Speed and Energy Optimization of a Carry Look-Ahead Adder (CLA) Using Uniform-Size Modules

Noura khdour 1212072

Dept. Of Electrical and Computer Engineering

Birzeit University

Jerusalem ,Palestine

1212072@student.birzeit.edu

***Abstract*-We all know that the carry look ahead adder is the one of the faster adders. In this paper we have implemented a 1-bit, 4-bits and 8-bits CLA circuit using static CMOS logic using Electric VLSI tool. This paper discuss the architecture and operation of the CLA adder focusing on its ability to generate carry signals in parallel. The use of uniform-size modules in the CLA adder enables optimized speed and energy consumption, making it a preferred choice for high-speed arithmetic operations in modern processors. Both carry look ahead and their component were simulated using LTspice software with 300nm CMOS technology, and power supply of 5Volts, leading to a delay of 0.000238884ps for 1 and 4 bit CLA, 0.00167219ps for 8-bit CLA and power consumption of 5W .**

Keywords: CLA, HA

I- INTRODUCTON

Modern microprocessors need to perform arithmetic operations quickly, efficiently using less energy, and without taking up too much space.Among these operations, adding binary numbers is the most fundamental , as many other calculations rely on addition. Addition is also important for tasks like cryptography parallel computing, and processing digital signals.Therefore designing adders that are fast, efficient, and use minimal space has become a major focus for researchers**[3].** So we introduce the carry look ahead as the best option.

A carry-lookahead adder or fast adder is a type of electronics adder used in digital logic A CLA improves speed by reduce the amount required to determine carry bit. It can be contrasted with the simpler but usually slower, ripple-carry adder for which the carry bit is calculated alongside the sum bit, and each stage must wait until the previous carry bit has been calculated to begin calculating its own sum bit and carry bit. The carry-lookahead Adder calculates one or more carry bits before the the sum which reduce the wait time for result**. [1]**

II- IMPLEMENTATION

To design an optimized carry look ahead we need several logic gates such as OR,AND and XOR gates and from these logic gates I implement the main component. My CLA adder contain three main parts first, PG generators , second , carry generator and third sum generator.

A- PG generators

The PG generators is a half adder. The half adder is a basic building block and smallest component.It have a AND and XOR gates .it takes two binary input and provides two.

The sum(P) output is generated from XOR gate and the carry (G) is generated from AND gate.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 1: truth table for HA

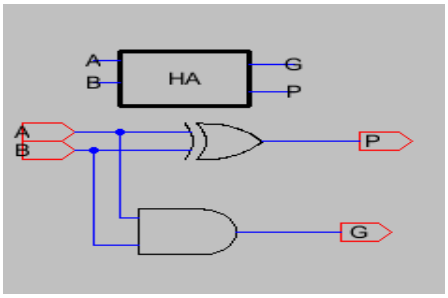


Figure 2: Half adder schematic

B- Carry generator

In this part I decided to divide the carry generator block into smaller blocks to make it easy for later when build the layout and the whole design.

A-B: Carry one generator (C1)

This part is build from AND and OR gates it takes three binary inputs and it produce the first carry out.

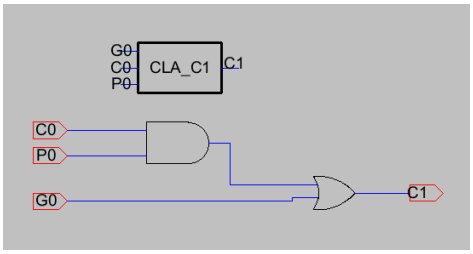


Figure 3: Carry one schematic

B-B- Generator of carry two(C2)

This part is build from 3-inputs ,2- inputs AND gate and 3-inputs OR gate it takes five binary input to produce the second carry

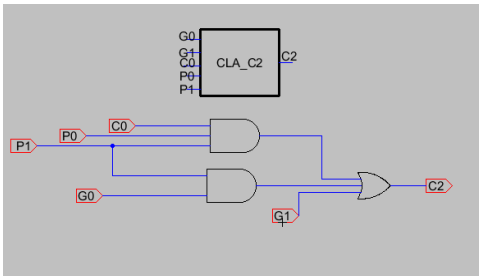


Figure 4: Carry two schematic

C-B- Generator of carry three(C3)

This part is build from 3-inputs,4-inputs,2-inputs AND gate and 4-inputs OR gates It takes seven binary input to produce the third carry.

