Computer Organisation and Architecture Lab

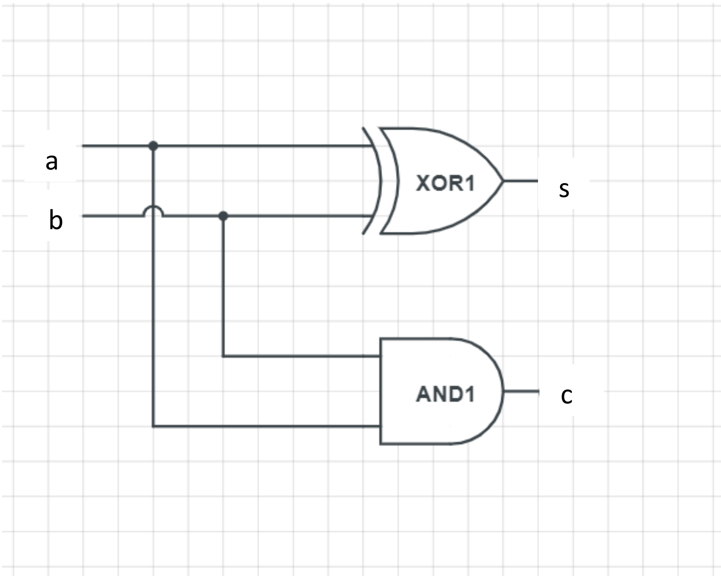
Group number: 65

Members: Pranav Mehrotra (20CS10085)

Saransh Sharma (20CS30065)

