INTERNATIONAL DEVELOPMENT PLANNING REVIEW
ISSN:1474-6743 | E-ISSN:1478-3401

IMPLEMENTATION OF EFFICIENT FIR FILTER

Ranjeeta Yadav
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Surekha Ghangas
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Krishna Kumar Verma
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Divyanshu Joshi
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Harsit Yadav
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Anmol Dev
ABES Engineering College, Ghaziabad, Uttar Pradesh, India

Abstract. In this paper, Design and implementation of FIR Filter done by using Han-Carlson Adder. To design Filter different blocks required which are Adders, Multipliers, and Delay elements. FIR Filters are easy to design and are less power consuming. Here for designing of multiplier Han-Carlson Adder used and the filter is designed using proposed multiplier and delay element(D-flip flop).[1] The reason of selecting Han-Carlson adder is their performance, it is one of the fastest adders as compare to other adders on other hand its circuit is complex but it has given efficient performance as compare to others adder. Our main focus is on the parameter of adders, Multiplier and Fir Filters such as Power, LUT utilization, area and delay to make an efficient FIR Filters using Vedic mathematics.[2] The proposed FIR Filter has designed and simulate with help Verilog HDL codes and their circuit diagram and Simulation result of the filter has synthesized on Xilinx Tool and Questa Sim.
Keywords: MAC Unit, RCA**,** CLA, Kogge stone, Han-carlson Adder.

Introduction. In this generation, Filters are in huge demand of electronic world. But VLSI industry is facing challenges while designing. To manage Low power consumption along with minimum delay in the circuit is one of the toughest tasks. Here, the main fo us is on various parameter which directly affect our Filters.[3] For designing of proposed Filter, Han-Carlson Adder chosen as best adder after a comparison done as shown in table 1. The main reason of selection of this adder is their parameter like (minimum delay) as shown in table 1. This is one of

the fastest Adder which provide minimum delay in the circuit. Digital multipliers are mostly used in designing new gadgets, and have many other applications in digital signal processing.[4] In this paper, the multiplier and filter both are design by the use of Han- Carlson Adder. Before selecting this Adders, a comparison made between different adders in order to get the best and efficient adder which is Han-Carlson adder.[5]

1. Mac Unit

MAC unit play a crucial role in the computing devices, especially in signal processor. MAC is used to perform different task like multiplication and accumulation.[6] At fundamental level MAC unit consist of adder, multiplier and accumulation.

2. Comparison of different adders on the bases of LUT and Delay

2.1 8-Bit Ripple Carry Adder

In this adder every carry bit gets undulated into the accompanying stage, it suggests that every full adder is the input-carry for the predominant next hugest full adder, and so forth.[7] Disregarding the way that RCA consume less region however delay in the circuit is (high defer time) this lead to high power use in juxtaposition with various adders. Conditions: S = input1 xor input2 xor input-carry & Carry-out = input1 * input2 + input2 * input-carry + inptu1 * inpt-carry (Here inptu1 and input2 are data sources and input-carry is carry input). For this Adder (LUTs, Delay) is (12,2.57ns).[8] In below figure 2.1(a) Behavioural Modelling and figure 2.1(b) Synthesis Report of 8-Bit Ripple Carry Adder shown:

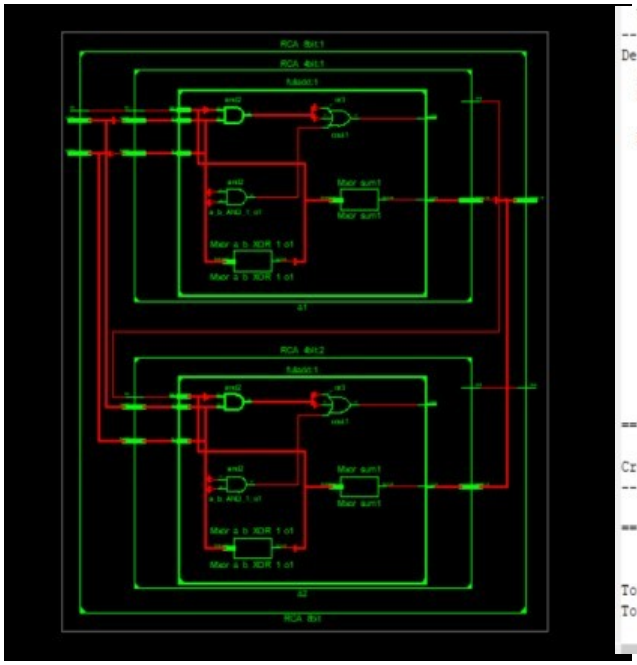

Figure. 2.1 (a)

```
Total number of paths / destination ports: 97 / 9
-------------------------------------------------------------------------
Delay:               2.571ns (Levels of Logic = 6)
  Source:            a<1> (PAD)
  Destination:       sum<6> (PAD)

  Data Path: a<1> to sum<6>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2   0.001   0.687  a_1_IBUF (a_1_IBUF)
     LUT5:I0->O            3   0.097   0.305  a1/a2/cout1 (a1/c<1>)
     LUT5:I4->O            3   0.097   0.389  a1/a4/cout1 (w)
     LUT5:I3->O            3   0.097   0.521  a2/a2/cout1 (a2/c<1>)
     LUT3:I0->O            1   0.097   0.279  a2/a3/Mxor_sum_xo<0>1 (sum_6_OBUF)
     OBUF:I->O                 0.000          sum_6_OBUF (sum<6>)
    ----------------------------------------
    Total                      2.571ns (0.389ns logic, 2.182ns route)
                                       (15.1% logic, 84.9% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

=========================================================================


Total REAL time to Xst completion: 10.00 secs
Total CPU time to Xst completion: 9.64 secs
```

