DESIGN OF A FLEXIBLE VLSI ARCHITECTURE FOR M-CHANNEL FILTER BANK LIFTING FACTORIZATIONS

Ruben Bartholomä, Thomas Greiner, Frank Kesel

Pforzheim University of Applied Sciences, Dept. of Engineering
Tiefenbronnerstr. 65, 75175 Pforzheim, Germany
{ruben.bartholomae, thomas.greiner, frank.kesel}@fh-pforzheim.de

ABSTRACT

In this paper we present an efficient VLSI architecture focusing on M-channel multirate filter banks using the lifting scheme. Since the M-channel lifting factorization results in a signal flow graph with a variable structure, the architecture must have a high degree of flexibility to allow the implementation of basically different lifting factorizations. The proposed architecture is convenient to realize arbitrary lifting factorizations on a variable amount of arithmetic resources, leading to an adaptability for the real-time requirements of various DSP applications.

1. INTRODUCTION

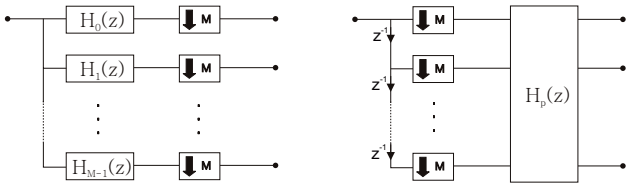
Due to the reduction of computational complexity, the lifting scheme [1], [2] enables an efficient realization of multirate filter banks used in applications like signal analysis and image compression. Many 2-channel lifting based VLSI architectures have been proposed [3], [4], [5]. Although the 2-channel decomposition has been largely investigated in the recent years, the M-channel decomposition [6] has been neglected up to now. Hence, there are no concepts on VLSI architectures for M-channel lifting factorizations available.

We present a VLSI architecture that is convenient for variable signal flow graphs (SFGs), resulting from M-channel lifting factorizations. Due to the fact that the lifting factorization is not a unique process and there exist different factorization strategies, the structure of the resulting SFGs may change, which has to be taken into account by the concept of the architecture. Moreover, our architecture has the ability to adjust the trade-off between resource utilization and maximum data throughput, resulting in an adaptability for the real-time requirements of different DSP applications.

The paper is organized as follows. In Section 2 a brief introduction into the M-channel lifting factorization is given. Section 3 describes the concept of the proposed VLSI architecture and in section 4 we present results of the FPGA implementation of the proposed architecture. A summary is given in section 5.

2. M-CHANNEL LIFTING FACTORIZATION

M-channel multirate filter banks for signal analysis are typically represented by M different filters $H_0(z)$ to $H_{M-1}(z)$ as shown in figure 1(a). Lifting factorization is based on the



(a) (b)

Fig. 1. (a) structure of a M-channel multirate filter bank and (b) its corosponding polyphase representation

representation of this filter bank in its polyphase structure as shown in figure 1(b) using the polyphase matrix of equation 1.

$$\mathbf{H}_p(z) = \begin{pmatrix} H_{0,0}(z) & H_{0,1}(z) & \cdots & H_{0,M-1}(z) \\ H_{1,0}(z) & H_{1,1}(z) & \cdots & H_{1,M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1,0}(z) & H_{M-1,1}(z) & \cdots & H_{M-1,M-1}(z) \end{pmatrix} \quad (1)$$

The basic idea of the lifting scheme is to factor the polyphase matrix into simpler upper and lower triangular matrices, which are referred to as lifting steps. A M-channel lifting step from channel j to i is defined by a lifting operator $\mathbf{\Lambda}_{i,j}[\lambda(z)]$:

$$\mathbf{\Lambda}_{i,j}[\lambda(z)] = \mathbf{I} + \lambda(z) \cdot \mathbf{e}_i \cdot \mathbf{e}_j^T, \quad (2)$$

where $\mathbf{I}$ is the $M \times M$ identity matrix, $\mathbf{e}_i$ denotes the i-th $M \times 1$ unit vector with $i, j \in \{0, 1, \cdots, M-1\}$ and $i \neq j$; $\lambda(z)$ denotes the corresponding FIR lifting polynomial. Figure 2 shows two kinds of SFGs resulting from a lifting step. The lifting scheme can be obtained by a re-