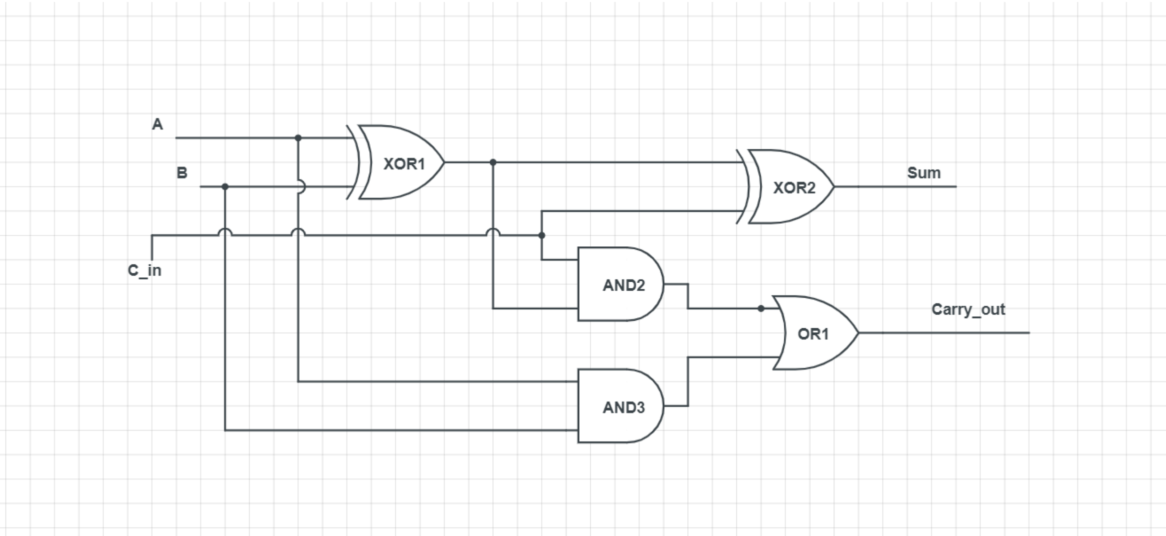
Q1) RCA using Verilog

i.) Half Adder



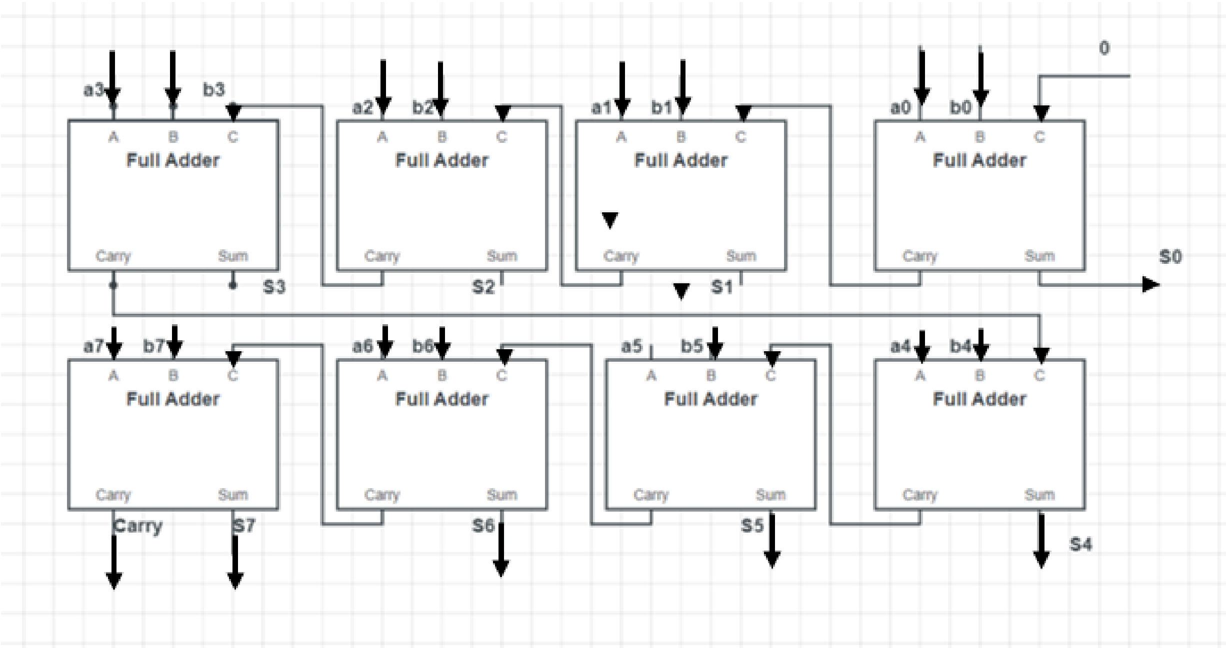
Truth Table of a Half Adder			
Input		Output	
A	b	s (Sum)	c (carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

