Ajeet Kokatay
862083784

Lab 5 Carry Lookahead Adder

No partner

Lab Report

Overview
In this lab we created a carry lookahead adder in verilog code.

New Concepts
We learned about the concept of a register.

Discussion:
The hardest part was generating the boolean expression for the carry lookahead adder. The rest was pretty simple. I found it interesting that we can use XOR to create an adding module.

Conclusion
The purpose of this part of the lab was to expand our knowledge of carry lookahead adders.

Questions

Question 1

No, the result can always be represented with 4 bits and a carry out bit. If this was not the case then we could not chain together adding modules. The carry out bit simply tells us that there is an extra carry digit for the next module to consider when doing arithmetic.

Question 2

Our 4 bit lookahead adder has less gate delay since all the carry digits are calculated at once.

Question 3

Verilog Code

Carry lookahead module (cla.v)

```
`timescale 1ns / 1ps

module cla(
    output [3:0] cout,
    input cin,
    input [3:0] x,
   input [3:0] y             );
    // Computing all gx
    wire g0, g1, g2, g3 ;
    assign g0 = x[0] & y[0] ;
    assign g1 = x[1] & y[1] ;
    assign g2 = x[2] & y[2] ;
   assign g3 = x[3] & y[3] ;
    // Computing all px
    wire p0, p1, p2, p3 ;
   assign p0 = x[0] + y[0] ;
    assign p1 = x[1] + y[1] ;
    assign p2 = x[2] + y[2] ;
    assign p3 = x[3] + y[3] ;
    // Computing all carries , your code here
    assign cout[0] = g0 | (p0 & cin);
    assign cout[1] = g1 | (p1 & cout[0]);
    assign cout[2] = g2 | (p2 & cout[1]);
    assign cout[3] = g3 | (p3 & cout[2]);
endmodule
```

Adding module (addlogic.v)

```
`timescale 1ns / 1ps
module addlogic(output r,      input x,           input y,           input cin );
  assign r= x ^ y ^ cin;
endmodule
```

Full carry lookahead module

```
module carrylookahead_st( input clk , input enable , input cin, input [3:0] x, input [3:0] y, output
cout, output [3:0] r);
    wire [3:0] c;   wire [3:0] ir1 ;           wire [4:0] ir2 ;
    // Compute Carries
    cla cx1 (c, cin, x, y);
    // Compute output ir1
    addlogic cx6 (ir1[0], x[0], y[0], cin);
    addlogic cx7 (ir1[1], x[1], y[1], c[0]) ;
    addlogic cx8 (ir1[2], x[2], y[2], c[1]) ;
    addlogic cx9 (ir1[3], x[3], y[3], c[2]) ;
    // Register
    assign ir2 = {c[3],ir1};
    // Results
    assign r = ir2[3:0] ;
    assign cout = ir2[4] ;

endmodule
```

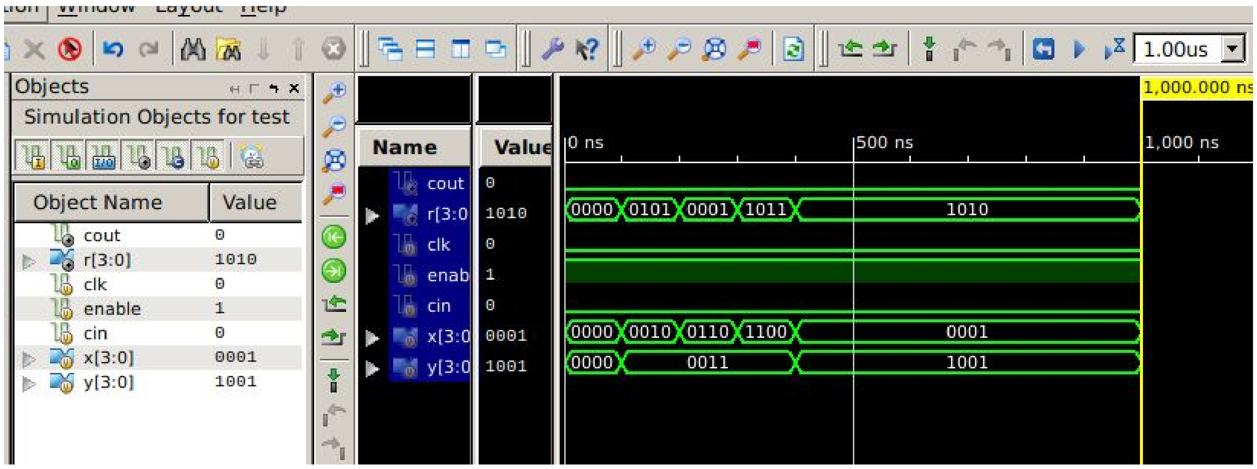
Verilog test bench

```
`timescale 1ns / 1ps
module test;

    // Inputs
    reg clk;
    reg enable;
    reg cin;
    reg [3:0] x;
    reg [3:0] y;

    // Outputs
    wire cout;
    wire [3:0] r;

    // Instantiate the Unit Under Test (UUT)
    carrylookahead_st uut (
            .clk(clk),
            .enable(enable),
            .cin(cin),
            .x(x),
            .y(y),
            .cout(cout),
            .r(r)
    );

    initial begin
            // Initialize Inputs2
            clk = 0;
            enable = 1;
            cin = 0;
            x = 0;
            y = 0;

            // Wait 100 ns for global reset to finish
            #100;
```

```
        // Add stimulus here
        x= 4'd2;
        y= 4'd3;
        #100;
        if(r != 4'd5) $display("Result is wrong 2 + 3 = %d", r);

        x= 4'd6;
        y= 4'd3;
        #100;
        if(r != 4'd9) $display("Result is wrong 6 + 3 = %d", r);

        x= 4'd12;
        y= 4'd3;
        #100;
        if(r != 4'd15) $display("Result is wrong 12 + 13 = %d", r);

        x= 4'd1;
        y= 4'd9;
        #100;
        if(r != 4'd10) $display("Result is wrong 1 + 9 = %d", r);
    end

endmodule
```

Question 4

Waveform screenshot



Window
Layout
Help
1.00us
Objects
Simulation Objects for test
Name
Value
0 ns
500 ns
1,000 ns
1,000.000 ns
cout 0
r[3:0] 1010 — 0000, 0101, 0001, 1011, 1010
clk 0
enab 1
cin 0
x[3:0] 0001 — 0000, 0010, 0110, 1100, 0001
y[3:0] 1001 — 0000, 0011, 1001

Object Name	Value
cout	0
r[3:0]	1010
clk	0
enable	1
cin	0
x[3:0]	0001
y[3:0]	1001