Figure. 2.1 (b)

2.2 8-Bit Carry Lookahead Adder

This is one of the most efficient and quickest Adders in gaining the resulting output; its speed is improved by diminishing the proportion of time expected to choose the carry bits.[9] As its carries in the moderate stages will be resolved to this point using carry produce and carry spread paying little mind to input carry for this Adder (LUTs, Delay) is (16, 2.438ns). In below figure 2.2(a) Behavioral Modeling and figure 2.2(b) Synthesis Report of 8-Bit Carry Lookahead Adder shown:

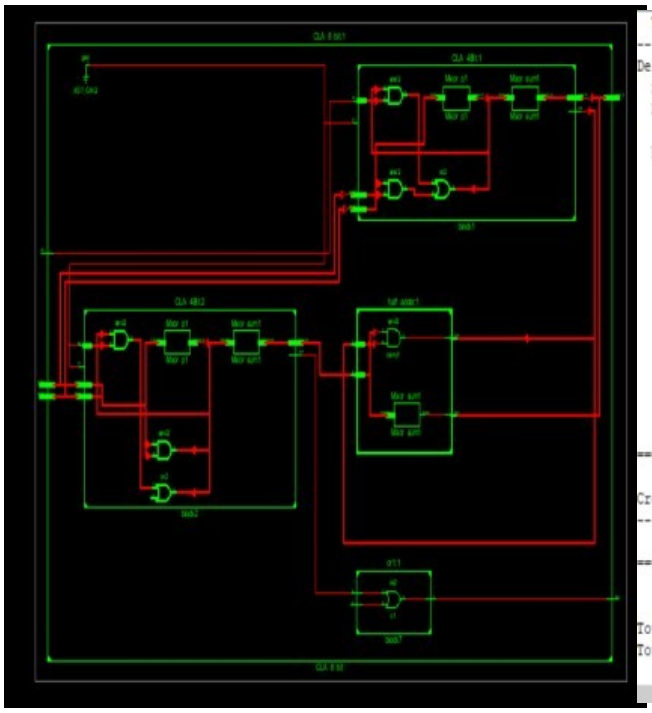

Figure. 2.2 (a)

```
Total number of paths / destination ports: 129 / 9
-------------------------------------------------------------------------
Delay:               2.438ns (Levels of Logic = 6)
  Source:            a<1> (PAD)
  Destination:       sum<7> (PAD)

  Data Path: a<1> to sum<7>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2   0.001   0.687  a_1_IBUF (a_1_IBUF)
     LUT5:I0->O            3   0.097   0.389  block1/c<2>1 (block1/c<2>)
     LUT3:I1->O            4   0.097   0.393  block1/c<3>1 (block1/c<3>)
     LUT6:I4->O            2   0.097   0.299  block5/carry1 (w<8>)
     LUT4:I3->O            1   0.097   0.279  block6/Mxor_sum_xo<0>1 (sum_7_OBUF)
     OBUF:I->O                 0.000          sum_7_OBUF (sum<7>)
    ----------------------------------------
    Total                      2.438ns (0.389ns logic, 2.049ns route)
                                       (16.0% logic, 84.0% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

=========================================================================


Total REAL time to Xst completion: 9.00 secs
Total CPU time to Xst completion: 8.20 secs
```

Figure. 2.2(b)

2.3 8-Bit Kogge Stone Adder

This adder is also known as parallel prefix adder, For higher speed this adder is apt appropriate and suitable. At same instant it pairs the time delay and to create the carry signals and here after significantly help in doing quick activities. The number of computational nodes is given by {**n[log$_2$n]- n+1**} and the optimal depth given by **log$_2$n.**[10] For this Adder (LUTs, Delay) is (15, 2.143ns). In below figure 2.3(a) RTL view and figure 2.3(b) Synthesis Report of 8-Bit Carry Lookahead Adder shown:

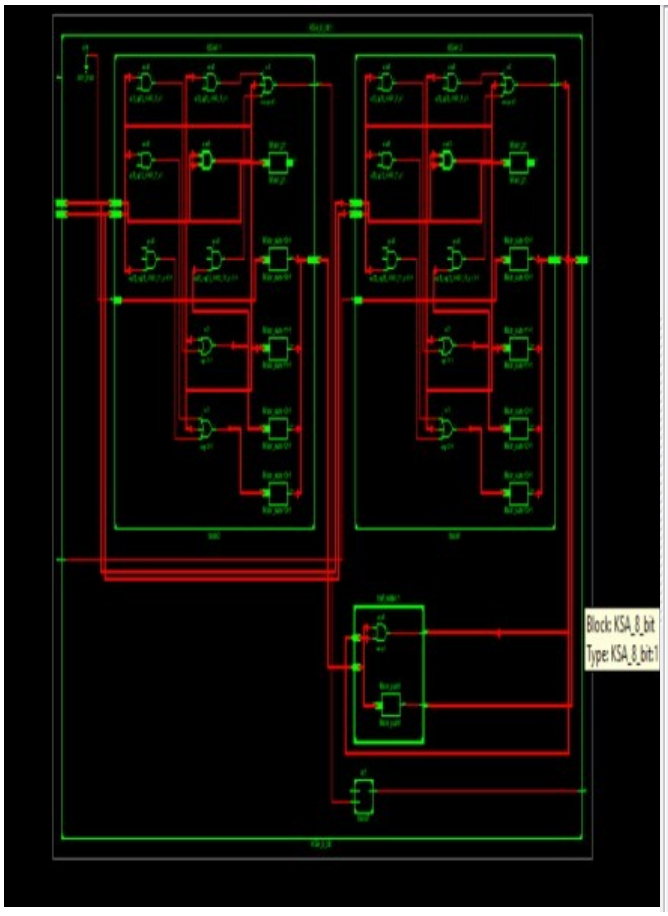

Figure. 2.3 (a)

```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 121 / 9
--------------------------------------------------------------------------------
Delay:               2.143ns (Levels of Logic = 5)
  Source:            a<4> (PAD)
  Destination:       sum<7> (PAD)

  Data Path: a<4> to sum<7>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             6   0.001   0.579  a_4_IBUF (a_4_IBUF)
     LUT4:I0->O            3   0.097   0.693  block2/Mxor_sum<1>_xo<0>1 (w<2>)
     LUT6:I1->O            2   0.097   0.299  block5/carry1 (w<8>)
     LUT4:I3->O            1   0.097   0.279  block6/Mxor_sum_xo<0>1 (sum_7_OBUF)
     OBUF:I->O                 0.000          sum_7_OBUF (sum<7>)
    ----------------------------------------
    Total                      2.143ns (0.292ns logic, 1.851ns route)
                                       (13.6% logic, 86.4% route)

=========================================================================

Cross Clock Domains Report:
--------------------------


=========================================================================


Total REAL time to Xst completion: 7.00 secs
Total CPU time to Xst completion: 7.87 secs
```

Figure. 2.3 (b)

2.4 8-Bit Han-Carlson Adder

This adder mostly used for faster speed and for lesser delay. This adder is a parallel prefix adder. It consists of different node and stages where input nodes are represented by White node. The number of computational nodes is given by {**n/2 [log$_2$n]**} and the optimal depth given by [**log$_2$n+1].**[11] Here for 8-bit (LUTs, Delay) is (15, 2.143ns). In below figure 2.4(a) Behavioral modeling and figure 2.4(b) Synthesis Report of 8-Bit Han-Carlson Adder shown:

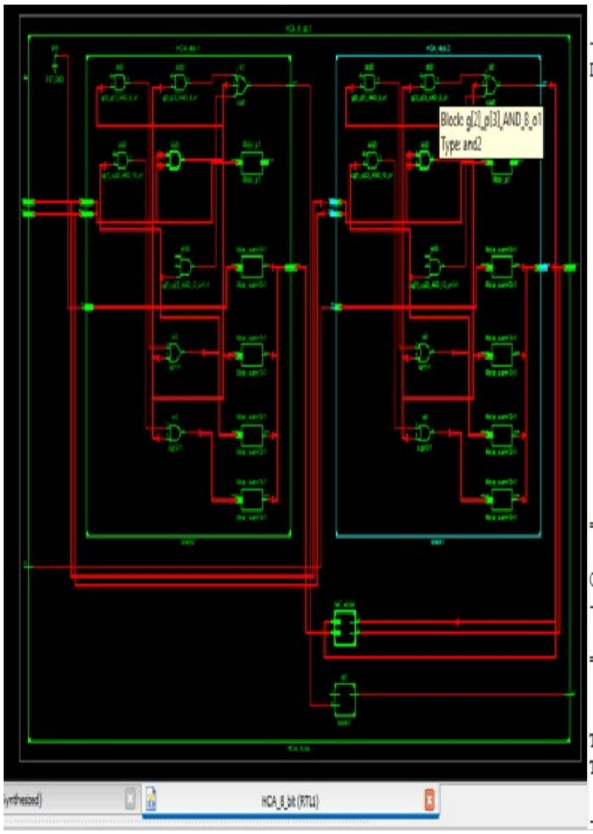

Fig. 2.4(a)

```
Total number of paths / destination ports: 121 / 9
--------------------------------------------------------------------------------
Delay:               2.143ns (Levels of Logic = 5)
  Source:            a<4> (PAD)
  Destination:       sum<7> (PAD)

  Data Path: a<4> to sum<7>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             6   0.001   0.579  a_4_IBUF (a_4_IBUF)
     LUT4:I0->O            3   0.097   0.693  block2/Mxor_sum<1>_xo<0>1 (w<2>)
     LUT6:I1->O            2   0.097   0.299  block5/carry1 (w<8>)
     LUT4:I3->O            1   0.097   0.279  block6/Mxor_sum_xo<0>1 (sum_7_OBUF)
     OBUF:I->O                 0.000          sum_7_OBUF (sum<7>)
    ----------------------------------------
    Total                      2.143ns (0.292ns logic, 1.851ns route)
                                       (13.6% logic, 86.4% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

=========================================================================


Total REAL time to Xst completion: 8.00 secs
Total CPU time to Xst completion: 7.85 secs

-->
```

Fig. 2.4(b)

From this table, KSA and HCA both are giving same delay and same LUT but in KSA number of Gray and Black cell is more as compare to HCA, these cells lead to more power consumption hence it is clear Han-Carlson adder is efficient adder and having minimum delay which is 2.143ns.

