Computer Organization and Architecture Laboratory

Assignment 1

Introduction to Verilog Programming

Group 30

Ashutosh Kumar Singh - 19CS30008

Vanshita Garg - 19CS10064

1. Design of Ripple Carry Adder (RCA)

(a) Half Adder

A half adder is a combinational circuit, which takes in two input bits, a and b, and produces the sum bit, s and the carry-out bit, c.

Truth Table :

Inputs		Outputs	
a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

We can write the boolean expression for s and c as :
$s = a \oplus b$
$c = a \cdot b$

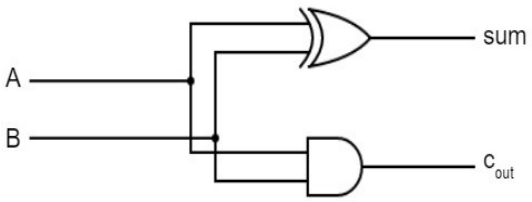

Figure 1: Half Adder

Relevant File(s) : `half_adder.v`, `half_adder_tb.v`

(b) Full Adder

A full adder is a combinational circuit, which takes in three input bits, a, b, and in addition a carry-in bit c_{in} , and produces the sum bit, s and the carry-out bit, c_{out}.

Truth Table :

Inputs			Outputs	
a	b	c_{in}	s	c_{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

We can write the boolean expression for s and c_{out} as :
$s = a \oplus b \oplus c_{in}$
$c_{out} = a \cdot b + b \cdot c_{in} + c_{in} \cdot a$

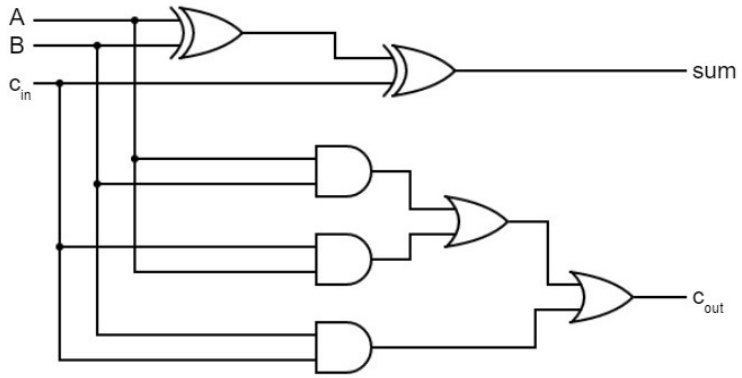

Figure 2: Full Adder

Relevant File(s) : `full_adder.v`, `full_adder_tb.v`

(c) Ripple Carry Adders

In this part, we first design an 8-bit ripple carry adder by cascading 8 full adders.
Then we cascade two 8-bit ripple carry adders to design a 16-bit ripple carry adder.
Similarly, we cascade two 16-bit ripple carry adders to design a 32-bit ripple carry adder.
Finally, we cascade two 32-bit ripple carry adders to design a 64-bit ripple carry adder.

Relevant File(s) : `rca_8_bit.v`, `rca_8_bit_tb.v`,
`rca_16_bit.v`, `rca_16_bit_tb.v`,
`rca_32_bit.v`, `rca_32_bit_tb.v`,
`rca_64_bit.v`, `rca_64_bit_tb.v`

