Methods for Regular VLSI Implementations of Wavelet Filters

Andreas Klappenecker*, Volker Baumgarte, Armin Nückel, and Thomas Beth
Universität Karlsruhe
Institut für Algorithmen und Kognitive Systeme,
Am Fasanengarten 5, D–76 128 Karlsruhe, Germany
klappi@ira.uka.de

ABSTRACT

We investigate three approaches to VLSI implementation of wavelet filters. The direct form structure, the lattice form structure, and an algebraic structure are used to derive different architectures for wavelet filters. The algebraic structure exploits conjugacy properties in number fields. All approaches are explained in detail for the Daubechies 4-tab filters. We outline the philosophy of a design method for integrated circuits.

Keywords: Wavelet filter, Daubechies wavelets, integrated circuits, VLSI.

1 INTRODUCTION

We investigate different methods to implement orthonormal wavelet filters as integrated circuits. Many applications of these filters, e. g. in video coding, require high performance and cost effective implementations, which can be achieved by full custom VLSI implementations. Our main interest is to investigate the relation between the mathematical structure of the filters and their physical implementations.

Wavelet filters can be realized in various ways. Basically, we can distinguish two different implementation strategies. In the first strategy, the functionality of the wavelet filters is realized by programming some processor structure, for example a general purpose or a digital signal processor. In the second strategy the data flow graph of the filters is mapped into a dedicated hardwired architecture, for example an application specific circuit. While the first strategy offers great flexibility, it usually does not achieve the high throughput of hardwired architectures and typically has a much higher power consumption (assuming a similar implementation technology). We discuss several techniques that are useful in the latter implementation strategy.

Usually, an application imposes several requirements on the choice of the wavelet filters. For example, it might be desirable to have some vanishing moments, some regularity, small support, etc. In the context of orthonormal wavelet filters, typically a few filters remain as possible candidates for a specific application. Since orthonormality in combination with other constraints leads in general to a rather complicated arithmetic structure of the wavelet coefficients, we can not expect that these filters are directly implementable. Almost always we have to approximate these filters in some way or another. We will see that some approximation techniques lead to elementary circuits that are particularly amenable to automatic routing and placement algorithms.

This paper is organized as follows: In the next section we recall some basic properties of wavelet filters and emphasize some aspects that are relevant for hardware design. The following three sections are devoted to different implementation approaches to wavelet filters. The first approach is based on the direct form structure, the second

* This work was supported by DFG under project Be 887/6-3.

approach is based on the lattice structure, and the last approach exploits conjugacy properties in number fields. We outline the design process and give some layout examples in section 6.

2 SOME WAVELET BASICS

Orthonormal bases of compactly supported wavelets of $L^2(\mathbb{R})$ were introduced by DAUBECHIES in her celebrated paper [4]. These bases are of the form $2^{-j/2}\psi(2^{-j}x-k)$, with $j,k\in\mathbb{Z}$, where ψ is a square integrable, compactly supported, real valued function. It was recently shown by LEMARIÉ-RIEUSSET[14] that these orthonormal wavelet bases can be constructed with the help of a multiresolution analysis, provided that the generating wavelet ψ is continuous.

Recall that a *multiresolution analysis* of $L^2(\mathbb{R})$ is a sequence of nested closed subspaces $V_j\subset V_{j-1}$ of $L^2(\mathbb{R})$ satisfying the following properties: $\bigcap_{j\in\mathbb{Z}}V_j=\{0\}$, the union $\bigcup_{j\in\mathbb{Z}}V_j$ is dense in L^2, the subspaces are linked by $f(x)\in V_j\iff f(2x)\in V_{j-1}$, and the subspace V_0 has an orthonormal basis of the form $\varphi(x-k)$, $k\in\mathbb{Z}$. The function φ is called *scaling function*. We will assume throughout this paper that the scaling function is compactly supported and real valued.

The nesting property of a multiresolution analysis allows to decompose an approximation space V_j into a coarser approximation space V_{j+1} that is orthogonally complemented by a closed vector space W_{j+1} comprising the missing details: $V_j=V_{j+1}\oplus W_j$. It turns out that these details can be described by wavelets. In fact, it can be shown that there exists an orthonormal basis $(\psi_{j,k})$ of W_j that is of the form $\psi_{j,k}=2^{-j/2}\psi(2^{-j}x-k)$, $k\in\mathbb{Z}$, cf. [5,16]. The multiresolution analysis determines the wavelet up to integral shifts and change of sign, see [5, Chapter 8].

The inclusions $V_0\subset V_{-1}$ and $W_0\subset V_{-1}$ lead after appropriate shifts of φ,ψ and possibly a change of sign to the relations

$$\varphi(x)=\sum_{n=0}^{2N-1}h_n\,2\,\varphi(2x-n)\qquad\text{and}\qquad\psi(x)=\sum_{n=0}^{2N-1}g_n\,2\,\varphi(2x-n),$$

where g_n is given by $(-1)^n h_{2N-1-n}$. These two equations play a key rôle in Mallat's Fast Wavelet Transform (FWT) algorithm. If we substitute $2^{-j}x-k$ for x in these two equations, then we obtain

$$2^{-1/2}\varphi_{j,k}=\sum_{m\in\mathbb{Z}}h_{m-2k}\,\varphi_{j-1,m}\qquad\text{and}\qquad 2^{-1/2}\psi_{j,k}=\sum_{m\in\mathbb{Z}}g_{m-2k}\,\varphi_{j-1,m}.\tag{1}$$

