Expt. No: 2 Hariharan A
Date: 07.02.2023 203002034

Design and Synthesis of Ripple Carry Adders

Aim:

- To model a 4-bit Ripple Carry Adder and an 8-bit Ripple Carry Adder using structural modeling.
- To compile, simulate and plot the results using Xilinx ISE Tools.
- To implement the proposed systems using Xilinx Tools and generate the synthesis report.

Software used:

Xilinx ISE Tools

Functional Description:

Full Adder:

Truth Table

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

Boolean Equation:

Inputs: A, B & Cin

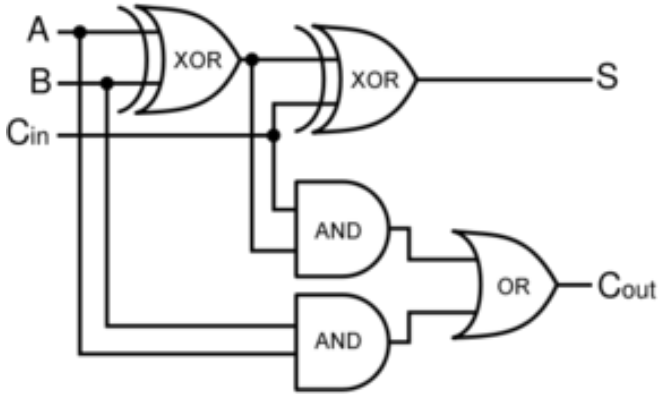
Outputs: Sum: A ^ B ^ Cin

Carry: (A & B) || (B & Cin) || (Cin & A)

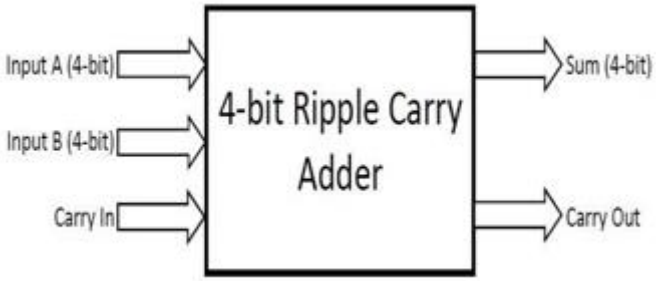
Hariharan A
203002034

Full Adder:

Logic Diagram:



4-Bit Ripple Carry Adder:

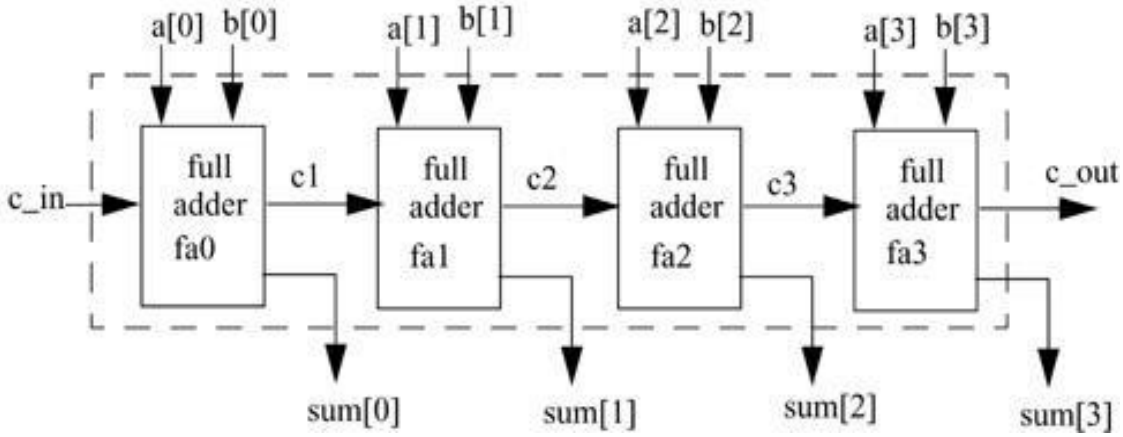


Hariharan A
203002034

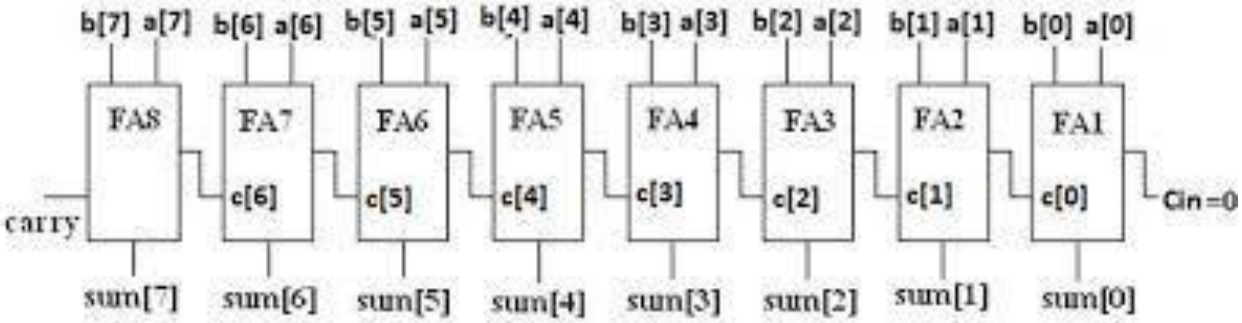
8-Bit Ripple Carry Adder:

Schematic Diagram:

4-Bit Ripple Carry Adder:



8-Bit Ripple Carry Adder:



Hariharan A
203002034

Modelling using Verilog HDL:
4-Bit Ripple Carry Adder:
Design module:

```
module RCA(sum,cout,a,b,cin); output [3:0] sum;
output cout; input [3:0] a,b; input cin;
wire c1,c2,c3;
fulladd f1(sum[0],c1,a[0],b[0],cin);
fulladd f2(sum[1],c2,a[1],b[1],c1);
fulladd f3(sum[2],c3,a[2],b[2],c2);

fulladd f4(sum[3],cout,a[3],b[3],c3);
endmodule

module fulladd(sum,cout,a,b,c); output sum,cout;
input a,b,c;
assign sum=a^b^c;
assign cout=(a&b)|(b&c)|(c&a);
endmodule
```

Stimulus file:

```
module stimulus; reg [3:0] A, B;
reg CIN;
wire [3:0] SUM; wire COUT;
RCA RippleCarryAdder(SUM, COUT,A,B,CIN);
Initial
Begin
$monitor($time," A= %b, B=%b, CIN= %b, COUT= %b, SUM= %b", A, B, CIN,
COUT, SUM);
end

Initial
Begin
A = 4'd0; B = 4'd0; CIN = 1'b0;
#10 A = 4'd3; B = 4'd4;
#10 A = 4'd2; B = 4'd5;
#10 A = 4'd9; B = 4'd9;
```

Hariharan A
203002034

```
#10 A = 4'd10; B= 4'd5;
#10 A = 4'd10; B = 4'd5; CIN = 1'b1;
end
endmodule
```

8-Bit Ripple Carry Adder:

Design module:

```
module RippleCarryAdder(sum,cout,a,b,cin);
output [7:0] sum;
Output cout;
Input[7:0] a,b;
wire c1,c2,c3,c4,c5,c6,c7;
fulladd f1(sum[0],c1,a[0],b[0],cin);
fulladd f2(sum[1],c2,a[1],b[1],c1);
fulladd f3(sum[2],c3,a[2],b[2],c2);
fulladd f4(sum[3],c4,a[3],b[3],c3);
fulladd f5(sum[4],c5,a[4],b[4],c4);
fulladd f6(sum[5],c6,a[5],b[5],c5);
fulladd f7(sum[6],c7,a[6],b[6],c6);
fulladd f8(sum[7],cout,a[7],b[7],c7);
endmodule

module fulladd (sum,cout,a,b,c);
output sum,cout;
input a,b,c;
assign sum=a^b^c;
assign cout=(a&b)|(b&c)|(c&a);
endmodule
```