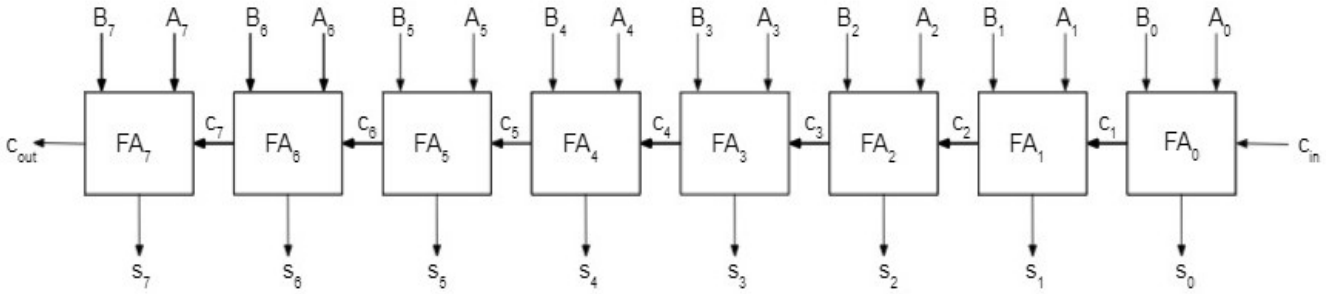

Figure 3: 8-bit Ripple Carry Adder

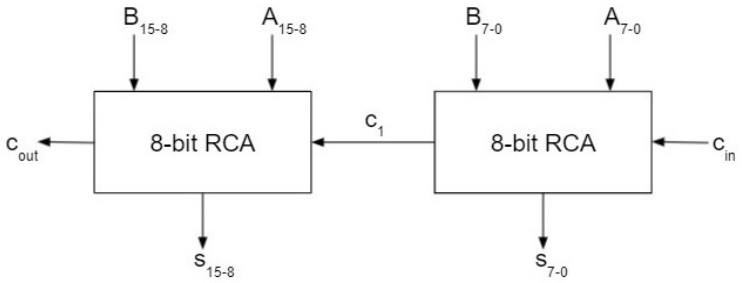

Figure 4: 16-bit Ripple Carry Adder

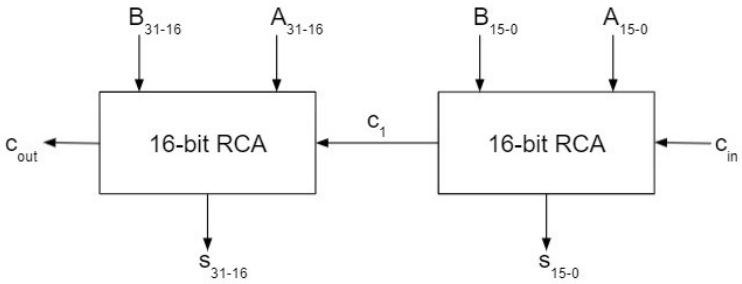

Figure 5: 32-bit Ripple Carry Adder

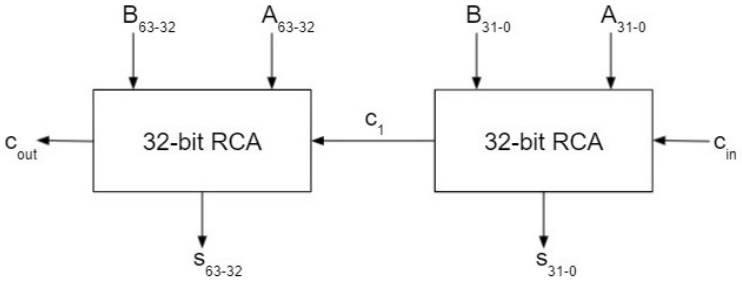

Figure 6: 64-bit Ripple Carry Adder

Longest delays observed:

- 8-bit RCA - 5.908 ns (0.993 ns logic, 4.915 ns route)
- 16-bit RCA - 11.236 ns (1.985 ns logic, 9.251 ns route)
- 32-bit RCA - 21.892 ns (3.969 ns logic, 17.923 ns route)
- 64-bit RCA - 43.204 ns (7.937 ns logic, 35.267 ns route)

For each ripple carry adder, we can see that they are a cascaded form of full adders. So, the critical path in each of them are :

- **For carry-out** : The critical path for calculating the carry-out is the path taken by the carry bits to travel from the first full adder to the last full adder, followed by the carry-out computation in the last full adder.
- **For sum** : The critical path for calculating the sum is the path taken by the carry bits to travel from the first full adder to the last full adder, followed by the sum computation in the last full adder.

For example, if we consider the 8-bit ripple carry adder (figure 3), the the critical path for computing the carry-out bit is the path C_1, C_2, C_3, C_4, C_5, C_6, C_7, followed by C_{out}, and the the critical path for computing the sum is the path C_1, C_2, C_3, C_4, C_5, C_6, C_7, followed by S_7.

