Computer Organization and Architechture Laboratory (CS39001)

ASSIGNMENT 1

INTRODUCTION TO VERILOG PROGRAMMING

Report Created by:

Group - 8

Monal Prasad - 19CS30030

Suhas Jain - 19CS30048

Structure of the folders and files submitted

1. **Part-1 (RCA)**
   - `half_adder.v` : Verilog code for half adder
   - `half_adder_tb.v` : Testbench for half adder
   - `full_adder.v` : Verilog code for full adder
   - `full_adder_tb.v` : Testbench for full adder
   - `ripple_carry_adder_4bit.v` : Verilog code for 4-bit ripple carry adder
   - `ripple_carry_adder_4bit_tb.v` : Testbench for 4-bit ripple carry adder
   - `ripple_carry_adder_8bit.v` : Verilog code for 8-bit ripple carry adder
   - `ripple_carry_adder_8bit_tb.v` : Testbench for 8-bit ripple carry adder
   - `ripple_carry_adder_16bit.v` : Verilog code for 16-bit ripple carry adder
   - `ripple_carry_adder_16bit_tb.v` : Testbench for 16-bit ripple carry adder
   - `ripple_carry_adder_32bit.v` : Verilog code for 32-bit ripple carry adder
   - `ripple_carry_adder_32bit_tb.v` : Testbench for 32-bit ripple carry adder
   - `ripple_carry_adder_64bit.v` : Verilog code for 64-bit ripple carry adder
   - `ripple_carry_adder_64bit_tb.v` : Testbench for 64-bit ripple carry adder

2. **Part-2 (CLA)**
   - `carry_look_ahead_4bit.v` : Verilog code for 4-bit carry look ahead adder
   - `carry_look_ahead_4bit_tb.v` : Testbench for 4-bit carry look ahead adder
   - `carry_look_ahead_4bit_aug.v` : Verilog code for augmented 4-bit carry look ahead adder with added PG and GG as output
   - `carry_look_ahead_4bit_aug_tb.v` : Testbench for 4-bit carry look ahead adder with added PG and GG as output
   - `carry_look_ahead_16bit.v` : Verilog code for 16-bit carry look ahead adder designed by precomputing the carry and using module `carry_look_ahead_4bit_aug.v`
   - `carry_look_ahead_16bit_tb.v` : Testbench for 16-bit carry look ahead adder designed by precomputing the carry and using module `carry_look_ahead_4bit_aug.v`
   - `carry_look_ahead_16bit_ripple.v` : Verilog code for 16-bit carry look ahead adder designed by rippling in the carry and using module `carry_look_ahead_4bit.v`
   - `carry_look_ahead_16bit_ripple_tb.v` : Testbench for 16-bit carry look ahead adder designed by rippling in the carry and using module `carry_look_ahead_4bit.v`

3. **Xilinx**
   - **Part-1** : Files related to Xilinx ISE stimulation of all the modules and test benches on Part-1 (RCA)
   - **Part-2** : Files related to Xilinx ISE stimulation of all the modules and test benches on Part-2 (CLA)

1. Ripple Carry Adder

(a) A Half Adder is defined as a basic four terminal digital device which adds two binary input bits. It outputs the sum binary bit s and a carry binary bit c .

A	B	$Sum\ (A \bigoplus B)$	$Carry\ (A.B)$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Files with corresponding code : `half_adder.v`
So, by using one XOR gate (for calculating Sum bit) and one AND gate (for calculating $Carry$ bit) we can easily design a half adder. Below is the figure of a basic half adder.

