VLSI Project 1: Report

Amit Vardhan Suryadevara

Department of Electrical and Computer Science, Purdue University

ECE 59500: VLSI

Prof. Saeed Mohammadi

November 14th, 2024

Abstract:

This project presents the design and analysis of an 8-bit Manchester adder implemented using complementary CMOS logic for Letni Corporation's high-performance chip. The design, optimized for speed and low power consumption, targets the accumulator section within a broader processing unit. Emphasizing static logic, the project explores complementary CMOS-based logic styles to achieve minimal propagation delay and high throughput, with design specifications assuming a 0.1 ns rise and fall time for the clock. Layout design, verification, and simulation are performed in Cadence Virtuoso, and the circuit's functionality is validated with a set of predefined hexadecimal input samples using the vector file. Power estimation, conducted both pre- and post-layout, provides insight into the adder's performance under various input conditions.

Introduction:

In modern digital systems, adders are essential for various arithmetic and logical functions, especially within accumulators in high-performance processors. Efficient adder designs contribute directly to the speed and power efficiency of a chip, making them critical in applications ranging from microcontrollers to high-speed computing systems. This project focuses on designing an 8-bit Manchester adder using Complementary CMOS transistor logic— CMOS logic consumes power only during the switching of states (i.e., dynamic power), making it highly efficient (consuming more power) \. In a static state, ideally, there is negligible power consumption. CMOS has excellent noise margins due to its high voltage swings between logic levels, which improves reliability in noisy environments. The design strategy aims to minimize propagation delay and optimize throughput, assuming a clock rise and fall time of 0.1 ns. To ensure high standards of reliability and functionality, the design undergoes a thorough layout, DRC, LVS verification, and post-layout simulations. Key specifications, such as propagation delay and power dissipation, are estimated with both pre-layout and post-layout testing to gauge the adder's real-world performance. The validation of the design's functionality involves applying a specific set of input patterns, ensuring robust performance across various input combinations. Power estimation for the Manchester adder considers different input vectors to reflect typical operational conditions

Procedure:

1.) To create the schematic of 8-Bit MCC using Complementary CMOS transistors, create NAND, NOR, INVERTER and XOR gates. The schematic is created using the Boolean expression for Carry, Delete, propagate and Sum,

1.) G = A.B   Generate carry,   (NAND Gate)

2.) D = Abar. bBar   delete carry, (NOR agte)

3.) Co $P = A \oplus B$ propagate carry (EXOR gate

4.) $Co = G + Pci$ (Carry)

5.) $S = P \oplus Ci$ (Sum)

These Boolean equations are used to construct the schematic.

Types of Gates	Number of gates	No of transistors in each gate
NOR	1	4
NAND	1	4
EXOR	3	12
Inverter	1	2
CMOS	0	4
Total		50

Total number of transistors in the 8-bit schematic= 50*8=400

The total number of transistors is 400 used in the 8-bit Manchester adder.

Types of Gates Used:

1. **AND Gates**:
    a. Number: 3
    b. Function: Perform logical multiplication, essential for generating intermediate signals used to determine the carry-out bit.
2. **OR Gates**:
    a. Number: 3

   b. Function: Combine the results of the intermediate signals to produce the final sum and carry-out bits.

3. **NOT Gate (Inverter)**:
   a. Number: 1
   b. Function: Invert the input signal, which is necessary for the logic operations within the adder.

Types of Transistors Used:

- **NMOS Transistors**:
  - Used in the pull-down network of logic gates.
  - Function: Ensure the outputs can be pulled down to ground effectively.
- **PMOS Transistors**:
  - Used in the pull-up network of logic gates.
  - Function: Ensure the outputs can be pulled up to the supply voltage efficiently.

Inputs:

   a. The inputs are typically the two binary digits (A and B) that need to be added, along with a carry-in bit (Cin).
   b. These inputs are processed through various gates to produce the output sum (S) and carry-out (Cout).

2. **Logic Gates Functionality**:
   a. **AND Gates**:

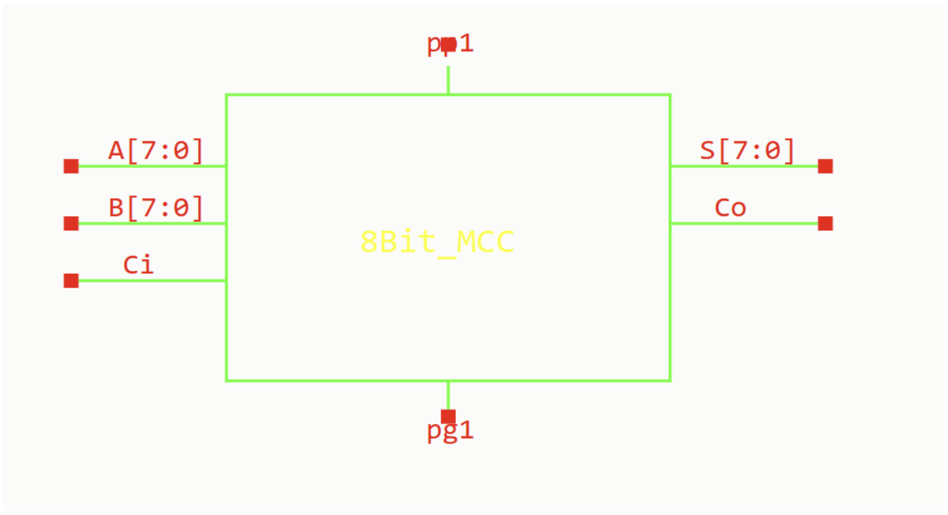
        i. These gates are used to generate the partial product terms, such as $A\cdot BA$ \cdot B, $A\cdot CinA$ \cdot Cin, and $B\cdot CinB$ \cdot Cin.
        ii. These terms are critical for determining whether a carry-out should be generated.

    b. **OR Gates**:
        i. These gates combine the intermediate terms to determine the final carry-out.

    c. **NOT Gate**:
        i. This gate inverts specific inputs to generate the required logical signals for the intermediate steps.
        ii. The NOT gate might invert one of the input signals for use in an AND gate, ensuring the correct operation of the sum bit calculation.

3. **Transistor-Level Implementation**:
    a. **NMOS Transistors**:
        i. Used in the pull-down networks within the AND and OR gates, controlling the flow of current and ensuring the output can be pulled down to ground when required.
    b. **PMOS Transistors**:
        i. Used in the pull-up networks within the AND and OR gates, ensuring the output can be pulled up to the supply voltage when needed.

Block Diagram:



Pre_Layout simulation:

Design specifications:

