CHAPTER 5

RECONFIGURABLE MULTIPLIER ON FIR FILTER FOR SDR RECEIVER

5.1 INTRODUCTION

Multiplications occur frequently in DSP systems, communication systems, and other application specific integrated circuits. The multiplication operation is present in many part of a digital system or digital computer, most suitably in signal processing, graphics and scientific computations. With advances in technology, various techniques have been proposed to design multipliers which offer high speed, low power compact VLSI implementation. These three parameters i.e., power, area and speed are always traded off.

Multiplication is the key in arithmetic operation and plays an important role. Unfortunately, the major source of power dissipation in digital signal processors is multipliers. Therefore, the design of multipliers for digital signal processing applications should be efficient while still being able to handle low-power applications. (Floyd 2005).Multipliers, being relatively complex units are deciding factors to the overall speed, area and power consumption of digital computers (Tokheim 1994). The diversity of application areas for multipliers and the ubiquity of multiplication in digital systems exhibit a variety of requirements for speed, area, power consumption, and other specifications.

Speed, area and hardware resources have been the major design factors and concerns in digital design. Parallel multipliers are combinational circuits and can be subjected to any standard combinational logic optimization. However, the complex structure of the multipliers imposes a number of difficulties for the EDA tools, as they simply cannot consider the multipliers as a whole; i.e., EDA tools have to limit the optimizations (Morris Mano et al 2003) to a small portion of the circuit and perform logic optimizations. On the other hand, multipliers are arithmetic circuits and considering arithmetic relations in the structure of multipliers can be extremely useful and can result in better optimization results.

Radios provide a means of basic wireless communication between individuals. While classic radios depend on the use of analog electronics, the introduction of software-based technologies in radios add flexibility and allow the developers to integrate radios into various software- based systems. However, with the use of software in a system classically understood to be purely based on electronics, new challenges and issues are introduced related to the software, computational platforms and interfaces used in realizing such system (Rafiquzzaman 2005). For example, issues such as algorithm implementation, processor cache performance and thread scheduling are important for software-based radios.

SDR is defined as a radio that accommodates a significant range of RF bands and air interface modes through software. Basically, SDR is able to realize significant portions of its radio functionality in software instead of hardware. By introducing software radio pieces in the critical radio transceiver path, the characterization of such radios involves the computational devices used (Cook 2003).

Looking at the diverse applications for SDR, which range from mobile handsets to base station level radios, it is evident that various classes of power utilization and computational density are needed to cover the wide spectrum of SDR needs (Rafiquzzaman 2005, Cook 2003). For such applications, three main computing devices are available: GPP, DSP and FPGA. Therefore, planning for radio hardware from a computational perspective needs to account for the end goal application and ultimate usage scenario.

SDR is fast becoming a crucial element of wireless technology. The use of SDR technology is predicted to replace many of the traditional methods of implementing transmitters and receivers while offering a wide range of advantages including adaptability, reconfigurability, and multifunctionality encompassing modes of operation, radio frequency bands, air interfaces, and waveforms. Research in this field is mainly directed towards improving the architecture and the computational efficiency of SDR systems.

SDR refers to wireless communication in which the transmitter modulation and the receiver demodulation are both generated through software. The main advantage of this approach is flexibility, as the software runs on one common hardware platform for any type of receiver configuration. The most computationally intensive part of the wideband receiver of a SDR is the Intermediate Frequency (IF) processing block. Digital filtering is the main task in IF processing. The computational complexity of FIR filters used in the IF processing block is dominated by the number of adders (subtractions). The proposed reconfigurable synthesizable multiplier blocks offer significant savings in area over the traditional multiplier blocks. They are implemented on FPGA hardware platforms. In

addition, software radio has recently gained much attention due to the need for integrated and reconfigurable communication systems.

In Section 5.2, the concepts of multiplier block are described. Binary Common subexpression Elimination method is discussed in the Section 5.3. The Section 5.4 explains the Efficient multiplier for transposed FIR filter followed by the Multiplier block synthesis in Section 5.5. The New MAG algorithm is presented in Section 5.6, followed by the presentation of results in Section 5.7. Conclusion is given in Section 5.8.

5.2 MULTIPLIER BLOCK

A popular technique for implementing the transposed form of FIR filters is the use of a multiplier block, instead of using multipliers for each constant.

