International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

A Power Efficient Single Precision Floating Point Arithmetic Unit Architecture based on Reversible Logic

Athira Anilan[1], Sreelekshmy S[2], Dr. S. Krishna Kumar[3]

[1]Athira Anilan: Student, Dept. of ECE, FISAT, Kerala, India
[2]Sreelekshmy S: Assistant Professor, Dept. of ECE, FISAT, Kerala, India
[3]Dr. S. Krishna Kumar: Professor, Dept. of ECE, FISAT, Kerala, India

---

Abstract- *Reversible logic is a promising field of research that finds applications in low power computing, quantum computing, optical computing, and other emerging technologies. A floating-point unit (FPU) is a part of a computer system specially designed to carry out operations on floating point numbers. In this paper an arithmetic unit based on IEEE-754 standard for floating point numbers has been implemented on FPGA Board. Here Floating Point Unit (FPU) follows IEEE single precision format. Various arithmetic operations such as, addition, subtraction multiplication division, square root and bit shifting on floating point numbers have been performed on arithmetic unit.*

Key Words: **Digital Signal Processing, Reversible logic, Floating Point Arithmetic.**

1. INTRODUCTION

Very Large Scale Integrated (VLSI) systems are ubiquitous in everyone's day-to-day activities. This has been due to the advancement in process technology and progress in high density coupled with the innovation of faster and energy efficient devices. The major driving force behind this technology revolution over the last few decades is the exponential growth of transistor density within a single chip as described by Moore's law. Due to the singular characteristic of negligible standby power, CMOS process has evolved as the prominent VLSI technology after Bipolar since large numbers of transistors can be integrated within a chip. As device dimensions (channel length, gate oxide thickness, junction depth etc) shrink, to achieve high integration, the supply voltage and the threshold voltage of the device must be scaled accordingly. This however increases the dynamic power dissipation as well as the standby power exponentially. This exponential increase in power dissipation is not a desirable characteristic and hence there is a need for the development of novel low power circuit techniques to keep on with the progress of device dimension scaling. Energy dissipation is an important consideration in VLSI design as transistor density increases. The performance of high-end processors is mainly limited by power dissipation. Today's processors employ the conventional irreversible logic design which erases a bit of information, after every logic operation. Reversible logic has received great attention in the recent years due to their ability to reduce the power dissipation which is the main requirement in low power VLSI design. It has wide applications in low power CMOS and Optical information processing, DNA computing, quantum computation and nanotechnology. Irreversible hardware computation results in energy dissipation due to information loss. According to Landauer's research, the amount of energy dissipated for every irreversible bit operation is at least KTln2 joules, where K is the Boltzmann's constant and T is the temperature at which operation is performed [1]. The heat generated due to the loss of one bit of information is very small at room temperature but when the number of bits is more as in the case of high speed computational works the heat dissipated by them will be so large that it affects the performance and results in the reduction of lifetime of the components. In 1973, Bennett showed that KTln2 energy would not dissipate from a system as long as the system allows the reproduction of the inputs from observed outputs [2].

Reversible logic supports the process of running the system both forward and backward. This means that reversible computations can generate inputs from outputs and can stop and go back to any point in the computation history. A circuit is said to be reversible if the input vector can be uniquely recovered from the output vector and there is a one-to-one correspondence between its input and output assignments, i.e. not only the outputs can be uniquely determined from the inputs, but also the inputs can be recovered from the outputs Energy dissipation can be reduced or even eliminated if computation becomes Information-lossless. Two constraints for reversible logic synthesis are: feedback is not allowed, and fan-out is not allowed (i.e., fan-out = 1). A gate with k inputs and k outputs is called a k*k gate. Several reversible gates have been proposed over the last decades. Some parameters used to characterize Reversible Logic circuits are Garbage Output (In a reversible logic circuit, the output that cannot be used further for computation process is called as garbage output), Constant Input (The input that is added to the function to make it reversible is called constant input), Quantum Cost (The number of 1×1 and 2×2 gates used to implement the reversible logic circuit is called as the quantum cost).

Adiabatic circuits are low power circuits which use "reversible logic" to conserve energy. Adiabatic logic is a fine grained energy recovery technique used in low power digital circuits which conserves the energy by giving the stored

energy back to the supply. Reversible logic that employs adiabatic switching principle reduces the dynamic power dissipation which is the major contributor of total power dissipation. In addition to reducing the power dissipation, it also boosts operation speed. Thus, this work aims to design a Single Precision Floating Point Arithmetic Unit using Reversible gates and adiabatic logic.

2. BACKGROUND

2.1 Design of a power efficient reversible floating point arithmetic unit

The principal components of the proposed Reversible Floating Point Arithmetic unit includes a reversible floating point adder/subtractor, a reversible floating point multiplier, a reversible floating point divider, a reversible floating point square root and a reversible 32-bit barrel shifter. Fig 1 shows the principal components of the Reversible Floating Point Arithmetic Unit suitable for digital signal processing applications.

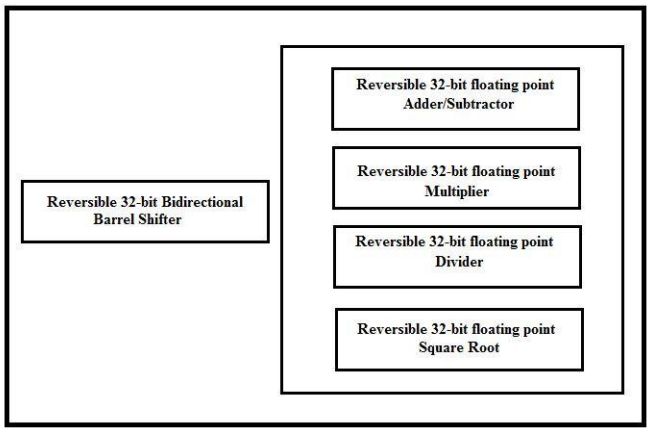

Fig-1: Principal components of reversible floating point arithmetic unit.

The algorithm for floating point multiplication is as follows:

- The exponents of the Multiplier (E1) and the multiplicand (E2) bits are added and the base value is subtracted from the added result. The subtracted result is put in the exponential field of the result block. The mantissa of the Multiplier (M1) and multiplicand (M2) are multiplied and the result is placed in the resultant field of the mantissa (truncate/round the result for 24 bits). i.e., M = Am * Bm
- S1 the signed bit of the multiplicand is XOR'd with the multiplier signed bit of S2. The result is put into the resultant sign bit.
- Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent.

The proposed design requires the implementation of a Reversible XOR for sign bit calculation, a Reversible Exponent Unit comprising of an 8-bit Reversible Propagate Generate Adder and an 8-bit Reversible Bias Subtractor, a 24x24 bit Reversible multiplier and a 47-bit Reversible Shift Register. The block diagram representation of the reversible Single Precision Floating Point Multiplier is shown in fig 2.