So, for designing of Multiplier and Filter, Han-Carlson adder is apt appropriate and suitable for the designing of Efficient Filter.

3. Design and implementation 8-bit Vedic multiplier

For 8x8 multiplier, firstly need to design 2x2 multiplier, then 4x4 multiplier and then 8x8 multiplier. To start designing of 2x2 multiplier need to know how two multiply two 2-bit numbers. Need to Convert the below fig 3(a) to a hardware equivalent as shown in fig 3(a).[12] here 3-AND Gates will work as 2-bit multipliers and 2-HA (Half Adder) to append the resultant to find out final result. Whole hardware information of this multiplier is given in such a way 'A' and B' are two numbers this two will be multiplied to get the product as 'q'. From this design verilog code easily written with the help of AND gates and HA (half adders).[13] Now need to made the design little simpler and more modular, firstly need to write the verilog code for Half

Adder after that instantiate this to get the final output as required product. From references of Fig 3(a) 2x2 multiplier designed.

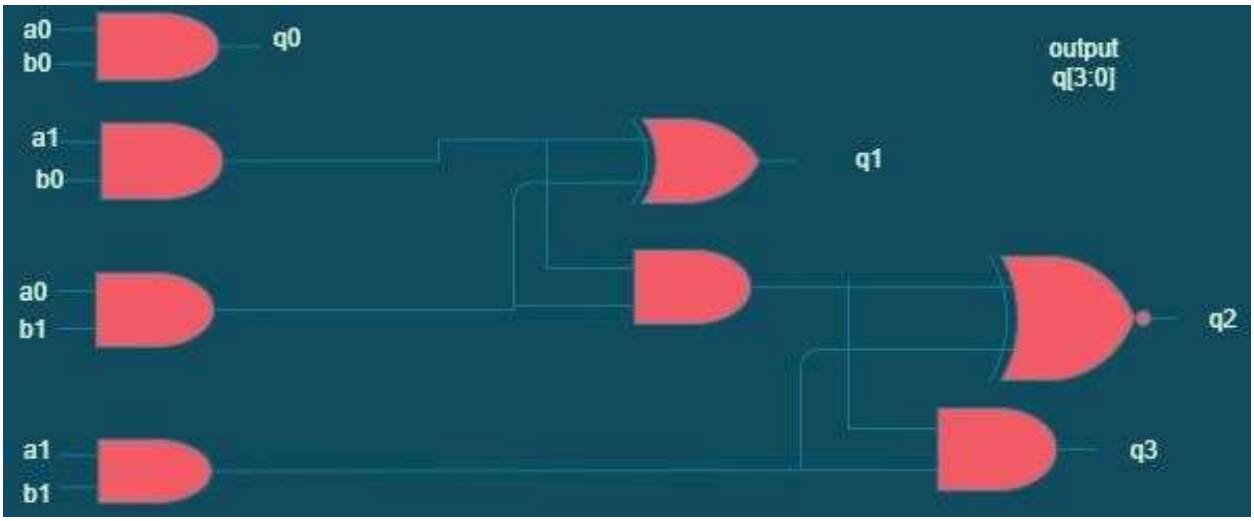

Figure. 3(a)

Similarly, for **4x4 multiplier**, by the use four such 2x2-multipliers and 3-adders which already implemented above, the main point is to design this multipliers as shown below in the figure 3(c).[14] Firstly, we need to design adders of 4-bit and 6-bit using verilog code and then properly instantiating with the help of 2x2 multiplier. Here Han-Carlson Adder is used to implement the above as shown in fig 3(c). With the help of Wallace tree architecture, level of addition is reduce from three to just two stages.[15] From references of Fig 3(c) 4x4 multiplier designed.