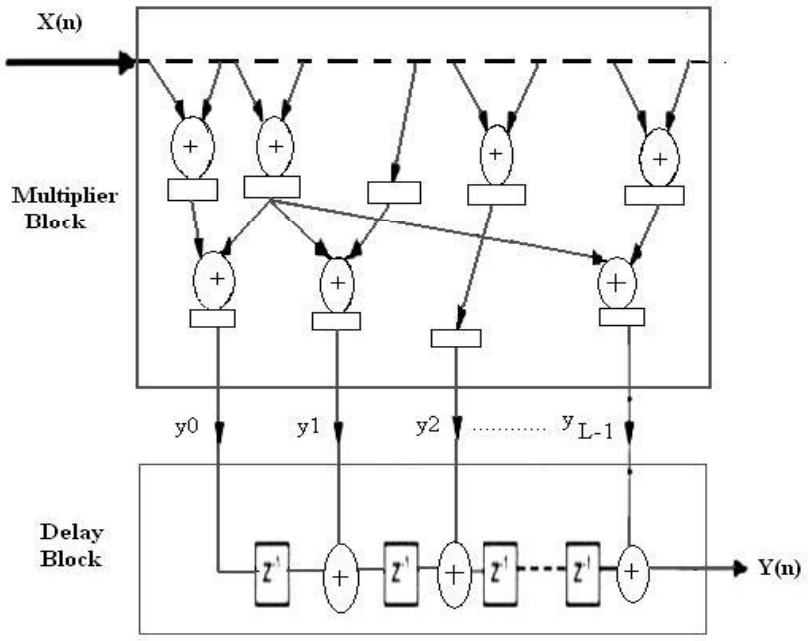

Figure 5.1 Replacing constant multiplications by multiplier block

The multiplications are with the set of constants h(k) and are replaced by an optimized set of additions and shift operations, involving computation sharing as shown in Figure 5.1. Further optimization can be done by factorizing the expression and finding common sub expressions.

The performance of this filter architecture is limited by the latency of the biggest adder. This uses a modified common sub expression elimination algorithm that minimizes the number of adders as well as the number of latches with significantly reduced area.

FIR filters are used extensively in mobile communication systems to perform tasks such as channelization, channel equalization, matched filtering and pulse shaping. The number of adders (subtractors) termed as Logic Operators (LOs) used to implement the coefficient multiplications determines the complexity of FIR filters. Hence, the methods that minimize the complexity of multiplication in FIR filters focus on reducing the number of LOs.

Among the approaches for reducing the LOs, the Common Subexpression (CSE) technique produced considerable reduction of LOs. The aim of CSE is to identify multiple occurrences of CSs that are present in the coefficient set and to eliminate the redundant multiplications. As the computation of multiple identical expressions needs to be implemented only once, the amount of hardware used can be reduced. This in turn reduces the area and power of FIR filters.

In this chapter, an algorithm (N-MAG) is proposed and implemented for designing FIR (Peiro et al 2002, Dempster et al 1995). Performances of this algorithm are compared with Binary Common Subexpression Elimination algorithm.

5.3 BINARY COMMON SUBEXPRESSION ELIMINATION (BCSE) METHOD

The complexity of linear phase FIR filters used in SDR receiver is dominated by the complexity of coefficient multipliers. The number of additions (subtractions) used to implement the coefficient multiplier determines the complexity of FIR filters. Many approaches including coefficient coding using efficient arithmetic schemes, coefficient optimization techniques, distributed arithmetic techniques, Read-Only Memory (ROM) - based designs, and CSE techniques have been proposed.

CSE methods based on CSD coefficients produce low complexity FIR filter coefficient multipliers. The most computationally intensive part of SDR receiver is the channeliser and it operates at the highest sampling rate (Mitola 2000). It extracts multiple narrowband channels from a wideband signal using a bank of FIR filters, called channel filters. Low power and high speed filters implemented with the minimum number of adders are required in the channelizers. Among the approaches for reducing the number of adders in the multipliers of FIR filters, the CSE technique produced the best hardware reduction since it deals with multiplication of one variable(input signal) with several constants(coefficients). The goal of CSE is to identify multiple occurrences of identical bit patterns that are present in the CSD representation of coefficients, and eliminate these redundant multiplication.

In Mehendale et al (1995), a graphical algorithm was proposed to identify and eliminate 2- bit subexpressions. A more efficient method was proposed in Hartley (1996) which eliminated the most commonly occurring 2-bit subexpressions. As an additional criterion in the subexpression

identification process, an estimation of a latch count improvement was also considered in Hartley (1996). A modification of the 2-bit CSE technique presented in Mehendale et al (1995) for identifying the "proper" patterns for elimination of common subexpression and to maximize the optimization of the impact was proposed in Pasko et al (1999). In Peiro et al (2002), the technique in Hartley (1996) was modified to minimize the logic depth into digital structure. In general, these methods considered the elimination of identical bit patterns within the coefficient known as Horizontal Common Subexpression Elimination (HCSE) and among different coefficients known as Vertical Common Subexpression Elimination (VCSE). In Vinod and Lai (2005), it has been shown that the HCSE technique offered better reduction of adders and critical path lengths than VCSE in FIR implementations.

