International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
www.ijset.com, Volume No.1, Issue No.4, pg : 55-60 01 Oct. 2012

# Pipelined FIR Filter Implementation using FPGA

**A.Kamaraj, C.Kalyana Sundaram, J.Senthilkumar**
Department of Electronics and Communication Engineering
Mepco Schlenk Engineering College, Sivakasi
Email : kamarajvlsi@gmail.com, gullycks@yahoo.co.in, senvimjag@gmail.com

***Abstract:*** **FIR filters are being designed using HDL languages to enhance the speed of the system. In the whole system if the speed of the individual block is enhanced, the overall speed of the system is enhanced. In order to attain effective utilization hardware is done by applying the pipelining technique. Pipelining is an implementation technique in which multiple instructions are overlapped in execution. The proposed design of this paper is an attempt to optimize the system speed with minimal cost and hardware. The central design concept is to build filters with higher operating frequency without sacrificing the performance of original filters.**

**To enhance system speed and reducing implementation complexity, a lot of work has been done in the process of achieving digital signal processing by use of the FPGA. In a filter the pipelining of multiplication is achieved by shifts and addition method. This paper describes the design of Third order low pass FIR filter with pipelined architecture. The design synthesis is done using Xilinx ISE 12.1 and implemented in Spartan-3E FPGA. By pipelining the delay of FIR filters can be reduced. Pipelined technique may reduce delay and enhances speed as compared to non-pipelined technique.**

***Keywords: FIR filter, Non Pipelining, Pipelining, FPGA.***

## I.INTRODUCTION

Filter as the name indicates remove unwanted components of signal. Filters are used for extraction of desired signal from noisy signal which consist of unwanted disturbances. Filters can be broadly described as a signal selection system. In the digital era, the digital filter, which has attracted people’s broad attention more and more, has been widely applied to communication, voice, image, automatic control, radar, aerospace, medicine and so on. As compared to analog filters, digital filters are having higher precision, better stability and reliability. Digital Filters do not have matching problems. Digital filters represent time division multiplexing which can complete some filtering tasks that a number of analog filters are incapable of doing. We can achieve digital filters by hardware circuits. Also we have an alternative approach as computer software programming.

There are two types of digital filters Finite impulse response filter and infinite impulse response filters. Speaking from the realization, IIR adopts recursive structure and uses the rational fraction which is equal to the ratio of two polynomials approximate to the frequency character, so it is able to get better frequency selection characters by use of less order but it is gained at the cost of the non-linear phase characters. At the same time, the existence of feedback slip requires the higher system stability and easily causes oscillation. Compared with the Infinite Impulse Response (IIR), the FIR filter is capable of meeting the strict requirements of the amplitude and phase characters, avoiding the drift and noise and so on, which is generated by the IIR filter, is easily achieved by hardware and has the precise linear-phase and high system stability. In recent years, with the rapid development of the Very Large Scale Integrated (VLSI) technology, FPGA (Field programmable gate array) has been widely applied because of its re-programmability, re-configurability, low-cost, high logic density and high reliability. FPGA, the structured internal logic array and rich connection resources, particularly can be useful to the digital signal processing. FPGA is a programmable logic device which is having user programmable features. This actually can minimize system design risk and maintenance and shorten the design cycle. To enhance system speed and reducing implementation complexity a lot of work have been done in the process of achieving digital signal processing by use of the FPGA.FIR algorithm can use the software to achieve in the Digital Signal Processor (DSP) or Central Processing Unit (CPU), but in a number of systems demanding high real-time, because of the ordinal implementation of the procedures, the software based on real-time often cannot meet the requirements. So it must use hardware to achieve a high speed. Now FPGA becomes a good choice.

## II.DIRECT FORM STRUCTURE OF FIR FILTER:

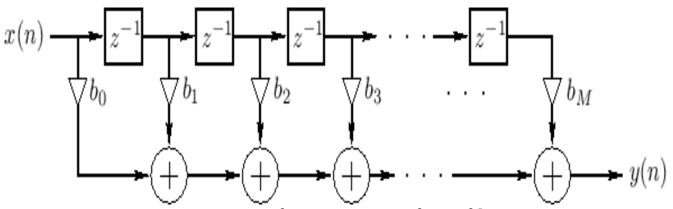


Figure 1.1 Direct form structure of FIR filter

There are three basic modules Adder, Delay and Multiplier modules. For an M tap filter, the number of adder and delay modules will be M-1, and the number of multiplier module will be M.

