Reconfigurable FPGA Implementation of FIR Filter using Modified DA Method

M. Backia Lakshmi[1], D. Sellathambi[2]

[1]PG Student, Department of Electronics and Communication Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu, India

[2]Assistant Professor, Department of Electronics and Communication Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu, India

Abstract: ***In signal processing, a digital filter is a system that performs mathematical operations on a sampled, discrete-time signal to reduce/enhance certain aspects of that signal. A digital filter usually contains of an ADC to sample the input signal, it is continued by a microprocessor and the peripheral components such as memory to store data and filter coefficients etc. In some of the high performance applications, instead of using a general purpose microprocessor, FPGA or ASIC can be used for expediting operations such as filtering. One type of digital filter is FIR filter which is stable and gives linear phase response. For an Nth order FIR filter, the generation of each output sample takes N+1 MAC operations. Memory based structures are well-suited for many DSP algorithms, which includes multiplication with a set of coefficients which remains fixed. For this purpose, Distributed Arithmetic architecture is used in FIR filter. Distributed arithmetic is one way to implement convolution without the multiplier unit, where the MAC operations can be replaced by a series of LUT access and summations. LUT are the kind of logic that used in DRAM based FPGAs. It is mainly used in the applications like Software Defined Radio (SDR), Digital up/down converters, Multi-Channel filters where the coefficients are changed during the run time. Hence LUT's are needed to be reconfigurable. This project achieves high-throughput by implementing the reconfigurable FIR filter using modified Distributed Arithmetic (DA) based approaches. For this implementation of reconfigurable FIR filter RAM based LUT's are used, and the implementation of such LUT's remains costlier. Thus a shared LUT design is proposed for reconfigurable FIR filter. Requirement of separate registers are eliminated by sharing the registers for different bit slices. Thus the DRAM based FIR filter reduces the number of bit slices. The proposed design is implemented in Xilinx Virtex-5 FPGA device (XC5VSX95T-1FF1136).***

Keywords: Modified Distributed Arithmetic (DA), Finite Impulse Response (FIR) Filter, Reconfigurable Implementation, Field Programmable Gate Array (FPGA), Look Up Table (LUT).

1. Introduction

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. It performs mathematical operations on a sampled or discrete time signal to reduce or enhance certain aspects of that signal. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution. LTI digital filters are generally classified as being finite impulse response (i.e., FIR), or infinite impulse response (i.e., IIR). As the name implies, an FIR filter consists of a finite number of sample values, reducing the above convolution sum to a finite sum per output sample instant. An IIR filter, however, requires that an infinite sum be performed.

The motivation for studying digital filters is found in their growing popularity as a primary DSP operation. Digital filters are rapidly replacing classic analog filters, which were implemented using RLC components and operational amplifiers. Analog filters were mathematically modeled using ordinary differential equations of Laplace transforms. They were analyzed in the time or *s* (also known as Laplace) domain. Analog prototypes are now only used in IIR design, while FIR are typically designed using direct computer specifications and algorithms [1]. An FIR with constant coefficients is an LTI digital filter. It is a stable filter. It gives linear phase response. It can be seen to consist of a collection of a "tapped delay line," adders, and multipliers. One of the operands presented to each multiplier is an FIR coefficient, often referred to as a "tap weight" for obvious reasons. Historically, the FIR filter is also known by the name "transversal filter," suggesting its "tapped delay line" structure. A perfectly linear-phase filter has a group delay that is constant over a range of frequencies. The symmetry properties intrinsic to a linear-phase FIR can also be used to reduce the necessary number of multipliers *L*. Recently, with the advent of software defined radio (SDR) technology, finite impulse response (FIR) filter research has been focused on reconfigurable realizations.

Distributed arithmetic is one way to implement convolution with multiplier less unit, where the MAC operations are replaced by a series of LUT access and summations. Distributed Arithmetic is a different approach for implementing digital filters [3]. The basic idea is to replace all multiplications and additions by a table and a shifter-accumulator. Basically each look up table is a bunch of single bit memory cells storing individual bit values in each of the cells. Distributed Arithmetic provides cost-effective and area-time efficient computing structures. The DA implementation of an FIR filter is particularly attractive for low-order cases due to LUT address space limitations. The outputs of a collection of low-order filters can be added together to define the output of a high-order FIR. To accelerate a DA filter, unrolled loops can be used. The input is applied sample by sample (one word at a time), in a bit-parallel form. In this case, for each bit of input a separate table is required. While the table size varies (input bit width equals number of filter taps), the contents of the tables are the same.

