Technology-Dependent Optimization of FIR Filters based on Carry-Save Multiplier and 4:2 Compressor unit

Burhan Khurshid and Roohie Naaz

***Abstract* - This work presents an FPGA implementation of FIR filter based on 4:2 compressor and CSA multiplier unit. The hardware realizations presented in this paper are based on the technology-dependent optimization of these individual units. The aim is to achieve an efficient mapping of these isolated units on Xilinx FPGAs. Conventional filter implementations consider only technology-independent optimizations and rely on Xilinx CAD tools to map the logic onto FPGA fabric. Very often this results in inefficient mapping. In this paper, we consider the traditional CSA-4:2 compressor based FIR filters and restructure these units to achieve improved integration levels. The technology optimized Boolean networks are then coded using instantiation based coding strategies. The Xilinx tool then uses its own optimization strategies to further optimize the networks and generate circuits with high logic densities and reduced depths. Experimental results indicate a significant improvement in performance over traditional realizations. An important property of technology-dependent optimizations is the simultaneous improvement in all the performance parameters. This is in contrast to the technology-independent optimizations where there is always an application driven trade-off between different performance parameters.**

***Index Terms*—FIR filters, FPGA, Look-up table, Technology Mapping, Carry-save Arithmetic**

Original Research Paper
DOI: 10.7251/ELS1620043K

I. INTRODUCTION

Finite Impulse Response (FIR) filters are basic components in many Digital Signal Processing (DSP) applications like multimedia & wireless communication, image processing, video & audio processing, speech recognition etc. [1], [2], [3]. Owing to their iterative nature, DSP computations differ drastically from general purpose computations. The non-terminating nature of DSP algorithms can be exploited to design efficient systems by exploiting the concurrencies both within iteration and among multiple iterations [4]. This has provided designers with sufficient impetus to look beyond the traditional software oriented solutions and consider some hardware platforms where the underlying resources can be utilized to develop a complete System on Chip (SoC) solution that best matches the algorithmic complexity by developing the right type of architecture.

Application Specific Integrated Circuits (ASIC) have long been used to develop custom architectures for realizing FIR filters. However, with ASICs the design flow is complicated and time consuming resulting in huge non-recurring engineering (NRE) costs [5]. This has typically reserved ASICs for high volume markets and for some specialized domains. FPGAs provide for reduced design time due to a simplified design flow and pre-fabricated nature which eliminates the requirement for verification of deep sub-micron effects. Some other advantages include reconfigurable design approach [6], [7], large scale integration [6], [8], availability of several intellectual property (IP) cores [9], reduced NRE costs [6], [7] etc. The design cycle in FPGAs has a strong computer aided design (CAD) support. The software handles the time consuming mapping, routing, placement and floor planning phases. The effectiveness of technology-independent optimizations that are generally well suited for ASICs [10] is thus limited in FPGAs. Thus, in order to get maximum performance from the target FPGA device, optimizations that are specific to the underlying technology have to be considered. This requires a complete knowledge about the target device. Also, the choice of the target device will have a prominent effect on the end performance of the system [11]. In this work, we carry out the hardware realization of 4:2 compressor and carry save adder (CSA) based multiplier units that are optimized for FPGAs with 6-input look-up tables (LUT). Since all modern FPGAs from Xilinx support 6-input LUTs [12], [13], [14], the filter realizations based on these individually optimized units should provide an improved performance.

FIR filters in general tend to have high arithmetic complexity due to the required number of multipliers, adders and delay elements. However, the main computational bottleneck is the multiply operation that requires a large

Manuscript received on July 20, 2015. Received in revised form on October 8, 2016. Accepted for publication on October 10, 2016.

Burhan Khurshid is with the Department of Computer Science and Engineering, National Institute of Technology Srinagar, Jammu and Kashmir, India, 190006 (phone: +91-9797875163; e-mail: burhan_07phd12@nitsri.net).

Roohie Naaz is with the Faculty of Computer Science and Engineering, National Institute of Technology Srinagar, Jammu and Kashmir, India, 190006 (e-mail: naaz310@nitsri.net).

computation time [15]. A variety of approaches have been used to speed-up the multiplier operation in a filter structure. These approaches either completely eliminate the existing multiplier unit or reduce its architectural complexity. A widely used multiplier-less approach is the one where the multiplier unit is replaced by some sort of memory. Two frequently used memory-based techniques are the direct ROM based implementation and distributed arithmetic (DA) based implementation. ROM based implementations replace the multipliers with LUTs resulting in faster output compared to multiply and accumulate design [16]. DA based techniques have high throughput processing capabilities and regularity resulting in cost-optimal structures [17], [18], [19]. In DA based multipliers the pre-computed partial products are stored in the memory elements and are later read out and accumulated to obtain the desired results. To minimize the logic requirements the authors in [20] present a new kind of FPGA implementation algorithm which is based on the Remainder theorem. Another approach uses the divided LUT method to decrease the computational complexity [21]. The problem with memory-based techniques is the increased on-chip memory requirements as the operand word-length increases. Low capacity FPGAs often switch to bit-serial arithmetic for realizing multiplier circuits [22]. Special purpose bit-serial implementations include power-of-two sum or difference approaches. This allows multiplication to be replaced with faster shift and addition operations [23], [24], [25]. Linear systolic structures have also been used as bit-serial architectures [4]. In these approaches the conventional 2-D bit-parallel architectures are transformed into linear 1-D bit-serial systolic structures. The drawback with bit-serial structure is their reduced speed.

Apart from the multiplier-less approaches several techniques have been used to reduce the complexity of the multiplication operation. In [26] the authors present an improved multiplier design based on Canonic Signed Digit (CSD) representation and Horner's scheme. Two multiplier structures have been proposed: a cascaded adder structure and an accumulator structure. Both take advantage of the fixed nature of filter coefficients and reduce the number of partial products resulting in an area efficient realization. Similarly constant coefficient multipliers have been reported in [27], [28], [29]. Residue Number System (RNS) is yet another approach that has been used for designing efficient high-speed multipliers [30]. The limited inter-moduli carry propagation and parallel computations make RNS desirable for add/multiply intensive applications. Similarly, the authors in [31] propose a novel design scheme based on the combination of sum of power of two (SPT) coefficients and carry-save addition for implementing fast multiplier blocks.

At system level, one of the frequently used modifications is to develop systolic architectures for the filter structures. Systolic designs posses a high potential to yield a high throughput rate with features like simplicity, regularity and modularity of structure [32]. The authors in [33] and [34] take a systolic approach for filter design by using multipliers based on direct ROM and DA approaches. Similarly, the work in [35] uses the systolic structures but focuses on replacing the original adder unit by using a parallel prefix adder (PPA) with minimal depth algorithm. Poly-phase decomposition has been used to design high-speed and low-power parallel filters [36], [7], [37], [38], [39], [40]. A modification of poly-phase decomposition is the Fast FIR algorithm (FFA). Filters based on FFA are area efficient utilizing fewer multiplier units [41], [42].

All the above mentioned approaches use technology-independent optimizations to enhance the performance of the filtering structures. In this paper, we take an alternate low-level design approach and propose realizations that are based on technology-dependent optimizations.

The rest of the paper is organized as follows. Section II discusses the basic FIR structure. Section III discusses some of the preliminary terminologies used in this paper. Section IV discusses the technology-dependent optimization of the 4:2 compressor and CSA multiplier unit. Synthesis and implementation is carried out in section V. Conclusions are drawn in section VI and references are listed at the end.