(d) A ripple carry adder normally calculates $a + b$, when the carry-in bit (c_{in}) is set to 0. Now, we want to subtract two n-bit numbers. So suppose we want to calculate $a - b$. So what we can do is, in place of b, we pass the bitwise complement of b, and set the initial carry-in bit as 1.
This will compute (a + bitwise complement of $b + 1$)
= (a + (1's complement of $b + 1$))
= (a + 2's complement of b)
= ($a - b$)
Thus, in this manner we can use a ripple carry adder to subtract two n-bit numbers.
The same is illustrated in the diagram below.

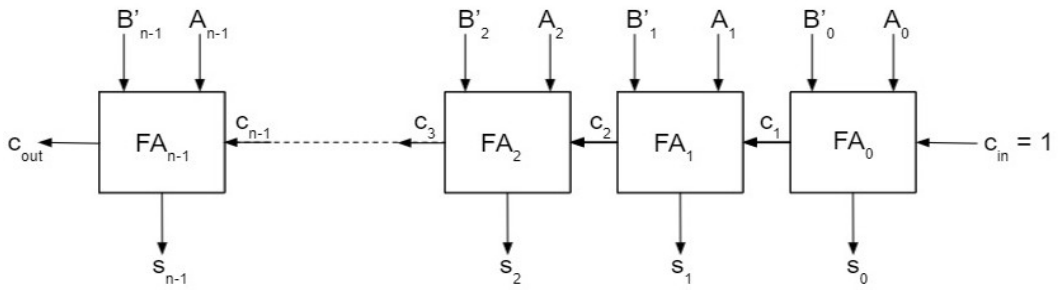

Figure 7: Difference of two numbers using RCA

2. Design of Carry Look-ahead Adder (CLA)

(a) 4-bit Carry Look-ahead Adder

For a carry look-ahead adder, for each i-th bit ($i = 0, 1, 2, 3$ for a 4-bit CLA), we define two functions :
$G_i = A_i \cdot B_i$ (carry generate)
$P_i = A_i \oplus B_i$ (carry propagate)
G_i represents the condition when a carry is generated in the i-th stage, independent of the other stages.

P_i represents the condition when an input carry C_i will be propagated to the output carry C_{i+1}.
So, we can write :
$C_{i+1} = G_i + P_i \cdot C_i$

If we expand this recurrence, we get the expressions for the carry functions :
(C_0 is the initial carry-in bit)
$C_1 = G_0 + P_0C_0$
$C_2 = G_1 + P_1G_0 + P_1P_0C_0$
$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$
(C_4 is the carry-out bit)

So, in this part, we design a 4-bit carry look-ahead adder with inputs as the two 4-bit numbers, a carry-in bit, and then using the mechanism described above, we output the 4-bit sum and the carry-out bit.
Relevant File(s) : `cla_4_bit.v`, `cla_4_bit_tb.v`

