# RECONFIGURABLE FIR FILTER WITH RADIX-4 ARRAY MULTIPLIER

**[1]S.KARTHICK, [2] Dr. S. VALARMATHY, [3]E.PRABHU**
[1]Assistant Professor (Sr.G), [2]Professor& Head
Department of Electronics and Communication Engineering,
Bannari Amman Institute of Technology, Sathyamangalam.
[3]Assistant Professor, Department of Electronics and Communication Engineering
Amrita School of Engineering, Amrita Vishwa Vidyapeetham University, Coimbatore, India
[1]skarthick1284@gmail.com, [2]atrmathy@rediffmail.com, [3]eprabhu85@gmail.com

## ABSTRACT

FIR filters are commonly used digital filters which find its major application in digital signal processing. In conventional FIR filter, the input vector form is delayed by one sample and then multiplied with filter coefficients which are subsequently accumulated by the adders. The drawbacks due to this are high device utilization and high power consumption. In order to compensate these drawbacks, we propose a reconfigurable FIR filter using radix-4 multiplier. The major changes in the proposed system are radix-4 multiplier for multiplication and change in the basic architecture of the FIR filter. In this method, we combine all the input tap values having similar co-efficient values and then multiplying those with the respective co-efficient. The proposed design is simulated and synthesized using Xilinx. The proposed method is compared with the existing FIR filter**.** From the results, it is observed that our proposed method has got better results by having less number of occupied slices and low power consumption. The power analysis report of a 8-tap FIR filter using the proposed approach consumes 60µW at 25MHz, 110µW at 50MHz, 170µW at 75MHz and 220µW at 100MHz compared with the existing approach which was implemented on Spartan-3E. Additionally, the proposed design was also tested for n-tap FIR filter implemented in Virtex-4 FPGA and compared with the existing technique, which shows that our approach minimizes the number of slices occupied by the design and reduces the power consumption.

***Keywords:*** *Reconfigurable FIR Filter, Radix 4 Multiplier, Power Consumption, Device Utilization.*

## 1. INTRODUCTION

Finite-Impulse Response (FIR) filters play a crucial role in many signal processing applications in communication systems. A wide variety of tasks such as spectral shaping, matched filtering, interference cancellation, channel equalization, etc. can be performed with these filters. [4] Traditionally, digital filters are achieved in Digital Signal Processor (DSP), but DSP based solution cannot meet the high speed requirements in some applications for its sequential structure. Nowadays, Field Programmable Gate Array (FPGA) technology is widely used in digital signal processing area because FPGA-based solution can achieve high speed due to its parallel structure and configurable logic, which provides great flexibility and high reliability in the course of design and later maintenance. In general, digital filters are divided into two categories, including Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters FIR filters are widely applied to a variety of digital signal processing areas for the virtues of providing linear phase and system stability. The FPGA-based FIR filters using traditional direct arithmetic costs considerable multiply-and-accumulate (MAC) blocks with the augment of the filter order.

Discrete Wavelet Transform (DWT) provide a time-frequency representation of a signal. They were developed to overcome the problem of time-frequency localization in short time Fourier transform (STFT) [4, 11]. Vast improvements in applications of DWT have been concluded to its hardware implementations. Hardware platforms such as FPGAs make easy the implementation of DWT [11, 13]; however, the operating frequency and occupied area are considerations that must be taken into account for the implementation of DWT. There are several architectures that can be used for hardware implementations of DWT. They are generally selected based on applications, desired speed, and availability of logical gates in the target hardware. A parallel, high speed implementation of

DWT using the Improved Distributed Arithmetic (IDA) is introduced in [11, 1]. A novel architecture for two-dimensional DWT is presented in [11, 8]. This architecture works in a non-separable fashion using a parallel filter structure with distributed control to compute all DWT resolution levels. Similarly, a VLSI implementation of DWT based on the lifting scheme is presented in [11, 14]. This architecture uses quantized coefficients and fixed point data precision to reduce hardware demands. The other reports present a filter-bank implementation of DWPT (Discrete Wavelet Packet Transform) with the ability to select the best wavelet packet in [11, 10, 3].

In this paper, we propose a reconfigurable FIR filter using radix 4multiplier to overcome the short comings of the conventional FIR filter. We make considerable changes in the existing FIR filter so as to improve the performance and lower the power dissipation. The major changes are employing the Radix 4 multiplier for multiplication and change in the basic architecture of the FIR filter. Use of Radix 4 multiplier leads to fewer digits, fewer cycles and fewer partial products which results in significant gain in speed over basic multipliers and higher performance. In our architecture, we combine all the input tap values having similar co-efficient values and then multiplying those with the respective co-efficients. This will result in decreasing the number of multiplications.

The rest of the paper is organized as follows: Section 2 gives the literature review. Section 3 describes the proposed FIR Filter. Section 4 presents results and discussions. Conclusions are summed up in Section 5.

## 2. REVIEW OF RELATED WORKS

A handful of researches have contributed in the area of Reconfigurable FIR Filter structure in DWT on FPGA platform. In this section the review of recent works is presented. J. Chaitanya*et al.* in [4], presented design and implementation of FIR filter using Distributed Arithmetic and Dynamic Distributed Arithmetic. Distributed Arithmetic structure increases the resource usage while pipeline structure increased the system speed. Their work also presented the Dynamic Distributed Arithmetic algorithm suitable for designing digital filter with varying co-efficient as compared to the conventional Distributed arithmetic algorithm. The extensive use of a Dynamic Distributed Arithmetic algorithm eliminated the multiplier unit which required more number of adders. Kuan-Hung Chen *et al.* in [8] presented a digit-reconfigurable finite impulse response (FIR) filter architecture with a very fine granularity. It provided a flexible yet compact and low-power solution to FIR filters with a wide range of precision and tap length. Based on their approach, an 8-digit reconfigurable FIR filter chip was implemented in a single-poly quadruple-metal 0.35-µm CMOS technology.

