REDUCED BINARY TREE FIR FILTERS

Artur Wróblewski, Marek Wróblewski, Christoph Saas and Josef A. Nossek

Munich University of Technology
Arcisstr. 21, 80333 Munich, Germany
e–mail: Artur.Wroblewski@ei.tum.de

ABSTRACT

In this paper we propose a novel implementation of FIR Filters based on parallel input serial output structure. Several simplifications have been achieved due to using reduced binary tree multipliers. Properties common to all coefficients of a filter have been exploited to modify and improve the performance of the structure as well as to decrease both the area and power consumption when compared to a state-of-the-art implementation.

1. INTRODUCTION

In recent years there has been a clear trend to move as many components of a system from the analog to the digital domain. In that struggle digital filtering has become one of the most challenging tasks. Digital filters can be made much more accurate than their analog counterparts, but they also need a large number of resources, especially multipliers, to perform the desired operation. Therefore, in terms of area and power consumption, digital filtering has become one of the most critical subsystems in many applications, like e.g. Software Defined Radio (SDR) [5]. Optimizing computational complexity of digital filters is a goal that has been widely investigated in the past years. Some approaches targeted algorithmic reduction of computational effort (e.g. [1], [6], [7]), some others aimed at simplifications in hardware complexity and architectural modifications (e.g. [8], [4], [2]). The goal of both approaches is similar: reduce the number or cost of used multipliers. In this work we present a method of implementing multipliers and, based on that, FIR filters. By applying the structure of the binary tree the properties common to all taps of a filter can be extracted, leading to significant reduction in computational complexity of multipliers and to savings in power consumption and area. Also, in some cases, shorter path delays and thus higher clock frequencies can be achieved.

At this point it should be stressed, that the following considerations apply to filters with *a priori* unknown coefficients for which the multipliers cannot in general be realized by means of shift-and-add operations. On the contrary, high-speed, full multipliers are required, as the filter is employed in applications like e.g. signal processing at intermediate frequency in a reconfigurable mobile communications system.

The idea to base a multiplier on a binary tree structure has been presented in [3]. The tree consists of several adders, which work in parallel often performing operations yielding the same result. Optimization potential can be exploited by replacing the adders with multiplexers, as in the reduced binary tree multiplier of Section 2. In a similar manner simplifications in performed arithmetics can be applied to a more general design like parallel-in-serial-out FIR filter. This will be explained in Section 3. The advantages of the modified filter are supported by experimental results of Sect. 4, while a conclusion is given in Section 5.

2. REDUCED BINARY TREE MULTIPLIER

Out of many multiplier structures presented in the past, the simplest solutions have been based on an array of full-adders. While very area-efficient, these multipliers are rather slow and very disadvantageous in terms of power consumption. Due to very high number of paths and hence resulting high glitching activity in the array, often alternative architectures are favored.

The widely-used, state-of-the-art approach is the Wallace-tree-multiplier. For word-lengths of more than 16 bits a booth-encoded Wallace-tree is used, while for smaller word-lengths a non-booth-encoded is preferred. Wallace-trees are extremely fast, have low power consumption but need more area than array-multipliers, especially for word-lengths of up to 16 bits. Nevertheless, they achieve very good power-area and delay-area products.

A different technique is to use a multiplier based on a binary tree (BT) (Fig.1). This approach is very efficient for up to 16 bit

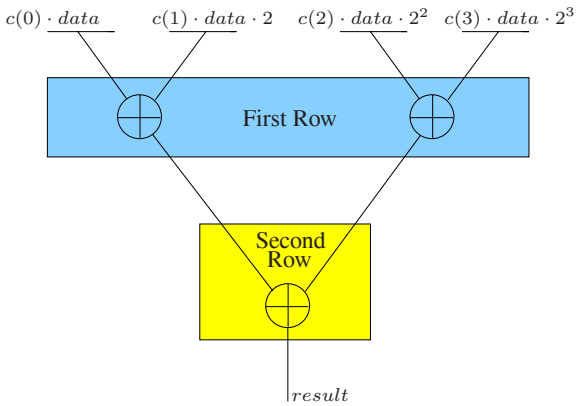
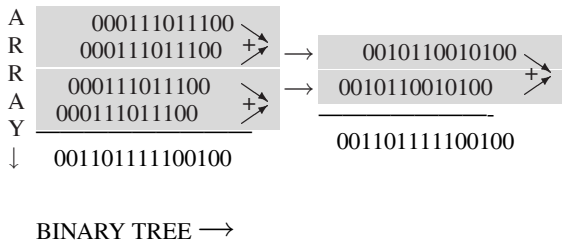


Figure 1: Binary Tree Multiplier

word-length and can be designed to be both faster and smaller than array multiplier while keeping the power-area and delay-area products only slightly larger than for Wallace-trees. From the BT-structure a reduced binary tree multiplier can be derived. It em-

ploys a reduced number of adders, which contributes to savings in area and power consumption.

