JOURNAL OF ADVANCEMENT IN ENGINEERING AND TECHNOLOGY

Journal homepage: http://scienceq.org/Journals/JAET.php

Research Article Open Access

Design and Implementation of FIR FILTER using MCM Architecture

E.Mallikarjuna[1*], L.S.Devaraj[2], R.Naresh Naik[3]

[1] Dept. of ECE, Intellectual Institute of Technology, Anantapur. India.
[2] Asst.Professor & HOD, Dept. of ECE, Intellectual Institute of Technology, Anantapur. India.
[3] Dept. of ECE, Intellectual Institute of Technology, Anantapur. India.
*Corresponding author: E.Mallikarjuna. E-mail: mallikarjuna.en@gmail.com

Received: June 12, 2015, Accepted: July 24, 2015, Published: July 24, 2015

ABSTRACT

To Design the low complexity bit-parallel multiple constant multiplications (MCM) operation, many efficient algorithms and architectures have been introduced in the last two decades. The MCM operation has been dominates the complexity of many digital signal processing systems. On the other hand, the digit-serial MCM design has becoming more popular and this design offers alternative low complexity MCM operations albeit at the cost of an increased delay. In this paper we are introducing high level synthesis algorithms, design architectures and CAD tools and also we address the problem of optimizing the gate level area in digit serial MCM design. The experimental results shows the efficiency of digit-serial MCM architectures and the proposed optimization algorithms in the design of MCM operations and finite impulse response filters.

Keyword: 0–1 integer linear programming (ILP), digit-serial arithmetic, finite impulse response (FIR) filters, CSE and GB algorithms, multiple constant multiplication.

INTRODUCTION

The Finite Impulse Response (FIR) filters play a very important role for the signal separation and signal restoration in Digital signal processing systems and control systems. And the FIR filters have great importance than IIR filters since their linear-phase and feed-forward implementations make them very stable high performance filters. A general FIR filter can be expressed as,

$$y[n] = \sum_{k=0}^{M-1} h[k]\, x[n-k]$$

$$y[n] = h[o]\ x[n] + h[1]\ x[n-1] + \ldots\ldots\ldots\ldots\ldots\ldots + h[M-1]\ x[n-M-1]$$

As shown in the above equation, the FIR filter consists of multiple constants multiplied with the input. Many times multiple constants are multiplied with the same input. These computations are called the multiple constant multiplications (MCM) block. Multiplication with a constant is called the constant multiplication. There are two types of multiple constants. First one is the single constant multiplication (SCM) and the other one is Multiple Constant Multiplication (MCM). Multiplication of single constant with an input is called Single Constant Multiplication. And multiplication of multiple constants with same input is called Multiple Constant multiplication (MCM).

The MCM block is very important to produce the constant multiplication operations in many digital signal processing systems (like DCT, FFT transformations), communication systems, Multiple input multiple output (MIMO) systems, error correcting codes and frequency control multiplications.

The MCM block in FIR filters can be implemented using the multiplier and multiplier-less algorithms as shown in Fig. 1 and 2 respectively. The multiplications are expensive in terms of area and power consumption when implemented in hardware. The relative cost of a multiplier is depends on the number of adders used in the multiplier architecture. For example, a k×k array multiplier has k times the logic area and twice the latency of the slowest ripple carry adder. But, recently many area, delay and power-efficient algorithms are implemented for multiplication purposes such as Wallace and Booth multipliers.

The multiplier based multiplication algorithms uses the full flexibility of a multiplier. But the FIR filter does not require the full flexibility of a multiplier because the MCM block consists of multiple constants which are fixed and obtained beforehand by the DSP algorithms. Hence, the multiplication of multiple constants with input data in MCM block is generally implemented with the shift-adds methods as shown in Fig. 2. Note that the here we are representing filter coefficients as constants since the coefficients are fixed constant variables. There are two types of arithmetic operations are used in the multiplication. First one is the bit-parallel design and another one is digit-serial design. The bit-parallel operators use extra hardware for implementations which occupy more area in FPGA and ASIC designs. But the shift operations are performed with free of cost in hardware, so the high-level algorithms uses the input data in parallel. In the digit-serial operators the data words are divided into digit sets which consists of d bits. The digit-serial operators occupy less amount of area and are independent of word length, they offer low complexity designs when compared to bit-parallel operators. But the digit-serial operators use D flip-flops for shift operators whereas in bit-parallel operators they are free.