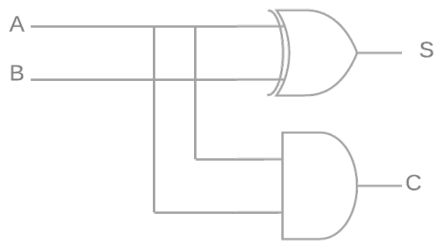

Figure 1: Half Adder

(b) Full Adder adds three inputs and produces two outputs. The first two inputs are a and b and the third input is an input carry as c_0. The output carry is designated as c and the normal output is designated as s which is SUM. A full adder logic is designed in such a manner that can take bigger inputs (4, 8, 16, 32 ...) together to create bigger adders and cascade the carry bit from one adder to the another.
Files with corresponding code : `full_adder.v`

Input			**Output**	
a	b	c_0	s	c
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

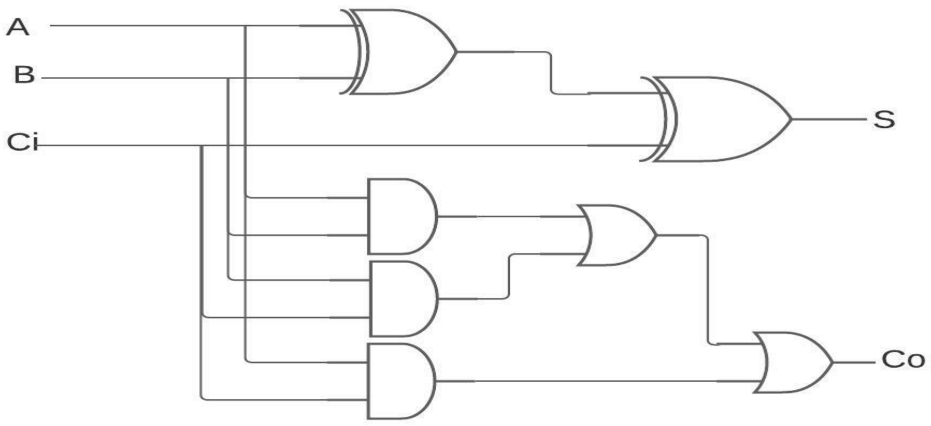

Figure 2: Full Adder

(c) Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage.
In this part we cascade 8 full adders to create a 8-bit ripple carry adder and then cascade 2 8-bit ripple carry adders to make a 16-bit ripple carry adder. Same thing is done while making 32-bit adders from 16-bit and 64-bit adders from 32-bit adders respectively (shown in the figures below).
Files with corresponding code : `ripple_carry_adder_8bit.v`, `ripple_carry_adder_16bit.v`, `ripple_carry_adder_32bit.v`, `ripple_carry_adder_64bit.v`