The first row of adders in Fig.1 can be obtained by splitting up the multiplier array into blocks. Each block consists of one adder summing up two numbers. Let us consider a multiplication of a data with 12 and a coefficient with 4 bits word-length. The array below shows the result of the multiplication of an input data $data$ = "000111011100" with a coefficient C = "1111" in a binary tree.



In the case of a four bit coefficient, for an array multiplier it holds:

$$result = data + data \cdot 2 + data \cdot 2^2 + data \cdot 2^3,$$

while for the binary tree:

$$result = (data + data \cdot 2) + (data \cdot 2^2 + data \cdot 2^3),$$

with the brackets resulting from forming groups of adders. The number of levels (rows) of a binary tree (BT) required to perform a multiplication is

$$rows = \lceil \log_2(n) \rceil \quad (1)$$

with n being the word-length of the coefficient. Depending on the bitmap of the coefficient C, the result of each addition of partial products (adders in the first row of Figure 1) can have only four distinguished values:

$$Sum \in (0, data, data \cdot 2, data + data \cdot 2) \quad (2)$$

From the four values only $data + data \cdot 2$ needs to be calculated since obtaining a 0 is trivial, $data$ is already available and

$$data \cdot 2 = shl(data, 1) \quad (3)$$

can be derived from the input with no computational effort by means of a shift operation. Here $shl(B, n)$ denotes the shift left by n bits performed on an input data B. From the above can be stated that to provide all possible results for the sums of partial products it is sufficient to include only one adder for the whole first row of Figure 1. To obtain a reduced binary tree we simplify:

$$\begin{aligned} result &= (data + data \cdot 2) + (data \cdot 2^2 + data \cdot 2^3) \\ &= (data + data \cdot 2) + 2^2(data + data \cdot 2). \end{aligned}$$

Clearly, to calculate the result of the multiplication, the sum $data + data \cdot 2$ has to be performed and its two shifted versions added up.

$$\begin{aligned} result &= (data + data \cdot 2) + 2^2(data + data \cdot 2) \\ &= (data + data \cdot 2) + shl((data + data \cdot 2), 2) \end{aligned}$$

The actual sums of partial products can be obtained by multiplexing the values defined in (2) depending on the coefficient's bitmap.

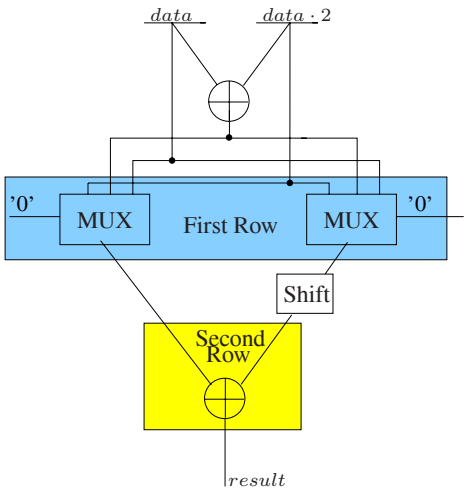


Figure 2: Reduced Binary Tree

From the above we can draw the structure of a reduced binary tree (RBT) multiplier as shown in Fig.2. Obviously, the advantage of the reduced number of adders, and thus the area savings, comes at the expense of additional delay introduced by the multiplexer. On the other hand, there's only one adder for the whole first row, which yields smaller area and reduced power consumption.

3. REDUCED BINARY TREE FIR

A disadvantage of the RBT-multiplier is, that the sum $data + data \cdot 2$ is always calculated, even if it is not needed due to the value of the coefficient. In some cases this could lead to increased power consumption when compared to the standard binary tree solution. Therefore it is controversial if it is reasonable to employ this technique in general purpose multipliers. However, in some applications, like FIR-filters, utilizing reduced binary tree multipliers leads to significant savings in area and power consumption.

3.1. MUX-multiplier

The most straightforward approach to apply RBT-multipliers to FIR filters is to replace standard multipliers with the modified structure. This doesn't change anything in the parallel-serial structure of the filter (Fig.3).

