# CORDIC-BASED ARCHITECTURES FOR THE EFFICIENT IMPLEMENTATION OF DISCRETE WAVELET TRANSFORMS

*Sven Simon, Peter Rieder, Christian Schimpfle and Josef A. Nossek*

Institute for Network Theory and Circuit Design
Technical University Munich
Arcisstr. 21, 80333 Munich, Germany
svsi@nws.e-technik.tu-muenchen.de

## ABSTRACT

In this paper, efficient implementations of the discrete wavelet transform (DWT) are examined. A new architecture, based on single steps of the CORDIC algorithm with less operations compared to classical lattice or transversal filter structures is presented. In addition to the computational savings, a generalized implementation of a single CORDIC-based approximate rotation as a module generator is sufficient to automate the full custom layout generation of the arithmetic part of the DWT. An exemplary VLSI implementation for a typical image processing application is presented.

## 1. INTRODUCTION

In recent years, wavelet transforms have gained a lot of interest, whereby orthogonal wavelet transforms [1] are used in many application fields, e.g. image processing. An orthogonal wavelet transform decomposes a signal in dilated and translated versions of the wavelet function. In the discrete case a filterbank structure is used to decompose the signal, whereby with each stage of the transform, the signal is filtered with a lowpass $G(z) = \sum_{i=0}^{n-1} g_i z^{-i}$ and the complementary highpass $H(z) = \sum_{i=0}^{n-1} h_i z^{-i}$, see Figure 3. Only the lowpass part of the signal is further processed with the next stage of the transform, what leading to multiresolution analysis of the given signal. In the following sections, the aspects of VLSI implementations of the DWT is examined and a new methodology to derive efficient architectures is presented.

## 2. THE ARCHITECTURE

A suitable architecture for the implementation of orthogonal wavelet filters are lattice structures [5]. Figure 1 shows a lattice filter implementation of one stage of Daubechies' wavelet transform of length $n = 4$ with its basic modules, namely orthogonal $2 \times 2$ rotations. By using these orthogonal rotations the orthogonality of the whole transform is structurally imposed [5] such that the possibilty of perfect reconstruction is guaranteed. For the lattice filter performing an orthogonal wavelet transform, G(z) must have at least one zero at $z = 1$. This is achieved, if the sum of rotation angles $\beta_k$ is chosen to be $-45^o$, for details see [4].

$$\sum_k \beta_k = -45^o.$$

A common method to execute orthogonal rotations is the CORDIC algorithm [6]. The rotation by any angle $\alpha$ is executed by a sequence of $(w+1)$ $\mu$-rotations ($w$ is the wordlength):

$$\mathbf{R}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} = \frac{1}{K_w} \prod_{s=0}^{w} \begin{bmatrix} 1 & -\sigma_s 2^{-s} \\ \sigma_s 2^{-s} & 1 \end{bmatrix}.$$

With $\frac{1}{K_w} = \prod_{s=0}^{w} \frac{1}{\sqrt{1+2^{-2s}}}$ being the scaling factor and $\sigma_s \in \{+1, -1\}$. This corresponds to the representation of $\alpha$ as

$$\alpha = \sum_s \sigma_s \alpha_s = \sum_s \sigma_s \arctan 2^{-s}.$$

This representation of an angle in the "$\arctan 2^{-s}$" basis is also the basic idea of CORDIC-based approximate rotations [2-4]. In order to avoid the full sequence of $\mu$-rotations, approximate rotations are used that approximate any rotation by a double rotation consisting of 2 equal $\mu$-rotations that only require very few shift and add operations.

$$\mathbf{R}(2\alpha_s) = \frac{1}{K_s^2} \begin{bmatrix} 1 & -\sigma 2^{-s} \\ \sigma 2^{-s} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\sigma 2^{-s} \\ \sigma 2^{-s} & 1 \end{bmatrix}.$$

