UNIVERSITY OF WESTMINSTER

WestminsterResearch
http://www.wmin.ac.uk/westminsterresearch

Efficient implementation of digital filters using novel reconfiguaration multiplier blocks (REMB).

Suleyman Demirsoy[1]
Andrew Dempster[2]
Izzet Kale[1]

[1]Cavendish School of Computer Science, University of Westminster

[2]School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia

Copyright © [2005] IEEE. Reprinted from Conference record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004: November 7-10, 2004, Pacific Grove, California, pp. 461-464.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Westminster's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners. Users are permitted to download and/or print one copy for non-commercial private study or research. Further distribution and any use of material from within this archive for profit-making enterprises or for commercial gain is strictly forbidden.

Whilst further distribution of specific materials from within this archive is forbidden, you may freely distribute the URL of WestminsterResearch. (http://www.wmin.ac.uk/westminsterresearch).

In case of abuse or copyright appearing without permission e-mail wattsn@wmin.ac.uk.

Efficient Implementation Of Digital Filters Using Novel Reconfigurable Multiplier Blocks

Süleyman Sırrı Demirsoy, Izzet Kale
Applied DSP and VLSI Research Group
University of Westminster
London, United Kingdom
{demirss, kalei}@wmin.ac.uk

Andrew G. Dempster
School of Surveying and Spatial Information Systems
University of New South Wales
Sydney, Australia
a.dempster@unsw.edu.au

***Abstract*— Reconfigurable Multiplier Blocks (ReMB) offer significant complexity reductions in multiple constant multiplications in time-multiplexed digital filters. In this paper the ReMB technique is employed in the implementation of a half-band 32-tap FIR filter on both Xilinx Virtex FPGA and UMC 0.18µm CMOS technologies. Reference designs have also been built by deploying standard time-multiplexed architectures and off-the-shelf Xilinx Core Generator system for the FPGA design. All designs are then compared for their area and delay figures. It is shown that, the ReMB technique can significantly reduce the area for the multiplier circuitry and the coefficient store, as well as reducing the delay.**

I. INTRODUCTION

Digital filters are by nature multiply-add intensive. The methodology, techniques and procedures deployed in realizing the associated hardware that undertake these tasks have matured over the years and are predominantly bound on utilizing the standard multiplier and adder structures in either fully parallel or time-multiplexed resource-sharing architectures.

However, there still remains a lot of redundancy in the arithmetic circuits and their associated computations as there is little sharing of the low-level intermediate calculations. The multiplier block approach has addressed this gap and has resulted in significant reduction in power, area and delay of the multiple constant multiplications in the fully-parallel structures [2]-[4].

Time-multiplexed designs are more efficient in terms of the resources needed. They time-share the available hardware, usually a multiply-accumulate block and a memory. FIR filters, IIR filters, filter-banks, poly-phase filters, adaptive filters can all be implemented as time-multiplexed structures. Figure 1(a) and (b) display two time-multiplexed filter architectures, namely Time Delay and Accumulate (TDA) and Tapped Delay Line (TDL) architectures.

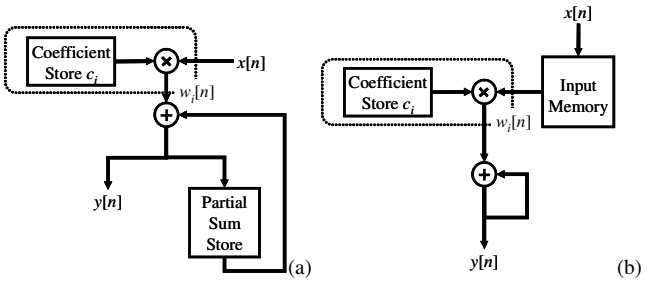

Figure 1 (a) Time-multiplexed TDA (transposed-direct form) FIR filter implementation, (b) Time-multiplexed TDL (direct-form) FIR filter implementation

In recent years, the application of the multiplier blocks to the time-multiplexed digital filter designs was also studied [1]. It was shown that, the redundancy can be reduced and the resulting specialized multiplier design can be much more efficient in terms of area and computational complexity compared to the general-purpose multiplier with its associated coefficient store. This novel methodology was named Reconfigurable Multiplier Blocks (ReMB) [1].

