EFFICIENT VLSI SUITED ARCHITECTURES FOR DISCRETE WAVELET TRANSFORMS

S. Simon, P. Rieder, J. A. Nossek
Technical University Munich, Germany
Sven.Simon@nws.e-technik.tu-muenchen.de

Abstract - In this paper, a variety of architectures for the discrete wavelet transform (DWT) is examined to derive an efficient VLSI implementation. The comparison leads to a lattice filter structure which uses single steps of the CORDIC algorithm. Due to the modular structure of the proposed architecture, this approach is especially suited for full custom design style using module generators to automate the manual design process.

INTRODUCTION

In recent years, wavelet transforms have gained a lot of interest. In this paper, architectures for the DWT are compared. The straight forward approach to realize the filter functions $H(z)$ and $G(z)$ of the DWT is the implementation as transversal filters. In the following, a polyphase structure for the filters is applied with the notation $H(z) = H_{even}(z) + z^{-1}H_{odd}(z)$, see e.g. [9]. The associated filter with $2n$ multipliers for the transversal filter approach is given in Figure 1. In order to reduce the number of multiplications, an FIR filter structure is proposed in [16], which makes use of the conjugation property of the wavelet filter coefficients: $h_i = (-1)^i\, g_{n-1-i}$, see Figure 2. This architecture needs only $1.5n$ multipliers such that 25% of the hardware resources compared to the previous approach can be saved [16]. Alternative to these transversal filters, lattice filters, proposed by Vaidyanathan and Hoang in [5], further reduce the number of multiplications. The associated lattice filter for a DWT of length $n = 4$ is shown in Figure 3, which needs n multipliers for the rotations plus 2 multipliers for the scaling factor K. The scaling factor could be implemented only once, using multiplexers but this would limit the throughput of the filter by a factor of two. For the comparisons in this paper, the architectures are presented in a way, that the same

0-7803-3134-6/96 $5.00 © 1996 IEEE

throughput for each architecture can be achieved by pipelining of each word level addition to give a fair comparison. Additional savings can be made if we do not follow this principle in certain cases, but these savings will not influence the conclusions of this paper.

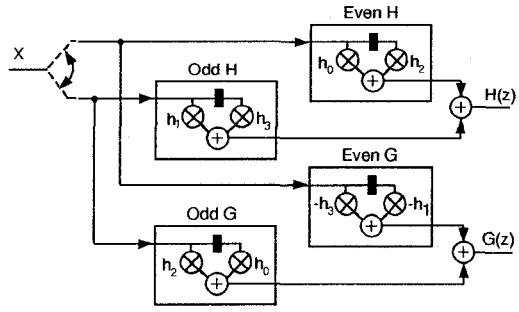

Figure 1: Transversal filter.

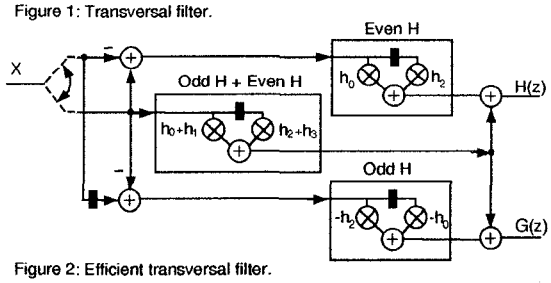

Figure 2: Efficient transversal filter.

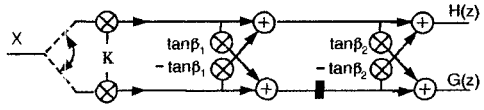

Figure 3: Lattice filter.

In [17], binomial quadratur mirror filters are used which furthermore reduce the number of multiplications to the expense of additional add operations compared to the previous architectures, see Figure 4. This methodology is applicable for the special case of orthogonal wavelet filters, e.g. in the case of Daubechies' wavelets of length n. In this case $H(z)$ has $p = n/2$ zeros at $z = -1$ and $G(z)$ has $p = n/2$ zeros at $z = 1$, respectively. Note, that p describes the number of vanishing moments. With the implementation of $(1 + z^{-1})^p$ or $(1 - z^{-1})^p$ being very simple, the computational complexity is essentially influenced by the rest of the transfer functions. For the Daubechies' wavelet transform of length $n = 4$ only two multiplications with $\sqrt{3}$ are necessary. The quantization of the coefficients determines the number of shift and add operations. While quantization does not affect the vanishing moments, the other important property of the wavelet

transform, namely orthogonality (perfect reconstruction), is lost (for the full wordlength) because of the approximation of the coefficients.