Clock specifications: Time period is 0.1 nSec

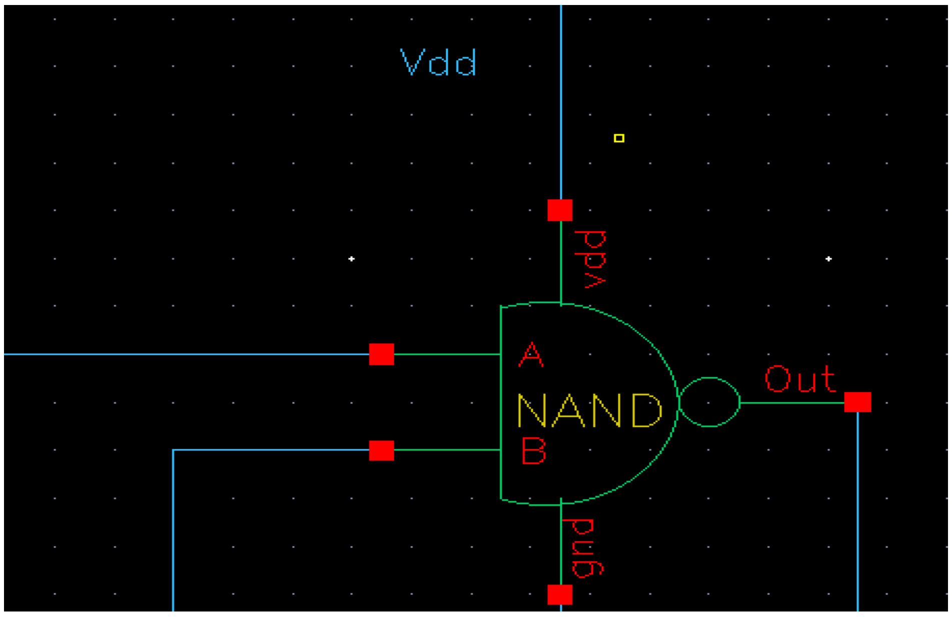
trise = tfall = 0.1nSec

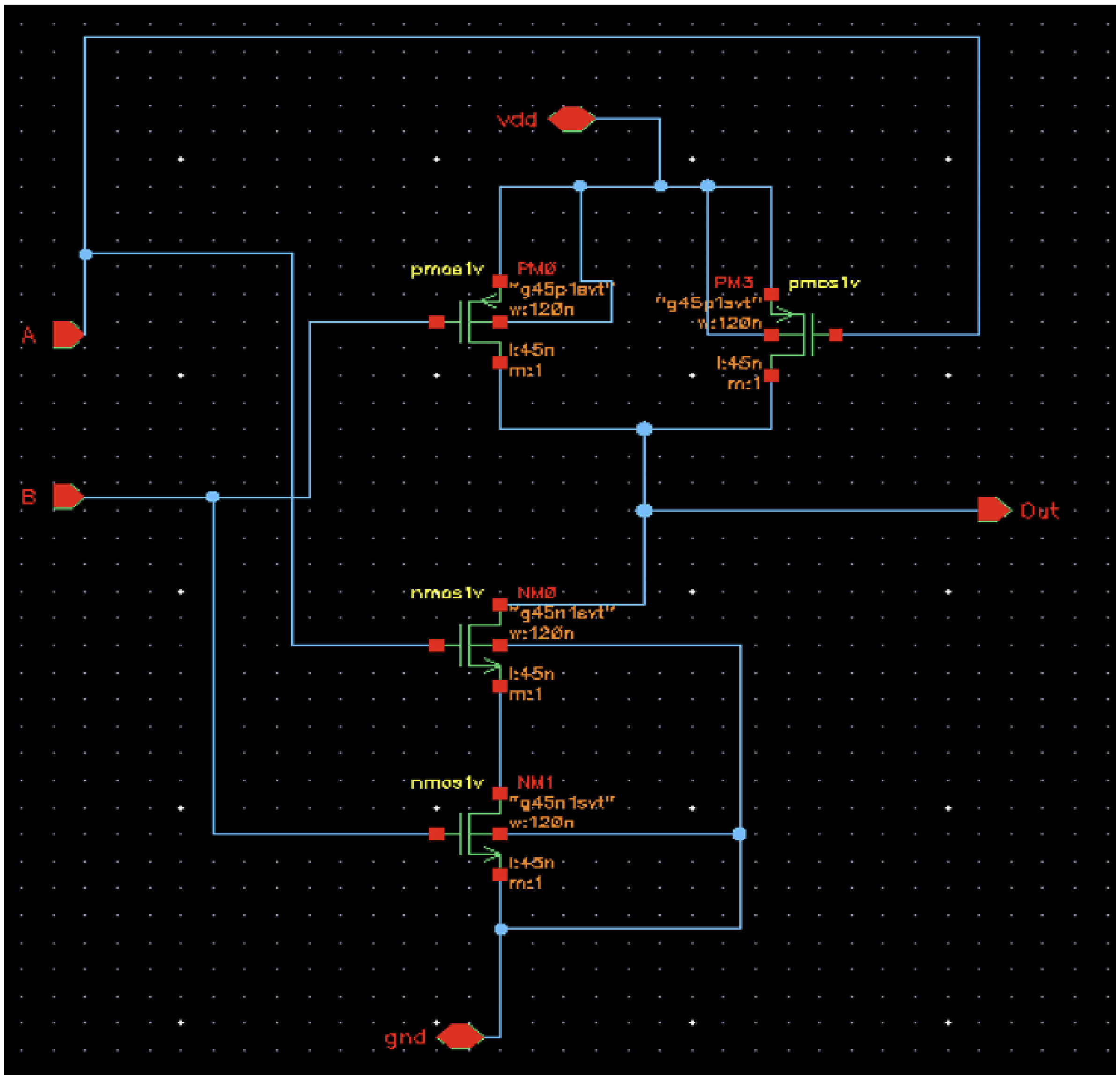
Logic : CMOS Transistors Supply

Voltage : 1.2 V

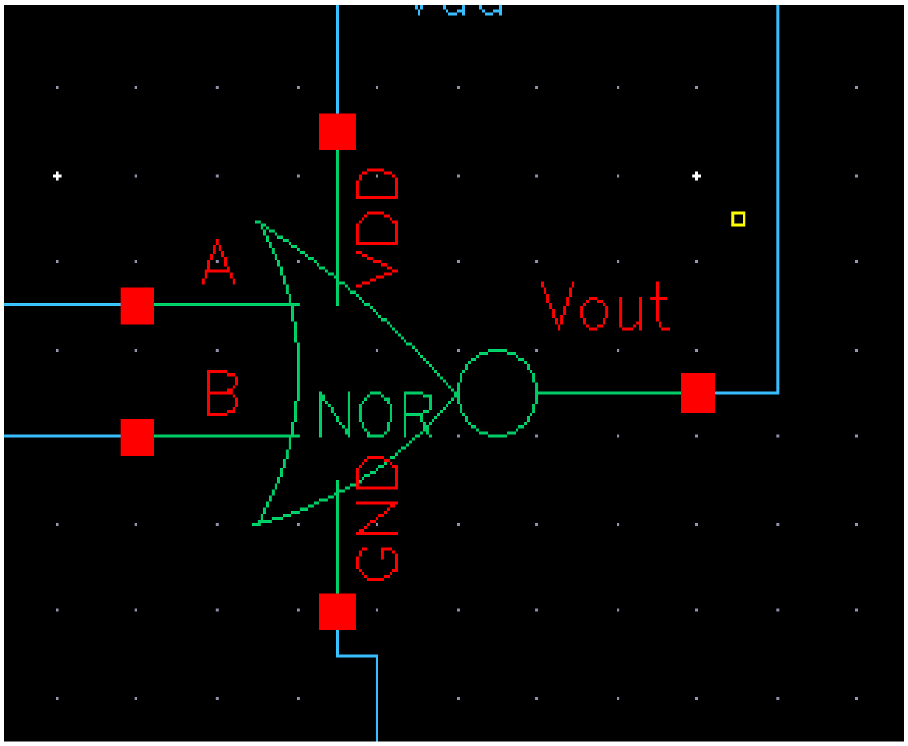
Schematic:

NAND Gate:

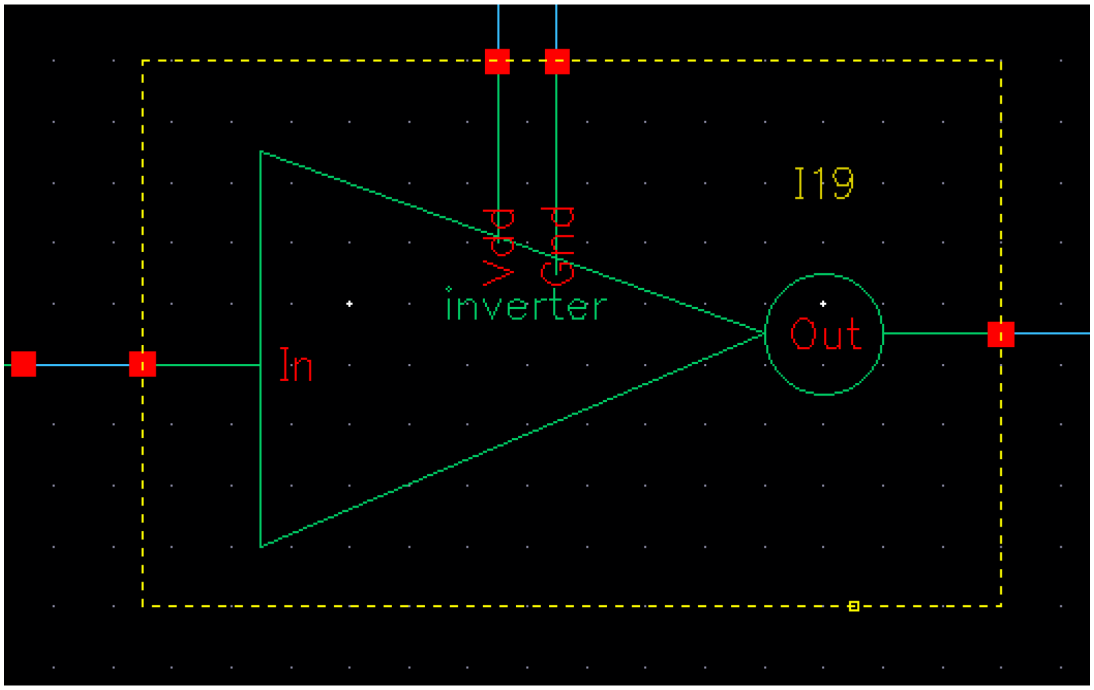

Vdd
vdd
A
NAND
B
Out
gnd


vdd
pmos1v
PM0
"g45p1svt"
w:120n
l:45n
m:1
PM3
pmos1v
"g45p1svt"
w:120n
l:45n
m:1
A
B
Out
nmos1v
NM0
"g45n1svt"
w:120n
l:45n
m:1
nmos1v
NM1
"g45n1svt"
w:120n
l:45n
m:1
gnd

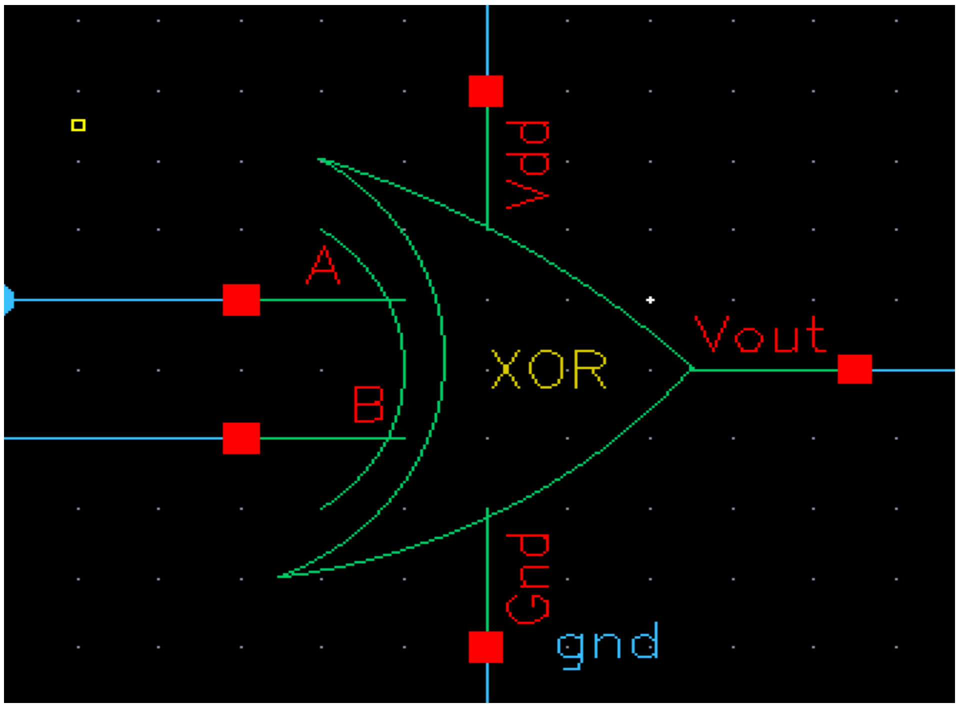
NOR GATE:



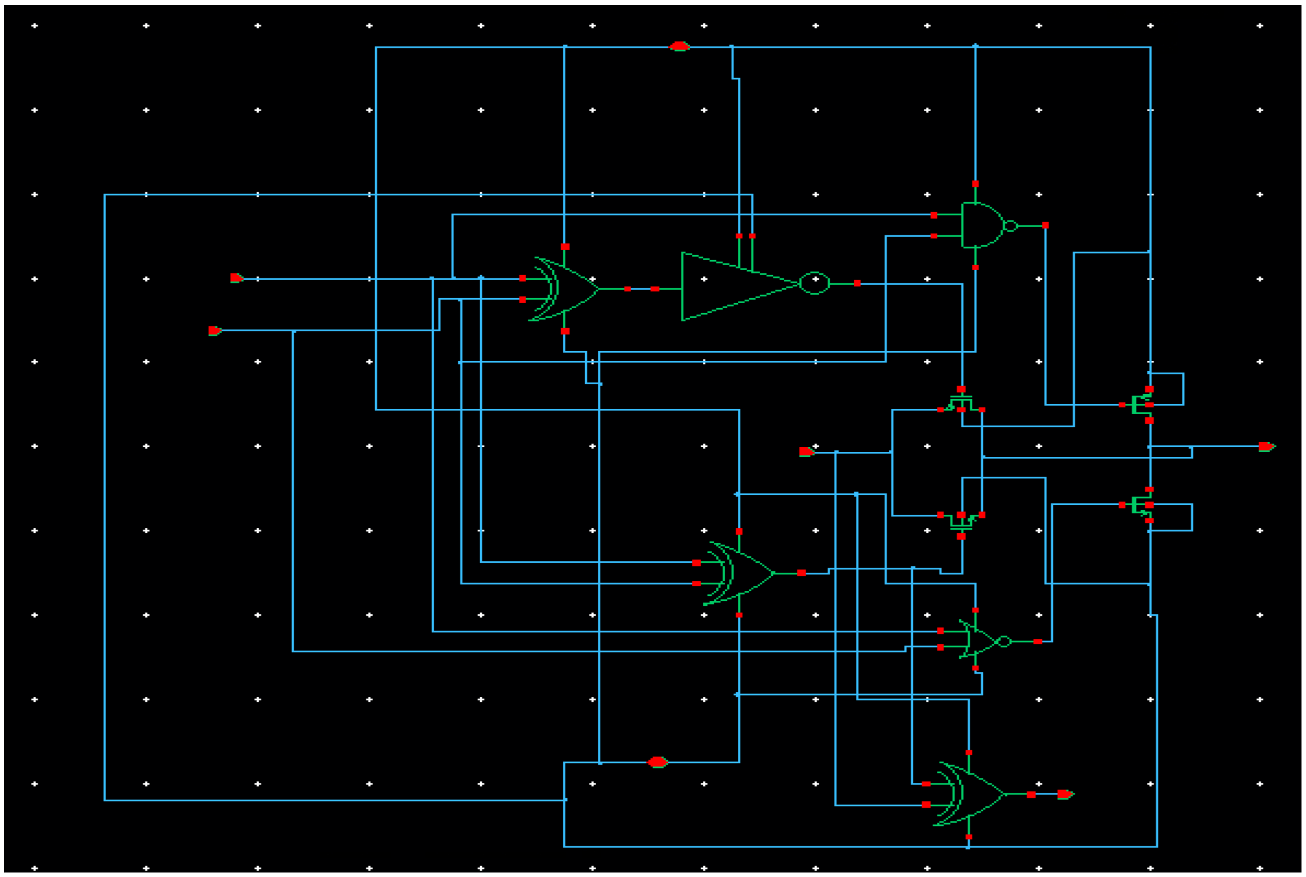
Inverter:



