FAST IMPLEMENTATION OF ORTHOGONAL WAVELET FILTERBANKS USING FIELD-PROGRAMMABLE LOGIC

U. Meyer-Baese, J. Buros, W. Trautmann, and F. Taylor

High Speed Digital Architecture Laboratory, University of Florida
Gainesville 32611-6130, U.S.
e-mail: {uwe,buros,wtraut,fjt}@alpha.ee.ufl.edu

ABSTRACT

Field-Programmable Logic (FPL) is on the verge of revolutionizing digital signal processing (DSP) in the manner that programmable DSP microprocessors did nearly two decades ago. While FPL densities and performance have steadily improved to the point where some DSP solutions can be integrated into a single FPL chip, they still have limited the use in high-precision high-bandwidth applications. In this paper it is shown that alternative implementation strategies can be found which overcome the precision/bandwidth barrier. The design of Daubechies length 4 and 8 filter is presented to compare FPL and programmable DSP solutions.

1. INTRODUCTION

FPLs appear in two forms, field programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs). FPGAs are fine grain devices consisting of small logic elements (LE) (e.g., Xilinx XC4000) and various different routing canals (short, local, and long-lines). CPLDs have larger logic blocks and fast busses connecting these array blocks (e.g., Altera FLEX [1]). The historical advantage of FPLs has been their "in circuit programmability" and support of "rapid prototyping." FPLs have been promoted in custom computing machine (CCMs) applications where they have been reported to achieve speed-up-factors ranging from 10-1000 compared with conventional workstations [2, Table 1]. FPLs provide DSP arithmetic support with fast carry chains (Xilinx XC4000, Altera FLEX) which are used to implement multiply-accumulates (MACs) at relatively high speeds. The designs using FPL typically exploited latent:

- parallelism: implementing multiple MAC calls
- efficiency: zero product-terms are removed
- pipelining: each LE has a register, therefore pipeline requires no additional resources

FPLs, nevertheless, currently have limited applicability in high-bandwidth high-precision applications. The geometric increase in area requirements for constant-speed arithmetic is a very challenging FPL design issue. Since DSP is acknowledged to be arithmetic-intensive, this presents an adoption barrier.

An emerging arithmetic-intensive DSP area is wavelet signal processing. In contrast to Fourier transforms, wavelets provide a far more robust representation of a dynamically changing signal or image. The orthogonal wavelet filter, developed by I. Daubechies, has motivated many contemporary wavelet-centric applications. For instance, the European Space Agency (ESA) distributes a collection of compress (60:1, without visible loss) satellite images on two CD-ROMs with 13K images covering a full year. ESA uses a maximum error compression scheme [3] whose most computational intensive component consists of a three stage DWT with Daubechies length 4 filter. The next generation of satellites the images size increases to 10K×20K pixel, which demands high performance DWT hardware solutions. The ability to implement such solutions using FPGAs, will offer the DSP community important new opportunities. This presumes, of course, that solutions can be realized which meet exacting bandwidth and precision requirements.

U. Meyer-Baese was supported by an European Space Agency fellowship.

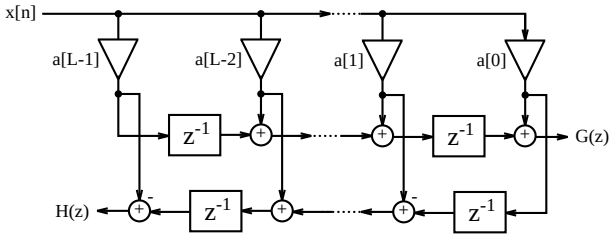

Figure 1: Orthogonal two channel filter bank in transporsed FIR form.