The input data for the FWT is an approximation sequence $\mathbf{a}_j=(2^{-j/2}\langle\varphi_{j,k}\mid s\rangle)_{k\in\mathbb{Z}}$. An elementary step of the FWT decomposes $\mathbf{a}_j$ in a coarser approximation sequence $\mathbf{a}_{j+1}$ and a detail sequence $\mathbf{d}_{j+1}$ given by $(2^{-(j+1)/2}\langle\psi_{j+1,k}\mid s\rangle)_{k\in\mathbb{Z}}$. It follows from the equations (1) that the sequences $\mathbf{a}_{j+1}$ and $\mathbf{d}_{j+1}$ can be obtained from $\mathbf{a}_j$ by convolution with the sequences (h_{-n}) and (g_{-n}) respectively, ignoring every second sample in the resulting output sequences.

Briefly, the abstract operations necessary to realize an elementary decomposition step are multiplications, additions, and delays. In the following sections we discuss various methods for fixed-point implementations of wavelet filters in dedicated hardware. It should be noted that concrete realizations of the three abstract operations, for example in CMOS technology, differ significantly in terms of silicon area. By far the most expensive operation is the multiplication. However, since the filter coefficients are fixed, it is always possible to use hardwired additions, subtractions, and shifts instead of universal multiplication units.

Throughout this paper we aim at bit-parallel fixed-point implementations using CMOS technology. Therefore, it seems appropriate to use the number of additions and subtractions to characterize the complexity of a hardwired constant multiplication (it should be noted that fixed shifts are then realized in CMOS technology by metal wires and do not require any active elements; CMOS realizations of adders and subtractors require roughly the same amount of area). Note that this complexity measure is in general not appropriate for digital-serial[17] or serial[7] implementations.

At this point it is instructive to recall the rational parametrization of scaling coefficients of length four or less, which can already be found in DAUBECHIES' paper[4]:

$$h_0(\nu) = \frac{1}{2}\frac{\nu(1+\nu)}{(1+\nu^2)},\ h_1(\nu) = \frac{1}{2}\frac{(1+\nu)}{(1+\nu^2)},\ h_2(\nu) = \frac{1}{2}\frac{(1-\nu)}{(1+\nu^2)},\ h_3(\nu) = \frac{1}{2}\frac{\nu(\nu-1)}{(1+\nu^2)}. \tag{2}$$

Apparently, not every choice of the parameter ν leads to filter coefficients that can be implemented without approximation (e. g. choose a real transcendent parameter ν). We describe three different approaches to bit-parallel fixed-point implementations of scaling and wavelet filter pairs $(h_n), (g_n)$. These approaches differ essentially in the way they approximate these filters.

3 DIRECT IMPLEMENTATION

In signal and image processing applications the input samples $\mathbf{a}_j$ are typically rational or integral. An elementary decomposition step of the FWT convolves this sequence with the scaling filter (h_{-n}) and the wavelet filter (g_{-n}) and then drops every second sample of the resulting two sequences. Clearly, it would not be particularly efficient to implement the decomposition step in such a way. The decimation after the convolution suggests a polyphase implementation[25,27] as in Figure 1, i. e., the input is subdivided into samples of even and odd indices and the resulting two sequences are convolved with the even indexed filter samples and the odd indexed filter samples respectively. One advantage of this architecture is that the critical path is reduced.

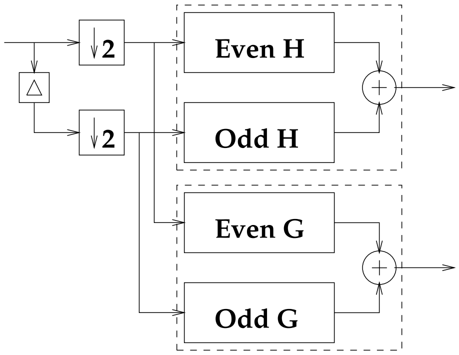

Figure 1: Polyphase realization of an elementary decomposition step. The symbol $\boxed{\Delta}$ denotes a delay.

We discuss in detail three different implementation strategies for the Daubechies 4-tab filters with scaling coefficients*

$$h_0 := \frac{1+\sqrt{3}}{8},\ h_1 := \frac{3+\sqrt{3}}{8},\ h_2 := \frac{3-\sqrt{3}}{8},\ h_3 := \frac{1-\sqrt{3}}{8}. \tag{3}$$

We mentioned in the previous section that multipliers are costly in terms of area and throughput. Of course, it is always possible to avoid multipliers for the multiplication with the fixed coefficients h_n and g_n. For example, the coefficient h_0 can be approximated with a precision of 8 bits by the dyadic rational $87/2^8$ which can also be expressed as the binary value 0.01010111_2. Thus, the multiplier can be simplified to contain only the required product terms, i. e., the multiplication with the constant 0.01010111_2 can be realized by four two-operand adders. The filter coefficient $(h_0, \ldots, h_3)$ can be approximated by the dyadic rationals $(87/2^8, 151/2^8, 41/2^8, -23/2^8)$. The binary expansion of the integer constants 87, 151, 41 and 23, namely

$$87 = 1010111_2,\ 151 = 10010111_2,\ 41 = 101001_2,\ 23 = 10111_2,$$

*Alternatively, we can use the sequence (h_{3-n}). This does not lead to any significant difference.

suggest implementations with $4, 4, 2$ and 3 adders respectively. Note that it is possible to deal with the sign of a filter coefficient in the accumulation of the intermediate results (e.g. use a subtractor instead of an adder). Therefore, the upper dashed box in Figure 1 can be implemented with 16 adders/subtractors and the complete decomposition step with 32 adders/subtractors.

