TIME-AREA EFFICIENT MULTIPLIER-FREE RECURSIVE FILTER ARCHITECTURES FOR FPGA IMPLEMENTATION

Mohammad Shajaan and *John Aasted Sørensen*

Electronics Institute, Technical University of Denmark
Building 349, DK-2800 Lyngby, Denmark
E-mail: ms@ei.dtu.dk and jaas@ei.dtu.dk

ABSTRACT

Simultaneous design of multiplier-free recursive filters (IIR filters) and their hardware implementation in Xilinx Field Programmable Gate Array (XC4000) is presented. The hardware design methodology leads to high performance recursive filters with sampling frequencies in the interval 15–21 MHz (17 bits internal data representation). It will be demonstrated that *time-area efficiency* and *performance* of the architectures are considerably above any known approach.

1. Introduction

In recent years the complexity of the Field Programmable Gate Arrays (FPGA's) have reached a level where they can be useful as a fundamental DSP-component. The functional structure of the XC4000 family is very constrained and complex, due to low level irregularity. This irregularity may result in dramatic time-area efficiency differences between equivalent realizations, making careful low level design and manual floorplanning necessary. This paper considers the necessary approaches to obtain optimal FPGA-designs, using multiplier-free an IIR filters example. Because of the recursive nature of the IIR filters, the realization of these filters are more difficult than FIR filters. In this paper an efficient method for the design and realization of the IIR filters based on cascaded biquads is presented.

2. IIR Filter Design

The background of the filter synthesis method is, that the transfer function of an IIR filter can be realized by cascaded biquads [1]. The transfer function expressed by biquads becomes:

$$H(z) = \prod_{i=1}^{N} \frac{a_{i,0} + a_{i,1}\, z^{-1} + a_{i,2}\, z^{-2}}{b_{i,0} + b_{i,1}\, z^{-1} + b_{i,2}\, z^{-2}} \qquad (1)$$

If the coefficients of the biquads are signed power-of-twos (SPT), the filter can be realized multiplier-free. This design problem can be formulated:

Design problem : *Find the best set of N biquads with SPT coefficients that give the least normalized ripple.*

In [2] a method is presented for optimization of a cascaded realization of linear phase FIR filters. This method can also be used to optimize a cascaded realization of biquads. The method starts with an IIR filter with infinite precision coefficients, and then by using an univariate search optimizes the cascaded biquads.

The implementation of a 10th order low-pass Butterworth filter is used to explain the hardware synthesis method. The frequency response of the multiplier-free filter is shown in Fig. 1. The normalized ripple in pass-band and stop-band are also shown. Note that the filter with quantized coefficients is not flat in the pass-band. The coefficients are chosen to be at most two SPT terms. Fig. 2 shows the possible zero and poles for a biquad with two SPT term coefficients and the word-length of the coefficients is 9; i.e., $\pm 2^{-p} \pm 2^{-q}$ where $0 \le p, q < 8$. The zero- and pole-grids show that these biquads can be used to realize the commonly used filters; of course, some IIR filters may have biquads that requires more resources.

3. Pipelining of a General Biquad

Performance of the filters realized by cascaded biquads depends on the efficiency of the general biquads. In this section a pipelining method is presented for general biquads. Fig. 3 shows a direct form realization of a biquad of Eq. 1. Note that $\frac{1}{b_0}$ is used in the biquad. Because the coefficients of the biquads consist of at most two SPT terms, each coefficient is realized by two wired shifts and one addition or subtraction denoted by $\boxtimes$. The critical logical paths are shown by dashed arrows. The critical paths contain 6 adders. These paths can be shortened by pipelining the all pole section and all zeros section of the biquads. The result of the pipelining is shown in Fig. 4, and the critical paths now contain just 4 adders. It is not possible to shorten the critical paths in this structure furthermore, thus one of them contains just one delay element that can only be moved in the path without changing its length. By using an alternative realization of this structure and cut-set retiming [3], it is possible to reduce the length of the critical path to 3 adders. The alternative structure is achieved by moving the adder denoted by (1) in Fig. 4, and applying cut-set (A), as shown on Fig. 5. Big dots denote delay elements appeared due to cut-set retiming. The critical path contains now just three adders; i.e., a 50 % improvement with respect to, the original structure.