ii.) Full Adder

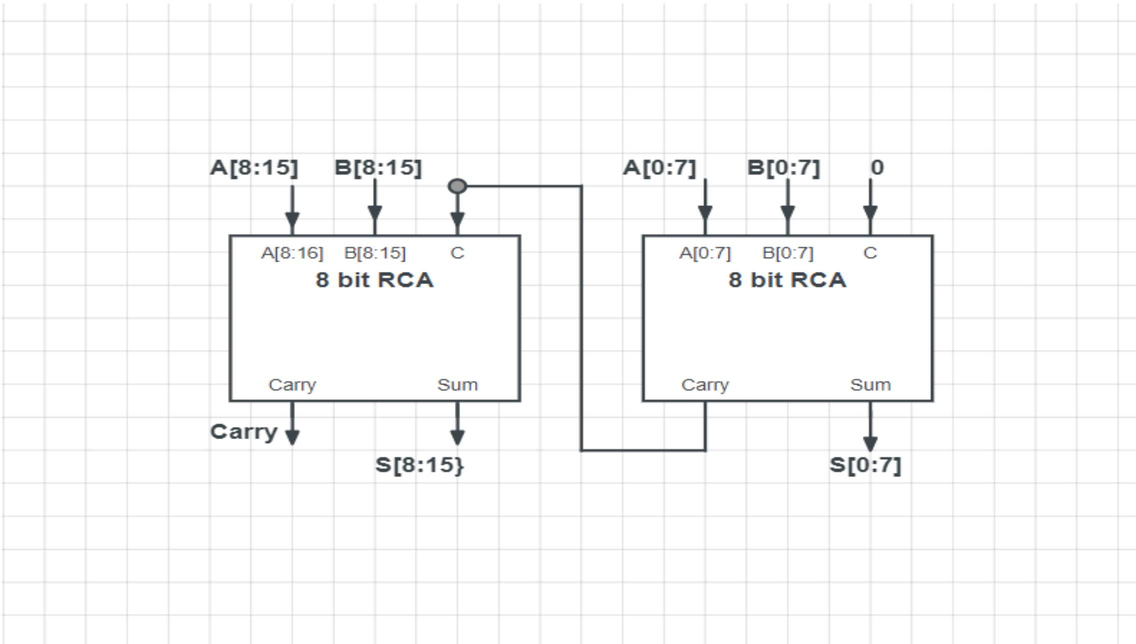


Truth Table of a Full Adder				
Input			Output	
A	B	C_in	Sum	Carry_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