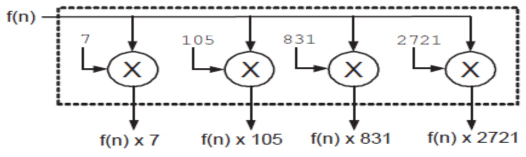


Fig. 1 A multiplier-based MCM example

Coming to the multiplier-less MCM block implementation, we have the many algorithms for MCM block implementation. The shift-adds method can be implemented by Digit-based Recoding, Common sub-expression Elimination and Graph-based algorithms.

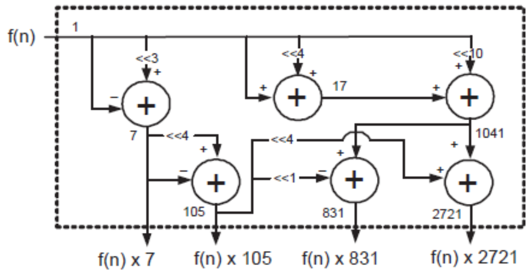


Fig. 2 A multiplier-less based MCM example

The first two are existing algorithms and third one is the proposed algorithm respectively.The reduction of gate-level area in FIR filters is achieved by reducing the number of addition/subtraction and shift operations. This problem is called MCM problem and formalized as 0-1 ILP (Integer Linear programming) problem. In existing systems that is In CSE algorithm is generated by using the different number representations called CSD representation. They provide certain improvement in the MCM problem. But the Graph-based algorithms use another representation called MSD representation which provide the better reductions of number of addition and shift operations in the MCM problem.

ALGORITHMS

Common-Subexpression Elimination Algorithm

The CSE algorithms are directly obtained from the digit-based recoding methods. In this, first we find common subpatterns in the representations of the constants after the constants are represented in conventional number representation such as binary and CSD formats. The main drawback in these algorithms is the performance of these algorithms depends on the number representation. This problem can be considered as NP-complete and its optimal solutions does in general not provide the optimal MCM solution. Recently, Dempster and Macleod proposes alternative number representations to find feasable solution for the MCM problem.

The main idea of common sub-expression elimination algorithm is to find the digit terms which are common between different constants and if they have common expressions then decrease the number of repeated expressions by sharing. It has many algorithms which deal with CSE and in most all algorithms have main three steps.

They are:

1. Identify the multiple expressions present in the input matrix of coefficients.
2. Then select one expression for elimination
3. Finally eliminate the same expression in all coefficients and share it.

This process is repeated until there are no more multiple expressions present in the multiple constants. There are two important metrics in this algorithm which are run time and quality of the solution.The following example explains CSE algorithm. Usually multipliers have large area and power and also multiplication is expensive in hardware. In MCM, since the constants are fixed and known before hand. Hence, multiplications are implemented by using shift and add/subtract methods. Suppose if you want to compute 23 * x. The constant 21 is represented in binary format as 10101. So, by using shift and add method we can compute 21x as below:

$$21x = (10101)_2 * x = x + x<<2 + x <<4$$

In this method, the complexity of the implementation is directly related to the number of non-zero digits in the constant representation. As told in the digit arithmetic operations we can reduce the number of non-zero digits by using signed digit representations. To reduce the number of non-zero digits we use canonical signed representation for CSE algorithm among all approaches. By this representation, common sub-expression elimination (CSE) will find the common expressions in binary representation of the constants. Let us consider the following example;

$$21* x = (10\underline{101})_2 * x = x+ x<<2+x<<4$$

$$13 * x = (10\underline{101}) * x= x+ x<<2+x<<3$$

In these two binary constant representations we have common expression 101. To implement these operations, we need four shift operations and four add operations. However, by using CSE algorithm if we eliminate the common expression "101" then we requires only three shift operations and three add operations. They are as follows;

$$F0 = (101)_2*x = x + x<<2$$

$$F1 = 21 * x = (10101)_2 * x = F0 + x << 4$$

$$F2 = 13 * x = (1101) * x = F0 + x<< 3$$

Graph Based Algorithm

The graph based algotims can be classified as bottom-up graph based methods and top-down graph based methods**.** The example for bottom-up methods are BH,BHM, n-RAG and Hcub methods. For top-down methods, Bernstein's Software-Oriented SCM Algorithm and the BBB Algorithm and Difference-Based Heuristics and the Diff.AG Algorithm are used. In this we considered only bottom-up graph based method and excluded the top-down method.The bottom-up methods constructs the graph representing the multiplier block. The graph construction uses heuristic process that determines the next graph vertex to add to the graph. In this there is no restriction of number representation, hence the Graph-based algorithms provide more freedom than other algorithms i.e. digit –based algorithms and CSE algorithms and produces less number of operations.

