**Computer Organization and Architecture Laboratory**

**Lab–1**

Group 48

Vinit Raj - 19CS10065

Aryan Agarwal - 19CS30005

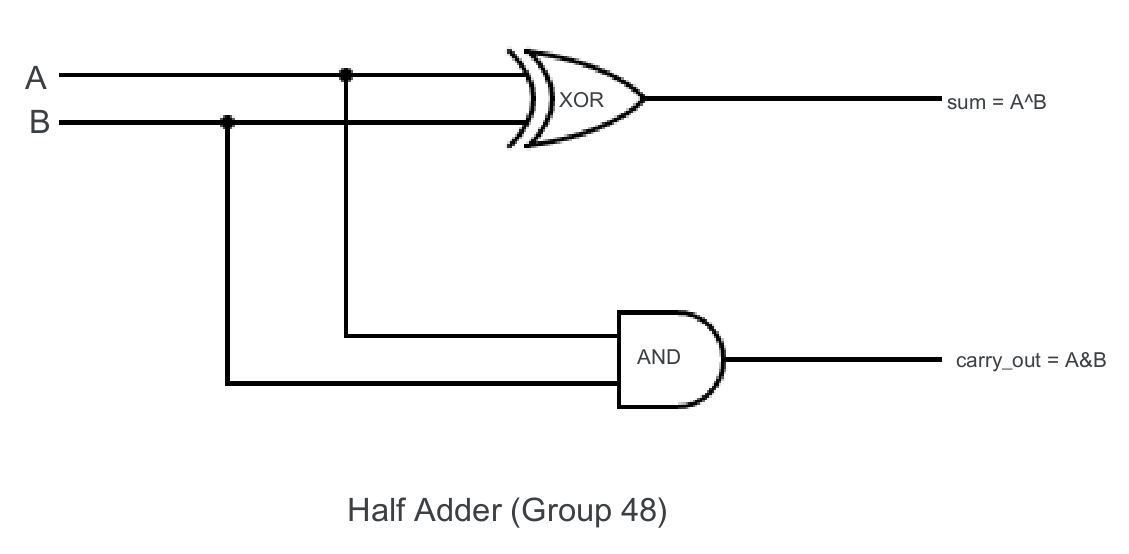
Q1)

a) Half Adder

Truth Table

| Input | | Output | |
| --- | --- | --- | --- |
| A | B | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Circuit Diagram