A CSE algorithm, using binary representation of coefficients for the implementation of higher order FIR filter with a fewer number of adders is used (Jongsun Park 2004). BCSE method is more efficient in reducing the number of adders needed to realize the multipliers when the filter coefficients are represented in the binary form. The observation is that the number of unpaired bits (bits that do not form Common Subexpressions (CSS)) is considerably fewer for binary coefficients compared to CSD coefficients, particularly for higher order FIR filter (Pasko et al 1999, Scholnik et al 2002, Gustafsson et al 2007).

Binary Horizontal Common Subexpression Elimination (BHCSE) deals with the elimination of redundant Binary HCSs (BHCSs) that occur within a coefficient. In general, an n-bit binary number can form $2^{(n-1)}$-1 BHCSs among themselves. For example, a 3-bit binary representation can form 4 BCSs, which are [0 1 1], [1 0 1], [1 1 0] and [1 1 1]. Other BCSs such

as [0 0 1], [0 1 0] and [1 0 0] do not require any adder for implementation and have only one nonzero bit.

These BHCSs can be expressed as follows where x is the input signal.

These BCSs can be expressed as:

$$[0\ 1\ 1] = x_2 = 2^{-1}x + 2^{-2}x \quad (5.1)$$

$$[1\ 0\ 1] = x_3 = x + 2^{-2}x \quad (5.2)$$

$$[1\ 1\ 0] = x_4 = x + 2^{-1}x \quad (5.3)$$

$$[1\ 1\ 1] = x_5 = x + 2^{-1}x + 2^{-2}x \quad (5.4)$$

where x is the input signal. A straightforward realization of these BCSs would require 5 adders. However, x_2 can be obtained from x_4 by a right shift operation (without using any adders) as follows.

$$x_2 = 2^{-1}x + 2^{-2}x = 2^{-1}(x+2^{-1}x) = 2^{-1}x_4 \quad (5.5)$$

Also, x_5 can be obtained from x_4 using an adder:

$$x_5 = x + 2^{-1}x + 2^{-2}x = x_4 + 2^{-2}x \quad (5.6)$$

Thus only 3 adders are needed to realize the BCSs (5.1)-(5.4). In general, the number of adders required for all the possible n-bit binary subexpressions is

$$2^{n-1} - 1 \quad (5.7)$$

In this method, the 3-bit BCSs (5.1)-(5.4) are chosen as it produces the best results. It is observed that the adder reductions are not significant when other 4-bit, 5-bit and 6-bit BCSs are used. As the bit length of the BCS increases, the number of adders required for all the possible subexpressions also increases. Therefore, BCSs with more number of bits do not offer considerable reduction of adders.

For instance consider coefficient h1= [0.001011000110] is grouped using 3-bit (including 2-bit) BHCSs as in Figure 5.2. This requires 2 Logical operators.

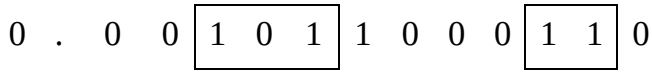

Figure 5.2 Coefficient h_1 grouped using 3–bit BHCs

Logic Depth (LD) is the number of adder steps in the maximal path of the decomposed multiplications, which determines the speed of the coefficient multiplications. Hence, LD is an important performance metric in the implementation of FIR filters. To illustrate the example, the coefficient below is considered.

h_4 = [0 . 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0]

Figure. 5.3 represents the implementation using the proposed method. The numerals adjacent to the data path represent the number of bitwise right shift. The bold dotted line shows the LD.

