LAB 6:

Carry Select Adder

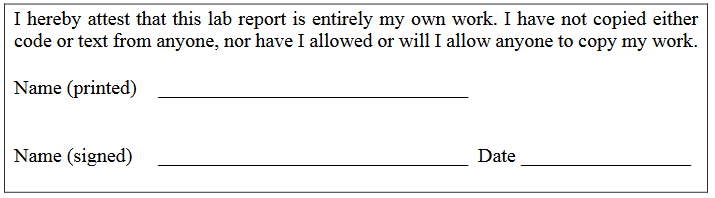
Professor Ronald Mehler

ECE 526L

Spring 2018

Garen Nikoyan

3/8/2018

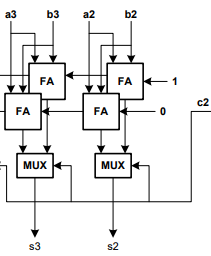


**Objective:**

The purpose of this lab is to become familiar with using loops to efficiently write a huge exhaustive test vector. Along with this, the use of the force function is used to force an error, in order to test if the test bench will properly find an error. And lastly, the use of `ifdef is learned in order to easily change if the forced error is applied or not.

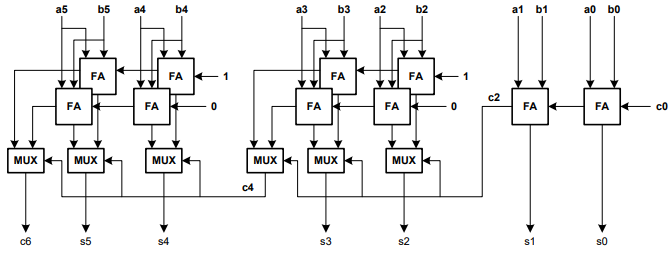
**Methodology:**

For this lab, the first step was to create a module for a behavioral full adder. It was made using the concatenation operator to properly assign the bits to the correct outputs. Next, a 2 input 1 output multiplexer was made by modifying the scalable multiplexer from Lab 5 to only accept one bit. Using the full adder and mux modules, a medium level module was creating which following the schematic below.



*Figure 1: Medium level module*

By instantiating this medium level module and the multiplexer module, a full 6-bit Carry Select Adder module was created, which follows the schematic below.



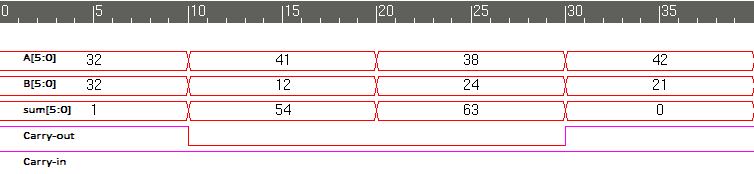
*Figure 2: Full 6-bit Carry Select Adder*

Lastly, the test benches were created. For this experiment, two test benches were created, a non-exhaustive and exhaustive version. The non-exhaustive test bench was kept short by testing only eight different cases. First it sets carry-in to 0, and checks if carry-out of the CSA will properly go to 0, to 1, and then it will test if the sum will properly go to all 0s, and then all 1s. The same tests were done with carry-in set to 1, with the vectors altered to accommodate the extra 1. This test method is adequate because it tests each bit of the sum and carry-out to see if it properly changes with the different inputs. The exhaustive test bench was done by using for loops to cycle through all possible combinations of A, B, and carry-in. If statements were used to check for errors in the outputs. A behavioral 6-bit adder module was created to compare the values to. Force statements were used to force the output of this behavioral adder to an incorrect value to check if the error checking works correctly. In order to easily enable and disable the force statement, `ifdef was used. By using `ifdef, the force statement could be enabled or disabled by simply commenting out a line at the top of the module, instead of having to dig through the module to find the force statement and comment it out.

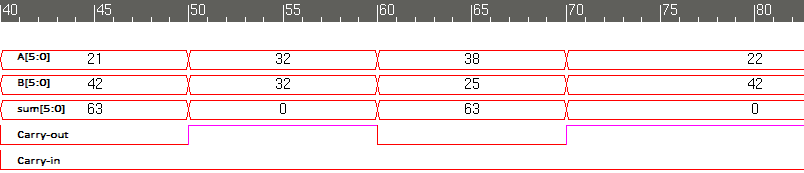
**Analysis:**

Looking at waveforms and the log, it can be seen that the modules worked correctly, as no error is triggered until the forced error is encountered. The non-exhaustive test bench uses different input values in each test vector. This is done to ensure that everything is working properly, and the correct output is not just a result of a bit carrying over from the previous test case. The exhaustive test bench also worked properly, not finding an error until the force statement was used to force an error into the behavioral adder’s output.

