Computer Organization and Architecture Laboratory

Assignment 3

Group 55:-

Kulkarni Pranav Suryakant - 20CS30029
Vineet Amol Pippal - 20CS30058

Question 1

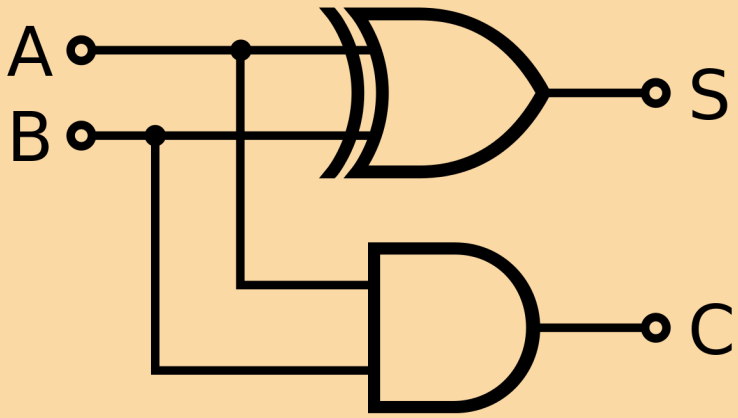
a). Half Adder

A Half Adder takes two bits (a and b) as input and gives the sum bit, s and the carry-out bit, c as output.

Truth Table for Half Adder:

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Circuit Diagram:



- The boolean expressions for Sum and Carry bit can be deduced using the truth table:

Sum = A XOR B
Carry = A AND B

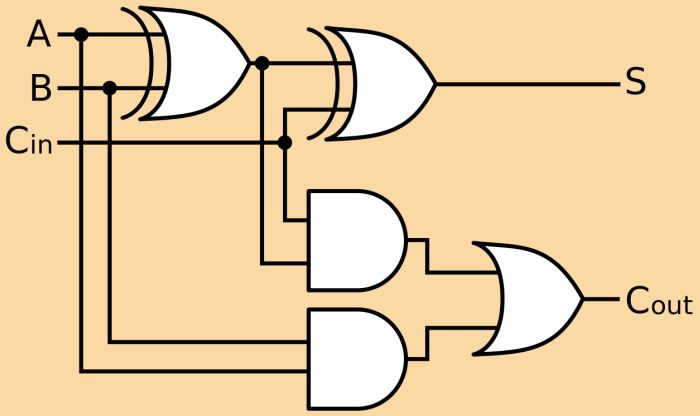
b). Full Adder

A Full Adder takes three bits (a, b and carry-in bit c_0) as input and gives the sum, s and the carry, c as output.

Truth Table for Full Adder:

a	b	c_0	s	c
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

- Circuit Diagram:



- The boolean expressions for Sum and Carry bit can be deduced using the truth table:

Sum = Cin XOR (A XOR B)
Carry = A AND B + Cin AND (A XOR B)

c). Longest Delays in the circuits

1. 64 Bit Full Adder:
   Net Delay: 22.343ns
   Logic Delay: 2.969ns
   Route Delay: 18.374ns

```
                          Gate     Net
Cell:in->out      fanout  Delay    Delay  Logical Name (Net Name)
----------------------------------------  ------------
 IBUF:I->O             2   0.001   0.925  a_1_IBUF (a_1_IBUF)
 LUT5:I0->O            3   0.124   0.550  s1/s1/s1/f2/cout1 (s1/s1/s1/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s1/f4/cout1 (s1/s1/s1/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s1/f6/cout1 (s1/s1/s1/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s1/f8/cout1 (s1/s1/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s2/f2/cout1 (s1/s1/s2/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s2/f4/cout1 (s1/s1/s2/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s2/f6/cout1 (s1/s1/s2/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s1/s1/s2/f8/cout1 (s1/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s1/f2/cout1 (s1/s2/s1/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s1/f4/cout1 (s1/s2/s1/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s1/f6/cout1 (s1/s2/s1/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s1/f8/cout1 (s1/s2/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s2/f2/cout1 (s1/s2/s2/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s2/f4/cout1 (s1/s2/s2/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s2/f6/cout1 (s1/s2/s2/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s1/s2/s2/f8/cout1 (tempCarry)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s1/f2/cout1 (s2/s1/s1/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s1/f4/cout1 (s2/s1/s1/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s1/f6/cout1 (s2/s1/s1/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s1/f8/cout1 (s2/s1/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s2/f2/cout1 (s2/s1/s2/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s2/f4/cout1 (s2/s1/s2/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s2/f6/cout1 (s2/s1/s2/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s2/s1/s2/f8/cout1 (s2/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s1/f2/cout1 (s2/s2/s1/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s1/f4/cout1 (s2/s2/s1/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s1/f6/cout1 (s2/s2/s1/tempCarry<5>)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s1/f8/cout1 (s2/s2/tempCarry)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s2/f2/cout1 (s2/s2/s2/tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s2/f4/cout1 (s2/s2/s2/tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  s2/s2/s2/f6/cout1 (s2/s2/s2/tempCarry<5>)
 LUT5:I3->O            1   0.124   0.399  s2/s2/s2/f8/h2/Mxor_sum_xo<0>1 (sum_63_OBUF)
 OBUF:I->O                 0.000          sum_63_OBUF (sum<63>)
----------------------------------------
Total                     22.343ns (3.969ns logic, 18.374ns route)
                                   (17.8% logic, 82.2% route)
```