The scaling factor $1/K_s^2$ can be factorized into a form suited for a VLSI implementation.

$$\frac{1}{K_s^2} = \frac{1}{\sqrt{1+2^{-2s}}^2} = (1-2^{-2s})(1+2^{-4s})(1+2^{-8s})\ldots$$

With respect to an efficient implementation of a wavelet transform, these approximate rotations are used to realize the necessary $2 \times 2$ rotations. With the sum of rotation angles being $-45^o$, the CORDIC $\mu$-rotation $\mathbf{R}(-\beta_0) = \mathbf{R}(-45^o)$ is always used. The scaling factor $\frac{1}{\sqrt{2}}$ is not critical, since the rotation $\mathbf{R}(-\alpha_0)$ appears not only in the analysis part but also in the synthesis part and the combination of the resulting scaling factors $\frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2}$ leads to a simple shift operation. The other rotations, whose sum of angles is $0^o$, are approximated with simple double rotations, of course, with different signs.

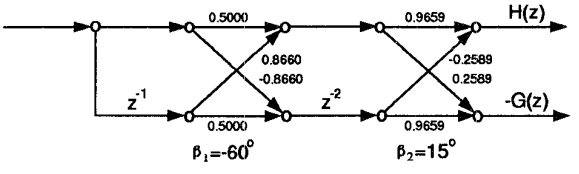


Figure 1: Classical lattice filter for implementing on stage of DWT.

0-7803-3073-0/96/$5.00 ©1996 IEEE

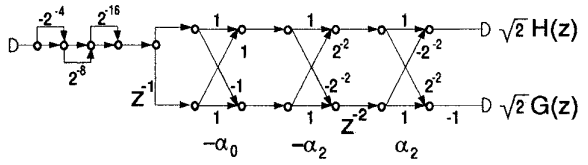


Figure 2: CORDIC-based filter for implementing stage of DWT.

For the wavelet transform of length $n = 4$ being used in this paper (see Figure 2), the CORDIC–based $\mu$–rotations lead to a suitable approximation of the exact wavelet transform by Daubechies ($-\alpha_0 - \alpha_2 = -59.03 \approx -60^\circ = \beta_1$ and $\alpha_2 = 14.03^\circ \approx 15^\circ = \beta_2$), whereby only very few shift and add operations are necessary.

In order to classify the computational complexity of the proposed architecture, the number of adders of the two architectures given in Figure 1 and 2 and the transversal filter implementation [13] are compared with variable wordlength, see Table 1. For the last two implementations, the coefficients are assumed to be realized with canonical signed digit code (CSD). Thus, a minimum number of adders is used for each coefficient because zero bits of the coefficient need not be implemented. Downsampling is realized by a demultiplexer at the input of each stage as shown in Figure 3. The scaling factor of the CORDIC based lattice structure (Figure 2) is assumed to be implemented twice because otherwise due to downsampling this would be the bottleneck and reduce the speed of the design. The coefficients in Table 1 are implemented with a wordlength of $w$ bit.

| word-length $w$ | CORDIC Lattice | Classical Lattice | Transversal Filter |
|---|---|---|---|
| 8 | 10 | 20 | 26 |
| 10 | 10 | 20 | 28 |
| 12 | 10 | 26 | 34 |
| 16 | 10 | 28 | 42 |
| 20 | 12 | 40 | 60 |

Table 1: Computational complexity of the DWT: number of adders.

In the following, the VLSI implementation of a DWT (3 stages) for a typical image processing application is presented. The pixels of the considered images have a resolution of 8 bit. Due to finite wordlength effects, which are examined for CORDIC stages in [8,9], several additional digits are added at the least significant part of the word. We choose 4 digits for this purpose in our implementation. Also the coefficient wordlength has to be chosen separately for each architecture of Table 1 depending on the specified quality of the DWT (SNR).