Formally, a Daubechies filter has compact support (finite length), provides perfect reconstruction in a two channel filter bank scheme, and fulfills the conjugate quadrature filter (CQF) property, i.e. $H(z) = z^{-N}G(-z^{-1})$. Using existing FPLs, the filter can be realized in a "unwrapped" fashion, preferred in the transposed FIR form to achieve maximum speed. In the two channel filter bank case each filter can be realized using a polyphase decomposition, i.e. $H(z) = H_0(z^2) + z^{-1}H_1(z^2)$, and $G(z) = G_0(z^2) + z^{-1}G_1(z^2)$ which increases the throughput by a factor two, see Figure 1. Further evaluation shows that the multiplier in the transposed FIR of a polyphase filter only has to be realized once since $G_1(z) = z^{-N/2}H_0(z^{-1})$ and $G_0(z) = -z^{-N/2}H_1(z^{-1})$. For orthogonal filters, this will result in more speed and savings of resources [4]. Filtering, using FFT methods, is preferred for filters of length 32 or larger [5], and, will therefore not be discussed here.

To compare different design options, a Daubechies length 4 and 8 FIR filter and two channel (synthesis) filter bank model is used. Three different computer arithmetic unit design strategies are considered. They are:

- Reduced Adder graph (RAG) - technique using a factored canonical signed digit (CSD) code.
- Distributed arithmetic (DA) - concept for realizing a sum of product.
- Residue Number System (RNS) - implementation using the concept of parallel, small word length, high bandwidth channels.

In the following sections the comparisons are made. First, however, we briefly describe the candidate FPL family and define some basic DSP building blocks.

2. BASIC DSP BLOCKS

Each logic block (LB) of a Altera FLEX 10K devices contains 8 LEs where each LE provides a $2^3 \times 1$ table and fast carry chain support in an arithmetic mode [1]. A EPF10K250A devices has, for example, 1520 LE blocks, 12160 LEs, and twenty 2K tables (called embedded array block, EAB). For implementing high-speed adders, it should be noted that LE tables are much slower ($\approx$ 3ns) than a fast carry chain ($<$ 0.5ns). Therefore the "fast adders" appearing in the literature [6], give a *slower* speed in FPLs than that using a ripple carry architecture with fast carry chains. The following table shows synthesis results (Altera optimization: 0 area/10 speed) for two's complement adders and array multiplier (i.e. Wallace tree) with a 4 pipeline stage delay data for 4ns devices where speed is measured in mega operations per second, and the number of required logic elements (LE).

	ADD				MUL	
Bits	8	16	26	32	9×9	12×12
MSPS	137	73	51	45	71	69
# LE	8	16	26	32	217	328

A naive wavelet filter bank design approach is to realize a FIR filter by simply combining the building blocks of adder, multiplier and register. The following table shows estimated resource requirements and speed of such a solution.

Type	#LE	Bits	Speed [MHz]
DB4	928	20	43.5*
DB8	1883	21	43.5*
DWT4	988	20	43.5*
DWT8	2030	21	43.5*

The reduced speed comes from routing problems. Typical a 30-40% decrease in performance can be observed for larger devices and designs, compared with the single building block in a smaller device (see DA section).

3. REDUCED ADDER GRAPH TECHNIQUE

Canonical signed digit (CSD) coding is a well studied computer arithmetic technique. In general, it can be assumed that an adder and subtrator are of equal complexity. Furthermore, in 2's complement and the CSD, 15 (decimal) can be coded as

$$15_{10} = 1111_2 = 16_{10} - 1_{10} = 1000(-1)_2 \qquad (1)$$

The integer $45_{10} = 101101_2$ has cost 3 (i.e. 3 adder are needed), but $5 \cdot 9 = (4+1)(8+1)$ has cost 2. There are 7 such integers below 100, namely 45,51,75,85,90,93, and 99 with lower cost after beeing factorized. In a filter bank realization a factorization of the coefficients is even more attractive, because most often several factors can be used in common. For instance, a halfbandfilter "F6" from Goodman and Carey (non-zero coefficients 346,208,44,9) has cost 9. Using the factor CSD code (a.k.a. RAG algorithms) reduced that to 5. In general finding the optimal RAG is a NP hard problem and heuristics must be applied. There are some obvious actions possible, however, to reduce the effort. Specifically:

- Remove all factors which are power of 2, because they can be accomplished by shift or hard-wiring.
- Realize all cost 1 coefficients.
- Using the cost 1 coefficients, start with the next coefficients with minimal costs.