EXOR Gate:

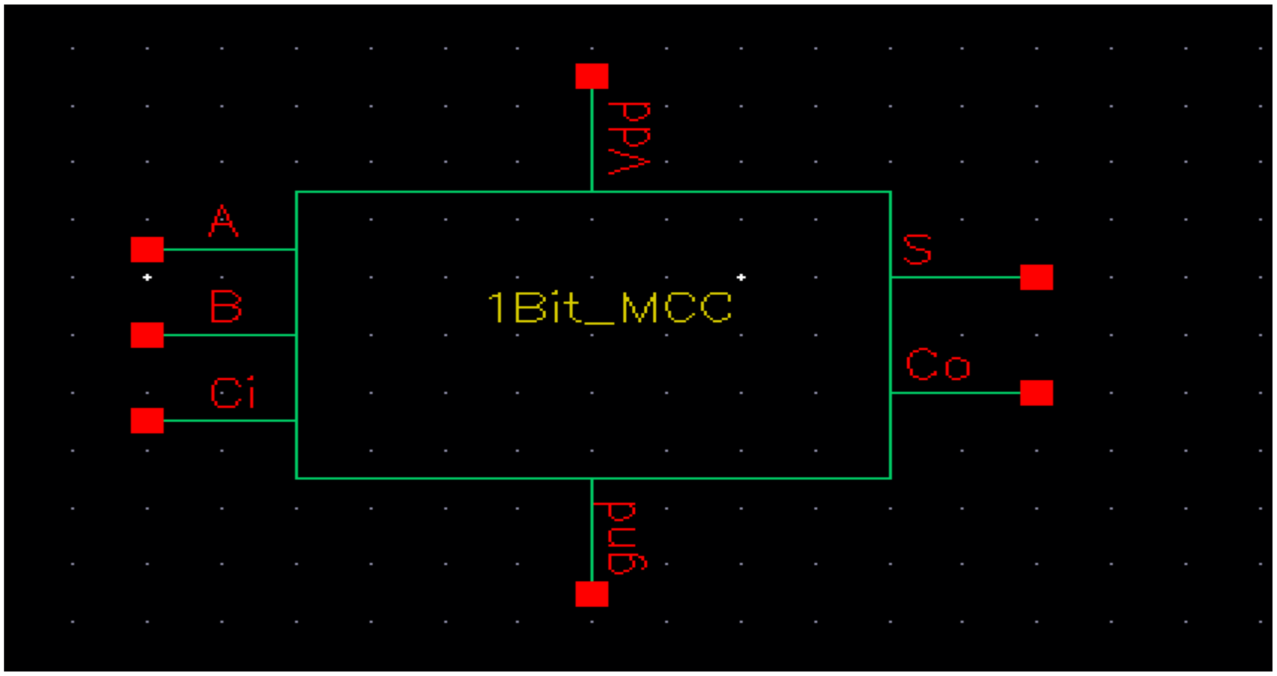


Schematic of 1-bit MCC using all the gates:

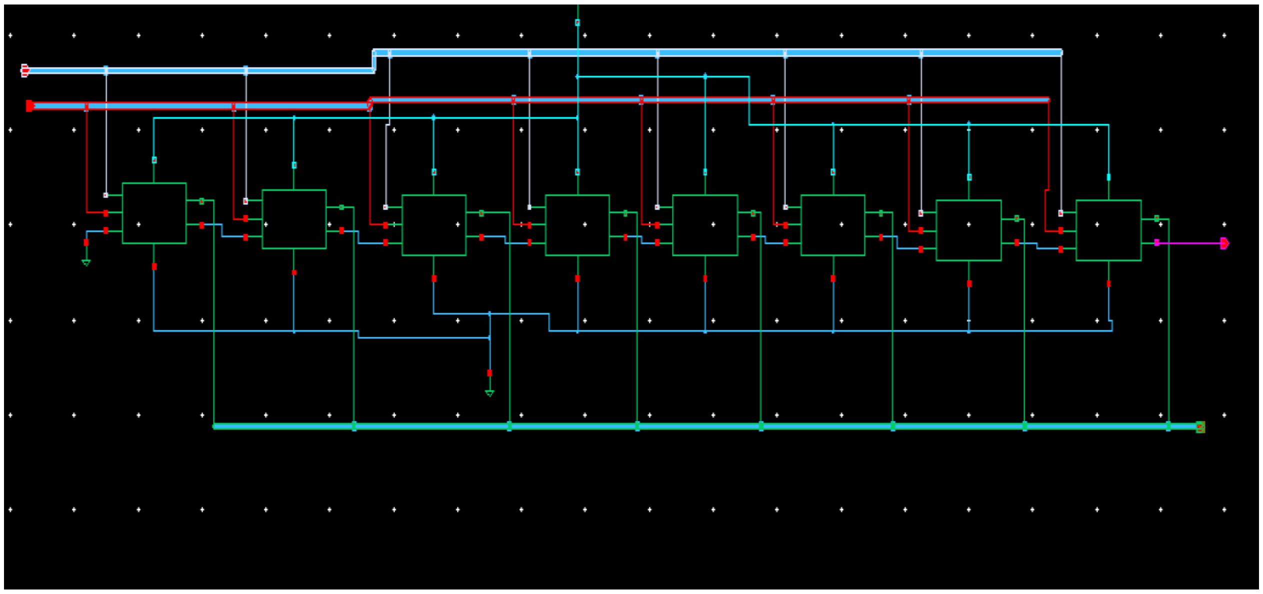


Total Number of Transistor count is 50

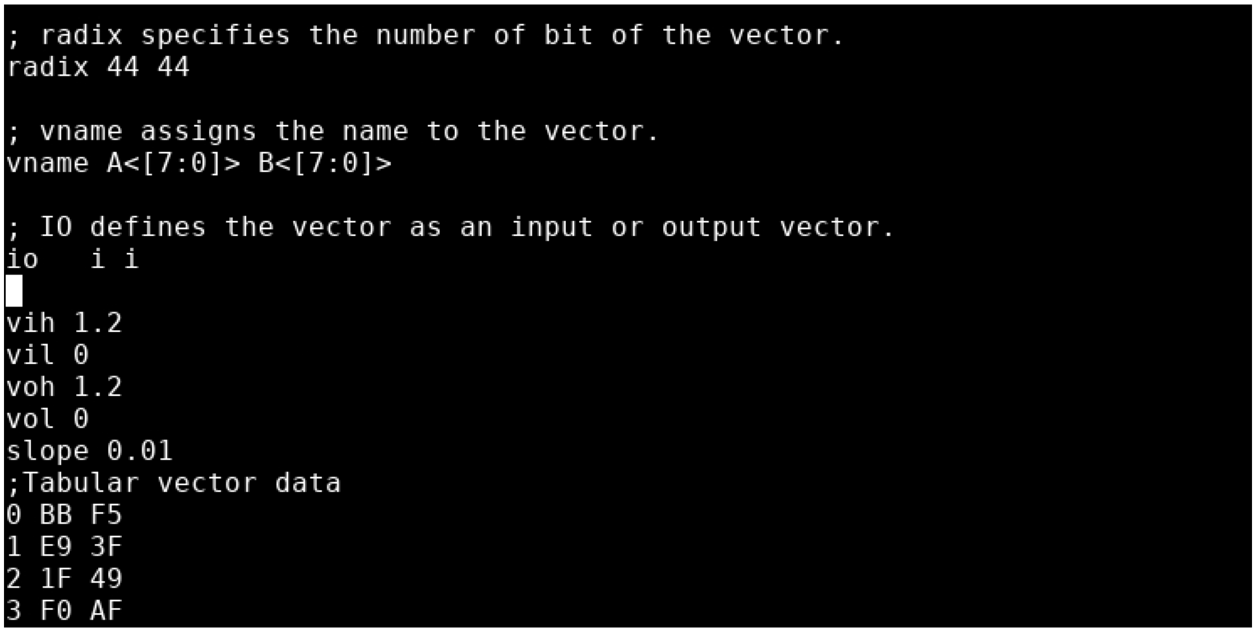
Symbol of 1 bit MCC



Using the symbol of 1-bit MCC, create an 8 bit schematic by connecting the 8 (1-Bit):



To generate the output using the vector file



```
; radix specifies the number of bit of the vector.
radix 44 44

; vname assigns the name to the vector.
vname A<[7:0]> B<[7:0]>

; IO defines the vector as an input or output vector.
io   i i

vih 1.2
vil 0
voh 1.2
vol 0
slope 0.01
;Tabular vector data
0 BB F5
1 E9 3F
2 1F 49
3 F0 AF
```

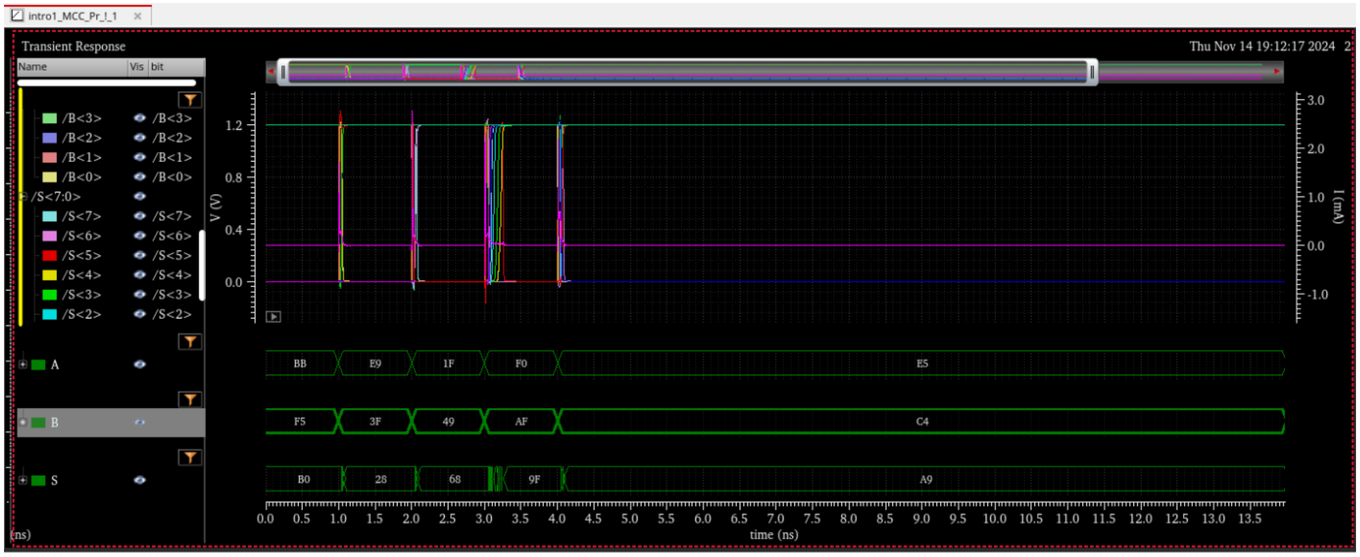
Output:

The output of the 8 bit MCC for the given pattern in the Hexadecimal.