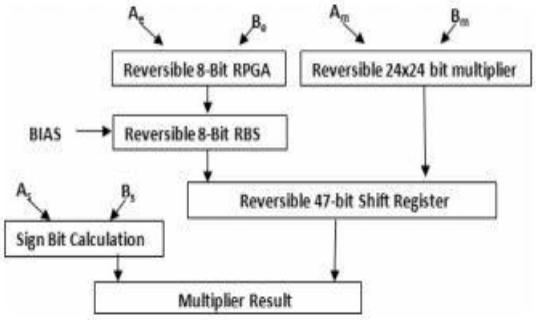

Fig- 2: Block Diagram of reversible Single Precision Floating Point Multiplier.

2.2 Design of a reversible floating point adder/subtractor

Floating Point Addition/Subtraction requires a reversible alignment unit, a reversible adder/subtractor unit, and a reversible normalization unit. Of the three units, the reversible alignment and the reversible normalization units have high quantum cost. Two works on reversible floating point adder reported that they have the limitation of high quantum cost. Hence, to bring down the quantum cost, an enhancement of these two units is proposed in this work by changing a few elements of the existing work, including the circuit design of the Reversible Leading Zero Detector [4].The algorithm for floating point addition/subtraction is as follows:

- Two numbers (say X1 and X2) can only be added if the exponents are the same i.e E1=E2.
- We assume that X1 has the larger absolute value of the 2 numbers. Absolute value of X1 should be greater than absolute value of X2, else swap the values such that Abs(X1) is greater than Abs(X2). Abs(X1)>Abs(X2).
- Initial value of the exponent should be the larger of the 2 numbers, since we know exponent of X1 will be bigger , hence Initial exponent result E3 = E1.
- Calculate the exponent's difference i.e. Exp_diff = (E1-E2).
- Left shift the decimal point of mantissa (M2) by the exponent difference. Now the exponents of both X1 and X2 are same.
- Compute the sum/difference of the mantissas depending on the sign bit.

If signs of X1 and X2 are equal (S1 == S2) then add the mantissas
If signs of X1 and X2 are not equal (S1!=S2) then subtract the mantissas.

- Normalize the resultant mantissa (M3) if needed. (1.m3 format) and the initial exponent.

The floating point addition/subtraction unit requires the implementation of an 8-bit reversible comparator, an 8-bit reversible subtractor, a 24-bit reversible shifter, a 24-bit reversible propagate generate adder/subtractor, a 24-bit reversible comparator, a 24-bit reversible subtractor and a 32-bit reversible leading zero detector.

The block diagram representation of reversible single precision floating point adder/subtractor is shown in fig 3.

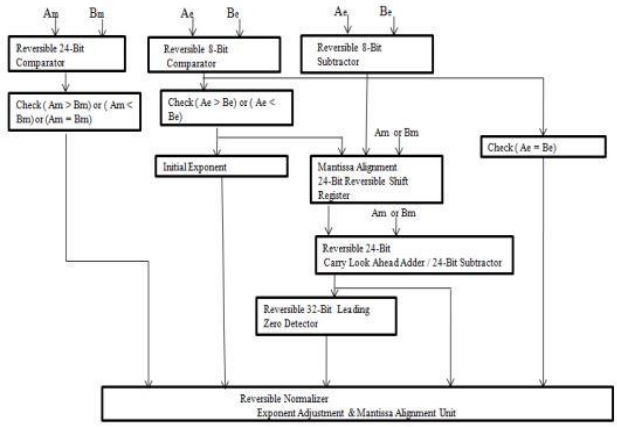

Fig-3: Block diagram representation of reversible single precision floating point adder/subtractor.

2.3 Design of reversible floating point square root

Square root operations are required in many complex DSP algorithms particularly in signal and image processing applications such as design of adaptive filters, in envelope detectors to recover the original signal in the demodulation process of a communication receiver etc. In comparison to the multiplication and division operations, the square root is a computationally intensive operation and has high latency. Existing square root units are based on the conventional irreversible logic and use fixed-point number representation. Due to these reasons, these designs have the drawbacks of power loss and lower dynamic range respectively. To avoid this drawback, a power and time efficient Reversible Single Precision Floating Point Square root is proposed using modified non-restoring algorithm.

The steps to compute floating point square root is as follows,

- Divide the significand 'n' into two groups exactly at the decimal point in both directions.
- From MSB, select one or two bits. (If 'n' is odd then select one bit, else two bits.)
- From the selected first group subtract '01'. If the result is positive, then root obtained is 1 else it is 0.
- Next append '01' and the root obtained to subtract from the remainder of previous stage.
- If the result of subtraction is negative, retain the previous remainder as the input for the next stage of subtraction and root is considered as 0, else consider the current difference as remainder and input for next stage and root as 1.
- Repeat steps 4 and 5 until the last group of two bits.

In the step 5, let the variable 'c' be used to represent borrow out (bout) and 'u' be the root. If the result of subtraction is negative then bout is 1. If bout is 1, u is 0 and input for the next stage is previous value of 'a' where 'a' represents the group of bits that has been selected for subtraction to subtract 01. If the result of subtraction is positive then bout is 0. If bout is 0, u is 1 and input for the next stage is the difference, d. From the step (5), it is clear that a Multiplexer is required as one block to determine one of the inputs to carry out the subtraction process for the next stage. Thus, the square root algorithm requires a Multiplexer and Subtractor as the basic building blocks. Hence, a Reversible Controlled-Subtract-Multiplex (RCSM) is proposed for the square root implementation. The symbolic representation of Reversible Controlled-Subtract-Multiplex block is shown in fig 4.

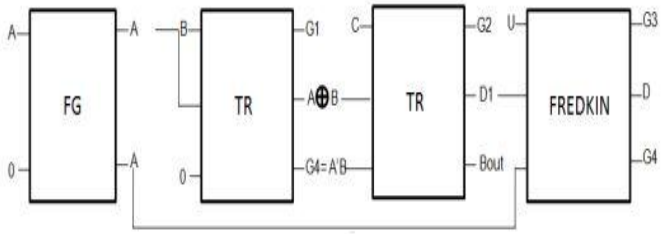

Fig- 4: Symbolic representation of the RCSM block.

The sign bit is the easiest bit to compute while calculating the square root of a number. If the sign of the input is positive, then the square root will be positive; if the sign of the input is negative, then there is no real square root. The exponent is represented in a biased form, where the actual exponent, e, is equal to the number formed by bits 23–30 minus 127. This means just a right shift of the exponent will not be sufficient. Thus, to compensate the bias effect, the right shifted exponent has to be added with a bias value of 127. In this work, the 23-bit significand is extended to 25-bits by pre-pending 01. As speed is the critical issue, the 25-bit is decomposed into four parts as a 12-bit, 6-bit, 4-bit and a 3-bit data. These four parts produce the first 14 bits of the significand. To produce the remaining bits, ten 1-bit square root stages are added by appending a zero in each stage. Finally, ten 1-bit square root stages are used in order to get the quotient Q [9 : 0] in which case one zero is appended in each stage to get the correct result.