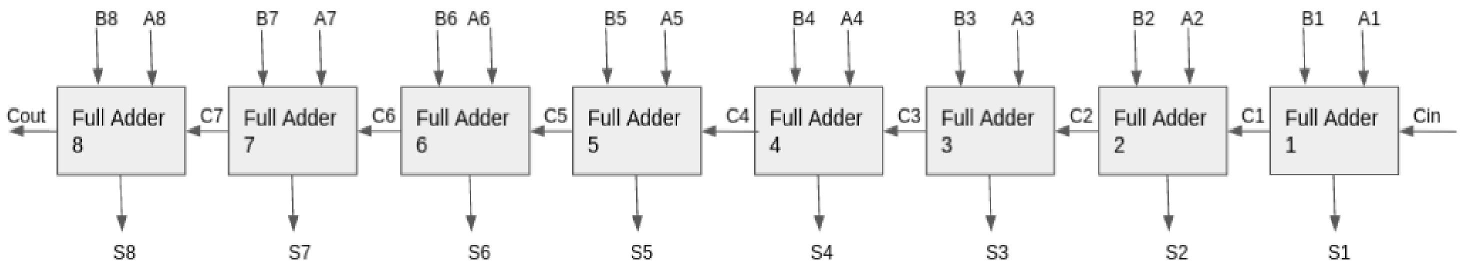

Figure 3: 8-bit Ripple Carry Adder

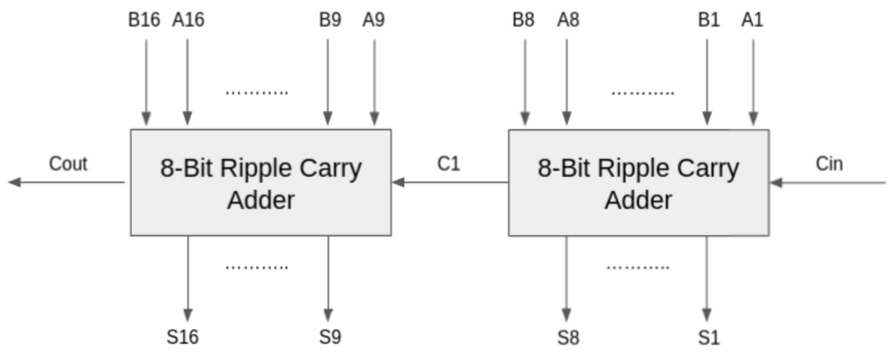

Figure 4: 16-bit Ripple Carry Adder

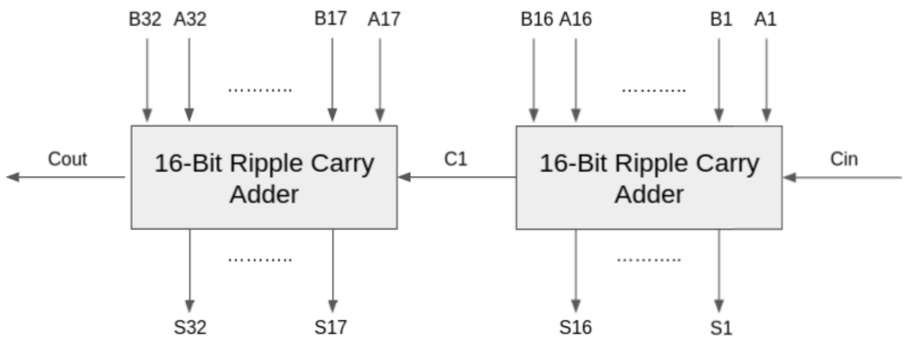

Figure 5: 32-bit Ripple Carry Adder

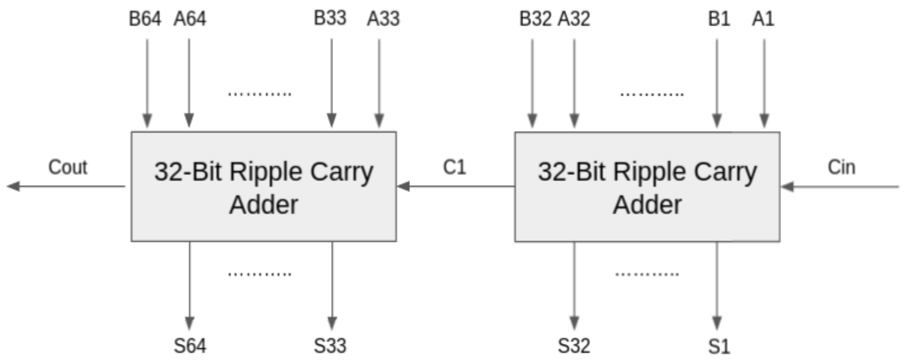

Figure 6: 64-bit Ripple Carry Adder

The longest delays obtained when these adders are stimulated on Xilinx ISE are given in the table below.

Adder Circuit	Total Delay	Logic Delay	Route Delay
Ripple Carry Adder 8 bit	5.908 ns	0.993 ns	4.915 ns
Ripple Carry Adder 16 bit	11.236 ns	1.985 ns	9.251 ns
Ripple Carry Adder 32 bit	21.892 ns	3.969 ns	17.923 ns
Ripple Carry Adder 64 bit	43.204 ns	7.937 ns	35.267 ns

Observation : As we can see the delay almost doubles if we double the number of bits which are given as input in the ripple carry adder because processing of one block always depends on the previous block. So, doubling the length of critical path basically doubles the delay.

