TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Integrierte Systeme

Dynamic Partial Self-Reconfiguration of FPGAs for Digital Broadcasting Receiver Systems

Michael Feilen

Vollständiger Abdruck von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr.-Ing. Wolfgang Utschick
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Walter Stechele
	2.	Prof. Raymond Knopp, Ph.D.

Die Dissertation wurde am 05.07.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.02.2017 angenommen.

Abstract

Partial reconfiguration of field-programmable gate arrays enhances the design-space by unconfined repurposing of logic elements to virtually enlarge the available resources of a programmable device. In this work, the benefits and drawbacks of partial FPGA reconfiguration in radio receiver systems will be outlined. The analysis comprises design considerations for self-adapting receiver systems, where resources are shared among different reconfigurable areas. The derivation of the resource requirements and explanation of the design complexity delivers an insight into the applicability of an adaptive receiver system using partial reconfiguration. In addition to the analysis of self-adapting systems, a framework for block-wise execution of receiver chain elements on partially reconfigurable systems will be introduced. Using this framework, the timing constraints for the execution and the delay implications of the receiver chain elements will be derived and analyzed by means of a prototype implementation. Given the previously derived approaches, an efficient algorithm for receiver chain partitioning will be introduced and analyzed in terms of efficiency and performance. Finally, it will be concluded that it is possible to implement resource-efficient adaptive receiver chains using partial FPGA reconfiguration and that new design tools are required to exploit the hardware possibilities of state-of-the art FPGAs.

Acknowledgments

Writing this thesis was an exciting journey, and I thank all people who accompanied me during the time of my studies with their inspiration and patience. First, I would like to thank my supervising Professor Walter Stechele for his scientific reasoning and constructive advise – without his support, this work would not have been possible. I express my sincere appreciation to Professor Andreas Herkersdorf, head of the Institute for Integrated Systems for his engagement, for valuable discussions and for allowing me to be a part of his research team. Furthermore, I am grateful for fruitful discussions and constructive criticism by my colleague and friend Matthias Ihmig, who assisted me during my project work. In addition, I am thankful for the aspiring guidance and constructive feedback of Dirk Koch, Michael Vonbun and Lothar Stolz, who replied to my questions with scientific precision and a lively sense of humor. It was a pleasure to be able to supervise excellent students at the Technical University of Munich, namely, in chronological order: Stefan Strasser, Yu Qi, Ali Adan Malik, Philipp Schmidbauer, Anton Zahlheimer, Daniel Münch, Christian Schwarzbauer, Markus Gnadl, Hussein Alasadi, Andreas Iliopoulos, Michael Ruf and Korbinian Berthold. I am very thankful for their substantial contribution to my research. With the same gratitude I thank all staff members and Ph.D. colleagues at the institute for a brilliant research climate that shaped my scientific thinking, and Professor Andreas Steil, for his valuable feedback to meaningful questions. Warm thanks also to Norbert Niklasch, head of the IZ40 Sensor Signal Processing department at IABG, who inspired and encouraged me to put my results into words. Finally, I would like to thank the Bundesministerium für Wirtschaft und Technologie and TÜV Rheinland for supporting my work in the context of the DEUFRAKO project under Grant 10 P 8012B.

With gratitude and love I thank Lena, Julius and Carla for their everlasting support.

Munich, June 2016 Michael Feilen

Contents

List of Figures

List of Tables

"Papa, wenn Deine Arbeit fertig ist,
feiern wir ein Fest."

— Julius - 2015

1 Introduction

Increasing computational demands, stricter power constraints for mobile operation and ambitions to reduce the chip count put field-programmable gate arrays (FPGAs) in direct competition with application-specific integrated circuits (ASICs). For the development of modern receivers for digital broadcasting, cost-effective FPGA-based implementations with minimum resource utilization are of concern. Dynamic partial reconfiguration (DPR) of FPGAs provides additional degrees of freedom for the optimization of a design in terms of resources. In how far partial reconfiguration is feasible for the optimization of a particular receiver implementation and to what extend it imposes implications on the FPGA system design is within the scope of the outlined research. Considering the derived implications, design strategies for the realization of resource-economic receivers for digital broadcasting will be proposed and analyzed.

Explanations given in the following chapters require an understanding of FPGA architectures and properties of broadcast receivers. Knowledge required to comprehend the concepts to be presented will be summarized further on.

1.1 Field-Programmable Gate Arrays

Field-programmable gate arrays are configurable integrated circuits for implementing logical functions. The internal structure of an FPGA varies among manufacturers and by device family. Technology-wise, different FPGA memory cell types exist, such as flash-, anti-fuse- and static random-access memory (SRAM)-based memory cell designs. In terms of configurability and logic density, SRAM-based FPGAs are the most versatile choice for many applications. At the time of writing, the two leading manufacturers of SRAM-based FPGAs are Xilinx and Altera with a combined market share of almost 90 %, cf. [Joh11]. The company Xilinx Inc. was founded by Ross Freeman and Bernard Vonderschmitt in 1984, based on their patented "Logic Cell Array" technology [Gra04]. Support for dynamic partial reconfiguration of Xilinx FPGAs has been available since 2003 [BBHN04], whereas Altera made DPR officially available with the introduction of the Stratix V devices in 2013 [Rhe13]. A detailed report about modern FPGA architectures and technologies of various vendors is provided by Kuon et al. in [KTR08].

Due to the better tool support at the time of writing, the hardware-specific analysis in this work has been accomplished with Xilinx FPGAs. The tools that have been used for system design, synthesis and software development are Xilinx Integrated Synthesis Environment (ISE), Xilinx Software Development Kit (SDK), Xilinx Embedded Development Kit (EDK) and the latest Xilinx Vivado design suite. Signal processing systems have been designed using Xilinx System Generator together with Matlab/Simulink from Mathworks.

FPGA Application Layer

FPGAs provide different resources, such as lookup tables (LUTs), flip-flops (FFs), shift registers, hardware multipliers denoted as digital signal processing (DSP) units or DSP48 units, block random access memory (BRAM) and other components. Residing in the *application layer*, these resources can be configured by the user to perform certain tasks. Programmable internal routing networks in the same layer allow to interconnect the mentioned FPGA components and link them to external input and output (I/O) pins. For physical interaction with these pins, configurable I/O Blocks (IOBs) and special high-speed interfaces such as multi-gigabit serial I/O transceivers are provided.

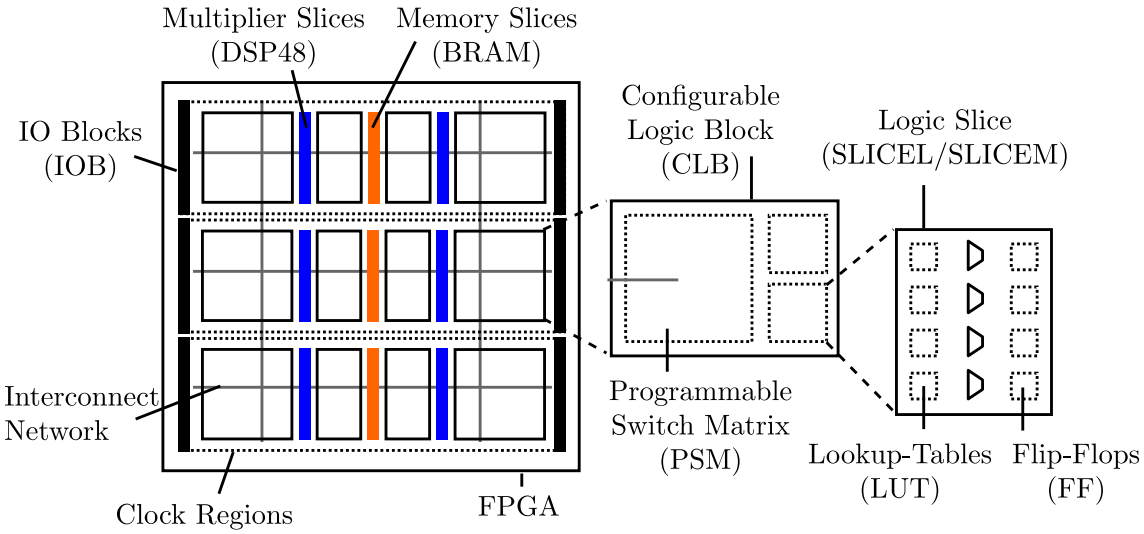


Figure 1.1: FPGA application layer resource floorplan.

Figure 1.1 shows a typical resource floorplan of an SRAM-based FPGA. An FPGA-specific set of LUTs and FFs is denoted as configurable logic block (CLB). The CLB resources are interfaced to the FPGA wire network using programmable switching matrices (PSM). In addition to this wire network, FPGAs comprise user-programmable clock signal trees to supply internal sequential logic elements. A clock signal can be distributed within a certain clock region, either directly driven by an external clock input pin or by a programmable phase-locked loop (PLL) from a digital clock manager (DCM). Once configured, the CLB slices, hardware multipliers, dedicated memory blocks and wiring resources resemble the user-defined logic functionality implemented by the FPGA.

FPGA Configuration Layer

The configuration and memory state of the application layer is controlled by the *configuration layer* of the FPGA. Once the configuration is accomplished, the FPGA application layer will resemble the user-defined digital circuity. Xilinx FPGAs are equipped with a selectable microprocessor access port (SelectMAP) and a Joint Test Action Group (JTAG) configuration access port accessible via external I/O. Additionally, modern FPGAs provide a serial peripheral interface (SPI) and a byte peripheral interface (BPI). Data written to these ports is forwarded to a configuration packet processor (CPP), which interprets

the supplied bits and provides read and write access to the frame data register (FDR) and control registers. In modern Xilinx FPGAs, an internal configuration access port (ICAP) gives access to the CPP from inside the application layer, on Altera FPGAs this counterpart is called partial reconfiguration control block (PRCB). By writing to this controller, FPGA self-reconfiguration can be performed, i.e. triggered internally by user-defined logic operations.

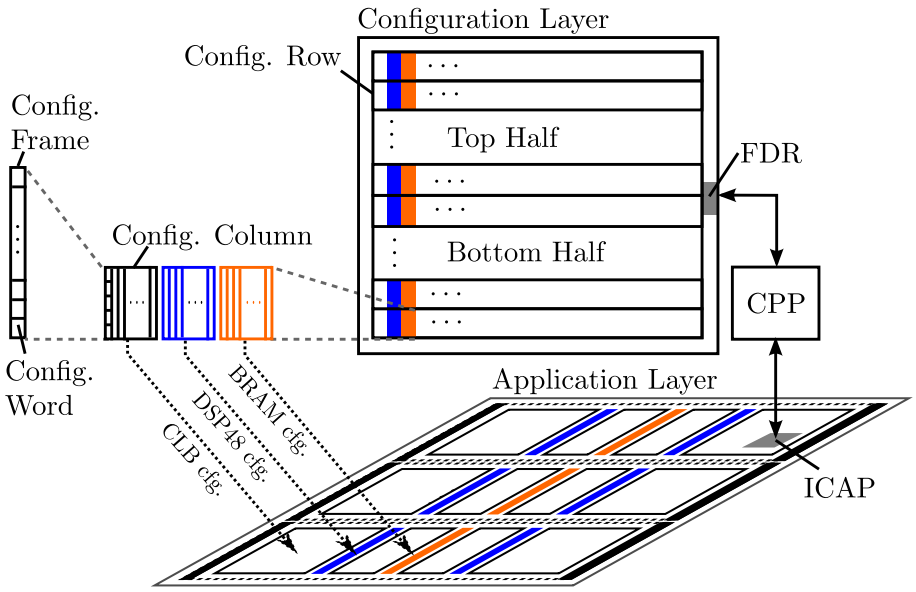


Figure 1.2: FPGA configuration layer and application layer tiling.

In Figure 1.2, the configuration layer is schematically sketched. It is partitioned into multiple configuration rows, on modern devices sometimes split into a top half and into a bottom half. Each configuration row contains multiple configuration columns for the different resource elements, such as CLBs, DSP48 slices, BRAMs or IOBs. A configuration column holds an integer number of configuration frames, which are the smallest addressable storage units in an SRAM-based Xilinx FPGA. A configuration frame is composed of multiple configuration words and the number of bits per configuration frame is obtained by multiplying the configuration interface width by the configuration word count. It is possible that the data of one configuration frame may affect multiple resource elements in the application layer, e.g. the wiring of multiple CLBs. Within the application layer domain, a configuration datastream can be supplied through the ICAP to the CPP. Upon successful synchronization, the CPP will forward the configuration frame information embedded into this stream to the input FDR. After a complete frame has been written to the FDR, the register contents are transferred to the FPGA configuration memory and the application layer changes will be applied. For consistency checking and comparison, it is possible to read back configuration frames through the output FDR.

Table 1.1 depicts the configuration interface parameters for state-of-the-art Xilinx Virtex FPGAs. Since the parameters are spread across different documents, individual references have been provided. Xilinx undiscloses the number of frames per column for newer FPGAs, wherefore they have not been provided in this work for Virtex-6 and Virtex-7 devices. Configuration interface widths of 32 bits per configuration word are

Xilinx FPGA Generation	$\frac{\#\text{Cfg.Words}}{\text{Cfg.Frame}}$	$\frac{\#\text{CLBs}}{\text{Cfg.Column}}$	$\frac{\#\text{CLB/DSP/BRAM Cfg.Frames}}{\text{Cfg.Column}}$
Virtex-7	101 [Xil15a]	50 [Xil14]	N/A
Virtex-6	81 [Xil15b]	40 [Xil14]	N/A
Virtex-5	41 [Xil12d]	20 [Xil12c]	36, 28, 30 [Xil12d]
Virtex-4	41 [Xil09b]	16 [Xil12c]	22, 21, 20 [Xil08]

Table 1.1: Xilinx Virtex FPGA configuration interface parameters.

used across all Virtex devices, together with a maximum configuration clock frequency of 100 MHz. Thus, read and write access to the CPP via the ICAP can be performed with a maximum rate of 3.2 Gbit/s. Note that the number of frames per column for BRAM resources refers to the interconnect configuration only, but not to the BRAM content, which requires a larger number of configuration frames [Xil12d]. It can be observed that, from one FPGA generation to the next, the configuration frame size increases in terms of words per frame, leading to an increased minimum configuration period.

FPGA Configuration Bitstream

The configuration datastream presented to the internal or external configuration interfaces is typically referred to as *bitstream*. In addition to command information it contains the configuration data with header information, such as row and column addresses for partial writes. In Xilinx FPGAs, synchronization to the bitstream is accomplished by the 32 bit sync-word `0xAA995566`, where `0x` indicates hexadecimal notation. The bitstream content is FPGA-specific and documented in the respective FPGA configuration user guide. Bitstreams with configuration data affecting only a subset of configuration frames are called *partial bitstreams*. Error detection and intellectual property protection can be employed by cyclic redundancy checksum (CRC) comparison commands and symmetric bitstream encryption according to the advanced encryption standard (AES). Further information on the bitstream format and other hardware-related details about the configuration architecture can be found in the respective FPGA configuration user guide.

Replacing specific portions in the FPGA application layer by writing partial bitstreams to the ICAP will be referred to as *partial self-reconfiguration* in this thesis. Achieving this without disturbing other application layer functions will be denoted as *dynamic partial self-reconfiguration*. Subsequently, the possibilities and constraints of this reconfiguration approach will be explained.

1.1.1 Dynamic Partial Self-Reconfiguration of FPGAs

Dynamic partial self-reconfiguration describes the ability of an FPGA to self-reconfigure an internal application layer partition, without interrupting the surrounding logic of this partition. A precise specification of the terming can be introduced as:

- **Dynamic** describes the ability to configure the FPGA (or portions of it) while a subset of logic resources and clock networks of the device remain operational. Typically, the continuously operating region is referred to as the static partition.

- **Partial** describes the ability to change parts of the FPGA application layer by updating a subset of configuration frames. The area in which one or multiple adjacent configuration columns and rows are changed is denoted as reconfigurable partition and is typically of rectangular shape in the resource floorplan.
- **Self-Reconfiguration** means that the FPGA provides an internal port to access the configuration layer from within the application layer. In case of Xilinx FPGAs, this port is called ICAP.

Temporal resource-multiplexing of FPGA resources is a major motivation for using partial FPGA reconfiguration, because re-using the FPGA resources for different features means being able to adapt to certain situations. A region of contiguous reconfigurable resources will herein after be referred to as reconfigurable partition or *DPR partition*. Typically, it is interfaced to the static partition using dedicated resource elements, further denoted as bus macros or *proxy logic*. Application subsets or features, specifically designed to be instantiated and to operate inside a DPR partition, will furthermore be referred to as *DPR module*.

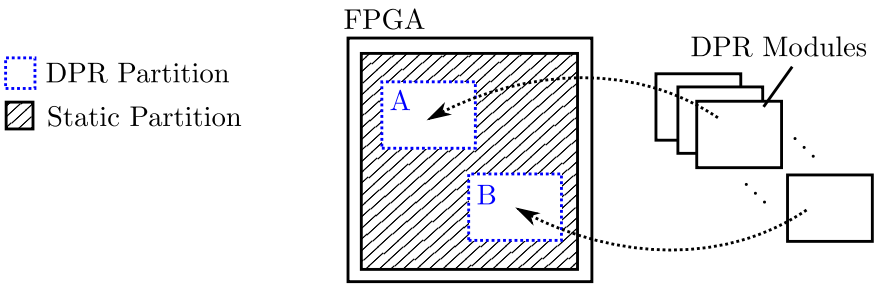


Figure 1.3: Schematic diagram of a partially-reconfigurable FPGA system.

In Figure 1.3 a conceptual DPR system is sketched with two DPR partitions A and B of equal size and multiple DPR modules. Writing the partial bitstream of a DPR module to the reconfiguration interface updates one of the specific DPR partitions as coded in the bitstream, i.e. the partition the DPR module has been specifically implemented for.

Since it took about 20 years for DPR to grow from a research topic into a state-of-the-art design methodology, it is necessary to introduce the research landmarks that lead to the concepts employed today. Subsequently, the current approaches will be put into context by a brief historical review.

A Historical Abstract of Dynamic Partial Reconfiguration

Conceptual ideas on dynamic FPGA reconfiguration appeared shortly after the first generation of FPGA devices emerged on the market in 1986. For example, in 1989, Gray and Kean announced "A new paradigm for computation" by the introduction of configurable array logic, referred to as CAL. The authors emblaze the possibilities of the structure in different case studies and emphasize the benefit to "restructure the hardware for a given algorithm"[GK89]. A few years later, Thomas C. Waugh presented SPLASH[Wau91], a reconfigurable linear logic array which allowed run-time reconfiguration of 32 individual Xilinx XC3090 FPGAs. These FPGAs did not provide self-reconfiguration, but reconfiguration was triggered externally.

A landmark coining the term *dynamic reconfiguration* was published in 1994 by Lysaght and J. Dunlop called "Dynamic Reconfiguration of FPGAs", where the authors describe the possibilities of run-time reconfiguration with state-of the art devices [LD94b]. At about the same time, the first adaptive signal processing implementation that made use of dynamic reconfiguration was presented in the work of Patrick Lysaght and Hugh Dick [LD94a]. In their publication, an implementation of a short-term autocorrelation function on a Xilinx 4005 and an Atmel AT6005 with externally-triggered reconfiguration is outlined. Furthermore, the authors compare the system complexity with a DSP system and conclude that, in terms of complexity, the autocorrelation FPGA setup was "not competitive today", and they predict that a "new hybrid device" with a microprocessor integrated into an FPGA would probably make the design competitive to a DSP.

In 1994 DeHon took the idea of time-multiplexed programmable hardware one step further and presented the concept of the dynamically programmable gate array (DPGA). The idea behind such a device is to have multiple logic configuration sets, so called "contexts", stored in an on-chip memory. The DPGA allows to quickly switch between the different contexts, and thus hardware functionality, at runtime. All context information is provided on load by a single configuration bitstream. The paper describes the benefits of "rapid reconfiguration" and mentions that DPGA array elements could be "reused in time". A first DPGA hardware prototype has been presented by Tau, Chen, Eslick and Brown at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology in 1995 (cf. [TCEB95]). The functionality and the routing of the gate array could be altered within a single clock cycle, thereby changing the functionality of a 4-input LUT together with a crossbar configuration. DeHon supervised this project and presented an analysis on the utilization of DPGAs in "DPGA Utilization and Application" one year later (cf. [DeH96]). As far as commercially available DPGAs are concerned, in 2008 Tabula, a company founded 2003 by Steve Teig, released the ABAX2 P1 DPGA called a 3PLD.

The temporal multiplexing of FPGA logic elements to subsequently execute blocks of a continuous processing pipeline has first been presented by Villasenor et al. in [VJS95]. The authors used a configurable logic array (CLAy31) and an erasable programmable read-only memory (EPROM) holding the configuration bitstreams, which were loaded by an external finite-state machine (FSM) on an Altera EP600 EPLD. Partial reconfiguration has not been used but instead the entire configuration of the FPGA has been swapped in a round-robin fashion. A similar idea for temporal multiplexing of processing blocks is revisited in Chapter 3 of this thesis.

Another landmark in reconfigurable computing is the work of Wirthlin and Hutchings, who in 1995 presented their Dynamic Instruction Set Computer, called DISC (cf. [WH95]), one of the first implementations using partial reconfiguration with the aim to reduce the reconfiguration time. DISC resembles a normal processor with the benefit of a run-time reconfigurable instruction set. A static "global controller" inside the National Semiconductor CLAy31 FPGA contains the necessary CPU components, such as status, data and address registers, program counter and instruction register. The column routing resources are used as shared control, data and address lines for the different reconfigurable modules. The authors quantify the reconfiguration overhead between 16 % and 71 % of the total operating time. About two years later, Wirthlin and Hutchings presented a metric to estimate the functional density of static and dynamic FPGA designs in [WH97]. The equations lead to the conclusion that, in general,

functional density reduces along with an increase in reconfiguration time. The authors propose to use partial reconfiguration of FPGAs to reduce the configuration time and thus increase the functional density of the system. Provided that the reconfigurable partition is small, the authors show that for a system with a high number of execution cycles per reconfigurable module, the functional density of a reconfigurable system exceeds the functional density of a static system. Another important work of the same decade is the time-multiplexed Artificial Neural Network system from Elderedge and Hutchings, presented 1996 (cf. [EH94]), where the logic of a Xilinx XC3090 FPGA is cyclically reconfigured between three distinct configurations.

Given the vital research on partial reconfiguration in the 1990s, Trimberger et al. proposed "the time multiplexed FPGA" in 1997 (cf. [TCJW97]). The work of Trimberger builds on top of the ideas of R. Ong, who filed a patent in 1995 for the design of an FPGA with DPGA functionality. The authors propose a new architecture based on a Xilinx XC4000E FPGA and suggest three modes of operation:

1. **Logic-Engine Mode:** Virtually enlarge the combinational logic by re-using, i.e. re-configuring, the FPGA LUTs within a "microcycle" or "user clock cycle". As the duration of a microcycle depends on the complexity of the time-multiplexed circuit, it is equal to one or multiple FPGA clock cycles. Trimberger suggests using flip-flops as intermediate buffers and also allow to feed-forward combinational outputs.

2. **Time-Share Mode:** Multiple LUT and Flip-flop resources can be reconfigured by a user-defined trigger. In contrast to (1), one reconfigurable module is executed for multiple FPGA clock cycles and then replaced by another module using dynamic reconfiguration.

3. **Static Mode:** The FPGA logic will not be affected by reconfiguration.

The approaches presented in this thesis require the FPGA to operate in time-share mode, where the configuration layer is re-written to update a subset of FPGA resources. Additionally, dynamic self-reconfigurability of the device is required, such that a continuously operating static FPGA partition can perform an update of a DPR partition by writing to the configuration layer controller. Although the first Xilinx Virtex devices supported partial reconfiguration, they did not provide an internal configuration controller. Hence, self-reconfiguration by internal wiring was not possible. With the introduction of the ICAP with the Xilinx Virtex-II Pro FPGA family, dynamic partial self-reconfiguration became available. Since then, several DPR architectures and applications for the Virtex-II Pro family have emerged. However, with one configuration frame spanning all primitives within a full device column, the Virtex-II FPGA application layer fabric did not allow a vertical, i.e. CLB row-wise, area partitioning, which was disadvantageous in terms of routing and area usage. With the introduction of configuration rows with a height of 16 CLBs per configuration column, Virtex-4 FPGAs abrogated this issue, thus enabling a fine-grained FPGA partitioning as presented in [LBM+06]. The improved tiling architecture of modern Xilinx FPGAs allow even more complex reconfigurable system designs with multiple DPR islands.

Today, improved vendor tool support makes the design of DPR systems much simpler as compared to the time when DPR was first introduced (cf. [Xil14]). However, designing

DPR systems is still challenging and requires to follow certain design steps and adhere to specific design rules. The process to create partially configurable systems will be referred to as *DPR system design flow*. Two popular flows will be subsequently explained: the difference-based flow and the partition-based flow.

1.1.2 Difference-Based DPR System Design Flow

The difference-based design flow is described in [Xil07] and was originally referred to as "Small-bit Manipulation Flow". Possible use cases are the manipulation of BRAM contents, LUT equations or changing I/O standards. Albeit intended to be used for minor application-layer modifications, the flow has also been used for large-scale design modifications (cf. [KBT08] and [MNH+11a]). A difference-based bitstream can be generated using the Xilinx Bitgen command line tool with the option `-r`. The tool compares the bitstream of a module A with the native circuit description (NCD) of the destination module B and generates a bitstream with the differences of A and B. The resulting bitstream will contain the configuration layer modifications for A that lead to an application layer realization of B. Since generating a differential bitstream for a new configuration requires the knowledge of the previous configuration, the difference-based flow is disadvantageous for designs with more than two DPR modules as subsequently depicted.

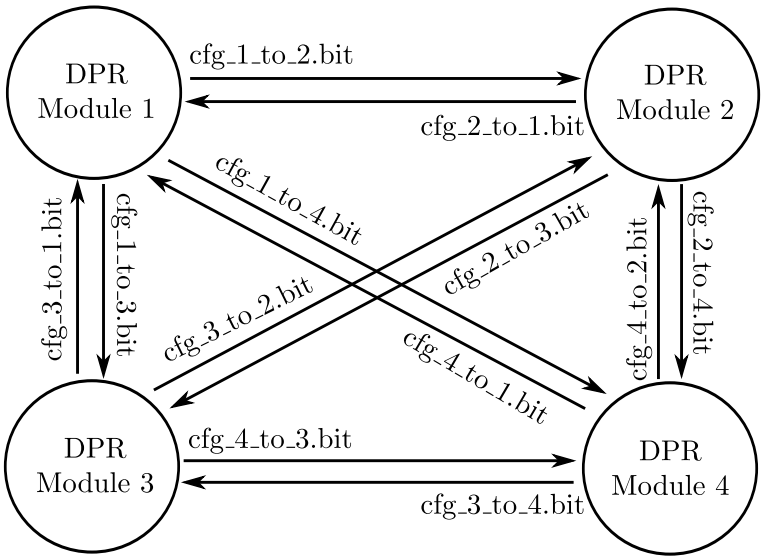


Figure 1.4: Difference-based reconfiguration flow with four DPR modules.

The graph in Figure 1.4 shows the differential bitstream configuration flow. Each edge of the graph represents a partial bitstream and the four DPR modules are represented by the nodes of the graph. Being able to switch from one module to any other module requires twelve independent differential bitstreams, and for a system with M DPR modules $M(M-1)$ partial bitstreams would be required. By using blanking bitstreams this number can be reduced to $2M$, i.e. one bitstream for loading and one for deletion, at the cost of one additional FPGA configuration cycle. Additionally, when using difference-based reconfiguration, changes in the signal routing might lead to errors in the static

partition or other reconfigurable partitions. This effect can be mitigated by the use of *blocker macros*, which feign that all routing resources outside the DPR partition have been occupied, thus forcing the routing algorithm to use interconnects inside the DPR partition only. At the time of writing, there is no vendor tool support for restricting the routing to certain areas. However, third party tools exist which make use of the Xilinx design language (XDL) for blocker macro generation [BKT11].

A rather modern design flow supported by Xilinx is the partition-based design flow, where routing and implementation of the DPR partitions can be performed independently as subsequently outlined.

1.1.3 Partition-Based DPR System Design Flow

The partition-based flow is the DPR system design flow recommended by Xilinx and is documented in [Xil12c] for Xilinx ISE designs and in [Xil14] for designs created with Xilinx Vivado. It replaces the older module-based reconfiguration flow, which sometimes is referred to as early access partial reconfiguration (EAPR) flow, in relation to the naming of the respective Xilinx user guide.

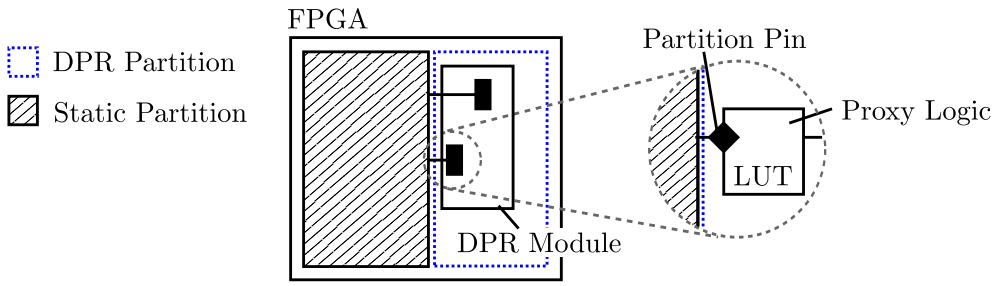


Figure 1.5: Partition pins and proxy logic locations for partition-based DPR systems.

Using the partition-based design flow requires the definition of DPR partition boundaries, either by using tools like Xilinx PlanAhead or by manual definition of area constraints. The reconfigurable partition is allowed to enclose slices, block RAM and DSP units. Resources like DCMs, PLLs and I/O resources must reside in the static partition. For every signal, a LUT in route-through mode is used as proxy logic providing a *partition pin* to connect the static and dynamic regions (cf. Figure 1.5). These partition pins are inserted automatically by the tool chain but can also be specified manually using location constraints. Since the proxy logic is effectively a part of the static logic, it can be placed anywhere inside the reconfigurable region. In partition-based designs, LUT elements are used as proxy logic, providing unidirectional asynchronous communication between the static and dynamic partitions. Resource-wise one LUT is allocated per transition wire and signal direction. The locations of the proxy LUTs are fixed within the area of a DPR partition and must therefore be known at implementation time of each DPR module. During reconfiguration, the signals inside the DPR partition are undefined. Therefore, connections to the static parts of the design should be decoupled to avoid glitching input signals. When using the partition-based design flow, decoupling can be achieved using enable flip-flops inside the static region as recommended by Xilinx in [Xil12c]. Since registering also reduces the combinational path delay, the timing performance of the circuity can be improved at the cost of a unit delay.

Exploiting the possibilities of DPR for digital broadcast receivers is part of the analysis presented in the subsequent chapters and will therefore be explained in further detail.

1.2 Digital Broadcasting Receivers

According to the Collins English dictionary, an (analog) radio receiver is defined as *"an apparatus that receives incoming modulated radio waves and converts them into sound"* (cf. [Col11]). Instead of converting radio waves into sound, a digital receiver converts radio waves into information. Improvements in encoding and decoding of this information are the main driver for the introduction of new terrestrial, satellite and cable-wire broadcasting standards deployed world-wide. Efficient coding usually means low-bitrate high-quality audio and video source-coding as well as near Shannon limit channel coding (cf. [ESL04]).

The way the information is coded is typically specified by an expert group and the specification the group members agree on are commonly referred to as *standard*. Next, the most important broadcasting standards related to this work will be introduced.

1.2.1 Selected Standards

The analysis presented in this work requires an introduction to the European terrestrial broadcasting standards terrestrial digital video broadcasting (DVB-T), digital audio broadcasting (DAB) and stereophonic analog frequency modulation (FM) including the radio data system (RDS). In Europe, these standards are operated mainly in the very high frequency (VHF) bands I to V according to ITU-R 432-7 (cf. [itu15]) as listed in Table 1.2.

VHF Band	Start of Band	End of Band	Standard
I	47 MHz	68 MHz	-
II	87.5 MHz	108 MHz	FM/RDS
III	174 MHz	230 MHz	DAB
IV	470 MHz	582 MHz	DVB-T
V	582 MHz	960 MHz	DVB-T

Table 1.2: European VHF frequency bands according to ITU-R 432-7.

For signal reception and physical layer decoding, each standard requires individual radio frequency (RF) hardware and baseband decoding chains. The baseband decoder can be implemented in hardware using FPGAs or ASICs or in software using Microprocessors. When implemented on an FPGA, dynamic partial reconfiguration can be utilized to reduce the resource consumption in comparison to static FPGA implementations, as derived further on in this work.

In order to classify the baseband decoder chains presented in the following chapters, important historical and technical cornerstones will be outlined in chronological order.

FM/RDS

After the ratification of the Copenhagen frequency plan in 1948 the first frequency-modulated audio broadcasts went on air in Europe. A few years later, the FM stereophonic multiplex (MUX) signal was standardized and enabled broadcasting of two independent audio channels. The desire to uniquely identify FM broadcasts, to transmit alternate frequency information and to carry traffic announcement signals lead to the development of RDS within the years 1975 and 1984 (cf. [Rds]). Later on, RDS was published as a standard by the International Electrotechnical Commission (IEC) in document 62106 Edition 3. The system will be explained in further detail in Chapter 2 in conjunction with an architecture for a reconfigurable FPGA-based FM receiver.

DAB

Research and development of the DAB system for the transmission of digital audio information dates back to the 1980s and was driven by several European research institutes, broadcasting companies and radio manufacturers within the EUREKA 147 project. First receiver system concepts based on orthogonal frequency division multiplexing (OFDM) were presented in 1989 by Floch et al. (cf. [LFHLC89]). Differential quadrature phase-shift keying (DQPSK) and punctured convolutional coding were proposed for DAB, allowing for efficient channel decoding using Viterbi's algorithm and channel equalization without dedicated channel estimation. Together with the invention of the MUSICAM audio codec presented in 1991 (cf. [DLU91]), the first generation DAB system was standardized in the early 1990s. In 2005, DAB+ emerged as next-generation radio system, using Reed-Solomon coding in concatenation with the convolutional code and high-efficiency advanced audio coding (HE-AAC) v2 for higher quality audio at lower bitrates. Nowadays, DAB and DAB+ are employed in several European countries as major digital technology for sound broadcasting. The system specification is published by the European Telecommunications Standards Institute (ETSI) in EN 300 401 [ets06]. The architecture of an FPGA-based DAB receiver for terrestrial reception in DAB mode I will be presented in Chapter 3 together with an approach to time-multiplex receiver components using partial reconfiguration of FPGAs.

DVB-T

In 1993, the Digital Video Broadcasting Group was formed by major European media interest groups to develop a new standard for digital video broadcasting (cf. [Dvb]). The standard for terrestrial video broadcasting (DVB-T) was ratified in 1997 and the first services went on air in Germany in 2002. Similar to DAB, DVB-T uses OFDM and convolutional coding. For transmit power efficiency reasons, coherent quadrature amplitude modulation (QAM) is used instead of DQPSK. Thus, in contrast to DAB, in DVB-T channel estimation and channel tracking are mandatory. In 2008, the successor DVB-T, called DVB-T2, was adopted by the ETSI in EN 302 755 [Ets08]. DVB-T2 promises to be 50 % more power efficient than DVB-T by using high-order QAM, fewer pilot signals and improved forward-error correction by concatenated low-density parity check code (LDPC) and Bose-Chaudhuri-Hocquenghem (BCH) codes. In terms of computational complexity, DVB-T2 is the most demanding terrestrial broadcasting standard. A proposal for the design of a reconfigurable FPGA-based DVB-T2 baseband decoder will be

outlined at the end of Chapter 3.

1.2.2 Receiver Design and Properties

Spectral efficiency and the possibility to use low-complexity channel estimation and equalization routines have made OFDM the modulation scheme of choice for state-of-the-art digital terrestrial broadcasting. According to Speth et al. an OFDM receiver can be split into an inner and an outer part (cf. [SFFM99]). The inner receiver compensates for all signal impairments such as timing offsets, frequency offsets, sampling clock offsets and channel distortions and forwards the equalized carrier information to the outer receiver, where channel decoding is performed. Figure 1.6 shows a simplified signal flow-graph of an OFDM receiver including the inner and outer parts.

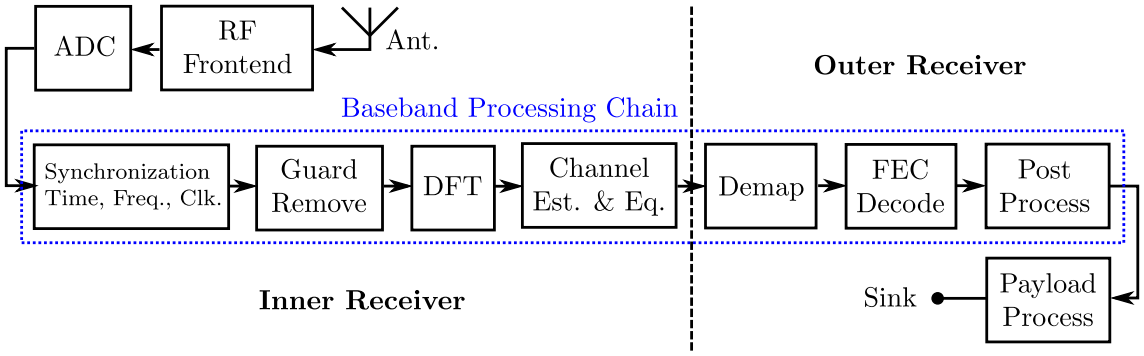


Figure 1.6: OFDM receiver signal flow graph.

The received signal is detected by the antenna and then down-converted in frequency and amplified in power using an RF frontend. Digitization of the frontend output signal is accomplished using an analog-to-digital converter (ADC) delivering a stream of quantized in-phase and quadrature values, referred to as *complex baseband signal*. The complex baseband signal is subsequently processed by a *baseband processing chain*, implemented on an FPGA and subject to optimization. At the output of the chain, the decoded bits are presented to a payload processor and forwarded to a sink for further processing. Although the OFDM chain shows the most important blocks for the decoding of various standards, each specific standard requires a tailored implementation.

The importance of the optimization of the baseband processing chain using DPR will be elaborated on in the forthcoming sections, whereas the design and optimization of other components, such as RF frontend, ADC or payload processor will not be concerned.

Baseband Processing Chain

The baseband processing chain is encircled by a blue dotted line in Figure 1.6. For simplicity reasons the diagram shows a feed-forward-oriented data flow, which hides the feedback structures of control loops typically used in receivers. The presence of feedback structures in block-wise executed receiver chains will be further investigated in Chapter 3.

At the input of the baseband chain, the time-domain synchronization stage corrects the frequency offset, the sampling rate offset and the temporal demodulation window offset. After the guard interval has been removed, the discrete Fourier transform (DFT)

operation converts the signal into the frequency domain. The DFT is typically calculated using the computationally efficient fast Fourier transform (FFT) algorithm. If present, the pilot signals are extracted from the modulated carrier bins and subsequently used for channel estimation. Using simple zero-forcing or more advanced equalization methods, the impairments of the terrestrial broadcast channel are equalized and the modulated symbols are demapped. The obtained log-likelihood ratio (LLR) values, also called soft bits, are fed to a forward error correction (FEC) decoder for channel decoding. Depending on the type of channel decoder, the LLR values need to be interleaved to spread error bursts across a frame of data. Broadcast receivers typically use convolutional interleavers to keep the receiver input-to-output latency at a minimum. The decoded bits are then post-processed, for example by an energy dispersal sequence, and provided to a payload processor and a sink.

Baseband Data-Flow

Although feedback is typically employed for synchronization and equalization control loops, the data flow inside the receiver is mostly feed-forward-oriented. The data rate of the sequentially processed information stream is typically decreasing from the baseband input to the decoder output, i.e. the data rate of the complex baseband stream is the highest and the data rate of the output payload bitstream the lowest in the chain. Due to the DFT operation, data needs to be processed block-wise, which inherently leads to a latency in decoding. Additionally, the DFT block-size may be increasing or decreasing at integer rates due to transmission framing. In case of broadcasting receivers, interleaver frames or large FEC frames typically determine the latency of the system. The listed data-flow properties are important for the analysis in Chapter 3 and Chapter 4.

Subsequently, the scope and structure of this work will be outlined by a description of the research focus covered in the following chapters.

1.3 Scope of this Work

The research outlined in this work focuses on analyzing the benefits, limitations and possibilities of temporal hardware resource multiplexing for digital receiver chain implementations using dynamic partial reconfiguration of FPGAs. Contributions of this thesis and references to related works cover:

- Design aspects and benefits of reconfigurable hardware for self-adapting broadcast receivers.
- Impacts of cyclic partial reconfiguration for block-wise execution of radio receiver components by time-multiplexing of FPGA resources.
- Receiver chain partitioning for cyclic partial reconfiguration.

Since the scope of this work affects interdisciplinary subjects, the elaborated results comprises contributions in the fields of signal processing and digital circuit design for real-time systems.

1.4 Structure of this Work

In each of the following chapters one individual aspect of temporal FPGA resource multiplexing for digital signal processing chains will be discussed. Figure 1.7 shows the structure of this thesis in a hierarchical diagram together with the major units of observation. The first chapter is mostly self-contained, which means that despite minor referencing it is possible to follow the analysis of the second and third chapter without reading the first chapter. The description of related works, related contributions and bibliography are explained individually in each chapter. An overview of the structure and content of the subsequent chapters is given further on.

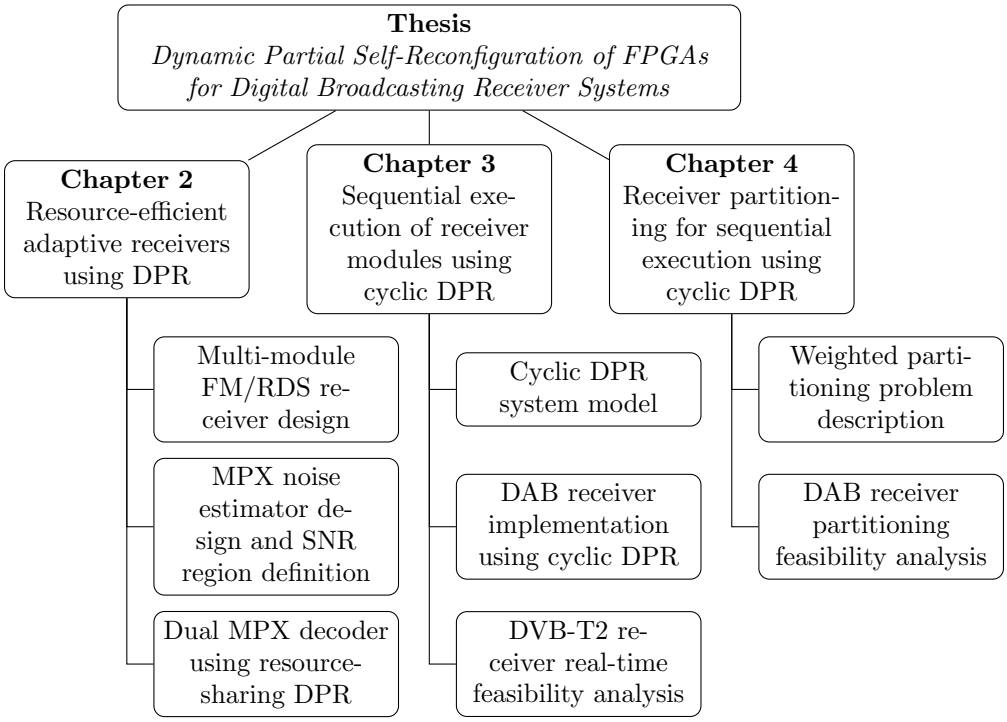


Figure 1.7: Structure of this thesis and major units of observation.

- **Chapter 2:** The design of a dynamically self-adapting FM/RDS radio receiver using DPR of a Xilinx Virtex-5 FPGA will be presented, where partial reconfiguration of the receiver is triggered using a signal-to-noise ratio (SNR) metric. In order to gain insight into the hardware complexity and system functionality, the receiver implementation, the hardware setup and the resource requirements will be outlined in detail. In addition, a method for receiver partitioning will be presented and a novel FM signal quality estimator will be proposed. Switching criteria for the reconfiguration of the dynamic partition will be derived from a receiver noise disturbance model and the tool-flow of the dynamically self-adapting system will be presented. Finally, a feasibility analysis for a multi-island resource-sharing reconfigurable system using vendor tools will be presented and the possibilities and limitations of the resource-sharing approach will be discussed.
- **Chapter 3:** The implications of cyclic reconfiguration for the sequential execution of signal processing chain elements will be presented in the third chapter.

Starting from a system model for cyclic reconfiguration, the requirements for a frame-wise execution of data will be derived. After presenting the buffering and latency implications on the processing chain, two feasibility studies for two digital broadcast receiver systems will be presented. Resource estimates for a low-cost Xilinx Spartan-6 FPGA will then be used to quantify the potential FPGA resource savings and the effect of buffering latency for real-world receivers.

- **Chapter 4:** An efficient method for processing chain partitioning of FPGA hardware designs using cyclic DPR will be elaborated. The partitioning problem will be formulated after the presentation of related works and an approach to solve the partitioning problem will be introduced. Using a weighted resource and latency metric, an approach for selecting suitable partitioning candidates will be derived and a novel approximation algorithm with linear time complexity will be presented. The chapter is concluded by a case study for the partitioning of a real-world DAB decoder chain.

- **Chapter 5:** A summary of the contributions and findings of this thesis will be presented in the last chapter, followed by an elaboration of potential future works.

2 FPGA Self-Reconfiguration for Adaptive Radio Receivers

The terrestrial broadcast transmission channel imposes impairments on the transmission signal leading to a signal degradation at the receiver. Since the signal reception quality influences the amount of information that can be recovered without error at the receiver, working groups usually define minimum receiver requirements by means of an impairment model. Thus, in conjunction with the specification of new radio standards, channel models, simulation parameters and receiver performance values are defined to ensure that a specified minimum performance can be achieved across all receiver implementations. Compared to implementations targeting a reception performance close to the theoretical optimum, minimum receiver requirements contain margins, leaving the developer with headroom for complexity of the employed decoding algorithms. Exploiting this headroom allows to use more or less complex algorithms and implementations. Adaptively switching between these more or less complex implementations using an FPGA is possible by dynamic partial reconfiguration. The analysis of reconfigurable architectures and the derivation of reconfiguration constraints for adaptive FPGA-based receiver systems is the research focus of this chapter.

Given an SNR-based reception quality metric, an adaptive FM receiver implementation using DPR will be presented. For this purpose, a novel SNR estimation routine will be derived and a set of meaningful SNR threshold values will be introduced. In addition to single-island and multi-island DPR solutions, a feasibility analysis for sharing FPGA resources of a single DPR partition between two DPR modules will be presented, where dynamic partial reconfiguration has been accomplished on a Xilinx Virtex-5 FPGA and the DPR system has been implemented using the vendor tool-flow as recommended by Xilinx.

The evaluation and design of adaptive signal processing chains is a wide research topic with many scientific contributions. Explaining the scope and categorizing the contributions of others is essential to put the matters of this work into context. Therefore, related works will be subsequently outlined together with a presentation of the achievements described in this work.

2.1 Related-Work and Contribution

Dynamic partial reconfiguration of FPGAs is used for various applications in reconfigurable computing, audio and video processing as well as in software-defined radio (SDR) systems. Works related to adaptive receiver chains and cognitive SDR systems are related to the analysis presented in this chapter and of particular importance in the following depiction of adaptive FPGA-based receivers.

In [DGRB04], Roland et al. dynamically reconfigure a phase-shift keying (PSK) filter

on a Virtex 1000E FPGA for a cellular communication system using a modular design approach. The presented system is not self-adapting and an external DSP is controlling the reconfiguration process. Similarly, Delahaye et al. presented a partially reconfigurable software-defined radio system in [DPML07], where the exchange of a constellation mapper, convolutional coder and finite impulse response (FIR) filter using DPR is described. An SNR-driven reconfiguration system for WiMAX systems on a Xilinx Virtex-4 SX35 FPGA has been developed by Chitty et al. in [CKPLM10]. In his work, he describes a link-adaption algorithm using an SNR estimation stage and SNR threshold values as reconfiguration trigger, which is similar to the work presented in this chapter. The system is designed using the Xilinx modular design flow with a fixed DPR partition, and, in comparison to the approaches presented in this work, Chitty does not use the ICAP for internal reconfiguration but uses an external computer to trigger a reconfiguration by writing the partial bitstream to the SelectMAP interface.

Lotse et al. have investigated in bit error rate (BER)-adaptive reconfiguration of modulation schemes and dynamically adapted the constraint length of a convolutional code given a certain channel scenario (cf. [LFDN09]). The authors used a Virtex-II Pro and operated the BER decision engine on the Power PC core, which is also used to trigger the reconfiguration. To save power, the authors propose to clear the DPR region if no receive signal is present. In contrast to the subsequently presented approaches, the signal quality detection engine is operating in a fixed-size reconfigurable partition. Furthermore, the system in [LFDN09] requires a feedback path to the transmitter for link-adaption. In [MMT+08], Manet et al. describe the benefits and drawbacks of dynamic partial reconfiguration for signal processing applications. He describes the problem of partition fragmentation, i.e. when the DPR partition size can not be changed during runtime, by "wasted" resources. The described problem affects all previous works.

In this work, a hierarchical reconfiguration approach is described to overcome partition fragmentation and share resources between otherwise fixed DPR partitions. Design approaches based on new third-party tools that show the feasibility of hierarchically reconfigurable systems have been presented in [KB14]. In the subsequent sections, practical use-cases for the application of these new tools will be illustrated by means of a self-adapting FM receiver system. Serving as a basis of the outlined work, the concept of a resource-sharing SNR-adaptive receiver was first discussed in [MF10]. Two years later, a similar reconfigurable FM receiver prototype was presented in [KTB+12]. In addition to the published material, the major contribution of the work described in this chapter is the comprehensible description of the processing chain complexity, modularization possibilities and reconfiguration approaches to realize an SNR-adaptive system with vendor tools. The results highlight the limits and possibilities of reconfigurable receiver systems and provide insights into the practical feasibility of reconfigurable FPGA-based receiver systems in addition to the theory.

Given the outlined state-of-the art, the major contributions of this work are:

- The design of an FPGA-based SNR-adaptive FM receiver system using multiple DPR partitions.
- The evaluation of hierarchical partial reconfiguration for FPGA resource-sharing between two adaptive receivers.
- The design of a novel SNR estimation method based on estimating the noise power

in the band-gaps of the demodulated FM signal.

- A comprehensive description of the FM receiver implementation and complexity analysis in context of dynamic partial reconfiguration.

Understanding the following sections requires an introduction to FM sound broadcasting at VHF band II to be provided further on. In addition, hardware implementations of FM receiver components and the design of a self-adapting FM receiver system using multiple DPR partitions will be described.

2.2 FM Sound Broadcasting

Frequency modulation is a wide-spread analog modulation scheme used for audio and voice communications. In order to understand the derivations in the subsequent sections, a brief introduction to the baseband representation of an FM signal will be outlined. The notation follows that of Werner in [WM06] and Kammeyer in [Kam08]. In the further course of this section, the FM multiplex (MPX) signal structure will be presented and the implications of noise disturbances in FM broadcasting will be explained.

Given is the complex-valued angular-modulated baseband signal $\underline{x}_{\mathrm{FM}}(t)$ with constant amplitude A_{FM} and time-variant phase $\phi(t)$ as follows

$$\underline{x}_{\mathrm{FM}}(t) = A_{\mathrm{FM}} \cdot \mathrm{e}^{\mathrm{j}\phi(t)}, \tag{2.1}$$

where underlining indicates complex baseband notation and j denotes the imaginary unit with the property $\mathrm{j}^2 = -1$. The carrier power of the constant-envelope signal in Equation 2.1 at a resistive load of 1 Ohm is equal to A_{FM}^2. The signal $\underline{x}_{\mathrm{FM}}(t)$ has a *time-variant* angular frequency $\omega(t)$ and the instantaneous angular frequency at a time instant t is equal to the first derivative of the phase function $\phi(t)$, i.e.

$$\omega(t) = \frac{d\phi(t)}{dt}. \tag{2.2}$$

Equation 2.2 states that the information carried in $\omega(t)$ is represented by the changes of the angular frequency over time, and this implies that the maximum frequency deviation of the baseband signal $\underline{x}_{\mathrm{FM}}(t)$ is determined by the peak values of $\omega(t)$. Therefore, the peak frequency deviation of $\underline{x}_{\mathrm{FM}}(t)$ can be expressed by

$$\Delta f_{\mathrm{MAX}} = \frac{1}{2\pi} \max\left(|\omega(t)|\right),$$

where $|\cdot|$ means taking the absolute value. Relating to a complex baseband representation of the FM signal, the peak frequency deviation reflects the maximum deviation of the instantaneous frequency from 0 Hz. Let the information carried in $\omega(t)$ be represented by a continuous real-valued signal $x_{\mathrm{LF}}(t)$, e.g. $x_{\mathrm{LF}}(t)$ could be an audio signal. According to the definition of frequency modulation, the instantaneous angular frequency $\omega(t)$ is a linear function of $x_{\mathrm{LF}}(t)$, i.e.

$$\omega(t) = K_{\mathrm{FM}} \cdot x_{\mathrm{LF}}(t), \tag{2.3}$$

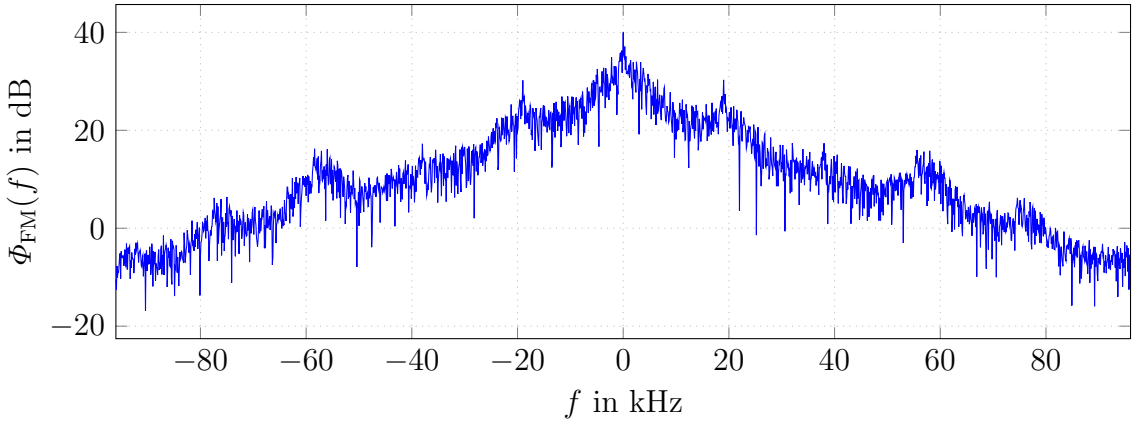


Figure 2.1: Double-sided power spectral density of FM baseband signal.

where K_{FM} denotes the modulation coefficient. Given a source-signal peak-to-peak constraint of $-1 \leqslant x_{\mathrm{LF}}(t) \leqslant 1$, a modulation coefficient of $K_{\mathrm{FM}} = 2\pi\Delta f_{\mathrm{MAX}}$ and the formulation in Equation 2.2, Equation 2.3 can be rewritten to

$$\frac{d\phi(t)}{dt} = 2\pi\Delta f_{\mathrm{MAX}} \cdot x_{\mathrm{LF}}(t). \tag{2.4}$$

Finally, Equation 2.4 can be reformulated by applying the second fundamental theorem of calculus to obtain the a signal model for frequency-modulated signals as

$$\phi(t) = 2\pi\Delta f_{\mathrm{MAX}} \int_0^t x_{\mathrm{LF}}(\tau)d\tau + \phi_0,$$

where ϕ_0 describes a the phase offset at $t = 0$. Figure 2.1 depicts the power spectral density (PSD) of a modulated FM audio baseband signal $\Phi_{\mathrm{FM}}(f)$ showing an approximately triangular-shaped spectrum, which is characteristical for frequency-modulated signals. The PSD was calculated using a received on-air audio program of 1 second duration.

In the early days of FM sound broadcasting, a monophonic low-frequency signal $x_{\mathrm{LF}}(t)$ has been used, either taken from a single channel audio source or from the sum of a stereo audio program. For stereo broadcasts, the LF signal has been modified while preserving backward-compatibility with older monaural receivers. According to the International Telecommunication Union (ITU) in ITU-R BS.450-3 [Itu01b], this is achieved by using the pilot-tone-based stereophonic multiplex signal, commonly referred to as MPX signal. Decoding the FM signal resembles the real-valued MPX signal. The windowed one-sided PSD of an audio broadcast is plotted in Figure 2.2.

As stated by the ITU in [Itu01a], in most countries the MPX signal consists of 4 components, where the continuous time-domain signals of the left and right MPX audio channels are denoted by $L(t)$ and $R(t)$, respectively:

- The sum audio channel $S(t) = \frac{1}{2}(L(t) + R(t))$, starting at around 15 Hz with a bandwidth of 15 kHz. Before MPX insertion, a pre-emphasis filter of first order with a time constant of 75 μs or 50 μs is applied to $S(t)$.

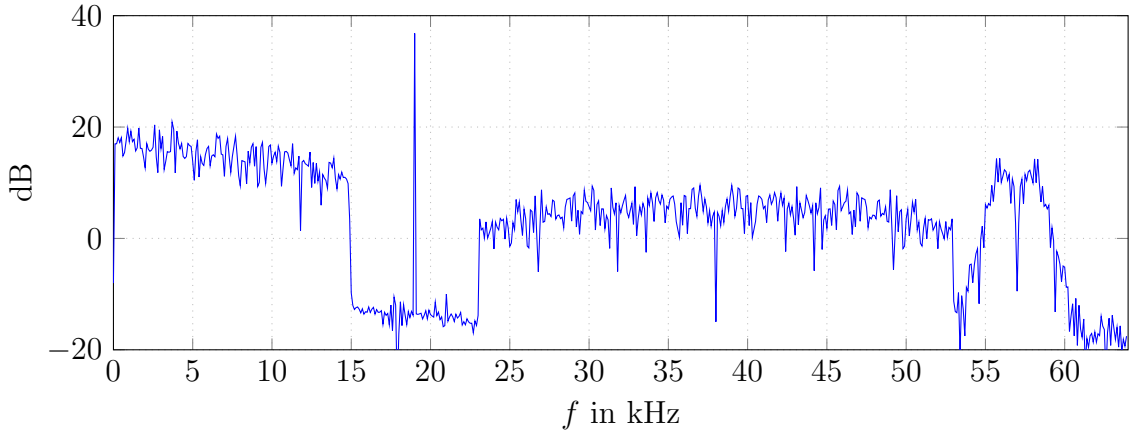


Figure 2.2: Windowed one-sided power spectral density of FM multiplex signal.

- The sinusoidal pilot tone at a frequency of 19 kHz, contributing at maximum 8 % to 10 % to the peak frequency deviation.

- The difference audio signal $D(t) = \frac{1}{2}(L(t) - R(t))$, located at 38 kHz with a one-sided bandwidth of 15 kHz. The difference signal is generated using amplitude modulation with suppressed carrier. Before MPX insertion, a pre-emphasis with a time constant of 75 μs or 50 μs is applied to $D(t)$.

