VLSI DESIGN LAB
GROUP PROJECT

Group 6:

PARVATHAREDDY VENKATA SRINIVAS -2023H1230169H
MEKA MARUTI NAGA PHANINDHRA-2023H1230176H
MADDALI SOHIT VENKATA SAI-2023H1230178H
KEESARI HEMANTH KUMAR-2023H1230190H

AIM:

1.Design and implement a 16-bit Square Root Carry Select adder using Cadence Virtuoso. Report the average power dissipation, propagation delay, and worst-case delay. Theoretically, demonstrate and explain the computation of a four input 4-bit Carry Save adder and mention the application and advantage of using Carry Save adder

Tools Used:

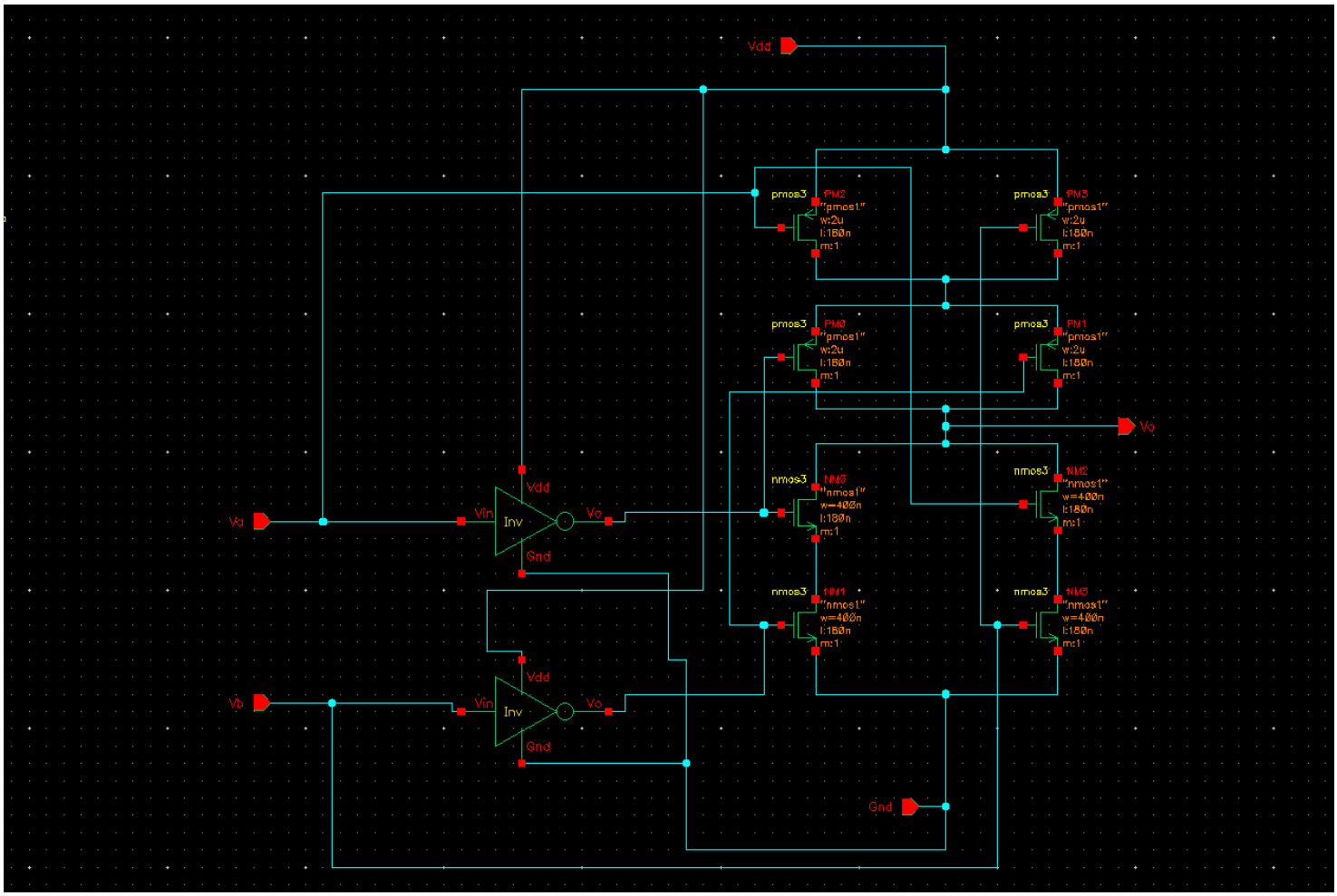
Cadence Virtuoso Analog Design Environment

Libraries Used:

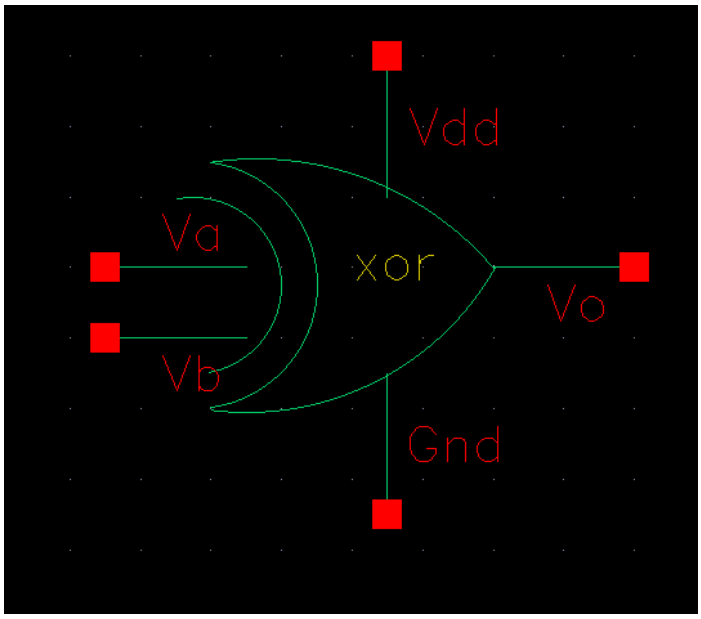
gpdk180 library, analoglib library.

Symbols Used:
Inverter
XOR
AND
OR
1-2 MUX
Full Adder
2-Bit Adder with Carry-0
2-Bit Adder with Carry-1
2-Bit Adder with Cin
3-Bit Adder with Carry-0
3-Bit Adder with Carry-1
4-Bit Adder with Carry-0
4-Bit Adder with Carry-1
5-Bit Adder with Carry-0
5-Bit Adder with Carry-1

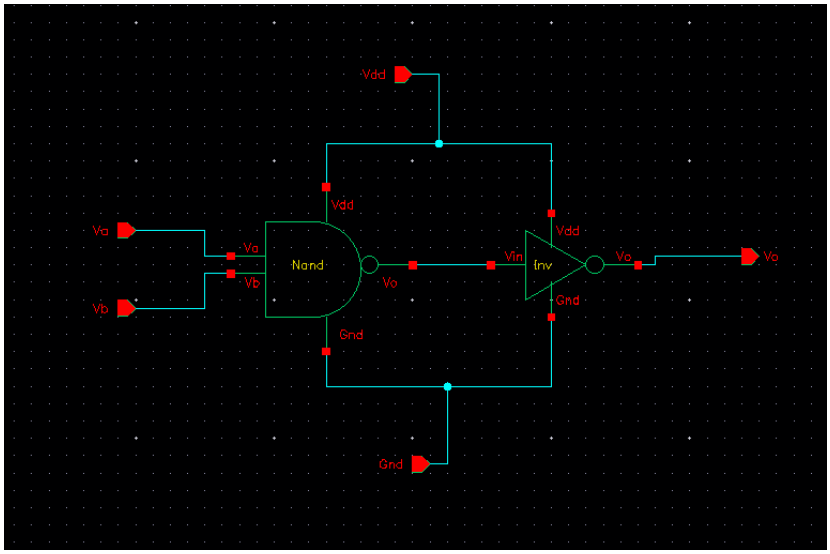
XOR Schematic:

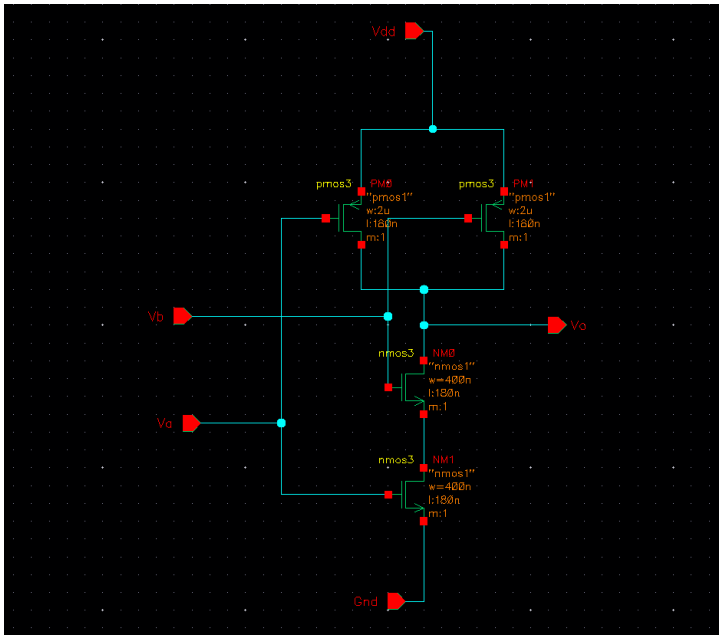


XOR Symbol:

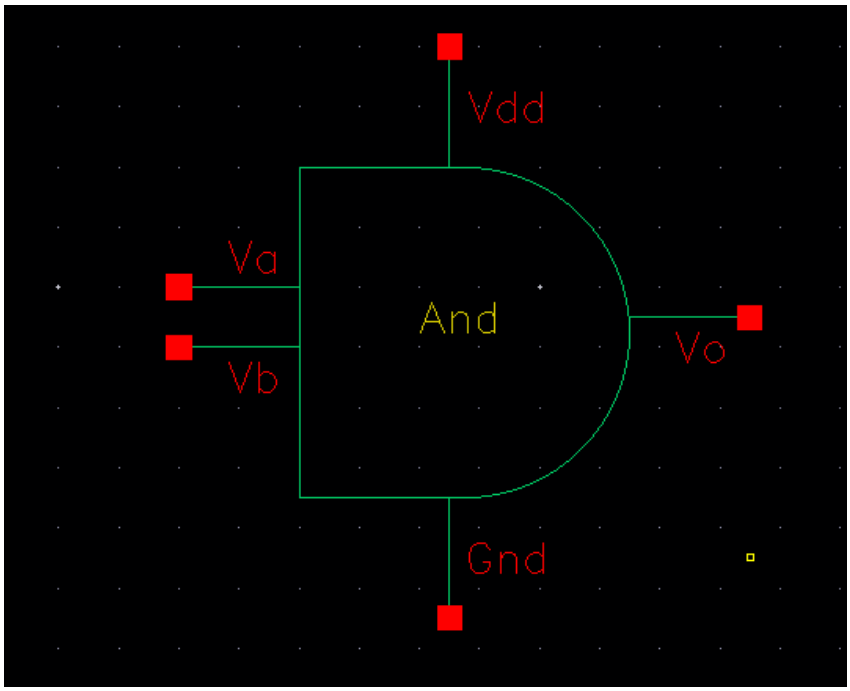


AND and NAND Schematic:

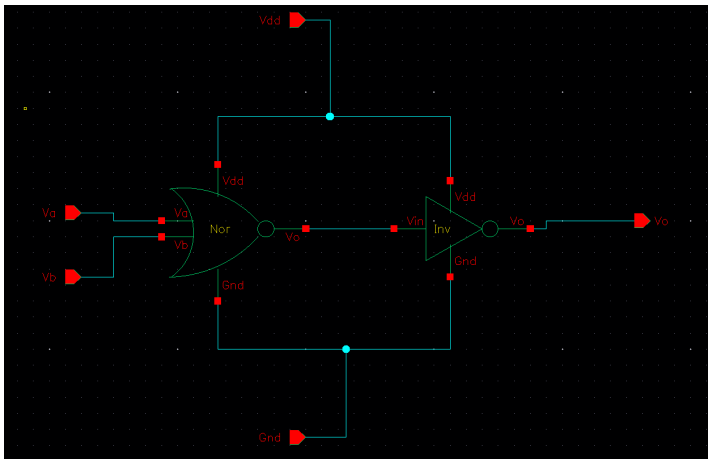


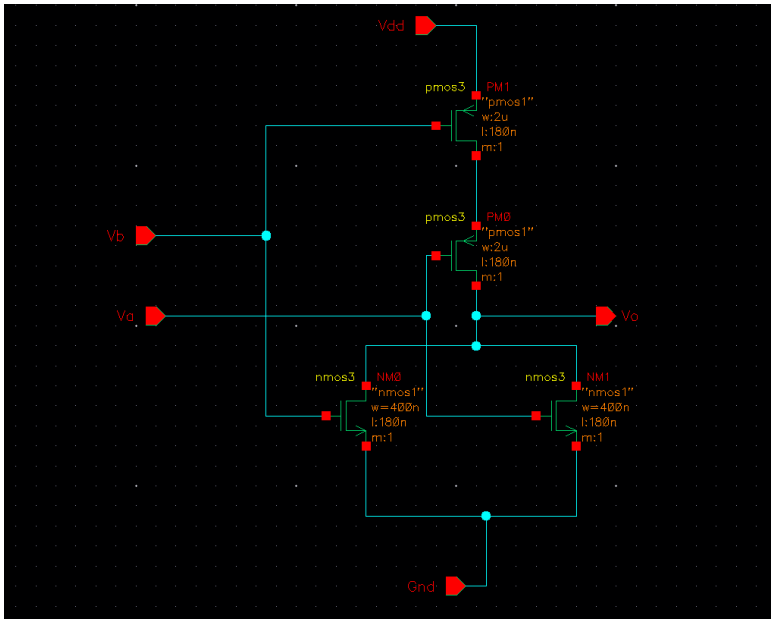


AND Symbol:

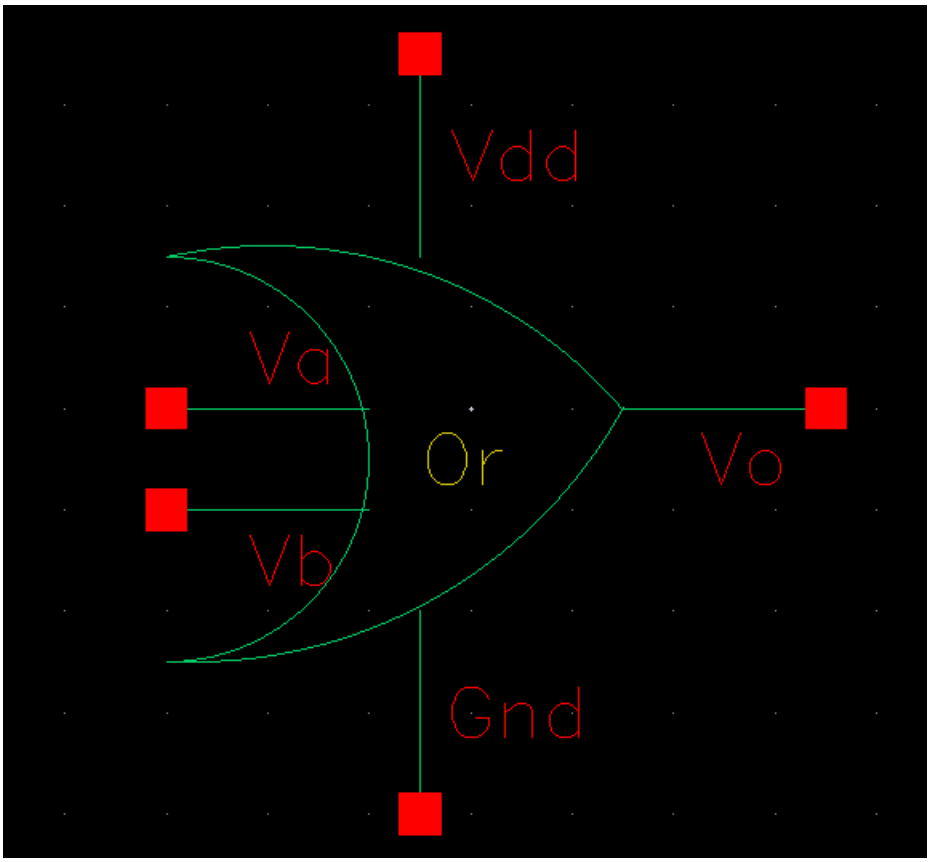


OR and NOR Schematic:

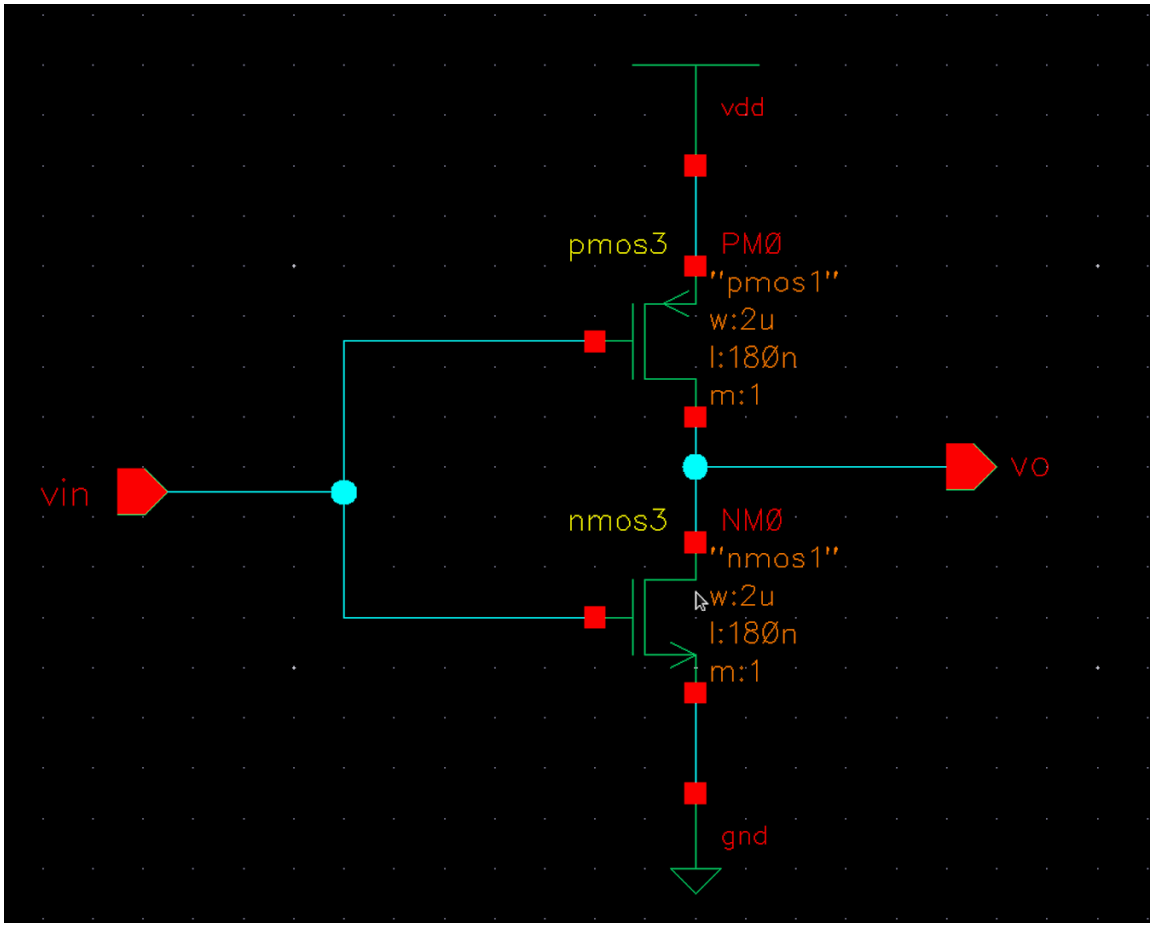




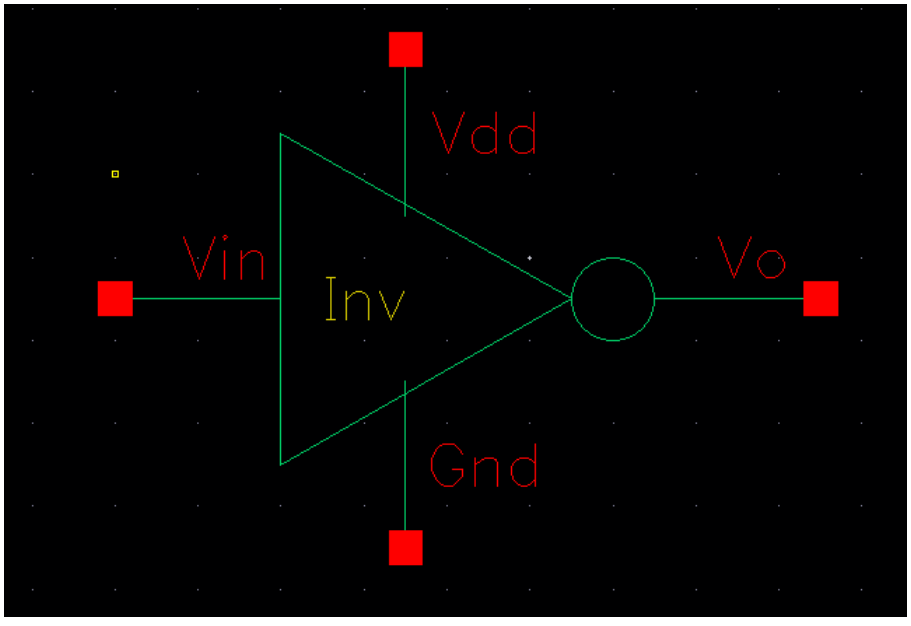
OR Symbol:



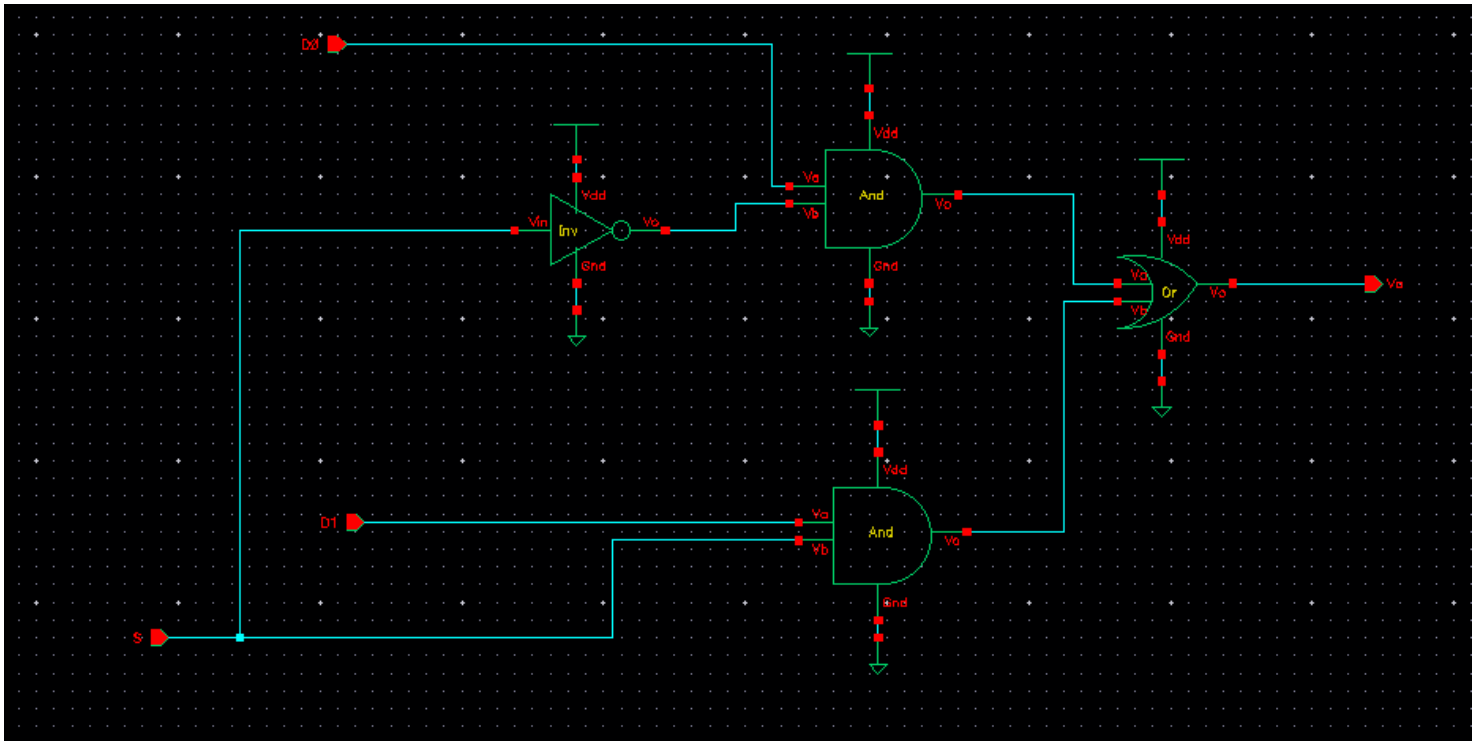
INVERTER:



