Hierarchcial Design

By Jake D. Karas

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021

Objective:

The objective of this lab was to understand and apply hierarchical logic design to generate a large circuit from "blocks" of smaller logic circuits. The design prompt this was tested with in this lab stated to create a four-bit binary number adder from four full adders.

Equipment Used:

- Multisim Software
- Verilog Modelsim Software

Methodology:

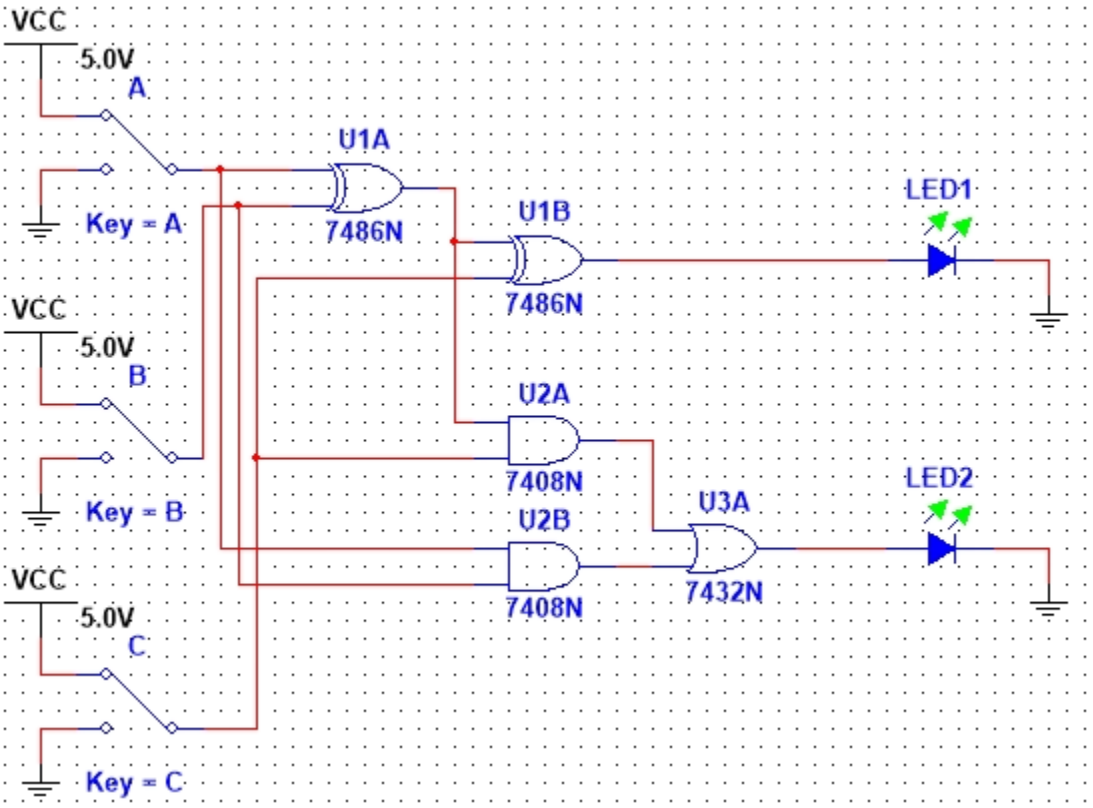
First, the truth table for both the sum and carry outputs a full adder was derived based on the circuit description. Using this truth table, K-Maps were developed and used to obtain the simplified equations for each output. This circuit was then drawn in Multisim to pre-test each truth table.

With the equations confirmed by testing the truth tables, the full adder was implemented in Modelsim via a Verilog program file utilizing the boolean equation programming method. A testbench file, along with a testvector, was then generated to test all input combinations for this circuit. After compiling all three files, a simulation was run, with the position of each input and output wave for each time period denoting a logic 0 or 1. Like the previous lab, a low position denoted a logic 0 while a high position implied a logic 1.