Using lattice filters instead, orthogonality is not violated. However, after the quantization of the lattice coefficients, the moments of the wavelet coefficients usually do not vanish anymore. The only possibility to conserve the most important first vanishing moment, that guarantees the wavelet being zero mean, is to choose the sum of lattice rotation angles being -45^o: $\sum \beta_i = -45^o$. This can be found in [4] in detail with a design method for orthogonal wavelets taking an efficient VLSI implementation into consideration. The basic idea is not to quantize in the domain of scalar coefficients, but in the domain of rotation angles. An elementary rotation of angle α_S is realized by one step of the CORDIC-algorithm, with $\alpha_S = \arctan 2^{-S}$. This is usually written in matrix form as follows:

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

For the Daubechies' wavelet transform of length $n = 4$ the angles 60^o and 15^o can be realized by the approximation of ($-\alpha_0 \; -\alpha_2 = -59.03 \approx -60^o = \beta_1$ and $\alpha_2 = 14.03^o \approx 15^o = \beta_2$), see Figure 5. Further examinations of this approximation in comparison to the DWT with exact angles can be found in [4]. Thus, only 3 CORDIC stages, which consists of two shift and add operations per stage and the shift and add operations of the scaling factor, have to be implemented.

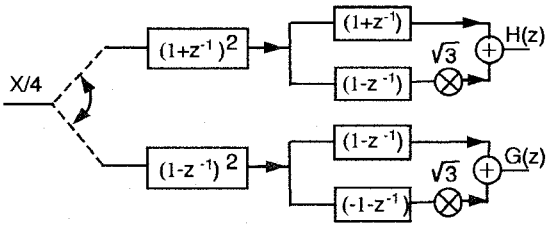

Figure 4: Binomial quadratur mirror filter.

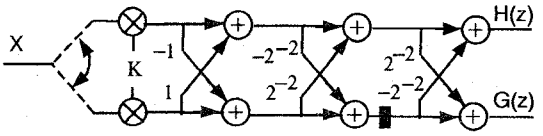

Figure 5: CORDIC lattice filter.

The hardware expense of the first three architectures could be estimated using the number of multiplications. Due to the additional

number of additions in Figure 4 and 5 and the fact that the coefficients are time invariant and can be implemented omitting zero bits for the multiplication, a more detailed analysis is carried out in the following. 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 series of one's are replaced by only two non zero bits.

In the following table the number of adders is listed for several wordlengths w.

Table 1: Number of adders using CSD for each coefficient of wordlength w.

w	1. Polyphase Filter	2. Efficient Pol. Filter	3. Bino. QMF	4. Classical Lattice	5. CORDIC Lattice
8	20	15	16	8	8
10	20	16	16	14	10
12	26	18	18	20	10
14	34	25	20	22	10
16	38	28	22	24	10
18	44	32	22	28	12

Obviously, the predicted difference of the required hardware resources of architectures 1, 2 and 3, still hold on the level of shift and add operations. The binomial QMF has a certain hardware offset due to the number of additions needed to realize the FIR filter part. If the wordlength increases, the realization of the multiplication by $\sqrt{3}$ needs additional adders.

Similarly, the implementation of the CORDIC based approach needs only additional digits for the improved accuracy of the scaling factor. As shown in Table 1, the CORDIC lattice filter saves between 60% and 75% of the additions compared to architecture 1. This is the main reason to choose this architecture our VLSI implementation. Additionally, the CORDIC stages lead to a modular design and very dense layout as shown in the following section.

VLSI IMPLEMENTATION

In this section, the implementation of a processing element and related aspects for the proposed architecture are presented. In [7, 9], 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 [9], 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 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.

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.

In order to achieve high throughput rates, the carry ripple path of a wordlevel addition has to be reduced. This can be achieved by two methodologies. Either redundant number systems are applied like carry save or like the signed digit number representation or the carry ripple path of the adders is pipelined using two's complement representation. In the first case, the propagation delay of an addition is reduced to those of two basic modules and is independent of the wordlength, but twice the number of adders and interconnections are necessary. In the second case, a register distribution is necessary which limits the carry path to b bit. The variable b adjusts the design to the specified throughput. Usually, the register distribution of data path designs is determined by cut-sets rules or as used in [13], but here a parameterized architecture with the variables b, w and s has to be examined. Therefore, we use another methodology from [10, 11].

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. Figure 14 contains the abbreviations and indices being used in the following considerations. The two equations below represent all possible directed paths for the upper part of the architecture

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

$$w_{p2}(in_i, out_j) = w(in_i, \bar{fa}_i) + w(\bar{fa}_{i-s}, \bar{fa}_j) + w(\bar{fa}_j, out_j).$$

The symmetry $w_{p1}(in_i, out_j) = w_{p2}(in_i, out_j)$ leads to the equation $w(in_i, va_i) = \bar{w}(in_i, \bar{va}_{i-s}) + \bar{w}(\bar{va}_{i-s}, \bar{va}_j) - w(va_i, va_j)$. If a systolic bit level architecture is required each full adder has one register at the carry output, such that $\bar{w}(\bar{va}_{i-s}, \bar{va}_j) = j - i + s$ and $w(in_i, va_i) = 1 + s$ if one register is assumed on the edge $e(in_i, \bar{va}_{i-s})$, see Figure 12. Otherwise, if the carry path is reduced to only b bit, we obtain