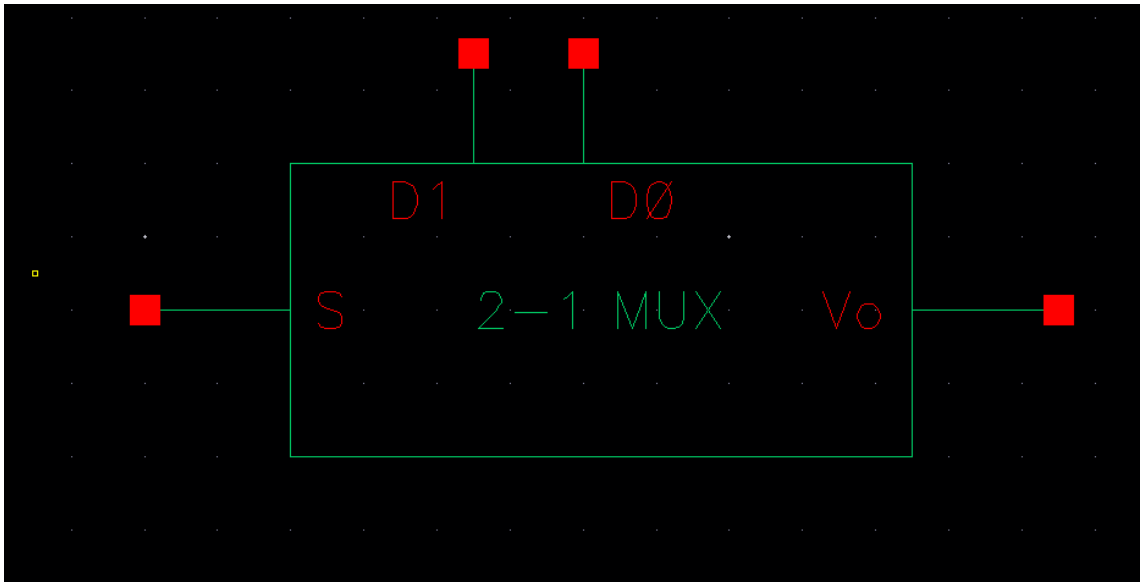
INVERTER Symbol:



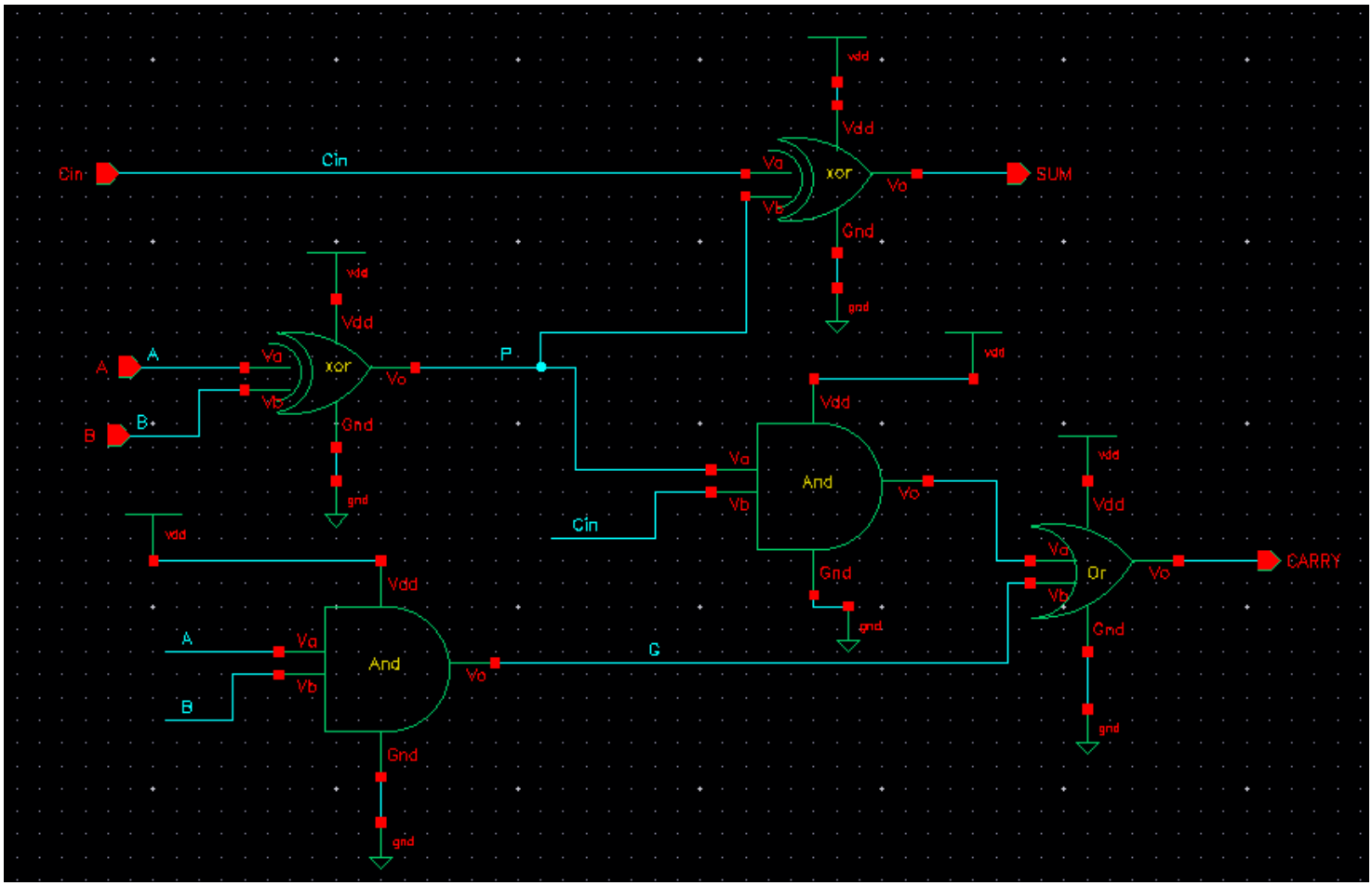
MUX 2-1 Schematic:



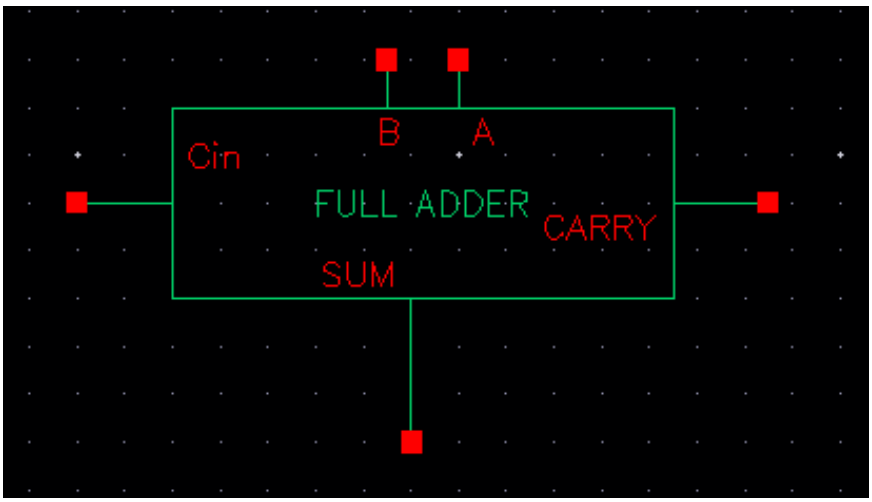
MUX 2-1 Schematic:



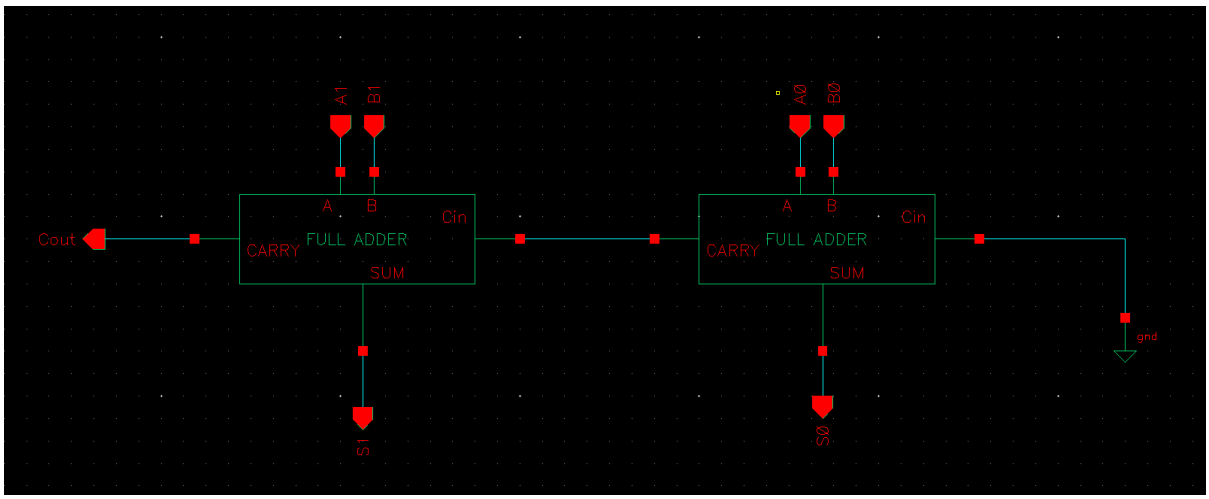
Full Adder circuit:



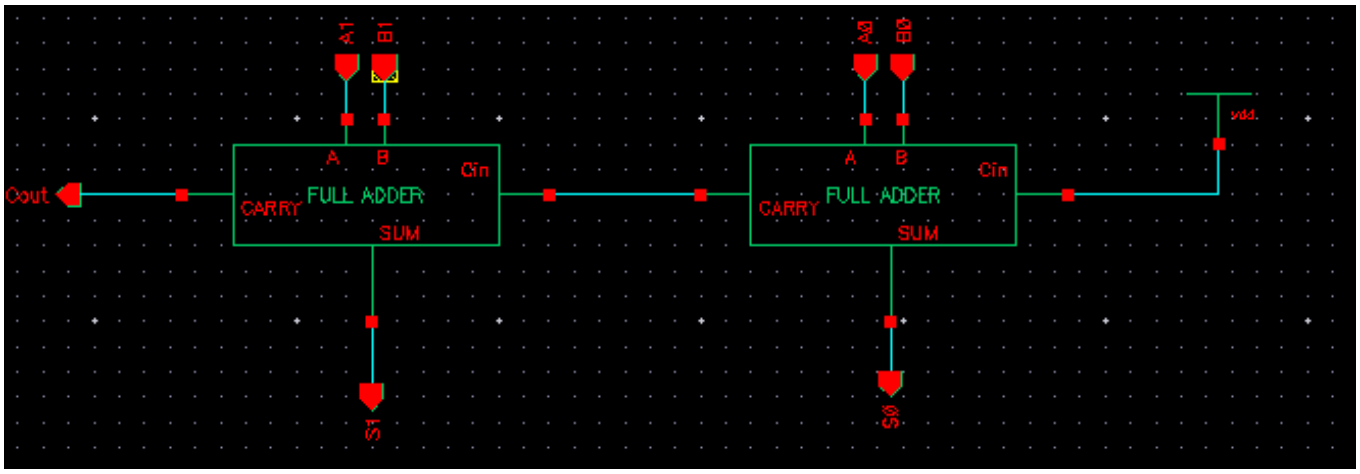
Full Adder Symbol:



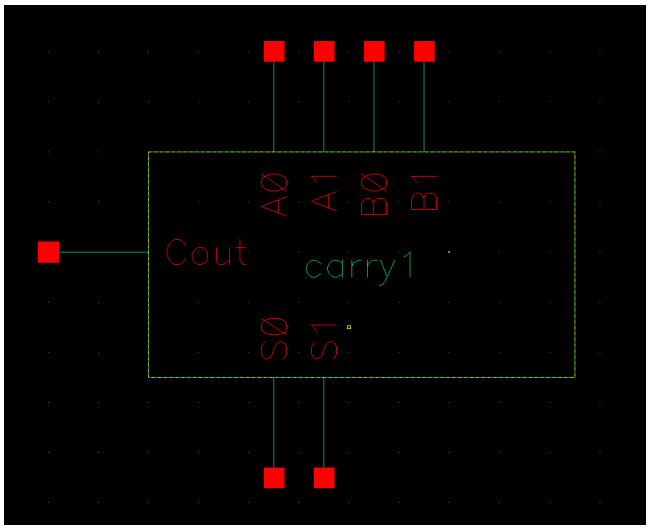
Full adder 2-bit Schematic for carry-0:

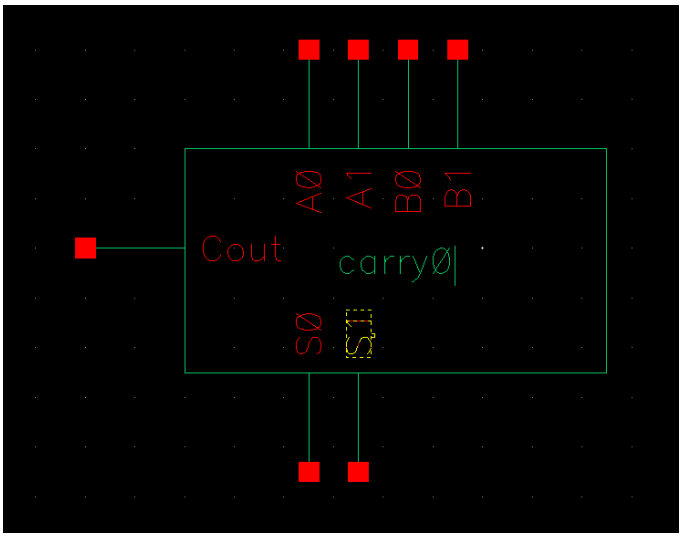


Full adder 2-bit Schematic for carry-1:

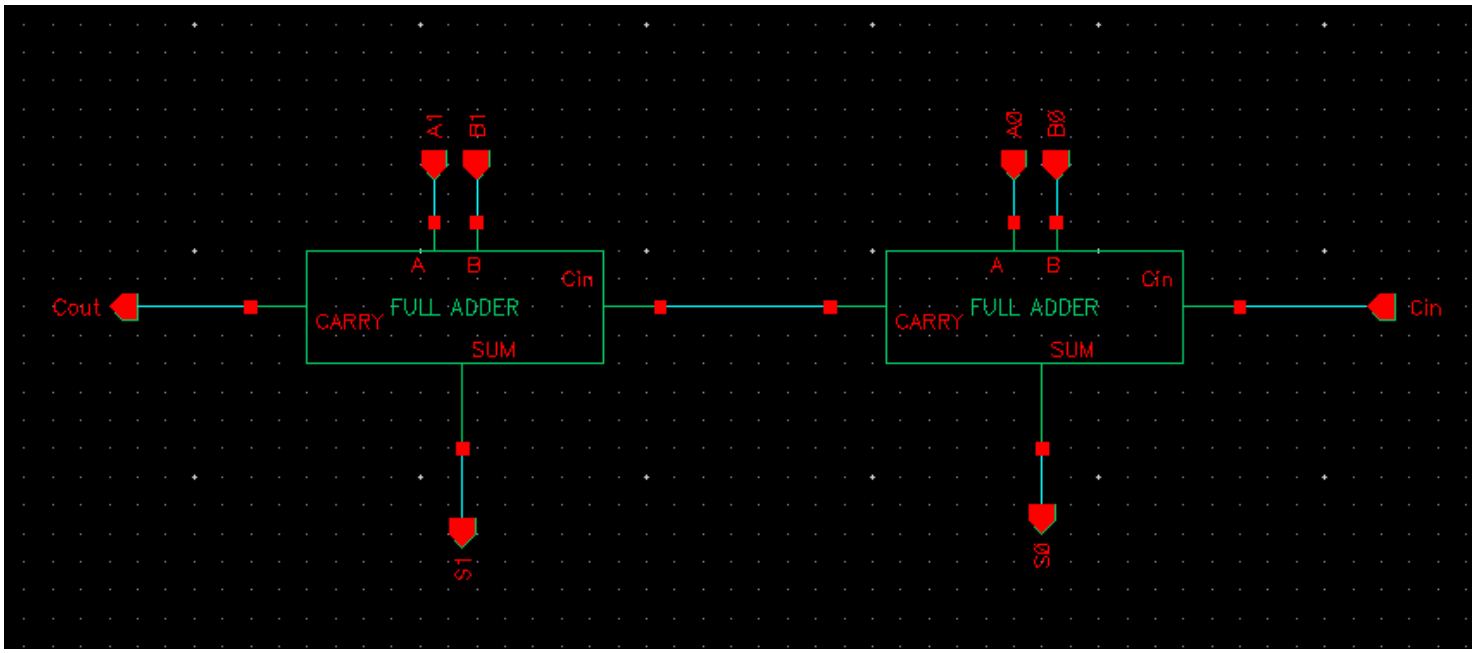


Symbol for 2bit Carry-0 and Carry-1:

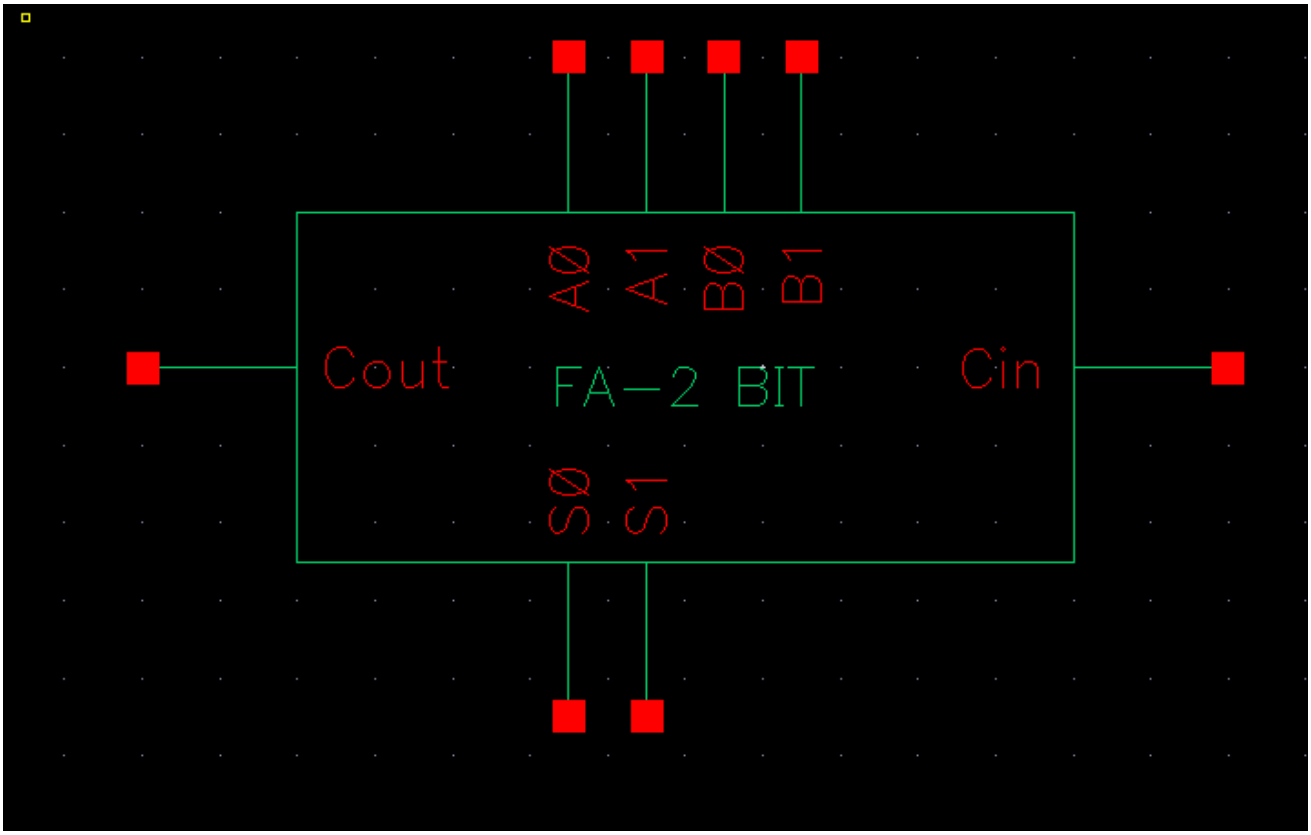




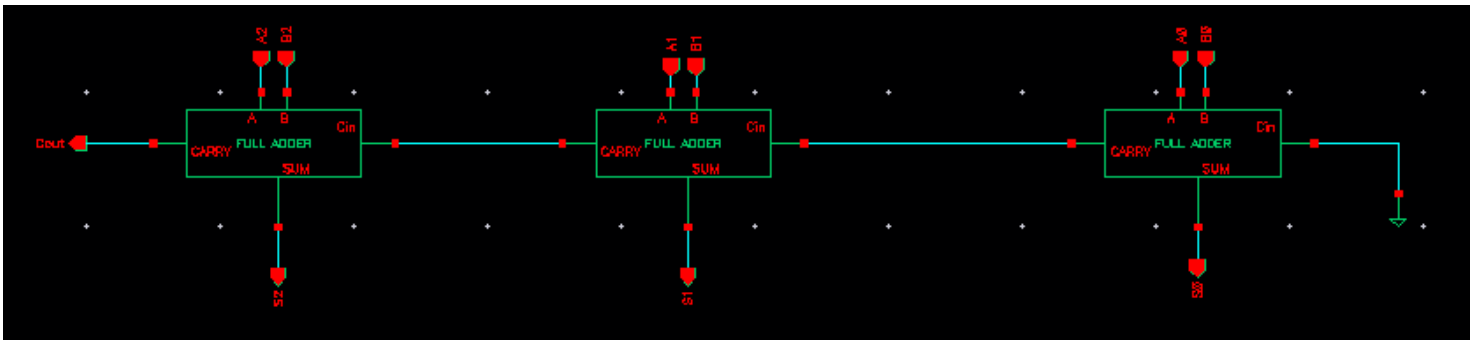
Full adder 2-bit Schematic:



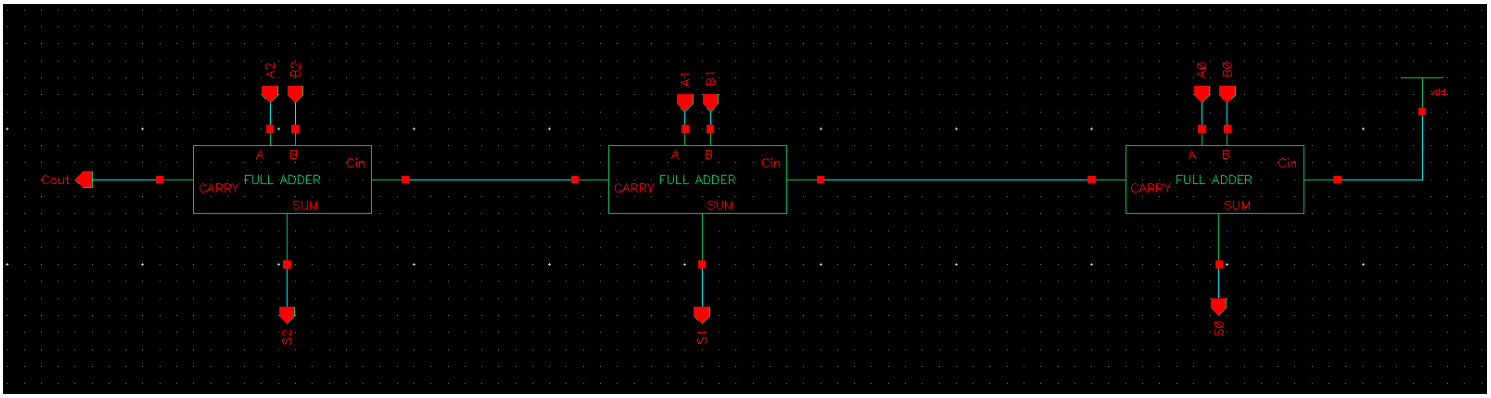
Full adder 2-bit Symbol:



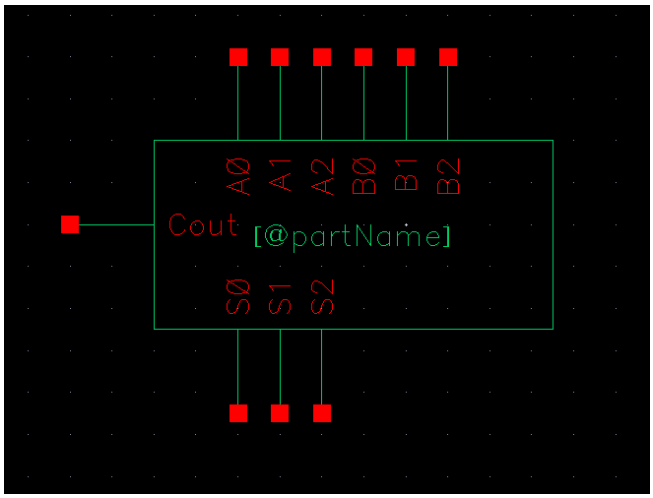
Full Adder 3-Bit Schematic for carry-0:

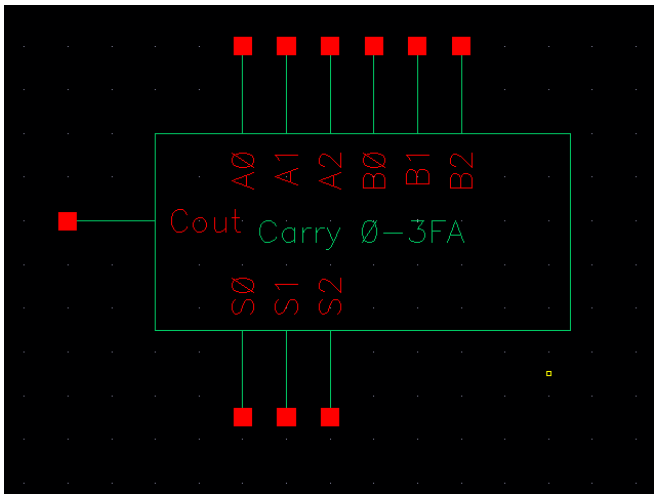


Full Adder 3-Bit Schematic for carry-1:

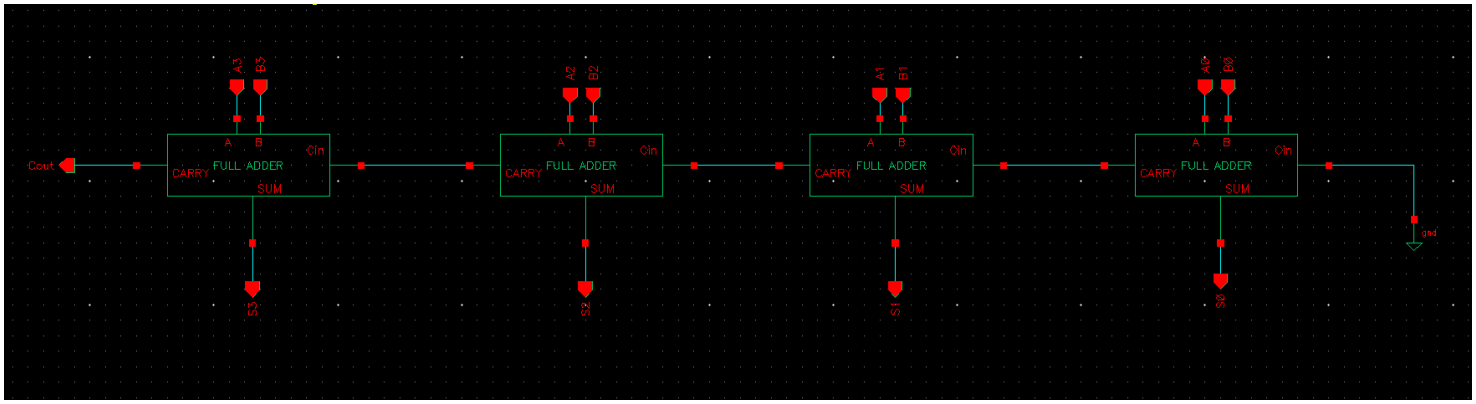


Full Adder 3-Bit Symbol for Carry-1 and Carry-0:

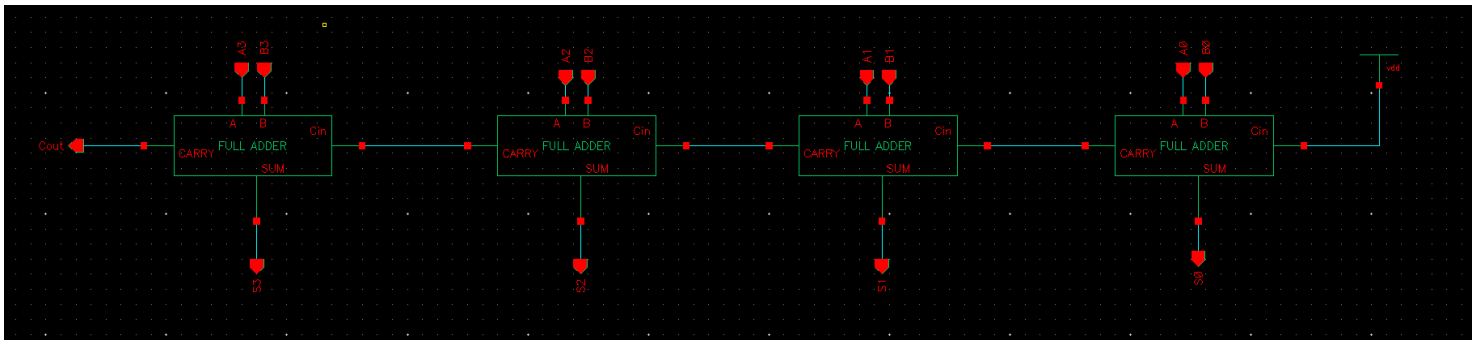




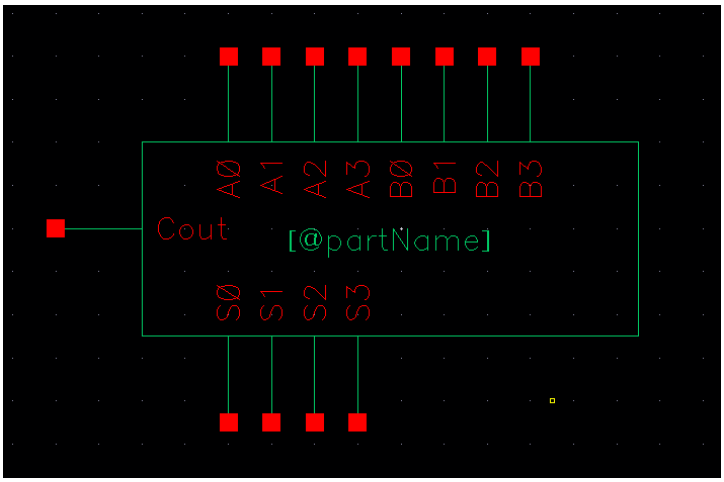
Full Adder 4 Bit Schematic for carry-0:

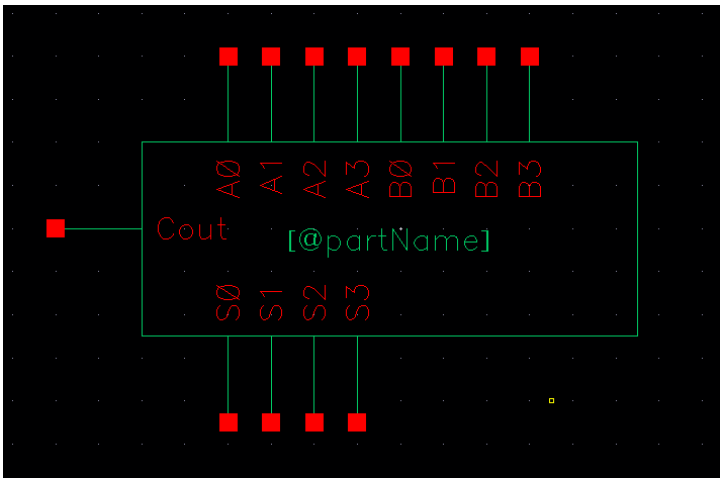


Full Adder 4 Bit Schematic for carry-1:

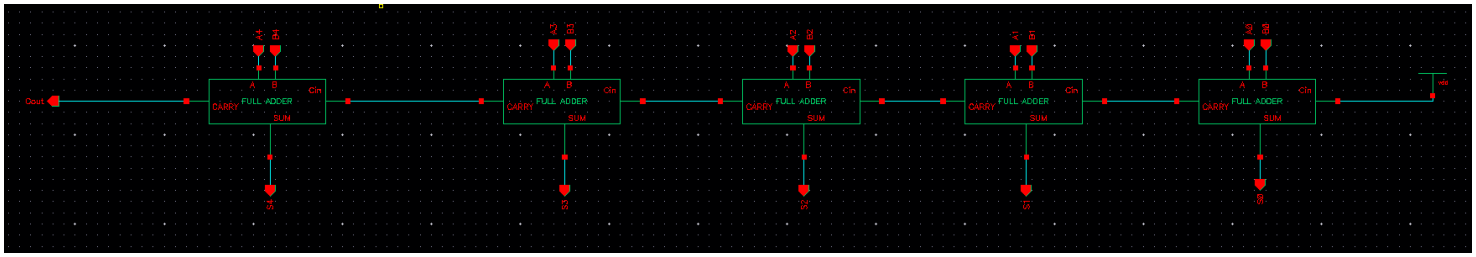


Full Adder 4-Bit Symbols for Carry-1 and Carry-0:

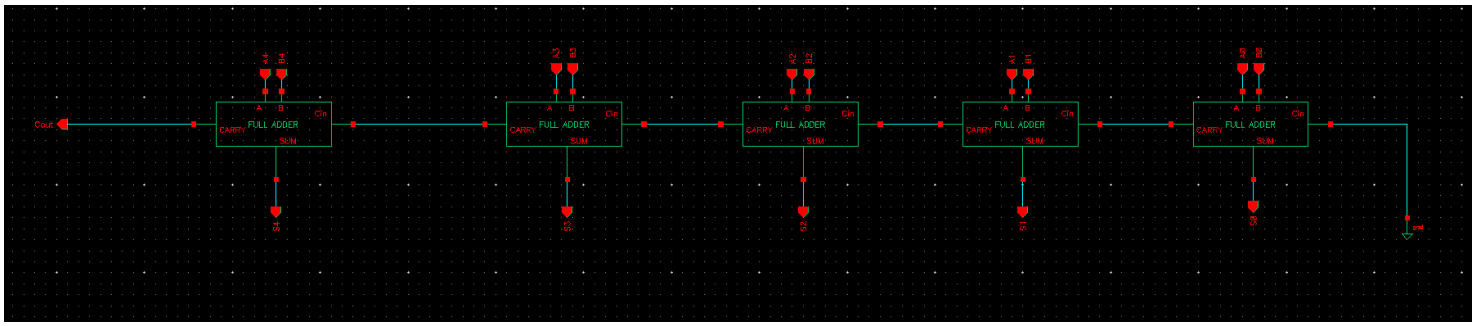




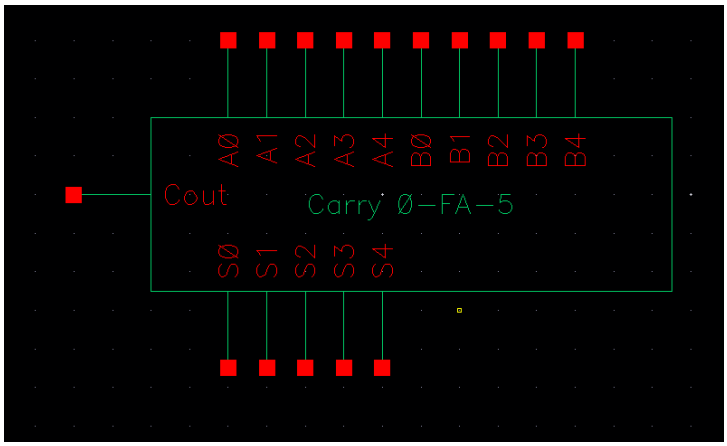
Full Adder 5-Bit Schematic for Carry-1:

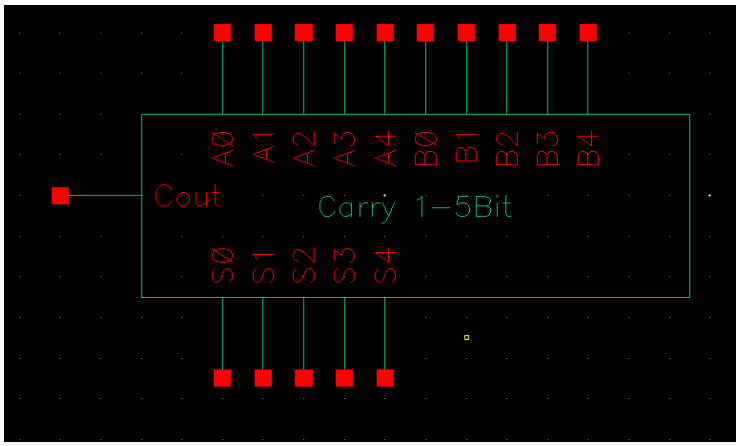


Full Adder 5-Bit Schematic for Carry-0:

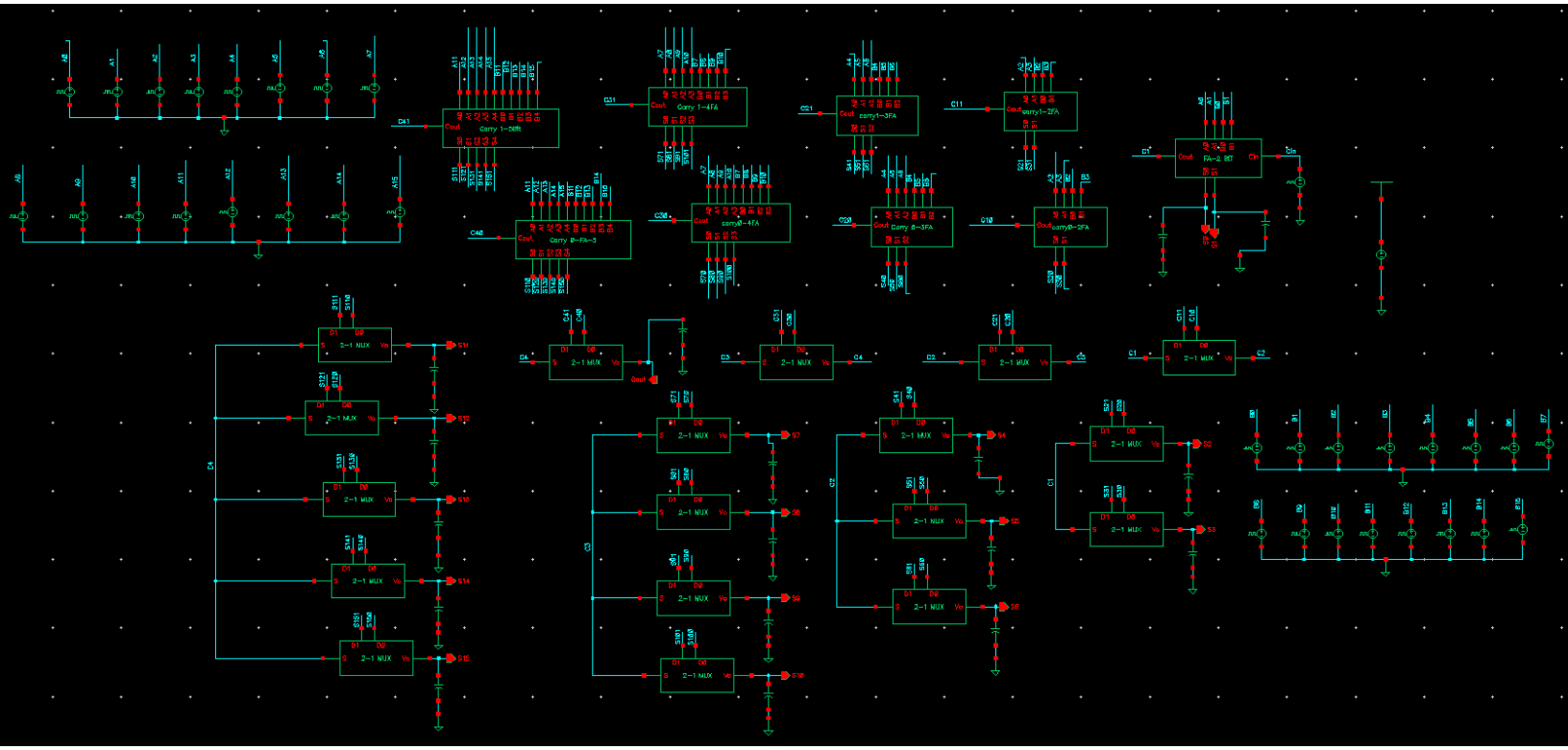


Full Adder 5-Bit Symbol for Carry-0 and Carry-1:





16-bit Square Root Carry Select Adder Circuit:



Worst Case Delay:

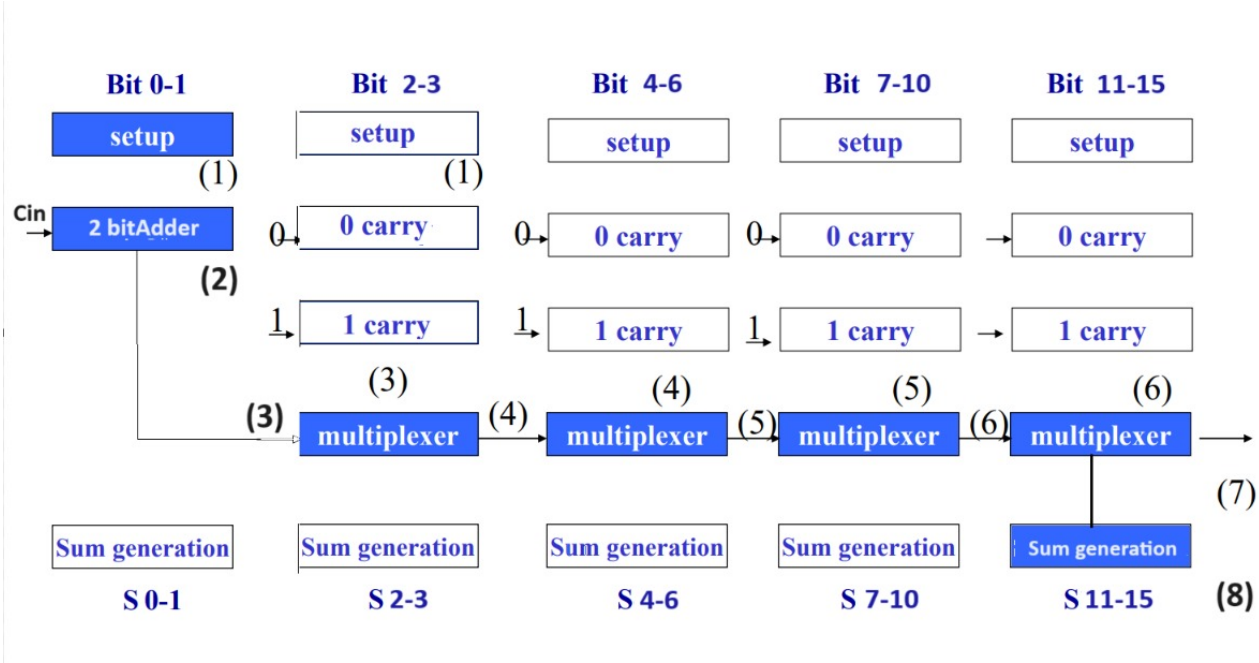
We will get worst case delay for the below input combination

Input	Value	Input	Value
A0	1	B0	0
A1	1	B1	0
A2	1	B2	0
A3	1	B3	0
A4	1	B4	0
A5	1	B5	0
A6	1	B6	0
A7	1	B7	0
A8	1	B8	0
A9	1	B9	0
A10	1	B10	0
A11	1	B11	0
A12	1	B12	0
A13	1	B13	0
A14	1	B14	0
A15	1	B15	0

Calculating TPLH and TPHL:

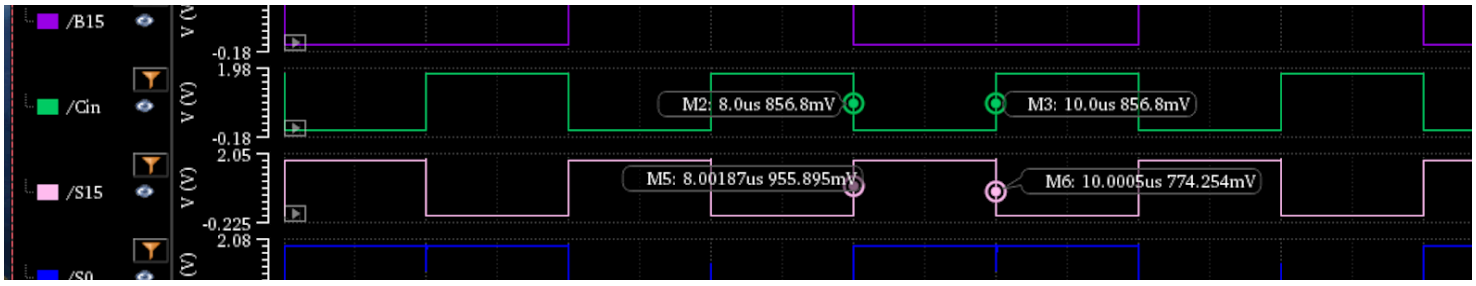
Worst case delay is from the time we have given the input to the time sum 15 is available and the critical path is as shown below

Critical path:



Theoretically we can calculate by

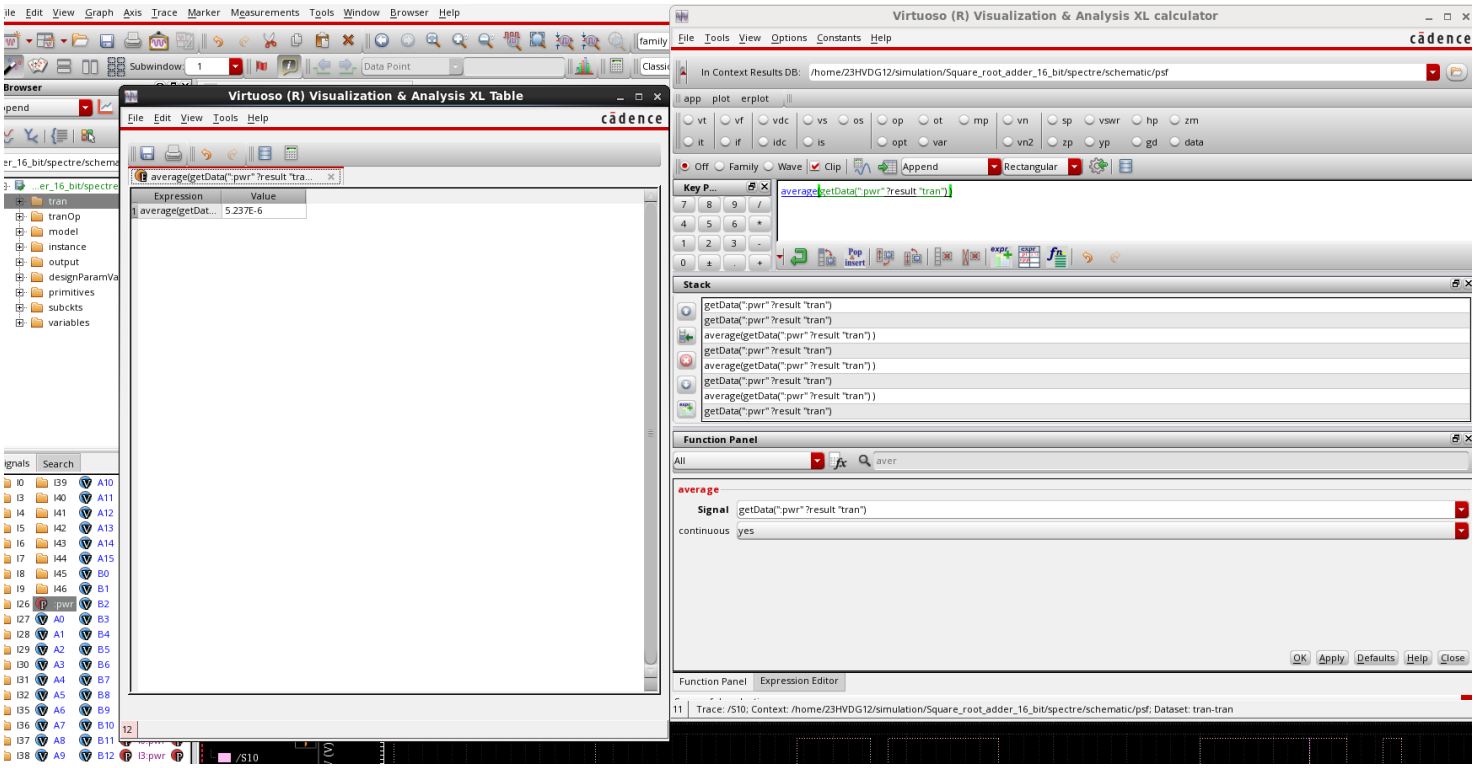
Tpropagation delay= Tsetup (1)+ 2Tcarry(2,3) + 3*MUX for Carry(4,5,6) + MUX for sum(7)



TPHL = 1.87ns
TPLH = 0.5ns

Propagation Delay: Tpd = (tPHL + tPLH)/2 = **1.185ns**
Worst case delay = Max(tPHL,tPLH) =**1.87ns**