A design technique for deriving highly efficient multipliers that operate on a limited range of multiplier values is presented by Turner, R.H. et al. in [14]. Using the technique, Xilinx Virtex field Programmable Gate Array (FPGA) implementation for a Discrete Cosine Transform (DCT) and poly-phase filter were derived with area reductions of 31%-70% and speed increases of 5%-35% when compared to designs using general-purpose multipliers. The technique gives superior results over other fixed coefficient methods and is applicable to a range of FPGA technologies. The two reconfigurable architectures of low complexity FIR filters are presented by Mahesh, R. et al. in [10], namely constant shifts method and programmable shifts method. They presented FIR filter architecture capable of operating for different word length filter coefficients without any overhead in the hardware circuitry. They showed that dynamically reconfigurable filters could be efficiently implemented by using common sub expression elimination algorithms. Their architectures have been implemented and tested on Virtex 2v3000ff1152-4 field-programmable gate array with a precision of 16 bits. Design examples show that their architectures offer good area, power reductions and speed improvements compared to the best existing reconfigurable FIR filter implementations.

The elaborate design of folded Finite-Impulse Response (FIR) filters based on pipelined multiplier arrays was presented by Bougas, P. *et al.* in [5]. The design was considered at the bit-level and the internal delays of the pipelined multiplier array were fully exploited in order to reduce hardware complexity. Both direct and transposed FIR filter forms were considered. The carry-save and the carry-propagate multiplier arrays were studied for the filter implementations. Partially folded architectures are also presented which are implemented by cascading a number of folded FIR filters. Their schemes are compared as to the aspect of hardware complexity with a straightforward implementation of a folded FIR filter based on the pipelined Wallace Tree multiplier. The comparison revealed that their approach required 20%-30% less hardware.

Meher, P. K. in [9] presented that the look-up-table (LUT) multiplier-based approach, where the memory elements store all the possible values of products of the filter coefficients could be an area-efficient alternative to DA-based design of FIR filter with the same throughput of implementation. By operand and inner-product decompositions, respectively, they designed the conventional LUT-multiplier-based and DA-based structures for FIR filter of equivalent throughput, where the LUT-multiplier-based design involves nearly the same memory and the same number of adders, and less number of input register at the cost of slightly higher adder-widths than the other. Moreover, they presented two approaches to LUT-based multiplication, which could be used to reduce the memory size to half of the conventional LUT-based multiplication. Besides, they presented a modified transposed form FIR filter, where a single segmented memory-core with only one pair of decoders are used to minimize the combinational area. They approached LUT-multiplier-based design which involves nearly 15% less area than the DA-based design for the same throughput and lower latency of implementation.

Efficient reconfigurable VLSI architecture for 1D 5/3 and 9/7 wavelet transforms adopted in JPEG2000 proposal, based on lifting scheme is presented by ChengyiXiong*et al.* in [5]. The embedded decimation technique based on fold and time multiplexing, as well as embedded boundary data extension technique, was adopted to optimize the design of the architecture. These reduce significantly the required numbers of the multipliers, adders and registers, as well as the amount of accessing external memory, and lead to decrease efficiently the hardware cost and power consumption of the design. Their architecture was designed to generate an output per clock cycle, and the detailed component and the approximation of the input signal are available alternately. Experimental results are presented, that their architecture have lower hardware complexity, thus it was adapted for embedded applications.

Benkrid*et al.* in [2] presented four different area-efficient Field-Programmable Gate-Array (FPGA) bit-parallel architectures of Finite Impulse Response (FIR) filters that smartly support the technique of symmetric signal extension while processing finite length signals at their boundaries. The key to this was a clever use of variable-depth shift registers which were efficiently implemented in Xilinx FPGAs in the form of Shift Register Logic (SRL) components. Comparisons with the conventional architecture of FIR filter with symmetric boundary processing show considerable area saving especially with long-tap filters. For instance, their architecture implementation of the 8-tap low Daubechies-8 FIR filter achieves ~ 30% reduction in the area requirement (in terms of slices) compared to the conventional architecture while maintaining the same throughput. Two of the above-cited architectures were dedicated to the special case of symmetric FIR filters. The first architecture presented highly area-efficient but required a clock frequency doubler. While doing this, it reduced the overall processing speed. Moreover, this speed penalty is cancelled in bi-phase filters which are widely used in multirate architectures. Their second symmetric FIR filter architecture saved less logic than the first architecture but overcomes its speed penalty as it matches the throughput of the conventional architecture.

## 3. PROPOSED RECONFIGURABLE FIR FILTER

In this paper, we have designed and developed a reconfigurable multiplier block structure for a FIR filter in DWT. Reconfigurable FIR Filter is built using a radix array multiplier block structure. Here, we combine all the inputs having the same coefficient together and then multiply the same with the respective coefficient using the radix array multipliers.

### 3.1 Conventional FIR Filter

FIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications, the other type being IIR. FIR means Finite Impulse Response and in simple terms, if we put in an impulse, that is, a single "1" sample followed by many "0" samples, zeroes will come out after the "1" sample has made its way through the delay line of the filter. In the common case, the impulse response is finite as there is no feedback in the FIR and the lack of feedback guarantees that the impulse response will be finite. The impulse response of a FIR filter is actually just the set of FIR coefficients and a FIR tap. The number of FIR taps is an indication of the amount of memory required to implement the filter and the number of calculations required. In effect, more taps means more stop band attenuation, less ripple, narrower filters, etc and in our case we use 32- tap FIR filters. In a FIR context, a "MAC" is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result and usually FIRs usually require one MAC per tap. Delay Line is the set of memory elements that implement the $Z^{-1}$ delay elements of the FIR calculation.

FIR filter is a filter whose impulse response is of finite duration, because it settles to zero in finite time. The impulse response of an $N^{th}$ order discrete-time FIR filter lasts for N + 1 samples, and then settles to zero. FIR filters can be discrete-time or continuous-time, and digital or analog. Direct realization of FIR filter is based on the direct implementation of this expression:

$$y[n] = \sum_{k=0}^{N-1} h[k].x[n-k] \tag{1}$$

where, $h$ is the coefficient value and $x$ is the input. Since h[n] and x[n] are finite duration sequences, their convolution is also finite in duration. The block diagram of the conventional 32 tap FIR filter is given in Fig. 1. We can see that the signal flow is strictly feed-forward, since all the paths connecting the input to the output flow is in the forward direction.