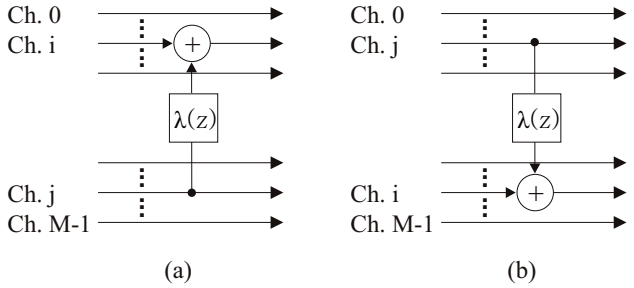


Fig. 2. SFG of a lifting step $\mathbf{\Lambda}_{i,j}[\lambda(z)]$ resulting from (a) an upper triangular matrix with $i < j$ and (b) a lower triangular matrix with $i > j$

peated extraction of lifting steps $m, n \in \mathbb{N}$ from the polyphase matrix, until the polyphase matrix has reduced to a diagonal matrix containing delay elements on its diagonal. This factorized representation of the polyphase matrix may have the advantages, that it requires less arithmetic operations and less delay elements to complete the calculation.

Next it will be explained how to extract lifting steps from a polyphase matrix. In order to clarify this iterative process, we use bracketed superscripts in polynomials and matrices as iteration counters. The extraction of a lifting step requires a polynomial division of two elements, selected either from a row or a column of the polyphase matrix $\mathbf{H}_p^{(m)}(z)$. Since different approaches are known to select these two elements, there exist principally different ways to extract lifting steps. The basic procedure is given as follows.

Choosing the element $H_{k,j}^{(m)}(z)$ as dividend and the element $H_{k,i}^{(m)}(z)$ as divisor for the polynomial division, gives result $\lambda^{(m+1)}(z)$ and remainder $H_{k,j}^{(m+1)}(z)$ of polynomial division as denoted in equation 3.

$$\frac{H_{k,j}^{(m)}(z)}{H_{k,i}^{(m)}(z)} = \lambda^{(m+1)}(z) + \frac{H_{k,j}^{(m+1)}(z)}{H_{k,i}^{(m)}(z)} \tag{3}$$

After determination of the result $\lambda^{(m+1)}(z)$ and the remainder $H_{k,j}^{(m+1)}(z)$ the remainders $H_{r,j}^{(m+1)}(z)$ of the remaining rows r whereas $r \in \{0, 1, \ldots, M-1\} \backslash \{k\}$ have to be reformulated in a manner as mentioned in 3. Since $\lambda^{(m+1)}(z)$ is already known the remainders $H_{r,j}^{(m+1)}(z)$ can be obtained using equation 4.

$$H_{r,j}^{(m+1)}(z) = H_{r,j}^{(m)}(z) - \lambda^{(m+1)}(z) \cdot H_{r,i}^{(m)}(z) \tag{4}$$

Finally, the $(m+1)$-th lifting step $\mathbf{V}^{(m+1)}(z)$ can be determined by the lifting operator $\mathbf{\Lambda}_{i,j}[\lambda^{(m+1)}(z)]$ with the result of polynomial division $\lambda^{(m+1)}(z)$ as shown in equation 5

$$\mathbf{V}^{(m+1)}(z) = \mathbf{\Lambda}_{i,j}[\lambda^{(m+1)}(z)], \tag{5}$$

where i denotes the column index of the divisor and j is the column index of the dividend. Hence, the previous polyphase matrix $\mathbf{H}_p^{(m)}(z)$ can be expressed by the new polyphase matrix $\mathbf{H}_p^{(m+1)}(z)$ and the lifting step $\mathbf{V}^{(m+1)}(z)$ as shown in equation 6.

$$\mathbf{H}_p^{(m)}(z) = \mathbf{H}_p^{(m+1)}(z) \cdot \mathbf{V}^{(m+1)}(z) \tag{6}$$