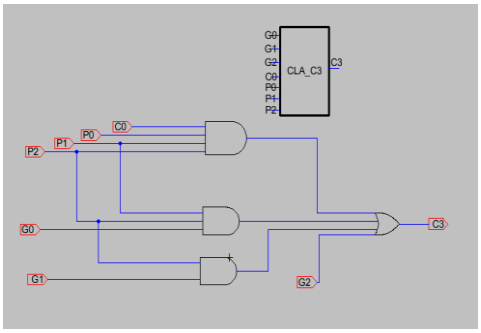


Figure 5: Carry three schematic

D-B- Generator of carry four(C4)

This part is build from 5-inputs, 4-inputs,3-inputs,2-inputs AND gate and 5-inputs OR gate . It takes 9 binary input to produce the fourth output.

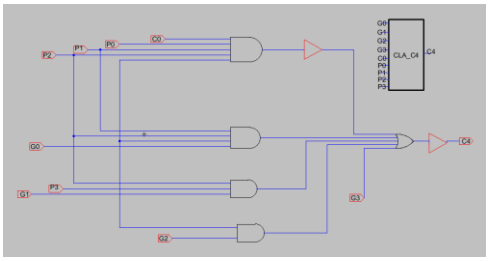


Figure 6: Carry four schematic

C-Sum generator

The main component in this part is XOR gate which is take the carry and propagate signal to produce the sum.

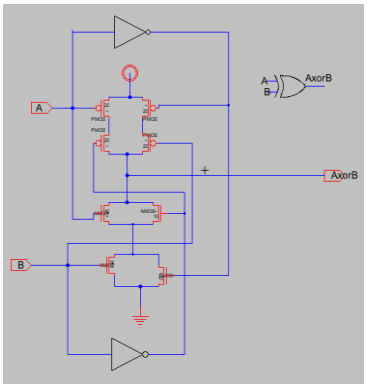


Figure 7: XOR gate schematic

III- THE PROPOSED STRUCTURE

A-Proposed 4-bit carry look ahead adder

The architecture of a 4-bit carry look ahead adder is important in designing larger adders, as 4-bit adders serve as fundamental building blocks. Therefore, If we optimizing the performance of a 4-bit CLA adder that can lead to significant improvements in the overall efficiency and speed of wider adder designs.

The traditional execution of CLA uses carry-generate (Gi) and carry-propagate terms (Pi) which are produced from half adder.If the input bits are denoted as Ai and Bi, then Gi and Pi terms can be expressed as the following equations

$$Gi=Ai.Bi \quad (1)$$

$$Pi=Ai\oplus Bi \quad (2)$$

The carry-out bits for CLA adder can be written in general as:

$$Ci+1=Gi+PiCi \quad (3)$$

Using equation (1), (2) and (3), carry-out equations for 4-bit CLA can be written as**:[2]**

$$C1=G0+P0C0 \quad (4)$$

$$C2=G1+P1G0+P1P0C0 \quad (5)$$

$$C3=G2+P2G1+P2P1G0+P2P1P0C0 \quad (6)$$

$$C4=G3+P3G2+P3P2G1+P3P2P1G0+P3P2P1P0C0 \quad (7)$$

And the sum for CLA can be written in general as:

$$Si=Pi\oplus Ci \quad (8)$$

All these equations are build above as shown in figures 3,4,5 and 6

As shown below in Fig 8 that my 4 bit CLA have four HA , four carries generator and four XOR gates .As you can see that carry is generated in parallel using the generate and propagate signals that produced from HA and the sum is calculated using the propagate signal and the computed carry.

