

Low Power Programmable FIR Filtering IP Cores Targeting System-on-a-Reprogrammable-Chip (SoRC)

Muhammad Akhtar Khan, Abdul Hameed
National Engg & Scientific Commission
Islamabad, Pakistan
Post Code 44000

A. T. Erdogan
University of Edinburgh
School of Engineering and Electronics
Edinburgh, EH9 3JL, Scotland, UK

***Abstract** - This paper presents the design and implementation methodology of some low power programmable FIR filtering IP cores targeting SoRC and compares their performance in term of area, power and speed. The paper analyzes the dynamic power consumption of the IP cores in the fabric of Field Programmable Gate Arrays (FPGAs). The target device is Virtex family (xcv1000). The cores are technology independent and runtime programmable. The results show an overall 61% power saving at an expense of 36% area (Slices used). The results demonstrate that the power saving is achieved by reducing switching activity in the cores. This reduction in switching activity is achieved by applying low power design techniques in to the MAC (multiply and accumulate) unit which is a power hungry component in FIR filters. The results are based on the comparison of 73 taps programmable cores with two 16X128 RAMs for data and coefficients.*

Keywords: Virtual Component, Reusable Filter Cores, Digital Filter, SoRC, System on a Reprogrammable Chip, SoC.

1 Introduction

The availability of multimillion gate FPGAs operating at speeds surpassing 300MHz has dramatically changed the design methodologies. Many designs, which previously could only achieve speed and cost-of-density goals in ASICs, are converting to much more flexible and productive reprogrammable solutions. The availability of multimillion gate FPGAs has started a shift of SoC (System-on-Chip) designs towards using Reprogrammable FPGAs, thereby starting a new era of System-on-a-Reprogrammable-Chip (SoRC) [8]. With the availability of new FPGA architectures designed for system level integration and FPGA design tools that are compatible with ASIC design methodologies, it is now possible to employ a similar if not identical design reuse methodology for ASICs and FPGAs.

Traditionally, DSP algorithms are implemented using general purpose DSP chips for low rate

applications, or special purpose DSP chip-sets and ASICs for higher rates. Advancements in FPGAs provide new options for DSP engineers. The FPGA maintains the advantages of custom functionality like an ASIC while avoiding the high development costs and the inability to make design modification after production.

There is a high demand for power conscious design techniques in SoC/SoRC at all levels of design abstraction [1]. One way to efficiently incorporate low power design techniques in SoC design is to implement low power architectures in IP cores [5]. These IP cores are re-used across various designs, which have the benefit of reducing the design effort as well as shortening time-to-market. The FIR filter is a widely used DSP algorithm in industrial applications such as mobile telephony, audio and image processing. These devices require efficient and flexible cores for the implementation of complex algorithms which in turn involve recursive and repetitive implementation of various FIR filters on these cores with very constrained power and area demands.

The digital circuits nowadays are characterized not only by their functionality but also by their area consumption, power consumption and speed performance in SoC/SoRC. It can be shown that the most significant factor affecting power consumption in a CMOS VLSI device is the switching power, which is expressed as follows:

$$P_{sw} = \frac{1}{2} k C_{load} V_{dd}^2 f \quad (1)$$

Where V_{dd} is the supply voltage, f is the clock frequency, C_{load} is the load capacitance of the gate and k is the switching activity factor, which is defined as the average number of times that a gate makes a logic transition ($1 \rightarrow 0$ or $0 \rightarrow 1$) in each clock cycle. Therefore, one way of reducing the power consumption is to reduce the amount of switched capacitance during its operation.

In our previous work [5] we have implemented a number of programmable soft FIR filter IP cores using low power architectures targeting ASIC implementation. We compared the performance of programmable cores in terms of area and power and also evaluated the cost of programmability by comparing them with their corresponding fixed coefficient cores.