II. BASIC FIR FILTER

An *N*-tap FIR filter is defined as:

$$\boldsymbol{y[n] = \sum_{k=0}^{N-1} h_k\, x[n-k]} \quad (1)$$

Where, h_k is the filter coefficient that determines the frequency behavior of the filter and *x[n-k]* are the time delayed samples of input sequence with $0 \le k \le N-1$. A direct mapping of equation 1 results in the Direct form realization of the FIR filter as shown in figure 1. The critical path of the Direct form consists of *N-1* adder units and one multiplier unit. Alternatively transposition theorem may be applied to obtain the Transposed form as shown in figure 2 [4]. The critical path in the Transposed form consists of one adder and one multiplier unit only. The Transposed and Direct form may be combined to have different Hybrid forms. However, in this paper we have only considered the realization of Transposed and Direct forms of FIR filters.

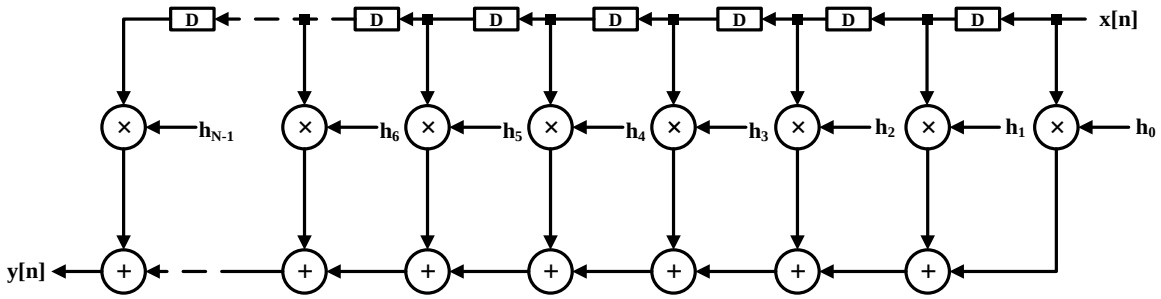

Fig. 1. Direct form realization of FIR filter

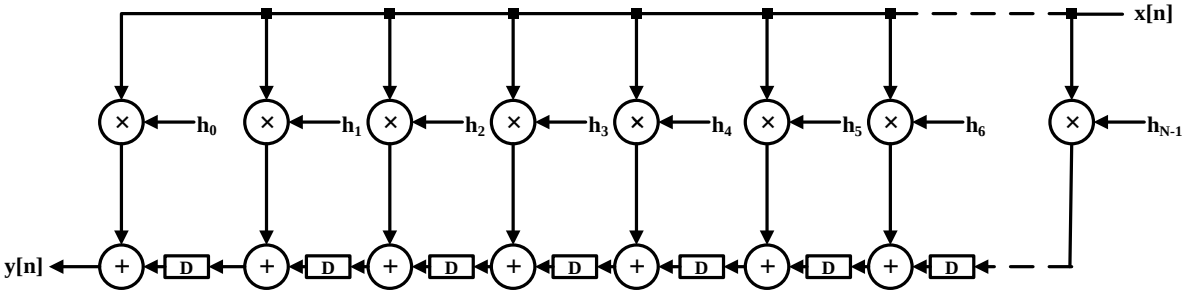

Fig. 2. Transposed form realization of FIR filter

III. Preliminary Terminologies

Logic synthesis is concerned with realizing a desired functionality with minimum possible cost. In the context of digital design the cost of a circuit is a measure of its speed, area, power or any combination of these. For graphical representations a combinational function may be represented as a directed acyclic graph (DAG) called the *Boolean network*. *Nodes* within this network represent logic gates, primary inputs (PI) and primary outputs (PO). Each node implements a *local function* and together with its predecessor nodes implements a *global function.* A *cone* of a node *v*, C_v, is a sub-graph that includes the node *v* and some of its non-PI predecessor nodes. Any node, *u* within this cone has a path to the root node *v*, which lies entirely in C_v. The *level* of the node *v* is the length of the longest path from any PI node to *v*. If node *v* is a PO node then the level will give the *depth* of the network. Thus network depth is the largest level of a node in the network. The critical path and area of a mapped Boolean network is measured by the depth and number of LUTs utilized by the network. A network is said to be *k-bounded* if the fan-in of every node does not exceed *k*.

IV. Technology-dependent Optimization

Technology-dependent optimization transforms the initial Boolean network into a circuit netlist that utilizes the target logic elements efficiently. The aim is to distribute the logic among the targeted elements with minimum possible depth and minimum resource utilization. The target element in majority of FPGAs is a *k*-LUT [43], [44]. An efficient utilization of this circuit element could lead to increased logic densities and reduced circuit depths.

Technology-dependent optimization using LUTs is a two step process. In the first step, the entire network is partitioned into suitable sub-networks. The individual nodes within each sub-network are then covered with suitable cones. The redundancies within each sub-network are exploited during the covering phase. The logic implemented by each cone is then mapped onto a separate LUT. In the second step, the netlist for the entire network is constructed by assembling the individually optimized sub-networks. The overall aim is to have a circuit implementation that uses minimum possible LUTs and has minimum possible depth.

The FIR structure considered in this paper is based on a multiplier unit and a 4:2 compressor unit. The multiplier is based on the CSA logic and generates two partial vectors. The 4:2 compressor then combines these partial vectors with those generated from the previous stage and generates two new partial vectors. A final adder stage at the output then combines these partial vectors and generates the final result. The CSA multiplier and 4:2 compressor ensure that there is no rippling of carry and the critical path is kept to a minimal. Direct and Transposed FIR structures based on these units are shown in figure 3 and figure 4 respectively. In each case, the input *x[n]* and the filter coefficients h_k are assumed to be in fixed-point 2's complement representation.

A. Technology-Dependent Optimization of the Multiplier unit

The multiplier unit is an array multiplier based on the carry-save logic. In carry-save logic the carry outputs are saved and used in the adder in the next row. This ensures that the additions at different bit-positions within a row are independent of each other. The details of a CSA multiplier are given in [4]. The schematic for 4-bit operands is shown in figure 5. The vector merging adder which is the main computational bottleneck in a CSA multiplier need not to be included as the partial sum and carry outputs are directly fed to the 4:2 compressor unit.