- The RDS signal is represented by a continuous stream of differentially-encoded binary phase-shift keying (DEBPSK) symbols centered around 57 kHz in the FM multiplex. Manchester coding is used for DC free transmission to avoid potential cross-talk in existing stereo decoder PLLs [KM99]. The encoded bits are shaped with a square root-raised cosine pulse with a roll-off factor of $\beta_{\mathrm{SRRC}} = 1$. With a gross data rate of 1187.5 bit/s, the Manchester-coded RDS signal has a one-sided bandwidth of 2.375 kHz and hence a two-sided bandwidth of roughly 4.8 kHz.

From the audio sum and difference signals, the left and right audio channels can be recovered by:

$$\begin{aligned} L(t) &= S(t) + D(t) = \frac{1}{2} \cdot [(L(t) + R(t)) + (L(t) - R(t))] \\ R(t) &= S(t) - D(t) = \frac{1}{2} \cdot [(L(t) + R(t)) - (L(t) - R(t))] \,. \end{aligned} \tag{2.5}$$

For a perfect audio channel separation based on Equation 2.5, a coherent demodulation of the difference signal is mandatory. Since the difference signal is in-phase with the 19 kHz pilot tone, coherent demodulation can be achieved by using a phase-locked oscillator running at 38 kHz. The RDS signal can be demodulated similarly, although an RDS carrier frequency estimation can also be performed by an independent carrier recovery loop.

In the next section, an SNR-adaptive receiver chain will be presented for an FM/RDS broadcast receiver chain.

2.3 A modularized FPGA-based FM Receiver

The design of a digital FM receiver is similar to the design of its analog counterpart. However, in terms of linearity, the digital receiver has advantages over an analog design, which in case of an FM broadcast receiver results in a better stereo separation and RDS demodulation. Since the complexity of the decoding algorithms of a digital receiver determines the FPGA resources needed for the actual implementation, the algorithms and implementation-specific considerations will be introduced. Depicting the implementation also helps understanding the design approaches outlined in the following sections focusing on partial reconfiguration.

2.3.1 Receiver Modules

The receiver has been designed with the goal in mind to obtain a modular hardware implementation for an SNR-adaptive operation in the FPGA. Therefore, the receiver chain has been sub-partitioned into four major processing blocks:

1. **FM demodulator:** The FM demodulator provides an estimate of the instantaneous frequency of the input signal. After FM demodulation, the FM multiplex signal is obtained, which contains the audio sum signal, the audio difference signal, a 19 kHz pilot carrier and the differentially-encoded BPSK data signal at 57 kHz.

2. **Monaural sum signal audio decoder:** The decoder extracts the monaural signal, which contains the sum of the left and right audio channels. It contains a combined decimation and low-pass stage to limit the LF audio signal to a frequency of 15 kHz.

3. **Difference signal stereo decoder:** The difference signal decoder coherently demodulates the audio difference signal L-R at 38 kHz and extracts the left and right audio components using the monaural sum signal. For coherent demodulation, i.e. in-phase signal combining, the stereo decoder requires a reference oscillator. The decoder derives the 38 kHz demodulation signal from the pre-filtered 19 kHz pilot carrier in the FM multiplex, which is in-phase to the 38 kHz modulated audio. This 19 kHz pilot tone is extracted using a digital phase-locked loop.

4. **The radio data system decoder:** The RDS decoder demodulates the DEBPSK data signal at 57 kHz. The decoder comprises of a carrier-recovery stage, a symbol clock recovery stage, a filtering stage and a differential decoding sage. The decoded bits are forwarded to a Xilinx Microblaze microcontroller to post-process the data and extract the payload information.

The presented blocks can be used in different configurations to satisfy certain functional requirements with different amounts of resources. Five different FM receiver configurations have been defined in total, using either all or only a subset the presented processing blocks as shown in Table 2.1.

The signal flow graph of the FM receiver is drawn in Figure 2.3. The blocks belonging to a certain configuration subset are highlighted in blue. Before the design and implementation of the receiver components will be described in detail, it is important to mention the sample rates used in the receiver.

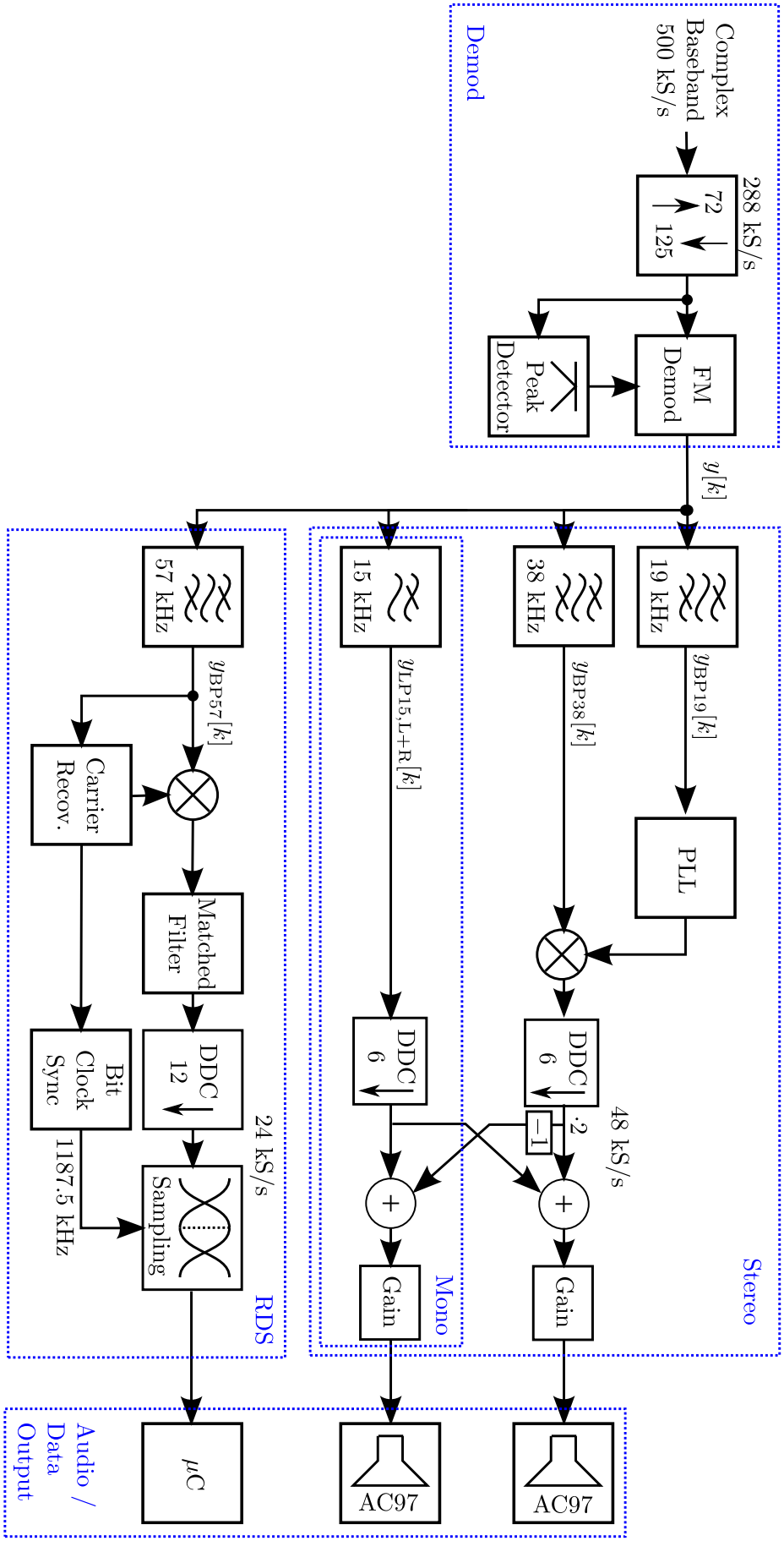


Figure 2.3: FPGA-based FM/RDS receiver signal flow-graph.

FM receiver configuration	Processing block
Demod + Stereo + RDS	1, 2, 3, 4
Demod + Mono + RDS	1, 2, 4
Demod + Stereo	1, 2, 3
Demod + Mono	1, 2
Demod + RDS	1, 4

Table 2.1: Overview of possible receiver configurations.

Sampling Rates and FPGA Clock Frequency

Designing a digital demodulator for a frequency-modulated signal gives the designer a certain amount of freedom when it comes to selecting the demodulation sample rates. The higher the sample rate, the more signal energy can be used in the demodulation process, but the higher the computational complexity. In turn, lower sampling rates reduce the computational burden, at the cost of a worse signal-to-noise and distortion ratio (SINAD) at the FM demodulator output (cf. [Ros89]). However, in the presence of strong noise or adjacent channel interference at the FM demodulator input, narrowband demodulation can increase the SINAD at the FM demodulator input, such that a higher sampling rate system with a narrow pre-filter performs similar to a system with lower sampling rate. Hence, in channels with negligible co-channel interference and noise, selecting the sampling rate and the FM demodulation bandwidth is a trade-off between complexity and error performance. For the FM receiver prototype presented in this work, the input sample rate of the FM demodulation stage has been selected according to the following criteria:

Criterion 1: The sampling frequency must satisfy the Nyquist constrains for the complex FM baseband signal, i.e. it must be chosen to be high enough to sample at least 90 % of the signal energy but as small as necessary to minimize the computational complexity.

Carson derived an estimate of the FM RF bandwidth for sinusoidal source signals in [Car22]. Since the spectrum of a frequency-modulated signal is infinite, the Nyquist rate must be related to Carson's constraints to involve a minimum percentage of signal energy in the demodulation process. For signal energy values of 90 % and 99 % the amount of RF signal energy intended to be used in the process of demodulation can be upper bounded by

$$\begin{aligned} B_{90\%} &= 2\,(\Delta f_{\mathrm{MAX}} + f_{\mathrm{MAX,LF}}) \leqslant f_{\mathrm{S}} \\ B_{99\%} &= 2\,(\Delta f_{\mathrm{MAX}} + 2 f_{\mathrm{MAX,LF}}) \leqslant f_{\mathrm{S}}, \end{aligned}$$

where $f_{\mathrm{MAX,LF}}$ is the highest frequency component in the unmodulated multiplex signal and f_{S} is the FM demodulation sampling rate for the high frequency (HF) baseband signal. For VHF transmissions the ITU-R BS.412-9 planning standards [Itu98] recommend a peak frequency deviation of $\Delta f_{\mathrm{MAX}} = 75\,\mathrm{kHz}$. The highest frequency component in the MPX is defined by the RDS signal bandwidth, such that $f_{\mathrm{MAX,LF}} \approx 59$ kHz. For broadcast FM, the Carson bandwidth is equal to $B_{90\%} = 268\,\mathrm{kHz}$ and $B_{99\%} = 386\,\mathrm{kHz}$, which means that the sample rate should be selected to be within this range. In this context it is important to mention that the Carson bandwidth is not related to the

sample rate required for reasonable FM baseband demodulation. Although in practice it may be beneficial if the demodulation sample rate is close to the Carson bandwidth, some demodulation algorithms require rates above $B_{90\%}$ or $B_{99\%}$ to ensure distortion-free signal decoding as stated in [Ros89].

Criterion 2: Sample rates that are a rational fraction of the RF input sample rate and the audio output sample rate are preferable, in order to simplify interpolation and decimation.

The Audio Codec 97 (AC97) on the FPGA board uses a sampling rate of 48 kHz and the FM demodulation sampling rate has been defined as an integer factor of the audio output sample rate. Thus, the HF sampling frequency can be selected between 288 kHz, 336 kHz or 384 kHz. Due to the fact that a sample rate closer to a bandwidth of $B_{90\%}$ reduces adjacent channel interference, a sample rate of $f_S = 288\,\text{kHz}$ has been selected as FM demodulation rate.

Prior to demodulation, a sample rate conversion module is required to convert the RF input sample rate of 500 kHz to the FM demodulation sample rate of $f_S = 288\,\text{kHz}$. Before resampling, a low-pass filter with a bandwidth of $\approx 280\,\text{kHz}$ is applied to avoid aliasing during down-sampling. In hardware, the sample rate conversion stage has been realized by a combined fractional rate polyphase resampling and filtering implementation as presented in [PM06].

Sampling Rate	Receiver Stage	FPGA cycles/sample
500 kS/s	RF front-end	72
288 kS/s	FM demodulation and MPX pre-filtering	125
48 kS/s	AC97 Audio output	750
24 kS/s	RDS sampling output	1500

Table 2.2: Receiver sampling rates at 36 MHz FPGA clock frequency.

From the sampling rates listed in Table 2.2 an FPGA clock frequency of 36 MHz has been determined as the least common multiple of the RF sample rate (500 kHz) and demodulation sample rate (288 kHz).

As already stated, the goal of the implementation was designing a modularized receiver system for self-adaptive dynamic partial FPGA reconfiguration. Hence, the algorithms have been selected to be reasonably efficient and state-of-the art, but apart from minor optimizations, the workflow was neither focused on building a high-quality receiver system, nor on finding a highly optimized design.

Fixed-Point Number Format

For most parts of the receiver chain, a fixed-point representation in 16.14 two's complement notation has been used, such that a 16 bit value comprises of 2 integer bits and 14 fractional bits, i.e. $1 \mathrel{\widehat{=}} 2^{14}$ and $-1 \mathrel{\widehat{=}} -2^{14}$. Using this notation, signals within the range of -2 to $1.999\ldots$ can accurately be represented with a dynamic range of $20 \cdot \log_{10}(2^{-14}) \approx -84.3\,\text{dB}$ full-scale (dBFS), where 0 dBFS refers to a value of 1.

In the next sections the receiver components and their respective FPGA implementation will be discussed and compared to the state of the art.

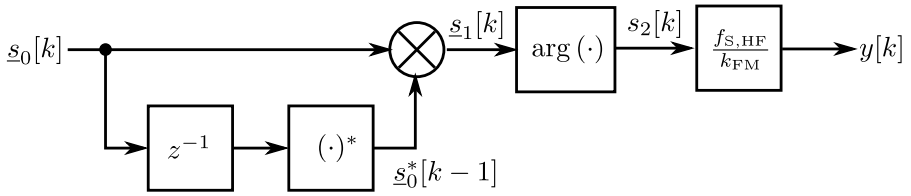


Figure 2.4: Arcus-tangent quadrature discriminator for FM demodulation.

FM Demodulation

Demodulation of the FM signal requires an accurate estimate of the instantaneous frequency of the received signal by calculating the time derivative of the phase. Various methods for digital FM demodulation have been proposed in literature (cf. [Ros89]). In this section, two popular demodulation principles will be presented together with a discussion of their properties and performances. Further on, the normalized complex-valued demodulator input signal will be denoted by

$$\underline{s}_0[k] = \frac{\underline{r}_{\mathrm{HF}}[k]}{\sqrt{\Re\{\underline{r}_{\mathrm{HF}}[k]\}^2 + \Im\{\underline{r}_{\mathrm{HF}}[k]\}^2}},$$

where $\underline{r}_{\mathrm{HF}}[k]$ is the complex baseband FM receive signal and $k \in \mathbb{N}$ denotes the sample index. The phase-derivative operation of an FM demodulator can well be depicted using the arcus-tangent *quadrature discriminator*. This forward-discriminator evaluates the phase difference of two consecutive complex samples to estimate the temporal derivative of the phase angle (cf. Equation 2.4). One can observe from the schematic in Figure 2.4 that the circuit requires one complex multiplication, one real multiplication and an arcus tangent look-up per input sample to generate one output sample.

The discriminator uses a forward structure to achieve a sample-wise differentiation of the phase by a complex conjugate multiplication. The mathematical relationship between the complex conjugate multiplication and the estimation of the phase difference is given by

$$\underline{s}_1[k] = \underline{s}_0[k] \cdot \underline{s}_0^*[k-1] = \mathrm{e}^{\mathrm{j}\phi[k]} \cdot \mathrm{e}^{-\mathrm{j}\phi[k-1]} = \mathrm{e}^{\mathrm{j}(\phi[k]-\phi[k-1])}$$

The signal at the output of the discriminator $s_2[k]$ is an estimate of the phase derivative, scaled by the sampling interval $T_{\mathrm{S}} = 1/f_{\mathrm{S}}$, as formulated in Equation 2.6.

$$\begin{aligned} s_2[k] &= \arg\left(\underline{s}_1[k]\right) = \phi[k] - \phi[k-1] = \Delta\phi(t) \\ &= \frac{\Delta\phi(t)}{T_{\mathrm{S}}} \cdot T_{\mathrm{S}} \\ &\approx \frac{d\phi(t)}{dt} \cdot T_{\mathrm{S}} \end{aligned} \tag{2.6}$$

The output of the discriminator is normalized by a scaling constant k_{FM} to project the output to the desired interval, e.g. $k_{\mathrm{FM}} = 2\pi$ clamps the output to values of $-1 \leqslant y[k] \leqslant 1$, i.e.

$$y[k] = \frac{\arg\left(\underline{s}_1[k]\right)}{T_{\mathrm{S}} \cdot k_{\mathrm{FM}}}.$$

The quadrature discriminator is suboptimal as it is prone to amplitude modulation of the FM signal, degrading the SNR at the demodulator output (cf. [Ros89]). Instead of using a forward demodulator, a feed-back structure can be used to reduce the AM distortion. A common principle for phase estimation is known in literature as PLL. A digital variant of the PLL is the digital phase-locked loop (DPLL), which operates on the same principles as its analog counterpart, i.e. it steadily minimizes the error between a reference signal and a feedback signal using a certain error metric. In the case of FM demodulation, the error signal is generated using a phase differentiation operation. This error signal is cascaded to a low-pass filter (loop filter) for image rejection and to increase the performance in the presence of noise. The error signal of the PLL is used to generate a feedback signal to continuously minimize the error.

Due to its computational simplicity, the PLL is widely used in communication systems for instantaneous frequency estimation. In [RPN09] Rice compared the resource utilization of different FM demodulator implementations for a Xilinx Virtex-4 FPGA and came to the conclusion that the DPLL is the most resource-efficient implementation. Regarding detector performance, the simplicity of the DPLL comes at the cost that it is not the optimum detector in terms of minimum mean square error performance in Gaussian noise channels, as shown by Boashash in [Boa92].

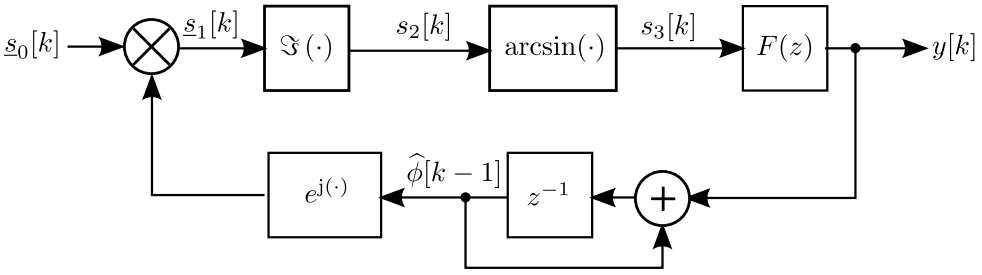


Figure 2.5: FM digital PLL discriminator signal flow.

The DPLL FM demodulation circuit is shown in Figure 2.5. The complex-valued input signal is multiplied by a phasor signal from the PLL feedback loop, which performs the discrimination of the phase as follows:

$$\underline{s}_1[k] = \underline{s}_0[k] \cdot \mathrm{e}^{-\mathrm{j}\widehat{\phi}[k-1]} = \mathrm{e}^{\mathrm{j}(\phi[k]-\widehat{\phi}[k-1])}.$$

The phase difference information is included in the real and imaginary parts of the signal. The imaginary part yields the signal $s_2[k]$, with

$$s_2[k] = \Im\{\underline{s}_1[k]\} = \sin(\phi[k] - \widehat{\phi}[k-1]),$$

where $-1 \leqslant s_2[k] \leqslant 1$. The arcsin function linearizes the output of the DPLL and the phase difference is obtained. The difference signal is then fed to a loop filter with transfer function $F(z)$ and order $K-1$. The DPLL has the order of K, which means that for a first-order DPLL the loop filter reduces to a simple gain value, i.e.

$$s_3[k] = \arcsin(s_2[k]) = \phi[k] - \widehat{\phi}[k-1] \approx \frac{d\phi(t)}{dt} \cdot T_\mathrm{S}.$$

An estimate of the phase is obtained by the integration of the loop output in the feedback path, such that

$$\widehat{\phi}[k] = \widehat{\phi}[k-1] + \frac{f_\mathrm{S}}{k_\mathrm{FM}} \cdot y[k].$$

The loop filter design is important for the DPLL to function in a noisy environment. By adjusting the loop filter transfer function, the SNR threshold performance of a DPLL can be improved by 2-3 dB compared to the previously introduced quadrature discriminator. Furthermore, the higher the loop filter order, the more degrees of freedom in designing for robustness. However, it is shown in [Ros89] that even a second order DPLL performs only 0.4 dB worse than higher order filters and in case of a first order DPLL, the difference is at maximum 1 dB.

The different output signals of the forward part of the DPLL are compared in Figure 2.6. Comparing the signals $s_2[k]$ and $s_3[k]$ indicates that in case of small phase differences the arcsin-operation is not necessary to recover the instantaneous frequency. In this case, the imaginary part of $\underline{s}_1[k]$ yields a sufficient approximation of the PLL output signal $y[k]$ [Ros89]. This leads to a considerable simplification of the hardware implementation since the PLL can be implemented without trigonometric calculation routines. Furthermore, using the sinusoidal instead of the linear phase differences a non-linear low-pass filtering of the output signal is implicit. Hence, in a computational efficient setup, the loop filter could also be omitted. The computational complexity of the modified loop is approximately equal to the previously presented forward FM demodulator (cf. Figure 2.4).

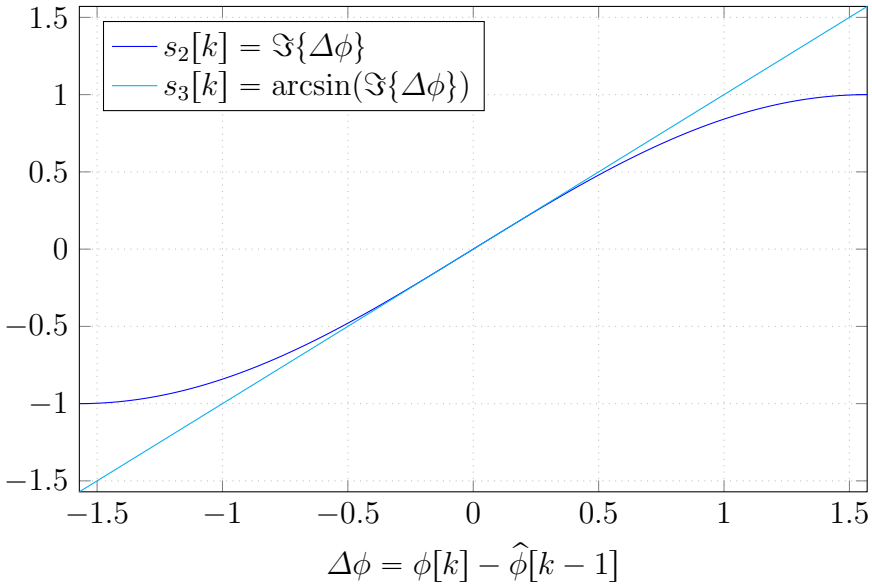


Figure 2.6: Digial PLL output signal and intermediate signal.

The FPGA implementation of the DPLL FM demodulator is shown in Figure 2.7. It is similar to the DPLL discriminator shown in Figure 2.5 including the previously stated simplifications. A single multiplexed DSP48 slice has been used for complex multiplication of the input signal with the estimated phasors using a two step multiply-accumulate cycle. The complex multiplication phasors have been derived using a Xilinx direct digital synthesis (DDS) intellectual property (IP) core. As the loop filter is zero

order, just the gain stage k_{FM} is present. The scheduling is accomplished by an FSM inside the demodulation subsystem of the Xilinx System Generator design.

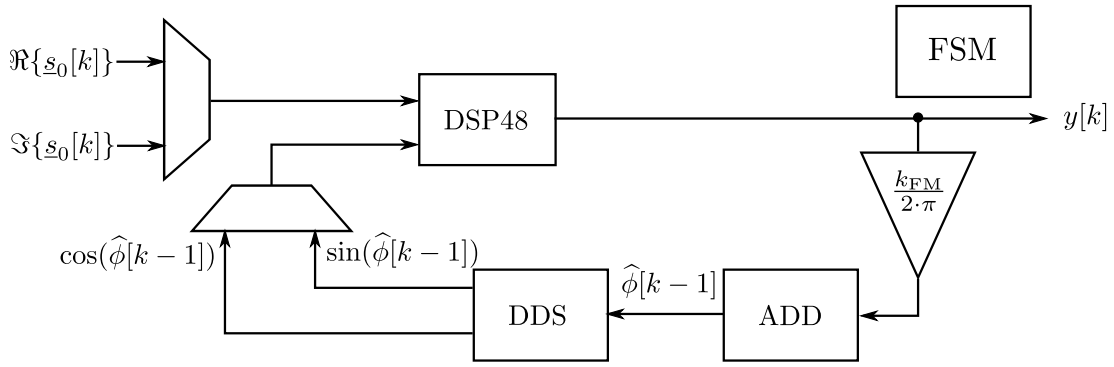


Figure 2.7: Digital PLL hardware implementation.

The demodulated FM multiplex signal is fed to the audio and RDS decoder chains, where it is post-processed for audio and data extraction. It is important to remark that all post-processing units use separate filtering and modulation stages as depicted by the architecture diagram of the receiver (cf. Figure 2.3). This means that low-pass and band-pass filtering for the extraction of the different spectral components is performed in the MPX signal domain, followed by down-conversion and filtering stages. Albeit easy to implement, filtering and multiplying the narrow-band FM components on the MPX sampling frequency is not efficient. A more efficient approach would use combined filtering and decimation on the output sampling frequency, i.e. 48 kHz, which has not been implemented in the presented FPGA receiver. However, combined decimation and filtering has been used to efficiently extract the audio information.

MPX Pre-Filtering

Prior to further processing, low-pass and band-pass filters have to be applied to the different MPX signal components. The narrow gaps between the different spectra of the different MPX components require a filter transition bandwidth of $\approx$ 4 kHz. Given a fixed-point representation of 16 bits, the stop-band attenuation has been chosen to be in the order of 70 dBfs. Meeting these requirements at the MPX sampling rate of 288 kS/s requires to use filter of high order as summarized in Table 2.3.

Signal	$y_{\mathrm{LP15,L+R}}[k]$	$y_{\mathrm{BP19}}[k]$	$y_{\mathrm{BP38}}[k]$	$y_{\mathrm{BP57}}[k]$
Filter type	FIR low-pass	FIR band-pass	FIR band-pass	FIR band-pass
Filter order	124	124	124	124
Passband / kHz	0-13.5	18.9-19.1	23.7-52.3	55.2-58.8, 63-144
Stopband / kHz	19-144	0-14.8, 23.2-144	0-18.5, 56-144	0-51, 63-144

Table 2.3: FM multiplex signal filter design parameters.

.

Given a resolution of 16 bit per filter coefficient and 125 coefficients 2 kbit of memory are required per filter. Although linear-phase FIR filters require more hardware resources

for the presented application, they have been preferred over the use of infinite impulse response (IIR) filters to avoid non-linear group-delays. Given an FPGA clock frequency of 36 MHz, a direct-form FIR filter with 125 coefficients requires one single DSP48 unit per MPX sample for the multiply-accumulate operation to finish. Hence, four DSP units are sufficient to filter all four branches in real-time.

Monaural Audio Decoder Design

Extracting the sum signal can be accomplished by subsequently decimating the monaural audio channel to a sampling rate of 48 kHz. Both operations can be combined by using an FIR structure as shown in Figure 2.8. Every f_S clock cycle one sample of the pre-filtered audio sum signal $y_{\mathrm{LP15,L+R}}[k]$ is fed to the combined filter. The sum signal is then post-filtered with another low-pass filter and sub-sampled at the audio output rate of 48 kHz, where the signal $y_{\mathrm{L+R}}[k]$ is obtained.

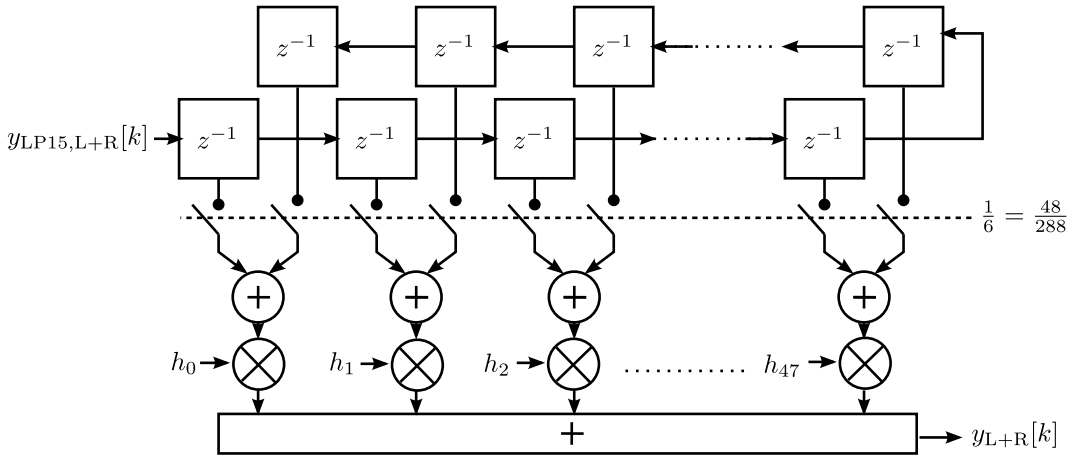


Figure 2.8: Combined filtering and decimation for FM audio signal extraction.

The FIR decimator is switched at a rate of 1/6-th and uses 96 coefficients. As the coefficients are symmetric only 48 multiplications are required per output sample. The de-emphasis filter was not implemented as part of the audio decoder and will therefore not be discussed in this work.

In the next section, the extraction of the stereo difference signal and the derivation of the left and right audio components will be described.

Stereo Difference Signal Decoder Design

The audio difference signal is centered around 38 kHz in the FM MPX with suppressed carrier. For coherent demodulation the carrier can be regenerated by the 19 kHz pilot tone. To accomplish this task, a sinusoidal with two times the angular frequency of the pilot signal must be generated. Furthermore, the generated tone must have a constant amplitude and be in-phase with the reference signal. In literature, different methods for carrier regeneration have been described (cf. [Kam08]). As discussed in the FM demodulation section, a DPLL is a suitable candidate for the estimation of the angular frequency as the output of the DPLL control loop is phase-locked to the input. The signal flow of a DPLL for angular-phase estimation is shown in Figure 2.9.

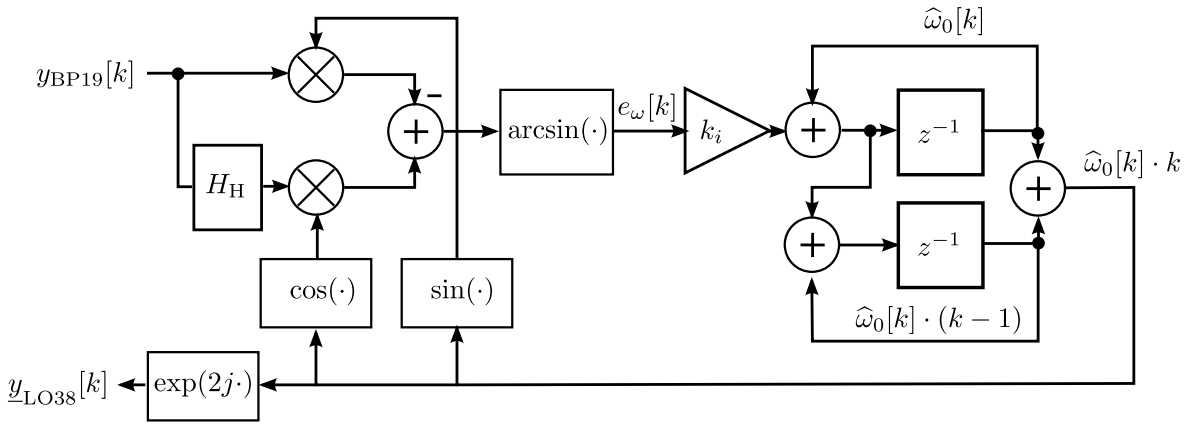


Figure 2.9: MPX pilot DPLL angular frequency estimator.

The band-pass filtered real-valued continuous wave pilot tone $y_{\mathrm{BP19}}[k]$ with a frequency of $f_0 = 19\,\mathrm{kHz}$ and an angular frequency of $\omega_0 = \frac{2\pi}{f_\mathrm{S}} \cdot f_0$ is fed to the PLL. Given a noiseless MPX signal, this tone can be modeled by

$$y_{\mathrm{BP19}}[k] = \cos(\omega_0 \cdot k). \tag{2.7}$$

If there is residual noise on the band-pass filtered MPX signal, the input signal exhibits an additional additive noise term. Due to the fact that the receiver oscillator is not perfectly synchronized to the oscillator of the transmitter, the angular frequency at the receiver deviates from the transmitted value. Since the deviations are assumed to be small, the angular frequency is known to be roughly $\omega_0 \approx \frac{2\pi}{f_\mathrm{S}} \cdot 19\,\mathrm{kHz}$. However, a more precise estimate of the angular frequency is required to coherently down-convert the stereo difference signal. At the input of the estimation circuit, a Hilbert filter with transfer function H_H is used to generate a complex-valued pilot tone signal, which is then fed to a phase comparator. The comparator is equal to that of the FM demodulator, where the imaginary part of the complex multiplication is taken and the arcsin-function is used to calculated the phase error $e_\omega[k]$ within each sampling interval according to Equation 2.8.

$$\begin{aligned} e_\omega[k] &= \arcsin\left[\Im\left\{\mathrm{e}^{\mathrm{j}\omega_0 k} \cdot \mathrm{e}^{-\mathrm{j}\widehat{\omega}_0[k]k}\right\}\right] \\ &= (\omega_0 - \widehat{\omega}_0[k]) \cdot k \end{aligned} \tag{2.8}$$

The error signal reflects the phase difference of the estimated phase and the phase of the pilot tone. The signal is amplified or attenuated by a gain constant k_i and the result is fed to two concatenated integration stages. The gain constant k_i is used to trade-off the noise robustness against the PLL lock time. The first integration stage accumulates the error signal to obtain an estimate of the angular phase, denoted by $\widehat{\omega}_0[k]$. In case the PLL is locked, the error signal approaches zero and the integrator output stays constant where

$$\widehat{\omega}_0[k] \approx \omega_0,$$

for sufficiently large values of k. The second integrator acts as a phase accumulator, where the estimated angular phase is used to generate the output signal $\widehat{\omega}_0[k] \cdot (k-1)$. In the next phase-comparison cycle at the input of the PLL, the estimated phase angle

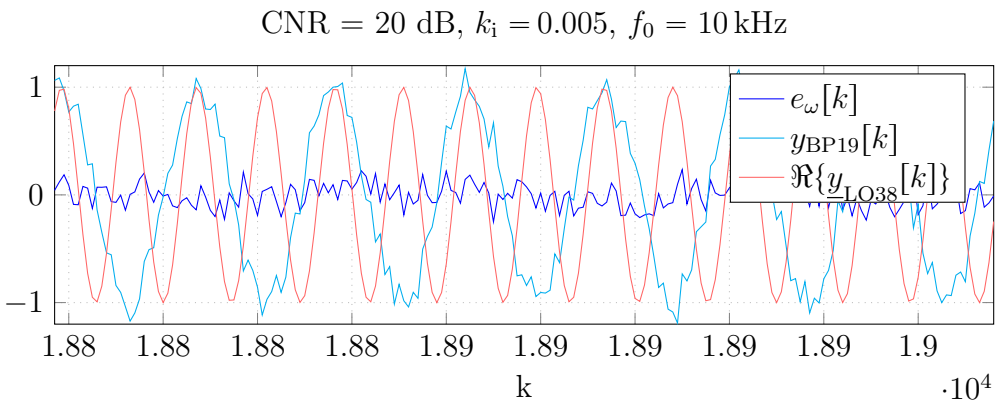


Figure 2.10: Pilot PLL input and output signals with additive white Gaussian noise.

is used to generate a new instance of the error signal. Figure 2.10 shows the steady-state carrier-to-noise ratio (CNR) performance of the PLL in the presence of additive white Gaussian noise (AWGN). An oscillator frequency of 10 kHz has been chosen for better visualization at a sample rate of 288 kHz. The plot shows that the PLL is able to generate a clean carrier output signal at twice the input angular frequency, even of the MPX pilot carrier is noisy. The complex-valued PLL output signal $\underline{y}_{\text{LO38}}[k]$ is used to coherently down-convert the band-pass filtered difference signal such that

$$y_{\text{LP15,L}-\text{R}}[k] = \Re\{\underline{y}_{\text{LO38}}[k] \cdot y_{\text{BP38}}[k]\}.$$

Then, another filtering and decimation stage equal to that in the monaural output chain is used to obtain the difference signal $y_{\text{L-R}}[k]$ from $y_{\text{LP15,L}-\text{R}}[k]$ according to Figure 2.8. Finally, the band-limited sum and difference signals are added and subtracted to obtain the left and right audio channels, $y_{\text{L}}[k]$ and $y_{\text{R}}[k]$, respectively, as follows:

$$\begin{aligned} y_{\text{R}}[k] &= 0.5 \cdot (y_{\text{L+R}}[k] + y_{\text{L-R}}[k]) \\ y_{\text{L}}[k] &= 0.5 \cdot (y_{\text{L+R}}[k] - y_{\text{L-R}}[k]). \end{aligned}$$

The hardware implementation of the pilot PLL is similar to the hardware implementation of the FM demodulation PLL. It uses one DSP48 slice on the FPGA and no trigonometric operation due to the approximate nature of the arcsin(·) and its argument, cf. Figure 2.6. Due to the similarity to the FM demodulator DPLL, a hardware flow-graph of the pilot PLL will be omitted.

Having explained the extraction of the audio components, the following section will describe the implementation of the RDS demodulator.

RDS Demodulator Design

The RDS BPSK decoder implementation needs to be self-synchronizing to operate as a stand-alone DPR module. Although incoherent detection of differentiallycoded BPSK is possible, it increases the bit error rate at the output of the detector [Kam08]. For coherent demodulation, the RDS carrier and the bit-clock have to be recovered from $y_{\text{BP57}}[k]$. Afterwards, the detected bits have to be passed to a differential decoder and

a block synchronization circuit. If the receiver is synchronized, the payload bits can be extracted and interpreted. Carrier synchronization could also be employed using the 19 kHz pilot tone. Given the goal of designing a modular receiver, in this work the carrier signal is derived from the bi-phase signal itself using a PLL with squaring feedback together with a band-pass filter.

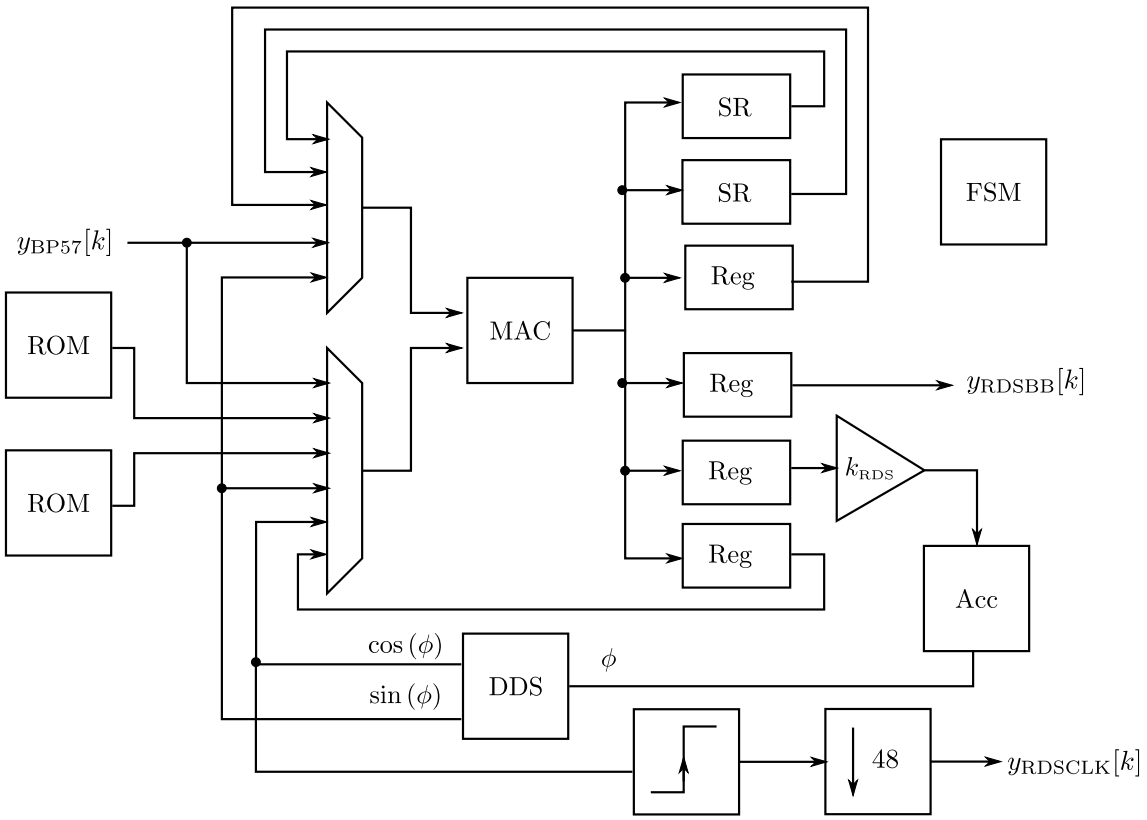


Figure 2.11: FM RDS carrier and bit-clock recovery hardware implementation.

The carrier recovery circuit is depicted in Figure 2.11. Since the FPGA clock frequency is significantly higher than the sample rate, the multiply and accumulate (MAC) unit is multiplexed within one sample cycle and used for squaring of the input signal, band-pass filtering of the squared signal, low-pass filtering of the DPLL output and for multiplication of the DPLL phasor signal. The circuit works as follows: First, one MPX input signal sample is fed to the MAC unit for squaring. The squared result is then written to an output shift register, which functions as delay line for a band-pass FIR filter to extract the second-order harmonic wave of the RDS signal. The FIR filter order is 14 and the coefficients are stored in one of the ROM tables at the MUX input and fed to the MAC unit together with the samples stored in the filter delay line. After processing, the band-pass filtered output is stored in one of the output registers. Re-using the MAC, the filtered signal is multiplied by the complex output of the DDS, which is locked to the second-order harmonic of the filtered signal. The resulting signal is then fed to the DPLL loop-filter for image rejection. For this purpose, the MAC unit is re-used again and the low-pass filter coefficients are provided by another ROM table. The order of the low-pass FIR filter is 62. The filtered output is then scaled and fed to the DPLL accumulator, which reflects the residual phase error of the DPLL. Subsequently, the error

phasor signal is fed to the DDS to generate a cosine and sine output signal at the RDS carrier frequency. The cosine DDS output line is tapped and fed to a rising-edge detector for bit-clock recovery. The resulting clock signal is down-sampled by a factor of 48 such that it can be used for bit detection in the following circuit. The filtered RDS baseband signal $y_{\mathrm{RDSBB}}[k]$ is subsequently fed to a 1023 order low-pass FIR filter and decimated by a factor of 12 to obtain the RDS baseband signal with a sampling frequency of 24 kS/s. The low-pass filter is matched to the pulse-shape of the RDS transmission signal and has been realized using a single multiplexed DSP48 slice. Performing 1023 MAC operations within a sample interval is possible since there are $\frac{36\,\mathrm{MHz}}{24\,\mathrm{kS/s}} = 1500$ clock cycles available per RDS sample. Using the derived clock signal $y_{\mathrm{RDSCLK}}[k]$, the differentially-coded RDS bits are detected by the signum function, i.e. by using the most significant bit (MSB) of the two's complement sample value. The stream of demodulated RDS bits is then forwarded to the Microblaze CPU for block synchronization and message extraction.

The complexity of the RDS decoder is depicted in sufficient detail to understand the implementation-specific demands in terms of FPGA resources. A more detailed description of the RDS detector and the recovery circuits can be found in [Str10].

RDS Demodulator Performance

In order to be able to derive meaningful SNR values for an adaptive receiver, it is important to describe the RDS detector performance in theory and in practice. For the subsequent analysis, it is assumed that the noise at the RDS demodulator input is approximately white and Gaussian. In this case, optimum detection (in the minimum mean square error sense) of the bi-phase-coded RDS bits requires a filter matched to the transmit pulse as stated in [Pro01]. In addition, Proakis states that the bit-error probability of coherent differentially-decoded BPSK $P_{b,\mathrm{coh}}$ under AWGN conditions can be calculated by

$$P_{b,\mathrm{coh}} = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \cdot \left[1 - \frac{1}{2} \cdot Q\left(\sqrt{\frac{E_b}{N_0}}\right)\right] \tag{2.9}$$

where E_b is the energy per bit, N_0 is the one-sided power spectral density of the noise at the demodulator input and $Q(\cdot)$ is the error function. In the RDS system specification, a simple rectangular low-pass filter is proposed, which is commonly referred to as integrate and dump (ID) filter. Although this filter functions as low-pass integrator, it is not optimum in terms of noise performance since the RDS transmit pulse has a square root-raised cosine transfer shape. The FIR band-pass filter in the previously described demodulator circuit has a bi-phase square-root-raised cosine transfer function and is thus matched to the RDS transmission pulse shaping. In order to decide in how far ID approaches the matched filter bound, BER simulations have been performed as outlined in Figure 2.12. The ID filter is close to the theoretical bit-error probability but, as explained, does not reach the matched filter bound.

The RDS payload data is framed into four 16 bit blocks, each one protected with a 10 bit shortened cyclic code checkword for bit-error detection and correction. The codewords are transmitted to a Xilinx Microblaze CPU in chunks of 32 bit via a memory-mapped interface connected to the processor local bus (PLB). Frame synchronization and payload decoding are done inside the CPU and upon successful decoding, the RDS information such as station label and text messages are displayed on a 16x2 character display using memory-mapped general purpose input and output (GPIO) pins. A

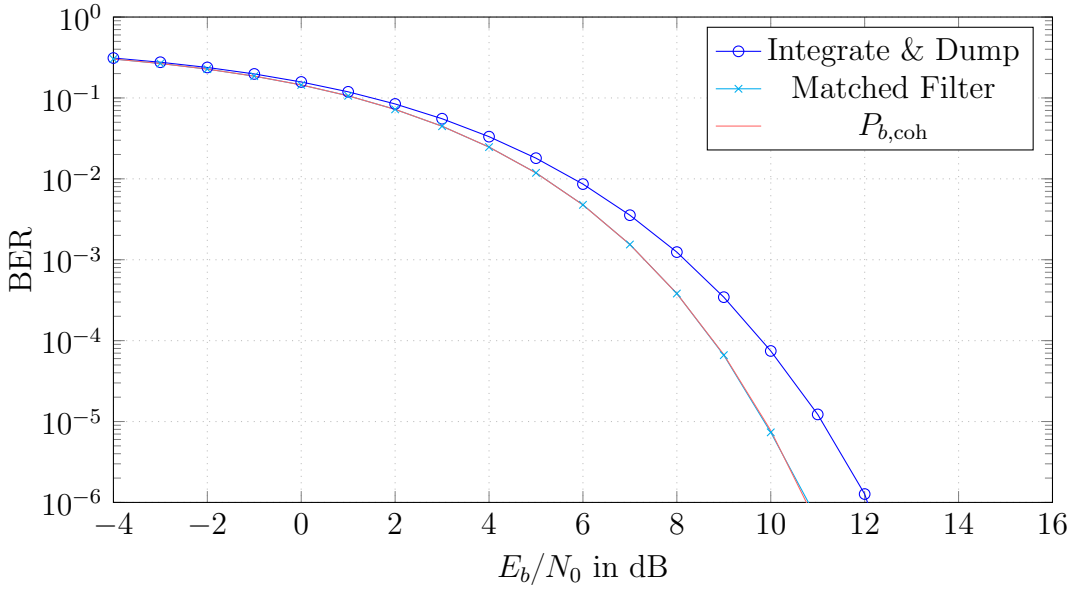


Figure 2.12: BER performance of different RDS demodulator implementations.

detailed overview of the RDS data processing chain is outlined in [Str10].

2.3.2 Synthesis and Hardware Setup

The described receiver components were developed using Xilinx System Generator version 11 with Matlab version 7.9. The Microcontroller code has been compiled using the Xilinx SDK and the Microblaze PLB FPGA system was designed using the Xilinx EDK. The Microblaze has been imported to System Generator using the EDK pcore. The final design has then been converted to very high speed integrated circuit hardware description language (VHDL) and imported to Xilinx ISE, where it has been synthesized and implemented using the toolchain version 11.4. Xilinx Plan Ahead has been used for I/O configuration and floorplanning. The generated bitstream can be loaded via JTAG using Xilinx Impact or via compact flash using the System ACE controller. Figure 2.13 visualizes the described development and implementation tool-flow.

After successful testing of the individual components, the receiver chain has been synthesized and implemented on a Xilinx Spartan-3A XtremeDSP 3400A development board. The digital baseband signal has been supplied from a PC to the FPGA using a Xilinx FMC Debug breakout board and a National Instruments data acquisition (DAQ) card with 8 bit digital I/O. The FPGA design runs fully synchronous and the clock-domain crossing happens between the PC and FPGA on the I/Q data path, i.e. the I/Q samples are provided by the first-in first-out buffer (FIFO) of the PC DAQ card and are extracted by the read clock signal of the FPGA. Thus, the FPGA functions as clock master providing a 1 MHz sampling clock to the DAQ card (IQ_CLK). On every rising clock edge, one FM baseband sample is clocked out of the DAQ card FIFO in an interleaved I/Q stream format. The values in the I/Q stream are represented in 7 bit two's complement integer notation (IQ_DATA) and streamed with a sampling rate of 500 kS/s. The MSB of the 8 bit signal is used as an I/Q synchronization signal (IQ_SYNC) and is high in case IQ_DATA carries an in-phase component and low in case a quadrature

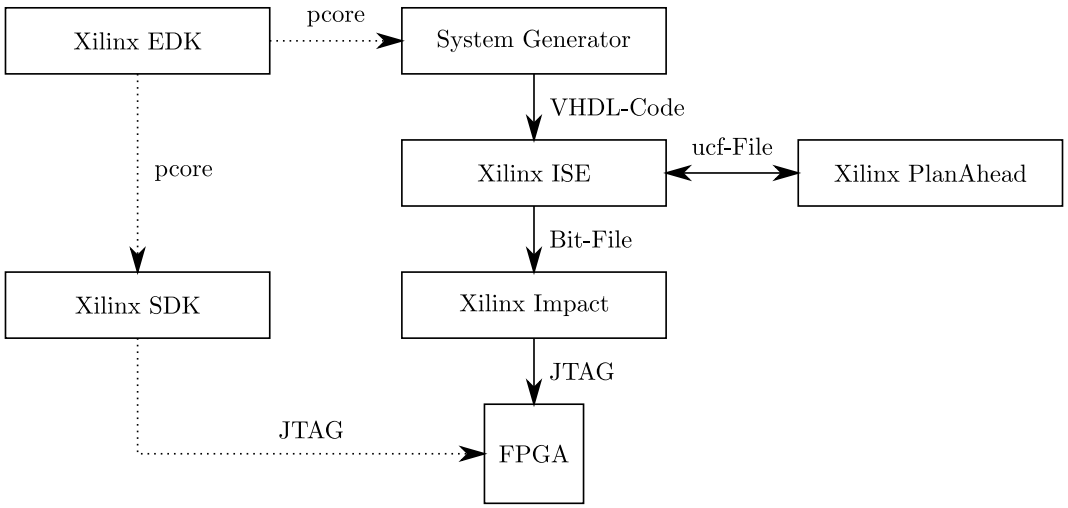


Figure 2.13: FM receiver development and implementation tool-flow.

component is signaled. The related I/O timing diagram is shown in Figure 2.14. In the diagram, the wire delays are indicated in IQ_DATA and IQ_SYNC by a small time shift relative to the edges of IQ_CLK. The Spartan-3A hardware setup and FPGA I/O schematic are shown in Figure 2.15.

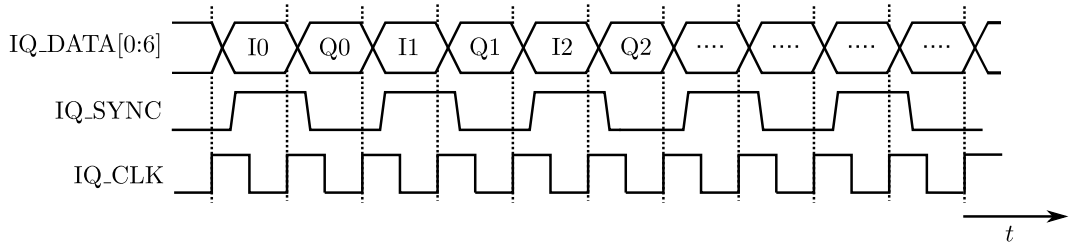


Figure 2.14: Complex FM baseband signaling and clocking.

Multiple Receiver Configurations

As already pointed out, the receiver design should follow a modularized approach to be able to re-use these modules in a DPR system. Modularization is required to obtain *independent* and *interchangeable* reconfigurable design elements. Interchangeable, because in a reconfigurable system, the number of I/O interfaces from and to the reconfigurable partition are fixed during design time. Also, to avoid signal glitches inside the FPGA fabric, the I/O gateway positioning inside the FPGA must be fixed [Xil12c], which is why different reconfigurable modules must have identical I/O port pins. Since clock administration components like BUFG, DCM, MMCM and PLL can not be used inside a reconfigurable partition, cf. [Xil12c], a common clock frequency for all modules is desirable. The modules must be independent to avoid interfering with the static partition and with other DPR modules.

For the system evaluation, five different receiver configurations have been synthesized and implemented on the Spartan-3A DSP FPGA:

- **Demod+Stereo+RDS**: All FM receiver components are present in the design.

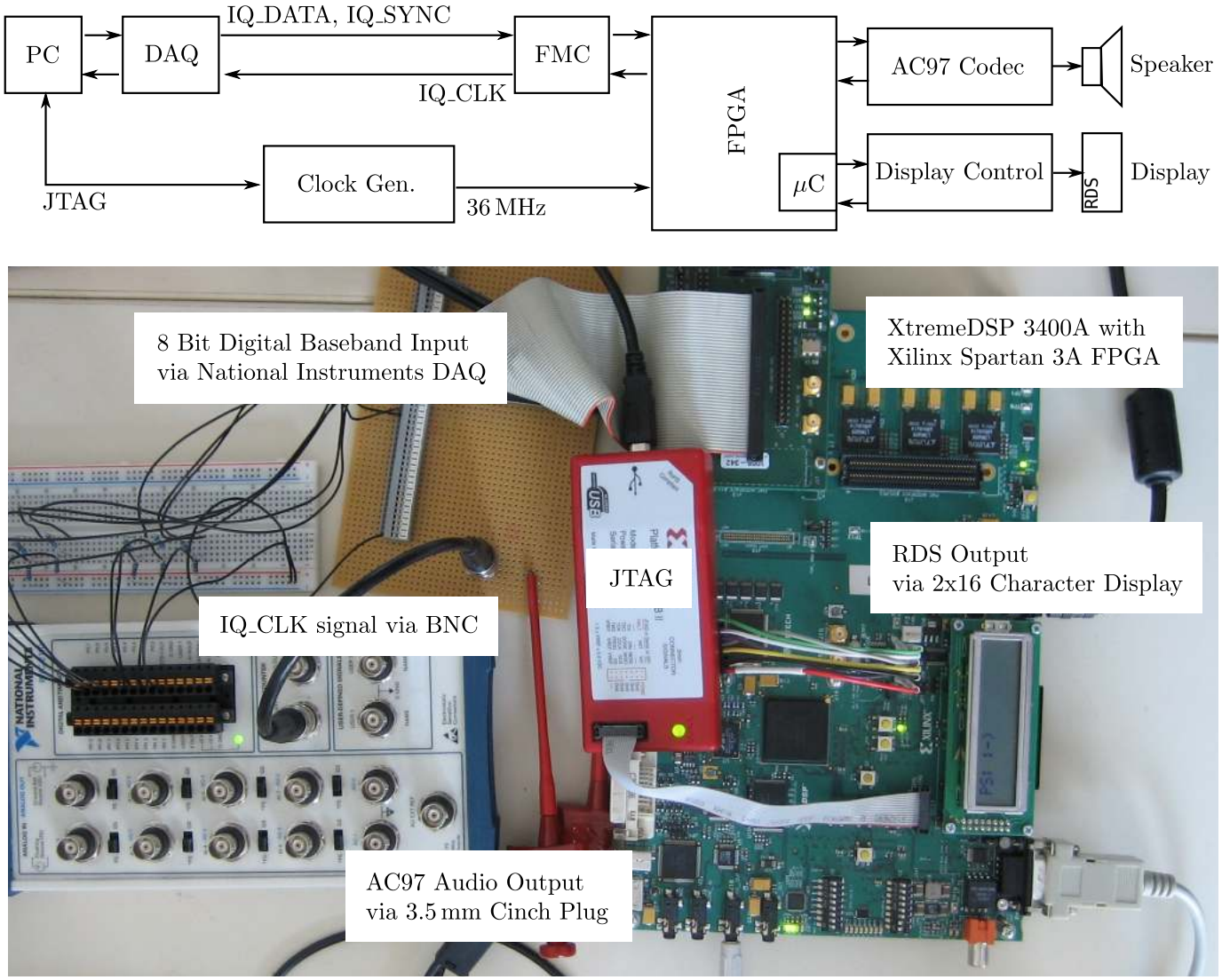


Figure 2.15: Xilinx Spartan-3A FPGA with data-flow to PC DAQ board.

Configuration	Slices	DSP48	18k BRAM
Demod + Stereo + RDS	4564	20	29
Demod + Mono + RDS	3098	13	23
Demod + Stereo	3391	14	19
Demod + Mono	2519	7	13
Demod + RDS	2068	11	16
XC3SD3400A	23872	126	126

Table 2.4: FM receiver resource consumption on Xilinx XC3SD3400A FPGA.

- **Demod+Mono+RDS**: The FM pilot PLL and stereo decoding parts are left out in the design.
- **Demod+Stereo**: The RDS processing chain is not part of the receiver.
- **Demod+Mono**: Only the FM demodulator and audio sum decoder are present in the design.
- **Demod+RDS**: Only the FM demodulator and the RDS decoder are present in the design.

In order to react on user input via GPIO, the Microblaze core has been kept as a part of the system in all configurations, even if RDS could not be provided to the processor. Keeping the Microblaze in the system also enables to use it for reconfiguration triggering in a reconfigurable setup, as described in the next section.