The aim of the work is to introduce new design methodologies and implementation techniques for programmable low power FIR IP Cores targeting the Xilinx Virtex family FPGA (xcv1000) and evaluation of their performance in terms of area, power and speed. The low power FIR techniques used in these cores are coefficient segmentation (seg) [4], block processing (BP) [3], and combination of coefficient segmentation and block processing (comb) [1]. In addition, the performance of three different types of multipliers (generic, booth and low power multiplier) is also evaluated. The cores demonstrated in this paper are fully parameterized and run time programmable. Parameterization and programmability in cores enhance the reusability of cores and user can change the personality of filter without any modification in hardware of filter. The programmable cores implemented in this paper achieve power saving not only by minimizing the switching activity at multiplier inputs but also by introducing low power multipliers.

The paper is organized as follows. A brief introduction of FIR filter cores with different types of multipliers is provided in section 2. Simulation results are discussed in section 3 and finally the paper concludes with section 4.

2 Implementation Work

A direct form N-tap FIR implementation is realized by the following equation.

$$Y(n) = \sum_{k=0}^{N-1} h_k \cdot x(n-k) \quad (2)$$

This means that the present and N-1 previous data samples of input $x(n)$ are multiplied by N-tap coefficients, and then are summed together to form the filter output $y(n)$. In our case we have implemented single data-path to realize the filter cores. It implies that with a single data-path unit, N cycles will be needed in order to generate a filter output. The block diagram of our reference programmable core which is based on generic Direct Form FIR architecture is illustrated in Figure 1. It

consists of two memory blocks, one for coefficient and one for input data stream, two registers for holding the coefficients and input data, an output register, a controller and a MAC unit.

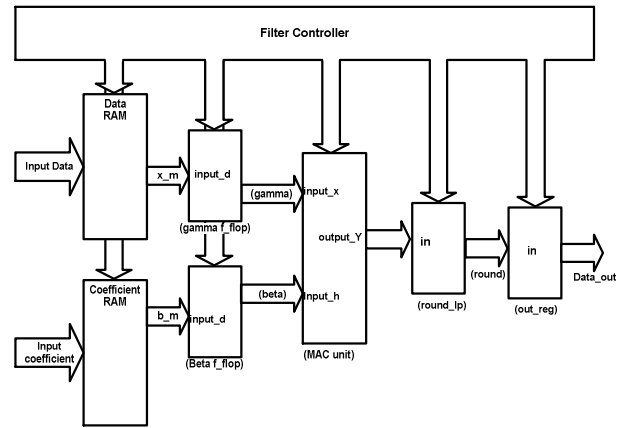


Figure 1: Block Diagram of Direct Form FIR filter

For introduction of programmability in core, the controller of the filter is designed such that it will have a control signal which is used to load the new coefficients in to the coefficient RAM, when required. For filter operation, first the coefficients of filter are loaded in to the coefficient RAM. After completion of loading, the load signal will become low and normal filtering operation will start till the next load high signal. During the loading process of filter coefficients, the filtering function is disabled. Both operations (loading of coefficient and normal filtering) are controlled by a controller. The block diagram of the controller is shown in Figure 2.

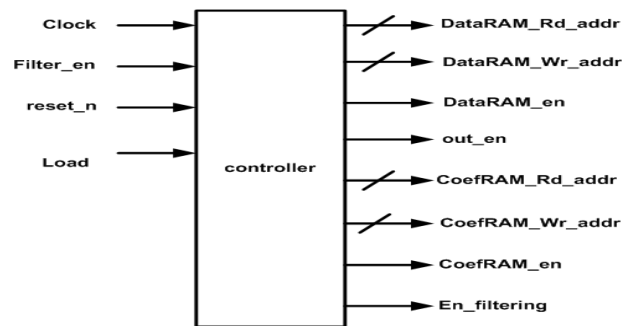


Figure 2: Block diagram of filter controller

A Direct Form filter core is considered as a reference core and power, area and speed results of this core are used for comparison with other cores designed with low power architectures (BP, seg, comb etc). All components mentioned in the block diagram are designed as a separate module in verilog HDL and are integrated in the top module.

Figures 3, 4 and 5 show the data flow diagrams of block processing (BP), segmentation (seg) and combine (comb) architectures. These architectures are designed such that they reduce switching activity at multiplier input and hence achieve a significant power saving. The implementation of these architectures also has some area overhead shown in tables 5, 6 and 7 respectively. The hardware implementation details of these architectures can be found in previous publications [1], [5].