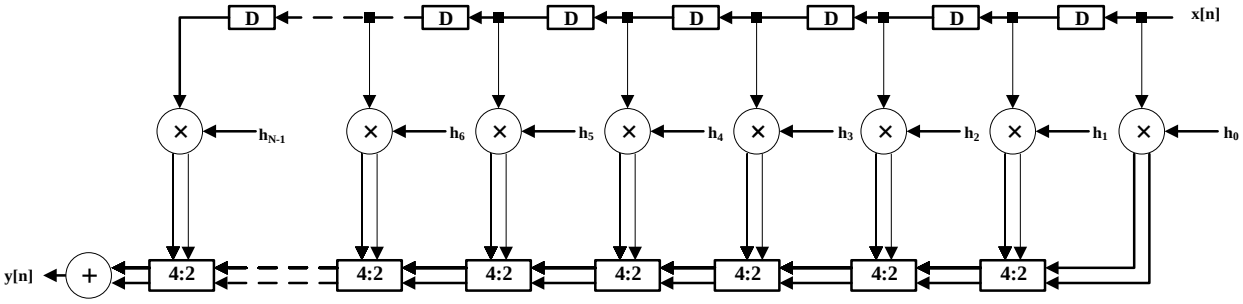

Fig. 3. Direct form FIR filter based on CSA multiplier and 4:2 compressor units

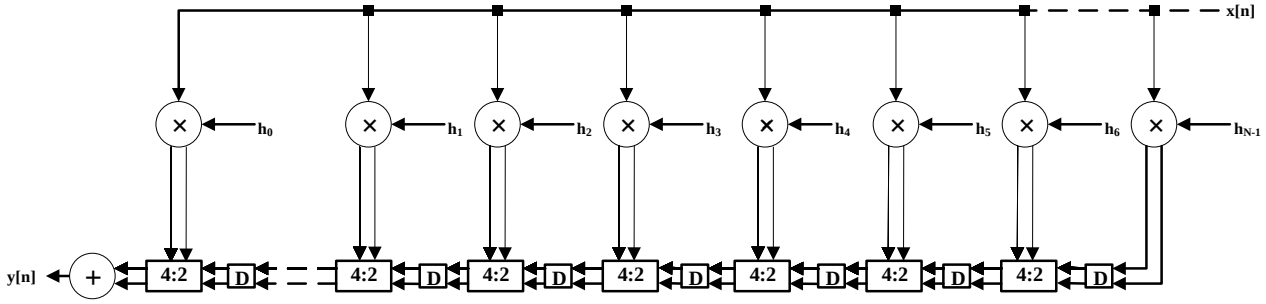

Fig. 4. Transposed form FIR filter based on CSA multiplier and 4:2 compressor units

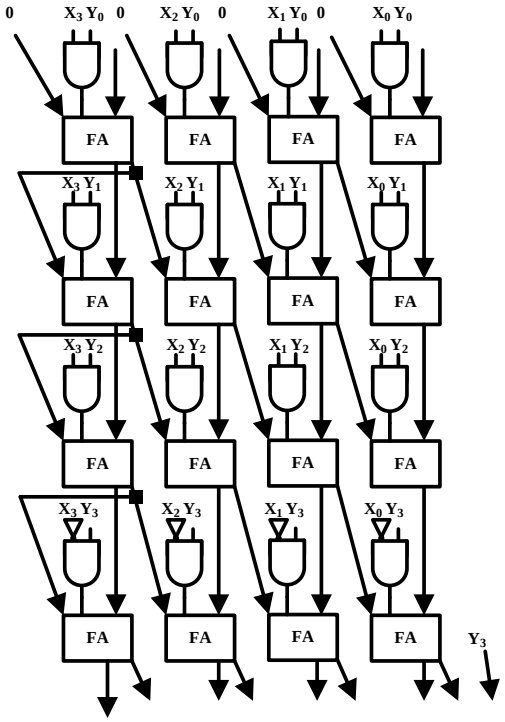

Fig. 5. Fixed-point 4-bit CSA multiplier

Figure 6 shows the Boolean network for the basic operating cell used in the CSA multiplier. The network is partitioned into two sub-networks corresponding to the sum (S) and carry (C) outputs. Each sub-network is separately mapped onto a circuit of LUTs by covering the individual nodes with suitable cones. A straight forward approach would be to cover each node within a sub-network with a separate cone. The sub-network is then traversed in a post-order depth-first fashion and the local function implemented by each cone is mapped onto a separate LUT. This is shown in figure 7(a). The overall depth at PO nodes S and C is three and the LUT count is seven. The number of LUTs may be reduced by decomposing the 3-input OR gate in the carry sub-network and duplicating the AND gate in each of the sum and carry sub-networks. The decomposed node is included in two separate cones and the sub-network is again traversed in a post-order depth-first fashion resulting in the realization of figure 7(b). The shaded nodes represent the duplicated logic. The circuit depth is now two and the LUT count is three. However, an optimal implementation may be obtained by exploiting the reconvergent PI nodes in the carry sub-network. Reconvergent nodes share the same inputs and can be exploited to reduce the number of PIs to a sub-network by realizing such paths within the LUT. This is shown in figure 7(c). The depth of the circuit is now reduced to one and the total LUT count is reduced to one as both sum and carry sub-networks are mapped onto a single 6-LUT with dual outputs. An $n \times n$ multiplier implemented using the realization of figure 7(c) will require n^2 LUTs and will have an overall depth of n.

It was mentioned in the introduction that the design cycle in FPGAs has a strong CAD support that handles the majority of the technology-dependent steps like mapping and placement and route (PAR). Technology-dependent optimizations mainly focus on improving the mapping of Boolean networks onto target LUTs. However, with modern CAD tools, both technology mapping and PAR are automated and the optimization process is not transparent to the user [14]. Thus any optimization done prior to the design entry may get over-rid during the mapping and PAR phases. To counter this issue, we re-define the coding strategy at the design entry phase. Instead of writing conventional behavioral codes, which are inferential in nature, we adopt an instantiation based coding

strategy, wherein a target element is directly called upon and the desired functionality is assigned to it. This ensures a controlled mapping. The following instantiations were used to map the circuit in figure 7.c.

LUT_1: LUT6_2 generic map (INIT => X"96660000E8880000") port map (C, S, c, s, b, a, '1', '1');

B. *Technology-Dependent Optimization of the 4:2 Compressor unit*

A 4:2 compressor unit takes four inputs and produces two output values, sum (S) and carry (C). A 4:2 compressor can be realized using a carry-save stage and a final ripple-carry stage as shown in figure 8. The carry save stage can be easily pipelined by inserting registers between the subsequent stages. The registers are shown as small dots in the schematic of figure 8. The critical path for such a realization consists of the delay associated with the final ripple carry chain. An *n*-bit 4:2 compressor unit will require 2*n* full adder units and will have a critical path of *n* full adders. Each full adder requires two LUTs; one for the sum and the other for the carry. Thus total LUT cost for an *n*-bit 4:2 compressor would be 4*n*. Assuming each LUT has a unit delay, the total critical path for such a realization would be *n*. Further *n* flip-flops would be required for effective pipelining of the structure.