Table 2.4 depicts the resource requirements of the different configurations in terms of slices, DSP48 units and 18 kb BRAMs for a Spartan-3A DSP FPGA running at 36 MHz. The number of FIR filters and the high FIR filter orders (cf. Table 2.3) are responsible for most of the BRAM utilization in the design. Due to the pilot PLL and additional mixing, the stereo decoder consumes most of the DSP48 slices. In terms of Slices, the *Demod+RDS* configuration seems to be the least demanding. However, without the additional logic to control the AC97 output circuit the *Demod+Mono* configuration is of equal complexity. To get a notion of the complexity in terms of resources it is important to mention that each Spartan-3A FPGA slice contains two 4-input look-up tables and two flip-flop registers [Xil11c]. The receiver configurations presented in the table have in common, that the FM demodulator is always present, whereas the MPX decoder modules are different. For the reconfigurable FPGA demonstrator derived further on in this chapter, only the demodulated MPX signal will be of concern and the FM demodulator will be static.

In Figure 2.16 a bar graph shows the relative resource consumption of the different module configurations. In the most complex configuration, which is *Demod+Stereo+RDS*, the receiver uses about 4564 slices, resulting in a rather small device utilization of 19.1 %.

Preparing the design for DPR

Although in literature [BY08] it has been reported that partial reconfiguration of a Spartan-3A FPGA is possible, self-reconfiguration requires external I/O wiring to the

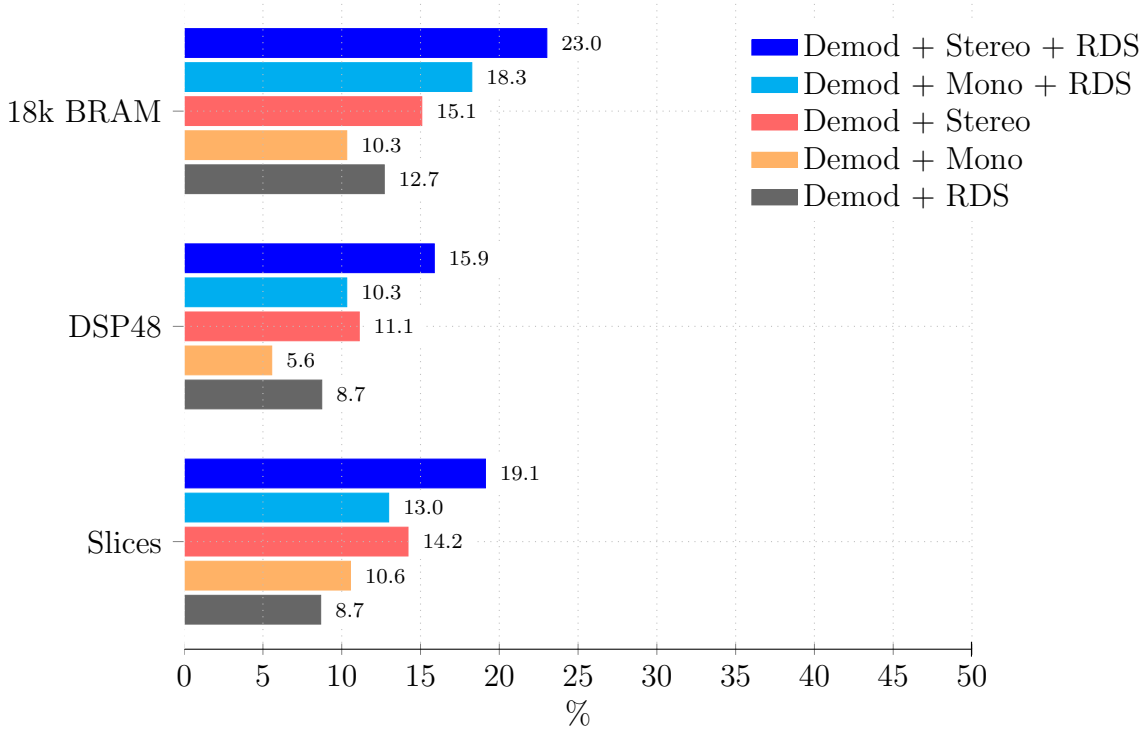


Figure 2.16: FM receiver relative resource consumption on Xilinx XC3SD3400A FPGA.

SelectMAP port. Furthermore, the Spartan-3A FPGA has not been designed for partial reconfiguration and, hence, there exists no vendor tool support. Therefore, the design was migrated to the Xilinx ML506 Virtex-5 FPGA board, which natively supports dynamic partial reconfiguration and provides an ICAP interface for internal wiring. The Virtex-5 FPGA is an XC5VSX50T with an FFG1136 package and the ML506 board is equipped with components which are mostly identical to the Spartan-3A board (cf. [Xil11b]). Most importantly, the XCCACE System Ace Interface, the XCF32 Platform Flash, the AD1981BJSTZ AC97 Codec and 2x16 character display/controller could be re-used. The FM receiver laboratory setup using the Virtex-5 FPGA board is shown in Figure 2.17. It is different from the setup shown in Figure 2.16 as instead of using a DAQ card, the baseband is supplied from the PC via the universal serial bus (USB) to a Cypress-FX2 microcontroller. The microcontroller sends the baseband data to a Xilinx Spartan-3 FPGA, which generates the previously described 8 bit digital I/O format. Inside the Spartan-3 a dual-port BRAM is used for clock-domain crossing and the digital output is connected to the input pin header of the ML506 board.

For the evaluation in a single-island reconfigurable system, three of the five module configurations have been migrated to the ML506 board, namely *Demod+Stereo*, *Demod+Mono* and *Demod+RDS*. These modules were later partitioned to obtain a more fine-grained modularization, which will be discussed in the following section. For partial-configuration support, the Microblaze design on the Spartan-3A was extended by an additional Xilinx HWICAP IP core, connected to the PLB. To increase the reconfiguration speed, the PLB and the HWICAP core were configured for a clock frequency of 100 MHz and a data width of 32 bit. Furthermore, a double data rate random-access memory (DDR-RAM) controller, a System ACE controller and a GPIO controller were added to

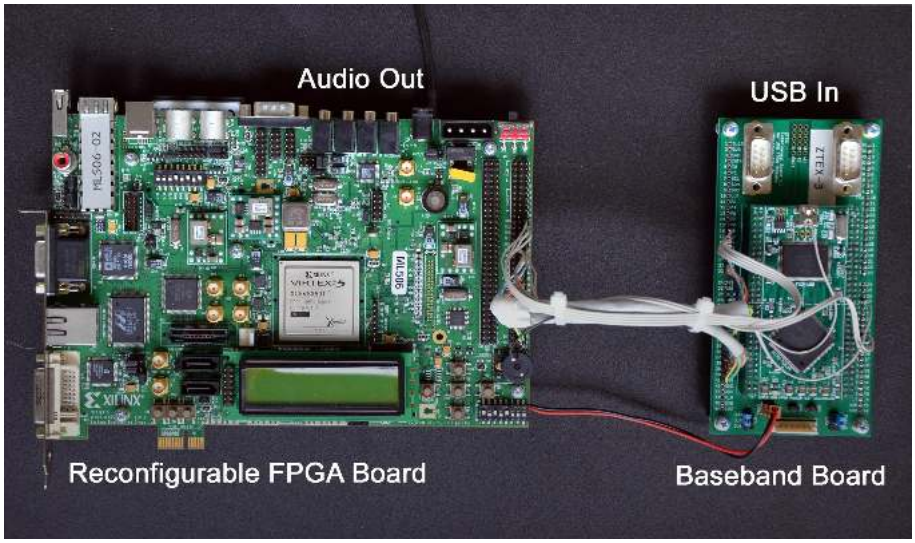


Figure 2.17: Xilinx ML506 Virtex-5 FPGA board connected to Spartan-3 USB board.

the design.

In the FPGA startup phase, the partial configuration bitstreams are buffered in the DDR2 memory to be accessible with negligible latency and high throughput during runtime. A partial reconfiguration controller based on the Xilinx EDK IP template has been used in the Virtex-5 design. It enhances the Microblaze CPU with memory mapped I/O. A DPR control unit was added to the design to enable or disable the gate registers from and to the reconfigurable partition. As explained in Section 1.1, these gate registers are necessary to decouple the logic in the static region from the reconfigurable partition. The DPR controller will also be used to reset and initialize the reconfigurable modules as subsequently discussed. Also, a more recent toolchain was used for development, namely the Xilinx System-Generator version 13, the Xilinx ISE version 13.1 and the Matlab/Simulink in version 7.10.

As outlined in the motivation of this chapter, the reconfigurable receiver should be self-adapting to the channel conditions. The realization of a self-adaptive system requires the knowledge of the signal quality and channel disturbances. Therefore, a novel estimation approach is motivated in the following section. The algorithm is based on a mean noise energy detection criterion and allows the design of a hardware-efficient SNR estimation stage for FM broadcast systems.

2.4 An MPX-based SNR Estimator for FM Radio

It can be observed that the noise power spectral density at the output of the FM demodulator is not constant but increases along with the frequency (cf. [Kam08]). Hence, the audio difference signal, which is required for stereo reception, is disturbed by a higher noise level than the audio sum signal. If the signal is weak compared to the noise, it might be better to output only the monaural signal since adding the difference signal will degrade the audio quality. In this case also the RDS decoding performance will be degraded. These circumstances motivate the design of an MPX decoder that can be adopted according to the receive signal quality, expressed by the SNR at the receiver

input. For the presented metric, threshold values can be derived, which can trigger a change in receiver complexity according to a certain constraint, for example:

- **Minimum power consumption:** Try to minimize the overall FPGA power consumption by minimizing the logic utilization and switching times, leading to a design where the allocated logic resources are kept at a minimum with respect to the minimum decoding quality requirements.
- **Maximum decoding quality:** The amount of FPGA resources occupied by the decoding algorithm is maximized in order to achieve a minimal decoding error rate, i.e. maximum SNR performance.
- **Service reliability:** The quality is balanced according to the channel conditions to optimize the service reliability for multiple receivers, e.g. to assure an acceptable error rate. In turn, also the amount of resources will be balanced among these receivers.

The choice of the optimization strategy depends on the use-case. In this work, the receiver is designed for maximum service reliability, such that the complexity of the demodulation algorithms is continuously adjusted in relation to the SNR at the input of the FM demodulator. The realization of this approach requires an SNR estimation device to be part of the receiver system. The design and FPGA implementation of an MPX-based SNR estimation algorithm for continuous operation in a reconfigurable FM receiver will in the following be presented.

2.4.1 Estimator Requirements and Restrictions

As described in the previous section, the FM demodulator is always part of the system and only the MPX decoder modules are different. Thus, in order to be independent of the FM demodulator implementation, an SNR estimation routine evaluating the MPX signal would be a preferable solution. This restriction prohibits the use of statistical methods evaluating the second and fourth order moments of the FM demodulator input signal as described in [PB00] and [XGXZCY13]. By providing the MPX signal, the SNR estimator should back-calculate the FM demodulator input SNR and trigger a change of the MPX decoder modules using partial reconfiguration. Furthermore, since in a DPR system the MPX decoders are changing, the SNR estimator should also be independent from the MPX decoder implementation. Another concern is that the estimator should be economic in terms of FPGA resources.

In summary, for the estimation scheme to be applicable, the following requirements have been formulated:

1. The estimation must be performed on the demodulator output signal to be independent from the FM demodulator implementation (analog or digital).
2. The estimation must not rely on other MPX-related implementations, e.g. it should work without evaluating the 19 kHz pilot tone or the RDS signal.
3. The estimation must be accurate enough to detect SNRs in the range between 0 dB to 30 dB with tolerable deviations.

4. The estimator must be realizable by a resource-efficient FPGA implementation.

The second requirement excludes methods based on the evaluation of the PLL tracking error variance or RDS bit-error statistics as presented in [DDHSW01] and [Trp91]. Bearing the formulated requirements in mind, an estimation approach will be proposed, which evaluates the MPX noise-energy to determine the SNR at the input of the FM demodulator. The approach is based on the fact that the noise in the FM signal can be detected after demodulation within the MPX PSD band-gaps where no audio signal energy is present. Averaging the noise power within the MPX band-gaps enables to estimate the SNR at the FM demodulator input. Since the mentioned approach was elaborated independently by Texas Instruments, it is necessarry to put it into historical context:

In December 2010, first theoretical feasibility studies of the proposed SNR estimation method were performed. In June 2011, Daniel Münch implemented the algorithm as part of his Master's Thesis with the title "Receive signal-dependent adaption of an FPGA-based Software Defined Radio" [Mü11]. The thesis was submitted to the Institute for Integrates Systems in September 2011. In his thesis, Münch presented a feasibility study and a possible hardware implementation. In 2013 Texas Instruments filed a patent describing the same estimation method, with minor implementation-specific differences (cf. [GMBV14]). In the patent, Gupta et al. utilize an FIR filter for noise power estimation, whereas in this work a resource-efficient high-order IIR resonator is used. Another difference is that the authors do neither specify the FIR filter requirements nor its complexity. The patent was published in November 2014 under the US patent publication number US 2014/0348328 A1.

In the next section, the effects of a noisy signal at the FM demodulator input are set in context to the noise at the demodulator output. Then, given the spectral characteristics of the FM MPX signal, the design of a band-gap-based noise estimator similar to [XSSK10] is proposed, that fulfills the previously formulated requirements.

2.4.2 FM Demodulation in Presence of Noise

The effects of noise on the FM demodulation process is covered in various text books and research articles. A comprehensive description of the problem has been presented by Rice in [Ric63]. Using the notation of Kammeyer (cf. [Kam08]), a brief summary on the effects of noise on the FM demodulation process will be given next. Following the convention introduced in Section 2.2, underlining indicates complex baseband notation.

In a distortion-free communication channel, where the transmitted FM signal $\underline{x}_{\mathrm{FM}}(t)$ is disturbed by zero-mean complex additive white Gaussian noise $\underline{n}(t)$, the receive signal $\underline{r}_{\mathrm{FM}}(t)$ can be described by:

$$\underline{r}_{\mathrm{FM}}(t) = \underline{x}_{\mathrm{FM}}(t) + \underline{n}(t)$$

Let the power of the FM transmit signal be defined by A_{FM}^2 and $N_0/2$ be the two-sided power spectral density of the white Gaussian noise in W/Hz. Given a typical stereophonic FM broadcast signal, the highest frequency component in the unmodulated MPX signal $f_{\mathrm{MAX,LF}}$ is generated by the RDS at a center frequency of 57 kHz. With an RDS bandwidth of approximately 3 kHz the frequency $f_{\mathrm{MAX,LF}}$ can be defined as $f_{\mathrm{MAX,LF}} \approx 60\,\mathrm{kHz}$ [SS08]. Using the bandwidth rule of Carson, as introduced in the

previous section, the carrier-to-noise ratio of the FM receive signal can be defined by

$$\rho_{\mathrm{FM}} = \frac{A_{\mathrm{FM}}^2}{N_0 \cdot 2(\Delta f_{\mathrm{MAX}} + 2f_{\mathrm{MAX,LF}})}. \tag{2.10}$$

The expression in Equation 2.10 defines a relation of signal and noise power and thus depends on the signal bandwidth term in the denominator. In the following it will be shown how to use the expression in Equation 2.10 in a real-world receiver.

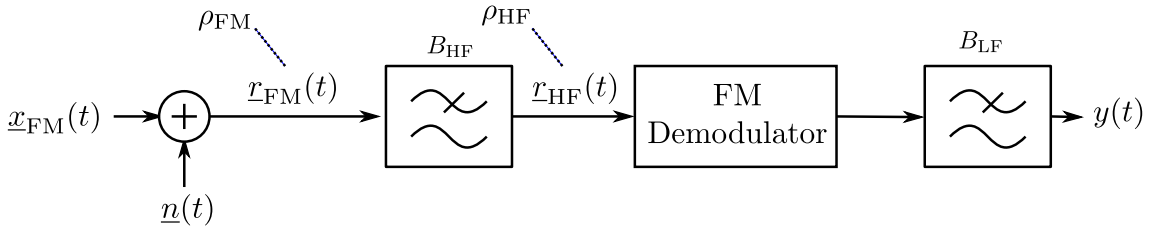


Figure 2.18: FM demodulation in presence of AWGN.

Figure 2.18 shows an FM demodulator flow-graph of a receiver with a low-pass filter in front of the FM demodulator and a low-pass filter at the output of the FM demodulator. The HF filter is required for noise suppression and adjacent channel rejection. Since in theory the FM signal is infinite in bandwidth, a portion of signal spectral energy is lost during the HF filtering process. This band-limitation is not necessarily harmful as in channels where N_0 is close to the FM signal PSD, the variation of the HF filter bandwidth can enhance the SNR at low-frequencies of the demodulator output (cf. [Kam08]). An exact quantification of the influences of the filtering process is difficult and requires the knowledge of the noise power and the HF filter characteristics. Therefore, in the following analysis it is assumed that the HF filter bandwidth is fixed and equal to the sampling frequency of the receiver, which had been chosen to comply with the Carson's banwidth rule as described in the receiver design Section 2.3, such that

$$f_{\mathrm{S}} = B_{\mathrm{HF}}.$$

The signal-to-noise ratio at the output of the HF filter is defined by ρ_{HF}, which will further on be used as an indication for the demodulation quality. Given the formulated assumptions of using a fixed HF filter bandwidth close to the Carson bandwidth, the signal-to-noise ratio at the demodulator input approaches

$$\rho_{\mathrm{HF}} \approx \rho_{\mathrm{FM}}. \tag{2.11}$$

In a digital receiver, the HF filter could also be interpreted as a band-limiter before analog-to-digital conversion, where the filter bandwidth would determine the minimum frequency required for sampling. To emphasize this relation, the HF filter bandwidth B_{HF} is defined to be double-sided and not time-variant, which creates an analogy to the required sampling frequency satisfying the Nyquist-Shannon theorem. For the following theoretical analysis, low-pass filtering is assumed to be ideal, i.e. both filters have a linear phase response and a rectangular spectral shape.

According to Equation 2.4, the LF signal $y(t)$ at the output of the FM demodulator is obtained by differentiating the phase of the filtered FM signal. Since the presence of

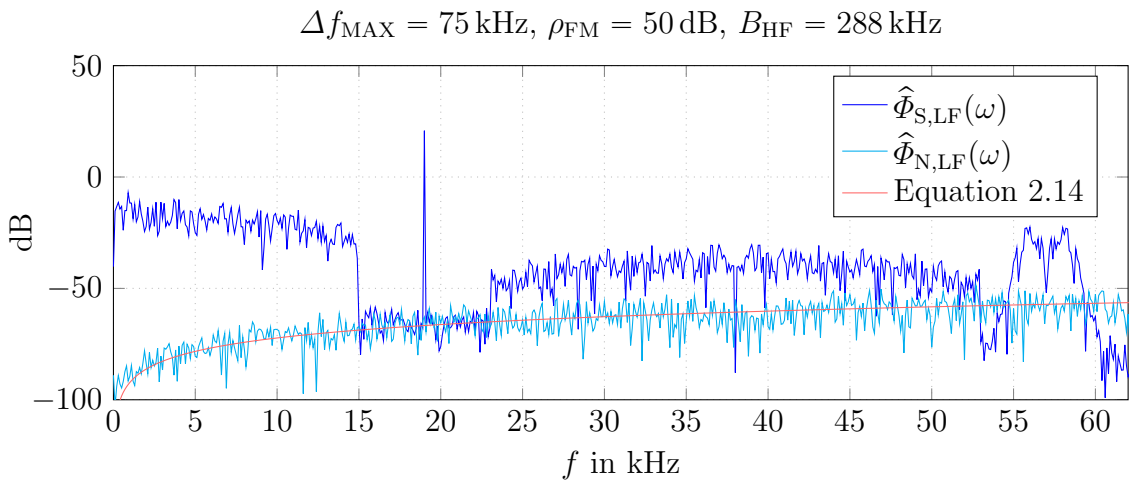


Figure 2.19: Simulation and theory of MPX signal and noise PSD.

.

AWGN adds an error phasor to the unit phasor, the phase differentiation stage, i.e. the FM demodulator, returns an erroneous LF signal. For further analysis the demodulation is assumed to be performed by a differential FM discriminator as stated in Equation 2.12 and as formulated by Kammeyer in [Kam08].

$$y(t) = \Im\left\{\frac{d\underline{r}_{\mathrm{HF}}(t)/dt}{\underline{r}_{\mathrm{HF}}(t)}\right\} \tag{2.12}$$

The demodulated signal is then processed by an ideal low-pass filter with transfer function $H_{\mathrm{LF}}(f)$ to obtain the filtered output signal $y(t)$.

$$H_{\mathrm{LF}}(f) = \begin{cases} 1, & \text{if} \quad -B_{\mathrm{LF}} \leqslant f \leqslant B_{\mathrm{LF}} \\ 0, & \text{else} \end{cases}, \tag{2.13}$$

where $B_{\mathrm{LF}} = f_{\mathrm{MAX,LF}}$. Given the assumption that the signal power of the transmitted signal is greater than the noise power, according to Kammeyer the differentiation of the phase error leads to a quadratic shaping of the noise power spectral density at the demodulator output. Kammeyer furthermore mentions that the coloring of the noise term can be approximated by a frequency-dependend density function, such that

$$\Phi_{\mathrm{N,LF}}(\omega) = \frac{N_0}{2A_{\mathrm{FM}}^2} \cdot \omega^2 \tag{2.14}$$

where $N_0/2$ is the two-sided power spectral density of the noise at the demodulator input. The presented linear approximation is accurate as long as the FM receive signal to noise ratio is above the FM threshold. The integration of the noise PSD within the LF bandwidth $f_{\mathrm{MAX,LF}}$ then gives the mean noise power

$$N_{\mathrm{MPX}} = \frac{1}{2\pi} \cdot 2 \int_0^{f_{\mathrm{MAX,LF}}} \frac{N_0}{2A_{\mathrm{FM}}^2} \cdot \omega^2 d\omega = \frac{2}{3}(2\pi)^2 \frac{N_0}{2A_{\mathrm{FM}}^2} f_{\mathrm{MAX,LF}}^3, \tag{2.15}$$

as also derived in [Kam08]. Figure 2.19 shows the right-sided theoretical PSD stated in Equation 2.14 together with the simulated noise power spectral density $\widehat{\Phi}_{\mathrm{N,LF}}(\omega)$ and the MPX signal power spectrum $\widehat{\Phi}_{\mathrm{S,LF}}(\omega)$. The simulations were conducted within a time frame of 1 second with a peak frequency deviation of $\Delta f_{\mathrm{MAX}} = 75\,\mathrm{kHz}$, a carrier-to-noise ratio of $\rho_{\mathrm{HF}} = 50\,\mathrm{dB}$ and an HF bandwidth of $B_{\mathrm{HF}} = 288\,\mathrm{kHz}$. The slopes of the curves confirm that within the MPX bandwidth of $f_{\mathrm{MAX,LF}} \approx 60\,\mathrm{kHz}$, the theoretical line follows the simulated values.

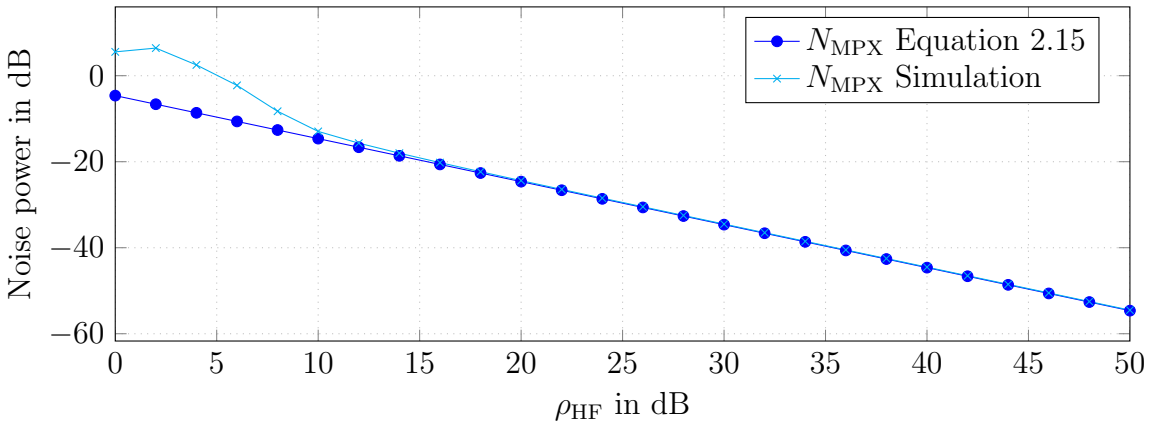


Figure 2.20: MPX noise power in relation to FM signal-to-noise ratio.

In Figure 2.20, the accumulated noise power inside the MPX signal is compared against the theory as stated in Equation 2.15 for $f_{\mathrm{MAX,LF}} = 60\,\mathrm{kHz}$ and a carrier-to-noise range between $0 \leqslant \rho_{\mathrm{HF}} \leqslant 50\,\mathrm{dB}$. The plot shows a good match between simulation and theory for carrier-to-noise ratios of 10 dB and higher. The FM threshold is responsible for the degradation of the estimation accuracy when $\rho_{\mathrm{HF}} \leqslant 10\,\mathrm{dB}$ (cf. [Ric63]), such that the approximation stated in Equation 2.14 does not describe the noise influences well enough anymore. Above that region, the estimated values closely approach the real MPX noise power. This motivates the idea to estimate the noise power within the MPX signal band-gaps and use this information as a metric for an SNR-adaptive receiver using DPR.

Further on, the design and hardware implementation of such an estimator will be presented, followed by a discussion on FPGA resources and computational complexity.

2.4.3 MPX-Based Noise Power Estimator Design

In the last section, it has been presented that the noise disturbance at the FM demodulator output increases quadratically in power with a linear increase in frequency. This might lead to a situation where the noise perturbation for high frequency signals in the MPX is too high to be decoded with an acceptable quality. Since the carrier-to-noise ratio at the demodulator input determines the noise power at the demodulator output, the SNR at the input of the FM demodulator determines whether decoding of high-frequency components in the MPX signal is feasible. Such a decision furthermore enables the design of an SNR-adaptive activation and deactivation stage for demodulator components of the receiver.

The MPX PSD shown in Figure 2.2 shows a gap of ≈ 4 kHz between the pilot tone and the audio signals, i.e. between 15 kHz and 19 kHz. Signals appearing within this

region are mainly caused by noise at the demodulator input. Furthermore, Equation 2.14 states that this noise is linearly affected by the SNR of the HF input signal. Given these relations, an efficient bandpass-filter-based noise energy estimator will be introduced.

Given an ideal band-pass filter with transfer function

$$H_{\mathrm{BP}}(\omega) = \begin{cases} 1, & \text{if} \quad \omega_a \leqslant |\omega| \leqslant \omega_b \\ 0, & \text{else,} \end{cases} \tag{2.16}$$

the mean noise power within the filter bandwidth of ω_a and ω_b can be obtained by integration of the two-sided noise PSD as stated by Equation 2.14. This relation is depicted by Equation 2.17.

$$N_{\mathrm{BP}} = \frac{1}{2\pi} \cdot 2 \int_{\omega_a}^{\omega_b} \frac{N_0}{2A_{\mathrm{FM}}^2} \cdot \omega^2 d\omega = \frac{N_0}{2\pi A_{\mathrm{FM}}^2} \cdot \frac{\omega_b^3 - \omega_a^3}{3} \tag{2.17}$$

Let the ideal bandpass filter in Equation 2.16 have the impulse response $h_{\mathrm{BP}}(t)$. According to Parseval's theorem [PM06], the mean noise power can be estimated by accumulating the energy within the band-gaps of the MPX spectrum as follows:

$$\widehat{N}_{\mathrm{BP}} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \left| y(t) * h_{\mathrm{BP}}(t) \right|^2 dt \quad t_1 > t_0, \tag{2.18}$$

where $*$ denotes the convolution operation and t_0, t_1 determine the start and end times of the averaging window. The estimated noise power $\widehat{N}_{\mathrm{BP}}$ approaches N_{BP} for $(t_1 - t_0) \to \infty$. Given the equality in Equation 2.11 and the HF SNR in Equation 2.10, the carrier-to-noise ratio at the FM demodulator input can be estimated by reformulating Equation 2.17 as follows:

$$\widehat{\rho}_{\mathrm{HF}} = \frac{\omega_b^3 - \omega_a^3}{6\pi \widehat{N}_{\mathrm{BP}}} \cdot \frac{1}{2(\Delta f_{\mathrm{MAX}} + 2 f_{\mathrm{MAX,LF}})} = \frac{\omega_b^3 - \omega_a^3}{6\pi \widehat{N}_{\mathrm{BP}} B_{\mathrm{HF}}} \tag{2.19}$$

The presented estimator requires the bandpass filter to be rectangular, which in a practical system is impossible to realize. Thus, the noise estimate will always be influenced by the residual energy of the adjacent MPX audio signals. Since the statistical properties of the audio signal inside the MPX are unknown, it is difficult to derive a mathematical model to quantify the variance of the estimator. Therefore, the robustness of the estimator has been evaluated in several simulations.

Before presenting an estimation scheme for $\widehat{\rho}_{\mathrm{HF}}$, the design of an efficient type of bandpass filter will be discussed and a method to use Equation 2.17 for non-rectangular filters will be derived.

An Efficient Multi-Stage IIR Bandpass Filter

Since the bandwidth of the MPX signal gaps is small compared to the MPX bandwidth, a bandpass filter of high order is required. As neither of the benefits of an FIR filter are required for narrowband energy accumulation, i.e. linear phase or finite impulse response, an IIR filter of low complexity can be used as subsequently described.

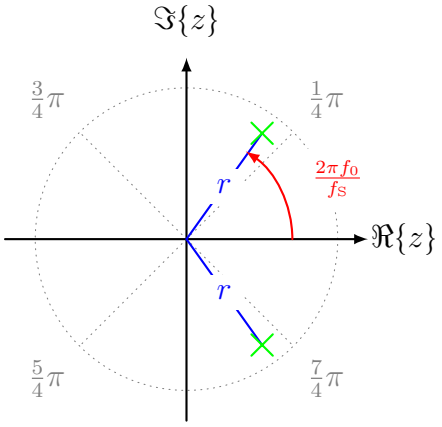


Figure 2.21: Poles of discrete second-order IIR resonator inside z-plane.

In fixed-point computing, higher order IIR filters are typically implemented as multiple cascaded IIR second-order filter sections (SOS) [PM06]. The complex-valued transfer function of a single SOS in the can be stated in the z-domain as

$$\underline{H}_{\mathrm{IIR}}(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}},$$

where b_0, b_1, b_2, a_1 and a_2 denote the filter coefficients. Note that albeit $z \in \mathbb{C}$, in accordance to literature for z underlining has been omitted. Since the band-gap in the MPX is narrow, a two-pole IIR resonator can be used instead of an IIR bandpass, which is less computationally complex as it has only poles and no zeros, thus reducing the number of multiplications and additions. The transfer function of a two-pole IIR resonator in the z-domain is formulated by

$$\underline{H}_{\mathrm{IIR}}(z) = \frac{b_0}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}}. \tag{2.20}$$

Deriving the difference equation from Equation 2.20 and feeding the demodulated FM MPX signal $y[k]$ as input to the IIR filter, the output of the resonator at each time instant k can be expressed by

$$y_{\mathrm{IIR}}[k] = b_0 \cdot y[k] - a_1 \cdot y_{\mathrm{IIR}}[k-1] - a_2 \cdot y_{\mathrm{IIR}}[k-2]. \tag{2.21}$$

According to [Smi08], the unity-gain filter coefficients for the two-pole resonator can be calculated given the center frequency f_0 and the pole radius r as follows

$$\begin{aligned} b_0 &= (1-r)\sqrt{1 + r^2 - 2r \cdot \cos\left(4\pi f_0 / f_S\right)} \\ a_1 &= -2r \cdot \cos\left(2\pi f_0 / f_S\right) \\ a_2 &= r^2. \end{aligned} \tag{2.22}$$

Figure 2.21 visualizes the relationship between the resonance frequency and the position of the two poles inside the unit circle. The distance of the poles to the unit circle controls the bandwidth of the IIR resonator, i.e. the closer the poles to the unit circle, the narrower the resonator bandwidth and vice versa. However, moving the poles closer

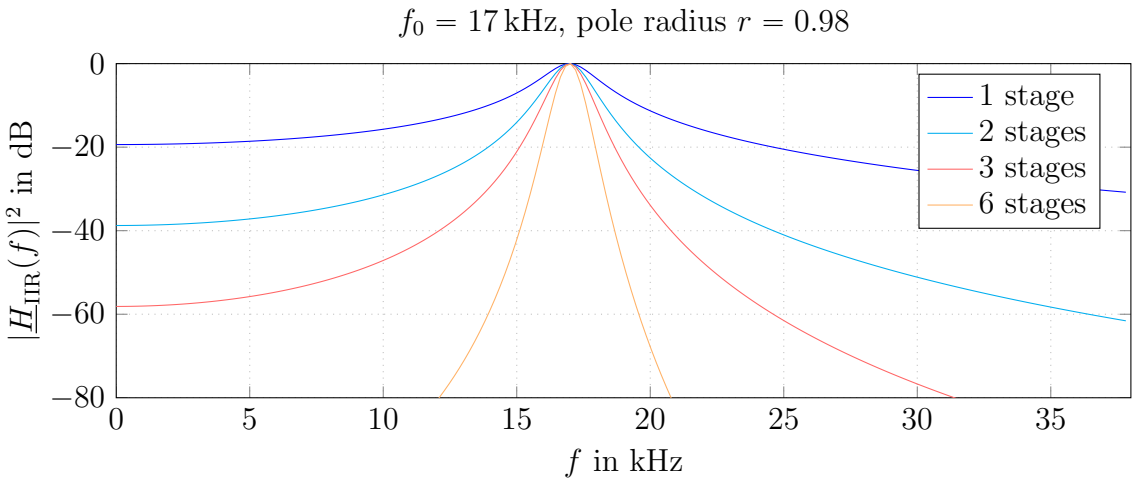


Figure 2.22: Frequency response of cascaded IIR two-pole resonator.

to the unit circle reduces the numerical stability of the filter and might render it infeasible for fixed-point implementations. A better approach is to cascade multiple stages of a less-sharp resonator to improve the numerical stability and achieve the required sideband suppression. The frequency response of a cascaded multi-stage IIR resonator with $r = 0.98$ and $f_0 = 17\,\mathrm{kHz}$ is shown in Figure 2.22. Using Equation 2.22, for a sample rate of $f_\mathrm{S} = 288\,\mathrm{kS/s}$ the filter coefficients can be calculated as

$$b_0 = 0.0144, a_1 = -1.8267, a_2 = 0.9604.$$

Since the shape of the presented IIR resonator is not rectangular (cf. requirement stated in Equation 2.16), Equation 2.18 can not be used to estimate the noise energy within the band-gap ω_a and ω_b. In order to obtain suitable values for ω_a and ω_b, the relation between the energy within the pass-band of the IIR resonator and the pass-band energy of a rectangular filter must be formulated. Let the pass-band bandwidth of the filter B_IIR be described as the region, where $|\underline{H}_\mathrm{IIR}(f)|^2 \geqslant -40\,\mathrm{dB}$. The frequency response of the 6-stage IIR filter in Figure 2.22 shows a pass-band region of $B_\mathrm{IIR} \approx 4\,\mathrm{kHz}$ at a center frequency of $f_0 = 17\,\mathrm{kHz}$. Outside this region, the sideband suppression is $|\underline{H}_\mathrm{IIR}(f)|^2 < -40\,\mathrm{dB}$. A stronger out-of-band rejection requires the concatenation of more than 6 IIR filter stages. The energy a single-stage resonator can collect within each side of the pass-band can be calculated by

$$E_\mathrm{IIR} = \int_{-B_\mathrm{IIR}/2}^{B_\mathrm{IIR}/2} \left(\frac{1}{2} \left| \underline{H}_\mathrm{IIR}\left(\mathrm{e}^{\mathrm{j}2\pi \frac{f+f_0}{f_\mathrm{S}}} \right) \right|^2 + \frac{1}{2} \left| \underline{H}_\mathrm{IIR}\left(\mathrm{e}^{\mathrm{j}2\pi \frac{f-f_0}{f_\mathrm{S}}} \right) \right|^2 \right) df, \tag{2.23}$$

and the energy density within the pass-band becomes

$$\Phi_\mathrm{IIR} = \frac{E_\mathrm{IIR}}{2\pi B_\mathrm{IIR}}.$$

Likewise, let the pass-band bandwidth of an ideal band-pass filter be defined as B. Then, the pass-band edges can be expressed by

$$\omega_a = \frac{2\pi}{f_\mathrm{S}} \left(f_0 - \frac{B}{2} \right) \quad \text{and} \quad \omega_b = \frac{2\pi}{f_\mathrm{S}} \left(f_0 + \frac{B}{2} \right). \tag{2.24}$$

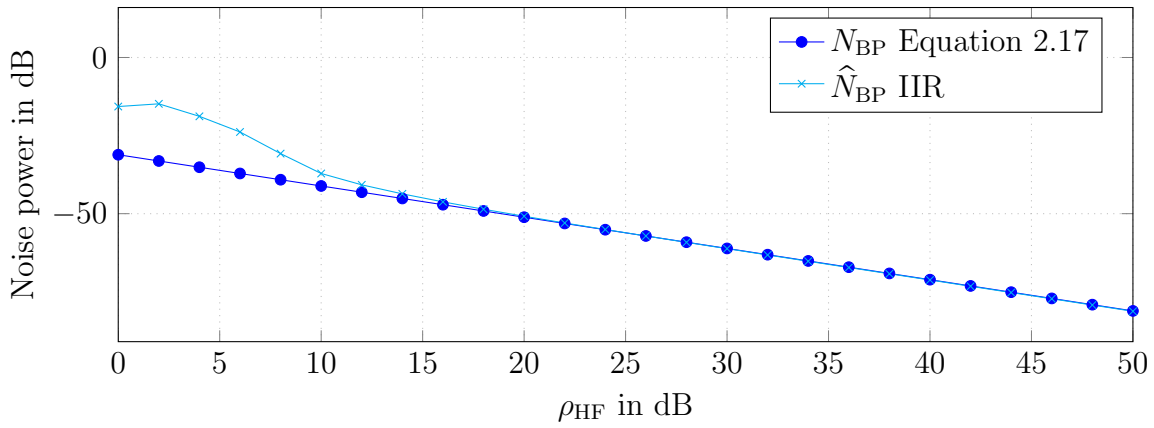


Figure 2.23: Noise power estimation performance using a six-stage IIR resonator.

Given a unity pass-band gain, i.e. an ideal band-pass filter, the accumulated energy inside the rectangular pass-band is

$$E = \frac{1}{2}(\omega_b - \omega_a) + \frac{1}{2}(-\omega_a + \omega_b) = \frac{2\pi}{f_{\mathrm{S}}} B, \tag{2.25}$$

and, clearly, since the filter has a pass-band gain of 1, the mean energy density within the pass-band is equal to

$$\Phi = 1.$$

Equation 2.25 states, that the energy a rectangular filter collects within the pass-band linearly increases with the bandwidth of the filter. Hence, if the mean energy of the pass-band of the IIR resonator filter is known, the bandwidth of an equivalent rectangular filter can be determined by

$$B = \frac{\Phi_{\mathrm{IIR}}}{\Phi} \cdot B_{\mathrm{IIR}} = \Phi_{\mathrm{IIR}} \cdot B_{\mathrm{IIR}}, \tag{2.26}$$

where $\Phi_{\mathrm{IIR}} < 1$ and, hence, $B \leqslant B_{\mathrm{IIR}}$. In conclusion, Equation 2.26 enables the translation of the IIR filter bandwidth into values for ω_a and ω_b using Equation 2.24. An analytical solution for the integral in Equation 2.23 has not been derived in this work but numerical simulations for a six-stage IIR resonator with $B_{\mathrm{IIR}} = 4\,\mathrm{kHz}$, $f_0 = 17\,\mathrm{kHz}$ and $r = 0.98$ have revealed a value of $\Phi_{\mathrm{IIR}} \approx 0.18$. Inserting these values into Equation 2.26 results in an IIR resonator filter that collects the energy of an equivalent ideal rectangular filter with $B \approx 720\,\mathrm{Hz}$.

SNR Estimation given the Mean Noise Power

Inserting the band-edges formulated in Equation 2.24 and the bandwidth translations in Equation 2.26 into Equation 2.19 and normalizing the HF bandwidth in the denominator to $B_{\mathrm{HF}}/f_{\mathrm{S}} = 1$ allows for deriving an estimated SNR value according to

$$\widehat{\rho}_{\mathrm{HF}} = \frac{\left(\frac{2\pi}{f_{\mathrm{S}}}\right)^3 \left(3 f_0^2 B + \frac{B^3}{4}\right)}{6\pi \widehat{N}_{\mathrm{BP}}}, \tag{2.27}$$

where using the values of the system the estimated SNR results in

$$\begin{aligned}\widehat{\rho}_{\mathrm{HF}} &= \frac{\left(\frac{2\pi}{288\,\mathrm{kHz}}\right)^3 \left(3(17\,\mathrm{kHz})^2 \cdot 720\,\mathrm{Hz} + \frac{(720\,\mathrm{Hz})^3}{4}\right)}{6\pi \widehat{N}_{\mathrm{BP}}} \\ &= \frac{3.4394 \cdot 10^{-4}}{\widehat{N}_{\mathrm{BP}}}.\end{aligned} \tag{2.28}$$

Note that in this expression $\widehat{N}_{\mathrm{BP}}$ refers to the mean accumulated power of the IIR filter output signal $y_{\mathrm{IIR}}[k]$ (cf. Equation 2.21). Given that the calculation is perfomed within an averaging period of N_{S} samples, the estimated mean noise power at the k-th sampling instant can be expressed by

$$\widehat{N}_{\mathrm{BP}} = \frac{1}{N_{\mathrm{S}}} \sum_{i=0}^{N_{\mathrm{S}}-1} |y_{\mathrm{IIR}}[k-i]|^2. \tag{2.29}$$

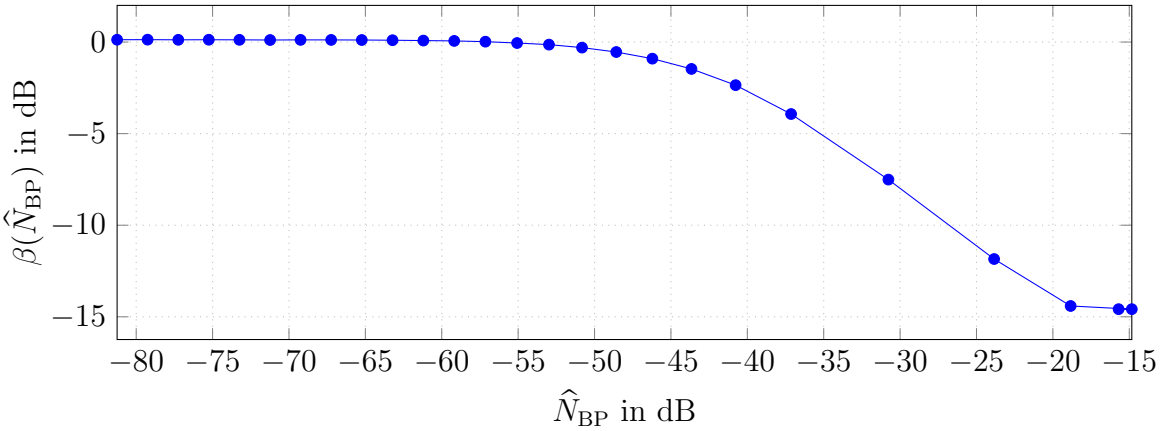


Figure 2.24: Noise power correction fuction.

The modified IIR-based noise estimator was tested in a simulation environment using the demodulator model as shown in Figure 2.18 and the equations presented in this section. Figure 2.23 shows that the carrier-to-noise ratio at the demodulator input can sufficiently be determined by the energy detection within the band-gaps of the MPX signal. The noise power in the simulation has been estimated by averaging one second of IIR in-band power. Furthermore, for high SNRs the estimation results in the simulations concur with the theory as stated in Equation 2.17. Since Equation 2.14 is inaccurate for low SNRs (cf. [Kam08]), the estimation increasingly deviates for $\rho_{\mathrm{HF}} < 15\,\mathrm{dB}$.

$$\widehat{N}_{\mathrm{BP,C}} = \widehat{N}_{\mathrm{BP}} \cdot \beta(\widehat{N}_{\mathrm{BP}}) \tag{2.30}$$

The non-linearity can be partially compensated by introducing a noise-energy dependent correction value $\beta(\widehat{N}_{\mathrm{BP}})$, stated in Equation 2.30, to improve the estimation at low SNRs. The function $\beta(\widehat{N}_{\mathrm{BP}})$ reflects a fraction of the theoretical noise value and the asymptotical estimated noise power and can been determined by simulations (cf. Figure 2.23 and the correction values in Figure 2.24). The slope of the correction function

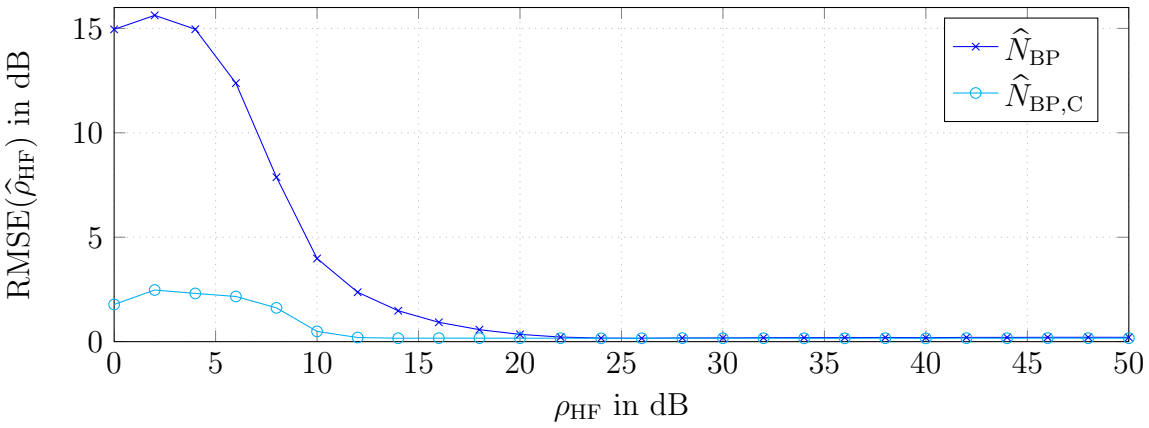


Figure 2.25: Root-mean-squared error of FM signal-to-noise estimator output in dB.

indicates that in case ρ_{HF} is low at the demodulator input, i.e. the noise power is high, the noise power estimate on average deviates up to 15 dB from the real value. The root mean-square error (RMSE) of the corrected and uncorrected SNR estimation values for different values of ρ_{HF} is plotted in Figure 2.25. The plot visualizes that post-correction improves the estimate for low ρ_{HF} values. The mean deviation for one second of averaging is within the range of 0.1 dB and 2.2 dB. The question in how far the detection performance is sufficient for SNR-adaptive reconfiguration of FPGA modules will be answered further on. Next, the hardware design and implementation complexity of the presented noise detector will be outlined.

2.4.4 Hardware Implementation

In hardware, the noise variance estimation is accomplished by a sequential execution of a single IIR second-order filter stage. This is possible since the FPGA clock frequency is more than 125 times the baseband sampling rate. The hardware implementation of the MPX-based noise estimator is shown in Figure 2.26. For block-wise processing in hardware, the sliding-window in Equation 2.29 has been implemented recursively, where the accumulated noise energy is calculated by

$$\widehat{N}_{\mathrm{ACC}}[k] = \widehat{N}_{\mathrm{ACC}}[k-1] + |y_{\mathrm{IIR}}[k]|^2,$$

with $\widehat{N}_{\mathrm{ACC}}[0] = 0$. An estimate of the noise variace is then obtained by

$$\widehat{N}_{\mathrm{BP}} = \frac{\widehat{N}_{\mathrm{ACC}}[N_{\mathrm{S}}]}{N_{\mathrm{S}}}.$$

The IIR memory registers of the SOS filters are implemented as a multiplexed register bank, i.e. in each filter iteration cycle the delay register corresponding to the active IIR filter is used. Generating a valid sample at the output of one SOS requires 7 FPGA clock cycles. Hence, the processing of one input sample to generate one output sample in a 6-SOS filter cascade requires 42 clock cycles in total. At the filter output, the sample energy is calculated by a squaring device and accummulated to obtain an estimate of the noise energy $\widehat{N}_{\mathrm{BP}}$. The energy calculation and accumulation take another 3 FPGA clock cylces, such that one noise energy accumulation cycle takes 45 FPGA cycles.

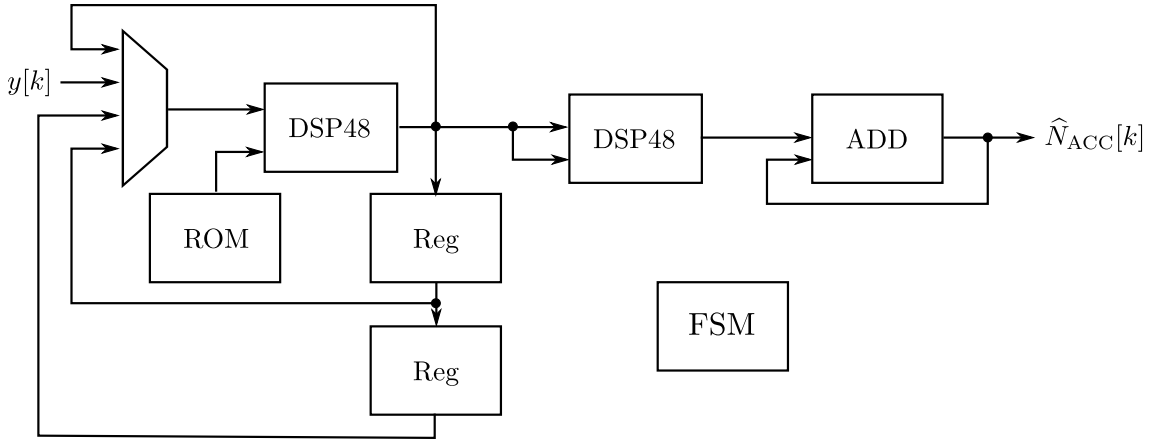


Figure 2.26: Hardware implementation of an MPX-based noise estimator.

The FPGA synthesis of the noise estimation algorithm resulted in a resource consumption of 193 slices, 2 DSP48 units and 1 BRAM. In conclusion, the hardware overhead for the estimator is neglible, given the available amount of resources in the XC5VSX50T FPGA. However, the synthesized design includes only the estimator and the control FSM, but no correction mechanism according to Equation 2.30. Before the estimation procedure starts, the output accumulation register is cleared to zero. Then, the noise power accumulation is perfomed and the output of the estimator is be fed to a DPR control unit, where the evaluation and correction of the value is performed. If the receiver is stationary, the more samples involved in the averaging, the closer the estimate $\widehat{N}_{\mathrm{BP}}$ approaches N_{BP}. In the fixed-point implementation of the system, a number of $N_{\mathrm{S}} = 5 \cdot 10^5$ samples has been evaluated as acceptable averaging duration [Mü11], resulting in a period of

$$\frac{5 \cdot 10^5 \,\mathrm{samples}}{288 \cdot 10^3 \,\mathrm{samples/s}} \approx 1.74 \,\mathrm{s}$$

per estimation cycle. The precision of the output accumulator will be concerned next.

Recall that the output of the FM demodulator DPLL generates values between -1 to 1, cf. Section 2.3, in two's complement fixed-point notation with 14 fractional bits. The number of fractional bits doubles after squaring in the DSP48 unit and the number of integer bits increases by $\log_2(N_s)$. Hence, in the worst case, the unsigned integer register at the estimator output requires

$$1 + 2 \cdot 14 + \left\lceil \log_2(5 \cdot 10^5) \right\rceil = 48 \,\mathrm{bits}$$

to store the accumulated output in full precision, where $\lceil \cdot \rceil$ denotes rounding towards infinity. However, in practical scenarios the regions of operation of ρ_{HF} will be small, such that the power of the noisy samples at the output of the IIR filter will be significantly below 1.

The function and design of the SNR evaluation unit is subsequently layed out in detail. Before that, meaningful SNR-related reconfiguration conditions will be presented, which will be used by a decision logic in the reconfigurable design, to switch from one configuration to another.

2.4.5 SNR-Related Reconfiguration Conditions

The SNR values obtained by the estimation unit are evaluated and used to trigger an FPGA reconfiguration when rising above or falling below a certain SNR threshold. Threshold values for the SNR have been derived for the receiver operation modes *Demod+Mono*, *Demod+Stereo* and *Demod+RDS*. A threshold will subsequently be defined as a lower bound on the SNR, where below this bound, the receiver will fail to function if in this mode.

Finding suitable switching SNR thresholds for monaural decoding is difficult since it depends on human perception whether the audio quality is acceptable or not. For example, the FM threshold could be taken as a switching value, since the FM decoding gain rapidly degrades if the SNR falls short on this level (cf. [Ric63]). However, as outlined by Rosenkranz in [Ros89], the FM threshold can vary for different FM demodulator implementations. Laboratory experiments outlined in [KTB+12] have revealed that, given the DPLL receiver implementation presented in Section 2.3, at an SNR of $\rho_{\mathrm{FM}} = 4\,\mathrm{dB}$ the audio signal is becoming so noisy, that it makes sense to switch-off audio decoding completely. Hence, if an SNR below $\rho_{\mathrm{FM}} = 4\,\mathrm{dB}$ is detected, the mono decoder can be removed from the reconfigurable partition and the partition can be used for the execution of other functions.

It has been derived in [SS08] that for stereo decoding the SNR at the demodulator input must be approximately 21 dB above the mono threshold for the audio quality to be sufficient. This is related to the fact that the audio difference signal is more prone to noise than the monaural sum signal since it is located at a center frequency of 38 kHz (cf. Equation 2.14). Thus, in case of stereo broadcasts it is feasible for the receiver to switch from stereo to mono if the estimated SNR falls below $\rho_{\mathrm{FM}} = 25\,\mathrm{dB}$.

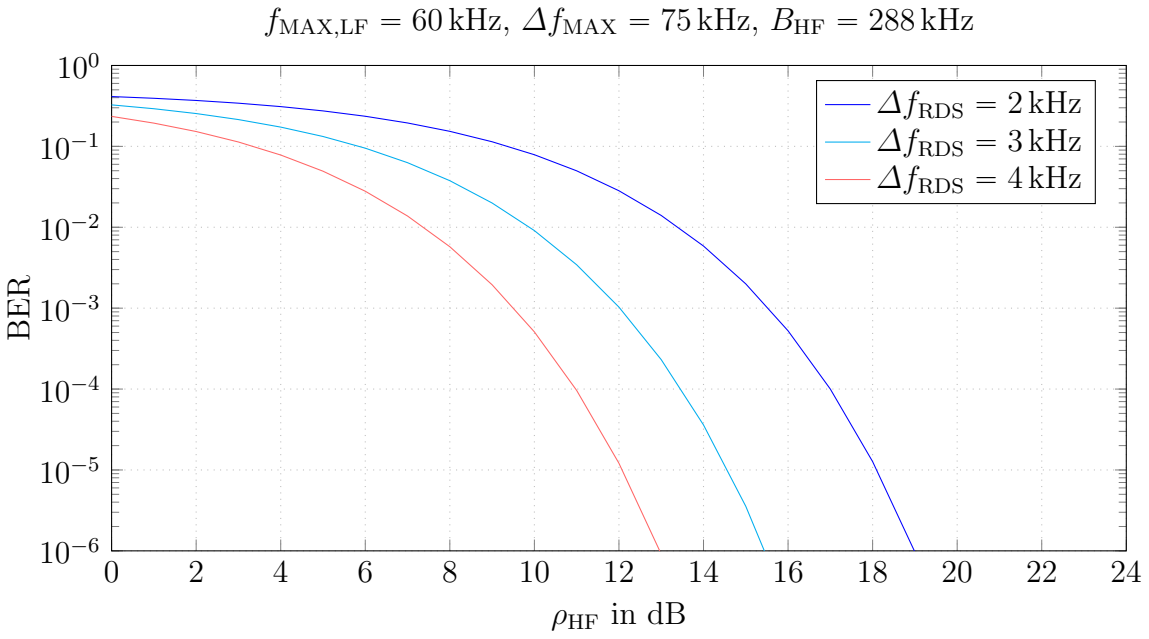


Figure 2.27: BER approximation for coherent RDS demodulation against FM CNR.

For RDS a decoding threshold can be formulated by means of the maximum bit-error-ratio that can be tolerated for error-free decoding. In [PS95] a BER of 10^{-2} before the

error correction code is taken as a threshold value for acceptable decoding performance. Comparing this BER with the theoretically achievable BER formulated by Equation 2.9, an MPX SNR requirement of $E_b/N_0 > 12$ dB is required at a center frequency of 57 kHz in the MPX signal. The energy per bit E_b can be derived from the FM carrier power, the FM frequency deviation proportion for RDS transmission and the RDS datarate of 1187.5 bit/s. In [Itu01b] an FM frequency deviation proportion of $\Delta f_{\mathrm{RDS}} = \pm 2\,\mathrm{kHz}$ is recommended for RDS transmissions. Combining the FM peak frequency deviation Δf_{MAX} and the FM carrier power A_{FM}^2 gives the signal energy of one RDS bit as

$$E_b = \frac{A_{\mathrm{FM}}^2 \cdot \left(\frac{\Delta f_{\mathrm{RDS}}}{\Delta f_{\mathrm{MAX}}}\right)^2}{1187.5\,\mathrm{bit/s}}.$$

Assuming that the additive noise within in RDS band is approximately white, Equation 2.14 can be used to approximate the noise power spectral density N_0 at a frequency of 57 kHz and a HF bandwidth of $B_{\mathrm{HF}} = 288\,\mathrm{kHz}$. Thus, Equation 2.9 can be used to express the relation between FM carrier power, RDS frequency deviation, MPX noise power and RDS bit-error-rate. In Figure 2.27 the relation is visualized for different RDS FM peak frequency proportions. The curves show that for $\Delta f_{\mathrm{RDS}} = \pm 2\,\mathrm{kHz}$ a bit-error rate of 10^{-2} is achieved when $\rho_{\mathrm{HF}} \geqslant 14\,\mathrm{dB}$. Below that threshold, the RDS decoder could be switched-off since it produces too many bit errors for reliable RDS decoding. In a reconfigurable FPGA design, it could alternatively be removed from the reconfigurable partition.

Configuration	SNR Threshold
Demod + Mono	$\rho_{\mathrm{HF}} \geqslant 4\,\mathrm{dB}$
Demod + RDS	$\rho_{\mathrm{HF}} \geqslant 14\,\mathrm{dB}$
Demod + Stereo	$\rho_{\mathrm{HF}} \geqslant 25\,\mathrm{dB}$

Table 2.5: SNR operation thresholds for different FM receiver module configurations.

