International Journal of LNCT, Vol 2(7)
ISSN (Online): 2456-9895

Efficient VLSI architecture for combinational and sequential circuits with the help of serial and parallel techniques

Kirti Dhakad[1] and Shraddha Shrivastava[2]
M. Tech. Scholar, LNCT, Bhopal[1]
Assistant Professor, LNCT, Bhopal[2]

Abstract

Evolution of combinational and sequential circuit in very high scale integrated circuit hardware description language (VHDL) and evaluates its performance with respect to logic speed, number of slice and maximum frequency. Structural implementation of linear feedback shift register, counter and XOR/ XNOR gate in VHDL is configurable in terms of different parameter. The target device used for implementation of combinational and sequential circuit is Xilinx software with Spartan-3 device family. The output waveforms and timing report are also discussed.

Keywords

Linear feedback shift register, Counter, XOR/ XNOR gate, Maximum frequency.

1.Introduction

A linear feedback shift register is a combination of series of flip flop and XOR or XNOR logic gates. Its output is pseudo randomly cycle through a sequence of binary values after certain number of clock cycle [1]. The repetition of random output depends on the number of stages in the LFSR. Therefore, it is an important component in communication system where, it play important role in various application such as cryptography application, CRC generator and checker circuit, gold code generator, for generation of pseudorandom sequence, for designing encoder and decoder in different communication channels to ensure network security, Design for Test (DFT) and Built in Self-Test design (BIST).

A linear feedback shift register is linear in the sense that its input bit is a linear function (XORing or XNORing) of LFSR previous state [2].

The main challenging areas in VLSI are performance, cost, testing, area, reliability and power. The demand for portable computing devices and communications system are increasing rapidly. These applications require low power dissipation for VLSI circuits. The power dissipation during the test mode is 100% more than in normal mode. Hence it is important aspect to optimize power during testing. Power optimization is one of the main challenges. Linear feedback shift registers have multiple uses in digital systems design. Here we have implemented a 32 bit length sequence on FPGA using VHDL with maximum length feedback polynomial to understand the memory utilization and speed requirement. Also, we have presented the comparison of performance analysis based on synthesis and simulation result as well identify the simulation problem for long bit LFSR. The target device we have used Xilinx Spartan 3A and performed simulation and synthesis using Xilinx ISE. The HDLs are VHDL and Verilog. We prefer VHDL for programming because it is widely used.

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All flip-flops are driven by a common clock, and all are set or reset simultaneously. In these few lectures, the basic types of shift registers are studied, such as Serial In - Serial Out, Serial In - Parallel Out, Parallel In-Serial Out, Parallel In-Parallel Out, and bidirectional shift registers. A special form of counter - the shift register counter, is also introduced.

2.Design synthesis flow diagram

Synthesis is the process that reduces and optimizes the HDL or graphical design logic.

Some third-party synthesis tools are available as a part of the FPGA vendor's complete development package. Simplicity's Simplify and Mentor Graphics' Leonardo Spectrum, Precision RTL and Precision Physical are some examples of third-party synthesis tools.

Xilinx offers ISE Project Foundation, which is a complete development application that includes a synthesis tool [3-8].

Although some FPGA vendors offer synthesis, they still recommend using a third-party's synthesis tools. The synthesis tool must be set up prior to actually synthesizing the design. The synthesis process takes this information and the user-defined constraints and produces the output netlist. A constraints file specifies information like the critical signal paths and clock speeds. Synthesis can begin after completing set-up. General synthesis flow for tools involves three steps: creating structural element, optimizing, and mapping. *Figure 1* shows a synthesis flow diagram.

