ALU Design, Working, Issues and Tradeoffs

Sai Praneeth Chokkarapu, Surya Poondla, Ruchitha Vucha, Sai Krishna Charan Dara

Electronics and Communication Engineering Department
International Institute of Information & Technology Hyderabad, India
saipraneeth.c@students.iiit.ac.in, surya.poondla@students.iiit.ac.in,
ruchita.vucha@students.iiit.ac.in, sai.krishna@students.iiit.ac.in

***Abstract*—Arithmetic Logic Unit(ALU) is one the most important component of any computing system like microprocessors, embedded chips, super computers, smart phones and sequential circuits. The ALU is present in very complex devices to the daily devices we use. ALU is a combinational circuit and it performs arithmetic and logic operations. For a circuit design, the important aspects are to minimize the delay, power and area as per the application. In this paper we cover various aspects of ALUs like working of 4-bit ALU designs, various approaches on how to optimize delay, how GDI technique is used to optimize delay and power, and its constraints and various techniques on how to test the ALUs.**

***Index Terms*—Arithmetic Logic Unit (ALU); Gate Diffusion Input (GDI); Full-Swing GDI (FS-GDI); Power optimization techniques; Testability of ALU; BIST in ALU**

I. Introduction

Portable electronic devices are playing a key role in day to day life. With the evolution of technology, these devices operate at low power and high speed. One of the important digital circuits is Arithmetic Logical Unit(ALU). ALUs of various bit-widths are frequently required in very large-scale integrated circuits (VLSI) from processors to application specific integrated circuits (ASICs). They are considered as an essential part in many applications such as Microprocessor design, digital signal processing, image processing, etc.

Arithmetic Part of ALU is made up of Full Adder and the logical part consists of Logical Gates such as AND,XOR,OR and XNOR. Any improvement in designing adder cell and Logical block would result in enhancing performance of ALU. In this paper we propose an ALU with a full swing GDI technique. Simulation environment is LTSpice using TSMC 180nm process.

The paper is organized as follows. Section I discusses various logic design techniques (CMOS, GDI, TG, N-PG) and comparison of these using parameters like delay, power and number of transistors and choosing the method that suits designing ALU. Section II covers Logic Design Techniques used in ALU. In Section III, various ALU Designs for Delay optimisation are discussed and compared them using various design criteria like power and transistor count. Voltage variation of the circuits is also discussed. Section IV talks about power optimisation in ALUs, various ALU architectures like tree architecture, chain architecture and GDI based architecture which has 8 Transistor based Full Adder and other blocks like logic block, multiplexers based on GDI technique and comparison of the architectures and finally in Section V, Design for testability in VLSI, analysis of different designs of testable ALU are discussed.

II. Gate Diffusion Input

A. Advantages of GDI over CMOS and PTL

Performance of logic circuits, once based on traditional CMOS technology are getting improved by developing of many logic design techniques. One form of logic that is popular in low-power digital circuits is pass-transistor logic (PTL). Pass transistor logic was formally based on NMOS transistor.

Some of the main advantages of PTL over standard CMOS design are:

1) Ratio-less logic - Output does not depend on W/L ratio of transistors
2) High speed - due to the small node capacitances
3) Low power dissipation - as a result of the reduced number of transistors
4) Lower interconnection effects - due to a small area

However, PTL implementations have few basic problems. The threshold drop across the single-channel pass transistors results in reduced current drive and hence slower operation at reduced supply voltages; this is particularly important for low power design since it is desirable to operate at the lowest possible voltage level. The “high” input voltage level at the regenerative inverters is not Vdd, the PMOS device in the inverter is not fully turned off, and hence direct-path static power dissipation could be significant [1].

Gate Diffusion Input(GDI) solves most of the problems of PTL. GDI approach allows implementation of a wide range of complex logic functions using only two transistors. This method is suitable for design of fast, low power circuits, using reduced number of transistors (as compared to CMOS and existing PTL techniques), while improving logic level swing and static power characteristics and allowing simple top-down design [2].

B. GDI Functions

- The GDI cell contains three inputs: G (common gate input of NMOS and PMOS), P (input to the source/drain of PMOS), and N (input to the source/drain of NMOS).
- Substrate(Bulk) of NMOS is connected to N and bulk of PMOS is connected to P.

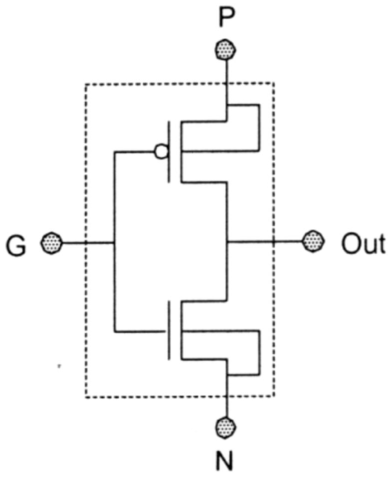

Fig. 1. Basic GDI Cell

TABLE I
VARIOUS LOGIC FUNCTIONS OF GDI CELL FOR DIFFERENT INPUT CONFIGURATIONS

N	P	G	Out	Function
'0'	B	A	$\overline{A}B$	F1
B	'1'	A	$\overline{A}+B$	F2
'1'	B	A	$A+B$	OR
B	'0'	A	AB	AND
C	B	A	$\overline{A}B+AC$	MUX
'0'	'1'	A	$\overline{A}$	NOT

GDI method uses simple basic cell as shown in Fig1.

Table I shows different boolean functions by changing N, P and G terminals of basic GDI cell. Generally these require 6 to 12 transistors (CMOS or PTL), but using GDI all these can be calculated using only 2 basic cells.

Small drawback of this GDI is swing restoration. For example, for F1 function when both A and B are 0, output is V_{tp} instead of 0 Voltage. So additional circuitry is needed for swing restoration logic.

It is shown that the delay of CMOS gate compared to GDI gate with same functional complexity is given by:

$$1.52 \leq \frac{t_{PHL(CMOS)}}{t_{PHL(GDI)}} \leq 2$$

where the high bound is for high C_{load} and the low bound is for C_{load} low.

C. Performance evaluation of GDI Cell

In ALU, we would be using different logical functions and multiplexers, so we compare GDI based implementation of delay, power and number of transistors with CMOS, Transmission gate (TG) and NMOS Pass Gate as shown in Fig 2 and schematic design in Fig 3.