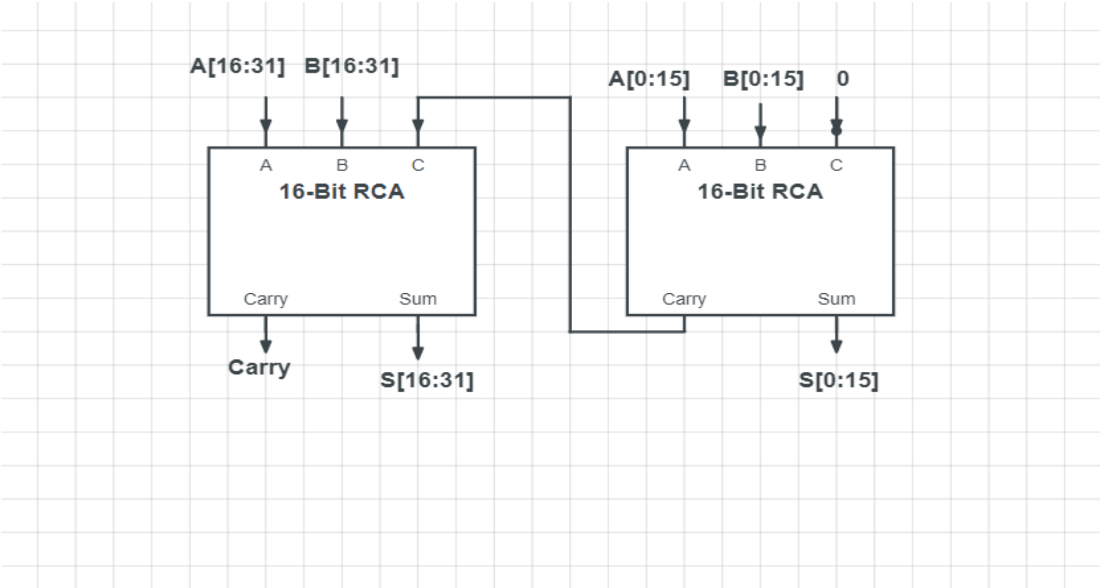
iii.) 8 – Bit Ripple Carry Adder



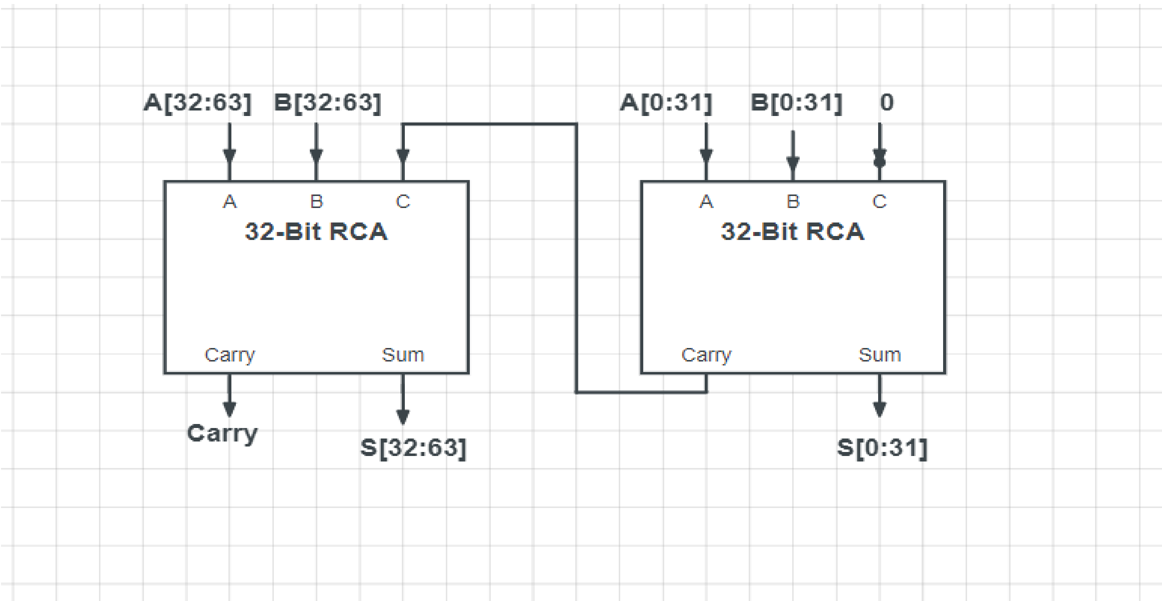
iv.) 16- Bit RCA



v.) 32-Bit RCA



vi.) 64- Bit RCA



Question: How can you use the above circuit, to compute the difference between two n-bit numbers?

Answer: Ripple Carry adder with input a and b and initial carry bit c_0 actually computes the binary sum of a and b, also adding c_0 to the LSB of the sum. Thus, c_0 is set to 0 so that RCA adds a and b bitwise and output the result.

For finding difference between two number a and b i.e., a-b we will convert the subtraction problem into addition problem:

i.e., a – b = a + (-b) where (-b) is the 2's complement of b.

we know that 2's complement of any number can be calculated by computing 1's complement of that number and then adding 1 to the LSB of the 1's complement. That is, (-b) = (~b) + 1 where (~b) is the 1's complement of b

let a = 100 (4) and b = 010 (2) and n=3

therefore, a – b = 100 – 010

-b = 101 (= ~b) + 1 = 110

a – b = 100 + (110) = 010 (carry_out = 1)

To calculate 1's complement of a number, we simply have to flip all the bits of that number. That is, we pass every bit of b to a NOT gate to flip the bits of the b. Therefore, whenever we need to calculate difference between a and b i.e., a-b we would first calculate 1's compliment of b by taking NOT of every bit of b with 1. Since we also need to add 1 to the 1's complement calculated, we would pass this 1 as c_0 to the RCA. Therefore, upon passing a, (~b) and 1 as input to RCA, the n-bit RCA can calculate the difference between a and b.

Synthesis Report

Circuit	Delay (in ns)	Logic Levels	Number of Slice LUTs	Number of bonded IOBs
Half Adder	1.066	3	2	4
Full Adder	1.246	3	2	5
8-Bit RCA	3.471	6	12	26
16-Bit RCA	6.167	10	24	50
32-Bit RCA	11.559	18	48	98
64- Bit RCA	22.343	34	96	194

Part-2 Carry Look- Ahead Adders

1. 4- Bit Carry Look-Ahead Adder

The prominent difference between a 4-Bit RCA and a 4- Bit CLA is that a CLA calculates all carries simultaneously instead of waiting for the carry from the previous block to be passed to the current block, thereby, reducing the delay in calculation of sum. C_i that is carry when ith bit of inputs is added can be represented in the form of C_0 (initial carry bit).