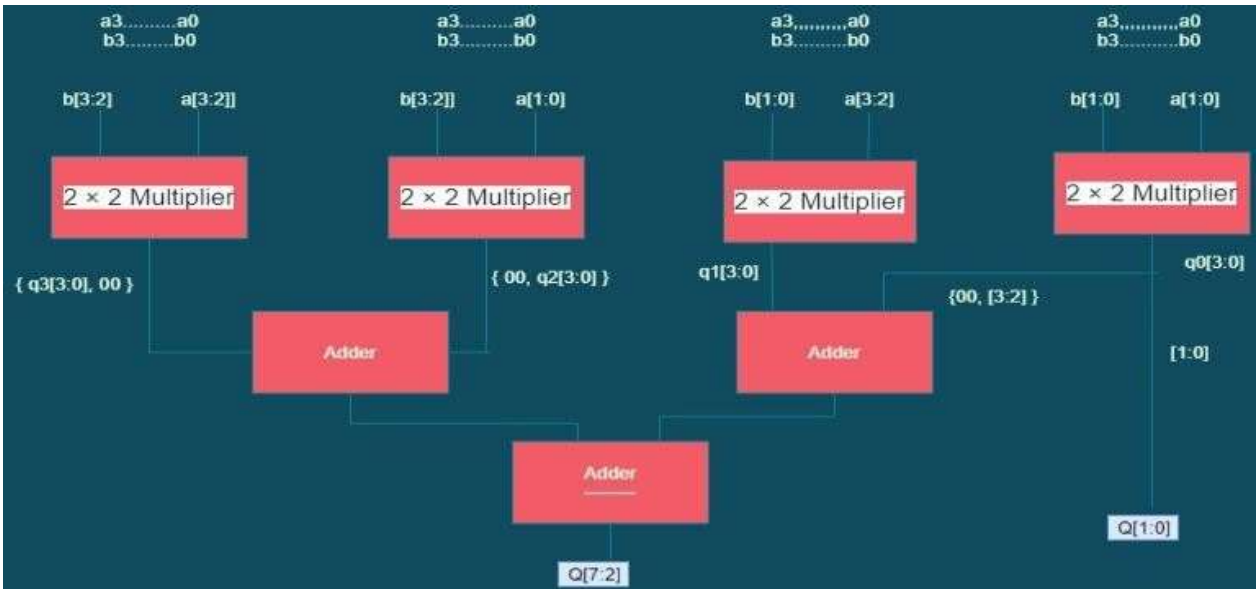

Fig. 3(c)

Again, the same procedure is followed to design **8x8-multilpier,** here four such 4x4 multiplier are used to design 8x8 multiplier.[16] For performing addition 8-bit and 12-bit adder required

with module and connections which is shown in the below figure. From references of Fig 3(d) 8x8 multiplier designed.

This design simplifies to code this in verilog easily. From the code RTL design and waveform of 8x8 multiplier generated which is shown in below figure 3(e). (for code need to email personally)

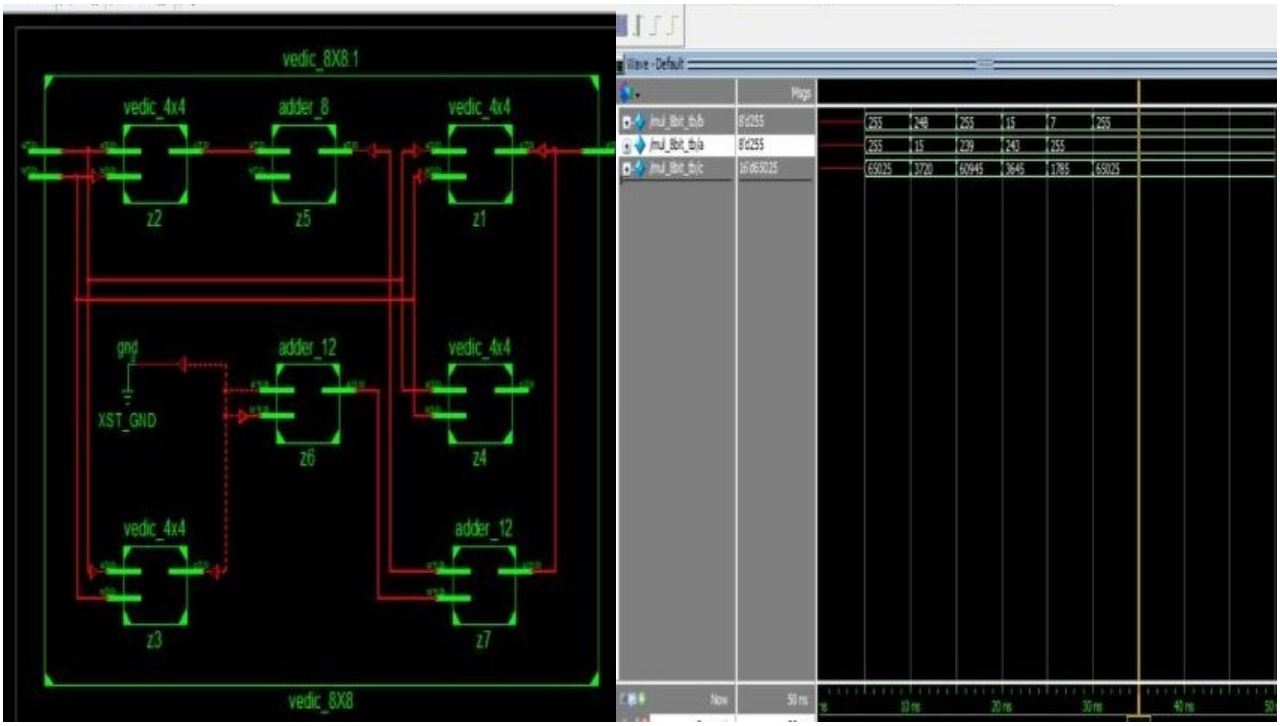

Figure. 3(e)

4. Implementation of FIR Filter

Filter design done by using Han-Carlson Adder. To design Filter different blocks required which are Adders, Multipliers, and Delay elements. FIR Filters are easy to design and are less power consuming.[17] Here multiplier will be designed through the help of Han-Carlson Adder and the Fir filter will be designed using proposed multiplier and (d-flip flop) as delay element.