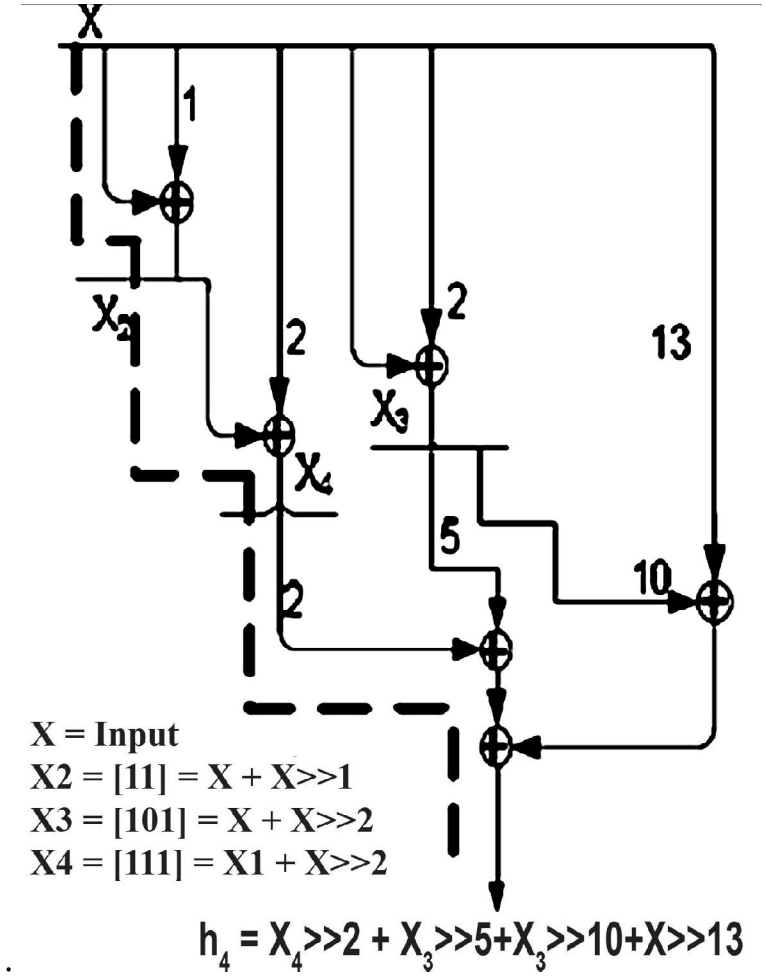

Figure 5.3 Implementation of example h_4 - 3-bit BSE

5.4 EFFICIENT MULTIPLIER FOR TRANSPOSED FIR FILTER

Figure 5.4 shows three full-parallel, fixed-coefficient FIR filter structures that are mathematically identical but differ in architecture. Derived from the standard FIR structure using cut-set retiming, the transposed FIR as in Figure 5.4.b yields an identical mathematical response but with several advantages for FPGA implementation :

(i) No input sample shift registers are required since each sample is fed to each tap simultaneously;

(ii) The pipelined addition chain maps efficiently;

(iii) Filter latency is reduced;

(iv) Identical tap coefficient magnitudes can share multiplication hardware because taps receive the input sample simultaneously.