These realizations of multiplications with fixed-point constants were directly induced by the binary number representation. Other number representations can lead to more area efficient multiplication units. For instance, the Canonical Signed Digit (CSD) number representation enjoys a particular popularity in the signal processing and the circuit design communities.[11,18,19,22] Radix-2 signed digit numbers are of the form $\sum b_i 2^i$ with $b_i \in \{-1, 0, 1\}$. A CSD number is a signed digit number with minimal number of non-zero digits such that the product of adjacent digits is always zero. For example, the four constants $87, 151, 41$ and 23 are represented by the following radix-2 CSD numbers:

$$87 = 10\overline{1}0\overline{1}00\overline{1}_{\text{CSD}},\ 151 = 1010\overline{1}00\overline{1}_{\text{CSD}},\ 41 = 101001_{\text{CSD}},\ 23 = 10\overline{1}00\overline{1}_{\text{CSD}},$$

where $\overline{1}$ denotes -1. This number representation now suggests an implementation of the constant multiplication units with $3, 3, 2$ and 2 two-operand adders/subtractors respectively. As a consequence, an 8-bit precision fixed-point arithmetic implementation of an approximation to the Daubechies 4-tab filters with $(3+3+2+2)*2+6 = 26$ adders/subtractors can be derived.

Although the number of digits is minimal in CSD number representations, the resulting arithmetic units are in general not optimal. The methods considered so far realize the operations $x \mapsto ax$, where a is a fixed dyadic rational, by shifting the input x appropriately and adding or subtracting these shifted inputs. However, it is often possible to reduce further the number of additions and subtractions by making use of intermediate results. For example, the composition of two constant multiplication units $x \mapsto a_1 x$ and $x \mapsto a_2 x$ leads to $x \mapsto (a_1 a_2)x$; the complexity in terms of additions and subtractions simply adds up. For instance, the multiplication with the composite number $85 = 5 * 17 = 1010101_2 = 1010101_{\text{CSD}}$ does not require three additions but can be realized with two additions, as follows from $x \mapsto 2^2 x + x$ and $x \mapsto 2^4 x + x$.

It so happens that each of the constants $87, 151, 41$ and 23 in our example can not be realized with less than $3, 3, 2$ and 2 additions/subtractions respectively. However, further optimizations are still possible, if we allow to use terms jointly for different constants.[3,6] This is particularly apparent in the transposed direct form[18,25] of FIR filters, where the input is first multiplied with the filter coefficients and then the intermediate results are appropriately accumulated and delayed, see Figure 2. Obviously, it is often possible to share hardware in the multiplier block. For example, the multiplication of one input sample with the four constants $87, 151, 41$ and 23 does not require 10 additions/subtractions but can be realized with 5 additions/subtractions, since 23 can be realized with two subtractions, $23 = (2^5 - 2^3) - 1$, and this can be used to reduce the complexity for other constants, e.g. $87 = 23 + 2^6$, $151 = 23 + 2^7$, and $41 = 2^6 - 23$. It follows that the polyphase implementation of the 8-bit precision version of the Daubechies 4-tab filters can be realized with 16 additions/subtractions.

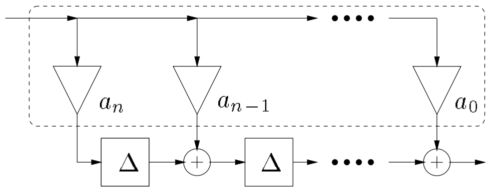

Figure 2: Transposed direct form of an FIR filter. The dashed box indicates the multiplier block.

The optimization problem for the filter block can be stated more formally as follows. Given a set OP of two-operand operators of the form $(a, b) \mapsto 2^p a \pm 2^q b$, with $p, q \in \mathbb{N}_0$, an OP-*chain* for the set of integers $I = \{n_1, \ldots, n_m\}$ is a sequence of integers

$$1 = a_0,\, a_1,\, \ldots,\, a_r \qquad \text{with the property that} \qquad a_k := a_i \diamond a_j,$$

where $\diamond \in$ OP and $i, j < k$ for all $k \in [1..r]$, such that $I \subseteq \{a_0, \ldots, a_r\}$. The length of the OP-chain is r. The OP-*sequence problem* is to find for a given set of integers I an OP-chain with minimal length r (provided it exists). Even if we restrict the set of operators to a single addition, i.e., OP $= \{+\}$, then it can be shown that the corresponding addition-sequence problem is NP-hard.[8] Therefore, it is unlikely that there exist polynomial time algorithms for this problem (this happens only if NP=P). However, this does of course not exclude the existence of efficient algorithms that deliver near-optimal addition chain sequences or more generally near-optimal OP-chain sequences. The interested reader is referred to [1–3,6,13,28] and the references therein.