$\bar{w}(\bar{va}_{i-s}, \bar{va}_j) = \lfloor j/b \rfloor - \lfloor (i-s)/b \rfloor$ and $w(in_i, va_i) = \lfloor i/b \rfloor - \lfloor (i-s)/b \rfloor$, see Figure 13. The value $w_{p1}(in_i, out_j) - w_{p1}(in_{i-1}, out_j) = 1$ for the systolic case means that data of neigbouring digits have a skew of 1 clock cycle.

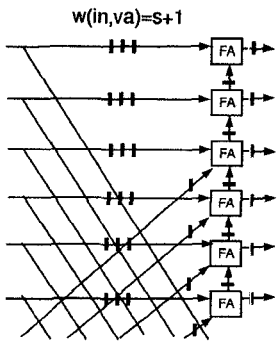

Figure 12: Systolic rotation.

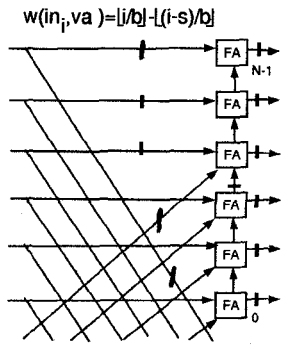

Figure 13: Reduced carry path.

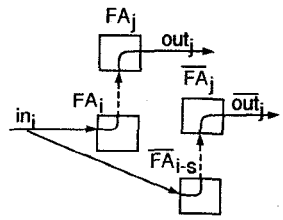

Figure 14: Considered paths.

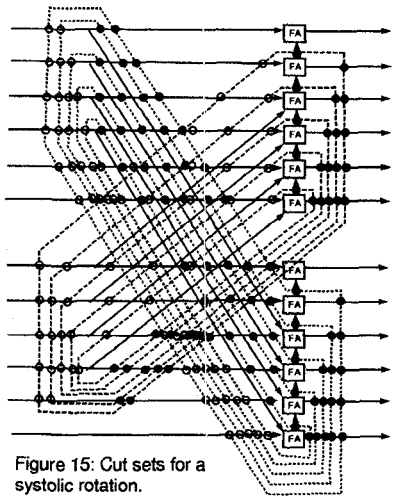

Figure 15: Cut sets for a systolic rotation.

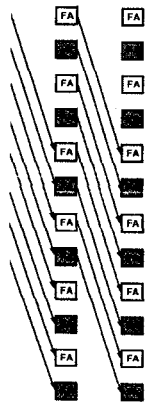

Figure 16: Bitwise interleaving.

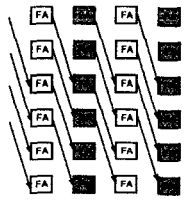

Figure 17: Wordwise interleaving.

These results can also be found applying the usual cut-set rules to the signal flow graph, see Figure 15 (moving negative and positive registers by cut sets), but here, a parameterized architecture with parameters b, w and s was analyzed. With regard to module generators, evaluations based on equations are more useful than moving registers in SFGs by hand.

The straightforward approach to implement the rotations applying the floorplan of the signal flow graph (SFG) in Figure 5, leads to channels for wires for all w bits, see e.g. Figure 15. The area necessary

for the routing channel can be reduced drastically if the architecture is modified on the bit level. Bit level architectures are known from e.g. [13]. In [14], an efficient bit level approach is presented, which rearranges the full adders such that binary rotations with a bitwise interleaving of the signal lines are used, see Figure 16.

Another bit level rearragement is shown in Figure 17. The interleaving of the single full adders is replaced by a horizontal interleaving of wordlevel additions and a bitwise interleaving of the signal lines. The size of the interconnection network only depends on the shift s and is identical zero if a rotation of 45^o is realized for both rearrangements. However, wires to realize the shift in Figure 17 are shorter in general because the wires of the Figure 16 connect adders with twice as many adder cells in between. For this reason the architecture proposed in Figure 17 is used for the implementations in this paper.

Obviously, the two different methodologies of interleaving have not only influence on the interconnection network but also on the final format (height and width) of the design. The architecture of Figure 16 will result in a layout with approximately the same format than the primary one of Figure 15, whereas the approach of Figure 17 results in a layout with half the width but double the length. This fact combined with further floorplanning considerations may influence the choice of the suited architecture in a general case.