To apply the ReMB method to the time-multiplexing systems, the coefficient store and the general-purpose multiplier in Fig. 1(a) and (b) were replaced by a multiplier block, which generates all the coefficient products, and a multiplexer select the required one as depicted in Fig 2(a). Initially, this method seems to incur redundancy due to wasting all the generated products but the selected one. However, it is shown that, by pushing the multiplexer deep into the multiplier block design, the redundancy can be reduced and the resulting specialized multiplier design can be more efficient in terms of area and computational complexity compared to the general-purpose multiplier plus the coefficient store [1].

Fig 2(b) shows a multiplier block that generates 784, 156, 600 and a multiplexer to select the desired coefficient product. It uses five adders and a 3-to-1 multiplexer. By pushing the multiplexing operation into the multiplier block, the same functionality can be implemented using three adders and three 2-to-1 multiplexers as given in Fig 2(c).

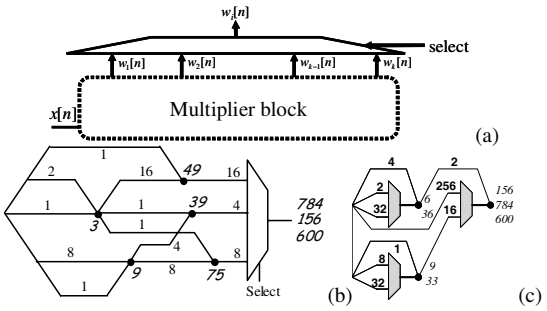

Figure 2 (a) Substitution of general-purpose multiplier plus the coefficient store of Fig. 1 with a multiplier block and a multiplexer, (b) a multiplier block with a multiplexer to choose one of the available outputs, (c) a reconfigurable multiplier block that produces the same outputs as (b).

A multiplexer connected to an input of an adder (in this context, adder refers either to an adder, subtractor or an adder/subtractor) together form the basic structure of the reconfigurable multiplier blocks. The size of the multiplexer and the functionality of the adder depend on the platform and the design criteria.

In Fig 2 (b) and (c) each node (•) corresponds to and adder. The edges represent the inputs to the adder. The numbers given at each edge shows the multiple of the signal achieved by a left-shift. If it is negative, then that particular signal is subtracted. The italic numbers next to the nodes are the product(s) generated by those nodes.

The area of a 2-to-1 multiplexer is considerably smaller than a full-adder for CMOS VLSI implementation. Moreover, the unnecessary evaluations of the products are avoided. Implementation of these circuits on Field Programmable Gate Arrays (FPGA) will benefit from the fixed resource FPGA environment. As an example, one of the most common FPGA platforms, the Xilinx Virtex device family contains Configurable Logic Blocks (CLB) with Look-Up Tables (LUT) to implement the combinational logic with up to four inputs. It also has dedicated circuitry around the LUT for fast addition and multiplication as shown in Fig 3(a). By utilizing the dedicated circuitry, a full-adder can be implemented using one LUT as an XOR gate. However, it is also possible to fit the 2-to-1 multiplexer to the same LUT as in Fig 3(b), reducing the area requirement by more than 50% for the design given in Fig 2(c).

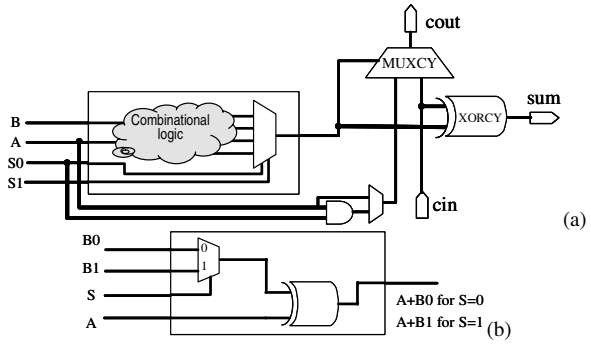

Figure 3 (a) A half-slice of the Virtex CLB (b) Configuration of an LUT for implementing a basic-structure