Differential equation of digital FIR Filter can be described as

$$y(n)= \sum_{i=0}^{M-1} x(n-i)h(i) \quad (1)$$

## III.FIR FILTER DESIGN:

FIR filter has been designed using Window method. Matlab special toolbox FDA Tool (Filter Design & Analysis) has been used to simulate and design filter, the filter amplitude-frequency characteristics meet the requirements .The filter coefficients exported to a text file, 8-order impulse response coefficients of FIR filters are as follows:

FIR Filter Design parameters has been given as follows:
1) Order of filter - 8
2) Window type-keiser window
3) Sampling Frequency fs=5000
4) Cut off frequency fc=1000
5) Input data size-16 bit
6) Filter Coefficients

## IV.DESIGN FLOW OF FIR FILTERS:

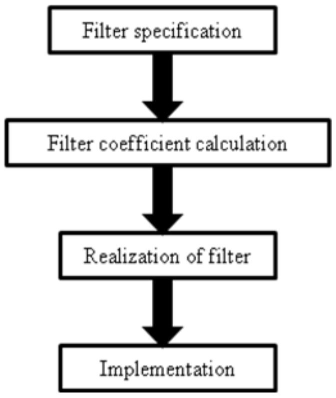


Figure 1.2 Design flow of FIR filters

The design of a digital filter involves following five steps as indicated in figure 1.2.

FILTER SPECIFICATION: This may include stating the type of filter, for example low pass filter, the desired amplitude and/or phase responses and the tolerances, the sampling frequency, the word length of the input data.

FILTER COEFFICIENT CALCULATION: The coefficient of a transfer function H (z) is determined in

this step, which will satisfy the given specification. The choice of coefficient calculation method will be influenced by several factors. The most important of which are the critical requirements i.e. specification.

REALIZATION: This involves converting the transfer function into a suitable filter network or structure.

IMPLEMENTATION: This involves producing the software code and/or hardware and performing the actual filtering.

## V.STRUCTURE OF 8-TAP FIR FILTER:

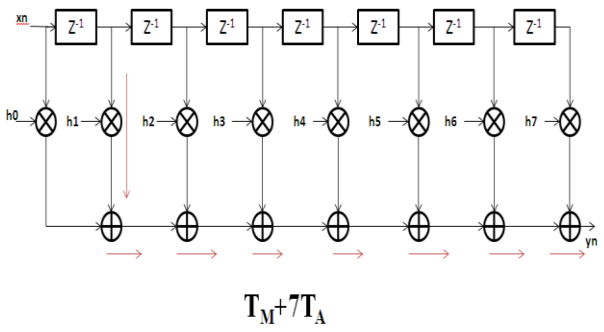


Figure 1.3 structure of 8-tap FIR filter

## VI.PIPELINING:

In pipelining, any operation along the critical path is broken into smaller, quicker operation with registers between levels, so as to get a smaller critical path delay. As a result there will be an increase in operating frequency which leads to higher throughput. Pipelining transformation leads to a reduction in the critical path, which can be exploited to either increase the clock speed or sample speed or to reduce power consumption at same speed. Moreover some real time application requires faster input rates where direct structures cannot be used. In such cases we can use pipelining by introducing latches along the critical data path.

The critical path or the minimum time required for processing a new sample is limited by 1 multiply time and 2 add times, i.e., if $T_M$ is the time taken for multiplication and $T_A$ is time needed for addition operation then the "sample period" ($T_{sample}$) is given by,

$$T_{sample} \geq T_M+2T_A.$$

Therefore, the sampling frequency ($s_{ample}$) (also referred to as the throughput or the iteration rate) is given by

$$f_{sample} = \frac{1}{T_M + 2T_A}$$

The direct form structure shown in the figure can only be used when the above equations are satisfied. But if some real-time applications demands a faster input rate (sample rate), then this structure cannot be use. In that case, the effective critical path can be reduced by using either pipelining or parallel processing.

The following points give the significances of pipelining;

- The speed of architecture (or the clock period) is limited by the longest path between any 2 latches or between an input and a latch or between a latch and an output or between the input and the output.
- This longest path or the “critical path” can be reduced by suitably placing the pipelining latches in the architecture.

