Time-area efficient multiplier-free filter architectures for FPGA implementation

Shajaan, Mohammad; Nielsen, Karsten; Sørensen, John Aasted

Published in:
I E E E International Conference on Acoustics, Speech and Signal Processing. Proceedings

Link to article, DOI:
10.1109/ICASSP.1995.479578

Publication date:
1995

Document Version
Publisher's PDF, also known as Version of record

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

Mohammad Shajaan, Karsten Nielsen, John Aasted Sørensen

Technical University of Denmark, Electronics Institute
E-mail: ms@ei.dtu.dk and jaas@ei.dtu.dk

ABSTRACT

Simultaneous design of multiplier-free filters and their hardware implementation in Xilinx Field Programmable Gate Array (XC4000) is presented. The filter synthesis method is a new approach based on cascade coupling of low order sections. The complexity of the design algorithm is $\mathcal{O}$(filter order). The hardware design methodology leads to high performance filters with sampling frequencies in the interval 20–50 MHz. Time-area efficiency and performance of the architectures are considerably above any known approach.

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 new DSP-component. Unlike standard gate arrays, 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 illustrates the necessary approaches to obtain optimal FPGA-designs, using multiplier-free filters as DSP algorithm examples.

Multiplier-free linear phase FIR filters by quantization of zeros

The background of the filter synthesis method presented is, that the transfer function of a linear phase FIR filter is symmetric or antisymmetric and can be factorized in fourth order, second order and first order sections with real coefficients. Every section represents a zero-group [4] denoted by R_i. The design problem can be formulated:

Design problem : Find the set $\{R_1, \ldots, R_N\}$ of the zero-groups which gives the least normalized peak ripple. N is the number of sections.

To the best of the authors knowledge, no efficient algorithm with linear time complexity exists to solve this design problem. Following, an $\mathcal{O}$(*filter order*) iterative algorithm is presented:

BEGIN ALGORITHM
- *Design the infinite precision FIR filter.*
- *Find the set of optimal zero-groups and the number of sections N*

WHILE *improvement achieved* OR *first iteration*
FOR $i = 1, \ldots, N$
- *Quantize R_i.*
- *Retain the currently optimal quantization of $R_{j \neq i}$, $j \in \{1, \ldots, N\}$.*
- *Compute the approximation error using the quantized values of R_i and the values of $R_{j \neq i}$.*
- *Select the best quantized R_i.*

An example will explain the algorithm in more detail. Fig. 1 shows a small search tree for a filter with three zero groups (R_1, R_2 and $R3$). At the first iteration a zero-group (R_1) is selected and quantized to the nearest zeros in the discrete space. Let the number of possible quantized zeros be three at each node (e.g. A, B and C at the first node). We choose A which gives the least normalized ripple with unquantized R_2 and R_3 at node 1. Afterwards R_2 is quantized and we choose D with quantized R_1 (i.e. A) and unquantized R_3. At the third node the last zero-group R_3 is quantized and F is chosen. At this stage a result is achieved, but it may be improved by repeating the search. The new iteration (second) differs from the first by quantizing R_1, and calculating normalized ripple using D and F as the rest of the filter. Second iteration results in a better solution (B, E, G). Since improvement was achieved, a new iteration (third) is started. Since no improvement is obtained, now the algorithm stops and the output of the algorithm is sections with zero-groups B, E and G.

The algorithm finds a semi-optimal filter. Since the zeros of the stopband section is placed on the unit circle, a good stopband attenuation is always achieved. However, the passband ripple is normally larger than the stopband ripple. Using a systematic approach, a large number of filters have been designed. Results are comparable with other approaches, despite the low algorithm complexity. The algorithm is very fast with linear time complexity, e.g. a 100th order filter can be designed in less than 90 seconds on a HP700 computer. The normalized peak ripple was calculated by using a

0-7803-2431-5/95 $4.00 © 1995 IEEE

frequency grid greater than $20 \cdot (filter\ order)$.

Hardware methodology by filter example

A hardware synthesis method leading to minimal, high performance hardware realizations of the multiplier-free filters has been developed. In the following, this method is illustrated by implementing a 33-tap multiplier-free filter example, with bandedges 0.3 and 0.5 for stopband and passband, respectively. Frequency responses of the original and multiplier-free filter are shown in Fig. 2. The coefficients are represented as Signed Power of Two (SPT) numbers with a 9 bit range, and normalized peak ripple is -50 dB. 16 bit data representation is used to make good noise properties possible.

