Computer Organization Laboratory CS39001

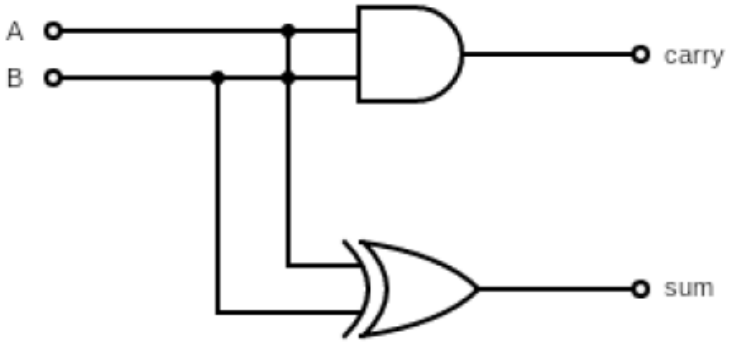
Assignment 01 - Introduction to Verilog Programming

Group – 26
Tuhin Mondal (22CS10087)
Diganta Mandal (22CS10062)

1) A) Half Adder Implementation:

Circuit diagram of half adder implemented in verilog

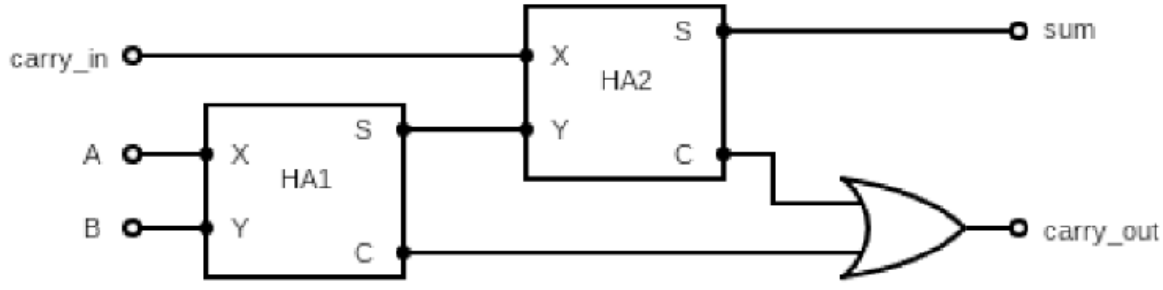
a	b	sum	carry_out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



B) Full Adder Implementation:

a	b	carry_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

Circuit diagram of full adder implemented in verilog



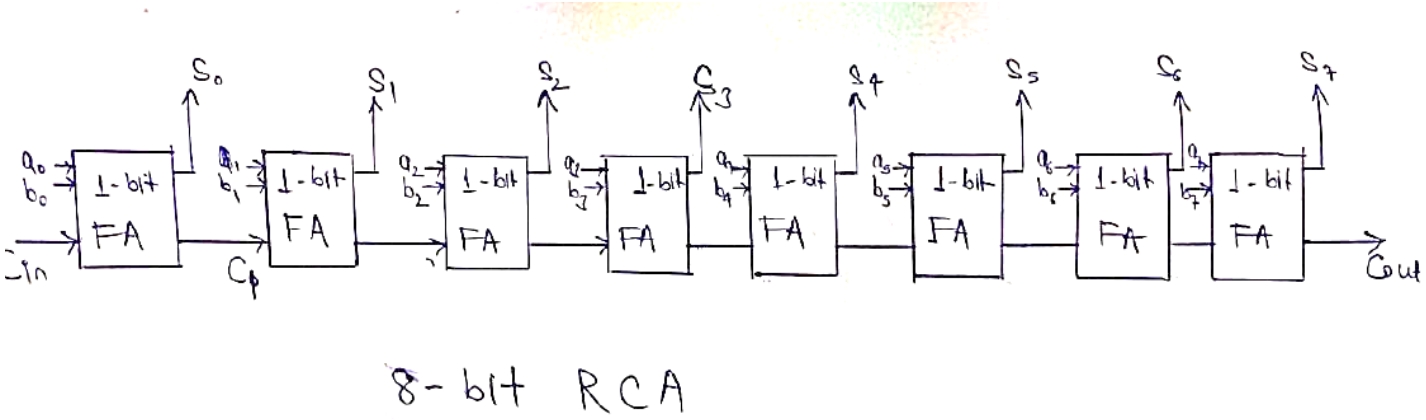
C) Two separate designs using behavioural and structural coding styles :