The described method can also be applied if divisor and dividend are selected from a column. In this case the $(n+1)$-th lifting step $\mathbf{W}^{(n+1)}(z)$ appears to the left hand side of the polyphase matrix as denoted in equation 7 and 8

$$\mathbf{W}^{(n+1)}(z) = \mathbf{\Lambda}_{i,j}[\lambda^{(n+1)}(z)] \tag{7}$$

$$\mathbf{H}_p^{(n)}(z) = \mathbf{W}^{(n+1)}(z) \cdot \mathbf{H}_p^{(n+1)}(z), \tag{8}$$

where j denotes the row index of the divisor and i is the row index of the dividend.

Equation 9 shows the factorized form of the original polyphase matrix $\mathbf{H}_p^{(0)}(z)$ with m right hand side and n left hand side triangular matrices using equation 6 and 8 respectively. $\mathbf{H}_p^{(m+n)}(z)$ is the remaining polyphase matrix after $m+n$ lifting steps.

$$\begin{aligned}\mathbf{H}_p^{(0)}(z) = \mathbf{W}^{(1)}(z) \cdot \ldots \cdot \mathbf{W}^{(n)}(z) \cdot \mathbf{H}_p^{(m+n)}(z) \cdot \\ \cdot \mathbf{V}^{(m)}(z) \cdot \ldots \cdot \mathbf{V}^{(1)}(z)\end{aligned} \tag{9}$$

In [6] it is shown that each M-channel filter bank with polyphase matrix $\mathbf{H}_p(z)$ with $det(\mathbf{H}_p(z)) = z^{-k}$, $k \in \mathbb{Z}$ can be decomposed into lifting steps, whereas the remaining polyphase matrix reduces to a diagonal matrix containing only delay elements on its diagonal. Considering the computational complexity, it may be more efficient to keep a polyphase matrix in diagonal form with FIR polynomials on its diagonal as mentioned in equation 10.

$$\mathbf{H}_p^{(m+n)}(z) = diag\left(\left[H_{0,0}^{(m+n)}(z), \ldots, H_{M-1,M-1}^{(m+n)}(z)\right]\right) \tag{10}$$

$diag$ represents a $M \times M$ diagonal matrix with the diagonal elements $H_{i,i}^{(m+n)}(z)$ (with $i \in \{0, 1, \ldots, M-1\}$), obtained as remainders from the factorization process. Each diagonal element can be represented by a SFG as shown in figure 3.

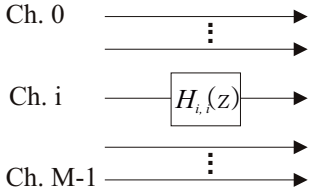


Fig. 3. SFG of the i-th diagonal element of the remaining polyphase matrix

3. VLSI ARCHITECTURE

In this section we present a VLSI architecture that is suitable for M-channel multirate filter banks, which are decomposed in a manner as formulated in equation 9. The SFG of a specific lifting factorization can be obtained by cascading the SFGs of lifting steps (figure 2) and diagonal elements (figure 3) according to equation 9. Since the structure of this SFG is variable and changes for different factorizations, the architecture has to be very flexible in order to realize arbitrary lifting factorizations.

3.1. Design Method

The concept of the proposed architecture is based on the partitioning of the SFG, followed by mapping the SFG partitions onto several so called Lifting Processing Elements (LPEs) as shown by the example of figure 4. The LPEs

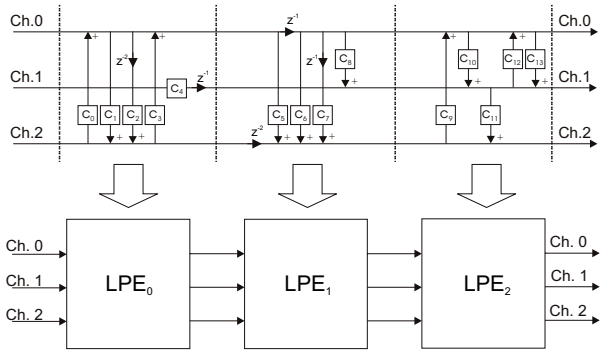


Fig. 4. Mapping of a SFG obtained by a factorized 3-channel filter bank onto a LPE array consisting of 3 LPEs

are chained to form a one dimensional array, where each LPE is receiving partial results from its predecessor. Such kind of configuration can be considered as a coarse grained pipeline, where the number of pipeline stages equals the number of LPEs.