We have another type of algorithm known as Hybrid algorithms which combine different algorithms possibly from different classes. For example Choo et al. constructs the multiplier block with fixed topology to compute the so called differential coefficients and then switches to a CSE algorithm for the multiplication the different coefficients.Among all these algorithms n-RAG and Hcub algorithms provides best solutions with smallest number of additions/subtract operations among all algorithms. The graph based algorithms have less restriction they are expected to outperform other methods.

In embedded systems area and run time management are most challenging issues. Many embedded systems uses the DSP algorithms for image processing and video processing which are very compute intensive. A custom hardware implementation of these algorithms can provide better timing of the embedded systems can be met. The main objective of these algorithms is to implement the multiple constant multiplications in digital filtering, processing, linear transforms etc by reducing number of addition/subtract operations. The optimization of MCM problem lead to improvement in various design parameters like area and delay. This problem is known as MCM problem.There are many methods for reducing the number of adders/ subtractors. Here we are implementing and investigate the following algorithms and compare the results with CSE and digit-based algorithms.

ARCHITECTURE OF MCM BLOCK:

The exact CSE algorithm will not provide the global solutions since all implementations of a constant are found from its representations that is binary, CSD or MSD representations. Also the MCM problem in the filters is NP complete problem due to the NP-completeness of the MCM problem. The generated 0-1 ILP solver could not provide the better solutions to the 0-1 ILP problem. Hence, we are proposed a new algorithm called the Graph-based heuristic algorithms for better gate-level area optimization.

The exact GB algorithms search for a better solution with the minimum number of operations using breadth-first and depth-first manners. The efficient Graph-based algorithms have two parts. They are optimal and heuristic parts. In the optimal part, if a target constant has a single operation to implement then it is synthesized. If there are unimplemented constants exists then the heuristic part finds the minimum number of intermediate constants for synthesizing the target constants. In previous chapter, we have seen the Graph-based algorithms, among those algorithms we chooses n-RAG and Hcub algorithms for implementation of constants. The RAG-n algorithm initially chooses a single unimplemented target constant with the smallest single coefficient cost and evaluated with a single operation which consists of one or two intermediate constants. The Hcub algorithm implements the target constant with a single intermediate constant which yields the better solutions for gate level area optimization. The RASG-n algorithm chooses the minimum number of intermediate constants that requires minimum number of shift operations than the RAG-n algorithm.

Thus, we found the minimum number of intermediate constants which implements the target constants in a single operation. If there are any unimplemented synthesized constants in each iteration then we favor one intermediate constant among all intermediate constants which have less hardware and enable the non-synthesizable constants with a reduced area. Let us consider the constant coefficient pair 29x and 43x, 59x and 89x. These constants can be implemented by using the Graph-based algorithm as follows. In the constant pair 29x and 43x, the best intermediate constant $7x = (111)_{bin}$ x is found to implement those constants with two addition, one subtraction and six shift operations as shown in Fig.4(a). The intermediate constant 7x cannot to be found in the binary representation of the 43x in the exact CSE algorithm.

Similarly, the constant pair 59x and 89x can be implemented by using the Graph-based algorithm with the intermediate constant $15x = (1000\bar{1})_{msd}$ x. The constants can be implemented with two subtraction, one addition and seven shift operations as shown in Fig.4(b). The intermediate constant 15x is not appearing in the Common sub-expression elimination algorithm.