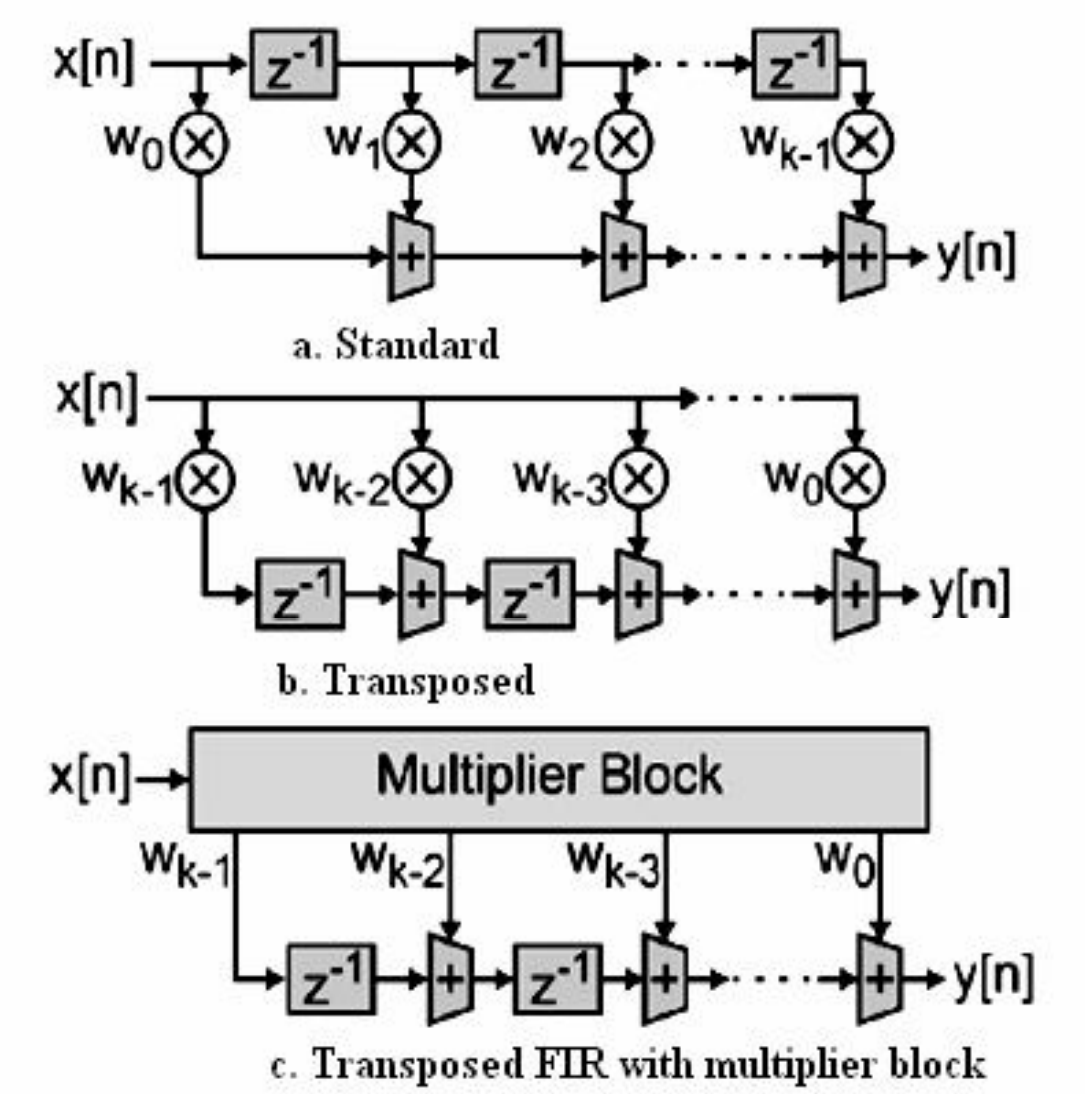

Figure 5.4 Full-parallel FIR filter structures

For maximum sampling rates, all multiplication hardware can be pipelined. In Figure 5.4 (c) the coefficient multipliers of the transposed FIR have been replaced with a multiplier block that generates all required multiples of the filter input using cascaded adds, subtracts and shifts. This filter architecture is known to be a highly area efficient method of implementing fixed-coefficient, full-parallel FIR filters. It is the multiplier block that determines filter implementation efficiency regardless of hardware platform.

The reconfigurability of the FIR filter is done by reconfiguring the multiplier block (MB) shown in Figure 5.4.(c). One of the key requirements to channelize in SDR is reconfigurability. The proposed architectures can be reconfigured by changing the values of coefficients stored in the LUT. Thus the same hardware architecture can be used for different filter specifications to achieve the necessary reconfigurability.

5.5 MULTIPLIER BLOCK SYNTHESIS

5.5.1 Minimized Adder Graph (MAG) Algorithm

MAG was defined by Dempster et al (1994) and generates minimum adder graphs consisting of shifts, adds and subtracts for implementing constant integer multiplication of individual values up to 12-bits. Multiplication by a constant is common when implementing DSP on FPGAs and efficient implementation has received attention from the research community (Scholnik 2002). Wirthlin and McMurtrey (2001) utilize the features of recent FPGA architectures to further reduce the area requirement of conventional constant coefficient multipliers. The implementation techniques described also allow for multiplication constants to be changed since it is not the structure of the logic that dictates multiplication values but rather look-up table contents.

Using the MAG algorithm, Dempster et al (1994) demonstrate a 16% average reduction in adder cost over CSD representation and show CSD to reduce average adder cost by 33% over binary representation. From an FPGA perspective, for single coefficients, these adder reductions also translate into hardware savings. In general, for each integer value in range, MAG finds numerous graphs for each value that all implement the required multiplication with the minimum number of adders/vertices.

Dempster et al (1994) suggest differentiating between these graphs by calculating the number of single-bit full adders required to implement each graph and selecting the graph requiring the least. This differentiation is motivated by attempting to minimize VLSI area consumed when implementing the multiplication. MAG also generates a table containing the adder cost of each integer value in range.

5.5.2 MAG Extensions and Modifications

The MAG implementation was extended beyond the 12-bit range (imposed by Dempster et al (1994) for physical RAM constraints) using the generic graph extension cases. The potential for greater differentiation between the multiple graphs found by MAG for each integer value in range was noted. Instead of using one level of differentiation with the single-bit full adder count metric, two metric levels (primary and secondary) are implemented to allow one ‘best graph’ to be selected.