Stimulus File:

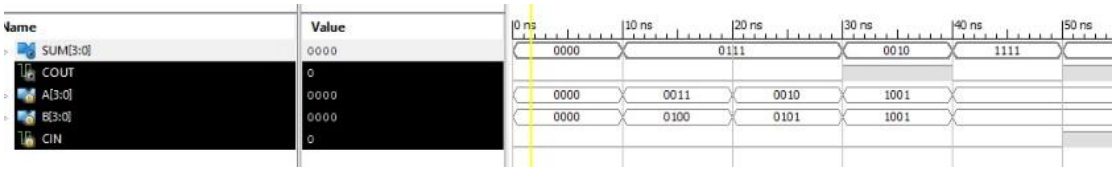
```
module stimulus;
reg [7:0] A, B;
reg C_IN;
```

Hariharan A
203002034

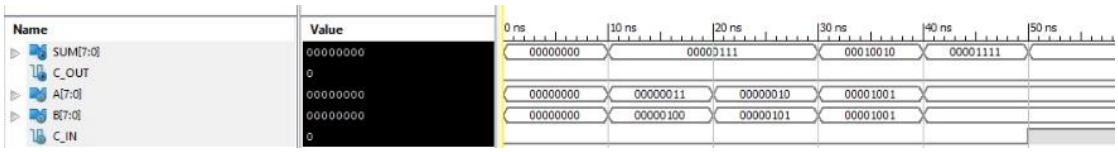
```
wire [7:0] SUM; wire C_OUT;
RippleCarryAdder FA1_4(SUM, C_OUT, A, B, C_IN);
initial
begin
$monitor($time," A= %b,B=%b,C_IN= %b,C_OUT= %b, SUM= %b",A, B, C_IN,
C_OUT, SUM);
end

Initial
begin
A = 8'd0; B = 8'd0; C_IN = 1'b0;
#10 A = 8'd3; B = 8'd4;
#10 A = 8'd2; B = 8'd5;
#10 A = 8'd9; B = 8'd9;
#10 A = 8'd10; B= 8'd5;
#10 A = 8'd10; B = 8'd5; C_IN = 1'b1;
end
endmodule
```

Simulation Results:

4-Bit Ripple Carry Adder:



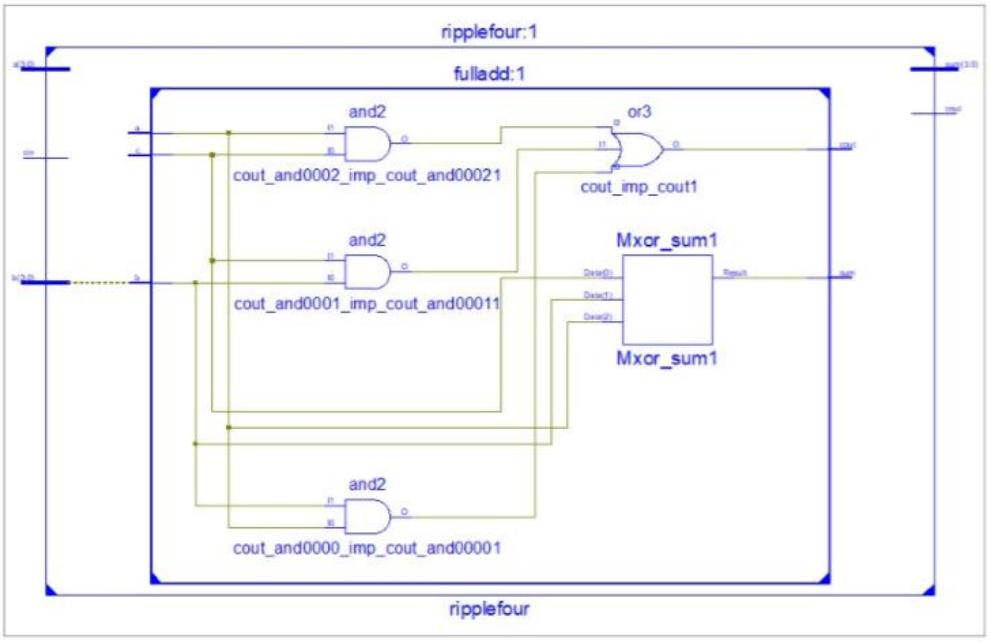
8-Bit Ripple Carry Adder:



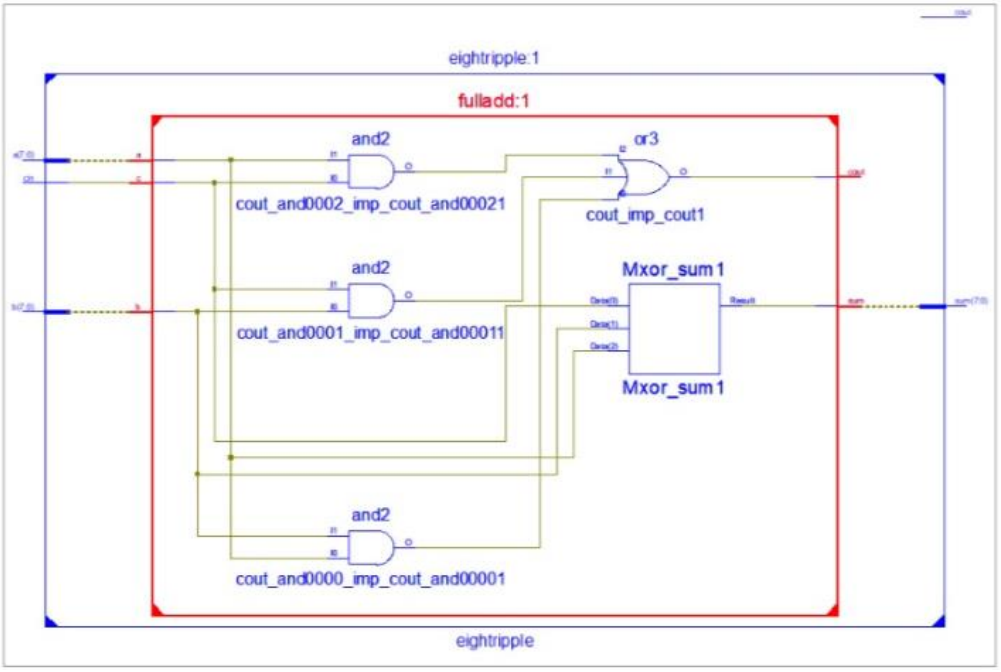
Hariharan A
203002034

RTL Schematic Diagram:

4-Bit Ripple Carry Adder:



8-Bit Ripple Carry Adder:



Hariharan A
203002034

Design Summary:

4-Bit Ripple Carry Adder:

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	8	4,896	1%
Number of occupied Slices	6	2,448	1%
Number of Slices containing only related logic	6	6	100%
Number of Slices containing unrelated logic	0	6	0%
Total Number of 4 input LUTs	8	4,896	1%
Number of bonded IOBs	14	158	8%
Average Fanout of Non-Clock Nets	1.71		

8-bit Ripple Carry Adder:

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	16	4,896	1%
Number of occupied Slices	12	2,448	1%
Number of Slices containing only related logic	12	12	100%
Number of Slices containing unrelated logic	0	12	0%
Total Number of 4 input LUTs	16	4,896	1%
Number of bonded IOBs	26	158	16%
Average Fanout of Non-Clock Nets	1.73		

Result:

Thus, a model for Ripple Carry Adder using Structural modelling was compiled, simulated, synthesized, and implemented.