LOGIC GATE COMPARISONS (GDI, CMOS, TRANSMISSION GATE, AND NMOS PASS GATE) USING THE CIRCUIT TOPOLOGIES FROMTABLE IV

Gate type in series	Logic expression	GDI			CMOS			TG			N-PG		
		Power (μW)	Delay (nsec)	# tr.	Power (μW)	Delay (nsec)	# tr.	Power (μW)	Delay (nsec)	# tr.	Power (μW)	Delay (nsec)	# tr.
MUX	$\overline{A}B+AC$	35.7	1.1	8	49.7	2.1	24	44.9	1.0	16	47.5	3.1	16
OR	$A+B$	26.3	1.2	8	32.9	1.7	12	36.2	1.3	16	32.6	2.7	16
AND	AB	25.7	0.9	8	34.1	1.4	12	30.8	0.8	16	30.1	2.8	16
F1	$\overline{A}B$	31.2	0.8	8	45.2	1.5	12	31.8	1.1	16	31.8	2.5	16
F2	$\overline{A}+B$	32.0	1.3	8	43.1	1.9	12	33.2	1.4	16	29.6	3.5	16

Fig. 2. Comparison

AND, OR, AND XOR CELLS USING GDI, CMOS, AND PTL DESIGN TECHNIQUES FOR TWIN-WELL PROCESS

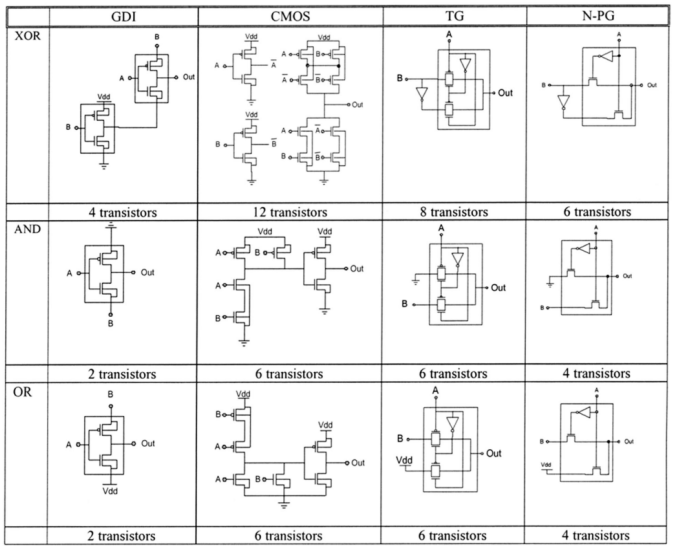

Fig. 3. Design

In order to perform a fair comparison between the techniques, the measurements were carried out from cells series with buffers and not from a single cell. GDI and TG test circuits contain 2 basic cells with one output buffer. N-PG contains two buffers - one after each cell. CMOS has no buffers in test circuits.

Among all the shown logical design techniques, GDI proves to be best with less count of transistors.

Even in some cases where delay and power are not least, we can observe that **Power-Delay** product is minimum among all the presented designs.

Hence, observing the advantages of GDI presented above , we use GDI techniques to construct both logical and arithmetic operations for Arithmetic Logical Unit(ALU).

III. PROPOSED DESIGNS OF ALU FOR DELAY OPTIMIZATION

The design of ALU is a critical part of the processor hence new approaches are being continuously developed to optimise the delay of ALU. The timing response of ALU depends on the components of the circuit.

ALU takes input operands A,B and Carry-in and performs the operation chosen using the input combination available on the select lines S and displays the output through G and Carry out.

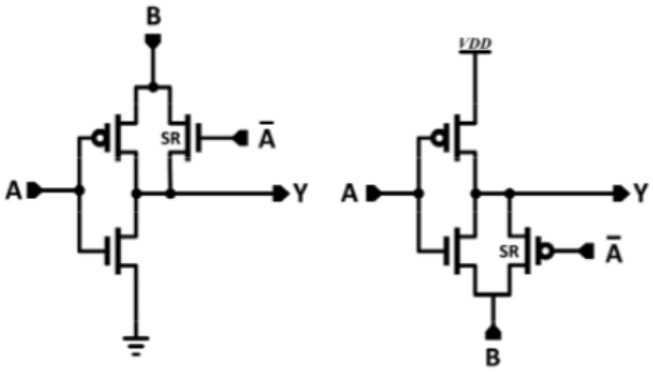

Fig. 4. Proposed Full Swing GDI cell

S2	S1	S0	Operations	Function
0	0	0	G = A - 1	DECREMENT
0	0	1	G = A + B	ADDITION
0	1	0	G = A + $\overline{B}$ + 1	SUBTRACTION
0	1	1	G = A + 1	INCREMENT
1	0	0	G = A ∧ B	AND
1	0	1	G = A ⊕ B	XOR
1	1	0	G = $\overline{A}$	XNOR
1	1	1	G = A ∨ B	OR

Fig. 5. Truth Table of the 4-bit ALU

A. Full Swing GDI

As we discussed above that GDI doesn't have full swing operation. Full-swing GDI cells proposed to improve the output swing of GDI gates as an alternative for swing restoration buffers, a swing restoration transistor used to improve the output swing of F1 and F2 gates as shown in fig 4.

B. ALU Design Type - I

In some processors the ALU is divided into an arithmetic unit and logical unit. This strategy plays an important role in the design of the following circuit. Using the select line S2 we can choose between arithmetic and logical operations. Full adder is used to perform arithmetic operations in ALU. We designed a separate Logical block which takes care of the logical operations. Full Swing GDI technique is introduced as an alternative to CMOS logic in all the blocks [5].

The proposed design of ALU consists of the following combinational digital blocks:

- Multiplexers
- Logical block
- Full adder

1) Multiplexers: Depending on the operation we need to choose among the different inputs to be given at operand B and then choose the output that is to be displayed. Multiplexer

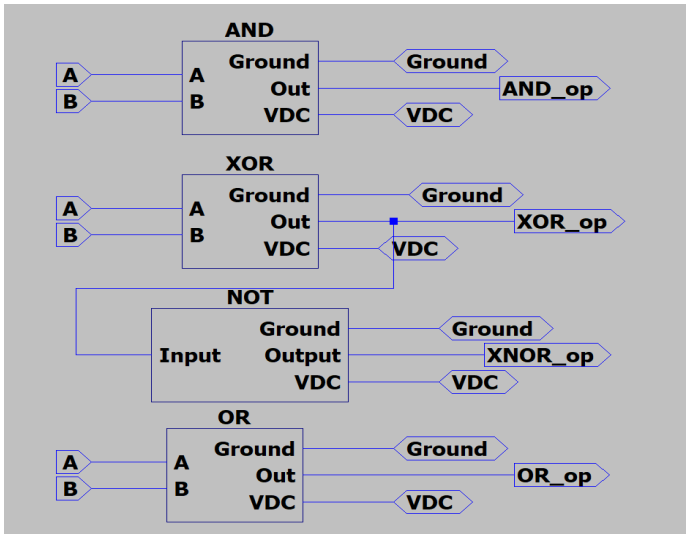

Fig. 6. Logical Block

is a combinational logic circuit used to select one of the several inputs and pass it onto the output. Multiplexer with 2^n inputs has n selection lines and one output.