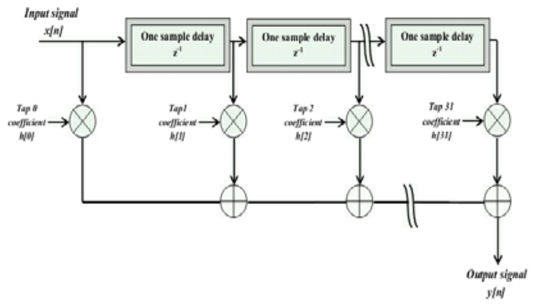


*Fig. 1. Conventional 32 Tap FIR Filter*

The transfer function of the FIR filter is defined by:

$$H(z) = \frac{y[z]}{x[z]} \tag{2}$$

**3.2 Proposed FIR Filter With Radix 4 Multiplier**

The working of conventional FIR filter has already been discussed in the earlier section. In conventional FIR filter, the input vector form is delayed by one sample and then multiplied with filter coefficients which are subsequently accumulated by the adders. In our proposed technique, we make certain changes to the existing FIR filter so as to improve the performance and lower the power dissipation. The major changes are employing the radix 4 multiplier for multiplication and change in the basic architecture of the FIR filter.

Multiplication is a heavily used arithmetic operation that figures prominently in signal processing applications. Multiplication is hardware intensive, and the main criteria of interest are higher speed, lower cost, and less VLSI area.The main concern in classic multiplication often realized by K cycles of shifting and adding, is to speed up the underlying multi-operand addition of partial products. In a DSP system, the multiplier must be fast and must havesufficient precision to support the desired application. A high quality filter will in general require more multiplications than one of lesser quality, so throughput suffers if the multiplier is not fast. Conventional FIR filter uses normal multiplier which is comparatively slow and of lesser quality. In our proposed FIR, we use radix-4 multiplier which is faster and yields good results. The use of radix 4 multipliers improves the FIR filter characteristics by improving the power efficiency and performance.

Booth multipliers allow the operation on signed operands in 2's-complement. They derive from array multipliers where, for each bit in a partial product line, an encoding scheme is used to determine if this bit is positive, negative or zero. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In radix 4 multiplication scheme, it handles more than one bit of the multiplier in each cycle which leads to fewer digits. This also requires fewer cycles, which means fewer partial products. The reduction in the number of cycles leads to significant gains in speed over basic multipliers. In radix array multipliers, the structure of the radix-r array multiplier architecture is the same as the plain array multiplier. Here, each partial product line operates on groups of m-bits instead of a single bit. This reduces the number of product lines to $\left(\frac{w}{\log_2 r}\right)$ where; w is number of bits of operands. The optimization of the partial product generation modules enables the performance and power improvement.

In radix 4 operation, based on two least significant bits (LSB) of multiplier, a pre-computed multiple of 'x' is added. It is implemented by the use of a multiplexer to select the value of 'x' based on the input select line (last two bits) and is given to adder for the addition process. The different case scenarios are given in Table 1. The architecture for this operation is given in Fig 2.

*Table. 1. Select Line And The Corresponding 'A' Value*

| ***Select Line*** | ***'x' value*** |
|---|---|
| 00 | 0 |
| 01 | X |
| 10 | 2x |
| 11 | 3x |

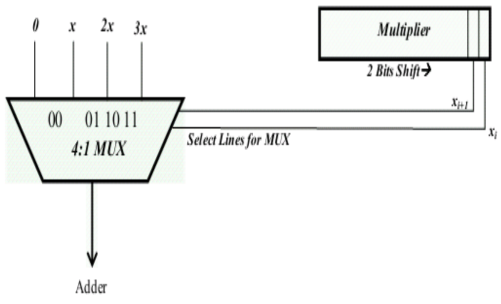


*Fig. 2. The Architecture For Radix Operation*

Alternately, rather than adding 3x, add –x and send a carry of 1 into the next radix-4 digit of the multiplier which is shown in Fig.3.

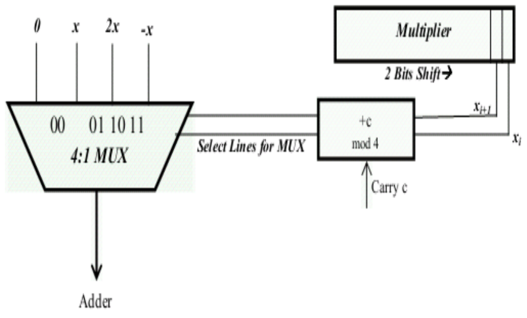


*Fig. 3. Alternative Architecture For Radix Operation*

Let the operand be represented by $X$ so that each of the bits of the operator is represented as $x_i$ for $0 \le i \le M-1$, where $M$ is the total number of bits. The most significant bit which represents the sign bit is given by $x_{M-1}$.If $X$ is positive, its representation is the same as for an unsigned number, and in case its negative, it is represented as $2^M - X$. Conversely, the value of the operand can be computed as follows:

$$X = \begin{cases} X, \; if \;\; x_{M-1} = 0 \\ X - 2^M, \;\; if \;\; x_{M-1} = 1 \end{cases} \qquad (3)$$

$X'$ is defined as:

$$X' = \sum_{i=0}^{M-2} x_i.2^i \qquad (4)$$

the above equation can be re-written as:

$$X = \begin{cases} X', if \;\; x_{M-1} = 0 \\ X' - 2^{M-1}, \;\; if \;\; x_{M-1} = 1 \end{cases} \qquad (5)$$

Then we have:

$$X = X' - x_{M-1} 2^{M-1} \qquad (6)$$

For considering 2's complement in radix 2m multiplication,

$$X' = \sum_{i=0}^{\frac{M}{m}-2} x_i 2^{i.k} \qquad (7)$$

where, $x_i$ is a $m$ bit digit. Then we have,

$$X = \begin{cases} X', if \;\; x_{M-1} = 0 \\ X' - x_{\frac{M}{m}-1} 2^{M-m}, \;\; if \;\; x_{M-1} = 1 \end{cases} \qquad (8)$$

Using analogous observation, we can write as:

$$X \times Y = X' \times Y' - X'y_{M-1}y_{\frac{M}{m}-1} 2^{M-m} - x_{M-1}x_{\frac{M}{m}-1} \sum_{j=0}^{\frac{M}{m}-1} b_j.2^{M-m+j} \qquad (9)$$

The working example of the multiplication procedure for 2's complement 8-bit wide radix-4 is given below in Fig. 3 and the example architecture is given in Fig. 4.for the inputs

X= -10, Y= -98;

$(10)_{dec}$ = $(00001010)_{binary}$ ; Taking 1's complement, we get $(1111\ 0101)_{binary}$ and taking 2's complement we get (11110110);

That is $(-10)_{dec}$ = $(11110110)_{binary}$.