The logic for CLA for 2 4-Bit input X and Y is shown as:

G[i] = X[i] & Y[i]

P[i] = X[i] $\oplus$ Y[i]

Where G_i denotes whether X[i] and Y[i] upon addition generate a carry on their own and P_i denotes whether the carry from previous block will be propagated forward or not.

Let C_0 be the initial input carry bit.

Sum[i] = P[i] $\oplus$ C[i]

C[i+1] = G[i] + P[i]C[i]

(Notation: A & B ➔ AB and A or B ➔ A + B)

Upon further expanding

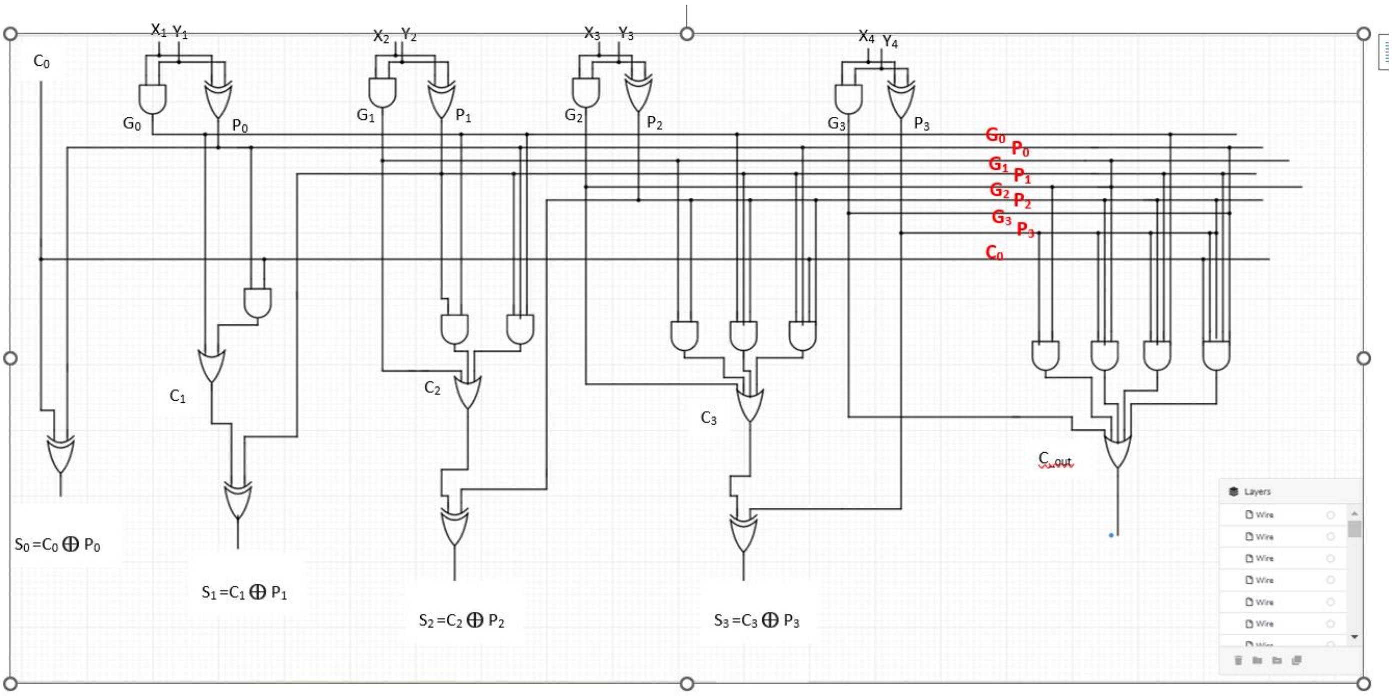
C[1] = G[0] + P[0]C[0]

C[2] = G[1] + P[1]G[0] + P[1]P[0]C[0]

C[3] = G[2] + P[2]G[1] + P[2]P[1]G[0] + P[2]P[1]P[0]C[0]

C[4] = G[3] + P[3]G[2] + P[3]P[2]G[1] + P[3]P[2]P[1]G[0] + P[3]P[2]P[1]P[0]C[0]

That is all C[1], C[2], C[3], C[4] can now be represented in terms of C[0].



