Design and Implementation of Block Based Transpose Form FIR Filter

O. Venkata Krishna¹, Dr. C. Venkata Narasimhulu², Dr. K. Satya Prasad³

¹(ECE, CVR College of Engineering, Hyderabad, India)

² (ECE, Geethanjali College of Engineering and Technology, Hyderabad, India)

³ (ECE, KL Deemed to be University, Vaddeshwaram, Guntur, India)

ABSTRACT

The transpose form configuration of Finite impulse response filter (FIR) does not support for block based processing so form FIR filter architecture is optimized and implemented. The basic Data Flow Graph (DFG) of transpose form FIR filter is converted into block based DFG and retiming is inserted in the DFG for low power consumption, reduced area and minimal delay. The generalized mathematical formulation is done for the retimed block based transpose form FIR filter and it is implemented with the block size of 4 for the filter length of 16 using Verilog Hardware Description Language (HDL). Later, it is synthesized using CADENCE-RTL compiler in TSMC 45nm CMOS library and power, area and delay reports are generated. The obtained results are compared with the few existing structures.

Keywords: Digital filters, Data Flow Graph, Retiming, low power, FIR, and HDL.

1. INTRODUCTION

The Digital Signal Processing (DSP) systems are being implemented on Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC), due to the reconfiguration and flexibility of FPGAs. The FPGA platform is more suitable for the optimizing the DSP systems in terms of area, power and delay. Digital filters are mostly used in DSP applications [1], such as biomedical applications, communication systems and mobile applications. For these applications, the digital filter must consume less power, reduced area and high speed. The FIR filters can be implemented in different architectures, such as, direct form structure, transpose form structure and hybrid structures.

Several FIR architectures are implemented in different styles to meet the specifications. For example, a FIR filter implemented by Mahesh et al [2] using programmable shift method (PSM) and Constant shift method (CSM) [8][11]. Park [3] also implemented a FIR filter based on distributed arithmetic structure in direct form and transpose form structures. But there is no any block based concept in transpose form structure. Mohanty et al [4] proposed block based structures and filter banks, which are not suitable for higher order filter lengths and applicable for 2-Dimensional (2D) filters. Mohanty also proposed [5], the reconfigurable block based transpose form filter and fixed length transpose form FIR filter for DSP applications [6].

The most preferred architectures of FIR filters in signal processing are transpose form structures. The transpose form FIR filter consists of inherent pipelining process. The pipelining in the digital filters design leads to reduction of critical path or delay, reduction of power consumption and increases the clock speed.

In this paper, the block based transpose form FIR filter is realized and mathematically formulated for reconfiguration applications. The DFG of transpose form FIR filter is converted and modified to reduce the power consumption, area and delay. Section-II, the computational analysis using DFG and data flow table (DFT) of transpose form FIR filter and mathematical formulations are presented. Section-III describes the realization of hard ware structure and the implementation approach of the proposed FIR filter. In section-IV, all the practical implemented synthesized circuits and simulation diagrams are presented and corresponding results compared with the existing structures in terms of Very Large Scale Integration (VLSI) design metrics, such as area power and delay etc.

2. ANALYSIS OF BLOCK BASED TRANSPOSE FORM FIR FILTER

The digital filters are designed to modify the frequency response properties of the given discrete input signal $x(n)$ to meet the some design requirements. The digital filter characterized by its transfer function or frequency response or unit sample response $h(n)$.

Equation (1) represents the general differential equation of digital filter.

$$y(n) = -\sum_{k=1}^{N-1} a_k y(n-k) + \sum_{k=0}^{N-1} b_k x(n-k) \quad (1)$$

If $a_k = 0$ for $1 \leq k \leq N-1$, the above equation becomes

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \quad (2)$$

Equation (2) is N tap finite impulse response filter with unit sample response $h(k) = b_k$ for $1 \leq k \leq N-1$ and $h(k) = 0$ otherwise.