4 LATTICE IMPLEMENTATION

The polyphase system in Figure 1 is completely determined once the four polyphase filters are known. In other words, this system can be expressed with the help of the polyphase component matrix $E(z)$ by

$$E(z^2)\begin{pmatrix} 1 \\ z^{-1} \end{pmatrix} = \begin{pmatrix} H_{Even}(z^2) & H_{Odd}(z^2) \\ G_{Even}(z^2) & G_{Odd}(z^2) \end{pmatrix}\begin{pmatrix} 1 \\ z^{-1} \end{pmatrix},$$

where $H(z) = H_{Even}(z^2) + z^{-1}H_{Odd}(z^2)$ is the scaling filter and $G(z) = G_{Even}(z^2) + z^{-1}G_{Odd}(z^2)$ is the wavelet filter. For orthogonal wavelet filters this matrix $E(z)$ is *paraunitary,*[24] that is, for all z on the unit circle ($|\, z \,| = 1$) the product $E^t(z^{-1})E(z)$ coincides (up to a fixed scalar factor) with the identity matrix. Therefore, the matrix $E(z)$ allows a (non-unique) factorization into orthogonal 2×2 matrices as follows:[26,25]

$$E(z) = c\, R(\theta_{N-1})D(z)R(\theta_{N-2}) \cdots D(z)R(\theta_0),$$

where c is a scalar factor, $D(z)$ is the diagonal matrix $\mathrm{diag}(1, z^{-1})$, and $R(\theta_k)$ is the 2×2 rotation matrix

$$R(\theta_k) = \begin{pmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{pmatrix}.$$

Assuming $\cos\theta_k \neq 0$ for all $k \in [0..N-1]$, the product can be re-written as

$$E(z) = d\, T(\theta_{N-1})D(z)T(\theta_{N-2}) \cdots D(z)T(\theta_0),$$

where the factor d is given by $c \prod_{k=0}^{n-1} \cos\theta_k$ and $T(\theta_k)$ is the matrix $\begin{pmatrix} 1 & -\tan\theta_k \\ \tan\theta_k & 1 \end{pmatrix}$.

This factorization leads directly to a lattice structured filter bank. Figure 3 shows the Daubechies 4-tab filter in lattice realization. One advantage of the lattice structure is that the number of multiplications is reduced. For example, the lattice implementation shown in Figure 3 can be realized with five multiplications and four additions as contrasted with the eight multiplications and six additions of the direct form implementation.

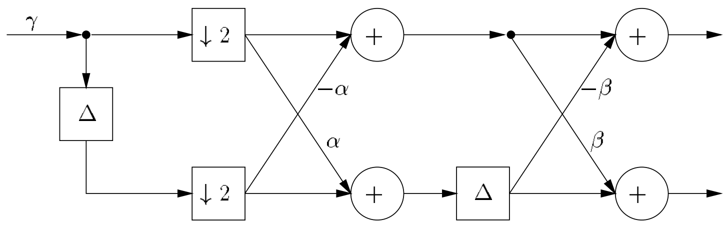

Figure 3: Lattice implementation of the Daubechies 4-tab filters with coefficients $\alpha = \tan(-\pi/3), \beta = \tan(\pi/12)$, and $\gamma = 2^{-1/2}\cos(-\pi/3)\cos(\pi/12)$.

Quantizing the values $\tan\theta_k$ and d to dyadic rationals leads to a lattice structure that is implementable. For example, rounding the values α, β, and γ in Figure 4 to 8-bits precision, leads to the constants $-443/2^8, 69/2^8$, and $87/2^8$. The integer constants $443, 69$, and 87 can be realized with $3, 2$, and 3 adders/subtractors respectively. It follows that it is possible to realize the Daubechies 4-tab filter in lattice form with 17 additions/subtractions.

Alternatively, each rotation can be expressed by elementary factors that are simple to implement.[20,21] For example, the CORDIC algorithm[11] can be used to approximate a matrix $T(\theta_k)$ with a precision of b-bits by a product of the following form:

$$C_b \prod_{k=0}^{b} \begin{pmatrix} 1 & -\sigma_k 2^{-k} \\ \sigma_k 2^{-k} & 1 \end{pmatrix}, \qquad \text{where} \quad \sigma_k \in \{-1, 1\} \quad \text{and} \quad C_b \in \mathbb{R}.$$

The prefactor C_b depends only on the precision b. Clearly, it is possible to use other elementary factors, see for example [10]. We give a layout example for a lattice implementation using elementary rotations in section 6.

5 ALGEBRAIC IMPLEMENTATION

The direct form and the lattice structure were used long before the advent of wavelets. In this section we use yet another approach that was introduced recently by the first author.[12] Recall that the wavelet filter is a mirrored version of the scaling filter with every second sign changed. The main idea of this section is to take advantage of this "symmetry" by use of conjugacy properties in number fields.

Input signal samples are usually integral or rational in signal processing applications. Assume that we want to perform the multiplication with the filter coefficients (3) in an exact way. Then we have to extend the field of rationals to a larger field that contains the quantity $\sqrt{3}$. The smallest field allowing exact calculation with this quantity is given by the quadratic number field $\mathbb{Q}(\sqrt{3}) = \mathbb{Q}(h_0, h_1, h_2, h_3)$.