2. 32 Bit Full Adder:

Net delay: 11.559ns

Logic Delay: 1.985ns

Route Delay: 9.574ns

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	2	0.001	0.925	a_1_IBUF (a_1_IBUF)
LUT5:I0->O	3	0.124	0.550	s1/s1/f2/cout1 (s1/s1/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s1/s1/f4/cout1 (s1/s1/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s1/s1/f6/cout1 (s1/s1/tempCarry<5>)
LUT5:I3->O	3	0.124	0.550	s1/s1/f8/cout1 (s1/tempCarry)
LUT5:I3->O	3	0.124	0.550	s1/s2/f2/cout1 (s1/s2/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s1/s2/f4/cout1 (s1/s2/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s1/s2/f6/cout1 (s1/s2/tempCarry<5>)
LUT5:I3->O	3	0.124	0.550	s1/s2/f8/cout1 (tempCarry)
LUT5:I3->O	3	0.124	0.550	s2/s1/f2/cout1 (s2/s1/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s2/s1/f4/cout1 (s2/s1/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s2/s1/f6/cout1 (s2/s1/tempCarry<5>)
LUT5:I3->O	3	0.124	0.550	s2/s1/f8/cout1 (s2/tempCarry)
LUT5:I3->O	3	0.124	0.550	s2/s2/f2/cout1 (s2/s2/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s2/s2/f4/cout1 (s2/s2/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s2/s2/f6/cout1 (s2/s2/tempCarry<5>)
LUT5:I3->O	1	0.124	0.399	s2/s2/f8/h2/Mxor_sum_xo<0>1 (sum_31_OBUF)
OBUF:I->O		0.000		sum_31_OBUF (sum<31>)
Total		11.559ns	(1.985ns logic, 9.574ns route)	
			(17.2% logic, 82.8% route)	

3. 16 Bit Full Adder:

Net Delay: 6.167ns

Logic Delay: 0.993ns

Route Delay: 5.174ns

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	2	0.001	0.925	a_1_IBUF (a_1_IBUF)
LUT5:I0->O	3	0.124	0.550	s1/f2/cout1 (s1/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s1/f4/cout1 (s1/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s1/f6/cout1 (s1/tempCarry<5>)
LUT5:I3->O	3	0.124	0.550	s1/f8/cout1 (tempCarry)
LUT5:I3->O	3	0.124	0.550	s2/f2/cout1 (s2/tempCarry<1>)
LUT5:I3->O	3	0.124	0.550	s2/f4/cout1 (s2/tempCarry<3>)
LUT5:I3->O	3	0.124	0.550	s2/f6/cout1 (s2/tempCarry<5>)
LUT5:I3->O	1	0.124	0.399	s2/f8/h2/Mxor_sum_xo<0>1 (sum_15_OBUF)
OBUF:I->O		0.000		sum_15_OBUF (sum<15>)
Total		6.167ns	(0.993ns logic, 5.174ns route)	
			(16.1% logic, 83.9% route)	

4. 8 Bit Full Adder:

   Net Delay: 3.471ns

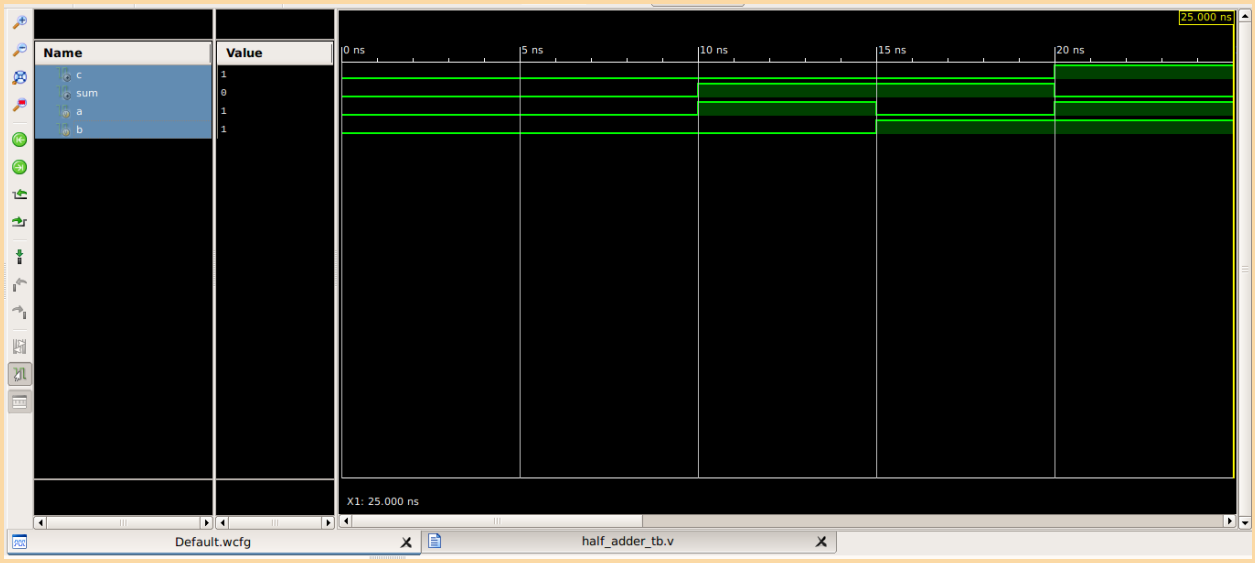
   Logic Delay: 0.497ns

   Route Delay: 2.974ns

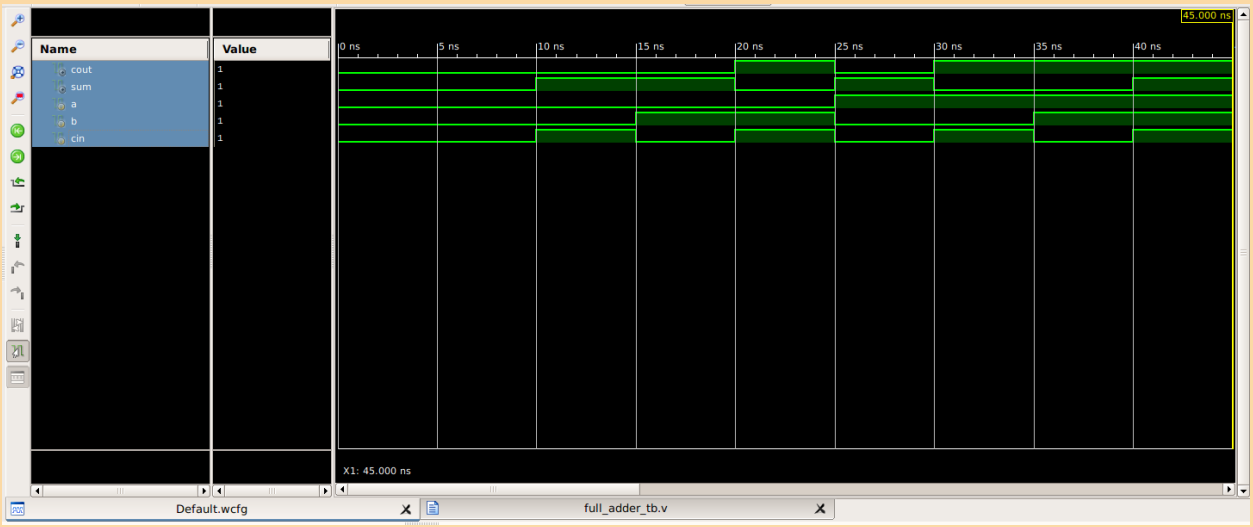
```
                              Gate     Net
Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
----------------------------------------  ------------
 IBUF:I->O             2   0.001   0.925  a_1_IBUF (a_1_IBUF)
 LUT5:I0->O            3   0.124   0.550  f2/cout1 (tempCarry<1>)
 LUT5:I3->O            3   0.124   0.550  f4/cout1 (tempCarry<3>)
 LUT5:I3->O            3   0.124   0.550  f6/cout1 (tempCarry<5>)
 LUT5:I3->O            1   0.124   0.399  f8/cout1 (cout_OBUF)
 OBUF:I->O                 0.000          cout_OBUF (cout)
----------------------------------------
Total                      3.471ns (0.497ns logic, 2.974ns route)
                                   (14.3% logic, 85.7% route)
```

Wave Outputs:

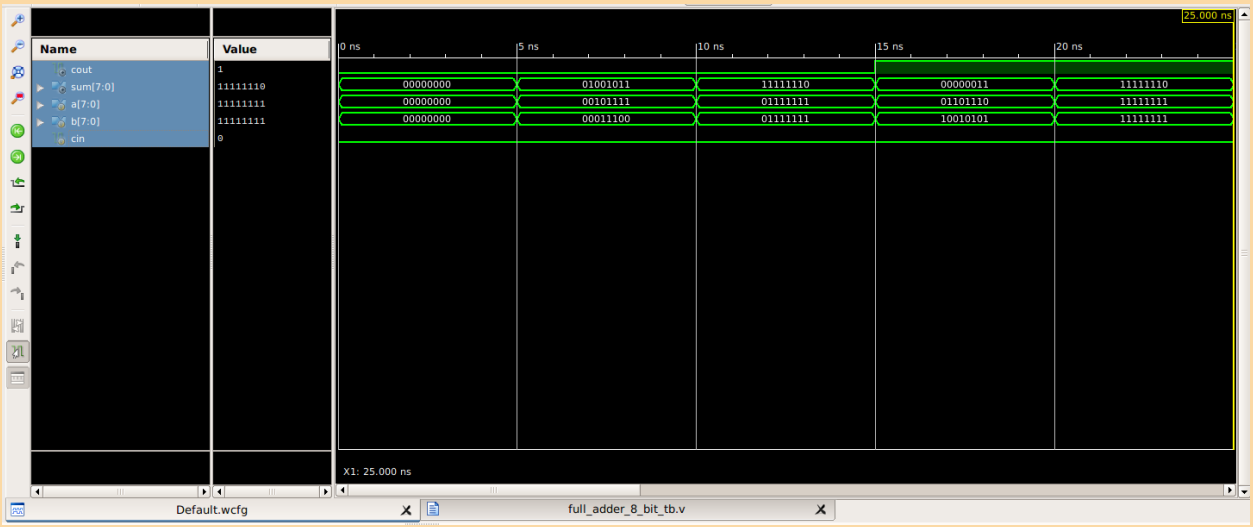
1. Half Adder:



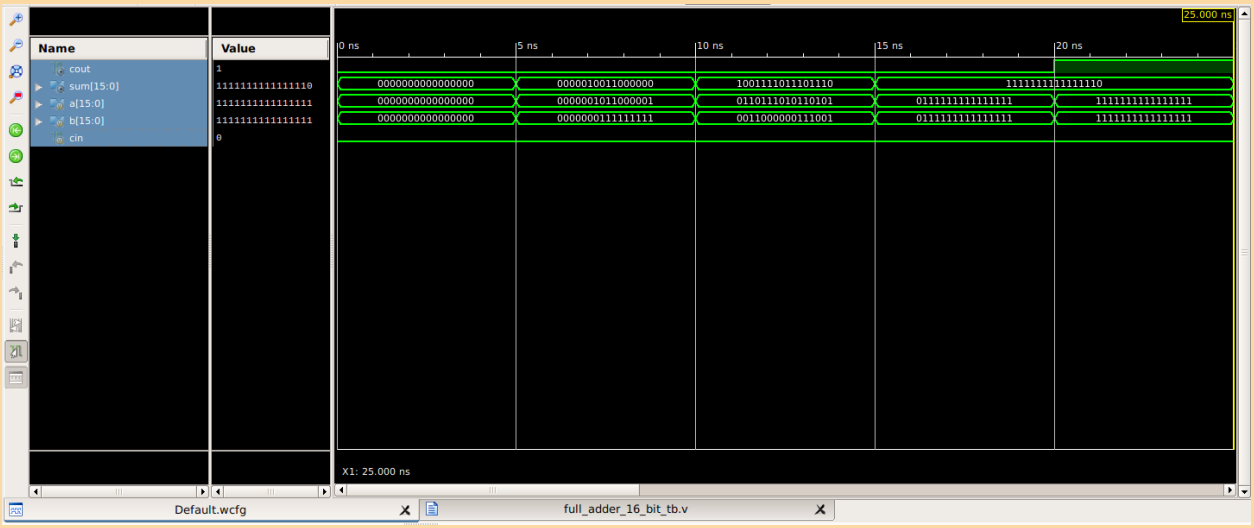
2. Full Adder:



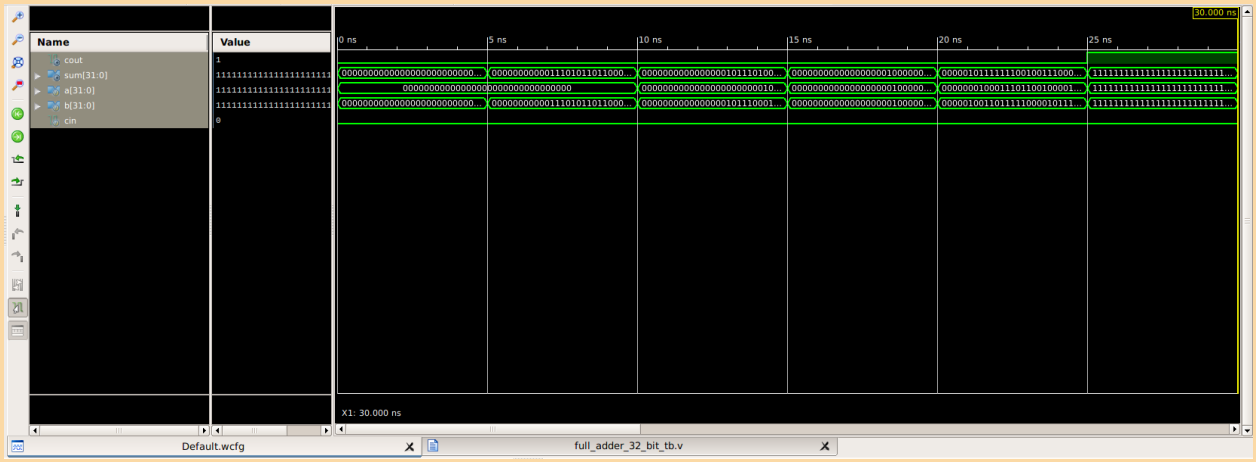
3. 8 bit Full Adder:



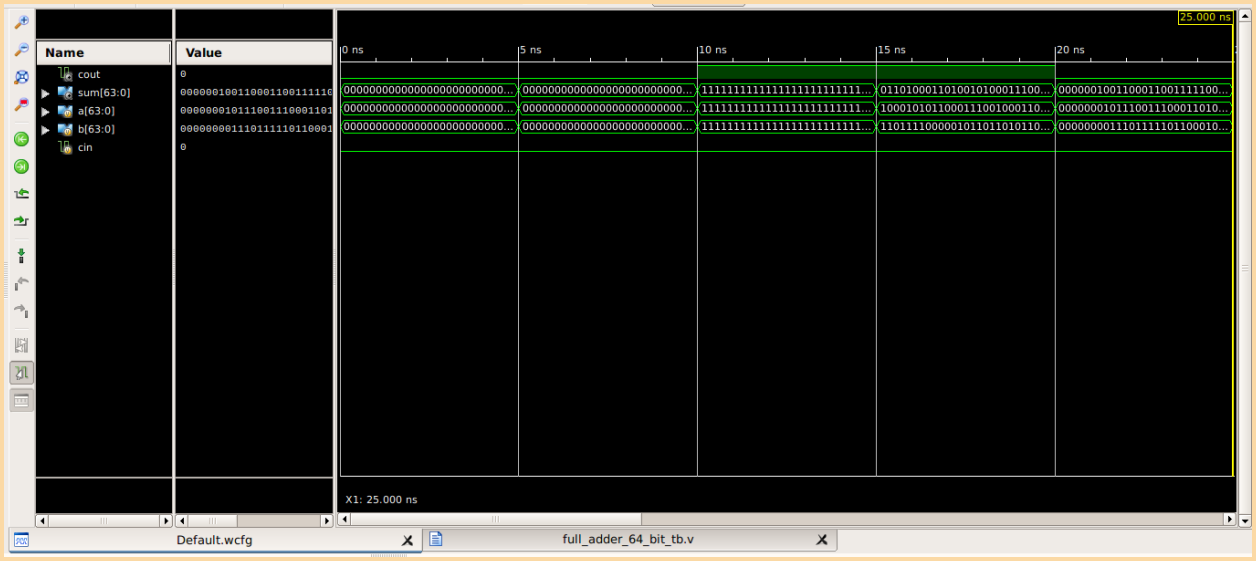
4. 16 bit Full Adder:



5. 32 bit Full Adder:



6. 64 bit Full Adder:



d). Difference between two n-bit numbers:

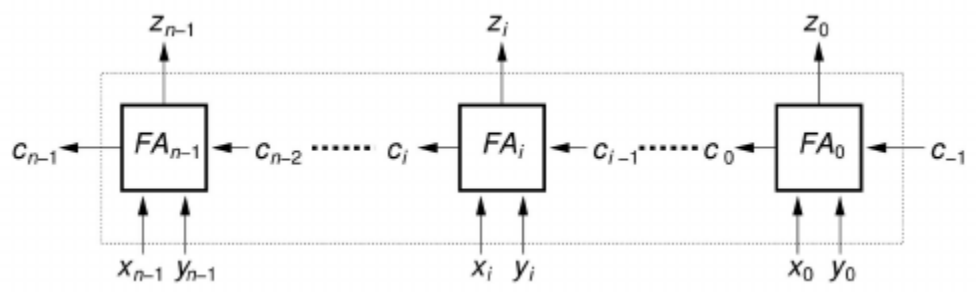
To calculate the difference between two numbers `a` and `b`, we can note that,