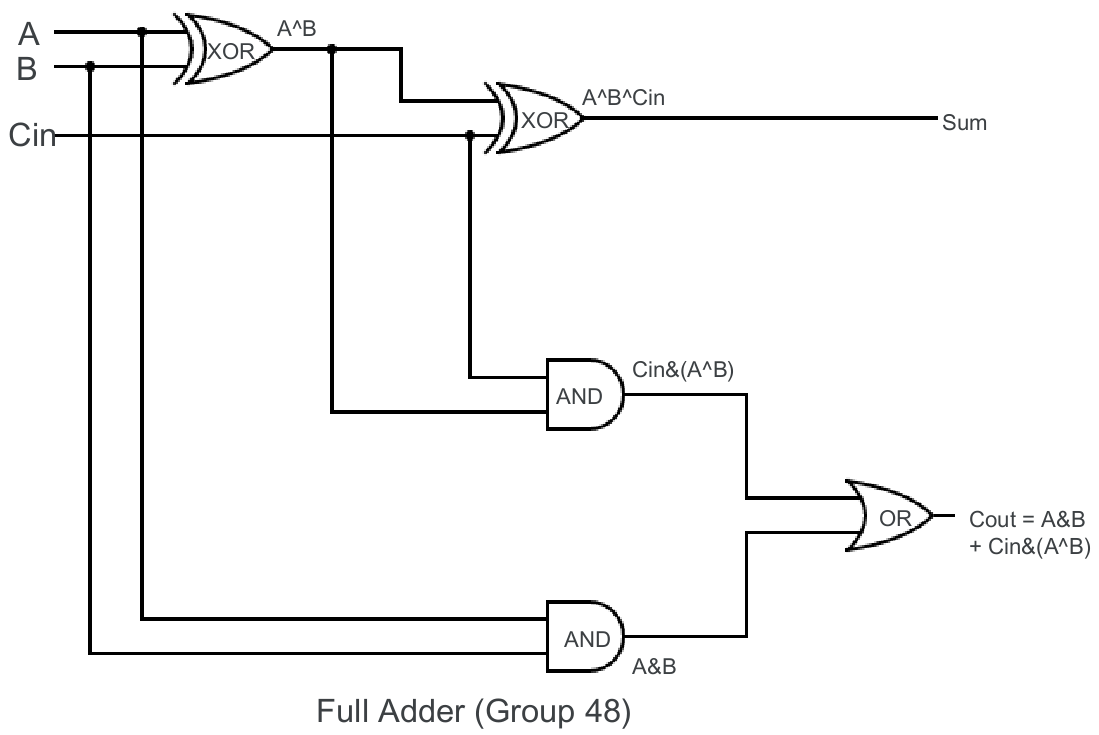


b) Full Adder

Truth Table

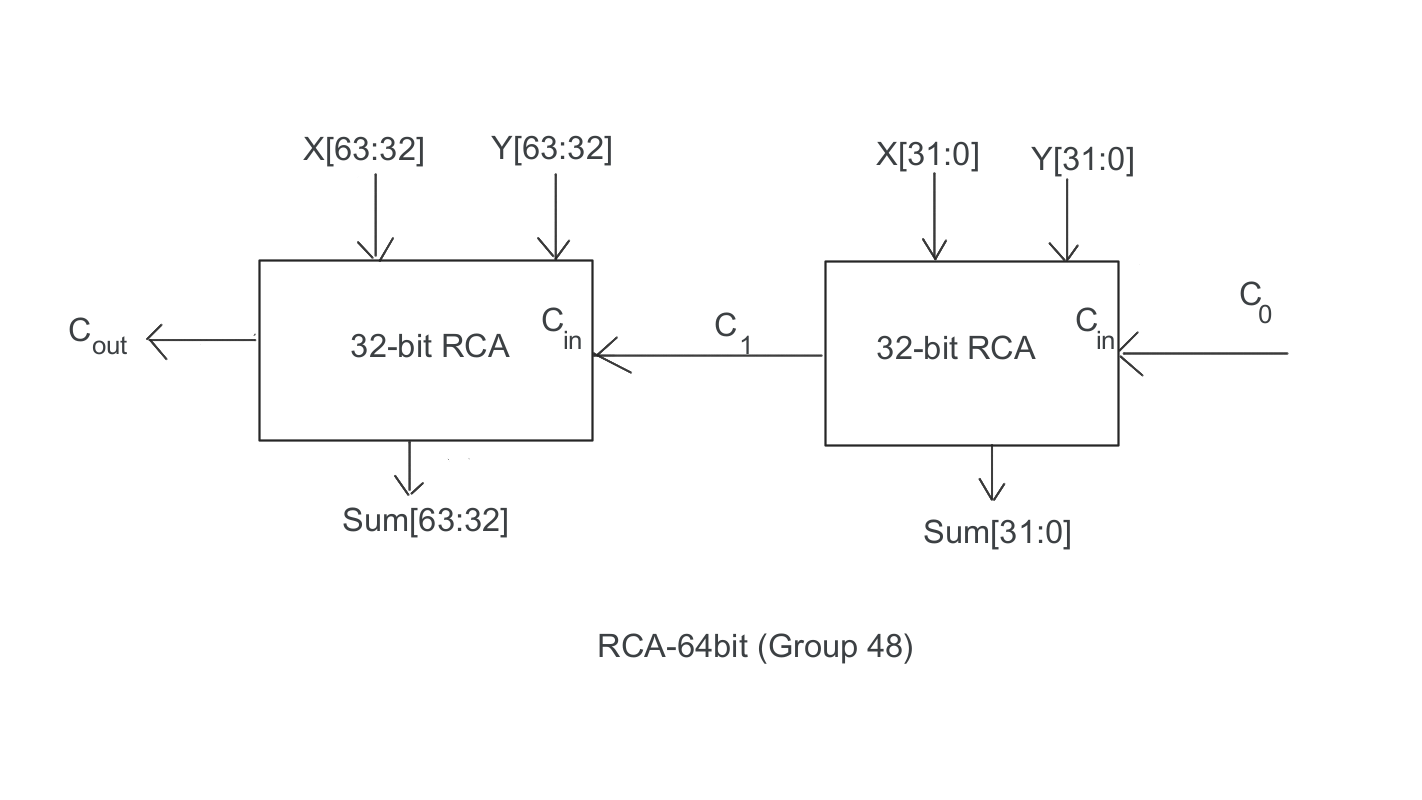
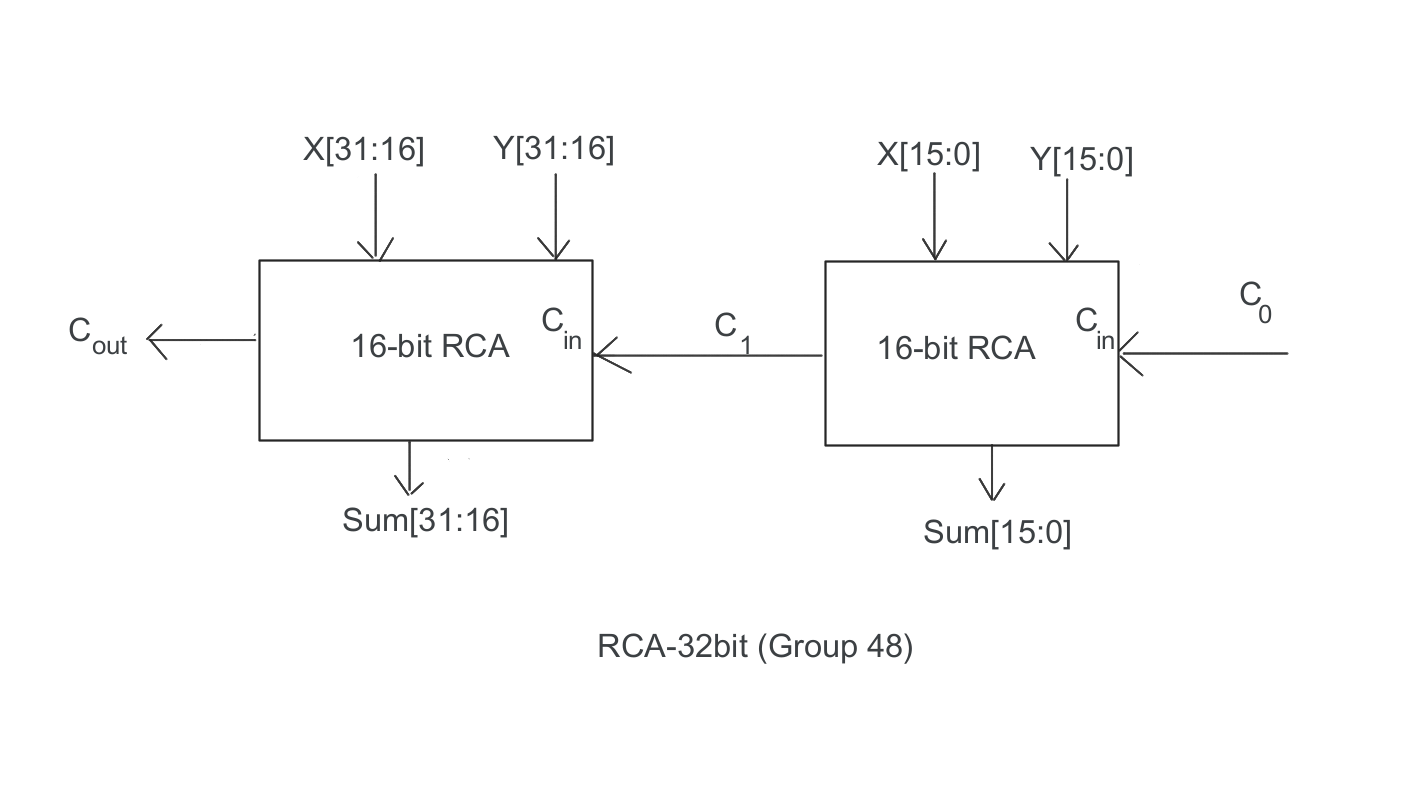