Can be found in the following files:

1) HALF_ADDER_BEHAVIOURIAL.v
2) HALF_ADDER_STRUCTURAL.v
3) FULL_ADDER_BEHAVIOURIAL.v
4) FULL_ADDER_STRUCTURAL.v

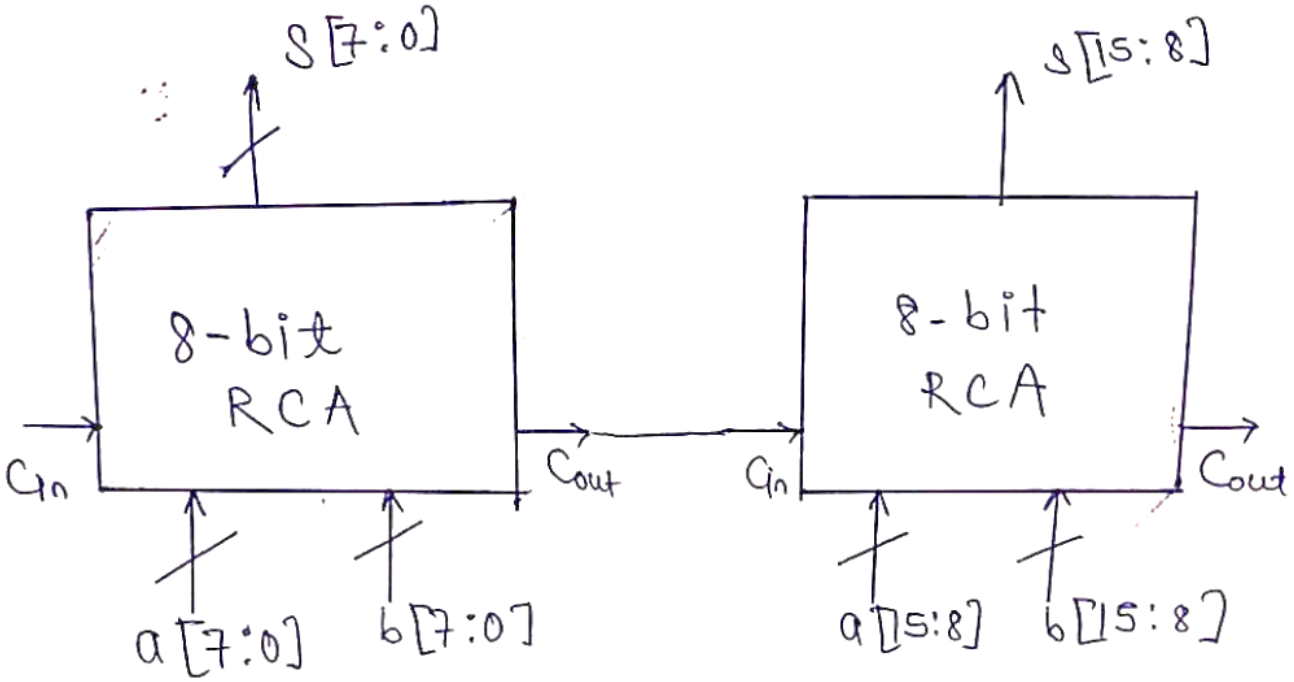
D) Ripple Carry Adder Implementation:

8 Bit RCA:

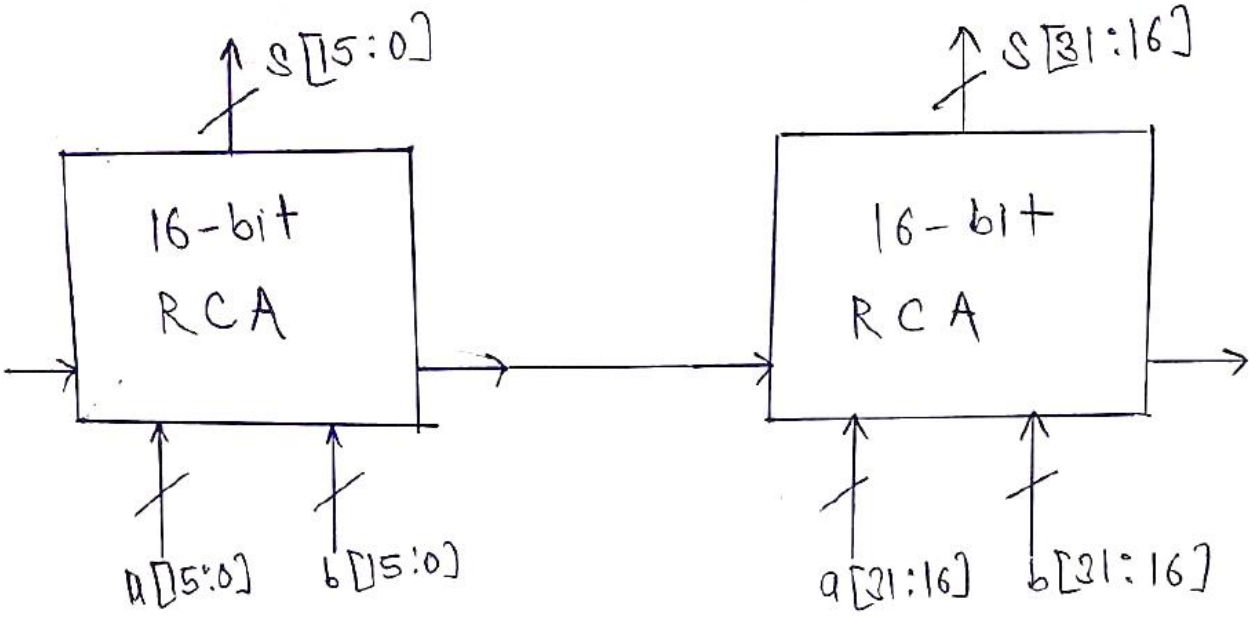


8-bit RCA

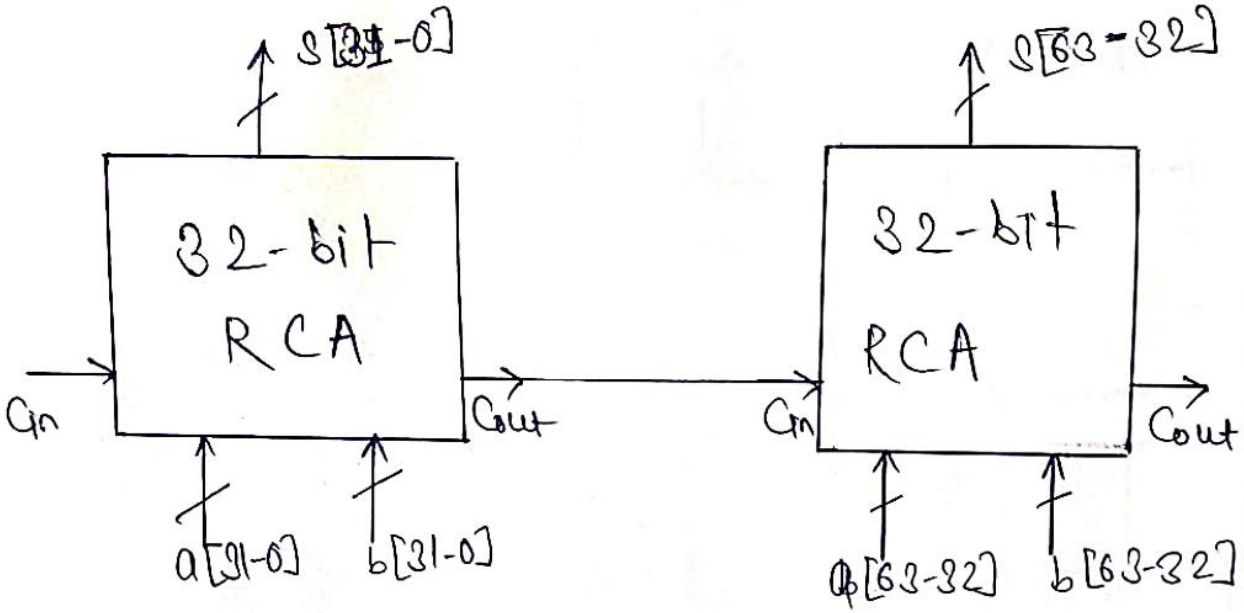
16 Bit RCA:



32 Bit RCA:



64 Bit RCA:



E) Using the above circuit to compute the difference between two n-bit numbers

Difference between two n-bit binary strings x and y (i.e., x-y) can be interpreted as the sum of binary strings x and y' where y' is the 2's complement of y. This implies that we can re-use a significant amount of circuitry implemented in addition to perform subtraction between two binary strings as well. Let us introduce a control line K as another input for the ripple carry adder that sends a single bit (0 or 1) as input. The input K is 1 if and only if subtraction is desired.