4. Hardware Synthesis

Zeros of a low-pass Butterworth filter are all located at z=-1; i.e., the sections in the numerator of the biquads are multiplier-free ($a_0 = a_2 = 0.5$ and $a_1 = 1$). $\frac{1}{b_0}$ is chosen to be one SPT term. These simple coefficients simplify implementation of Butterworth filters. The coefficients of the denominator (given in Tab. 1) are found using the optimization algorithm in [2].

The scaling factors (S_i in Tab. 1) ensure that there will be no overflow in the adders and the dynamic range of the adders is optimally used. All scaling factors are limited to be one SPT term to ease their realization.

The example is targeted for approx. 1.4 XC4005 FPGA's. Each XC4005 has 196 (14x14) configurable logic blocks (CLB). The hardware realization is synthesized in a 4 stage process:

Stage One : The filter architecture is defined as a linear systolic array, shown in Fig. 6. By applying systolization cut-sets between the biquads, the filter architecture in *level 1* is a linear, temporally and spatially local systolic array. *Level 2* systolization involves pipelining of a biquad, which is explained in the previous section. The elements (adders and registers) are marked in Fig. 5 to show the mapping process. R5 is the pipeline register due to the level 1 systolization.

Stage Two : The next stage, in the implementation of the biquads, is mapping of the adders and registers to a processor array, which shows the relative location of the adders and registers in FPGA. The mapping can be done in many ways depending on the available routing resources. The only way to find the optimal location is to try all the possible orderings; this requires a lot of time. By placing the elements in a way that preserves the natural flow of data, it is possible to achieve a very good result. Fig. 7 shows an efficient processor array for the shown biquad.

Stage Three : The processor array is then mapped to a bit level structure graph to examine possibility of reduction and bit level systolization. Then the length of the elements in the section processor array are determined, and the next step is mapping of the section processor array to the FPGA. Fig. 10 shows the bit level structure graph of the biquad number 1.

Unlike FIR sections [2], it is not possible to realize biquads without internal truncation or rounding, thus the biquads are recursive and a realization without truncation needs adders of infinite length. Therefore all the adders are limited to be 17 bits adders. It is not possible to apply bit level systolization due the recursive nature of the biquads.

Considering the XC4000 family structure constraints, three efficient types of the processing elements (PE) can be realized. These three PE's can realize all types of operations required by a digital filter with SPT coefficients, and are at the same time the most optimal usage of the CLB's of the FPGA's. The processor elements are shown in Fig. 8. The array of the processor elements are shown in Fig. 9. The elements in the dotted squares in Fig. 7 are mapped to the same processor element in the FPGA.

Stage Four : Having the section processor array, the next stage in the implementation, is to floor-plan and to realize the processor elements with the required word-length and route the design. Fig. 11 shows an efficient floor-planing for the 10th order Butter worth filter in two XC4005 FPGA's. Fig. 12 shows the routed design of the 5 biquads. Note that biquad 3 is realized using U-formed elements.

The efficiency of the routing depends on the quality of the CAD-system used. Normally, these systems use simulated annealing to rout the design. Therefore, different routing will result in different timing properties. This makes it difficult to predict the result, but an estimate can be given for speed grade 5 XC4000 FPGA's:

$$T[\text{ns}] \simeq 5.5 + \frac{W-1}{2}\,1.5 + N_a\,6 + N_a\,10 \qquad (2)$$

where W is the word-length of the adders and N_a is the number of the adders in the longest logical path. The three first terms in Eq. 2 are due to carry initialization, carry propagation and carry out (from the top full-adder), respectively. The 4th term is the estimated routing delay per adder in the longest logical path. In this example the maximum number of the adders in the longest logical paths in the biquads is 2, and the word-length of the adders is 17. The delay for signal passing through these adders becomes $T = 49.5$ ns; i.e.,

$$f_{throughput} \simeq \frac{1}{T} = 20.2 \text{ MHz} \qquad (3)$$

An implementation of the multiplier-free filter in two XC4005 FPGA (using XACT) showed that throughput of 21.6 MHz is possible. Each biquad was implemented in 54 CLB's; i.e., the multiplier-free Butterworth filter requires 5·54=270 CLB's.