3.2. First row at filter input

From Figures 2 and 3 we notice, that the sum $data + data \cdot 2$ is being calculated for every coefficient independently. Since the incoming $data$ is the same for all multipliers, the sum $data + data \cdot 2$ is also equal. Therefore it is sufficient to evaluate this common value only once at the input of the filter. The adder responsible for performing this sum (adder on the top in Fig. 2) can be removed from all RBT-multipliers. Thus, the resulting structure looks like a binary tree multiplier (Fig. 1) with the adders of the first row replaced by multiplexers. This circuit, which is not a multiplier anymore and will be referred to as *common first adder* (CFA), has a shorter delay time. The FIR filter utilizing CFAs (Fig. 4), which

Bits	Wallace			RBT			CFA			CTR		
	17 taps	25 taps	75 taps	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)
8	1803	2862	9084	1507 (17%)	2423 (16%)	7835 (14%)	1372 (24%)	2101 (27%)	6754 (26%)	1620 (10%)	2530 (12%)	7611 (17%)
16	12057	17333	52253	8719 (28%)	12763 (27%)	39123 (26%)	7869 (35%)	10956 (37%)	34180 (35%)	8425 (30%)	12184 (30%)	40041 (24%)
32	80810	117034	353468	47922 (41%)	66201 (44%)	203916 (43%)	42831 (47%)	59025 (50%)	178206 (50%)	45718 (44%)	63747 (46%)	207679 (42%)

Table 1: Comparison of power consumption (in mW) of four architectures (Wallace - Wallace tree, RBT - reduced binary tree multiplier, CFA - first row adders moved to the filter input, CTR - adders from two rows moved to the input). SAV - savings in percent.

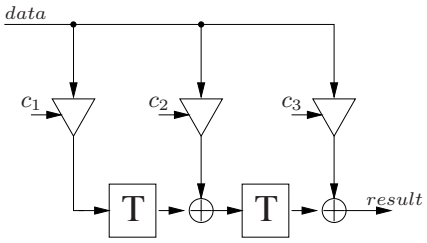


Figure 3: Parallel-In Serial-Out FIR filter.

is obtained at the expense of two additional registers, will have two important advantages:

- The critical path in the multipliers will be shorter, resulting in a filter with a higher admissible clock frequency and
- the area and power consumption can be reduced.

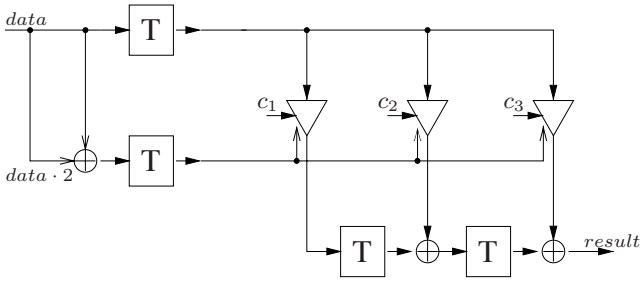


Figure 4: First row adders can be moved to the filter input.

3.3. Two rows at input

Common adders in conjunction with multiplexers can replace not only the first, but also the second row of the binary tree. In FIR structure, the resulting substitute for the multiplier, here referred to as *common two rows* (CTR), contains only one single multiplexer in place of the first two rows of the multiplier. This multiplexer has to switch between 16 possible output combinations out of which 9 can be derived from other results while the remaining 7 additions need to be pre-calculated. Also here the main drawback is that the computations are performed no matter if it is required for the given set of coefficients or not. The hardware employed to obtain these numbers is common for all coefficients and thus can be moved to the input of the filter. A register bank has been added to accumulate the 7 independent results. The resulting FIR-filter is depicted in Figure 5 with six out of seven adders placed at the input of the filter as to form an arithmetic unit.

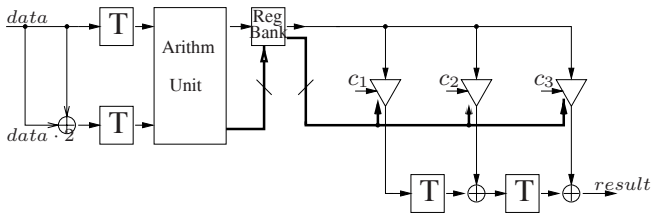


Figure 5: Reduced FIR with two rows moved to the input.

As presented in Section 4, the speed of the above described filter circuit is very high, but both the area and power consumption are significantly larger than in the design with only one row replaced with multiplexers. The reason for that is the area occupied by one 1-of-16 multiplexer, which is higher than that of 3 adders that have been replaced in the binary tree.

4. EXPERIMENTAL RESULTS

To prove the advantages of the reduced binary tree FIR-filters presented here several designs have been implemented in a high-level description language. These designs have been tested for functional equivalence with the state-of-the-art approach. The delay and area have been estimated using commercially available tools, while power consumption was obtained from simulations on transistor level. We present here three design examples for word-lengths ranging from 8 to 32 bits. The same word-length has been used for data and filter coefficients. The input data has been obtained from a matlab model of a communications system in order to emulate a practically relevant input stream. The simulations extended over several thousand cycles at 50 MHz. In tables 1-3