For each integer, the primary metric selects the subset of graphs with minimum logic depth and from that subset; the secondary metric selects the graph with lowest vertices sum to minimize bit-widths (Samueli 1989). These new metrics reflect the low FPGA area observations described previously and in general, leave only one best graph per integer, whereas

the sole ‘single-bit full adder’ metric selects multiple graphs from which an arbitrary choice must be made.

It is noted that up to adder cost 3, for each integer in range, the modified MAG implementation stores and allows extension/branching from every graph found. This allows full searching of the graph space up to and including cost 4. From cost 4 graphs upwards (i.e. beyond 12-bits), only one ‘best graph’ (as selected by the new primary/secondary metric system described previously) is stored and used per integer for extending to higher cost graphs (Gustafsson 2007). This restriction is for practical implementation concerns of available physical RAM and algorithm run-time.

5.6 NEW MAG ALGORITHM

5.6.1 Efficient Multiplier Design

Circuits for shift-and-add multiplication by one or more fixed-point constant coefficients can be built using both adders and subtractors. Graph-based methods are used to illustrate these circuits, where each vertex represents an adder and has two inputs and an edge represents a shift. After Bull and Horrocks (1991), algorithms are developed that use fewest adders for multiplication by a single integer (Dempster et al 1994 Gustafsson et al 2002) and by several integers (Dempster et al 1995).

Subexpression Elimination (SE) introduced by Hartley (1991) also aims to reduce adders in fixed-point multiplication. It combines pairs (Hartley 1991) or groups (Potkonjak et al 1996, Pasko 1999) of digit patterns that recur in Signed-Digit (SD) representations of coefficients. A single adder can produce each pattern (subexpression), reducing overall adder count. CSD and other representation (Potkonjak 1996) have been used.

MAG algorithm is used to design shift-and-add circuits that multiply by an integer using fewest possible adders. It considers the circuit as a graph, made up of two-input adders. It performs an exhaustive search of all possible graph topologies (i.e. interconnections of the two input adders) and produces two tables. One contains the number of adders required to produce the integer multiplier. The other contains the partial products that are the outputs of the adders in the circuit (other than the circuit's output) – known as "fundamentals".

5.6.2 N-MAG Algorithm

The new algorithm aims to produce shift-and-add multiplication using fewest adders and the complete description is given as below.

Initialisation process

e_1, e_2 = coefficient values

C_1, C_2 = optimal adder costs of e_1, e_2 taken from the MAG cost table. e_1, e_2 are selected without loss of generality such that $C_1 \geq C_2$

C = cost of multiplier block

Process 1 : Select the values e_1, e_2 and divide e_1, e_2 by 2 until they become odd. Any even number can be generated by shifting an odd number. This step simplifies many graphical algorithms.

Process 2 and 3 for the removal of the Trivial cases

Process 2 : Omit the trivial case 1.

If $e_1 = e_2$ and $C = C_1$

i.e., if the coefficients are the same, both outputs are the same.

Process 3 : Omit the trivial case 2.

If $C_2 = 0$ and $C = C_1$

i.e., if one of the coefficient has cost '0' , it means that it is power of 2 and it requires no adder. Hence there is no contribution to the overall cost of the circuit.

Process 4 : Look for cases where e_2 is a fundamental of e_1. In this case $C=C_1$.

This search loads all of the possible fundamentals for e_1 from the MAG table and identifies e_2.

After the process 4, examine all the possible cases

where $C = C_1$

when $C = (C_1 + 1)$, it reaches the optimal point and terminate the algorithm.

Process 5 : Find all the sets of fundamentals of the coefficients.

Calculate C_c

where C_c = maximum number of common fundamentals.

Set $C = C_1 + C_2 - C_c$

If $C = (C_1 + 1)$, then terminate the algorithm.

Process 6 : Generate a new set of table as similar to those used by MAG table where all possible graphs are described that can synthesis e_1, e_2 at cost C_1+1, C_2+1

Again execute from process 4 and terminate if a case is found where $C = C_1 + 1$.

After process 6, all possible cases are examined where $C = C_1 + 1$. If $C = C_1 + 2$, then it is optimal and the algorithm can terminate.

5.7 SYNTHESIS RESULTS

In this section, the synthesis results of the proposed design are shown in Table 5.1. The simulation results of FIR filter is shown in Figure 5.5 and RTL schematic of FIR filter is shown in the Figure 5.6.