A careful floor-planning is also necessary to get a good performance and a high throughput, and some experiences have shown that an automatic routing using the design software such as XACT is not sufficient to obtain high performance.

As mentioned, a low-pass Butterworth filter has simple coefficients in the numerator of the biquads and for some other types of filters, at least two SPT terms must be used for each coefficient. In this case the implementation is more difficult and experiments show that the maximum throughput rate is reduced to about 17 MHz for an all pole section (An all-zeros section can be realized with very high throughput rate regardless of the number of SPT terms used in coefficients [2]).

5. Comparison and Conclusion

The design of an IIR filter is compared with the two fastest designs [4, 5] from the literature. The comparison is based on the Time-Area efficiency index:

$$I = \frac{\text{Filter order}(N) \cdot \text{Word-length}(B)}{\text{Area}(A) \cdot \text{Sampling Period}(T)} \qquad (4)$$

and the result is shown in the following table.

Ref.	f_s	N	A [CLB]	B [bits]	I
[4]	50 KHz	15	317	16	0.038
[5]	10 MHz	4	576	12	0.83
This work	21.6 MHz	10	270	17	13.60

Multiply **AC**cumulate rate is defined by

$$\text{MAC-rate} = (\text{Number of coefficients}) \cdot f_s \qquad (5)$$

The MAC-rate of the above filter with 22 coefficients becomes 475.2 MHz. The realizations in [4] and [5] achieve 1.6MHz and 100MHz MAC-rates, respectively.

This work introduces a new efficient architecture, which achieves a performance higher than the known approaches. The architecture uses FPGA technology to realize programmable dedicated hardware for IIR filters with highly quantized coefficients.

6. References

[1] L. R. Rabiner et. al., *"Theory and Application of Digital Signal Processing,"* Prentice-Hall 1975.

[2] M. Shajaan et. al., *"Time-Area Efficient Multiplier-Free Filter Architectures for FPGA Implementation,"* ICASSP 95, pp. 3251–3254.

[3] S. Y. Kung, *"VLSI Array Processors,"* Prentice Hall, 1988.

[4] M. Wahab et. al., *"FPGA-based DSP Systems,"* The Oxford 1993 International Workshop on Field Programmable Logic and Application, pp. 291–298.

[5] C. Chou et. al., *"FPGA Implementation of Digital Filters,"* ICSPAT 93, pp. 80–88.

i	S_i	$1/b_{i,0}$	$b_{i,1}$	$b_{i,2}$
1	2^{-1}	1	$-1-2^{-4}$	$1-2^{-2}$
2	2^{-1}	2^{-1}	$-1-2^{-1}$	$1-2^{-5}$
3	2^{-1}	2^{-1}	$-1-2^{-1}$	$2^{-1}+2^{-5}$
4	1	2^{-1}	$-1-2^{-2}$	$2^{-2}+2^{-5}$
5	2^{-1}	2^{-1}	$-1-2^{-1}$	$2^{-2}+2^{-4}$

Table 1: The coefficients of the 10th order filter.