The above equation can be expressed in the z- transform is given by,

$$Y(z) = \sum_{k=0}^{N-1} b_k z^{-k} X(z)$$

For the computational analysis, the DFGs are drawn in the transpose form for the filter length $N=8$ as shown in figures. The figure 1 represents the DFG for the input $x(n)$ and output $y(n)$ and figure 2 describes the DFG for the input $x(n-1)$ and output $y(n-1)$ respectively.

Figure1: DFG of transpose form FIR filter for the length of 8 for output $y(n)$ to input $x(n)$.

Figure 2: DFG of transpose form FIR filter for the length of 8 for output $y(n-1)$ to input $x(n-1)$.

In the DFG1 and DFG2, the multiplied values of coefficients with input values and corresponding accumulation paths are shown in the data flow tables (DFT1) and DFT2 of figure 3. The accumulation path of the product values are indicated by arrows in DFT1 and DFT2. From the observation of the DFT1 and DFT2, we conclude that, the five values in the each column of the data flow graphs are same.

CCS	b(7)	b(6)	b(5)	b(4)	b(3)	b(2)	b(1)	b(0)
x(n-5)	x(n-5)b(7)	x(n-5)b(6)	x(n-5)b(5)	x(n-5)b(4)	x(n-5)b(3)	x(n-5)b(2)	x(n-5)b(1)	x(n-5)b(0)
x(n-4)	x(n-4)b(7)	x(n-4)b(6)	x(n-4)b(5)	x(n-4)b(4)	x(n-4)b(3)	x(n-4)b(2)	x(n-4)b(1)	x(n-4)b(0)
x(n-3)	x(n-3)b(7)	x(n-3)b(6)	x(n-3)b(5)	x(n-3)b(4)	x(n-3)b(3)	x(n-3)b(2)	x(n-3)b(1)	x(n-3)b(0)
x(n-2)	x(n-2)b(7)	x(n-2)b(6)	x(n-2)b(5)	x(n-2)b(4)	x(n-2)b(3)	x(n-2)b(2)	x(n-2)b(1)	x(n-2)b(0)
x(n-1)	x(n-1)b(7)	x(n-1)b(6)	x(n-1)b(5)	x(n-1)b(4)	x(n-1)b(3)	x(n-1)b(2)	x(n-1)b(1)	x(n-1)b(0)
x(n)	x(n)b(7)	x(n)b(6)	x(n)b(5)	x(n)b(4)	x(n)b(3)	x(n)b(2)	x(n)b(1)	x(n)b(0)

CCS	b(7)	b(6)	b(5)	b(4)	b(3)	b(2)	b(1)	b(0)
x(n-6)	x(n-6)b(7)	x(n-6)b(6)	x(n-6)b(5)	x(n-6)b(4)	x(n-6)b(3)	x(n-6)b(2)	x(n-6)b(1)	x(n-6)b(0)
x(n-5)	x(n-5)b(7)	x(n-5)b(6)	x(n-5)b(5)	x(n-5)b(4)	x(n-5)b(3)	x(n-5)b(2)	x(n-5)b(1)	x(n-5)b(0)
x(n-4)	x(n-4)b(7)	x(n-4)b(6)	x(n-4)b(5)	x(n-4)b(4)	x(n-4)b(3)	x(n-4)b(2)	x(n-4)b(1)	x(n-4)b(0)
x(n-3)	x(n-3)b(7)	x(n-3)b(6)	x(n-3)b(5)	x(n-3)b(4)	x(n-3)b(3)	x(n-3)b(2)	x(n-3)b(1)	x(n-3)b(0)
x(n-2)	x(n-2)b(7)	x(n-2)b(6)	x(n-2)b(5)	x(n-2)b(4)	x(n-2)b(3)	x(n-2)b(2)	x(n-2)b(1)	x(n-2)b(0)
x(n-1)	x(n-1)b(7)	x(n-1)b(6)	x(n-1)b(5)	x(n-1)b(4)	x(n-1)b(3)	x(n-1)b(2)	x(n-1)b(1)	x(n-1)b(0)

Figure 3: DFT for output $y(n)$ to input $x(n)$ with respect to DFG1 and DFT2 output $y(n-1)$ to input $x(n-1)$ with respect to DFG2.