The DA implementation in ROM based LUT has fixed coefficients. Hence the LUT is fixed i.e. coefficients cannot be changed. The fixed LUT's require more memory space and delay. Thus the complexity of the system increases. To overcome this problem Modified DA method is implemented in RAM based LUT. Modified DA architecture is used to obtain an area time-power-efficient implementation of FIR filter in FPGA [2]. FIR filter coefficients can be changed dynamically in RAM based LUT. In this proposed method LUT's are shared by DA units which results in less memory space and delay.

2. DA Based Implementation of FIR filter

The distributed arithmetic is a method of computing the sum of products:

$$y(n) = \sum_{n=0}^{N-1} c[n] * x[n] \quad (2.1)$$

where, c[n] → coefficients; x[n] → variables.

In the existing approach, the ROM based LUT's are fixed. The LUT's which are fixed in the sense the coefficients stored in the LUT's cannot be changed during run time. The same coefficients are used for the full operation of FIR Filters.

It is useful to implement in the case of adder operations in which the multiply and Accumulate (MAC) operation is performed in addition to the multipliers. Hence it increases the cost of the process as well as the memory. So it increases the number of bit slices.

The FIR filter coefficients are dynamically changed during run time. The conventional DA implementation used for the implementation of an FIR filter assumes that impulse response coefficients are fixed, and this behavior makes it possible to use ROM-based LUTs. The memory requirement for DA-based implementation of FIR filters, however, exponentially increases with the filter order. Modified DA architecture is used to obtain an area time-power-efficient implementation of FIR filter in FPGA [5]. The mapping function is presented as a lookup table (DA-LUT) that includes all the possible linear combinations of the coefficients and the bits of the incoming data samples. In a modification of distributed arithmetic concept was proposed that allows decomposing DA-LUT into LUTs of sizes available in given FPGA architecture. For higher order filters, the size of the LUT also increases exponentially with the order of the filter. For a filter with N coefficients, the LUT have 2^N values. This in turn reduces the performance [6].

Therefore, for higher order filters, LUT size to be reduced to reasonable levels. To reduce the size, the LUT can be subdivided into a number of LUTs, called LUT partitions. Each LUT partition operates on a different set of filter taps. The results obtained from the partitions are summed. Suppose the length LK inner product, then equation,

$$y = \sum_{k=1}^{LK} A_k X_k \qquad (2.2)$$

Then the sum can be partitioned into L independent K^{th} parallel DA LUTs resulting in,

$$y = \sum_{I=0}^{L-1} [\sum_{n=0}^{N-1} X_{LI} + A_{LI+n}] \qquad (2.3)$$

For 3rd order filter

- Number of partition = 2
- 2 LUT tables are used. Each has 2 inputs.
- Memory location = no .of partition * 2^n = $2*2^2$ =8 location.
- n=number of inputs of LUT

A Reconfigurable FIR filter whose filter coefficients dynamically change during runtime plays an important role in the software defined radio systems, multichannel filters, and digital up/down converters [4].

3. Proposed Approach

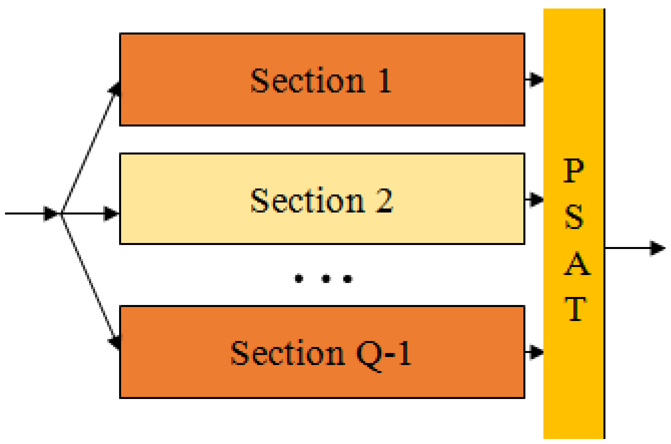


Figure 1: Overview diagram of proposed method

The number of sections depends upon the input bit width shows in figure 1. In figure 2 shows that the functional block diagram of proposed method.