To simplify the algorithm it's helpful to have an optimal CSD table available [7, 8]. If we realize a Daubechies filter of length 8, quantized to 10 bits (9 mantissa plus sign), then it follows that

n	0	1	2	3	4	5	6	7
$h[n]$	164	511	448	20	132	22	24	8
Cost	2	1	1	1	1	2	1	0

In a RAG algorithm, one can use the factor 3 two times, (i.e. $22 = (3+8)2$ and $24 = 3 \cdot 8$), and therefore reduce the total cost to 8. The following table shows the data for the DB length 8 filter in CSD code and the two channel filter banks of length 4 and 8.

Type	#LE	Bits	Speed [MHz]
DB8	946	21	81*
DWT4	364	20	44.2
DWT8	1992	21	81*

For the DB8 and DWT8 filter pipelined adders were used to speed up the design, i.e. instead of a single 21 bit adder, three 7 bit adder were used and additional pipeline registers added. But such pipeline adders including input and output registers need significant resources as it can be seen from the following table.

Bits	#LE	Speed [MHz]
9-15	39-69	135.1
16-22	93-135	118.9

4. DISTRIBUTED ARITHMETIC

The distributed arithmetic (DA) concept [9] is an alternative to compute a "sum of product"

$$y = \sum_{n=0}^{N-1} h[n]x[n] \qquad (2)$$

by combining all bits of the $x[n]$ at a single position b to one word according to

$$y = \sum_{n=0}^{N-1} h[n] \sum_{b=0}^{B-1} x_b[n]2^b = \sum_{b=0}^{B-1} \sum_{n=0}^{N-1} \underbrace{h[n]x_b[n]}_{f(h,x)}. \qquad (3)$$

It should be noted that for DA with an iterative realization, the latency depends on the number of bits B and *not* on the number of coefficient N, as is the case for a general programmable DSP. This basic principle can be speed up by computing several of the

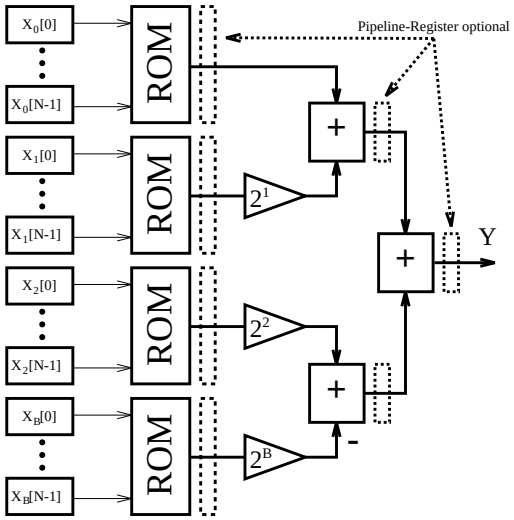

Figure 2: Maximum speed DA implementation.

$f(h, x)$ in parallel. The fastest way is if all LUT are computed in parallel, as shown in Figure 2.

The "fast" approach is limited by the available "fan in" of the LUTs in FPLs to 4-8 TAP filter. For Altera Flex 10K we can use the 4 input LE or the $2^8 \times 8$ EAB ROM tables. For a 2 channel filter bank using DA the following observations are made:

1. A polyphase decomposition (or symmetry) reduces the effort essential to half the necessary fan in.
2. The CQF symmetry can *not* be utilized in DA designs.
3. The fastest pipelining can theoretical be achieved with LE (ca. 130 MHz). EABs provide 86.2 MHz theoretical performance.

The following table shows results for different DA designs. It should be noted by combining 4 length 4 filter the performance drops from 106 to 65.2 MHz. But using the EAB the necessary routing is essential reduced and the performance for the 2 channel filter bank only drops from 86.2 to 84.7 MHz.

Type	#LE	#EAB	Bits	Speed [MHz]
DB4	643	–	20	106
DB8	515	16	21	86.2
DWT8	2737	–	20	65.2
DWT8	1110	32	21	84.7

There is also an essential number of LE necessary for the EAB design because the adders (20 or 21 bit) again have to be pipelined.