Table 5.1 Synthesis results of the proposed FIR filter

2vp2fg256-6	Area Used	Utilization
Number of Slices	59 out of 1408	4 %
Number of Slice Flip Flops	85 out of 2816	3 %
Number of 4 input LUTs	52 out of 2816	2%
Number of bonded IOBs	22 out of 140	16 %
Number of GCLKs	1 out of 16	6 %

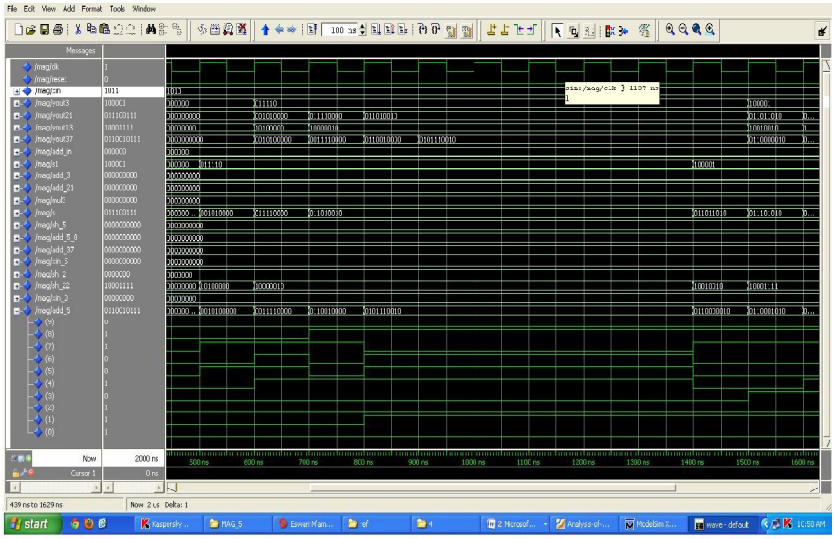

Figure 5.5 Simulation results of the proposed design

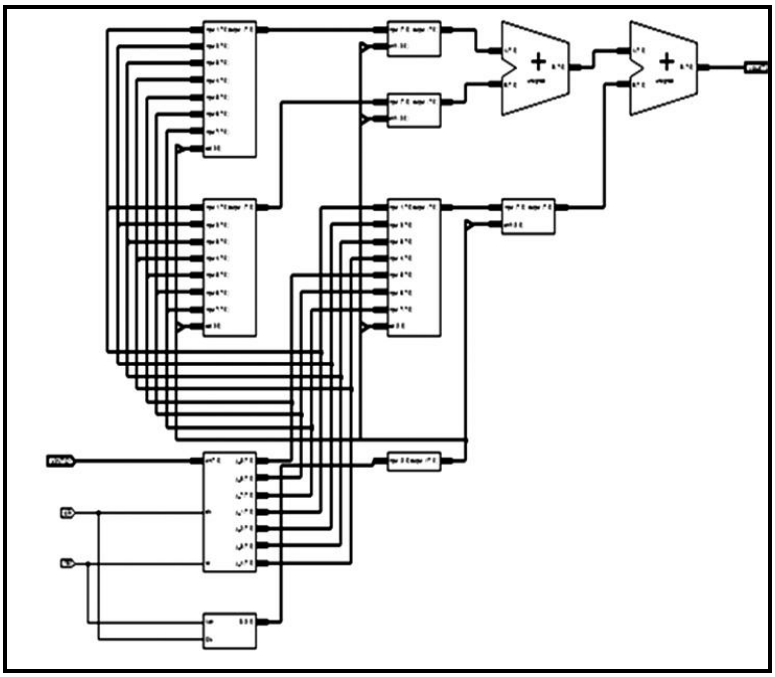

Figure 5.6 RTL schematic of the proposed FIR filter

Table 5.2 shows the comparison of the synthesis results for FIR filter design using BCSE and N-MAG algorithm. Figure 5.7 demonstrates the comparison of area utilization and performance comparison for POWER is shown in Figure 5.8.