To avoid overflow, the word has to be extended also at the most significant part. Table 2 gives the number and reason of additional digits for our implementation.

| Digits | MSB | LSB | Reason |
|---|---|---|---|
| 4 | | x | Finite wordlength effects |
| 3 | x | | Energy compression, 3 stages |
| 1 | x | | Scaling factor 1/2 for 45-rotation |

Table 2: Wordlength considerations.

Thus, we obtain a wordlength $w$ of 16 bit and the number of 10 add operations for each stage of the DWT.

## 3. VLSI IMPLEMENTATION

In this section, the implementation of a processing element and related aspects for the proposed architecture are presented. In [7,10], the control flow of folded VLSI architectures is examined. Folding is used to improve hardware utilization of the filterbank because downsampling reduces the number of computations in each stage. In [10], Parhi and Nishitani compare two types of architectures, a word level folded architecture and a digit-serial architecture. If latency plays an important role the word level folded architecture should be preferred otherwise the digit-serial approach due to simpler and local interconnections. The filterbank with the digit-serial approach is shown in Figure 3. The filter stages with few parallel digits need less area than a full parallel implementation. Taking these results concerning the control flow into account, we focus our examinations on the data path of the CORDIC-based implementation for both, the word level folded and digit-serial architecture.

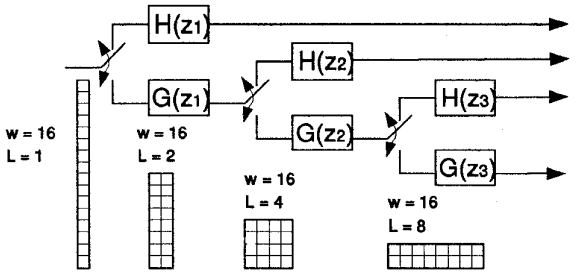


Figure 3: Filterbank with a digit-serial approach with $w/l$ parallel digits.

### 3.1. The Wordlevel Folded Architecture

The wordlevel parallel implementation uses one stage of the filterbank for the data computed in all stages of the filterbank. Thus, a processing element with a high throughput rate is needed compared to an unfolded or digit-serial version. Usually, redundant number systems like carry save or a signed digit number representation (SDNR) are used to achieve a high throughput rate because the carry ripple propagation delay $O(w)$ is reduced to one digit only $O(1)$. Unfortunately, the usage of e.g. carry save arithmetic requires twice the number of full adders and wires. Specifically for our architecture, the same throughput can be achieved with two's complement representation instead of redundant number implementations but with less area consumption, see Table 3. The reason for this fact is that the used CORDIC stages in our application have a small shift of $s = 2$ which leads to an efficient register distribution as derived in the following paragraphs.

| | Two's complement | Carry Save |
|---|---|---|
| FA | 10 | 20 |
| Reg | 13 | 10 |
| Area | 14.3 | 23.3 |

Table 3: Area consumption in full adder units per bit slice. (FA=full adder, Reg=Register, $Area_{Reg} = Area_{FA}/3$)

If the signal flow graph SFG of Figure 2 is used as a floorplan, the routing channel to realize the lattice structure needs a relatively large area $O(w)$. This can be reduced

drastically to $O(s)$, if we use the adders of each rotation in one row alternately in a modified floorplan. This modified floorplan for a $s = 2$ rotation on bit level with full adders $fa$ and interconnections is given in Figure 4. As shown in Figure 4 the SFG is suited for a bit slice architecture.

The shifter is implemented by local wire cells which are identical for each bit slice and which realize the global interconnections by abutment, as shown in Figure 5. The layout of the complete processing element (PE) for the wordlevel parallel implementation of one stage of the DWT is given in Figure 6.