A	**B**	**S**	**Co**
BB	F5	B0	1
E9	3F	28	1
1F	49	68	0
F0	AF	9F	1
E5	C4	A9	1
CB	11	DC	0
EE	33	21	1
11	00	11	0
1A	A3	BD	0
C4	6B	2F	1
1F	F1	10	1
55	AC	01	1
C3	E2	A5	1

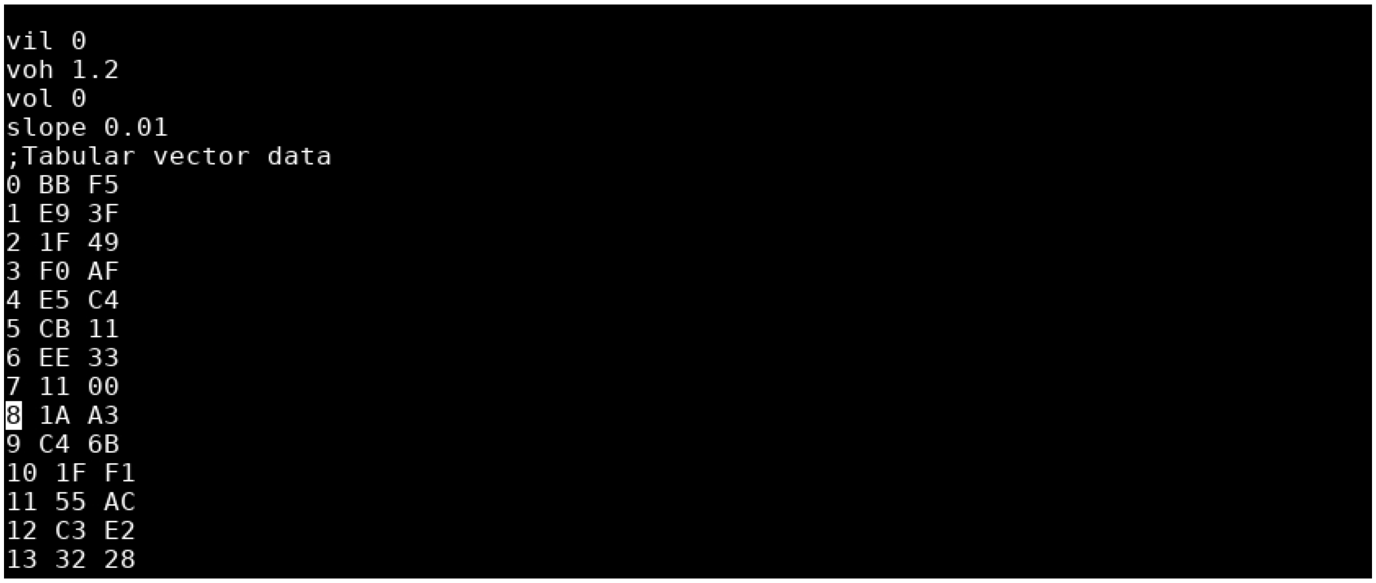
32	28	5A	0
FF	FF	FE	1

The output is shown in the graph:

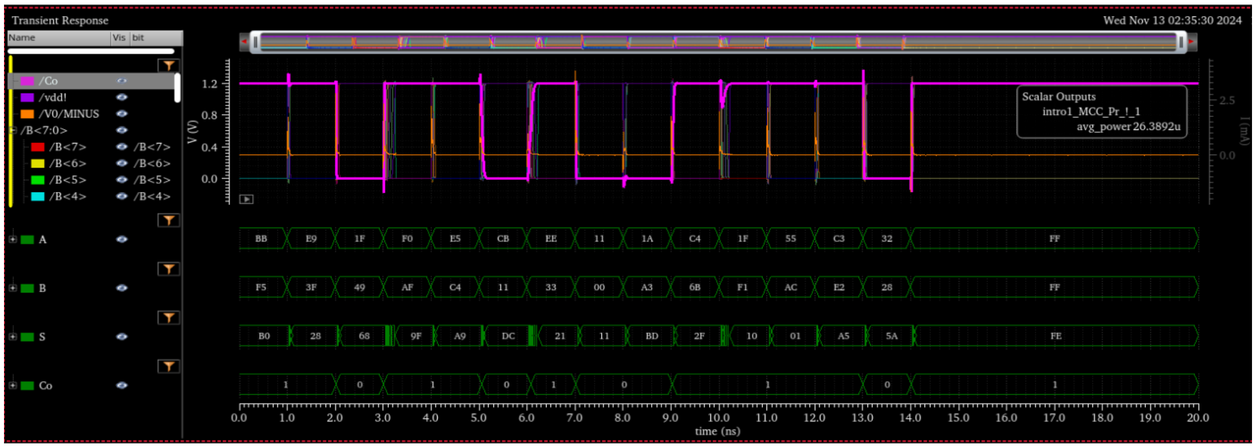


For entire inputs:

Vector file:



```
vil 0
voh 1.2
vol 0
slope 0.01
;Tabular vector data
0 BB F5
1 E9 3F
2 1F 49
3 F0 AF
4 E5 C4
5 CB 11
6 EE 33
7 11 00
8 1A A3
9 C4 6B
10 1F F1
11 55 AC
12 C3 E2
13 32 28
```



Explaination:

This output demonstrates that the primary benefit of the MCC adder is its reduced carry propagation delay. Traditional adders, like ripple-carry adders, suffer from slow carry propagation as each bit must wait for the previous bit's carry to be resolved. The MCC adder addresses this by using Manchester carry chains, which allow for faster carry signal propagation and thus quicker addition operations.
The waveform demonstrates rapid transitions in the carry-out signal, indicating a consistent and predictable performance. The MCC adder’s structured approach to carry propagation ensures reliable and repeatable timing characteristics, which are crucial for synchronous digital circuits. It also excels in scenarios where speed and efficiency are critical. Its ability to swiftly handle carry signals and lower power consumption makes it a preferred choice for high-performance digital systems, particularly in comparison to more straightforward but slower adder designs like ripple-carry adders.

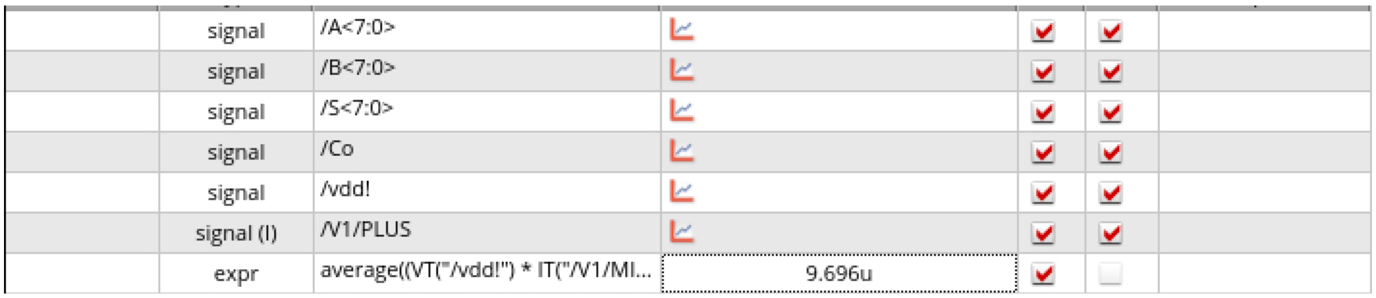
Average Power dissipation:

Calculating the average power dissipation of the 8-bit MCC using the calculator in cadence.
For 5 inputs:

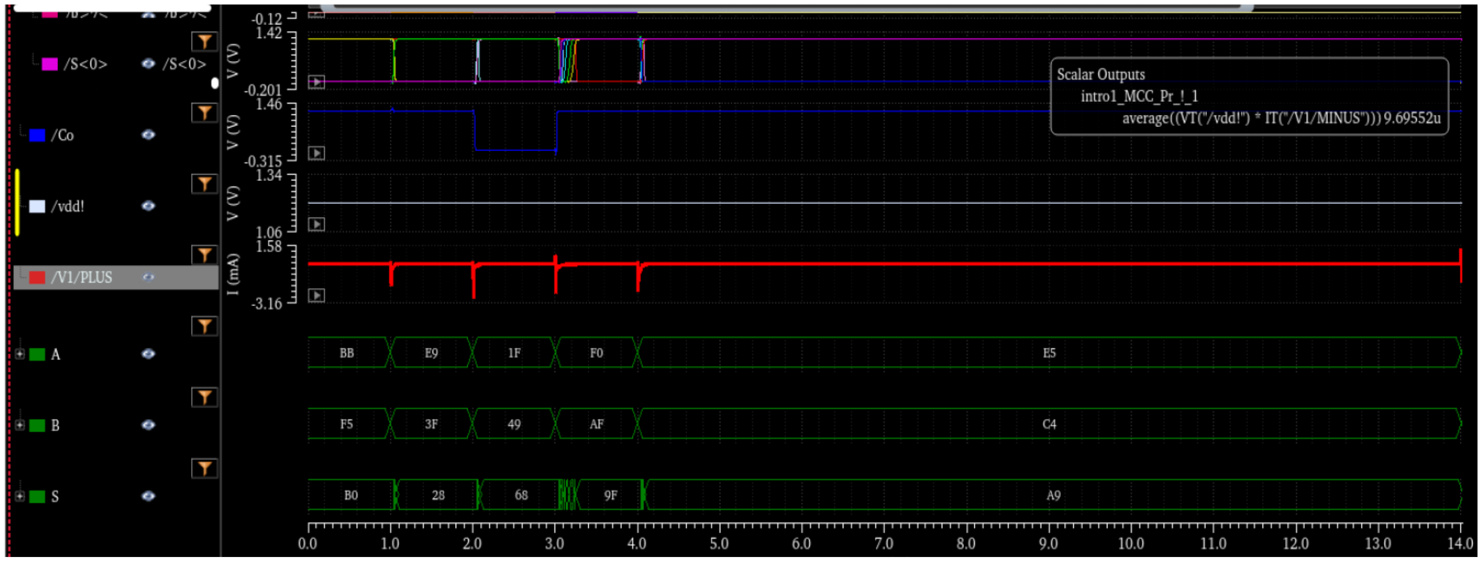
Formula:

average((VT("/vdd!") * IT("/V1/MINUS")))

Power generation:

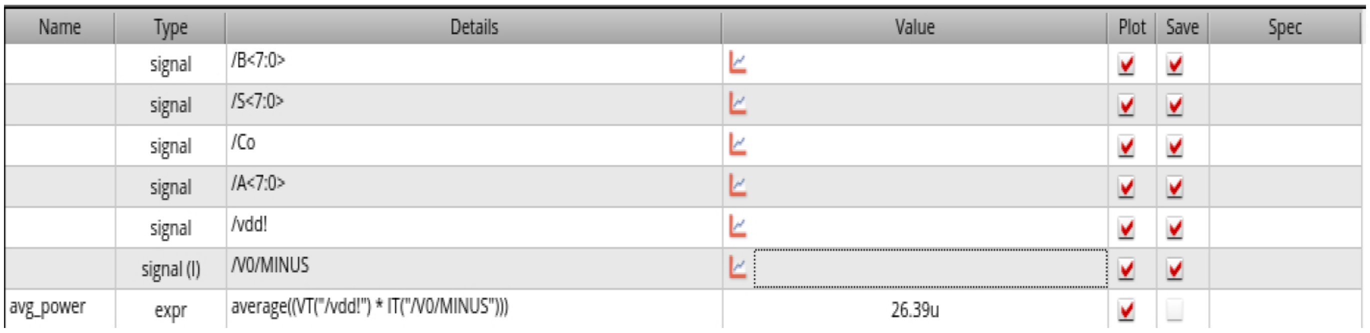


	signal	/A<7:0>	📈	✓	✓	
	signal	/B<7:0>	📈	✓	✓	
	signal	/S<7:0>	📈	✓	✓	
	signal	/Co	📈	✓	✓	
	signal	/vdd!	📈	✓	✓	
	signal (I)	/V1/PLUS	📈	✓	✓	
	expr	average((VT("/vdd!") * IT("/V1/MI...	9.696u	✓		

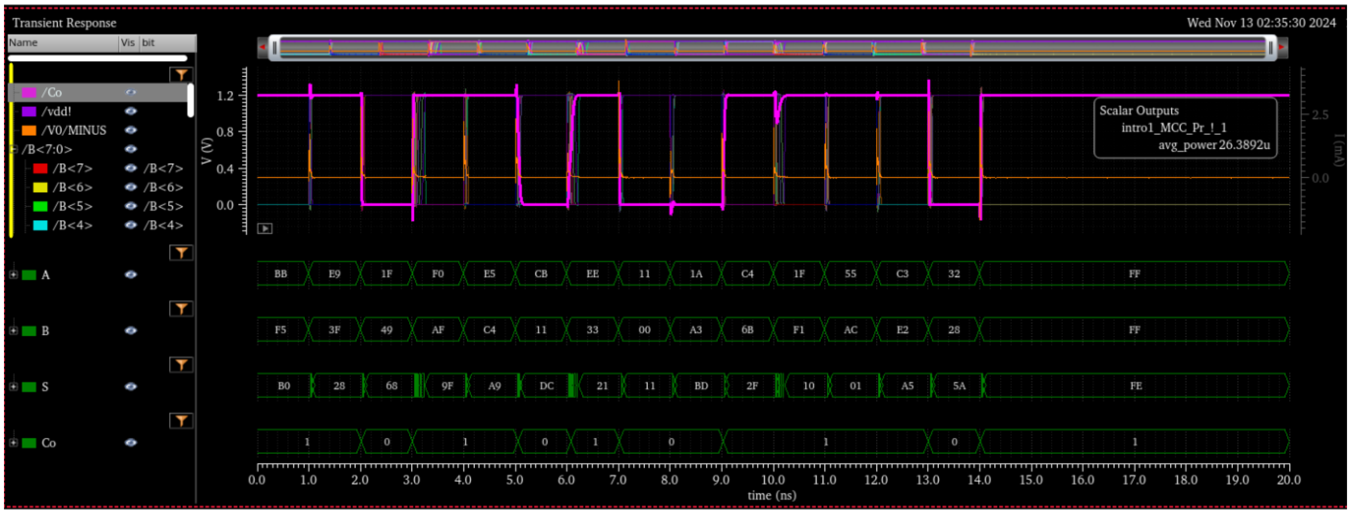


Scalar Outputs
intro1_MCC_Pr_!_1
average((VT("/vdd!") * IT("/V1/MINUS"))) 9.69552u

For 14 inputs:



Name	Type	Details	Value	Plot	Save	Spec
	signal	/B<7:0>	📈	✓	✓	
	signal	/S<7:0>	📈	✓	✓	
	signal	/Co	📈	✓	✓	
	signal	/A<7:0>	📈	✓	✓	
	signal	/vdd!	📈	✓	✓	
	signal (I)	/V0/MINUS	📈	✓	✓	
avg_power	expr	average((VT("/vdd!") * IT("/V0/MINUS")))	26.39u	✓		

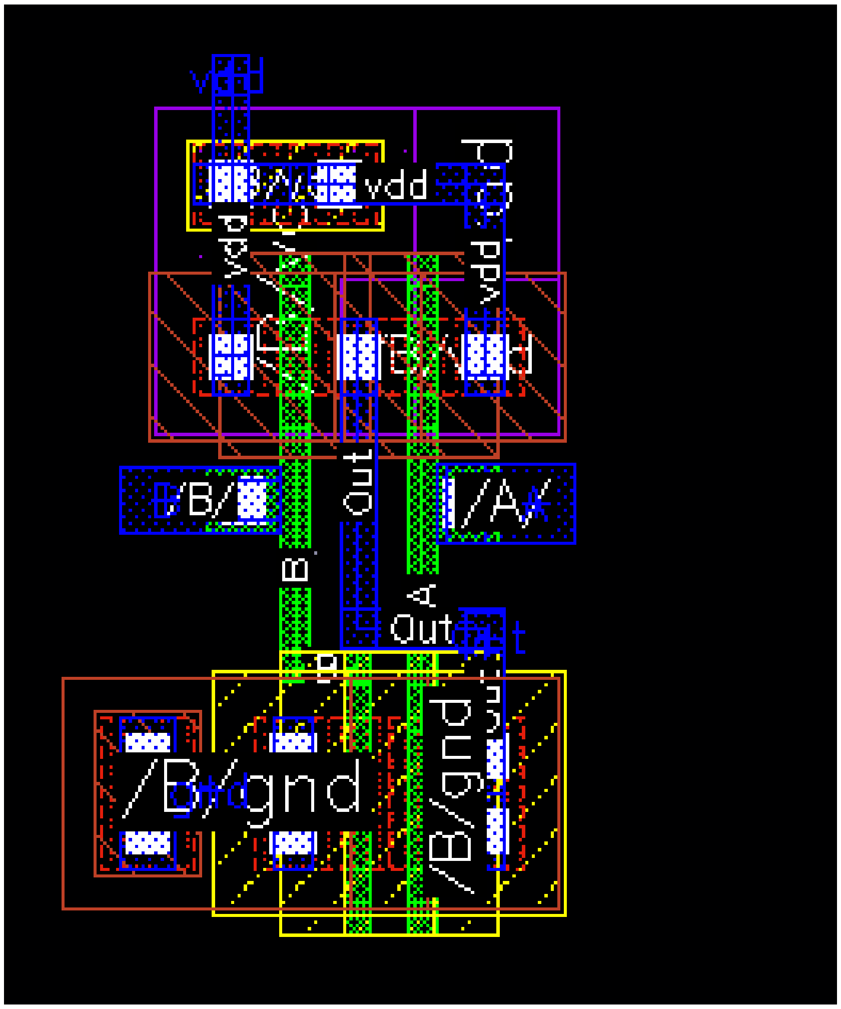


Explaination:
The difference in power consumption between the 5-input and 14-input sequences for the 8-bit MCC adder reflects the variations in switching activity and complexity of carry propagation across the adder stages. With the 5-input sequence resulting in 9.6 µW and the 14-input sequence resulting in 26 µW, the higher power usage for the 14-input sequence indicates increased switching activity, as more bits are likely toggling and generating carry operations. Since the power consumed in digital circuits largely depends on switching events, the number of transitions and the propagation of carries directly impact dynamic power consumption. This shows that the power consumption in CMOS is so high.

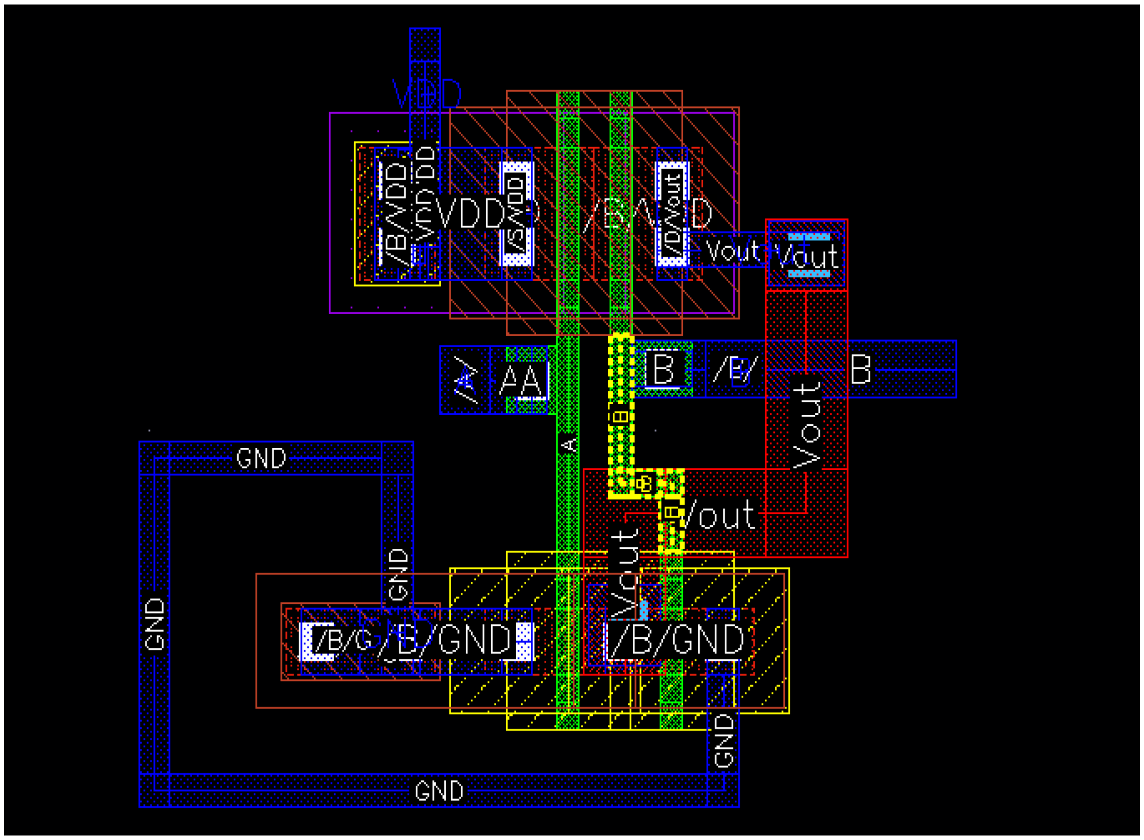
The 14-input sequence, more carry bits are generated and need to propagate through the carry chain, increasing the power due to more frequent switching in the transistors. This also leads to greater capacitive charging and discharging across the adder's circuitry, hence a rise in dynamic power. Conversely, the 5-input sequence has fewer bits actively engaged in generating and propagating carries, leading to a lower overall switching activity and therefore lower power consumption. This variation demonstrates how input sequences with more carry operations lead to greater dynamic power usage in adders like the MCC.
The 8-bit MCC adder's efficient carry propagation mechanism significantly reduces overall power consumption by ensuring quick and reliable carry signal transmission. This fast propagation is less power-intensive compared to ripple-carry adders, where each bit's addition depends on the previous bit. The MCC adder can compute carries more quickly than ripple-carry adders due to the Manchester Carry Chain, leading to improved speed.