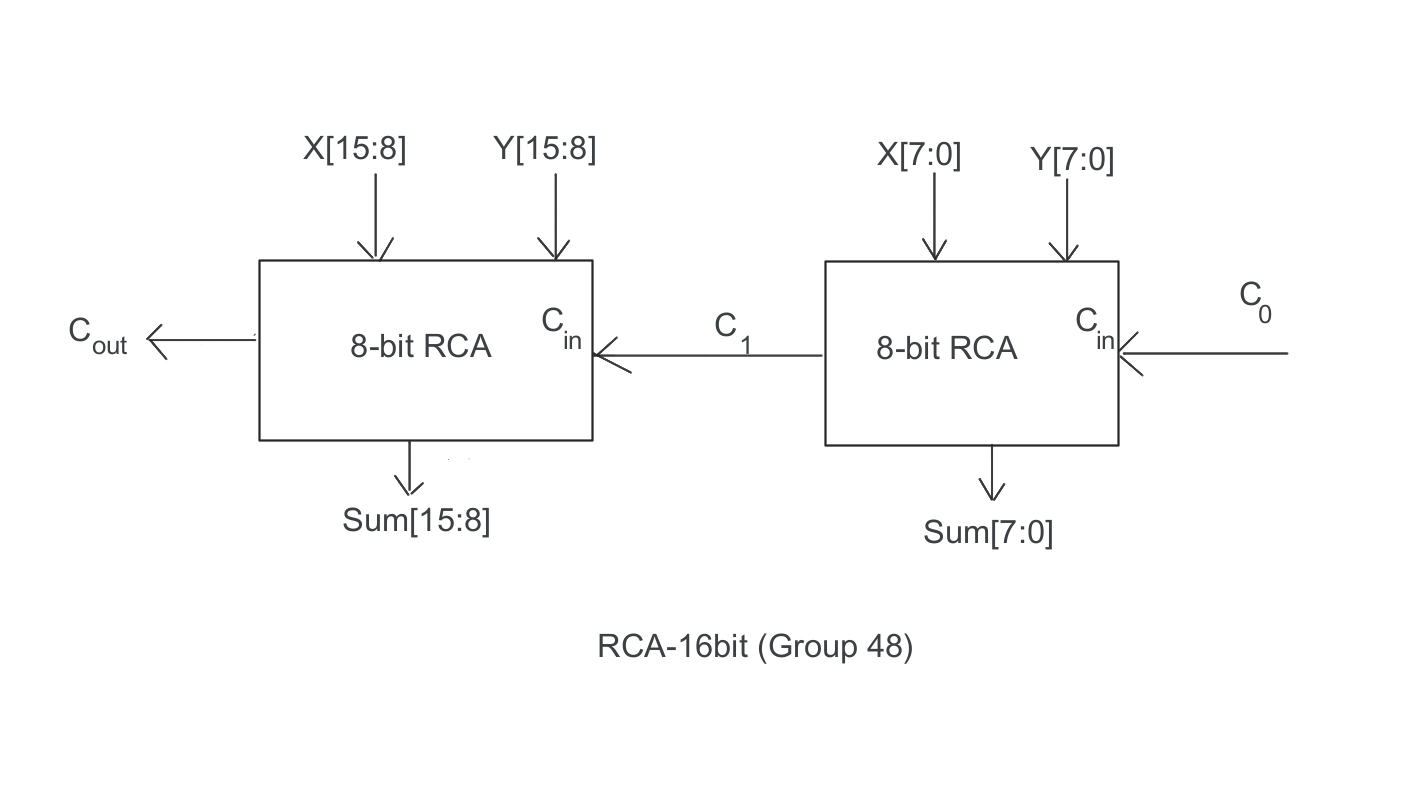
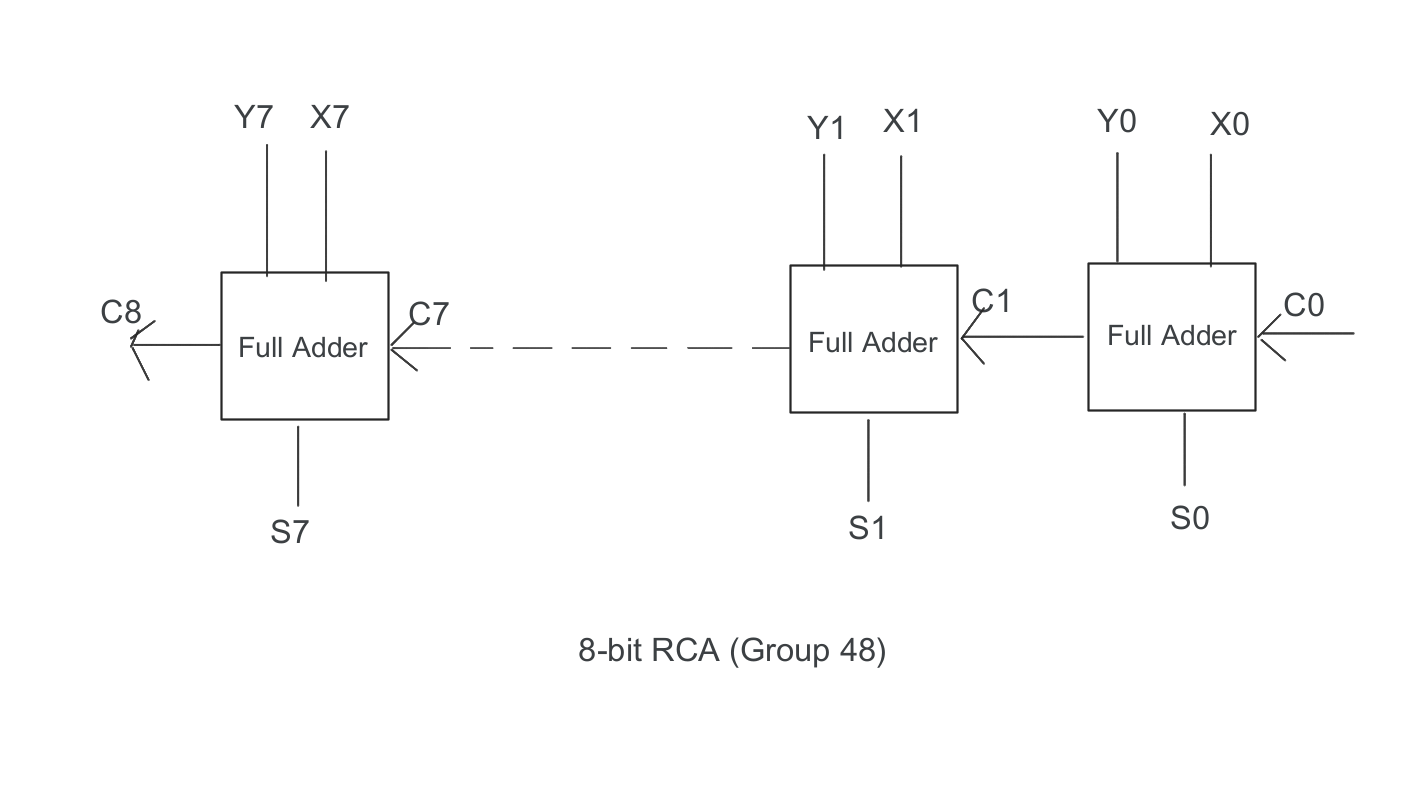
| Input | | | Output | |
| --- | --- | --- | --- | --- |
| A | B | CARRY IN | CARRY OUT | SUM |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Circuit Diagram