A scaling filter with algebraic coefficients is said to have the *conjugacy property* iff there exists an automorphism $\sigma \in \mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ that maps the scaling coefficient (h_n) into its "mirrored" form (h_{2N-1+n}). Two different Galois automorphism of the normal extension $\mathbb{Q}(\sqrt{3})/\mathbb{Q}$ are induced by $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, namely the identity and the automorphism $\sigma : \sqrt{3} \mapsto -\sqrt{3}$. Applying σ to the Daubechies filter coefficients shows that this filter satisfies the conjugacy property:

$$\sigma\left(\frac{1+\sqrt{3}}{8}, \frac{3+\sqrt{3}}{8}, \frac{3-\sqrt{3}}{8}, \frac{1-\sqrt{3}}{8}\right) = \left(\frac{1-\sqrt{3}}{8}, \frac{3-\sqrt{3}}{8}, \frac{3+\sqrt{3}}{8}, \frac{1+\sqrt{3}}{8}\right).$$

The number field $\mathbb{Q}(\sqrt{3})$ is in particular a two-dimensional vector space over the rationals. A basis for this vector space is e.g. given by $B = \{1, \sqrt{3}\}$. Expressing the Daubechies scaling filter coefficients (3) in this basis yields:

$$m_0 = (1/8, 1/8),\ m_1 = (3/8, 1/8),\ m_2 = (3/8, -1/8),\ m_3 = (1/8, -1/8).$$

The convolution of the rational input signal with the Daubechies filter leads then to scalar multiplications of input signal samples with the vectors $m_0, \dots, m_3$. Note that the mirrored sequence (h_{3-n}) yields the *same* coefficients with respect to the conjugate basis $\sigma B = \{1, -\sqrt{3}\}$.

We define the vector-valued filter $M(z)$ by $m_0 + m_1 z^{-1} + m_2 z^{-2} + m_3 z^{-3}$. Applying this filter to a (scalar) input sequence yields a vector valued output sequence. In other words, projecting the output of this filter onto its first vector-component yields the same result as the convolution of the input signal with the filter $M_1(z) = 1/8 + 3/8z^{-1} + 3/8z^{-2} + 1/8z^{-3}$. Similarly, the projection onto the second vector-component yields the same output as the convolution of the signal with the filter $M_2(z) = 1/8 + 1/8z^{-1} - 1/8z^{-2} - 1/8z^{-3}$.

Let us write the filter $M(z)$ in polyphase form: $M(z) = M_{Even}(z^2) + z^{-1}M_{Odd}(z^2)$. The output of the filter $M_{Even}(z^2) + z^{-1}M_{Odd}(z^2)$ gives the scaling filtered signal, provided we interpret the vector-valued output as numbers of $\mathbb{Q}(\sqrt{3})$ with respect to the basis $B = \{1, \sqrt{3}\}$. Analogously, the output of $M_{Even}(z^2) - z^{-1}M_{Odd}(z^2)$ yields the wavelet filtered signal, if we interpret the vector-valued output coefficients as numbers of $\mathbb{Q}(\sqrt{3})$ with respect to the basis $\sigma B = \{1, \sqrt{3}\}$. The dashed box in Figure 4 shows how this can be implemented efficiently. This architecture for algebraic wavelet filters with conjugacy property (which are of course Smith-Barnwell[23] QMFs) resembles the polyphase implementation of Esteban-Galand[9] QMFs.

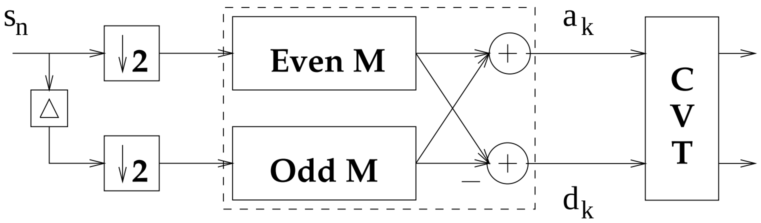

Figure 4: Algebraic wavelet QMFs satisfying the conjugacy property can be implemented with vector-valued filters.

We can view the architecture in Figure 4 as a black box that gets a rational signal sequence as input and outputs rational scaling and wavelet filtered signal sequences. In Figure 4, the back conversion of a_k and d_k to rationals is done in the box CVT. As mentioned before, the results a_k and d_k are represented with respect to the bases B and σB. Therefore, the vectors $a_k = (a_{k,1}, a_{k,2}) \in \mathbb{Q}$ and $d_k = (d_{k,1}, d_{k,2}) \in \mathbb{Q}$ represent the numbers

$$(a_{k,1}, a_{k,2})(1, \sqrt{3})^t = a_{k,1} + \sqrt{3}\,a_{k,2} \qquad \text{and} \qquad (d_{k,1}, d_{k,2})(1, -\sqrt{3})^t = d_{k,1} - \sqrt{3}\,d_{k,2}.$$

A rational approximation to these numbers is used in finite precision implementations. The back conversion unit CVT can be realized with four constant multiplication units and two additions/subtractions.

For example, rounding $\sqrt{3}$ to 8-bits precision yields the constant $443/2^8$. A multiplication with this constant can be realized with 3 additions/subtractions. Therefore, the back conversion unit in our example can be realized with $(1 + 3) + (1 + 3) = 8$ additions/subtractions. It is easy to see that an implementation of $M(z)$ can be achieved with 10 additions/subtractions. Consequently, we can realize the complete decomposition with 18 additions/subtractions.

It is possible to improve this by choosing another basis of the field extension $\mathbb{Q}(\sqrt{3})/\mathbb{Q}$. For example, if we use the basis $B = \{1, (1 + \sqrt{3})/8\}$, then we obtain the filter coefficients:

$$m_0 = (0, 1),\ m_1 = (1/4, 1),\ m_2 = (1/2, -1),\ m_3 = (1/4, -1).$$

The dashed box can then be implemented with 7 additions/subtractions. Rounding the elements $(1 + \sqrt{3})/8$, $(1 - \sqrt{3})/8$ to 8-bit precision yields the constants $87/2^8$ and $-23/2^8$. The multiplication with these constants can be implemented with 3 and 2 additions/subtractions. Thus, the back conversion unit can be realized with 7 additions/subtractions. This gives 14 additions/subtractions in total for the complete decomposition.