(d) We can compute the difference between two n bit numbers X and Y by adding the X and two's complement of Y. The two's complement of an N-bit number is defined as the result of subtracting the number from 2^N. This is also equivalent to taking the ones' complement and then adding one, since the sum of a number and its ones' complement is all 1 bits. Note also that in twos-compliment arithmetic, the value of the second operand must not only be inverted, but 1 must be added to it. For this reason, when performing subtraction, the carry input into the LSB should be a 1 and not a zero.

$Difference = X+$(1's complement of $Y + 1$)
$Difference = X+$(2's complement of Y)
$Difference = X - Y$

The figure below shows how we can accomplish both addition and subtraction from the same circuit. When we want to add 2 numbers $(X + Y)$ we can set the P bit to 0 and when we want to subtract two numbers $(X - Y)$, we can set the P bit to 1. Setting the P bit to 1, complements each bit of Y and sets carry input bit to 1.

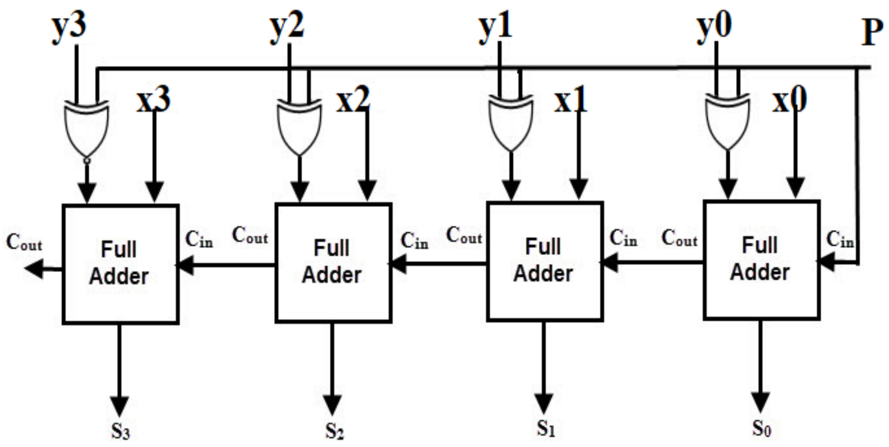

Figure 7: Ripple carry adder-subtractor circuit

2. Carry Look Ahead Adder

(a) The carry output Boolean function of each stage in a 4 stage carry look-ahead adder can be expressed

Carry Signals :

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

Generate Signals :

For all individual bits :

$G_i = A_i \cdot B_i$

Propagate Signals :

For all individual bits :

$P_i = A_i \bigoplus B_i$

The diagram explaining the inner structuring of the 4-bit CLA is given below.

