# Computer Organization and Architecture Laboratory

## Assignment 1

## Introduction to Verilog Programming

Group 53
Umang Singla - 20CS10068
Siddharth Vishwakarma - 20CS10082

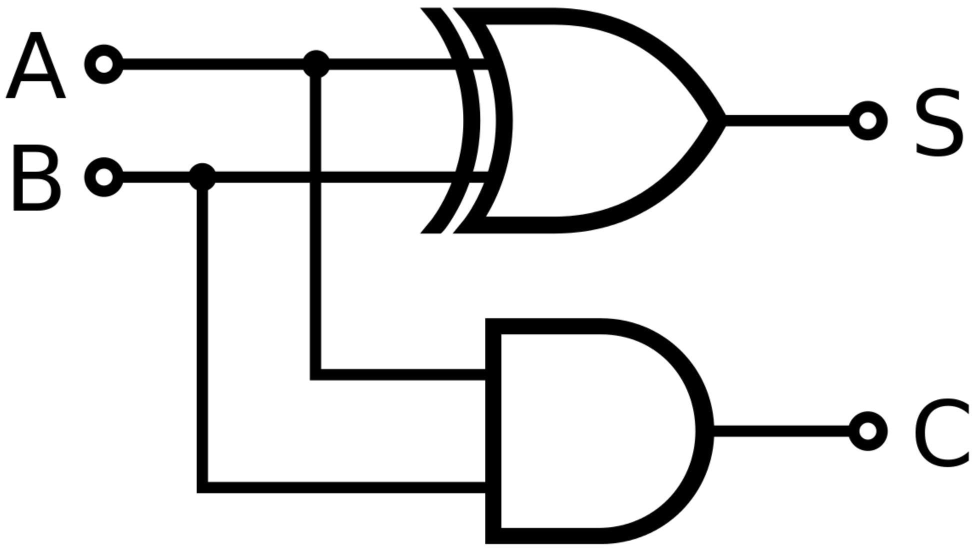
## Half Adder

Half Adder is a combinational arithmetic circuit that adds two bits (A and B) and outputs a sum bit (S) and a carry bit (C).

**Truth Table for Half Adder**

| **A (input)** | **B (input)** | **C (output)** | **S (output)** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

We can write the boolean expression for outputs as:
S = $A \otimes B$
C = $A.B$



**Relevant File(s):** *half_addr.v, test_half_addr.v*

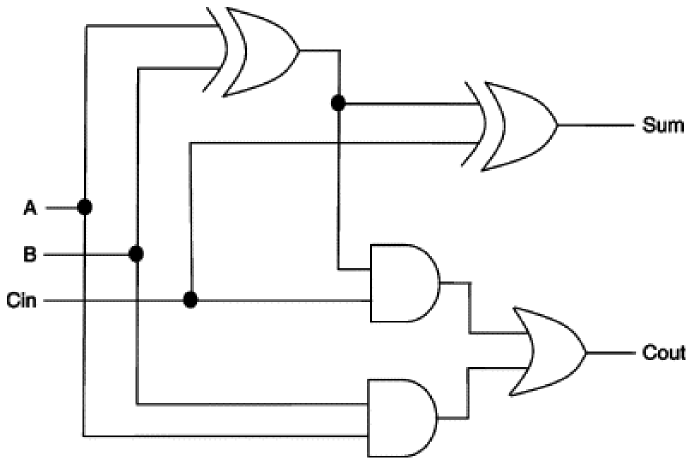
## Full Adder

Half Adder is a combinational arithmetic circuit that adds three bits (A, B, and C-IN) and outputs a sum bit (S) and a carry bit (C-Out).

### Truth Table for Full Adder

| A (input) | B (input) | C-IN (input) | Sum (output) | C-Out (output) |
|---|---|---|---|---|
| 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 |

We can write the boolean expression for outputs as:
Sum = C-IN ⊗ (A ⊗ B)
C = A.B + A.C-IN + B.C-IN



**Relevant File(s):** *full_addr.v, test_full_addr.v*

## Ripple Carry Adder

In the first step of this section, we cascade 8 full adders to create an 8-bit ripple carry adder. Next, a 16-bit ripple carry adder is created by cascading two 8-bit ripple carry adders.
To create a 32-bit ripple carry adder, we similarly cascade two 16-bit ripple carry adders.
The 64-bit ripple carry adder is created by cascading two 32-bit ripple carry adders.