Hence, if the SNR estimator signals that the SNR has fallen below a certain threshold, the reconfiguration controller in the FPGA can initialize a DPR of the FM multiplex decoder partition to become more or less complex. The SNR thresholds for the different reconfigurable partitions are summarized in Table 2.5, where ρ_{HF} denotes the CNR at the FM demodulator input. It becomes apparent that the SNR threshold distance is approximately 10 dB from one configuration to another. This means that, for example, if RDS is working at a BER of 10^{-2}, the signal power needs to be increased by a factor of 10 for stereo reception to be feasible. From mono to stereo, the required increase in power is even higher with roughly a factor of 100. These relatively large SNR gaps motivate the use of a dynamically-adaptable receiver, where the active decoder is switched according to the estimated SNR. In a reconfigurable FPGA design, the presented approach can be used to trade the resources inside the reconfigurable region with respect to the resource and SNR requirements. The possibilities and limits of such a receiver implementation will be described in the next section.

2.5 An SNR-Adaptive FM Receiver using Partial Reconfiguration of FPGAs

In the previous section, SNR thresholds for feasible demodulation of the three main FM signal components have been derived, which will be used for the design of three different reconfigurable FPGA receiver prototypes. These prototypes will be presented with incremental complexity, i.e. the simplest approach is presented first and the most complex approach is presented last. The first approach uses a single reconfigurable partition for receiver operations, furthermore denoted as single-island design. In this design, the receiver configurations presented in the SNR threshold table will be exchanged according to the FM demodulator SNR. Subsequently, a multi-partition reconfigurable receiver will be presented, comprising of multiple demodulation chains. In this multi-island design, multiple reconfigurable partitions will be reconfigured according to the derived SNR constraints. In the last section an approach will be presented, where a single partition is sub-divided into smaller partitions for a fine-grained reconfiguration. This approach will be referred to as resource-sharing design.

2.5.1 Single-Island Design

In the single-island design, the logic of one particular receiver configuration is placed inside a single reconfigurable partition. The static logic is enclosing the reconfigurable partition and the signals are gated from and to the different logic areas using flip-flops at the reconfiguration borders. A soft-core Xilinx Microblaze CPU is used to initiate reconfiguration tasks and to handle system events. The CPU is configured to use an 8 kB BRAM for data and instruction caching and one DSP48 unit for hardware multiplication. The PLB clock frequency is set to 100 MHz with a bus width of 32 bit and interfaces a DDR2 RAM controller via data-cache link and instruction-cache link, an XPS_HWICAP module, a DPR control module and a compact flash module. The Xilinx HWICAP module is described in [Xil11a] and is operated without direct memory access (DMA). It is reported in [LKLJ09] that using the HWICAP PLB interface without DMA is not optimum in terms of ICAP data throughput. For high-performance reconfiguration, DMA-based implementations would be preferable (cf. [HP11]). The DPR control unit is memory-mapped via registered I/O and is used to enable or disable the reconfiguration gateways and to reset the logic inside the reconfigurable partition. Aside from the configuration and RDS signal gateways, there is one 8 bit baseband I/Q input gateway and one 16 bit pulse-code modulation (PCM) audio output gateway. The hardware configuration and generation was accomplished using Xilinx XPS. A clock distribution unit is used to generate the different clock frequencies in the design. Figure 2.28 shows the described configuration. The partition-based DPR flow was used for the design of the reconfigurable partitions (c.f. [Xil12c]). The bitstreams for the different configurations are cached in an external on-board DDR2 memory and loaded on demand by the configuration control block.

In relation to the three different SNR threshold regions, three partial bitstreams have been generated for the different configurations. The required resources of these bitstreams are summarized in Table 2.6. Comparing the required number of slices with the Spartan-3A resource consumption in Table 2.16, the Virtex-5 design seems to require fewer logic and BRAM resources, whereas the number of DSP48 units stayed the same.

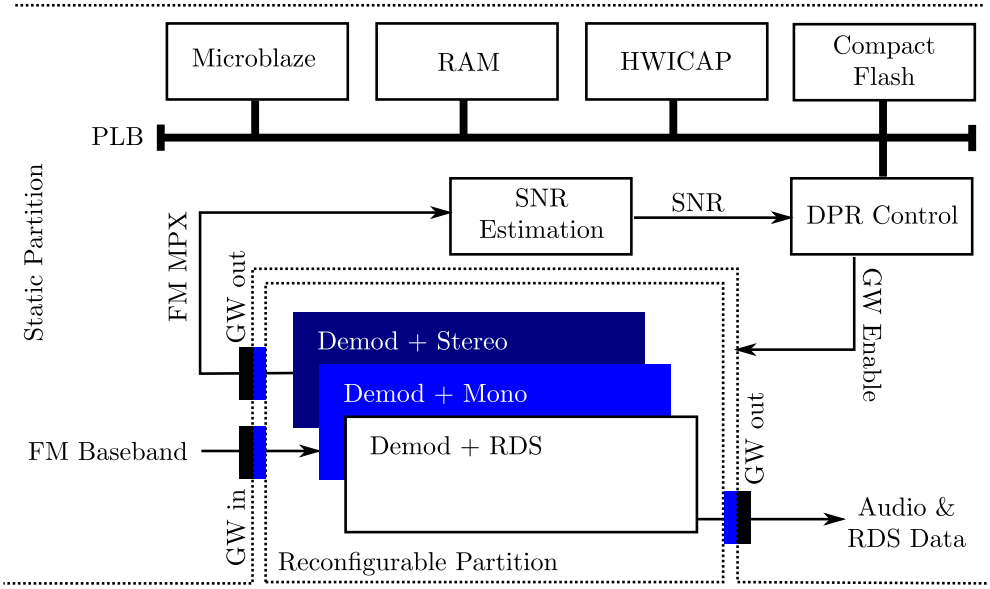


Figure 2.28: Single-island reconfigurable FM receiver system design.

This is because the logic and BRAM resources of the Spartan-3A FPGA are not directly comparable to the Virtex-5 fabric, as Virtex-5 FPGA slices contain four 6-input LUTs and four flip-flop registers in contrast to the two 4-input LUTs and two flip-flop registers available in the Spartan-3A FPGA. Additionally, as far as BRAMs are concerned, a Virtex-5 FPGA BRAM can store twice as much data as compared to its counterpart on the Spartan-3A FPGA. Given the reconfiguration interface logic and resource floorplanning for DPR, the reconfigurable Virtex-5 FM receiver is considerably more complex as compared to the previously introduced static Spartan-3A design.

Configuration	Slices	DSP48	36k BRAM
Demod + Stereo	1963	14	10
Demod + Mono	1375	7	8
Demod + RDS	1449	11	10
XC5VSX50T	8160	288	132

Table 2.6: FM receiver resource consumption on Xilinx XC5VSX50T FPGA.

For example, in the Virtex-5 design the FPGA area constraints must be tailored to the FPGA resource and configuration layout, i.e. the constraints must be defined such that they span an integer multiply of configuration frames, which is the smallest reconfigurable entity of a Xilinx FPGA (cf. Section 1.1 in Chapter 1). Furthermore, the reconfigurable area must enclose enough CLBs, BRAMs and DSP48 units to implement the design. As stated in [Xil12c], a Virtex-5 configuration frame is 1 CLB column wide and 20 CLB rows high, and one CLB includes 2 slices. The XC5VSX50T FPGA has 120 rows and 34 columns available for floorplanning [Xil12e]. As a consequence, module-based partial reconfiguration is more feasible in horizontal direction, i.e. across the CLB columns. The PLB components and the Microblaze CPU have been generated using the Xilinx EDK. The exported *pcore* has been used as target platform for the software compilation

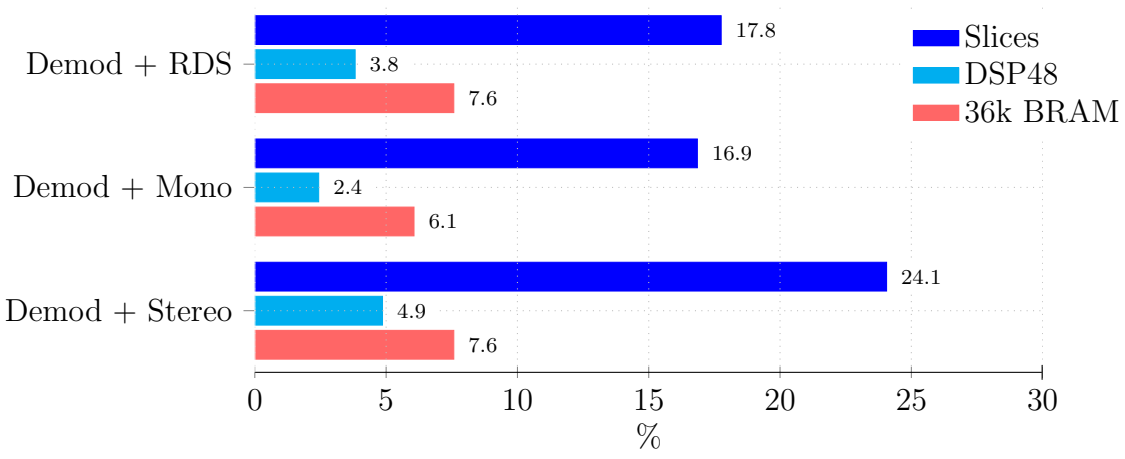


Figure 2.29: FM receiver relative resource consumption of Xilinx XC5VSX50T FPGA.

process in the Xilinx SDK. The DPR control software has been provided as C code and compiled using the Xilinx SDK. Xilinx System Generator has been used to design the reconfigurable module configurations as listed in Table 2.6.

Generating a SystemACE configuration from the implemented designs involves multiple steps (cf. Figure 2.30). First, a VHDL description of the top-level entity needs to be available, which comprises of the static system and the initial configuration of the reconfigurable module. Therefore, the Microblaze design, the DPR decoupling logic, the input processing logic and the AC97 output logic in the static partition must be available. In case of the FM receiver, the first configuration is the *Demod+Stereo* configuration. Next, auto-generated VHDL code is exported from within System Generator and synthesized afterwards. Upon successful synthesis, the netlist of all partitions is available. Area-constrained mapping, placing and routing is performed by a tool command language (TCL) script using the options `RUN_NGDBUILD`, `RUN_MAP` and `RUN_PAR`. For each configuration, area constraints have been provided by a user constraint file (UCF), which can be created manually or can alternatively be generated using Xilinx PlanAhead [Mü11]. The TCL script also generates a directory structure for a unique identification of every DPR configuration. After the static design and the reconfigurable modules have been jointly implemented, the initial (full) bitstream and the partial bitstreams are generated[1]. The bistreams are further on provided to Xilinx GenAce and copied to the compact flash memory card together with the compiled DPR control code for the Microblaze design. Using the partition-based design flow, the size of the partial bitstreams is proportional to the size of the reconfigurable area. More specifically, the size does not depend on the logic utilization inside the reconfigurable area, but on the size of the rectangular partition that spans the FPGA floorplan. Furthermore, since the Xilinx toolchain does not provide a shrinking of the reconfigurable partition during FPGA operation, re-allocating unused resources inside one reconfigurable partition to other partitions is not possible.

In the final design, the software must initiate a reconfiguration when a trigger signal is being received. The trigger signal may be provided by the DPR control circuit at the PLB or via GPIO. The DPR-related software tasks of the Microblaze CPU are depicted in the flowchart in Figure 2.31, showing three states:

[1]The partial bitstream is generated using Xilinx Bitgen with the options `-w -g ActiveReconfig:Yes`.

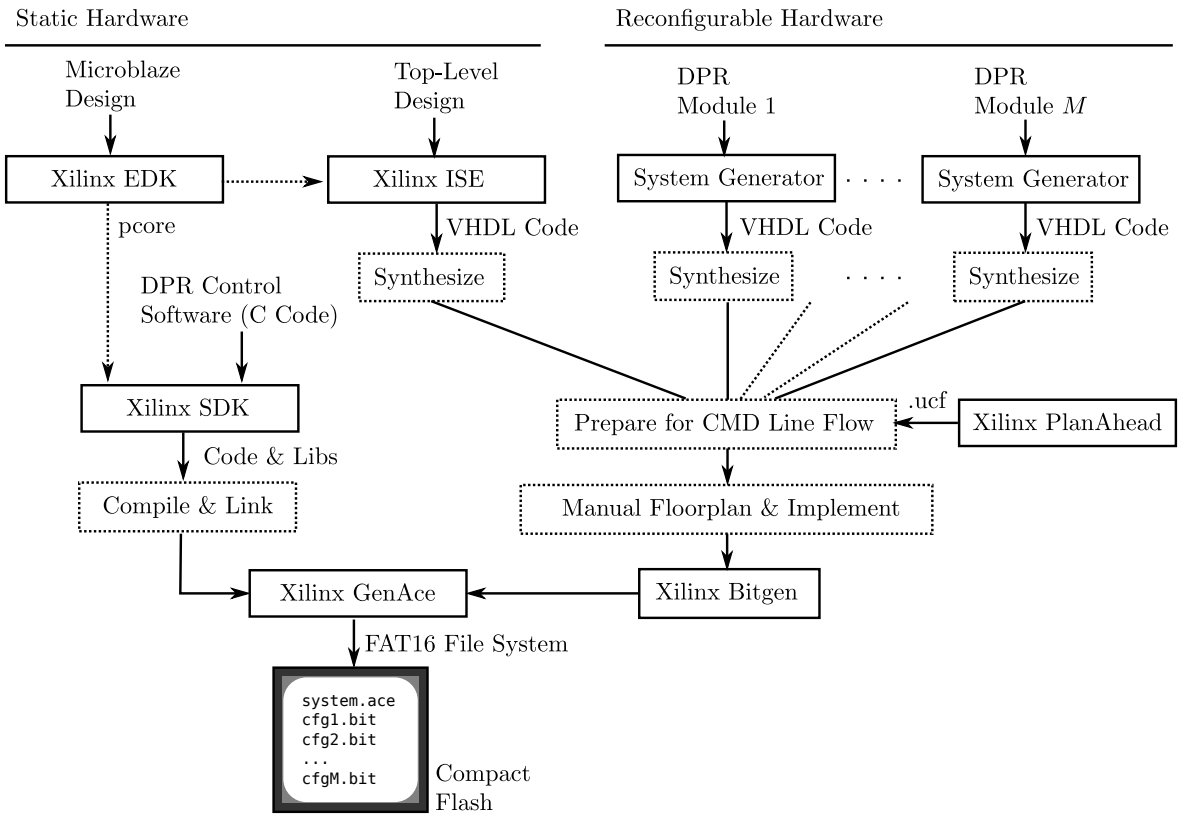


Figure 2.30: DPR receiver system design tool-flow.

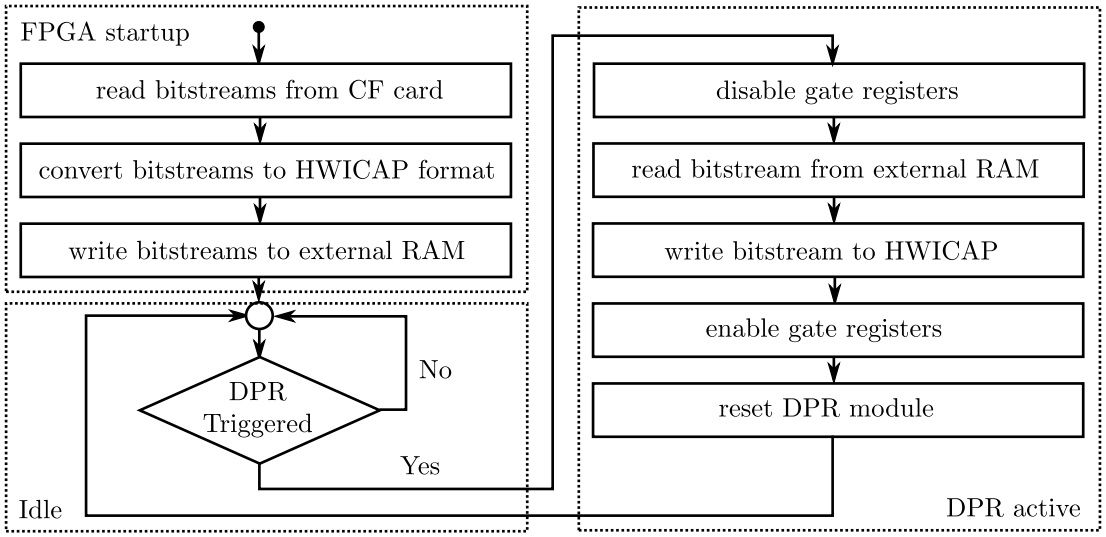


Figure 2.31: Microblaze software bringup and reconfiguration loop flowchart.

- **FPGA startup:** During the startup phase, the bitstreams are read from the compact flash card, converted into the HWICAP format by swapping the byte endianess and stored in the external DDR RAM. Subsequently, the receiver enters the idle state and wait for a reconfiguration trigger signal.
- **Idle:** In the idle state the CPU waits for a reconfiguration trigger and the reconfigurable partition continuously processes the incoming data. Upon trigger detection, the Microblaze CPU enters the DPR active state to initiate a reconfiguration of the DPR partition.
- **DPR active:** When entering the DPR active state, the DPR gate registers are disabled to decouple the reconfigurable partition from the static partition. Then, the partial bitstream of the DPR module is loaded from the external DDR memory and written to the HWICAP. After successful reconfiguration, the gate registers interfacing the DPR partition are enabled and the DPR module is reset. Finally, the CPU goes back into the idle state.

The reconfiguration interface throughput mainly depends on the clock frequency of the HWICAP and PLB. The reconfiguration time can be estimated from the mean reconfiguration throughput and the size of the partial bitstream. For the single-island Virtex-5 design, the HWICAP has been configured to a clock frequency of 100 MHz and a width of 32 bits, which gives a theoretical peak throughput of 400 MBytes/s. In practical systems, this rate is seldom achieved (cf. [LKLJ09]) and requires dedicated reconfiguration engines. For example, in [HP11] Hoffman et al. have presented a high-speed dynamic partial reconfiguration controller (HSDPRC) for Virtex-5 FPGAs with a maximum write throughput of 418.5 MB/s using a DMA engine, a PowerPC memory controller (PPC440MC) and overclocking the ICAP to 133 MHz. Using a 200 MHz Microblaze, a Xilinx multi-port memory controller IP and a 100 MHz ICAP clock frequency, the maximum reconfiguration speed Hoffman could achieve was 178.6 MB/s. For the DPR setup in this work, the reconfiguration throughput has not been measured but has been bounded by measurement results presented in literature. Table 2.7 summarizes the reconfiguration port throughput reported for comparable systems using the HWICAP block at the PLB without DMA.

DPR System and Clock Frequencies			DPR Throughput	
FPGA	CPU	ICAP	Max. Theory	Measured
Virtex-4 FX20 [LKLJ09]	MB@100 MHz	100 MHz	400 MB/s	14.5 MB/s
Virtex-4 FX20 [LKLJ09]	PP@300 MHz	100 MHz	400 MB/s	19.1 MB/s
Virtex-5 XC5VFX70T [KDHS14]	MB@100 MHz	100 MHz	400 MB/s	19 MB/s
Spartan-6 XC6SLX45T[Sch11]	MB@66 MHz	20 MHz	40 MB/s	4.8 MB/s
MB: Xilinx Microblaze Soft Core, PP: PowerPC Hard Core				

Table 2.7: Reconfiguration performance with HWICAP at PLB without DMA.

The values emphasize that the measured throughputs are more than an order of magnitude below the theoretical maximum. Due to the fact that the data needs to be transferred across the CPU local bus, the slowdown can most effectively be counteracted by using direct memory transfers to the ICAP interface. For a Virtex-5 FPGA, Kulkarni

quotes an effective DPR throughput of 19 MB/s. Given the size of the partial bitstream of 622, 795 Bytes, the duration to load a new set of configuration frames into the single-island DPR partition equals 33 ms. For the bitstream size of the prototype system, the reconfiguration duration for other systems and throughput values are outlined in Table 2.8.

Publication	ICAP Interface	DPR Throughput	DPR Duration
Liu in [LKLJ09]	PLB	14.5 MB/s	43 ms
Kulkarni in [KDHS14]	PLB	19 MB/s	33 ms
Hofmann slow in [HP11]	DMA	178.6 MB/s	3.5 ms
Hofmann fast in [HP11]	DMA	418.5 MB/s	1.6 ms

Table 2.8: Reconfiguration time estimates for single-island DPR partition.

The resource requirements of the complete single-island DPR system is given in Table 2.9. In comparison to the previously stated resource consumption of the receiver (cf. Table 2.6), the new design is quite large consuming 67 % of the available FPGA slices. Measures to overcome this drawback are presented in the following section, where the receiver FM demodulator is separated from the MPX decoders. As already pointed out, the noise estimator is quite resource efficient, consuming only 2.3 % of the FPGA slices plus two DSP48 units and one BRAM.

Configuration	Slices	DSP48	36k BRAM
Noise Estimator	193	2	1
Complete DPR Design	5466	19	43
XC5VSX50T	8160	288	132

Table 2.9: Resource consumption of single-island DPR receiver.

The final demonstration platform is shown in Figure 2.32, consisting of a PC, a Xilinx Spartan-3 FPGA and a reconfigurable Xilinx Virtex-5 FPGA (XC5VSX50T). The PC generates a complex baseband signal at a sample frequency of 500 kHz and transmits the data to the Spartan-3 FPGA via USB. The reconfigurable Virtex-5 device reads the data from a parallel GPIO interface and processes it internally. The PC generates a modulated FM stereo broadcast signal including an RDS service and the SNR of these signals can be varied by adding white Gaussian noise to the respective stream. The reconfigurable FM receiver prototype demonstrates the concept of an SNR-adaptive cognitive radio and serves as a template for further investigations of more complex applications.

In the presented setup, the FM baseband signal is received, decoded and the respective SNR is estimated in the Virtex-5 FPGA. According to the SNR of the received FM signal, the MPX decoding routines are self-adapting. When switching between different designs using one reconfigurable partition, in terms of resources this partition needs to provide enough headroom to satisfy the requirements of the most demanding DPR module implementation. For the FM receiver, this means that the reconfigurable area must include enough resources to allow for stereo decoding. The single-island design does then allow for an SNR-adaptive switching between different receiver configurations. As a

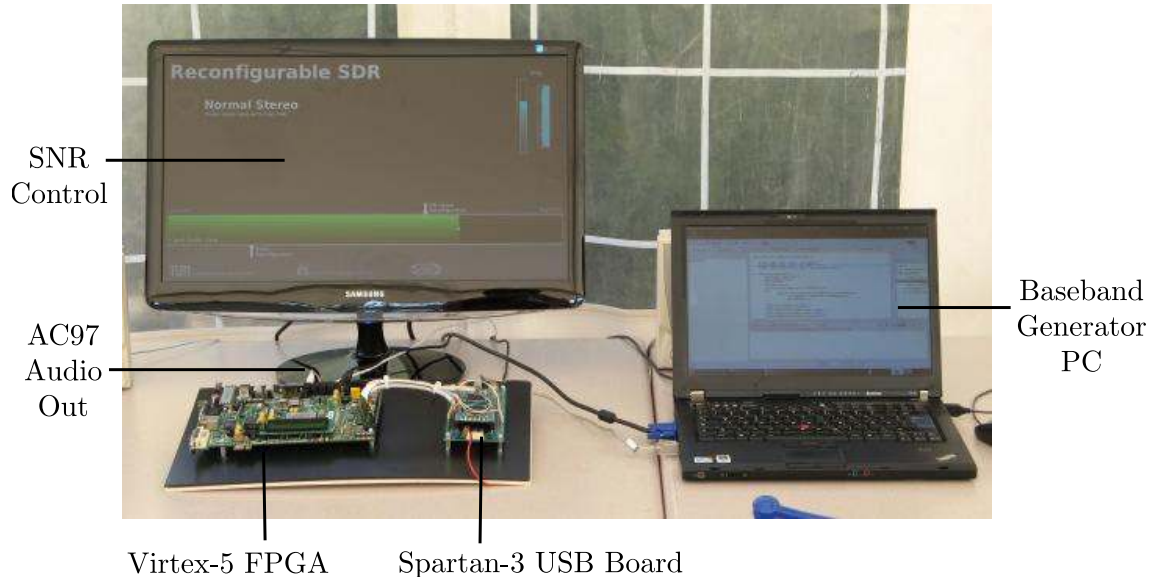


Figure 2.32: Reconfigurable broadcast FM receiver demo system.

future work, it would be interesting to analyze the savings in dynamic power consumption if the reconfigurable partition are de-activated, i.e. in case the SNR falls below a given threshold.

In the following section, a modified multi-island DPR receiver design will be presented, where multiple MPX decoders operate in different DPR partitions.

2.5.2 Multi-Island Design

In the scope of this work, two multi-island systems were designed and implemented: a dual-island and a triple-island reconfigurable receiver system, where the FPGA comprises one static and two or three reconfigurable partitions. Due to the fact that the demodulation part is required to be present in all DPR configurations, the FM demodulator was separated from the MPX decoder in the multi-island design. This leads to a considerable reduction in resource utilization for the reconfigurable partitions as outlined further on.

Equal to the single-island design, the static partition includes a Microblaze microcontroller and a reconfiguration control unit. Additionally, multiple PLLs and estimation stages are employed for FM demodulation and SNR estimation. The DPR modules contain the MPX signal decoders, which are supplied by the data of the FM demodulation DPLLs in the static partition. Each partition can hold one of the following demodulator types: Stereo demodulator, mono demodulator and RDS decoder. Figure 2.33 shows the described hardware setup with two reconfigurable islands. The reconfigurable partitions are denoted as DPR partition 1 and DPR partition 2 and each partition can be reconfigured individually without interrupting the other. The reconfigurable system has been implemented using the tool-flow as presented in the previous section, cf. Figure 2.30.

In Table 2.10, the resource requirements for the different reconfigurable MPX decoding modules are quantified. With FM demodulator and MPX decoder being split, the dual-partition system occupies 73.2 % of the slice resources. In comparison to the single-island design, which requires 67 % of the slice resources, the dual-partition design is

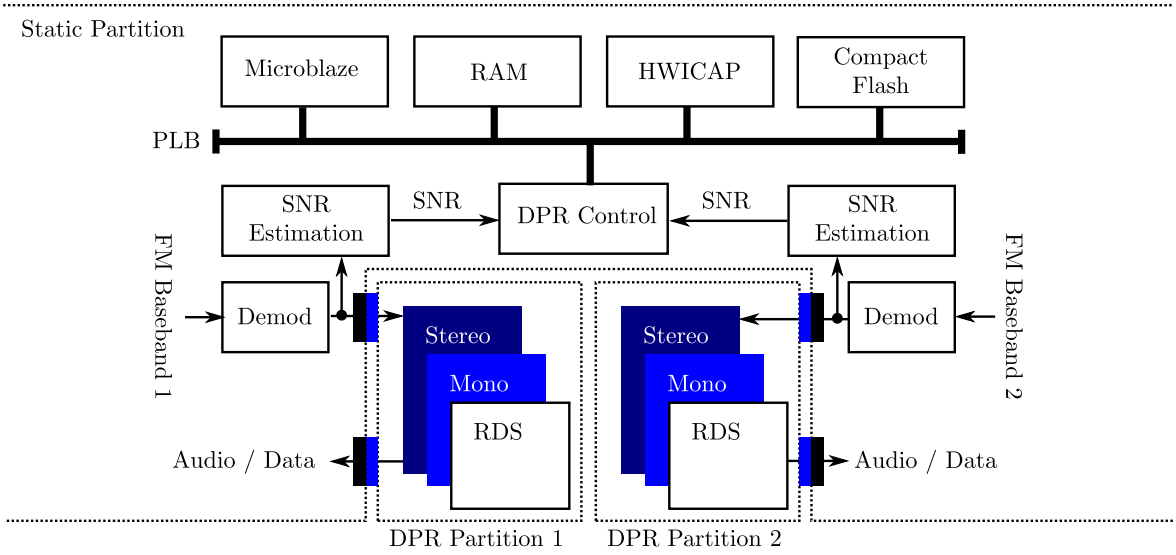


Figure 2.33: Dual-partition reconfigurable FM receiver design.

more efficient in terms of resources per MPX decoder. Regarding the interfacing, the FM receiver input and demod units require 10.8 % of the slice resources as each I/O unit includes a PLL and a fractional polyphase resampler to down-convert from 500 kS/s to 288 kS/s baseband rate. The stereo configuration is the most demanding MPX decoder in terms of slices and DSP48 units, whereas the RDS configuration requires most of the BRAM resources. Similar to the single-island design, the amount of available logic resources inside each DPR partition has to be defined with respect to the most complex design to handle the complexity of all DPR modules. Using the multi-island receiver, different configuration permutations can be operational on the device, e.g. the receiver can have two stereo demodulators, or one mono and one RDS demodulator, two RDS demodulators, etc.

Type	Configuration	Slices	DSP48	36k BRAM
FM receiver input	Demod	888	5	5
FM receiver output	AC97	31	0	2
MPX DPR module	Stereo	804	9	3
MPX DPR module	Mono	458	2	2
MPX DPR module	RDS	503	6	5
Dual-partition DPR system		5976	31	47
Triple-partition DPR system		6934	40	52
XC5VSX50T FPGA		8160	288	132

Table 2.10: Multi-island receiver resource requirements.

In case the SNR estimator of a specific decoding branch detects a value below the previously introduced decoding threshold, the respective partition is replaced, similar to the approach presented in [CKPLM10]. If the noise power increases above a level

where decoding is not feasible anymore, the MPX decoder in question is replaced by an empty bitstream. The CPU evaluates the estimated SNR values and is able to trigger a reconfiguration of DPR partition 1 or 2. While the FM-MPX decoding chain is reconfigured via DPR, the SNR estimation and the FM signal demodulation stay active in the static part of the device.

Possibilities on how to achieve a sharing of FPGA resources among different DPR partitions will be evaluated in the next section. Furthermore, it will be described in how far a fine-grained resource-sharing approach is possible to implement with state-of-the-art Xilinx tools.

2.6 Resource-Efficient Concurrent Receivers using DPR

The multi-island prototype represents a flexible FM demodulation system, where different decoders can be swapped on-the fly using the SNR of the input signal as triggering metric. However, although the presented implementation exemplifies that SNR-adaptive reconfiguration can be applied to multiple islands, it also shows that with fixed-size DPR partition areas there are limited possibilities of re-using the unused logic resources when switching from complex MPX modules to simpler MPX modules. Thus, using multi-island reconfiguration does not bring benefits in terms of resource savings as, similar to the single-island design, each MPX decoder partition utilizes the same number of configuration columns and rows.

Using the previously discussed DPR approaches, the logic resources of the reconfigurable areas need to hold the most complex DPR module designed to be executed inside the DPR partition. In case of a non-uniform module resource-distribution, the design suffers from resource fragmentation, resulting in a sub-optimum resource utilization. Thus, if DPR modules utilize only a small fraction of the available resources in a DPR partition, it would be desirable to free the unused resources and allocate them to adjacent DPR regions. In this section, the concept of such a resource-sharing approach will be described and the possibilities to utilize reconfiguration on a more fine-grained level will be discussed.

2.6.1 Motivation

In a multi-module DPR design, a set of modules can be mapped in various ways to the reconfigurable partitions by permutation. Given M DPR modules and P reconfigurable partitions, the number of configuration sets with unique partition-to-module mapping is M^P. If the order of the modules is ignored, such a set of modules is denoted as multiset [Knu97] and the number of element permutations can be calculated by

$$K = \binom{M+P-1}{P},$$

where $\binom{\cdot}{\cdot}$ denotes the binomial coefficient. Hence, for a dual-module design with $M = 3$ modules and $P = 2$ partitions the number of possible module-to-partition mappings is $K = 6$. Given the premise that always a minimum of two MPX modules are active in the reconfigurable design, Figure 2.34 shows the amount of resources required for a resource-shared dual-module DPR implementation using the values of Table 2.10. The

accumulated utilization of slices, BRAMs and DSP48 units is indicated in the figure by dashed lines. Similarly, for a triple-module receiver, Figure 2.35 shows the different set realizations of MPX decoders with $K = 10$. The dual-decoder bargraphs show that in terms of slices and DSP48 units the set (6) is the most demanding, whereas set (1) requires most of the BRAM resources. The amount of resources available in the DPR partition is a design criterion that defines the limit on the realizable tuple. In practice, a limit must be defined for each of the three resource elements and all tuples that do not exceed this limit in terms of resources are suitable candidates to operate inside the DPR region. Or the other way around, if the amount of resources inside the reconfigurable partition is insufficient to operate a set of resource-demanding module pairs, the design can not be realized.

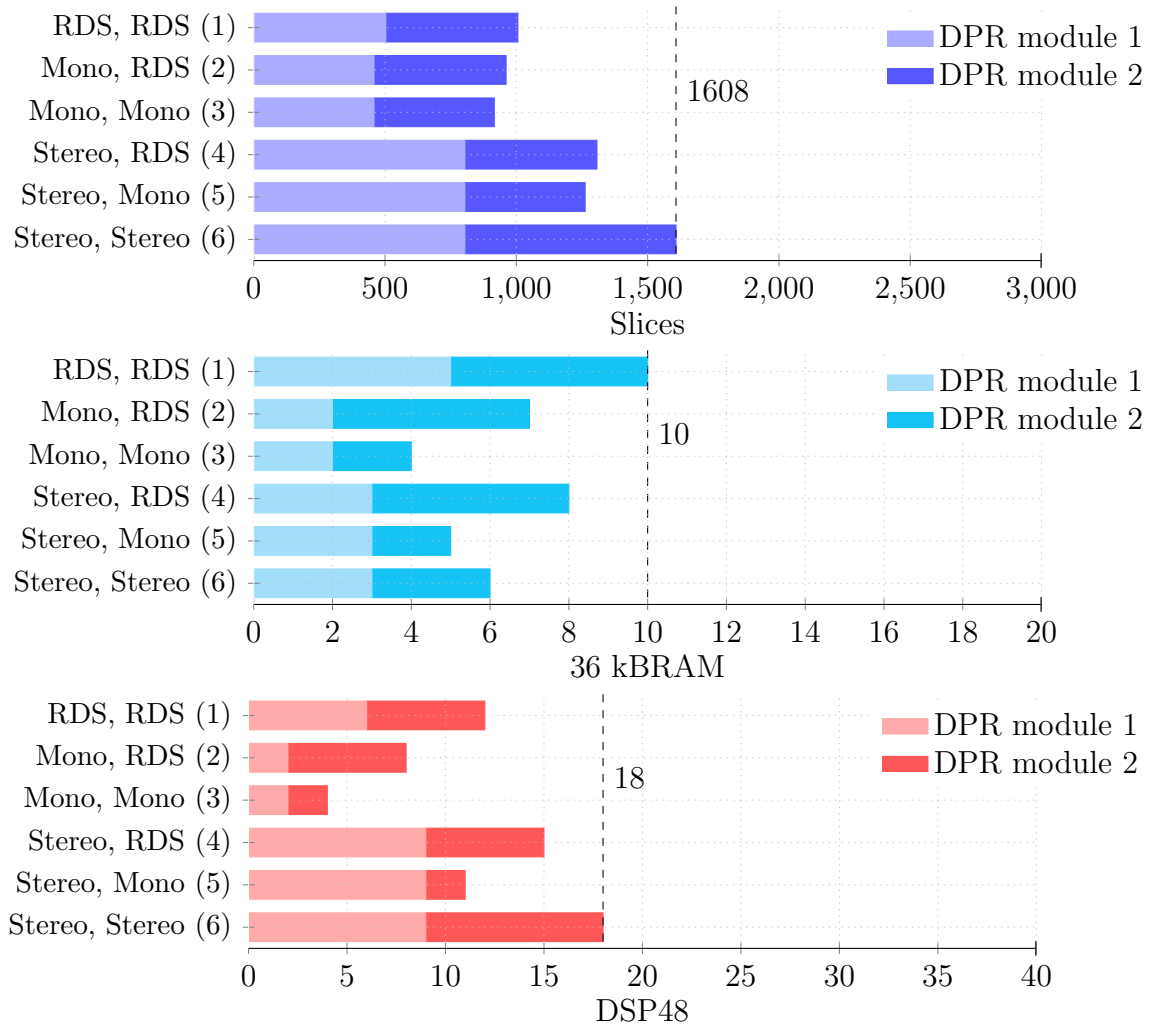


Figure 2.34: MPX dual-decoder accumulated module resources on XC5VSX50T FPGA.

Following the Xilinx tool-flow requirements, in partition-based dual-island designs, enough resources must be reserved to realize the most complex module configuration in both DPR partitions. Hence, more resources must be allocated than eventually required. In the following, approaches are introduced that potentially lead to more resource-economic implementations.

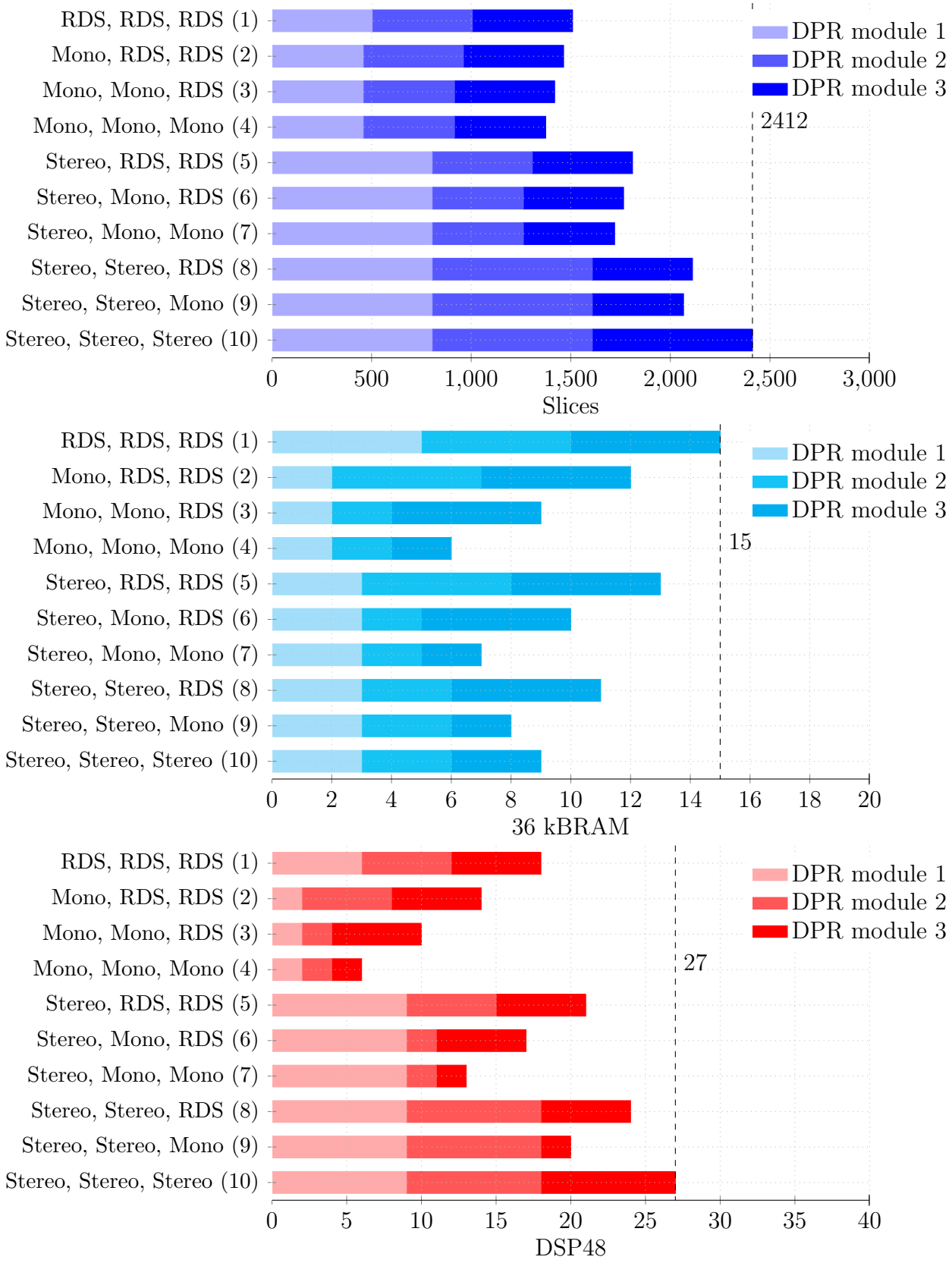


Figure 2.35: MPX triple-decoder accumulated module resources on XC5VSX50T FPGA.

2.6.2 Proposed System

Given are multiple decoder implementations with different FPGA resource requirements, in the following denoted by *large* and *small*. The decoders will be implemented as DPR modules and connected either to the left or to the right border of the reconfigurable partition, cf. Figure 2.36. Regarding the application, it is assumed that *either* the left partition or the right partition is holding a large decoder configuration, but never both at the same time. The resources in the reconfigurable partition will be shared among both configurations and the area will be constrained such that the following setup can be realized:

1. A large decoder may be present in the left DPR module implementation and a small decoder in the right DPR module implementation.
2. A small decoder may be present in the left and right DPR module implementation.
3. A small decoder may be present in the left DPR module implementation and a large decoder in the right DPR module implementation.

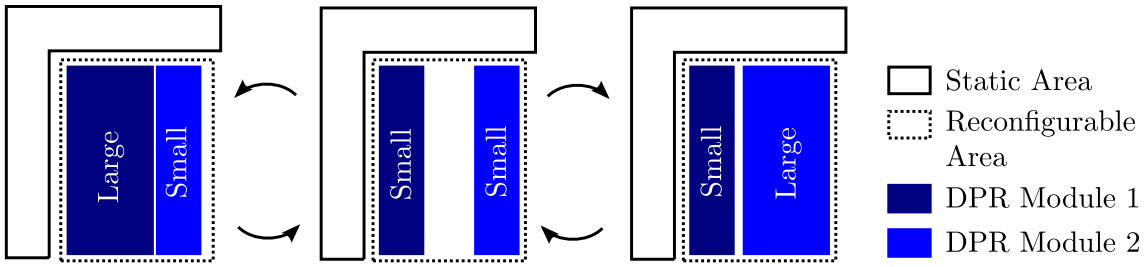


Figure 2.36: Dual resource-sharing reconfigurable system design.

In the proposed system, the resources in the center of the reconfigurable island among the left and right DPR modules are shared. For a multi-island DPR system as presented in Section 2.5.2, the reconfigurable area must be large enough to hold two times the largest design. In a resource-sharing design the required area can be reduced since only the largest and the small design will ever be part of the system. Clearly, the bigger the difference between the large designs and the small design, the more resources can be saved compared to the traditional multi-island DPR design.

Considering the presented MPX decoder modules, the small implementation may refer to the mono MPX decoder and the large implementation may refer to the stereo MPX decoder or the RDS MPX decoder. The accumulated resources of four interesting dual-decoder configurations are summarized in Table 2.11 including the relative resource requirements compared to a dual-stereo decoder. The table shows that the resource consumption is reduced for all configurations except for the BRAM utilization in configuration II. This motivates the evaluation of a resource-sharing MPX decoding system, as further on proposed.

2.6.3 Resource-Shared Dual-Decoder Case Study

In the following, a case-study for a dual-decoder resource-sharing system will be presented. For the case-study, only the stereo and mono MPX decoders have been analyzed,

Configuration	Slices	DSP48	36k BRAMs
I: Mono, Mono	916 (57 %)	4 (22 %)	4 (67 %)
II: Stereo, RDS	1307 (81 %)	8 (44 %)	8 (133 %)
III: Stereo, Mono	1262 (78 %)	15 (61 %)	5 (83 %)
IV: Stereo, Stereo	1608 (100 %)	18 (100 %)	6 (100 %)

Table 2.11: MPX accumulated module resources for resource-sharing implementation.

leading to an implementation as depicted in Figure 2.37 with configurations I, III and IV (cf. Table 2.11).

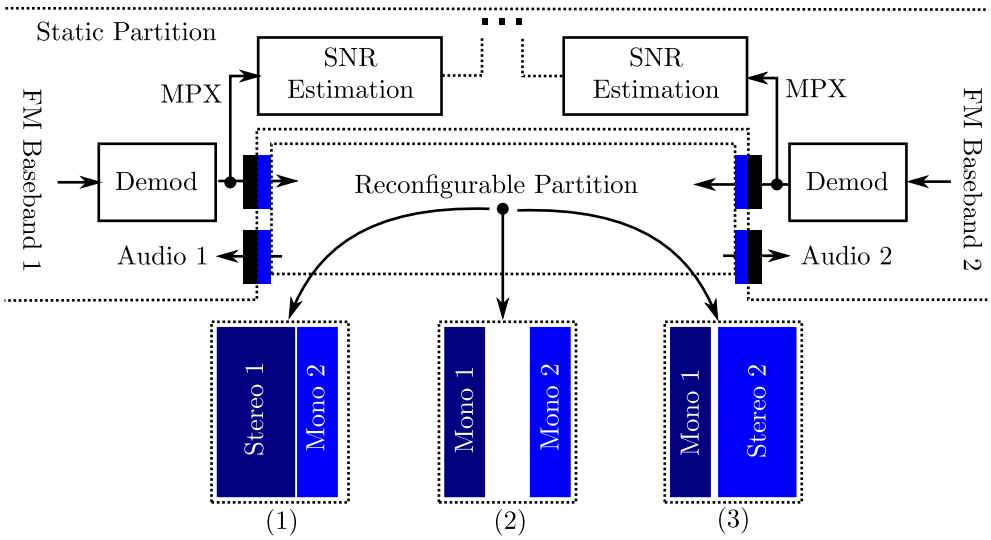


Figure 2.37: Dual resource-sharing reconfigurable FM receiver system design.

Two demod block partition interfaces exist at the left and right border of the reconfigurable partition. The resource-sharing approach prohibits the use of two concurrently operating large decoders, i.e. two stereo MPX decoder blocks. Given this constraint and comparing the requirements of a multi-island implementation with the resource-sharing approach, theoretically, the number of slices can be reduced by 22 %, compared to an unoptimized static implementation. For SNR-adaptive switching a prioritization between the left and right stereo DPR module must be included. Note that prioritization also means decision-coupling in terms of reconfiguration since the DPR module with higher priority can decide whether the other module will be reconfigured in case more resources are needed.

Next, the realization of the resource-sharing approach will be evaluated in a feasibility study.

Technical Feasibility and Floorplanning Considerations

Before putting effort into the design of a resource-sharing DPR system, it has to be ensured that the Xilinx FPGA fabric is capable to implement the required functionality, which, upon request, was positively confirmed by Xilinx. In addition, reconfigurability is

only ensured when the area constraints for the resource-sharing partitions are correctly defined. For example, the area must be partitioned in multiples of a configuration column. In reference to Table 1.1 in the introduction, a Virtex-5 configuration frame is 20 CLBs high and 1 CLB wide. Sharing the resources of a configuration column among two reconfigurable areas leads either to a failing implementation (`ERROR:XCad`) or to a situation where one receiver is likely to disturb the other (cf. [Mü11]). Since the static part of the system is imported after implementation, it is theoretically possible to share a configuration column with the reconfigurable partitions. However, practical evaluations have shown that this procedure leads to glitches in the static partition, which is why it is not recommended. Another reason not to mix the static and reconfigurable partition within one configuration column is that the reconfigurable areas should be aligned to the clock region boundaries, which are also 20 CLBs high and aligned with the configuration column layout. This is also the reason why the design of a horizontally expanding DPR area is preferred over a vertically expanding DPR area. Bearing these considerations in mind, constraining the placement of the sub-module to a limited resource region is accomplished using the following area group directives:

- Area constraints for **Black Box**:

```
AREA_GROUP "dpr_partition" RANGE=SLICE_X38Y20:SLICE_X53Y99;
AREA_GROUP "dpr_partition" RANGE=DSP48_X4Y8:DSP48_X5Y39;
AREA_GROUP "dpr_partition" RANGE=RAMB36_X3Y4:RAMB36_X4Y19;
```

- Area constraints for **Stereo 1**:

```
AREA_GROUP "stereo_left" RANGE=SLICE_X38Y20:SLICE_X47Y99;
AREA_GROUP "stereo_left" RANGE=DSP48_X4Y8:DSP48_X4Y39;
AREA_GROUP "stereo_left" RANGE=RAMB36_X3Y4:RAMB36_X3Y19;
```

- Area constraints for **Mono 1**:

```
AREA_GROUP "mono_left" RANGE=SLICE_X38Y20:SLICE_X43Y99;
AREA_GROUP "mono_left" RANGE=DSP48_X4Y8:DSP48_X4Y39;
AREA_GROUP "mono_left" RANGE=RAMB36_X3Y4:RAMB36_X3Y19;
```

- Area constraints for **Mono 2**:

```
AREA_GROUP "mono_right" RANGE=SLICE_X44Y20:SLICE_X53Y99;
AREA_GROUP "mono_right" RANGE=DSP48_X5Y8:DSP48_X5Y39;
AREA_GROUP "mono_right" RANGE=RAMB36_X4Y4:RAMB36_X4Y19;
```

- Area constraints for **Stereo 2**:

```
AREA_GROUP "stereo_right" RANGE=SLICE_X48Y20:SLICE_X53Y99;
AREA_GROUP "stereo_right" RANGE=DSP48_X5Y8:DSP48_X5Y39;
AREA_GROUP "stereo_right" RANGE=RAMB36_X4Y4:RAMB36_X4Y19;
```

For the feasibility study, audio drop-outs in adjacent partitions have not been concerned, i.e. the continuity of both MPX decoders is not a strict requirement. Hence, it is acceptable if the audio output of an MPX decoder turns silent during the reconfiguration of another DPR partition.

The approaches for the realization of a resource-sharing DPR design will be subsequently presented.

Evaluated Approaches

In this section, the approaches for a resource-sharing DPR FPGA system according to Figure 2.37 will be discussed. The Xilinx ISE 14.4 toolchain has been used for all subsequently described DPR implementations.

Hierarchical Partition-Based DPR Approach

In the first approach, hierarchical reconfiguration has been used with the partition-based design concept as described in Section 1.1.3 in the introduction. In opposition to the standard partition-based flow, for this approach the reconfigurable region is split into sub-regions, which are implemented by nested partitions (sub-partitions) inside the main DPR partition as visualized in Figure 2.38. For the realization of the three different modes of operation, four partial bitstreams are required. The main DPR partition is still required to block the resources for the sub-partitions during the implementation of the static design, such that no static logic is placed inside that area. Interfacing the sub-partitions is possible by forcing the partition pins to static locations inside the left and right sides of the DPR partition using the constraint directive `PIN "<instance>" LOC=<slice>"`. With the presented approach, the monaural decoder on one side could continue to operate without interruption, even if the decoder on the opposite side is being reconfigured from mono to stereo or vice versa. Moving the stereo decoder from the left side to the right side or vice versa needs two reconfiguration cycles. However, the reconfiguration from stereo to mono requires only one reconfiguration cycle.

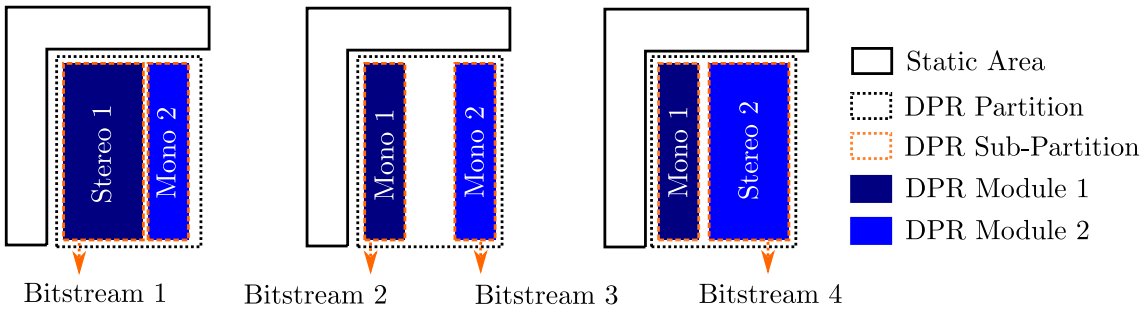


Figure 2.38: Dual resource-sharing reconfigurable system design.

Implementing the described hierarchical DPR system with the Xilinx toolchain failed with error 152 (`ERROR: HierarchicalDesignC`) due to missing support for nested reconfigurable partitions. Xilinx technical support [Mü11] confirmed that the implementation of resource-sharing systems by nested area groups is not possible. Since the partition-based flow turns out to be not suitable for hierarchical reconfiguration, in the next section, a difference-based resource-sharing system design approach will be introduced.

Difference-Based DPR Approach

The second resource-sharing system will be implemented with the difference-based partial reconfiguration flow as explained in Section 1.1.2 in the introduction. Similar to the partition-based flow, the difference-based flow requires the I/O interfaces between the static part and the DPR partition to be constrained to a fixed location. Additionally, it must be ensured that the clock networks of the left and right configurations are not interrupting each other during reconfiguration. Therefore, the difference-based flow requires to provide clock inputs to the left and right sub-modules at fixed locations. The position of the clock inputs are fixed by pin location constraints together with the I/O interfaces, similar to the partition-based implementation.

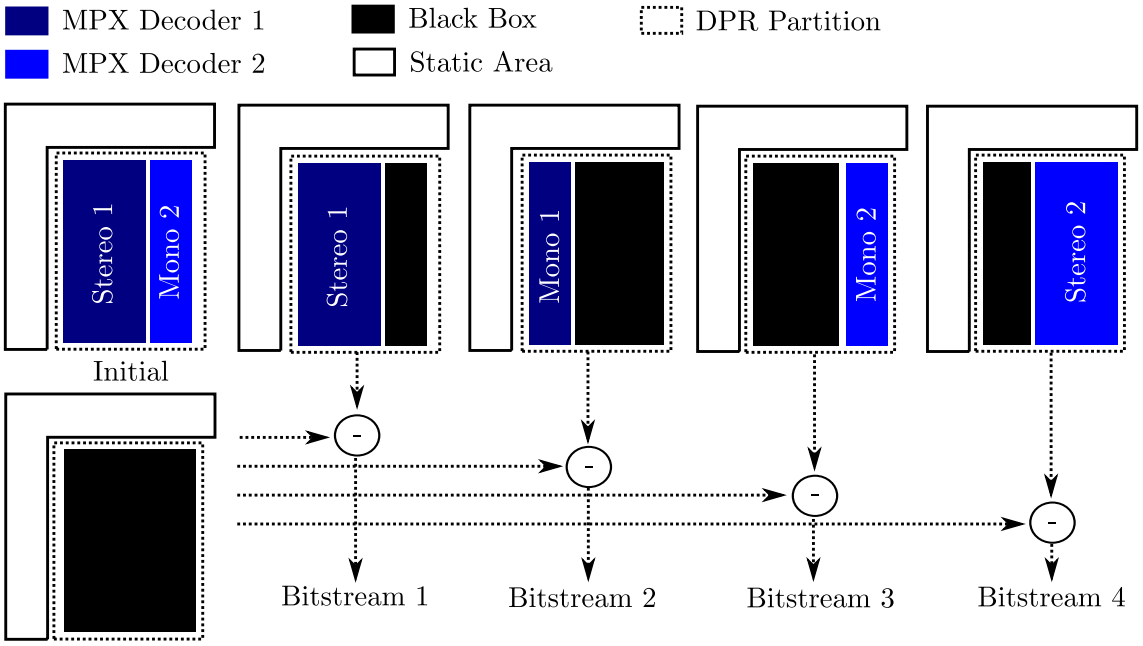


Figure 2.39: Dual resource-sharing reconfigurable system design.

Figure 2.39 shows the work-flow for bitstream generation. The initial system has been implemented from scratch using a stereo decoder on the left side and a mono decoder on the right side enclosed by the static partition. A black box system is then generated comprising of the static system and an empty DPR partition. Finally, four different MPX decoder realizations are implemented, each constrained as stated above and with a black box dummy region. Using the black box system and the implemented MPX decoder realizations, Xilinx Bitgen is used with the `-r` option to generate four differential bitstreams. Equal to the partition-based system, moving the stereo decoder from one side to another needs two reconfiguration cycles and the reconfiguration from stereo to mono requires only one cycle.

The analysis of the bitstreams using Xilinx Impact revealed that the placement of the resources could be constraint correctly, whereas the routing was leaking out of the constraint area into the neighboring partitions. The leakage is shown in Figure 2.40 and, since the routing crosses the black box region, leads to malicious configurations and interference with the neighboring DPR modules. Upon observation, the Xilinx technical support suggested using the undocumented area group constraint options `BOUNDARYCROSS=NO`

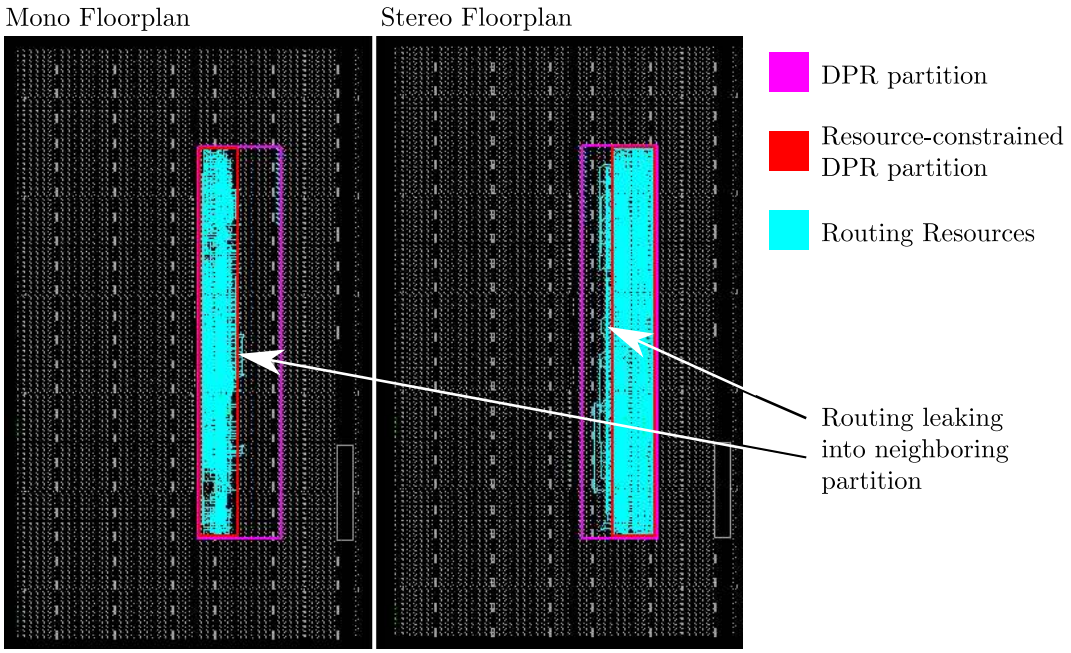


Figure 2.40: FPGA floorplan showing routing leakage for adjacent partitions.

and `CONTAINED=ROUTE`. However, even by using these constraints the routing was still prone to leaking into the neighboring partitions. Hence, using the difference-based system design approach the resource-sharing design could not be realized with Xilinx tools.

Since a resource-sharing DPR design could not be implemented using the vendor toolchain, the possibility of constraining the routing by using third-party tools is proposed. In [KB14] Koch et al. describe the tool-flow for hierarchical reconfiguration with the tool GoAhead. Instead of black boxes, the GoAhead tool utilizes blocker macros to constrain the routing to a certain region. The blocker macros are defined by XDL and added to the respective module before implementation. GoAhead has not been used in this work but has been reported to work with Virtex-5 FPGA designs in [BWF+13] together with a bus-macro reconfiguration approach.