Post Layout Simulation

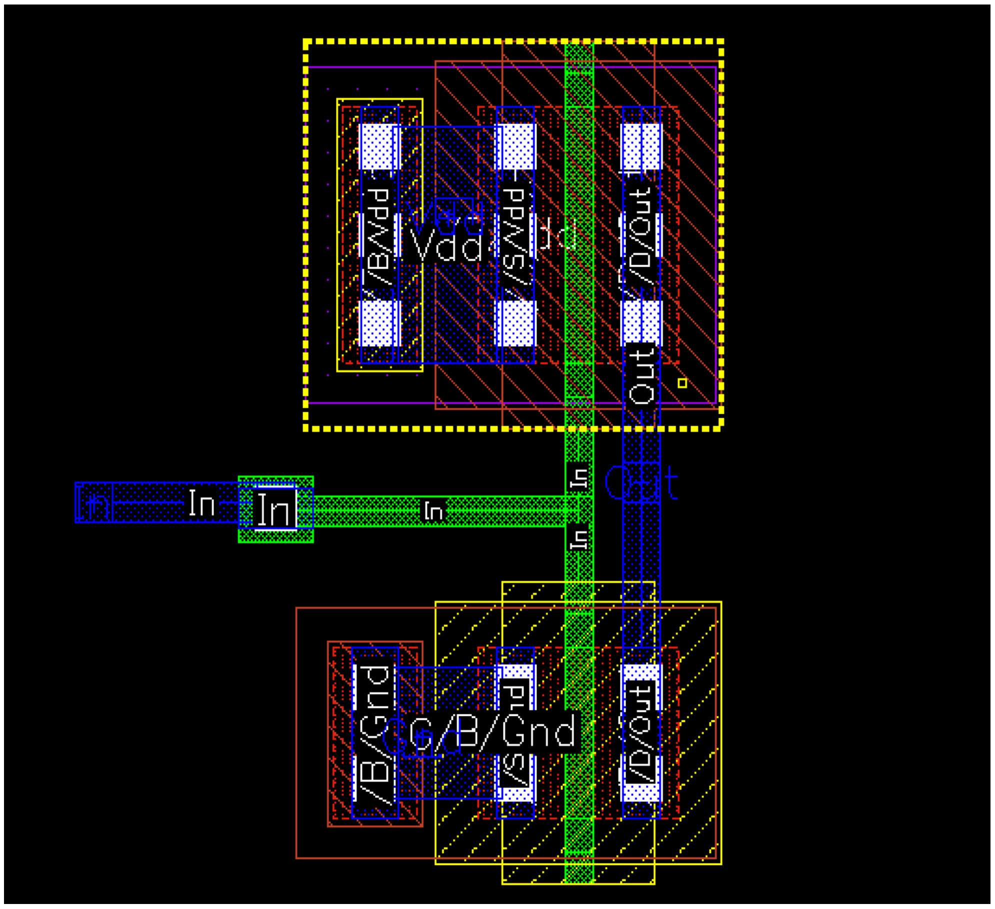
NAND GATE:



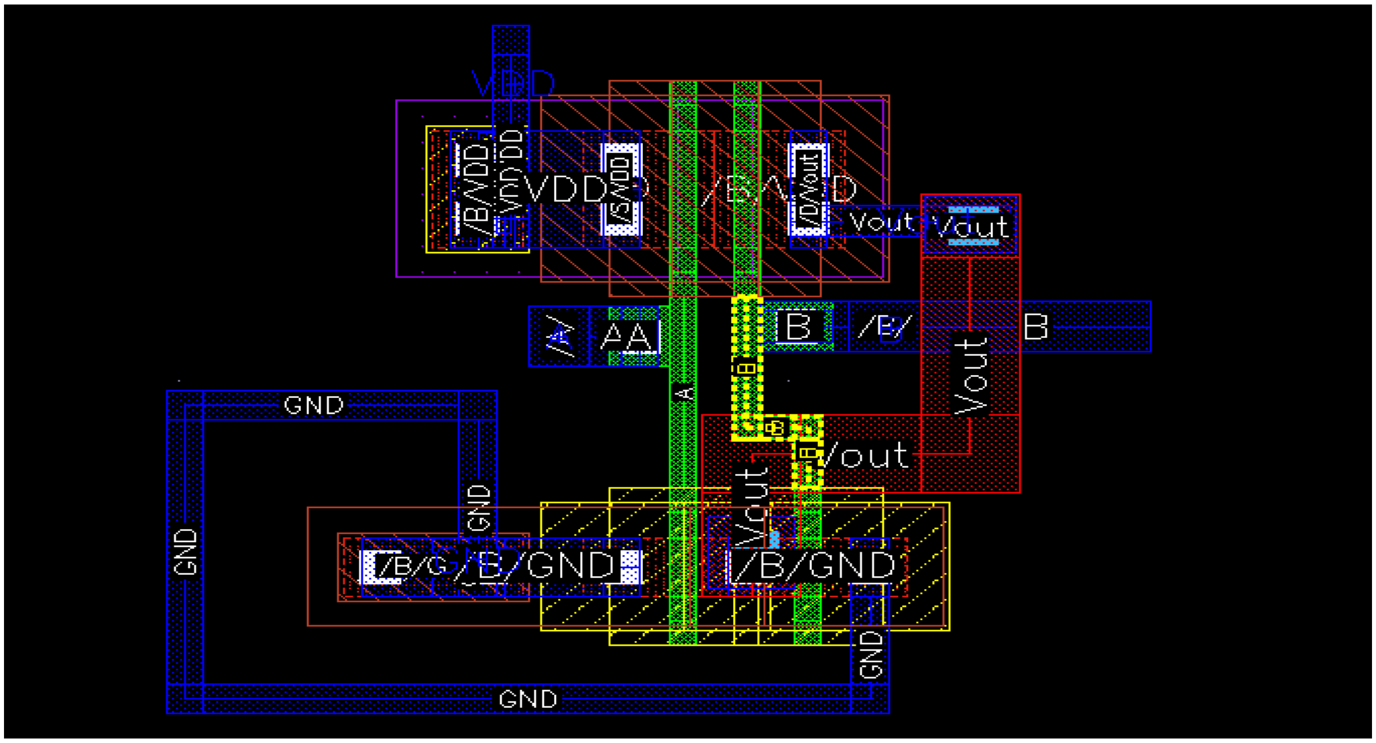
NOR Gate:


VDD
/B/VDD
VDD
/S/VDD
/B/VDD
/D/Vout
Vout
Vout
/A/
AA
B
/B/
B
Vout
A
GND
GND
Vout
Vout
GND
/B/G
/B/GND
/B/GND
GND
GND

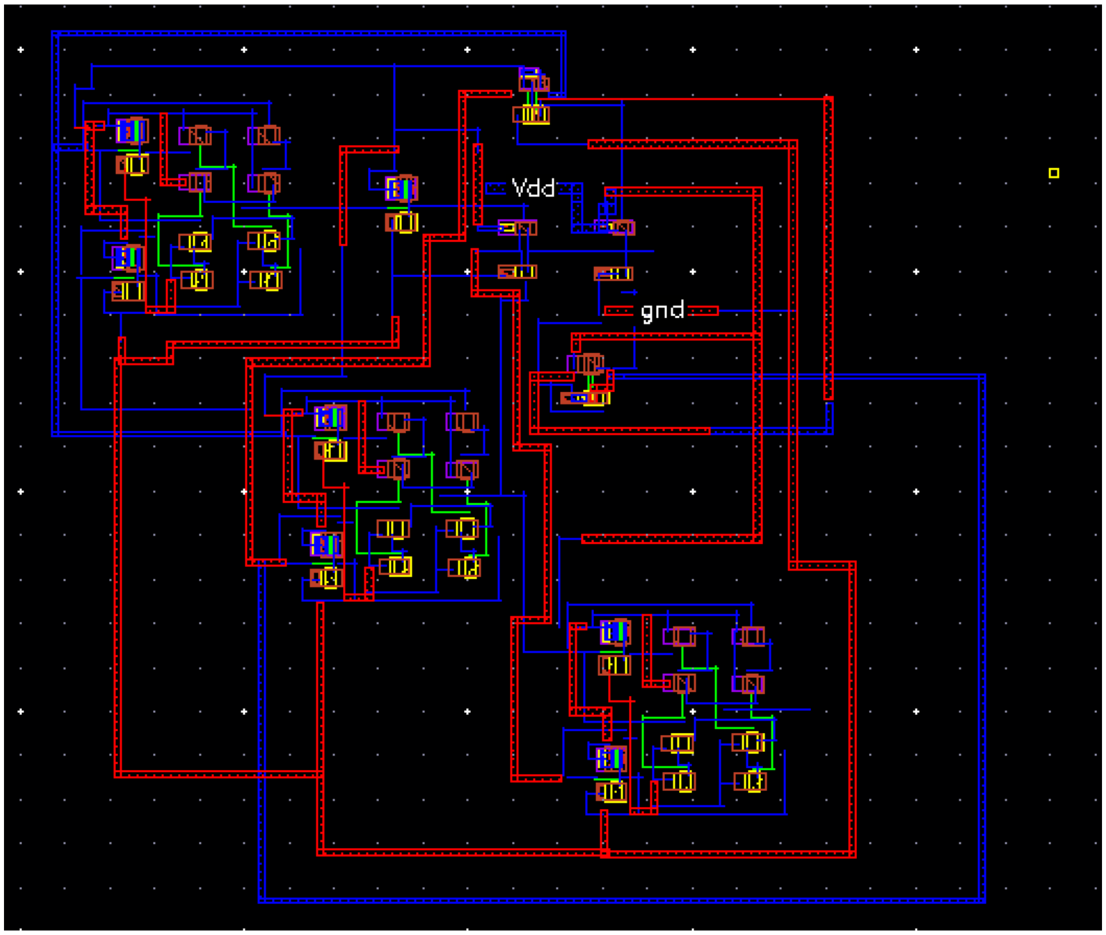
Inverter:


In
In
In
In
In
In
Vdd
Vdd
/B/Vdd
/S/Vdd
/D/Out
Out
Out
Gnd
G/B/Gnd
/B/Gnd
/S/Gnd
/D/Out