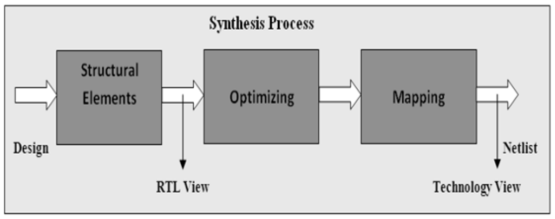

Figure 1 Design synthesis flow diagram

The first step in the synthesis process takes the HDL design and compiles it into structural elements. The next step involves optimizing the design, making it smaller and faster by removing unnecessary logic and allowing signals to arrive at the inputs or output faster. The goal of the optimizing process is the make the design perform better without changing the circuit's functions. The final step in the synthesis process maps or associates the design to the vendor specific architecture [9-15]. The mapping process takes the design and maps or connects it using the architecture of the specific vendor. This means that the design connects to vendor specific components such as look-up tables and registers. The optimized netlist is the output of the synthesis process. This netlist may be produced in one of several formats. Edif is a general netlist format accepted by most implementation tools, while '.xnf' format is specific to Xilinx and is only recognized by Xilinx's implementation. In addition to the optimized netlist, many synthesis tools like Simplify will produce a netlist for gate-level simulation and other report files. Stimulus applied to this netlist instead of the original HDL design produces the functional-level simulation, which lets the designer verify that the synthesis process hasn't changed the design's functions.

3.Proposed methodology

A. Liner feedback shift register

This sequential device loads the data present on its inputs and then moves or "shifts" it to its output once every clock cycle, hence the name Shift Register. 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", 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. 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. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

- Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- Parallel-in to parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

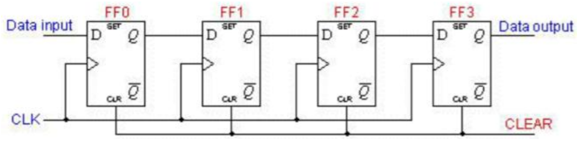

Figure 2 Flow diagram of serial in serial output shift register

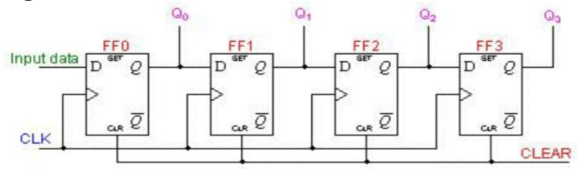

Figure 3 Flow Diagram of Serial in Parallel Output Shift Register

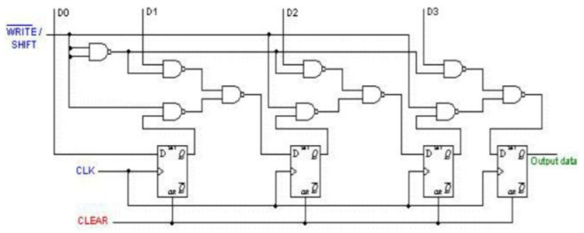

Figure 4 Flow diagram of parallel in serial out shift register

A four-bit parallel in - serial out shift register is shown below. The circuit uses D flip-flops and NAND gates for entering data (i.e. writing) to the register.

D0, D1, D2 and D3 are the parallel inputs, where D0 is the most significant bit and D3 is the least significant bit. To write data in, the mode control line is taken to LOW and the data is clocked in. The data can be shifted when the mode control line is HIGH as SHIFT is active high. The register performs right shift operation on the application of a clock pulse, as shown in the table below.

Table 1 Application of a clock

	Q_0	Q_1	Q_2	Q_3	
Clear	0	0	0	0	
Write	1	0	0	1	
Shift	1	0	0	1	
	1	1	0	0	1
	1	1	1	0	01
	1	1	1	1	001
	1	1	1	1	1001

For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The following circuit is a four-bit parallel in - parallel out shift register constructed by D flip-flops.

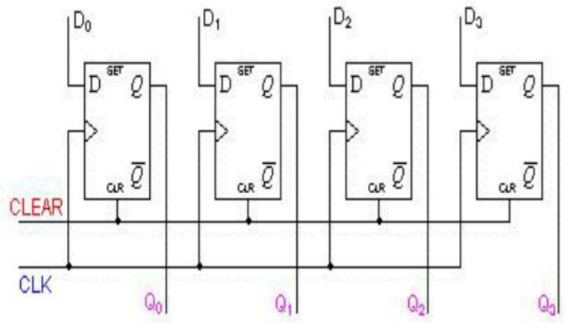

Figure 5 Flow diagram of parallel in parallel out shift register

B. Counter

This sequential device loads the data present on its inputs and then moves or "shifts" it to its output once every clock cycle, hence the name Shift Register.
A counter basically consists of several single bit "D-Type edge trigger", one for each data bit, either a logic "0" or a "1", 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 counter 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.