For the realization of 6-bit reversible unsigned square root the input radicand be a 6-bit number N5N4N3N2N1N0. The output quotient be Q2Q1Q0. Fig 5 shows the realization of 6-Bit Reversible Unsigned Square root using RCSM. In Figure 5 N5 N4 N3 N2 N1 N0 represents the 6-bit unsigned input radicand while R4, R3, R2, R1 and R0 represents the remainder and U2, U1 and U0 represents the square root. Since the input is a 6-bit number, 3 stages are used to produce the roots U2, U1 and U0. RCSM represents the Reversible Controlled-Subtract–Multiplex. In similar way we can design 12-bit, 4-bit, 3-bit and 1-bit unsigned square root.

To increment the exponent, an 8-Bit Reversible Adder needs to be designed. To right shift the exponent, an 8-Bit Reversible Right Shifter is to be designed. To left shift the mantissa, a 25-bit Reversible Left Shifter is to be designed. To implement f(n), one 12-bit reversible unsigned square root, one 6-bit reversible unsigned square root, one 4-bit reversible unsigned square root, one 3-bit reversible unsigned square root and ten 1-bit reversible unsigned square root units are to be designed.

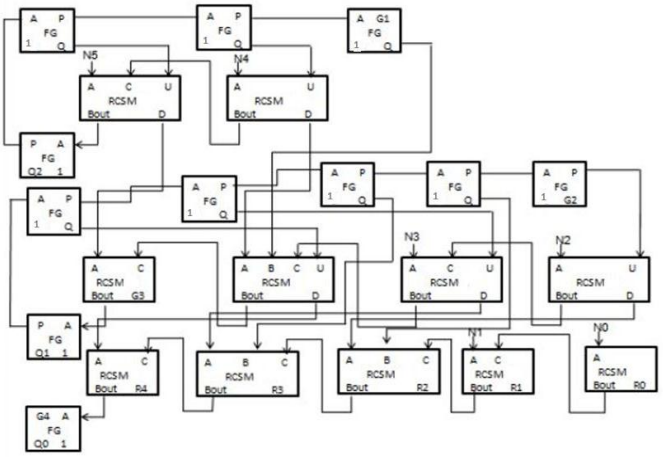

Fig-5: Realization of 6-Bit Reversible Unsigned Square Root

Fig 6 shows the parallel implementation of Reversible Single Precision Floating Point Square Root.

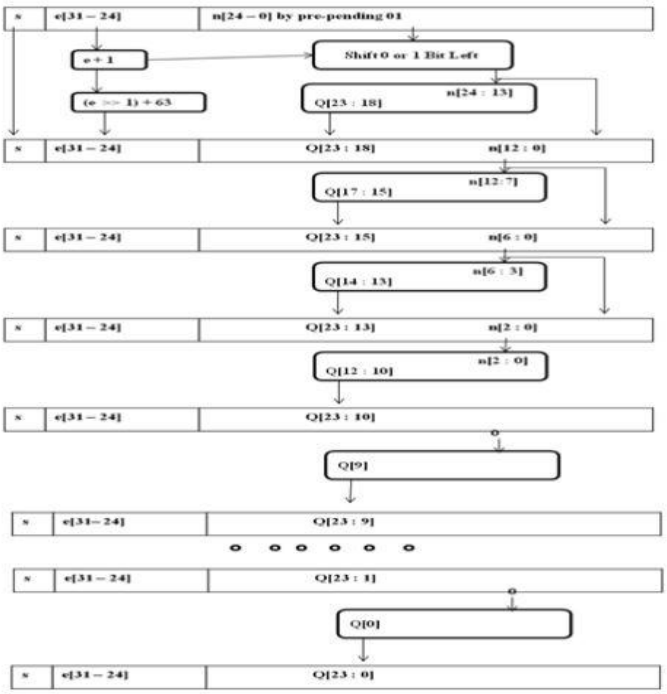

Fig- 6: Parallel implementation of reversible single precision floating point square root

2.4 Design of reversible floating point divider

Fast division methods start with a close approximation to the final quotient and produce twice as many digits of the final quotient on each iteration. Newton-Raphson and Goldschmidt fall in this category. Goldschmidt method is often characterized as non-self-correcting and more suitable for hardware implementation while Newton-Raphson method is characterized as self-correcting and more suitable for software implementation. Hence, Goldschmidt algorithm has been chosen in this work to implement an Efficient Reversible Floating Point Divider on FPGA with low latency. Goldschmidt algorithm was designed for the fractional part of the floating-point division. It relies on the availability of a very fast multiplier.

The basic Idea of this algorithm is assume Numerator (N) and Denominator (D) to satisfy the constraint $1 \leq N$ and $D < 2$ (considering them as normalized significands of floating point numbers). The basic division method is performed as follows Quotient Q = N/D. The Goldschmidt's algorithm points at finding a sequence K1, K2, K3…. Ki such that the product ri = D. K1. K2. K3. K4. K5……. Ki approaches 1 as i goes to infinity. Thus we have qi = N. K1.K2. K3 .K4.K5…….. Ki → Q. Fig 7 shows the basic idea of Godschmit's algorithm.

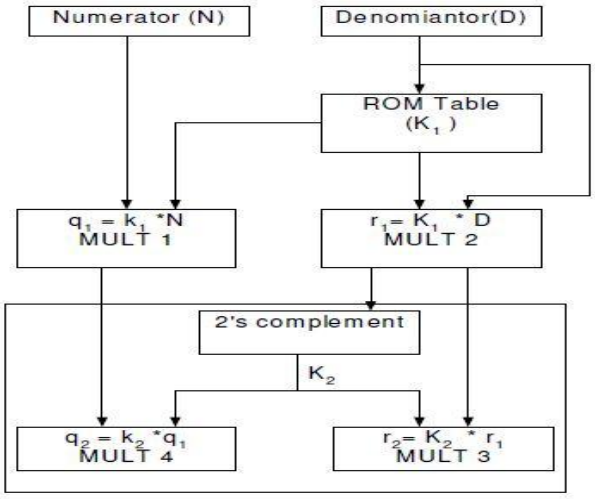

Fig- 7: Basic Idea of Goldschmidt's Algorithm

Based on Goldshmit's algorithm we can design a reversible floating point divider.