The hierarchical portion of this lab was taking the full adder circuit and utilizing four of them to make a 4-bit binary number adder. The easiest method of chaining four of these together in Modelsim was to treat the full adder as a user-defined function and connect the inputs and outputs correctly utilizing gate-level modeling. Another testbench, this time utilizing the monitor function rather than a testvector, was then created, and a simulation was performed in the same manner as before, except this time, there were four output waves instead of two.

Multisim Circuit Diagram:

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021



Truth Tables and K-Maps:

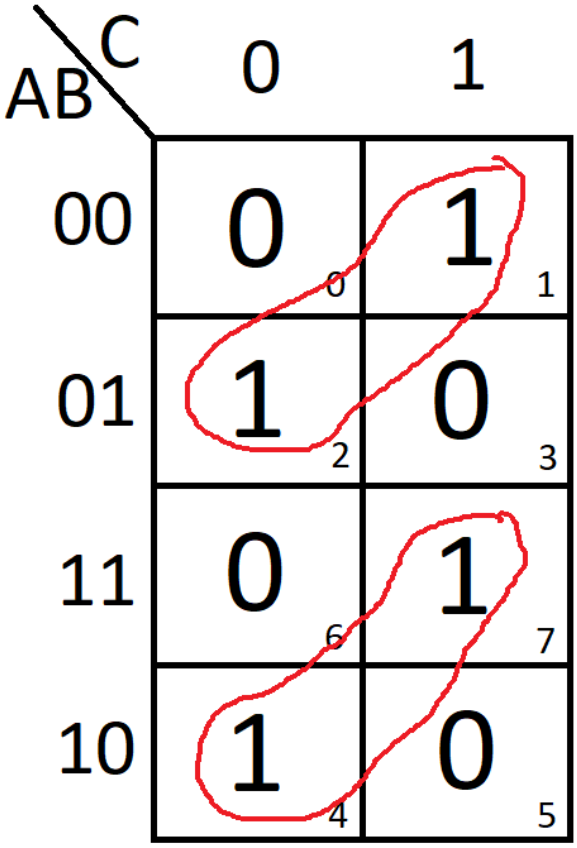
Sum Bit:

Truth Table:

A	B	C_{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

K-Map:

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021



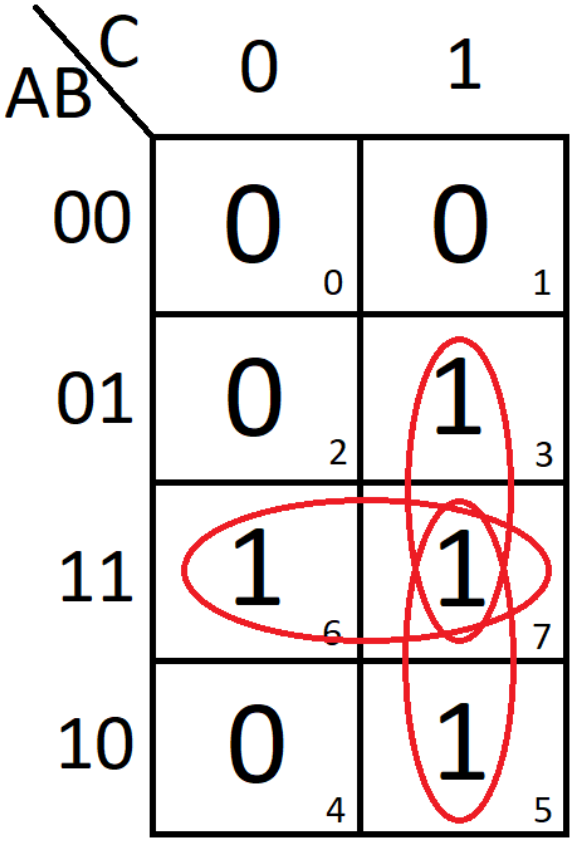
$$S = A'(B \oplus C) + A(B \oplus C) = A \oplus B \oplus C$$