Implementation Details:

The task is to modify a ripple carry adder to perform both addition and subtraction of two n-bit binary numbers. The design introduces a control line, K, which determines the operation:

When K = 0 (Addition):
The circuit functions as a standard ripple carry adder.
The bits of y are passed unchanged to the adder.

When K = 1 (Subtraction):
The bits of y are flipped (1's complement).
The circuit then adds x and the 1's complement of y.
An additional 1 is added to the result to account for the 2's complement operation.

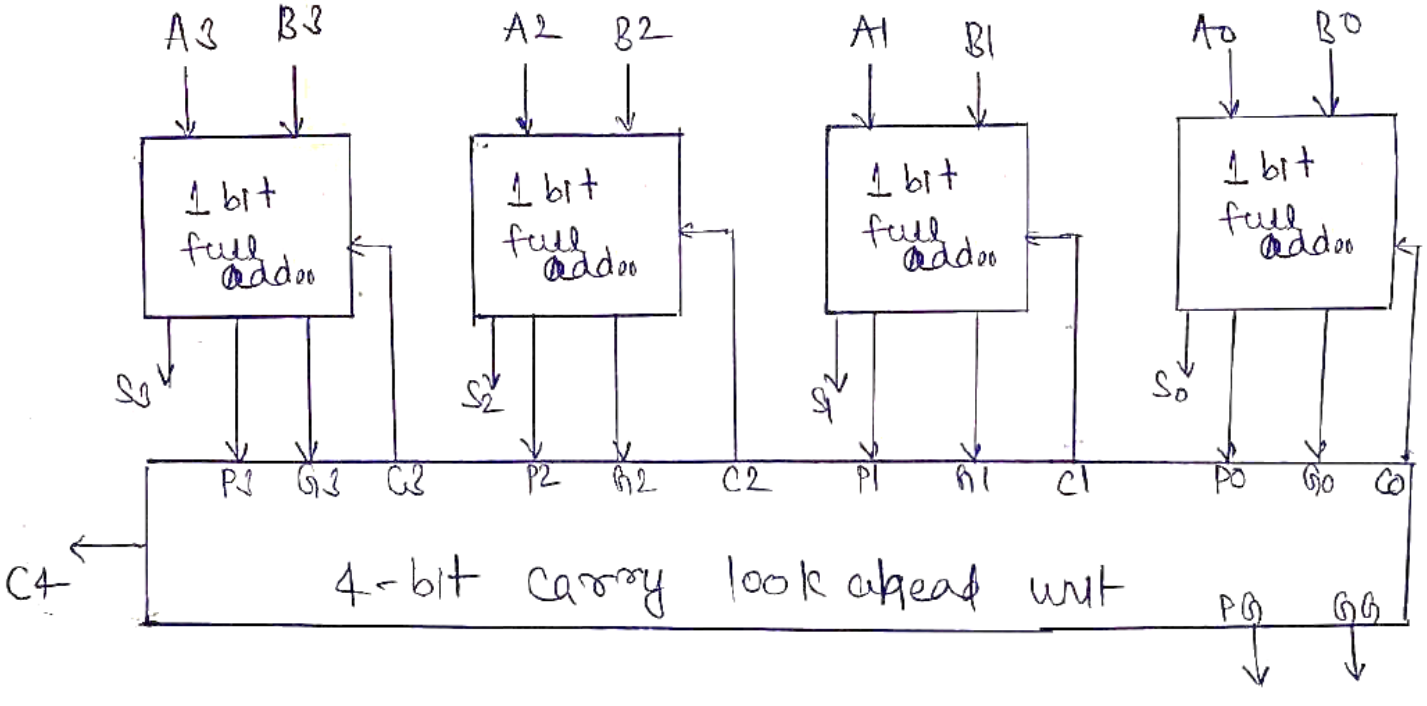
Design Changes:
Add a control line, K, to the ripple carry adder.
Use XOR gates to flip the bits of y based on the value of K.
The carry-in for the first full adder is set to K.