Counter 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.

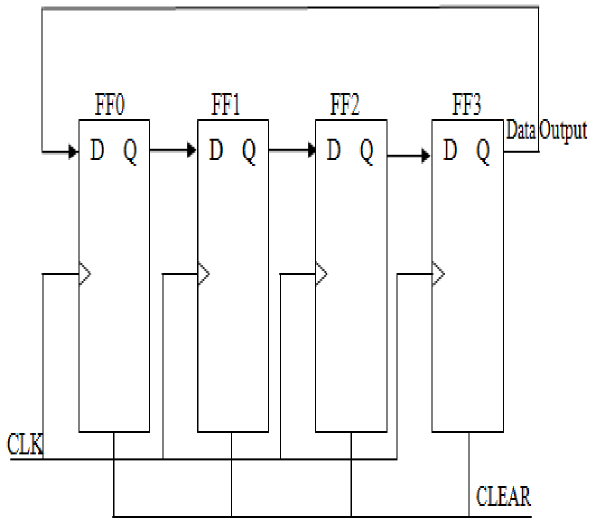

Figure 6 Block diagram of 4-bit ring counter

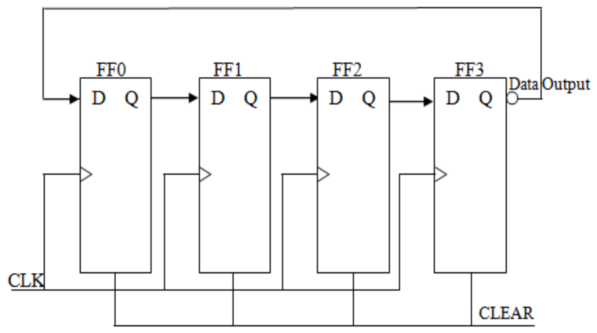

Figure 7 Block Diagram of 4-bit johnson counter

4.XOR/ XNOR circuit

In order to generate the EBC of three-input XOR/XNOR circuits, four steps are taken. Initially, three-input XOR and its complement are represented by one binary decision tree (BDT) in order to share common sub circuits. The BDT is achieved by some cascaded 2 ×1 MUX blocks, which are denoted by simplified symbol controlled with input variables at each correspondent level. This construction simply implements the min-terms of the three-input XOR/XNOR function. The step is followed by applying reduction rules to simplify the BDT representation. These include elimination, merging, and coupling rules. The major task of the coupling rule, in simple terms, is to obtain all the possible equivalent trees by interchanging the order of the controls. The trees are acquired by impacting the state matrix on the corresponding control matrix where the multiply and add operators operate as follows. The result of applying *the* reduction rules to the tree. Afterward, as the inputs into the first level are 0's and 1's of the function's truth table, the 0 and 1 can be replaced by the Y' and Y, respectively.