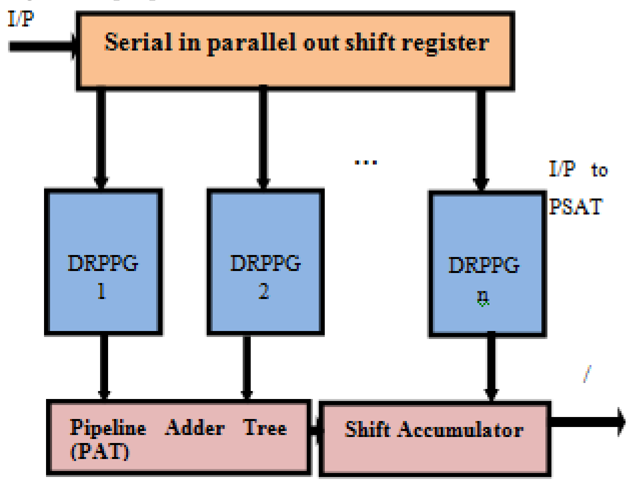

I/P
Serial in parallel out shift register
...
I/P to PSAT
DRPPG 1
DRPPG 2
DRPPG n
Pipeline Adder Tree (PAT)
Shift Accumulator

Figure 2: Functional block diagram

3.1 Shift Register

In digital circuits, a shift register is a cascade of flip flops which shares the same clock. The output of each flip flop is connected to the "data" input of the next flip flop in the chain. The resulting circuit shifts by one position the “bit array” stored in it, shifting in the data. More generally, a shift register may be multidimensional, such that it’s "data in" and stage outputs are themselves bit arrays: this is implemented simply by running several shift registers of the same bit-length in parallel.

Shift registers can have both parallel and serial inputs and outputs. These are often configured as 'serial-in, parallel-out' (SIPO) or as 'parallel-in, serial-out' (PISO). There are also types that have both serial and parallel input and types with serial and parallel output. There are also 'bidirectional' shift registers which allow shifting in both directions: Left to Right or Right to Left. The serial input and last output of a shift register can also be connected to create a 'circular shift register'.

3.2 Word shift register

In digital circuits, a shift register is a cascade of flip flops, sharing the same clocks, in which the output of each flip flop is connected to the “data” input of the next flip flop in the chain. It results in a circuit that shifts by one position the “bit array” in it. Here the input and filter coefficients are given separately in a serial bit by bit fashion.

A shift register basically consists of several single bit “D-Type Data Latches”, one for each data bit, either a logic “0” or a “1”. In which they are connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on. Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration. The number of individual data latches required to make up a single Shift Register device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches. In figure 3 shows MAC architecture using modified DA.

Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices. Shift register IC’s are generally provided with a *clear* or *reset* connection so that they can be “SET” or “RESET” as required.

3.3 Multiplier

The high speed multiplication operation plays vital part in Digital Signal Processor (DSPs) as well as in general processor. Finite Impulse Response (FIR) filter with higher speed is of great importance. FIR filter is also called convolution filter since convolution is the fundamental concept of designing FIR filter. In Digital signal processing, the multiply-accumulate operation is a common step that computes the product of two numbers and adds the product to an accumulator. The hardware unit which performs this operation is known as Multiplier-Accumulate unit or MAC unit.

3.4 Accumulator

Accumulator is a register which is used to hold the output of the ALU or Multiplier-Accumulate unit. DSP processors typically have from one to four accumulators. Several low-power design techniques have been applied to the design of a power efficient multiplier-accumulator (MAC) array. The MAC array is designed to have a programmable resolution so that the blocks corresponding to the least significant bits can be deactivated when a lower resolution is sufficient. A MAC unit consists of a multiplier and accumulator. MAC unit is an inevitable component in many digital signal processing (DSP) applications involving multiplications and/or accumulations.