5. RESIDUE NUMBER SYSTEM (RNS)

The silicon area associated with a constant-speed fixed-point MAC unit is generally considered to geometrically increase with word-length. The antithesis is the Residue Number System (RNS) which has a linear relationship between MAC silicon area and speed [10]. The RNS, therefore, provides an opportunity to overcome the precision barrier in high-performance FPL applications. The RNS mechanics are well understood. RNS integer arithmetic is performed concurrently in parallel non-communicating small word length channels. An RNS system is defined in terms of a basis set $\{m_1, m_2, \ldots, m_L\}$ of relatively prime positive integers. The dynamic range of the resulting system is $M = \prod_{l=1}^{L} m_l$. RNS arithmetic is defined with respect to the ring isomorphism

$$\mathbb{Z}_M \cong \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_L} \tag{4}$$

Specifically, $\mathbb{Z}_M = \mathbb{Z}/(M)$ which corresponds to the ring of integers modulo M. The mapping of an integer X into the RNS is defined to be the L-tuple $X = (x_1, x_2, \ldots, x_L)$ where $x_l = X \mod m_l$, for $l = 1, 2, \ldots L$. Defining $\square$ to be either the algebraic operation $+, -$ or $*$, it follows that if $0 \leq Z < M$, then

$$Z = X \square Y \mod M \tag{5}$$

is isomorphic to $Z = (z_1, z_2, \ldots, z_L)$ where

$$z_l = x_l \square y_l \mod m_l \qquad l = 1, 2, \ldots, L \tag{6}$$

Here it is evident that the RNS arithmetic is performed in parallel within channels whose word width is bounded by $w_l = \lceil \log_2(m_l) \rceil$ where typically $w_l \leq$ 8-bits. In practice, most RNS arithmetic systems use small RAM or ROM tables to implement the modular mappings $z_l = x_l \square y_l \mod m_l$.

RNS systems have been built as custom VLSI devices [11], GaAs, and LSI [10]. For a small wordlengths, the RNS has been shown to provide a significant speed-ups [12] using the $2^4 \times 2$ bit tables found in a Xilinx XC4000 FPGAs. For larger moduli, the $2^8 \times 8$ bit tables belonging to the Altera FLEX CPLDs are beneficial in designing RNS arithmetic and RNS-to-integer converters. With the ability to support larger moduli, the design of high-precision FPL systems becomes a practical reality.

A wide variety of *modular* addition designs exist [13]. Using LEs only, the design of Fig. 3(a) is viable for FPLs. The Altera FLEX CPLD contains a small number of 2K Bit ROMs or RAMs (EABs) which can be configured as $2^8 \times 8, 2^9 \times 4, 2^{10} \times 2$ or $2^{11} \times 1$ tables which can be used for modulo m_l correction. The next table shows re-designed [14] 6, 7, and 8-bit modulo adder.

Bits	Pipe	6	7	8
MPX	0	41.3 MSPS 27 LE	46.5 MSPS 31 LE	33.7 MSPS 35 LE
MPX	2	76.3 MSPS 16 LE	62.5 MSPS 18 LE	60.9 MSPS 20 LE
MPX	3	151.5 MSPS 27 LE	138.9 MSPS 31 LE	123.5 MSPS 35 LE
ROM	3	86.2 MSPS 7 LE 1 EAB	86.2 MSPS 8 LE 1 EAB	86.2 MSPS 9 LE 2 EAB

Although the ROM shown in Fig 3 provides high-speed, the ROM itself produces a four cycle pipeline delay and the number of ROMs is limited. ROMs, however, are mandatory for the scaling schemes discussed in the next section. The multiplexed-adder (MPX-Add) has comparatively a reduced speed even if a carry chain is added to each column. The pipelined version usually needs the same number of LEs as the un-pipelined version but runs about twice as fast. Maximum throughput occurs when the adders are implemented in two blocks (where each block has 8 LEs for Altera FLEX 10K devices) within six-bit pipelined channels.