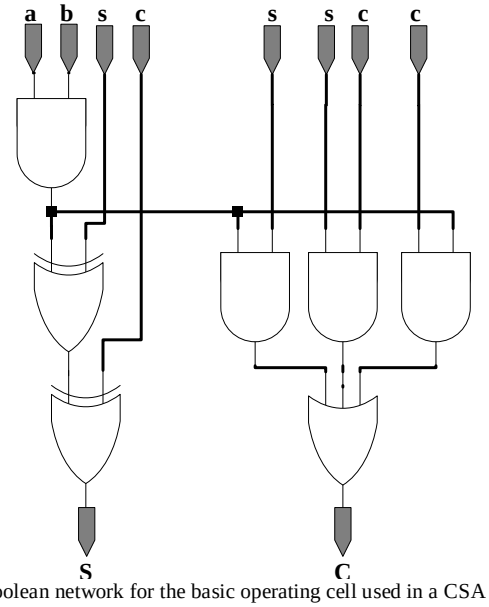

Fig. 6. Boolean network for the basic operating cell used in a CSA multiplier

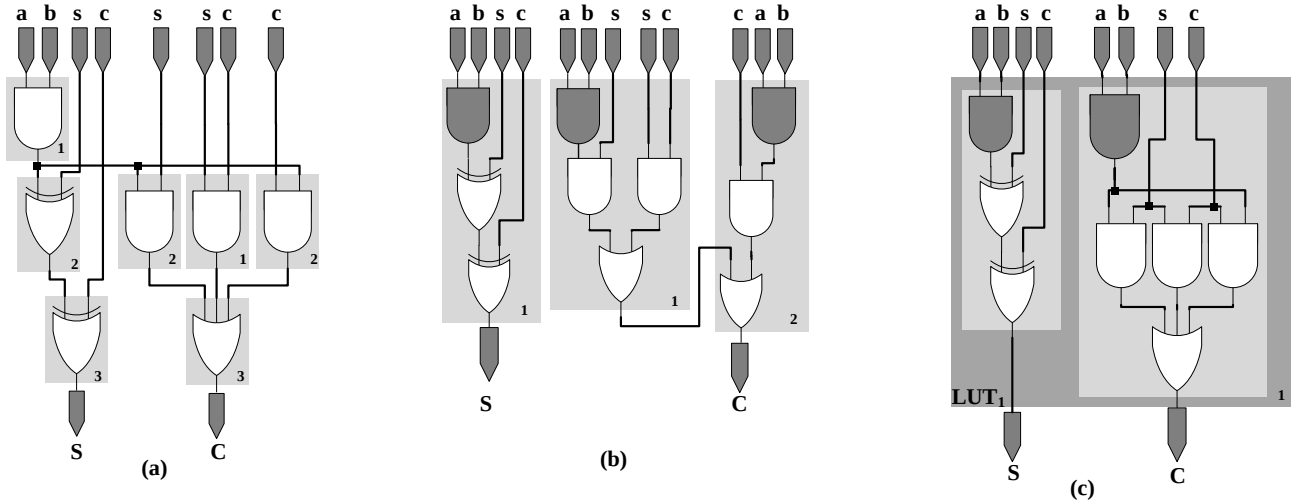

Fig. 7. Technology optimization for the basic cell. a) Direct mapping. b) Using node decomposition and logic duplication. c) Using reconvergent paths.

For technology-dependent optimization, we first consider the covering process at a higher level. The adder units are covered in an inclined fashion as shown in figure 9. Each covered portion implements a two-stage ripple carry chain, with the difference that the sum output is being rippled and the carry output is retained. The critical path of such a realization will consist of the delay associated with each covered portion. Pipelining the circuit would require placing registers at the output node of each shaded portion. This is shown as small dots in figure 9. The entire structure would require approximately *n* flip-flops for pipelining.

Let us now consider the covering process at a lower level. Figure 10(a) and 10(b) shows the block diagram and the corresponding parent network for a single shaded portion. For technology-dependent optimization we again directly re-structure the parent network. This is done by duplicating the logic at node *Z*. Node duplication enables the parent network to be partitioned into three separate networks corresponding to the outputs Out_1, Out_2 and Out_3. Each network is covered with a suitable cone and the logic function implemented by each cone is mapped onto a separate LUT. The overall realization is shown in figure 11. LUT_1 is used to realize a single three-input function. LUT_2 is used in the dual mode to realize two five-input functions corresponding to the outputs Out_1 and Out_3. The overall depth of the circuit is one. Since the critical path of the 4:2 compressor unit is limited by the depth of each shaded portion, the overall critical path is one. Also note that the LUT cost is two. The 4:2 compressor will thus have the same hardware utilization as a binary adder.

The following instantiations were used to map the circuit in figure 18(c).

LUT_1: LUT3_L generic map (INIT => X"E8") port map (Out_2, c, b, a);

LUT_2: LUT6_L generic map (INIT => X"E88E8EE896696996") port map (Out_1, Out_3, e, d, c, b, a,

'1');

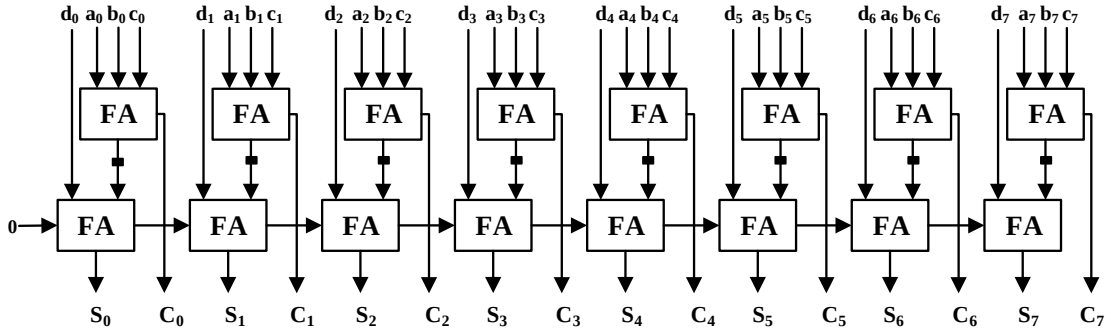

Fig. 8. Top-level schematic for an 8-bit 4:2 compressor unit based on CSA logic

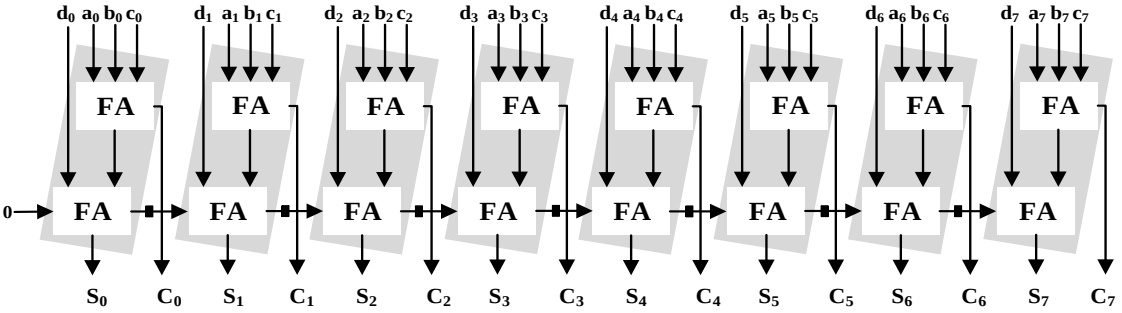

Fig.9. Covering of individual adder units. Each shaded portion represents a two-stage RCA chain

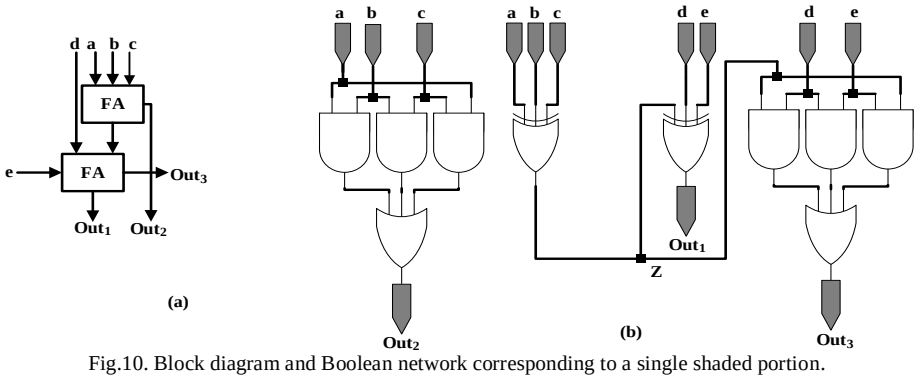

Fig.10. Block diagram and Boolean network corresponding to a single shaded portion.