Note that the output of the wavelet filter changes only every second clock cycle. Interleaving the output of the scaling and the wavelet filter leads to further savings. The two additions and the two subtraction in the dashed box can be replaced by two switchable adder/subtractor units. The back conversion unit can be reduced using a module realizing both $x \mapsto (1 \pm \sqrt{3})/8$. This can be implemented with 3 additions/subtractions using a switchable adder/subtractor unit. This reduces the complexity of the back conversion unit to 4 additions/subtractions. As a consequence, the complete decomposition with interleaved scaling and wavelet filter output can be realized with 9 additions/subtractions.

At first sight, the method presented in this section seems to exploit peculiar properties of this very example. However, it can be shown[12] that each orthonormal wavelet filter pair can be approximated with arbitrary precision by wavelet filters allowing an implementation of this type. Thus, the situation is very similar to lattice implementations, where each orthonormal wavelet filter can be approximated with arbitrary precision by conjugate quadrature filters having "rotation" matrices $T(\theta_k)$ with dyadic rational entries. The algebraic structure allows the same optimization techniques as the direct form implementation. Moreover, since a change of base may lead to filter coefficients with lower complexity in terms of additions and subtractions, it is often possible to obtain further savings. As we have seen, the overhead of the back conversion unit is rather small. Allowing interleaved output, this overhead can even be reduced further.

6 MATHEMATICALLY INFLUENCED CIRCUIT DESIGN

In signal processing, hardware modules and their syncronization are strongly related to the algebraic structure of the implementations' mathematical specification. We feel that a thorough understanding of the algebraic structures of these specifications may greatly influence the high level hardware design process. The relation between mathematical structures and algorithmic architectural consequences is the key topic in algebraic algorithm synthesis. This mathematical algorithm and architecture engineering approach can be used to define topics like the structure and the synchronization of hardware oriented algorithms. With the appearance of high level design tools, it became possible to integrate "intelligent" libraries within a hardware design environment. It is widely acknowledged in the hardware design community that the use of abstraction to describe hardware structures is of special importance. Typically, aspects like simulation performance, documentation or design interchange are mentioned in the literature. The use of high level Hardware Description Languages (HDLs) leads through various transformations and optimizations from an algorithmic description to a physical implementation, see Figure 5.

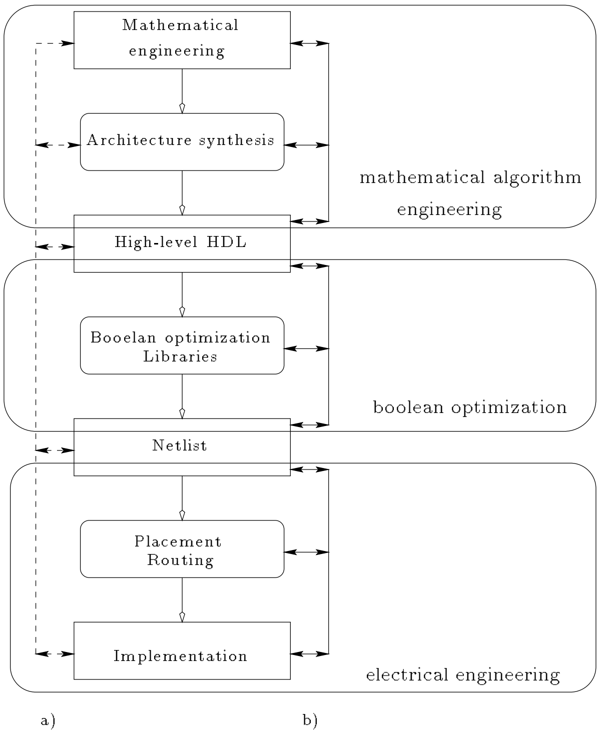

Figure 5: This figure characterizes the mathematical algebraic algorithm synthesis approach as it is supported by nowadays commonly used design environments, dotted lines a), and the idealized required situation b).

A high level HDL supports the development of implementation technology independent models, which is crucial for the mathematical, algebraic design approach. Technology independent high level HDLs support the implementation of "intelligent" libraries that can be used without any detailed knowledge about the target technology. Thus, this design style allows the user to concentrate on the relationship between architectures, mathematical structures and the corresponding algorithmic optimization.

Unfortunately, many commercially used HDLs do not consequently support this design style. For example, several languages do not guarantee that the simulated behavior of a high level model is equivalent to the behavior of the synthesized circuit. This makes a mathematical design approach very difficult, since it is very hard to modularize the design flow. The dotted line on the left of Figure 5 characterizes the badly modularized design flow. From the viewpoint of informatics, some commonly used combinations of HDLs and corresponding synthesis and CAD tools already contain inconsistencies within the HDL-definition. This is not very critical, if the designer is a highly educated electrical engineer, since such a user is able to understand the details of the technology specific implementation. The mathematical educated user will run in major problems using badly defined languages. If an implementation, e.g. a CMOS-Chip, is shown to be incorrect, it is hard to decide at which step of the technology refinement process an error was occurring.

The idealized design flow is shown in Figure 5 b). This approach is characterized by a demand to a design environment, namely that there is a clear interface between the mathematical engineering task and the aspects concerning boolean optimization and electrical engineering. Clearly, the simulated behavior based on an high level HDL should be equivalent to the circuits behavior in any case.