4-bit ALU implements four arithmetic and four logical operations as shown in the Fig 5. The design requires 4x1 multiplexers to choose among the four inputs required for different arithmetic operations and also choose among the four logic operations performed by the Logical block. In the next stage we need a 2x1 multiplexer to choose between the output of the logical block and full adder.

A 2x1 multiplexer using Full Swing GDI technique can be implemented using 6 transistors itself whereas using CMOS logic 12 transistors are needed. Full Swing GDI technique 4x1 multiplexer can be implemented using 18 transistors itself.

2) Logical Block: This block takes the input operands A & B and performs the following logical functions on them : AND, XOR, XNOR & OR. Even though using a separate block for logical functions increases the transistor count the delay is decreased as the logical block and full adder can operate simultaneously now. Full Swing GDI AND, XOR and OR gates are implemented as shown in Fig 3. Logical block is implemented in LTSpice using the Full Swing GDI AND, XOR and OR gates as shown in Fig 8.

3) Full Adder: Full adder plays an important role in ALU as it performs all the arithmetic operations. Transistor count, delay and power are the criterion for the design of full adders. Most of the time adder lies in the critical path and hence optimizing adders timing response is one of our main concerns. Full adder adds the 3 inputs A,B & Carry in and produces 2 outputs Sum & Carry out. Logical expressions of

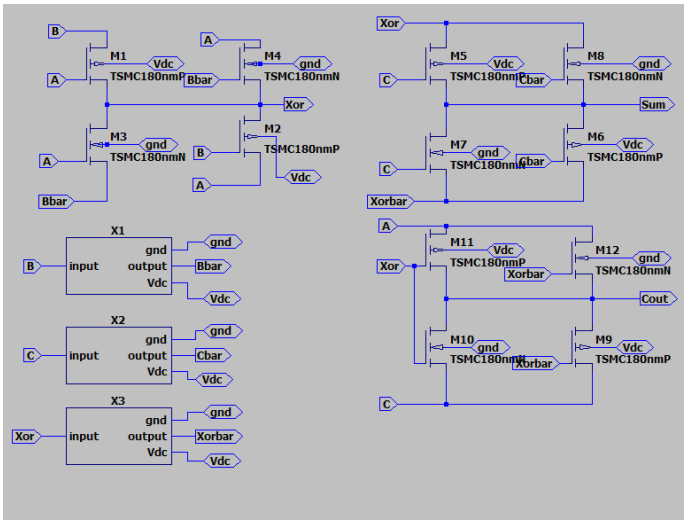

Fig. 7. Full Adder

outputs in terms of inputs are as follows:

$$Sum = A\ \boldsymbol{XOR}\ B\ \boldsymbol{XOR}\ C_{in} \quad (1)$$

$$C_{out} = A\ \boldsymbol{AND}\ B + B\ \boldsymbol{AND}\ C_{in} + A\ \boldsymbol{AND}\ C_{in} \quad (2)$$

The timing response of full adder can be optimised by proper selection of logic styles. Different styles favor different performance aspects. The equations (1) and (2) can be rewritten as follows:

$$Sum = \overline{C_{in}}(A\ \boldsymbol{XOR}\ B) + C_{in}(A\ \boldsymbol{XNOR}\ B) \quad (3)$$

$$C_{out} = (A\ \boldsymbol{XOR}\ B)C_{in} + \overline{(A\ \boldsymbol{XOR}\ B)}A \quad (4)$$

Here AXORB can be used as an intermediate result for computing both sum and carryout. Sum is obtained by the combination of product of XOR with $\overline{C_{in}}$ and XOR's inverted version XNOR with C_{in}. Carryout can be obtained by multiplexing A & C_{in} with AXORB acting as a select line. The delay is increased by the presence of an inverter in the critical path. Full adder using Conventional CMOS logic requires 40 transistors. Full Swing GDI Full adder[27] is implemented in this logic style using 18 transistors as shown in Fig 9.

4) Implementation of one-bit ALU: One-bit ALU is implemented by integrating the multiplexers, logic block and the full adder as shown in Fig 10.

Arithmetic Operations are performed by the full adder on operand A by choosing the respective operand B with the help of a 4x1 multiplexer. The values of operand B are chosen among the four possible values B, $\overline{B}$, 0 and 1, by the input combination available on the select lines s0, s1. Sum and C_{out} are the outputs of the sum of these 3 operands A, B, C_{in}. Implementation of arithmetic operations :

- Decrement: $s2 = 0, s1 = 0, s0 = 0$
  The output of 4x1 mux is 1. Hence the 2nd input of the full adder is 1. As C_{in} = 0, A is summed with 1 which represents -1 in 2's complement in the full adder .
  Full adder output = A-1.
- Addition: $s2 = 0, s1 = 0, s0 = 1$
  The operand B is passed as 2nd input to the full adder and C_{in} = 0. Now A is summed up with B.
  Full adder output = A+B.
- Subtraction: $s2 = 0, s1 = 1, s0 = 0$
  Now $\overline{B}$ is the 2nd input of the full adder. Subtraction is performed by summing 2's complement A with B's complement with C_{in} = 1.
  Full adder output = A + $\overline{B}$ + 1 = A - B.
- Increment: $s2 = 0, s1 = 1, s0 = 1$
  0 is passed as 2nd input to the full adder and C_{in} = 1. Now A is summed with 1.
  Full adder output = A + 1.

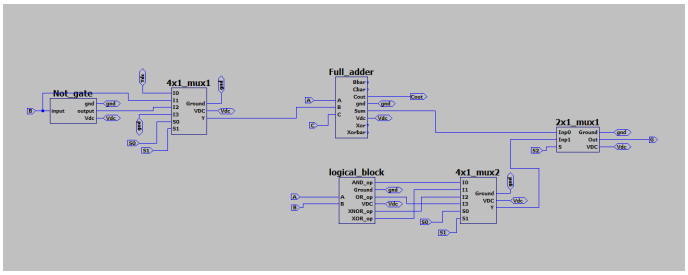

Fig. 8. One-bit ALU

Logical Block performs the 4 logical operations AND, OR, XOR, XNOR on A & B. A 4x1 multiplexer is used to choose among these 4 outputs based on the select lines s0, s1. Finally the select line s2 is used to choose between the arithmetic and logical outputs.

5) Integration of four bit ALU: Four bit ALU is implemented by cascading the one-bit ALUs in four stages as shown in Fig 11. The carryout of ALU is given as carryin for the ALU in the next stage.