MAC unit is used for high performance digital signal processing systems. The DSP applications include filtering, convolution, and inner products. The MAC inputs are obtained from the memory location and given to the multiplier block. The functionality of MAC unit enables high-speed filtering and other processing typical for DSP applications. Multiplication-and-accumulate operations are typical for digital filters. Therefore, the functionality of the MAC unit enables high-speed filtering and other processing typical for DSP applications. Since the MAC unit operates completely independent of the CPU, it can process data separately and thereby reduce CPU load.

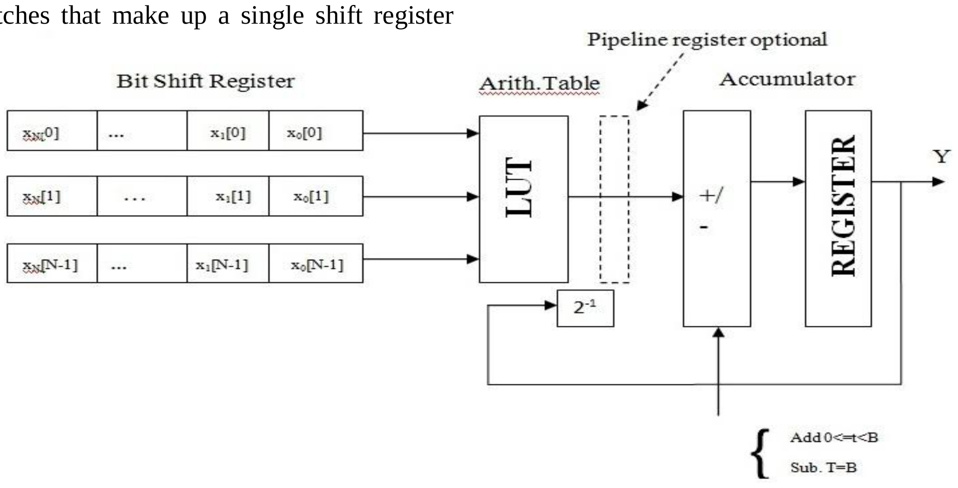


Figure 3: MAC architecture using modified DA

3.5 Pipeline registers

Pipeline registers preserve the result of a previous stage's execution so that the stage's hardware can be used for the next instruction. Enables are not needed for the registers since they are update on every clock cycle. Pipelining results in speed enhancement which reduces the power consumption.

3.6 Bit shift registers

Inputs are given as parallel bits simultaneously and coefficients are predefined in the LUT.

3.7 Look up table

Look up table is a basic building block of processor. This configuration is to reduce memory size, logic gate count and improve the speed of operation. Digital filters are becoming ever-present in audio applications. As a result, good digital filter performance is important to audio system design. Digital filters differ from conventional analog filters by their use of finite precision to represent signals and coefficients and finite precision arithmetic to compute the filter response.

The FPGA utilizes lookup tables (LUT) to implement multi-level functions in order to maximize node sharing in a Boolean network. Since the invention of FPGAs in the mid-1980s, Look-up-tables (LUTs) have been the basis of FPGA logic blocks. A K-LUT is a single-output memory with K address lines that can implement any Boolean function that uses up to K variables. The earliest FPGAs used 4-LUTs, established as the best LUT size to maximize area efficiency. The extra outputs can be added onto their LUTs which is a straightforward modification. Since the nature of a LUT's implementation in hardware, is a tree structure. The LUTs in modern FPGAs are reduced to smaller LUTs. In the look-up-table (LUT)-multiplier-based approach, the memory elements stores all the possible values of products of the filter coefficients which could be an area-efficient and this is an alternative to DA-based design of FIR filter.