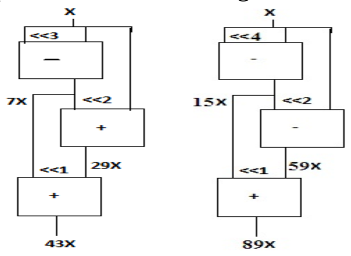


Fig.3 GB algorithm for the implementation of 29,43,59,89 coefficients

The digit-serial implementation of constant pairs 29x and 43x, 59x and 89x are shown in the Fig.4.8. In the figure, the full adders perform either addition or subtraction operations and the D flip-flops perform the left shift operations.

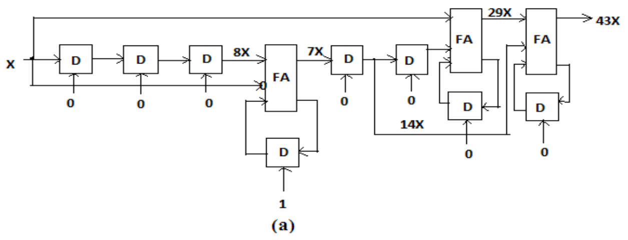


Fig.4 (a) Digit-serial design of shift-adds implementation of constants 29x and 43x

To implement the constant pairs 29x and 43x, 59x and 89x using the digit-based recoding, exact CSE and Graph-based algorithms, the number of addition/subtraction and shift operations required are shown in Table-1.

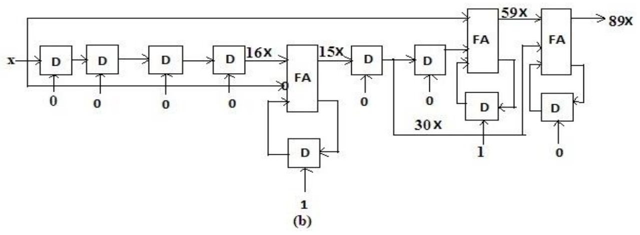


Fig.4 (b) Digit-serial design of shift-adds implementation of constants 59x and 89x

Table 1 Comparison table for number of operations

Algorithms	Additions /subtractions	Shifts
Digit-based recoding	13	44
Exact CSE	9	25
Graph-based	6	13

The 4-TAP FIR Filter Implementation Using MCM Block

The implementation architecture of the four tap Finite Impulse response filter with the Multiple Constant Multiplication (MCM) block is shown in Fig. 5. The architecture consists of following elements,

a. MCM block
b. Digit-serial Adder
c. 8-bit Register

The Multiple Constant Multiplication (MCM) block is implemented using the Digit-based algorithm, CSE algorithm and GB algorithms. The digit-serial adders are required to add the constants to produce the final filter output. The 8-bit register is used to store the output results of each adder, since the next adder will requires the previous output as one of the input. In the implementation architecture, x(n) represents the input data, y(n) represents the output data of the filter and D is not a flip-flop which represents the storage element.

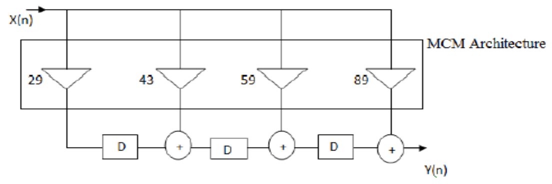


Fig .5 Four tap FIR filter implementation with MCM architecture

IMPLEMENTATION RESULTS

GB algorithm can be applied for any coefficient pair combinations. Hence GB algorithm is used and number of operations is reduced drastically than other algorithms. Four filter coefficient 29,43,59,89 values are taken for digit serial FIR filter design. X(n) is taken as a input sequence and Y(n) is taken as output sequence.

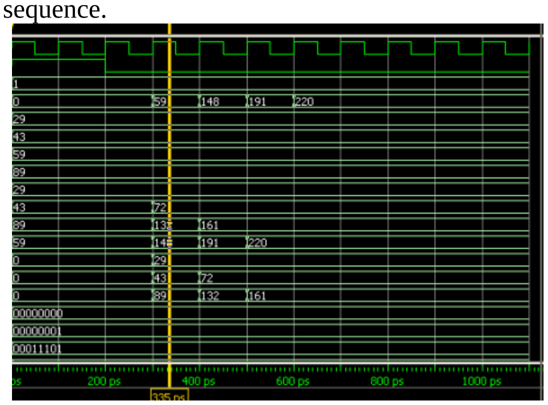


Fig .6 Output for FIR filter with CSE algorithm

Fig.7 displays 4 tap filter with GB algorithm. This simulation result was displayed by modelsim software. These are the simulation results displayed by modelsim software. After completion of simulation process in Modelsim tool, synthesis process is takes place to calculate gate count and delay report. Fig.8 shows the RTL schematic view of FIR filter with MCM architecture.