C. ALU Design Type - II

In this design Full Swing GDI technique is used to realise all the circuits. The implementation of multiplexers is similar to the previous design. There is no separate logical block as Adder performs arithmetic and logical operations as well. This design maintains low power and also decreases the transistor count[4].

1) Full Adder: The outputs of Full adder can be expressed in terms of the inputs using XOR, AND & OR operations as shown below:

$$Sum = A\boldsymbol{XOR}B\boldsymbol{XOR}C_{in} \quad (5)$$

$$C_{out} = \overline{C_{in}}(A\boldsymbol{AND}B) + C_{in}(A\boldsymbol{OR}B) \quad (6)$$

Sum is achieved by the XOR of the inputs A, b and Carry in. C_{out} is achieved by multiplexing the AND and OR operation through Carry in. Full adder is implemented using

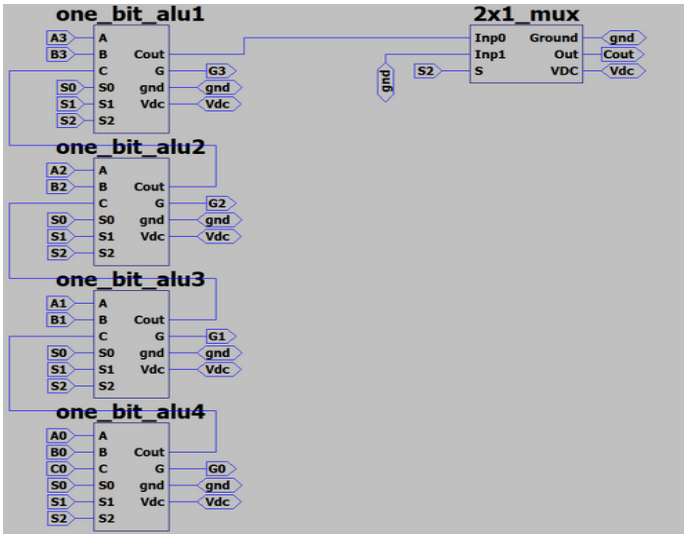

Fig. 9. Four-bit ALU

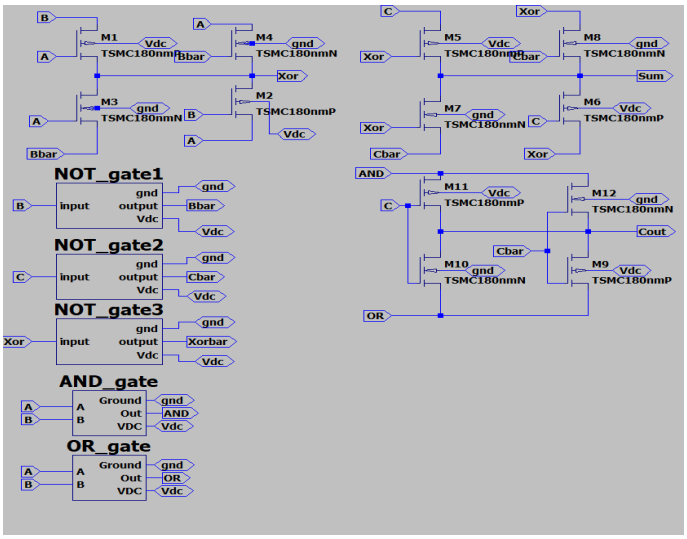

Fig. 10. Full Adder II

only 22 transistors. The Full adder circuit[27] is implemented in LTSpice using Full Swing GDI AND, OR & XOR circuits as shown in Fig 12.

2) Implementation of one-bit ALU: As the full adder itself performs the logical operations for the implementation of one bit ALU we need a 4x1 multiplexer to choose among the logical outputs and then we need a 2x1 multiplexer to choose between the logical and arithmetic operations. 4-bit ALU is realised by cascading the one-bit ALUs in 4 stages as shown in the previous design.

D. Verification of simulation results

The proposed designs of 4-bit ALU are implemented in LTSpice using 180nm TSMC CMOS process. The simulations are done with a power supply of 1.8V. The transistor sizing of

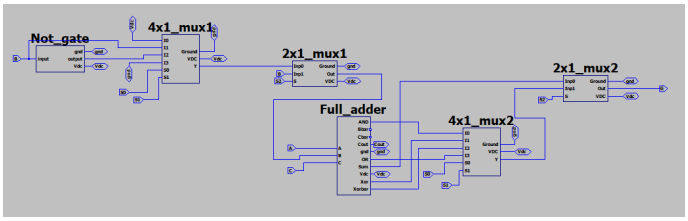

Fig. 11. One-bit ALU II

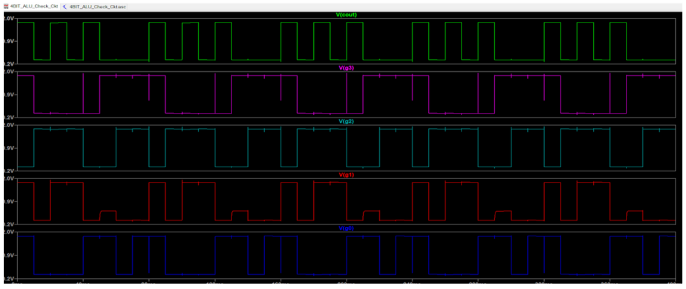

Fig. 12. Output Waveform

PMOS and CMOS are as follows: $(W/L)_P$ = 720 nm /180 nm $(W/L)_N$= 360 nm /180 nm.Here average power is calculated from all possible operations of ALU. The output waveform of the four-bit alu for the test inputs A = 1100 B = 0101 is shown in Fig 12. From the analysis and simulations it is observed that the delay of Design-I is less compared to that of the Design-II. But Design-II performs better in terms of Power and transistor count.

E. Voltage Variation

From testing it is found that the circuit works for input voltage greater than 0.9 when we assume voltage $>$ 0.9 as high.

IV. Power Optimization

A. Importance Of Power Optimization

As we know that ALU is a very essential component of the circuits, it is almost busy all the time, and makes the ALU one of the most power hungry components. If we optimize the ALU design then we can reduce the power consumption of the whole circuit too. The power optimization can be done in several ways like transistor sizing [17], threshold voltage scaling [18] at semiconductor design level, clock gating [20], power gating [19] at logic level, Dynamic Voltage Scaling at system level [21]. The power reduction can also be done on individual components of the circuit.