Since the amount of available LPEs is a variable parameter, the architecture features a scalability with respect to the maximum data throughput. By increasing the number of available LPEs, the arithmetic unit of one LPE has to process fewer operations of the whole SFG, which finally results in a higher data throughput.

The partitioning of a SFG requires to formulate the SFG as a sequence of so called LPE instructions. LPE instructions are considered as built-in instruction codes, which control the data paths. The finite state machine of a LPE is implicitly defined by looping through the instruction sequence of the LPE. A lifting step $\mathbf{\Lambda}_{i,j}[\lambda(z)]$ containing a polynomial $\lambda(z)$ with $n \in \mathbb{N}$ coefficients can be realized by n LPE instructions. For this purpose the lifting step $\mathbf{\Lambda}_{i,j}[\lambda(z)]$ has to be decomposed into n lifting steps with only one polynomial coefficient, since each LPE instruction can process a single coefficient. Equation 11 shows this decomposition step.

$$\mathbf{\Lambda}_{i,j}[\lambda(z)] = \prod_{k=0}^{n} \mathbf{\Lambda}_{i,j}[c_k \cdot z^k] \text{ , with } \lambda(z) = \sum_{k=0}^{n} c_k \cdot z^{-k} \quad (11)$$

With consideration of the SFGs data dependencies, the set of all LPE instructions is distributed among all LPEs, while each LPE processes the same number of LPE instructions. If the total number of LPE instructions is an integer multiple of the number of available LPEs, the architecture performs an optimal resource utilization, since all arithmetic units of the data path are busy in each clock cycle. Using this design technique enables the necessary flexibility to adapt a LPE to any mapped SFG partition.

3.2. LPE Design

A LPE basically consists of one multiplier, one adder, a variable amount of registers and a memory for LPE instructions [7]. Figure 5 shows the architecture of a LPE, whose elements are briefly discussed below.

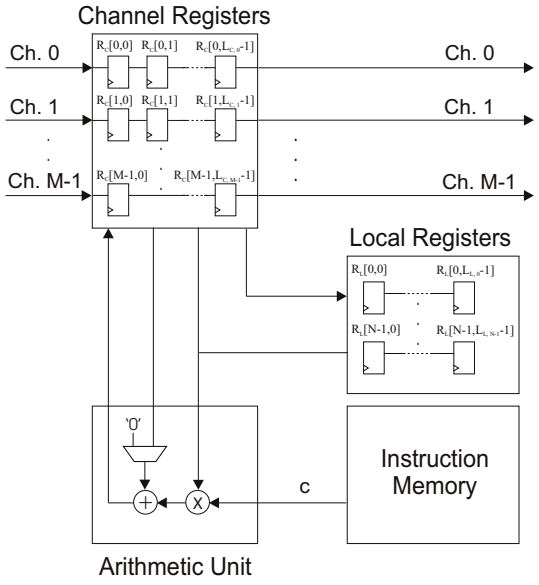


Fig. 5. Basic architecture of a LPE

The registers of a LPE are classified into channel registers $R_C[i,j]$ and local registers $R_L[k,l]$, whereas each category consists of several shift register banks subscripted by i and k respectively. Channel registers are embedded into the data path of the channels. They are used for communication between the neighbouring LPEs and for the realization of delay elements, resulting from the residual polyphase matrix. Local registers store partial results, which

are only needed within the particular LPE. They are used to realize delay elements of lifting steps. The actual length of the shift register banks $L_{C,i}$ and $L_{L,k}$ and the total number of local shift register banks N in each LPE is adapted to its mapped SFG partition.