**Relevant File(s):** *Ripple_Carry_Adder_8bit.v*
*Ripple_Carry_Adder_16bit.v*
*Ripple_Carry_Adder_32bit.v*
*Ripple_Carry_Adder_64bit.v*

## How can you use the above circuit, to compute the difference between 2 n bits numbers?

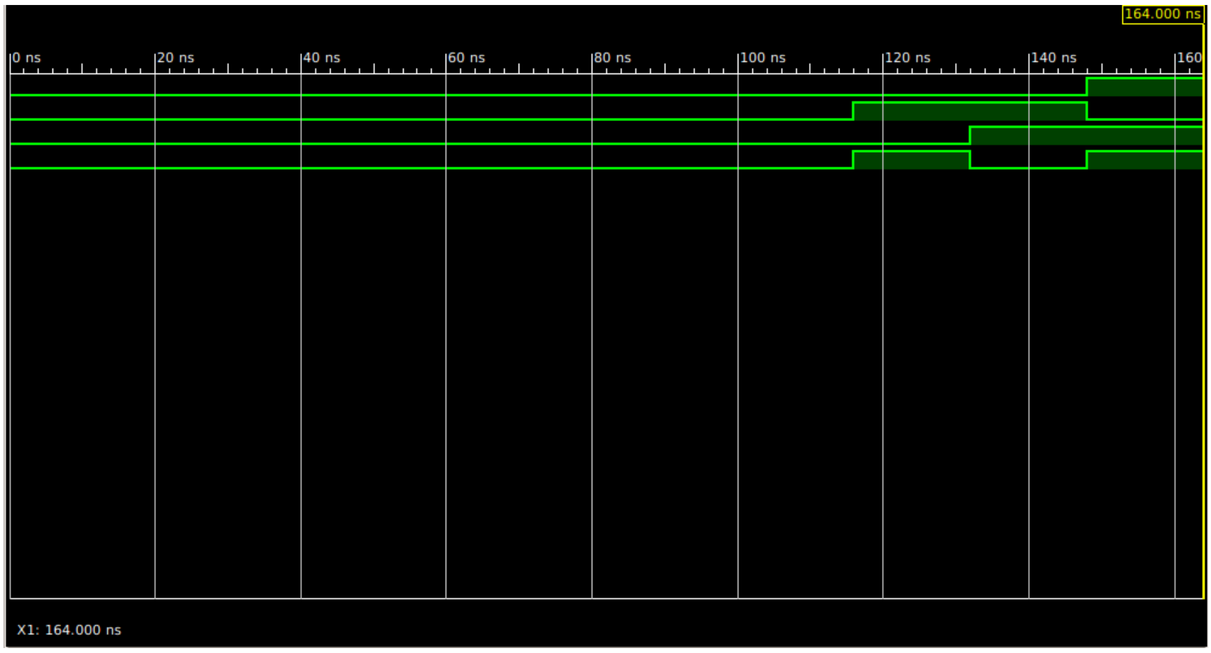
A ripple carry adder is used to calculate the sum of two n bits numbers. But we can modify our inputs to the adder so as to calculate the difference between 2 numbers as follows:

- a - b = a + (-b)
- -b is 2's complement of b, which can be computed as -b = ~b + 1, where ~b is 1's complement of b.
- So, we can calculate the difference between two numbers by passing a and ~b as input and 1 as carry-in.