For example, consider a lattice architecture as discussed in section 4. Figure 6 shows an example for such an architecture that was proposed recently by Rieder, Gerganoff, Götze, and Nossek.[20] The rotations are already in a factored form that is directly amenable to implementation.

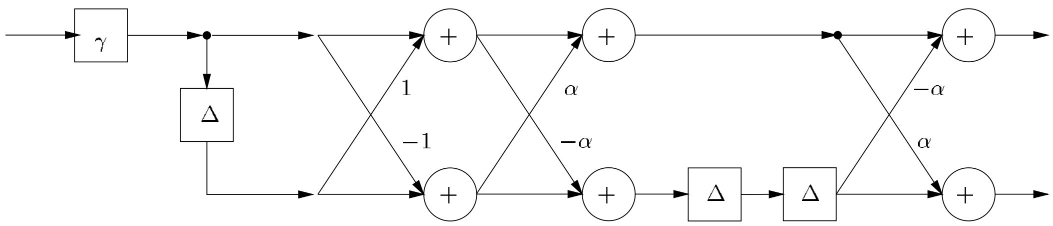

Figure 6: Lattice structure of an approximation to the Daubechies 4-tab filter using elementary rotations as proposed in [21]. The value of α is 2^{-2}.

However, the mathematical methods in architecture development are not able to replace the engineering process. Figure 8 shows the result of an automatic place and route algorithm. The result of geometric optimization of the same architecture is Figure 7. Numerous NP-hard problems make it difficult to automatize the layout process completely,[15] so that geometric and electrical optimization is still the domain of highly skilled persons.

7 CONCLUSION

We discussed different architectures for wavelet filters that are suitable for implementation in dedicated hardware. Using an internal precision of 8-bits, we showed that the direct form structure, the lattice structure, and the algebraic structure of the Daubechies 4-tab filter can be implemented with 16, 17, and 14 additions/subtractions respectively. In general, the direct and the algebraic structure is well-suited for high-precision implementation, whereas the lattice structure leads to bit-slice architectures that are advantageous for low-precision implementations. We decided to proceed by example, so that our description of the various architectures is detailed enough. We outlined in section 6 the design flow for the mathematical influenced engineering of integrated circuits.

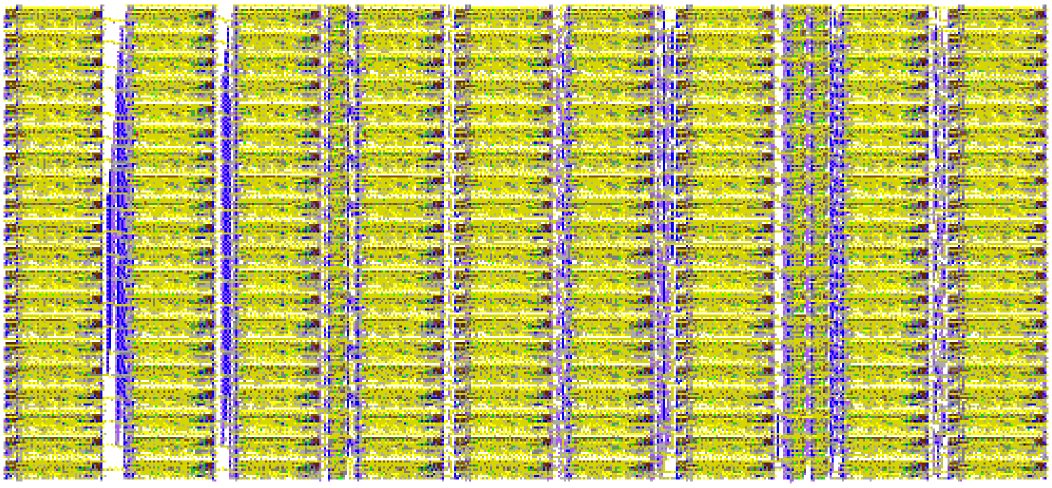

Figure 7: Lattice form implementation using a 1μ dual metal CMOS process. Adders and subtractors clearly dominate the overall area. The input is on the left. From left to right: a constant multiplication (3 additions and subtractions), a register, two rotations (with two additions/subtractions), two registers, one rotation.