Table 5.2 Slices, LUT and power comparison

Parameter	BCSE	N-MAG
Slices	81	39
LUT	64	37
Power (mW)	365	298

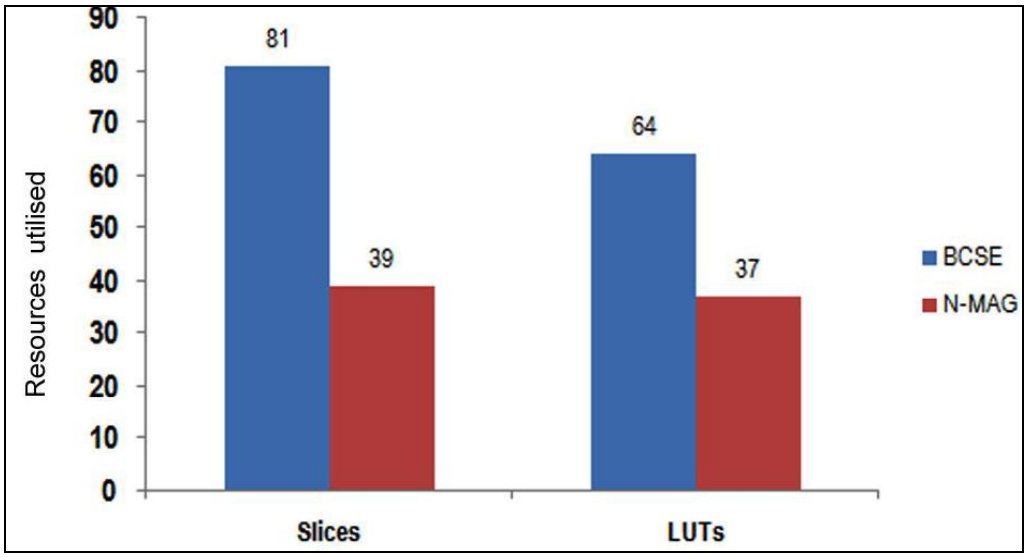

Figure 5.7 Comparison of resource utilisation

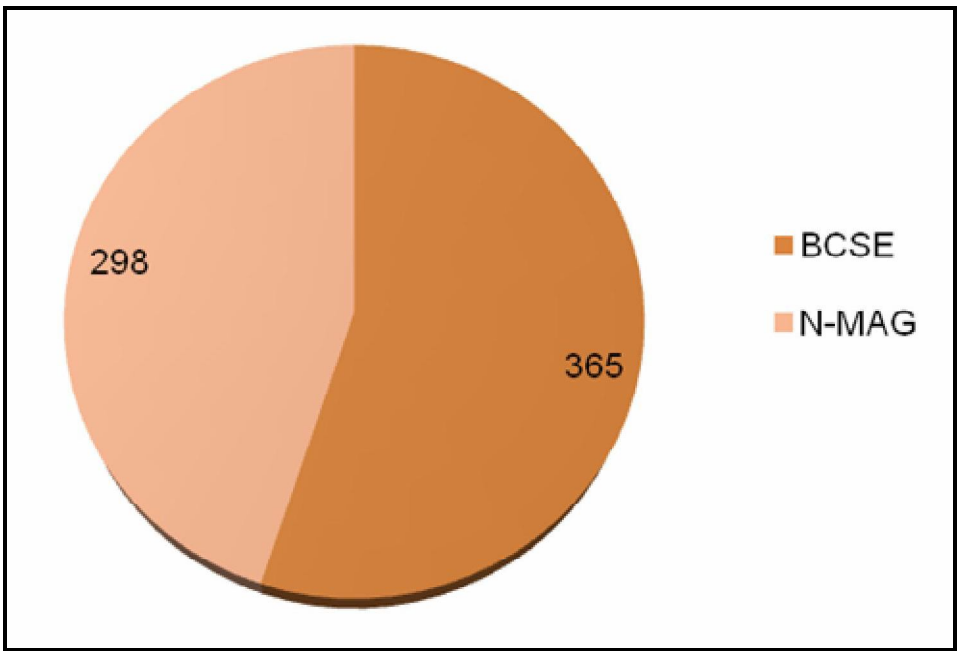

Figure 5.8 Comparison of power(mW)

5.8 CONCLUSION

In this chapter, an algorithm (N-MAG) has been proposed and implemented for designing FIR. The new algorithm aims to implement the multiplier block using fewest adders. Optimality proofs are provided with this algorithm in the sense that costs are minimum and list of fundamentals are exhaustive. The implementation results show a 18% power reduction. Further upto 51% reduction in area slices and 42% reduction in LUT are achieved.

The classic research community optimization metric of minimizing multiplier block adder cost has been demonstrated to minimize FPGA hardware for full-parallel pipelined FIR filters. Reducing flip-flop count through minimizing multiplier logic depth has been shown to yield the lowest area solutions. The new MAG algorithm has been defined to embody this design principle. The results presented establish a clear area advantage of N-MAG over prior algorithms for typical filter parameters .