# ASSIGNMENT 3 REPORT

*Ripple Carry Adder (8, 16, 32 and 64 bit)*


## Jatin Gupta (20CS10087)

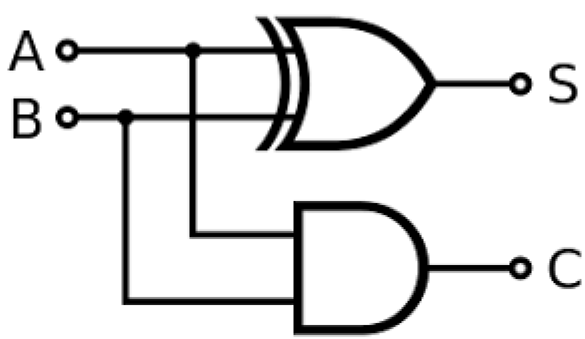
## Rushil Venkateswar (20CS30045)

Group 69
Computer Organisation Laboratory


## HALF ADDER

(Module File: half_addr.v, Test: half_addr_TestBench.v)

Circuit Diagram:




S=A^B

C = A.B

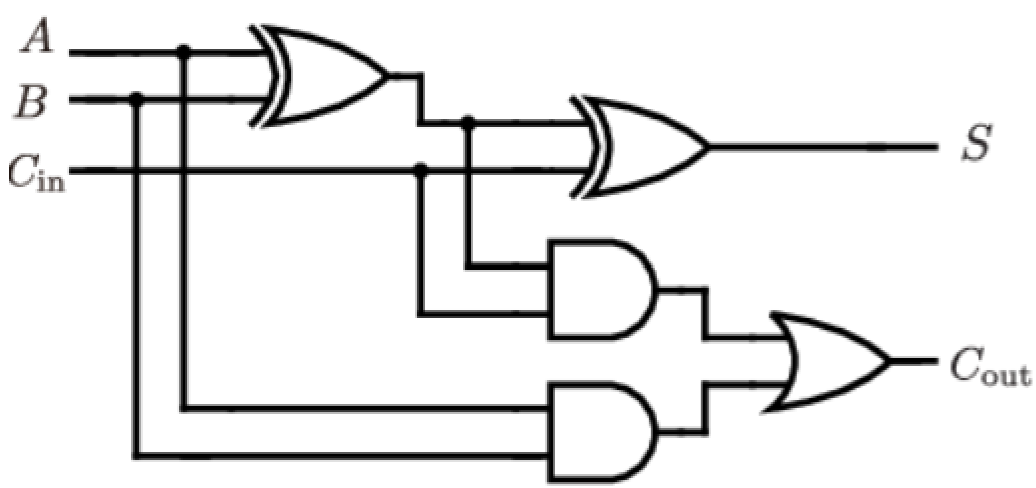
Truth Table:

| Input 1 (A) | Input 2 (B) | Sum (S = A^B) | Carry (C = A . B) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

## FULL ADDER

(Module File: full_addr.v, Test: full_addr_TestBench.v)

Circuit Diagram:




S=A^B^Cin

Cout = A.B + B.Cin + Cin.A

Truth Table:

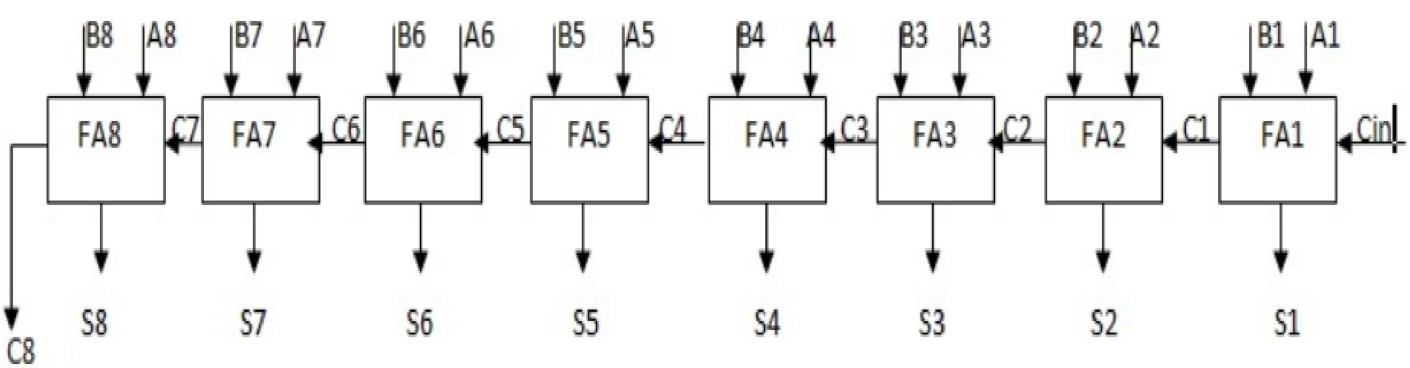
| INPUT 1 (A) | INPUT 2 (B) | Cin | SUM (S=A^B^Cin) | Carry (Cout) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