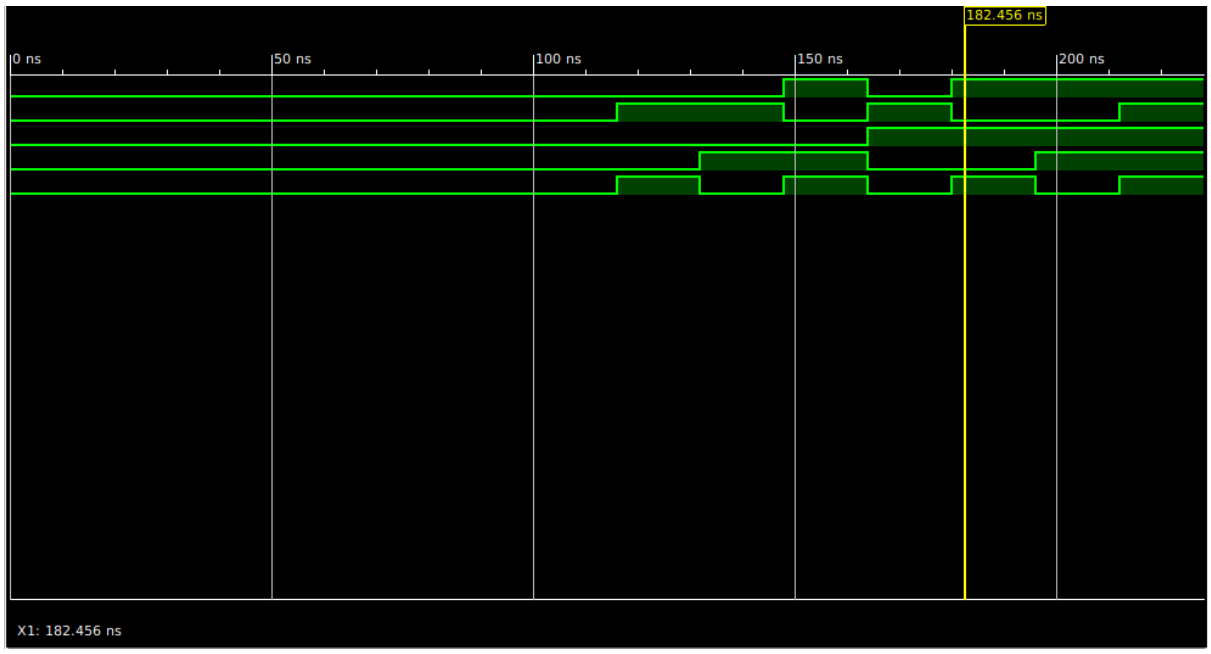
## Synthesis Summary

| Circuit | Total Delay (in ns) |
|---|---|
| Half Adder | 1.066 |
| Full Adder | 1.246 |
| 8-bit RCA | 3.471 |
| 16-bit RCA | 6.167 |
| 32-bit RCA | 11.559 |
| 64-bit RCA | 22.343 |

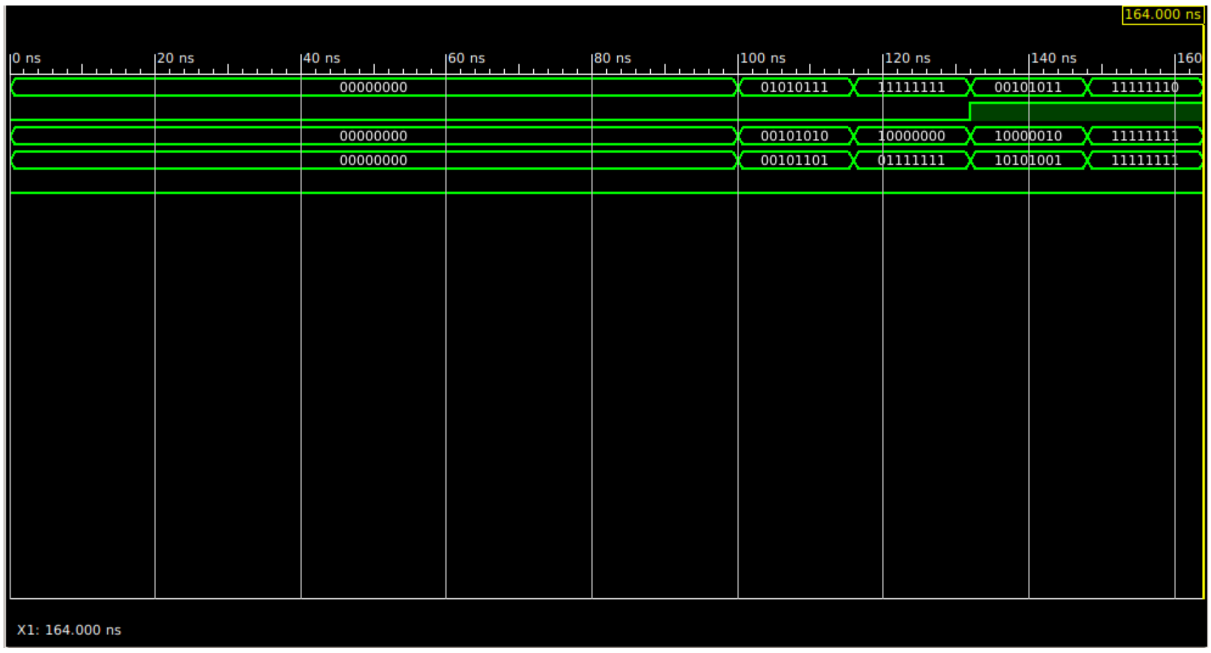
## Behavioral Simulation Screesshots:



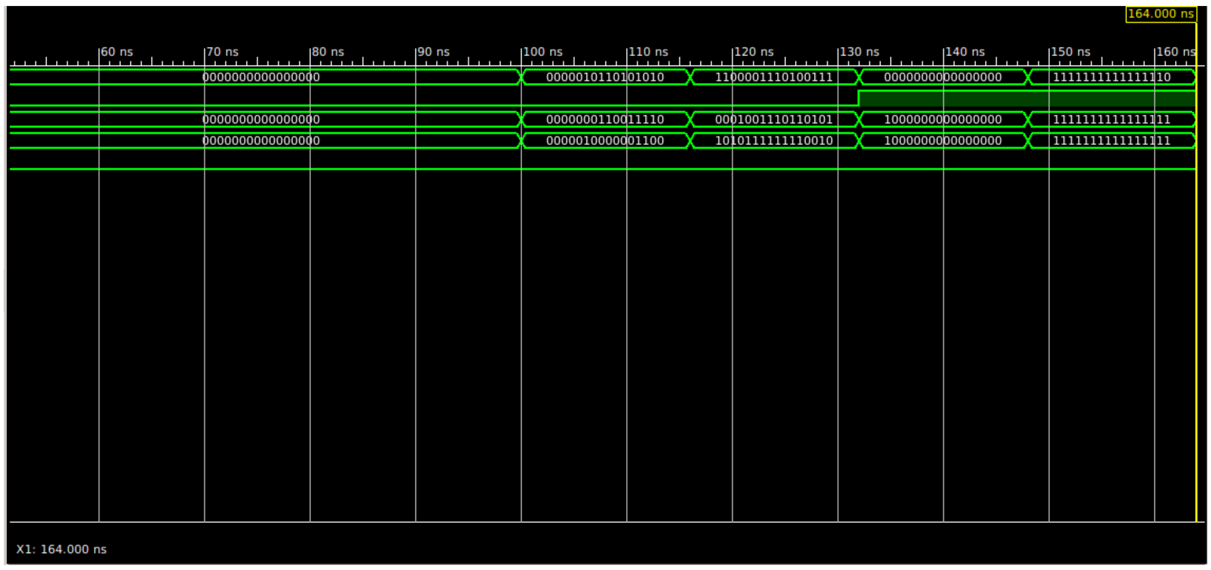
HALF ADDER



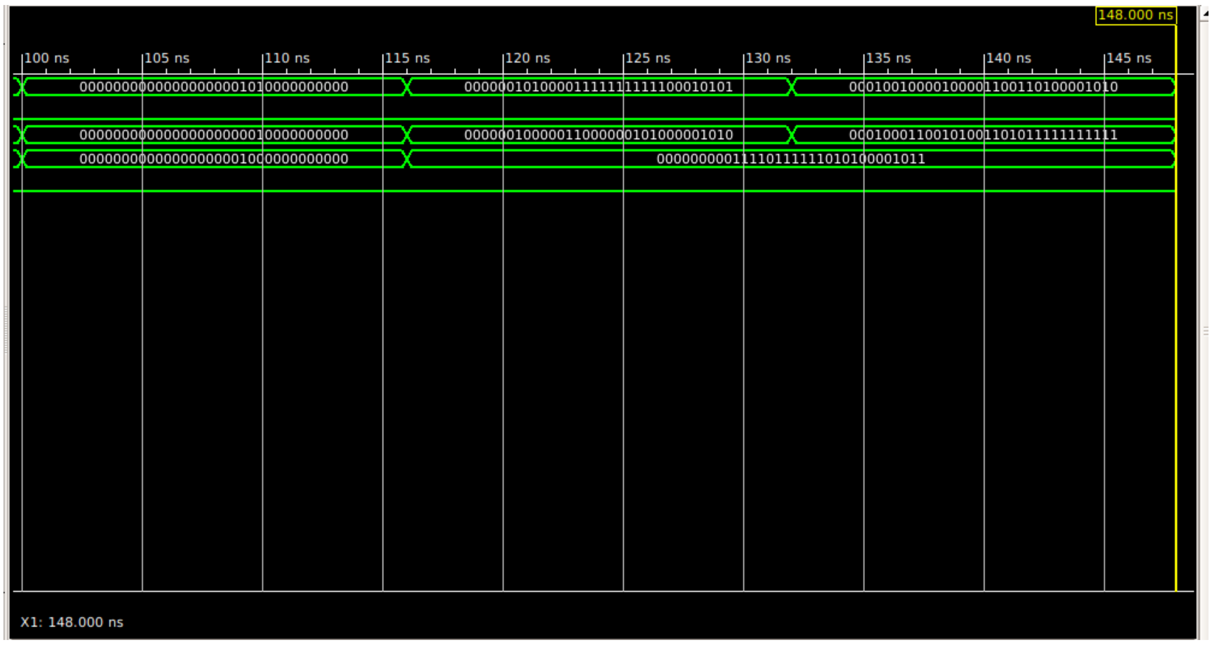
FULL ADDER



**8-BIT RCA**

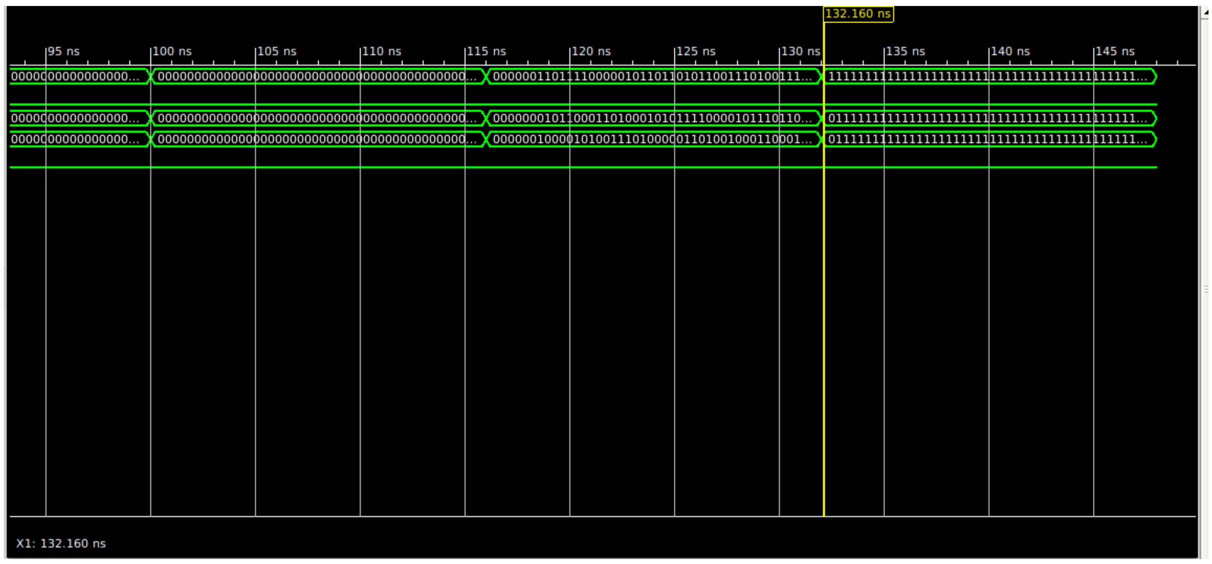


**16-BIT RCA**



```
00000000000000000000101000000000000   00000010100001111111111111100010101   00010010000100001100110100001010
00000000000000000000010000000000000   00000010000011000000101000001010    00010001100101001101011111111111
00000000000000000000100000000000000   00000000011110111111010100001011
```

**32-BIT RCA**



```