The filter example has 10 complex conjugated zero-groups realized by symmetric 2nd order sections with a 1-2 SPT term combination, i.e. first and second coefficient is a sum of 1 and 2 SPT terms, respectively. Three quadruple zero-groups are realized by 4th order sections of different complexity (1-3-3 and 2-3-3). The section ordering and scaling factors are determined by noise considerations, because of the extreme sensitivity between output noise and section ordering [4]. Different section orderings showed a theoretical 80dB output noise difference between the chosen section ordering and worstcases. The hardware methodology is based on scaling factors restricted to power of two values, and no truncation internally in the section.

The example is targeted for a FPGA hardware prototyping PC-board, at the time present configured with a quadratic array of four XC4005 devices. The hardware realization is synthesized in a 4 stage process:

In **Stage One** the filter architecture is defined as a linear systolic array in 2 levels, shown in Fig. 3. By applying systolization cut-sets [1] between sections, the filter architecture seen at level 1 is a linear, temporally and spatially local systolic array. Each section in the filter architecture is also realized by a linear systolic array (Fig. 3, level 2), with fine-grain processing elements (PE's) as the fundamental components. The set of PE's are devised on the basis of detailed knowledge of the XC4000 architecture constraints, and the multiplier-free section structures to be implemented. Exactly 3 PE's are necessary to make an efficient realization of all section structures possible. The three PE's are generated as different combinations of a basic operation module. Fig. 3 shows high level representations of both the basic operation module and the three PE's. A 2-bit bitslice of each PE can be implemented in one, optimal used configurable logic block (CLB).

In **Stage Two**, mapping of section structures to PE's and the section (level 2) floorplanning is carried out, considering the major constraints imposed by the routing architecture. The physical shape and relative placement of PE's is highly restricted, by the use of the dedicated carry logic. The constraints of this resource implies, that optimal section processorarrays have to be realized with a horisontal PE-topology.

The section structures are chosen by comparing filter synthesis results with hardware complexity. All sections are realized by transpose form structures, shown in Fig. 4. The triangled symbol represents an adder/subtracter unit realizing 'multiplications', by adding/subtracting hardwired shifted operands. The original passband (4th order) section structures results in inefficient hardware realizations. An efficient map of the original structures to the above defined PE's is not possible, since the number of delay elements is less than the number of arithmetic units. Furthermore, three arithmetic units share combinatorial paths in both 4th order structures, resulting in relatively poor performance. By 2nd level systolization in these section structures, far more efficient structures are generated. Fig. 4 shows the systolization cut-sets, leading to the dramatic increase in time-area efficiency. The number of registers matches the number of aritmetic units (complete map to PE's), and the longest combinatorial path is reduced to two arithmetic units (better temporal locality).

Rules have been specified to automize both mapping to PE's and floorplanning the section processorarray. The mapping of the systolized section structures to section processorarrays for the filter example is shown in Fig. 5.

In **Stage Three**, bitlevel reduction mechanisms are applied using a bitlevel structure graph (BSG). This bitlevel representation form was developed to reveal the complete, somewhat irregular bitlevel structure of the sections, making total dedication and further bitlevel (level 3) systolization possible. All redundant bitlevel operations are eliminated by specified reduction rules in the upper and lower BSG. In practice this stage involves two substages, graph construction leading to BSG1 and elimination leading to a fully minimized graph, named BSG2. Fig. 6 shows BSG1 for a stopband section with coefficients ($a = 2^{-1},\ b = 2^0 - 2^{-5}$). The upper structure is reduced by a wordlength adjustment cut (1) and a scaling cut (2), and the lower structure is reduced by a cut (3). All bitlevel elements above (1), (2) and under (3) are redundant and can be removed (reductions due to cut (1) have been carried out in the figure). Due to the simplicity and low coefficient wordlength of the example section, reductions are not remarkable. In general, especially for more complex structures, the reduction mechanisms have a considerable effect. From the minimized graph (BSG2), the final wordlengths of the PEs are determined, and the practical hardware implementation is thereafter trivial, using BSG2 and a library of PE's (Xilinx hardmacro's).

In **Stage Four**, the final floorplanning (level 1) is carried out, considering the FPGA-topology and the

fixed placement of memory connections and communication channels on the PC board. The linear systolic array is mapped directly to a linear FPGA array, with multiple sections in every FPGA. Fig. 7 shows four XC4005 chip plots of the realization. Timinganalysis showed a maximal delay of 49 ns, giving a sampling frequency of 20 MHz. The total amount of resources used by the systolic array was 538 CLB of 784 CLB. Further resources are needed for memory control.

Structure classification

Different section structures including transpose form, direct form and two lattice structures have been analyzed by time-area effiency considerations. Effiency has two primary aspects: (1) *Resource usage.* Map to PE. Number of registers realized outside PE's. (2) *Performance.* Temporal locality. Number of arithmetic units in longest combinatorial path.