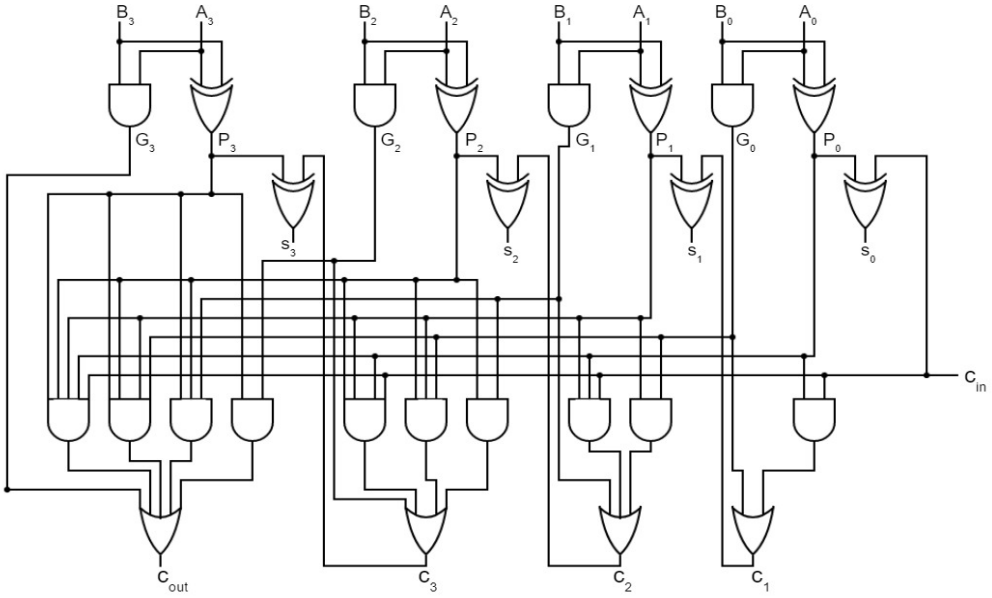

Figure 8: 4-bit Carry Look-ahead Adder

(b) In this part, we compare the delays of a 4-bit carry look-ahead adder with a 4-bit ripple carry adder.
Relevant File(s) : `cla_4_bit.v`, `cla_4_bit_tb.v`,
`rca_4_bit.v`, `rca_4_bit_tb.v`
Delay for 4-bit CLA - 2.123 ns (0.249 ns logic, 1.847 ns route)
The critical paths for a 4-bit carry look-ahead adder are :
For carry-out : The path taken by all the P_i and G_i bits to arrive at the 4 AND gates (4 leftmost AND gates in the diagram) and then their results going to the OR gate gives us the critical path for the carry-out bit.
For sum : The path taken by the P_i and G_i bits to arrive at the 3 AND gates (before the OR gate from where we get C_3) and then their results going to the OR gate (which gives C_3), followed by C_3 reaching the next OR gate (along with P_3) gives us the critical path for the sum.

Delay for 4-bit RCA - 3.244 ns (0.497 ns logic, 2.747 ns route)
The critical paths in this case are :

- **For carry-out** : The path taken by the carry bits to travel from the first full adder to the last full adder, followed by the carry-out computation in the last full adder.
- **For sum** : The path taken by the carry bits to travel from the first full adder to the last full adder, followed by the sum computation in the last full adder.

(c) 16-bit Carry Look-ahead Adder

(i) In this part, we augment the already created 4-bit CLA by adding two more output ports - P and G. P represents the block propagate for the next level of hierarchy, and G represents the block generate for the next level of hierarchy.
Relevant File(s) : `cla_4_bit_augmented.v`, `cla_4_bit_augmented_tb.v`

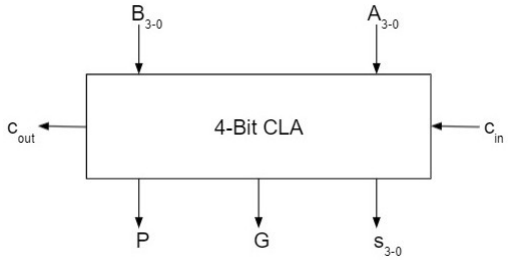

Figure 9: 4-bit Carry Look-ahead Adder (augmented)

(ii) 16-bit Carry Look-ahead Adder (using Look-ahead Carry Unit)
In this part, we first design a Look-ahead Carry Unit, which takes the block propagates and generates from the previous level as input, and computes the carry bits, and the block propagate and generate for the next level.
Relevant File(s) : `lookahead_carry_unit.v`, `lookahead_carry_unit_tb.v`

Then, we integrate this Look-ahead Carry Unit with four 4-bit augmented CLAs to design a 16-bit Carry Look-Ahead Adder.
Relevant File(s) : `cla_16_bit.v`, `cla_16_bit_tb.v`