`a - b = a + (-b)`

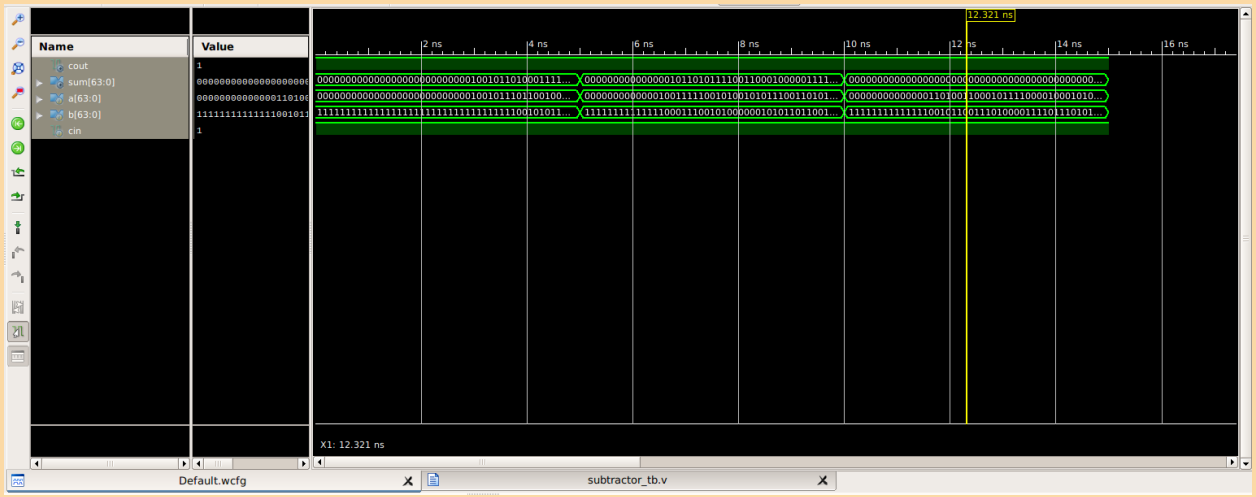
Therefore, to perform a subtraction operation, we can add `a` with the 2's complement of `b`:

⇒ a + (2's complement of b)
= a + (1's complement of b) + 1

Therefore, a ripple carry adder can take `a`, `2's complement of b` as number inputs and `1 as the input carry` to calculate `a - b` i.e. the difference between two n-bit numbers.



Difference between two n-bit numbers using a ripple carry adder.




```
a =          81277482720, b =          446278127, carry_in = 1, carry_out = 1, a-b =          80831204593
a =  1400000000000000, b =  1000000000000000, carry_in = 1, carry_out = 1, a-b =  400000000000000
a =       928391830841483, b =       928391830841483, carry_in = 1, carry_out = 1, a-b =                0
```

Simulated output

Question 2

a) 4-bit Carry Look-Ahead Adder (CLA):

For a carry look-ahead adder, we define two variables as Carry Generate (G) and Carry Propagate (P) where,

$$G_i = A_i \cdot B_i$$
$$P_i = A_i \oplus B_i$$

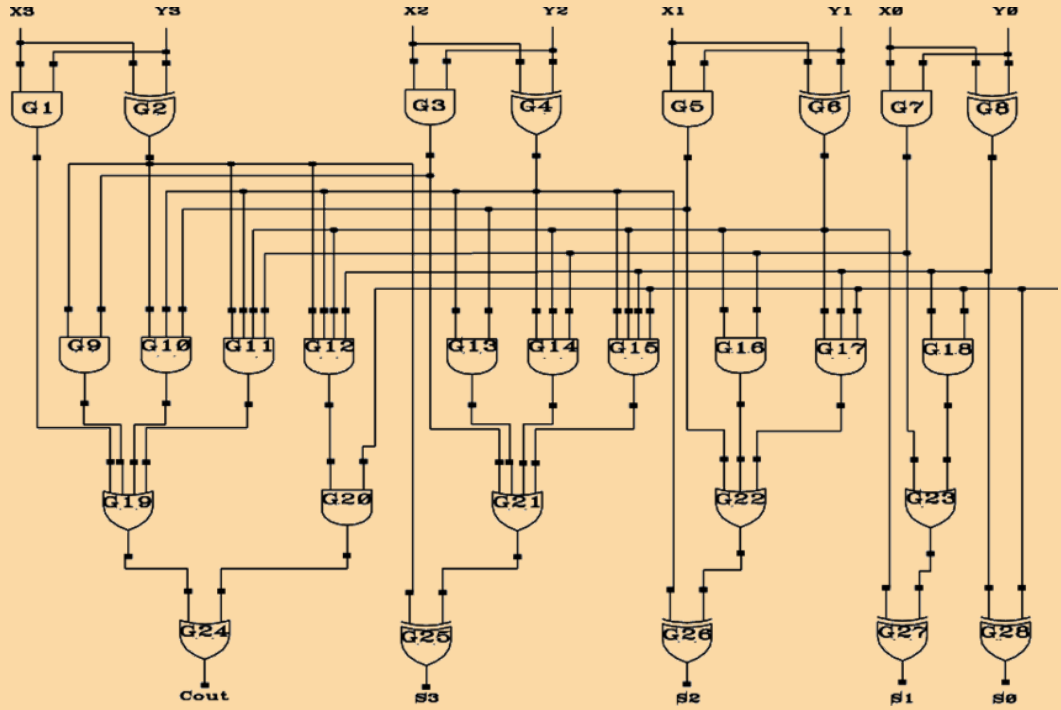
For the ith stage, G_i represents the condition of generation of a carry independent of other states and P_i represents the condition of an input carry C_i being propagated to the output carry C_{i+1}.
Therefore,

$$C_{i+1} = G_i + P_iC_i$$

Taking C_0 as the initial carry bit, we have:

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$
$$C_4 = \text{Carry\_out} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

Henceforth, we can design a 4-bit CLA, taking two 4-bit numbers and a carry as input and giving a 4-bit sum and a carry as output.



b) Comparing delays between 4-bit CLA and 4-bit RCA:

Here, we compare the delays of a 4-bit Ripple Carry Adder and a 4-bit Carry Look-Ahead Adder after setting the **KEEP_HIERARCHY** option in the synthesizer to **TRUE**.

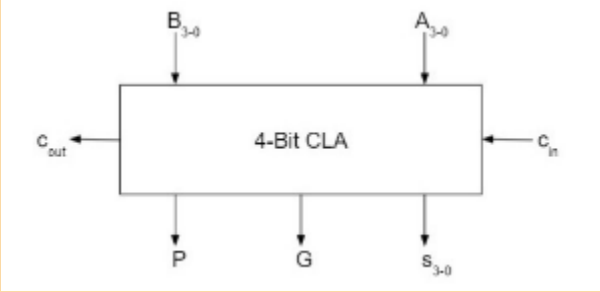
- **Delay for 4-bit RCA**: **5.565ns**
  For an RCA, the carry bits travel from the first full adder to the last full adder, followed by computation of the output carry in the last full adder. For sum, the carry bits travel from the first full adder to the last, followed by computation of the last bit of sum in the last full adder.
- **Delay for 4-bit CLA: 2.123ns**
  For a CLA, the critical path for the carry bit is given by the path taken by the Propagate and Generate bits in arriving at the 4 leftmost AND gates and their result going to the OR gate. The critical path for the sum is given by the path taken by Pi and Gi bits to arrive at the 3 AND gates (before the OR gate which gives us C3) and their result going to the OR gate, followed by C3 reaching the next OR gate (along with P3).

c) 16-bit Carry Look-ahead Adder:

i) 4-bit Carry Look-ahead Adder (Augmented):

For a carry look-ahead adder (augmented), the block propagate and generate are given as output for the carry look-ahead unit to use, instead of generating an output carry. The same design can hence be combined to construct 16, 32, 64 bit adders, etc. The block propagate and generate from lower levels can be used instead of a rippling carry:

```
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])
```



4-bit Carry Look-ahead Adder (Augmented)

ii) Design and integration of Lookahead Carry Unit:

To make a 16-bit CLA, we cascade four 4-bit CLAs, but instead of rippling the carry output from one block to the other, delay in circuit is minimized by a look-ahead unit. The look-ahead unit calculates these carries simultaneously and so there is no delay in waiting for the carry from previous blocks. Moreover, the same design could be used for building higher order adders.

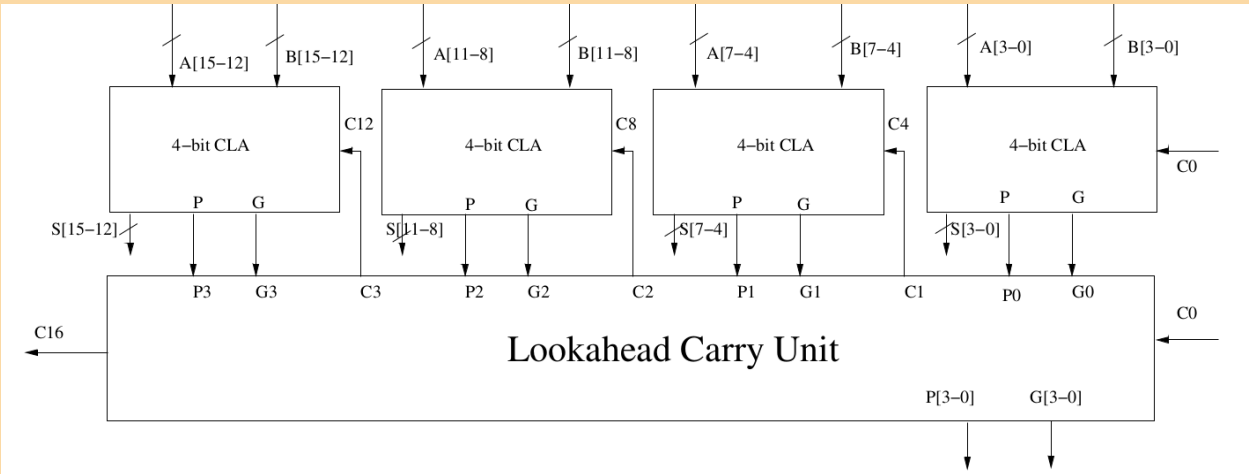
$$carry_i = G_{i-1} \mid P_{i-1} \,\&\, carry_{i-1} \;,\; 1 \le i \le 4$$