c) Following are the longest delays :

* 8 bit RCA - 9.949 ns
* 16 bit RCA - 18.717 ns
* 32 bit RCA - 36.253 ns
* 64 bit RCA - 71.325 ns



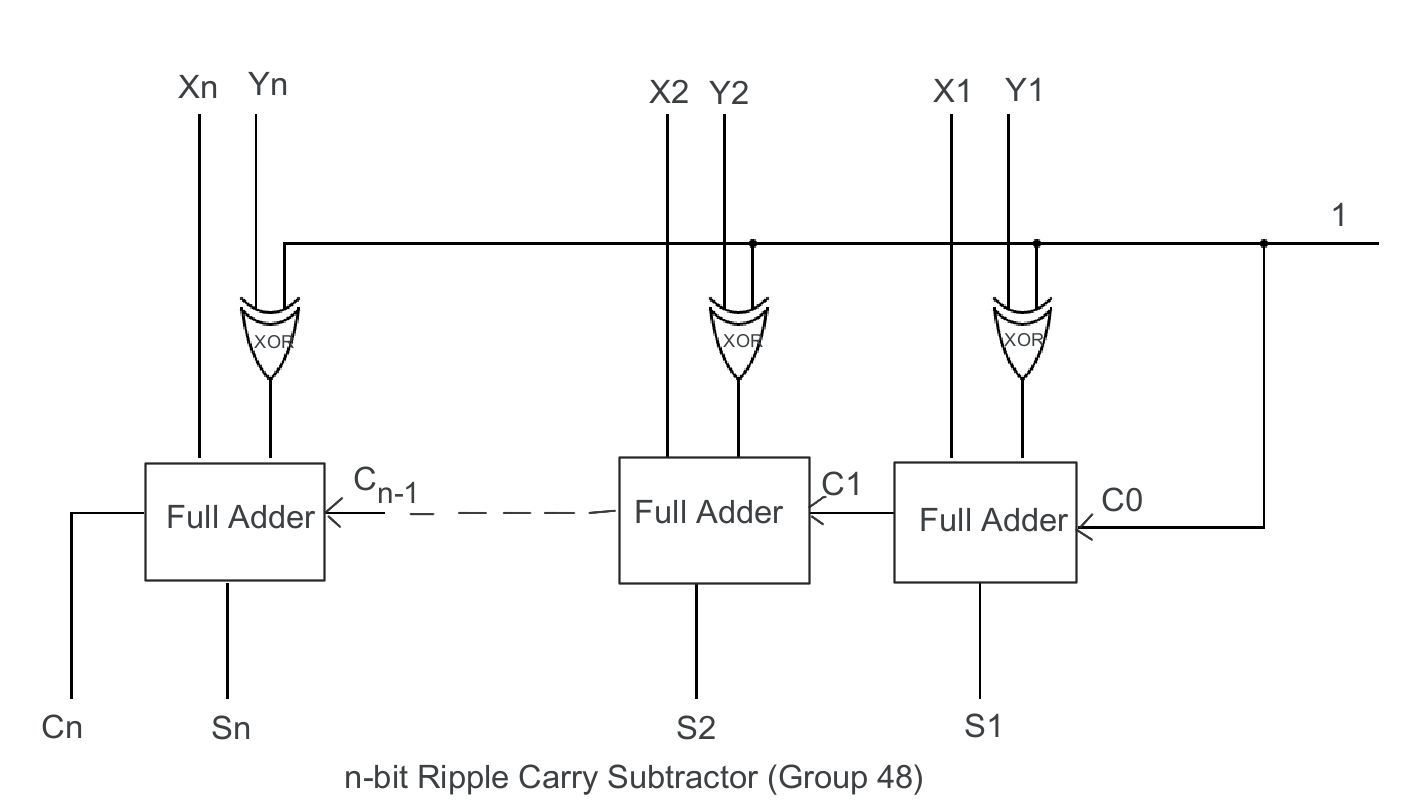
Carry is rippled into the next adder, so the next adder has to wait for the previous adder for carry computation, hence the time delays doubles every time.

d) X-Y is the same as X+(-Y).

We will first find the 2s complement of Y then add it to X using Ripple Carry Adder to get the result.

2s complement of a number = 1s complement of number + 1.

1s complement of a number is basically the inverted bits in all positions. We can use XOR gate for this as ( bit^1 = ~bit ). For adding the 1 in 1s complement we can put C0 to 1.  
  
Here is the required circuit :



**Q2)**

a)

Ci = Gi + PiCi-1

Gi = Ai.Bi (where . denotes the bitwise AND operation)

Pi = Ai ⊕ Bi (where ⊕ denotes the bitwise XOR operation)

Si = Pi ⊕ Ci-1 (where ⊕ denotes the bitwise XOR operation)

