Diane Shan

862148900

EE120A Section 23

Jack Huang

Lab 5 – Datapath Components - Adders

**Overview**

In this lab, we looked at how to design a carrylookahead adder. We derived equations and computed values to figure out how to design it. Then we created and tested Full adder using structural Verilog. Our results were that the code runs according to the specifications. The design adds together numbers successfully by calculating the carry from the first set of inputs and then going to the next set of inputs and calculating that carry and so on. Using the functionality of the carrylookahead adder, there is no gate delay.

**New Concepts**

Ripple Carry Adder – logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder

FPGA – a field programmable gate array which is an integrated circuit designed to be configured by a customer or a designer after manufacturing

**Analysis**

*Procedure*:

1. Look at given equations for c1 and c2
2. Derive equations for c3 and c4
3. Compute all ci and then compute zi using zi = (xi^yi^ci)
4. Connect four full-adders to create a 4-bit adder
5. Simulate the system and observe the outputs
6. Try adding smaller numbers and then larger numbers
7. Create and test a Full adder using structural verilog
8. Then use the testbench to test all the code and demonstrate that application performs according to specs

Equations for c3 and c4:

c3 = g2 + p2c2 = g2 + p2(g1 + p1g0 + p1p0c0) = g2 + p2g1 + p2p1g0 + p2p1p0c0

c4 = g3 + p3c3 = g3 + p3(g2 + p2g1 + p2p1g0 + p2p1p0c0) =

g3 + p3g2 + p3p2p1g0 + p3p2p1p0c0

**Records**

*Schematics/Diagrams:*

Diagram

Description automatically generated with medium confidenceDiagram

Description automatically generated

*Code:*

design.sv

module falogic(

output r, // We label our output as r instead of z

input x,

input y,

input cin

);

wire t1;

xor cx1 ( t1, x,y );

xor cx2 ( r, t1, cin );

endmodule

module register\_logic(

input clk,

input enable ,

input [4:0] Data ,

output reg [4:0] Q ) ;

// on real FPGA board which has clk signal, we use the following always statement:

// always @(posedge clk )

// begin

// if ( enable) begin

// Q = Data;

// end

// end

// endmodule

// for simulation, we force the statement to execute without clk signal:

always @(\*) begin

if ( enable) begin

Q = Data;

end

end

endmodule

module carrylogic(

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

assign cout[0] = g0 | ( p0 & cin) ;

assign cout[1] = g1 | ( p1 & ( g0 | (p0 & cin) ) ) ;

assign cout[2] = g2 | ( p2 & ( g1 | ( p1 & ( g0 | (p0 & cin) ) ) ) ) ;

assign cout[3] = g3 | ( p3 & ( g2 | (p2 & ( g1 | ( p1 & ( g0 | (p0 & cin) ) ) ) ) )) ;

endmodule

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

carrylogic cx1 ( c, cin, x, y ) ;

// Compute R

falogic cx6 ( ir1[0], x[0], y[0], cin ) ;

falogic cx7 ( ir1[1], x[1], y[1], c[0] ) ;

falogic cx8 ( ir1[2], x[2], y[2], c[1] ) ;

falogic cx9 ( ir1[3], x[3], y[3], c[2] ) ;

// Register

register\_logic cx10 ( clk, enable, {c[3],ir1}, ir2 ) ;

// Results

assign r = ir2[3:0] ;

assign cout = ir2[4] ;

endmodule

testbench.sv

module carrylookahead\_tb;

// Inputs

reg cin;

reg [3:0] x;

reg [3:0] y;

reg clk;

// Outputs

wire cout;

wire [3:0] r;

reg [3:0] rx;

integer index ;

// Instantiate the Unit Under Test (UUT)

carrylookahead\_st uut (

.clk(clk),

.cin(cin),

.x(x),

.y(y),

.cout(cout),

.r(r)

);

initial begin

$dumpfile("dump.vcd"); $dumpvars;

// Initialize Inputs

cin = 'd0;

y = 'd0;

// r = x + 0 ; cout = 0;

$display("TC11 ");

for (index=0; index < 15; index = index + 1) begin

x = index ;

#100;

if ( r != x ) $display ("Result is wrong");

if ( cout != 1'b0 ) $display ("Result is wrong - Carryout ");

end

// r = x + 1 ;

cin = 1'b1;

y = 4'b0;

$display("TC12 ");

for (index=0; index < 15; index = index + 1) begin

x = index ;

#100;

if ( r != (x + 'd1) ) $display ("Result is wrong %b %b" , r, (x+1) );

if ( cout != 1'b0 ) $display ("Result is wrong -Carryout ");

end

// r = x + y + 1 ;

cin = 1'b1;

$display("TC13 ");

for (index=0; index < 8; index = index + 1) begin

x = index ;

y = index ;

#100;

if ( r != (x + y +1 ) ) $display ("Result isw rong %b %b" , r, (x+y) );

if ( cout != 1'b0 ) $display ("Result is wrong -Carryout ");

end

// r = x + y + 1 ;

cin = 1'b1;

$display("TC14 ");

for (index=8; index < 16; index = index + 1) begin

x = index ;

y = index ;

rx = x + y + cin;

#100;

if ( r != rx ) $display ("Result is wrong %b %b" , r,rx );

if ( cout != 1'b1 ) $display ("Result is wrong -Carryout ");

end

end

endmodule

*Waveform*:

A computer screen capture

Description automatically generated with medium confidence

**Discussion**

The system works according to the provided specifications. It passed all the tests in the testbench and there weren’t any problems encountered. It did take some more work to figure out how to design the adder but there were no issues that resulted in system resign or big modifications. There are not really any possible ways to improve the system as the system was designed to reduce gate delay and it simply adds numbers together.

**Conclusion**

The purpose of this lab was to learn about another different type of circuit and also to learn about the design of adders, synthesis, and implementation. We had to design an adder in this lab and also learned about the design of special purpose registers. We implemented a four-bit adder with a carry look ahead to reduce the gate delay and was able to make it work with verilog and made sure it added all the numbers correctly with extensive testing.

**Questions**

Is it possible for two 4-bit numbers and a carry-in to result in a number too big to represent using 4 sum bits and a carry-out bit?

* It is not possible for two 4-bit numbers and a carry-in to result in a number too big to represent using 4 sum bits and a carry-out bit because the biggest number would be 1111 + 1111 which would result in a carry of 1 and that equals 11110 which is a 4-bit number plus the carry.