0000000000000000...   00000000000000000000000000000000000000000000...   000000110111110000010110110101100111010011 1...   11111111111111111111111111111111111111111111...
0000000000000000...   00000000000000000000000000000000000000000000...   0000000101100011010001010111100001011101 10...   01111111111111111111111111111111111111111111...
0000000000000000...   00000000000000000000000000000000000000000000...   0000001000010100111010000011010010001100 01...   01111111111111111111111111111111111111111111...
```

**64-BIT RCA**

## Relevant Files for Test Bench:

- *test_half_adder.v*
- *test_full_adder.v*
- *test_RCA_8bit.v*
- *test_RCA_16bit.v*
- *test_RCA_32bit.v*
- *test_RCA_64bit.v*
- *test_RCA_difference.v*

**Carry Look-ahead Adder (CLA)**

**(a) 4-bit Carry Look-ahead Adder**

For a carry look-ahead adder we can compute output carry, $C_{i+1}$ from input carry bit, $C_i$ as:

$C_{i+1} = G_i + P_i \cdot C_i$

where,

$G_i = A_i \cdot B_i$ (carry generator)

$P_i = A_i \oplus B_i$ (carry propagator)

If we expand this recurrence relation, we get the expressions for the carry functions :

($C_0$ is the initial carry-in bit)

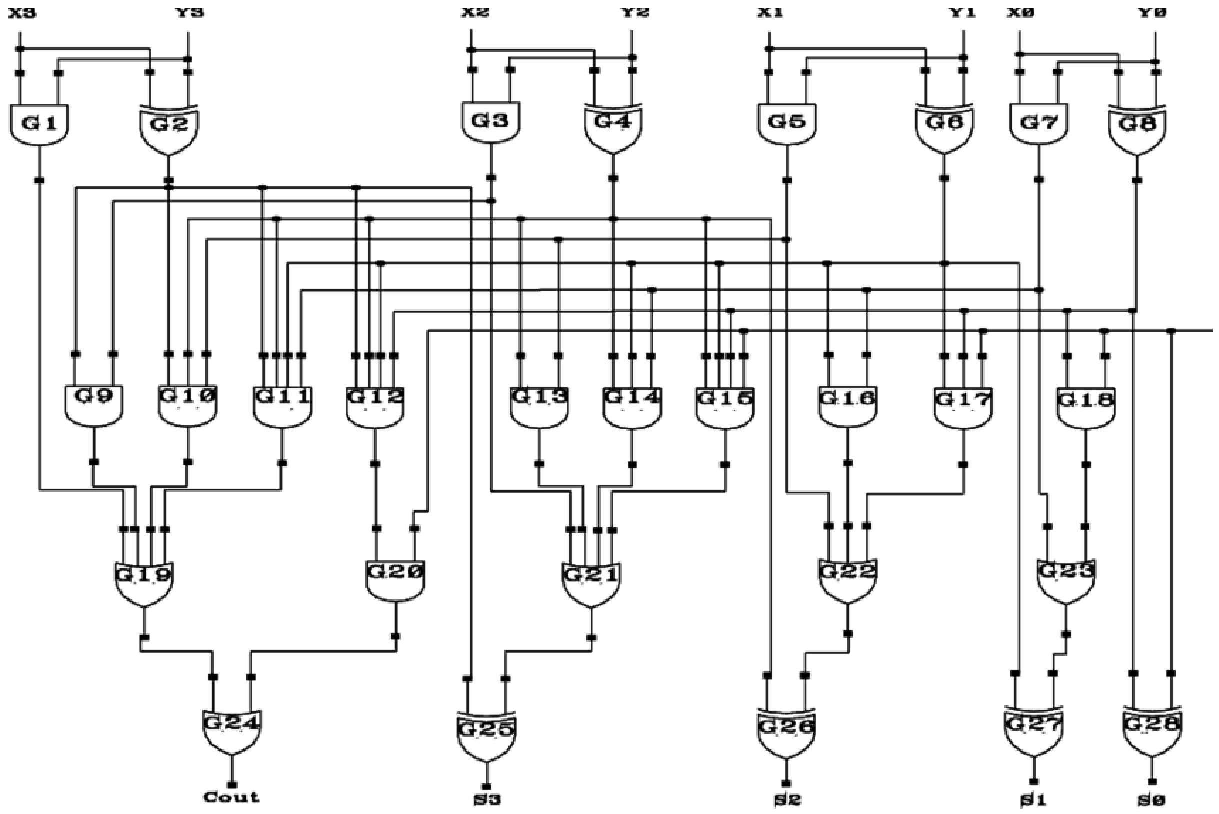
$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