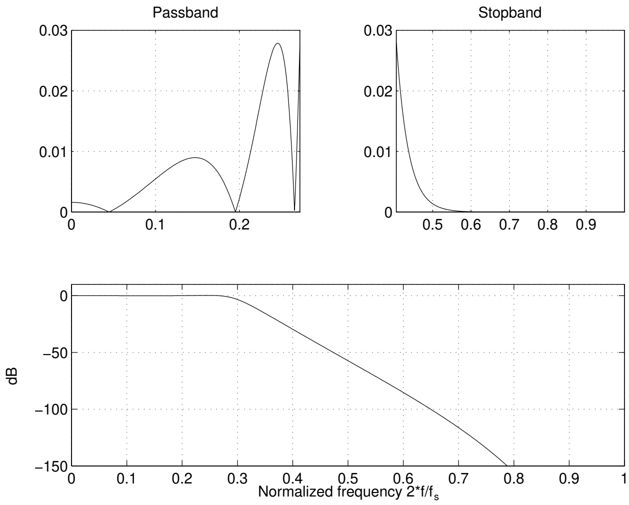

Figure 1: Multiplier-free filter example.

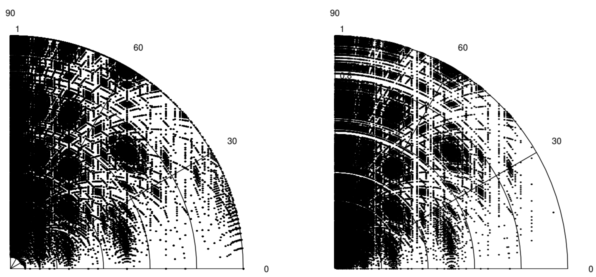

Figure 2: Zeros- and pole-grid for a direct form realization of biquads with 2 SPT term coefficients.

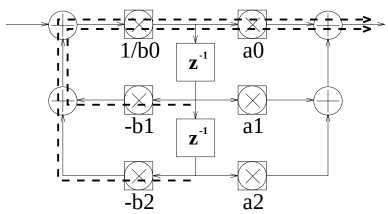

Figure 3: General biquad. Dashed arrows show the critical paths.

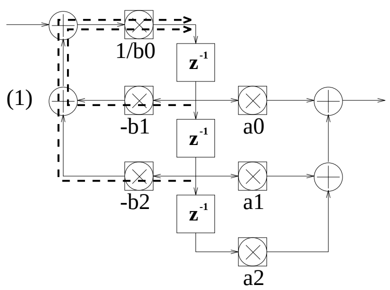

Figure 4: Pipelined biquad.

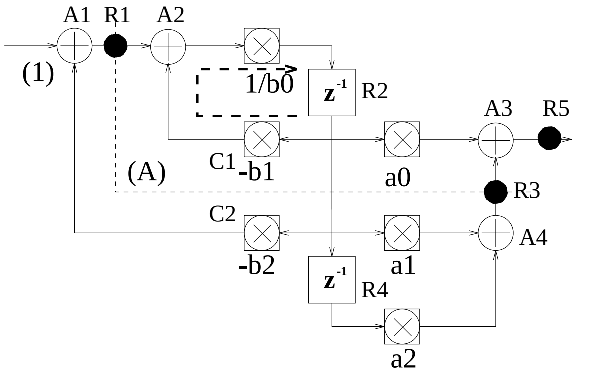

Figure 5: Alternative realization of a pipelined biquad.

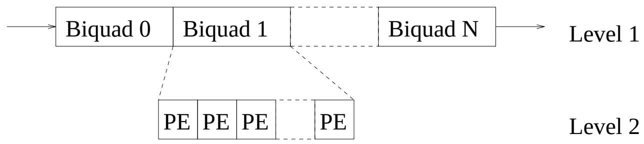

Figure 6: Level 1 and 2 systolization.

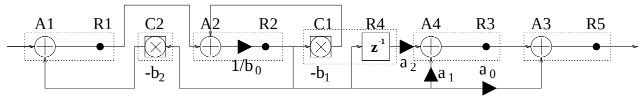

Figure 7: The processor array for the biquad given in Fig. 5.