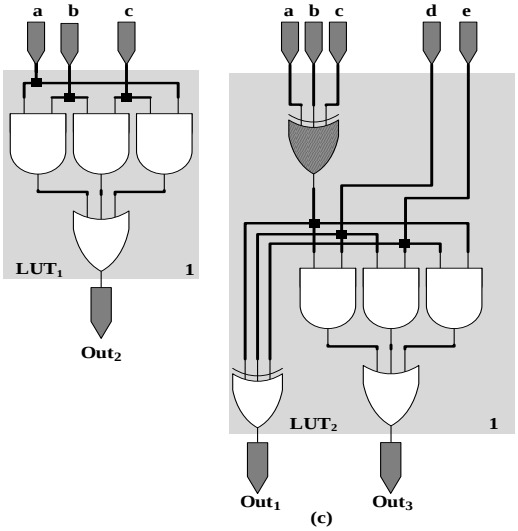

Fig. 11. Technology optimized realization for each shaded portion

V. Synthesis, Implementation and Results

The implementation in this work targets 6-LUT FPGAs. In particular we have considered devices from Virtex-5 and Virtex-6 FPGA families from Xilinx. The implementation is carried for different filter orders and an operand word-length of 16 bits. The parameters considered are area, timing and power dissipation. Area is measured in terms of different FPGA resources utilized. Timing analysis may be static or dynamic. Static timing analysis gives information about the critical path delay and operating frequency of the design. Static timing analysis is done post synthesis and post PAR. However, the metrics obtained after synthesis are often not accurate enough due to the programmability of the FPGA which allows for interconnect delays to change significantly between iterations. Therefore, the metrics presented in this paper are post PAR and have been recorded for a high optimization level with area and speed as optimization goals. Dynamic timing analysis verifies the functionality of the design by applying test vectors and checking for correct output vectors. Dynamic timing analysis gives information about the switching activity of the design, which is captured in the value charge dump (VCD) file. Power dissipation is given by the sum of static power dissipation and dynamic power dissipation. Static power dissipation is device specific and is mainly determined by the specific FPGA family. Dynamic power dissipation is related to the charging and discharging of capacitances along different logic nodes and interconnects. Dynamic power dissipation mainly consists of the logic power, clock power and signal power [45]. Logic power depends on the amount of on-chip resources being utilized by the design. Clock power is proportional to the operating frequency. Signal power depends on the switching activity and the density of the interconnects. For simulation and metrics generation similar test benches have been used and are typically designed to represent the worst case scenario (in terms of switching activity) for data entering into the filter. Design entry is done using VHDL. As mentioned earlier instantiation based coding strategy is used. The constraints relating to synthesis and implementation are duly provided and a complete timing closure is ensured. Synthesis and implementation is carried out in Xilinx ISE 12.1 [46]. Power analysis is done using the Xpower analyzer tool.

A. Area Analysis

Table 1 provides a comparison of the different FPGA resources utilized by the technology optimized FIR filter against the traditional implementation and the one based on the Xilinx multiply-adder IP v 2.0. The operand length in each case is 16 bits and the filter order is 16. Target device is xc5vlx50-2ff324 from Virtex-5. Both technology optimized and traditional implementations rely on the optimization strategies of the Xilinx CAD tool, however, the initial restructuring ensures that the end performance is better in technology-dependent optimizations. Note that the multiply-adder IP v 2.0 is used with an optimum latency value (-1).

TABLE 1
RESOURCE UTILIZATION FOR TECHNOLOGY OPTIMIZED AND IP BASED TRANSPOSED FIR FILTER.

Filter Design	LUTs	Flip-flops	Slices	DSPs
Transposed [this work]	254 (223[A])	185 (185)	143 (93)	0
Transposed [traditional]	323 (307)	271 (253)	311 (289)	0
Transposed [IP v 2.0]	273 (267)	503 (503)	243 (243)	30

[A] When area optimized

Next we compare our implementation against the various realizations reported in [47] and [10]. In [47] the authors have considered the Direct, Transposed and Hybrid realizations of the FIR filter. Two sets of results have been reported; one for symmetric coefficients and the other for asymmetric coefficients. For symmetric coefficients Direct and Transposed forms have been considered for a filter order of 120. Each form is implemented using the architecture based on generic multiplier (GM) and shift-add (SA) operation. For asymmetric coefficients Direct, Transposed and three different Hybrid forms (Hybrid-I-3, Hybrid-II-15, and Hybrid-III-15) have been considered for a filter order of 75. Table 2 provides the comparison for symmetric coefficients and table 3 provides the comparison for asymmetric coefficients. The devices considered are xc5vlx50-2ff324 and xc6vlx75t-2ff484 from Virtex-5 and Virtex-6 respectively. It is observed that in each case the structures based on the technology optimized multiplier and 4:2 compressor units use the underlying fabric efficiently. It should be noted that the authors in [47] have used Xilinx ISE 13.1 design suite as the synthesis tool while our implementation is carried out in Xilinx ISE 12.1 design suite. Using a higher version will only enhance the performance of the designs.

In [10] the authors present an algorithm that achieves performance speed up by enabling an efficient use of the embedded arithmetic blocks and custom compression trees. Different filter realizations have been considered that include filters based on canonic signed digit (CSD) arithmetic utilizing 6:3 compression trees, systolic architectures for transposed realizations and filters designed using IP multiply-accumulate (MAC) units, unfolded MAC architecture and one generated through MATLAB Filter design and analysis (FDA) tool. Table 4 provides the area comparisons in terms of number of slices utilized. The target device is xc5vlx50-2ff324 from Virtex-5. Again the technology optimized structures show an efficient utilization of the underlying fabric. Apart from that only general logic elements are utilized and no special primitives or macro-support is consumed.

TABLE 2
RESOURCE UTILIZATION FOR DIFFERENT FILTER REALIZATIONS WITH SYMMETRIC COEFFICIENTS.

Filter Design	Architecture	XC5VLX50-2FF324 (Virtex-5)				XC6VLX75T-2FF484 (Virtex-6)			
		LUTs	FFs	Slices	DSPs	LUTs	FFs	Slices	DSPs
Direct [47]	GM	4357	1904	1542	35	2826	1904	938	59
	SA[A]	3838	1904	1274	--	3785	1904	1085	--
	SA[D]	4013	1904	1377	--	4021	1904	1146	--
Transposed [47]	GM	4236	3886	1169	48	3777	3886	1018	57
	SA[A]	5094	3886	1402	--	5020	3886	1332	--
	SA[D]	5070	3886	1390	--	5054	3886	1354	--
Direct [this work]	Tech. optimized[A]	2508	1904	1216	--	2535	1904	1956	--
	Tech. optimized[D]	4975	1904	1359	--	5047	1904	2241	--
Transposed [this work]	Tech. optimized[A]	**1802**	**1888**	**1094**	--	**1795**	**1857**	**914**	--
	Tech. optimized[D]	3952	1888	1286	--	3839	1857	1221	--

[A] When area optimized
[D] When delay optimized

TABLE 3
RESOURCE UTILIZATION FOR DIFFERENT FILTER REALIZATIONS WITH ASSYMMETRIC COEFFICIENTS.