In Figures 3 and 4, the waveforms of the non-exhaustive test bench can be found. It works properly in both carry-in=0 and carry-in=1 cases. The sum output also properly cycles through 0s and 1s.

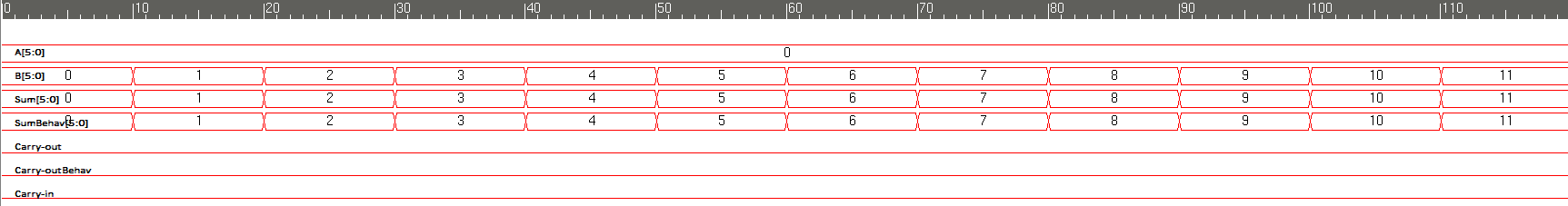


*Figure 3: Non-exhaustive testbench with Carry-in=1*

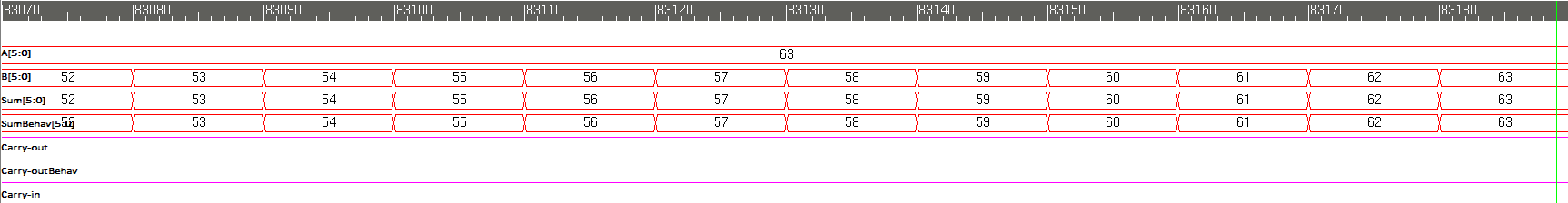
**

*Figure 4: Non-exhaustive testbench with Carry-in=0*

In Figure 5 and 6, the first set of 12 and last set of 12 waveforms of the No Error Exhaustive test bench can be seen, respectively. The only notable cases not tested by this test vector is a case where both inputs and carry-in are all 0, or all 1. Only the outputs were taken into account when thinking of test vectors.

