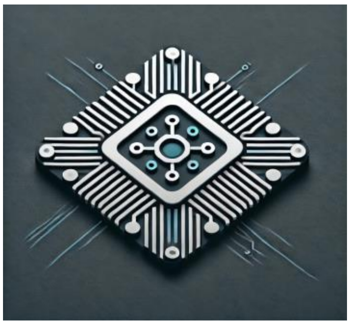


Project report on

3:2 COMPRESSOR USINGCMOS

Submitted by:

ANIRUDH PAL

A report submitted for the partial fulfilment of the requirements of the course

ECL- CMOS

Submission Date: 11/11/2024

Under the guidance of:

Dr. Paritosh Peshwe

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

Chapter 1: Introduction

A 3:2 compressor is a digital circuit used in arithmetic operations, particularly in multi-bit adders and multipliers. It takes three binary inputs and produces two outputs (a sum and a carry), compressing three bits of information into two bits. The purpose of a 3:2 compressor is to simplify addition by reducing the number of partial sums, especially in applications like multiplication where a large number of bits need to be added together.

Components and Functionality:

Compone

XOR Gates: The circuit has two XOR gates, each labeled "XNOR_XOR." These are used to calculate intermediate XOR results.

XNOR Gates: The circuit also appears to use XNOR functionality, potentially as part of the "XNOR_XOR" blocks.

Multiplexer (mux): This component selects between two values based on a control signal to produce the carry output.

LED Indicators: There are two LED indicators labeled "sum" and "CARRY," which display the outputs of the circuit.

Functionality

The 3:2 compressor circuit takes three inputs, AAA, BBB, and CCC, and produces two outputs: a sum (S) and a carry (CARRY). Here's how it works:

1. **Sum Calculation:**

The two XOR gates produce the intermediate results of A⊕BA \oplus BA⊕B and A⊕B⊕CA \oplus B \oplus CA⊕B⊕C.

The final sum output (S) is the XOR of all three inputs: S=A⊕B⊕CS = A \oplus B \oplus CS=A⊕B⊕C.

Carry Calculation:

The carry output is based on the majority function, where the carry is high if at least two of the three inputs are high.

This is implemented using the equation: CARRY=(A⊕B)·C+(A⊕B)·A\text{CARRY} = (A \oplus B) \cdot C + (A \oplus B) \cdot ACARRY=(A⊕B)·C+(A⊕B)·A

The multiplexer (mux) uses the intermediate XOR values to determine the correct carry output.

2. **LED Indicators:**

The "sum" LED shows the state of the sum output, while the "CARRY" LED displays the carry output**.**

Use Cases:

3:2 compressors are commonly used in digital arithmetic circuits where high-speed addition and multiplication are required. Their primary purpose is to simplify and accelerate multi-bit addition operations by reducing the number of partial sums that need to be calculated. Here are some key use cases for 3:2 compressoFuture Enhancements:

Multipliers: Used in multipliers like Wallace trees to reduce partial products quickly, making multiplication faster.

Parallel Adders: Used in high-speed adders to handle multiple binary additions in parallel.

Digital Signal Processing (DSP): Speeds up operations like FFTs and convolutions.

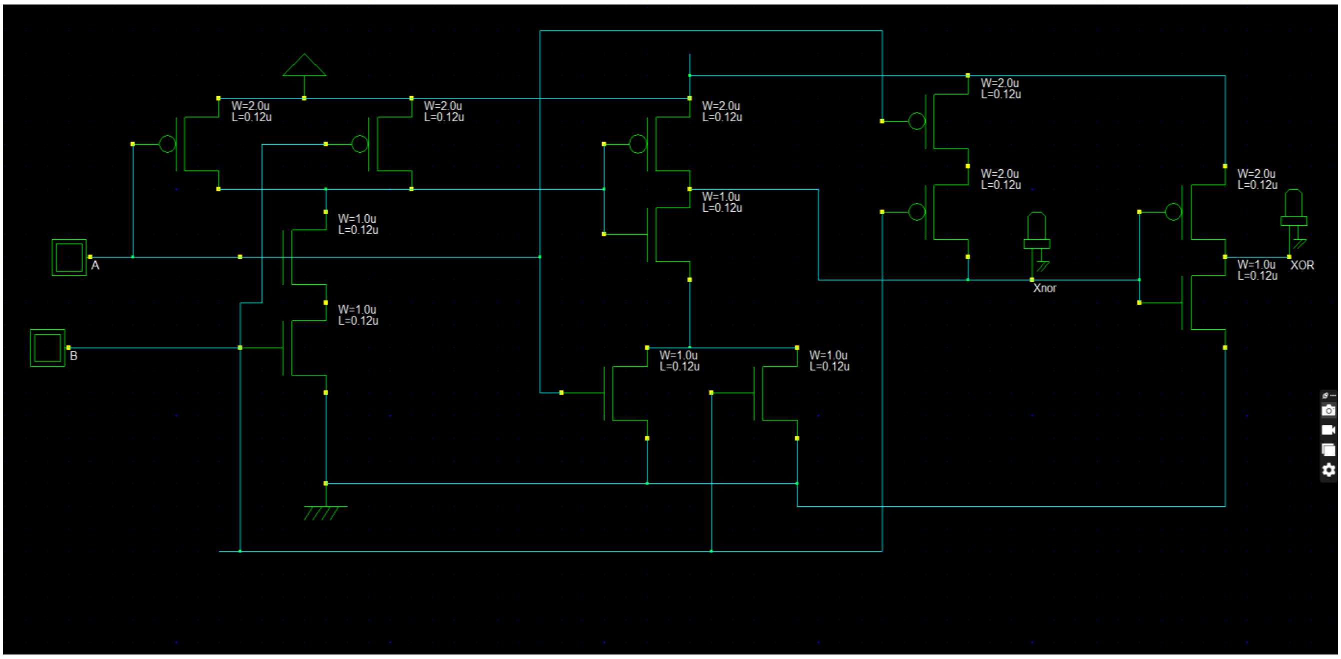
ALUs (Arithmetic Logic Units): Improves speed in CPUs and GPUs by optimizing addition and multiplication.

AI/ML Accelerators: Enhances performance in matrix operations and convolution tasks for AI processing.