	Delay	Power	Transistor Count
ALU Design Type - I	0.122375 ns	54.927 µW	294
ALU Design Type - II	0.120147 ns	48.627 µW	286

Fig. 13. Simulation results

B. Literature Review

There are many methods to reduce the power leakage. The power Dissipation is mainly due to Switching Power (Dynamic Power), Short circuit Power(Static Power), Leakage(Static Power) [22] [24]

$$P_{total} = P_{Switching} + P_{Shortcircuit} + P_{Leakage}$$

A detail study was done on optimizing the dynamic power. In this section we will optimize dynamic power dissipation and see various architectures, combinations of the various methods mentioned in previous section for 4-bit ALUs.
The below Architectures have been implemented in LTSPICE software.x

- CMOS based tree architecture
- CMOS based Chain architecture
- GDI based optimized 8T Full adder structure.

C. Analysis and Implementations

1) CMOS based tree architecture: In this architecture all the building blocks have been implemented based on CMOS logic. A 1-bit ALU design which is adder independent has been implemented. The 4-bit ALU is implemented like an extension of the 1-bit adder, it is based on a 4-bit ripple carry adder [23]. The technology used is 180nm technology.

The logic operations are separated from the adder logic.

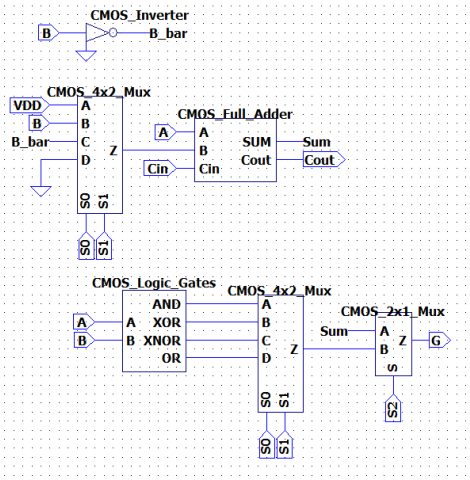

Fig. 14. 1-Bit ALU Adder independent Structure

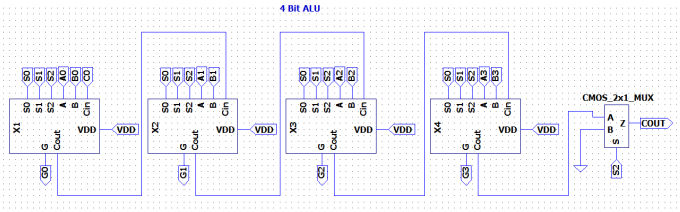

Fig. 15. 4-Bit ALU Structure

The test-bench output for the ALU designs is shown in Figure(16). The average power of the circuit is shown in the Figure(17). The average power is calculated at the

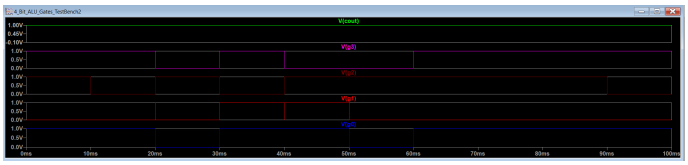

Fig. 16. 4-Bit ALU Working

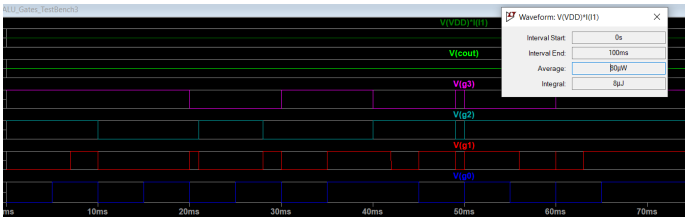

Fig. 17. Average power

time of switching the operands. The average power depends on the frequency of the pulses, as the frequency increases the average power dissipation increases. The switching of operands causes a change in the current which causes the power dissipation of the circuit.
The average power for tree architecture is approximately around 80μ Watts.

2) CMOS based Chain Structure: The Chain Architecture is mainly based on the applications [22], it has a smaller area compared to the previous architecture. The placing of the components changes the power dissipation. CMOS based 1-bit ALU was implemented. The adder block is same as the previous architecture, the logic block has been divided into two sub-blocks. The average power of the 4-bit ALU chain

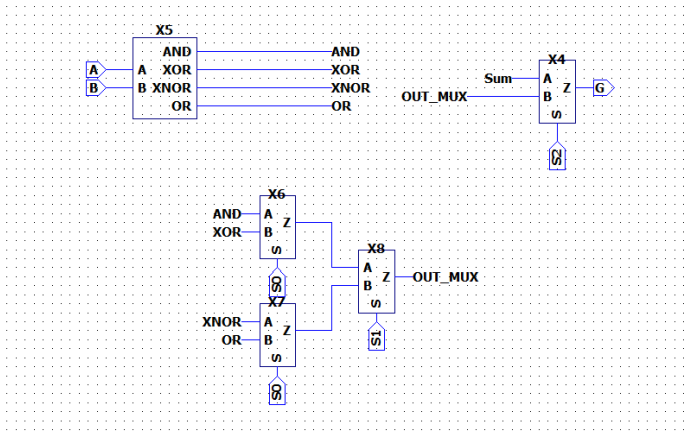

Fig. 18. Chain Structure of operations

structure is shown in Figure(19) The average power of the architecture is approximately 84μ Watts. The power can be further optimized based on the Application. The placement

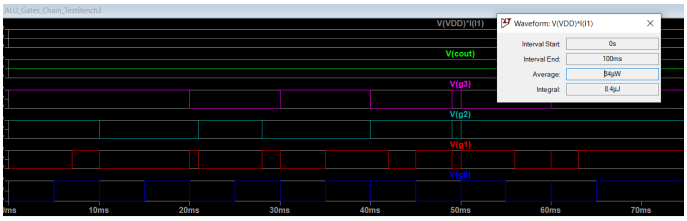

Fig. 19. Average Power of Chain Structure

of components can be done based on Benchmark Dhrystone [22] [24] to place the components to reduce the power in the critical paths.

3) GDI Based 8T Full adder: In this architecture we modify each of the blocks [23] like MUX, adder, logic block and implement them with GDI mechanism. The GDI technique is technique very efficient in optimizing the power dissipation, this is because in GDI based designs the number of transistors is reduced largely, in return reduces the power dissipation of the circuit.

The Full Adder is a key factor to power dissipation but we could not optimize its structure in the previous architectures. Here we optimize the full adder and the Multiplexers based on GDI method. We can optimize the full adder in various ways like 8 transistors [23] , 10 transistors [25], 12 transistors etc. .Here we implemented a 8 transistor based Full Adder design based on [23] [26].