MODULE GENERATOR AND RESULTS

In this section, a module generator and the produced layouts are presented. The digit serial approach divides the number of parallel digits by 2 after each stage of the filterpair $G(z)$ and $H(z)$. Thus, the implementation of each rotation depends not only on the parameter s but also on the dataformat. Although, the design consists of modular CORDIC stages, a fully handcrafted implementation would enforce the manual design of each rotation due to the variations of these parameters. This very time consuming process could be avoided if the layout of an arbitrarily CORDIC stage would be generated automatically. A module generator for a parameterized CORDIC rotation was developed using the Cadence Design Framework with the Skill programming language. The program consists of instructions on how to build up the layout of a rotation by abutment of basic cells. If the cells are modified in the design using hierarchy the program will produce a correct design without further modifications. Wiring is generally realized by cells which are placed on top of the other if necessary. Thus, if the design has to be changed the designer has only to deal with the modifications

of cells. A bit slice architecture was used which led to a regular design on bit level. As the scaling factor is implemented twice to avoid a bottleneck in the design, the scaling factor differs from a rotation only by wiring. The choice between a CORDIC rotation and scaling factor could easily be implemented in the generator program because the selection of different wiring cells is sufficient.

In order to use a bit slice structure, the wiring to realize the multiplications by 2^{-s} can be realized by cells as shown in Figure 9, instead of wires which cross several bit slices. These modular wiring cells realize a shift over several bit slices by abutment and need less area in many cases than straight horizontal and vertical wires.

Figure 9: Shift wire cells for an area efficient bit slice structure.

We choose a wordlength of 16 bits for our implementation for the following reasons. For an image processing application, the pixel resolution of the image is typically 8 bit. The energy compression in each stage of the DWT enforces 3 additional bits (three stages) for at the most significant part of the word. Additionally, due to the 45^o rotation another bit is necessary because the scaling by $1/2$ has to be considered. Furthermore in order to reduce finite wordlenght effects 4 digits are added at the least significant part of the word. These effects are examined in [8] for CORDIC stages.

Thus, the data path of a DWT for both the digit serial and world level folded architecture can be realized in few hours with this module generator. A resulting layouts in 1μ technology is shown in Figure 11. The density of the design of Figure 11 is 3965 transistors per mm^2 with a size of $1.7x1.1mm^2$ and a number 7456 transistors. This is comparable to the design of a single 16 bit fix point multiplication in size and complexity what underlines the efficiency of the derived design.

CONCLUSION

In this paper, we examined the properties concerning full custom VLSI implementation of filters for the DWT. We showed that the proposed CORDIC based lattice filter results in a very efficient implementation

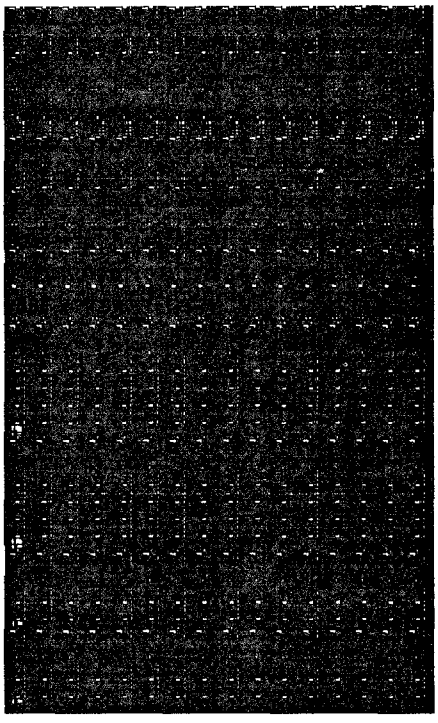

Figure 11: Layout of a parallel implementation of $G(z)$ and $H(z)$.

using less silicon area than the architectures known from literature. The analysis of the architecture on different abstraction levels enabled us to produce a VLSI suited design considering aspects like pipelining, interleaving of additions on bit level and modularity. Furthermore, due to the modular structure of the architecture, a module generator to generate arbitrary designs automatically was developed. 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] 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

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

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

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

[12] S.Simon, P. Rieder, C. Schimpfle and J.A. Nossek, , CORDIC-Based Architectures for the Efficient Implementations of DWTs *ISCAS 1995*.

[13] S. Y. Kung, VLSI Array Processors, Prentice Hall, New Jersey, 1989.

[14] M. Sauer, Algorithmustransformationen beim Entwurf anwendungsspezifischer integrierter Schaltungen, Verlag Shaker, Ph.D. thesis 1995

[15] 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.

[16] O. Rioul and P. Duhamel, Fast Algorithms for Discrete and Continuous Wavelet Transforms, *IEEE Trans. on Information Theory*, Vol 38, No 2, March 1992, pp. 569-586.

[17] A.N. Akansu, R.A. Haddad, and H. Caglar, The Binomial QMF-Wavelet Transform for Multiresolution Signal Decomposition, *IEEE Trans. on Signal Processing* Vol. 41, No 1, January 1993, pp. 13-19.