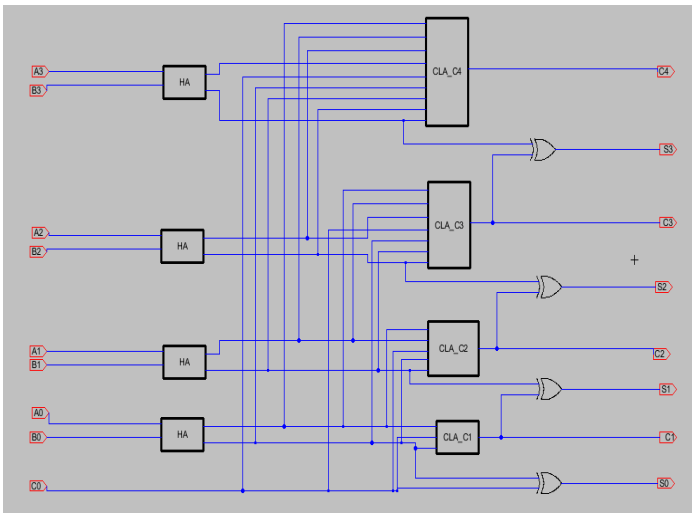


Figure 8: 4-bit carry look ahead adder

B- Proposed 8-bit carry look ahead adder

We can easily build a 8-bit CLA adder by using two 4-bit CLA adder . Using the carry out (C4) from the first adder as the carry-in for the second CLA adder ,then Combine the sum outputs to get the final 8-bit result.

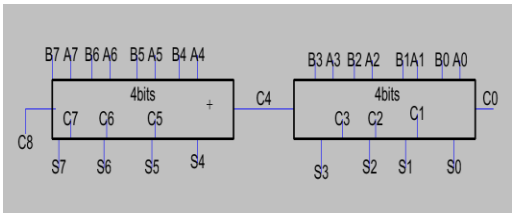


Figure 9:Icon of 8-bit CLA

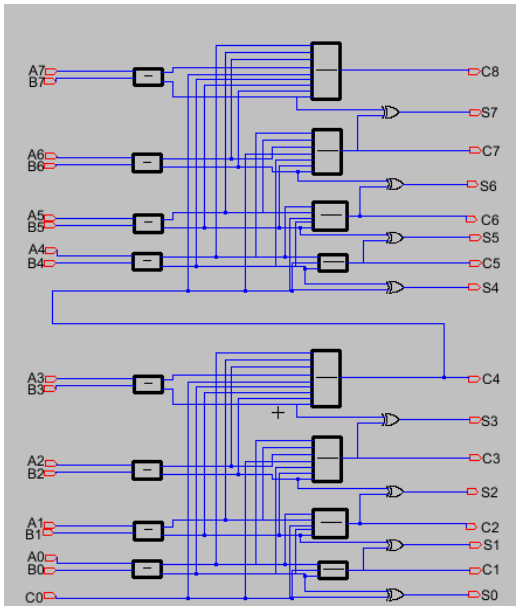


Figure 10:8-bit CLA

C- Proposed 1-bit carry look ahead adder

Actually , building a 4-bit adder from 1-bit adders is straightforward, it is not efficient .Cascading 1-bit CLAs would introduce significant delay. Each 1-bit CLA would have to wait for the carry signal from the previous stage, resulting in a ripple effect and slower overall operation. Additionally, cascading 1-bit CLAs would increased complexity and reduced performance.

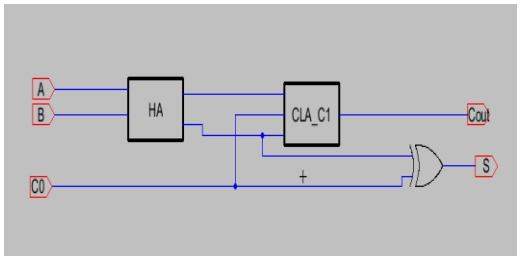


Figure 12:1-bit CLA

IV-LAYOUT

In this section, the layout of the design is shown.