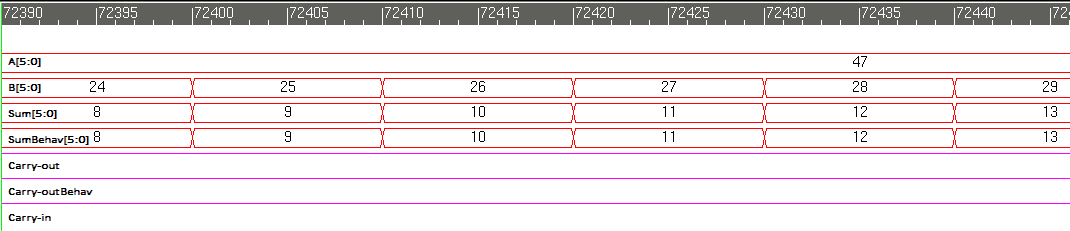


*Figure 5: No Error Exhaustive test bench first set of 12 waveforms*

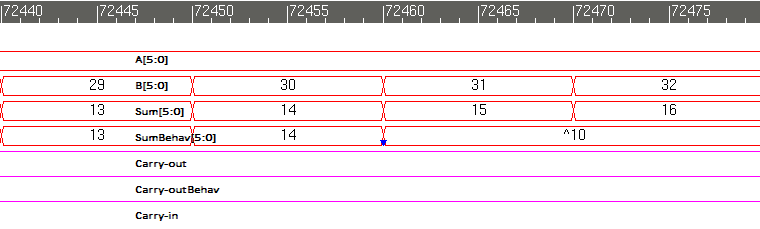
**

*Figure 6: No Error Exhaustive test bench last set of 12 waveforms*

In Figures 7 and 8, the waveforms of the exhaustive test bench with the forced error is shown. As can be seen in Figure 7, the adder is working correctly before the error is forced. The vectors immediately before the error are also shown in the log. In Figure 8, at around time 72460ns, the error is forced, and as can be seen in the log, the error is displayed and the test bench is stopped properly using $finish.

**

*Figure 7: Exhaustive test bench with Error, waveforms before error*

**

*Figure 8: Exhaustive test bench with Error, showing where error is forced*

**Modules:**

**Carry Select Adder**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #6 Garen Nikoyan, Spring 2018

\*\*\* Carry Select Adder

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: Carry\_Sel\_Adder.v Created by: Garen Nikoyan, 3/8/2018 \*\*\*

\*\*\* -Revision History

\*\*\* 3/1/2018: First draft

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module models a a 6-bit Carry Select Adder \*\*\*

\*\*\* This module instantiates a full adder and multiplexer module, along

\*\*\* with a mid level module which combines the two into a 2 bit adder, to create a a full

\*\*\* 6 bit carry select adder with carry-in and carry-out

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns/1ns

module Carry\_Sel\_Adder(c6, sum, A, B, c0);

input [5:0] A, B;

input c0;

output [5:0] sum;

output c6;

Full\_Adder FA0(c0, A[0], B[0], c1, sum[0]);

Full\_Adder FA1(c1, A[1], B[1], c2, sum[1]);

Twobitadder TBA0(c2, A[3:2], B[3:2], co0, co1, sum[2], sum[3]);

mux mux0(co1, co0, c2, c4);

Twobitadder TBA1(c4, A[5:4], B[5:4], co2, co3, sum[4], sum[5]);

mux mux1(co3, co2, c4, c6);

endmodule

**Medium Level Module**

`timescale 1ns/1ns

module Twobitadder(cin, A, B, coT, coB, sum0, sum1);

input [1:0] A, B;

input cin;

output sum0, sum1;

output coT, coB;

Full\_Adder FA0(1'b1, A[0], B[0], co1, S0);

Full\_Adder FA1(co1, A[1], B[1], coT, S1);

Full\_Adder FA2(1'b0, A[0], B[0], co2, S2);

Full\_Adder FA3(co2, A[1], B[1], coB, S3);

mux mux0(S2, S0, cin, sum0);

mux mux1(S3, S1, cin, sum1);

endmodule

**Full Adder**

`timescale 1ns/1ns

module Full\_Adder(cin, A, B, cout, out);

input cin, A, B;

output out, cout;

assign {cout, out} = cin + A + B ;

endmodule

**2:1 Mux**

module mux(A, B, SEL, OUT);

output reg OUT;

input A, B;

input SEL;

always @\*

OUT = SEL ? B : A; // if SEL=1, OUT=B, if SEL=0, OUT=A

// if SEL=x, then A=B causes OUT=A=B, and A!=B, OUT=x

endmodule

**Behavioral Adder**

module BehavAdder(cin, A, B, coutb, outb);

reg [6:0] temp;

input cin;

input [5:0] A, B;

output coutb;

output [5:0] outb;

always @\* temp = cin + A + B ;

assign coutb = temp[6];

assign outb = temp[5:0];

endmodule

**Testbench:**

**Non-exhaustive Test Bench**

`timescale 1 ns / 1 ns

module NonExhTB();

reg [5:0] A, B;

wire [5:0] sum;

wire c6;

reg c0;

Carry\_Sel\_Adder UUT(c6, sum, A, B, c0);

initial

$monitorb ("sum = %d A = %d B = %d carryout = %d carryin = ", sum, A, B, c6, c0);

initial begin

$vcdpluson;

c0 = 1'b1; A = 6'b100000; B = 6'b100000;