2. 4-Bit Carry Look Ahead Adder (augmented)

   The previous circuit will be modified so that the block outputs propagate P and generate G instead of carry-out. This P and G will help us in constructing 16-Bit and higher order CLA using 4-Bit CLA.

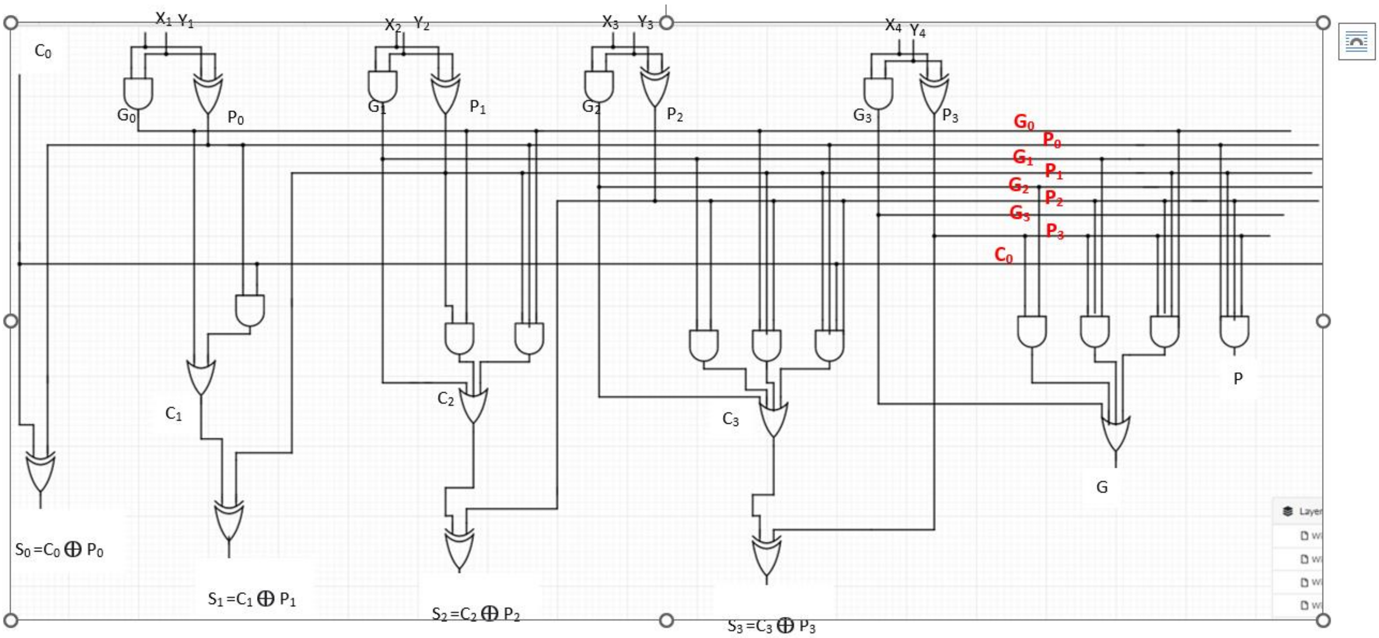
   The expressions of C[i] and Sum[i] remains the same, i.e.,