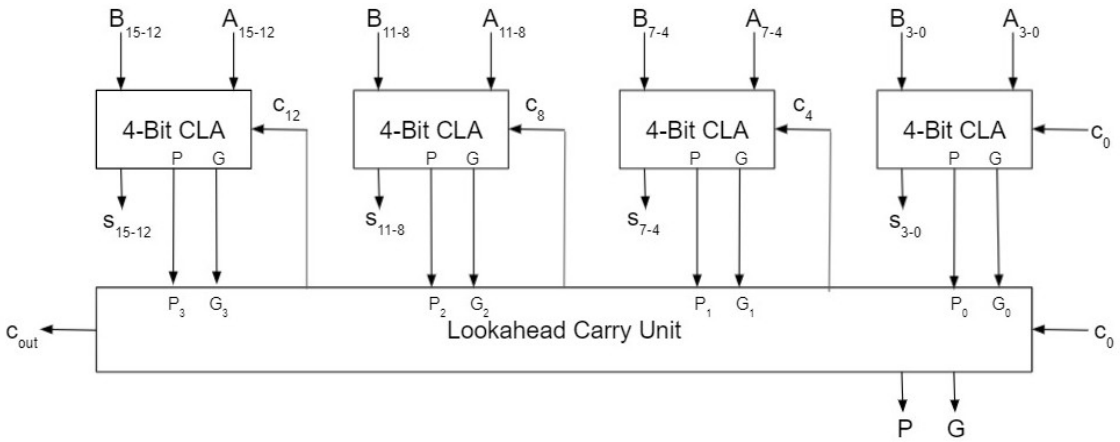

Figure 10: 16-bit Carry Look-ahead Adder (using Look-ahead Carry Unit)

(iii) In this part, we need to compare the delays of the 16-bit Carry Look-ahead Adder designed using the Look-ahead Carry Unit, with a 16-bit Carry Look-ahead Adder when the carry is just rippled in without using the second layer of look-ahead (shown in the diagram below).
Relevant File(s) : `cla_16_bit.v`, `cla_16_bit_tb.v`, `cla_16_bit_ripple.v`, `cla_16_bit_ripple_tb.v`

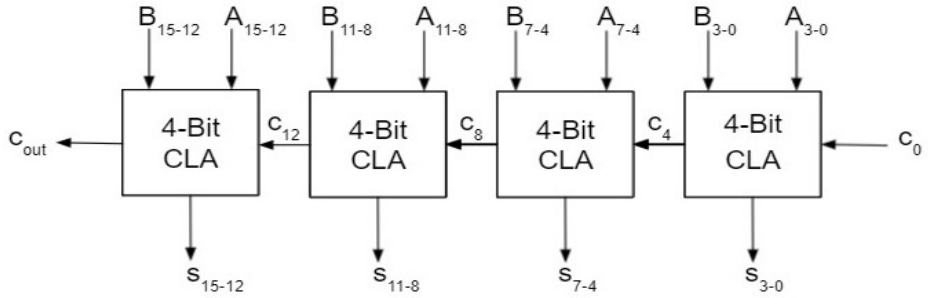

Figure 11: 16-bit Carry Look-ahead Adder (with rippling carry)

Delay for 16-bit CLA using the look-ahead carry unit - 5.243ns (0.745ns logic, 4.498ns route)
The critical paths for the carry-out and sum in this case are :

- **For carry-out** : We get all the P_i and C_i bits from the 4-bit CLAs simultaneously. Then we compute C_{out} as $G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$. The computation for this using the AND and OR gates in the look-ahead carry unit is the critical path.
- **For sum** : The sum is computed from the 4-bit CLAs itself, hence the critical path for the sum is the same as described above for the 4-bit CLA.

Delay for 16-bit CLA when the carries are rippled in - 6.167ns (0.993ns logic, 5.174ns route)
The critical paths for the carry-out and sum in this case are :

- **For carry-out** : In this case, the critical path for the carry-out is the path taken by the carry bits C_4, C_8, C_{12} to ripple through all the 4-bit CLAs to reach the last 4-bit CLA, followed by the computation in the last 4-bit CLA to get C_{out}.
- **For sum** : Similarly, the critical path for calculating the sum is the path taken by the carry bits C_4, C_8, C_{12} to ripple through all the 4-bit CLAs to reach the last 4-bit CLA, followed by the computation in the last 4-bit CLA to get S_{15-12}.