$(98)_{dec}$ = $(01100010)_{binary}$ ; Taking 1's complement, we get $(10011101)_{binary}$ and taking 2's complement we get (10011110);

That is $(-98)_{dec}$ = $(10011110)_{binary}$.
That is $X \times Y$ = $(11110110)_{binary} \times$ $(10011110)_{binary}$.

The operation is carried out by multiplying two bits of Y with X and performing 2 bit shift every time. Here the X is multiplied with Y pairs of 10, 11,01 and 10 in the respective position and 2 bit left shift is carried out. The output values for the different pair inputs are given in table 2.

*Table: 2. Output Values For The Different Pair Inputs*

| | Input | Bit Pair | Result Value |
|---|---|---|---|
| ***Case 1*** | 11110110 | 00 | 00000000 |
| ***Case 2*** | 11110110 | 01 | 11110110 |
| ***Case 3*** | 11110110 | 10 | 111101100 |
| ***Case 4*** | 11110110 | 11 | 1111100010 |

In our case second multiplicand (Y) will be broken down to four set of pair bits so as to perform the multiplication with the first multiplicand in the radix multiplication procedure. That is Y=10011110, will be split to {10}, {01}, {11},{10} in the process of radix multiplication.

That is, in radix 4 multiplication scheme, it handle more than one bit of the multiplier in each cycle which leads to fewer digits, fewer cycles and fewer partial products which results in significant gains in speed over basic multipliers and higher performance.

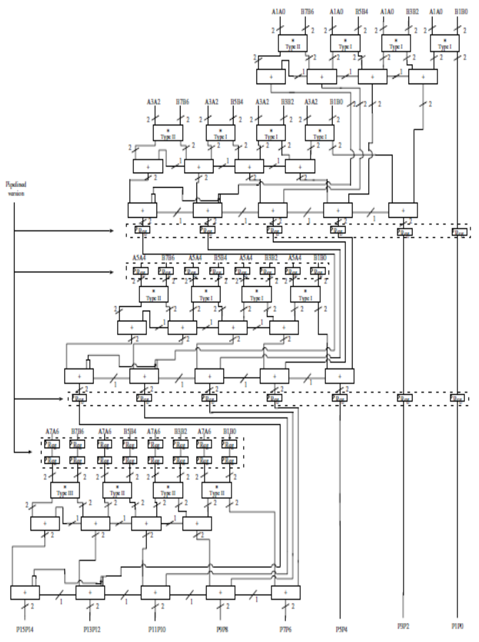


*Fig.4. Example Architecture Of A 8-Bit Wide 2's Complement Radix-4 Array Multiplier*

Basic architecture of the FIR structure is also changed to improve the performance of the system. Here emphasis is given to reducing the number of multiplications performed so as to reduce power and improve efficiency. The idea behind is to combine all the input tap values having similar co-efficient values and then multiplying those with the respective co-efficients. This will result in decreasing the number of multiplications. That is, instead of multiplying all the tap values with co-efficient values, here all the tap values with the same co-efficients are added and finally multiplied with the co-efficient value.

Initially, all input taps having same co-efficient values are found out. The input tap values can be represented by x[n], where $0 \le n \le 31$ as we are considering 32 tap FIR filter. The co-efficient values are represented by h[n]. In the existing technique, the output is found using

$$y[n] = x(31)\times h(0) + x(30)\times h(1) + ... + x(0)\times h(31) \tag{10}$$

The 32 number of multiplications happen inside the FIR filter for the given an input. In our proposed technique, sample delay of input is taken and adds those pairs of input taps with same co-efficient values first and then multiply by the respective co-efficient values. Suppose $x(0), x(2), x(6), x(31)$ are having the same co-efficient values of $a$, $x(1), x(5), x(16)$ are having the same co-efficient values of $b$, $x(19), x(20)$ are having the same co-efficient values of $c$ and likewise for other co-efficients, then according our proposed technique, we have:

$$y[n] = [x(0)+x(2)+x(6)+x(31)]\times a + [x(1) + x(5)+x(16)]\times b + ... + [x(19)+x(20)]\times c \tag{11}$$

From these we can observe a definite decrease in number of multiplications needed and it results in an improved FIR filter. For multiplication, we use radix 4 array multiplier which again results in lower power, higher speed and better performance. The block diagram of the proposed technique is given in Fig 5.

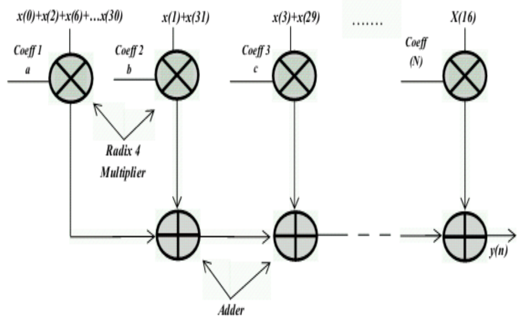


*Fig. 5.The Block Diagram Of The Proposed FIR Filter*

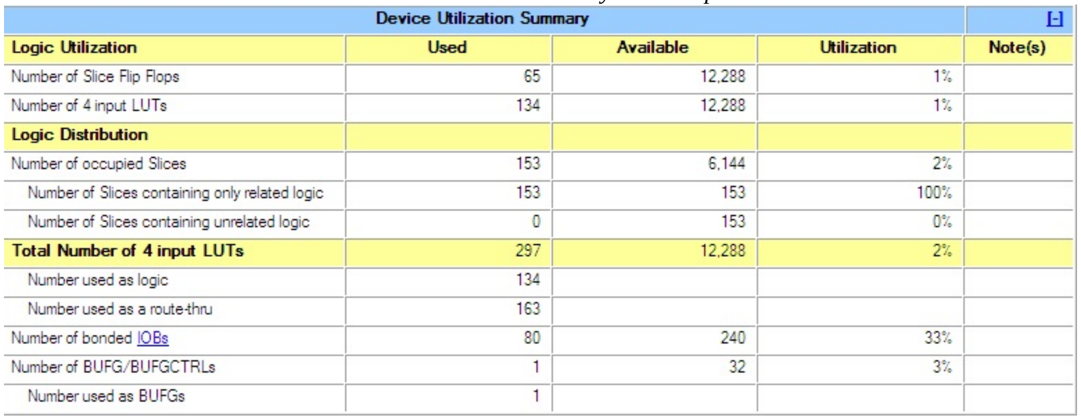
## 4. RESULTS AND DISCUSSION

The experimental results of our proposed FIR filter are presented below. The proposed design is simulated and synthesized using Xilinx 10.1 version. In section 4.1 results obtained for n-tap FIR filter are presented.