2.7 Summary

A discussion of the related work for SNR-adaptive reconfigurable systems revealed that literature is missing a detailed analysis of SNR-adaptive receivers for partially reconfigurable FPGA systems and that resource-sharing self-adapting FPGA receiver systems are currently not employed. Therefore, a detailed outline of the design and implementation of a digital FM broadcast receiver has been given, together with an insight on the DPR module complexity. Next, the resource consumption of the receiver using a Xilinx Spartan-3 FPGA and the interfacing to audio and baseband I/O has been described. For the system to be self-adaptive, a novel FM SNR estimation technique based on the estimation of MPX band-gap noise has been presented and the routine has been dimensioned for the receiver in question. Furthermore, SNR threshold values for the different MPX decoding modules have been derived, by applying the ITU recommendations to the FM receiver design parameters. Since the Spartan-3A FPGA does not support

partial self-reconfiguration, the FM receiver and SNR estimator have been ported to a Xilinx Virtex-5 FPGA (XC5VSX50T) and the resource consumption for receiver realizations with different complexity has been outlined. Subsequently, the implementation of a self-adapting system using dynamic partial reconfiguration has been presented and the resource requirements for such a system have been quantified. Further on, a more fine-grained multi-island receiver implementation was obtained, by separating the MPX decoders from the FM demodulation stage. In the modified design, the MPX decoder modules strongly varied in resource consumption, which motivated the idea to share the resources of a DPR partition among multiple MPX decoders using hierarchical partial reconfiguration. It has been outlined that with such a system the number of slices can be reduced by 22 % compared to an unoptimized static implementation. Albeit Xilinx confirmed that the hardware supports hierarchical reconfiguration, the realization of the resource-sharing system failed since the Xilinx tool chain does neither support nested DPR partitions, nor does it support constraining the routing when difference-based reconfiguration is used. The chapter is concluded by a proposal on how to circumvent these drawbacks using third-party tools like GoAhead, which support hierarchical reconfiguration on Xilinx Virtex-5 FPGAs. Note that the elaborated results are not limited to Xilinx Virtex-5 devices but are also applicable to newer Xilinx FPGAs.

The feasibility of continuous temporal multiplexing of FPGA resources for the sequential execution of receiver chain elements will be analyzed in the following chapter.

3 Cyclic FPGA Reconfiguration for Sequential Processing of Receiver Modules

Sharing the logic resources among different DPR partitions can potentially reduce the resource occupation of a particular implementation as discussed in the previous chapter. In this chapter, approaches for sharing the logic resources in time by continuous reconfiguration of one single-island DPR partition will be of concern.

In traditional FPGA designs, a processing chain typically consists of concatenated processing elements (PEs), concurrently processing the data, i.e. all PEs may be active in parallel. An example of such a processing chain with the PEs $p_1, p_2, \ldots, p_5$ is depicted in Figure 3.1.

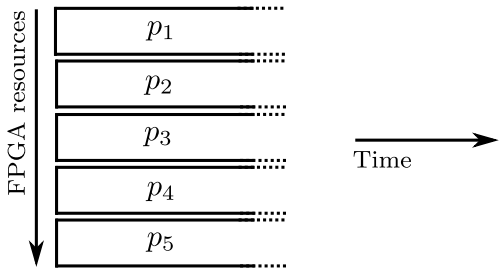


Figure 3.1: Concurrent execution of processing elements in traditional designs.

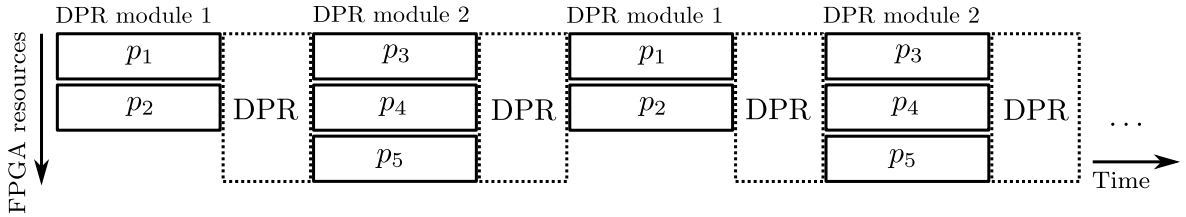


Figure 3.2: Trading FPGA resources against time using cyclic DPR.

For the subsequent analysis, it is assumed that only a subset of concatenated PEs process the data concurrently. Such a PE subset will in the following be referred to as DPR module. DPR modules will be activated *sequentially* in time, i.e. one after another, and thus can not process the data concurrently anymore but must process the data block-wise. Sequential processing of data is achieved by time-multiplexing the FPGA resources inside a DPR partition as expressed by Figure 3.2, where the first two

PEs are processed concurrently in DPR module 1 and the last three PEs are processed concurrently in DPR module 2. After a portion of data has been processed inside each DPR module, the DPR partition is reconfigured. The outlined approach trades FPGA resources against execution time and will subsequently be referred to as *cyclic DPR*[1].

Sequential module-wise processing requires the data to be processed in chunks. A chunk or *frame* refers to a finite portion of data propagating through a DPR module within one execution period. In this section, it will be shown that for real-time decoding the duration of a frame is tightly coupled to the processing delay and buffer capacity. Furthermore, the implications of cyclic DPR on the DPR module clock frequency will be outlined and it will be discussed how existing receiver chains have to be modified in order to be capable to work in a cyclic DPR environment. Hardware parameters of the DPR environment and of the DPR modules will be described by means of a system model for the feasibility analysis of cyclic DPR systems. The presented aspects are strongly connected and need to be analyzed in a combined context as shown further on.

Before the introduction of a suitable system model for cyclic DPR, the prior-art on time-multiplexing of FPGA resources will be provided and the contribution of this work will be classified. As a proof of concept, the effects of partitioning and sequential execution on a DAB receiver chain will be discussed in terms of real-time performance, FPGA resources and latency. In addition, the same DAB receiver chain will be used inside a cyclic DPR hardware implementation and analyzed in further detail. A feasibility study for a cyclic DPR system for DVB-T2 baseband decoding and a brief summary will conclude this chapter.

3.1 Related-Work and Contribution

As outlined in the first chapter of this work, the idea of time-multiplexing FPGA resources to virtually enlarge the available logic gained momentum with Trimberger's publication "*A Time-Multiplexed FPGA*" (cf. Section 1.1). An architecture for real-time operation of run-time reconfigurable signal processing systems was introduced by Eilers et al. in [ESK03]. The authors present an analysis of different buffering schemes to hide the reconfiguration latency of a Xilinx Virtex-II 1500 FPGA. In contrast to this work, cyclic reconfiguration or the design of a reconfiguration system model are not in the scope of their research. In 2007 Claus et al. presented a reconfigurable design based on a Xilinx Virtex-II Pro FPGA, where multiple reconfiguration columns can be dynamically reconfigured to switch between different video-processing functions [CZMS07]. Here, similar to previous works, the FPGA is not cyclically reconfigured, but the DPR module configurations are written to the ICAP on demand. The idea of cyclic time-multiplexing of FPGA resources to realize a resource-efficient radio receiver was presented by Ihmig et al. in 2008 and introduced as "*Reconfigurable sequential approach*" (cf. [IAH08]). Ihmig's work includes a specification of the system components for a sequential reconfigurable architecture, for example: external memory, a reconfiguration scheduler and a buffer manager. Similar components will be introduced in the cyclic DPR system model of this work. Ihmig also provides a task-graph for time-slotted processing of DAB receiver components, which can be directly related to the reconfiguration flow model presented in this chapter. Since no analytic framework is derived to determine the reconfiguration

[1] In literature, sometimes the term time-multiplexed FPGA or TM-FPGA is used.

system performance, the influence of buffering on the processing delay of the system are not further analyzed by Ihmig. Based on a modified DAB receiver implementation for non-reconfigurable systems, a receiver implementation tailored for cyclic DPR will be outlined in the next sections. Inspired by the idea of cyclic reconfiguration, in [PLMK09] Popp et al. derived a high-level reconfiguration cost model given a "*basic reconfigurable architecture*" comprising of a reconfigurable FPGA, a configuration controller and external memory. Popp's hardware architecture and timing model is equal to the model presented in this work, albeit Popp did not explicitly distinguish between context loading and saving time, but assumed intermediate data loading and saving to have the same duration. Popp's work does consider memory throughput for context loading and saving, but not for intermediate buffering during module execution. Similarly, in [FIZS11] saving and loading of the DPR module context was not explicitly distinguished but combined in a single term. In turn, Becker et al. mention buffering and buffer sizes given the throughput constraints in [BLC09], but do not include context loading and saving in their model and also not the delay induced when multiple modules are processed sequentially. Finally, although Popp and Becker introduce system parameters for cyclic DPR, they miss to show detailed architectures depicting how such a reconfigurable system could be realized in practice.

Based on the outlined works of others, and on own works related to cyclic DPR, an extended cyclic DPR system model will be presented. Using a modified DAB receiver implementation, potential resource savings and real-time performance of the sequentially executed DAB module chain will be quantified as discussed in [FIIS12]. In addition, the effects of cyclic sequential processing of DVB-T2 baseband decoder modules will be outlined in a proof-of-concept study, as published in [FISS12].

In relation to the state-of-the art, the major contributions presented in this chapter comprise of:

- A framework for the analysis of cyclic DPR system including processing delay, memory capacity and throughput constraints.
- A feasibility analysis and implementation of a DAB receiver chain for the usage inside a cyclic DPR system.
- A hardware implementation and reconfiguration scheduler to quantify the resource consumption of a cyclic DPR receiver system for DAB.
- A feasibility analysis of a cyclic DPR-based accelerator for DVB-T2 baseband processing.

Prior to giving a profound explanation in how far the previously mentioned design parameters are related, a system model of a single-island time-multiplexed FPGA architecture will be derived next.

3.2 System Model

Although time-multiplexing models have been discussed in related works, it is necessary to enhance these frameworks for the modularization of reconfigurable sequential processing chains with explicit focus on using DPR for real-time processing. In this section, a timing and delay model for the sequential execution of receiver chain elements will be presented. The model covers the influences of partial FPGA reconfiguration on real-time constraints and resource consumption and assists in quantifying the real-time capability of a specific DPR-based hardware implementation.

3.2.1 Cyclic Reconfiguration Flow

A functional element of a processing chain will further be denoted as PE. A PE is defined to have a known execution time, no feedback to previous elements and a predefined maximum input and output data throughput required for real-time processing. Inside the processing chain, the PE execution order is assumed to be strictly sequential and inherently cyclic. Digital receiver chains belong to this class of processing chains and the functional subset of a signal processing chain such as filtering, channel decoding or demodulation can be defined as a processing element (cf. [SFHB12]). Furthermore, it is assumed that a processing element mainly interacts with his adjacent neighboring PE and that the data throughput of feedback paths can be neglected.

Let a processing chain be defined by a sequence of N independent concatenated PEs as depicted in Figure 3.3, where the output data throughput of the n-th element is denoted by γ_n. Since the input data throughput of the n-th PE is equal to the output data throughput of PE $n-1$, the input data throughput of the n-th PE is γ_{n-1}.

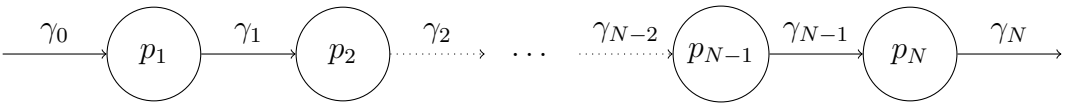


Figure 3.3: Sequential chain of processing elements.

It is important to mention, that the data throughput is related to the PE functionality and is not to confuse with the maximum throughput of the hardware interface used to carry the information from or to a PE. Given an FPGA hardware implementation of the processing elements, it is assumed that the resource consumption and the number of execution cycles of all PEs are known a priori and that there exists a hardware communication interface with a fixed maximum data throughput for the data transfer from and to the PEs. The linear data dependency of the PEs allows to wrap a concatenated subset of PEs into a larger PE entity, while still preserving the properties stated in this section. Such a larger PE entity will be further be referred to as *reconfigurable module* or DPR module. A DPR module will be defined to include the hardware functionality of one or multiple PEs. Since a DPR module only contains a subset of the PEs of the chain, the resource requirements to realize the functionality of one module is smaller in comparison to the accumulated logic utilization of all PEs in a static system, which is the major motivation for the cyclic DPR approach.

In accordance to the properties of the PEs, a chain of DPR modules is assumed to have no feedback elements and each module is carrying the functionality of one or more

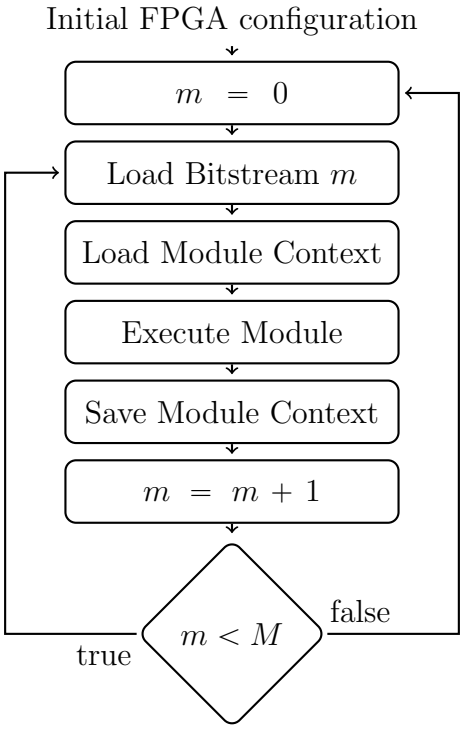


Figure 3.4: Cyclic module reconfiguration flow graph.

disjoint processing elements. Within each module reconfiguration cycle a new bitstream is presented to the ICAP and the m-th DPR module is loaded into the DPR partition. The DPR module is then reset and the context memory of the DPR module is eventually restored. During the execution a frame, a chunk of data is transferred from memory to the input of the module and the output produced by the DPR module is written back to memory. Upon completion of a frame the state of the DPR module (i.e. its context) is written to memory. Then, the bitstream of the next DPR module is written to the ICAP. The process starts over with the first DPR module after the M-th DPR module has been processed. The outlined sequence is shown in Figure 3.4. According to Popp and Feilen (cf. in [PLMK09] and [FIZS11]), the duration of one DPR module chain execution cycle will be denoted by T_{CYC} and can be expressed by

$$T_{\mathrm{CYC}} = \sum_{m=1}^{M} T_{\mathrm{DPR},m} + T_{\mathrm{LD},m} + T_{\mathrm{EX},m} + T_{\mathrm{SV},m}, \tag{3.1}$$

where $T_{\mathrm{DPR},m}$ represents the time to write the bitstream of the m-th DPR module to the ICAP, $T_{\mathrm{LD},m}$ represents the time to initialize the m-th DPR module, $T_{\mathrm{EX},m}$ is the time to process a frame with the m-th DPR module and $T_{\mathrm{SV},m}$ is the time to save state of the m-th DPR module to memory. As a reference, the cycle time parameters are summarized in Table 3.1.

Many broadcasting standards follow a certain transmission framing (cf. Section 1.2), such that processing states become cyclically finite, which can be exploited in the design of DPR systems to reduce the effort for context storage and recovery. This can also be beneficial in case context saving and recovery can not be realized, e.g. if access to the internal DPR module state is not provided.

Symbol	Description
$T_{\mathrm{DPR},m}$	Time to write the DPR module m into the DPR partitions.
$T_{\mathrm{LD},m}$	Time to reset, load the state and initialize the m-th DPR module.
$T_{\mathrm{EX},m}$	Time to process a chunk of data inside the m-th DPR module.
$T_{\mathrm{SV},m}$	Time to save the state of the m-th DPR module.

Table 3.1: Cyclic DPR module-related task durations according to Popp and Feilen.

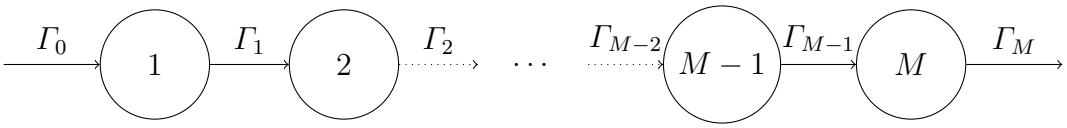


Figure 3.5: Sequential chain of DPR modules.

In order to quantify the duration of a reconfiguration cycle in Equation 3.1, the timing parameters need to be expressed by implementation-specific parameters of a DPR module. Furthermore, it will be shown that the execution time, i.e. the processing duration, of the sequential chain of modules also depends on the throughput of the memory interface and ICAP controller. For both it is necessary to include the hardware-related parameters in the system model and relate them to the parameters of the DPR module.

3.2.2 Module Throughput and Data Framing

As discussed in the beginning of this chapter, a chunk of data will be fed to each DPR module, denoted as *frame*[2]. A frame corresponds to a portion of data with a certain duration and with a defined rate. It is assumed that each frame supplied to the DPR chain has a constant average data rate of γ_0 and a pre-defined duration $\mathrm{T_{FRAME}}$.

In the following, Γ_{m-1} will describe the minimum DPR module input throughput and Γ_m will express the minimum output throughput of a DPR module as shown in Figure 3.5. During the execution period of a DPR module, an equivalent of $\mathrm{T_{FRAME}}$ seconds of data must be consumed and produced. The chain of DPR modules is defined to be real-time capable, if it can consume the periodically sampled data from the source at the same or at a higher rate as the source produces the data. Therefore, the instantaneous input and output throughput of a DPR module must be higher than the average data throughput of a PE. From this observation follows that, in order to be real-time capable, the input and output data throughput of the DPR module must at least be increased by a factor of $\frac{\mathrm{T_{FRAME}}}{T_{\mathrm{EX},m}}$, which increases the data rate at the input and output of a DPR module according to

$$\begin{aligned} \Gamma_{m-1} &\geqslant \frac{\mathrm{T_{FRAME}}}{T_{\mathrm{EX},m}} \cdot \gamma_{\mathrm{IN},m} \\ \Gamma_m &\geqslant \frac{\mathrm{T_{FRAME}}}{T_{\mathrm{EX},m}} \cdot \gamma_{\mathrm{OUT},m}, \end{aligned} \tag{3.2}$$

[2]In this chapter, a frame is defined as a portion of data taken from a periodically sampled source and must not be confused with an FPGA configuration frame.

where $\gamma_{\mathrm{IN},m}$ describes the throughput of the PE or chain input feeding the m-th DPR module and $\gamma_{\mathrm{OUT},m}$ denotes the throughput of the first PE of the subsequent DPR module in the chain or, alternatively, of the chain output. The outlined relations are depicted in Figure 3.6.

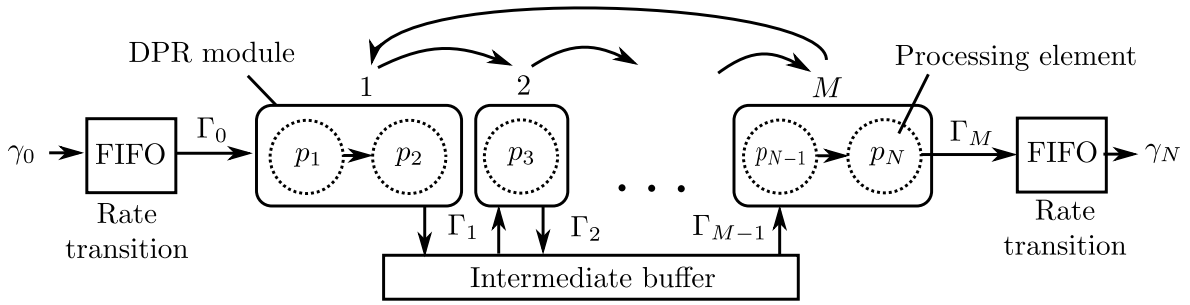


Figure 3.6: Cyclic execution flow-graph of DPR modules with throughput annotation.

Recall that, if the DPR module input and output throughput values can not be achieved, the real-time requirements of the system are not satisfied. If the input and output throughput values are higher than the required minimum, there will be an idle time left until the arrival of the next input frame, i.e. in this case $T_{\mathrm{CYC}} < \mathrm{T}_{\mathrm{FRAME}}$ and $\mathrm{T}_{\mathrm{IDLE}} = \mathrm{T}_{\mathrm{FRAME}} - T_{\mathrm{CYC}}$. This leads to the following observation: Since the data arrives at the first DPR module with a rate of γ_0 and since $\Gamma_0 > \gamma_0$, it is impossible to use the DPR system without prior buffering. In the following, buffering is assumed to be accomplished by a FIFO buffer with capacity $\mathrm{T}_{\mathrm{BUF}} \geqslant \mathrm{T}_{\mathrm{FRAME}}$. In case the FIFO buffer can not be written to and read from at the same time, a double-buffer with capacity $2 \cdot \mathrm{T}_{\mathrm{BUF}}$ will be required as shown in [FIZS11]. To avoid an infinite growth of the input buffer, the sequential chain of DPR modules must process the data within a duration of at least T_{CYC}, where the first module consumes a whole input frame, such that the DPR cycle time is upper bounded by the duration of the input data frame, i.e.

$$\mathrm{T}_{\mathrm{FRAME}} \geqslant T_{\mathrm{CYC}}. \tag{3.3}$$

Due to time-multiplexing of the DPR partition and execution of the DPR modules, the DPR chain exhibits a processing delay. Buffering of the input signal introduces an additional latency in the order of a duration of a frame. The accumulated DPR system latency will subsequently be referred to as T_{DELAY}. Expressing the delay exactly requires precise knowledge of the access latency and execution timing of the DPR modules. The relations of the timing parameters described so far are depicted in Figure 3.7. By inspecting the timing diagram, the DPR processing delay can be defined as the time between the arrival of the first sample of an input frame to the time the first output sample of the last DPR module has been written to memory. Alternatively, the delay could be described by the time where the first DPR module just finished processing the last sample of an input frame to the time the last output sample of the last DPR module has been written to memory. There are other possibilities to describe the input-to-output delay, but they will always be related to the execution time of the first DPR module and the last DPR module in a subsequent DPR cycle. In the figure, the input-to-output delay is

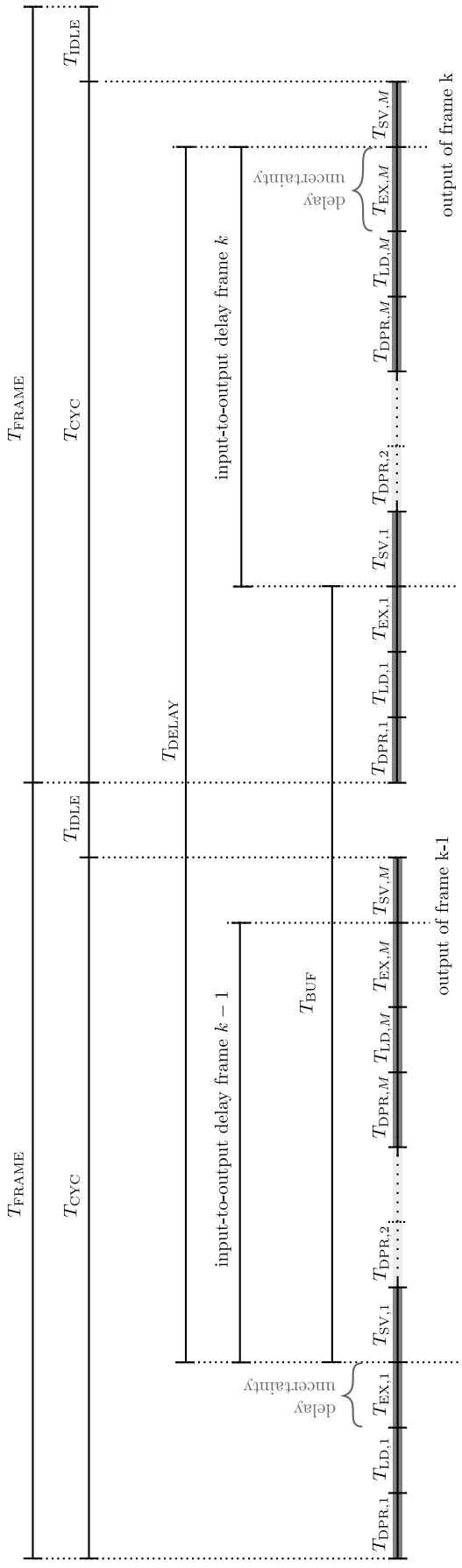


Figure 3.7: Module execution timing diagram and DPR processing delay.

defined as the time between the end of the execution of the first module and the end of the execution of the last module in the next cycle, such that

$$T_{\mathrm{DELAY}} = \mathrm{T}_{\mathrm{FRAME}} - T_{\mathrm{EX},1} - T_{\mathrm{LD},1} - T_{\mathrm{CFG},1} + T_{\mathrm{CYC}} - T_{\mathrm{SV},M}. \tag{3.4}$$

In case no implementation knowledge about the DPR module is available, the worst-case delay must be defined as the time between the beginning of the execution of the first module and the end of the execution of the last module in the next cycle, i.e.

$$T_{\mathrm{DELAY}} = \mathrm{T}_{\mathrm{FRAME}} - T_{\mathrm{LD},1} - T_{\mathrm{CFG},1} + T_{\mathrm{CYC}} - T_{\mathrm{SV},M}.$$

Given that the sum of the idle time, the configuration time and the loading and saving times are negligible in relation to the cycle time, the worst-case delay can be upper bounded by

$$T_{\mathrm{DELAY}} \leqslant 2 \cdot \mathrm{T}_{\mathrm{FRAME}}. \tag{3.5}$$

Note that Equation 3.5 serves also as an upper bound for real-time processing, i.e. for Equation 3.3 to be satisfied. The presented reconfiguration flow and the DPR module constraints will be used for the analysis in the following sections of this chapter.

Next, a hardware model will be introduced to determine the application-specific processing delay and buffering requirements of a cyclic DPR implementation.

3.2.3 Hardware Model

The hardware model describes all hardware components which have an influence on the reconfiguration timing. Consequently, it enables to express the timing parameters that make up the cycle time in Equation 3.1 by hardware-related parameters. A signal-flow diagram of the components of the system model is shown in Figure 3.8, which is similar to the models presented in literature (cf. [ESK03], [PLMK09] and [IAH08]). In this work, the hardware model comprises of a dynamically reconfigurable FPGA with an ICAP reconfiguration interface and an external memory peripheral. The external memory is interfaced by a memory controller from which access is provided to the static partition of the FPGA. Although modern FPGAs have internal BRAM resources which could be used as storage memory, these resources are rather limited in capacity as compared to external memory. Therefore, without loss of generality, external memory will be used in the model for bitstream storage for the DPR modules and as an input and intermediate buffer for processing. Also, due to the restrictions that I/O elements can not be placed inside the reconfigurable partition (cf. Section 1.1), the model implies that external memory will only be accessible from within the static FPGA partition. For internal buffering, i.e. for intermediate buffering inside the DPR module, the BRAM resources of the DPR partition can be used as scratch buffers.

Upon startup, the FPGA is configured with one static partition containing the reconfiguration FSM, a memory controller and a dynamic partition for the execution of the DPR module. The baseband input and data output ports are bridged to the static part of the FPGA via external I/O pins. For the description of the system, two sets of parameters will be introduced: hardware-related parameters, listed in Table 3.2, and reconfigurable system design parameters, outlined in Table 3.3. Typically, the hardware-related parameters are defined by an FPGA platform, whereas the design-related parameters depend on the functionality of the DPR modules. Thus, if the platform is fixed, the designer can

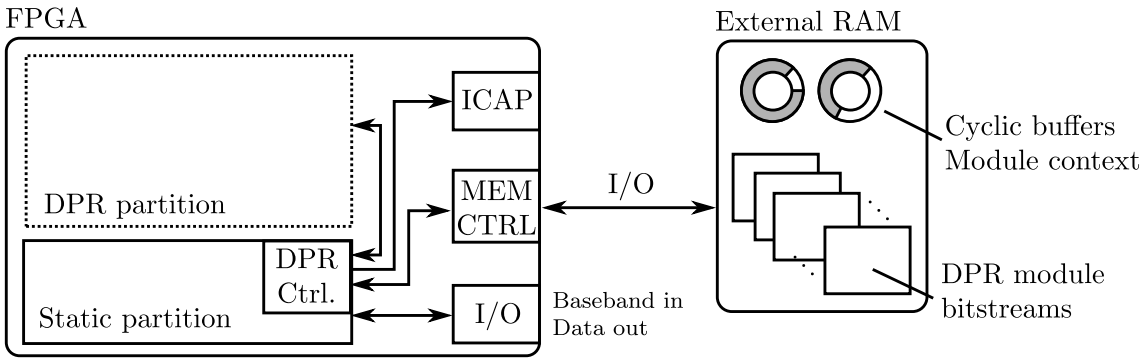


Figure 3.8: Single-island FPGA reconfiguration hardware model.

only have an influence on the DPR system performance by modifying the design-related parameters, which are determined by the actual DPR module implementation.

Symbol	Description
f_{EX}	DPR partition clock frequency in cycles/s.
f_{ICAP}	Configuration interface (ICAP) clock frequency in cycles/s.
$W_{\mathrm{B,ICAP}}$	Configuration interface width in bits.
Γ_{ICAP}	Configuration interface throughput in bits/s.
γ_{MEM}	Mean memory controller I/O throughput in bits/s.

Table 3.2: Hardware-related system model parameters.

Symbol	Description
$N_{\mathrm{B,BIT},m}$	Configuration bitstream size in bits.
$N_{\mathrm{C,EX},m}$	Execution duration in clock cycles.
$N_{\mathrm{C,LD},m}$	Number of clock cycles for context loading.
$N_{\mathrm{C,SV},m}$	Number of clock cycles for context saving.
$N_{\mathrm{B,LD},m}$	Context loading interface width in bits.
$N_{\mathrm{B,SV},m}$	Context saving interface width in bits.

Table 3.3: DPR module implementation-related parameters.

Recall that the reconfiguration flow depicted in Figure 3.4 is cyclic and a priori defined. Hence, the memory access pattern for the different reconfiguration activities is also predefined. The memory access pattern of the model for one reconfiguration and execution cycle is presented in Figure 3.9, where the memory controller is assumed to have a minimum of two write ports and two read ports. The graph shows four different memory access states, related to the four DPR cycle time parameters as outlined in Table 3.1, i.e. FPGA configuration, context loading, DPR module execution and context writeback. Receiving a continuous stream of input data at a fixed rate of γ_0 and transmitting a continuous output stream at a rate of γ_N (cf. Figure 3.3) is depicted in the diagram by a continuous write/read I/O transfer spanning all four DPR module processing tasks.

Let the average throughput of the external memory interface be defined by γ_{MEM}. For cyclic DPR operation to be possible, it is required that the memory interface can provide at least the throughput needed for a continuous transfer of frame information from the baseband source to the input buffer and from the output buffer to the output port. Hence, in order to leave enough headroom for other I/O operations it is necessary that

$$\gamma_{\mathrm{MEM}} > \gamma_0 + \gamma_N.$$

Deducting the throughput required for continuous transfer to the input buffer and from the output buffer, the mean memory throughput is reduced, such that

$$\Gamma_{\mathrm{MEM}} = \gamma_{\mathrm{MEM}} - \gamma_0 - \gamma_N, \tag{3.6}$$

where Γ_{MEM} reflects the memory throughput available for other tasks during DPR module execution. Note that the output FIFO buffer included in Figure 3.6 might be omitted if the FPGA data output port is able to cope with the high-throughput bursts of the last DPR module. This is the case if the data sink provides a buffer itself as shown in the subsequently presented DAB receiver feasibility study. In this case, the external memory interface gains a throughput margin of γ_N. In case the external memory exhibits access latencies, it is important to optimize the memory access pattern for input and output data streaming for maximizing the average memory throughput Γ_{MEM}.

For further analysis the maximum ICAP throughput will be defined by

$$\Gamma_{\mathrm{ICAP}} = f_{\mathrm{ICAP}} \cdot W_{\mathrm{B,ICAP}}, \tag{3.7}$$

where f_{ICAP} is the ICAP clock frequency and $W_{\mathrm{B,ICAP}}$ the interface width in bits. Typically, the throughput of the external memory interface is much higher than the throughput of the ICAP interface, i.e. $\Gamma_{\mathrm{MEM}} \gg \Gamma_{\mathrm{ICAP}}$. In this case, the ICAP throughput is dominating the reconfiguration time (cf. [BLC09]). On the other hand, if the memory interface is slower than the ICAP, the memory throughput determines the reconfiguration time. Accounting for both cases, the reconfiguration time of the DPR partition can be expressed by

$$T_{\mathrm{DPR},m} = \max\left(\frac{N_{\mathrm{B,BIT},m}}{\Gamma_{\mathrm{ICAP}}}, \frac{N_{\mathrm{B,BIT},m}}{\Gamma_{\mathrm{MEM}}}\right), \tag{3.8}$$

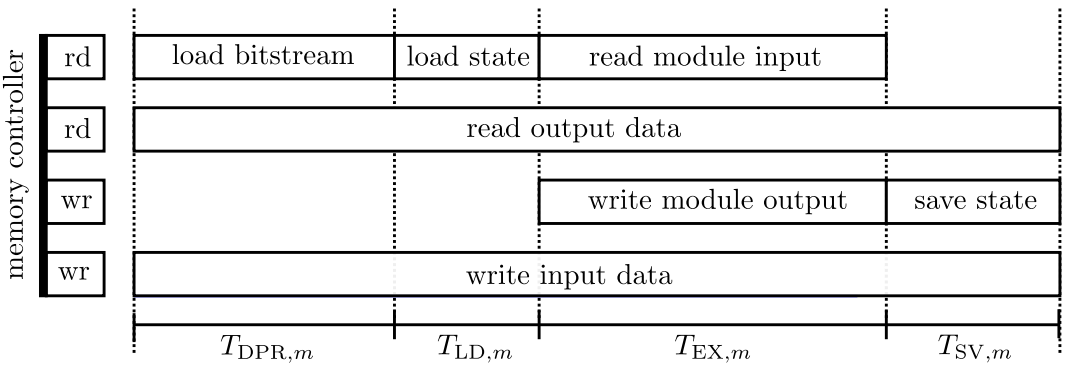


Figure 3.9: Memory access pattern during DPR module processing.

where $N_{\mathrm{B,BIT},m}$ defines the size of the partial bitstream in bits. Although the DPR partition is assumed to be constant in area, in case differential reconfiguration is used, the size of the partial bitstream may vary for different DPR modules. Given the clock frequency of the FPGA and the number of execution cycles per DPR module, the module execution time parameter as listed in Table 3.1 can be expressed as

$$T_{\mathrm{EX},m} = \max\left(\frac{N_{\mathrm{C,EX},m}}{f_{\mathrm{EX}}}, \frac{\Gamma_{m-1} + \Gamma_m}{\Gamma_{\mathrm{MEM}}} \cdot \mathrm{T_{FRAME}}\right), \tag{3.9}$$

with $N_{\mathrm{C,EX},m}$ being the number of execution cycles of the m-th DPR module required to process one frame of duration $\mathrm{T_{FRAME}}$ and f_{EX} being the clock frequency of the DPR module. This means that the execution time is either limited by the processing rate of the FPGA or by the data throughput of the memory interface. Reducing the module execution time by optimizing the implementation reduces the processing delay and the memory transfer overhead of the system. Clearly, the number of execution cycles per DPR module is one of the most important parameters and depends on the amount of data that has to be processed while the DPR module is active. Given a fixed portion of input data, the faster the data can be processed within one reconfiguration cycle, the smaller the input frame duration (cf. Equation 3.3). In turn, the longer the activity time, the longer the data at the input must be held back to be processed in the next iteration. Holding back the data at the input introduces processing delay, which might not be tolerable for the processing system in question (cf. [FIZS11]). The effects of reducing or increasing the data throughput of a DPR module will be discussed in the next section.

The time to load and save the FPGA context, i.e. the bringup and shutdown time of a DPR module, depends on the functionality of the module itself. Clearly, stateless modules are preferable as they reduce the configuration cycle time with $T_{\mathrm{LD},m} = T_{\mathrm{SV},m} = 0$. If the modules are not stateless, the times for context switching can be evaluated by

$$\begin{aligned} T_{\mathrm{LD},m} &= \max\left(\frac{N_{\mathrm{C,LD},m}}{f_{\mathrm{EX}}}, \frac{N_{\mathrm{B,LD},m}}{\Gamma_{\mathrm{MEM}}}\right) \text{ and} \\ T_{\mathrm{SV},m} &= \max\left(\frac{N_{\mathrm{C,SV},m}}{f_{\mathrm{EX}}}, \frac{N_{\mathrm{B,SV},m}}{\Gamma_{\mathrm{MEM}}}\right), \end{aligned} \tag{3.10}$$

where $N_{\mathrm{C,LD},m}$ and $N_{\mathrm{C,SV},m}$ refer to the number of cycles for context loading and saving. Similar to the other cycle time parameters, the context recovery process is dominated by the slowest interface, which means that either the memory throughput or the FPGA clock frequency will be dominating.

Using Equations 3.8, 3.9 and 3.10 all timing parameters in Equation 3.1 can be expressed by hardware-related and implementation-related parameters. In the next section, an existing DAB receiver implementation will be modified to operate in a cyclic DPR environment and the cycle time parameters will be determined by FPGA platform parameters and implementation-specific parameters of the DPR modules.

3.3 Cyclic DPR for DAB Receivers - Part I: Feasibility Analysis

In addition to the brief introduction to DAB outlined in Section 1.2, for the following analysis it is important to introduce the DAB framing architecture for transmission mode I as presented in Figure 3.10. The complex baseband signal is transmitted as a sequence of DAB frames, where each DAB frame comprises of four common interleaved frames (CIFs), one fast information channel (FIC) frame, a null OFDM symbol and a phase reference OFDM symbol. A CIF is made up of 18 OFDM symbols and the FIC comprises of three OFDM symbols. At a sample rate of 2.048 MS/s a continuous stream of 2552 complex baseband samples make up one OFDM symbol with 2048 samples enclosing the useful carrier information and 504 samples of guard interval, also denoted as cyclic prefix. Hence, a DAB frame consists of 76 OFDM symbols of duration T_S and one null symbol of duration T_0. If the null symbol is being transmitted, no energy is radiated, such that it can be used to determine the start of a DAB frame. The reference symbol is needed for differential demodulation of the first OFDM symbol in the DAB frame and can optionally be used for channel estimation. The lengths of the described entities are listed in Table 3.4. From the CIFs the main service channel (MSC) can be extracted by decoding a selected set of OFDM symbols as defined in the FIC. The MSC carries the audio and program associated data (PAD) payload information and the FIC is used to signal configuration parameters and service information. The information in the MSC and in the FIC are both encoded by a convolutional channel code and subsequently interleaved and DQPSK-mapped onto the orthogonal subcarriers. A DAB receiver needs to synchronize to the transmitted baseband signal to subsequently decode the payload information of the FIC and MSC for appropriate audio playback.

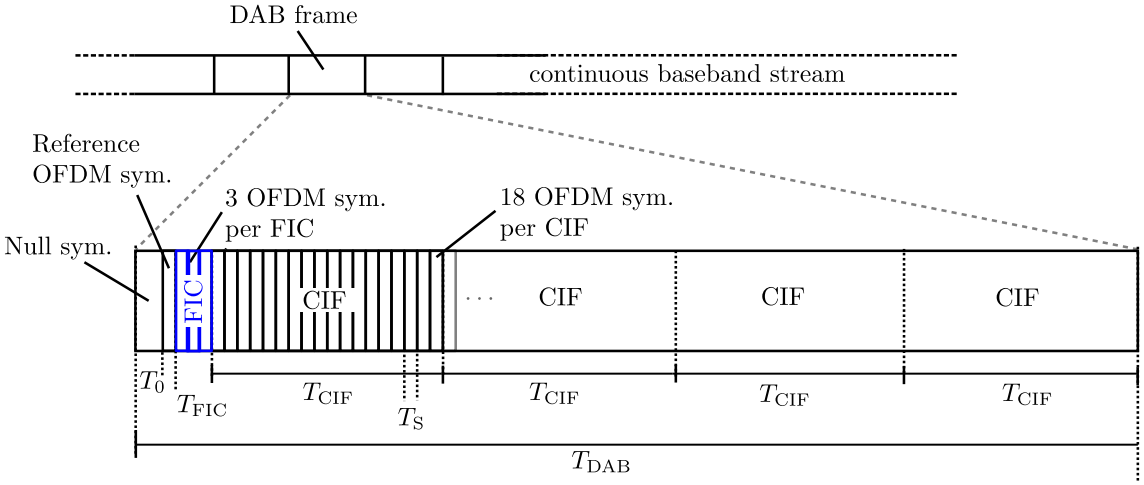


Figure 3.10: Framing structure of DAB baseband stream.

An FPGA-based DAB baseband receiver chain is shown in Figure 3.11. The receiver chain was developed by Ihmig et al. (cf. [IAH10]) to operate on a high-performance Lyrtech SDR platform using a Xilinx Virtex-4 SX35 FPGA. In order to operate on a low-cost FPGA platform from ZTEX equipped with a Xilinx Spartan-6 FPGA the receiver chain was migrated and extended by Gnadl in [Gna12]. In the Spartan-6 implementation, the FPGA acts as a baseband decoder accelerator for a PC, where the baseband is

Frame type	Symbol	Duration
OFDM symbol	T_S	$\frac{2{,}552}{2{,}048}$ ms
Null symbol	T_0	$\frac{2{,}656}{2{,}048}$ ms
FIC	T_{FIC}	$3 \cdot T_S$
CIF	T_{CIF}	$18 \cdot T_S$
DAB frame	T_{DAB}	$T_0 + T_S + T_{FIC} + 4 \cdot T_{CIF} = 96\,\text{ms}$

Table 3.4: Duration of the different DAB framing units.

supplied via USB and the decoded audio transport stream is send back via the same interface to the PC for playback. Although a Spartan-6 FPGA has been used for the implementation, the methods described in this chapter can be applied to all Xilinx FPGAs with partial self-reconfiguration support. Similar to the FM receiver presented in Chapter 2, the DAB receiver has been implemented with the goal in mind to develop a laboratory prototype for research. It has neither been explicitly optimized in terms of reception robustness nor in terms of resource utilization.

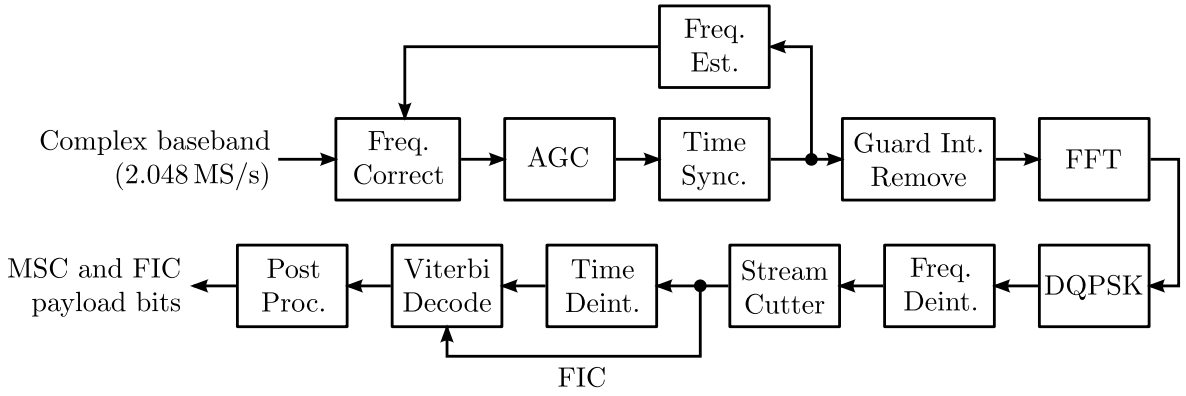


Figure 3.11: DAB receiver processing element chain.

The receiver has been designed for terrestrial reception and decodes DAB signals in transmission mode I [ets06]. Although the architectural concepts derived in this section can be transferred to all DAB transmission modes, the DAB receiver timings are different in modes II, III and IV. The PC provides the complex baseband signal with a sample rate of 2.048 MS/s in 8 bit two's complement notation for the in-phase and quadrature channels to the FPGA. After frequency offset compensation, the baseband signal is amplified by an automatic gain control (AGC) stage and subsequently processed by a guard-interval correlation unit and energy-detection unit to estimate and track the frequency offset and the DFT window offset. The DQPSK-modulated carriers are obtained by feeding the unguarded part of the OFDM symbol to a 2048-point FFT of which 1536 useful carriers are subsequently evaluated. Next, incoherent (differential) demodulation is used to equalize the channel distortions in the frequency-domain. Amplitude and phase information of the carriers are then fed to a frequency-domain de-interleaver and a payload bit extraction and time de-interleaving stage for the MSC. Finally, the soft-bits of the FIC and MSC are supplied to a Viterbi decoder for channel decoding. The post-processing stage includes a pseudo-random binary sequence (PRBS) for energy

Chain Element	Functional Description
1: Freq. Correct	Frequency offset compensation by complex multiplication.
2: AGC	Automatic gain estimation and correction for valid amplitude range.
3: Time Sync.	DAB frame and DFT window offset estimation and tracking.
4: Freq. Est.	Frequency offset estimation by guard interval correlation.
5: Guard Int. Remove	Guard interval removal.
6: FFT	Fast Fourier transform with 2048 bins.
7: DQPSK	Frequency-domain differential quadrature phase-shift keying.
8: Freq. Deint.	Carrier de-interleaving of current symbol.
9: Stream Cutter	Extracting the MSC bits needed for time-deinterleaving.
10: Time Deint.	Bit-wise convolutional de-interleaving.
11: Viterbi Decode	Viterbi channel decoding.
12: Post Proc.	Payload frame post-processing.

Table 3.5: DAB receiver processing elements description.

Figure 3.12: ZTEX USB-FPGA-Module 1.11c with Xilinx Spartan-6 LX25.

dispersal and forwards the payload bits to the USB transport layer and then back to the PC. The receiver chain elements are listed in sequential order in Table 3.5.

The receiver has been implemented on a ZTEX USB-FPGA module 1.11c mounted on a debug board (cf. [ZTE]) as shown in Figure 3.12. The ZTEX module comprises of an XC6SLX25 Spartan-6 FPGA, a Cypress CY7C68013A EZ-USB FX2 microcontroller (cf. [Cyp]) running at a clock frequency of 48 MHz, an external 64 MByte DDR synchronous dynamic random access memory (SDRAM) with a clock frequency of 200 MHz and a 128 kbit electrically erasable programmable read-only memory (EEPROM). The clock frequency of the FPGA can be derived from the 48 MHz clock signal provided by the crystal connected to the EZ-USB microcontroller using DCM and PLL resources in the FPGA. The firmware of the microcontroller handles the USB communication to the FPGA and is read from the EEPROM on startup.

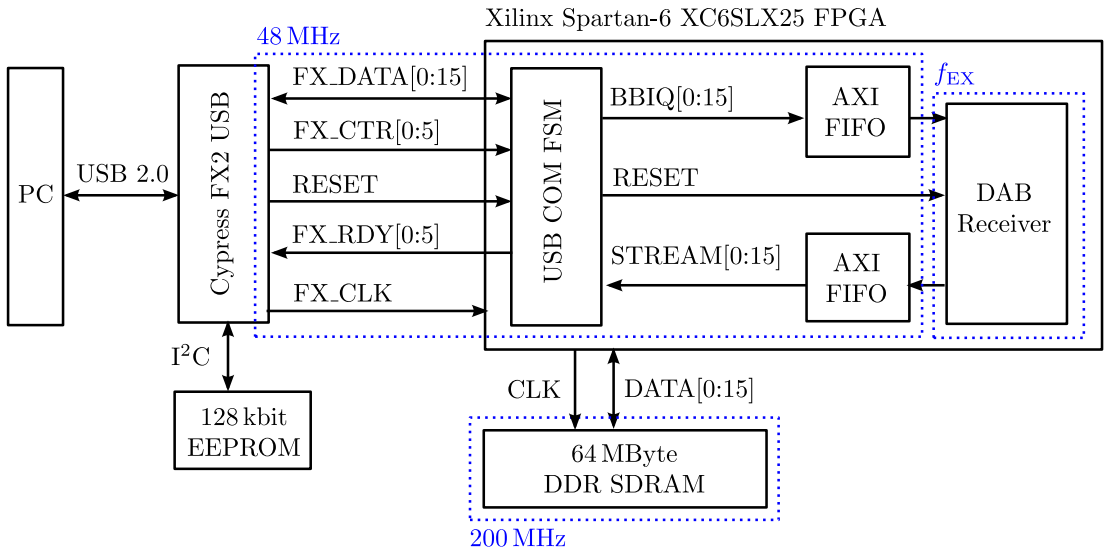


Figure 3.13: DAB receiver system architecture on ZTEX FPGA platform.

A schematic of the FPGA system environment is shown in Figure 3.13. Two advanced extensible interface bus (AXI) streaming FIFOs are used to decouple the DAB receiver I/O channels with the USB_COM_FSM module and handle the clock domain crossing. The USB_COM_FSM module controls the communication between the EZ-USB FX2 microcontroller and the AXI FIFOs by evaluating and translating the FX2 FIFO control signals FX_CTR and FX_RDY to corresponding ready and valid signals from and to the AXI FIFOs. Back-pressure is used in the receiver chain to control the data-flow from the last PE to the first PE using AXI streaming FIFOs as shown in Figure 3.14. The ARM Advanced Microcontroller Bus Architecture (AMBA) specification defines that the AXI master supplies the AXI slave with data using the handshaking signals `tready` and `tvalid`. If both signals are asserted, the data transfer will be initiated. Using AXI FIFOs between all PEs allows an arbitrary partitioning of the receiver chain into sub-chains without changing the data-flow control among the PEs, which is useful for DPR module chain partitioning.

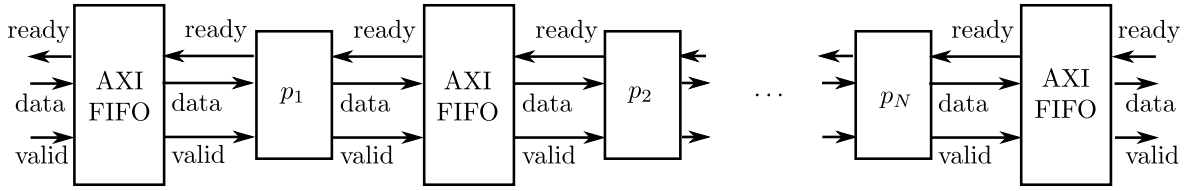


Figure 3.14: DAB receiver PE synchronization using AXI streaming FIFOs.

3.3.1 Resource Utilization and Dominating Processing Elements

The processing element with the longest processing duration in relation to the duration of a DAB frame determines the minimum clock frequency of the system in order to be real-time capable. For the receiver in question, the FFT requires most of the FPGA cycles in the chain as it has been designed for minimum resource usage. The FFT core has been generated using the Xilinx LogiCORE IP generator as documented in [Xil12a]. In Table 3.6 multiple FFT implementations are listed together with their latencies and resource requirements. The *Radix-2 Lite* implementation has been used in the Spartan-6 receiver and requires 26,600 cycles to perform an FFT operation for one OFDM symbol with 2048 bins.

Implementation	Latency[3]	Queue Time[4]	Slices	BRAMs	DSP48
Pipelined Streaming	4,223 cycles	2,048 cycles	994	12	18
Radix-4	5,267 cycles	7,313 cycles	752	15	18
Radix-2	13,434 cycles	15,582 cycles	359	11	6
Radix-2 Lite	24,612 cycles	26,660 cycles	298	7	4

Table 3.6: Xilinx implementation options for 2048-point FFT on Spartan-6 FPGA.

Given the FFT processing time of 26,660 cycles per OFDM symbol and a DAB frame duration of 96 ms, a minimum FPGA clock frequency of 21.1 MHz is required to demodulate all 76 OFDM symbols in real-time as stated in Equation 3.11. Here, it is important to mention that DAB does not require a continuous FFT operation to demodulate a single audio stream in the MSC as employed by Stolz et al. in [SFS12]. In case the FFT processing time can be reduced, the Viterbi decoder is likely to dominate the processing chain as subsequently laid out.

$$f_{\text{EX,MIN}} = \frac{76 \cdot 26,660\,\text{cycles}}{96\,\text{ms}} = 21.106\,\text{MHz} \tag{3.11}$$

The Viterbi decoder is the second most demanding block in the decoding chain and is used to decode the punctured convolutional code with constraint length 7. The convolutional code has a mother code rate of $r = 1/4$ using 3 different polynomials. Similar to the FFT, different implementations were analyzed for the Spartan-6 FPGA implementation using the Xilinx LogiCORE IP Viterbi Decoder (cf. [Xil12b]) with 4 bits per LLR and a traceback length of 84. In Table 3.7, two different implementation options are listed, i.e. parallel and serial.

[3]Number of clock cycles from first input sample to first output sample.
[4]Number of cycles until 2048-point FFT calculation is complete.

Implementation	Latency[5]	Queue Time[6]	Slices	BRAMs	DSP48
Parallel	363	1	2,076	2	0
Serial	354	14	594	2	0

Table 3.7: Xilinx implementation options for Viterbi decoder on Spartan-6 FPGA

The serial Viterbi decoder implementation has been used in the DAB receiver implementation as it requires fewer slices compared to the parallel implementation. However, the number of cycles per output bit is increased by a factor of 14. Given an audio broadcast with a worst-case data rate of 384 kbit/s, unequal error protection (UEP) encoding at a code rate of $r \approx 0.35$ in protection level 1 results in 26,624 LLR soft-bit quadruples which need to be processed within the duration of one CIF. Since there are 4 CIFs per DAB frame $4 \cdot 26,624$ LLR quadruples are received during a DAB frame. On top comes the FIC with a fixed code rate of $r = 1/3$ with 3072 depunctured LLR quadruples per DAB frame. For such a configuration, the minimum FPGA clock frequency can be calculated to be 16.24 MHz as shown by Equation 3.12.

$$f_{\text{EX,MIN}} = \frac{4 \cdot 14 \cdot (354 + 26,624) + 14 \cdot (354 + 3072)\,\text{cycles}}{96\,\text{ms}} = 16.237\,\text{MHz} \tag{3.12}$$

The resource utilization of the processing elements of the DAB receiver are outlined in Table 3.8 and the resource utilization of the receiver together with the USB communication interface is stated in Table 3.9 (cf. system design in Figure 3.13). Note that the USB communication logic and AXI FIFO buffers resemble the environment of the receiver and do not contain any baseband decoder functionality. Synthesis and implementation has been accomplished using Xilinx ISE 13.4.

As already discussed, the FFT and Viterbi implementations are the most demanding in terms of processing time and resources. The frequency estimation block comes third in complexity followed by the frequency synchronization and gain amplification stage which both operate at baseband rate. In the implementation of Gnadl (cf. [Gna12]) the time deinterleaver has been placed inside the FPGA BRAM memory occupying 29 BRAM units. To relax the memory requirements of the receiver system, the external SDRAM has been used to store the time deinterleaver values, reducing the BRAM consumption from 29 to 2 in this configuration. Another difference to the work of Gnadl is that in the present implementation only 2 BRAMs instead of 6 BRAMs have been used for the USB COM FSM, which has shown to be sufficient for real-time operation of the receiver. It needs to be remarked that the additional SDRAM memory interface resources for the external time deinterleaver memory have not been traced in the receiver design phase and are thus not listed as a part of the system environment in Table 3.9. However, the resources of the system environment are not of concern in this section but will be addressed in the next section, where the static receiver system resources will be compared against the cyclic DPR implementation.

The output throughput of the PEs and the relative resource consumption are also presented in Table 3.8. From the input to the output of the chain, the throughput is

[5]Number of input LLRs the decoder will consume to produce the first output bit.
[6]Number of cycles required for the computation of one output bit given an input LLR quadruple.

n	p_n	Slices	FFs	LUTs	BRAMs	DSP48	$\frac{\gamma_n}{\text{Mbit/s}}$
1	Freq. Correct	127	306	344	0	4	32.768
2	AGC	99	195	148	0	3	32.768
3	Time Sync.	64	86	85	0	0	32.325
4	Freq. Est.	291	581	811	2	5	32.325
5	Guard Remove	7	10	9	0	0	25.941
6	FFT	358	1024	625	7	8	25.941
7	DQPSK	37	76	49	2	6	25.941
8	Freq. Deint.	49	29	50	3	0	9.1
9	Stream Cut.	38	31	46	0	0	4.533
10	Time Deint.	52	63	71	3	0	4.533
11	Viterbi	713	1946	1415	3	0	0.416
12	Post Proc.	14	17	4	0	0	0.416

Table 3.8: Resource utilization and data throughput of DAB receiver PEs.

Description	Slices	FFs	LUTs	BRAMs	DSP48
DAB Receiver	1849	4364	3657	20	26
USB COM FSM	175	416	297	2	0
AXI FIFOs	34	60	70	2	0
Total	2058	4840	4024	24	26
XC6SLX25	3758	30064	15032	52	38

Table 3.9: DAB receiver resource utilization on ZTEX module 1.11c.

continuously decreasing. For the output throughput of the last PE an audio broadcast with a worst-case rate of 384 kbit/s has been assumed. Together with an output throughput of 32 kbit for the FIC, the maximum throughput at the Viterbi decoder output results in 416 kbit/s.

Reducing the resource utilization by partitioning the DAB receiver into DPR modules and sequentially executing these modules using cyclic DPR will be subsequently depicted.

3.3.2 Framing and Context Lifespans

The color of the nodes in the receiver flow-graph in Figure 3.15 reflect the context lifespan of each processing element. The graph visualizes the framing period after which the context of an element reaches a pre-defined state. For example, the context of an orange element reaches a pre-defined state after one CIF has been processed. Recall that context lifespans are important when designing a cyclic DPR system since during one DPR module execution cycle it is beneficial to store as little context information as possible to minimize the cycle time (cf. Equation 3.1). For DAB receiver elements with infinite framing, the context information must always be saved and restored. It is important to keep in mind that the framing is implied by the constraints of the broadcasting standard,

whereas the context lifespan is defined by the actual implementation. The context of

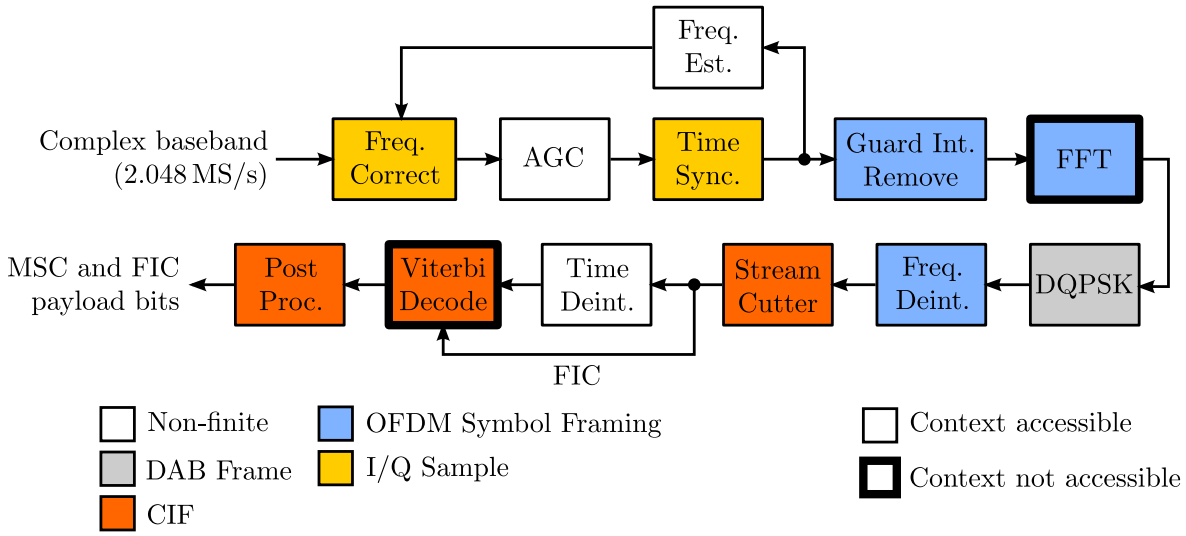


Figure 3.15: Annotated DAB chain graph with different context lifespans.

a module is sometimes not accessible as there are no signals provided to the memory elements of the module, which typically is the case for IP cores. As shown in Figure 3.15, the Viterbi decoder IP and the FFT IP blocks hide their internal state such that it can not be accessed for context storage and recovery. Apart from these two blocks, the elements in the DAB chain provide access to context information. It is important to bear in mind that elements with a non-finite framing and an inaccessible context can not easily be used inside a reconfigurable architecture. In this case only an ICAP read-back enables to realize a cyclic DPR design, cf. [JTHT10] and [LD09], which has not been considered in this work. Regarding context information the convolutional time-deinterleaver is a special case since its execution basically means modifying its context. Hence, no explicit context loading and saving are required. Given the context lifespans and accessibility information in the annotated graph, a suitable framing unit for cyclic DPR needs to be defined. Special attention must be paid to PEs with inaccessible context as they are likely to determine the minimum frame duration. For example, for the Viterbi decoder and the demodulator at least one CIF needs to be processed to assure that the PE context settles in a known state. As the FFT PE context is related to an OFDM symbol-wise framing, which is a subset of one CIF, the Viterbi decoder determines the framing of the DPR design by having the longest frame duration. Following this idea, either a DAB frame-based or a CIF frame-based execution cycle can be used, i.e.