As already mentioned, the sequential behaviour of a LPE is defined by a sequence of LPE instructions, which are stored in the instruction memory. The instruction code for each LPE instruction is mainly formed by an opcode opc, several register addresses and a constant c embedding a polynomial coefficient. The five distinct types of LPE instructions are shown in table 1, whereas b refers any channel register and a refers any channel or local register. The instruc-

opc	Arithmetic Operation	Parallel Shift Operation
MAC	$b \leftarrow b + a * c$	NO
MUL	$b \leftarrow a * c$	NO
MAC_SFT	$b \leftarrow b + a * c$	YES
MUL_SFT	$b \leftarrow a * c$	YES
NOP	—	NO

Table 1. LPE instructions

tion MUL performs a coefficient multiplication and MAC performs a coefficient multiplication followed by an addition. MAC_SFT and MUL_SFT are extended versions of MAC and MUL, which concurrently shift a partial result from a channel register into a local shift register bank. Lifting steps are realized by MAC and MAC_SFT instructions. MUL and MUL_SFT are used for implementing polynomials of the residual polyphase matrix. NOP instructions have to be inserted into the instruction sequence of a LPE, if the number of allocated LPEs does not divide the total number of LPE instructions.

4. RESULTS

To demonstrate the efficiency and flexibility of our architecture, we present some results about the FPGA implementation of a lifting factorization, resulting from a 3-channel multirate filter bank [6]. Due to the lifting factorization, the computational complexity reduced from 24 to 14 multiplications compared to a non-factorized filter bank. The filter bank was implemented on a Xilinx Virtex II FPGA with a 18 bit data path for coefficients and intermediate results. Table 2 summarizes the results. We have realized the lifting factorization with different number of LPEs to show the adaptability respecting the maximum data throughput of our architecture. The data throughput shown in table 2 concerns the sum of all three channels and resource usage is measured in terms of gate equivalent counts, as reported by the place and route tool.

# LPE	# Register	Resources / # GE	Throughput / MSPS
1	10	8231	18.2
2	13	13371	38.8
3	16	18215	53.7
4	19	22953	69.6
5	22	27681	96.5
7	28	37238	146.1

Table 2. Results of 3-channel filter bank architectures

5. CONCLUSION

We have presented a VLSI architecture based on lifting factorizations for M-channel multirate filter banks. The architecture is capable to implement arbitrary lifting factorizations on a variable amount of arithmetic resources. Moreover, we have presented results about the implementation of the proposed architecture on a Xilinx Virtex II FPGA.

6. REFERENCES

[1] I. Daubechies, W. Sweldens, *Factoring wavelet transforms into lifting step*, Tech. Rep., Bell labs, 1996

[2] W. Sweldens, *The lifting scheme - A construction second generation wavelets*, SIAM J. Math. Anal, vol. 29, no. 2, pp. 511, 1997

[3] K. Andra, C. Chakrabarti, T. Acharya, *A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform*, IEEE Transactions on Signal Processing, vol. 50, no. 4, pp. 966, April 2002

[4] P.-Y. Chen, *VLSI Implementation for One Dimensional Multilevel Lifting-Based Wavelet Transform*, IEEE Transactions on Computers, vol. 53, no. 4, pp. 386, April 2004

[5] C.-T. Huang, P.-C. Tseng, L.-G. Chen *Efficient VLSI Architectures of Lifting based Discrete Wavelet Transform by Systematic Design Method*, IEEE International Symposium on Circuits and Systems, vol. 5, pp. 565, 2002

[6] Y.-J. Chen, K. Amaratunga *M-Channel Lifting Factorization of Perfect Reconstruction Filter Banks and Reversible M-Band Wavelet Transforms*, IEEE Transactions on Circuits and Systems II, Analog and Digital Signal Processing, vol. 50, no. 12, pp. 963, December 2003

[7] R. Bartholomä, T. Greiner, F. Kesel *A Lifting based VLSI Architecture for M-Channel Multirate Filter Banks*, Tech. Rep., Pforzheim University, July 2004