In this paper, we apply ReMB technique to a 32-tap half-band FIR filter to demonstrate its benefits on both FPGA and ASIC implementations. Furthermore, we implement two reference designs of the same filter using standard time-multiplexed filter architectures. For the FPGA implementation, we also and compare our design with the readily available, off-the-shelf implementation with Xilinx Core Generator system. For the ASIC implementation, the proposed and the reference designs are implemented in UMC 0.18um CMOS technology. Section 2 of the paper will give the design details of the reference filters and the ReMB filters for fixed-point implementation. Section 3 will discuss the implementation issues specific for FPGA and ASIC. The area and delay figures for all designs and comment on the savings achieved by ReMB technique are also reported. Section 4 will conclude the paper.

II. DESIGN DETAILS

We designed the 32-tap half-band FIR filter in Matlab. Due to the nature of the half-band filter, the coefficients are symmetric and every other coefficient is zero except the middle coefficient.

For the fixed-point implementation of this filter, we quantized the coefficients to 10-bits rounding the exact coefficients towards the nearest integer. Assumed data word-length is 16 bits. The main reason for the 10-bits coefficient word-length was the algorithm that generated the ReMB structure.

Fig 4 displays the quantized coefficient set. There are nine distinct numbers in this set: 256, 162, -50, 26, -15, 8, -4, 2, -1.

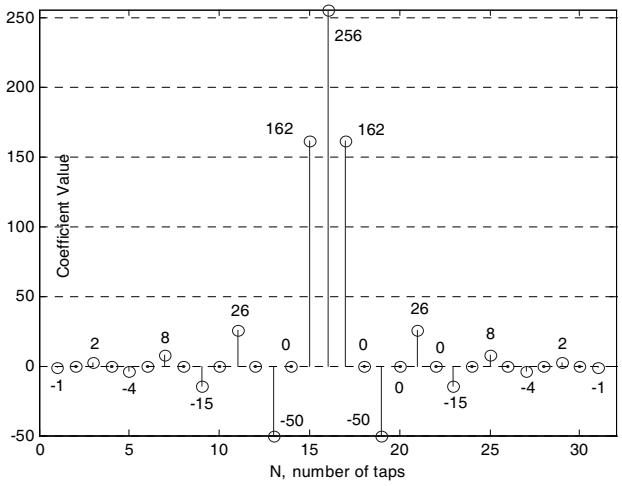

Figure 4 The quantized coefficients for the half-band FIR filter.

A typical time-multiplexed TDL filter architecture, which is used as a reference design, is shown in Fig 5(a). All the coefficients are stored in a coefficient memory and the incoming input samples are stored in an input memory. A simple controller operates to process one filter-tap per-cycle. In Fig 5(b), only the distinct non-zero coefficients (their

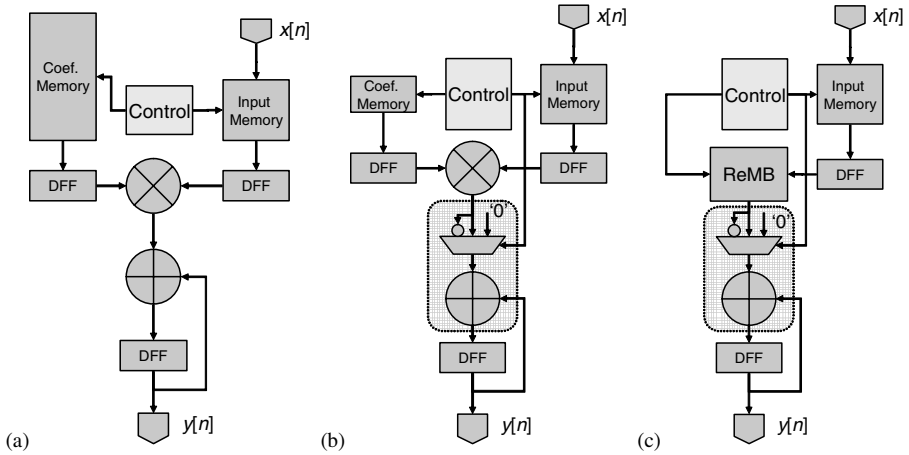

Figure 5 (a), (b) Time-multiplexed TDL strucutres implemented as reference filters (c) The proposed filter implementation using ReMB technique