Result:
If K is 0, the circuit performs addition.
If K is 1, the circuit performs subtraction by adding x to the 2's complement of y.

This design effectively combines the addition and subtraction operations in a single circuit using minimal modifications. The subtraction algorithm leverages 2's complement arithmetic. To compute x - y, the algorithm first finds the 2's complement of y, which is calculated as y'=y''+1, where y'' is the 1's complement of y (flipping all the bits). Then, x−y can be computed by adding x and y'. After flipping the bits of y to get y'', 1 is added to account for the 2's complement. The final result of x−y is obtained by adding this 1 to the intermediate sum.

2) Carry Look Ahead Adder (CLA)

A) Implementation of 4 Bit Carry Look Ahead Adder (CLA)



Boolean equations for the carry look-ahead adder that determines the values of *sum* bits (S0, S1, S2, S3) in terms of carry bits (C0, C1, C2, C3) and propagate signals (P0, P1, P2, P3).

$S0 = P0 \wedge C0$

$S1 = P1 \wedge C1$

$S2 = P2 \wedge C2$

$S3 = P3 \wedge C3$

C0 is the input carry bit.

Boolean equations of the look-ahead carry generation for the 4 carry bits (C1, C2, C3, C4) in terms of the generate (G0, G1, G2, G3) and propagate (P0, P1, P2, P3, P4) signals:

$C1 = G0 + P0.\ C0$

$C2 = G1 + P1.\ G0 + P1.\ P0.\ C0$

$C3 = G2 + P2.\ G1 + P2.\ P1.\ G0 + P2.\ P1.\ P0.\ C0$

$C4 = G3 + P3.\ G2 + P3.\ P2.\ G1 + P3.\ P2.\ P1.\ G0 + P3.\ P2.\ P1.\ P0.\ C0$

C0 is the input carry bit.

Boolean equations for the generate and propagate signals in look-ahead carry generation:

$G0 = X[0]\ \&\ Y[0]$

$G1 = X[1]\ \&\ Y[1]$

$G2 = X[2]\ \&\ Y[2]$

$G3 = X[3]\ \&\ Y[3]$

$P0 = X[0] \wedge Y[0]$

$P1 = X[1] \wedge Y[1]$

$P2 = X[2] \wedge Y[2]$

$P3 = X[3] \wedge Y[3]$

X and Y are 4-bit binary strings. X[0] denotes the least and X[3] denotes the most significant bit.

D) Implementation of 16 Bit Carry Look Ahead Adder (CLA) using 4 Bit CLA

Look-ahead carry unit generates carry bits at a higher level of hierarchy. It takes 2 4-bit binary strings, block propagate signals P and block generate signals G along with an input carry bit cin as inputs and gives 4 carry bits along with block propagate and block generate signals (produced for the next level of hierarchy) as outputs. The circuit is implemented using primitive logic gates only.

The boolean equations governing the values of the carry bits are as follows.