## VII.STRUCTURE OF 8-TAP PIPELINED FIR FILTER:

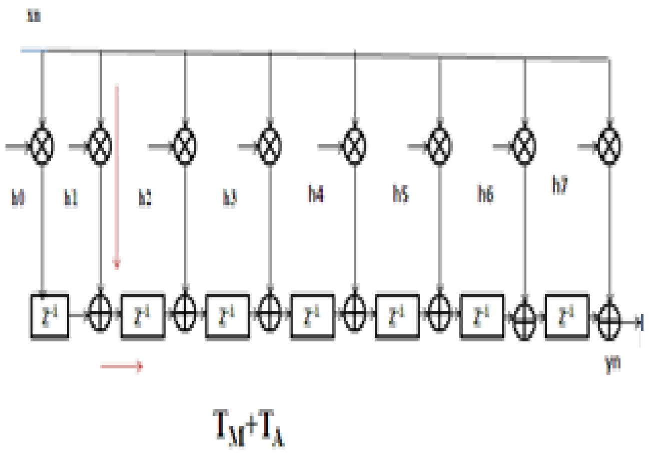


Figure 1.4: Structure of pipelined FIR filter

Hence from the figures 1.3 and 1.4, its clear that by introducing additional latches the critical path is reduced from TM+7TA to TM+TA.

## VIII.BOOTH MULTIPLIER:

Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier is among the fastest. However, they suffer from a bad regularity. Hence, when regularity, high performance and low power are primary concerns, Booth multipliers tend to be the primary choice. Booth multipliers allow the operation on signed operands in 2’s-complements. 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.

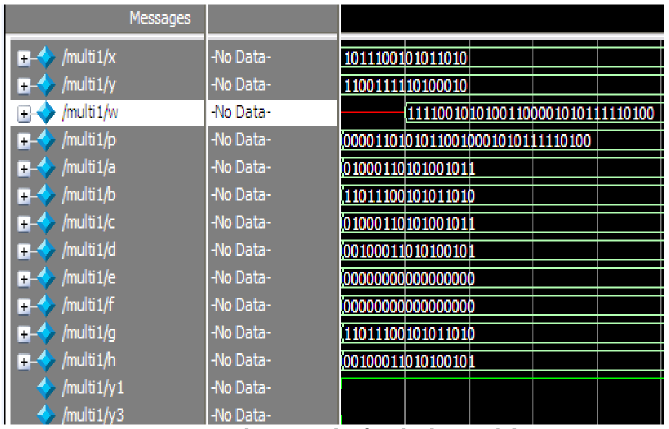
The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In this algorithm each partial product line operates on 2 bits at a time, thereby reducing the total number of the partial products. This is particularly true for operands using 16 bits or more.

BOOTH MULTIPLICATION TABLE:

| X(i) | X(i-1) | X(i-2) | y |
|---|---|---|---|
| 0 | 0 | 0 | +0 |
| 0 | 0 | 1 | +y |
| 0 | 1 | 0 | +y |
| 0 | 1 | 1 | +2y |
| 1 | 0 | 0 | -2y |
| 1 | 0 | 1 | -y |
| 1 | 1 | 0 | -y |
| 1 | 1 | 1 | +0 |

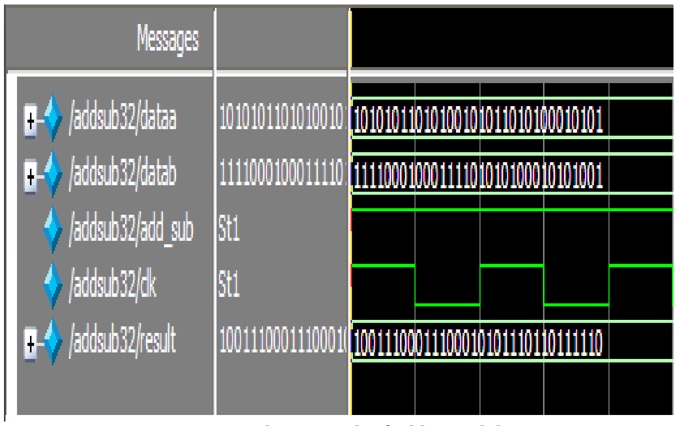
Table 1.1: Radix 4 modified Booth algorithm for odd values of i