3. PROPSED ARCHITECTURE

3.1 Proposed reversible 32-bit bidirectional barrel shifter

In the existing literature, there is limited work on the design of reversible barrel shifters and the existing reversible barrel shifter designs are capable of a single operation i.e left rotation [7]. In this work we propose design methodologies for (n,k) reversible barrel shifters that are capable of performing different types of shift/rotate operations: the reversible logical right shifter supports the logical right shift operation, the reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate, the reversible bidirectional logical shifter supports logical right shift and logical left shift operations, the reversible bidirectional arithmetic and logical shifter supports logical right shift, arithmetic logical left shift and arithmetic left shift operations, the reversible universal bidirectional shifter that supports bidirectional logical shift, arithmetic shift and rotate operations.

The proposed design methodologies of reversible barrel shifters are capable of shift or rotate the input data in both direction and can also be used for logical or arithmetic shifting. The reversible design methodologies of barrel shifters are based on the Fredkin gate and the Feynman gate. The Fredkin gate is used as it can implement the 2:1 MUX with minimum quantum cost, minimum number of ancilla inputs and minimum number of garbage outputs. The Feynman gate is used to avoid the fanout as fanout is not allowed in reversible logic.

A reversible bidirectional arithmetic and logical shifter is capable can perform logical right shifting, arithmetic right shifting, logical left shifting and arithmetic left shifting operations. Figure 8 shows the diagram of an (n,k) reversible bidirectional barrel shifter. The chain of n/2 Fredkin gates controlled by the control signal left are added at the inputs and outputs of the (n,k) reversible universal right shifter. As can be seen in Fig 8 there is an additional Fredkin gate after the (n,k) reversible universal right shifter that is to preserve the sign bit in case of arithmetic left shift operation when the value of control signal sla is 1. To copy the sign bit of the input data for arithmetic right shift an additional Feynman gate is used as shown in figure 8. All the operations that can be performed by a (n,k) reversible universal bidirectional shifter are shown in Table 1.

Table- 1: Operations on (n,k) reversible universal bidirectional shifter

left	sra	rotate	Operation Performed
0	1	0	Reversible arithmetic right shift
0	0	1	Reversible right rotation
0	0	0	Reversible logical right shift
1	0	0	Reversible arithmetic left shift
1	0	1	Reversible left rotation
1	0	0	Reversible logical left shift

For left = 1, the (n,k) reversible universal bidirectional shifter will perform all type of left shifts such as logical left shift or arithmetic left shift or left rotate operation. When left = 0 the (n,k) reversible universal bidirectional shifter will work as (n,k) reversible universal right shifter that can perform the operation such as logical right shift or arithmetic right shift or right rotate operation. The working of (n,k) reversible universal bidirectional shifter can be explained as follows:

Stage I: In this stage, the input data in−1, in−2, in−3 ...., i2, i1, i0 is provided to the n/2 Fredkin gates which are controlled by the control signal left. When the value of control signal lef t is 1 the input data is reversed otherwise it remains same.

Stage II: The outputs generated by the n/2 Fredkin gates are used as the inputs of the (n,k) reversible universal right shifter. The (n,k) reversible universal right shifter performs the right shift (logical or arithmetic or rotate) on its inputs and generates the right shifted outputs (logical or arithmetic or rotate).

Stage III: The stage III consists of a single Fredkin gate and will be needed only if the arithmetic left shift operation is performed. The additional Fredkin gate used in the design as shown in Figure 8 helps in changing the 0th bit of the output generated by the (n,k) reversible universal right shifter based on the condition: lef t = 1 and sla = 1, to perform the arithmetic left operation. Thus, when the condition is satisfied the Fredkin gate will change the 0th bit of the output generated by the (n,k) reversible universal right shifter to the sign bit i_{n-1}. If this stage is used, then in the final stage that consists on n/2 Fredkin gates and performs

the operation of reversing the data, the outputs will be arithmetic left shifted result of the original inputs.

Stage IV: This stage consists of n/2 Fredkin gates as shown in Figure 8 which are controlled by the control signal left. For the value of control signal left = 1 the inputs passed to this stage are reversed by these n/2 Fredkin gates to produce the final output as logically left shifted input data or arithmetic left shifted input data or left rotated input data. Otherwise when left = 0, the final outputs will be same as the outputs of the Stage III which is logically right shifted input data or arithmetic right shifted input data or right rotated input data.

Fig 8 shows an (n,k) reversible bidirectional barrel shifter.

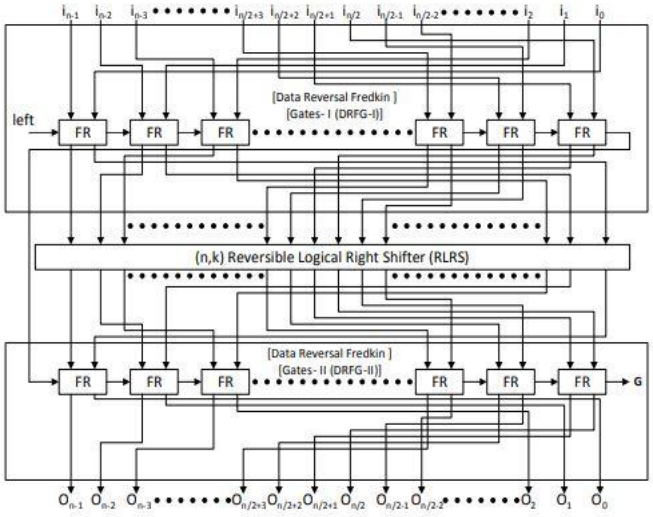

Fig- 8: An (n,k) reversible bidirectional barrel shifter.

3.2 Proposed reversible floating point multiplier

In this section, we present a new design of a 2*2 bit reversible Wallace tree multiplier to be used as a multiplier module in the design of 24x24 bit multiplier. The technique used here is called operand decomposition. The IEEE754 standard defines the format for floating point numbers of several levels of precision. This work focuses on the binary32 precision, commonly called single precision, which defines how a single floating point number is stored in 32 bits. The standard details three fields for a single precision floating point number: A single sign bit S, an 8 bit biased exponent E, and a 23 bit trailing significand T. The field E is regarded as an unsigned integer that represents the signed exponent of the floating point number with a bias of 127 applied. Normal floating point numbers are those that are normalized, use all available precision of the format, and have an implicit leading 1 added to trailing significand. Furthermore our design focuses on normal single precision floating point multiplication. In the existing work it requires more number of clock cycles to generate the output. But in this work the output is obtained within one clock cycle. To get the mantissa part, a reversible right shifter is used to obtain the result within one clock cycle .The block diagram representation of the proposed Reversible Single Precision Floating Point Multiplier is shown in fig 9.