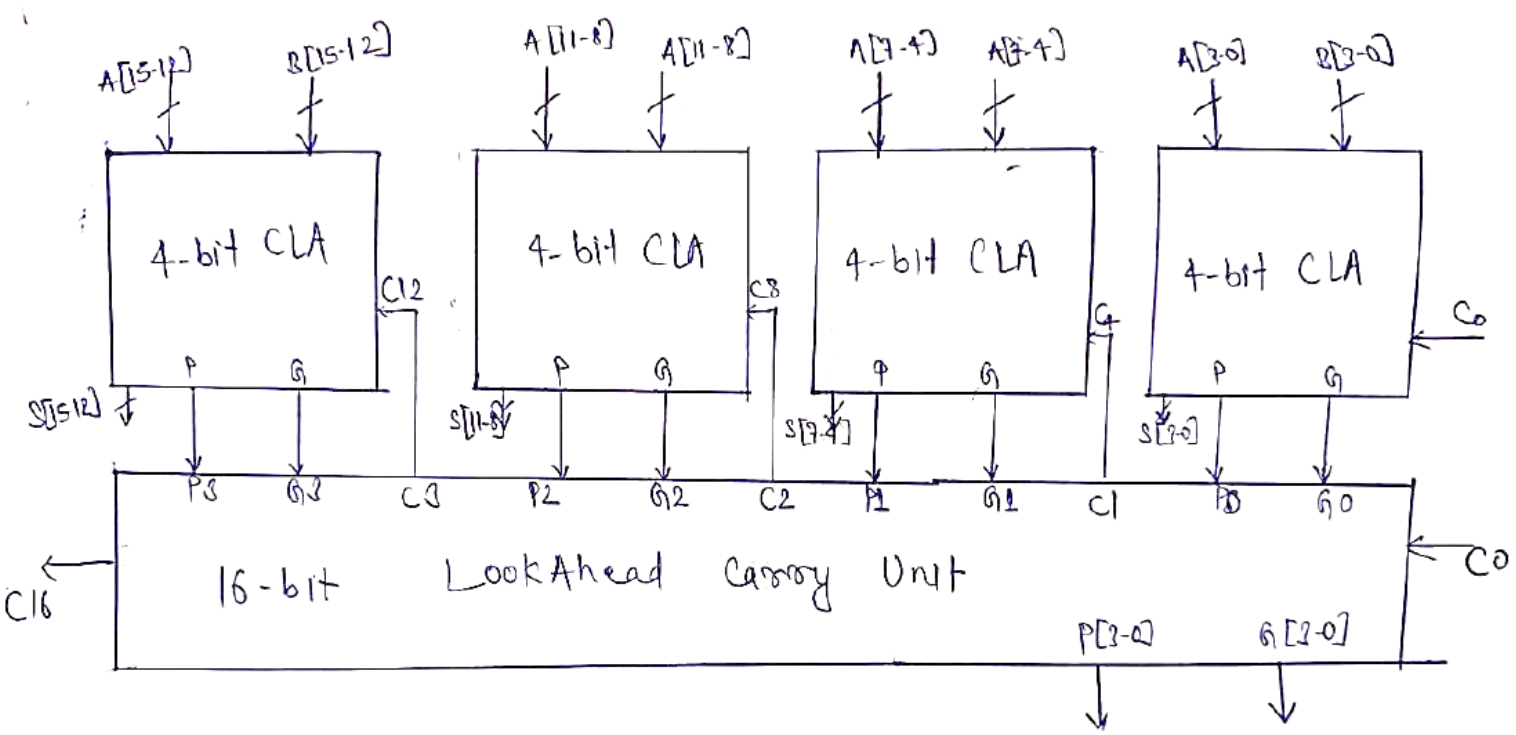
$C0 = G0 \mid (P0 \,\&\, cin)$
$C1 = G1 \mid (P1 \,\&\, C0)$
$C2 = G2 \mid (P2 \,\&\, C1)$
$C3 = G3 \mid (P3 \,\&\, C2)$

Expanded form of the above equations :

$C0 = G0 \mid (P0 \,\&\, cin)$
$C1 = G1 \mid (P1 \,\&\, (\, G0 \mid (P0 \,\&\, cin)))$
$C2 = G2 \mid (P2 \,\&\, (\, G1 \mid (P1 \,\&\, (\, G0 \mid (P0 \,\&\, cin)))))$
$C3 = G3 \mid (P3 \,\&\, (G2 \mid (P2 \,\&\, (\, G1 \mid (P1 \,\&\, (\, G0 \mid (P0 \,\&\, cin)))))))$

Block propagate and block generate signals for the next level of hierarchy :

$P = P3.\, P2.\, P1.\, P0$
$G = G3 + P3.\, G2 + P3.\, P2.\, G1 + P3.\, P2.\, P1.\, G0$



The delay in an n-bit Ripple Carry Adder (RCA) is proportional to n (O(n)), as each bit's addition depends on the carry from the previous bit, causing the carry to "ripple" through all stages sequentially. In contrast, a Carry Look-Ahead Adder (CLA) reduces this delay to O(log n) by computing carry signals in parallel using generate and propagate functions, allowing for much faster addition, especially as the number of bits increases. Thus, the CLA significantly outperforms the RCA in terms of speed for larger bit widths.