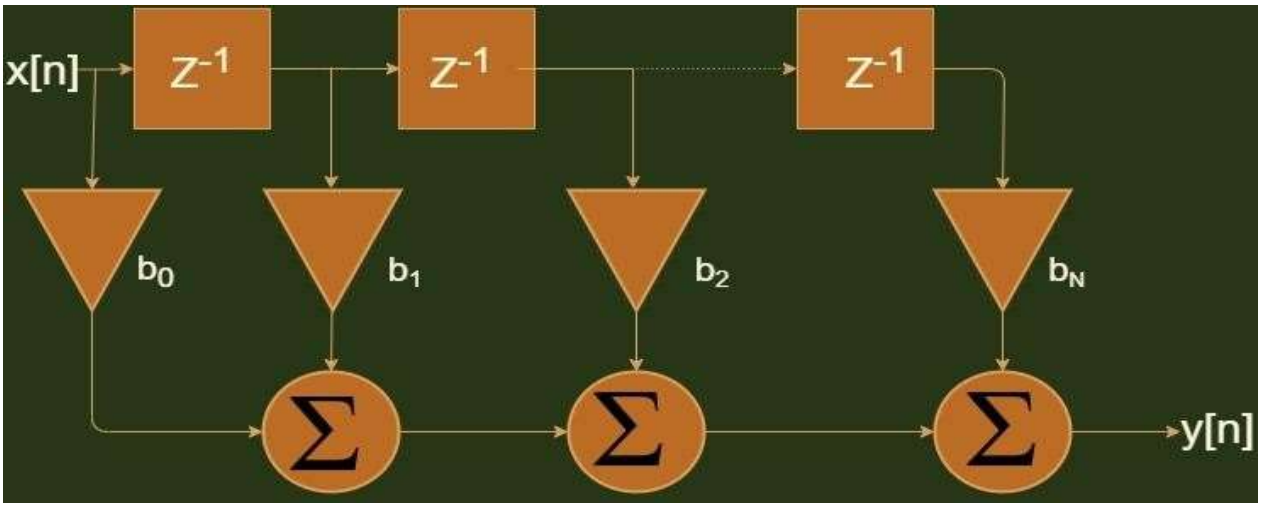

Figure.4(a)

This project is to implement a low pass FIR filter in Verilog. In this project, the regular implementation of fir filter is done as we can see in the following fig 4 (a). The implementation of FIR Filter done by using D-flip flop, Vedic multiplier and Han-Carlson adder. Each block consists of one-multiplier, one-N-bit register, and one-adder. Verilog is used to implement and design Han-Carlson adder, and by instantiating these adders, Vedic multiplier is designed. and these adders and multipliers are used in FIR filter. From the reference of Fig. 4(a) verilog coding done and the Behavioral modeling of Fir Filter we can see in Fig. 4(b).

RTL view of FIR Filter:

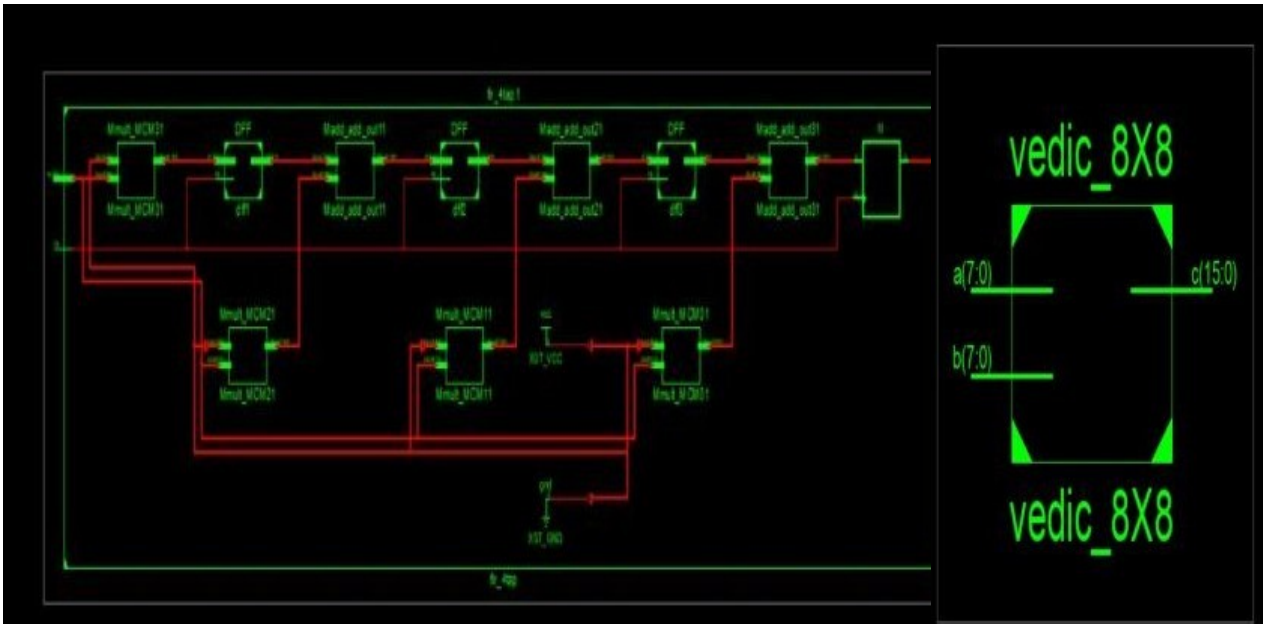

Figure. 4(b)