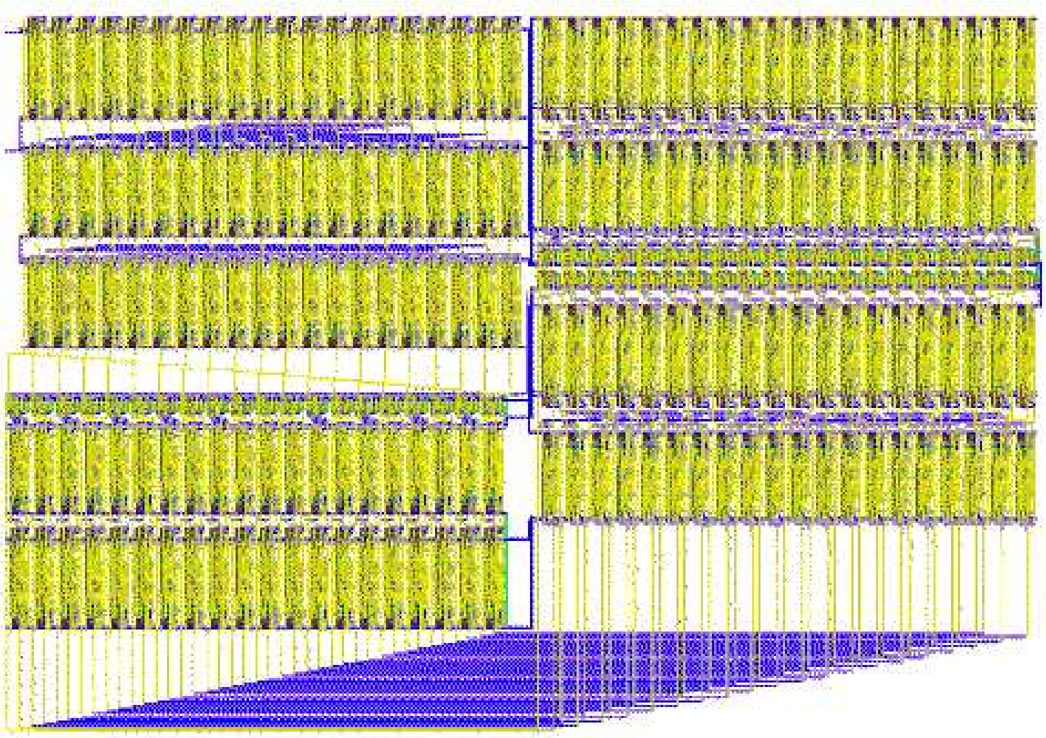

Figure 8: The result of a standard automatic place and route algorithm for the architecture shown in Figure 6.

Acknowledgements

We wish to thank Professor Nossek and his group for their fruitful cooperation. In particular, we thank Peter Rieder and Sven Simon for many helpful discussions.

8 REFERENCES

[1] F. Bergerson, J. Berstel, S. Brlek, and C. Duboc. Addition chains using continued fractions. *J. of Algorithms*, 10:403–412, 1989.

[2] R. Bernstein. Multiplication by integer constants. *Software – Practice and Experience*, 16(7):641–652, 1986.

[3] D. R. Bull and D. H. Horrocks. Primitive operator digital filters. *IEE Proceedings*, 138(3):401–412, 1991.

[4] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

[5] I. Daubechies. *Ten Lectures on Wavelets.* CBMS-NSF Reg. Conf. Series Appl. Math. SIAM, 1992.

[6] A. G. Dempster and M. D. Macleod. Use of minimum-adder multiplier blocks in FIR digital filters. *Trans. Circuits and Systems II*, 42(9):569–577, 1995.

[7] P. B. Denyer and D. Renshaw. *VLSI Signal Processing: A Bit-Serial Approach.* Addison-Wesley, 1985.

[8] P. Downey, B. Leong, and R. Sethi. Computing sequences with addition chains. *SIAM J. Computing*, 10(3):638–646, 1981.

[9] D. Esteban and C. Galand. Application of quadrature mirror filters to split-band voice coding schemes. In *1977 IEEE Int. Conf. ASSP*, pages 191–195, 1977.

[10] J. Götze, S. Paul, and M. Sauer. An efficient Jacobi-like algorithm for parallel eigenvalue computation. *IEEE Trans. on Computers*, 42(9), 1993.

[11] K. Hwang. *Computer Arithmetic – Principles, Architecture, and Design.* John Wiley & Sons, 1979.

[12] A. Klappenecker. Algebraic wavelet filters. *Submitted*, 1996.

[13] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, 1981.

[14] P.-G. Lemarié-Rieusset. Ondelettes generalisées et fonctions d'échelle à support compact. *Rev. Mat. Iberoamericana*, 9(2):333–371, 1993.

[15] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* J. Wiley & Sons and B. G. Teubner, 1990.

[16] S. Mallat. Multiresolution approximation and wavelets. *Trans. Am. Math. Soc.*, 315:69–88, 1989.

[17] K. K. Parhi and T. Nishitani. VLSI architectures for discrete wavelet transforms. *IEEE Trans. VLSI Systems*, 1(2), 191-202 1993.

[18] P. Pirsch. *Architekturen der digitalen Signalverarbeitung.* B. G. Teubner Stuttgart, 1996.

[19] G. W. Reitwiesner. Binary arithmetic. In F. L. Alt, editor, *Advances in Computers*, volume 1, pages 231–308. Academic Press, 1960.

[20] P. Rieder, K. Gerganoff, J. Götze, and J. A. Nossek. Parametrization and implementation of orthogonal wavelet transforms. *Proc. Int. Conf. on Acoustic, Speech and Signal Processing, Atlanta*, 1996.

[21] P. Rieder, J. Götze, and J. A. Nossek. Efficient implementation of orthogonal wavelet transforms. *IEEE Trans. on Circuits and Systems II*, 1996. To appear.

[22] H. Schmeck. *Analyse von VLSI-Algorithmen*. Spektrum Akademischer Verlag GmbH, 1995.

[23] M. J. T. Smith and T. P. Barnwell. Exact reconstruction techniques for tree structured subband coders. *IEEE ASSP*, 34:434–441, 1986.

[24] P. P. Vaidyanathan. Quadrature mirror fitler banks, M-band extensions and perfect-reconstruction techniques. *IEEE ASSP Magazine*, pages 4–20, 1987.

[25] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[26] P. P. Vaidyanathan and P. Q. Hoang. Lattice structures for optimal design and robust implementation of two-channel perfect reconstruction filter banks. *IEEE Trans. Acoust., Speech, Signal Processing*, 36:81–94, 1988.

[27] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall, 1995.

[28] Y. Wu. Strength reduction of multiplications by integer constants. *ACM SIGPLAN Notices*, 30(2):42–48, 1995.