The design of 8 Transistors Full Adder is shown in

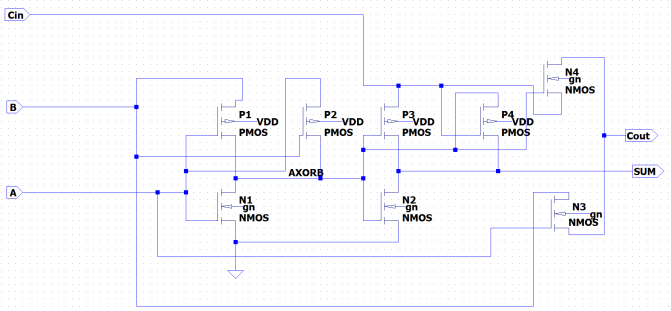

Fig. 20. 8 Transistors based FULL Adder design

Figure(20), Logic block based of GDI design is implemented as shown in Figure(3) and 4x1 Mux shown in Figure(22) and 2x1 Mux shown in Figure(21) are based on GDI.

We can observe that the Full Adder transistors have reduced considerably from 28 transistors to 8 transistors and the traditional AND gate, OR gate and XOR gate based on CMOS logic had 6, 6, 12 transistors respectively but using GDI technique we can reduce it to 3, 3, 4 transistors respectively, Similarly 4x1 Mux based on CMOS had 32 transistors and using GDI the number of transistors reduced to 12 and the 2x1 Mux based on CMOS logic had 12 transistors

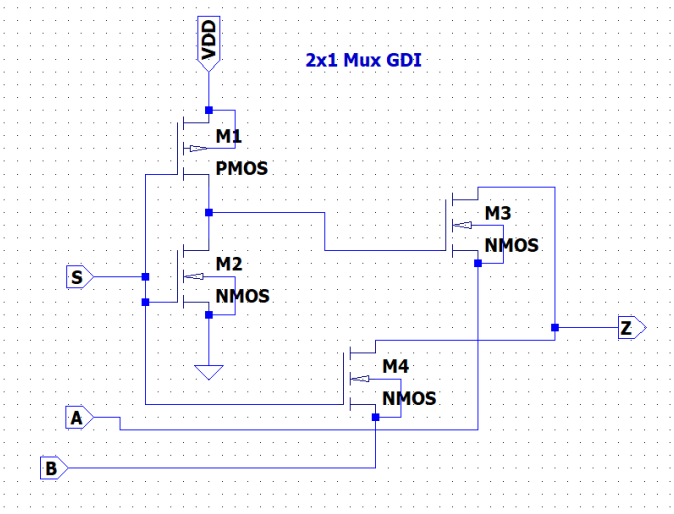

Fig. 21. 2x1 Mux design based on GDI

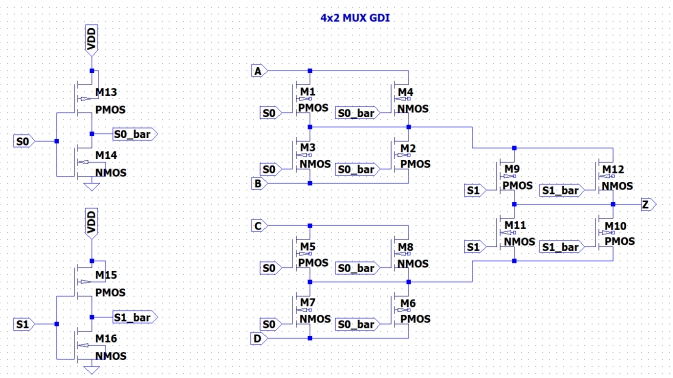

Fig. 22. 4x1 Mux design based on GDI

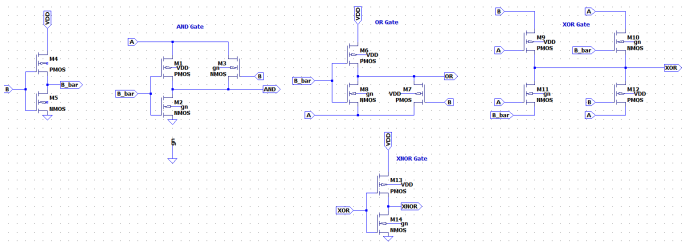

Fig. 23. Logic Block design based on GDI

which reduced to 4 transistors using GDI method.
The average power of GDI based architecture is around 47μ Watts, it can be seen in Figure(24). We can observe that the GDI technique has considerably reduced the power dissipation and Area [26] compared to the other architectures.

D. Conclusion

The CMOS based architectures have low power dissipation for individual gates but when they are combined, all the low power dissipation adds up to form a large power dissipation.

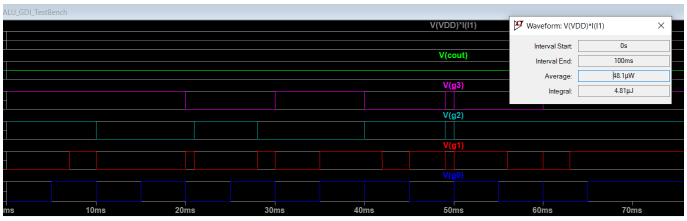

Fig. 24. Average Power of GDI Architecture

We can optimize it by optimizing the W/L ratios but we need to compromise on the delays [3], various placing techniques [22]. The GDI Technique is very helpful as it optimizes power(both Static and Dynamic!!), delay and the number of Transistors, which in turn reduce the area. GDI technique is very useful but the drawback is it is tough to fabricate. The comparison table for Power optimized architecture can be seen in Figure(25).

Design	Avg Power Dissipation	Min Avg Power	Total Number of Transistors	Area
CMOS based Tree Architecture	80u Watts	50u Watts	312	Moderate
CMOS based Chain Architecture	84u Watts	52u Watts	320	Less
GDI Based 8T Architecture	48.1u Watts	22u Watts	184	Least

Fig. 25. Comparison of architectures

V. Testability in ALUs

A. Importance of VLSI testing

Following the so-called Moore's law, the scale of ICs has doubled every 18 months. VLSI devices with many millions of transistors are commonly used in today's computers and electronic gadgets. This is a direct result of the steadily feature size of the transistors and interconnecting wires from tens of microns to tens of nanometers, with current submicron technologies based on a feature size of less than 100 nm. The reduction in feature size increases the probability that a manufacturing defect in the IC will result in a faulty chip. A very small defect can easily result in a faulty transistor or interconnecting wire. Furthermore, defects created during the manufacturing process are unavoidable and as a result, some number of ICs is expected to be faulty.

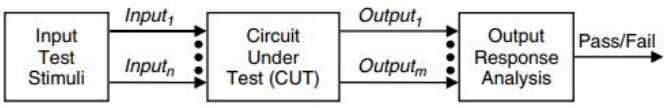

Fig. 26. Basic testing approach

Testing typically consists of applying a set of test vectors to the inputs of the circuit under test (CUT) while analyzing the output responses, as in figure (26). Testing is performed at various stages in the lifecycle of a VLSI device, including during the VLSI development process, the system manufacturing process etc.