To obtain the temporal local architecture with a propagation delay $O(1)$, each interconnection must at least contain one timing element, register or latch. These elements are added and moved form the boundaries to inner interconnections known as pipelining. In [11,12], methodologies to evaluate register or latch distributions from design goals are presented. Let $w_p(i,j), w_e(i,j)$ be the number of latches of a path $p$ or edge $e$ form node $i$ to $j$. The index $e$ is omitted in the following. The following equations represent all possible directed paths for the upper part of the architecture.

$$w_{p1}(in_i, out_j) = w(in_i, fa_i) + w(fa_i, fa_j) + w(fa_j, out_j)$$

$$w_{p2}(in_i, out_k) = w(in_i, fa_{i-s}) + w(fa_{i-s}, fa_k) + w(fa_k, out_k)$$

The symmetry $w_{p1}(in_i, out_j) = w_{p2}(in_i, out_j)$ leads to the equation $w(in_i, fa_i) = w(in_i, fa_i - s) + s$ and at least one latch should be at the output path such that $w(fa_j, out_j) \geq 1$. In order to choose the minimum number of latches, we obtain $w(in_i, fa_i) = s+1$ and $w(fa_j, out_j) = 1$. This leads to the distribution shown in Figure 4.

The value $w_{p1}(in_i, out_j) - w_{p1}(in_{i-1}, out_j) = 1$ means that data of neigbouring digits have a skew of 1 clock cycle. These results can also be found applying the usual cut-set rules to the signal flow graph, but here a parameterized architecture with parameter $s$ was analyzed. With regard to module generators evaluations based on equations are more useful than moving registers in SFGs by hand.

### 3.2. The Digit-Serial Architecture

The digit serial approach divides the number of parallel digits by 2 after each demultiplexer, see Figure 3. Thus, the implementation of each rotation depends not only on the parameter $s$ but also on the dataformat. A module generator for a parameterized CORDIC $\mu$–rotation was developed such that the full custom layout of the arithmetic core of the whole filterbank structure can be generated. The Cadence Design Framework with Skill programming laguage was used for this generator.

The two's complement number representation was chosen again because with a wordlength of 16 bit we obtain only 8 parallel digits for the first stage of the filterbank. Thus, the throughput is determined by the propagation delay of an 8 bit carry ripple adder (in our technology: 5.2ns). Figure 7 shows the result of a generated layout for a shift 2 rotation.

## 4. CONCLUSION

In this paper, a novel architecture for the discrete wavelet transfrom (DWT) is presented. The proposed architecture, based on single stages of the CORDIC algorithm, reduces hardware complexity compared to classical lattice or transversal filter structures. As this reduction depends on the specific problem, a classical image processing application was chosen for comparison for which the the hardware complexity was reduced by a factor of 2 to 4.

The fact that CORDIC rotations are the basic element of the architecture, enables to automate the layout generation of an arbitrary lattice filter by a module generator for a single parameterized CORDIC stage. This module generator and a word parallel, bit slice architecture with localized shift wire cells have been implemented. The presented methodology cannot only be applied to filterbanks specified for the DWT but to classical lattice filter architectures in general. Future research will focus on how to generalize this methodology for DSP synthesis.

## REFERENCES


[1] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, 92.

[2] J. Götze and G.J. Hekstra. Adaptive Approximate Rotations for Computing the EVD. In M. Moonen and F. Cathoor, editors, *Algorithms and Parallel VLSI Architectures.* Elsevier Science Publishers, 1994.

[3] J. Götze, S. Paul, and M. Sauer. An Efficient Jacobi-like Algorithm for Parallel Eigenvalue Computation. *IEEE Trans. on Computers*, 42(9):1058–1065, 9 1993.

[4] P. Rieder, K. Gerganoff, J. Götze, and J.A. Nossek. Parameterization and Implementation of Orthogonal Wavelet Transforms. *submitted to IEEE ICASSP 96*, Atlanta.

[5] P.P. Vaidyanathan and P. Hoang. Lattice Structures for Optimal Design and Robust Implementation of Two-Channel Perfect-Reconstruction QMF Banks. *IEEE Trans. on ASSP*, 36(1), January 1988.