## IX.SIMULATION RESULTS:



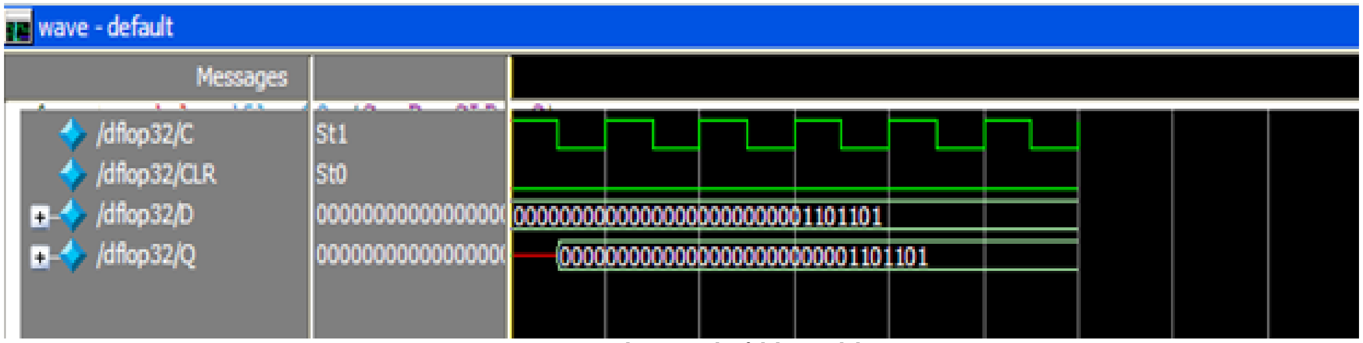
```
Messages
/multi1/x    -No Data-  1011100101011010
/multi1/y    -No Data-  1100111110100010
/multi1/w    -No Data-  11110010101001100001010111110100
/multi1/p    -No Data-  00001101010110010001010111110100
/multi1/a    -No Data-  010001101010010 11
/multi1/b    -No Data-  1101110010101101 0
/multi1/c    -No Data-  010001101010010 11
/multi1/d    -No Data-  001000110101001 01
/multi1/e    -No Data-  000000000000000000
/multi1/f    -No Data-  000000000000000000
/multi1/g    -No Data-  1101110010101101 0
/multi1/h    -No Data-  001000110101001 01
/multi1/y1   -No Data-
/multi1/y3   -No Data-
```

Figure 1.5: Simulation result of multiplier module



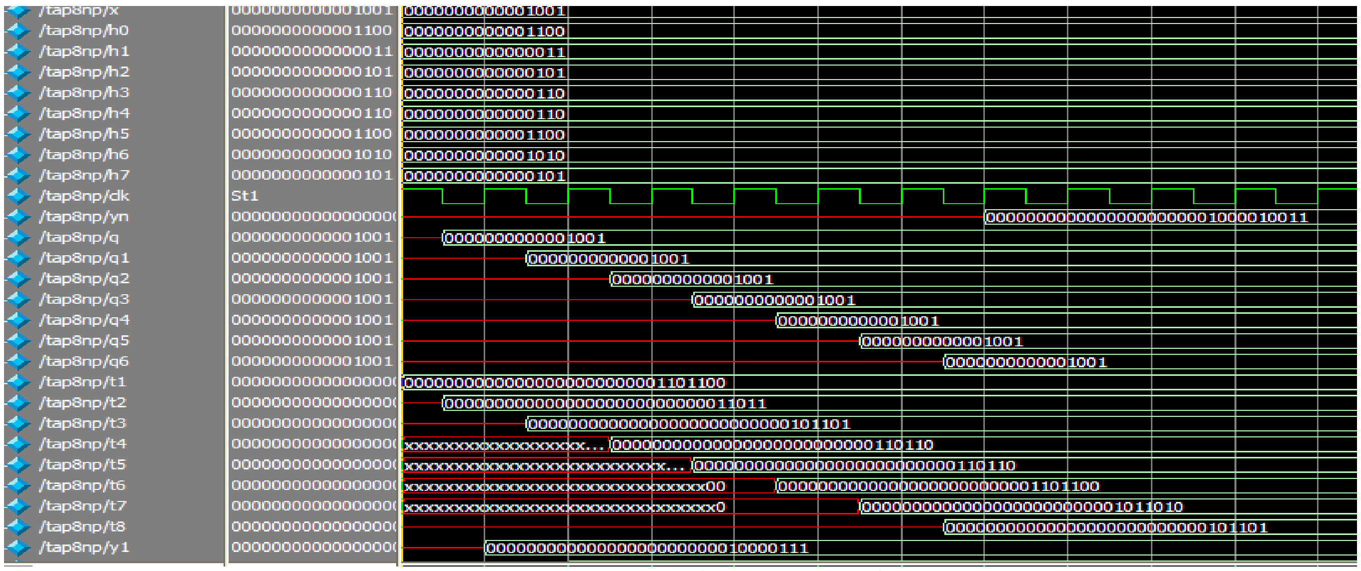
```
Messages
/addsub32/dataa    10101011010100101  10101011010100101011010100010101
/addsub32/datab    11110001000111101  11110001000111101010100010101001
/addsub32/add_sub  St1
/addsub32/clk      St1
/addsub32/result   10011100011100010  10011100011100010101110110111110
```