This is high redundancy in the normal transpose form FIR filter. This redundancy can be reduced in the above FIR structures for the two consecutive inputs by introducing the block based inputs concept. Here, the non-overlapped sequence input blocks are used. Now two modified data flow tables DFT3 and DFT4 are presented in the figure 4 corresponding to non-overlapped input blocks to avoid the redundancy in normal transpose form FIR filters.

CCS	b(7)	b(6)	b(5)	b(4)	b(3)	b(2)	b(1)	b(0)
x(n-5)	x(n-10)b(7)	x(n-10)b(6)	x(n-10)b(5)	x(n-10)b(4)	x(n-10)b(3)	x(n-10)b(2)	x(n-10)b(1)	x(n-10)b(0)
x(n-4)	x(n-9)b(7)	x(n-9)b(6)	x(n-9)b(5)	x(n-9)b(4)	x(n-9)b(3)	x(n-9)b(2)	x(n-9)b(1)	x(n-9)b(0)
x(n-6)	x(n-6)b(7)	x(n-6)b(6)	x(n-6)b(5)	x(n-6)b(4)	x(n-6)b(3)	x(n-6)b(2)	x(n-6)b(1)	x(n-6)b(0)
x(n-4)	x(n-4)b(7)	x(n-4)b(6)	x(n-4)b(5)	x(n-4)b(4)	x(n-4)b(3)	x(n-4)b(2)	x(n-4)b(1)	x(n-4)b(0)
x(n-2)	x(n-2)b(7)	x(n-2)b(6)	x(n-2)b(5)	x(n-2)b(4)	x(n-2)b(3)	x(n-2)b(2)	x(n-2)b(1)	x(n-2)b(0)
x(n)	x(n)b(7)	x(n)b(6)	x(n)b(5)	x(n)b(4)	x(n)b(3)	x(n)b(2)	x(n)b(1)	x(n)b(0)

CCS	b(7)	b(6)	b(5)	b(4)	b(3)	b(2)	b(1)	b(0)
x(n-11)	x(n-11)b(7)	x(n-11)b(6)	x(n-11)b(5)	x(n-11)b(4)	x(n-11)b(3)	x(n-11)b(2)	x(n-11)b(1)	x(n-11)b(0)
x(n-9)	x(n-9)b(7)	x(n-9)b(6)	x(n-9)b(5)	x(n-9)b(4)	x(n-9)b(3)	x(n-9)b(2)	x(n-9)b(1)	x(n-9)b(0)
x(n-7)	x(n-7)b(7)	x(n-7)b(6)	x(n-7)b(5)	x(n-7)b(4)	x(n-7)b(3)	x(n-7)b(2)	x(n-7)b(1)	x(n-7)b(0)
x(n-5)	x(n-5)b(7)	x(n-5)b(6)	x(n-5)b(5)	x(n-5)b(4)	x(n-5)b(3)	x(n-5)b(2)	x(n-5)b(1)	x(n-5)b(0)
x(n-3)	x(n-3)b(7)	x(n-3)b(6)	x(n-3)b(5)	x(n-3)b(4)	x(n-3)b(3)	x(n-3)b(2)	x(n-3)b(1)	x(n-3)b(0)
x(n-1)	x(n-1)b(7)	x(n-1)b(6)	x(n-1)b(5)	x(n-1)b(4)	x(n-1)b(3)	x(n-1)b(2)	x(n-1)b(1)	x(n-1)b(0)

Figure 4: The modified Data flow tables DFT3 and DFT4 for transpose form FIR filter with block size of 2 and N=6.