Two efficient *forms* have been defined, each representing one of the above aspects. The *Adjusted Form* representing an optimal map to PE's, and *Maximal Form* representing a full systolic multiplier-free section structure, with only one arithmetic unit in every combinatorial path. The maximal form leads to a temporally local section processorarray. A maximal form structure leads to the highest performance that can be achieved by 2nd level systolization.

A library of systolized multiplier-free section structures on the two defined forms has been generated. The practical classification is based on 3 complexity parameters representing resource usage, performance and pipeline delay. The library of systolized section structures and the attached classification parameters makes it possible to determine the optimal structures for every application.

Bitlevel (level 3) systolization

Higher performance is achieved by bitlevel systolization on the basis of BSG2. Effective bitlevel cuts (I) and (II) for the stopband section example is shown in Fig. 6. The result of a bitlevel cut-set is a performance increase at the cost of an increase in resource usage. The table below shows implemented and estimated results for the two worstcase 2nd and 4th order sections in the different performance classes, that can be reached with 16 bit data representation.

Class	$Fs = \frac{1}{T}$ [Mhz]	*Area* (1-2) [CLB]	*Area* (1-3-3) [CLB]
Ajusted	20-23	30	75
Maximal	29-33	46	99
Maximal(I)	37-42	57	115
Maximal(II)	44-50	68	131

Using a time-area efficiency index $\frac{Filter\ order \cdot Wordlength}{Area \cdot T}$, the hardware methodology yields both efficiency and performance considerably above any known approach on the XC4000, XC3100 and XC3000 series. Comparison has been done with the best known approaches, presented in [2] and [3].

Since the FPGA-component has the same flexibiliy as the digital signal processor (in terms of programmability), it is reasonable to compare with this traditional DSP technology. This can be done by calculating the equivalent signal processor multiply accumulate (MAC)-rate, $N \cdot F_s$. Realizing the 33-tap filterexample (corresponding to a 27-tap infinite precision filter) in the above defined 4 performance classes, results in MAC-rates of 672 MHz, 928 MHz, 1.22 GHz and 1.41 GHz, using 538, 763, 922 and 1081 CLB respectively. With these relative modest amount of ressources, the hardware methology thus generates filterarchitectures with MAC-rates, between one and two order of magnitude larger than state-of-the-art signal processors.

Conclusions

A new multiplier-free filtersynthesis method with $\mathcal{O}(filter\ order)$ complexity has been presented. Despite the low algorithm complexity, results compares well with other known approaches in most situations. Furthermore, a hardware methodology synthesizing minimized, wordparallel and bitparallel multiplier-free filter architectures has been presented. The total dedication to the Xilinx-architecture and DSP-algorithm leads to both efficiency and performance considerably above any known approaches. Efficiency is retained over a broad performance spectrum 20-50 MHz (16 bit). In general, the FPGA-technology is very promising as a future fundamental DSP-component, offering the best from the both the signal processor (programmability, flexibility) and semi/full custom VLSI-technology (speed, parallelism, dedication).

[1] S. Y. Kung. VLSI array processors. Prentice Hall, 1988.

[2] J. B. Evans. Efficient FIR-filter architectures suitable for FPGA implementation. IEEE transactions on CAS-II. Vol. 41, No. 7, 1994. pp. 490–493.

[3] B. Feher. Efficient synthesis of distributed vector multipliers. Microprocessing and microprogramming 38. 1993, pp. 345–350

[4] L. R. Rabiner et.al. Theory and Application of Digital Signal Processing. Prentice-Hall 1975.