A- Half adder layout

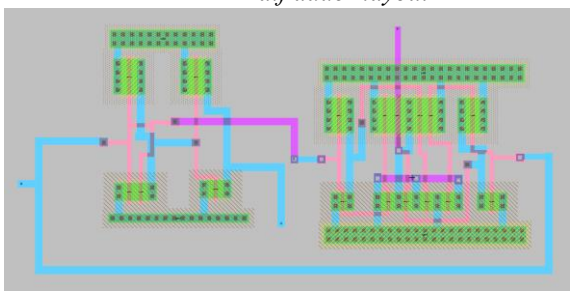


Figure 13:HA layout

B- Carry one generator layout

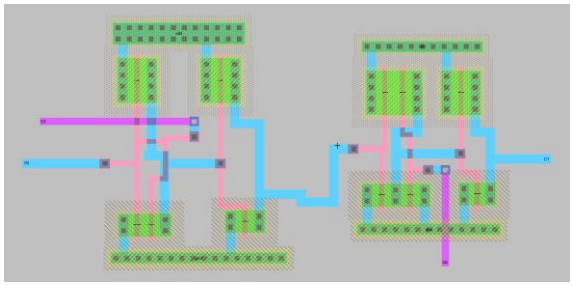


Figure 14: C1generator layout

C-Carry two generator layout

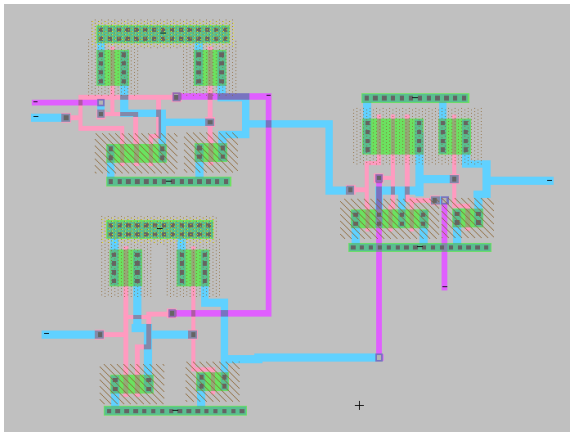


Figure 15: C2generator layout

D- Carry three generator layout

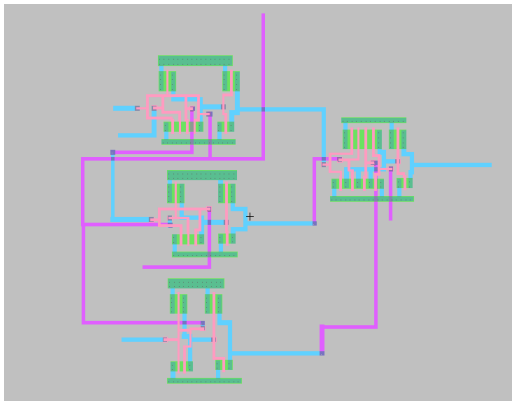


Figure 16: C3 generator layout

E- Carry four generator layout

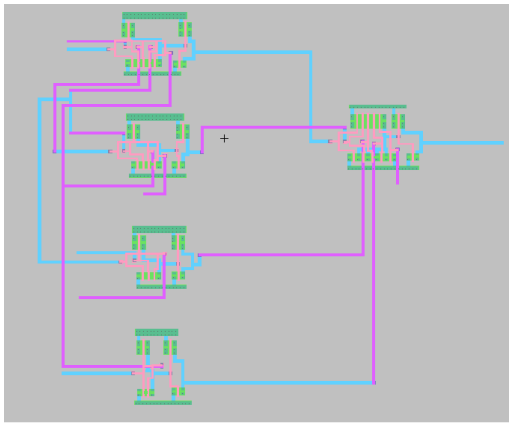


Figure 17: C4 generator layout

F-1-bit carry look ahead adder layout

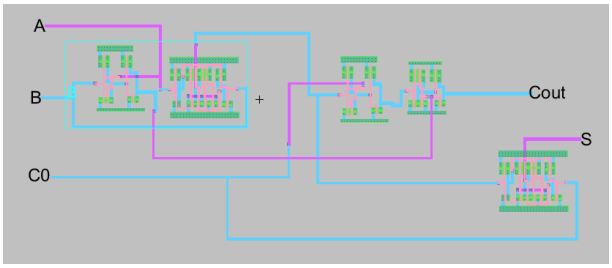


Figure 18: 1-bit CLA layout

G-4-bit carry look ahead layout

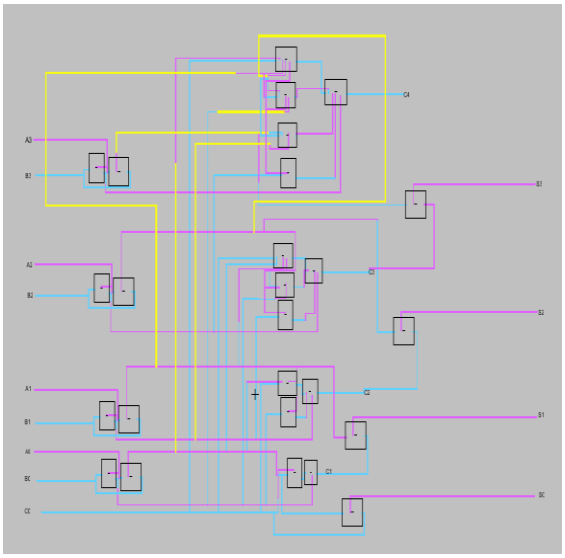


Figure 19: 4-bit CLA layout

H-8-bit carry look ahead layout

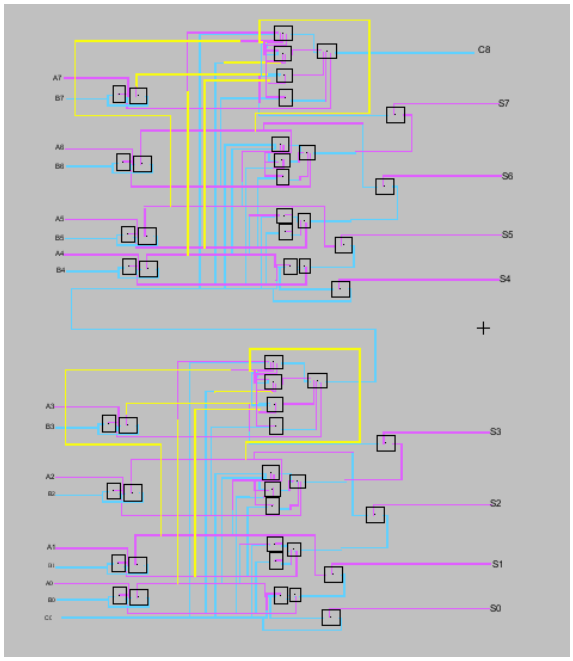


Figure 20: 8-bit CLA layout