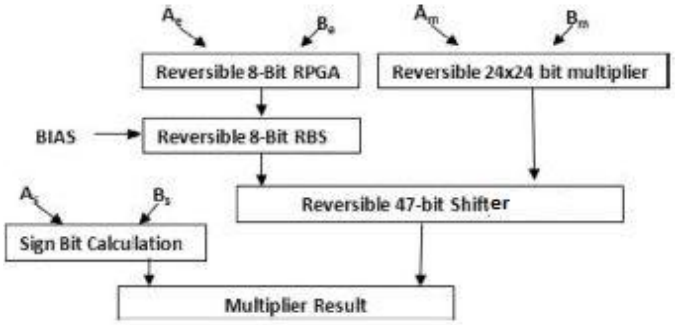

Fig- 9: Block diagram of the proposed reversible single precision floating point multiplier.

3.3 Proposed reversible single precision floating point Arithmetic unit

An ALU is a fundamental building block of many types of computing circuits, including the central processing_unit (CPU) of computers, FPUs, and graphics processing units (GPUs). Also, ALU is the basic building block of any computing system. In this section we present a design of a reversible single precision floating point Arithmetic and Logical unit composed of reversible floating point adder/subtractor, reversible floating point multiplier, reversible floating point divider, reversible floating point square root and a reversible floating point bidirectional barrel shifter. The architecture of a reversible single precision floating point arithmetic unit is given in fig 10.

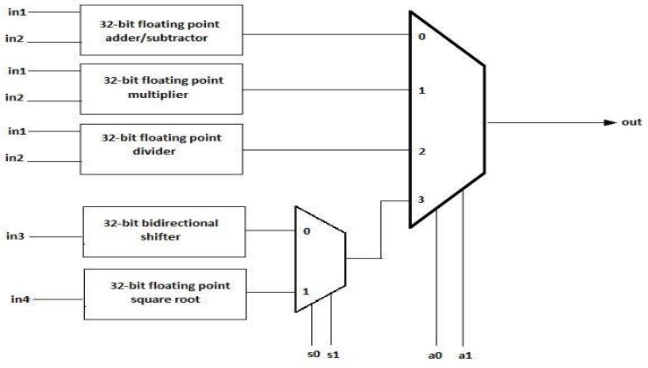

Fig- 10: Architecture of reversible single precision floating point arithmetic unit

3.4 Proposed reversible single precision floating point MAC unit

The real time digital signal processing applications are greatly extended by the advancements in VLSI (Very Large Scale Integrated Circuit) technology. As a part of digital

signal processing, the FIR (Finite Impulse Response) filter has so many applications especially it is well-suited for elimination of PLI. Power line Interference is the most common type of noise in the ECG signal caused by absorbing the electromagnetic radiation by the human body from 50Hz frequency power lines. Low power and high speed filtering is essential to eliminate the noise from biomedical raw signals and make the monitoring device portable. Basically the FIR filter consists of multiplier and an accumulator that contains the sum of the previous consecutive products. A digital athematic unit called MAC can also perform the same operation of multiplication and accumulation. In this work, we designed a MAC unit with the proposed reversible floating point multiplier and reversible floating point adder. The block diagram of a MAC unit consists of a multiplier, an adder and an accumulator. The accumulator is an accumulation register (we use PIPO register). This PIPO register is used to store data as shown in fig 11. Parallel In Parallel Out register uses D flip flop. Thus this PIPO will be acting as the accumulator in MAC unit and as a delay unit when considering filter.

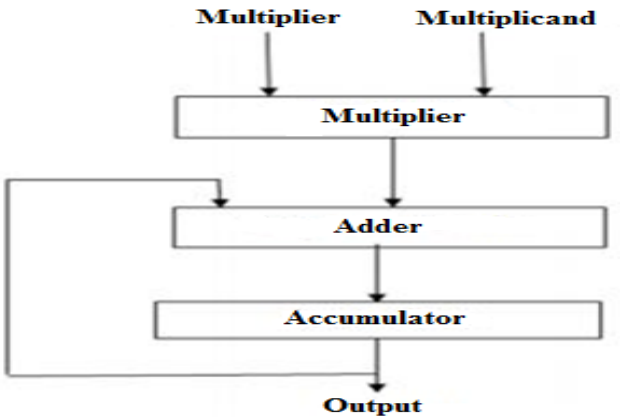

Fig- 11: Block diagram of MAC unit

In this proposed work, the designed MAC unit is reused at each tap of the filter based on allocation of a specific time slot for that tap. So the hardware requirement is greatly reduced by using a single MAC instead of multiple units. The filter can be implemented in many ways depending on the number of multipliers and accumulators available. In this paper we have implemented using a Single MAC Based FIR Filter unit. The block diagram is shown in fig 12 which consists of one multiplexers and Single MAC Based FIR Filter unit. The multiplexer is used to select only one input at a time which is fed to the multiplier at a given time. Here we consider x (n) as a one input and h (n) has 32 coefficients, so we used 32:1 mux of two quantities. These multiplexers select first one sample i.e. x (n) and first coefficients h (0) applies to MAC unit. A MAC unit is a single bit MAC unit. The output of this will be saved in accumulator which will be wide bits. In the next clock cycle it selects sample x (n) and next coefficient h (1) and performs MAC operation on these inputs. So this will be apply for all the bits one by one and final output will be y (n) which is saved in accumulator. The multiplexer is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Generally, the selection of each input line in a multiplexer is controlled by an additional set of inputs called control lines and according to the binary condition of these control inputs, either HIGH or LOW the appropriate data input is connected directly to the output. Normally, a multiplexer has an even number of 2n data input lines and a number of control inputs that correspond with the number of data inputs. The main advantage of using Single MAC Based FIR Filter unit is that it provides a less delay compared to that of multiple MAC's which are used in MAC based FIR filter. The power reduction is achieved through the usage of a MAC unit inside the filters that reduce the total activity and therefore the dynamic power.

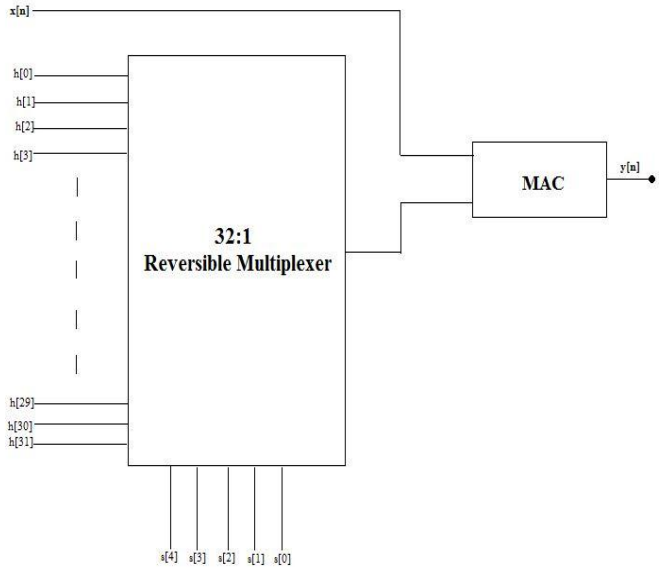

Fig- 12: MAC based FIR filter

4. IMPLEMENTATION RESULTS AND DISCUSSIONS

4.1 Proposed reversible 32-bit bidirectional barrel shifter

The functionality of the proposed reversible 32-bit Barrel Shifter is verified by using Cadence Nc-launch. Fig 4.1.1and 4.1.2 shows the simulation result of the proposed Reversible 32-bit barrel shifter (Arithmetic and Rotation).