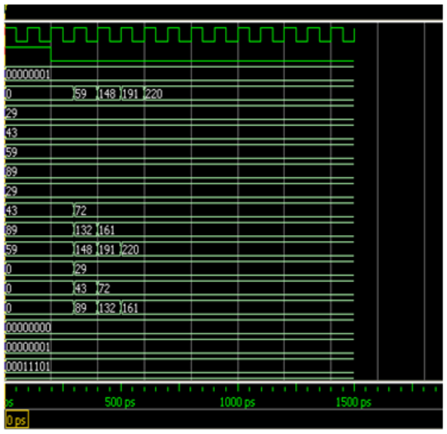


Fig .7 Simulation result of FIR filter using Graph-based algorithm

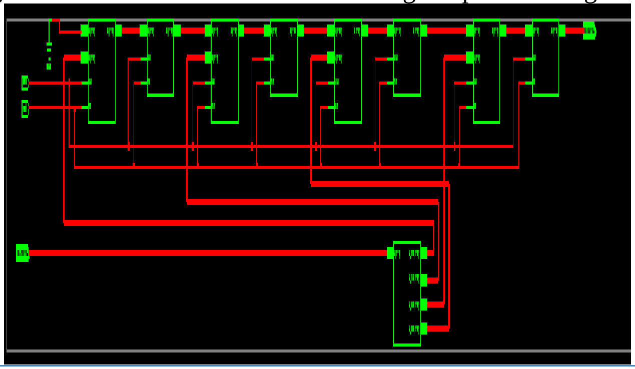
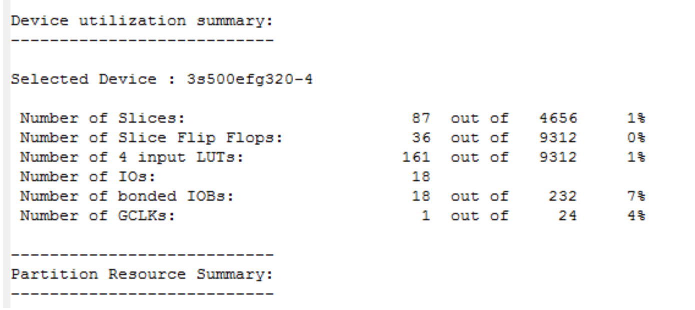


Fig.8 RTL schematic diagram

FIR FILTER DEVICE UTILIZATION REPORT: FIR filter with digit size d=4 was synthesized separately for device utilization in terms of gate count with each different algorithms. Fig 9 and Fig.10 shows the device utilization report of FIR filter with GB algorithm.

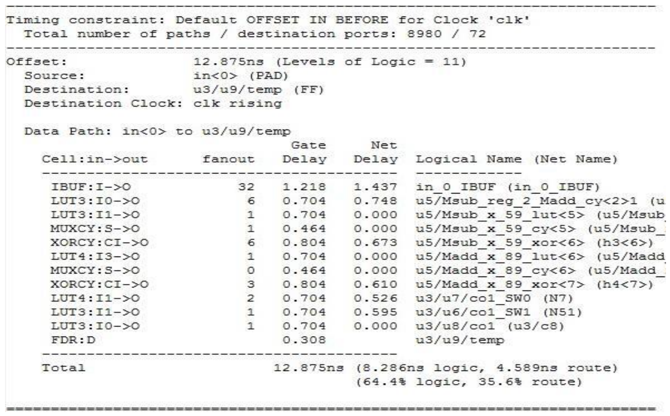


```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                       87  out of   4656     1%
 Number of Slice Flip Flops:             36  out of   9312     0%
 Number of 4 input LUTs:                161  out of   9312     1%
 Number of IOs:                          18
 Number of bonded IOBs:                  18  out of    232     7%
 Number of GCLKs:                         1  out of     24     4%


---------------------------
Partition Resource Summary:
---------------------------
```

Fig. 9 Device utilization report

explains that the FIR filter with GB algorithm utilized less number of gate count and delay than other two algorithms.