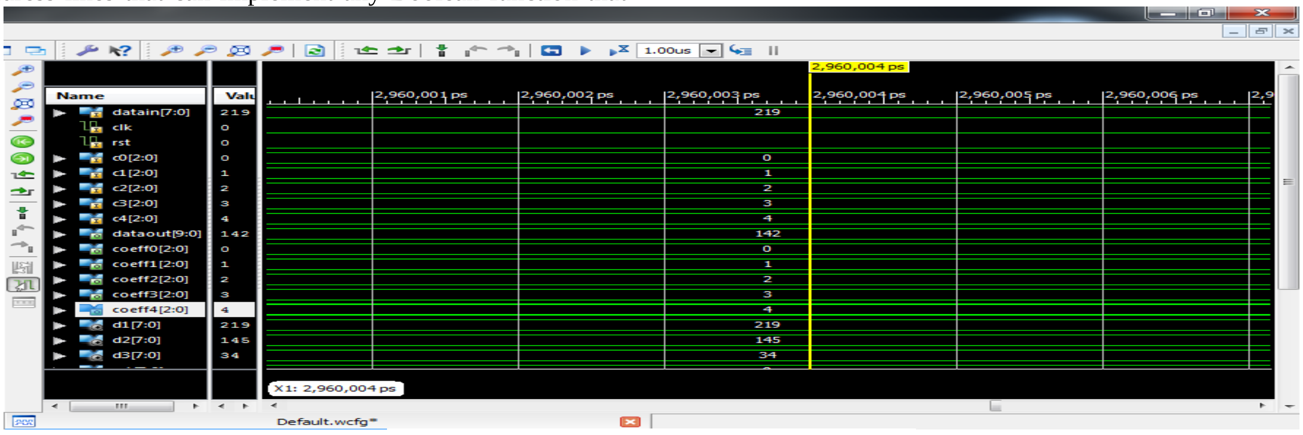
4. Performance Evaluation

Based on the evaluation done between ROM and RAM based Delay the proposed method (i.e) RAM based LUT's has more benefits. The area utilization has much reduced (nearly 20% reduced) for RAM based LUT's. And also the memory usage, delay consumption gets reduced. Simulation results for ROM and RAM Based LUT shown in figure 4 and 5.The comparative analysis shown in table 3.1.

Table 3.1. Performance Analysis

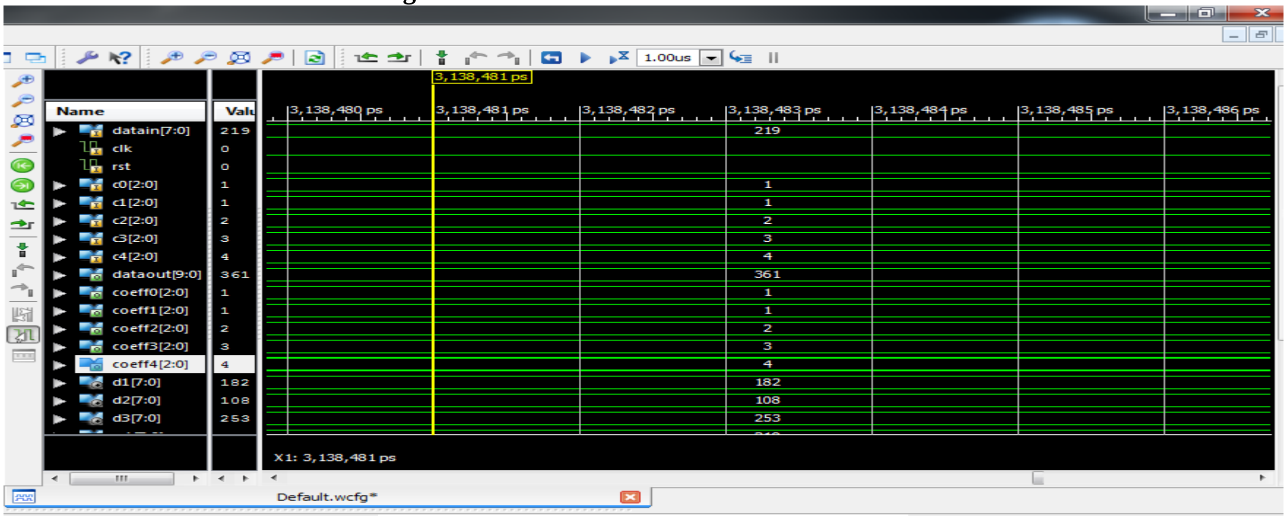
Content	*Existing (ROM LUT)*	*Proposed (RAM LUT)*
Area Utilization	65%	41%
Delay	14.530ns logic	12.944ns logic
Memory Usage	325148 Kilobytes	322844 Kilobytes

Here the FIR filter taken for result is 5tap. And the input bit width is 8.