   $Sum[i] = P[i] \oplus C[i]$

   $C[i+1] = G[i] + P[i]C[i]$

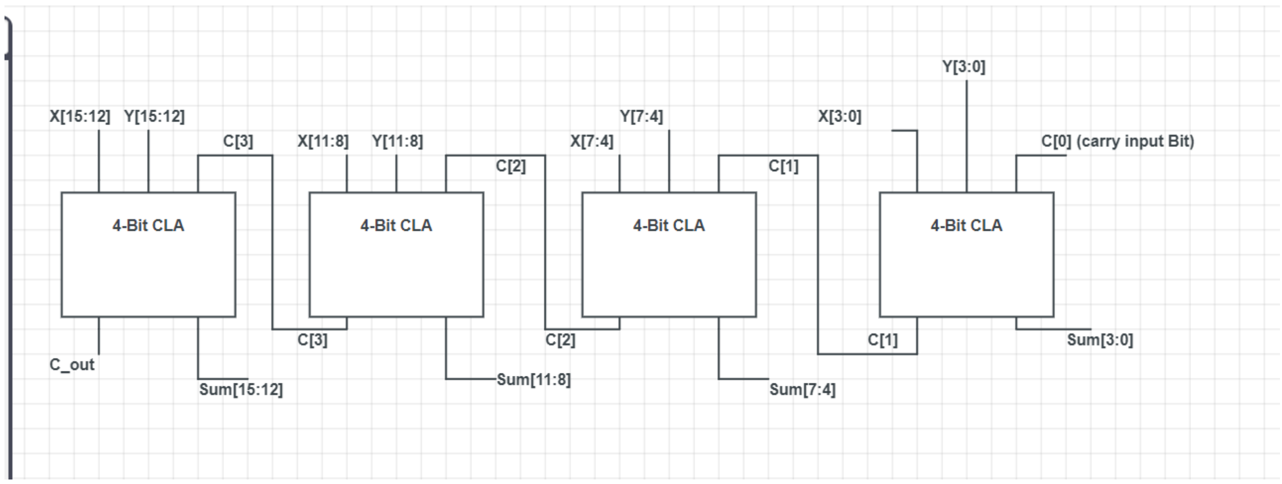
   $P = P[3]P[2]P[1]P[0]$

   $G = G[3] + P[3]G[2] + P[3]P[2]G[1] + P[3]P[2]P[1]G[0] + P[3]P[2]P[1]P[0]C[0]$



3. 16-Bit Carry Look-Ahead Adder

   16-Bit CLA can be constructed by connecting 4 4-Bit CLA and rippling the carry of one block to the next block. C[3], C[2] and C[1] are the internal carries of the circuit.



4. 16- Bit Carry Look- Ahead Adder (Look- Ahead Carry Unit)

   The previous circuit has to wait for carry from previous block to do further calculations. We can reduce this delay by using an additional look-ahead unit along with 4 CLA's which can calculate carries using propagate and generate of the 4 CLA's. let P[0],P[1],P[2],P[3] be the propagates of the 4 blocks of CLA and G[0],G[1],G[2],G[3] be the generates of the 4 blocks of CLA. Using this 16-Bit CLA, delay is further reduced and this circuit can be then used for constructing higher order CLA's.

Equations involved:

C_0 is the input carry

$Sum[i] = P[i] \oplus C[i]$

$C[i+1] = G[i] + P[i]C[i]$

$C[1] = G[0] + P[0]C[0]$

$C[2] = G[1] + P[1]G[0] + P[1]P[0]C[0]$

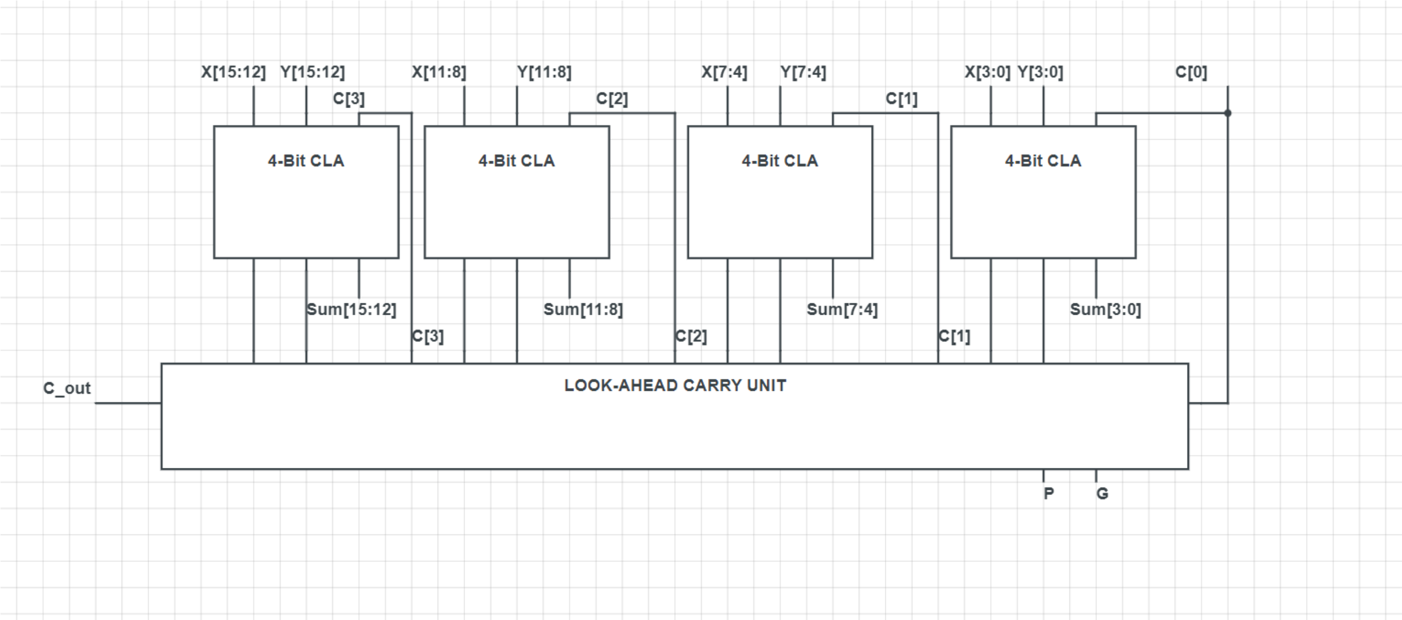
$C[3] = G[2] + P[2]G[1] + P[2]P[1]G[0] + P[2]P[1]P[0]C[0]$

$C[4] = G[3] + P[3]G[2] + P[3]P[2]G[1] + P[3]P[2]P[1]G[0] + P[3]P[2]P[1]P[0]C[0]$

Also propagate and generate of this circuit would be:

$P = P[3]P[2]P[1]P[0]$

$G = G[3] + P[3]G[2] + P[3]P[2]G[1] + P[3]P[2]P[1]G[0] + P[3]P[2]P[1]P[0]C[0]$



Synthesis Report

Component	Logic Delay	Route Delay	Total Delay (in ns)	Number of Slice LUTs	Logic Levels	Number of bonded IOBs
4-Bit RCA	0.497	2.697	3.194	8	10	14
4-Bit CLA	0.249	1.874	2.123	6	4	14
4-Bit CLA (augmented)	0.249	2.114	2.363	9	4	15
Look Ahead Carry Unit	0.373	2.623	2.996	7	5	15
16-Bit RCA	0.993	5.174	6.167	24	10	50
16-Bit CLA (with LCU)	0.745	4.563	5.30	43	11	52
16-Bit CLA (Ripple)	0.993	5.174	6.167	24	14	50

Component	Worst Case Slack	Best Case Achievable
16- Bit CLA with LCU	0.426 ns	4.844 ns
16-Bit CLA with Ripple	0.271 ns	4.229 ns