V- SIMULATION BASED ON SCHEMATIC

Simulation is an important step in verifying the functionality of a designed circuit before physical implementation. To determine the improvement the proposed CLA adder can bring we need to simulation the design after implemented it ,In order to do so the simulation has been conducted by using LTspice tool with the technology scale of 300nm.The source voltage used is 5v .The CMOS size width are 10,20 for NMOS and PMOS respectively.

A- Simulation of 1-bit CLA adder

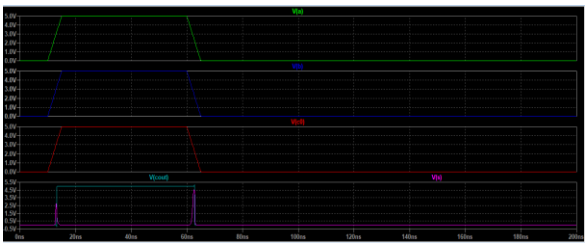


Figure 21: 1-bit CLA simulation

B- Simulation of 4-bit CLA adder

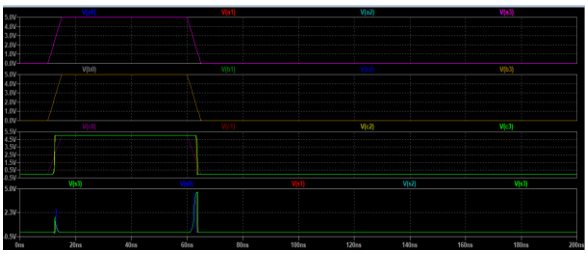


Figure 22: 4-bit CLA simulation

C- Simulation of 8-bit CLA adder

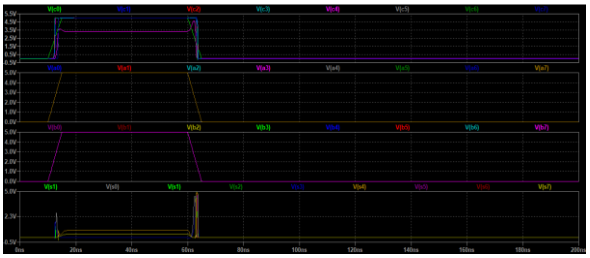
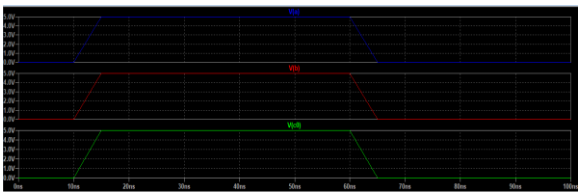