Equations for the propagate signals P

* P1 = A1 ⊕ B1
* P2 = A2 ⊕ B2
* P3 = A3 ⊕ B3
* P4 = A4 ⊕ B4

Equations for the generate signals G

* G1 = A1 . B1
* G2 = A2 . B2
* G3 = A3 . B3
* G4 = A4 . B4

Equations for the sum signals S

* S1 = P1 ⊕ C0
* S2 = P2 ⊕ C1
* S3 = P3 ⊕ C2
* S4 = P4 ⊕ C3

Equations for the carry signals C

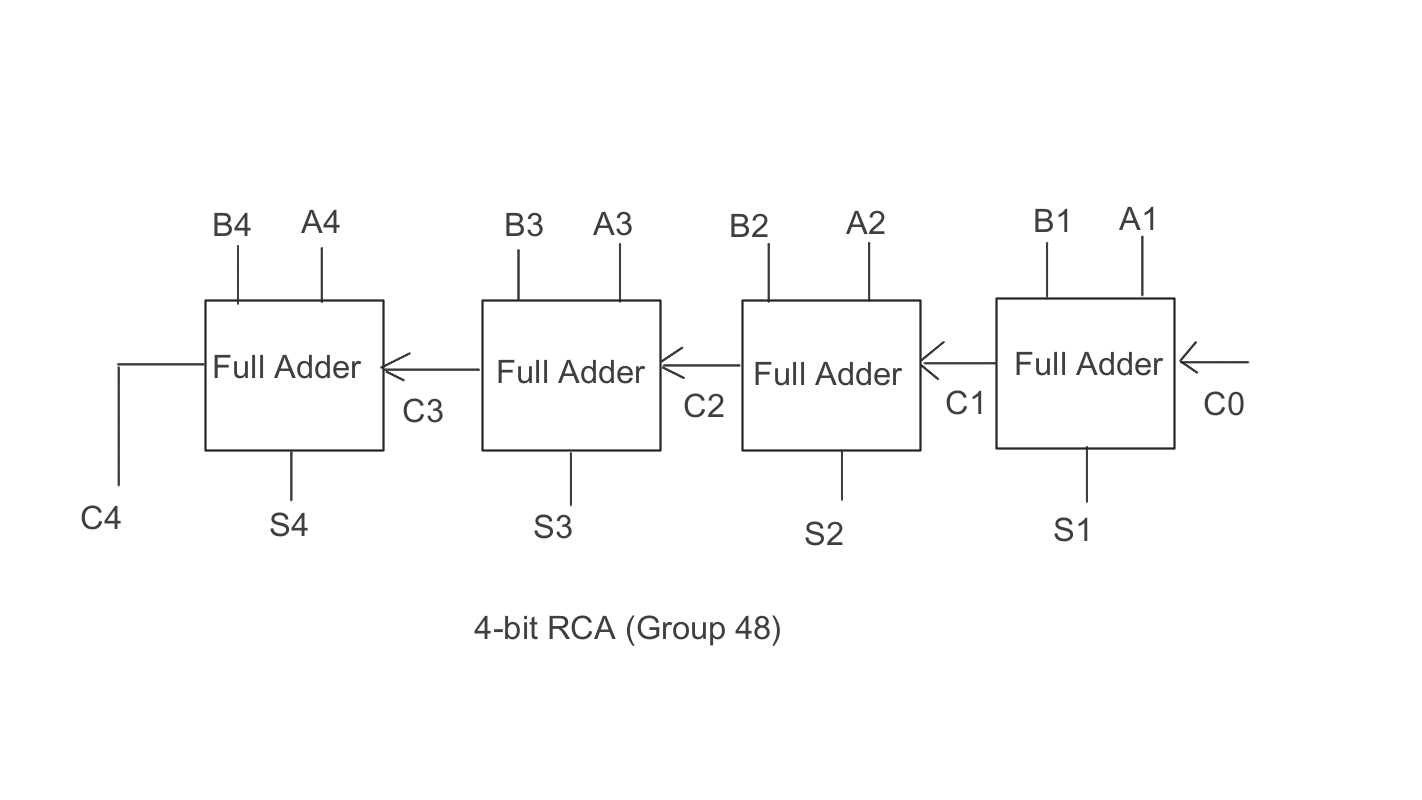
* C1 = G0 + P0C0
* C2 = G1 + P1G0 +P1P0C0
* C3 = G2 + P2G1 + P2P1G0 + P2P1P0C0
* C4 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C0