The DFT3 is the data flow table for the output of $y(n)$ and DFT4 for the output of $y(n-1)$. There is no redundancy in DFT3 and DFT4, which can be observed from the entries of the tables. The gray cells represent the output of $y(n)$ and other values for output $y(n-1)$. Now the DFG1 is completely transformed into a new DFG corresponding to the DFT3 and DFT4 is referred as DFG3. The DFG3 is the equivalent flow diagram for the computations of DFT3 and DFT4 with non-overlapped blocks of 2 for the filter length of 8 which is shown in figure 5.

Figure 5: Modified DFG of block based transpose form FIR filter for the length of 8.

This DFG3 is further optimized using the concept of retiming. The retiming is method which reduces the power, area and delay in VLSI circuits, by changing the positions of the delay elements like flip flops. This change can not alter the characteristics of the circuit. Retiming is mostly used in the synchronous designs for many applications. Due to the retiming, circuit switching activity is reduced, hence the power consumption decreases. Actually, the dynamic power dissipation is reduced in static CMOS circuits [7]. In this paper, the DFG3 is retimed to obtain the advantages of retiming and named as DFG4, which is block based retimed transpose form FIR filter as shown in figure 6.

Figure 6: Retimed DFG of block based Transpose form FIR filter for the length of 8.

In the comparison of DFG3 and DFG4 for block based transpose form FIR filters, note that both structures consists of equal number of adders and multipliers. Only the delay elements or D flip flops are less in the retimed FIR filter structure.

3. MATHEMATICAL FORMULATION OF PROPOSED STRUCTURE

This retimed block based transpose form FIR filter is mathematically formulated to design and implement the proposed architecture for low power and less delay FIR filter with optimized area. For the implementation of proposed FIR filter, the input block size is assumed as $L=4$ for every cycle. It process the input samples and produce L number of outputs [8].

The filter output for k^{th} block is generally represented by y_k using the following relation.

$$Y_k = X_k \cdot b \quad (3)$$

Where, b is coefficient weight vector taken as $b = [b(0), b(1), b(2), \dots, b(N-1)]^T$

The input matrix X_k is taken as

$$X_k = [X_k^0 \ X_k^1 \ X_k^2 \ \dots \ X_k^{N-1}] \quad (4)$$

Where X_k^i is the $(i+1)$ th column of X_k are defined as

$$X_k^i = [x(4k-i) \ x(4k-i-1) \ x(4k-i-2) \ x(4k-i-3)]^T \quad (5)$$

Substitute (4) in (3)

$$y_k = \sum_{i=0}^{N-1} X_k^i \cdot b(i) \quad (6)$$

Suppose N is composite number and decomposed as $N=ML$, the index $i = l + 4m$ for $0 \leq l \leq 3$ and $0 \leq m \leq 3$; Substituting $i = l + 4m$ in (5), we have

$$X_k^i = X_k^{l+4m} = X_{k-m}^l \quad (7)$$

Substitute (7) in (4)

$$\mathbf{X}_k = [X_K^0 \ X_K^1 \ X_K^2 \ X_K^3 \ X_{K-1}^0 \ X_{K-1}^1 \ X_{K-1}^2 \ X_{K-1}^3 \ X_{K-2}^0 \ X_{K-2}^1 \ X_{K-2}^2 \ X_{K-2}^3 \ X_{K-3}^0 \ X_{K-3}^1 \ X_{K-3}^2 \ X_{K-3}^3] \quad (8)$$

Substituting (8) in (3)

$$\mathbf{y}_k = \sum_{l=0}^3 \sum_{m=0}^3 X_{k-m}^l \cdot b(i+4m) \quad (9)$$

The input matrix $\mathbf{X}_k$ of (8) has the following features, the data block X_K^0 is the current block, while $\{X_{K-1}^0, X_{K-2}^0, X_{K-3}^0\}$ are blocks delayed by 1, 2, 3 cycles. The overlapped blocks $\{X_{K-1}^1, X_{K-2}^1, X_{K-3}^1\}$ are 1, 2, 3 clock cycles delayed version of overlapped block X_K^1 .