Filter Design	XC5VLX50-2FF324 (Virtex-5)			XC6VLX75T-2FF484 (Virtex-6)		
	LUTs	FFs	Slices	LUTs	FFs	Slices
Direct [47]	3398	1184	1148	3166	1185	891
Hybrid-I-3 [47]	3821	1212	1179	3542	1212	985
Hybrid-I-15 [47]	3941	**546**	1105	3751	**538**	1020
Hybrid-I-15 [47]	4121	598	1120	3672	606	988
Transposed[47]	3617	2204	980	3585	2204	955
Direct [this work][A]	1538	1184	1137	1553	1184	1189
Direct [this work][D]	2980	1192	1420	3088	1214	1515
Transposed [this work][A]	**1362**	1222	**960**	**1390**	1222	**871**
Transposed [this work][D]	2479	1222	1084	2759	1222	918

TABLE 4
RESOURCE UTILIZATION FOR DIFFERENT FILTER REALIZATIONS

Filter Design	Filter Order	Bit-Slices	DSP Blocks
CSD 6:3 compressor [10]	7	444	--
Transposed Systolic [10]	7	114	7
2-Unfolded MAC [10]	7	14	14
FDA HDL pipelined [10]	18	653	18
IP Systolic MAC [10]	18	548	9
Transposed CSD pipelined [10]	18	1071	--
Pipelined DSP48 MAC [10]	18	14	18
Direct form [This work]	7	162	--
Transposed form [This work]	7	132	--
Direct form [This work]	18	320	--
Transposed form [This work]	18	240	--

B. Timing Analysis

Table 5 provides a comparison of the critical path delay and maximum clock frequency for the technology optimized FIR filter against the traditional implementation and the one based on the Xilinx multiply-adder IP v 2.0. The realization considered is transposed. The operand length is 16 bits and the filter order is 16. Target device is xc5vlx50-2ff324 from Virtex-5.

Tables 6 and 7 provide a comparison of the critical path delay for the technology optimized realization and those reported in [47]. Table 6 is for symmetric coefficients with a filter order of 120 and table 7 is for asymmetric coefficients with a filter order of 75. The devices considered are xc5vlx50-2ff324 and xc6vlx75t-2ff484 from Virtex-5 and Virtex-6 respectively. The input operand length in each case is 16 bits.

Technology optimized structures are implemented with minimum possible depth; therefore, the critical path delays are quite low. Since clock frequency is also a strong function of the propagation and routing delays associated with the critical path, a minimum depth circuit also ensures high operating frequencies. This is indicated in table 8 where maximum clock frequency is compared against the various designs implemented in [10].

TABLE 5
TIMING ANALYSES FOR TECHNOLOGY OPTIMIZED AND IP BASED TRANSPOSED FIR FILTER

Filter Design	Critical path (ns)	Max. clock frequency (MHz)
Transposed [This work]	6.632	581.654
Transposed [Traditional]	10.541	375.38
Transposed [IP v 2.0]	8.751	480.517

TABLE 6
CRITICAL PATH DELAY FOR DIFFERENT FILTER REALIZATIONS WITH SYMMETRIC COEFFICIENTS

Filter Design	Architecture	Critical path delay (ns)	
		XC5VLX50-2FF324	XC6VLX75T-2FF484
Direct [47]	GM	29.7	29.9
	SA[A]	37.6	31.2
	SA[D]	27.7	23.7
Transposed [47]	GM	11.5	8.8
	SA[A]	15.8	11.7
	SA[D]	10.9	8.4
Direct [this work]	Tech. optimized[A]	30.873	27.547
	Tech. optimized[D]	21.64	17.287
Transposed [this work]	Tech. optimized[A]	11.741	7.946
	Tech. optimized[D]	**8.088**	**4.66**

C. Power Analysis

Technology-dependent optimization reduces the power dissipation in two ways. First, the high activity switching nodes within a network are hid within the LUTs in the final circuit netlist. This reduces the overall switching activity associated with the logic nodes [48]. Second, technology-dependent optimization results in a minimal depth circuit with a high logic density. This reduces the length of interconnects. Since interconnects in FPGAs are reconfigurable switches, there is a further reduction in the switching activity and thus the power dissipated. The analysis is done for a constant supply voltage and maximum operating frequency in each case. Test benches were designed for worst-case switching activity and the filter functionality was verified for more than 1000 input signals. The design node activity from the simulator database along with the power constraint file (PCF) was used for power analysis in the Xpower analyzer tool. Table 9 gives the detailed power dissipation for the technology optimized FIR filter against the traditional implementation and the one based on the Xilinx multiply-adder IP v 2.0. The target device is Virtex-5 and the filter order and input bit-width is 16.

The power dissipated in clocking resources varies with the clock frequency. Since technology optimized design operates at slightly higher frequency, the power dissipated by clocking resources is more. Power dissipated by on-chip resources

TABLE 7
CRITICAL PATH DELAY FOR DIFFERENT FILTER REALIZATIONS WITH ASSYMMETRIC COEFFICIENTS

Filter Design	Critical path delay (ns)	
	XC5VLX50-2FF324	XC5VLX50-2FF484
Direct [47]	25.5	27.3
Hybrid-I-3 [47]	22.8	26.4
Hybrid-I-15 [47]	22.8	25.2
Hybrid-I-15 [47]	22.2	23.3
Transposed[47]	11.4	12.1
Direct [this work][A]	33.79	31.06
Direct [this work][D]	20.82	22.528
Transposed [this work][A]	11.741	9.916
Transposed [this work][D]	**9.028**	**7.62**

TABLE 8
MAXIMUM CLOCK FREQUENCY FOR DIFFERENT FILTER REALIZATIONS

Filter Design	Filter Order	Clock frequency (MHz)
CSD 6:3 compressor [10]	7	182.3
Transposed Systolic [10]	7	423
2-Unfolded MAC [10]	7	535
FDA HDL pipelined [10]	18	302.51
IP Systolic MAC [10]	18	500
Transposed CSD pipelined [10]	18	520
Pipelined DSP48 MAC [10]	18	535
Direct form [This work]	7	523.04
Transposed form [This work]	7	**605.692**
Direct form [This work]	18	468.92
Transposed form [This work]	18	560.51

(logic + DSP) is lesser for technology optimized design because of the efficient utilization of the underlying resources. A reduction in switching activity due to hiding of nodes and reduction of interconnects results in lower power dissipation in the signals.

TABLE 9
POWER DISSIPATION FOR TECHNOLOGY OPTIMIZED AND IP BASED FIR FILTERS

FPGA Resource	Power Dissipation (mW)		
	T'posed [This work]	T'posed [Traditional]	T'posed [IP v 2.0]
Clock	27.45	26.43	26.75
Logic	3.41	6.002	2.03
DSP	--	--	3.58
Signals	6.23	7.51	8.11
I/Os	9.12	9.14	11.1
Quiescent	529.91	529.91	529.91
Total	576.12	578.992	581.47

Tables 10 and 11 provide comparison of the power dissipation by the technology optimized realizations and those reported in [47]. Table 10 is for symmetric coefficients with a filter order of 120 and table 11 is for asymmetric coefficients with a filter order of 75. The devices considered are xc5vlx50-2ff324 and xc6vlx75t-2ff484 from Virtex-5 and Virtex-6 respectively. The input operand length in each case is 16 bits.