Expanding recursively we get,

$$carry_1 = G_0 \mid (P_0 \,\&\, carry_0) = G_0 \mid (P_0 \,\&\, cin)$$
$$carry_2 = G_1 \mid (P_0 \,\&\, G_0) \mid (P_1 \,\&\, P_0 \,\&\, cin)$$
$$carry_3 = G_2 \mid (P_2 \,\&\, G_1) \mid (P_2 \,\&\, P_1 \,\&\, G_0) \mid (P_2 \,\&\, P_1 \,\&\, P_0 \,\&\, cin)$$
$$carry_4 = G_3 \mid (P_3 \,\&\, G_2) \mid (P_3 \,\&\, P_2 \,\&\, G_1) \mid (P_3 \,\&\, P_2 \,\&\, P_1 \,\&\, P_0 \,\&\, cin)$$

For propagate and generate:

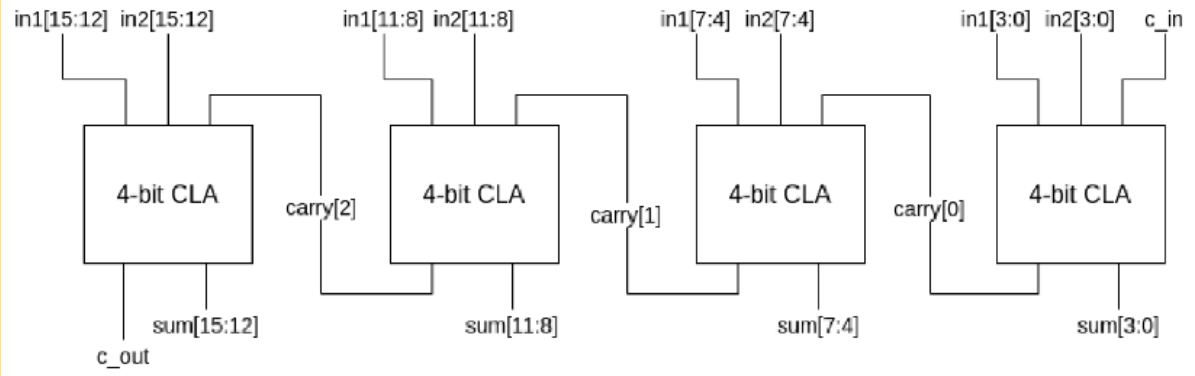
```
p = P[3] & P[2] & P[1] & P[0]
g = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1]) | (P[3] & P[2] & P[1] & G[0])
```



16-bit Carry Look-ahead Adder (with LCU)

iii) 16-bit Carry Look-Ahead Adder (rippling carry):

To make a 16-bit CLA, we cascade four 4-bit CLAs by rippling the output carry from one block to another as shown below (`carry[2:0]` represents internal carries being rippled from one block to another):



16-bit Carry Look-ahead Adder (with rippling carry)

Comparison of delays for 16-bit CLA with LCU and 16-bit CLA with

Delay for 16-bit CLA using the look-ahead carry unit: 5.446ns

Four CLAs provide P_i and C_i. Cout is computed by:

$$\mathbf{cout = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0}$$

and so the gates used to compute cout comprise the critical path. CLAs also give the sum through the same path as cout. Overall, the additional LCU unit reduces the delay.

Delay for 16-bit CLA using ripple carry: 6.167ns

The carry bits C_4, C_8, C_{12} ripple through all 4-bit CLAs to reach the last one, where further computation is done to get cout. Sum also takes the same path.

<u>iv) Speed and LUT comparison between 16-bit CLA (with LCU) and 16-bit RCA</u>

Delay for 16-bit CLA using the look-ahead carry unit: 5.446ns

As described previously, P_i and C_i are provided by the four CLAs simultaneously. Cout is computed by:

$$\mathbf{cout = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0}$$

Therefore, the gates used to compute cout comprise the critical path. CLAs also give the sum through the same path as cout.

Delay for 16-bit RCA: 18.717ns

For carry, the critical path comprises the path taken by the carry bits to travel from the first to the last full adder, and to compute output carry in the last full adder thereafter. Similarly, the path taken by the carry bits to travel from the first to the last full adder and the computation of sum comprise the critical path for sum.

LUT Cost Comparison:

Slice LUTs utilized for 16-bit LCA with look-ahead carry unit: 43 out of 63400

Slice LUTs utilized for 16-bit RCA: 80 out of 63400

Module	Delay Time	LUT Cost
16 Bit LCA	5.446ns 0.745ns (logic) 4.701ns (route)	43 = 10 (LUT3) + 9 (LUT4) + 24 (LUT5)
16 Bit RCA	18.717ns 4.093ns (logic) 14.624ns (route)	80 = 80 (LUT2)

Comparison Summary