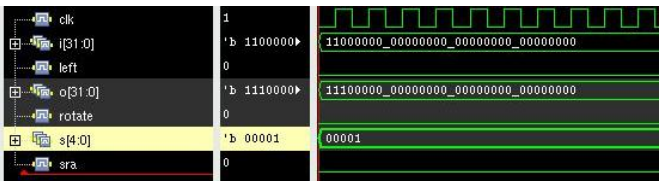

Fig-4.1.1: Output waveform of proposed arithmetic barrel shifter.

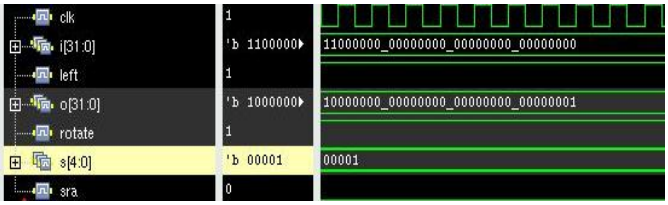

Fig- 4.1.2: Output waveform of proposed rotational barrel shifter

Fig 4.1.3 shows the schematic diagram of proposed 32-bit Bidirectional barrel shifter.

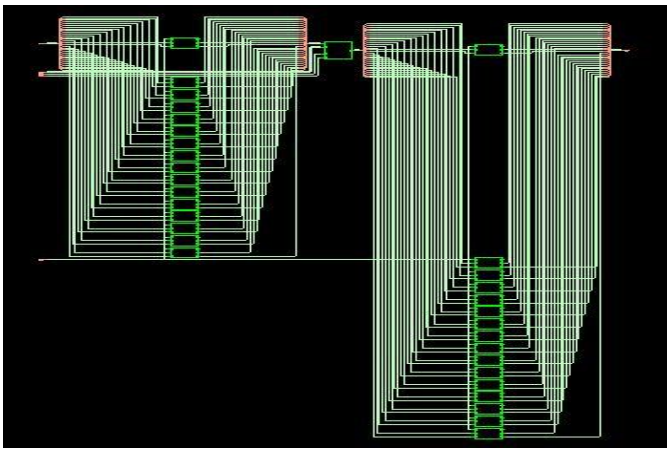

Fig - 4.1.3: Schematic diagram of proposed bidirectional barrel shifter.

4.2 Proposed reversible floating point multiplier

The functionality of the proposed reversible single precision floating point multiplier is verified by using Cadence Nc-launch. Fig 4.2.1 shows the simulation result of the proposed Reversible Single Precision Floating Point Multiplier.

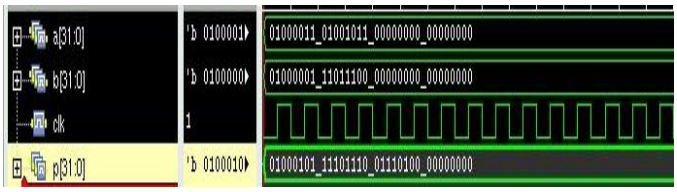

Fig - 4.2.1: Output waveform of proposed reversible floating point multiplier

Fig 4.2.2 shows the schematic diagram of proposed reversible floating point multiplier.

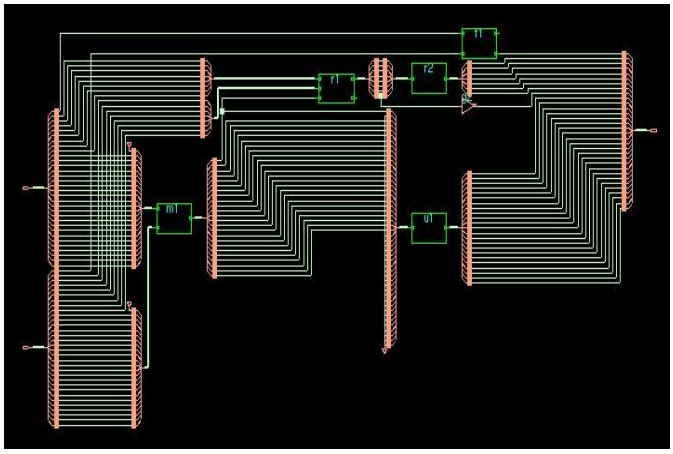

Fig - 4.2.2: Schematic diagram of proposed reversible floating point multiplier.

4.3 Proposed reversible single precision floating point arithmetic unit

The functionality of the proposed reversible single precision floating point arithmetic unit is verified by using Cadence Nclaunch. Figure below shows the simulation result of the proposed Reversible Single Precision Floating Point Arithmetic Unit which includes reversible floating point adder/subtractor, reversible floating point multiplier, reversible floating point divider, reversible floating point square root, reversible floating point barrel shifter.

Fig 4.3.1 shows the output waveform of proposed reversible floating point adder/subtractor.

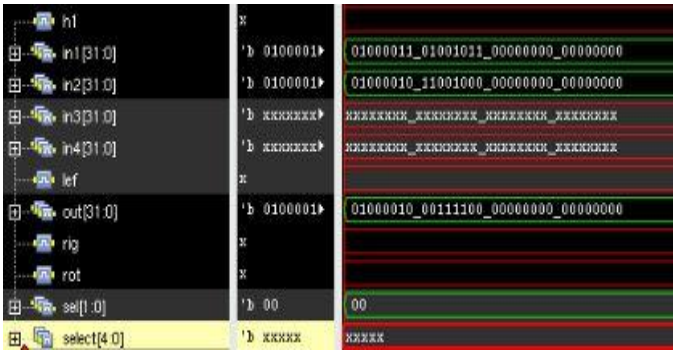

Fig - 4.3.1: Output waveform of proposed reversible floating point adder/subtractor.

Fig 4.3.2 shows the output waveform of proposed reversible floating point multiplier.

Fig - 4.3.2: Output waveform of proposed reversible floating point multiplier.

Fig 4.3.3 shows the output waveform of proposed reversible floating point divider.