## 4.1 Synthesized Results for n-Tap FIR Filter

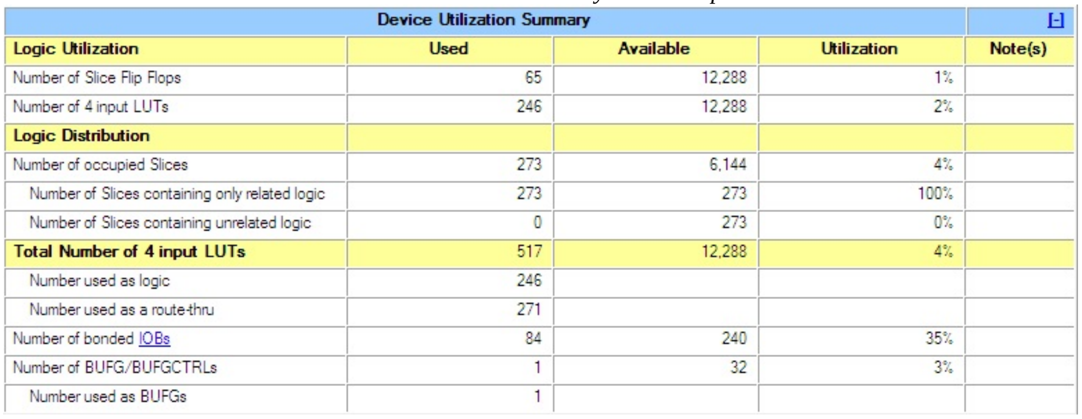
The system under test is n-Tap FIR filter for various taps= 6, 10, 13, 20, 28, 41, 61 for which the Synthesized device utilization report is presented in the following tables [3-9].

*Table: 3 Device Utilization Summary For 6-Tap FIR Filter.*



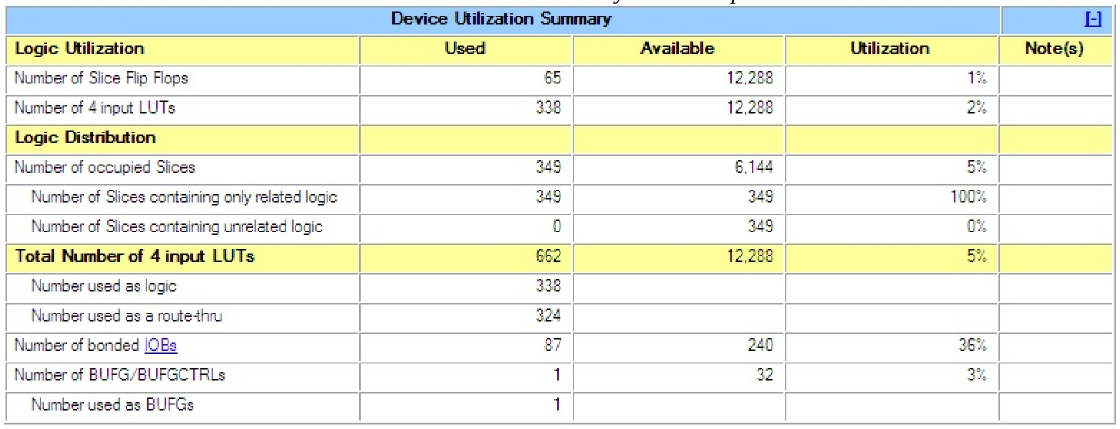
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 134 | 12,288 | 1% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 153 | 6,144 | 2% | |
| Number of Slices containing only related logic | 153 | 153 | 100% | |
| Number of Slices containing unrelated logic | 0 | 153 | 0% | |
| **Total Number of 4 input LUTs** | 297 | 12,288 | 2% | |
| Number used as logic | 134 | | | |
| Number used as a route-thru | 163 | | | |
| Number of bonded IOBs | 80 | 240 | 33% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 4 Device Utilization Summary For 10-Tap FIR Filter.*



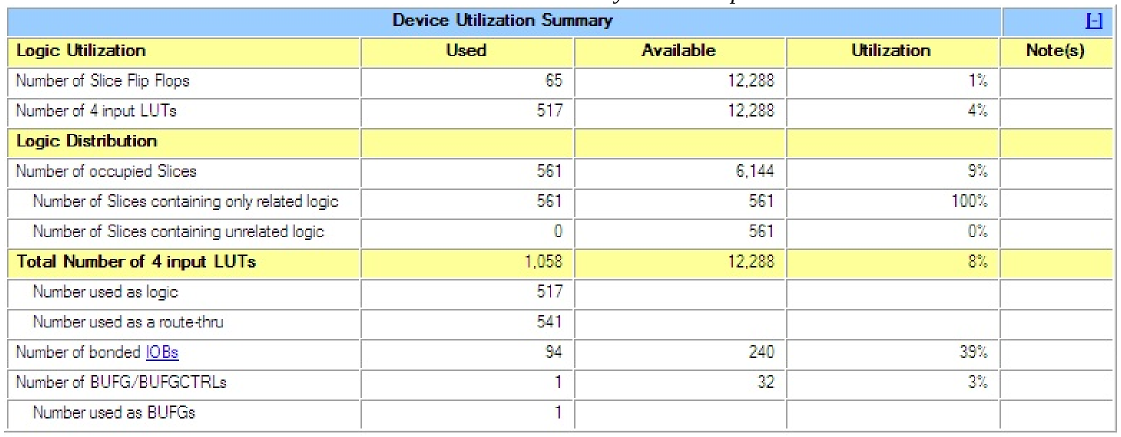
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 246 | 12,288 | 2% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 273 | 6,144 | 4% | |
| Number of Slices containing only related logic | 273 | 273 | 100% | |
| Number of Slices containing unrelated logic | 0 | 273 | 0% | |
| **Total Number of 4 input LUTs** | 517 | 12,288 | 4% | |
| Number used as logic | 246 | | | |
| Number used as a route-thru | 271 | | | |
| Number of bonded IOBs | 84 | 240 | 35% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 5 Device Utilization Summary For 13-Tap FIR Filter.*