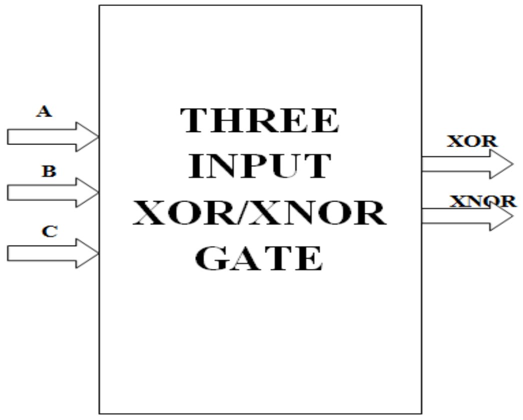

Figure 8 Block diagram of simple three input XOR/XNOR gate

5.Simulation result

All the designing and experiment regarding algorithm that we have mentioned in this paper is being developed on Xilinx 14.1i updated version. Xilinx 9.2i has couple of the striking features such as low memory requirement, fast debugging, and low cost. The latest release of ISE^{TM} (Integrated Software Environment) design tool provides the low memory requirement approximate 27 percentage low. ISE 14.1i that provides advanced tools like smart compile technology with better usage of their computing hardware provides faster timing closure and higher quality of results for a better time to designing solution. ISE 14.1i Xilinx tools permits greater flexibility for designs which leverage embedded processors. The ISE 14.1i Design suite is accompanied by the release of chip scope Pro^{TM} 14.1i debug and verification software. By the aid of that software we debug the program easily. Also included is the newest release of the chip scope Pro Serial IO Tool kit, providing simplified debugging of high-speed serial IO designs for Virtex-4 FX and Virtex-5 LXT and SXT FPGAs. With the help of this tool we can develop in the area of communication as well as in the area of signal processing and VLSI low power designing. To simplify multi rate DSP and DHT designs with a large number of clocks typically found in wireless and video applications, ISE 14.1i software features breakthrough advancements in place and route and clock algorithm offering up to a 15 percent performance advantage. Xilinx 14.1i Provides the low memory requirement while providing expanded support for Microsoft windows Vista, Microsoft Windows XP x64, and Red Hat Enterprise WS 5.0 32-bit operating systems.

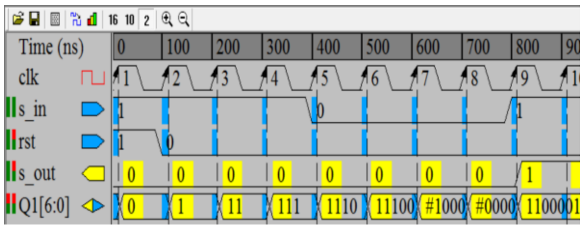

Figure 9 Output waveform of 8-bit serial in serial out shift register

The VHDL languages were used to design the VLSI architecture modules and are synthesized Spartan-3 (xc3s50-4pq208) device family. It is observed from the table 1 that the 2-bit SISO for number of slice 2, slice flip flop 3 and input output buffer 4, 4-bit SISO for number of slice 3, slice flip flop 5 and input output buffer 6, 8-bit SISO for number of slice 5, slice flip flop 9 and input output buffer 10, 16-bit

SISO for number of slice 10, slice flip flop 17 and input output buffer 18 and 32-bit SISO for number of slice 19, slice flip flop 33 and input output buffer 34.

Table 2 Device utilization summary of SISO shift register

Number of Bits	Number of Slice	Slice Flip Flop	IOBs	GCLKs
2-bit	2	3	4	1
4-bit	3	5	6	1
8-bit	5	9	10	1
16-bit	10	17	18	1
32-bit	19	33	34	1

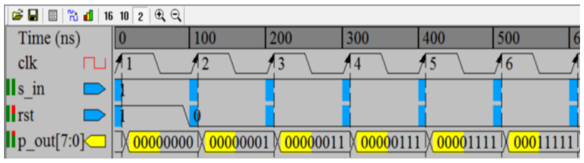

Figure 10 Output waveform of 8-bit serial in parallel out shift register

The VHDL languages were used to design the VLSI architecture modules and are synthesized Spartan-3 (xc3s50-4pq208) device family. It is observed from the table 2 that the 2-bit SIPO for number of slice 2, slice flip flop 3 and input output buffer 4, 4-bit SIPO for number of slice 3, slice flip flop 5 and input output buffer 6, 8-bit SIPO for number of slice 5, slice flip flop 9 and input output buffer 10, 16-bit SIPO for number of slice 10, slice flip flop 17 and input output buffer 18 and 32-bit SIPO for number of slice 19, slice flip flop 33 and input output buffer 34.