Name	Valu
datain[7:0]	219
clk	0
rst	0
c0[2:0]	0
c1[2:0]	1
c2[2:0]	2
c3[2:0]	3
c4[2:0]	4
dataout[9:0]	142
coeff0[2:0]	0
coeff1[2:0]	1
coeff2[2:0]	2
coeff3[2:0]	3
coeff4[2:0]	4
d1[7:0]	219
d2[7:0]	145
d3[7:0]	34

Figure 4: Simulation result for ROM Based LUT



Name	Valu
datain[7:0]	219
clk	0
rst	0
c0[2:0]	1
c1[2:0]	1
c2[2:0]	2
c3[2:0]	3
c4[2:0]	4
dataout[9:0]	361
coeff0[2:0]	1
coeff1[2:0]	1
coeff2[2:0]	2
coeff3[2:0]	3
coeff4[2:0]	4
d1[7:0]	182
d2[7:0]	108
d3[7:0]	253

Figure 5: Simulation result for RAM Based LUT

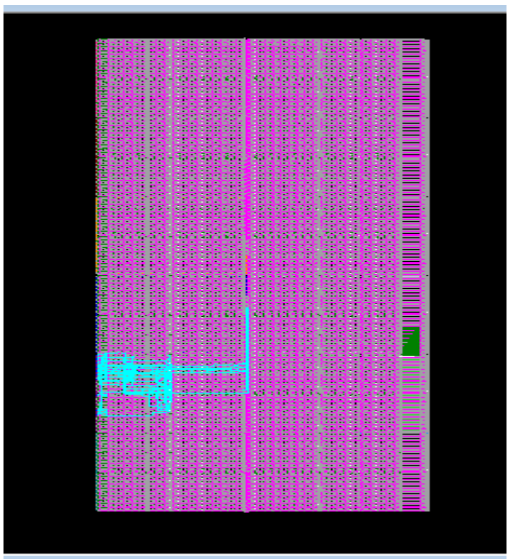


Figure 6: Floorplan Design View of ROM Based LUT

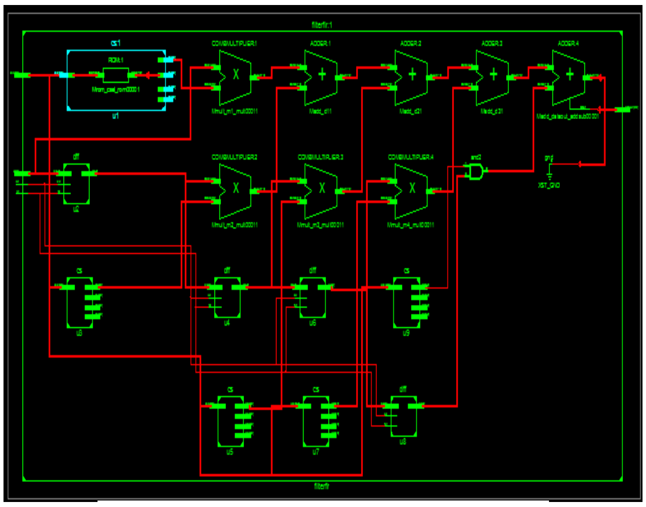


Figure 7: RTL View of ROM Based LUT

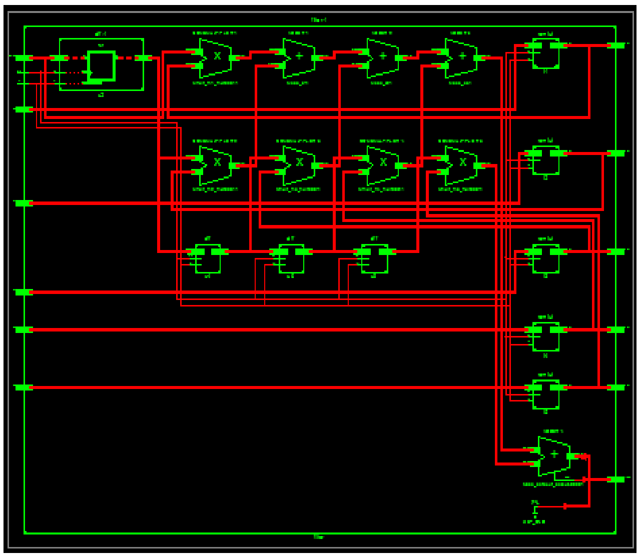


Figure 8: RTL view of RAM Based LUT

In figure 6 and 7 shows that the floorplan design view and RTL view of ROM based LUT. The RAM based LUT is best one compared with ROM based LUT. The RTL view and floorplan design view of RAM based LUT shown in figure 8 and 9.

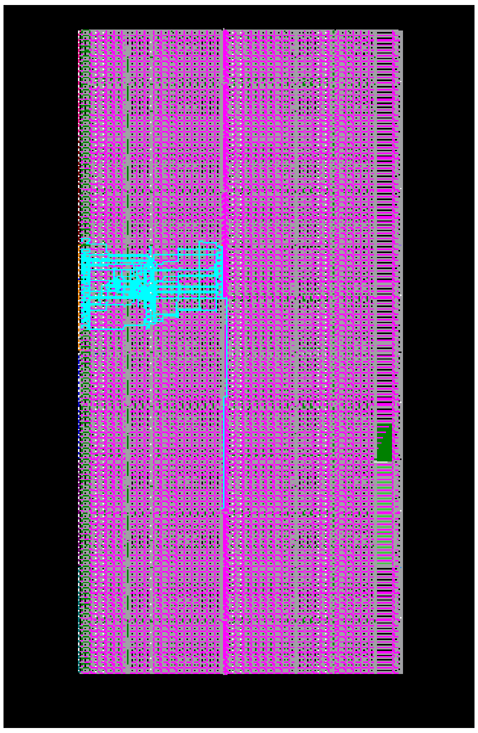


Figure 9: Floorplan Design View of RAM based LUT

5. Conclusion

Since ROM based LUT's reduces the complexity of multiplier structures, due to the linearity between coefficients and filter order, it increases the memory requirement. And also during the run time the FIR filter coefficients changing. For that we proposed the high throughput reconfigurable FIR filter using modified DA method. In reconfigurable DA based FIR filters the Look up Tables are implemented in RAM. A shared LUT design is suggested for the implementation of RAM which substantially reduces the hardware cost. Since RAM is an erasable one, it reduces the memory as well as bit slices (nearly 40%). The bit slice is a basic building block of a processor. The proposed approach supports beyond MHZ sampling rate. It is found to produce high throughput than the existing approach.