Bits	Wallace			RBT			CFA			CTR		
	17 taps	25 taps	75 taps	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)	17 taps (SAV)	25 taps (SAV)	75 taps (SAV)
8	0.100	0.148	0.448	0.083 (18%)	0.122 (18%)	0.369 (18%)	0.076 (24%)	0.112 (25%)	0.338 (25%)	0.125 (-24%)	0.181 (-21%)	0.537 (-20%)
16	0.394	0.581	1.680	0.291 (26%)	0.430 (26%)	1.294 (23%)	0.282 (29%)	0.415 (29%)	1.254 (26%)	0.413 (-5%)	0.605 (-4%)	1.819 (-8%)
32	1.263	1.860	5.588	1.087 (14%)	1.601 (14%)	4.811 (14%)	1.079 (15%)	1.588 (15%)	4.759 (15%)	1.509 (-19%)	2.204 (-18%)	6.583 (-17%)

Table 2: Comparison of area (in mm^2) of four architectures (Wallace - Wallace Tree, RBT - reduced binary tree multiplier, CFA - first row adders moved to the filter input, CTR - adders from two rows moved to the input). SAV - savings in percent.

a comparison of power consumption, area and delay for the four structures is given. The results are presented for three FIR low-pass filters with 17, 25 and 75 taps. Most savings in power consumption can be achieved with the structure of Fig. 4. This structure is also the best in terms of area. However it is slightly slower than the Wallace-tree reference for larger word-lengths. The structure, where the multiplier has simply been replaced by a reduced binary tree is the slowest one, but achieved savings in power and area are significant. The fastest is the design where both binary tree rows have been replaced by a multiplexer. Even if the multiplexer used here is very large, considerable savings in power consumption can be observed. Obviously, in most cases, the best choice is the reduced binary tree with first row adders moved to the filter input. Especially for the practical word-lengths (up to 16 bits) this structure offers best performance in terms of cost and outperforms the state-of-the-art Wallace-tree approach in all benchmarks. Please note, that the results presented in Tables 1 and 2 correspond to designs with the highest area efficiency, while Table 3 shows the highest speed possible without the area increase taken into account. Neverteless, the presented savings in area and power consumption have been achieved with digital filters being able to run at 80 MHz (32 bits word-length) or more (smaller word-lengths). Filters running at such high frequencies are able to meet even the very stringent requirements set on applications like Software Defined Radio. Thus, binary tree FIR filters seem to be a very good alternative to the state-of-the-art implementation in almost all practical applications.

Bits	Wallace	RBT	CFA at input	2 CTR at input
8	1.49	1.82	1.4	1.05
16	2.29	3.25	2.62	1.93
32	2.88	4.8	4.03	3.33

Table 3: Comparison of multiplier delay (in ns) of four architectures (Wallace - Wallace Tree, RBT - reduced binary tree multiplier, CFA - first row adders moved to the filter input, CTR - adders from two rows moved to the input).

5. CONCLUSION

In this paper we have presented an advantageous way to implement parallel-in-serial-out FIR digital filters. The solution is based on reduced binary tree multipliers in which inter-communities have been exploited to obtain overall complexity reduction. The resulting design provides more flexibility to the designer, offering more freedom in choosing delay-power and delay-area trade-off than the state-of-the-art solution. In fact, it makes possible to design low-power high-speed FIR filters accepting an increased area. On the other hand, low-power, low-area design is possible with this method, if speed is not an issue. However, even in that case, the achieved critical path delays are small enough to satify the requirements of most practical designs. All implementations presented here provide significant savings in power consumption (up to 50 %) and area (up to 29 %), which makes them highly desirable in applications like e.g. mobile communications systems.

6. REFERENCES

[1] R. E. Crochiere and L. E. Rabiner. *Multirate Digital Signal Processing*. Prentice Hall, 1983.

[2] Pearson D.N. and Parhi K.K. Low power FIR filter architectures. *Proc. ISCAS'95, Seattle, WA*, 1:231–234, May 1995.

[3] Abu-Shama E., Maaz M.B., and Bayonmi M.A. A Fast And Low Power Muliplier Architecture. *IEEE 39th Midwest Symposium on Circuits And Systems*, 1:53–56, August 1996.

[4] K. Hwang. *Computer Arithmetic*. John Wiley & Sons, New York, 1979.

[5] J. Mitola. *Software Radio Architectures*. Wiley-Interscience, New York, 2000.

[6] K. Muhammad and K. Roy. On complexity Reduction of FIR digital filters using constrained least squares solution. *International Conference on Computer Design, ICCD'97, Austin, TX*, pages 196–, October 1997.

[7] A.P. Vinod, E.M-K. Lai, A.B. Premkumar, and C.T. Lau. Hardware Efficient FIR Filter Implementation Using Subfilters For Digital Receivers. *Proc. International Symposium on Signal Processing and its Applications, ISSPA 2003, Paris*, July 2003.

[8] A. Wróblewski, Marek Wróblewski, and J. A. Nossek. FIR Filters With Reduced Word-Length. *Proc. European Conference on Circuit Theory and Design, ECCTD'03, Kraków*, September 2003.