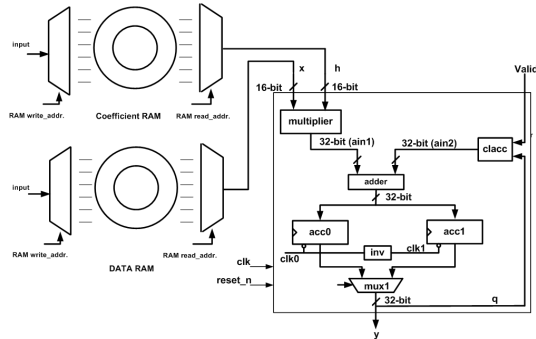


Figure 3: Data Flow Diagram of BP Technique

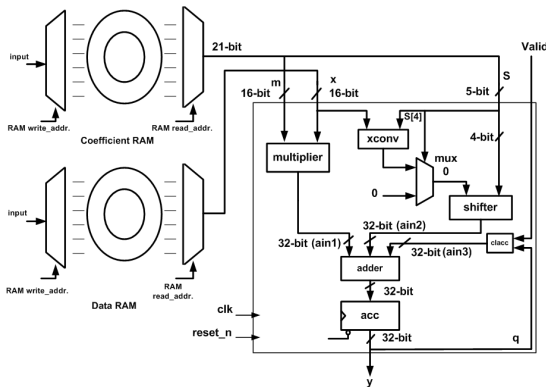


Figure 4: Data Flow Diagram of Seg. Technique

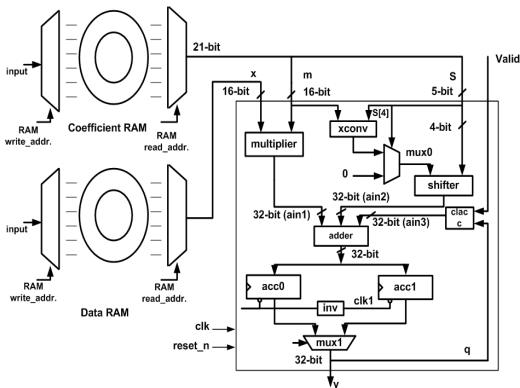


Figure 5: Data Flow Diagram of Comb Technique

In order to study the effects of different multiplier architectures on power, area and speed, the cores are

implemented with three different types of multipliers (generic, booth and low power). The architecture of the low power multiplier [10] exploits the fact that the power saving in FPGAs is achieved when there is symmetry in design. So in our low power multiplier, larger multiplier is decomposed in to a number of small ones. For example a $4n \times 4n$ bit multiplier is constructed by four $2n \times 2n$ bit multipliers which in turn each requires four $n \times n$ multipliers. This can be shown by the following equation.

$$A.X = (A_H \cdot 2^n + A_L) \cdot (X_H \cdot 2^n + X_L) \quad (3)$$

$$= A_H \cdot X_H \cdot 2^{2n} + (A_H \cdot X_L + A_L \cdot X_H) \cdot 2^n + A_L \cdot X_L \quad (4)$$

A total of 16 partial product generated in this way have to be aligned before being added, as shown in Figure 6.

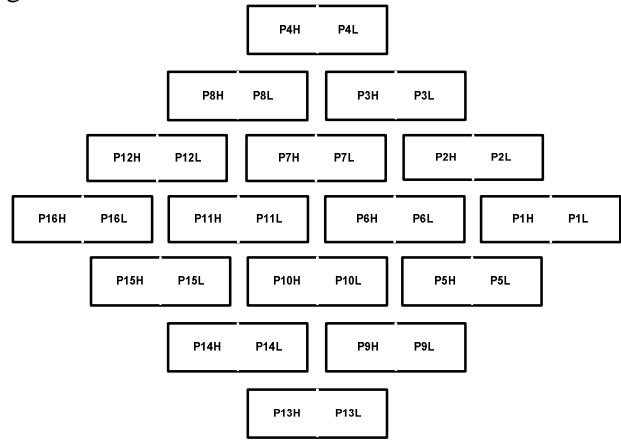


Figure 6: 16 partial products alignment

Figure 7 shows the block diagram of booth multiplier's partial product generation mechanism. The output of these partial products is then added using carry save adders (csa) and finally a carry-look-ahead (cla) adder.