absolute value) are stored in the memory. The controller in this case gets a bit more intelligent to address the correct coefficient for each tap. A multiplexer selects either the coefficient-product or its inverse or zero to be accumulated. The sizes of the controller block and the coefficient memory reflect the difference between the architectures of Fig 5(a) and (b).

Fig 5(c) shows the proposed implementation of the filter using ReMB. The coefficient store and the general-purpose multiplier in Fig 5(b) are replaced with a ReMB structure that performs multiplication for the distinct coefficients stored in the coefficient memory. The complexity of the controller is kept same since it generates the same control signals as in Fig 5(b).

The ReMB block used in the filter is shown in Fig 6. It is generated by an algorithm described in [1]. It comprises seven basic structures of the smallest size (a 2-to-1 multiplexer connected to one input of an adder). The coefficients of the filter are generated at the output of the basic-structure at layer 3 by selecting particular inputs of the multiplexers. Select signals are not shown on the diagram for simplicity.

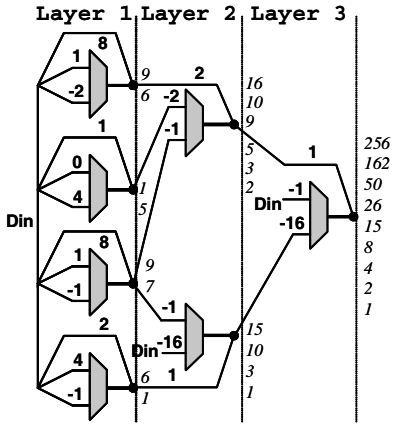

Figure 6 The ReMB design used in the proposed filter implementation. Din is the input signal to the block.

Table 1 shows the set of select values required to produce each coefficient of the filter. In the table, basic structures are indexed from 0 to 6 starting from the top of layer 1 downward and then layer 2 and layer 3. The select value of '0' means that the top branch of the multiplexer is selected. An 'X' value means that particular basic structure is not involved in generating the coefficient.

A separate decoder is designed to produce these select signals by using the output of the main controller that was used to address the coefficient memory.

TABLE I SELECT VALUES REQUIRED FOR EACH COEFFICIENT

Coef	Select signals for each basic-structure						
	0	**1**	**2**	**3**	**4**	**5**	**6**
256	0	0	1	X	0	0	0
162	1	1	0	X	0	0	0
50	1	1	0	0	0	1	0
26	1	0	0	1	0	1	0
15	0	0	X	X	0	X	1
8	0	X	X	0	1	X	1
4	1	X	X	1	1	X	1
2	1	X	X	0	1	X	1
1	1	1	X	X	0	X	1

III. IMPLEMENTATION

The filters are implemented in VHDL using HDL Designer™ and synthesized using Leonardo Spectrum™. The FPGA implementations are realized on a Virtex FPGA with model number XCV300BG432-4. They are Placed and Routed (PAR) using Xilinx ISE 5.2 software. Area and delay figures reported in this section for the FPGA designs are obtained after PAR.

Furthermore another reference design using off-the-shelf Xilinx Coregen™ software is also implemented from the parametrizable MAC FIR core (version. 3.0) [5].

The ASIC implementations are targeted for the UMC 0.18um CMOS technology. They are not placed and routed and all the results reported here are obtained after synthesis.