Carry Bit:
Truth Table:

A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

K-Map:

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021



$$C_{out} = AB + AC_{in} + BC_{in} = (A \oplus B)C_{in} + AB$$

ModelSim Code:

Full Adder Main Code File:

```
module Full_Adder (a, b, c_in, sum, c_out);
input a, b, c_in;
output sum, c_out;

assign sum = (a ^ b) ^ c_in;
assign c_out = (c_in & (a ^ b)) | (b & a);
endmodule
```

Full Adder Testbench:

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021

```
module Full_Adder_tb();
reg a, b, c_in;
wire sum, c_out;

reg [2:0] testvectors [7:0];
integer i;

Full_Adder r1(.a(a), .b(b), .c_in(c_in), .sum(sum), .c_out(c_out));

initial begin
        $readmemb ("C:/Users/Jake/Documents/ModelSim/Lab 5/Full_Adder_Testvector.tv", testvectors);
end
always begin

for (i=0; i<8; i=i+1) begin
        {a, b, c_in} = testvectors[i];
        #10;
end
end
endmodule
```

Full Adder Testvector:

```
000
001
010
011
100
101
110
111
```

Four Bit Adder Main Code File:

```
module Four_Bit_Adder (A0, A1, A2, A3, B0, B1, B2, B3, Cin, S0, S1, S2, S3, Cout);
input A0, A1, A2, A3, B0, B1, B2, B3, Cin;
output S0, S1, S2, S3, Cout;
wire C0, C1, C2;

Full_Adder FA0 (A0, B0, Cin, S0, C0);
Full_Adder FA1 (A1, B1, C0, S1, C1);
Full_Adder FA2 (A2, B2, C1, S2, C2);
Full_Adder FA3 (A3, B3, C2, S3, Cout);
endmodule
```

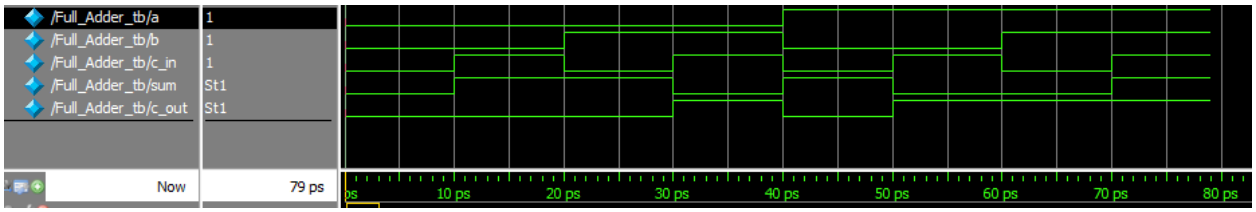
Four Bit Adder Testbench:

```
module Four_Bit_Adder_tb();
reg A0_in, A1_in, A2_in, A3_in, B0_in, B1_in, B2_in, B3_in, Cin_in;
wire S0_out, S1_out, S2_out, S3_out, Cout_out;

initial
begin
$monitor("simtime=%g, A0_in=%b, A1_in=%b, A2_in=%b, A3_in=%b, B0_in=%b, B1_in=%b, B2_in=%b, B3_in=%b, Cin_in=%b, S0_out=%b, S1_out=%b, S2_out=%b, S3_out=%b, Cout_out=%b",
$time, A0_in, A1_in, A2_in, A3_in, B0_in, B1_in, B2_in, B3_in, Cin_in, S0_out, S1_out, S2_out, S3_out, Cout_out);
end

Four_Bit_Adder uut (.A0(A0_in), .A1(A1_in), .A2(A2_in), .A3(A3_in), .B0(B0_in), .B1(B1_in), .B2(B2_in), .B3(B3_in), .Cin(Cin_in),
.S0(S0_out), .S1(S1_out), .S2(S2_out), .S3(S3_out), .Cout(Cout_out));

initial
begin

#0 A0_in= 1'b0; A1_in= 1'b0; A2_in= 1'b0; A3_in= 1'b0; B0_in= 1'b0; B1_in= 1'b0; B2_in= 1'b0; B3_in= 1'b0; Cin_in= 1'b0; //0000 + 0000
#10 A0_in= 1'b1; A1_in= 1'b0; A2_in= 1'b1; A3_in= 1'b0; B0_in= 1'b1; B1_in= 1'b0; B2_in= 1'b1; B3_in= 1'b0; Cin_in= 1'b0; //0101 + 0101
#10 A0_in= 1'b1; A1_in= 1'b1; A2_in= 1'b1; A3_in= 1'b0; B0_in= 1'b1; B1_in= 1'b0; B2_in= 1'b0; B3_in= 1'b1; Cin_in= 1'b0; //0111 + 1001
#10 A0_in= 1'b1; A1_in= 1'b0; A2_in= 1'b0; A3_in= 1'b1; B0_in= 1'b1; B1_in= 1'b0; B2_in= 1'b1; B3_in= 1'b1; Cin_in= 1'b0; //1001 + 1101
#10 A0_in= 1'b1; A1_in= 1'b1; A2_in= 1'b1; A3_in= 1'b1; B0_in= 1'b1; B1_in= 1'b1; B2_in= 1'b1; B3_in= 1'b1; Cin_in= 1'b0; //1111 + 1111

end
endmodule
```