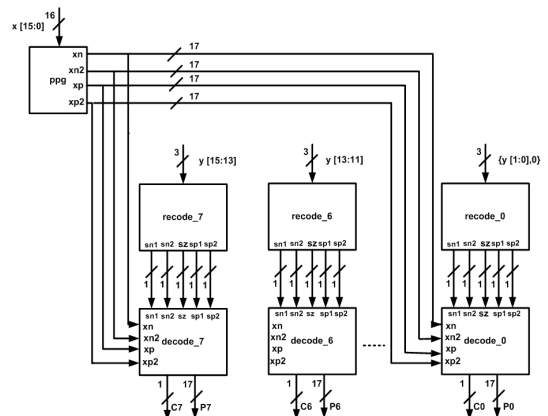


Figure 7: Block diagram of Booth multiplier

In the figure 7, the ppg unit takes a 16-bit number X as a multiplicand, process the input number and produce four (X_{n1} , X_{n2} , X_{p1} , X_{p2}) outputs which will be used later in decode units as follows. These four outputs are generated based upon radix-4.

- X_{n1} : $-X$ with sign extension, 17-bit number.
- X_{n2} : $-2X$, 17-bit number.
- X_{p1} : X with sign extension, 17-bit number
- X_{p2} : $2X$, 17-bit number.

The recode unit takes 3-bit chunks (radix-4) of the multiplier number Y as shown in Figure 8. Each 3-bit chunk can have 8 possible permutations. Based upon these 8 possible options, 5 outputs are produced. The decode unit processes the 9 inputs (4 from ppg and 5 from recode unit) and produces 2 outputs (partial product and carry) shown in Table 1. Each recode unit has one corresponding decode unit to process the inputs.

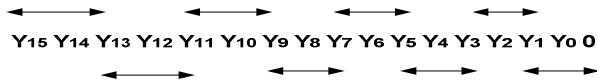


Figure 8: 3-bit chunks for Recode unit

Table 1: Recode-Decode relation in Booth

Recode Unit Output	Decode Unit Output
$S_{n2} = 1$	$P = X_{n2}, Co = 1$
$S_{n1} = 1$	$P = X_{n1}, Co = 1$
$S_z = 1$	$P = 0, Co = 0$
$S_{p1} = 1$	$P = X_{p1}, Co = 0$
$S_{p2} = 1$	$P = X_{p2}, Co = 0$

3 SIMULATION RESULTS

We have analyzed the power consumption, area and speed of the different programmable low power FIR filtering IP cores implemented with three different types of multipliers. The low power techniques used in these IP cores were, direct form (ufdf), block processing (BP), segmentation (seg) and combination of block and segmentation (comb). The multipliers used were: generic multiplier (G_mult), low power multiplier (technique of implementing large multiplier using small one i.e. 16x16 with 4x4) and booth multiplier. All IP cores were implemented for an example of 73-tap (16-bit data and coefficient with) low pass filter. The cores were designed using verilog HDL and then implemented using Xilinx ISE 7.1i tool. Simulations were performed using Model

Sim. For power evaluation the Xilinx XPower tool were used. In all cases the clock rate of 10 MHz was used.

3.1 Power Results

Tables 2, 3 and 4 shows the power result of programmable low power FIR filter cores with three different types of multipliers.

Table 2: Power analysis with generic multiplier

Core Name	Total Power (mW)	Toggle Rate (%)	Saving (%)
Ufdf_g_mult	21	70	-
BP_g_mult	20	70	5
Seg_g_mult	21	71	0
Comb_g_mult	8	70	61

Table 2 shows the power analysis of different low power programmable cores with a generic multiplier. It is evident from the results that the power consumption of the combined core (combination of both block processing and segmentation techniques) has the lowest value (61% power saving). This saving is due to the maximum reduction in switching activity at multiplier input, due to the combination of both, block processing architecture and coefficient segmentation architecture. Because of this low switching activity in this architecture, the total power is 8mW and a power saving of 61% is achieved.