For high throughput DSP systems it is more appropriate to quantify the power efficiency through energy analysis. In [49] the authors define three energy related parameters for FIR systems. These include Energy per operation (EOP), which is the average amount of energy required to compute one operation; Energy throughput (ET) which is the energy dissipated for every output bit processed and Energy density (ED) which is the energy dissipated per FPGA slice. Tables 12 and 13 provide these metrics for the technology optimized realizations and those reported in [47]. The devices considered are xc5vlx50-2ff324 and xc6vlx75t-2ff484 from Virtex-5 and Virtex-6 respectively. The input operand length in 16 bits.

TABLE 10
POWER DISSIPATION FOR DIFFERENT FILTER REALIZATIONS WITH SYMMETRIC COEFFICIENTS

Filter Design	Architecture	Power Dissipation (mW)	
		XC5VLX50-2FF324	XC6VLX75T-2FF484
Direct [47]	GM	797	1477
	SA[A]	870	1559
	SA[D]	812	1498
Transposed [47]	GM	785	1516
	SA[A]	848	1537
	SA[D]	804	1503
Direct [this work]	Tech. optimized[A]	749.13	1242.97
	Tech. optimized[D]	737.72	1039.82
Transposed [this work]	Tech. optimized[A]	733.61	1023.97
	Tech. optimized[D]	**722.08**	**991.29**

TABLE 11
POWER DISSIPATION FOR DIFFERENT FILTER REALIZATIONS WITH ASSYMMETRIC COEFFICIENTS

Filter Design	Power Dissipation (mW)	
	XC5VLX50-2FF324	XC5VLX50-2FF484
Direct [47]	820	1523
Hybrid-I-3 [47]	762	1493
Hybrid-I-15 [47]	834	1551
Hybrid-I-15 [47]	787	1507
Transposed[47]	760	1487
Direct [this work][A]	745.78	1059.76
Direct [this work][D]	**658.54**	**851.26**
Transposed [this work][A]	755.65	1062.76
Transposed [this work][D]	740.25	962.08

TABLE 12
ENERGY ANALYSIS FOR DIFFERENT FILTER REALIZATIONS WITH SYMMETRIC COEFFICIENTS

Filter Design	Architecture	EOP (nJ)		ET (nJ/bit)		ED (nJ/Slice)	
		XC5VLX50-2FF324	XC6VLX75T-2FF484	XC5VLX50-2FF324	XC6VLX75T-2FF484	XC5VLX50-2FF324	XC6VLX75T-2FF484
Direct [47]	GM	23.671	44.16	0.012329	0.023	0.015351	0.047079
	SA[A]	32.712	48.64	0.017038	0.025333	0.025677	0.044829
	SA[D]	22.5	35.5	0.011719	0.01849	0.01634	0.030977
Transposed [47]	GM	9.027	13.34	0.004702	0.006948	0.007722	0.013104
	SA[A]	13.4	17.98	0.006979	0.009365	0.009558	0.013498
	SA[D]	8.763	12.625	0.004564	0.006576	0.006304	0.009324
Direct [this work]	Tech. optimized[A]	23.127	34.24	0.012045	0.017833	0.019019	0.017505
	Tech. optimized[D]	15.964	17.975	0.008315	0.009362	0.011747	0.008021
Transposed [this work]	Tech. optimized[A]	8.613	8.136	0.004486	0.004238	0.007873	0.008024
	Tech. optimized[D]	**5.84**	**4.62**	**0.003042**	**0.002406**	**0.004541**	**0.003784**

TABLE 13
ENERGY ANALYSIS FOR DIFFERENT FILTER REALIZATIONS WITH ASSYMMETRIC COEFFICIENTS

Filter Design	**EOP (nJ)**		**ET (nJ/bit)**		**ED (nJ/Slice)**	
	XC5VLX50-2FF324	**XC6VLX75T-2FF484**	**XC5VLX50-2FF324**	**XC6VLX75T-2FF484**	**XC5VLX50-2FF324**	**XC6VLX75T-2FF484**
Direct [47]	20.91	41.5779	0.017425	0.034648	0.018214	0.046664
Hybrid-I-3 [47]	17.3736	39.4152	0.014478	0.032846	0.014736	0.040015
Hybrid-I-15 [47]	19.0152	39.0852	0.015846	0.032571	0.017208	0.038319
Hybrid-I-15 [47]	17.4714	35.1131	0.01456	0.029261	0.015599	0.03554
Transposed	8.664	17.9927	0.00722	0.014994	0.008841	0.018841
Direct [this work][A]	25.19991	32.91615	0.021	0.02743	0.022164	0.027684
Direct [this work][D]	13.7108	19.17719	0.011426	0.015981	0.009655	0.012658
Transposed [this work][A]	8.872087	10.53833	0.007393	0.008782	0.009242	0.012099
Transposed [this work][D]	**6.682977**	**7.33105**	**0.005569**	**0.006109**	**0.006165**	**0.007986**

VI. CONCLUSIONS

This paper focused on the realization of FIR filters using technology optimized multiplier and 4:2 compressor unit. The results presented in this paper showed that technology-dependent optimizations have a direct impact on area, delay and power dissipation of the design. Different filter forms (Direct, Transposed and Hybrid) were implemented and it was shown that for a particular form, the technology optimized realizations will always have an improved performance. Another key feature of technology-dependent optimization is that the same optimization results in the improvement of all the performance parameters (area, speed and power). This is generally not the case with technology-independent optimization where there is always an application driven trade-off that drives the design process. However, performance speed-up through technology-dependent optimization strongly relies on the amount of control the designer has over the mapping process. In this paper, we tackled this issue by modifying the coding strategy and writing instantiation based codes to map the behavior of the optimized Boolean networks. This complicates the design entry and although an efficient mapping is achieved, a complete control over the mapping process still remains a bottleneck in technology-dependent optimizations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

REFERENCES

[1] A. Mirshekari and M. Mosleh, “Hardware Implementation of a Fast FIR Filter with Residue Number System,” 2nd International conference on Industrial Mechatronics and Automation, pp 312-315, 30-31 May, 2010.

[2] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing: Principles, Algorithms and Applications," Prentice Hall, 1996.

[3] A. Antonion, "Digital Filters: Analysis, Design and Application," McGraw Hill, 1993.

[4] K. K. Parhi, “VLSI Digital Signal Processing Systems Design and Implementation,” Wiley, 1999.

[5] R. Woods, J. McAllister, G. Lightbody and Y. Yi, “FPGA-based Implementation of Signal Processing Systems,” Wiley, 2008.

[6] R. Tessier and W. Burleson, “Reconfigurable Computing for DSP: A Survey,” Journal of VLSI Signal Processing, Vol. 28, pp. 7-27, 2001, Kluwer Academic Publisher.

[7] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung, “Reconfigurable Computing: Architecture and Design Methods,” IEEE Proceedings. Computer Digital Technology, Vol. 152, No. 2, March 2005.

[8] R. Naseer, M. Balakrishnan, and A. Kumar, “Direct Mapping of RTL Structures onto LUT-Based FPGAs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No. 7, July 1998.

[9] M. K. Jaiswal and R. C. C Cheung, “Area-efficient architectures for double precision multiplier on FPGA, with run-time-reconfigurable dual single precision support,” Microelectronics journal 44 pp. 421 – 430, 2013.

[10] H. M. Kamboh and S. A. Khan, "An Algorithmic Transformation for FPGA Implementation of High Throughput Filters," Proceedings of the 7th International Conference on Emerging Technologies, 2011.

[11] A. Perwaiz, S. A. Khan and H. M. Kamboh, "Optimization for Quantization & Embedded Resources on FPGA," Proceedings of International Conference on Emerging Technologies, 2011.