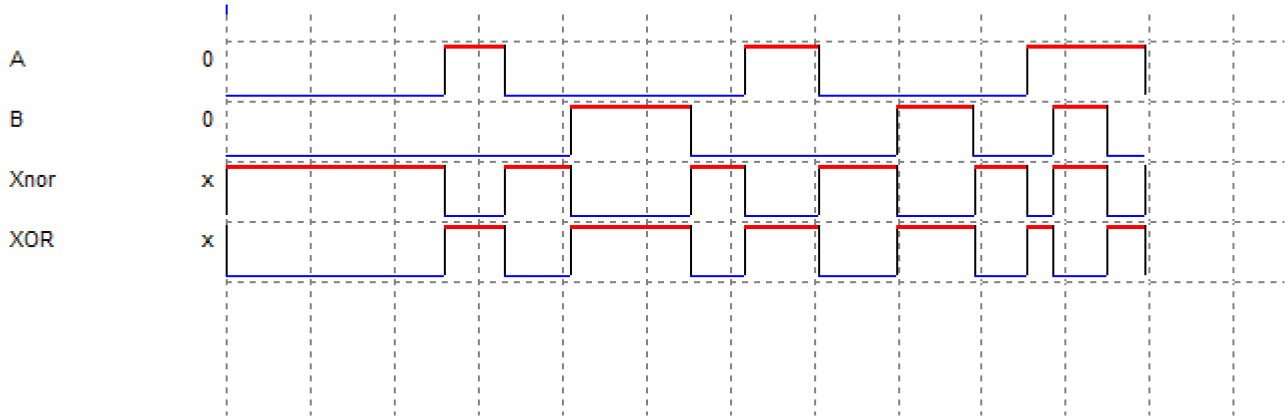
Image/Video Processing: Speeds up pixel operations in real-time image and video processing.battery-operated devices.

Chapter 2: DSCH circuit

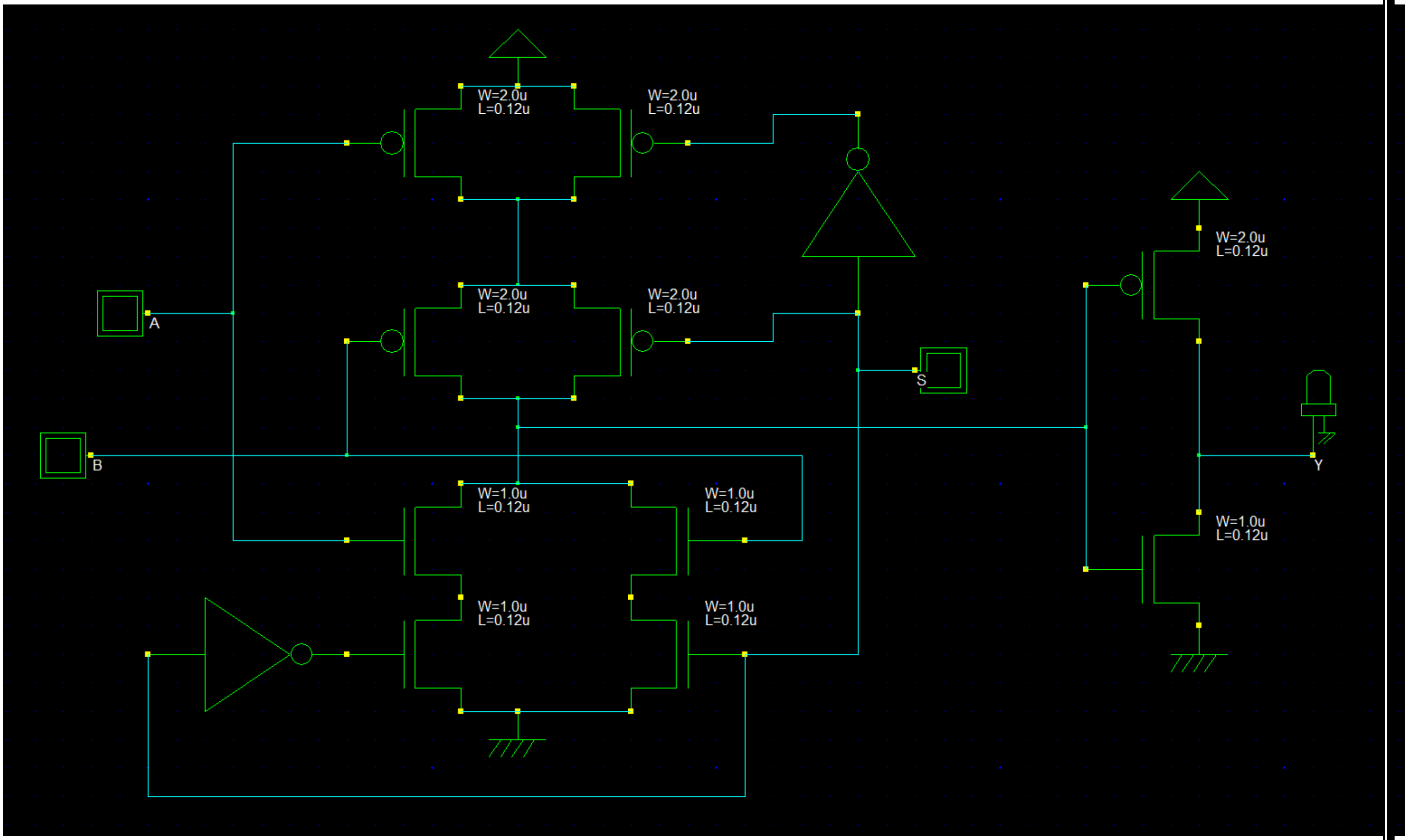
XNOR and XOR Circuit:



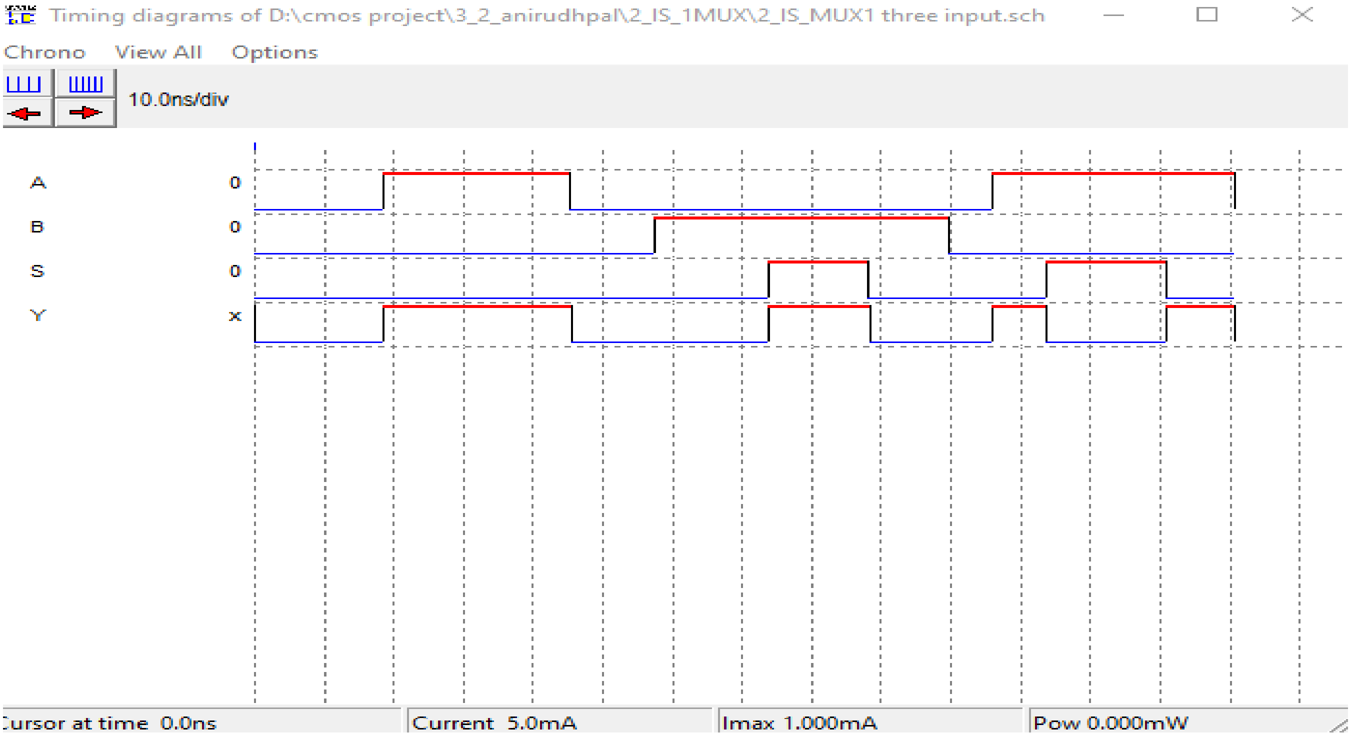
Output:



2:1 Mux Circuit:



output:


Timing diagrams of D:\cmos project\3_2_anirudhpal\2_IS_1MUX\2_IS_MUX1 three input.sch