Table 3: Power analysis with low power multiplier

Core Name	Total Power (mW)	Toggle Rate (%)	Saving (%)
Ufdf_lp_mult	19	66	-
BP_lp_mult	19	71	0
Seg_lp_mult	9	68	52
Comb_lp_mult	8	70	58

Table 3 shows the power results of different programmable cores with low power multiplier (lp_mult). Comparing the results of tables 1 and 2 it is clear that effect of low power multiplier on power saving is more dominating in ufdf, BP and seg cores but have no effect in comb core.

Table 4: Power analysis with booth multiplier

Core Name	Total Power (mW)	Toggle Rate (%)	Saving (%)
Ufdf_booth	26	69	-
BP_booth	22	70	15
Seg_booth	9	69	65
comb_booth	9	73	65

Table 4 shows the power results of cores with booth multiplier. It is clear from the Figure 7 that the booth multiplier does not have regular structure. Keeping in view the layout and resources of FPGA, it is clear that the dynamic power consumption of components having block style structure is always low as compared to distributed style structures when these structures sit on an FPGA fabric. This is because the majority (at least 60%) [4] of power is dissipated in the collection of interconnect resources and logic cell interface circuitry that a design utilizes. The structure of booth multiplier is distributed type that is why it consumes more power as compared to other type of multipliers which has a block style structure. The high resource utilization of distributed style structure of booth multiplier on an FPGA is very clear from the area result in Table 5, 6 and 7, where the area of cores with booth multiplier is more than the area of cores having other multipliers. The paper [4] shows that the dynamic power dissipation shares of interconnect; logic and clocking resources in a typical FPGA are 60%, 16% and 14% respectively. Interface circuitry includes the input multiplexers used to select CLB inputs from the wiring tracks and the output buffers used to drive signals on to the interconnect fabric [6]. For distributed FPGA implementation, to realize a logic function, it requires more interconnect resources to be used for linking those isolated CLBs. Thus, as soon as a signal leaves CLBs, it immediately drowns in the capacitance discontinuity caused by the general purpose routing network and hence produces large switching power consumption. Therefore, using block style components, the interconnect power can be minimized.

3.2 Area Results

The Tables 5, 6 and 7 show the Area results of programmable low power FIR filtering processor cores with three different type of multipliers. The area results show that the combined architecture which is the most complex architecture, consumes the highest area as compared to other architectures. In addition to this, it is also clear from the data presented in tables 5, 6 and 7 that the combined architecture with booth multiplier consumes the highest area as compared to the combined architectures with other multipliers.

Table 5: Area analysis with generic multiplier

Core Name	No of Slices	Input LUTs	No of BRAMS	Bonded IOBs
Ufdf_g_mult	318 (2%)	614 (2%)	2 (6%)	59 (14%)
BP_g_mult	409 (3%)	667 (2%)	2 (6%)	59 (14%)
Seg_g_mult	350 (2%)	680 (2%)	3 (9%)	63 (15%)
Comb_g_mult	435 (3%)	706 (2%)	3 (9%)	64 (15%)

Table 6: Area analysis with low power multiplier

Core Name	No of Slices	Input LUTs	No of BRAMS	Bonded IOBs
Ufdf_lp_mult	324 (2%)	626 (2%)	2 (6%)	59 (14%)
BP_lp_mult	421 (3%)	678 (2%)	2 (6%)	59 (14%)
Seg_lp_mult	390 (3%)	750 (3%)	3 (9%)	63 (15%)
Comb_lp_mult	461 (3%)	759 (3%)	3 (9%)	64 (15%)

Table 7: Area analysis with booth multiplier

Core Name	No of Slices	Input LUTs	No of BRAMS	Bonded IOBs
Ufdf_booth	518 (4%)	975 (3%)	2 (6%)	59 (14%)
BP_booth	616 (5%)	991 (4%)	2 (6%)	59 (14%)
Seg_booth	566 (4%)	1080 (4%)	3 (9%)	63 (15%)
Comb_booth	655 (5%)	1065 (4%)	3 (9%)	64 (15%)

3.3 Speed Results

The Tables 8, 9 and 10 show the speed results of programmable low power FIR filtering IP cores with three different type of multipliers. From speed results of different FIR cores, it is clear that the generic multiplier has over all best result when implemented with different low power cores. This is because of its simplicity in implementation.