Figure 1.6: Simulation result of adder module



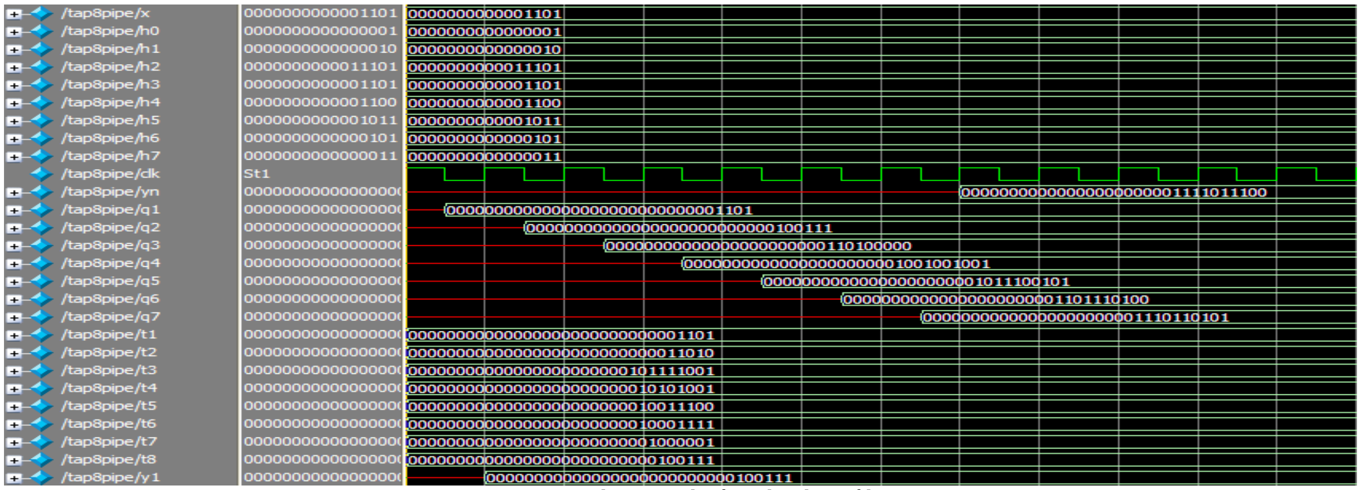
```
wave - default
Messages
/dflop32/C      St1
/dflop32/CLR    St0
/dflop32/D      00000000000000000(   00000000000000000000000000001101101
/dflop32/Q      00000000000000000(   00000000000000000000000000001101101
```

Figure 1.7: Simulation result of delay module



```
/tap8np/x     0000000000001001     0000000000001001
/tap8np/h0    0000000000001100     0000000000001100
/tap8np/h1    000000000000011      000000000000011
/tap8np/h2    000000000000101      000000000000101
/tap8np/h3    000000000000110      000000000000110
/tap8np/h4    000000000000110      000000000000110
/tap8np/h5    0000000000001100     0000000000001100
/tap8np/h6    0000000000001010     0000000000001010
/tap8np/h7    000000000000101      000000000000101
/tap8np/dk    St1
/tap8np/yn    00000000000000000(   00000000000000000000000010000100 11
/tap8np/q     0000000000001001     0000000000001001
/tap8np/q1    0000000000001001     0000000000001001
/tap8np/q2    0000000000001001     0000000000001001
/tap8np/q3    0000000000001001     0000000000001001
/tap8np/q4    0000000000001001     0000000000001001
/tap8np/q5    0000000000001001     0000000000001001
/tap8np/q6    0000000000001001     0000000000001001
/tap8np/t1    00000000000000000(   0000000000000000000000000001101100
/tap8np/t2    00000000000000000(   00000000000000000000000000000011011
/tap8np/t3    00000000000000000(   000000000000000000000000000101101
/tap8np/t4    00000000000000000(   xxxxxxxxxxxxxxxxxxx...  0000000000000000000000000000110110
/tap8np/t5    00000000000000000(   xxxxxxxxxxxxxxxxxxxxxxxxxx...  0000000000000000000000000000110110
/tap8np/t6    00000000000000000(   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00  0000000000000000000000000001101100
/tap8np/t7    00000000000000000(   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0  0000000000000000000000000001011010
/tap8np/t8    00000000000000000(   0000000000000000000000000000101101
/tap8np/y1    00000000000000000(   0000000000000000000000000010000111
```