($C_4$ is the carry-out bit)



Circuit Diagram for 4-bit carry look-ahead adder

**(b)** In this section, we have compared the delay of 4-bit Ripple Carry Adder and 4-bit Carry Look-ahead Adder.

**The delay for 4-bit Ripple Carry Adder**: 5.558 ns (1.624 ns logic, 3.934 ns route).
**The delay for 4-bit Carry Look-ahead Adder**: 2.122ns (0.756 ns logic, 1.366 ns route).

As can be seen, the Carry Look-ahead Adder is quicker than the Ripple Carry Adder since it computes the carry bits in advance while the Ripple Carry Adder waits for the sum bit to be computed. As a result, Carry Look-Ahead Adder speeds up computations.

**(c) 16-bit Carry Look-ahead Adder**

**(i)** In this section, we add the two additional output ports, P and G, to the 4-bit CLA that has already been developed. P stands for the block propagated for the following level of hierarchy, while G stands for the block generated for that level.
The P and G can be computed as:
$P = P_3 . P_2 . P_1 . P_0$
$G = G_3 | (P_3 . G_2) | (P_3 . P_2 . G_1) | (P_3 . P_2 . P_1 . G_0)$

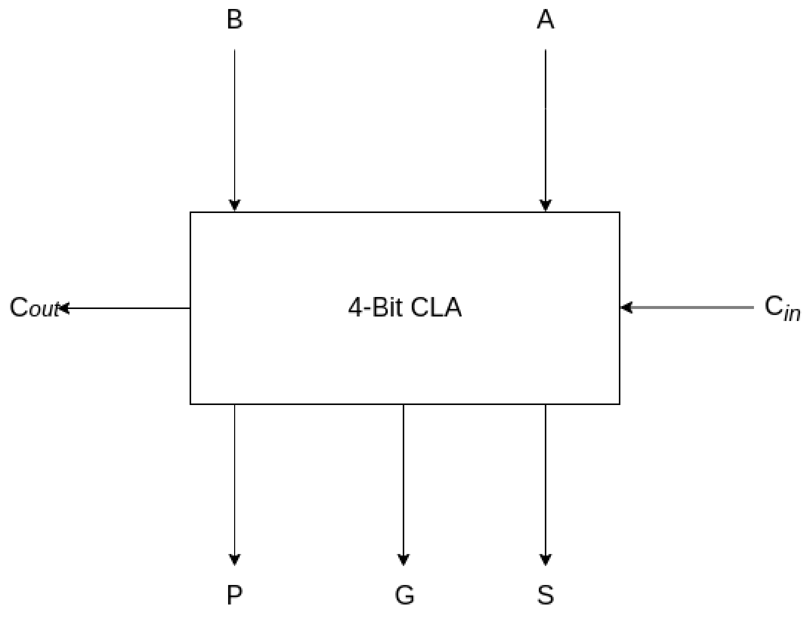


*Figure - 4-bit carry look ahead adder (augmented)*

**(ii)** In this part, we integrate four 4-bit augmented CLAs to create 16-bit Carry Look-ahead Adder as shown in diagram:

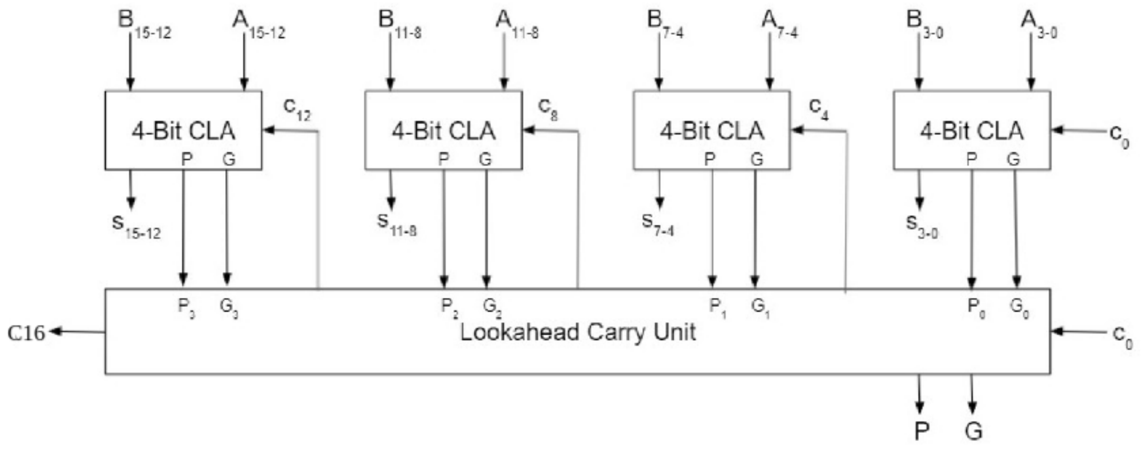


*Figure - 16-bit carry look ahead adder*

**(iii)** In this part, we are required to compare 16-bit Carry Look-ahead Adder designed using Look-ahead Carry unit with that of one when carry is rippled in without using the Look-ahead carry unit.

**Delay for 16-bit CLA using Look-ahead carry unit:** 3.784 ns (1.128 ns logic, 2.656 ns route).
**Delay for 16-bit CLA without using Look-ahead carry unit:** 3.825 ns (1.128 ns logic, 2.697 ns route).

It can be clearly seen that Look-ahead carry unit design computes more quickly than ripple carry design. It is because the Look-ahead carry unit precomputes all the carry bits by taking input all $P_i$ and $G_i$ bits simultaneously from 4 bit-CLA whereas in ripple carry design it waits for the 4 bit CLA unit to compute carry bit. Hence the computation becomes fast in Look-ahead carry design.

**(iv)**

- **Speed Comparison:**

  **Delay for 16-bit CLA using Look-ahead carry unit:** 3.784 ns (1.128 ns logic, 2.656 ns route).
  **Delay for 16-bit RCA:** 18.710 ns (4.600 ns logic, 14.110 ns route)

  The carry bits in 16 bit CLA are precomputed using $P_i$ and $G_i$ inputs from 4 bit CLA hence each 4-bit unit doesn't wait for the input of previous 4 bit CLA unit whereas in 16-bit RCA the current sum and carry bit are computed from the output of previous full adder. Hence the computation in case of 16-bit RCA takes more time than 16-bit CLA.

- **LUT Cost Comparison:**

| **Module** | **Total Delay ( in ns)** | **LUTs used** |
|---|---:|---:|
| CLA 4 bit | 2.122 | 9 |
| CLA 16 bit with Look-ahead unit | 3.784 | 48 |
| CLA 16bit without Look-ahead unit | 3.825 | 34 |
| RCA 4-bit | 5.558 | 20 |
| RCA 16 bit | 18.710 | 80 |