RTL view of FIR Filter as we can see in below figure 4(c)

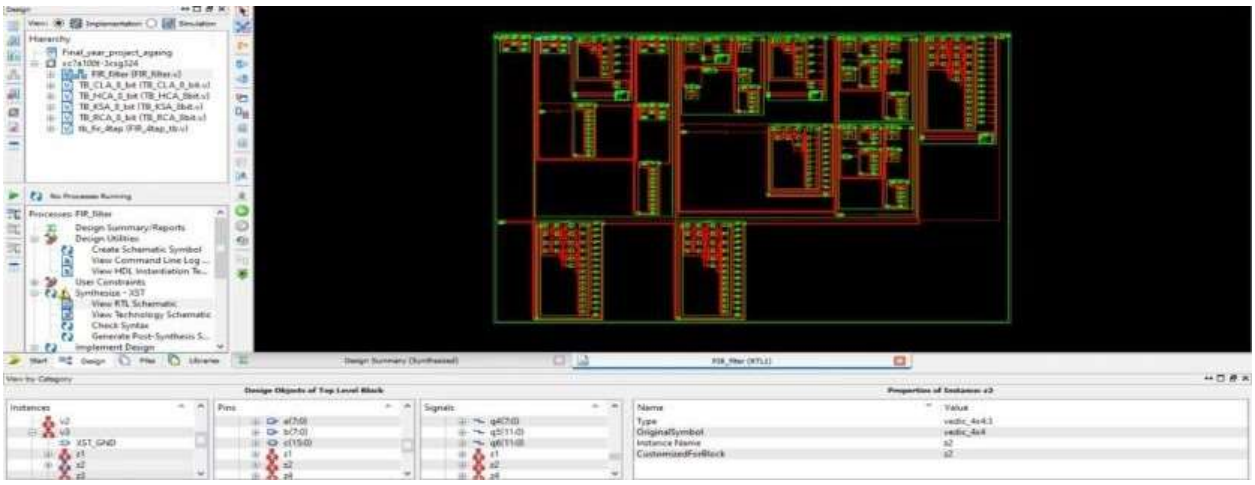

Figure. 4(c)

Simulation result of FIR Filter as we can see in figure 4 (d).

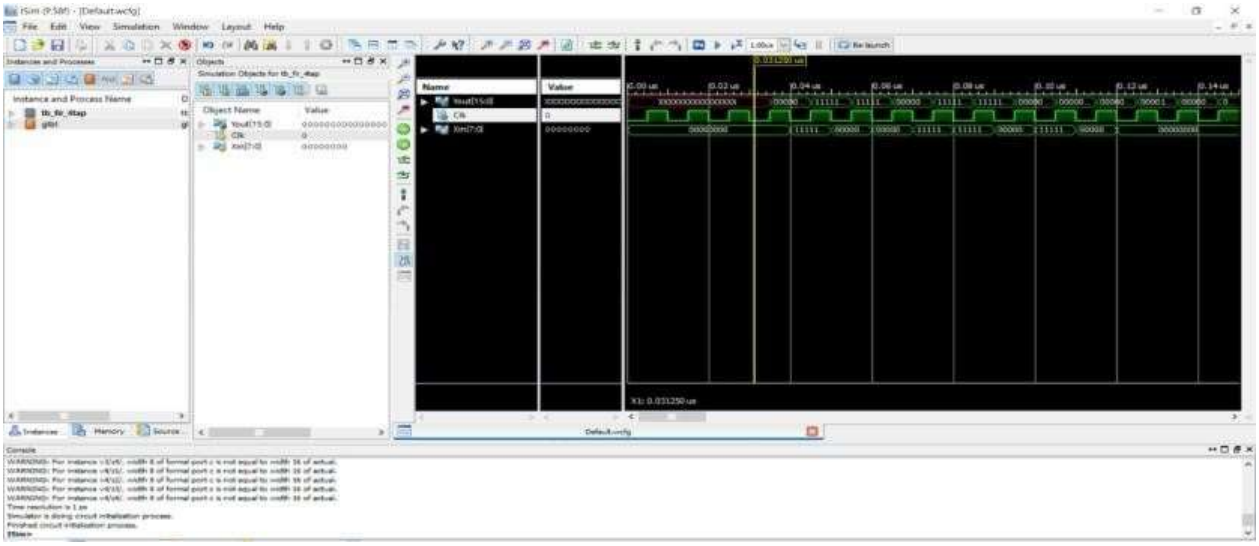

Fig. 4(d)

After implementing FIR Filter through Han-Carlson Adder and Vedic-Multiplier, we have seen an improvement in our filter and we got a delay of 7.139ns, which is less as compared as to the FIR Filter made by different types of adders (CLA, RCA, KSA).