In future it can be implemented with fault toleration techniques. The idea is to show that parallel filters can be protected using error correction codes (ECCs) in which each filter is the equivalent of a bit in a traditional ECC. An error-correcting code is an algorithm for expressing a sequence of numbers such that any errors which are introduced can be detected and corrected (within certain limitations) based on the remaining numbers. Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors.

6. Acknowledgment

I would like to thank my supervisor, Department of Electronics and Communication Engineering, Parisutham Institute of Technology and science, Thanjavur, for his help and guidance to enable me to propose this paper.

References

[1] Jayasudha, N., Sathiya, K.G., "Reconfigurable Architectures for FIR filter with Low-Power Consumption", IEEE conf. Communication and Embedded syst, Feb 2013.

[2] M. Kumm, K. Moller, and P. Zipf, "Dynamically reconfigurable FIR filter architectures with fast reconfiguration," in Proc. 8th Int. Workshop ReCoSoC, Jul. 2013, pp. 1–8.

[3] P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in Proc. IEEE/IFIP 19th Int. Conf. VLSI-SOC, Oct. 2011, pp. 428–433.

[4] Dongwon Lee., Georgia Inst. of Technol., Atlanta"Reconfigurable and Area efficient Architecture for symmetric FIR filters with power of two coefficients, IEEE Conf. Nov. 2007.

[5] P. K. Meher, "Hardware-efficient systolization of DA-based calculation of finite digital convolution," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 53, no. 8, pp. 707–711, Aug. 2006.

[6] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 7, pp. 1327–1337, Jul. 2005.