Table 8: Speed analysis with generic multiplier

Core Name	Min period (ns)	Max Frequency (MHz)	Setup time (ns)	Hold time (ns)
ufdf_g_mult	37.387	26.747	9.013	28.271
Seg_g_mult	35.638	28.060	11.283	13.400
BP_g_mult	64.322	15.547	9.883	13.214
com_g_mult	55.908	17.887	8.464	11.646

Table 9: Speed analysis with low power multiplier

Core Name	Min period (ns)	Max Frequency (MHz)	setup time (ns)	Hold time (ns)
ufdf_lp_mult	46.023	21.728	9.013	22.449
Seg_lp_mult	45.197	22.125	10.702	14.178
BP_lp_mult	82.246	12.159	9.883	12.121
Com_lp_mult	82.338	12.145	10.921	11.716

Table 10: Speed analysis with booth multiplier

Core Name	Min period (ns)	Max Frequency (MHz)	setup time (ns)	Hold time (ns)
Ufdf_booth	40.682	24.581	9.963	21.240
Seg_booth	36.091	27.708	9.501	12.994
BP_booth	56.744	17.623	9.347	12.513
Com_booth	56.236	17.782	11.479	12.229

4 CONCLUSION

In this paper complete design and implementation methodology of a number of low power programmable FIR filtering IP cores targeting SoRC is presented. The performance of these cores is compared in term of area, power and speed. An overall 61% power saving is achieved with the implementation of combined architecture (seg+block), using generic and low power multiplier, at an expense of 36% area overhead in number of Slices used in FPGA. Target device is Virtex (xcv1000) FPGA. In terms of size, Virtex (xcv1000) FPGA is suitable for SoRC. The results are based on comparison of 73 taps programmable FIR filtering processor cores with 16x128 RAMs for data and coefficients. The cores are synthesized using Xilinx ISE 7.1i and power is evaluated using XPower tool. The work will be extended to implement low power cores using FPGA's on-chip resources (block multiplier and block RAM etc) targeting SoRC design, will be explored further.

5 REFERENCES

- [1] A. T. Erdogan, M. Hasan and T. Arslan "Algorithmic low power FIR cores"; IEE Proceedings–Circuits, Devices and Systems, Vol.150, No. 3, June 2003, pp. 155-160
- [2] A. T. Erdogan and T. Arslan, "Low power implementation of linear phase FIR filters for single multiplier CMOS based DSPs"; IEEE International Symposium on Circuits and Systems (ISCAS'98), 1998.
- [3] T. Arslan and A. T. Erdogan, "Data block processing for low power implementation of direct form FIR filters on

single multiplier CMOS DSPs"; IEEE International Symposium on Circuit and Systems (ISCAS'98), 1998.

- [4] T. Arslan and A. T. Erdogan, "A Coefficient segmentation algorithm for low power implementation of FIR filters"; IEEE International Symposium on Circuit and Systems (ISCAS'99), 1999.

- [5] Muhammad Akhtar Khan and A.T. Erdogan, "Parameterized and Programmable Low Power Soft FIR Filtering IP Cores", WSEAS Transactions on Circuits and Systems, Issue 10, Volume 3, pp. 2156-2161, December 2004, ISSN 1109-2734.

- [6] Li Shang, Alireza S Kaviani and Kusuma Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family". http://www.ece.queensu.ca/hpages/faculty/shang/papers/fpga_a02.pdf

- [7] Eric Kuss and Jan M Rabaey, "Low-Energy Embedded FPGA Structure", EECS Department, University of California at Berkeley, CA, USA.

- [8] Xilinx, "Xilinx Design Reuse Methodology for ASIC and FPGA Designers". www.xilinx.com/ipcenter/designreuse/docs/Xilinx_Design_Reuse_Methodology.pdf

- [9] Gregory Roy Goslin, "A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance", Xilinx, Inc. 2100 Logic Dr. San Jose, CA 95124.

- [10] Israel Koren, "Computer Arithmetic Algorithms", 2nd Edition, A K Peters, Ltd. 63 South Avenue Natick, MA 01760.