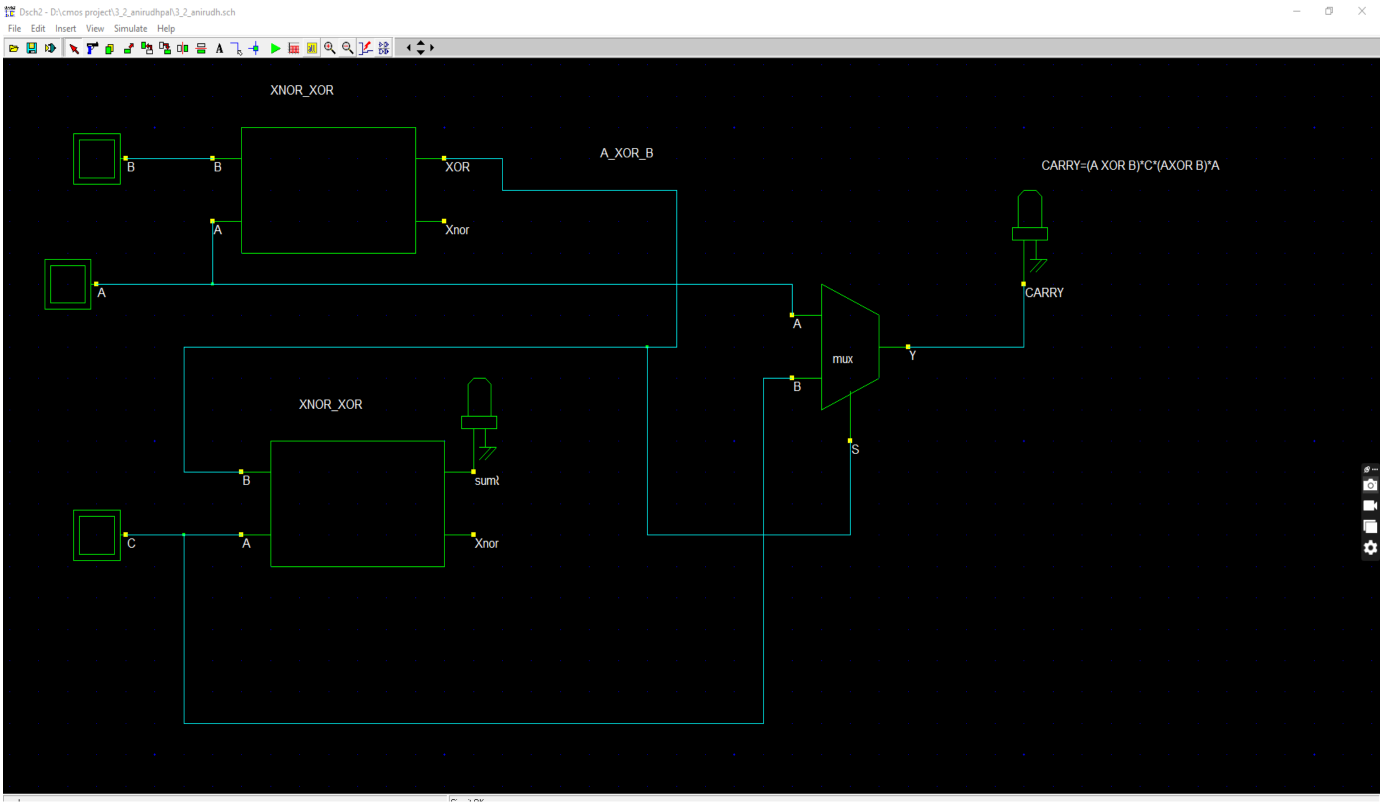
ursor at time 0.0ns

Current 5.0mA

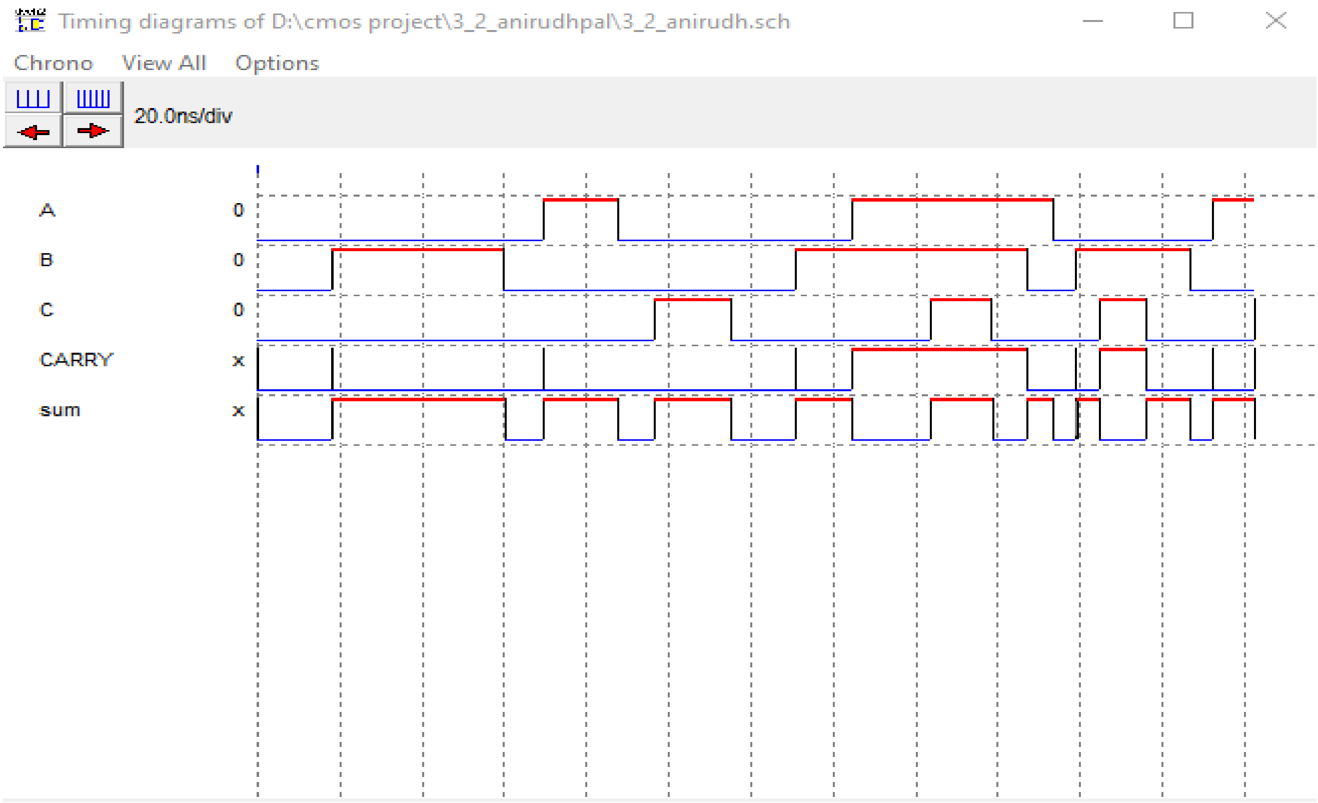
Imax 1.000mA

Pow 0.000mW

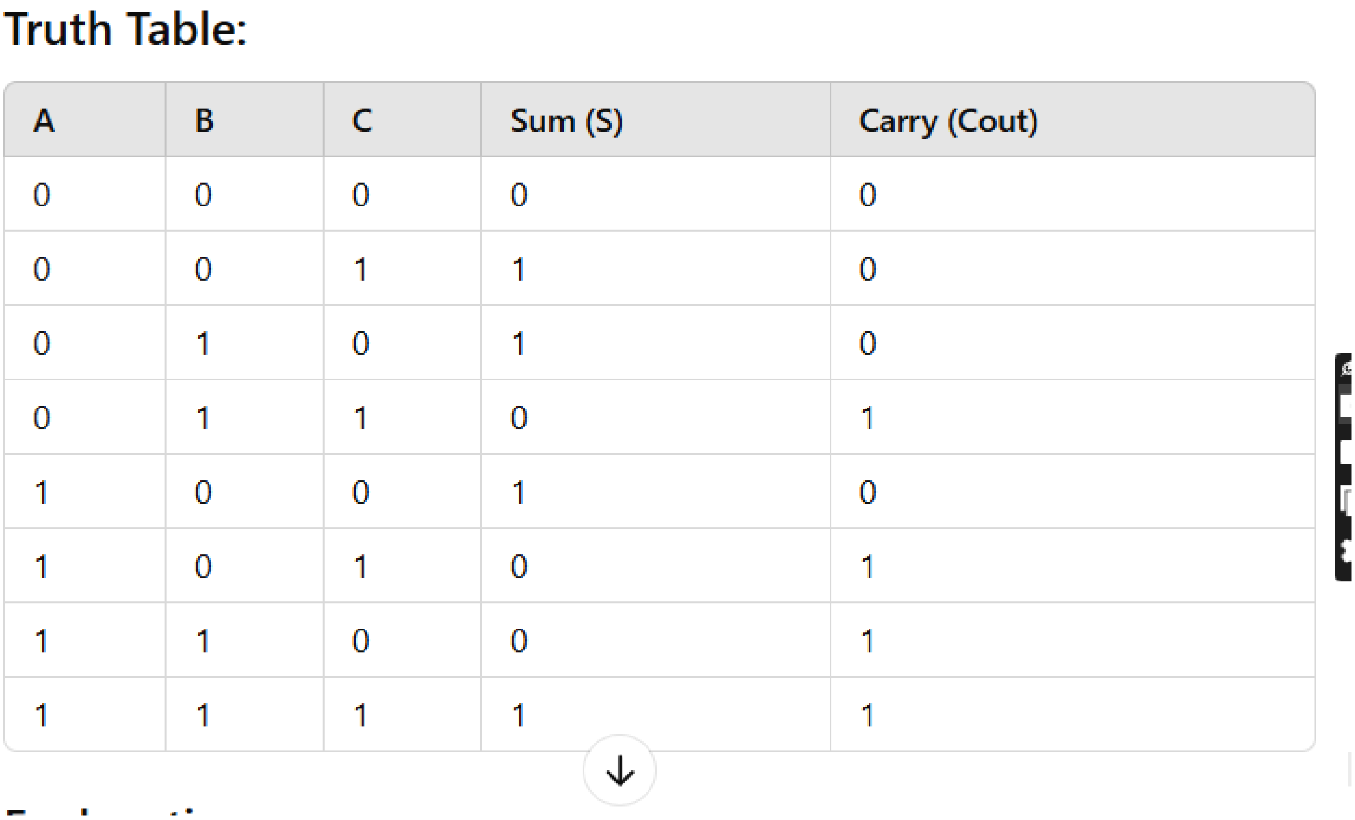
3:2 Compressor Circuit:


Dsch2 - D:\cmos project\3_2_anirudhpal\3_2_anirudh.sch
File Edit Insert View Simulate Help
XNOR_XOR
B
B
XOR
Xnor
A
A
A_XOR_B
CARRY=(A XOR B)*C*(AXOR B)*A
CARRY
A
mux
Y
B
S
XNOR_XOR
sum
B
C
A
Xnor

output:



3:2 Compressor Circuit Truth table

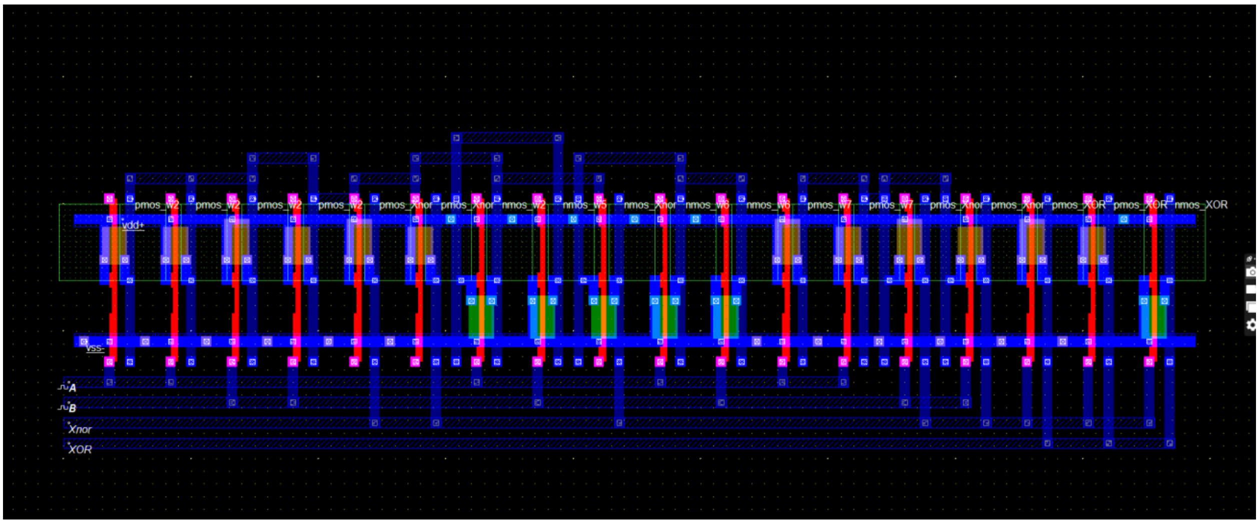


Truth Table:

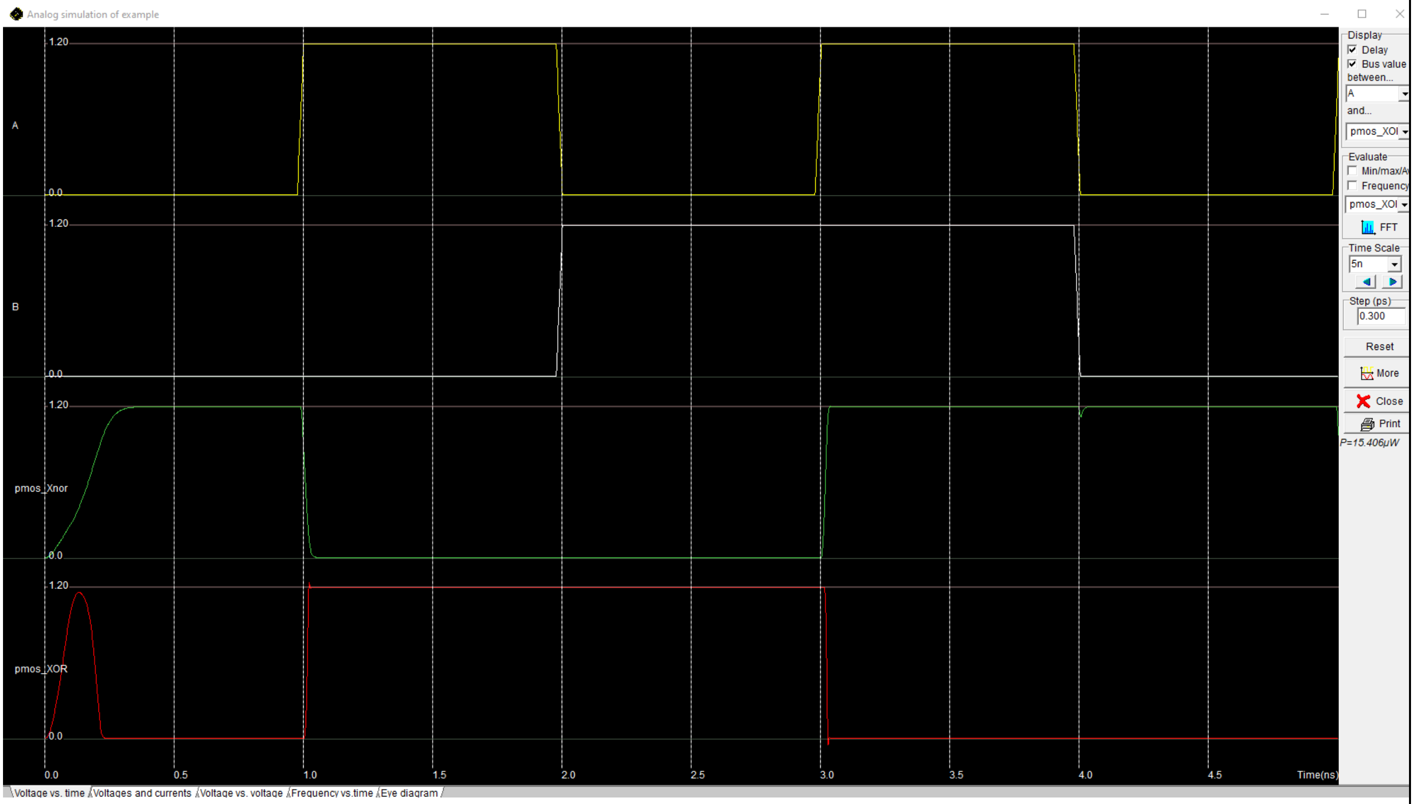
A	B	C	Sum (S)	Carry (Cout)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Chapter 3 Layout in Microwind:

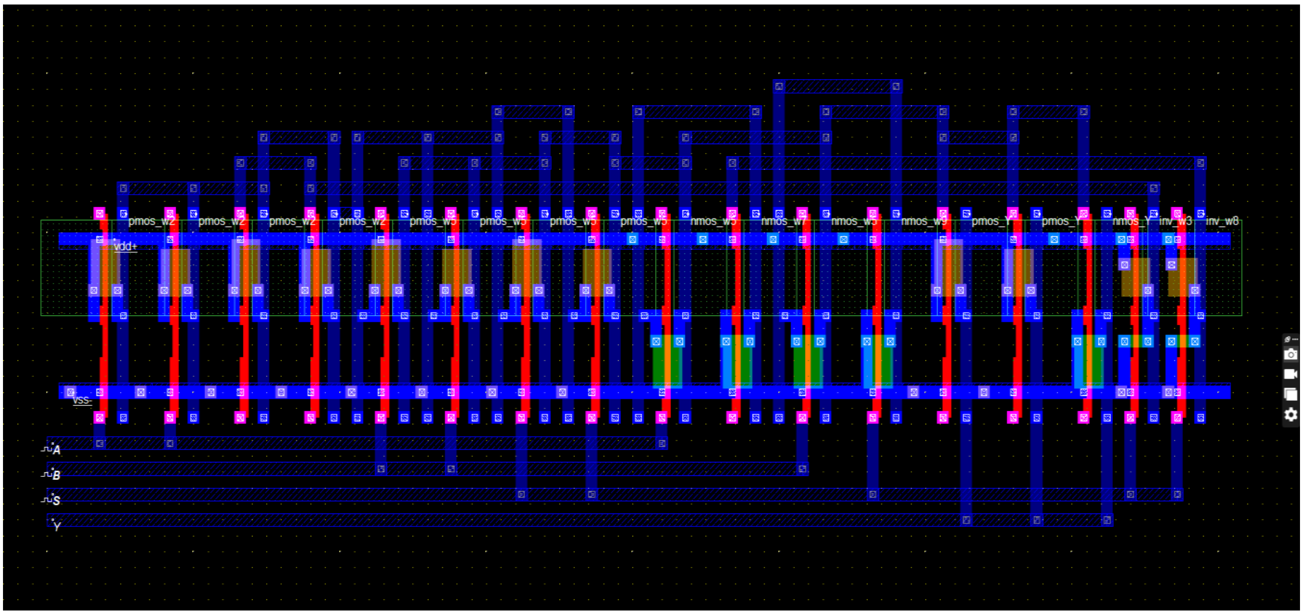
XNOR and XOR Layout:



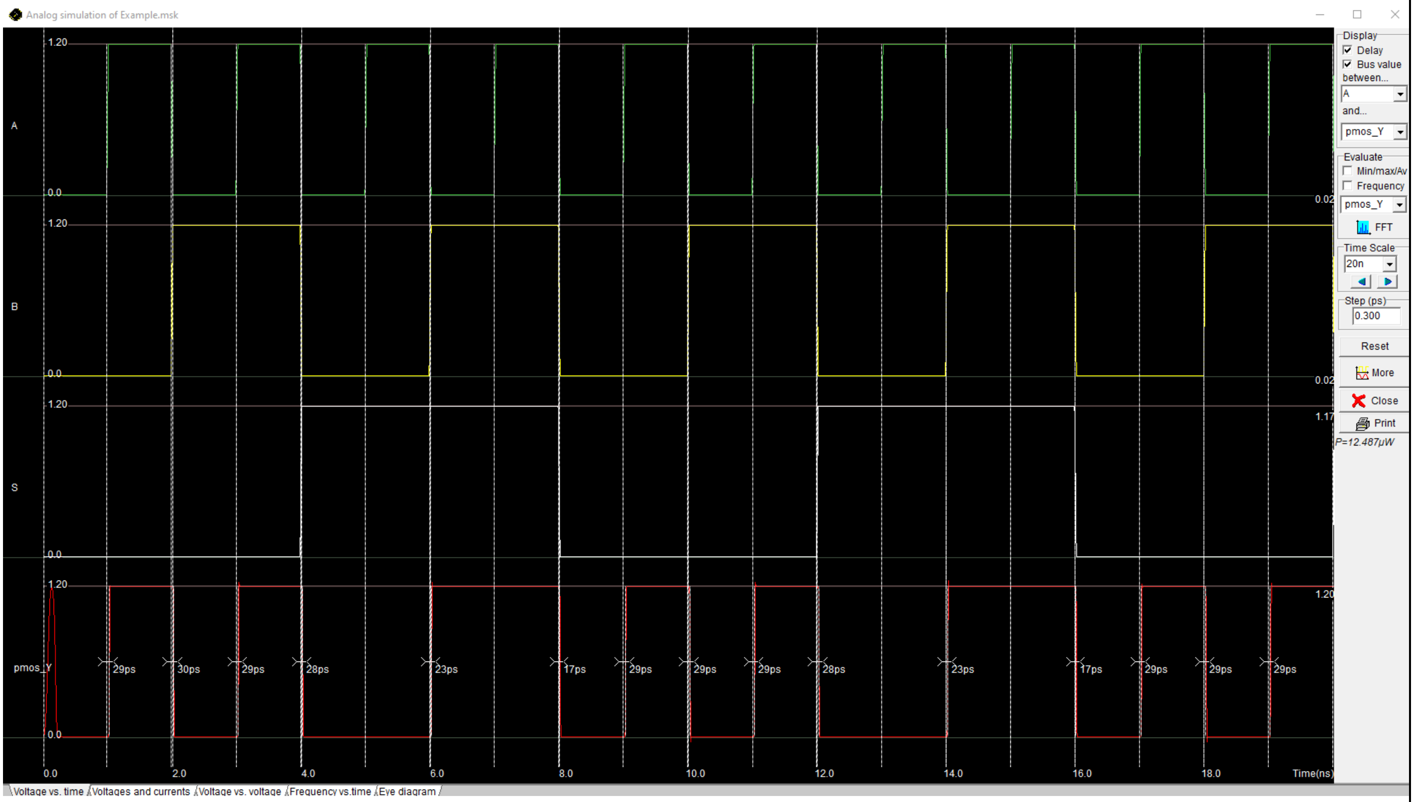
output:



2:1 MUX Layout:

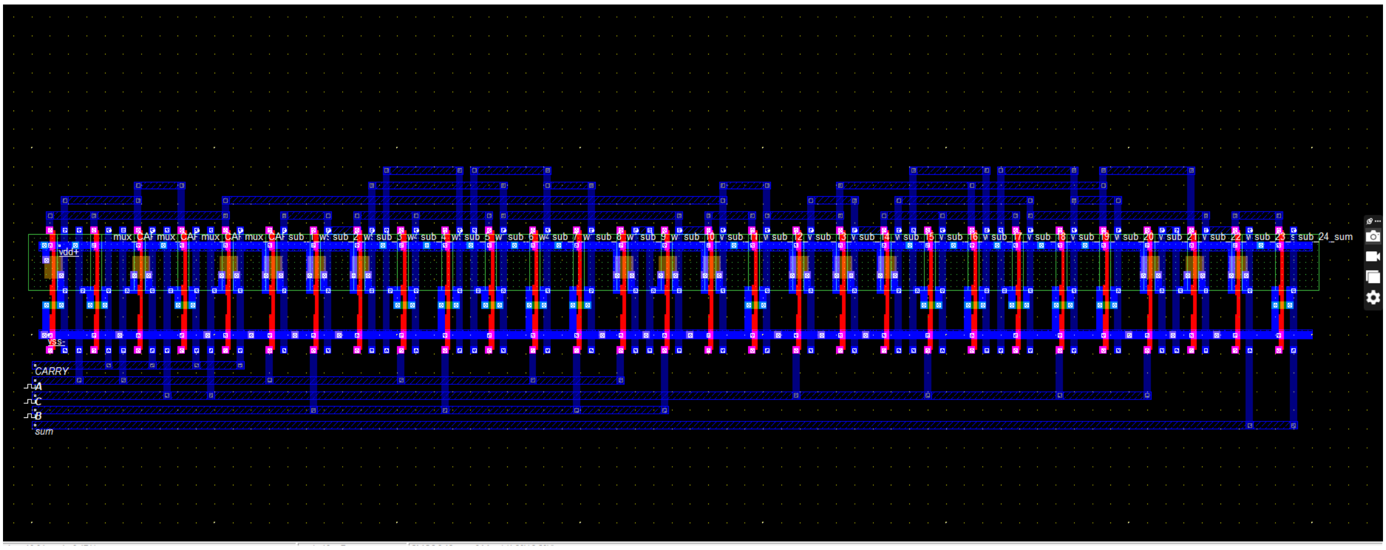


output:

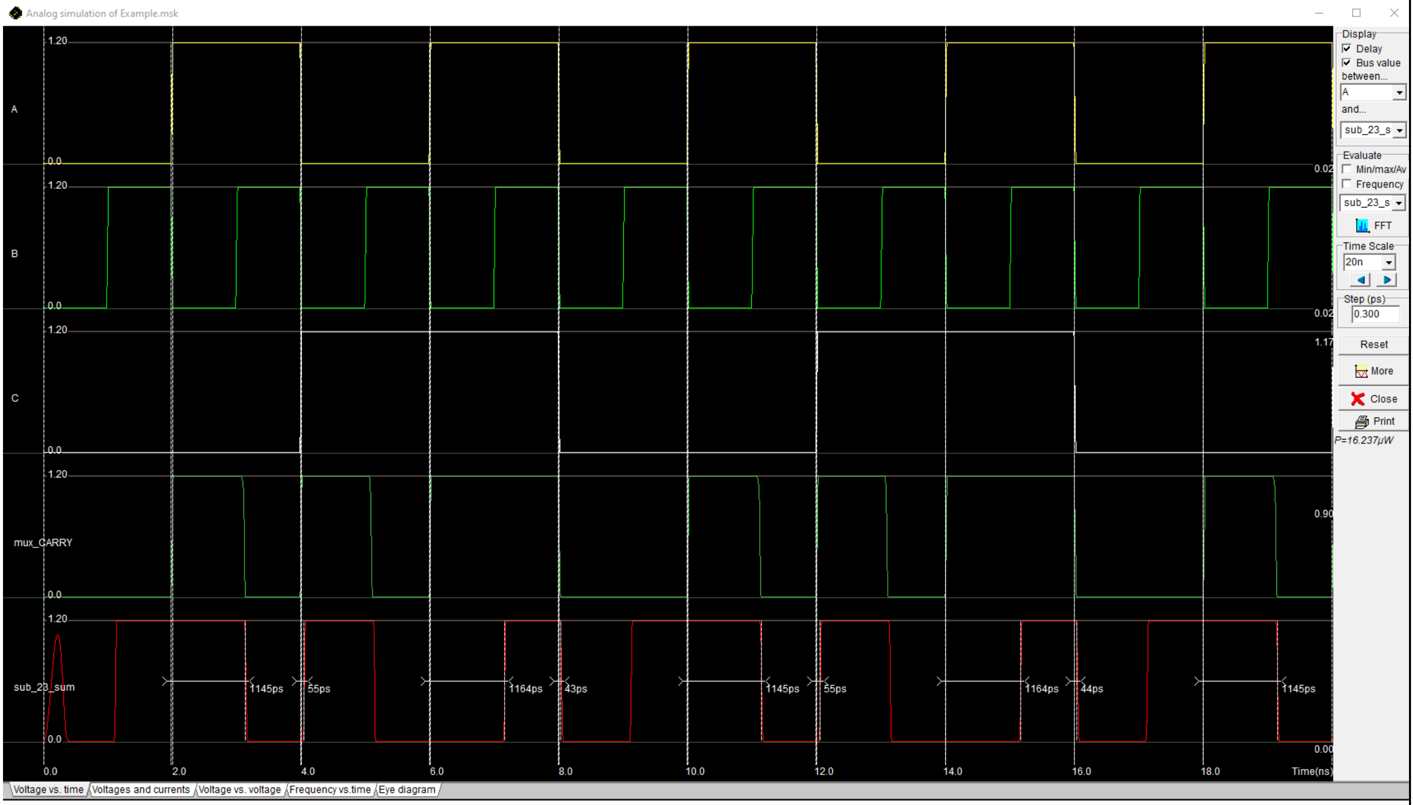


Display
☑ Delay
☑ Bus value
between...
A
and...
pmos_Y
Evaluate
□ Min/max/Av
□ Frequency
pmos_Y
Time Scale
20n
Step (ps)
0.300
P=12.487µW