- DAB frame-based execution with $T_{FRAME} = T_{DAB}$, and
- CIF-based execution with $T_{FRAME} = T_{CIF}$.

Using CIFs instead of DAB frames enforces to store and to recover the context of the DQPSK demapper, as the context is valid for the duration of one DAB frame. For DAB frame-based execution, the context of the AGC and frequency estimator may be neglected, as the memory used for averaging by an exponential window can be reset at the beginning of a DAB frame period. Since the time deinterleaver state is stored in the external memory, the whole chain can be executed without context saving and recovery

when a DAB frame-based processing is employed. Since context storage and recovery makes the system design more complicated, DAB frame-based processing has been used for the cyclic DPR prototype. Having described the possible durations of a cyclic DPR input frame, in the next section the partitioning of the receiver processing elements into DPR modules will be presented.

3.3.3 Receiver Partitioning

Partitioning the DAB receiver requires the data-flow to be controlled by the PEs themselves rather than having an additional data-flow control unit, which is accomplished by the AXI FIFO design as previously outlined. Partitioning criteria and metrics have to be defined in order to find suitable DPR modules for the DAB receiver. Both are accurately laid out in detail in Chapter 4 together with a partitioning proposition for the DAB receiver in question and will therefore not be discussed further in this section. In short, the approach for finding suitable DPR modules for cyclic DPR is based on finding the candidates where the resulting DPR modules require approximately the same amount of resources, i.e. have minimum variance in terms of resources. In addition, minimizing the DPR module output throughput is of concern for finding suitable partitioning candidates. Applying these constraints, the DAB receiver has been partitioned into three DPR modules as shown in Figure 3.16 with the accumulated resource utilization and throughput as provided in Table 3.10.

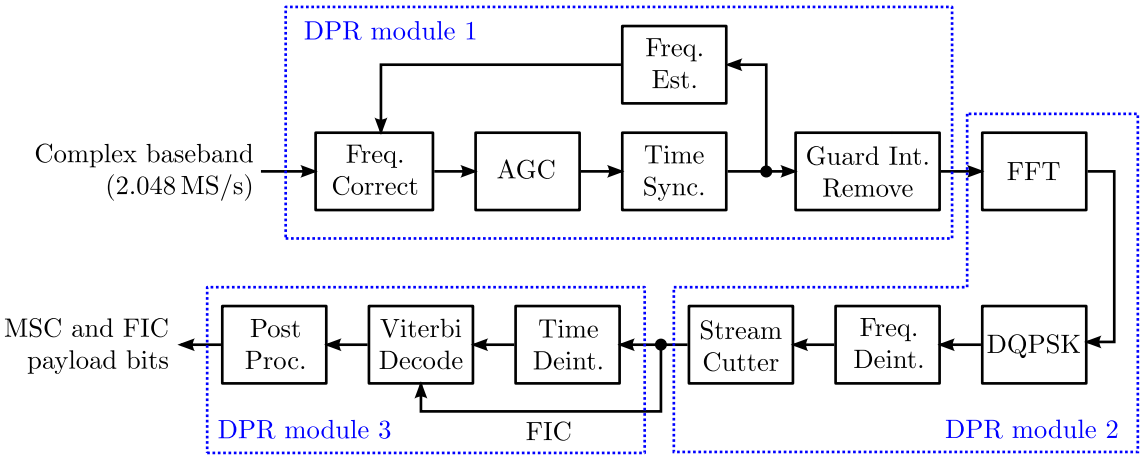


Figure 3.16: DAB chain partitioning into three DPR module.

The input data rate of the first DPR module is determined by the resolution and sample rate of the complex baseband signal fed to receiver using 16 bit per complex value at a rate of 2.048 MS/s. The output rate of DPR module 1 (i.e. the input rate of DPR module 2) is related to the 2048 samples of an OFDM symbol quantized with 16 bit per complex value and transmitted within a period of approximately 1.246 ms. For continuous operation, this gives a throughput of approximately 26.3 Mbit/s. Considering the length of the null symbol in relation to the DAB frame length, the average throughput at the output of module 1 reduces down to 25.941 Mbit/s. The LLR values at the output of module 2 are fed to the time-deinterleaver with 4 bits per soft-bit. In the worst-case scenario, the MSC interleaver sequence comprises of 26624 LLRs per CIF. Given the soft-bit resolution of 4 bits and 4 CIFs per DAB frame, the maximum MSC LLR throughput

from DPR module 2 to DPR module 3 results in

$$\frac{26624 \cdot 4 \cdot 4}{96\,\text{ms}} = 4.4373\,\text{Mbit/s}.$$

Together with the FIC throughput of 96 kbit/s, the maximum output throughput of module 2 turns out to be 4.533 Mbit/s. Since this configuration represents the highest throughput scenario, the following analysis is based on decoding a single DAB service using this worst-case configuration.

m	PEs	Slices	FFs	LUTs	BRAMs	DSP48	$\frac{\gamma_{\text{IN},m}}{\text{Mbit/s}}$	$\frac{\gamma_{\text{OUT},m}}{\text{Mbit/s}}$
1	p_1, p_2, p_3, p_4, p_5	588	1178	1397	2	12	32.768	25.941
2	p_6, p_7, p_8, p_9	482	1160	770	12	14	25.941	4.533
3	p_{10}, p_{11}, p_{12}	779	2026	1490	6	0	4.533	0.416

Table 3.10: DPR module resource utilization and I/O throughput.

When comparing the resources of the DPR modules to the total system resources outlined in Table 3.8, a sequential execution of DPR modules seems promising for a resource-economic implementation. However, for the operation of the cyclic DPR system additional control logic in the static region of the FPGA is required. Before the hardware requirements of the static and reconfigurable regions are elaborated, in the next section the real-time constraints of the system will be determined given the DPR modules as defined in Table 3.10 and the data framing as outlined in Section 3.3.2.

3.3.4 Memory Throughput and Execution Time

Recalling the cyclic DPR system model, the execution time of a DPR module is either upper bounded by the memory interface throughput or by the number of execution cycles per frame in relation to the clock frequency of the DPR module (cf. Equation 3.9 and 3.8). In order to determine the correct upper bound, it is necessary to derive the external memory throughput of the system. The theoretical maximum memory transfer rate can be obtained by the interface clock rate, bit width and access methodology. On the ZTEX hardware platform the 64 Mbyte DDR memory is interfaced using a bi-directional 16 bit data bus at a clock rate of 200 MHz, which gives a theoretical maximum throughput of

$$\gamma_{\text{MEM}}^{\text{MAX}} = 200\,\text{MHz} \cdot 2 \cdot 16\,\text{bit} = 6.4\,\text{Gbit/s},$$

Access latency of SDRAM or inefficient memory controller implementations might reduce the maximum throughput, such that the theoretical maximum can not be achieved in practical systems. Therefore, for the subsequent analysis a conservative throughput of 50 % of the maximum rate will be assumed. Given this assumption, the maximum memory throughput available in the cyclic DPR system becomes

$$\gamma_{\text{MEM}} = \frac{\gamma_{\text{MEM}}^{\text{MAX}}}{2} = 3.2\,\text{Gbit/s},$$

As stated in Equation 3.6, the I/O throughput of the DPR chain has to be deducted from this value, such that the remaining available throughput for other memory accesses

can be formulated to be

$$\Gamma_{\mathrm{MEM}} = 3.2\,\mathrm{Gbit/s} - (\gamma_{\mathrm{IN},1} + \gamma_{\mathrm{OUT},3}) \approx 3.167\,\mathrm{Gbit/s}, \tag{3.13}$$

where $\gamma_{\mathrm{IN},1} = 32.768\,\mathrm{Mbit/s}$ and $\gamma_{\mathrm{OUT},3} = 0.416$ Mbit/s, as stated in Table 3.10. Having derived the available memory throughput of the ZTEX system, the correct upper bound on the execution times for reconfiguration and module execution will be determined next.

Memory Throughput during Reconfiguration

Before deriving the DPR system parameters, it is important to mention that Xilinx does not officially support partial reconfiguration with the Spartan-6 FPGA series. However, several authors report that difference-based DPR on the Spartan-6 FPGA is feasible (cf. [KBT10] and [BYT11]). The Spartan-6 FPGA provides an ICAP with a data width of $W_{\mathrm{B,ICAP}} = 16\,\mathrm{bits}$. As mentioned by Xilinx in [Xil11d], the ICAP clock signal should be limited to a maximum frequency of $f_{\mathrm{ICAP}} = 20\,\mathrm{MHz}$, although ICAP clock rates of $f_{\mathrm{ICAP}} = 100\,\mathrm{MHz}$ have also been successfully tested in [MNH+11b].

According to Equation 3.7 of the cyclic DPR model, the ICAP throughput for 100 MHz clock rate can be derived as

$$\Gamma_{\mathrm{ICAP}} = 100\,\mathrm{MHz} \cdot 16\,\mathrm{bit} = 1.6\,\mathrm{Gbit/s},$$

where for 20 MHz the maximum data rate becomes

$$\Gamma_{\mathrm{ICAP}} = 20\,\mathrm{MHz} \cdot 16\,\mathrm{bit} = 320\,\mathrm{Mbit/s}.$$

For the subsequent analysis both values will be taken into account. Since the available memory throughput is much higher than the required throughput of 1.6 Gbit/s at 100 MHz ICAP clock rate, i.e. $\Gamma_{\mathrm{MEM}} \gg \Gamma_{\mathrm{ICAP}}$, the reconfiguration time is dominated by the throughput of the ICAP interface and not by the throughput of the memory interface. Looking at the memory access pattern in Figure 3.9 underlines this observation as no other task is accessing the external memory during reconfiguration. Given the ICAP throughput, the configuration time can be deducted from the size of the partial bitstream to be written to the ICAP during reconfiguration. The uncompressed bitstream of the ZTEX Spartan-6SLX25 comprises of 6,440,432 bits (cf. [Xil15c]). Given the resources of the DPR modules as outlined in Table 3.10 and taking into account the Spartan-6 FPGA resources listed in Table 3.8, the differential bitstream is assumed to have an average size of 1/3rd of the FPGA bitstream size[7], which equals $N_{\mathrm{B,BIT},m} = 2,146,810\,\mathrm{bits}$. According to Equation 3.8 the reconfiguration time for an ICAP clock frequency of 100 MHz can be calculated as

$$T_{\mathrm{DPR},m} = \frac{2,146,810\,\mathrm{bit}}{1.6\,\mathrm{Gbit/s}} \approx 1.34\,\mathrm{ms},$$

for all m, i.e. for all DPR modules, and for 20 MHz ICAP clock frequency the reconfiguration time resembles to

$$T_{\mathrm{DPR},m} = \frac{2,146,810\,\mathrm{bit}}{320\,\mathrm{Mbit/s}} \approx 6.71\,\mathrm{ms}.$$

[7] An evaluation with DPR modules for the DAB receiver revealed an average differential-bitstream size of approximately 983,040 bits (cf. [Ili12]). More conservative values have been chosen for the evaluation in this work.

The configuration time of the DPR partition will subsequently be used to determine the execution delay and the real-time capability of the system.

Memory Throughput during DPR Module Execution

In order to determine the execution time and the effective module throughput of the DPR modules, the number of execution cycles have to be evaluated. The number of cycles for the initialization and execution of a DPR module for DAB frame-based and CIF-based execution are provided in Table 3.11. In the receiver implementation, no context loading and saving has been employed, which is why $N_{\mathrm{C,SV},m} = 0$ for all m. The values for $N_{\mathrm{C,LD},m}$ in the table refer to the number of cycles for the initialization of the m-th module. Due to the cyclic nature of the DPR module context, in case of DAB frame-based processing ignoring the context is not an issue as elaborated in Section 3.3.2. In contrast, a CIF-based receiver can not be realized without context saving and loading. Since the amount of context information required to store and load is negligible, it is still possible to compare the values derived for the CIF-based receiver to the DAB frame-based receiver within the scope of a feasibility study.

DPR module	$T_{\mathrm{FRAME}} = T_{\mathrm{DAB}}$			$T_{\mathrm{FRAME}} = T_{\mathrm{CIF}}$		
m	$N_{\mathrm{C,LD},m}$	$N_{\mathrm{C,SV},m}$	$N_{\mathrm{C,EX},m}$	$N_{\mathrm{C,LD},m}$	$N_{\mathrm{C,SV},m}$	$N_{\mathrm{C,EX},m}$
1	15	0	155,648	15	0	38,912
2	11	0	2,229,900	11	0	557,475
3	5	0	1,562,101	5	0	390,532

Table 3.11: Number of cycles for the initialization and execution of the DPR modules.

Inspecting the numbers in Table 3.11 reveals that the second DPR module requires most of the processing cycles. The reason for the dominant cycle count is that the second module includes the FFT block with the FFT queue time dominating the system as stated in Section 3.3.1. The third module requires about 70 % of the cycles of the second module, where the Viterbi implementation is the dominant processing element. In terms of execution cycles, the first module is able to process the data more than 14 times faster than the second module. Moreover, as shown by the resource listing in Table 3.10, the input and output throughput of the first module is the highest among all modules.

Therefore, in relation to the memory access pattern in the system model (cf. Figure 3.9), the effective module throughput of the first DPR module determines the peak memory throughput during one module execution cycle. Note that the effective module throughput is related to the module execution time, which in turn is determined by the FPGA execution frequency. Given the number of execution cycles as outlined in Table 3.11, an DPR module execution frequency of f_{EX} and the DPR module input and output throughput values in Table 3.10, the effective DPR module throughput can be calculated according to Equation 3.2. Recall that the peak memory throughput during the execution period of a DPR module is defined by the accumulated effective input and output throughput rates of the DPR module as outlined in Equation 3.9. Figure 3.17 shows the peak memory input and output throughput during the execution of the DPR modules for different FPGA module execution clock rates. In addition, the previously

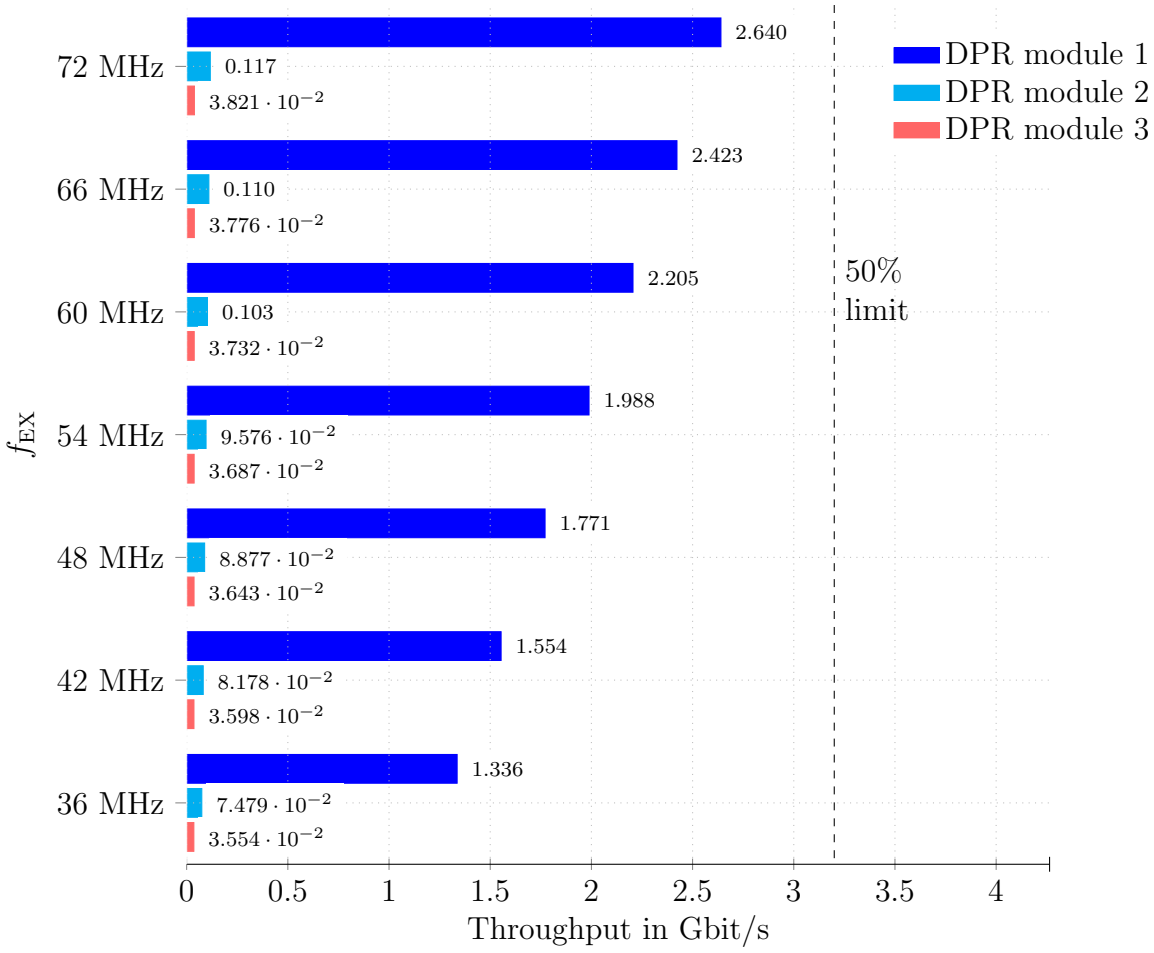


Figure 3.17: Peak memory throughput during DPR module execution.

introduced limit of 50 % of the maximum theoretical throughput is shown by a dashed vertical line. As already pointed out, the first DPR module exhibits the highest memory throughput during execution as it can process data much faster than the other modules. It can also be observed that even for an execution frequency of $f_{\mathrm{EX}} = 72\,\mathrm{MHz}$ the memory interface can cope with the throughput emerging during the execution of DPR module 1. It can be concluded from the graph that the execution time of a DPR module is not limited by the memory interface but determined by the FPGA clock frequency and the number of execution cycles per DPR module (cf. left term in Equation 3.9).

In conclusion, the memory interface is fast enough to cope with the transfer rates during reconfiguration and during the DPR module execution period. In the next section, the execution times of the cyclic DPR system will be determined.

3.3.5 Real-Time Constraints and Latency

The cyclic DPR system model presented in Section 3.2 applies timing restrictions to the sequential execution of the DPR module chain. For real-time operation it is important that the processing of the current frame has been finished before the arrival of a new frame. As stated in the system model, the processing duration is referred to as cycle time and it can be calculated with the knowledge of the execution times according to Equation 3.1. Recall that it is important that $\mathrm{T}_{\mathrm{FRAME}} \geqslant T_{\mathrm{CYC}}$ is satisfied for real-time operation to be possible. Considering this constraint, for the DAB receiver chain the minimum clock frequency for real-time operation can be determined by

$$f_{\mathrm{EX,MIN}} = \frac{1}{\mathrm{T}_{\mathrm{FRAME}} - 3 \cdot T_{\mathrm{DPR},m}} \sum_{1}^{3} N_{\mathrm{C,LD},m} + N_{\mathrm{C,EX},m}. \tag{3.14}$$

For DAB frame-based processing with $\mathrm{T}_{\mathrm{FRAME}} = 96\,\mathrm{ms}$ and 20 MHz ICAP clock frequency, using Equation 3.14 gives a minimum module execution clock frequency of $f_{\mathrm{EX}} = 52\,\mathrm{MHz}$ for real-time processing. Feeding the ICAP with a 100 MHz clock leads to a minimum DPR module execution clock frequency of $f_{\mathrm{EX}} = 43\,\mathrm{MHz}$. For DAB frame-based execution, the module execution timing diagram is presented in Figure 3.18 for various module execution clock frequencies and an ICAP clock frequency of 20 MHz. Since the number of load cycles $N_{\mathrm{C,LD},m}$ are negligible in comparison to the number of execution cycles, both values are combined in the graph. The reconfiguration time is indicated in yellow, the execution and initialization times of the modules in dark blue (1), light blue (2) and red (3) and the idle time is shown in gray. The frame duration is indicated by a vertical dashed line. If the DPR cycle time crosses this line, the real-time constraint is violated and module clock frequency or ICAP throughput are not feasible. The upper plot shows the DPR processing cycle timing for a frame duration of one DAB frame and the lower plot for two DAB frames. When comparing the system with a single DAB frame to the dual DAB frame system shown in Figure 3.19, the minimum execution frequency decreases by 11.7 % since the reconfiguration time is lower in proportion to the cycle time. Comparing the minimum clock frequency of the non-DPR system stated in Equation 3.11 to the DPR implementation, the execution frequency is increased by more than a factor of two, and albeit fewer resources being active at a time when using cyclic DPR system, the increase in clock frequency is a major drawback since it leads to a linearly proportional increase in dynamic power consumption of the

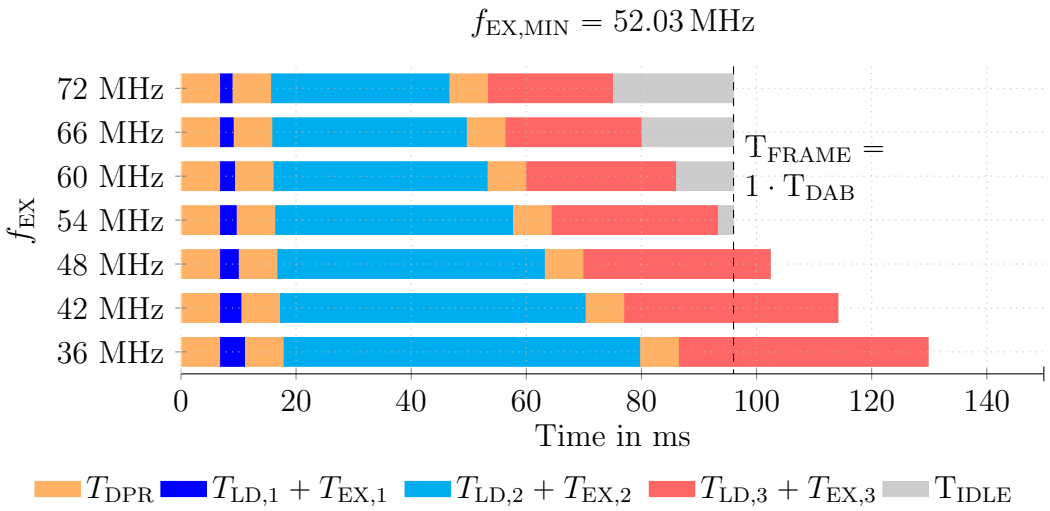


Figure 3.18: Cycle time for DAB frame-based execution with 20 MHz ICAP.

FPGA[8] (cf. [Xil09a]). In addition to the increased dynamic power consumption, the steady communication with external memory also leads to more power being consumed by the cyclic DPR system.

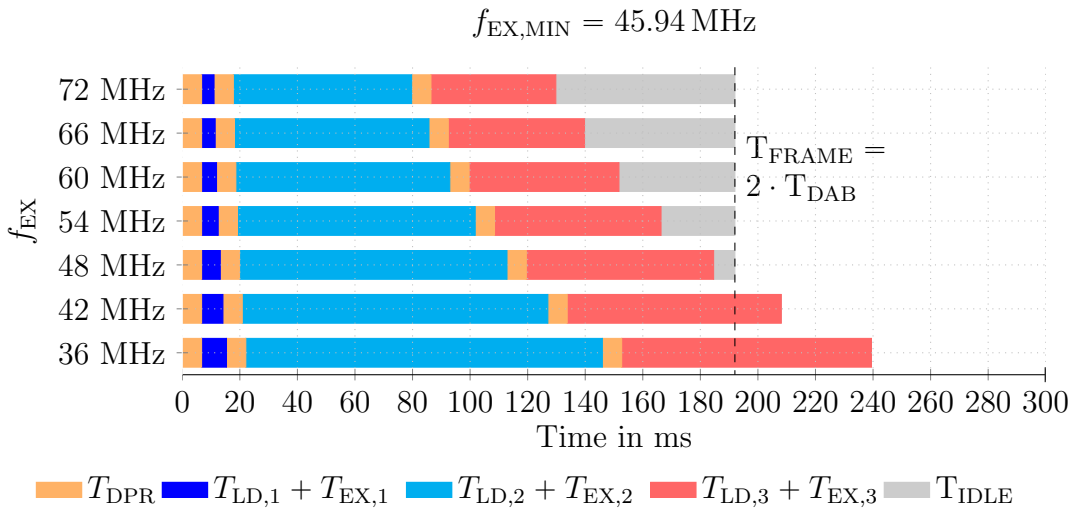
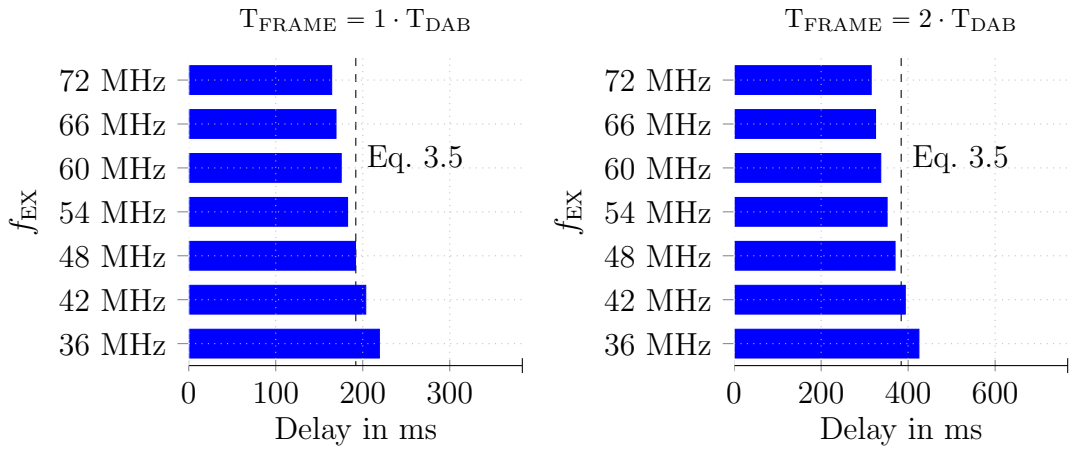


Figure 3.19: Cycle time using a duration of two DAB frames with 20 MHz ICAP.

As expected, it can be observed that for $f_{EX} > f_{EX,MIN}$ the system exhibits an idle time. Although idle times indicate that the execution clock frequency is higher than necessary, they lead to a reduction in DPR system latency, as derived in the system model (cf. Equation 3.4). Figure 3.20 shows the additional system latency due to cyclic DPR operation for an ICAP clock frequency of 20 MHz and DAB frame-based processing. The dashed line in the Figure shows the upper bound on the delay for real-time systems as presented in Equation 3.5. For values crossing the delay bound, the execution frequency is not sufficient for real-time operation. From the figure can be deducted that systems

[8] Note that the DPR operation itself leads to an increase in power consumption, albeit the logic elements inside the DPR partition being idle during DPR as evaluated in [Cla11].

with idle times exhibit a lower DPR system latency compared to systems without idle time. For the DAB receiver system, the additional delay is around 192 ms if one DAB frame is processed and twice as much if two DAB frames are processed within one DPR cycle. The worst-case end-to-end playback delay of the DAB system due to the convolutional time-interleaving is 384 ms as stated in [ets06]. Using cyclic DPR on a DAB frame basis, the overall system delay increases to 576 ms. The additional DPR delay can be minimized by using smaller frame sizes within a DPR processing cycle as discussed further on.



(a) Frame duration T_{DAB}

(b) Frame duration $2 \cdot T_{DAB}$

Figure 3.20: Latency for DAB frame-based execution with 20 MHz ICAP.

Although CIF-based processing is not realizable in a DPR system without context handling, the behavior can be an approximated by the cycle times calculated in Table 3.11. Using the cycle times to determine the minimum execution frequency according to Equation 3.14 leads to a system where $f_{EX,MIN} = 254.79$ MHz for $f_{ICAP} = 20$ MHz. The significant increase in execution clock frequency is related to the fact that an ICAP clock frequency of 20 MHz with CIF-based execution does not leave a lot of margin for execution given that three reconfiguration cycles require 20.13 ms of a 24 ms CIF duration. In this case the reconfiguration interface throughput has a stronger influence on the real-time capability of the system and the problem can be mitigated by increasing the ICAP frequency to 100 MHz. The cycle time of the DPR system for CIF-based processing with an ICAP frequency of 100 MHz is depicted in Figure 3.21. In this case, the minimum execution frequency for real-time operation is 49.41 MHz which is roughly 1/5th of the execution frequency when using an ICAP clock of 20 MHz.

The worst-case DPR system delay of the CIF-based system is 48 ms which is 1/4th of the delay of the DAB frame-based system. Hence, for CIF-based processing, the end-to-end DAB system delay can be calculated to be 432 ms, resulting in a reduction of 25 % over the DAB frame-based design.

Discussion

A scheme for the sequential execution of DPR modules has been presented together with equations to determine the real-time capability and cyclic DPR system delay. Deriving

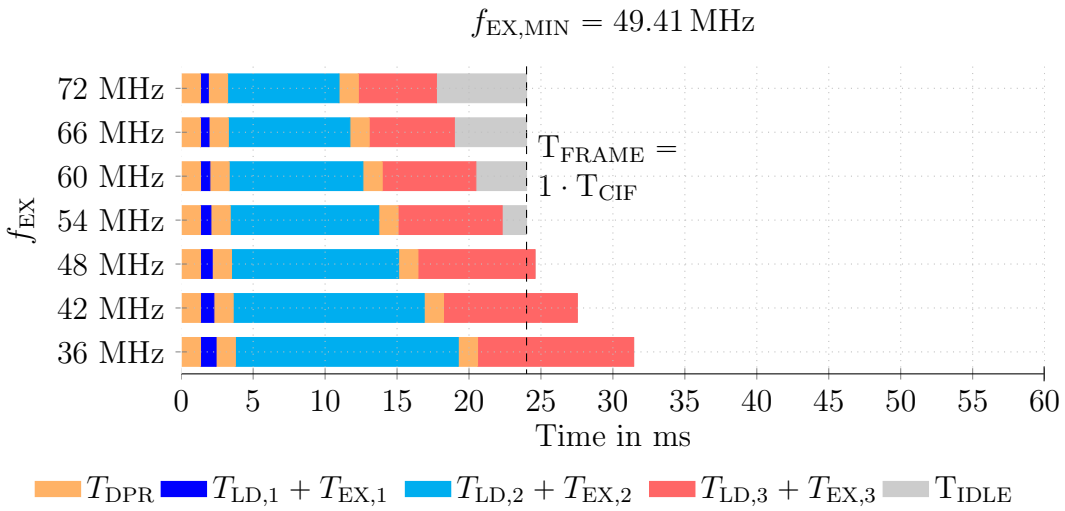


Figure 3.21: Cycle time for CIF-based execution with 100 MHz ICAP.

DPR modules from an existing chain of processing elements requires the clock frequency of the cyclic DPR design to be higher than the clock frequency of the non-DPR design for the system to be real-time capable. If increasing the clock frequency is not possible due to latency or timing violations in the design, the PEs must be redesigned to accomplish the task in fewer cycles, which in turn might increase the resource utilization (cf. FFT implementation in Table 3.6). If the PE requires fewer cycles for the task to finish, the throughput of the memory interface might saturate the execution time (cf. limit in Figure 3.17). In cases where increasing the clock frequency is infeasible, either the reconfiguration time must be reduced or the duration of the input frame must be increased (cf. Equation 3.14). A side-effect of increasing the input frame duration is a proportional increase of the DPR system delay according to Equation 3.5. Reducing the frame duration might increase the requirements for context handling, making the implementation more complex and requiring more resources and memory bandwidth. For a real-time feasibility evaluation early in the design phase it might be sufficient to look at dominating blocks like the FFT and Viterbi cores for the DAB receiver (cf. Section 3.3.1).

Given the system model for round-robin execution as derived in Section 3.2, the design of a hardware-based reconfiguration engine will be outlined, intended to serve as a generic platform for cyclic DPR systems [Ili12]. The implementation has been used to test the feasibility of the DAB receiver prototype and will be described in the subsequent sections.

3.4 Cyclic DPR for DAB Receivers - Part II: Hardware Implementation

A hardware-based DPR framework for the DAB receiver has been developed and a Spartan-6 FPGA has been chosen as implementation target since it promised to be the lowest-cost Xilinx FPGA platform with partial self-reconfiguration capability. Although partial reconfiguration is not officially supported by Xilinx it has been reported to be possible by Koch, Bayar and Meyer et al. up to an ICAP frequency of 100 MHz (cf.

[KBT10], [BYT11] and [MNH+11b]). The DAB receiver DPR modules have been implemented on the ZTEX Spartan-6 FPGA as described in the last section. For DPR to be applicable, the system environment of the DAB receiver needed to be enhanced as subsequently explained.

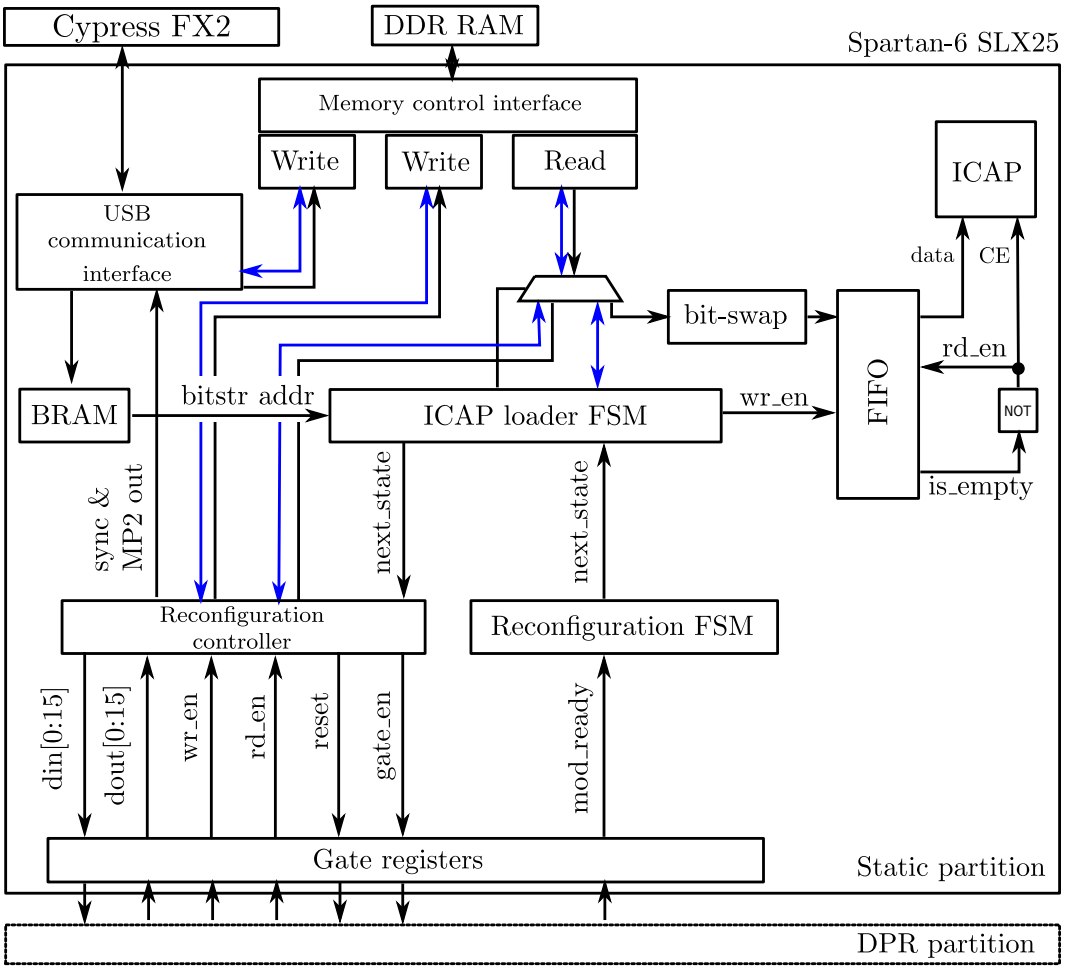


Figure 3.22: Cyclic DPR system and DAB receiver signal and control flow.

3.4.1 Static Environment of DPR System

The static logic resources required for the operation of the DAB receiver will in the following be referred to as *static environment*. In a DPR system, the static environment resides in the static partition. The major components of the static environment of the cyclic DPR system are presented in Figure 3.22. The Cypress FX2 microcontroller and the DDR memory are interfaced to the Spartan-6 SLX25 FPGA via external I/O pins as discussed in Section 3.3 and the cyclic reconfiguration tasks are handled without additional microprocessors. Baseband and bitstream information are transferred across the USB 2.0 port to the DDR memory using a configurable memory controller block (MCB) provided by Xilinx (cf. [Xil10]). Prior to buffering the bitstream in memory, the bit- and byte-order of the data are swapped in order to be compatible to the native ICAP format

(cf. [Xil15c]). The MCB provides up to 6 different 32 bit memory ports of which two can be used for bi-directional read/write operations and the entity has been interfaced to the system using a Xilinx memory interface generator (MIG) IP core. The MIG has been configured to provide three uni-directional 32 bit ports, one for reading and two for writing. FIFO buffers are used inside the MIG core to optimize read and write access to and from memory and to decouple the memory clock frequency of 200 MHz and the FPGA clock frequency of 48 MHz. Control signals to the memory interface are indicated by blue lines in the graph. According to Figure 3.9 in the system model, memory accesses between the ICAP bitstream loader state-machine and the DPR partition read operations are non-overlapping in time. Therefore, the memory read port is multiplexed between the ICAP loader FSM and the reconfiguration controller. In contrast, transfers from the reconfigurable module to memory overlap with the continuous transfer of baseband data into memory. Therefore, the USB communication interface and the reconfiguration controller utilize independent 32 bit write interfaces. The ICAP loader encapsulates the logic required to read the bitstream data from memory and write it into the ICAP interface, thereby changing the logic configuration inside the DPR partition. A dual-port FIFO is used to decouple the memory transfer cycles from the ICAP write cycles and as soon as the FIFO receives new data, it will be written into the ICAP with a clock frequency of 100 MHz. Real-time operation is feasible with this setup as the FPGA clock frequency exceeds the minimum FPGA clock frequency of 43 MHz as outlined in Section 3.3.5.

Subsequently, the functionality of the different blocks inside the static environment will be explained.

- **USB comm. interface**: The USB communication controller handles the synchronous data transfer from and to the USB host using the Cypress FX2 microcontroller. It is responsible for writing the bitstream data to external memory and the bitstream addresses to an internal BRAM. After the bitstreams have been stored, the communication controller enters a streaming mode in which the baseband data is written to memory and where the MP2 audio payload data is transferred back to the host.

- **Reconfiguration Controller**: The reconfiguration controller evaluates the state signal provided by the reconfiguration FSM through the ICAP loader FSM. The output signal of the FSM reflects the index of the reconfigurable module of the DPR chain. This module index is evaluated by the ICAP loader FSM, which is explained in the next section.

- **Reconfiguration FSM**: In case the active DPR module completed processing the input frame, the reconfiguration FSM signals the index of the next bitstream by counting the rising edges of the mod_ready signal.

- **ICAP loader FSM**: The bitstream address information is evaluated by the ICAP loader FSM, which evaluates the next_state signal from the reconfiguration FSM and presents the address of the pending DPR module to the memory controller. The bitstream information then passes through the bit-swap block and a rate-transition FIFO before it is written to the ICAP data port.

- **Gate registers**: During reconfiguration the signals from the reconfigurable partition to the static partition might randomly change during the reconfiguration process. Therefore, the static and dynamic partition are separated by registers to avoid signal glitches.
- **Memory control interface**: The MCB of the Spartan-6 FPGA is interfaced using the MIG IP core, which is a wrapper providing read and write interfaces to the external DDR memory. The memory control interface provides two dedicated ports to the reconfiguration controller for baseband streaming and data writeback via the USB communication interface.

The following signals have been used for communication between the static and dynamic partitions:

- **Data input (din):** Complex baseband signal transferred from memory and supplied to the DPR partition.
- **Data output (dout):** Decoded MP2 payload data streamed to the USB communication interface from the active DPR module.
- **Write enable (wr_en):** Enable signal from the static partition to the DPR partition indicating that the current sample at the data input port is valid.
- **Read enable (rd_en):** Enable signal from the DPR partition to the static partition indicating that the current sample at the data output port is valid.
- **Enable (gate_en):** Enables or disables the gate registers to decouple the static partition from the DPR partition.
- **Reset (reset):** Reset the reconfigurable module to the initial state.
- **Ready (mod_ready):** If the processing task of a DPR module has been completed, the ready signal is asserted for one clock cycle. The reconfiguration FSM evaluates this signal and decides if it is necessary to switch to the next configuration. The mod_rdy signal is raised after a FIC frame has been processed and after an MSC frame has been processed.

A detailed explanation of the system architecture can be found in [Ili12] and will be omitted in this work. In the following, the bitstream generation flow will be outlined together with the observations made during Spartan-6 dynamic partial reconfiguration.

3.4.2 DPR Simulation and Bitstream Generation Flow

Prior to implementing the DPR system, the DAB receiver system outlined in Figure 3.22 and the functionality of the DPR modules have been verified in a simulation testbed. A screenshot of the components of the simulation testbed is provided in Figure 3.23, where the DPR modules 1, 2 and 3 are denoted by block_a, block_b and block_c. The testbed includes all three DPR modules and the static logic inside a single static system configuration. The DPR modules are virtually loaded and unloaded by multiplexing their I/O ports. In comparison to Figure 3.22, the CE signal of the ICAP has been

tied to constant low and an additional FSM resets the dual-port FIFO every 4 clock cycles, thus simulating the ICAP consuming the bitstream data. After successful testing and debugging, the DPR system has been prepared for the use in a real reconfiguration environment as subsequently explained.

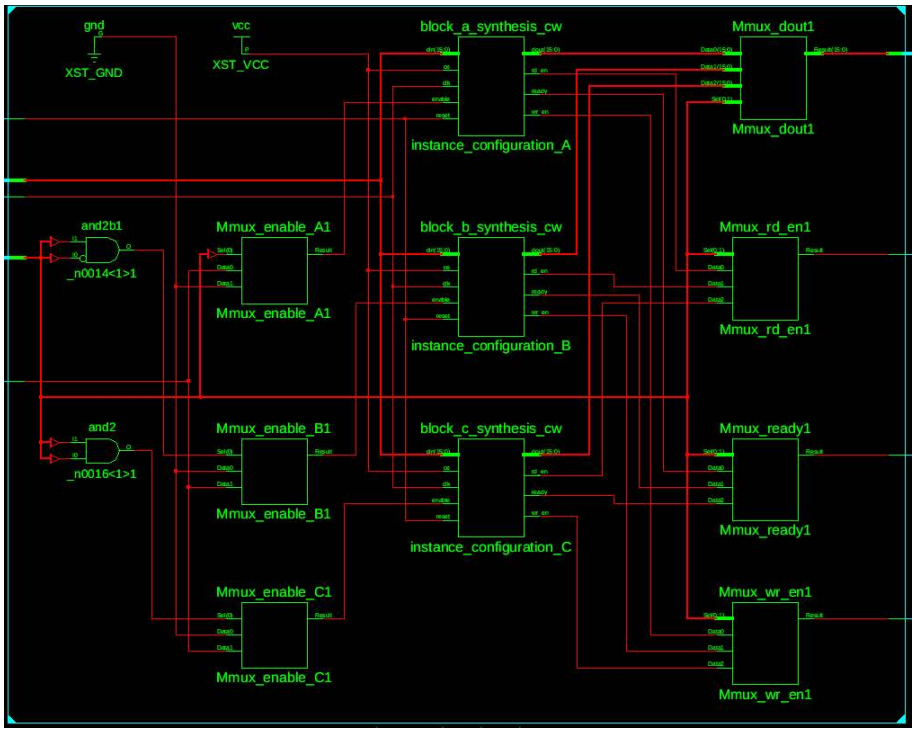


Figure 3.23: Screenshot of the RTL simulation model for DAB receiver modules.

The configuration bitstreams have been generated using the differential generation tool-flow as described by Meyer et al. in [MNH+11b] and by Schmidbauer in [Sch11]. Given the mentioned tool-flow, a functional prototype DPR system with five small DPR modules has been realized as documented in [Sch11]. This test system has been developed independently of the DAB receiver and solely for DPR evaluation purposes of Spartan-6 FPGAs. In the test system, the size of the DPR modules was small in comparison to the DPR partition size as shown in Figure 3.24. In the evaluation DPR system, the ICAP was driven with a 12 MHz clock frequency and the DPR modules have shown to be fully functional, indicating a successful reconfiguration process. Upon successful evaluation of the Spartan-6 DPR test system, the same bitstream generation tool-flow has been used for the cyclic DPR DAB receiver system.

For the DAB receiver the exported VHDL code, generated by Xilinx System Generator for all three DAB receiver modules, is synthesized in addition to the static logic of the top-level design. For each synthesis operation one netlist file with constraint information (NGC) is obtained, which is required for successive implementation. Since the static partition needs to be operational during reconfiguration, the logic placement and wire routing need to be the same for all DPR modules. Therefore, the static logic needs to be implemented before the implementation of the DPR modules and the placement and routing information of the static partition are imported and subsequently supplied to the DPR module implementation process. For differential DPR, a total number of four

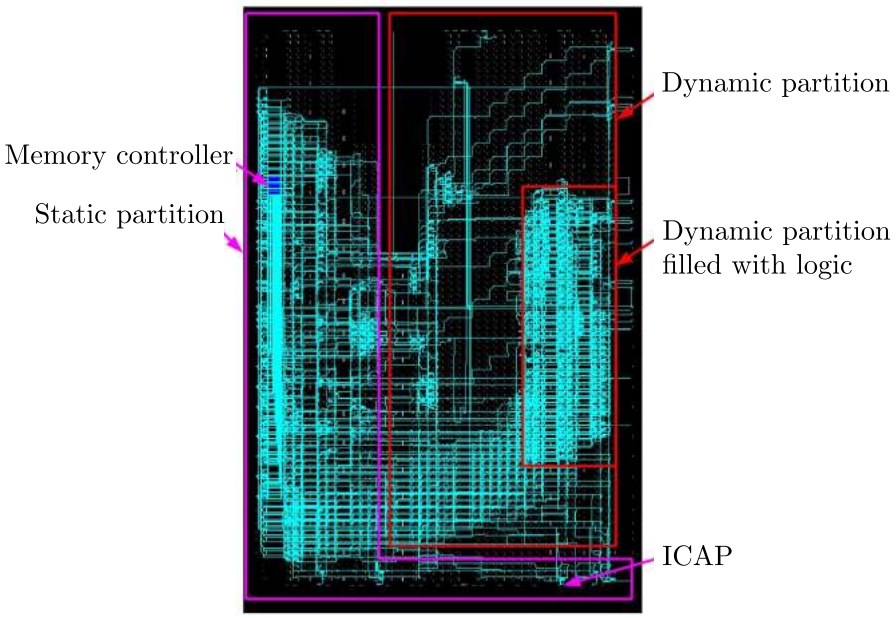


Figure 3.24: Single-island test system for Spartan-6 FPGA reconfiguration.

bitstreams are required, comprising of the initial non-differential bitstream including the first DPR module configuration, and three differential bitstreams containing the differences between the DPR modules. Figure 3.25 shows the bitstream generation procedure in a simplified diagram.

Finally, albeit intensive testing and debugging of the DPR system, **a functional DPR design for the DAB receiver could not be realized reliably** with the approaches outlined by Meyer and Schmidbauer, i.e. with the same techniques that have successfully been applied in the evaluation phase. The reason for the malfunctioning of the system is that although the Xilinx tools constrain the logic placement by the area constraints provided, the routing constraints are ignored by the tools, such that the a reconfiguration of the DPR partition might lead to a reconfiguration of a wire routing matrix inside the static partition. This has been observed to be the case if the DPR module requires too many routing resources, such that the Xilinx place and route tool starts using wires outside the DPR partition. This issue did not appear in the test and evaluation system since the DPR module has sufficiently been separated from the static partition (cf. Figure 3.24). In case the DPR partition resources are sufficiently utilized, such that routing resources outside the DPR partition are occupied, routing matrix reconfigurations inside the static region can then lead to glitches in the static logic during DPR module reconfiguration. Note that this issue has already been reported in Section 2.5, where the use of blocker macros has been proposed to circumvent this drawback. As already pointed out, blocker macros or other tools for XDL macro generation have not been evaluated in this work. Since the system architecture could be verified by means of a DPR simulation model on the ZTEX board, the feasibility of the cyclic DPR implementation could be verified and the remaining uncertainty of the cyclic DPR systems constitutes in the size of the partial bitstreams.

The resource utilization and system constraints of the DPR-based design will be com-

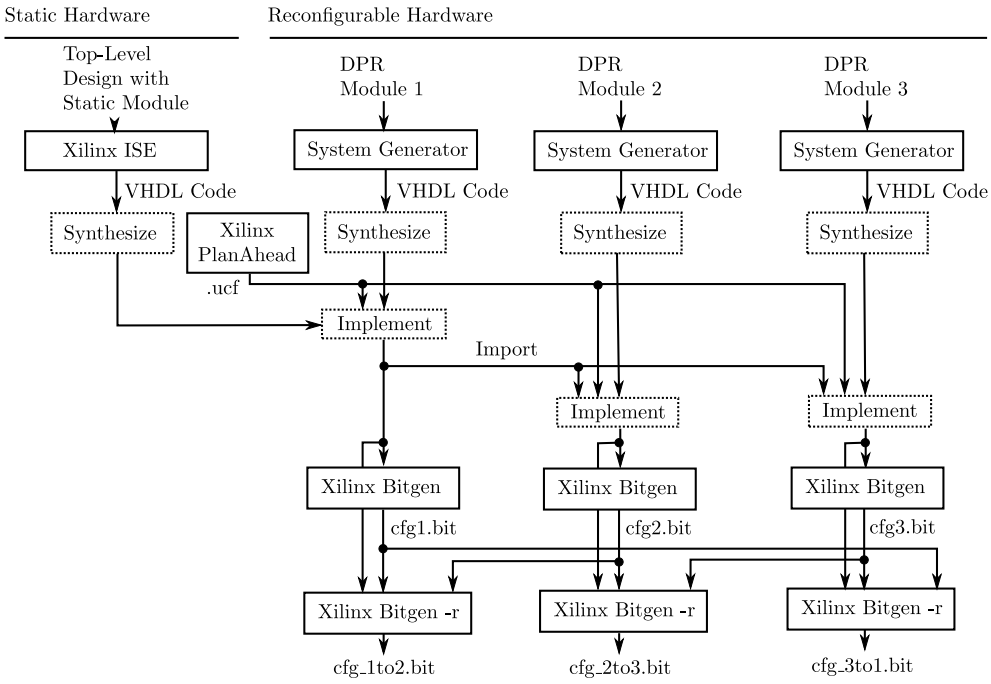


Figure 3.25: Difference-based bitstream generation tool-flow for the DAB receiver.

pared to the static DAB receiver system in the next section.

3.4.3 Resource Utilization and Comparison

The resource occupation of the three DPR modules has previously been outlined in Table 3.10. Inside the single-island cyclic DPR receiver system, enough space needs to be reserved for the DPR partition, where the minimum amount of required resources elements is determined by the worst-case resource occupancy of the DPR modules. Regarding slices and, hence, also FF and LUT resources, the third DPR module dominates the size requirements with 779 slices. In terms of BRAM resources, the second DPR module is the most demanding with 12 BRAMs, where the FFT is the dominating processing element in the chain. In terms of DSP48 units, the second module is the most demanding among the DPR modules with a total requirement of 14 DSP units. The resource requirements of the DPR partition and the components of the static partition of the DPR system are summarized in Table 3.12.

In the following, the DAB receiver implementation as presented in Section 3.3 will be denoted as *original* implementation. The static DPR system environment requires no DSP resources and only a few BRAM resources for the USB I/O FIFOs and for the ICAP loader FIFO, which is where the bitstream memory addresses are stored. In comparison to the DAB receiver interface resources listed in Table 3.9, the USB COM FSM requires more slice resources in the DPR design due to the increased complexity for bitstream uploading, address uploading and memory interface communication. In comparison to the static environment of the DAB receiver implementation presented in Section 3.3, the required amount of slices is roughly increased by a factor of 4.

At this point it is important to bear in mind that the resource listing of the static

Description	Slices	FFs	LUTs	BRAMs	DSP48
DPR Partition	779	2026	1490	12	14
USB COM FSM	471	747	841	4	0
DPR MEM I/O	117	165	314	0	0
ICAP LOADER	77	143	241	1	0
DPR FSM	7	7	8	0	0
DPR I/O GATE	1	37	1	0	0
Others	160	229	374	0	0
Total	1612	3354	3269	17	14
XC6SLX25	3758	30064	15032	52	38

Table 3.12: DPR-based DAB receiver resource utilization on ZTEX module 1.11c.

environment in the original system misses the resource requirements of the DDR memory interface controller. However, the original DAB receiver implementation can also function without external memory by using an additional amount of 29 BRAMs. This modification in turn would make the area constraining of the DPR partition impractical on the XC6SLX25 FPGA. Therefore, the following resource comparison between the static environment of the original system and the DPR system, i.e. between Table 3.9 and Table 3.12, is considered as biased in favor of the original system. Albeit this bias, it will be shown that the cyclic DPR system is still more resource-efficient in comparison to the original implementation since the increased amount of resources of the static environment in the cyclic DPR system is compensated by the reduction in FPGA resources for the DAB receiver. In comparison to the original system, in the cyclic DPR system the amount of DAB receiver slices is approximately reduced by a factor of 2.4. Since the DAB receiver occupies the major proportion of the FPGA slices, the DPR system turns out to require fewer resources in comparison to the original implementation.

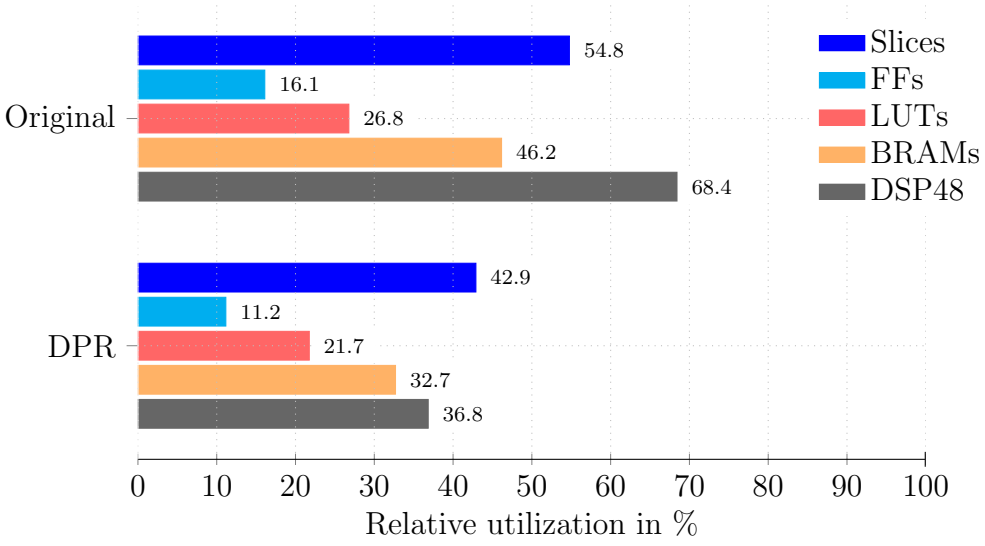


Figure 3.26: Relative resource consumption for DAB receiver on XC6SLX25.

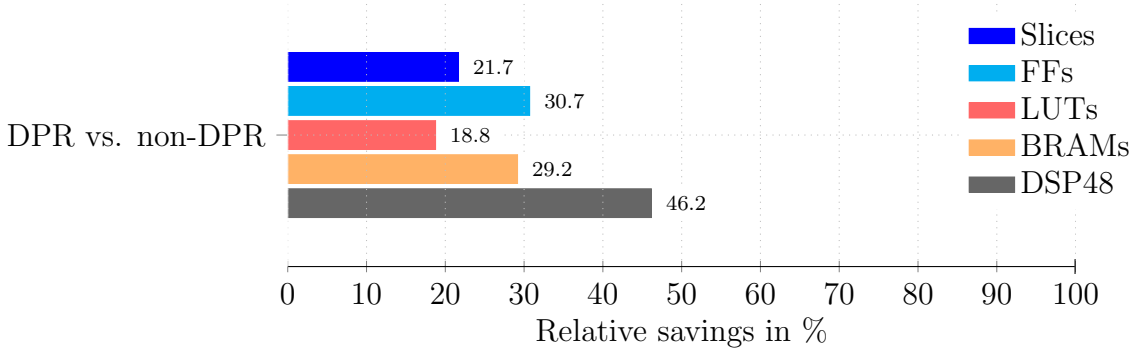


Figure 3.27: Relative resource saving using DPR compared to static design.

The relative resource utilization of the original non-DPR design and the cyclic DPR implementation is outlined in Figure 3.26 with respect to the resources available on the Xilinx Spartan-6 XC6SLX25 FPGA on the ZTEX board. For all resource elements, the cyclic DPR system turns out to require fewer resources in comparison to the static design. The relative resource savings are depicted in Figure 3.27. The graph expresses that, especially in terms of DSP48 resources, the cyclic DPR design is 46 % more resource-efficient in comparison to the original design.