Figure 23: 8-bit CLA simulation

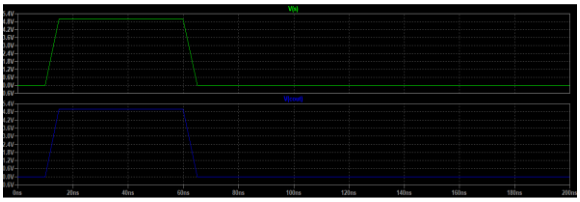
VI- SIMULATION BASED ON LAYOUT

In this section ,beside on simulation the layout using LTspice software we need also to measure the area ,power ,rise time, fall time and the propagation delay based on layout using spice code The results appear as the following:

A- Simulation of 1-bit CLA adder



(a)Inputs



(b) output

Figure 24: 1-bit CLA simulation layout

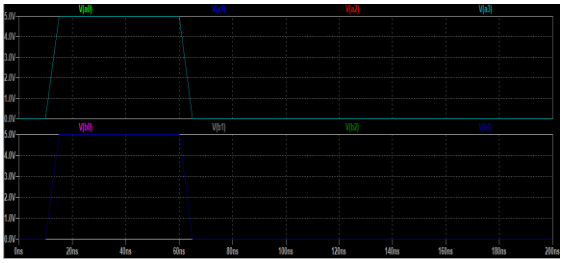
```
vdd vdd 0 DC 5
vA A 0 pwl 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
vB B 0 pwl 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
vC0 C0 0 pwl 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
.measure tran rise_time_S TRIG v(S) VAL=0.5 RISE=1 TARG v(S) VAL=4.5 RISE=1
.measure tran fall_time_S TRIG v(S) VAL=4.5 FALL=1 TARG v(S) VAL=0.5 FALL=1
.measure tran delay_S TRIG v(A) VAL=2.5 RISE=1 TARG v(S) VAL=2.5 RISE=1
.measure tran power_Dissipation AVG   v(vdd) *i(vdd)
.tran 0 0.1us
.include C:\Electric\C5_models.txt
```

Figure 25: code of simulation

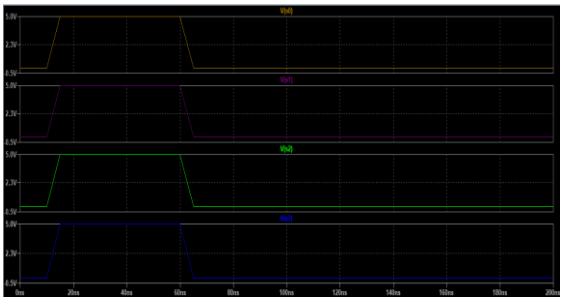
```
rise_time_s=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
fall_time_s=4e-009 FROM 6.05e-008 TO 6.45e-008
delay_s=2.38884e-016 FROM 1.25049e-008 TO 1.25049e-008
power_dissipation: AVG(v(vdd))=5 FROM 0 TO 1e-007
```