3:2 Compressor Layout:



output:


Analog simulation of Example.msk

Display
☑ Delay
☑ Bus value
between...
A
and...
sub_23_s

Evaluate
□ Min/max/Av
□ Frequency
sub_23_s

Time Scale
20n

Step (ps)
0.300

P=16.237µW

Chapter 3 code:

```
     * 3:2 Compressor Circuit Netlist  for WinSpice
    * Define inputs, outputs, and internal connections
    * Inputs: A, B, C
    * Outputs: S (Sum), Cout (Carry)
    * Define subcircuits for basic gates using pass transistor logic
    .subckt ckt_one 1 3 6 7
    .model pmod pmos level=54 version=4.7
    .model nmod nmos level=54 version=4.7

    M1 2 1 0 0 nmod w=100u l=10u
    M2 5 3 2 2 nmod w=100u l=10u
    M5 7 3 0 0 nmod w=100u l=10u
    M6 6 5 7 7 nmod w=100u l=10u
    M8 7 1 0 0 nmod w=100u l=10u
    M11 10 6 0 0 nmod w=100u l=10u

    M3 5 3 4 4 pmod w=100u l=10u
    M4 5 4 1 1 pmod w=100u l=10u
    M7 6 5 4 4 pmod w=100u l=10u
    M9 9 3 4 4 pmod w=100u l=10u
    M10 6 1 9 9 pmod w=100u l=10u
    M12 10 6 4 4 pmod w=100u l=10u
    .ends

    * Define subcircuit for pass_and
    .subckt pass_and 1 2 3
    M1 3 1 0 0 nmod w=100u l=10u
    M2 3 2 0 0 nmod w=100u l=10u
    .ends

    * Define subcircuit for pass_or
    .subckt pass_or 1 2 3
    M1 3 1 0 0 pmod w=100u l=10u
    M2 3 2 0 0 pmod w=100u l=10u
    .ends

    * Define subcircuit for pass_inv (inverter)
    .subckt pass_inv 1 2
    M1 2 1 Vdd Vdd pmod w=100u l=10u
    M2 2 1 0 0 nmod w=100u l=10u
    .ends

    * Input sources
    V1 A 0 DC 0
    V2 B 0 DC 0

    V3 C 0 DC 0
    Vdd Vdd 0 DC 5

* XOR Gate for A XOR B
X1 A B AxorB_unused AxorB ckt_one
```

```
* XOR Gate for (A XOR B) XOR C - Sum Calculation
X2 AxorB C S_unused S ckt_one

* AND Gate for (A XOR B) * C
X3 AxorB C And1_out_unused And1_out pass_and

* AND Gate for A * B
X4 A B And2_out_unused And2_out pass_and

* OR Gate for Carry Calculation: Cout = And1_out + And2_out
X5 And1_out And2_out Cout_unused Cout pass_or

* Define MUX logic using pass transistor subcircuits
* MUX selection: S and Cout outputs
Va 10 0 DC 0
Vb 11 0 DC 5
Vs 12 0 PULSE(0 5 0 0 0 100m 200m)

* Inverter for MUX selection signal
Xs_not 12 MUX_not pass_inv

* MUX AND gates for selection logic
Xand_s 10 MUX_not S_MUX pass_and
Xand_cout 11 12 Cout_MUX pass_and

* OR gate for final MUX output
Xor_mux S_MUX Cout_MUX Output_MUX pass_or

* Simulation Control
.tran 1ns 10ns
. run
plot v(A) v(B) v(C) v(sum) v(carry)
.endc
.end
```

Chapter 3 code Explanation:

This **WinSpice netlist** is designed to implement a **3:2 compressor** circuit using **pass transistor logic** and **MUX** (multiplexer) logic. Here's a brief explanation of the code:

1. Subcircuits for Basic Gates:

- **ckt_one**: Defines a custom **XOR** gate using NMOS and PMOS transistors (pass-transistor logic).
- **pass_and**: Defines a basic **AND** gate using two NMOS transistors.
- **pass_or**: Defines a basic **OR** gate using two PMOS transistors.
- **pass_inv**: Defines a **NOT** gate (inverter) using PMOS and NMOS transistors.

2. Inputs:

- **V1, V2, V3**: Input voltages for the three input bits AAA, BBB, and CCC, respectively.
- **Vdd**: Supply voltage for the circuit (5V).

3. XOR Gates:

- **X1**: Calculates A⊕BA \oplus BA⊕B (XOR of AAA and BBB).
- **X2**: Calculates the final **Sum (S)** as (A⊕B)⊕C(A \oplus B) \oplus C(A⊕B)⊕C.

4. AND Gates:

- **X3**: Computes (A⊕B)·C(A \oplus B) \cdot C(A⊕B)·C for the carry calculation.
- **X4**: Computes A·BA \cdot BA·B for the carry calculation.

5. OR Gate for Carry:

- **X5**: Computes the final **Carry (Cout)** as (A⊕B)·C+A·B(A \oplus B) \cdot C + A \cdot B(A⊕B)·C+A·B.

6. Multiplexer (MUX) Logic:

- **MUX selection signal**: Defined by Vs, a **pulse** signal toggling between 0 and 5V.
- **Inverter Xs_not**: Inverts the MUX selection signal.
- **AND gates Xand_s and Xand_cout**: Select the **Sum (S)** and **Carry (Cout)** outputs based on the MUX logic.
- **Final OR gate Xor_mux**: Combines the selected **Sum** and **Carry** outputs.

7. Simulation Control:

- **trans 1ns 10ns**: Defines the transient analysis, simulating the circuit for 10ns with a time step of 1ns.

print train: Specifies that the voltages at AAA, BBB, CCC, SSS, and CoutCoutCout should be printed during the simulation.

In Summary:

This code models a **3:2 compressor** circuit using basic pass-transistor logic gates (XOR, AND, OR) and a **multiplexer (MUX)** to select between the sum and carry outputs. It is intended to simulate the behavior of the compressor with a transient analysis, providing the voltage signals for each input and out