6.1 CONCUSION:

After analyzing various adders and Vedic multipliers and combining different sets of adders and multipliers, some FIR filters are designed, which are described in this review paper. Out of these FIR Filters the best option on the basis of performance is Han-Carlson Adder based on Vedic mathematics. But it consumes high power.

Another suitable FIR filter is Carry Look Ahead Adder based on Vedic Mathematics which offers decent performance along with lesser power consumption. But the advantage is over CLA is speed. Han-Carlson Adder beats CLA in speed comparison. Finally, we can conclude that FIR Filter using Han-Carlson is one of the best and efficient Filter.

FUTURE SCOPE:

In VlSI field there are a lot of opportunities to procced this for higher research. This Fir Filter can be improved further in terms of parameter and their design to achieve more efficient result.However, Proposed filter design process also have some limitationsat some process corners that can be investigated and improved thoroughlyfor more efficient and accurate result.

REFERENCES:

[1] D. Yadav A. Mittal and A. Nandi, 'Comparative study of 16-order FIR filter design using different multiplication techniques', *IET Circuits, Devices Syst.*, vol. 11, no. 3, pp. 196–200, 2017, doi: 10.1049/iet-cds.2016.0146.

[2] R. Jagadale, 'Implementation of fir filter using vhdl', pp. 274–277.

[3] M. V Kaveri and S. Madhav, 'IMPLEMENTATION OF PROGRAMMABLE FIR FILTER USING', no. Icirca, pp. 585–589, 2018.

[4] A. Singh, V. Niranjan, and S. Nagaria, 'Efficient FIR filter design using booth multiplier for VLSI applications', *2018 Int. Conf. Comput. Power Commun. Technol. GUCON 2018*, pp. 581–584, 2019, doi: 10.1109/GUCON.2018.8674998.

[5] J. Thomas S. Jinesh, and R. Pushpangadan, 'Comparative study of performance vedic multiplier on the basis of adders used', *2015 IEEE Int. WIE Conf. Electr. Comput. Eng. WIECON-ECE 2015*, pp. 325–328, 2016, doi: 10.1109/WIECON-ECE.2015.7443929.

[6] P. S. Howal, M. C. Patel, and K. P. Upla, 'HDL implementation of digital filters using floating point vedic multiplier', *IEEE Int. Conf. Circuits Syst. ICCS 2017*, vol. 2018-Janua, no. Iccs, pp. 274–279, 2018, doi: 10.1109/ICCS1.2017.8326004.

[7] N. Udaya Kumar, Y. Rama Lakshmanna, K. Anirudh Kumar Maurya, and K. Bala Sindhuri, 'Design and implementation of 32-bit adders using various full adders', *2017 Innov. Power Adv. Comput. Technol. i-PACT 2017*, vol. 2017-Janua, pp. 1–6, 2017, doi: 10.1109/IPACT.2017.8245176.

[8] M. a Beach, G. V Tsoulos, and S. C. Swales, 'Performance Enhancement of', vol. 5, no. 2, pp. 1086–1090, 1996.

[9] G. Arya, and N. Varshney, 'Design and Execution of Enhanced Carry Increment Adder using Han-Carlson and Kogge-Stone adder Technique', *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 1163–1170, 2019, doi: 10.1109/ICECA.2019.8822194.

[10] Ananthakrishnan, A. Ajit, P. V. Arathi, K. Haridas, N. M. Nambiar, and S. Devi, 'FPGA Based Performance Comparison of Different Basic Adder Topologies with Parallel Processing Adder', *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 87–92, 2019, doi: 10.1109/ICECA.2019.8821925.

[11] N. Meghana, B. Koyada, M. O. Jaleel, and P. R. Jeripotula, 'A comparative study on adders', *Proc. 2017 Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2017*, vol. 2018-Janua, no. 0, pp. 2206–2230, 2018, doi: 10.1109/WiSPNET.2017.8300155.

[12] D. Yaswanth, S. Nagaraj, and R. V. Vijeth, 'Design and analysis of high speed and low area vedic multiplier using carry select adder', *Int. Conf. Emerg. Trends Inf. Technol. Eng. ic-ETITE 2020*, pp. 8–12, 2020, doi: 10.1109/ic-ETITE47903.2020.369.

[13] E. J. Rao, T. Ramanjaneyulu, and K. J. Kumar, 'Advanced multiplier design and implementation using Hancarlson adder', *2018 Int. Conf. Intell. Innov. Comput. Appl. ICONIC 2018*, pp. 1–5, 2019, doi: 10.1109/ICONIC.2018.8601252.

[14] S. Akhter and S. Chaturvedi, 'Modified Binary Multiplier Circuit Based on Vedic Mathematics', *2019 6th Int. Conf. Signal Process. Integr. Networks, SPIN 2019*, pp. 234–237, 2019, doi: 10.1109/SPIN.2019.8711583.

[15] S. Nithya, B. Pavithra, and P. Puvitha, 'HIGH - SPEED AND LOW - POWER CARRY SKIP ADDER USING HAN-CARLSON ADDER',Department of ECE, Dr.NGP Institute of Technology, Coimbatore. 1', vol. 4, no. June, pp. 1–8, 2017.