ECE472 Lab 1

Ashtyn Moehlenhoff
Torben Rasmussen

Part 1: 16 bit ripple adder

Desciription: In order to implement the 16 bit ripple adder, we simply added on to the 8 bit ripple adder. We increased our array sizes to 16 bits and connected 8 more full adders. You can see in figure 1 that the sum of A and B (for a small sample) is correct.

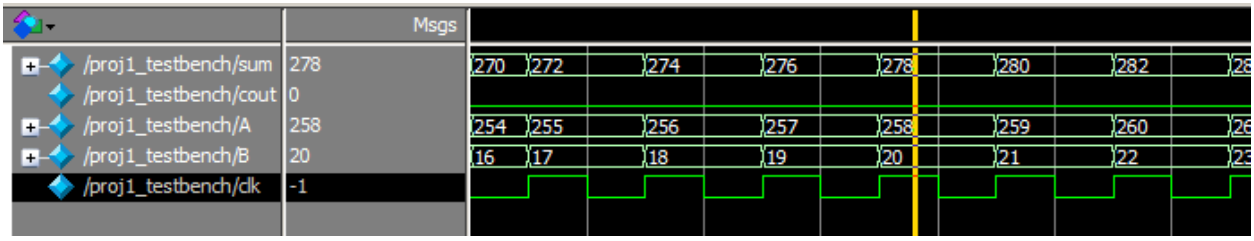

Figure 1: 16 bit ripple adder

Part 2: 16 bit carry-lookahead adder

Desciription: First we got a 4 bit carry-lookahead adder working by making a lookahead module and a fulladder_LA module. We used both these modules in a ripple_adder_LA module to calculate the sum of two 4 bit numbers. In order to create a 16 bit CLA we just strung together four 4 bit CLAs (connecting their carry-in and carry-outs). Figure 2 is a simulation of our 16 bit CLA which adds A and B, you can see the correct sums being calculated for a small sample size.

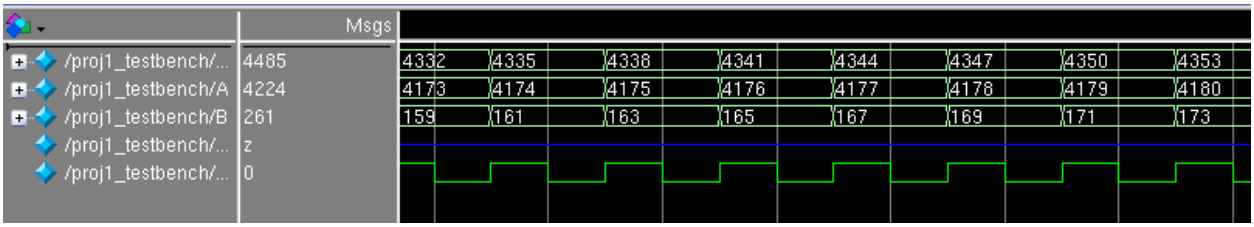

Figure 2: 16 bit carry-lookahead adder

Part 3: Simulating with delay

Description: You can see in Figure 3 and Figure 4 that it takes more time to calculate the sum of A and B (because we added in a delay for each logic function). We have circled the sum of A and B in red in the diagrams. We displayed the worst case for A and B, which is when A = 0xFFFF. This is worst case because it will generate a carry on the addition of each bit.

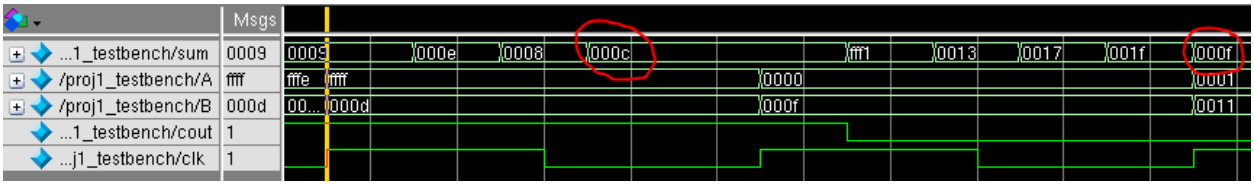

Figure 3: 16 bit ripple adder with delays

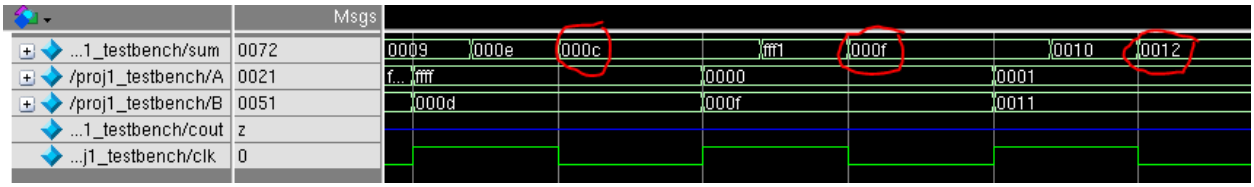

Figure 4: 16 bit carry-lookahead adder with delays

Part4: Reflection

Estimated Time: 6hours

Most Valuable: Getting familiar with Verilog syntax and Modelsim.