[6] J.E. Volder. The CORDIC trigonometric computing technique. *IRE Trans. Electr. Comput.*,EC-8:330–334,59.

[7] G. Knowles. VLSI Architecture for the Discrete Wavelet Transform. *Electronic Letters*, 26(15), 19 July 1990, pp. 1184-1185.

[8] Y.H. Hu. The Quantization Effects of the CORDIC Algorithm, *IEEE Trans. on Signal Processing*, Vol. 40, NO. 4, April 1992, pp. 834-844.

[9] S. Paul, J. Götze, M. Sauer. Error Analysis of CORDIC-based Jacobi Algorithms, *IEEE Trans. on Computers*, Vol. 44, No7, July 1995, pp. 947-951

[10] K. K. Parhi and T. Nishitani, Folded VLSI Architectures for Discrete Wavelet Transform, *ISCAS 1993*, pp. 1734-1737.

[11] S.Simon, E. Bernard, M. Sauer and J.A. Nossek, A New Retiming Algorithm for Circuit Design, *ISCAS 1994*, pp. 4.35-4.37.

[12] S.Simon, J. Hofner and J.A. Nossek, Retiming of Circuits Containing Multiplexers, *ISCAS 1995*, pp. 1736-1739.

[13] A.S. Lewis and G. Knowles, Image Compression Using the 2-D Wavelet Transform, *IEEE Trans. on Image Processing*, Vol 1, No 2, April 1992, pp. 244-250.