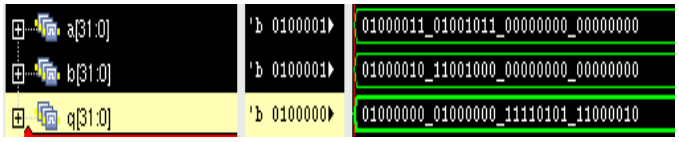

Fig- 4.3.3: Output waveform of proposed reversible floating point divider.

Fig 4.3.4 shows the schematic diagram of proposed reversible floating point divider.

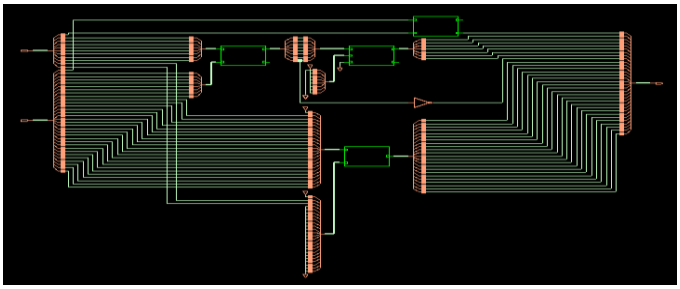

Fig - 4.3.4: Schematic diagram of proposed reversible floating point divider.

Fig 4.3.5 shows the output waveform of proposed reversible floating point shifter.

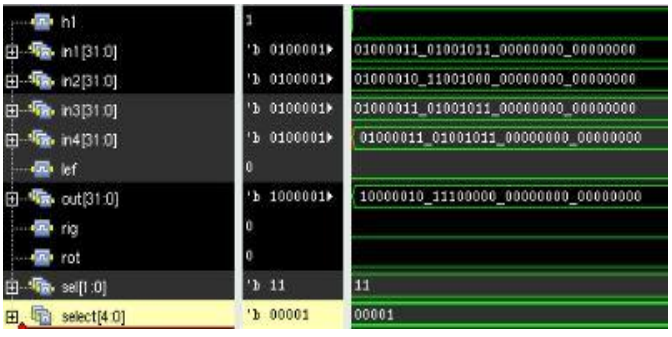

Fig- 4.3.5: Output waveform of proposed reversible floating point shifter.

Fig 4.3.6 shows the schematic diagram of the proposed Reversible Single Precision Floating Point Arithmetic Unit.

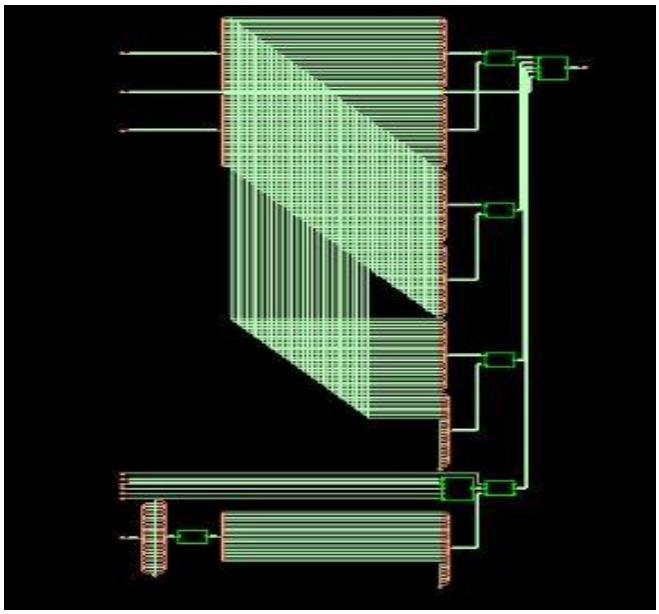

Fig- 4.3.6: Schematic diagram of the proposed Reversible Single Precision Floating Point Arithmetic Unit.

4.4 Proposed reversible single precision floating point MAC unit

The proposed Reversible Single Precision Floating Point MAC unit is simulated using Cadence NCLaunch. Fig 4.4.1 shows the simulation result of the proposed Reversible Single Precision Floating Point MAC unit.

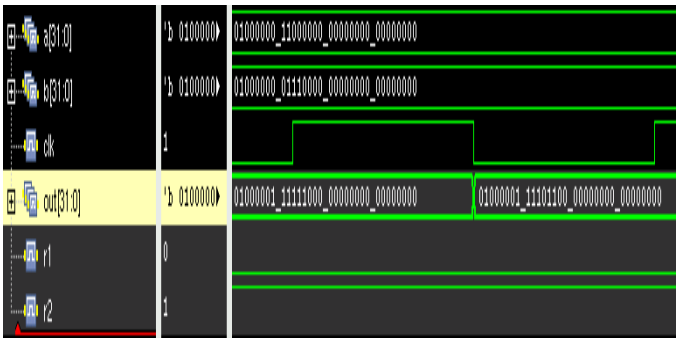

Fig- 4.4.1: Output waveform of reversible floating point MAC unit.

Figure 4.4.2 shows the schematic diagram of the proposed Reversible floating point MAC unit.

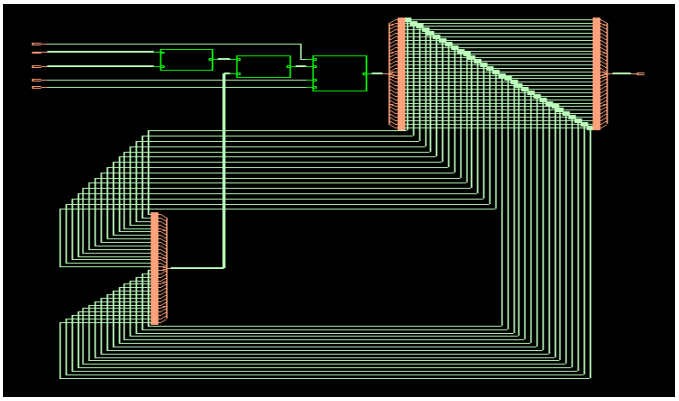

Fig- 4.4.2: Schematic diagram of reversible floating point MAC unit.

4.5 Proposed reversible floating point MAC based FIR filter

The proposed Reversible Single Precision Floating Point MAC based FIR filter is simulated using Cadence NCLaunch. Figure 4.5.1 shows the simulation result of the proposed Reversible Single Precision Floating Point MAC based FIR filter.