The input matrix $\mathbf{X}_k$ is decomposed into 3 small matrices R_k^l , such that R_k^0 contains 4 blocks $\{X_K^0, X_K^1, X_K^2, X_K^3\}$ and R_k^1 contains $\{X_{K-1}^0, X_{K-1}^1, X_{K-1}^2, X_{K-1}^3\}$. The coefficient vector $\mathbf{b}$ is decomposed into small vectors $C_m = \{b(4m), b(4m+1), b(4m+2), b(4m+3)\}$ where $0 \leq m \leq 3$.

Here R_k^m is symmetric and satisfy the identity

$$R_k^m = R_{k-m}^0 \quad (10)$$

From (10) R_k^m is m clock cycle delay with respect to R_k^0 , the equation (9) can be expressed in R_{k-m}^0 and C_m as

$$\left. \begin{aligned} \mathbf{y}_k &= \sum_{m=0}^3 \mathbf{r}_k^m \\ \mathbf{r}_k^m &= R_{k-m}^0 \cdot C_m \end{aligned} \right\} \quad (11)$$

The above relation can be expressed in z- transform as recurrence relation

$$Y(z) = R_k^0(z)[z^{-1}(z^{-1}(z^{-1}C_3 + C_2) + C_1) + C_0] \quad (12)$$

Where $R_k^0(z)$ and $Y(z)$ are the z- domain values of R_k^0 and $\mathbf{y}_k$ respectively. The above z- transform recurrence relation is equivalent to the DFG4 which was shown in the above figure.

The next section deals with the implementation of proposed architecture and internal blocks of the transposed form FIR filter structure for the length N=16 and block size L= 4 and related design issues.

4. IMPLEMENTATION OF PROPOSED ARCHITECTURE

In this section, the block based transpose form FIR filter with the constraint of retiming for the low power, less delay and less area is implemented. The figure 7 shows the architecture of proposed FIR filter, which consists of a Register Cell (RC), one Coefficient Unit (CU), four Product Units (PUs) and one Pipeline Adder Circuit (PAC).

Figure 7: Architecture of FIR filter for the length $N=16$ and block size $L=4$.

The block of 4 input samples are applied to the RC for the k^{th} cycle, it internally consists of delay flip flops to rearrange the samples corresponding to the algorithm and it produce 4 rows of input samples R_k^0 in parallel as shown in figure 8.

Figure 8: The internal circuitry of Register cell (RC)

These 4 rows of R_k^0 are applied to M (where $M=4$) number of PUs in the structure. The M weight coefficient vectors from CU also transmitted to PU. The 4 coefficient vectors C_0 are transmitted to PU4, C_1 to PU3, C_2 to PU2, and C_3 for PU1 respectively as shown in above figure. Then a matrix multiplication is taken place between the input samples R_k^0 from RC and coefficient weighted vectors C_m (where m is 0 to 3). The each PU internally consists of four inner product cells (IPC) as shown in the figure 9. The 4 rows of input samples from RC are going to row wise to the each inner product cell. Then IPC multiplies these values with 4 coefficient vectors and generates the r_k^m as the output. Similarly, each PU produce r_k^m , such as, r_k^0 , r_k^1 , r_k^2 , and r_k^3 . Here 4 PUs worked on parallel processing and produce 4 blocks of result r_k^m (where m is 0 to 3).

Parallel processing means that, the multiple outputs are computed in parallel for a clock period. The parallel processing and pipelining in the architecture of PAC is used to reduce the power consumption, for the reduction of critical path or delay, and which also can be improves the clock speed. Parallel processing and pipelining techniques are dual each other [9] [10]. A computation can be pipelined and it also can be parallel processed.

Figure 9: Product Unit block with four internal IPCs

The internal circuit of each IPC block is shown in the figure 10. The actual multiplication of inputs with filter coefficients is carried out by this cell. It multiplies the each input sample with the corresponding coefficient and produce the output as denoted by $r(4k)$. Similarly, the four IPCs produce the outputs are $r(4k-1)$, $r(4k-2)$ and $r(4k-3)$ respectively. The combination of these four outputs is generally denoted by r_k^m , is one of the output elements of PU. According to this procedure, the each PU produce 4 outputs referred as r_k^0 , r_k^1 , r_k^2 , and r_k^3 . These four PU outputs are given to the next block which is PAC.