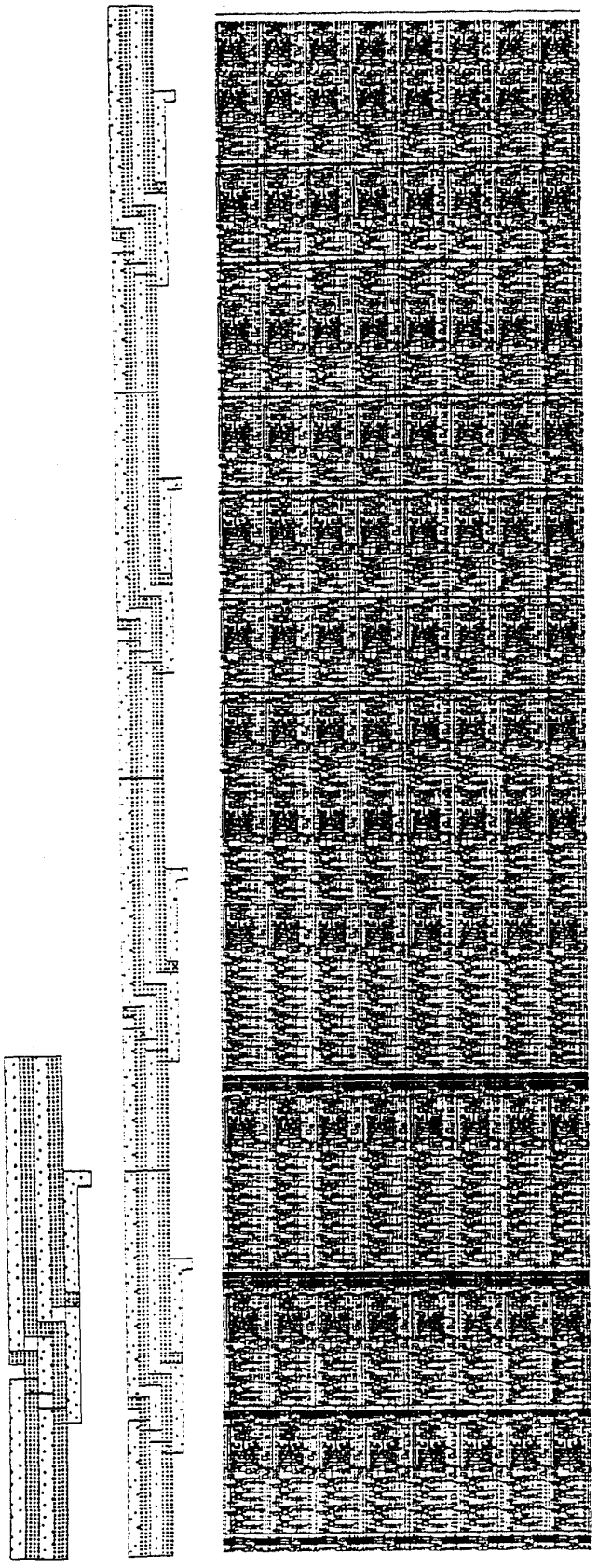

Figure 5: Local wire cells for global interconnections.

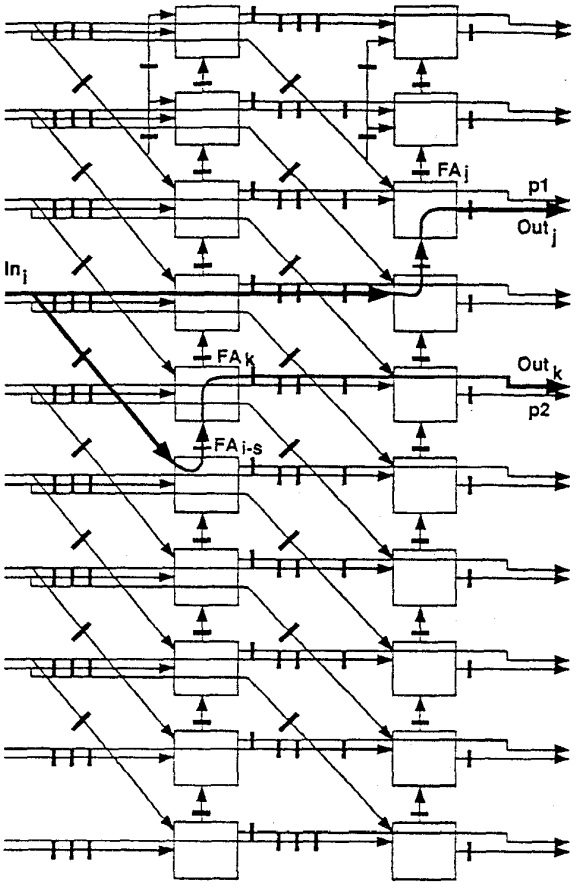


Figure 4: Systolic implementation of a rotation.

Figure 6: Bit slices for 8bit of one PE applying floorplan of Fig. 4.

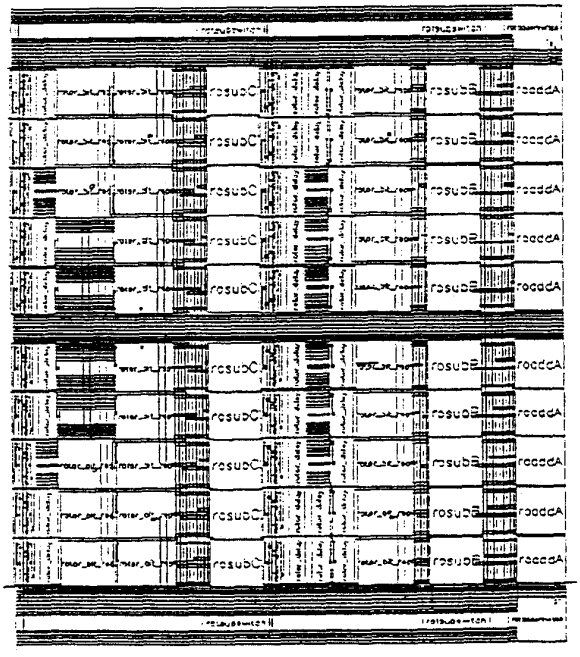

Figure 7: A digit-serial rotation.