Figure 26: 1-bit CLA Measurements

B-Simulation of 4-bit CLA adder



(a) 4-bit inputs



(b) 4-bit output

Figure 27: 4-bit CLA Simulation layout

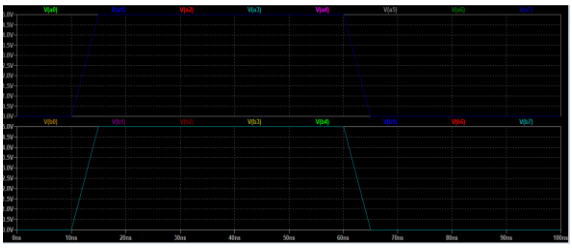
```
Vdd vdd 0 DC 5
VA0 A0 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VA1 A1 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VA2 A2 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VA3 A3 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VB0 B0 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VB1 B1 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VB2 B2 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
VB3 B3 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
Vc0 C0 0 PWL 0 0 10n 0 15n 5 30n 5 60n 5 65n 0 100n 0
.measure tran rise_time_S0 TRIG v(S0) VAL=0.5 RISE=1 TARG v(S0) VAL=4.5 RISE=1
.measure tran rise_time_S1 TRIG v(S1) VAL=0.5 RISE=1 TARG v(S1) VAL=4.5 RISE=1
.measure tran rise_time_S2 TRIG v(S2) VAL=0.5 RISE=1 TARG v(S2) VAL=4.5 RISE=1
.measure tran rise_time_S3 TRIG v(S3) VAL=0.5 RISE=1 TARG v(S3) VAL=4.5 RISE=1
.measure tran fall_time_S0 TRIG v(S0) VAL=4.5 FALL=1 TARG v(S0) VAL=0.5 FALL=1
.measure tran fall_time_S1 TRIG v(S1) VAL=4.5 FALL=1 TARG v(S1) VAL=0.5 FALL=1
.measure tran fall_time_S2 TRIG v(S2) VAL=4.5 FALL=1 TARG v(S2) VAL=0.5 FALL=1
.measure tran fall_time_S3 TRIG v(S3) VAL=4.5 FALL=1 TARG v(S3) VAL=0.5 FALL=1
.measure tran delay_S TRIG v(A3) VAL=2.5 RISE=1 TARG v(S0) VAL=2.5 RISE=1
.measure tran power_Dissipation AVG  v(vdd) *i(vdd)
.tran 0  0.1us
.include C:\Electric\C5_models.txt
.end
```

Figure 28: code of Simulation

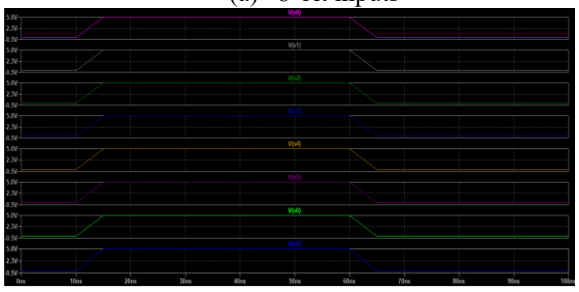
```
rise_time_s0=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s1=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s2=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s3=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
fall_time_s0=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s1=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s2=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s3=4e-009 FROM 6.05e-008 TO 6.45e-008
delay_s=2.38884e-016 FROM 1.25049e-008 TO 1.25049e-008
power_dissipation: AVG(v(vdd))=5 FROM 0 TO 1e-007
```

Figure 29: 4-bit CLA Measurements

C- *Simulation of 8-bit CLA adder*



(a) 8-bit inputs



(b)8-bit output

Figure 30: 8-bit CLA Simulation layout

```
VDD VDD 0 DC 5
VA0 A0 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA1 A1 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA2 A2 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA3 A3 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA4 A4 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA5 A5 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA6 A6 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VA7 A7 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB0 B0 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB1 B1 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB2 B2 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB3 B3 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB4 B4 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB5 B5 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB6 B6 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VB7 B7 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
VC0 C0 0 PWL 10N 0 15N 5 30N 5 60N 5 65N 0 100N 0
.MEASURE TRAN RISE_TIME_S0 TRIG V(S0) VAL=0.5 RISE=1 TARG V(S0) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S1 TRIG V(S1) VAL=0.5 RISE=1 TARG V(S1) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S2 TRIG V(S2) VAL=0.5 RISE=1 TARG V(S2) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S3 TRIG V(S3) VAL=0.5 RISE=1 TARG V(S3) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S4 TRIG V(S4) VAL=0.5 RISE=1 TARG V(S4) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S5 TRIG V(S5) VAL=0.5 RISE=1 TARG V(S5) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S6 TRIG V(S6) VAL=0.5 RISE=1 TARG V(S6) VAL=4.5 RISE=1
.MEASURE TRAN RISE_TIME_S7 TRIG V(S7) VAL=0.5 RISE=1 TARG V(S7) VAL=4.5 RISE=1
.MEASURE TRAN FALL_TIME_S0 TRIG V(S0) VAL=4.5 FALL=1 TARG V(S0) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S1 TRIG V(S1) VAL=4.5 FALL=1 TARG V(S1) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S2 TRIG V(S2) VAL=4.5 FALL=1 TARG V(S2) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S3 TRIG V(S3) VAL=4.5 FALL=1 TARG V(S3) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S4 TRIG V(S4) VAL=4.5 FALL=1 TARG V(S4) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S5 TRIG V(S5) VAL=4.5 FALL=1 TARG V(S5) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S6 TRIG V(S6) VAL=4.5 FALL=1 TARG V(S6) VAL=0.5 FALL=1
.MEASURE TRAN FALL_TIME_S7 TRIG V(S7) VAL=4.5 FALL=1 TARG V(S7) VAL=0.5 FALL=1
.measure tran delay_S TRIG v(A0) VAL=2.5 RISE=1 TARG v(S7) VAL=2.5 RISE=1
.measure tran power_Dissipation AVG  v(vdd) *i(vdd)
.tran 0 0.1us
.INCLUDE C:\Electric\C5_models.txt
```