Figure 10: Internal circuit of Inner product cell (IPC) in the PU.

These four outputs are passing through PAC block, which consists of delay elements and carry save adders, as shown in the figure 11. In this block, the total partial product outputs are added by a pipeline addition and produce L number of outputs y_k , where L value is 4. Finally, the output of the Transpose form FIR filter architecture produces 4 blocks of output for four

input block samples. The pipelining is used in this PAC block to obtain the optimization in the filter.

Figure 11: The internal circuit of the PAC.

5. RESULTS AND PERFORMANCE COMPARISON

The entire FIR filter design is implemented in Xilinx Synthesis Tool (XST) using Verilog HDL for the target device of FPGA vertex-5 and simulated using ISE simulator. The top level synthesized module of FIR filter and simulation outputs from the ISE Simulator are presented in the figure 12 and figure 13 respectively.

Figure 12: Top level module of block based transpose form FIR filter for the length $N=16$ and block size $L=4$ using XST.

Figure 13: Simulation outputs of proposed FIR filter using ISE Simulator.

The Table. I represents the complete design summary of the proposed FIR filter using XST tool for the FPGA Vertex 5. Here, the design blocks are mapped to the technology blocks in the FPGA. The device utilization percentages and number of available blocks and number of used blocks are shown in the table I.

Table I Device utilization summary for the proposed FIR filter using XST

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	129	64000	0%
Number of Slice LUTs	192	64000	0%
Number of fully used LUT-FF pairs	128	193	66%
Number of bonded IOBs	98	640	15%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48Es	80	256	31%

The blocks of FIR filter are coded using Verilog HDL, next synthesized using Encounter RTL Compiler in TSMC 45nm and TSMC 180nm CMOS technology from CADENCE. The RTL compiler gives nanometer performance goals, reduces the chip area, lowers power and improves timing closure. The reports are generated for power consumption, area and delay using this synthesis. The top module of FIR filter architecture from RTL compiler tool is shown in the figure 14.

Figure 14: The complete sturcture of proposed FIR filter using TSMC 45-nm CMOS technology by RTL Compiler.

The comparision of the two different TSMC technologies using RTL compiler is described in table II. The FIR filter synthesized results for TSMC 45-nm CMOS library and TSMC 180-nm CMOS technolgy are tabulated from th reports generated by RTL Compiler tool. The power consumption of the FIR filter in 180nm technology is very much greater than the 45nm technolgy, that means more power optimization is taken place in the 45nm technolgy due to the constraints given with repect to power reduction in the filter. Here, the number of delay elements also reduced using retiming, hence the delay is improved in the 45nm technology. The clock speed corresponding to FIR filter delay in the 45nm is high i.e 222 MHz.

The area also reduced in the advanced CMOS technology 45nm using appropriate constarints for the area. The number of Flip Flops are reduced and optimized adders and multipliers are used in the design of FIR filter. From the RTL compiler synthesis tool, the leakage power is very less comparively dynamic power in both 45nm and 180nm technologies. The leakage power in 45nm is 157nW and leakage power in 180nm is 602 nW.It is negigible comparitively dynamic power. The toatl power is 326 μ W for proposed FIR filter using 45nm technology, which is optimized power.

Table II Synthesis results of 45nm and 180nm CMOS technologies using RTL compiler.