Several other RNS basic building blocks were designed. This includes modulo adder for the index domain, i.e. modulo multiplier, Converter (BIN→RNS and RNS→BIN) and a $\epsilon-$CRT. Altera

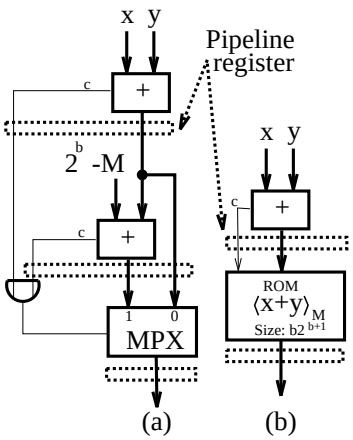

Figure 3: Modular addition with CPLD. (a) MPX-Add and MPX-Add-Pipe. (b) ROM-Pipe.

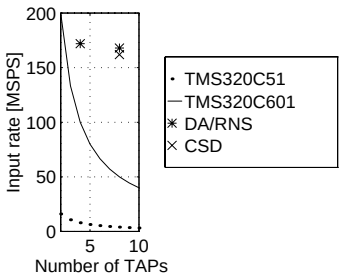

Figure 4: Comparison of programmable DSPs and FPL two channel DWT filter banks.

VHDL version 7.1 does not allow generic clauses. `gawk` and `c` programs have been developed for an automatic generation of the basic building blocks, simply by specifying the desired blocks and the moduli set. Within seconds a complete set of error free, highly optimized VHDL code blocks is produced for a specific moduli set.

Using this build blocks single DB4 RNS channel have been designed in 6, 7, and 8 RNS arithmetics as shown in the following table:

Type	#LE	#EAB	Bits	Speed [MHz]
DB4	233	2	6	73.5
DB4	260	2	7	86.2
DB4	287	4	8	82.6

6. CONCLUSION

The Figure 4 shows in conclusion that all three proposed design strategies outperform the fastest commercial available programmable DSP for DWT of length larger than 4. For the complexity for this medium precision (10 Bit coefficient) design the RAG i.e. factor CSD code design gives the best results. But the effort for DA and RNS design are not dependent on the specific coefficient values and for higher precision this strategies may be superior.

7. REFERENCES

[1] Altera Corporation, "Data Sheet," FLEX 10K CPLD Family, June 1996.

[2] R. Hartenstein, J. Becker, and R. Kress, "Costum computing machines vs. hardware/software co-design: From a globalized point of view," in *Lecture Notes in Computer Science*, Sept. 1996, vol. 1142, pp. 1142:65–76.

[3] M. Acheroy, J.-M. Mangen, and Y. Buhler., "Progressive wavelet algorithm versus jpeg for the compression of meteosat data," in *SPIE, San Diego*, 1995.

[4] Z. Mou and P. Duhamel, "Short-length fir filters and their use in fast nonrecursive filtering," *IEEE Transactions on Signal Processing*, vol. 39, pp. 1322–1332, June 1991.

[5] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 569–586, Mar. 1992.

[6] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[7] Uwe Meyer-Baese, *Schnelle digitale Signalverarbeitung*, Oldenbourg Verlag, 1998.

[8] A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in fir digital filters," *IEEE Transactions on Circuits and Systems II*, vol. 42, pp. 569–577, Sept. 1995.

[9] S. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE Transactions on Acoustics, Speech and SignalProcessing Magazine*, pp. 4–19, July 1989.

[10] M. Soderstrand, W. Jenkins, G. Jullien, and F. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press Reprint Series. IEEE Press, 1986.

[11] J. Mellott, M. Lewis, F. Taylor, and P. Coffield, "ASAP – a 2D DFT VLSI processor and architecture," in *International Symposium on Circuits and Systems*, May 1996, pp. 261–264.

[12] V. Hamann and M. Sprachmann "Fast Residual Arithmetics with FPGAs". *Proceedings of the Workshop on Design Methodologies for Microelectronics*, Smolenice Castle, Slovakia, pages 253 - 255, Sept. 1995.

[13] M. Bayoumi, G. Jullien, and W. Miller, "A VLSI implementation of residue adders," *IEEE Transactions on Circuits and Systems*, pp. 284–288, Mar. 1987.

[14] A. Garcia, U. Meyer-Baese, and F. Taylor, "Pipelined Hogenauer CIC filters using field-programmable logic and residue number system," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 1998, vol. 5, pp. 3085–3088.