File with corresponding code : `carry_look_ahead_4bit.v`

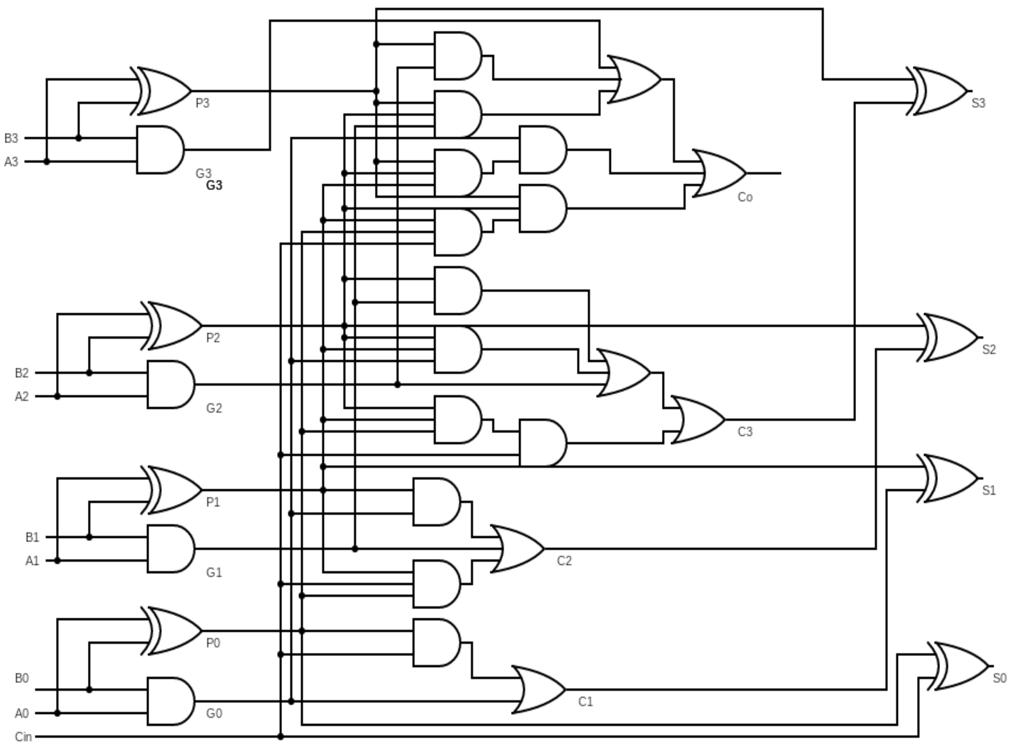

Figure 8: 4-bit Carry Look Ahead Adder

(b) In this part we are comparing the delays of a 4-bit ripple carry adder (Figure 9) and 4-bit carry look ahead adder (Figure 8). These are two different kinds of digital binary adders based on the carry determining technique. Time delays in calculating sum by the adders differs in both of them.

Delay for 4-bit carry look ahead adder : 2.123 ns (0.249 ns logic, 1.847 ns route)
File used to calculate : `carry_look_ahead_4bit.v`
Critical Path : The path which is used to calculate carry-out bit, in the top portion of the diagram(Figure 8), the path where all the P_i and G_i bits arrive and then AND is taken, after which we take subsequent OR is the path which takes highest amount of computation in the whole circuit, making it the critical path.

Delay for 4-bit ripple carry adder : 3.244 ns (0.497 ns logic, 2.747 ns route)
File used to calculate : `ripple_carry_adder_4bit.v`
Critical Path : The path which is used for calculating both the sum and carry bits is the same as the path by which carry bit travels between subsequent full adders. Since calculation of one carry bit is dependent on the input of previous carry bit. This path takes the highest amount of time and hence is critical.
Diagram of 4-bit ripple carry adder is shown below