Figure 31: code of Simulation

```
rise_time_s0=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s1=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s2=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s3=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s4=4.00781e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s5=4.00782e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s6=4.00782e-009 FROM 1.0501e-008 TO 1.45088e-008
rise_time_s7=4.00782e-009 FROM 1.0501e-008 TO 1.45088e-008
fall_time_s0=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s1=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s2=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s3=4e-009 FROM 6.05e-008 TO 6.45e-008
fall_time_s4=4e-009 FROM 6.04999e-008 TO 6.44999e-008
fall_time_s5=4e-009 FROM 6.04999e-008 TO 6.44999e-008
fall_time_s6=4e-009 FROM 6.04999e-008 TO 6.44999e-008
fall_time_s7=4e-009 FROM 6.04999e-008 TO 6.44999e-008
delay_s=1.67219e-015 FROM 1.25049e-008 TO 1.25049e-008
power_dissipation: AVG(v(vdd))=5 FROM 0 TO 1e-007
```

Figure 32: 8-bit CLA Measurements

VII-RESULTS ANALYSIS AND COMPARISON

A-Area

The compactness and optimization of the 8-bit CLA is a result of dividing the module to submodules of 4-bit CLA which simplifies the overall design, where each block can be optimized for the desired lowest size, then combined efficiently.Also,reducing the number of pmos that used in AND gate that would reduce the number of required transistor thus reduce the area. In addition, building the CLA using half adder instead of full adder that will also reduce required transistor.

Parameter	Value (um^2)
Cell Layout Area 1x1 CLA	(346.35um*142.5um)
Cell Layout Area 4x4 CLA	(1128um*615.45um)
Cell Layout Area 8x8 CLA	(1657.2um*1333.8um)

Table1:Area of different size CLA

B-Power

In a Carry Look-Ahead Adder (CLA), power consumption is influenced by several factors, including the speed of computation and the complexity of the carry-generation logic.The CLA computes carry signals in parallelThis parallel processing involves additional gates and interconnections, leading to higher dynamic power consumption due to increased switching activity.After simulated the layout I foumd that Power Dissipation is 0.005uW

C-Delay/speed

The delay in a Carry Look-Ahead Adder (CLA) is significantly reduced compared to traditional ripple carry adders due to its efficient carry computation mechanism.

Parameter	Value (ps)
1x1 CLA	0.000238884
4x4CLA	0.000238884
8x8CLA	0.00167219

Table2:Delay of different size CLA

VIII-CONCLUSION

In this paper, a detailed 8-bit CLA architecture have been proposed using CMOS technology, with multiple 4-bit CLA . By focusing on balancing the three desired metrics to meet the demands,The CLA remains a highly effective solution for achieving faster addition in digital circuits, striking a balance between speed, area, and power efficiency. Its design makes it a preferred choice in applications requiring high-speed arithmetic operations.

NOTE: Similarity report is 14%

REFERENCES

[1] V. Pudi and K. Sridharan, "Low Complexity Design of Ripple Carry and Brent–Kung Adders in QCA," IEEE Transactions on Nanotechnology, vol. 11, no. 1, pp. 105-119, 2012.

[2] M. Hasan, M. S. Islam, M. R. Ahmed, “Performance improvement of 4-bit static CMOS carry look-ahead adder using modified circuits for carry generate and propagate terms,” Science Journal of Circuit, Systems and Signal Processing, vol. 8, no. 2, pp. 76-81, 2019.