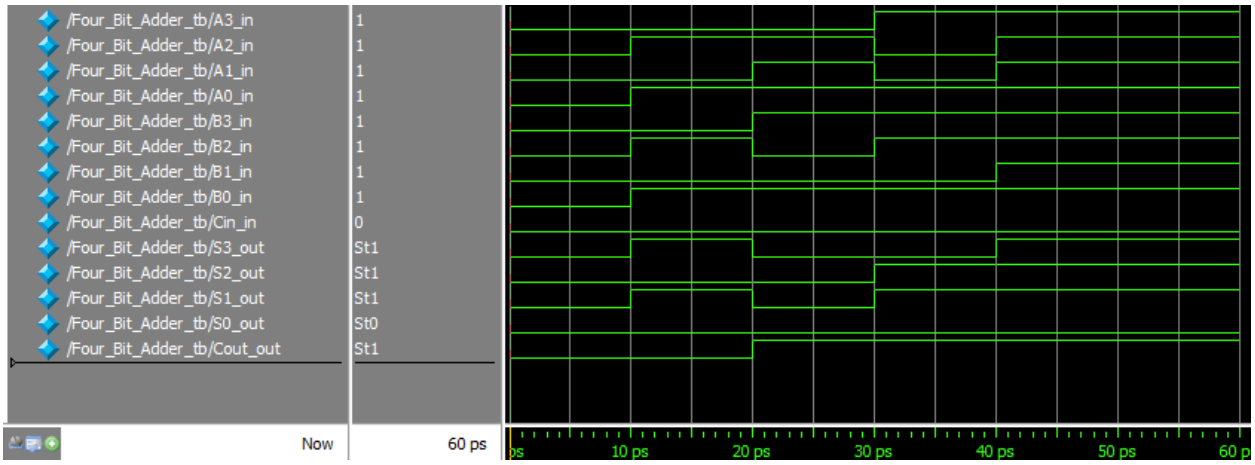
ModelSim Output Waves:

Full Adder:

Hierarchical Design
Jake D. Karas
U0000008780
10/21/2021



Four Bit Adder:



Conclusion:

Hierarchical logic design is the process of combining “blocks” of small, simple circuits to create a larger and more complex one. Utilizing this approach can save time and help break a logic circuit up into a sum of its components rather than just be a ton of logic gates filling up an entire paper. As shown above, utilizing this approach still yields correct results, although it is much easier to utilize hierarchical design than straight-up attempting to connect many logic gates together.