Because of the diversity of VLSI defects, it is difficult to generate tests for real defects. Many fault models have been proposed [6] but no single fault model accurately reflects the behavior of all possible defects that can occur. Different fault models are used to detect detect different kinds of faults. Some examples are, a stuck-at fault model assumes a signal line is faulty and is stuck at either logic 1 or logic 0, an input pattern(IP) fault model allows change of response to a module input pattern, a cell fault model allows the function of any single module in the circuit to change to any other function of the same number of inputs. The cell fault model is a widely used fault model.

Let's say the input operand size of the circuit is N. There are different levels of testability [6] associated to a CUT. Some examples are Lin-testability, C-testability, I-testability, CI-testability, pI-testability etc. A circuit design is said to be Lin-testable (linearly testable) if it is testable with a test set whose size is proportional to N. A circuit is C-testable (constant-testable) if it is testable with a test set whose size is constant irrespective of the circuit's size. One must note that in design for testability (DFT), a circuit design is evaluated based on the fault model assumed and then testability of the circuit changes as per the design and fault model.

B. Literature survey

Prior work has shown that an array structure improves an ALU's testing properties [7]. However, the array-type ALU has a major disadvantage that its worst case delay is linearly proportional to N, where N is the input operand size of the ALU. Carry-lookahead circuitry in the adder changes the worst case delay from being linear in N to logarithmic in N but increases the hardware cost and the C-testability of the circuit is lost [8]. Moreover, the adder is found to be untestable, that is, some faults are undetectable, when a more general functional fault model is considered [9] [10].

Consider an 8-bit tree ALU using a carry-lookahead adder as in figure (27). Each LM module executes various logical operations on a pair of operand bits A_i and B_i and is controlled by the values applied to the control input bus Q. Arithmetic operations are performed using the modules gp, GP, C, FA, and SM. Much of the tree ALU's circuitry is easy to test. The testing difficulty associated with the tree ALU arises from the carry-lookahead circuitry. Although testable for all single stuck line faults, the gp, C, and GP modules are not fully testable. So, some design changes have to be done so that it becomes fully testable. Methods for designing fully testable ALUs with different levels of testability are proposed in [10] [11] [12]. There are different techniques used in design for testability. For example, there

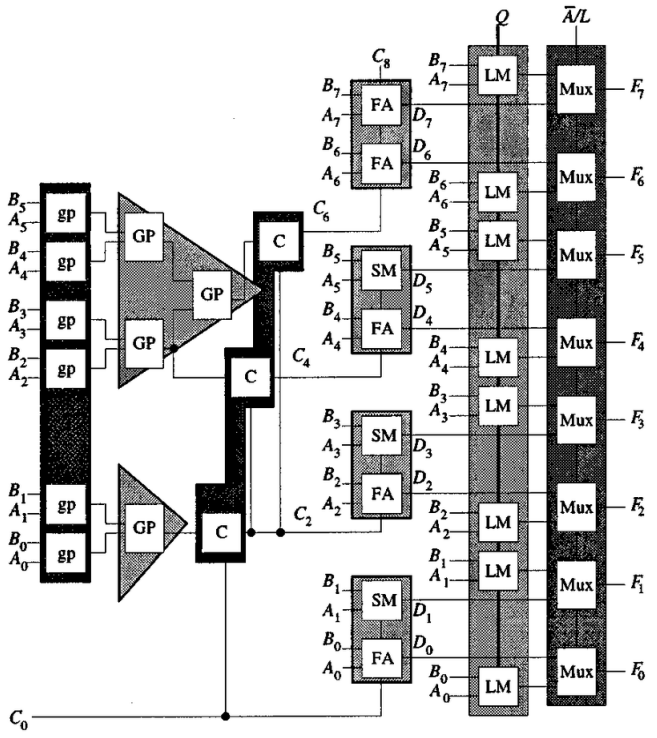

Fig. 27. An 8-bit tree ALU utilizing a carry-lookahead adder

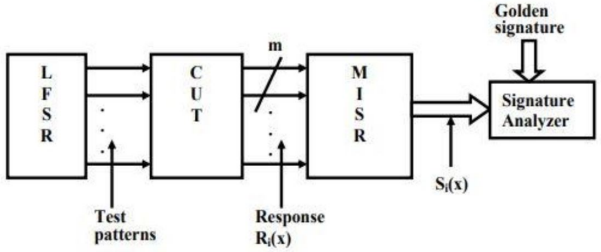

Fig. 28. BIST logic generic structure

are scan-path design techniques and Built-in self-test(BIST). In the literature, BIST is considered to be the best and ideal technique in design for testability. Several BIST designs have been proposed assuming different fault models resulting in a trade off with power, area and delay.

C. Design techniques in DFT

While scan-path design techniques have steadily improved the testability of complex digital circuits, built-in self-test techniques can improve it more. BIST is a DFT technique in which a portion of a circuit on a chip, board, or system is used to test the digital logic circuit itself. In BIST, the input test vectors and responses do not have to be shifted in and out and so the BIST designs significantly improve increase testing speed. Moreover, we can use BIST and scan-path techniques together. In a typical BIST implementation, we would test all functional circuit modules using on-chip test generators and response verifiers. We would then use the scan path to test registers, initialize test generators, or recover the test-response signature.

Despite these obvious speed advantages, BIST techniques do have their share of problems:

- BIST circuitry wastes space. The additional BIST circuitry does not extend chip functionality, yet increases the circuitry that must be tested.
- Faults that affect memory, which are stuck-open faults, are hard to detect. This fault may convert a combinational CMOS circuit into a sequential one.

In BIST, it is not possible to store all the outputs of a CUT on chip. So, signatures are generated by compressing the outputs and these signatures are later used to compare with the golden signature to detect faults in CUT. As shown in figure (28), BIST design methodology typically consists of LFSR block to generate test patterns for inputs of CUT, then CUT and then PASS-FAIL signal from the signature analyser shows whether the CUT is faulty or fault free by comparing the test signature value with reference signature value from MISR.

Several BIST designs have also been proposed in [13] [14] [15] [16]. In [13], a design is proposed to implement BIST technique on optimized ALU using Cadence Encounter Platform. It focused on optimizing ALU and testing BIST concept. In ALU, Vedic multiplier and carry save adder were included. Overall 31% reduction in area and 42% reduction in power had been reported. In [14], a TPG is proposed which has lower hardware overhead and was used in BIST. Switching activity of CUT was reduced using proposed TPG and BIST. Fault coverage of 100% was achieved and area overhead is also reduced. Authors in [15] proposed reconfigurable BIST logic which detects faults across the ALU block mapped on the FPGA. In [16], authors proposed a technique based on pI testability for implementing CMOS ALU covering all functional faults.