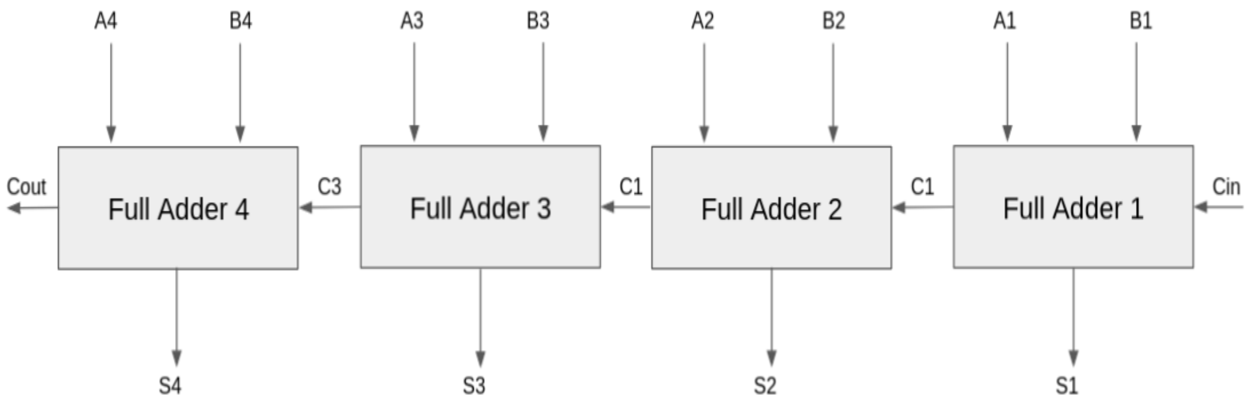

Figure 9: 4-bit ripple carry adder

(c) **(i)** In this part we modify the 4-bit CLA that we created before by adding 2 new outputs ports, these are P and G. P represents the block propagate signal and G represents the block generate signal for the next level of hierarchy. These are useful as if we want to design circuits which take bigger bit size as inputs we can use these blocks as modular components and easily design those circuits.

File with corresponding code : `carry_look_ahead_4bit_aug.v`

(ii) Using the augmented 4-bit carry look ahead adder we have designed the 16-bit carry look ahead adder in this part (shown in figure below). In the 4 individual 4-bit carry look ahead adders the 4 sum bits are calculated making a total of 16 sum bits. In the 16-bit CLA input carry bits for each of these 4 individual 4-bit carry bits ($C_4, C_8, C_{12}, C_{out}$) are pre-calculated so that they can be fed as input. In this block output PG and GG bits are also calculates so that in case we want to make a higher-level carry look ahead adder in the future, we can combine more such blocks and make it in a easy and modular way.

File with corresponding code : `carry_look_ahead_16bit.v`

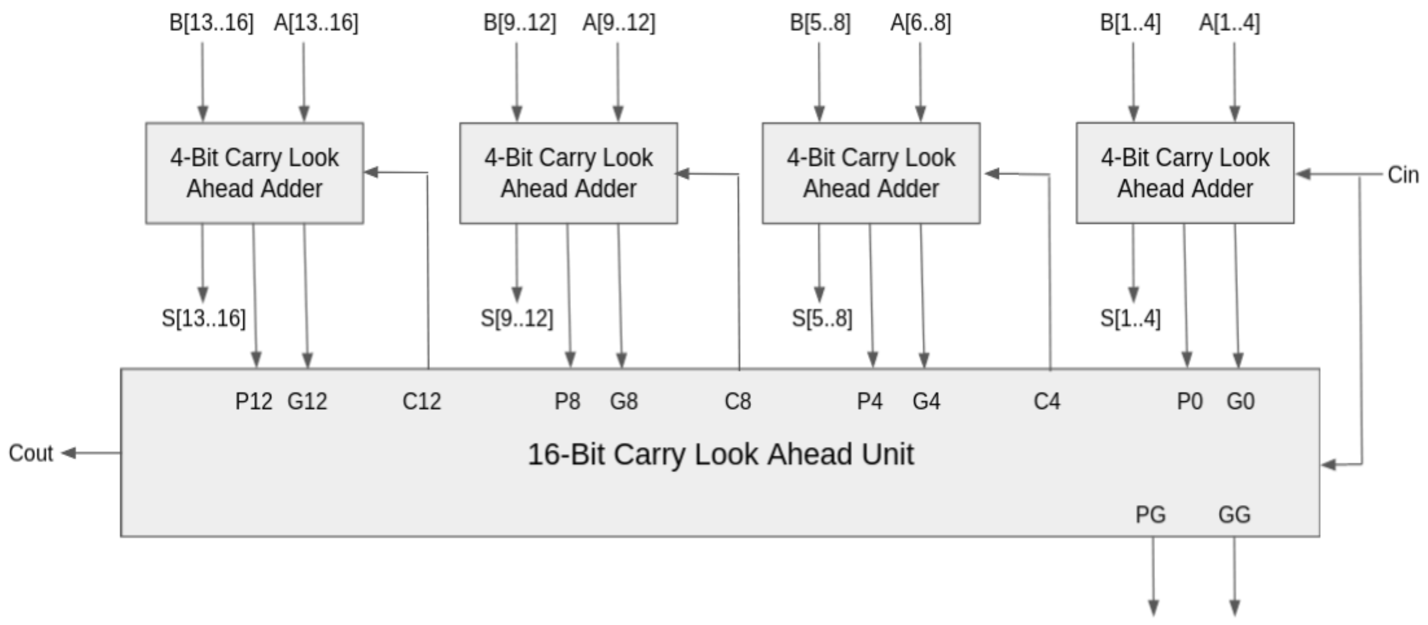
B[13..16]
A[13..16]
B[9..12]
A[9..12]
B[5..8]
A[6..8]
B[1..4]
A[1..4]
4-Bit Carry Look Ahead Adder
4-Bit Carry Look Ahead Adder
4-Bit Carry Look Ahead Adder
4-Bit Carry Look Ahead Adder
Cin
S[13..16]
S[9..12]
S[5..8]
S[1..4]
P12
G12
C12
P8
G8
C8
P4
G4
C4
P0
G0
Cout
16-Bit Carry Look Ahead Unit
PG
GG