## RIPPLE CARRY ADDERS

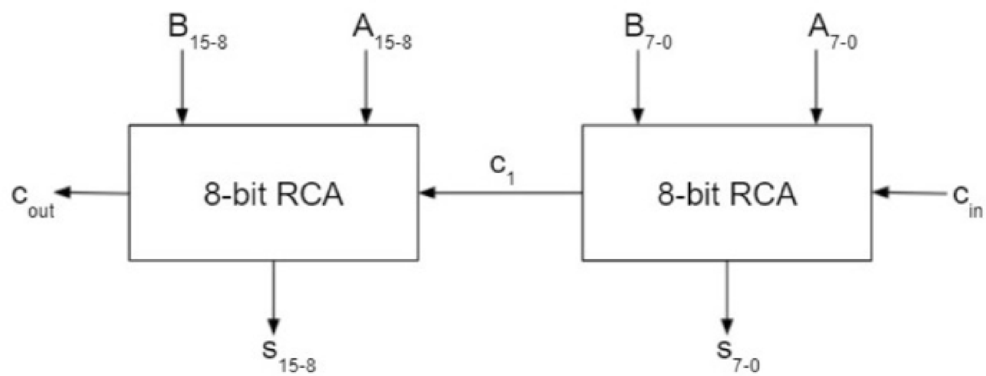
Here, we used 8 full adders to design an 8-bit ripple carry adder. Then we used two 8-bit ripple carry adders to design a 16-bit ripple carry adder. Further, we used two 16-bit ripple carry adders to design a 32-bit ripple carry adder. Finally, we used two 32-bit ripple carry adders to design a 64-bit ripple carry adder.

Circuit Diagrams:

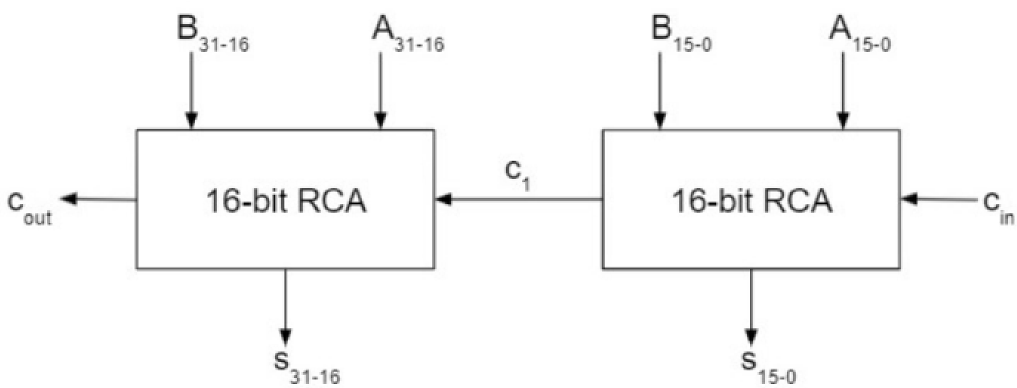
1. 8-bit ripple carry adder (Module File: RCA_8.v, Test: RCA_8_TestBench.v):



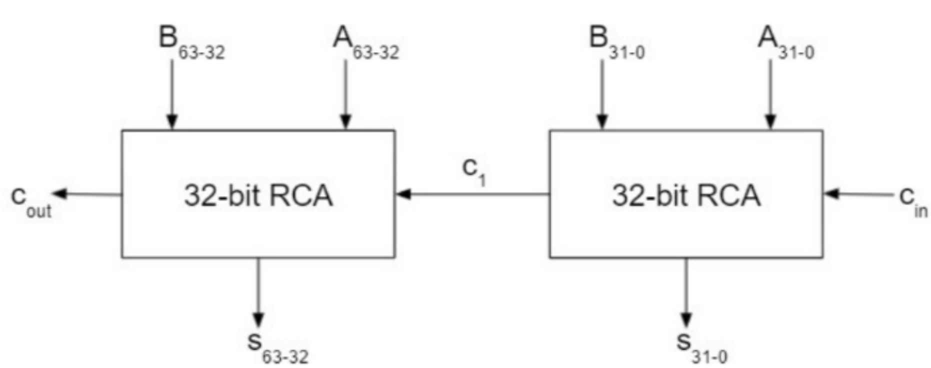
2. 16-bit ripple carry adder (Module File: RCA_16.v, Test: RCA_16_TestBench.v):




3. 32-bit ripple carry adder (Module file: RCA_32.v, Test: RCA_32_TestBench.v):




4. 64-bit ripple carry adder (Module File: RCA_64.v, Test: RCA_64_TestBench.v):



# SYNTHESIS SUMMARY

| Circuit | Total delay (ns) | Logic delay (ns) | Route delay (ns) | Number of Slice LUTs |
|---|---|---|---|---|
| Half adder | 1.066 | 0.125 | 0.941 | 2 |
| Full adder | 1.246 | 0.125 | 1.121 | 2 |
| 8-bit adder | 3.471 | 0.497 | 2.974 | 12 |
| 16-bit adder | 6.167 | 0.993 | 5.174 | 24 |
| 32-bit adder | 11.559 | 1.985 | 9.574 | 48 |
| 64-bit adder | 22.343 | 3.969 | 18.374 | 96 |

## Q. How can you use the above circuit, to compute the difference between two n-bit numbers?

Let us assume that the 2 inputs are A and B. Both are signed 64-bit numbers, so the range of both numbers is from [-2^32 to 2^32 - 1].

The **RCA_sub** module defined in our code computes A – B. The resultant difference is stored in **‘diff’**.

To achieve this, the 64-bit RCA Adder module is utilised.

A – B

=> A + (–B)

The 2’s complement notation of **–B** is **1** + ~**B**

=> A + (1 + ~B)

We **compute the NOT of B** and then pass **A,** ~**B** and **1** as the inputs to the 64-bit RCA Adder.

(Module File: RCA_subtraction.v, Test Bench: RCA_subtraction_TestBench.v)