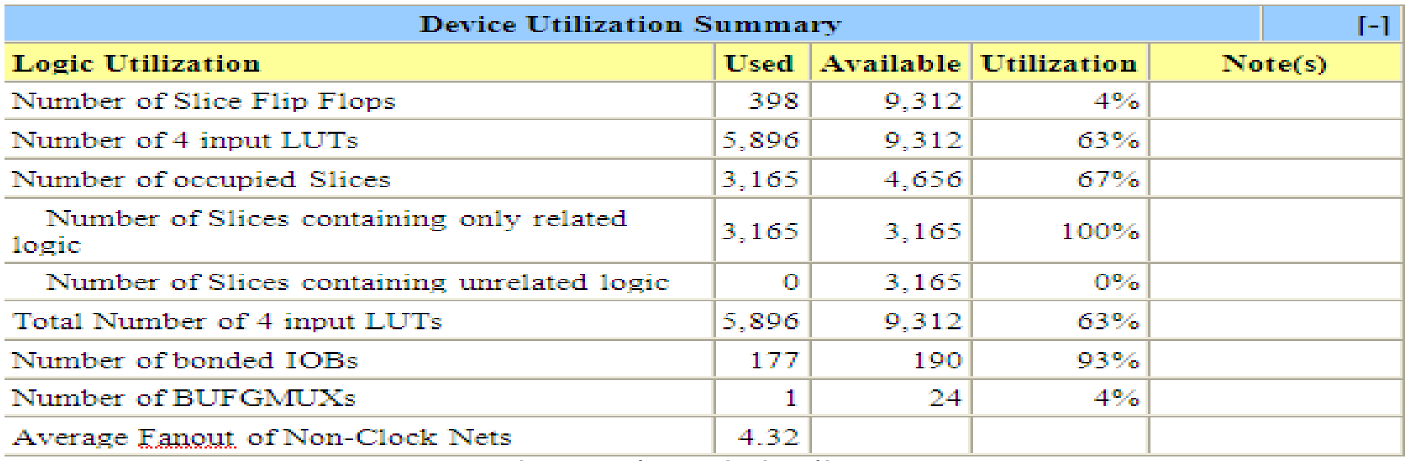
Figure 1.8: Simulation result of Non pipelined FIR filter



```
/tap8pipe/x     0000000000001101     0000000000001101
/tap8pipe/h0    0000000000000001     0000000000000001
/tap8pipe/h1    0000000000000010     0000000000000010
/tap8pipe/h2    0000000000011101     0000000000011101
/tap8pipe/h3    0000000000001101     0000000000001101
/tap8pipe/h4    0000000000001100     0000000000001100
/tap8pipe/h5    0000000000001011     0000000000001011
/tap8pipe/h6    000000000000101      000000000000101
/tap8pipe/h7    000000000000011      000000000000011
/tap8pipe/dk    St1
/tap8pipe/yn    00000000000000000(   0000000000000000000001111011100
/tap8pipe/q1    00000000000000000(   0000000000000000000000000000001101
/tap8pipe/q2    00000000000000000(   000000000000000000000000000100111
/tap8pipe/q3    00000000000000000(   00000000000000000000000110100000
/tap8pipe/q4    00000000000000000(   000000000000000000000100100100 1
/tap8pipe/q5    00000000000000000(   0000000000000000000001011100101
/tap8pipe/q6    00000000000000000(   0000000000000000000001101110100
/tap8pipe/q7    00000000000000000(   0000000000000000000001110110101
/tap8pipe/t1    00000000000000000(   00000000000000000000000000000001101
/tap8pipe/t2    00000000000000000(   00000000000000000000000000000011010
/tap8pipe/t3    00000000000000000(   00000000000000000000000101111001
/tap8pipe/t4    00000000000000000(   000000000000000000000000010101001
/tap8pipe/t5    00000000000000000(   000000000000000000000000010011100
/tap8pipe/t6    00000000000000000(   000000000000000000000000010001111
/tap8pipe/t7    00000000000000000(   0000000000000000000000000001000001
/tap8pipe/t8    00000000000000000(   0000000000000000000000000000100111
/tap8pipe/y1    00000000000000000(   0000000000000000000000000000100111
```