Figure 10: 16-bit Carry Look Ahead Adder

(iii) In this part we are designing two different 16-but adders by combining four 4-bit carry look ahead adders in two different ways. First we make a 16-bit adder by using the augmented 4-bit carry look ahead adder (Figure 10). In the 4 individual 4-bit carry look ahead adders the 4 sum bits are calculated making a total of 16 sum bits. In the 16-bit CLA input carry bits for each of these 4 individual 4-bit CLAs are pre-calculated so that they can be fed as input. Secondly we make a 16 bit adder by making sort of a 16-bit ripple carry adder as we make previously, only difference being that it is made up of 4-bit carry look ahead adders rather than 4-bit ripple carry adders (Figure 11). In such a circuit only the carry bits that are being transferred between individual 4-bit blocks are being rippled, rest all the carry bits within each blocks are computed simultaneously.

Delay for 16-bit carry look ahead adder : 5.243 ns (0.745 ns logic, 4.498 ns route)
File used to calculate : `carry_look_ahead_16bit.v`
Critical Path : We get all the P (block propagate) and G(block generate) signals from all the blocks simultaneously and by this time sum has also been calculated by all the individual blocks, after this step we compute our output carry i.e.
$C_{out} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$,
which takes the highest amount of time. Therefore this is the critical path in this circuit.

Delay for 16-bit carry look ahead adder with carry rippled in : 6.167ns (0.993ns logic, 5.174 ns route)
File used to calculate : `carry_look_ahead_16bit_ripple.v`
Critical Path : In this case the critical path is the path taken by the carry bits C_4, C_8, C_{12}(in Figure 11) to ripple through individual 4-bit CLAs. An individual CLA block cannot start computation and calculate $S_{1..16}$ before the input carry bit for a particular block arrives. This process end at the last block where finally C_{out} is calculated.