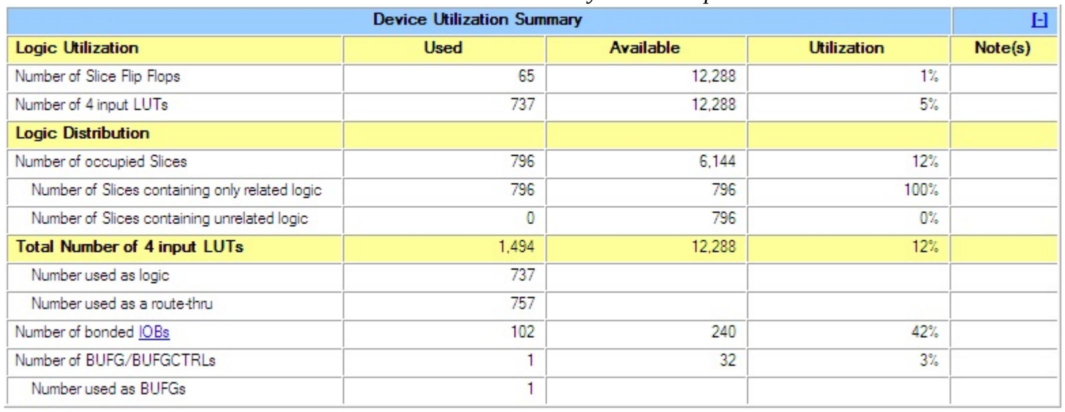
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 338 | 12,288 | 2% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 349 | 6,144 | 5% | |
| Number of Slices containing only related logic | 349 | 349 | 100% | |
| Number of Slices containing unrelated logic | 0 | 349 | 0% | |
| **Total Number of 4 input LUTs** | 662 | 12,288 | 5% | |
| Number used as logic | 338 | | | |
| Number used as a route-thru | 324 | | | |
| Number of bonded IOBs | 87 | 240 | 36% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 6 Device Utilization Summary For 20-Tap FIR Filter.*



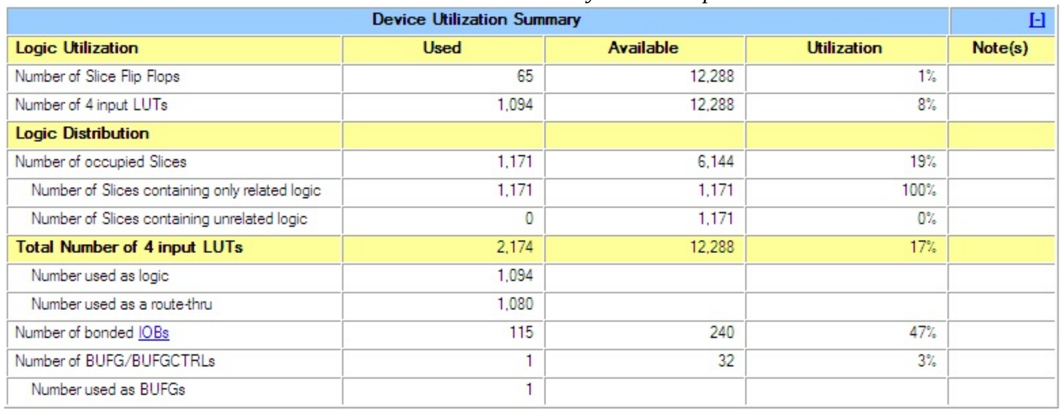
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 517 | 12,288 | 4% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 561 | 6,144 | 9% | |
| Number of Slices containing only related logic | 561 | 561 | 100% | |
| Number of Slices containing unrelated logic | 0 | 561 | 0% | |
| **Total Number of 4 input LUTs** | 1,058 | 12,288 | 8% | |
| Number used as logic | 517 | | | |
| Number used as a route-thru | 541 | | | |
| Number of bonded IOBs | 94 | 240 | 39% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 7 Device Utilization Summary For 28-Tap FIR Filter.*



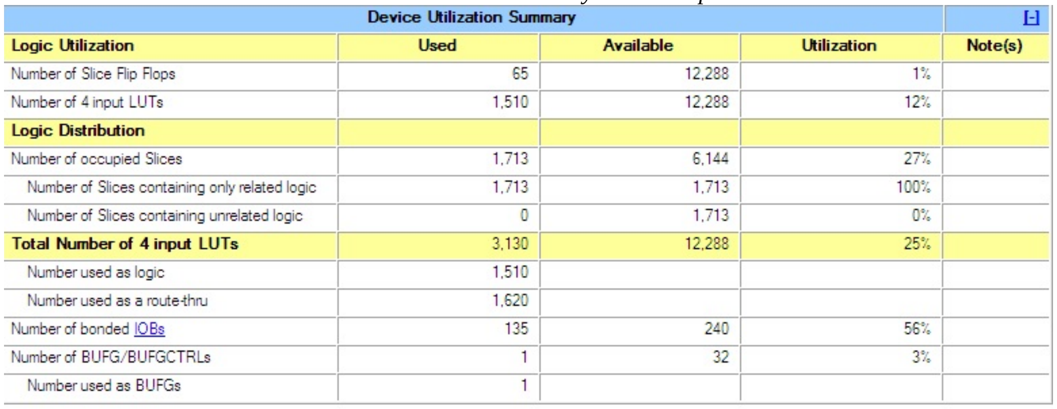
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 737 | 12,288 | 5% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 796 | 6,144 | 12% | |
| Number of Slices containing only related logic | 796 | 796 | 100% | |
| Number of Slices containing unrelated logic | 0 | 796 | 0% | |
| **Total Number of 4 input LUTs** | 1,494 | 12,288 | 12% | |
| Number used as logic | 737 | | | |
| Number used as a route-thru | 757 | | | |
| Number of bonded IOBs | 102 | 240 | 42% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 8 Device Utilization Summary For 41-Tap FIR Filter.*



| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 1,094 | 12,288 | 8% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 1,171 | 6,144 | 19% | |
| Number of Slices containing only related logic | 1,171 | 1,171 | 100% | |
| Number of Slices containing unrelated logic | 0 | 1,171 | 0% | |
| **Total Number of 4 input LUTs** | 2,174 | 12,288 | 17% | |
| Number used as logic | 1,094 | | | |
| Number used as a route-thru | 1,080 | | | |
| Number of bonded IOBs | 115 | 240 | 47% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