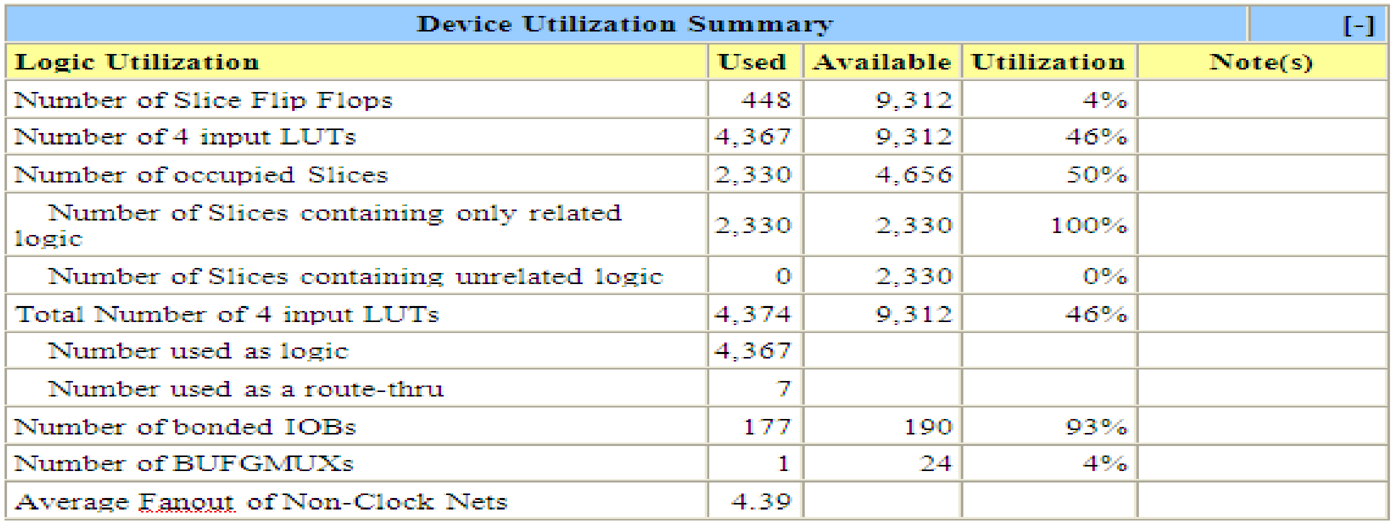
Figure 1.9: Simulation result of pipelined FIR filter

## X.SYNTHESIS REPORTS:



| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 398 | 9,312 | 4% | |
| Number of 4 input LUTs | 5,896 | 9,312 | 63% | |
| Number of occupied Slices | 3,165 | 4,656 | 67% | |
| Number of Slices containing only related logic | 3,165 | 3,165 | 100% | |
| Number of Slices containing unrelated logic | 0 | 3,165 | 0% | |
| Total Number of 4 input LUTs | 5,896 | 9,312 | 63% | |
| Number of bonded IOBs | 177 | 190 | 93% | |
| Number of BUFGMUXs | 1 | 24 | 4% | |
| Average Fanout of Non-Clock Nets | 4.32 | | | |

Figure 1.10: Synthesis report of Non pipelined FIR filter on Spartan-3E



| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 448 | 9,312 | 4% | |
| Number of 4 input LUTs | 4,367 | 9,312 | 46% | |
| Number of occupied Slices | 2,330 | 4,656 | 50% | |
| Number of Slices containing only related logic | 2,330 | 2,330 | 100% | |
| Number of Slices containing unrelated logic | 0 | 2,330 | 0% | |
| Total Number of 4 input LUTs | 4,374 | 9,312 | 46% | |
| Number used as logic | 4,367 | | | |
| Number used as a route-thru | 7 | | | |
| Number of bonded IOBs | 177 | 190 | 93% | |
| Number of BUFGMUXs | 1 | 24 | 4% | |
| Average Fanout of Non-Clock Nets | 4.39 | | | |

Figure 1.11: Synthesis report of pipelined FIR filter on Spartan-3E

## XI.PIPELINING (Vs) NON PIPELINING:

| | **Non pipelining** | **Pipelining** |
|---|---|---|
| **LUTS** | 5896 | 4367 |
| **Minimum period** | 44.069 ns | 10.192 ns |
| **Maximum frequency** | 22.692MHZ | 98.116 MHZ |

Table 1.2: Comparison of Non pipelined and Pipelined FIR filter

## XII.RESOURCE UTILIZATION IN FPGA:

FPGA editor view in Xilinx describes about the device utilization for the design. The architectural view of FPGA shown in Figure 1.12 and 1.13 provides the details of amount of resources occupied in the device for Non-pipelined and pipelined FIR filter. By comparing both the figures, it is evident that the device utilized for pipelined FIR filters is much reduced when compared with that of the non-pipelined FIR filters.