Note that when presenting these resource reductions there is a caveat, which in general has to be considered when speaking about resource reduction in DPR systems: Recall that each Xilinx FPGA is organized in configuration rows and columns of fixed size (cf. Section 1.1) and that the DPR partition size must be fixed in the floorplanning phase. This means that, unless the DPR partition size fits exactly the resources available in the rectangular reconfigurable FPGA area, there will be a remaining amount of unused resources inside the DPR partition. Moreover, it is unlikely that the DPR modules occupy exactly the same amount of resources inside the DPR partition. Hence, although the cyclic DPR system can still beat the original system in terms of resource occupation, the mentioned resource fragmentation leads to an effective reduction of available resources, meaning that the freed resources can neither be assigned back to the static environment nor to other DPR modules, and that due to these circumstances it is likely that more resources will be occupied or blocked than required in total. Considering these remarks and the constraints induced by the cyclic DPR system model, it is still possible to reduce the effective amount of resources of a cyclic DPR system, which is the case for the presented DAB receiver system. Since DSP48 and BRAM resources are of special concern in digital broadcast receiver systems, time-multiplexing of these components increases the degrees of freedom for the design of DSP systems. Given that low-cost FPGAs usually provide fewer DSP48 and BRAM resources in comparison to their expensive counterparts, cyclic DPR seems to be a viable optimization option in terms of system costs, especially for DSP applications.

So far the resource benefits of the DAB receiver have been evaluated and a DPR flow architecture tailored to the DAB receiver has been presented. The implementation of the DPR DAB system requires a more complex static environment system architecture in comparison to the original implementation. The presented DPR static environment architecture can be adopted for similar receiver chains by changing only a few design parameters, such as the memory mapping and the DPR control state machines. Compared

to the original design, in the cyclic DPR design intermediate data buffers and bitstream storage memory are required, which increases the memory requirements of such a system as outlined in Section 3.2. In the next section, these increased requirements will be quantified for the DAB receiver.

3.4.4 Cyclic DPR Receiver Memory Requirements

In Section 3.2 it has been explained that a cyclic DPR system requires input data buffering and intermediate data buffering to function. Furthermore, for independent operation it is necessary to buffer the DPR bitstreams. The static environment of the DAB receiver must store the incoming data and the bitstream information.

Baseband Input Buffer

Since the DAB receiver operates on DAB frames with $T_{FRAME} = 96\,\text{ms}$ duration, the input buffer has been designed to store the complex baseband samples of at least two DAB frames, which at a sample rate of 2,048 MHz and 8 bits per I and Q sample results in a memory requirement of

$$2 \cdot 2.048 \cdot 2\,\text{MByte/s} \cdot 96\,\text{ms} = 786.432\,\text{kBytes}.$$

Intermediate DPR Module Data Buffer

The buffer size of the symbol buffer bridging the DPR modules 1 and 2 is mainly related to the useful symbol duration and the number of symbols per DAB frame. As defined in Table 3.4 and shown in Figure 3.10, one CIF contains 18 OFDM symbols and one DAB frame consists of 4 CIFs carrying the MSC information. Hence, a DAB frame carries 72 MSC symbols. In addition, a DAB frame includes 3 FIC symbols and one reference symbol, which is required by the DQPSK stage. Altogether, 76 OFDM symbols have to be stored in the intermediate buffer to be transferred from DPR module 1 to DPR module 2. Since only the useful part of the OFDM symbols are forwarded, each symbol has a duration of 1 ms in time, which results in

$$76 \cdot 1\,\text{ms} \cdot 2.048\,\text{MS/s} \cdot 2\,\text{Bytes} = 311.296\,\text{kBytes}$$

of required symbol storage memory. As already mentioned, the time deinterleaver has been redesigned to allow for storing the values in the external memory with 4 bits per LLR. The design decision to use external memory instead of BRAMs resulted from the fact that the memory resource consumption of the time deinterleaver is very high compared to the amount of logic resources (cf. [Gna12]). Although increasing the number of bits per LLR increases the reliability information of a demapped bit and thus improves the channel decoding performance, a choice of 4 bits per LLR shows a good trade-off between memory requirement and decoding robustness since the BER performance only improves marginally by spending more bits. Given a worst-case MSC bitrate of 386 kBit/s (using UEP) the deinterleaver memory requirement becomes

$$\frac{416\,\text{CUs}}{\text{CIF}} \cdot 16\,\text{CIFs} \cdot \frac{4\,\text{bit}}{\text{LLR}} \cdot \frac{64\,\text{LLRs}}{\text{CU}} = 1,703,936\,\text{bit} = 212.992\,\text{kBytes},$$

which is the amount of data that has to be transferred per DAB frame from DPR module 2 to DPR module 3. Apart from the MSC information stored in the time deinterleaver, the FIC information needs to be stored in memory before it can be decoded in the third DPR module. The intermediate FIC buffer capacity is designed to hold 3 FIC symbols with 768 bits per symbol and 4 bit per LLR, such that

$$3\,\text{sym} \cdot \frac{768\,\text{LLRs}}{\text{sym}} \cdot \frac{4\,\text{bit}}{\text{LLR}} = 9,216\,\text{bit} = 1.152\,\text{kBytes}$$

of additional memory is required to store the FIC soft-bit information. The amount of data per DAB frame in the cyclic DPR DAB receiver is summarized in Table 3.13. Note that the output of the third DPR module does not need to be buffered but is fed to the USB communication interface and forwarded to the USB host PC. The output of DPR module 1 dominates the memory requirements and the intermediate buffer must be designed to store at least 311.296 kBytes of data. Since the context of the convolutional time deinterleaver depends on the previously decoded DAB frame its memory needs to be persistent, which is why an additional amount of 212.992 kBytes of data needs to be available for continuous deinterleaving. Since the OFDM symbol buffer is only required as intermediate buffer for the DPR modules 1 and 2, the OFDM symbol buffer can be re-used and overwritten by the FIC LLR values generated by the second DPR module, such that no additional buffer space needs to be allocated.

Description	Transfer direction	Size in kBytes
OFDM symbol buffer	Module 1 → Module 2	311.296
MSC LLR buffer	Module 2 → Module 3	212.992
FIC LLR buffer	Module 2 → Module 3	1.152

Table 3.13: Amount of data transferred between DPR modules per DAB frame.

Bitstream Buffers and Memory Map

The memory requirements of the cyclic DPR system are summarized in Table 3.14. As already explained, in comparison to the original receiver implementation, the cyclic DPR system requires additional memory for buffering and bitstream storage. Although the amount of memory can be reduced by CIF-based processing, changing the frame duration imposes other challenges on the system design, as explained in Section 3.3.5. Using BRAMs for buffering is not recommended as they are likely too small and typically required for buffering tasks in the PEs of the receiver. Given that the external memory has already been available on the ZTEX board, it could be utilized without additional cost. Hence, for FPGA systems with external memory, using cyclic DPR to reduce the system resources is feasible. For tailored FPGA systems it needs to be investigated whether the reduction in FPGA resources by using cyclic DPR justifies the costs of adding a dedicated external memory chip to the bill of materials.

The size of the DPR bitstreams typically increases with the size of the DPR partition area. Since a stable cyclic DPR system could not be established with the Spartan-6 FPGA, no reliable bitstream size information is available to determine the memory

Description	Size in kBytes
Baseband buffer	786.432
Intermediate buffer	311.296
Time deinterleaver	212.992
Bitstream buffer	$\sum_{m}^{3} \frac{N_{\mathrm{B,BIT},m}}{8}$

Table 3.14: Buffer memory requirement of the cyclic DPR DAB receiver.

requirements for bitstream storage. For a complete device reconfiguration 6,440,432 configuration bits need to be written to the XC6SLX25 FPGA (cf. [Xil15c]). Thus, in the worst case scenario, roughly $3 \cdot \frac{6{,}440{,}432}{8 \cdot 1000} \approx 2400\,\mathrm{kBytes}$ of bitstream storage capacity must be reserved in the cyclic DPR receiver system.

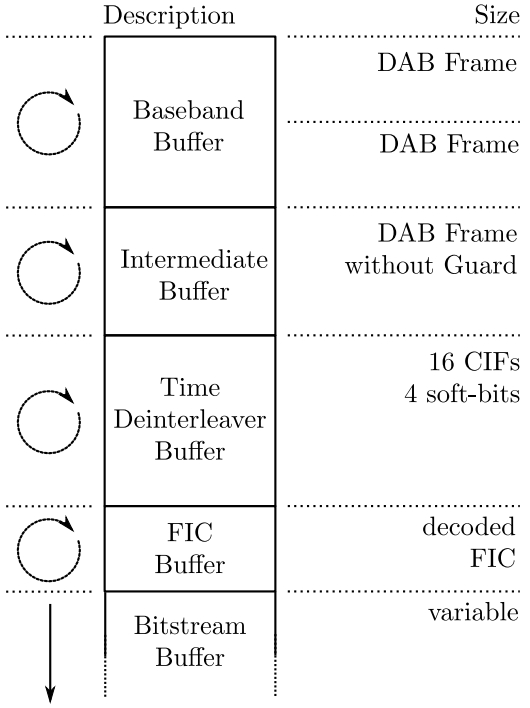


Figure 3.28: External memory map of the cyclic DPR DAB receiver.

The memory partitioning of the respective components is shown in Figure 3.28. The memory is word addressable with 16 bits per word and apart from the bitstream memory region, all other buffer memory regions are of fixed size. The cyclic DPR system environment as well as the memory partitioning approach can be applied to new cyclic DPR broadcast receiver designs.

In the next section, a feasibility study for a DVB-T2 baseband decoder using cyclic DPR will be presented, where in contrast to the cyclic DPR DAB receiver presented in this section, the system feasibility will be evaluated given estimates of the resource occupation and timing behavior of the receiver implementation.

3.5 Feasibility Analysis for a DVB-T2 Baseband Decoder using Cyclic DPR

In this section, a feasibility study for a DVB-T2 baseband decoder chain on a Xilinx Kintex FPGA using cyclic DPR will be presented with the help of the cyclic DPR system model. Given realistic system constraints, the DPR system model allows for an exploration of the real-time capability and latency of the cyclic DPR system, such that it is not strictly necessary to have an implementation of the system available. Taking a DVB-T2 baseband decoder as an example, the following analysis will introduce the necessary steps for conducting such an feasibility study to explore the cyclic DPR design space. Similar to the previously introduced baseband chains, the feasibility analysis encloses baseband demodulation only, which means that decoding and interpretation of the baseband frames including the transport stream is not part of this work.

An overview on the DVB-T2 baseband system, the receiver architecture and DPR module partitioning will be outlined next.

3.5.1 System Architecture

The DVB-T2 baseband receiver shown in Figure 3.29 is assumed to be similar to the generic OFDM receiver presented in Section 1.2.2. For cyclic DPR, the receiver chain will be split into two parts with approximately equal computational complexity, namely a demodulator part, in the following denoted by *DEMOD* and a forward error correction part, referred to as *FEC*. The motivation for splitting the system into two parts is that the resources inside the DPR partition can be reused in time by both parts. The DEMOD part contains frequency and time offset estimation and compensation routines, the FFT stage, channel estimation and equalization routines as well as a cell deinterleaver whereas the FEC part contains an LLR deinterleaver, a demapper, an LDPC decoder and a BCH decoder.

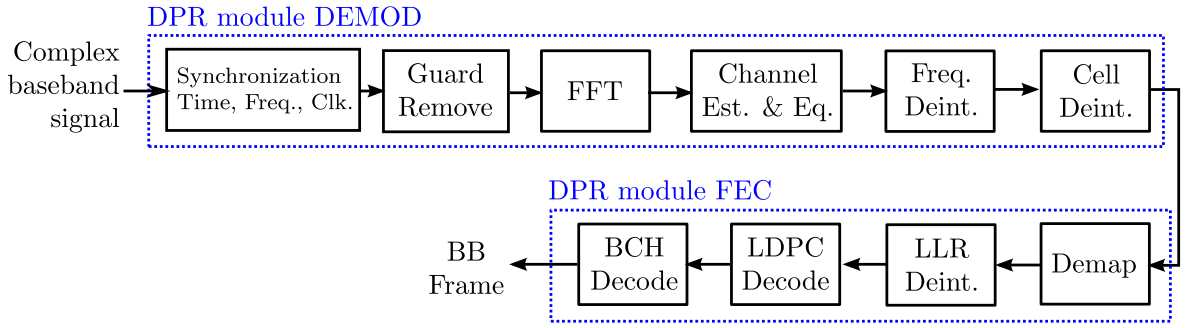


Figure 3.29: Signal flow-graph of a DVB-T2 receiver chain and DPR module partitioning.

To simplify the calculations, the following analysis does not include L1 signaling frames, i.e. no P1 and P2 header symbols, but only the data symbols, which make up the major payload in the DVB-T2 receiver system as outlined in [Ets08]. In terms of hardware complexity the FEC decoder is likely to consume most of the resources (c.f. [KVW+10] and [CA11]). When the LDPC codeword must be calculated using a high number of iterations, high DVB-T2 payload data rates of up to 50.3 Mbit/s make

real-time decoding a computationally complex task. Therefore, in the following it is proposed to allocate both channel decoders to a separate DPR module. Let the DVB-T2 receiver be partitioned into two modules, namely FEC and DEMOD, which are subsequently executed and reconfigured.

The DPR modules are executed in a dedicated DPR partition and the design follows the single-island reconfiguration design similar to the DAB receiver prototype presented in Section 3.3. After a certain number of module execution cycles, the DPR partition is reconfigured and another module is made ready for processing. The proposed time-multiplexing approach is depicted in Figure 3.30. Each DPR module is executed a certain number of times before it is replaced by the other DPR module, i.e. the DEMOD part is executed $N_{EX,DEM}$ times and the FEC part is executed $N_{EX,FEC}$ times. After one execution cycle the DEMOD module is replaced by the FEC decoder module and vice versa.

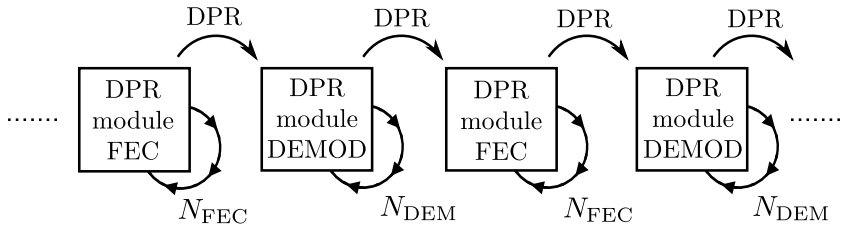


Figure 3.30: Cyclic execution of FEC and DEMOD modules using DPR.

Implementation	Slices	36k BRAMs	DSP48
Xilinx Kintex 7K160T	25350	325	600
Xilinx LDPC/BCH decoder IP	14008 (55.3 %)	71 (21.8 %)[9]	16 (2.7 %)

Table 3.15: Resource requirements for FEC part on Xilinx Kintex FPGA.

Literature reports that real-time capability and acceptable decoding performance for the DVB-T2 LDPC decoder requires either spending many FPGA resources (cf. [LNJ+11]) or using higher clock rates to achieve a minimum number of iterations for belief propagation. This is an indication that the FEC decoder is the most performance-critical part of the system. Therefore, a DVB-C2 LDPC/BCH decoder IP core [Xil] offered by Xilinx and developed by Creonic has been taken as a reference to determine the size of the DPR partition. The IP core includes all components as defined in DEMOD module, i.e. a soft-decision demapper, a block deinterleaver, an LDPC decoder, a BCH decoder and a descrambler (cf. Figure 3.29). The resource requirements of the implementation generated with Xilinx ISE 13.4 are presented in Table 3.15. An uncompressed bitstream for a complete configuration of the Kintex 7K160T FPGA comprises of 53,540,576 bits (cf. [Xil15a]). Since the decoder IP consumes 55.3 % of the slice resources, the number of bits for the partial bitstream will subsequently be defined in proportion to the resource utilization. To account for fragmentation when defining the reconfigurable area

[9] The datasheet provided by Xilinx does not specify whether the number of BRAMs refers to 18k BRAMs or 36k BRAMs.

(cf. Section 2.6), a total of 60 % FPGA utilization is assumed to be allocated to the DPR partition. Hence, the proportional size of the partial bitstream to configure the DPR partition will be conservatively estimated to be 32,124,346 bits. This portion of the FPGA can then be used for both, the DEMOD and the FEC module implementations. Xilinx 7 series FPGAs provide an ICAP interface and full DPR support and as defined in [Xil14], where the ICAP interface has a width of 32 bit and supports a maximum clock frequency of 100 MHz. Considering the outlined values, the reconfiguration time for the DPR partition can be estimated to be

$$T_{\mathrm{DPR}} = \frac{32,124,346\,\mathrm{bits}}{32\,\mathrm{bits}\cdot 100\,\mathrm{MHz}} \approx 10\,\mathrm{ms}. \tag{3.15}$$

After the introduction of the time-multiplexing approach using partial reconfiguration, the realtime constraints of such a system will be presented and parameters for the execution times $\mathrm{N_{EX,DEM}}$ and $\mathrm{N_{EX,FEC}}$ will be derived.

3.5.2 Real-Time Constraints

For a broadcast receiver to be realtime capable, the implementation must cope with the data throughput defined in the system specification. Since in DVB-T2 different bandwidths, guard interval lengths and different FFT sizes may be used, an analysis for all configurations of the system might become cluttered. In order to account for the computationally most demanding scenario, a DVB-T2 signal bandwidth of 8 MHz with an elementary period of $\tau = 7/64\,\mu$s is considered for the analysis. Additionally, the guard interval fraction will be defined to be $\Delta = 1/128$. In this configuration the OFDM symbol duration $\mathrm{T_S}$ can be calculated by

$$\mathrm{T_S} = (\Delta + 1)\cdot\tau\cdot\mathrm{N_{FFT}} = \frac{903}{8192}\,\mu\mathrm{s}\cdot\mathrm{N_{FFT}}, \tag{3.16}$$

where $\mathrm{N_{FFT}}$ defines the number of complex samples for the Fourier transform operation. To allow for Doppler shifts, tone reservation and pilot carriers only $\mathrm{N_C}$ of $\mathrm{N_{FFT}}$ bins are used as data carriers. Table 3.5.2 gives an overview on the number of used carriers for the different FFT sizes.

$\mathrm{N_{FFT}}$	1024	2048	4096	8192	16384	32768
$\mathrm{N_C}$	853	1705	3409	6817	13633	27265

Table 3.16: Number of used carriers $\mathrm{N_C}$ for $\mathrm{N_{FFT}}$ FFT bins.

For the data carriers either 4-, 16-, 64- or 256-QAM mapping with either 2, 4, 6 or 8 bits per constellation point can be used, such that the number of bits per OFDM symbol can be specified to be

$$\mathrm{N_{LLR}} = \mathrm{N_C}\cdot q, \text{where } q \in \{2,4,6,8\}.$$

Concerning the channel coding, the broadcaster can decide between two FEC codeword lengths together with different code rates. Since the performance for the worst case processing load is of major concern, the analysis is based on the LDPC long code with

64800 bits per codeword, neglecting the LDPC short code. Decoding one long codeword requires the FEC module to be executed

$$\mu = \frac{\mathrm{N_{LLR}}}{64800}$$

times per OFDM symbol. Hence, the higher the constellation alphabet and the more carriers are in use, the higher the required execution frequency of the FEC decoder per OFDM symbol. With the derived system constraints, the execution time of the cyclic DPR system can be evaluated. Let the execution time per OFDM symbol of the FEC part further be denoted by $\mathrm{T_{EX,FEC}}$ and the execution time of the DEMOD part by $\mathrm{T_{EX,DEM}}$. Furthermore, it is assumed that context recovery can be neglected, i.e. $T_{\mathrm{SV,FEC}} = 0$ and $T_{\mathrm{SV,DEM}} = 0$. With the parameters derived so far it can be stated that in a system without reconfiguration delay and without setup times, realtime processing is possible if the following inequality is satisfied

$$\mathrm{T_{EX,DEM}} + \mathrm{T_{EX,FEC}} \cdot \mu + C \leqslant \mathrm{T_S}, \tag{3.17}$$

where $C > 0$ since the reconfiguration time T_{DPR} is nonzero and the FEC and DEMOD modules exhibit an initialization time. Considering that the processing is done on a symbol-wise basis in bursts of $\mathrm{N_{SYM}}$ OFDM symbols, the execution time to process $\mathrm{N_{SYM}}$ symbols, including the reconfiguration and setup delays, reflects the cycle time T_{CYC} of the DPR system, i.e.

$$T_{\mathrm{CYC}} = \mathrm{N_{SYM}} \cdot \mathrm{T_S}.$$

According to the cyclic DPR system model the upper bound on the cycle time for real-time operation is

$$\mathrm{T_{FRAME}} \leqslant \mathrm{N_{SYM}} \cdot \mathrm{T_S}. \tag{3.18}$$

With the knowledge of $T_{\mathrm{LD,DEM}}$ reflecting the initialization time for the DEMOD implementation to start the processing and $T_{\mathrm{LD,FEC}}$ being the setup time for the FEC implementation, the maximum frame duration $\mathrm{T_{FRAME}}$ can be derived by

$$\mathrm{T_{FRAME}} = \mathrm{N_{SYM}} \cdot (\mathrm{T_{EX,DEM}} + \mathrm{T_{EX,FEC}} \cdot \mu) + T_{\mathrm{LD,DEM}} + T_{\mathrm{LD,FEC}} + 2 \cdot T_{\mathrm{DPR}}, \tag{3.19}$$

where in this case the constant additive part C in Equation 3.17 is determined by

$$C = \frac{T_{\mathrm{LD,DEM}} + T_{\mathrm{LD,FEC}} + 2 \cdot T_{\mathrm{DPR}}}{\mathrm{N_{SYM}}}. \tag{3.20}$$

Hence, with an arbitrary large number of processed OFDM symbols $\mathrm{N_{SYM}}$ the constant C can be made arbitrary small and the condition for realtime-capability in the reconfigurable system can be stated with

$$\mathrm{T_{EX,DEM}} + \mathrm{T_{EX,FEC}} \cdot \mu < \mathrm{T_S}. \tag{3.21}$$

Bearing this condition in mind, it is possible to define the minimum number of symbols $\mathrm{N_{SYM}}$ required to achieve real-time capability. In the following part of the analysis the question on suitable parameters for $T_{\mathrm{LD,DEM}}$ and $T_{\mathrm{LD,FEC}}$ will be answered and the execution times $\mathrm{T_{EX,DEM}}$ and $\mathrm{T_{EX,FEC}}$ for a DVB-T2 receiver system will be specified.

3.5.3 Feasibility Analysis

The execution times of the FEC and DEMOD accelerators must be specified in order to cope with the worst-case throughput of a DVB-T2 broadcast, i.e. when a 32k FFT together with a 256-QAM modulation is used. The symbol duration in this mode is $\mathrm{T_S} = 3612\,\mu\mathrm{s}$ and the number of FEC executions per symbol is approx. $\mu = 3.4$. Since the complexity of the demodulation part varies with the FFT size and modulation alphabet. The number of execution cycles for the 2048-point pipelined streaming FFT implementation is stated to be 4223, cf. Table 3.6 in the previous section. For a DPR module execution frequency of 100 MHz the execution time of a 2048-point FFT results in $42.23\,\mu\mathrm{s}$. For the execution time of half the FFT size, i.e. for 1024 FFT bins, the execution time is further on assumed to be $\mathrm{T_{EX,DEM,1k}} = 25\,\mu$. Linearly scaling this value returns an execution time of $\mathrm{T_{EX,DEM,1k}} \cdot 32 = 800\,\mu\mathrm{s}$ for a 32k FFT.

In contrast to the DEMOD part, the FEC decoder must be designed to operate on a finite frame length and, thus, the execution time $T_{\mathrm{EX,FEC}}$ should be fixed as well. Given the previously derived parameters and reformulating Equation 3.21, the limit for the FEC execution time can be calculated as

$$\mathrm{T_{EX,FEC}} < \frac{\mathrm{T_S} - \mathrm{T_{EX,DEM,1k}} \cdot 32}{\mu} = 835\,\mu\mathrm{s} \tag{3.22}$$

The value derived in Equation 3.22 requires an FEC throughput of 78 Mbit/s. With respect to the inequality, the analysis can be started from a value of $\mathrm{T_{EX,FEC}} = 800\,\mu\mathrm{s}$, i.e. with a throughput of 81 Mbit/s. Next, the initialization latency of the FEC and DEMOD modules will be approximated by $T_{\mathrm{LD,DEM}} = 0.5 \cdot \mathrm{T_{EX,DEM}}$ and $T_{\mathrm{LD,FEC}} = 0.5 \cdot \mathrm{T_{EX,FEC}}$. Up to this point, all important parameters of the cyclic DPR system have been derived. Next, system aspects such as memory consumption and DPR module throughput will be discussed.

DPR Module Throughput

For the configuration derived in the last section, the minimum number of execution cycles for the FEC and DEMOD parts are depicted in Table 3.5.3. The numbers show that the minimum number of execution cycles for the DEMOD part $\mathrm{N_{EX,DEM}}$ linearly increases with the FFT size, which is due to the fact that the symbol duration $\mathrm{T_S}$ also linearly increases with an increase of the FFT size, cf. Equation 3.16. Since the produced data throughput at the FFT output stays almost constant with a varying FFT size, also the minimum number of execution cycles for the FEC part $\mathrm{N_{EX,FEC}} = \mathrm{N_{SYM}}$ stays constant. If the mapping constant q increases, i.e. if the QAM cells carry more bits, both accelerators have to cope with a higher data throughput and, hence, the number of execution cycles per multiplex cycle increases.

Figure 3.31 shows in how far the FEC decoder throughput influences the number of execution cycles for the FEC and DEMOD parts. It can be observed that the higher the FEC throughput the smaller the number of execution cycles per reconfiguration cycle. In the most demanding configuration, i.e. using a 32k FFT with 256 QAM mapping, the throughput of 81 Mbit/s exceeds the plot with approximately 600 execution cycles for the FEC decoder module. For an FEC throughput of around 100 Mbit/s, which is around twice the maximum throughput of a DVB-T2 BB frame, the implementation starts becoming feasible even for the most complex configuration. Due to the finite

FFT	4-QAM		16-QAM		64-QAM		256-QAM	
	$N_{EX,DEM}$	$N_{EX,FEC}$	$N_{EX,DEM}$	$N_{EX,FEC}$	$N_{EX,DEM}$	$N_{EX,FEC}$	$N_{EX,DEM}$	$N_{EX,FEC}$
1k	319	9	466	25	863	69	5869	619
2k	160	9	233	25	431	69	2897	610
4k	80	9	117	25	216	69	1441	607
8k	40	9	59	25	108	69	720	606
16k	21	9	30	26	55	70	361	608
32k	11	10	15	26	28	71	182	613

Table 3.17: Minimum cycle times for $T_{EX,FEC} = 800\,\mu s$ and $T_{EX,DEM,1k} = 25\,\mu s$.

LDPC code length, the number of FEC cycles reduces with the number of QAM cells per second produced at the output of the demodulator. Hence, in a 4k FFT / 16 QAM configuration the number of FEC execution cycles is always lower than the number of DEMOD cycles. A reversal of this trend can be observed if more bits are required to be processed in time, for example when using a 32k FFT / 256 QAM configuration. In this case, the FEC becomes the dominant block. If the FEC throughput is sufficiently high, i.e. at 108 Mbit/s, a higher order FFT results in a reduced the number of execution cycles for the DEMOD module because the duration of an OFDM symbol increases linearly with the FFT length (cf. Equation 3.16). Given the OFDM symbol duration T_S the execution duration of the DEMOD module can be derived from the number of execution cycles.

So far the relationship between the number of execution cycles $N_{EX,FEC}$ and N_{FFT} and the throughput of the FEC and DEMOD modules have been described. Next, the relation between the number of execution cycles and the memory consumption of the system will be outlined, including an analysis of the DPR system latency as described by the cyclic DPR system model.

3.5.4 Memory Constraints

In each of the DEMOD and FEC decoders a burst of N_{SYM} symbols must be processed during the DPR module activity cycle. This requires to pre-buffer N_{SYM} symbols of data before the cyclic DPR processing of the FEC and DEMOD modules can be initiated (cf. Figure 3.7). While the two DPR modules are reading the first N_{SYM} symbols from the buffer, the incoming stream of the following N_{SYM} symbols needs to be buffered to be processed next. If double-buffering is used the input buffer must be designed to hold two complete frames, i.e. $T_{BUF} = 2 \cdot T_{FRAME}$. If the input FIFO can be read and written at the same time, only one frame needs to be buffered. Hence, the number of symbols should be kept as small as possible in order to reduce the memory consumption and thus the cyclic DPR system delay. For the DVB-T2 receiver system, this requires the FEC decoder throughput to be as high as possible. In the following, the delay and buffer memory requirements for different FEC decoder rates will be determined.

The complex baseband stream is fed to the input of the DEMOD part. Given an elementary period of $7/64\,\mu s$ and a spectral bandwidth of 8 MHz, the OFDM sample rate required for demodulation is approximately 9.143 MS/s. Streaming the complex baseband signal with 8 bits per complex component results in an baseband datarate of

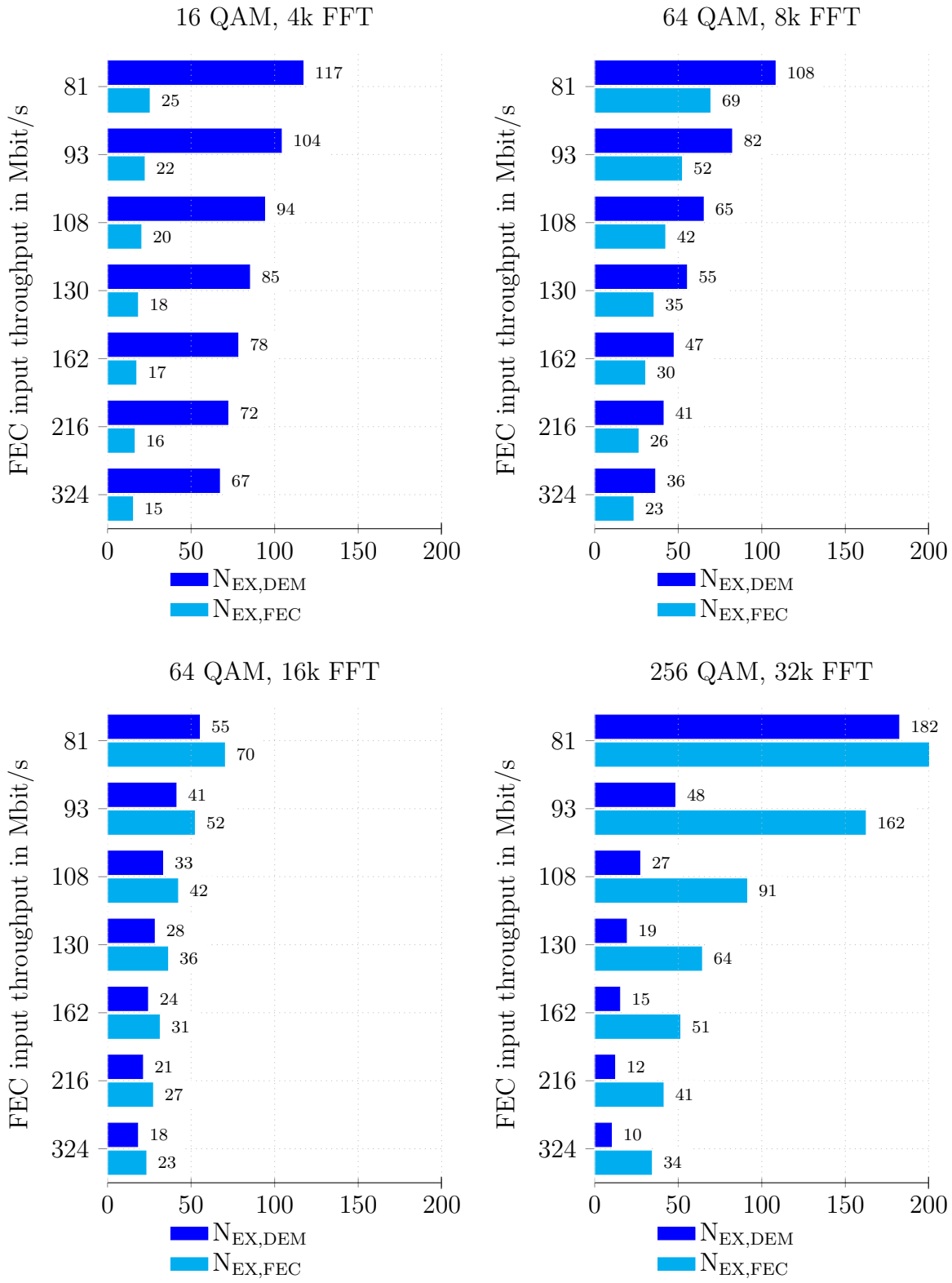


Figure 3.31: Number of DPR module execution cycles for DVB-T2 baseband decoder.

146.29 Mbit/s. The DVB-T2 physical layer pipe (PLP) stream is generated after inner and outer channel decoding have been accomplished and can have a maximum data rate of 50.3 Mbit/s. The sum of the input and output data rates is 196.6 Mbit/s. Thus, the required buffer capacity in bits for the input sample stream becomes

$$\mathrm{N_{B,IO}} = (\gamma_{\mathrm{OUT}} + \gamma_{\mathrm{IN}}) \cdot 2 \cdot \mathrm{T_{FRAME}}.$$

Since $\mathrm{T_{FRAME}}$ depends on the number of OFDM symbols the required memory for input buffering increases with an increase of $\mathrm{N_{EX,DEM}} = \mathrm{N_{SYM}}$. At the output of the DEMOD module a burst of complex equalized carrier values are forwarded to the channel decoder, which must be buffered at the output of the QAM cell[10] deinterleaver (cf. Figure 3.29). The carrier information is assumed to be quantized with 8 bit for in-phase and 8 bit for quadrature component. The buffer memory at the output of the DEMOD module must be designed to have a storage capacity of

$$\mathrm{N_{B,CELL}} = \mathrm{N_C} \cdot \mathrm{N_{SYM}} \cdot 16\,\mathrm{bit}. \tag{3.23}$$

Using a receiver configuration with 32k FFT bins and 256 QAM mapping the QAM cell memory storage requirement per OFDM symbol reaches its maximum. In conclusion, the memory capacity of the system must be large enough to store at least

$$\mathrm{N_B} = \mathrm{N_{B,CELL}} + \mathrm{N_{B,IO}} + 2 \cdot N_{\mathrm{B,BIT}} \tag{3.24}$$

bits of information, where $N_{\mathrm{B,BIT}} \approx 32.1$ Mbit per partial bitstream. In Figure 3.32, the memory consumption and the delay induced by the cyclic DPR operation is shown in relation to the FEC decoder throughput. The bargraph shows that at high FEC throughput rates the buffer memory and delay values for different FFT and QAM configurations converge because the processing duration of the FFT becomes the dominating part in the system. The memory required for bitstream storage dominates the buffer capacity with approximately 8 MBytes in total. For the cyclic DPR system it is proposed to use the configuration with a FEC throughput of 108 Mbit/s. In this configuration the worst-case input buffer size required for real-time operation is $\mathrm{N_B} = 13.7$ MByte and the worst case delay induced by cyclic DPR is 195 ms. Using a 4k FFT with 16 QAM, in the best case an additional latency of 84.9 ms can be observed. The buffer values and the delay values seem feasible for the operation in professional broadcast receivers.

Discussion

Without in-depth implementation knowledge the conducted feasibility study delimits the requirements for the real-time capability of a cyclic DPR system for a DVB-T2 baseband decoder. Reusing the hardware resources available in the DPR partition of the FPGA enables to reduce the logic resources required to implement a DVB-T2 baseband decoder in comparison to a static system. In the DVB-T2 system, the area re-use has been assumed to be 60 % of the overall Kintex FPGA resources (cf. Section 3.5.1). Reduced execution clock frequencies or resource-economic DEMOD and FEC implementations[11] can be employed if larger DPR system delays and buffer I/O buffers are acceptable.

In the next section, the benefits and drawbacks of a cyclic DPR system will be summarized.

[10] A QAM cell is defined as the complex value of the equalized FFT carrier bin.

[11] In the sense that resource-economic implementations require fewer FPGA resources but more execution cycles per operation.

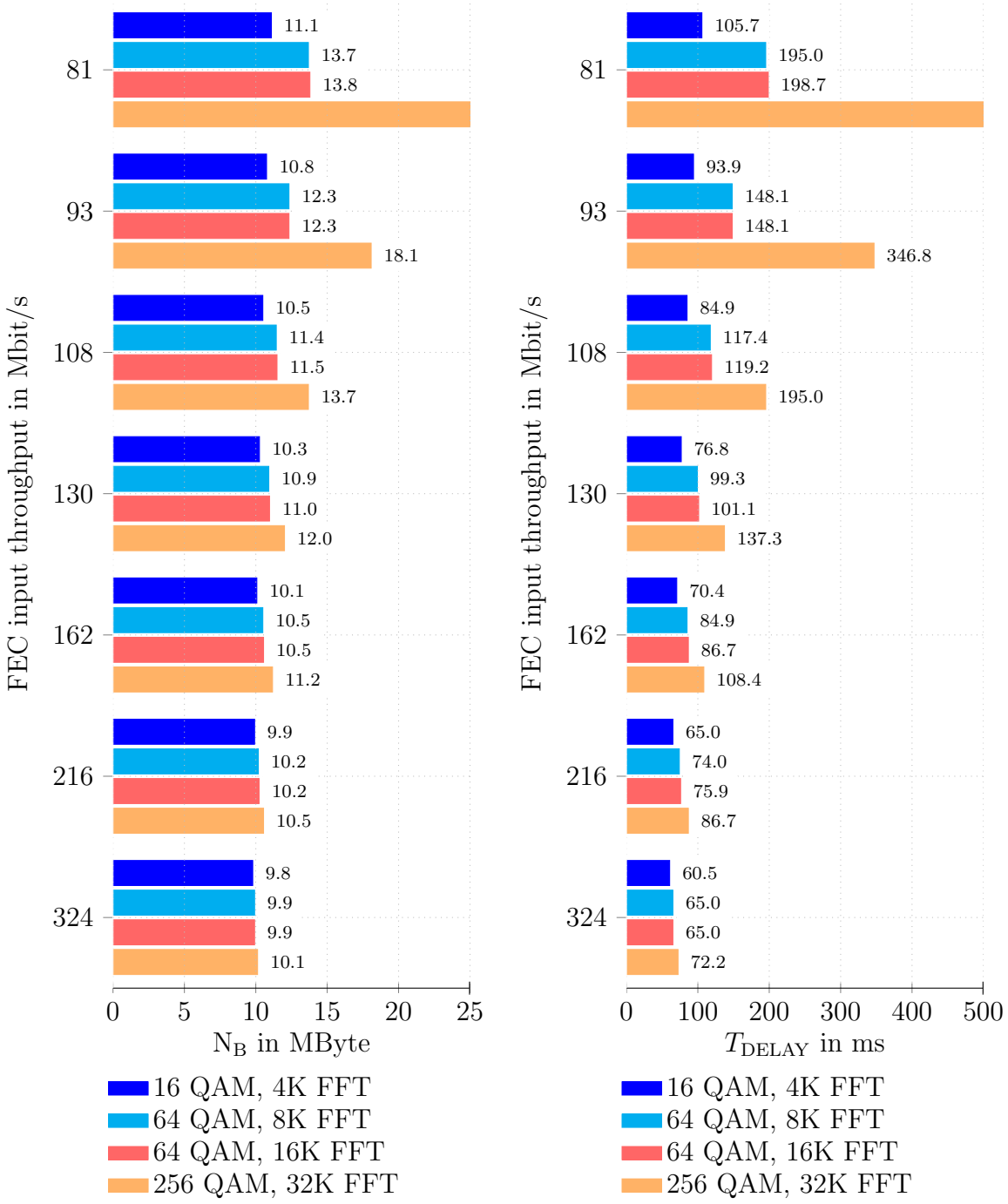


Figure 3.32: Buffer memory and processing delay for DVB-T2 baseband decoder.

3.6 Summary

Related work on cyclic dynamic partial FPGA reconfiguration revealed that existing analytical frameworks for cyclic DPR are incomplete. This motivated the introduction of a cyclic DPR model as presented in Section 3.2. The model enables deriving the timing parameters for the execution of DPR modules given the FPGA platform and DPR module implementation. Using these timing parameters, real-time constraints for the cyclic execution have been formulated. The elaborated theoretical framework has been applied to an existing hardware system from ZTEX and a receiver chain implementation for DAB. After the introduction of the system parameters, the timing constraints of the DAB receiver elements have been discussed and it has been derived that the FFT implementation of the DAB receiver is the bottleneck of the system. This bottleneck determines the minimum execution frequency of 21.2 MHz in the non-DPR receiver system. Together with a suitable partitioning of the DAB receiver into three DPR modules, the frame duration of the receiver elements have been discussed and it has been concluded that a DAB frame-based processing leads to a receiver system with negligible context handling. In relation to the duration of a DAB frame the real-time constrains of the cyclic DPR implementation have been analyzed. The analysis revealed that the memory throughput of the ZTEX platform does neither impose limitations on the transfer load during DPR module execution nor on the bitstream write cycle. Using the number of cycles per DPR module as an upper bound on the execution duration, the analysis revealed that for a 20 MHz ICAP frequency a minimum execution frequency of 52 MHz is required for the DPR modules to be real-time capable. Furthermore, for DAB-based processing a worst-case system delay of 192 ms has been determined. A hardware implementation of the cyclic DPR system on the ZTEX FPGA platform has been outlined. Albeit that DPR on Spartan-6 FPGAs has been reported to be feasible, cyclic DPR could not be realized on this platform, due to insufficient constraining of the signal routing, resulting in interference with the static partition. Using a DPR simulation model the system could be proven as functional. The resource occupation in the cyclic DPR receiver is reduced by a minimum of 18 % per resource element in comparison to the static implementation. The effects of cyclic DPR can also be evaluated without detailed implementation knowledge, as shown by a feasibility study for a DVB-T2 baseband decoder. Using two DPR modules and a minimum FEC throughput of 108 MBit/s the DPR delay of such a system lies between 80 ms and 200 ms. In this configuration 14 MBytes of external memory will be required for intermediate buffers and partial bitstream storage. Finally, as the PEs in cyclic DPR systems typically operate at a higher clock frequency compared to non-DPR designs and since the ICAP and memory resources are continuously accessed, cyclic DPR systems are in general less power efficient compared to non-DPR designs.

The problem of partitioning a sequential chain of PEs into suitable DPR modules for the cyclic execution inside a DPR partition will be addressed in the next chapter. In order to minimize the resource fragmentation when a fixed-size DPR partition is employed, the subsequently presented approach enables finding partitioning solutions with minimum resource variance while also considering the DPR module communication constraints.

4 High-Level Receiver Partitioning for Cyclic FPGA Reconfiguration

The benefits and drawbacks of cyclic DPR systems have been discussed in Chapter 3. As these systems use a single-island DPR partition, it is necessary to appropriately design the DPR modules for this kind of operation. In Chapter 2 it has been concluded that the size of the DPR partition needs to be defined in the design phase and can not be changed during the cyclic DPR operation. Hence, it is beneficial to utilize as many of the limited resources inside the single DPR partition as possible. Since resource elements such as DSP48 slices or BRAMs are of prime importance for signal processing systems, the optimization should be performed with respect to all reconfigurable FPGA resource elements and not solely to a specific type of resource. While this is one goal when partitioning the processing chain into DPR modules, another major concern is reducing the DPR module throughput in the cyclic DPR system, as outlined in the previous chapter. Given both objectives, a partitioning scheme for weighted resource partitioning and joint data throughput minimization will be presented, using a linear combination of partitioning metrics to find DPR module candidates with minimum data throughput and minimum resource utilization variance. In order to classify the approach to be presented, an overview on related works on time-multiplexing of FPGA resources will be given in the next section.

4.1 Related-Work and Contribution

Methods for the identification of suitable partitioning candidates, derived from a predefined set of processing elements and for the operation in single-island cyclic DPR system are of concern in this chapter. Graph-based partitioning approaches for static and time-multiplexed FPGA designs have been presented by Chang, Andersson and Kao et al. in [CMS99], [AK00] and [KTHL07]. Operating on a net-list representation of the processing chain, the major objective in the presented works is to find an optimum schedule for partitions of the processing graph. As the schemes require a circuit description of the chain to be applicable and since neither the memory transfer implications of the cyclic DPR system nor the problem of resource fragmentation are considered, the approaches can not be used to obtain suitable partitioning sets for cyclic DPR. High-level partitioning of sequential modules for cyclic DPR systems is not directly covered in existing literature to the best of the authors' knowledge. However, the partitioning of sequential processing elements as described in this work is related to the problem of partitioning sequential number sequences, for which Zobel et al. presented a solution in [Zob00]. Zobel's partitioning concept has been adopted to be applicable to a set of PR resources in a sequential processing chain as presented in [FIIS12] and [FIVS13]. The analysis is based on the assumption that the number of resources of a PE is approxi-

mately constant for different synthesis and implementation runs as presented in [IFH12]. Considering this constraint, the contributions of this chapter are:

- The presentation of a partitioning approach to derive a set of DPR modules with minimized resource variance and memory throughput for cyclic DPR systems.
- The derivation of a low-complexity tree traversal algorithm to efficiently solve the partitioning problem by exploiting the properties of digital receiver chains.
- A case study for the high-level partitioning of a DAB receiver, which has been introduced in Chapter 3.

The partitioning method to be proposed does not require full PE implementation knowledge, but can also be applied using resource and throughput estimates. Before describing the approach in detail, an introduction to the partitioning problem will be given.

4.2 The Partitioning Problem

Let the set of PEs of a sequential processing chain with N elements be defined by

$$\boldsymbol{P} = \{p_1, p_2, \ldots, p_N\}.$$

The elements in $\boldsymbol{P}$ are linearly ordered and appear in ascending order with respect to their position in the sequential processing chain as defined by the cyclic DPR system model the previous chapter. The goal is to sub-partition the chain into M different sequential modules, where $1 \leqslant M \leqslant N$, such that the set $\boldsymbol{P}$ is divided into M mutually disjoint sub-sets $\boldsymbol{L}_{m,n}$, each forming a sub-sequence of processing elements

$$\boldsymbol{L}_{m,n} =< p_n, p_{n+1}, p_{n+2}, \ldots >, \forall\, m = 1, \ldots, M,$$

where the set index n is the index of the first PE in the set, and the elements of the set $\boldsymbol{L}_{m,n}$ form a DPR module. As $\bigcup_{m=1}^{M} \boldsymbol{L}_{m,n_m} = \boldsymbol{P}$, all PEs are included in the DPR module sets. In the following, an example with $\boldsymbol{P} = \{p_1, p_2, p_3, p_4\}$ PEs and $M = 2$ DPR modules will be provided to better elucidate the partitioning approach. In this case, the following three valid partitioning sets exist:

$$\begin{aligned} \boldsymbol{L}_{1,1} &= < p_1 >, \boldsymbol{L}_{2,2} =< p_2, p_3, p_4 > \\ \boldsymbol{L}_{1,1} &= < p_1, p_2 >, \boldsymbol{L}_{2,3} =< p_3, p_4 > \\ \boldsymbol{L}_{1,1} &= < p_1, p_2, p_3 >, \boldsymbol{L}_{2,4} =< p_4 > \end{aligned} \tag{4.1}$$

Subsequently, let the set $\boldsymbol{T}_k$ contain a sequence of reconfigurable module sub-sets $\boldsymbol{L}_{m,n}$, i.e.

$$\boldsymbol{T}_k = \{\boldsymbol{L}_{1,n_1}, \boldsymbol{L}_{2,n_2}, \ldots, \boldsymbol{L}_{M,n_M}\},$$

where the number of valid partitioning sets $\boldsymbol{T}_k$ is defined by K. As shown in the example in Equation 4.1, the first set in $\boldsymbol{T}_k$ is always $\boldsymbol{L}_{1,1}$ since $n_1 = 1$ for the first element in the set. This means that the first set $\boldsymbol{L}_{1,n_1}$ always contains p_1, i.e. the first PE, as the first element in the set. This leads to the conclusion that the problem of PE chain partitioning can be interpreted as finding suitable starting indices $n_2, n_3 \ldots, n_M$ of the DPR module

sets $\boldsymbol{L}_{m,n}$ in $\boldsymbol{T}_k$. Since all PEs of the chain can be characterized by a common set of parameters, e.g. their resource consumption, processing time, output data rate, etc. , different chain partitionings lead to different realizations of the DPR modules. This in turn has an effect on the overall system performance as some realizations might be more, some might be less efficient given the throughput and area constraints of a cyclic DPR system.

In the following, an approach for finding suitable partitioning candidates will be derived, reflected by the partition starting indices $n_2, n_3 \ldots, n_M$. Given these candidates, a decision metric will be defined to obtain a set of DPR modules with minimum resource variance and minimum output data transfer rate in order to reduce the effects of resource fragmentation in the DPR partition and to reduce the memory transfer load during DPR module operation.

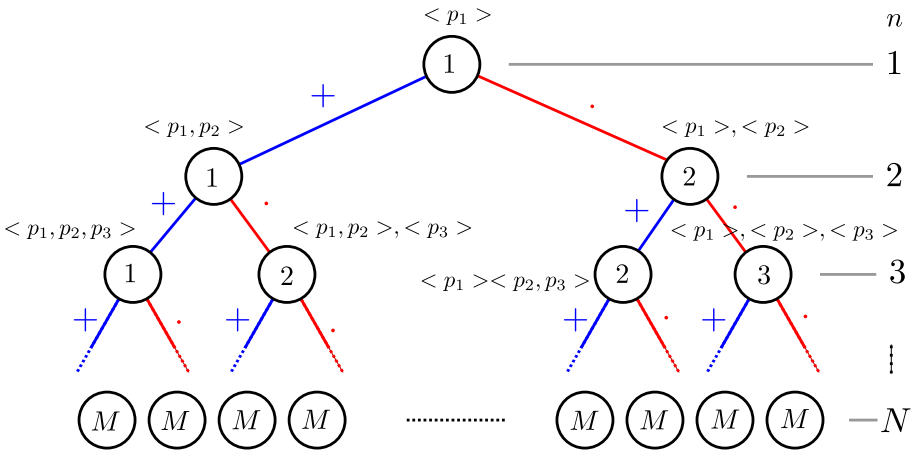


Figure 4.1: Binary tree with possible partitioning solutions inside the leafs.

As already stated, the position of the first PE in $L_{1,1}$ is fixed. Thus, solving the partitioning problem reduces to finding $M-1$ indices out of $N-1$ possible values. The number of solutions K for finding a suitable set of starting indices $n_2, n_3 \ldots, n_M$ is reflected by the binomial coefficient

$$K = \binom{N-1}{M-1} = \frac{(N-1)!}{(M-1)!(N-M)!}, \; N > M > 1. \tag{4.2}$$

Clearly, for $N = M$ and for $M = 1$ the problem is trivial as there exists only a single solution, i.e. $K = 1$. Using exhaustive search reveals all K partitioning sets as follows: As a first step, the set $\boldsymbol{S} = \{\boldsymbol{L}_{1,n=1} =< p_{n=1} >\}$, i.e. $n = 1$, must be be formed. Then, subsequently all possible partitioning sets can be generated by applying the operators *extension* and *composition* to the PE sets in the steps $n = 2, \ldots, N$ as defined by Zobel in [Zob00], where the operator $+$ is used for composition, which means adding another element to the current set $\boldsymbol{L}_{m,n_m}$ in $\boldsymbol{S} = \{\boldsymbol{L}_{1,1}, \ldots, \boldsymbol{L}_{m-1,n_{m-1}}, \boldsymbol{L}_{m,n_m}\}$, and the operator $\cdot$ as extension, which means close the current set $\boldsymbol{L}_{m,n_m}$ and extend $\boldsymbol{S}$ with the new set $\boldsymbol{L}_{m+1,n_{m+1}} =< p_{n_{m+1}} >$. This procedure can be visualized by a binary tree of depth N, where each of the K leafs contain a valid partitioning solution $\boldsymbol{T}_k$ as depicted in Figure 4.1.

The values inside the nodes of the tree reflect the cardinality $|\boldsymbol{S}|$ and the leafs of the tree reflect the partitioning solutions $\boldsymbol{T}_k$. To avoid creating a set with less than M partitions, new composition nodes are not created if $M - |\boldsymbol{S}| \leqslant N - n$. If $|\boldsymbol{S}| \geqslant M$ the creation of extension nodes is omitted created to avoid generating sets with more than M modules. Partitioning candidates of interest are obtained by comparing all leafs and selecting a candidate that fits best certain *performance metrics*. As previously outlined, the goal is to find a candidate in the leafs, where the resource occupation of the modules is balanced and the memory throughput is minimal. Performance metrics to find such a candidate will be derived in the next section.

4.3 Performance Metrics for DPR Module Sets

Before formulating a suitable partitioning metric, it is necessary to discuss the characteristics of a partitioning solution in terms of logic resource balancing and in terms of memory throughput minimization. Given the observations made in the previous chapters, the performance of a partitioning candidate will be classified by how it fulfills the following criteria:

(a) **Minimum resource variance:** As outlined in the previous chapters, it is important to minimize the difference in resource consumption among the reconfigurable modules to reduce the effect of resource fragmentation in the DPR partition and increase the resource utilization of the FPGA.

(b) **Minimum output data throughput:** According to the cyclic DPR system model, reducing the data throughput at the output of the DPR modules leads to a reduction in memory load. Hence, it is desirable that the accumulated data at the output of the set of DPR modules is minimum (cf. Table 3.8 in previous chapter). Following this criterion, a partitioning candidate with reduced memory data transfer load can be obtained.

Clearly, if the output data rate in the chain decreases or increases from the first PE to the last PE, an optimal solution satisfying criterion (a) is not very likely to satisfy criterion (b). Hence, optimizing for both rules results in a compromise solution, and it is important to formulate an optimization approach where the DPR system designer has the choice to prefer either the one or the other criterion. Therefore, cost functions considering FPGA resources and DPR module throughput will be introduced and later merged to derive a combined partitioning metric to jointly account for (a) and (b).

4.3.1 Minimum Resource Variance Metric

In order to get a suitable metric to account for optimization criterion (a), an error metric expressing the resource utilization variance of the DPR module sets has to be defined. The amount of resources of the DPR partition is usually defined in terms of number of slices $r_{\mathrm{S,DPR}}$, lookup-tables $r_{\mathrm{L,DPR}}$, flip-flops $r_{\mathrm{F,DPR}}$ as well as BRAM $r_{\mathrm{B,DPR}}$ and DSP units $r_{\mathrm{D,DPR}}$ (cf. [BSSK09]). In relation to the available resources inside the DPR partition, let the normalized resource occupation of a processing element be reflected by

the utilization vector

$$\boldsymbol{r}_{\mathrm{p}}(p_n) = \left(\frac{r_{\mathrm{S,n}}}{r_{\mathrm{S,DPR}}}, \frac{r_{\mathrm{F,n}}}{r_{\mathrm{F,DPR}}}, \frac{r_{\mathrm{L,n}}}{r_{\mathrm{L,DPR}}}, \frac{r_{\mathrm{B,n}}}{r_{\mathrm{B,DPR}}}, \frac{r_{\mathrm{D,n}}}{r_{\mathrm{D,DPR}}} \right),$$

where the number of slices of p_n is reflected by $r_{\mathrm{S,n}}$, the number of lookup-tables by $r_{\mathrm{L,n}}$, the amount of flip-flops by $r_{\mathrm{F,n}}$ and the number of BRAMs and DSP units is expressed by $r_{\mathrm{B,n}}$ and $r_{\mathrm{D,n}}$. The notation includes a certain redundancy as flip-flops and LUTs are nested inside the slice resources. This property can subsequently be exploited to obtain a more fine-grain or more coarse-grain resource partitioning. Clearly, for all processing elements p_n, the entries in the vector $\boldsymbol{r}_{\mathrm{p}}(p_n)$ must not exceed a value of 1 since this would mean that there are not enough resources inside the reconfigurable partition to carry the logic of the respective processing element, i.e.

$$[\boldsymbol{r}_{\mathrm{p}}(p_n)]_i \leqslant 1,\ \forall\, p_n \in \boldsymbol{P},\ i \in \boldsymbol{r}_{\mathrm{p}}(p_n), \tag{4.3}$$

where $[\cdot]_i$ denotes the i-th element of a vector or an ordered set. Using the definition of the normalized resource occupation vector for a processing element, the normalized resource occupation of a reconfigurable module can be formulated by

$$\boldsymbol{r}_{\mathrm{L}}(\boldsymbol{L}_{m,n}) = \sum_{k=n}^{n+|\boldsymbol{L}_{m,n}|-1} \boldsymbol{r}_{\mathrm{p}}(p_k),$$

where $|\cdot|$ denotes the cardinality of the set. The same resource occupation limit as formulated in Equation 4.3 holds for the resources of a reconfigurable module. This means that a module is realizable if all resource elements do not exceed a value of 1, i.e.

$$[\boldsymbol{r}_{\mathrm{L}}(\boldsymbol{L}_{m,n})]_i \leqslant 1,\ \forall\, \boldsymbol{L}_{m,n} \in \boldsymbol{T},\ i \in \boldsymbol{r}_{\mathrm{L}}(\boldsymbol{L}_{m,n}).$$

In addition, the mean resource occupation of a set of DPR modules can be defined by

$$\boldsymbol{r}_{\mu} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{r}_{\mathrm{L}}(\boldsymbol{L}_{m,n}) = \frac{1}{M} \sum_{n=1}^{N} \boldsymbol{r}_{p}(p_n),$$

which is equal to the area cost definition of Moullec as defined in [Mou11]. The resource assignment with minimum-variance among the reconfigurable modules is reached if the partition sets inside $\boldsymbol{T}_k$ satisfy

$$\boldsymbol{r}_{\mathrm{L}}(\boldsymbol{L}_{m,n}) = \boldsymbol{r}_{\mu}, \forall\, \boldsymbol{L}_{m,n} \in \boldsymbol{T}_k, \tag{4.4}$$

which means that all reconfigurable modules have the same resource utilization, i.e. the resources of the DPR modules exhibit no variance and the system is optimally balanced with respect to criterion (a). Finding a set of DPR modules with minimum resource variance can be accomplished by using the root mean square error function as distance metric for the reconfigurable modules $\boldsymbol{L}_{m,n}$ inside a set $\boldsymbol{T}_k$, such that

$$\boldsymbol{\epsilon}_{\mathrm{r}}(\boldsymbol{T}_k) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\boldsymbol{r}_{\mathrm{L}}([\boldsymbol{T}_k]_i) - \boldsymbol{r}_{\mu})^2}. \tag{4.5}$$

Hence, the values of the elements in the vector $\boldsymbol{\epsilon}_r(\boldsymbol{T}_k)$ reflect the performance of the set with respect to criterion (a). Depending on the implementation of the PEs and on the amount of FPGA resources, some resources such as DSPs or BRAMSs may have higher or lower importance in the cost function $\boldsymbol{\epsilon}_{\mathrm{r}}(\boldsymbol{T}_k)$. Therefore, a weighting vector

$$\boldsymbol{\omega} = (\omega_{\mathrm{S}}, \omega_{\mathrm{F}}, \omega_{\mathrm{L}}, \omega_{\mathrm{B}}, \omega_{\mathrm{D}}),$$

is introduced, with the elements $\omega_{\mathrm{S}}, \omega_{\mathrm{F}}, \omega_{\mathrm{L}}, \omega_{\mathrm{B}}$ and ω_{D} linearly reflecting the importance of the different FPGA resources during optimization. The scalar product of the normalized weighting vector and the distance metric in Equation 4.5 give an indication for the accumulated weighted distance by

$$\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k) = \frac{\boldsymbol{\epsilon}_{\mathrm{r}}(\boldsymbol{T}_k) \cdot \boldsymbol{\omega}}{(1,1,1,1,1) \cdot \boldsymbol{\omega}}.$$

The values of $\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k)$ indicate how good or bad criterion (a) is satisfied for a weighted set of resources. A set $\boldsymbol{T}_{\hat{k}}$ can be considered optimal according to criterion (a) if $\epsilon_{\mathrm{rw}}(\boldsymbol{T}_{\hat{k}})$ is minimal compared to all other sets. Note that there might exist multiple partitioning sets with equal values for $\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k)$, i.e. for which criterion (a) is equally satisfied. Generally, if $\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k) = 0$ the set has minimum resource variance and is optimum according to criterion (a).