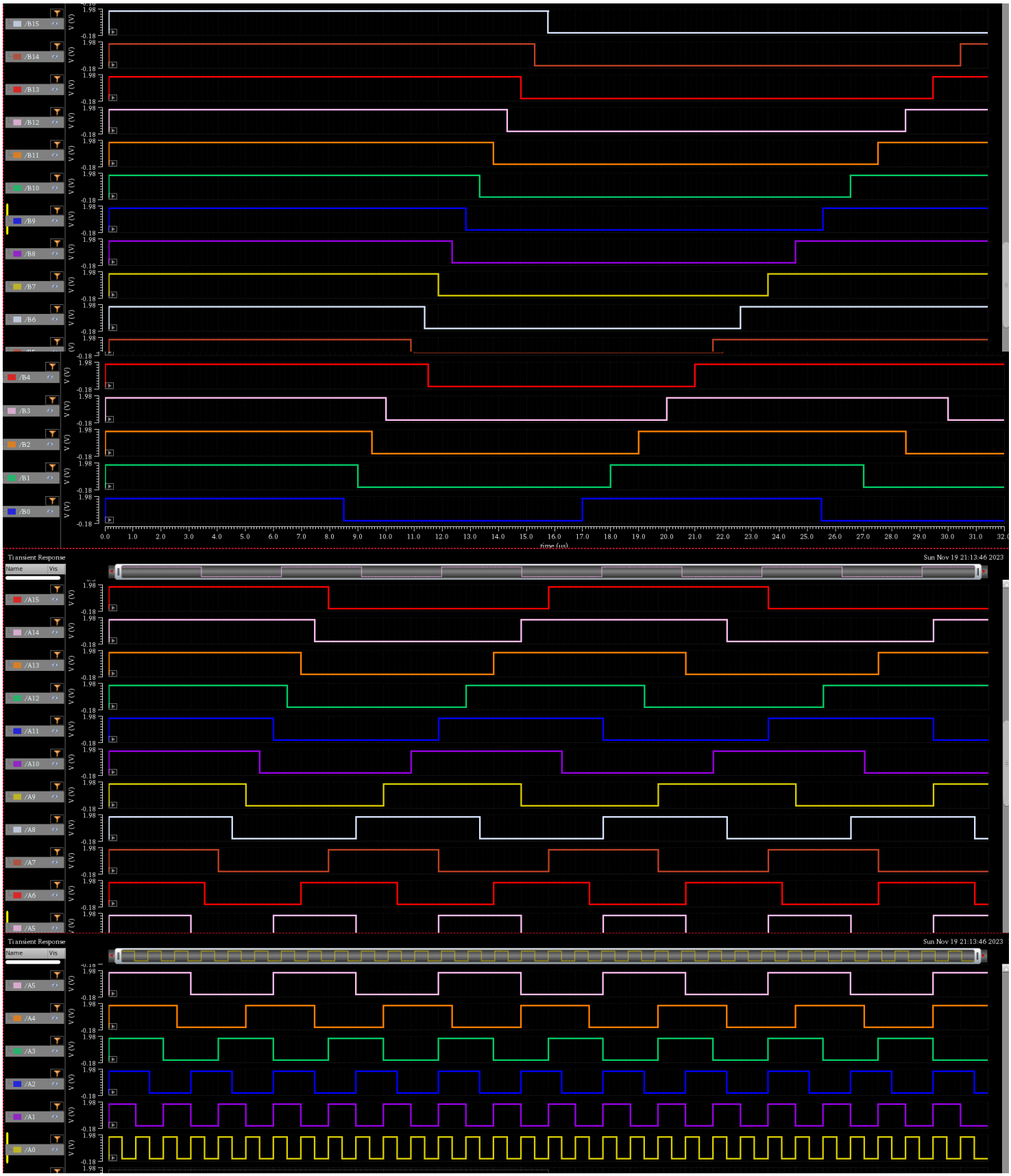
Average Power Dissipation:

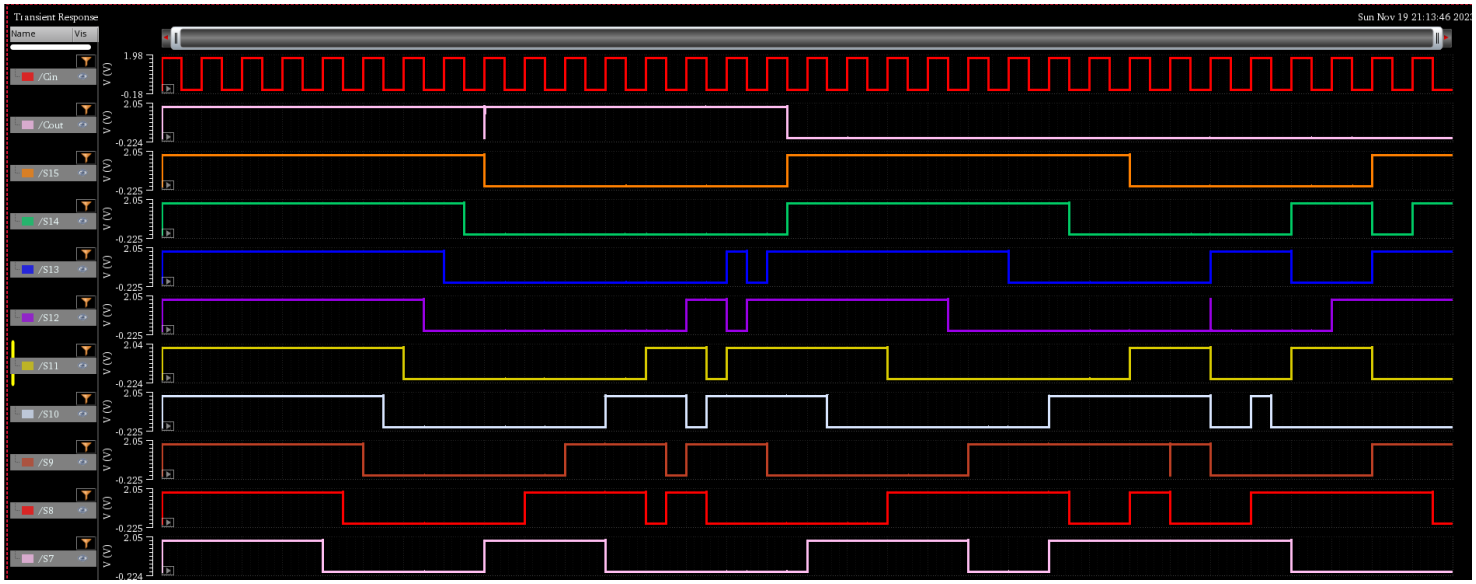


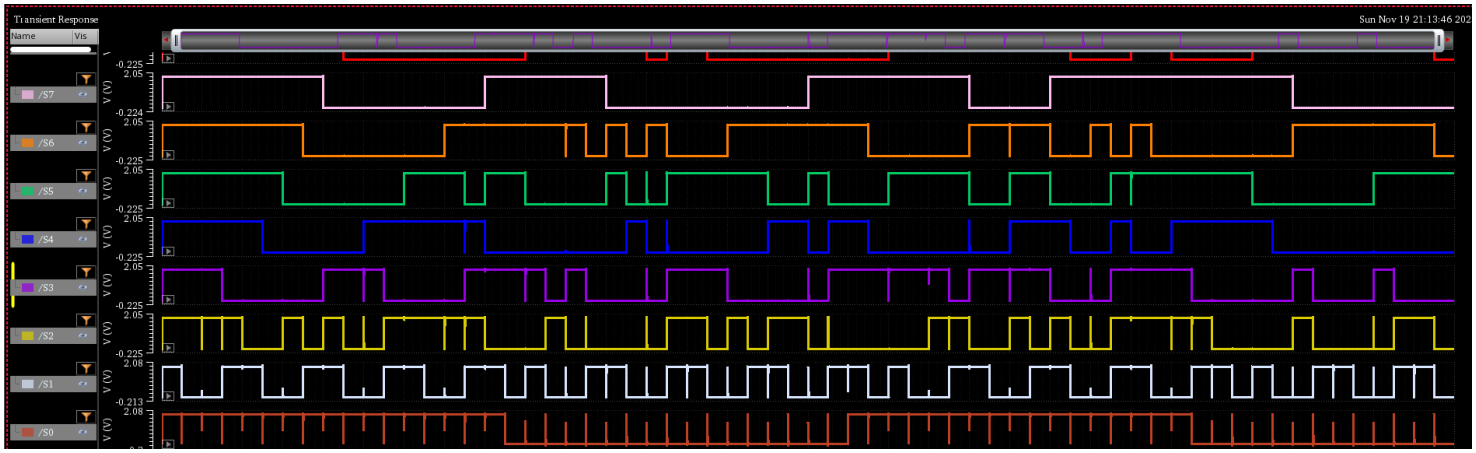
Average power dissipation is 5.237 uW

Output Waveforms:

The Below waveform depicts the input combinations for A0-A15 and B0-B15

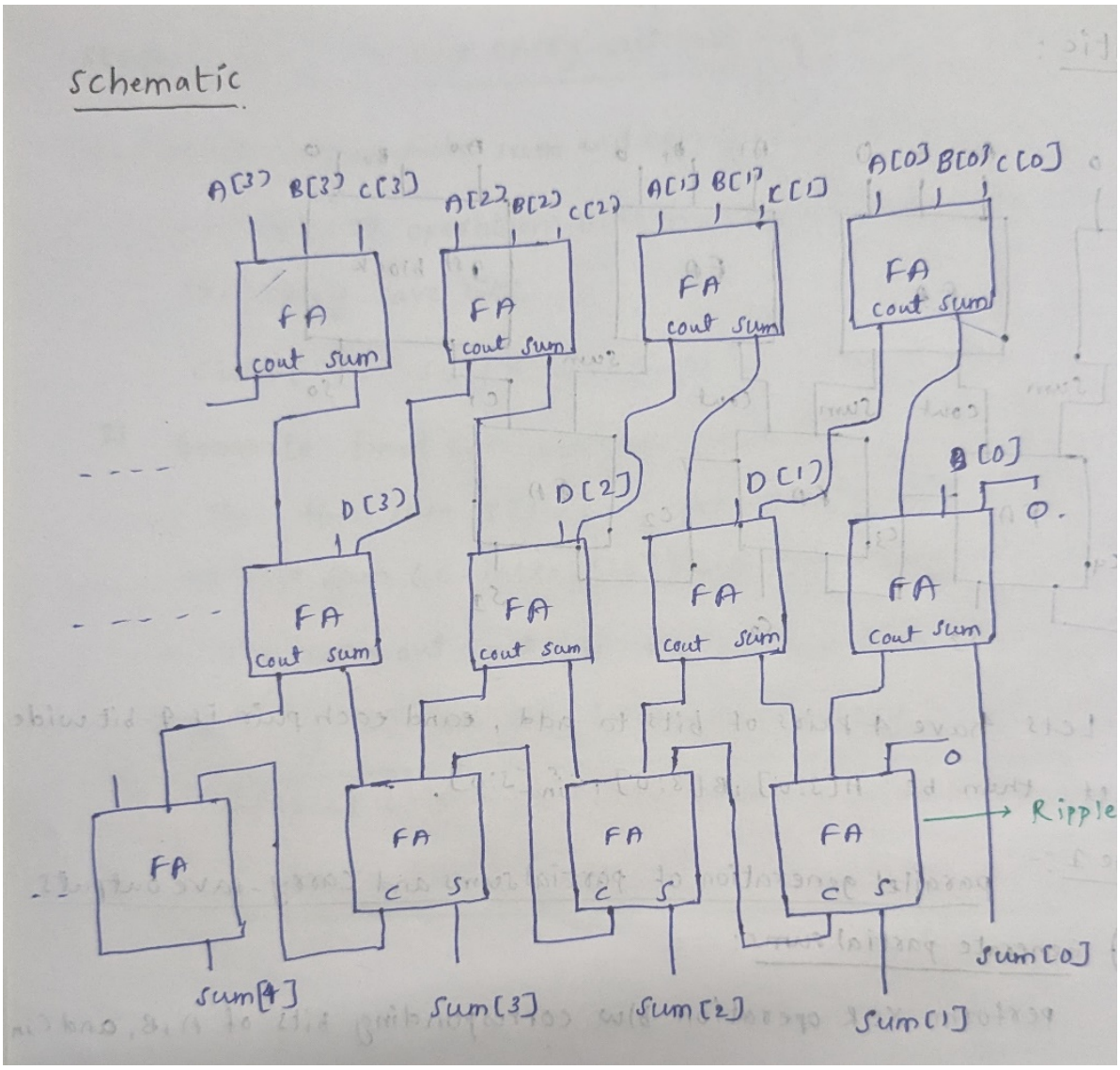




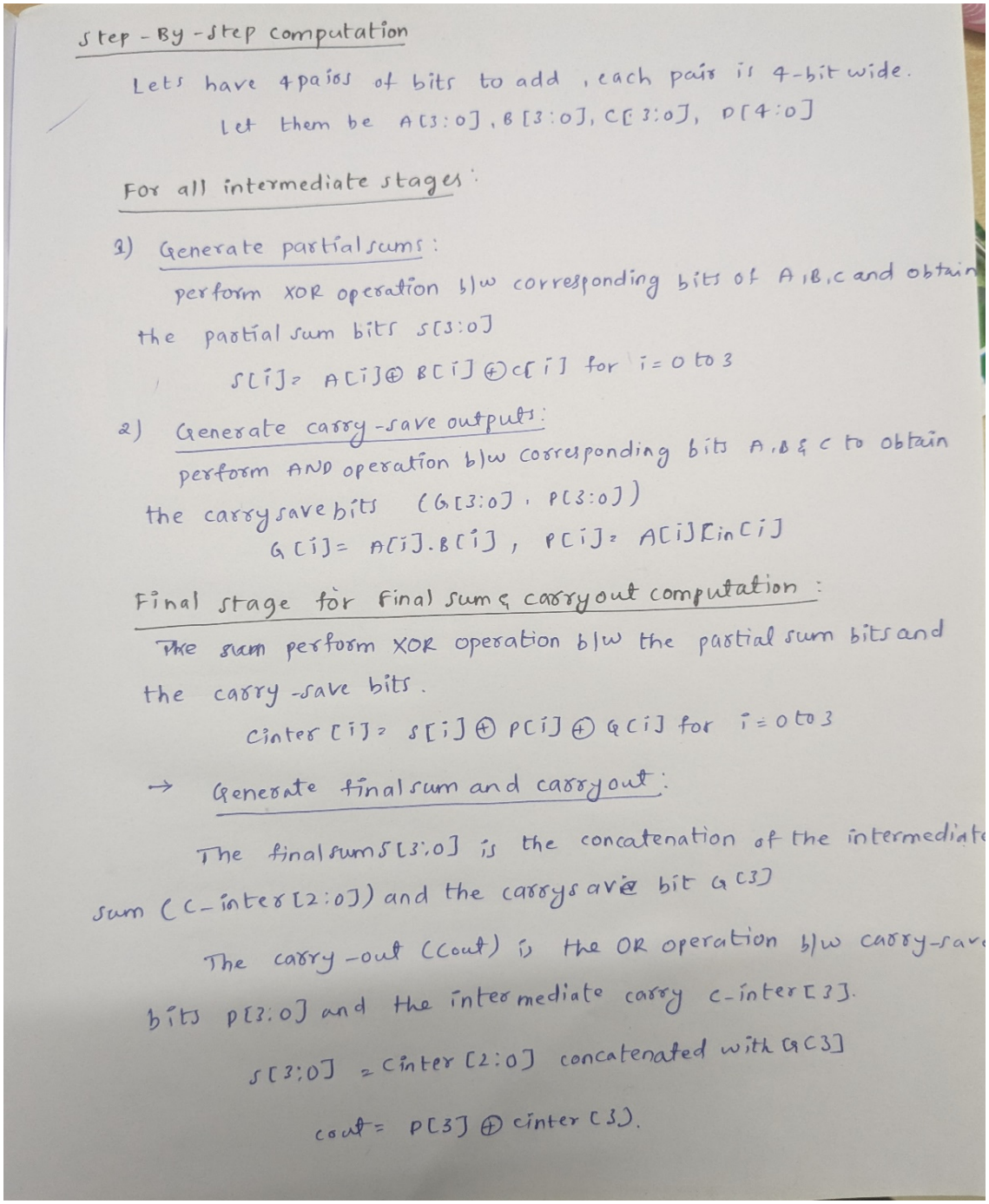


We can see outputs S0-S15 and Cout in the above waveforms

CARRY SAVE ADDER:



STEP BY STEP COMPUTATION:



Step - By - Step computation

Lets have 4 pairs of bits to add, each pair is 4-bit wide.

Let them be A[3:0], B[3:0], C[3:0], D[4:0]

For all intermediate stages:

1) Generate partial sums:

perform XOR operation b/w corresponding bits of A, B, C and obtain the partial sum bits S[3:0]

$S[i] = A[i] \oplus B[i] \oplus C[i]$ for $i = 0$ to 3

2) Generate carry-save outputs:

perform AND operation b/w corresponding bits A, B & C to obtain the carry save bits (G[3:0], P[3:0])

$G[i] = A[i].B[i]$, $P[i] = A[i] [in C[i]$

Final stage for Final sum & carry out computation:

The sum perform XOR operation b/w the partial sum bits and the carry-save bits.

$Cinter[i] = S[i] \oplus P[i] \oplus G[i]$ for $i = 0$ to 3

→ Generate final sum and carry out:

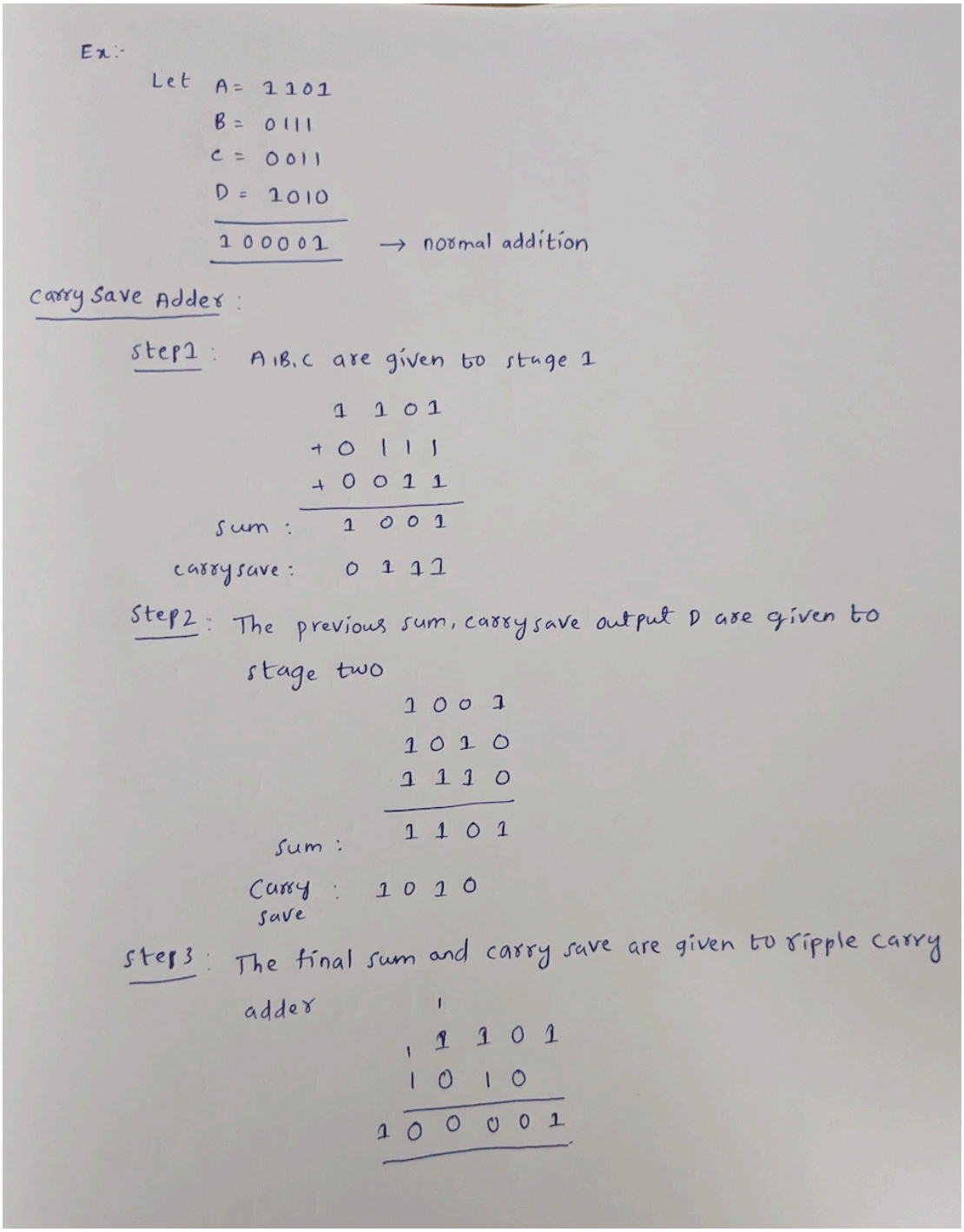
The final sum S[3:0] is the concatenation of the intermediate sum (C_inter[2:0]) and the carry save bit G[3]

The carry-out (Cout) is the OR operation b/w carry-save bits P[3:0] and the intermediate carry c_inter[3].

S[3:0] = Cinter[2:0] concatenated with G[3]

$cout = P[3] \oplus cinter[3]$.

EXAMPLE:



Ex:-

Let A = 1101
B = 0111
C = 0011
D = 1010

100001 → normal addition

Carry Save Adder :

Step1 : A, B, C are given to stage 1

```
   1 1 0 1
 + 0 1 1 1
 + 0 0 1 1
 ---------
sum :  1 0 0 1
carry save : 0 1 1 1
```

Step2 : The previous sum, carry save output D are given to stage two

```
   1 0 0 1
   1 0 1 0
   1 1 1 0
   -------
sum :  1 1 0 1
Carry save : 1 0 1 0
```

Step3 : The final sum and carry save are given to ripple carry adder

```
    1
  1 1 0 1
  1 0 1 0
 ---------
1 0 0 0 0 1
```

Applications:

Carry-Save Adders (CSAs) have various applications in digital arithmetic and computing due to their efficiency in certain operations. Here are some applications:

Multiplier Design:

CSAs are commonly used in the design of multipliers. They help optimize the partial product addition process, especially in high-performance multipliers used in applications such as digital signal processing (DSP) and graphics processing units (GPUs).

Parallel Prefix Adders:

CSAs are essential components in the design of parallel prefix adders, which are high-speed adder structures commonly used in modern processors. Examples include the Brent–Kung adder, Kogge-Stone adder, and other parallel prefix adder structures.

Fast Arithmetic Units:

In applications where speed is crucial, such as high-frequency trading or real-time signal processing, CSAs can be employed in arithmetic units to improve the overall speed and efficiency of addition operations.

FPGA and ASIC Implementations:

CSAs are suitable for implementation in Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) due to their regular and parallel structure. They can be optimized for specific applications, offering advantages in terms of area efficiency and speed.

Digital Signal Processing (DSP):

CSAs find applications in DSP algorithms, where the addition of multiple operands is a common operation. Their parallel structure can accelerate the processing of data streams in applications such as audio and image processing.

Quantum Computing:

In the emerging field of quantum computing, where qubits are manipulated to perform computations, CSAs or their quantum counterparts are explored for efficient addition operations. Quantum CSAs can be used in quantum arithmetic circuits.

Energy-Efficient Computing:

For applications where power consumption is a critical consideration, such as in mobile devices or battery-operated systems, CSAs can offer advantages in terms of energy efficiency due to their parallel nature and reduced carry propagation delay.

Digital Filters:

CSAs can be employed in the implementation of digital filters, where the summation of multiple terms is a common operation. They contribute to the optimization of the critical path in filter structures.

Advantages :

Parallelism:

One of the primary advantages of CSAs is their parallelism. They perform addition in a parallel manner, which means that all the bits are processed simultaneously. This parallel processing reduces the critical path delay compared to traditional ripple-carry adders, leading to faster operation.

High-Speed Addition:

Due to their parallel structure and reduced carry propagation delay, CSAs are well-suited for applications that require high-speed addition, such as in digital signal processing, graphics processing, and other computationally intensive tasks.

Multiplier Optimization:

CSAs are commonly used in multiplier designs to optimize the addition of partial products. In multiplication operations, CSAs help reduce the number of levels of adders, improving overall efficiency and speed.

Regular and Symmetric Structure:

CSAs have a regular and symmetric structure, making them suitable for implementation in hardware, including ASICs (Application-Specific Integrated Circuits) and FPGAs (Field-Programmable Gate Arrays). This regularity simplifies the design and layout process.

Reduced Energy Consumption:

In certain applications, the parallelism of CSAs can lead to reduced energy consumption compared to other adder architectures. This is particularly important in power-sensitive devices and applications.

Efficient for Multiple Operand Addition:

CSAs are efficient when it comes to adding more than two operands simultaneously. This makes them suitable for applications where the addition of multiple numbers is a common operation, such as in digital filters or certain types of arithmetic circuits.

Conclusion:

Hence designed and implemented a 16-bit Square Root Carry Select adder using Cadence Virtuoso. Reported the average power dissipation, propagation delay, and worst-case delay.
Theoretically, demonstrated and explained the computation of a four input 4-bit Carry Save adder and mentioned the applications and advantages of using Carry Save adder