Least Valuable: Debugging Modelsim, dealing with crashes in Modelsim on the lab computers, etc…

Suggestions: More information on defining inputs and outputs for each function so we would have had a more clear idea of what we were doing.

```
module fulladder(a, b, cin, sum, cout);
input a, b, cin;
output sum, cout;
assign #2 sum = a ^ b ^ cin;
assign #2 cout = a & b | a & cin | b & cin;
endmodule


module ripple_adder(a, b, sum, cout);
input [15:0] a, b;
output [15:0] sum;
output cout;
wire [15:0] c; // used for carry connections
assign c[0]=0;
fulladder f0(a[0], b[0], c[0], sum[0], c[1]);
fulladder f1(a[1], b[1], c[1], sum[1], c[2]);
fulladder f2(a[2], b[2], c[2], sum[2], c[3]);
fulladder f3(a[3], b[3], c[3], sum[3], c[4]);
fulladder f4(a[4], b[4], c[4], sum[4], c[5]);
fulladder f5(a[5], b[5], c[5], sum[5], c[6]);
fulladder f6(a[6], b[6], c[6], sum[6], c[7]);
fulladder f7(a[7], b[7], c[7], sum[7], c[8]);
fulladder f8(a[8], b[8], c[8], sum[8], c[9]);
fulladder f9(a[9], b[9], c[9], sum[9], c[10]);
fulladder f10(a[10], b[10], c[10], sum[10], c[11]);
fulladder f11(a[11], b[11], c[11], sum[11], c[12]);
fulladder f12(a[12], b[12], c[12], sum[12], c[13]);
fulladder f13(a[13], b[13], c[13], sum[13], c[14]);
fulladder f14(a[14], b[14], c[14], sum[14], c[15]);
fulladder f15(a[15], b[15], c[15], sum[15], cout);
endmodule

module proj1_testbench;
wire [15:0] sum;
reg [15:0] A, B;
wire cout;
reg clk;
// instantiate a ripple_adder (DUT = Device Under Test)
//ripple_adder_LA DUT(A, B, sum);
add16_LA DUT(A, B, sum);
//ripple_adder DUT(A, B, sum, cout);
// Generate a clock that changes ever 5 time units (a period of 10)
always
#5 clk = ~clk;
// Initialize signals
initial begin
clk = 1'b0; //initalize clk value to 0 at t=0
A = 16'hFFF9; // initialize input A to 00000000
B = 16'h0001; // initialize input B to 00000000
end
// add one to the value of A at each positive clock edge
always @(posedge clk)
A = A + 1;
always @(posedge clk)
B = B + 2;
endmodule


module ripple_adder_LA(a, b, cin, cout, sum);
input [3:0] a, b;
input cin;
output cout;
output [3:0] sum;
wire [4:0] c; // used for carry connections
wire [3:0] g;
wire [3:0] p;
wire P,G;
assign c[0]=cin;
fulladder_LA f0(a[0], b[0], c[0], sum[0], p[0], g[0]);
```



```
fulladder_LA f1(a[1], b[1], c[1], sum[1], p[1], g[1]);
fulladder_LA f2(a[2], b[2], c[2], sum[2], p[2], g[2]);
fulladder_LA f3(a[3], b[3], c[3], sum[3], p[3], g[3]);

lookahead f4(p[3],p[2],p[1],p[0],g[3],g[2],g[1],g[0],c[0],c[1],c[2],c[3],c[4],P,
 G);
assign cout = c[4];
endmodule

module fulladder_LA(a, b, cin, sum, p, g);
input a, b, cin;
output sum, p, g;
assign #2 sum = a ^ b ^ cin;
assign #1 p = a | b;
assign #1 g = a & b;

endmodule

module lookahead(p3,p2,p1,p0,g3,g2,g1,g0,c0,c1,c2,c3,c4,P,G);
input p3,p2,p1,p0,g3,g2,g1,g0,c0;
output c1,c2,c3,c4,P,G;

assign #2 c4 = g3 | (p3 & g2) | (p3 & p2 & g1) | (p3 & p2 & p1 & g0) | (p3 & p2
& p1 & p0 & c0);
assign #2 c3 = g2 | (p2 & g1) | (p2 & p1 & g0) | (p2 & p1 & p0 & c0);
assign #2 c2 = g1 | (p1 & g0) | (p1 & p0 & c0);
assign #2 c1 = g0 | (p0 & c0);
assign #1 P = p3 & p2 & p1 & p0;
assign #1 G = g3 & g2 & g1 & g0;


endmodule

module add16_LA(a, b, sum);
input [15:0] a, b;
output [15:0] sum;
wire [15:0] temp_sum;
wire [3:0] sum0;
wire [3:0] sum1;
wire [3:0] sum2;
wire [3:0] sum3;

wire carry0;
wire carry1;
wire carry2;
wire carry3;

ripple_adder_LA f0(a[3:0], b[3:0], 0, carry0, sum0);
ripple_adder_LA f1(a[7:4], b[7:4], carry0, carry1, sum1);
ripple_adder_LA f2(a[11:8], b[11:8], carry1, carry2, sum2);
ripple_adder_LA f3(a[15:12], b[15:12], carry2, carry3, sum3);

assign sum = {sum3,sum2,sum1,sum0};

endmodule
```