Next, a cost function to derive suitable partitioning candidates according to criterion (b) will be described.

4.3.2 Minimum Output Data Throughput Metric

Reducing the accumulated throughput at the output of all DPR modules minimizes the buffering overhead and the memory transfer load of the DPR system, which is a requirement formulated by optimization criterion (b). Let the normalized data throughput at the output of a processing element be defined as

$$\gamma_{\mathrm{p}}(p_n) = \frac{\gamma_n}{\gamma_{\mathrm{MEM}}},$$

where γ_n is the data throughput of the n-th PE and γ_{MEM} is the data throughput of the memory interface to the intermediate buffer as described in the previous chapter. Given that the *last* PE of a DPR module defines the amount of data that has to be transferred to the intermediate memory, a throughput function for the m-th DPR module needs to be defined, reflecting the throughput of the last element in the set $\boldsymbol{L}_{m,n}$ by

$$\gamma_{\mathrm{L}}(\boldsymbol{L}_{m,n}) = \gamma_p([\boldsymbol{L}_{m,n}]_{n+|\boldsymbol{L}_{m,n}|-1}). \tag{4.6}$$

In relation to the previous definitions, a DPR system can be considered to be realizable if Equation 4.6 satisfies the maximum throughput condition

$$\gamma_{\mathrm{L}}(\boldsymbol{L}_{m,n}) \leqslant 1, \; \forall \; \boldsymbol{L}_{m,n} \in \boldsymbol{T}_k.$$

Since the partitioning goal is minimizing the accumulated throughput inside the DPR system, it is sufficient to minimize the mean throughput of all DPR modules inside $\boldsymbol{T}_k$, defined by

$$\epsilon_{\gamma}(\boldsymbol{T}_k) = \frac{1}{M} \sum_{i=1}^{M} \gamma_{\mathrm{L}}([\boldsymbol{T}_k]_i).$$

Hence, the partitioning set $\boldsymbol{T}_k$ minimizing $\epsilon_\gamma(\boldsymbol{T}_k)$ can be referred to as optimum with respect to to criterion (b). In order to jointly consider criterion (a) and (b), i.e. minimum module throughput and minimum FPGA resource variance, a cost function will be presented in the next section.

4.3.3 Combined Throughput and Variance Minimization Metric

Since minimizing either the functions $\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k)$ or $\epsilon_\gamma(\boldsymbol{T}_k)$ produces candidates which are either optimum in the sense of (a) or (b), for joint optimization a linearly-weighted combination metric will be formulated as

$$\epsilon(\boldsymbol{T}_k, \lambda) = \lambda\epsilon_\gamma(\boldsymbol{T}_k) + (1-\lambda)\epsilon_{\mathrm{rw}}(\boldsymbol{T}_k), \tag{4.7}$$

where $\lambda < 0.5$ leads to partitioning candidates in favor of resource balancing and where $\lambda > 0.5$ leads to candidates with minimum data throughput. Hence, minimizing $\epsilon(\boldsymbol{T}_k, \lambda)$ returns a compromise between throughput minimization and resource balancing. In general, the best partitionings with respect to (a) and (b) minimize the values for $\epsilon(\boldsymbol{T}_k, \lambda)$ given the weighting constraints of the designer. Since the cost function is applied to a finite set of partitioning candidates $\boldsymbol{T}_k$, there might exist multiple partitioning solutions with the same metric value.

4.4 A Reduced-Complexity Partitioning Problem Solver

Performing the leaf comparison requires that all partitioning sets of the tree have been generated. Since the problem is of factorial complexity (cf. Equation 4.2), brute-force methods might become infeasible in case the problem is not well conditioned as subsequently explained.

For a worst-case value of $M = \lceil N/2 \rfloor$ the perpendicular foot of Pascal's triangle points to the maximum value of Equation 4.2, where $\lceil\cdot\rfloor$ denotes rounding to the nearest integer towards infinity. In this case, the number of leafs in the partitioning tree in Figure 4.1 can be calculated by

$$K_{\mathrm{MAX}} = \frac{(N-1)!}{\left(\left\lceil\frac{N}{2}\right\rfloor - 1\right)! \cdot \left(N - \left\lceil\frac{N}{2}\right\rfloor\right)!}. \tag{4.8}$$

Exhaustively calculating and comparing the joint metric values of $\epsilon(\boldsymbol{T}_k, \lambda)$ using Equation 4.7 for all K_{MAX} solutions might become infeasible since, for example, for a chain with $N = 50$ PEs and $M = 25$ desired DPR modules, there are as much as $K_{\mathrm{MAX}} \approx 6.3 \cdot 10^{13}$ possible partitioning solutions. Given the joint metric formulated the previous section, it is possible to describe a heuristic algorithm to reduce the search space and to efficiently obtain a suitable partitioning set for a signal processing chain with many PEs. Deriving this simplified approach requires the introduction of the following observations and assumptions:

I. If the data rate at the output of the PEs is monotonically increasing or decreasing with increasing PE index n, also $\epsilon_\gamma(\cdot)$ increases or decreases monotonically.

II. If $r_{\mathrm{L}}(\cdot)$ is small in relation to $\boldsymbol{r}_\mu$, the resource error $\epsilon_{\mathrm{rw}}(\cdot)$ is likely to dominate the joint error function $\epsilon(\cdot, \lambda)$.

Algorithm 1: Heuristic linear chain partitioning

```
Data: Set of PEs P
Result: A heuristic partitioning set S ∈ T_k

S = {L_{1,1} =< p_1 >}
startdepth = 1
mincost = Infinity(1,M)
for m = 1 : M do
    for n = startdepth : N do
        if numelgreat(r_L(L_{m,n}), r_μ, 2) then
            if ε(S, λ) < mincost(1,m) then
                mincost(1, m) ← ε(S, λ)
                store ← S
                if m − |S| < N − n then
                    composition(S, p_n)
                end
            else
                startdepth = n
                break
            end
        else
            if m − |S| < N − n then
                composition(S, p_n)
            end
        end
    end
    if m ⩽ M then
        extension(S, p_{startdepth−1})
    end
end
```

III. The PEs are assumed to exhibit a moderate variance in terms of FPGA resources.

From these constraints Algorithm 1 can be formulated, which traverses the binary partitioning tree from top to bottom and in each step discards the node with the higher cost. The algorithm works as follows: After the initialization of the set $\boldsymbol{S}$ with the first PE, a composition is performed (e.g. enlarging the current partition) after at least two resource elements have surpassed their mean values in $\boldsymbol{r}_\mu$. If the accumulated resources cross this threshold, the throughput error $\epsilon_\gamma(\boldsymbol{S})$ is going to dominate the error function $\epsilon(\boldsymbol{S}, \lambda)$. At this point, the costs of the partition in the current step n are being checked in accordance to assumption II. If the costs are smaller than the minimal costs seen so far, the *mincost* vector is updated and the current configuration is stored before proceeding to the next step $n+1$. In case the actual costs exceed the minimal costs, the break condition ends the loop and an extension operation with the last PE is performed, thus creating a new partition, since the previous configuration was the best seen so far. In conclusion, the two optimization goals stated in Section 4.3 determine the branch selection in the algorithm:

- *Extension* is likely to be applied, if the accumulated resources of the current partition is equal or greater than the ideal one, cf. goal (a).
- *Composition* is likely to be applied, if the accumulated throughput can be reduced, cf. goal (b).

The branch selection can be modified by adjusting λ, i.e. for $\lambda = 1$, the algorithm solely optimizes for memory throughput (favor compositions) and for $\lambda = 0$ the algorithm solely optimizes for weighted resources (favor extensions). Since the output data rate of

the PEs in the signal processing chain is likely to decrease with increasing n, the elements in the *mincost* vector increase monotonically, cf. observation I. The algorithm terminates after all PEs have been assigned, with the partitioning candidate being stored in the leaf. Next, the suitability of the metrics derived in the last section and the performance of the heuristic algorithm will be evaluated by partitioning the DAB receiver chain as presented in the previous chapter.

4.5 DAB Decoder Chain Partitioning

In this section the proposed metrics will be used to find a weighted partitioning candidate for the DAB receiver implementation as presented in Chapter 3. The receiver comprises of $N = 12$ PEs and has been designed for a Xilinx Spartan-6 XC6SLX25 FPGA. The feedback path from the frequency offset estimator to the correction unit is resolved in the implementation by using a DAB frame-based processing as previously outlined. The number of DPR modules have been defined to be $M = 3$ and in order to favor slice and BRAM resources over FFs and LUTs, the resource weighting vector has been defined to be $\boldsymbol{w} = (10, 1, 1, 5, 1)$. Since the memory throughput of the DAB receiver PEs is low compared to the throughput of the FPGA memory interface, the memory throughput is preferred over the resource variance reduction by setting $\lambda = 0.9$.

n	p_n	Slices	FFs	LUTs	BRAMs	DSP48	$\frac{\gamma_n}{\text{Mbit/s}}$
1	Freq. Sync.	127	306	344	0	4	32.768
2	AGC	99	195	148	0	3	32.768
3	Time E.+Sync.	64	86	85	0	0	32.325
4	Freq. Est	291	581	811	2	5	32.325
5	Guard Rem.	7	10	9	0	0	25.941
DPR module m_1		**588**	**1178**	**1397**	**2**	**12**	**25.941**
6	FFT	358	1024	625	7	8	25.941
7	DQPSK	37	76	49	2	6	25.941
8	Freq. Deint.	49	29	50	3	0	9.1
9	Bitcut	38	31	46	0	0	4.533
DPR module m_2		**482**	**1160**	**770**	**12**	**14**	**4.533**
10	Time Deint.	52	63	71	3	0	4.533
11	Viterbi	713	1946	1415	3	0	0.416
12	Post Process	14	17	4	0	0	0.416
DPR module m_3		**779**	**2026**	**1490**	**6**	**0**	**0.416**

Table 4.1: Resources and output data rates of the receiver PEs and accumulated resources of the balanced DPR modules.

Given these optimization parameters, the partitioning solution selected by the heuristic algorithm is identical to the optimum candidate found by exhaustive search. This is due to the fact that the parameters of the DAB receiver PEs are well conditioned given the optimization constraints in Section 4.4. Since the algorithm has no memory, it might only find a local optimum and optimizing the algorithm for different processing element chains can be regarded as future work.

The resource consumption and the output data rate of the different PEs are listed in Table 4.1 as well as the accumulated resources of the three DPR modules of the optimum partitioning candidate. The available resources of the Spartan FPGA are included in the last row of the table. Since from the numbers it is hard to determine in how far the resources are balanced in relation to the average number of resources, Figure 4.2 illustrates the accumulated resource consumption of the DPR modules by five stacked graphs for each of the three DPR modules, including a plot of the PE output throughput. The mean value $\boldsymbol{r}_\mu$ of the different resource elements is indicated by a blue horizontal line. The closer the top of the resource stacks approaches this line, the more balanced the partitioning solution is according to criterion (a). It can be observed from the graph that the slice and BRAM allocation per module are balanced according to the weighting vector $\boldsymbol{w}$.

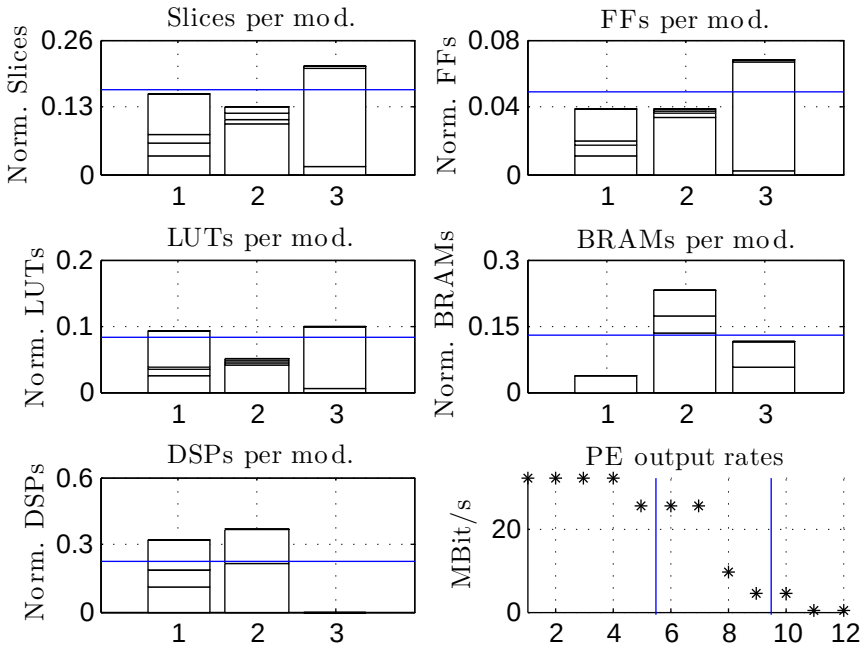


Figure 4.2: Weighted resource partitioning of PEs favoring slices and BRAMs.

In the throughput graph, the output data rates are monotonically decreasing from the first PE to the last PE in the chain, which reinforces the assumptions made in Section 4.4. The blue vertical lines reflect the intercept point of the modules in the chain, with their respective output rate γ_L. Recall that if $\lambda = 0$ the data throughput per module is neglected and only the FPGA resources are considered in the partitioning problem. In this case the guard interval removal stage is allocated to the second module m_2 instead of m_1 causing the memory load to increase by approx. 20 % (cf. Table 4.1) at the output of DPR module m_1. Although this partitioning solution is marginally better in terms of resources, the difference in data rate is significant.

4.5.1 Weighting of Single Resource Elements

In this section it will be evaluated how the resource allocation looks like if only one resource element can be considered in the DAB receiver partitioning process. In this

case, it can be shown that the formulated requirement to obtain a weighted partitioning solution for slices *and* BRAMs can not be met. For $\boldsymbol{w} = (1, 0, 0, 0, 0)$ an optimized partitioning for slices only has been generated. To jointly optimize for throughput, lambda is again set to $\lambda = 0.9$. For this configuration, the partitioning set depicted in Figure 4.3 is obtained.

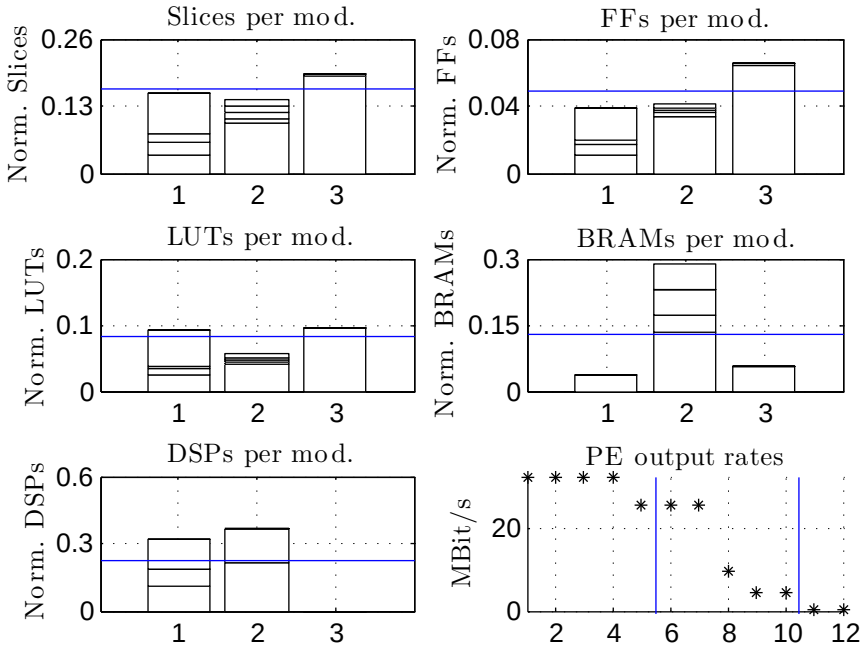


Figure 4.3: Non-weighted resource partitioning of PEs favoring slices only.

Regarding an uniform distribution of slices among the DPR modules, the partitioning solution in Figure 4.3 shows to be slightly superior to the weighted partitioning in Figure 4.2. However, the slices-only optimization comes at the downside of a peak in BRAM occupation in DPR module m_2. Thus, this implementation would require 20 % more BRAM resources in the reconfigurable partition as compared to the weighted partition solution, which in comparison requires only 7.2 % more slices. By looking at the graphs it can be observed that the resource distribution is worse than in Figure 4.2 but still acceptable, i.e. for the DAB receiver example optimizing for slices only would give a reasonably good partitioning candidate.

For a further evaluation, the weighting vector is defined to be $\boldsymbol{w} = (0, 0, 0, 1, 0)$ to generate a partitioning set with a minimum variance in BRAM occupation. The results of this design choice are outlined in Figure 4.4. The graphs show that the amount of required BRAMs can be reduced by 25 %, while in turn the utilization of all other resources increased, resulting in a strongly imbalanced resource partitioning. The outlined examples show that without proper weighting, the resulting partitioning might turn out to be imbalanced in terms of resources and that using a weighting vector with multi-resource preference is likely to result in a better partitioning candidate selection.

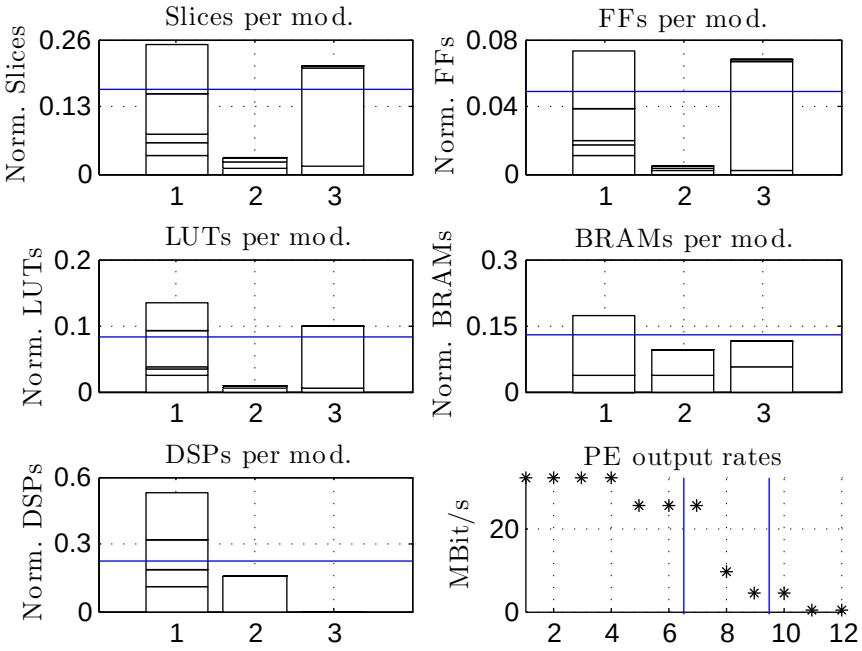


Figure 4.4: Non-weighted resource partitioning of PEs favoring BRAMs only.

4.6 Summary

The problem of partitioning a sequential chain of processing elements into reconfigurable modules has been described and a suitable metric to evaluate the quality of the distribution of the resources among a set of DPR modules has been presented. Together with an adjustable resource weighting vector, which allows the designer to prefer certain resource elements in the metric calculation, the presented evaluation approach jointly allows for finding a partitioning candidate with minimal memory throughput. Using the metrics derived, a heuristic algorithm with linear complexity has been proposed in order to quickly obtain a partitioning solution. The DAB receiver partitioning case study has shown that the presented approach has advantages when a weighted set of FPGA resources is considered in the partitioning process. Compared to net-list-based approaches, the given approach is applicable for high-level analysis early in the DPR system design phase, given an a priori knowledge of the resources or even PE resource estimates.

The major findings of this thesis are concluded in the next chapter together with an outlook on future prospects on dynamic partial reconfiguration of FPGAs for digital broadcasting receivers.

5 Conclusion and Outlook

Analyzing the benefits, limitations and possibilities of hardware resource multiplexing for digital receiver chain implementations using dynamic partial reconfiguration of FPGAs constitutes the major focus of this work. In this context, the feasibility of adaptive reconfiguration of broadcast receivers using DPR of Xilinx FPGAs has been evaluated by means of an adaptive FM receiver system implementation, where the adaptation routines have been included in the static system to autonomously trigger a reconfiguration of the dynamic partition. Routines and metrics for signal quality estimation have been elaborated and evaluated for the use inside this adaptive system. It could be shown that the noise power within the spectral gaps of the FM MPX signal can be evaluated for signal quality estimation with low computational complexity. Given the output of the estimation routine, switching thresholds for the FM multiplex component decoders have been defined in relation to the FM receiver demodulation algorithms. Analyzing the DPR system components revealed that in certain scenarios a resource reallocation can reduce the resource consumption of an adaptive FPGA system. The achievable resource gain increases if the DPR modules exhibit a high variance in resource utilization. Freeing and reallocating resources to other reconfigurable partitions is possible using hierarchical partial reconfiguration. The analysis conducted in this work revealed that hierarchical reconfiguration is promising for implementing resource-economic receiver systems on FPGAs. However, it could be shown that even with differential reconfiguration hierarchical DPR systems can not be realized reliably with existing vendor tools. Hierarchical reconfiguration using nested area groups has turned out to be not supported either. However, third party tools exist that leverage hierarchical DPR and make it possible to exploit the DPR possibilities of the Xilinx FPGA fabric.

The sequential execution of receiver modules using cyclic DPR has been discussed and proposed as a feasible method to weigh processing time against FPGA resources. The introduction of a cyclic reconfiguration system model enabled to quantify the trade-offs between area consumption, execution time and context handling of the DPR modules. Compromising between DPR system latency and processing frame duration is possible, if the reconfiguration time can be made arbitrary small in comparison to the DPR module execution time. In case the reconfiguration interface is slow, longer frames need to be processed, increasing the system latency and buffer memory requirements. A feasibility study furthermore outlined the correlation between framing duration and context handling for a DAB receiver. In order to minimize context write and read operations, frames of longer duration in relation to the transmission framing of the DAB system turned out to be preferable. Using AXI FIFOs for processing element communication simplified the partitioning of the DAB receiver chain processing elements since the AXI protocol implicitly handles the control-flow among the elements. This allowed using the same processing elements inside the DPR modules without major modifications. The implementation of the cyclic DPR receiver system on a Spartan-6 FPGA outperformed the static implementation in terms of resource usage at the cost of an increased process-

ing element clock frequency. However, although reported in literature, DPR could not be accomplished reliably with the vendor DPR tool flow since the routing of the design can not be constraint. It is recommended to use third party tools to reliably use DPR on the Spartan-6 FPGA platform.

Evaluating the feasibility of cyclic DPR receiver systems has shown to be possible even without full implementation knowledge, as presented by means of a case study for a DVB-T2 receiver. This allows to gain insights into the feasibility of cyclic DPR systems early in the design phase of a certain implementation. Partitioning the receiver processing chain into a set of DPR modules requires to account for the properties of the cyclic DPR system. It has been derived that it is desirable to use a set of DPR modules with minimum output data throughput and minimum variance in resource utilization. For the DAB receiver system it could be concluded that a weighting of resource elements, such as slices, DSP units and BRAMs, is necessary to obtain suitable partitioning candidates. Although solving the partitioning problem is of factorial time complexity, the analysis revealed that for a chain of receiver processing elements with monotonically decreasing output throughput and low variance in resource utilization a heuristic partitioning algorithm with linear time complexity is feasible.

Recapitulating, the results of the preceding chapters have shown that dynamic partial self-reconfiguration provides additional degrees of freedom when optimizing existing FPGA-based digital broadcasting receiver systems in terms of resource utilization. Taking advantage of cyclic DPR by partitioning existing receiver implementations into reconfigurable modules can potentially lead to a reduction in FPGA resources. Reducing the resource requirements means that smaller FPGAs might provide a sufficient amount of resources for the task in question, thus reducing the system cost. Severe limitations arise from the DPR vendor tool support, imposing design constraints to DPR floorplanning tasks such as signal routing and area allocation. With the development of new and more versatile tools, future work could potentially include an analysis on how hierarchical configuration can be used on a fine grain level to share resources between multiple small DPR partitions. Refining the insights of SNR-adaptive receiver systems and investigating in how far DPR can be used in bi-directional communication systems with timing restrictions is considered as another major field of research. Analyzing the trade-off between the number of DPR modules and resource occupation for cyclic DPR systems with a larger number of processing elements is a subject that also needs further investigation.

Bibliography

[AK00] P. Andersson and K. Kuchcinski. Performance Oriented Partitioning for Time-multiplexed FPGAs. In *Proceedings of the 26th Euromicro Conference*, volume 1, pages 60 –66 vol.1, 2000. doi:10.1109/EURMIC.2000.874616.

[BBHN04] B. Blodget, C. Bobda, M. Huebner, and A. Niyonkuru. Partial and Dynamically Reconfiguration of Xilinx Virtex-II FPGAs. In Jürgen Becker, Marco Platzner, and Serge Vernalde, editors, *Field Programmable Logic and Application*, volume 3203 of *Lecture Notes in Computer Science*, pages 801–810. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30117-2_81.

[BKT11] C. Beckhoff, D. Koch, and J. Torresen. Migrating Static Systems to Partially Reconfigurable Systems on Spartan-6 FPGAs. In *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pages 212–219, May 2011. doi:10.1109/IPDPS.2011.144.

[BLC09] Tobias Becker, Wayne Luk, and Peter Y. K. Cheung. Parametric Design for Reconfigurable Software-Defined Radio. In *Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications*, pages 15–26, March 2009. doi:10.1007/978-3-642-00641-8_5.

[Boa92] B. Boashash. Estimating and Interpreting the Instantaneous Frequency of a Signal. II. Algorithms and Applications. *Proceedings of the IEEE*, 80(4):540–568, April 1992. doi:10.1109/5.135378.

[BSSK09] P. Banerjee, M. Sangtani, and S. Sur-Kolay. Floorplanning for Partial Reconfiguration in FPGAs. In *22nd International Conference on VLSI Design*, pages 125–130, January 2009. doi:10.1109/VLSI.Design.2009.36.

[BWF+13] C. Beckhoff, A. Wold, A. Fritzell, D. Koch, and J. Torresen. Building partial systems with GoAhead. In *23rd International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–1, September 2013. doi:10.1109/FPL.2013.6645634.

[BY08] S. Bayar and A. Yurdakul. Self-reconfiguration on Spartan-III FPGAs with compressed partial bitstreams via a parallel configuration access port (cPCAP) core. In *Research in Microelectronics and Electronics, 2008. PRIME 2008. Ph.D.*, pages 137–140, June 2008. doi:10.1109/RME.2008.4595744.

[BYT11] S. Bayar, A. Yurdakul, and M. Tukel. A Self-Reconfigurable Platform for General Purpose Image Processing Systems on Low-Cost Spartan-6 FPGAs. In *6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–9, June 2011. doi:10.1109/ReCoSoC.2011.5981513.

[CA11] Xiaoheng Chen and Venkatesh Akella. Exploiting Data Level Parallelism For Energy Efficient Implementation of LDPC Decoders and DCT on a FPGA. *ACM Trans. Reconfigurable Technol. Syst.*, 4(4):37:1–37:17, December 2011. doi:10.1145/2068716.2068723.

[Car22] John R. Carson. Notes on the Theory of Modulation. *Proceedings of the Institute of Radio Engineers*, 10(1):57–64, February 1922. doi:10.1109/JRPROC.1922.219793.

[CKPLM10] Sri Hanuma Chitti, Gaurav Kulkarni, Andreas Popp, and Yannick Le Moullec. Flexible and Reconfigurable Implementation of Link Adaptation Algorithms. *Wireless Personal Communications*, 54(1):83–93, 2010. doi:10.1007/s11277-009-9712-5.

[Cla11] Christopher Claus. *Zum Einsatz dynamisch rekonfigurierbarer eingebetteter Systeme in der Bildverarbeitung.* Dissertation, Technische Universität München, München, 2011. (Accessed May 11 2016). URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20110218-1002365-1-2.

[CMS99] D Chang and M Marek-Sadowska. Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs. *IEEE Transactions on Computers*, 48(6):565 –578, June 1999. doi:10.1109/12.773794.

[Col11] *Collins English Dictionary.* Collins UK, 11th edition, 2011.

[Cyp] Cypress Semiconductor Corporation. *Datasheet for EZ-USB FX2LP USB Microcontroller High-Speed USB Peripheral Controller.*

[CZMS07] C. Claus, J. Zeppenfeld, F. Müller, and W. Stechele. Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System. In *Design, Automation Test in Europe (DATE) Conference and Exhibition*, pages 1 –6, April 2007. doi:10.1109/DATE.2007.364642.

[DDHSW01] N. Da Dait, M. Harteneck, C. Sandner, and A. Wiesbauer. Numerical modeling of PLL jitter and the impact of its non-white spectrum on the SNR of sampled signals. In *Southwest Symposium on Mixed-Signal Design (SSMSD)*, pages 38–44, 2001. doi:10.1109/SSMSD.2001.914934.

[DeH96] André DeHon. DPGA Utilization and Application. In *Proceedings of the 1996 ACM Fourth International Symposium on Field-programmable Gate Arrays*, FPGA '96, pages 115–121, New York, NY, USA, 1996. ACM. doi:10.1145/228370.228387.

[DGRB04] Jean-Philippe Delahaye, Guy Gogniat, Christian Roland, and Pierre Bomel. Software Radio and Dynamic Reconfiguration on a DSP/FPGA platform. *Frequenz Journal*, 58(5-6):152–159, 2004.

[DLU91] Y.F. Dehery, M. Lever, and P. Urcun. A MUSICAM source codec for digital audio broadcasting and storage. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3605–3608 vol.5, April 1991. doi:10.1109/ICASSP.1991.151054.

[DPML07] J.-P. Delahaye, J. Palicot, C. Moy, and P. Leray. Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform. In *16th IST Mobile and Wireless Communications Summit*, pages 1 –5, July 2007. doi:10.1109/ISTMWC.2007.4299250.

[Dvb] History of Digital Video Broadcasting (DVB). (Accessed May 11 2016). URL: https://www.dvb.org/about/history.

[EH94] J.G. Eldredge and B.L. Hutchings. RRANN: The Run-time Reconfiguration Artificial Neural Network. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 77–80, May 1994. doi:10.1109/CICC.1994.379763.

[ESK03] D. Eilers, H. Steckenbiller, and R. Knorr. Architecture Template with Dynamic Buffering for Runtime Reconfiguration of Adaptive Embedded Communication Systems. In *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT)*, pages 383–386, December 2003. doi:10.1109/FPT.2003.1275782.

[ESL04] M. Eroz, Feng W. Sun, and Lin N. Lee. DVB-S2 low-density-parity-check-codes with near Shannon limit performance. *International Journal of Satellite Communications and Networking*, 22(3), May 2004. doi:10.1002/sat.787.

[ets06] EN 400 401: Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers v1.4.1. European Standard, European Telecommunications Standards Institute (ETSI), January 2006. (Accessed May 11 2016). URL: http://www.etsi.org/deliver/etsi_en/300400_300499/300401/01.04.01_40/en_300401v010401o.pdf.

[Ets08] EN 302 755: Digital Video Broadcasting (DVB) Second generation framing structure (DVB-T2) v1.1.1. European Standard, European Telecommunications Standards Institute (ETSI), August 2008. (Accessed May 11 2016). URL: http://www.etsi.org/deliver/etsi_en/302300_302399/302307/01.02.01_60/en_302307v010201p.pdf.

[FIIS12] M. Feilen, A. Iliopoulos, M. Ihmig, and W. Stechele. Partitioning and Context Switching for a Reconfigurable FPGA-based DAB Receiver. In *Conference on Design & Architectures for Signal & Image Processing (DASIP)*, pages 22 –28, October 2012.

[FISS12] M. Feilen, M. Ihmig, C. Schwarzbauer, and W. Stechele. Efficient DVB-T2 decoding accelerator design by time-multiplexing FPGA resources. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 75–82, August 2012. doi:10.1109/FPL.2012.6339244.

[FIVS13] M. Feilen, A. Iliopoulos, M. Vonbun, and W. Stechele. Weighted partitioning of sequential processing chains for dynamically reconfigurable FPGAS. In *23rd International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, September 2013. doi:10.1109/FPL.2013.6645521.

[FIZS11] M. Feilen, M. Ihmig, A. Zahlheimer, and W. Stechele. Real-time signal processing on low-cost-FPGAs using dynamic partial reconfiguration. In *13th International Symposium on Integrated Circuits (ISIC)*, pages 110–113, December 2011. doi:10.1109/ISICir.2011.6131921.

[GK89] J. P. Gray and T. A. Kean. Configurable Hardware: A New Paradigm for Computation. In *Proceedings of the Decennial Caltech Conference on Advanced Research in VLSI*, pages 279–295, Cambridge, MA, USA, 1989. MIT Press. URL: http://dl.acm.org/citation.cfm?id=90897.90945.

[GMBV14] P. Gupta, S. Murali, J. Balakrishnan, and S. Vishwakarma. Signal quality estimation and control, November 27 2014. US Patent App. 13/899,868.

[Gna12] Markus Gnadl. A Digital Audio Broadcasting (DAB) Receiver on Low-Cost Spartan-6 FPGAs. Bachelor's Thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, February 2012.

[Gra04] T. Grant. *International Directory of Company Histories*. Number Bd. 64 in Gale virtual reference library. St. James Press, 2004.

[HP11] John C. Hoffman and Marios S. Pattichis. A High-Speed Dynamic Partial Reconfiguration Controller Using Direct Memory Access Through a Multiport Memory Controller and Overclocking with Active Feedback. *International Journal of Reconfigurable Computing*, 2011:10, 2011. doi:10.1155/2011/439072.

[IAH08] M. Ihmig, N. Alt, and A. Herkersdorf. Resource-efficient Sequential Architecture for FPGA-based DAB Receiver. In *Proceedings of the 5th Karlsruhe Workshop on Software Radios*, pages 101–107, March 2008.

[IAH10] M. Ihmig, N. Alt, and A. Herkersdorf. Implementation and fine-grain partitioning of a DAB SDR receiver on an FPGA-DSP platform. In *Proceedings of the 6th Karlsruhe Workshop on Software Radios*, March 2010.

[IFH12] M. Ihmig, M. Feilen, and A. Herkersdorf. On the Accuracy of sum-based Logic and Power Estimates in Hardware-accelerated SDR Systems. In *Proceedings of the 6th Karlsruhe Workshop on Software Radios*, March 2012.

[Ili12] Andreas Iliopoulos. Sequential Execution of DAB Receiver Modules using DPR on Spartan-6 FPGAs. Master's thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, September 2012.

[Itu98] ITU-R BS.412-9: Planning standards for terrestrial FM sound broadcasting at VHF. Recommendation, International Telecommunication Union, December 1998. (Accessed May 11 2016). URL: https://www.itu.int/rec/R-REC-BS.412/en.

[Itu01a] ITU-R BS.1194-2: System for multiplexing frequency modulation (FM) sound broadcasts with a sub-carrier data channel having a relatively large transmission capacity for stationary and mobile reception. Technical report, International Telecommunication Union, June 2001. (Accessed May 11 2016). URL: https://www.itu.int/rec/R-REC-BS.1194-2-199812-I/en.

[Itu01b] ITU-R BS.450-3: Transmission standards for FM sound broadcasting at VHF. Technical report, International Telecommunication Union, November 2001. (Accessed May 11 2016). URL: https://www.itu.int/rec/R-REC-BS.450/en.

[itu15] ITU-R V.431-8: Nomenclature of the frequency and wavelength bands used in telecommunications. Recommendation, International Telecommunication Union, June 2015. (Accessed May 11 2016). URL: https://www.itu.int/rec/R-REC-V.431/.

[Joh11] Jeff Johnson. List and comparison of FPGA companies, 2011. (Accessed May 11 2016). URL: http://www.fpgadeveloper.com/2011/07/list-and-comparison-of-fpga-companies.html.

[JTHT10] K. Jozwik, H. Tomiyama, S. Honda, and H. Takada. A Novel Mechanism for Effective Hardware Task Preemption in Dynamically Reconfigurable Systems. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 352 –355, September 2010. doi:10.1109/FPL.2010.76.

[Kam08] K.-D. Kammeyer. *Nachrichtenübertragung*. B.G. Teubner, Reihe Informationstechnik, Stuttgart, Deutschland, 4th edition, March 2008.

[KB14] D. Koch and C. Beckhoff. Hierarchical reconfiguration of FPGAs. In *24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, September 2014. doi:10.1109/FPL.2014.6927491.

[KBT08] D. Koch, C. Beckhoff, and J. Teich. ReCoBus-Builder - A Novel Tool and Technique to Build Statically and Dynamically Reconfigurable Systems for FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 119–124, September 2008. doi:10.1109/FPL.2008.4629918.

[KBT10] D. Koch, C. Beckhoff, and J. Tørrison. Advanced partial run-time reconfiguration on Spartan-6 FPGAs. In *International Conference on Field-Programmable Technology (FPT)*, pages 361–364, December 2010. doi:10.1109/FPT.2010.5681426.

[KDHS14] A. Kulkarni, T. Davidson, K. Heyse, and D. Stroobandt. Improving Reconfiguration Speed for Dynamic Circuit Specialization using Placement Constraints. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–6, December 2014. doi:10.1109/ReConFig.2014.7032534.

[KM99] D. Kopitz and B. Marks. *RDS: The Radio Data System*. Artech House mobile communications library. Artech House, 1999.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[KTB+12] Dirk Koch, Jim Torresen, Christian Beckhoff, Daniel Ziener, Christopher Dennl, Volker Breuer, Jürgen Teich, Michael Feilen, and Walter Stechele. Partial Reconfiguration on FPGAs in Practice - Tools and Applications. In *ARCS Workshops*, pages 1–12, February 2012.

[KTHL07] C. C. Kao, T. C. Tai, Y. Y. Hwang, and Y. T. Lai. A Sequential Circuit Partitioning Algorithm for Dynamically Reconfigurable FPGAs. In *International Conference on Communications, Circuits and Systems (ICCCAS)*, pages 1185–1188, July 2007. doi:10.1109/ICCCAS.2007.4348258.

[KTR08] Ian Kuon, Russell Tessier, and Jonathan Rose. FPGA Architecture: Survey and Challenges. *Found. Trends Electron. Des. Autom.*, 2(2):135–253, February 2008. URL: http://dx.doi.org/10.1561/1000000005, doi:10.1561/1000000005.

[KVW+10] C. Kocks, A. Viessmann, A. Waadt, C. Spiegel, A. Burnic, G.H. Bruck, P. Jung, Jaeyoel Kim, YeonJu Lim, and Hyeon Woo Lee. A DVB-T2 receiver realization based on a software-defined radio concept. In *4th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 1 –4, March 2010. doi:10.1109/ISCCSP.2010.5463488.

[LBM+06] Patrick Lysaght, Brandon Blodget, Jeff Mason, Jay Young, and Brendan Bridgford. Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs. In *International Conference on Field-programmable Logic and Applications (FPL)*, pages 1–6. IEEE, 2006. doi:10.1109/FPL.2006.311188.

[LD94a] Patrick Lysaght and Hugh Dick. Implementation of Adaptive Signal Processing Architectures Based On Dynamically Reconfigurable FPGAs. In *Proceedings of European Association for Signal Processing (EUSIPCO)*, pages 1871–1874, 1994.

[LD94b] Patrick Lysaght and John Dunlop. Dynamic Reconfiguration of FPGAs. In *Selected Papers from the Oxford 1993 International Workshop on Field Programmable Logic and Applications on More FPGAs*, pages 82–94, Oxford, UK, UK, 1994. Abingdon EE&CS Books. URL: `http://dl.acm.org/citation.cfm?id=188565.188605`.

[LD09] V. Lai and O. Diessel. ICAP-I: A reusable interface for the internal reconfiguration of Xilinx FPGAs. In *International Conference on Field-Programmable Technology (FPT)*, pages 357 –360, December 2009. `doi:10.1109/FPT.2009.5377616`.

[LFDN09] Jörg Lotze, Suhaib A Fahmy, LE Doyle, and J Noguera. An FPGA-based Autonomous Adaptive Radio. *ACM SIGCOMM Conference*, 2009. (Accessed May 11 2016). URL: `http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final57.pdf`.

[LFHLC89] B. Le Floch, R. Halbert-Lassalle, and D. Castelain. Digital sound broadcasting to mobile receivers. *IEEE Transactions on Consumer Electronics*, 35(3):493–503, August 1989. `doi:10.1109/30.44309`.

[LKLJ09] Ming Liu, W. Kuehn, Zhonghai Lu, and A. Jantsch. Run-time Partial Reconfiguration speed investigation and architectural design space exploration. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 498–502, August 2009. `doi:10.1109/FPL.2009.5272463`.

[LNJ+11] Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard. A shuffled iterative bit-interleaved coded modulation receiver for the DVB-T2 standard: Design, implementation and FPGA prototyping. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pages 55 –60, October 2011. `doi:10.1109/SiPS.2011.6088949`.

[MF10] Walter Stechele Michael Feilen, Matthias Ihmig. Concept and Design of an SNR-adaptive DRM+/FM Receiver using Dynamic Partial Reconfiguration (DPR) of FPGAs. In *11th Workshop Digital Broadcasting*, Erlangen, Germany, September 2010.

[MMT+08] P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati, V. La Barba, P. Cuvelier, B. Rousseau, and P. Gelineau. An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications. *EURASIP Journal on Embedded Systems*, 2008:1–11, 2008. `doi:http://dx.doi.org/10.1155/2008/367860`.

[MNH+11a] J. Meyer, J. Noguera, M. Hübner, L. Braun, O. Sander, R.M. Gil, R. Stewart, and J. Becker. Fast Start-up for Spartan-6 FPGAs using Dynamic Partial Reconfiguration. In *Design, Automation and Test in Europe (DATE) Conference*, pages 1–6, March 2011. `doi:10.1109/DATE.2011.5763244`.

[MNH+11b] J. Meyer, J. Noguera, M. Hübner, L. Braun, O. Sander, R.M. Gil, R. Stewart, and J. Becker. Fast Start-up for Spartan-6 FPGAs using Dynamic Partial Reconfiguration. In *Design, Automation Test in Europe (DATE) Conference and Exhibition*, pages 1 –6, March 2011.

[Mou11] Y. Le Moullec. A First Step Towards High-Level Cost Models for the Implementation of SDRs on Multiprocessing Reconfigurable Systems. In *14th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–5, October 2011.

[Mü11] Daniel Münch. Receive signal dependent adaption of an FPGA-based software-defined radio receiver system. Master's thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, September 2011.

[PB00] D.R. Pauluzzi and N.C. Beaulieu. A comparison of SNR estimation techniques for the AWGN channel. *IEEE Transactions on Communications*, 48(10):1681–1691, October 2000. doi:10.1109/26.871393.

[PLMK09] A. Popp, Y. Le Moullec, and P. Koch. Fast Feasibility Estimation of Reconfigurable Architectures. In *4th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 117 –122, May 2009. doi:10.1109/ICIEA.2009.5138181.

[PM06] John G. Proakis and Dimitris K. Manolakis. *Digital Signal Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 4th edition, 2006.

[Pro01] John G. Proakis. *Digital Communications*. Electrical Engineering Series. McGraw-Hill, 2001.

[PS95] R. Andersson P. Scomazzon. A high bit-rate data broadcasting system using the terrestrial FM radio network. Technical review, European Broadcasting Union (EBU), May 1995. (Accessed May 11 2016). URL: https://tech.ebu.ch/docs/techreview/trev_264-scomazzon.pdf.

[Rds] History of the Radio Data System (RDS). (Accessed May 11 2016). URL: http://www.rds.org.uk/2010/RDS-History.htm.

[Rhe13] Sven Rheindt. Dynamisch Partielle Rekonfiguration auf Altera 28 nm FPGAs. Bachelor's Thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, July 2013.

[Ric63] S.O. Rice. Noise in FM Receivers. In New York M. Rosenblatt(ed.) Wiley, editor, *Symposium of Time Series Analysis Proceedings*, 1963.

[Ros89] Werner Rosenkranz. Digitale Systeme und optimierte Algorithmen zum Empfang frequenzmodulierter Signale. Habilitationsschrift, Universität Erlangen-Nürnberg, 1989.

[RPN09] M. Rice, M. Padilla, and B. Nelson. On FM Demodulators in Software Defined Radios Using FPGAs. In *IEEE Military Communications Conference (MILCOM)*, pages 1–7, October 2009. doi:10.1109/MILCOM.2009.5379759.

[Sch11] Philipp Schmidbauer. Rekonfiguration von Spartan 6 FPGAs. Bachelor's thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, September 2011.

[SFFM99] M. Speth, S.A. Fechtel, G. Fock, and H. Meyr. Optimum Receiver Design for Wireless Broad-Band Systems Using OFDM. *IEEE Transactions on Communications*, 47(11):1668–1677, November 1999. doi:10.1109/26.803501.

[SFHB12] Nimish Sane, John Ford, Andrew I. Harris, and Shuvra S. Bhattacharyya. Prototyping scalable digital signal processing systems for radio astronomy using dataflow models. *Radio Science Journal*, 2012. arXiv:arXiv/1204.4696, doi:10.1029/2011RS004924.

[SFS12] L. Stolz, M. Feilen, and W. Stechele. An Optimized Software-defined Digital Audio Broadcasting (DAB) Receiver for x86 Platforms. In *7th Karlsruhe Workshop on Software Radios (WSR)*, Karlsruhe, Germany, March 2012.

[Smi08] Julius O. Smith. *Introduction to Digital Filters: With Audio Applications*. Music signal processing series. W3K, October 2008.

[SS08] Felix Schad and Andreas Steil. Laboruntersuchung über Versorgungskriterien für eine UKW-FM Monoabstrahlung. Technical Report, Fachhochschule Kaiserslautern im Auftrag der Landeszentrale für Medien und Kommunikation Rheinland-Pfalz (LMK), September 2008.

[Str10] Stefan Strasser. Entwicklung eines FM-Empfängers für die Xilinx Spartan-3A FPGA-Plattform. Master's thesis, Technische Universität München (TUM), Lehrstuhl für Integrierte Systeme, September 2010.

[TCEB95] Edward Tau, Derrick Chen, Ian Eslick, and Jeremy Brown. A First Generation DPGA Implementation. In *In Proceedings of the Third Canadian Workshop on Field-Programmable Devices*, pages 138–143, 1995.

[TCJW97] S Trimberger, D Carberry, A Johnson, and J Wong. A Time-multiplexed FPGA. In *Proceedings of the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 22 –28, April 1997. doi:10.1109/FPGA.1997.624601.

[Trp91] Z. Trpovski. Reliability testing method for RDS based on the PI code statistics. *IEEE Transactions on Consumer Electronics*, 37(4):884–891, November 1991. doi:10.1109/30.106954.

[VJS95] J. Villasenor, C. Jones, and B. Schoner. Video Communications Using Rapidly Reconfigurable Hardware. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):565–567, Dec 1995. doi:10.1109/76.475899.

[Wau91] T.C. Waugh. Field programmable gate array key to reconfigurable array outperforming supercomputers. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 6.6/1–6.6/4, May 1991. doi:10.1109/CICC.1991.164051.

[WH95] M.J. Wirthlin and B.L. Hutchings. A Dynamic Instruction Set Computer. In *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pages 99–107, April 1995. doi:10.1109/FPGA.1995.477415.

[WH97] M. J. Wirthlin and B. L. Hutchings. Improving Functional Density Through Run-Time Constant Propagation. In *In ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 86–92, 1997.

[WM06] M. Werner and O. Mildenberger. *Nachrichten-Übertragungstechnik: Analoge und digitale Verfahren mit modernen Anwendungen.* Studium Technik. Vieweg+Teubner Verlag, 2006.

[XGXZCY13] Chun Xian Gao, Yong Xiu Zhang, En Cheng, and Fei Yuan. Investigation of SNR Estimation Algorithms of FM Signal for the Underwater Acoustic Channel. *Journal of Computers*, 8(8):2042–2050, August 2013. doi:10.4304/jcp.8.8.2042-2050.

[Xil] Inc. Xilinx. DVB-C2 LDPC/BCH Decoder IP Core. (Accessed May 11 2016). URL: http://www.xilinx.com/products/intellectual-property/1-411yls.html.

[Xil07] Difference-Based Partial Reconfiguration v2.0. XAPP290, Xilinx, Inc., December 2007. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf.

[Xil08] Correcting Single-Event Upsets in Virtex-4 Platform FPGA Configuration Memory v1.0. XAPP988, Xilinx, Inc., March 2008. (Accessed May 11 2016). URL: http://application-notes.digchip.com/077/77-43209.pdf.

[Xil09a] Power Consumption at 40 and 45 nm v1.0. WP298, Xilinx, Inc., April 2009. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/white_papers/wp298.pdf.

[Xil09b] Virtex-4 FPGA Configuration User Guide v1.11. UG071, Xilinx, Inc., June 2009. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug071.pdf.

[Xil10] Spartan-6 FPGA Memory Controller v2.3. UG388, Xilinx, Inc., August 2010. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug388.pdf.

[Xil11a] LogiCORE IP XPS HWICAP v5.01a. DS586, Xilinx, Inc., June 2011. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap/v5_01_a/xps_hwicap.pdf.

[Xil11b] ML505/ML506/ML507 Evaluation Platform v3.1.2. UG347, Xilinx, Inc., May 2011. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf.

[Xil11c] Spartan-3 Generation FPGA User Guide v1.8. UG331, Xilinx, Inc., June 2011. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf.

[Xil11d] Xilinx LogiCORE IP AXI HWICAP v2.0 Data Sheet. DS817, Xilinx, Inc., June 2011. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/ip_documentation/axi_hwicap/v2_00_a/ds817_axi_hwicap.pdf.

[Xil12a] LogiCORE IP Fast Fourier Transform v8.0 Data Sheet. DS808, Xilinx, Inc., July 2012. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/ip_documentation/ds808_xfft.pdf.

[Xil12b] LogiCORE IP Viterbi Decoder v8.0 Data Sheet. PG027, Xilinx, Inc., January 2012. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/ip_documentation/viterbi/v8_0/pg027_viterbi_decoder.pdf.

[Xil12c] Partial Reconfiguration User Guide v14.1. UG702, Xilinx, Inc., May 2012. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug702.pdf.

[Xil12d] Virtex-5 FPGA Configuration User Guide v3.11. UG191, Xilinx, Inc., October 2012. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug191.pdf.

[Xil12e] Virtex-5 FPGA User Guide v5.4. UG190, Xilinx, Inc., March 2012. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf.

[Xil14] Vivado Design Suite User Guide: Partial Reconfiguration v2014.4. UG909, Xilinx, Inc., November 2014. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_4/ug909-vivado-partial-reconfiguration.pdf.

[Xil15a] 7 Series FPGAs Configuration v1.10. UG470, Xilinx, Inc., June 2015. (Accessed May 11 2016). URL: http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf.

[Xil15b] Virtex-6 FPGA Configuration v3.9. UG360, Xilinx, Inc., November 2015. (Accessed May 11 2016). URL: `http://www.xilinx.com/support/documentation/user_guides/ug360.pdf`.

[Xil15c] Xilinx Spartan-6 FPGA Configuration User Guide v2.8. UG380, Xilinx, Inc., November 2015. (Accessed May 11 2016). URL: `http://www.xilinx.com/support/documentation/user_guides/ug380.pdf`.

[XSSK10] Haifeng Xiao, Y.Q. Shi, Wei Su, and J. Kosinski. An Investigation of Non-Data-Aided SNR Estimation Techniques for Analog Modulation Signals. In *IEEE Sarnoff Symposium*, pages 1–5, April 2010. `doi:10.1109/SARNOF.2010.5469706`.

[Zob00] Zobel, Justin and W. Dart, Philip. Partitioning Number Sequences into Optimal Subsequences. In *Journal of Research and Practice in Information Technology*, volume 32, pages 121–129, May 2000.

[ZTE] ZTEX GmbH. USB-FPGA-Module 1.11c with Spartan-6 XC6SLX25. (Accessed May 11 2016). URL: `http://www.ztex.de/usb-fpga-1/usb-fpga-1.11.d.html`.

List of Personal Publications

[FIIS12] M. Feilen, A. Iliopoulos, M. Ihmig, and W. Stechele. Partitioning and context switching for a reconfigurable FPGA-based DAB receiver. In *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–8, October 2012.

[FIS10] M. Feilen, M. Ihmig, and W. Stechele. Concept and Design of an SNR-adaptive DRM+/FM Receiver using Dynamic Partial Reconfiguration (DPR) of FPGAs. In *11th Workshop Digital Broadcasting*, Erlangen, Germany, September 2010.

[FISS12] M. Feilen, M. Ihmig, C. Schwarzbauer, and W. Stechele. Efficient DVB-T2 decoding accelerator design by time-multiplexing FPGA resources. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 75–82, August 2012. doi:10.1109/FPL.2012.6339244.

[FIVS13] M. Feilen, A. Iliopoulos, M. Vonbun, and W. Stechele. Weighted partitioning of sequential processing chains for dynamically reconfigurable FPGAS. In *23rd International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, September 2013. doi:10.1109/FPL.2013.6645521.

[FIZS11] M. Feilen, M. Ihmig, A. Zahlheimer, and W. Stechele. Real-time signal processing on low-cost-FPGAs using dynamic partial reconfiguration. In *13th International Symposium on Integrated Circuits (ISIC)*, pages 110–113, December 2011. doi:10.1109/ISICir.2011.6131921.

[FSHS11] M. Feilen, L. Stolz, C. Hausl, and W. Stechele. Improving the performance of Digital Radio Mondiale Plus (DRM+) by LDPC channel coding. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5, June 2011. doi:10.1109/BMSB.2011.5954939.

[IFH12a] M. Ihmig, M. Feilen, and A. Herkersdorf. Analytical Design Space Exploration Based on Statistically Refined Runtime and Logic Estimation for Software Defined Radios. In *15th Euromicro Conference on Digital System Design (DSD)*, pages 445–452, September 2012. doi:10.1109/DSD.2012.23.

[IFH12b] M. Ihmig, M. Feilen, and A. Herkersdorf. On the Accuracy of sum-based Logic and Power Estimates in hardware-accelerated SDR systems. In *7th Karlsruhe Workshop on Software Radios (WSR)*, "Karlsruhe, Germany", March 2012.

[KTB+12] D. Koch, J. Torresen, C. Beckhoff, D. Ziener, C. Dennl, V. Breuer, J. Teich, M. Feilen, and W. Stechele. Partial reconfiguration on FPGAs in practice; Tools and applications. In *ARCS Workshops (ARCS)*, pages 1–12, February 2012.

[SFS12] L. Stolz, M. Feilen, and W. Stechele. "An Optimized Software-defined Digital Audio Broadcasting (DAB) Receiver for x86 Platforms". In *"7th Karlsruhe Workshop on Software Radios (WSR)"*, "Karlsruhe, Germany", March 2012.

[VWF+13] M. Vonbun, S. Wallentowitz, M. Feilen, W. Stechele, and A. Herkersdorf. Evaluation of hop count advantages of network-coded 2D-mesh NoCs. In *23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 134–141, September 2013. doi:10.1109/PATMOS.2013.6662166.

List of Abbreviations

AC97	Audio Codec 97
ADC	analog-to-digital converter
AES	advanced encryption standard
AGC	automatic gain control
AMBA	Advanced Microcontroller Bus Architecture
ASIC	application-specific integrated circuit
AWGN	additive white Gaussian noise
AXI	advanced extensible interface bus
BCH	Bose-Chaudhuri-Hocquenghem
BER	bit error rate
BPI	byte peripheral interface
BRAM	block random access memory
CIF	common interleaved frame
CLB	configurable logic block
CNR	carrier-to-noise ratio
CPP	configuration packet processor
CRC	cyclic redundancy checksum
DAB	digital audio broadcasting
DAQ	data acquisition
DCM	digital clock manager
DDR-RAM	double data rate random-access memory
DDS	direct digital synthesis
DEBPSK	differentially-encoded binary phase-shift keying
DFT	discrete Fourier transform
DMA	direct memory access
DPGA	dynamically programmable gate array
DPLL	digital phase-locked loop
DPR	dynamic partial reconfiguration
DQPSK	differential quadrature phase-shift keying
DSP	digital signal processing
DVB-T	terrestrial digital video broadcasting
EAPR	early access partial reconfiguration
EEPROM	electrically erasable programmable read-only memory
EPROM	erasable programmable read-only memory
ETSI	European Telecommunications Standards Institute

FDR	frame data register
FEC	forward error correction
FF	flip-flop
FFT	fast Fourier transform
FIC	fast information channel
FIFO	first-in first-out buffer
FIR	finite impulse response
FM	frequency modulation
FPGA	field-programmable gate array
FSM	finite-state machine
GPIO	general purpose input and output
HE-AAC	high-efficiency advanced audio coding
HF	high frequency
I/O	input and output
ICAP	internal configuration access port
ID	integrate and dump
IEC	International Electrotechnical Commission
IIR	infinite impulse response
IOB	I/O Block
IP	intellectual property
ITU	International Telecommunication Union
JTAG	Joint Test Action Group
LDPC	low-density parity check code
LLR	log-likelihood ratio
LUT	lookup table
MAC	multiply and accumulate
MCB	memory controller block
MIG	memory interface generator
MPX	multiplex
MSB	most significant bit
MSC	main service channel
MUX	multiplex
NCD	native circuit description
NGC	netlist file with constraint information
OFDM	orthogonal frequency division multiplexing
PAD	program associated data
PCM	pulse-code modulation
PE	processing element

PLB	processor local bus
PLL	phase-locked loop
PLP	physical layer pipe
PRBS	pseudo-random binary sequence
PRCB	partial reconfiguration control block
PSD	power spectral density
PSK	phase-shift keying
PSM	programmable switching matrices
QAM	quadrature amplitude modulation
RDS	radio data system
RF	radio frequency
RMSE	root mean-square error
SDR	software-defined radio
SDRAM	synchronous dynamic random access memory
SelectMAP	selectable microprocessor access port
SINAD	signal-to-noise and distortion ratio
SNR	signal-to-noise ratio
SOS	second-order filter sections
SPI	serial peripheral interface
SRAM	static random-access memory
TCL	tool command language
UCF	user constraint file
UEP	unequal error protection
USB	universal serial bus
VHDL	very high speed integrated circuit hardware description language
VHF	very high frequency
XDL	Xilinx design language