Table 3 Device utilization summary of SIPO shift register

Number of Bits	**Number of Slice**	**Slice Flip Flop**	**IOBs**	**GCLKs**
2-bit	2	3	4	1
4-bit	3	5	6	1
8-bit	5	9	10	1
16-bit	10	17	18	1
32-bit	19	33	34	1

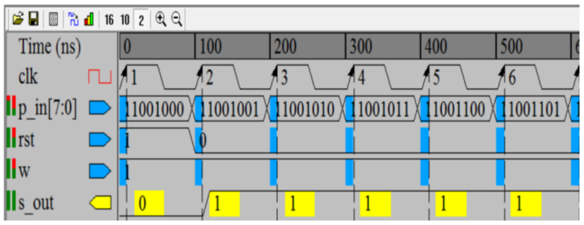

Figure 11 Output waveform of 8-bit parallel in serial out shift register

The VHDL languages were used to design the VLSI architecture modules and are synthesized Spartan-3 (xc3s50-4pq208) device family. It is observed from the table III that the 2-bit PISO for number of slice 1, slice flip flop 2 and input output buffer 5, 4-bit PISO for number of slice 2, slice flip flop 4 and input output buffer 7, 8-bit PISO for number of slice 5, slice flip flop 9 and input output buffer 11, 16-bit PISO for number of slice 9.
Slice flip flop 17 and input output buffer 19 and 32-bit PISO for number of slice 18, slice flip flop 34 and input output buffer 35.

Table 4 Device Utilization Summary of PISO Shift Register

Number of Bits	**Number of Slice**	**Slice Flip Flop**	**IOBs**	**GCLKs**
2-bit	1	2	5	1
4-bit	2	4	7	1
8-bit	5	9	11	1
16-bit	9	17	19	1
32-bit	18	34	35	1

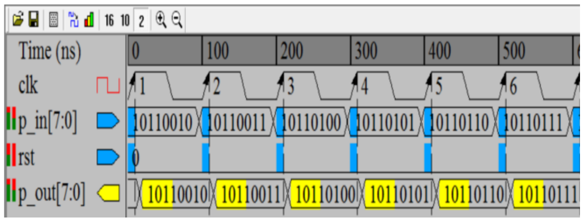

Figure 11 Output waveform of 8-bit parallel in parallel out shift register

The VHDL languages were used to design the VLSI architecture modules and are synthesized Spartan-3 (xc3s50-4pq208) device family. It is observed from the table IV that the 2-bit PISO for number of slice 1, slice flip flop 2 and input output buffer 5, 4-bit PIPO for number of slice 2, slice flip flop 4 and input output buffer 9, 8-bit PIPO for number of slice 5, slice flip flop 8 and input output buffer 17, 16-bit PIPO for number of slice 9, slice flip flop 16 and input output buffer 33 and 32-bit PIPO for number of slice 18, slice flip flop 32 and input output buffer 65.

Table 5 Device utilization summary of PIPO Shift register

Number of Bits	Number of Slice	Slice Flip Flop	IOBs	GCLKs
2-bit	1	2	5	1
4-bit	2	4	9	1
8-bit	5	8	17	1
16-bit	9	16	33	1
32-bit	18	32	65	1

Table 4 shows the illustration of the performance of proposed method discussed in this research work in term of maximum frequency, minimum period, arrival time before max input clock and max output required time. From the above graphical representation it can be inferred that the proposed algorithm gives the best performance as compared with previous algorithm.

Table 6 Timing summary of shift register

Register	Minimum Period	Maximum Frequency	Arrival time before max input clock	Max output required time
SISO	1.586 ns	630.517 MHz	2.459 ns	6.496 ns
SIPO	1.586 ns	630.517 MHz	2.459 ns	6.496 ns
PISO	2.081 ns	480.538 MHz	3.291 ns	6.271 ns
PIPO	1.586 ns	2.234 MHz	3.709 ns	6.271 ns

Table 6 Shows the illustration of the performance of proposed method discussed in this research work in term of maximum frequency, minimum period, arrival time before max input clock and max output required time. From the above graphical representation it can be inferred that the proposed algorithm gives the best performance as compared with previous algorithm.