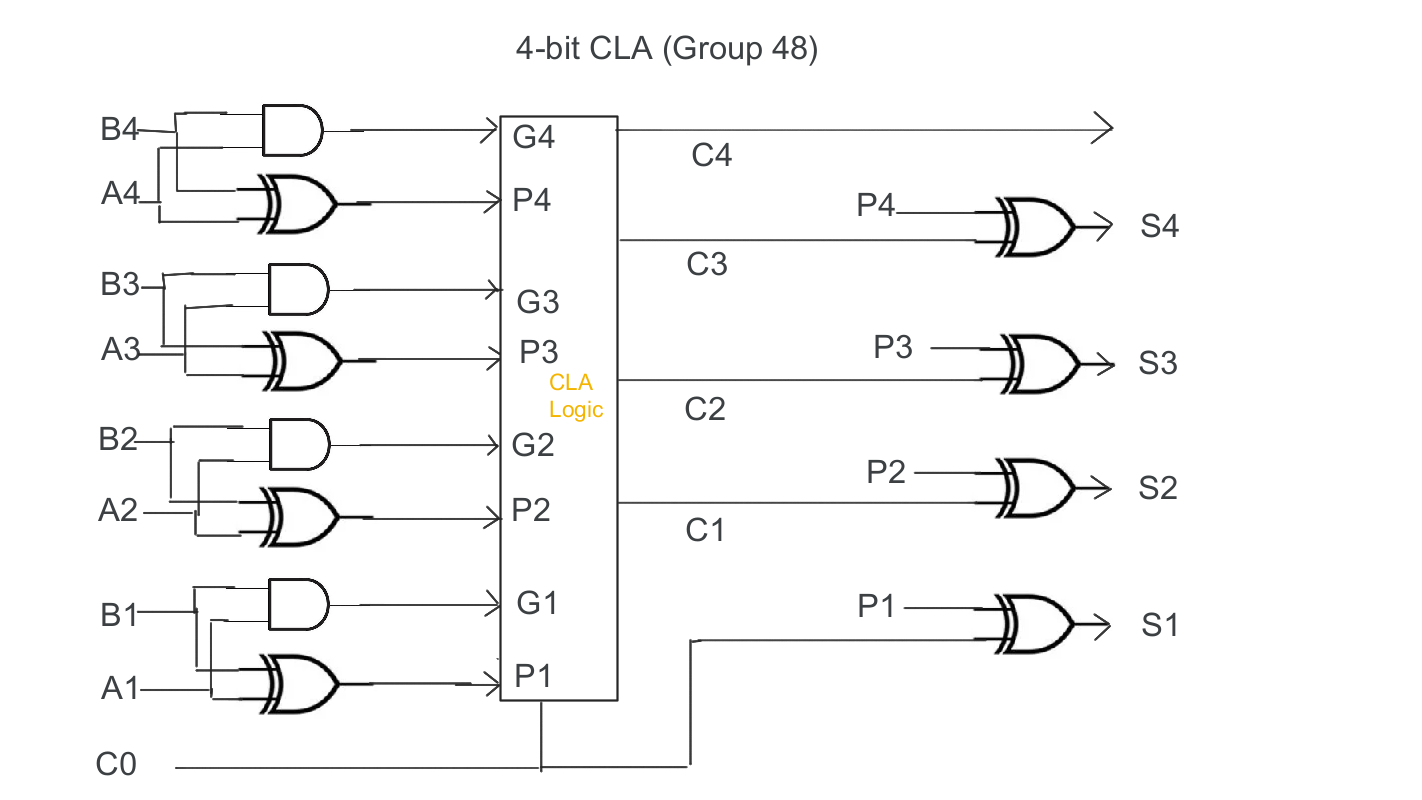
b)

4 bit CLA - 2.123 ns

4 bit RCA - 5.565 ns

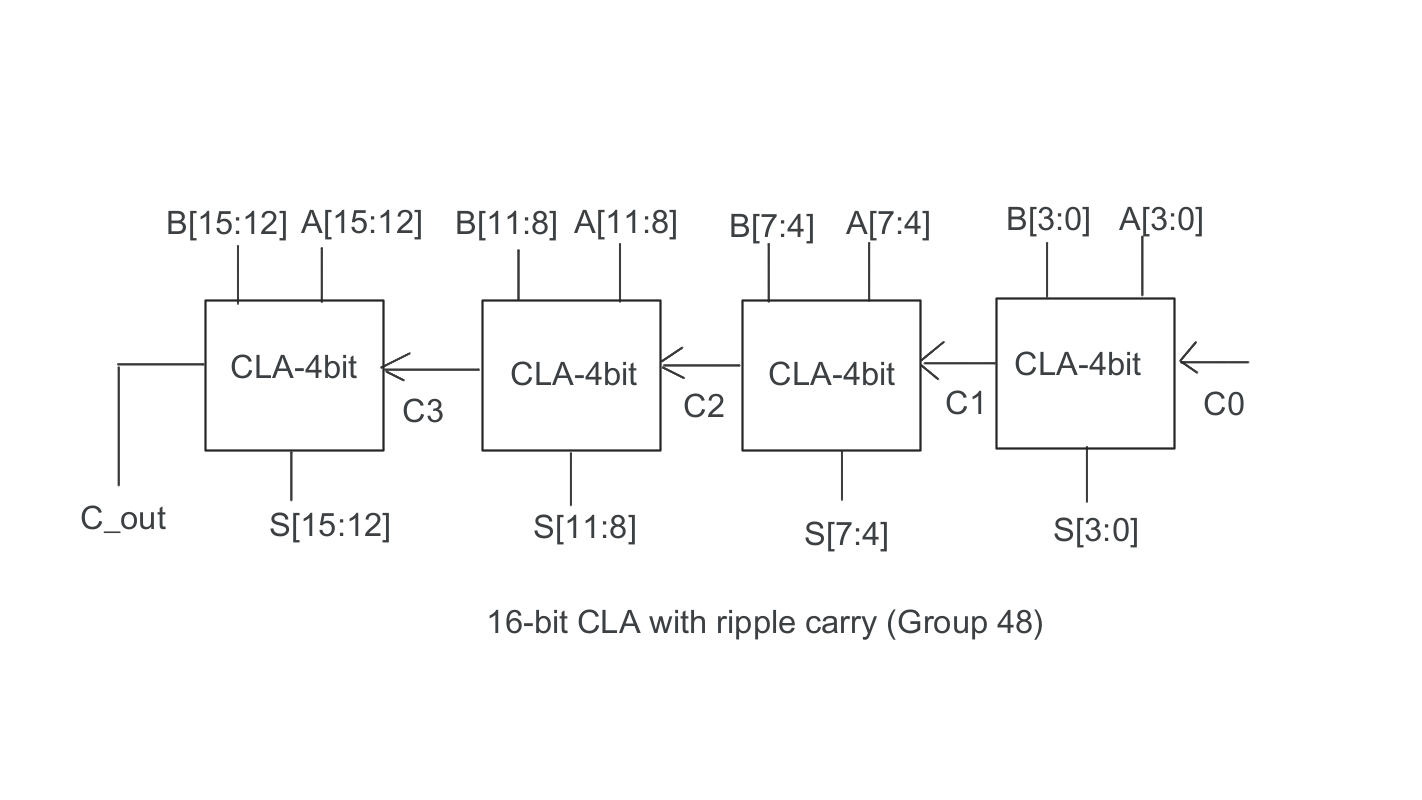
The RCA takes more time as each full adder has to wait for the previous adder to generate its output, but in CLA the carry is not dependent on the carry of the previous adder hence it is faster.