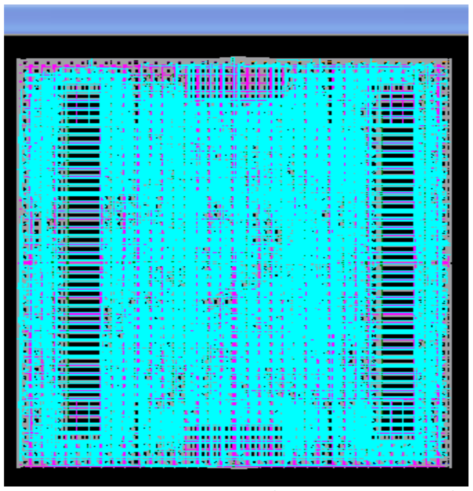


Figure 1.12 FPGA of Non pipelined FIR filter

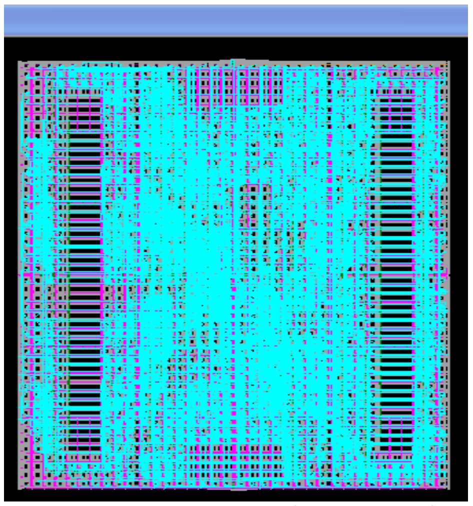


Figure 1.13 FPGA of Pipelined FIR filter

## XIII.CONCLUSION:

This paper mainly describes design and implementation of pipelined FIR filters using FPGA. As the proposed design of this paper is to enhance the speed of the FIR filter, by introducing pipelining operating frequency of the FIR filter is increased approximately 4 times compared to non pipelined filter. Hence the speed and device utilization (LUTs) of the system is improved.

## XIV.FUTURE WORK:

The design and implementation of 8-tap FIR filter is discussed in this paper. Further the design can be extended to n-tap filter to be used in DSP applications.

## XV.REFERNCES:

[1] Jiang Xiaoyan and Bao Yujun , "FIR Filter Design Based on FPGA", 2010 International Conference on Computer Application and System Modeling (ICCASM 2010).

[2] Jieshan Lin and Huang Shizhen, "An Design of the 16-order FIR Digital Filter Based on FPGA", 2009, International Conference on Information Science and Engineering (ICISE2009).

[3] Zhang Chi and Guo Li Li, Design of FIR filter with Matlab and running on FPGA, Applied Science and Technology. vol. 33, no. 6, pp.83, Jun. 2006

[4]Sami Khorbotly, Joan E. Carletta, Robert J. Veillette "A Methodology for Implementing Pipelined Fixed-Point Infinite Impulse Response Filters" 41st Southeastern Symposium on System Theory University of Tennessee Space Institute Tullahoma, TN, USA, March 15-17, 2009

[5] A. Shaw and M. Ahmed, "Pipelined recursive digital filters: a general look-ahead scheme and optimal approximation," *IEEE Trans. On Circuits and Systems II: Analog & Digital Signal Processing*, vol. 46, no. 11, pp. 1415– 1420, Nov. 1999.

[6] Chao-Huang Wei, Hsiang-Chieh Hsiao, Su-Wei Tsai "FPGA Implementation of FIR Filter with smallest Processor" IEEE, 2005.

[7] Mohamed Al Mahdi Eshtawie, and Masuri Bin Othman "An Algorithm Proposed for FIR Filter Coefficients Representation". International Journal of Applied Mathematics and Computer Sciences 4;1 2008

[8]Keshab K.Parhi "VLSI digital signal processing systems" Jhon Wiley and Sons publishing company, New Delhi, 2008.