Table 7 Comparison result for shift register

Design	**Bit**	**Minimum period**	**Arrival time before max input clock**	**Number of slice**	**Slice Flip Flop**
Previous Design	8-bit	2.363 ns	2.880 ns	8	13
Proposed Design		1.586 ns	2.459 ns	5	9
Previous Design	16-bit	2.540 ns	3.142 ns	15	24
Proposed Design		1.586 ns	2.459 ns	10	17
Previous Design	32-bit	2.750 ns	3.262 ns	24	43
Proposed Design		1.586 ns	2.459 ns	19	33

6.Conclusion

In this paper, we have discussed the VHDL implementation of configurable linear feedback shift register by number of slice, number of flip flop, input output bounded, minimum period, arrival time before max input clock, arrival time after max input clock and maximum frequency. The proposed shift register is 37.5% improvement of number of slice, 30.67% improvement of slice flip flop, 14.61% improvement of arrival time before max input clock and 32.88% improvement of minimum period. Its means that the proposed linear feedback shift registers is high speed compared to previous design.

References

[1] Mishra S, Tripathi RR, Tripathi DK. Implementation of configurable linear feedback shift register in VHDL. In Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES), International Conference on 2016 (pp. 342-6). IEEE.

[2] Kuorilehto M, Kohvakka M, Suhonen J, Hämäläinen P, Hännikäinen M, Hamalainen TD. Ultra-low energy wireless sensor networks in practice: Theory, realization and deployment. John Wiley & Sons; 2008.

[3] Hathwalia S, Yadav M. Design and analysis of a 32 bit linear feedback shift register using VHDL. International Journal of Engineering Research and Applications. 2014; 4:99-102.

[4] Sewak K, Rajput P, Panda AK. FPGA implementation of 16 bit BBS and LFSR PN sequence generator: a comparative study. In Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on 2012 (pp. 1-3). IEEE.

[5] Gopal L, Mahayadin NS, Chowdhury AK, Gopalai AA, Singh AK. Design and synthesis of reversible arithmetic and logic unit (ALU). In computer, communications, and control technology (I4CT), international conference on 2014 (pp. 289-93). IEEE.

[6] Gupta A, Malviya U, Kapse V. Design of speed, energy and power efficient reversible logic based vedic ALU for digital processors. In engineering (NUiCONE), 2012 nirma university international conference on 2012 (pp. 1-6). IEEE.

[7] Dixit A, Kapse V. Arithmetic & logic unit (ALU) design using reversible control unit. Development. 1998; 32:16-23.

[8] Premananda BS, Ravindranath YM. Design and synthesis of 16 bit ALU using reversible logic gates. Int. J. Adv. Res. Commun. Eng. 2013; 2(10):2278-1021.

[9] Morrison M, Ranganathan N. Design of a reversible ALU based on novel programmable reversible logic gate structures. In VLSI (ISVLSI), IEEE computer

society annual symposium on 2011 (pp. 126-31). IEEE.

[10] Mamataj S, Das B, Rahaman A. An optimized realization of ALU for 12-Operations by using a control unit of reversible gates. International Journal of Advanced Research in Computer Science and Software Engineering. 2014; 4(1):496-502.

[11] Mamataj S, Das B, Rahaman A. An Ease implementation of 4-bit Arithmetic Circuit for 8 Operation by using a new reversible COG gate. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. 2014; 3(1).

[12] Landaurer R. Irreversibility and heat generation in the computational process. IBM Journal of Research and Development. 1961; 5:183-91.

[13] Bennett CH. Logical reversibility of computation. IBM Journal of Research and Development. 1973; 17(6):525-32.

[14] Aradhya HR, Kumar BP, Muralidhara KN. Design of control unit for low power AU using reversible logic. Procedia Engineering. 2012; 30:631-8.

[15] Rao KP, Vani RM, Hunagund PV. Gain enhancement of linear four element microstrip antenna array. International Journal of Advanced Technology and Engineering Exploration. 2018 Jun 1;5(43):151-9.