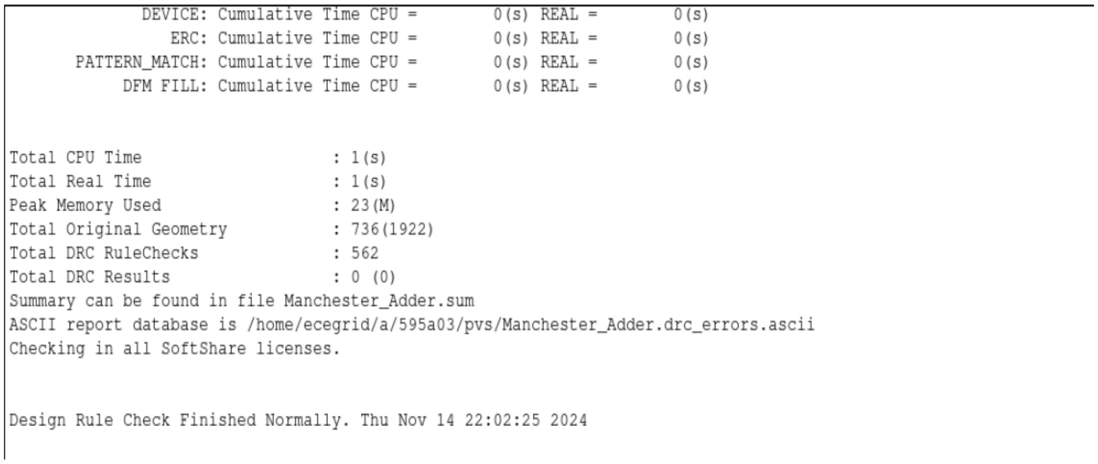
EXOR Gate:


VDD
/B/VDD
VDD
/S/VDD
/B/\/\D
/D/Vout
Vout
Vout
/A/
AA
B
/B/
B
Vout
A
/out
GND
GND
Vout
/B/G /B/GND
/B/GND
GND
GND

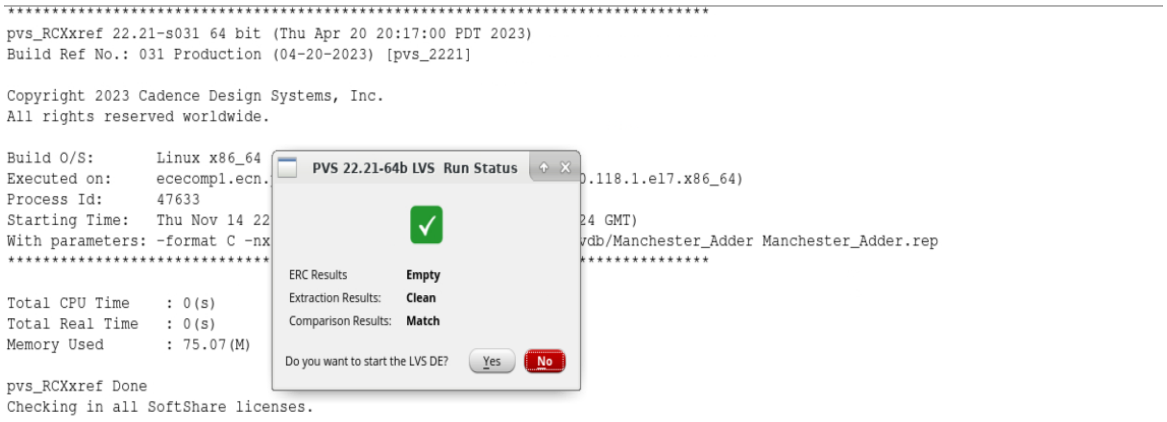
1-Bit MCC:


Vdd
gnd

DRC and LVS check:



```
              DEVICE: Cumulative Time CPU =        0(s)  REAL =        0(s)
                 ERC: Cumulative Time CPU =        0(s)  REAL =        0(s)
       PATTERN_MATCH: Cumulative Time CPU =        0(s)  REAL =        0(s)
            DFM FILL: Cumulative Time CPU =        0(s)  REAL =        0(s)


Total CPU Time                  : 1(s)
Total Real Time                 : 1(s)
Peak Memory Used                : 23(M)
Total Original Geometry         : 736(1922)
Total DRC RuleChecks            : 562
Total DRC Results               : 0 (0)
Summary can be found in file Manchester_Adder.sum
ASCII report database is /home/ecegrid/a/595a03/pvs/Manchester_Adder.drc_errors.ascii
Checking in all SoftShare licenses.


Design Rule Check Finished Normally. Thu Nov 14 22:02:25 2024
```



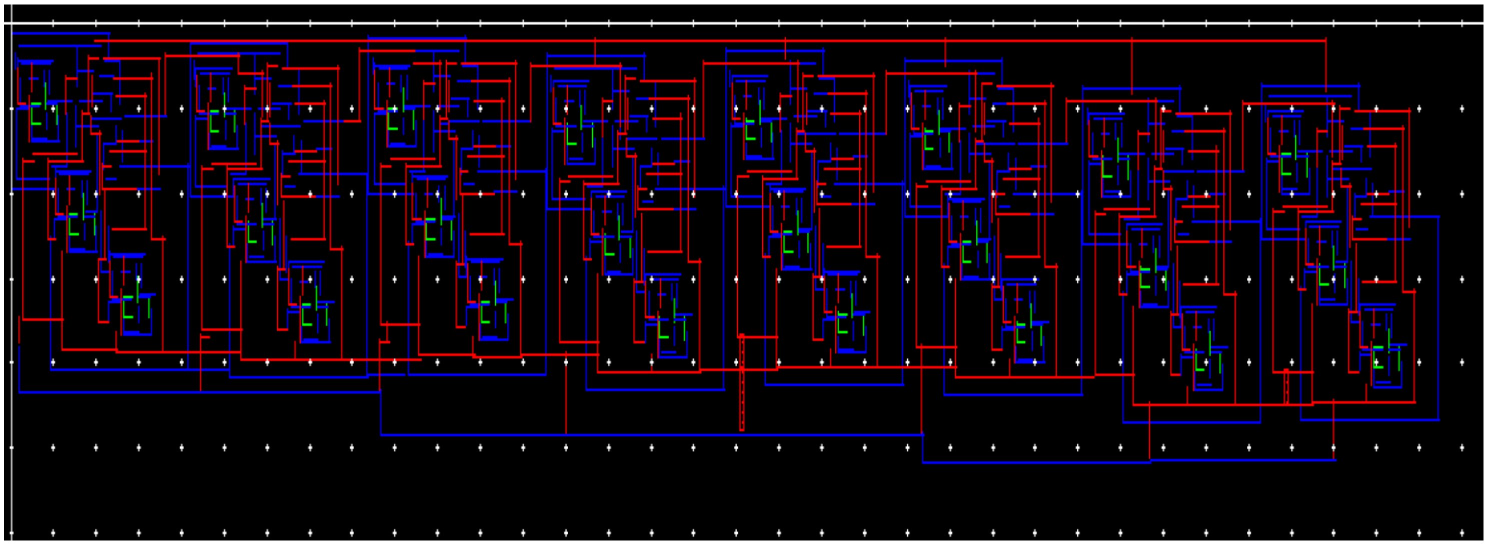
```
************************************************************************************
pvs_RCXxref 22.21-s031 64 bit (Thu Apr 20 20:17:00 PDT 2023)
Build Ref No.: 031 Production (04-20-2023) [pvs_2221]

Copyright 2023 Cadence Design Systems, Inc.
All rights reserved worldwide.

Build O/S:       Linux x86_64
Executed on:     ececomp1.ecn.                      0.118.1.el7.x86_64)
Process Id:      47633
Starting Time:   Thu Nov 14 22                      24 GMT)
With parameters: -format C -nx                      vdb/Manchester_Adder Manchester_Adder.rep
********************************                    ***************

Total CPU Time    : 0(s)
Total Real Time   : 0(s)
Memory Used       : 75.07(M)

pvs_RCXxref Done
Checking in all SoftShare licenses.
```

PVS 22.21-64b LVS Run Status

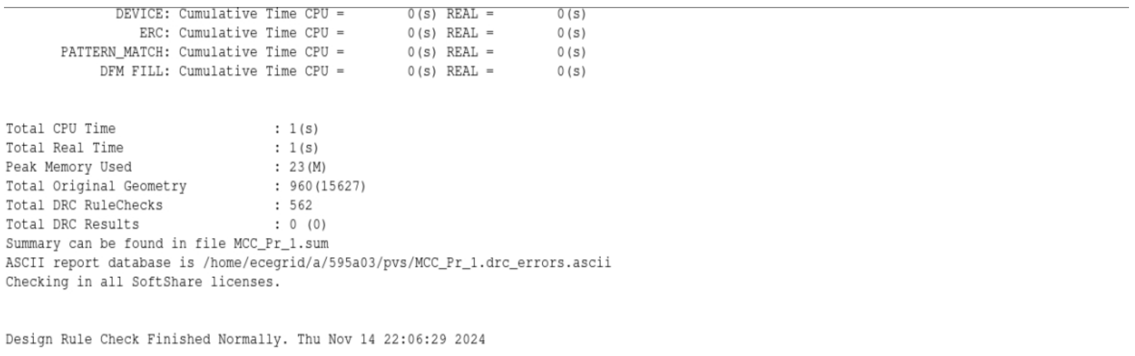
ERC Results	Empty
Extraction Results:	Clean
Comparison Results:	Match

Do you want to start the LVS DE? Yes No

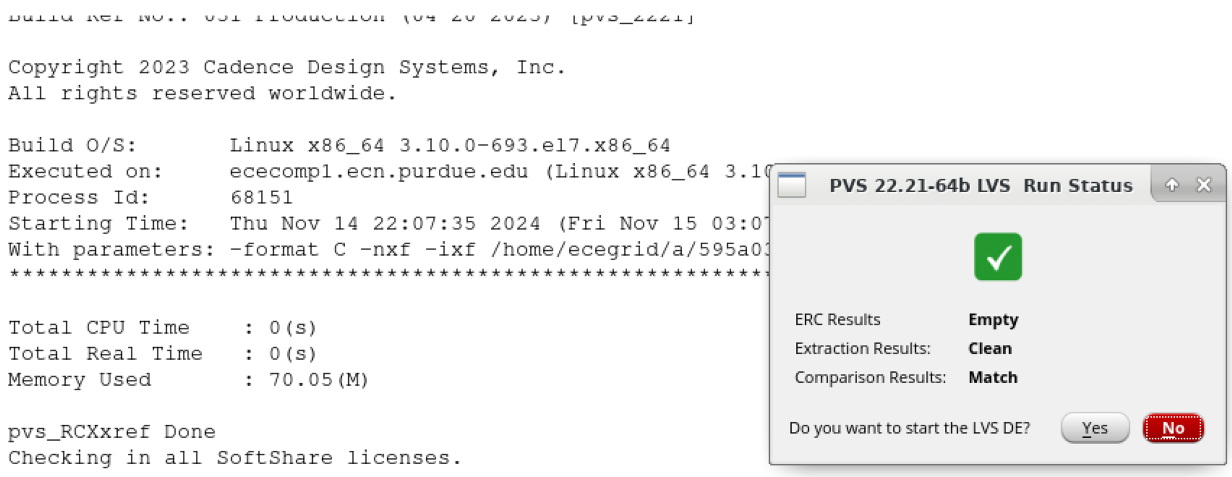
8-bit MCC:



DRC and LVS Check:



```
              DEVICE: Cumulative Time CPU =         0(s)  REAL =         0(s)
                 ERC: Cumulative Time CPU =         0(s)  REAL =         0(s)
       PATTERN_MATCH: Cumulative Time CPU =         0(s)  REAL =         0(s)
            DFM FILL: Cumulative Time CPU =         0(s)  REAL =         0(s)


Total CPU Time                    : 1(s)
Total Real Time                   : 1(s)
Peak Memory Used                  : 23(M)
Total Original Geometry           : 960(15627)
Total DRC RuleChecks              : 562
Total DRC Results                 : 0 (0)
Summary can be found in file MCC_Pr_1.sum
ASCII report database is /home/ecegrid/a/595a03/pvs/MCC_Pr_1.drc_errors.ascii
Checking in all SoftShare licenses.


Design Rule Check Finished Normally. Thu Nov 14 22:06:29 2024
```



```
Build Ref No.: 051 Production (04-20-2023) [pvs_2221]

Copyright 2023 Cadence Design Systems, Inc.
All rights reserved worldwide.

Build O/S:        Linux x86_64 3.10.0-693.el7.x86_64
Executed on:      ececomp1.ecn.purdue.edu (Linux x86_64 3.10
Process Id:       68151
Starting Time:    Thu Nov 14 22:07:35 2024 (Fri Nov 15 03:07
With parameters: -format C -nxf -ixf /home/ecegrid/a/595a03
**********************************************************

Total CPU Time     : 0(s)
Total Real Time    : 0(s)
Memory Used        : 70.05(M)

pvs_RCXxref Done
Checking in all SoftShare licenses.
```

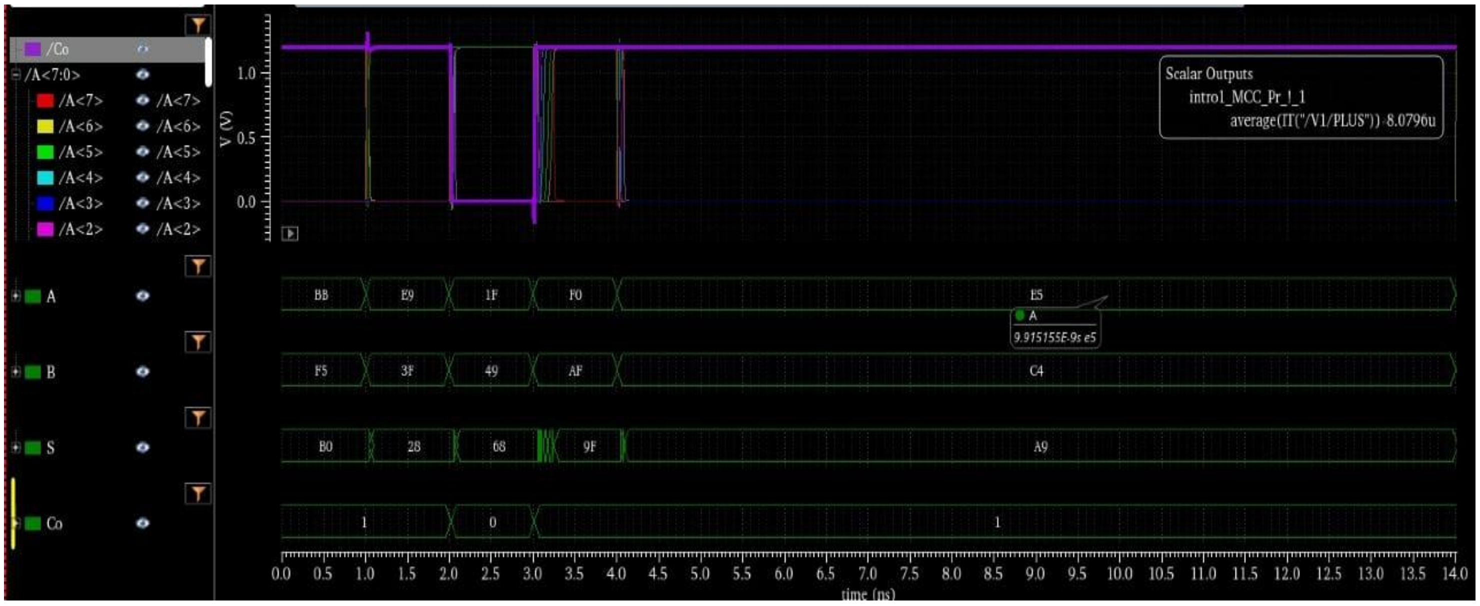
PVS 22.21-64b LVS Run Status

✓

ERC Results	**Empty**
Extraction Results:	**Clean**
Comparison Results:	**Match**

Do you want to start the LVS DE? Yes No

Power Simulation



```
Scalar Outputs
    intro1_MCC_Pr_1_1
        average(IT("/V1/PLUS")) -8.0796u
```

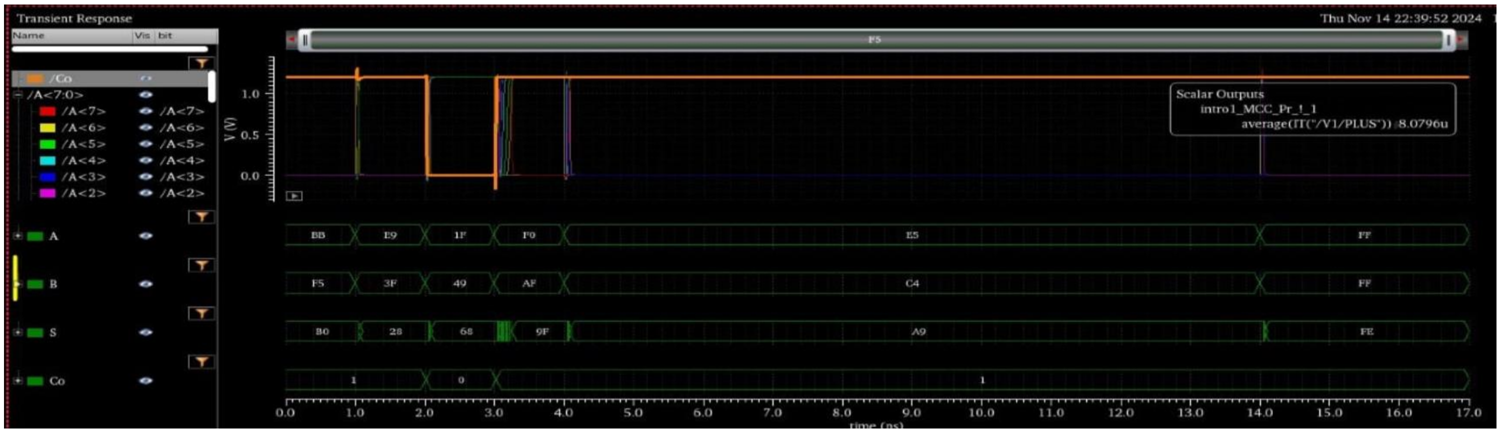
Adders Comparision:

Types	**Ladner Fischner**	**Brent Kung**	**Han-Carlson**	**Kogge-stone**	**Manchester Adder (CMOS)**
Logic depth	**4**	**5**	**5**	**4**	**5**
Area	**High**	**Low**	**Moderate**	**High**	**High**
Power	**Moderate**	**Moderate**	**Moderate**	**High**	**High**
Layout	**Moderate**	**Low**	**Moderate**	**High**	**Moderate**
Fan Out	**Moderate**	**High**	**Moderate**	**High**	**High**

Concluision(comparison)

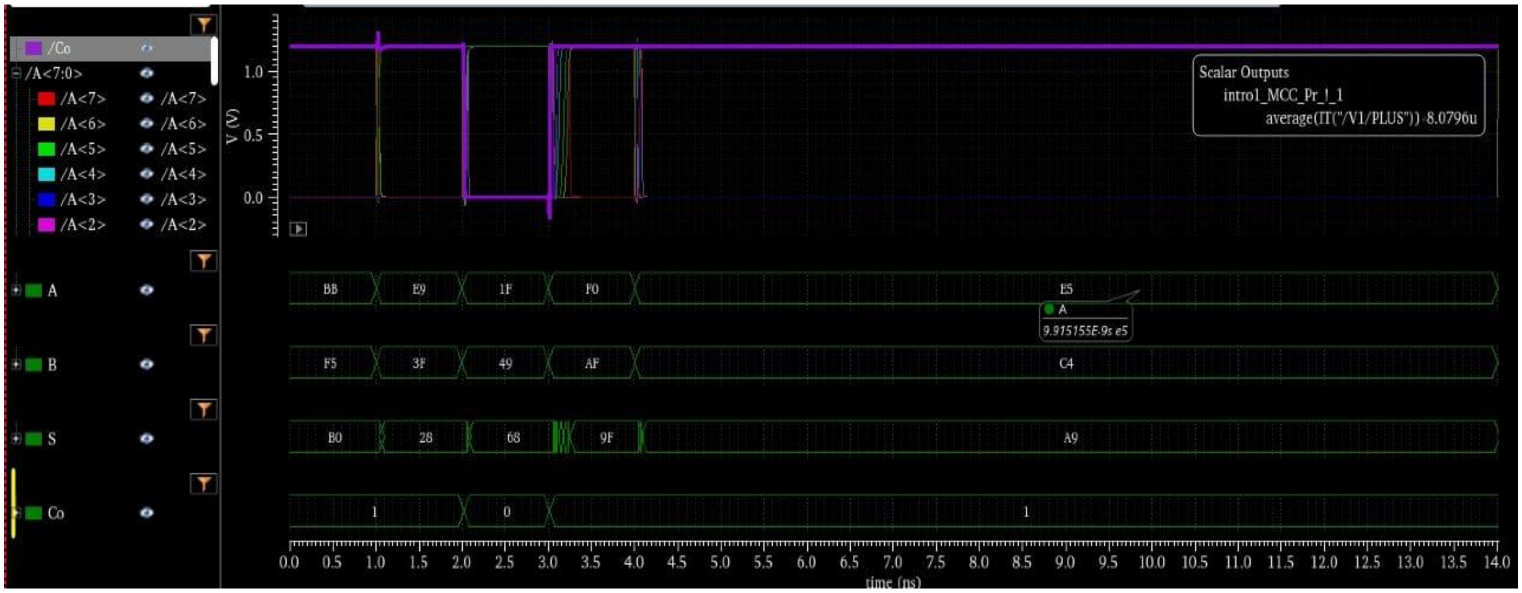
The power dissipation:

Pre layout =26.382uW



Scalar Outputs
intro1_MCC_Pr_!_1
average(IT("/V1/PLUS")) :8.0796u

Post layout = 8.079uW



Scalar Outputs
intro1_MCC_Pr_!_1
average(IT("/V1/PLUS")) 8.0796u

The propagation delay:
Pre layout = 90. 5 psec
Post Layout= 211.3 psec

For the Manchester Adder using the CMOS logic the Propagation delay is less compared to the other logic and the power dissipation is high, this is because of the number of transistors used in this logic (400 transistors)
The circuit complexity for the Manchester adder (CMOS logic) is high but the output is efficient. The CMOS logic gives the accurate output compared to other logic