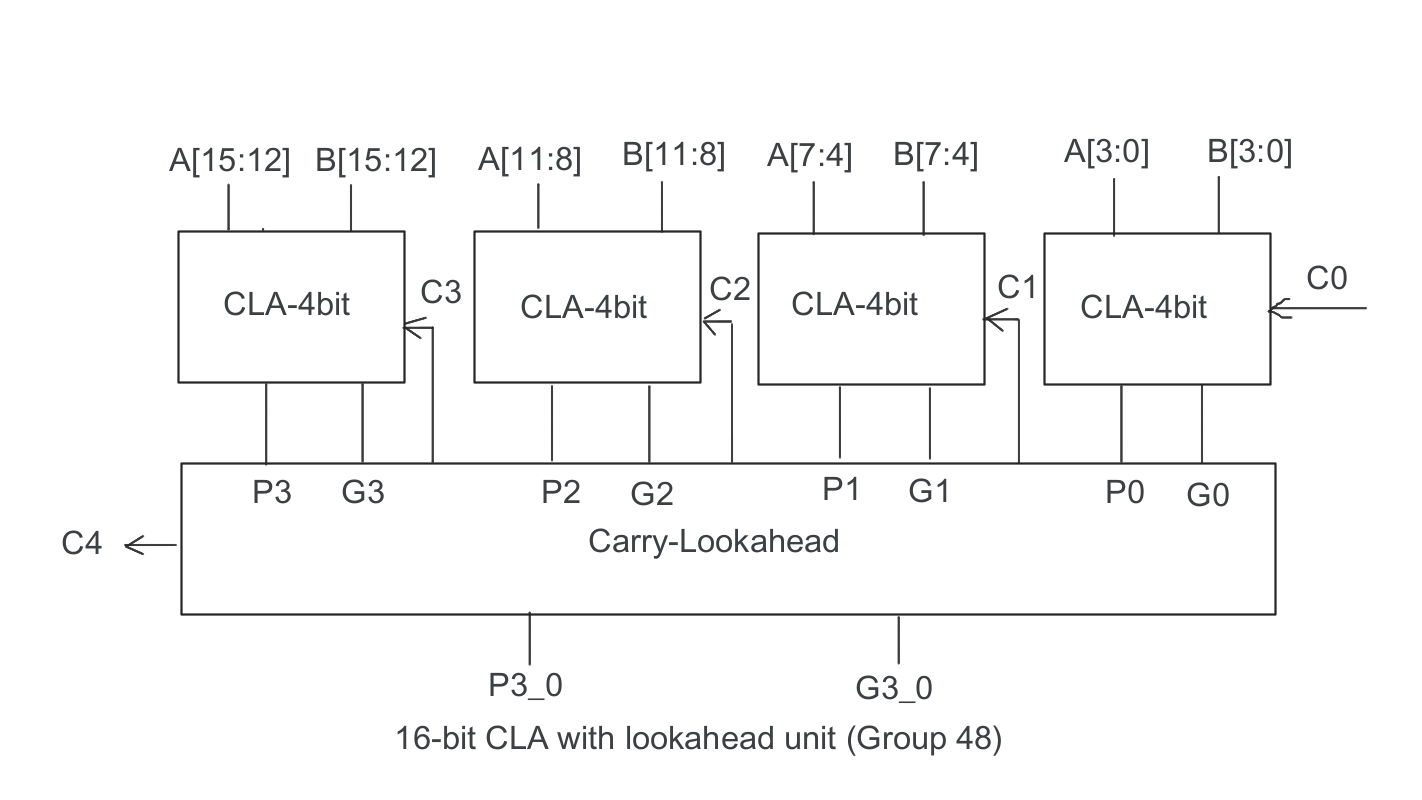




c)

i) Tested

ii)

iii) Delay for obtaining the sum and final carry for:

* 16 bit CLA (with carry rippled in, without using the second layer of lookahead): 6.167 ns
* 16 bit CLA (using the second layer of lookahead): 5.237 ns

While using the ripple method the carry for the next adder block is dependent on full computation of the previous block but using a second level lookahead unit makes this process much faster by computing the carry using the propagate and generating signals of each block.

iv) Max time delay for obtaining the sum and final carry for:

* 16 bit RCA 18.717 ns
* 16 bit CLA (using the second layer of lookahead): 5.237 ns

LUT (lookup-table cost of the FPGA) (Number of Slice LUTs) for:

* 16 bit RCA : 80 LUT out of 63400
* 16 bit CLA (using the second layer of lookahead): 42 LUT out of 63400