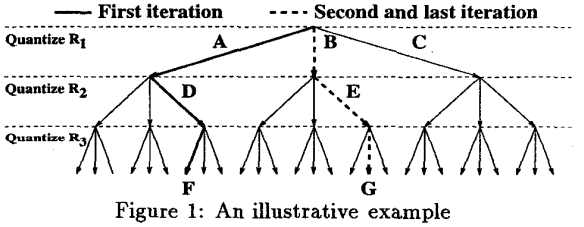

Figure 1: An illustrative example

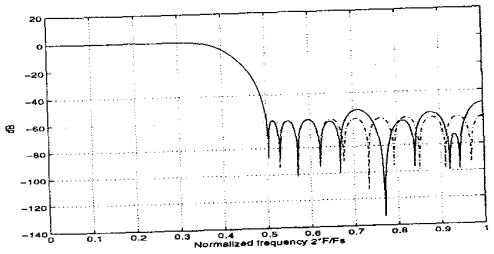

Figure 2: Frequency responses

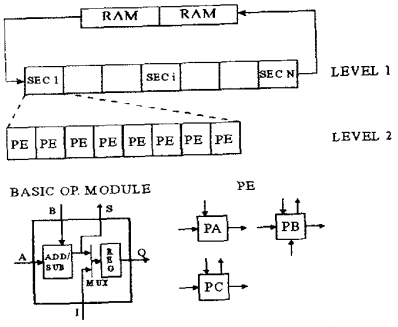

Figure 3: Systolic filterarray and PE's

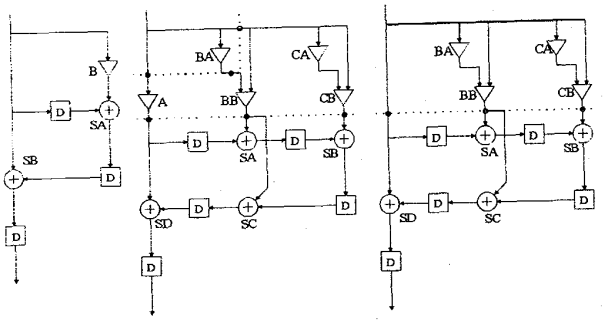

Figure 4: Systolized filter sections

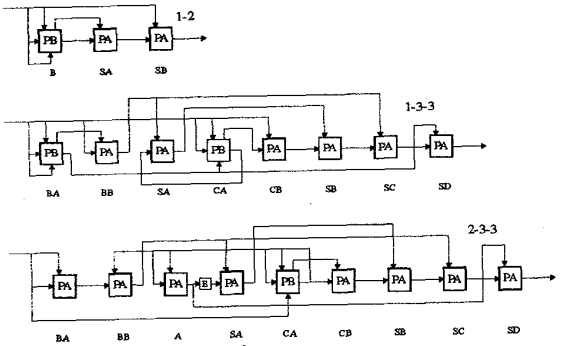

Figure 5: Section processorarrays

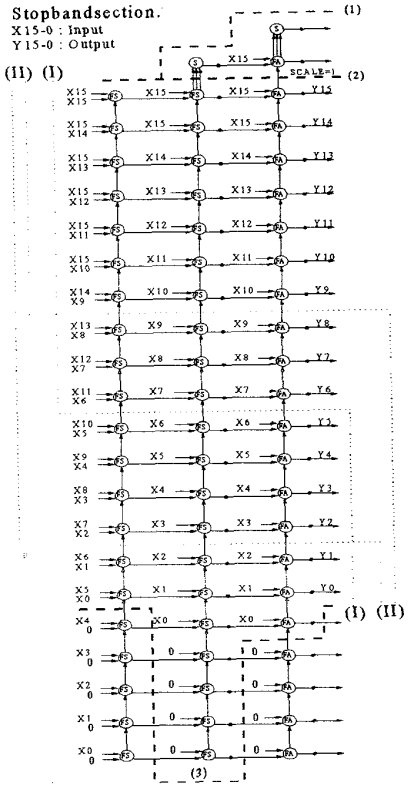

Figure 6: Bitlevel Structure Graph (BSG)

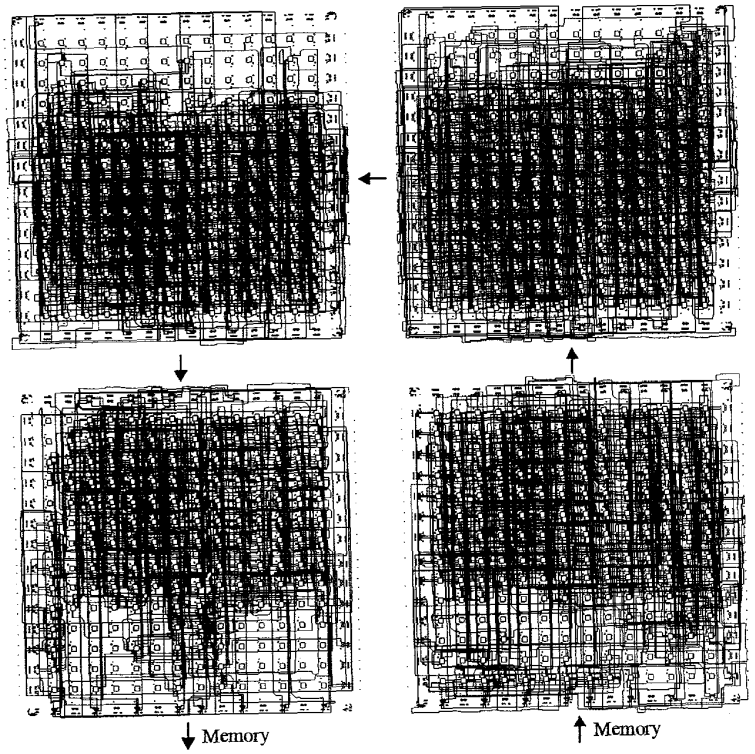

Figure 7: Chip plots of systolic filterarray