[12] Xilinx, “Virtex-5 Family Overview,” DS100 (v 5.0) Feb. 6, 2009. www.xilinx.com.

[13] Xilinx, “Virtex-6 Libraries Guide for HDL Designs,” UG623 (v 12.3) September 21, 2010. www.xilinx.com.

[14] Xilinx, “Spartan-6 Family Overview,” DS160 (v 2.0) October 25, 2011. www.xilinx.com.

[15] X. L. Hu, F.Wang, M.Zhang, "Hardware Implementation of FIR Filter," Proceedings of International Conference on Multimedia Technology, p. no. 341-343, July 26-28, 2011.

[16] P. K. Meher, “Low-Latency Hardware-Efficient Memory-Based Design for Large Order FIR Digital Filters,” 6th International Conference on Information, Communication and Signal Processing, 10-13 Dec., 2007.

[17] G. Wei and W. F. Ying, “The Implementation of FIR Low-Pass Filter Based on FPGA and DA,” 4th International Conference on Intelligent Control and Information Processing, Beijing, China, June 9-11, 2013.

[18] L. Zhao, W. H Bi and F. Liu, “Design of Digital FIR Band pass Filter Using Distributed Algorithm Based on FPGA,” Electronic Measurement Technology, 30(7):101-104, 2010.

[19] G. R. Goslin, “A Guide to using FPGAs for application-specific DSP performance,” in Xilinx application note, 1998.

[20] P. Shi and Y. J. Yu “Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders,” IEEE Circuits and Systems Society, vol. 58 Issue: 1, 126 – 136, Jan. 2011.

[21] N. Mu and G. Liu, “Study on the FPGA Implementation Algorithm of Effective FIR Filter Based on Remainder Theorem,” 2nd International Conference on Consumer Electronics, Communication and Networks, 21-23 April, 2012.

[22] Y. Zhon and P. Shi, “Distributed Arithmetic for FIR Implementation on

FPGA," International Conference on Multimedia Technology, Hangzhou, China, 2011.
[23] J. E. Carletta and M. D. Rayman, "Practical Considerations in the Synthesis of High Performance Digital Filters for Implementation on FPGAs," FPL 2002, LNCS 2438, pp. 886-896, Springer, 2002.
[24] J. Chou, S. Mohanakrishnan and J. B. Evans, "FPGA Implementation of Digital Filters," Proceedings 4th International Conference on Signal Processing Applications and Technology, 1993.
[25] R. Woods, S. Ludwig, J. Heron, D. Trainor and S. Gehring, "FPGA Synthesis on the XC6200 using IRIS and Trianus/Hades," Proceedings 5th IEEE Symposium on FPGA based Custom Computing Machines, pp. 155-164, 1997.
[26] X. Yuan, T. Ying and G. Chunpeng, "Improved Design of Multiplier in the Digital Filter," International Conference on Computer and Communication Technologies in Agriculture Engineering, 2010.
[27] K. Chapman, "Constant Coefficient Multipliers for the XC4000E," Xilinx Technical Report 1996.
[28] M. J. Wirthlin and B. McMurtrey, "Efficient Constant Coefficient Multiplication Using Advanced FPGA Architectures," International Conference on Field Programmable Logic and Applications (FPL), 2001.
[29] M. J. Wirthlin, "Constant Coefficient Multiplication Using Look-Up Tables," Journal of VLSI Signal Processing, vol. 36, pp. 7-15, 2004.
[30] A. Mirshekari and M. Mosleh, "Hardware Implementation of a Fast FIR Filter with Residue Number System," 2nd International Conference on Industrial Mechatronics and Automation, 2010.
[31] J. E. Carletta and M. D. Rayman, "Practical Considerations in the Synthesis of High Performance Digital Filters for Implementation on FPGAs," FPL 2002, LNCS 2438, pp. 886-896, Springer, 2002.
[32] R. Uma and J. Ponnian, "Systolic FIR Filter Design with Various Parallel Prefix Adders in FPGA: Performance Analysis," International Symposium on Electronic System Design, Kolkatta, 19-22 Dec., 2012.
[33] P. K. Meher, "Low-Latency Hardware-Efficient Memory-Based Design for Large Order FIR Digital Filters," 6th International Conference on Information, Communication and Signal Processing, 10-13 Dec., 2007.
[34] P. K. Meher, S. Chanderasekaram and A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic," IEEE Transactions on Signal Processing, vol. 56. no. 7, July, 2008.
[35] R. Uma and J. Ponnian, "Systolic FIR Filter Design with Various Parallel Prefix Adders in FPGA: Performance Analysis," International Symposium on Electronic System Design, Kolkatta, 19-22 Dec., 2012.
[36] Y. C. Tsao and K. Choi, "Area Efficient Parallel FIR Digital Filter Structures for Symmetric Convolutions Based on Fast FIR Algorithm," IEEE Transactions Very Large Scale Integration (VLSI) Systems, vol. 20, no. 2, pp. 366–371, Feb. 2010.
[37] V. D. Pavlovic, N. Doncov and D. Ciric, "1D and 2D Economical FIR Filters Generated by Chebyshev Polynomials of the First kind," International Journal of Electronic, 2013.
[38] S. Haykin, "Adaptive Filter Theory," Prentice Hall, 1991.
[39] C. Cheng and K. K. Parhi, "Hardware Efficient Fast Parallel FIR Filter Structures Based on Iterated Short Convolution," IEEE Transactions Circuits Systems-I, Reg. Papers, vol. 51, no. 8, pp. 1492–1500, Aug. 2004.
[40] M. Nikolic and M. Lutovac, "Sharpening of the Multistage Modified Comb Filters," Serbian Journal of Electrical Engineering, vol. 8, no. 3, pp. 281- 291, 2011.
[41] Y. C. Tsao and K. Choi, "Area Efficient VLSI Implementation for Parallel Linear-Phase FIR Digital Filters of Odd Length Based on Fast FIR Algorithm," IEEE Transactions on Circuits and Systems-II Express Briefs, vol. 59, no. 6, June 2012.
[42] Y. C. Tsao and K. Choi, "Hardware-Efficient VLSI Implementation for 3-Parallel Linear-Phase FIR Digital Filters of Odd Length," IEEE International Symposium on Circuits and Systems, Seoul, 2012.
[43] A. Ling, D. P. Singh, and S. D. Brown, "FPGA Technology Mapping: A Study of Optimality," IEEE Proceedings Design Automation Conference, pp. 427-432, June 2005.
[44] J. H. Anderson and Q. Wang, "Area-Efficient FPGA Logic Elements: Architecture and Synthesis," 16th Asia and South Pacific Design Automation Conference (ASP-DAC), January 2011.
[45] L. Deng, K. Sobti, Y. Zhang and C. Chakarbarti, "Accurate Area, Time and Power models for FPGA based Implementations," Journal of Signal Processing Systems, Springer, 2011.
[46] http://www.xilinx.com.
[47] L. Aksoy, P. Flores and J. Monteiro, " A Tutorial on Multiplierless Design of FIR Filters: Algorithms and Architectures,"
[48] B. Khurshid and R. N. Mir, "Power Efficient Implementation of Bit-Parallel Unrolled CORDIC Structures for FPGA Platforms," International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), 2015.
[49] P. K. Meher, S. Chanderasekaran and A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization using Distributed Arithmetic," IEEE Transactions on Signal Processing, vol. 56, No. 7, July 2008.