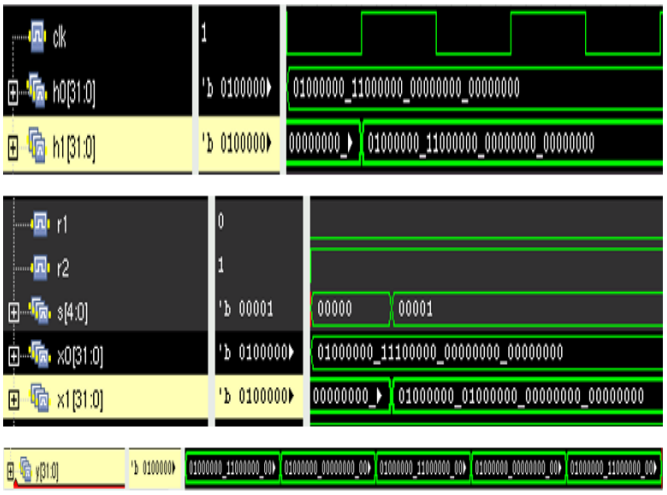

Fig- 4.5.1: Output waveform of reversible floating point MAC based FIR filter.

Figure 4.5.2 shows the schematic diagram of the proposed Reversible floating point MAC based FIR filter.

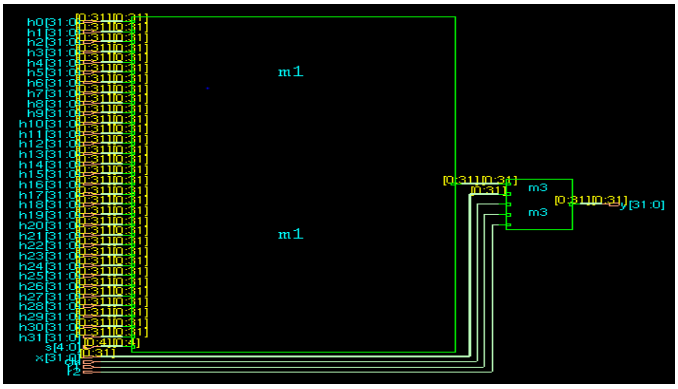

Fig- 4.5.2: Schematic diagram of reversible floating point MAC based FIR filter.

The performance of the Proposed Reversible Arithmetic unit is analyzed in terms area and power of the particular operation namely, addition/subtraction, multiplication, division, bidirectional barrel shifter and square root. The area and power consumption of the proposed reversible modules is generated by Cadence Encounter RTL tool. Table 1 shows the area and power consumption of the proposed reversible modules.

Table- 2: Area and power of proposed reversible arithmetic modules

Modules	Tech	Power (mW)	Area (μm²)
Reversible floating point adder/subtractor	180nm	1.58	17251
Reversible floating point multiplier	180nm	6.03	85355
Reversible floating point divider	180nm	3.63	65879
Reversible bidirectional barrel shifter	180nm	0.36	6799
Reversible floating point square root	180nm	0.21	4314
Reversible floating point arithmetic unit	180nm	12.18	182506

To make the design versatile for DSP applications, a MAC unit is designed with the proposed reversible modules. By utilizing the reversible floating point multiplier and reversible floating point adder, the proposed reversible floating Point MAC unit consumes less power. Using the proposed MAC unit, a reversible floating point MAC based FIR filter is implemented. The area and power consumption

of the proposed reversible floating point MAC unit and reversible MAC based filter is generated by Cadence Encounter RTL tool. Table 2 shows the area and power consumption of the proposed reversible floating point MAC unit and reversible MAC based filter.

Table- 3: Area and power of proposed reversible MAC unit and MAC based filter.

Modules	Tech	Power (mW)	Area (μm^2)
Reversible floating point MAC unit	180nm	9.55	115024
Reversible MAC based FIR filter	180nm	14.55	138405

An efficient 32 bit Reversible Single Precision Floating Point Arithmetic unit and a MAC unit as per IEEE 754 standard suitable for DSP applications has been proposed, designed and implemented. The proposed reversible single precision floating point MAC unit is used for the application of MAC based FIR filter. Simulation results and comparative studies with existing arithmetic unit modules indicate that the proposed modules are computational efficient, consume less power.

6. CONCLUSIONS

Reversible logic circuits preserve information as they compute, and thereby offer a promising design paradigm for low-power computing. In traditional logic circuits switching activity dissipates at least kTln2 Joules of heat every time a bit of information is lost, and since reversible logic circuits seek to minimize such information loss, they can push the boundary of low-power computing. This thesis proposes power efficient design of a 32- bit single precision floating Point Arithmetic Unit based on reversible logic optimized for Digital Signal Processors. On comparison with the existing floating point arithmetic operations using conventional logic it is inferred that the proposed reversible floating point arithmetic operations have the advantages of lower power dissipation. This is solely due to the circuit implementation with Reversible logic circuits employing adiabatic logic. Further, using the proposed arithmetic units a MAC unit is designed. By utilizing the proposed reversible single precision floating Point MAC unit, a MAC based FIR filter is designed. Thus, the proposed design is computationally as well as power efficient.

My objective for future work is to design an ADC can be designed which needs an implementation of a counter. By using reversible logic and gates we can design an ADC. Logic unit is also a fundamental unit for computing systems. Thus, incorporating a reversible Logic unit in the architecture has to be considered and analyzed in detail in future proposals. CORDIC implementation plays a crucial role in Digital Signal Processing algorithms. A reversible CORDIC architecture may be included in the proposed architecture to realize trigonometric functions.

ACKNOWLEDGMENT

The authors would like to thank Dr. S Krishna Kumar for his valuable discussions and support.

REFERENCES

[1] R. Landauer, “Irreversibility and heat generation in the computing process,” IBM J. Research and Development, vol.5, no. 3, pp. 183-191, 1961.

[2] C. H. Bennett, “Logical reversibility of computation,” IBM J. Research and Development, vol.17, no. 6, pp. 525-532, 1973.

[3] IEEE standard for binary floating-point arithmetic, New York ANSI/IEEE Std. 754, 1980.

[4] N. TrungDuc, M. Rodney Van, A space-efficient design for a reversible floating point adder in quantum computing, J. Quantum Phys. (2013) arXiv:1306.3760 [quant-ph].

[5] A.V. AnanthaLakshmi, Gnanou Florence Sudha, “A novel power efficient 0.64-GFlops fused 32-bit reversible floating point arithmetic unit architecture for digital signal processing applications”, Microprocessors and Microsystems, vol.51, pp.366–385 , Jan 2017.

[6] L.Yamin, C. Wanming,“ A new non-restoring square root algorithm and its VLSI implementation”, Proceedings of the IEEE International Conference, 1996, pp. 538–544 .

[7] K. Saurabh, H. Thapliyal, N. Ranganathan, “Design of a reversible bidirectional barrel shifter”, Proceedings of 11th IEEE International Conference on NanoTechnology, 2011, pp. 463–468.

[8] Inwook Kong and Earl E. Swartzlander, “A Goldschmidt Division Method With Faster Than Quadratic Convergence”, IEEE Transactions on Very Large Scale Integration (VLSI) SYSTEMS, VOL. 19, NO. 4, APRIL 2011.