*Table: 9 Device Utilization Summary For 61-Tap FIR Filter.*



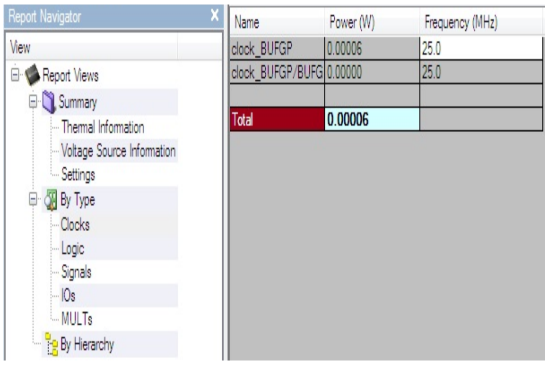
| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 65 | 12,288 | 1% | |
| Number of 4 input LUTs | 1,510 | 12,288 | 12% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 1,713 | 6,144 | 27% | |
| Number of Slices containing only related logic | 1,713 | 1,713 | 100% | |
| Number of Slices containing unrelated logic | 0 | 1,713 | 0% | |
| **Total Number of 4 input LUTs** | 3,130 | 12,288 | 25% | |
| Number used as logic | 1,510 | | | |
| Number used as a route-thru | 1,620 | | | |
| Number of bonded IOBs | 135 | 240 | 56% | |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% | |
| Number used as BUFGs | 1 | | | |

The synthesized report of the proposed FIR filter is compared with the existing technique in section 4.3 in terms of number of occupied slices.

## 4.2 Power Analysis Report For the proposed 8-Tap FIR Filter

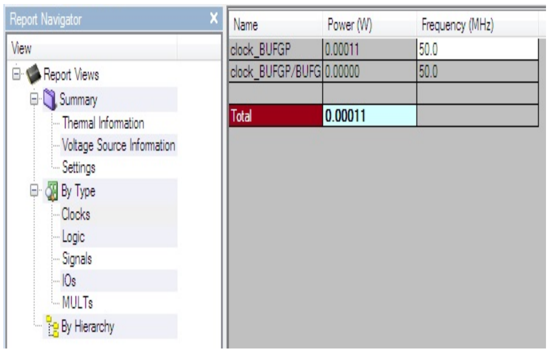
The system under test is the proposed 8-Tap FIR filter for which the Xpower Analysis report is generated at different frequencies and the same is presented in tables[10-13].

*Table: 10 Xpower Analysis Report For 8-Tap FIR At 25mhz*



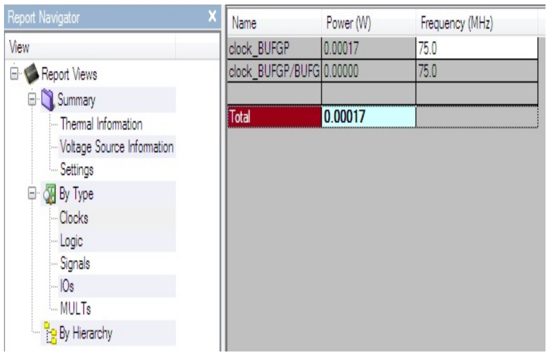
| Name | Power (W) | Frequency (MHz) |
|---|---|---|
| clock_BUFGP | 0.00006 | 25.0 |
| clock_BUFGP/BUFG | 0.00000 | 25.0 |
| Total | 0.00006 | |

*Table: 11 Xpower Analysis Report For 8-Tap FIR At 50mhz*



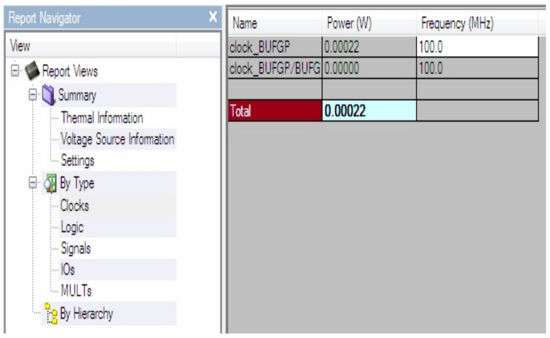
| Name | Power (W) | Frequency (MHz) |
|---|---|---|
| clock_BUFGP | 0.00011 | 50.0 |
| clock_BUFGP/BUFG | 0.00000 | 50.0 |
| Total | 0.00011 | |

*Table: 12 Xpower Analysis Report For 8-Tap FIR At 75mhz*



| Name | Power (W) | Frequency (MHz) |
|---|---|---|
| clock_BUFGP | 0.00017 | 75.0 |
| clock_BUFGP/BUFG | 0.00000 | 75.0 |
| Total | 0.00017 | |

*Table: 13 Xpower Analysis Report For 8-Tap FIR At 100mhz*



| Name | Power (W) | Frequency (MHz) |
|---|---|---|
| clock_BUFGP | 0.00022 | 100.0 |
| clock_BUFGP/BUFG | 0.00000 | 100.0 |
| Total | 0.00022 | |

The generated power analysis repot indicates that the proposed system consumes 60µW at 25MHz, 110µW at 50MHz, 170µW at 75MHz and 220µW at MHz. The comparisons of our proposed results are discussed in the following section.

### 4.3 Comparison of N- Tap FIR Filters Implemented For Different Devices

Design of n-tap FIR filter was verilog coded in Xilinx 10.1 version and was implemented on several FPGA devices, for the same the sysnthesis report and the Xpower analysis report are compared such that the power consumption and device area utilization is compared with recent techniques presented in [6] and [12].

*Table: 14 Comparison Of Number Of Slices Occupied By The Proposed N-TAP FIR Filter With Respect To The Existing* techniques

| Filter taps | Add shift method [13] | MAC filter[13] | **Proposed method** |
|---|---|---|---|
| | Virtex-4 | Virtex-4 | Virtex-4 |
| 6 | 264 | 219 | 153 |
| 10 | 475 | 418 | 273 |
| 13 | 387 | 462 | 349 |
| 20 | 851 | 790 | 561 |
| 28 | 1303 | 886 | 796 |
| 41 | 2178 | 1660 | 1171 |
| 61 | 3284 | 1947 | 1713 |

The proposed n-Taps FIR Filters were implemented on Vertix-4 FPGA and compared with the existing technique presented in table-12 above. The comparison shows that the proposed technique reduces the number of slice required in comparison to the add shift method and MAC filter method in [6].