Technology	Power consumption(μ W)	Area (μm^2)	Delay (ns)	Clock frequency (M Hz)
TSMC 45-nm CMOS	326	2445	4.487	222.866
TSMC 180-nm CMOS	16967	91925	7.391	135.299

The table III presents the comparison between the existing FIR filter structures from survey and proposed structure of FIR filter. The number of multipliers required for the proposed design is 64 same as the direct form architecture [4] and transpose form structure [5], but the less number of delay elements or FFs comparatively all the existing structures. Due to the less number of delay elements the over delay for the proposed FIR filter is reduced to 4.487ns. The clock speed for the proposed transpose form FIR filter is achieved for 45nm technology is 222MHz. The adder blocks also 197, which is lesser than filter existing architectures [2] and [3]. The area occupied by the proposed structure is very much smaller than the existing FIR filter structures. The more area optimization is achieved by this FIR filter in 45nm technology.

Table III Comparison between different FIR filters parameters.

	Multipliers	Adders	FF	Power consumption	Area (μm^2)
Direct form structure	64	60	120	55.3 mW	112723
Park et al	0	71	296	24.9 mW	56325
Mahesh et al	0	114	360	28.7 mW	67949
Mohanty et al	64	60	312	59.3 mW	123489
Proposed	64	60	197	326 μW	2445

6. CONCLUSION

The optimized block based transpose form FIR filter is realized using retiming with less number of delay elements for the low power consumption, low area and high speed. The basic transpose form FIR filter DFG is converted into modified DFG to avoid the redundancy for block based inputs. The retiming is introduced by changing the location of D flip flops for the optimization of the FIR filter. The constraints are applied in the synthesis tool to reduce the delay, area and power consumption of FIR filter. The entire block based transpose form FIR filter structure is implemented in Verilog HDL code and simulated using ISE simulator. The design synthesized using XST tool and again synthesized using RTL compiler for two different technologies, such as, TSMC 45nm and TSMC 180nm CMOS technology. From the comparison between these two technologies, the 45nm technology gives better results in terms of area, delay and power consumption. The area and utilization summary is given by XST tool and power report and delay reports are obtained by RTL synthesis tool.

REFERENCES

- [1] A. Umasankar and N. Vasudevan, "Design And Analysis of Various Slice Reduction Algorithm for Low Power and Area Efficient FIR Filter", ICCTET13, IEEE Conf. July 2013.
- [2] R Mahesh and A.P Vinod, "New Reconfigurable Architectures for Implementing FIR Filters with low Complexity" IEEE Transactions, Computer Aided Design Integr. Circuits Syst., Vol. 29, no 2, pp. 275-288, Feb. 2010

- [3] S. Y. Park and P. K. Meher, "Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 7, pp. 511–515, Jul. 2014.
- [4] B. K. Mohanty and P.K.Meher " A gigh performance energy efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm" *IEEE Trans. Signal Process.*, vol. 61,no.4, pp. 921-932, feb. 2013
- [5] B. K. Mohanty and P. K. Meher, "A high- performance FIR Filter Architecture for Fixed and Reconfigurable Applications," *IEEE Trans. on VLSI systems*, vol. 24, issue 2, pp.444 –452, 2016.
- [6] A. P. Vinod and E. M. Lai, "Low power and high-speed implementation of FIR filters for software defined radio receivers," *IEEE Trans. Wireless Commun.*, vol. 7, no. 5, pp. 1669–1675, Jul. 2006.
- [7] Keshab k. Parhi "VLSI Digital Signal Processing Systems- Design and Implementation" john wiley & sons, in 1999.
- [8] B. K. Mohanty and P. K. Meher, "A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm," *IEEE Trans. Signal Process.*, vol. 61, no. 4, pp. 921–932, Feb. 2013.
- [9] R. Mahesh and A. P. Vinod, "A new common sub-expression elimination algorithm for realizing low-complexity higher order digital filters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 217–219, Feb. 2008.
- [10] J. Park, W. Jeong, H. Mahmoodi-Meimand, Y. Wang, H. Choo, and K. Roy, "Computation sharing programmable FIR filter for low-power and high-performance applications," *IEEE J. Solid State Circuits*, vol. 39, no. 2, pp. 348–357, Feb. 2004.
- [11] K.-H. Chen and T.-D. Chiueh, "A low-power digit-based reconfigurable FIR filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 617–621, Aug. 2006.