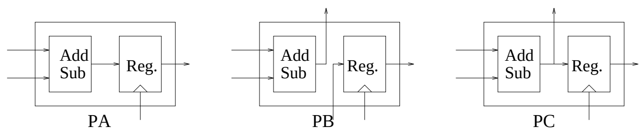

Figure 8: The three efficient processing elements for XC4000 FPGA's.

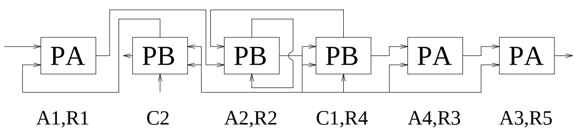

Figure 9: The mapping of a biquad to the processor element in FPGA.

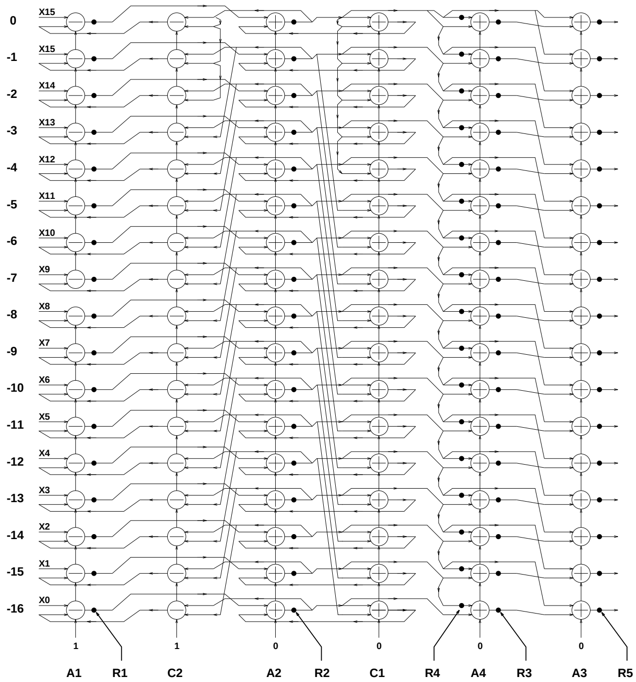

Figure 10: The bit level structure graph for the biquad number 1 ($s = 2^{-1}$, $1/b_0 = 1$, $b_1 = -1 - 2^{-4}$, $b_2 = 1 - 2^{-2}$, $a_0 = a_2 = 2^{-1}$, and $a_1 = 1$).

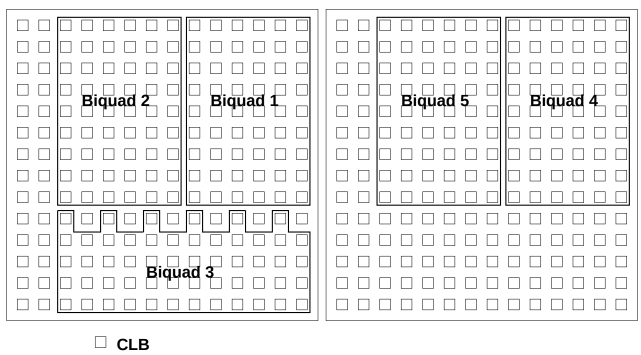

Figure 11: The floor planning of the 10th Butterworth filter in two XC4005 FPGA's.

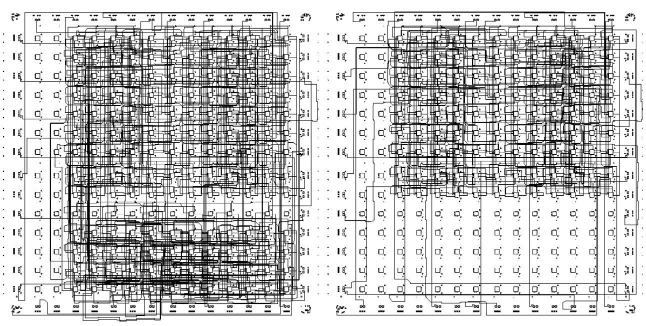

Figure 12: The chip plots of FPGA 1 and 2.