*Table: 15 Comparison of Power consumption by the proposed 8-TAP FIR Filter with the existing techniques*

| Method | 25MHz | 50MHz | 75MHz | 100MHz |
|---|---|---|---|---|
| | Spartan-3E FPGA | | | |
| | Power consumption (mW) | | | |
| MAC_booth[14] | 100 | 140 | 180 | 210 |
| Linear Phase Folding Booth[14] | 110 | 150 | 190 | 230 |
| Shift add [14] | 90 | 120 | 140 | 110 |
| Serial multiplier and serial adder[14] | 112 | 126 | 141 | 155 |
| Booth without DPDT [14] | 469 | 879 | 1279 | 1703 |
| Booth with DPDT using Reg [14] | 326 | 596 | 881 | 1137 |
| Booth with DPDT using AND gate [14] | 868 | 669 | 1018 | 1283 |
| Proposed dynamic power | 0.06 | 0.11 | 0.17 | 0.22 |

The generated power analysis repot indicate that the proposed system tested for a 8-tap FIR filter consumes 60µW at 25MHz, 110µW at 50MHz, 170µW at 75MHz and 220µW at 100MHz compared with the existing approach [12] which was implemented on Spartan-3E and presented in table-15 which consumes less power.

## 5. CONCLUSION

In this paper, we propose a reconfigurable FIR filter using radix 4multiplier. In our method, we combine all the input taps having similar co-efficient values and then multiplying those with the respective coefficients is carried out using radix-4 multiplier. The proposed design is simulated and synthesized using Xilinx 10.1 version. Our proposed method is compared with the recent techniques presented for design of FIR filter. The results obtained for the proposed method has less number of occupied slices and reduced power consumption. The generated power analysis report indicates that the proposed method for a 8-tap FIR filter consumes 60µW at 25MHz, 110µW at 50MHz, 170µW at 75MHz and 220µW at 100MHz compared with the existing approach which was implemented on Spartan-3E. Further, the proposed concept was also tested for n-tap FIR filter implemented on Vertix-4, FPGA and compared with the existing technique. To conclude, our approach minimizes the total number of slices occupied and reduces the power consumption.

## REFERENCES

[1]Amir HosseinGholamipour, FadiKurdahi, Ahmed Eltawil and Mazen A.R. Saghir, "Reducing Reconfiguration Overhead for Reconfigurable Multi-Mode Filters Through Polynomial Time Optimization and Joint Filter Design,"(4)

[2] Benkrid, Abd.S. and Benkrid,K., "Novel Area-Efficient FPGA Architectures for FIR Filtering With Symmetric Signal Extension," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* May-2009, Vol:17, Issue:5, Page(s):709 – 722.(11)

[3] Bougas, P., Kalivas, P., Tsirikos, A., Pekmestzi, K.Z., "Pipelined array-based FIR filter folding," *Circuits and Systems I: Regular Papers, IEEE*

*Transactions on* Jan. 2005, Vol: 52 , Issue: 1, Page(s): 108 – 118.(8)

[4] J.Chaitanya, Ch.Venkateswarlu and P.Sunitha, “Area Efficient Reconfigurable Coefficient Based FIR Filter,” *International Journal of Engineering Research & Technology (IJERT)* on Aug 2012, Vol. 1, Issue 6.(1)

[5] ChengyiXiong, JinwenTian and Jian Liu, “Low complexity reconfigurable architecture for the 5/3 and 9/7 discrete wavelet transform,” *Journal of Systems Engineering and Electronics on* Jun-2006, Vol:17, Issue 2, Pages 303–308.(10)

[6] A. Hemalatha and A. Shanmugam, “A Novel Shift and Add Algorithm for Low Power and Area Efficient FIR Filter,” *European Journal of Scientific Research*, Vol.49, Issue No.4 (2011), pp. 634-641.(13)

[7] Kuan-Hung Chen and KwyroLee, “Low-Power and Area-Efficient FIR Filter Implementation Suitable for Multiple Taps,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* February 2003,Vol:11, Issue:1,Page(s):150-153.(12)

[8] Kuan-Hung Chen and Tzi-Dar Chiueh, “A Low-Power Digit-Based Reconfigurable FIR Filter,” *Circuits and Systems —II: Express Briefs, IEEE Transactions on* Aug 2006, Vol: 53, Issue: 8.(5)

[9] Meher, P. K, “New Approach to Look-Up-Table Design and Memory-Based Realization of FIR Digital Filter,” Circuits and Systems I: Regular Papers, *IEEE Transactions on* March 2010, Vol: 57, Issue:3, Page(s): 592 – 603.(9)

[10] Mahesh, R. Vinod, A.P. “New Reconfigurable Architectures for Implementing FIR Filters With Low Complexity,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* Feb. 2010, Vol: 29, Issue: 2, Page(s): 275 – 288.(7)

[11] Mohsen AmiriFarahani, SaeidMirzaei and HosseinAmiriFarahani, “Implementation of a Reconfigurable Architecture of Discrete Wavelet Packet Transform with Three Types of Multipliers on FPGA,” *Electrical and Computer Engineering (CCECE), on* May-2011 Page(s): 001459 – 001462.(2)

[12] T Ramesh Reddy and Dr. K. Soundara Rajan, “Low Power and Low Area Digital FIR Filter Using Different Multipliers and Adders,” *International Journal of Engineering Research & Technology (IJERT),* Vol. 1, Issue 3, 2012.(14)

[13] SüleymanSırrıDemirsoy, Izzet Kale and Andrew G. Dempster, “Efficient Implementation Of Digital Filters Using Novel Reconfigurable Multiplier Blocks,” *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on* Nov-2004, Volume: 1, Page(s): 461-464.(3)

[14] Turner, R.H. Woods, R.F. “Highly efficient, limited range multipliers for LUT-based FPGA architectures,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* Oct. 2004, Vol: 12, Issue: 10 Page(s): 1113 - 1118 .(6)