TABLE II AREA AND DELAY FIGURES FROM THE FILTER IMPLEMENTATIONS

	FPGA Implementations (Virtex XCV300BG432-4)				ASIC Implementations (UMC 0.18um CMOS)		
Filters	**Coregen**	**Fig 5(a)**	**Fig 5(b)**	**Fig 5(c)**	**Fig 5(a)**	**Fig 5(b)**	**Fig 5(c)**
Area (comb. Logic)	299 LUT	223 LUT	231 LUT	190 LUT	1637 gates	1725 gates	1394 gates
Area (D Flip-Fops)	323 FF	61 FF	61 FF	55 FF	380 gates	380 gates	339 gates
Delay	11.88 ns *	27.3 ns	26.7 ns	23.6 ns	6.52 ns	6.39 ns	7.17 ns
Notes	Latency 40 cycles for 31 taps. There is pipelining in the circuit at several stages.	Area excludes the coefficient and input memory.	Area excludes the coefficient and input memory.	No coefficient memory is required. Area excludes input memory.	Area excludes the coefficient and input memory.	Area excludes the coefficient and input memory.	No coefficient RAM is required. Area excludes input memory.

There is no quantization in the data-path of the any of the designs. The input data are 16-bits and the output data are 30-bits wide with full-precision.

Table II displays the area and delay figures for all implementations. The units of area for the FPGA implementations are LUT count for combinational logic and D Flip-Flop (FF) count for the sequential logic. For the ASIC implementations, the area is reported in terms of gate count.

No pipelining is applied to the filters given in Fig 5. Critical path delays are reported for the full combinational logic in the multiply-and-accumulate circuits. However, the filter generated by Coregen™ is pipelined as the latency of the filter is more than the number of filter taps.

Table II also shows that, the coefficient memory and the input memory are not included to the area figures for the filters given in Fig 5. The area figure for the Coregen filter, on the other hand, includes the memory for coefficients but not the input data.

The effect of the increased controller complexity in Fig 5(b) can be observed in the area figures for both FPGA and ASIC implementations. However, the multiplexer in the multiply-and-accumulate path in Fig 5(b) only contributes to the area for the ASIC implementations. For the Virtex implementation, the components inside the dashed-line in Fig 5(b) and (c) can be fitted into one LUT, which in turn means the multiplexer comes free.

The area savings achieved by the ReMB technique for the FPGA and the ASIC implementations of this particular example is around 20%. The ReMB block given in Fig 6 is not optimal in the sense of the number of basic-structures [1]. A better ReMB design, which would share more intermediate partial-products, would increase the area savings.

The decrease in the critical path delay for the FPGA implementations is due to the reduced logic depth of the multiplier since the extra multiplexer stages between adders do not contribute to the delay. However, for the ASIC implementations, the critical path delay increased a little due to multiplexers. The reduced logic-depth of the adder network in the multiplier avoided a large increase in the delay. Reduced logic-depth also contributes to lower-power since less glitches are produced.

If the pipelining was considered, the delays associated with all implementations would be similar. Even then, the area of the ReMB filter would be smallest because of the addition of the same amount of latches or flip-flops to all of the filters.

IV. CONCLUSIONS

We have implemented a half-band 32-tap FIR filter using the ReMB technique and compared it with the reference designs for FPGA and ASIC implementations.

ReMB technique reduced area for both FPGA and ASIC implementations around 20 %.

The critical-path delay in the FPGA implementations is reduced due to efficient basic structure mapping and less logic depth in multiplier. However, the multiplexers resulted in a slight increase in the delay of ASIC implementation.

Pipelining the filter structure would reduce the delay of all circuits to be comparable with Coregen design. The ReMB filter would still be the smallest area if the pipelining was considered.

REFERENCES

[1] Demirsoy S. S., "Complexity reduction in digital filters and filter banks", Ph.D. Thesis, University of Westminster, October 2003

[2] Dempster A.G. and Macleod M.D., "Use of minimum-adder multiplier-blocks in FIR digital filters", *IEEE Trans. CAS-II,* vol. 42, no. 9, pp. 569-577, November 1995.

[3] Bull D.R. and D.H Horrocks, "Primitive operator digital filters", IEE Proceedings-G, vol. 138, no. 3, pp. 401-412, June 1991.

[4] Potkonjak M., M.B. Srivastava and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination", *IEEE Trans. on CAD of ICS*, vol. 15, no. 2, pp. 151-165, February 1996.

[5] Xilinx Inc, "MAC FIR 3.0", Product Specification, March 2003