(iv) Speed Comparison :
Delay for 16-bit CLA using the look-ahead carry unit - 5.243ns (0.745ns logic, 4.498ns route)
The critical paths for the carry-out and sum in this case are :

- **For carry-out** : We get all the P_i and C_i bits from the 4-bit CLAs simultaneously. Then we compute C_{out} as $G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$. The computation for this using the AND and OR gates in the look-ahead carry unit is the critical path.
- **For sum** : The sum is computed from the 4-bit CLAs itself, hence the critical path for the sum is the same as described above for the 4-bit CLA.

Delay for 16-bit RCA - 11.236 ns (1.985 ns logic, 9.251 ns route)
The critical paths in this case are :

- **For carry-out** : The path taken by the carry bits to travel from the first full adder to the last full adder, followed by the carry-out computation in the last full adder.

- **For sum** : The path taken by the carry bits to travel from the first full adder to the last full adder, followed by the sum computation in the last full adder.

LUT Cost Comparison :
Slice LUTs used for 16-bit CLA - 43
Slice LUTs used for 16-bit RCA - 32

Table for Delays and No. of LUTs Used

Module	Total Delay (in ns)	Logic Delay (in ns)	Route Delay (in ns)	No. of LUTs Used
`rca_4_bit`	3.244	0.497	2.747	8
`rca_8_bit`	5.908	0.993	4.915	16
`rca_16_bit`	11.236	1.985	9.251	32
`rca_32_bit`	21.892	3.969	17.923	64
`rca_64_bit`	43.204	7.937	35.267	128
`cla_4_bit`	2.123	0.249	1.874	6
`cla_4_bit_augmented`	3.029	0.373	2.656	10
`cla_16_bit`	5.243	0.745	4.498	43
`cla_16_bit_ripple`	6.167	0.993	5.174	24

List of Verilog files submitted

Part 1 - Ripple Carry Adder

- `half_adder.v` - Module for half adder
- `half_adder_tb.v` - Testbench for half adder
- `full_adder.v` - Module for full adder
- `full_adder_tb.v` - Testbench for full adder
- `rca_4_bit.v` - Module for 4-bit Ripple Carry Adder by cascading 4 full adders
- `rca_4_bit_tb.v` - Testbench for 4-bit Ripple Carry Adder
- `rca_8_bit.v` - Module for 8-bit Ripple Carry Adder by cascading 8 full adders
- `rca_8_bit_tb.v` - Testbench for 8-bit Ripple Carry Adder
- `rca_16_bit.v` - Module for 16-bit Ripple Carry Adder by cascading two 8-bit Ripple Carry Adders
- `rca_16_bit_tb.v` - Testbench for 8-bit Ripple Carry Adder
- `rca_32_bit.v` - Module for 32-bit Ripple Carry Adder by cascading two 16-bit Ripple Carry Adders
- `rca_32_bit_tb.v` - Testbench for 32-bit Ripple Carry Adder
- `rca_64_bit.v` - Module for 64-bit Ripple Carry Adder by cascading two 32-bit Ripple Carry Adders
- `rca_64_bit_tb.v` - Testbench for 64-bit Ripple Carry Adder

Part 2 - Carry Look-ahead Adder

- `cla_4_bit.v` - Module for 4-bit Carry Look-ahead Adder
- `cla_4_bit_tb.v` - Testbench for 4-bit Carry Look-ahead Adder
- `cla_4_bit_augmented.v` - Module for 4-bit Carry Look-ahead Adder with augmented functionality to compute block propagate and generate for the next level of hierarchy
- `cla_4_bit_augmented_tb.v` - Testbench for the augmented 4-bit Carry Look-ahead Adder
- `lookahead_carry_unit.v` - Module for the look-ahead carry unit
- `lookahead_carry_unit_tb.v` - Testbench for the look-ahead carry unit
- `cla_16_bit.v` - Module for 16-bit Carry Look-ahead Adder using the look-ahead carry unit
- `cla_16_bit_tb.v` - Testbench for 16-bit Carry Look-ahead Adder using the look-ahead carry unit
- `cla_16_bit_ripple.v` - Module for 16-bit Carry Look-ahead Adder using rippling carry
- `cla_16_bit_ripple_tb.v` - Testbench for 16-bit Carry Look-ahead Adder using rippling carry