FIR FILTER DELAY SYNTHESIS REPORT



```
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 8980 / 72
-------------------------------------------------------------------------
Offset:              12.875ns (Levels of Logic = 11)
  Source:            in<0> (PAD)
  Destination:       u3/u9/temp (FF)
  Destination Clock: clk rising

  Data Path: in<0> to u3/u9/temp
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            32   1.218   1.437  in_0_IBUF (in_0_IBUF)
     LUT3:I0->O            6   0.704   0.748  u5/Msub_reg_2_Madd_cy<2>1 (u
     LUT3:I1->O            1   0.704   0.000  u5/Msub_x_59_lut<5> (u5/Msub
     MUXCY:S->O            1   0.464   0.000  u5/Msub_x_59_cy<5> (u5/Msub_
     XORCY:CI->O           6   0.804   0.673  u5/Msub_x_59_xor<6> (h3<6>)
     LUT4:I3->O            1   0.704   0.000  u5/Madd_x_89_lut<6> (u5/Madd
     MUXCY:S->O            0   0.464   0.000  u5/Madd_x_89_cy<6> (u5/Madd_
     XORCY:CI->O           3   0.804   0.610  u5/Madd_x_89_xor<7> (h4<7>)
     LUT4:I1->O            2   0.704   0.526  u3/u7/co1_SW0 (N7)
     LUT3:I1->O            1   0.704   0.595  u3/u6/co1_SW1 (N51)
     LUT3:I0->O            1   0.704   0.000  u3/u8/co1 (u3/c8)
     FDR:D                     0.308          u3/u9/temp
    ----------------------------------------
    Total                     12.875ns (8.286ns logic, 4.589ns route)
                                       (64.4% logic, 35.6% route)

=========================================================================
```

Fig.10 Delay synthesis report

Table -2: Delay and Gate Count Comparison

FIR filter algorithm	Delay (in nsec)	Device utilization
Digit-based Recoding	14.203	323
Exact CSE	15.528	321
Graph-based	12.875	299

CONCLUSION:

Thus the implementation of digit serial FIR filter was implemented with low complexity MCM architectures using GB algorithm. Hence this MCM approach drastically reduces the system complexity, area and delay and FPGA hardware real time implementation has performed with spartan3 version. Device utilization and delay values are compared for hardware implementation. Future enhancement of this paper is to design MCM architecture with more coefficient pairs for FIR filter implementation.

REFERENCES

1. L. Wanhammar, DSP Integrated Circuits. New York:

Academic, 1999.
2. M. Ercegovac and T. Lang, Digital Arithmetic. San Mateo, CA: Morgan Kaufmann, 2003.
3. R. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 43, no. 10, pp. 677–688, Oct. 1996.
4. I.-C. Park and H.-J. Kang, "Digital filter synthesis based on minimal signed digit representation," in Proc. DAC, 2001, pp. 468–473.
5. L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 6, pp. 1013–1026, Jun. 2008.
6. A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 42, no. 9, pp. 569–577, Sep. 1995.
7. Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," ACM Trans. Algor., vol. 3, no. 2, pp. 1–39, May 2007.
8. L. Aksoy, E. Gunes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," J. Microprocess.Microsyst., vol. 34, no. 5, pp. 151–162, Aug. 2010.
9. P. Cappello and K. Steiglitz. Some Complexity Issues in Digital Signal Processing. IEEE Trans. on Acoustics, Speech, and Signal Processing,32(5):1037.1041, October 1984.
10. R. Hartley and P. Corbett. Digit-Serial Processing Techniques. IEEE TCAS II, 37(6):707-719, 1990.
11. L. Aksoy, C. Lazzari, E. Costa, P. Flores, and J. Monteiro, "Efficient shift-adds design of digit-serial multiple constant multiplications," in Proc. Great Lakes Symp. VLSI, 2011, pp. 61–66.
12. P. Flores, J. Monteiro, and E. Costa, "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications," in Proc. Int. Conf. Comput.-Aided Design, Nov. 2005, pp. 13–16.

Citation: E.Mallikarjuna et al.. Design and Implementation of FIR FILTER using MCM Architecture. J. of Advancement in Engineering and Technology. V3I2. DOI: 10.15297/JAET.V3I2.05