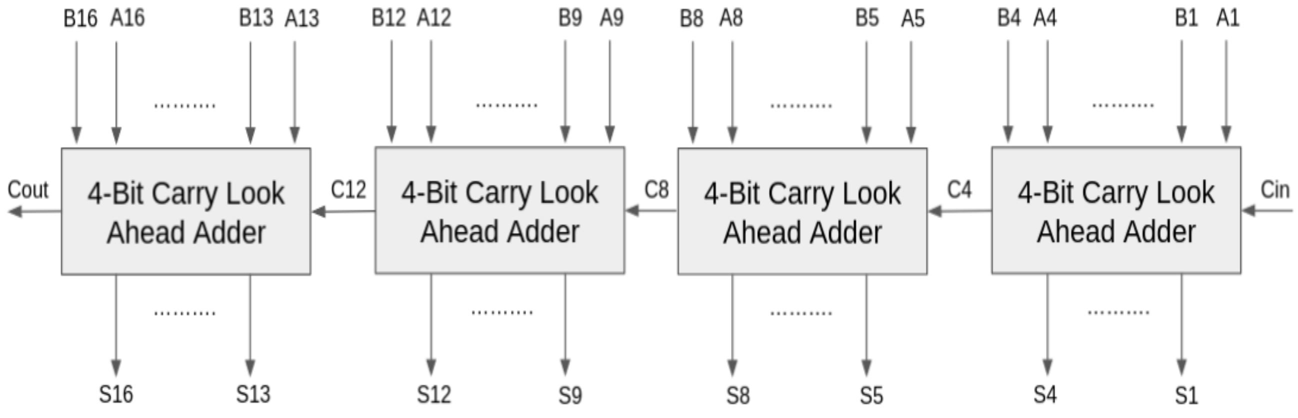
B16 A16
B13 A13
B12 A12
B9 A9
B8 A8
B5 A5
B4 A4
B1 A1
Cout
4-Bit Carry Look Ahead Adder
C12
4-Bit Carry Look Ahead Adder
C8
4-Bit Carry Look Ahead Adder
C4
4-Bit Carry Look Ahead Adder
Cin
S16
S13
S12
S9
S8
S5
S4
S1

Figure 11: 16-bit adder by rippling in the carry from Carry Look Ahead Adders

(iv) Speed Comparison :

Delay for 16-bit carry look ahead adder : 5.243 ns (0.745 ns logic, 4.498 ns route)
File used to calculate : `carry_look_ahead_16bit.v`
Critical Path : We get all the P (block propagate) and G(block generate) signals from all the blocks simultaneously and by this time sum has also been calculated by all the individual blocks, after this step we compute our output carry i.e. C_{out} as $G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$, which takes the highest amount of time. Therefore this is the critical path in this circuit.

Delay for 16-bit ripple carry adder : 11.236 ns (1.985 ns logic, 9.251 ns route)
File used to calculate : `ripple_carry_adder_16bit.v`
Critical Path : The path which is used for calculating both the sum and carry bits is the same as the path by which carry bit travels between subsequent full adders. Since calculation of one carry bit is dependent on the input of previous carry bit. This path takes the highest amount of time and hence is critical.

Lookup Table Cost Comparison :

LUT cost for 16-bit carry look ahead adder : 43
LUT cost for 16-bit ripple carry adder : 32
Reason : Lookup table is a customized truth table which is loaded with values that are relevant to our FPGA, based on your specific needs and instructions. We can even think of your custom Lookup Table or LUT as a small piece of RAM that is loaded whenever you power up your FPGA chip. As we can observe from the theory and figures in the report that CLAs have higher number of logic gates and require more expensive operations to calculate the carry bits. But because the carry bits are being precomputed and not rippled, it works faster. Because of this reason we require a higher number of LUTs in a CLA.

Appendix A

Adder Circuit	Total Delay	Logic Delay	Route Delay	LUT Cost
`ripple_carry_adder_4bit.v`	3.244 ns	0.497 ns	2.747 ns	8
`ripple_carry_adder_8bit.v`	5.908 ns	0.993 ns	4.915 ns	16
`ripple_carry_adder_16bit.v`	11.236 ns	1.985 ns	9.251 ns	32
`ripple_carry_adder_32bit.v`	21.892 ns	3.969 ns	17.923 ns	64
`ripple_carry_adder_64bit.v`	43.204 ns	7.937 ns	35.267 ns	128
`carry_look_ahead_4bit.v`	3.209 ns	0.373 ns	2.836 ns	6
`carry_look_ahead_4bit_aug.v`	2.123 ns	0.249 ns	1.874 ns	10
`carry_look_ahead_16bit.v`	5.243 ns	0.745 ns	4.498 ns	43
`carry_look_ahead_16bit_ripple.v`	6.167 ns	0.993 ns	5.174 ns	24