D. Observations and Analysis

Different designs for fully testable ALUs are proposed in [10] [11] [12]. An analysis of their designs can be found in figure (29). It shows the gate overhead, number of IP faults, fault coverage for Unmodified, Linearly testable and Constant testable ALUs. We can observe that as the testability of circuit design changes from unmodified to Lin-testable to C-testable, the number of gates increases due to additional circuitry added to improve the testability. So, this means there occurs a trade off between testability of the design, hardware and area. Consequently, due to the increase in the number of gates, the power and delay might also increase. From the 2nd row of the table in figure (29), we can observe that the number

of IP faults also increased due to additional gates. The reason why one checks for IP faults and stuck line faults in analysis is that a design is said to be testable under the widely used cell-fault model only if every circuit module can be tested for all of its possible IP faults. Coming to the 3rd row of the table in figure (29), we can observe that it is not possible to completely detect all the faults in the unmodified design. And in Lin-testable and C-testable designs, using a reasonable number of tests it is possible to get 100% coverage and hence they are full testable designs.

Circuit property	Circuit type	8-bit ALU	16-bit ALU
Number of gates overhead (%)	Unmodified	481	915
	Lin-testable	494 (2.7)	1052 (15.0)
	C-testable	603 (25.4)	1249 (36.5)
Number of IP faults	Unmodified	752	1624
	Lin-testable	824	1768
	C-testable	824	1768
Number of IP tests and coverage (%)	Unmodified	312 (85.11)	672 (83.5)
	Lin-testable	384 (100)	816 (100)
	C-testable	195 (100)	195 (100)

Fig. 29. GATE OVERHEAD, FAULT NUMBERS, AND FAULT COVERAGE OF VARIOUS ALU DESIGNS

REFERENCES

[1] W. AI-Assadi, A. P.Jayasumana and Y. K.Malaiya, "Pass-transistor logic design", International Journal of Electronics, 1991, vol. 70, no. 4, pp. 739-749.

[2] A. Morgenshtein, A. Fish, and I. Wagner, "Gate-diffusion input (GDI):a power-efficient method for digital combinatorial circuits," IEEE Transactions on Very Large Scale Integration (VLSI) Systems IEEE Trans. VLSI Syst., vol. 10, no. 5, pp. 566–581, 2002.

[3] N. Weste, D. Harris, CMOS VLSI Design a Circuits and Systems Perspective, 4th Ed, Addison-Wesley, 2011

[4] M. A. Ahmed and M. A. Abdelghany, "Low power 4-Bit Arithmetic Logic Unit using Full-Swing GDI technique," in Proceedings of 2018 International Conference on Innovative Trends in Computer Engineering, ITCE 2018, 2018, vol. 2018–March, no. Itce, pp. 193–196.

[5] Mahmoud Aymen Ahmed , M. A. Mohamed El-Bendary , Fathy Z. Amer , Said M. Singy :Delay Optimization of 4-Bit ALU Designed in FS-GDI Technique

[6] Wang, L.T., Wu, C.W. and Wen, X., 2006. VLSI test principles and architectures: design for testability. Elsevier.

[7] T. Sridhar and J. P. Hayes, "Design of easily testable bit-sliced systems," IEEE Trans. Comput., vol. C-30, pp. 842–854, Nov. 1981.

[8] A. D. Friedman, "Easily testable iterative systems," IEEE Trans. Comput., vol. C-22, pp. 1061–1064, Dec. 1973.

[9] R. D. Blanton and J. P. Hayes, "Properties of the input pattern fault model," in Proc. 1997 Int. Conf. Computer Design, Oct. 1997, pp. 372–380.

[10] R. Blanton and J. P. Hayes, "Testability of onvergent tree circuits," IEEE Trans. Comput., vol. 45, pp. 950–963, Aug. 1996.

[11] R. D. Blanton and J. P. Hayes, "Design of a fast, easily testable ALU," in Proc. 14th VLSI Test Symp., Apr. 1996, pp. 9–16.

[12] R. D. Shawn Blanton and J. P. Hayes "On the Design of Fast, Easily Testable ALUs", IEEE Transactions on VLSI SYSTEMS, VOL. 8, NO. 2, APRIL 2000

[13] Ravi L.S, Chitra C P, Chandana., "Optimized ALU with BIST Implementation Using Cadence Encounter Platform" International Journal of Research and Scientific Innovation (IJRSI), Volume IV, Issue VIS, June 2017

[14] Seongmoon Wang, "A BIST TPG for Low Power Dissipation and High Fault Coverage," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.15, no.7,pp.777,789,July2007.

[15] Jamuna S, Dinesha P, Kp Shashikala, Kishore Kumar K, "Design and Implementation of BIST logic for ALU on FPGA", IEEE.

[16] E. Cerny, M.Aboul Hamid, G.Bois, J.Cloutier, "BIST in CMOS ALU", IEEE Design and Test of Computers.

[17] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for low power cmos circuits", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 15:665– 671, 1996..

[18] Y.-T. Ho and T.-T. Hwang, "Low power design using dual threshold voltage", In Proceedings of the Asia and South Pacific Design Automation Conference, pages 205–208, 2004.

[19] H. Jiang, M. Marek-Sadowska, and S. Nassif. "Benefits and costs of power-gating technique", IEEE International Conference on Computer Design: VLSI, 2005 in Computers and Processors, pages 559– 566, 2005.

[20] V. Tiwari, S. Malik, and P. Ashar, "Guarded evaluation: Pushing power management to logic synthesis/design", 9th International Symposium on Low Power Design, pages 221–226, 1995.

[21] L. Shang, L. Peh, and N. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks", In Proceedings of International Symposium on High-Performance Computer Architecture, pages 91–102, 2003

[22] Yu Zhou, Hui Guo, "Application Specific Low Power ALU Design", IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008

[23] NUR UMAIRA BINTI ZULKIFLI, "A LOW POWER AND FAST CMOS ARITHMETIC LOGIC UNIT", University Tun Hussein Onn Malaysia

[24] Priyanka Yadav, Gaurav Kumar, Sumita Gupta, "Design and Implementation of 4-Bit Arithmetic and Logic Unit Chip with the Constraint of Power Consumption", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)

[25] Balaji Ramakrishna S, Adiniru Guru Prasad, Anand P, "High Performance GDI-ALU Using 10T Adder Cells", 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT-2018), 2018

[26] Sreeja S Kumar, Rakesh S, "Design of a 4-bit Arithmetic and Logic Unit using 9T Full Adder with Optimized Area and Speed", International Journal of Engineering and Advanced Technology (IJEAT), 2019

[27] M. Shoba and R. Nakkeeran, "GDI based full adders for energy efficient arithmetic applications," Eng. Sci. Technol. an Int. J., vol. 19, no. 1, pp. 485–496, 2016.