$displayb("Testing if carryout=1 properly with carry-in = 1");

#10 A = 6'b101001; B = 6'b001100;

$displayb("Testing if carryout=0 properly with carry-in = 1");

#10 A = 6'b100110; B = 6'b011000;

$displayb("Testing if all bits go to 1 with carry-in = 1");

#10 A = 6'b101010; B = 6'b010101;

$displayb("Testing if all bits go to 0 with carry-in = 1");

#10 c0 = 1'b0; A = 6'b010101; B = 6'b101010;

$displayb("Testing if carryout=0 properly with carry-in = 0");

#10 A = 6'b100000; B = 6'b100000;

$displayb("Testing if carryout=1 properly with carry-in = 0");

#10 A = 6'b100110; B = 6'b011001;

$displayb("Testing if all bits go to 1 with carry-in = 0");

#10 A = 6'b010110; B = 6'b101010;

$displayb("Testing if all bits go to 0 with carry-in = 0");

#20 $finish;

end

endmodule

**Exhaustive Test Bench**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #6 Garen Nikoyan, Spring 2018

\*\*\* Carry Select Adder

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: ExhTB.v Created by: Garen Nikoyan, 3/8/2018 \*\*\*

\*\*\* -Revision History

\*\*\* 3/1/2018: First draft

\*\*\* 3/27/2018: Corrected error checking, forced error, and `ifdef

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module exhaustively tests a 6-bit Carry Select Adder \*\*\*

\*\*\* It will force an error if "`define no\_force" is commented out

\*\*\* It has built in error checking which will display a message and stop

\*\*\* simulation if an error is found

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1 ns / 1 ns

//`define no\_force // if commented out, runs code with forced error

module ExhTB();

reg [5:0] A, B;

wire [5:0] sum, outb;

wire c6, coutb;

reg c0;

integer i, j;

Carry\_Sel\_Adder UUT(c6, sum, A, B, c0);

BehavAdder(c0, A, B, coutb, outb);

initial

$monitorb ("A = %d B = %d Carryin = %b Sum = %d Carryout = %b ||| SumBehav = %d CarryoutBehav = %b", A, B, c0, sum, c6, outb, coutb);

initial begin

$vcdpluson;

// Testing with Carryin = 0

c0 = 0; A = 0; B = 0;

for(i=0; i<64; i=i+1) begin

for(j=0; j<64; j=j+1) begin

#10 B = B + 1'b1;

// Turning monitor off after first dozen results

if(A==0 && B==12) $monitoroff;

else;

// If statement to check for errors in either sum or carryout, stops simulation and displays a message if error is found

if (sum!=outb | c6!=coutb) begin

$display ("ERROR \n A = %d B = %d Carryin = %b Sum = %d Carryout = %b SumBehav = %d CarryoutBehav = %b", A, B, c0, sum, c6, outb, coutb);

$stop;

end

else;

end

#10 A = A + 1'b1;

end

// Testing with Carryin = 1

c0 = 1'b1; A = 0; B = 0;

for(i=0; i<64; i=i+1) begin

for(j=0; j<64; j=j+1) begin

#10 B = B + 1'b1;

// Turning Monitor back on for last dozen results

if(A==63 && B==52) $monitoron;

else;

// If statement to check for errors in either sum or carryout, stops simulation and displays a message if error is found

if (sum!=outb | c6!=coutb) begin

$display ("ERROR \n A = %d B = %d Carryin = %b Sum = %d Carryout = %b SumBehav = %d CarryoutBehav = %b", A, B, c0, sum, c6, outb, coutb);

#80 $finish;

end

else;

// if no\_force is defined, forced error is not applied

// if no\_force is not defined, sum is forced to a value that causes an error when test is 75% complete

`ifdef no\_force

`else begin

if(A==47 && B==19) $monitoron;

else;

if(A==47 && B==31) force outb=10;

else;

end

`endif

end

#10 A = A + 1'b1;

end

$display("Exhaustive test was successful with no errors.");

#10 $finish;

end

endmodule

**Log:**

**Non-exhaustive Log**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Mar 15 21:11 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

Testing if carryout=1 properly with carry-in = 1

sum = 1 A = 32 B = 32 carryout = 1 carryin = 1

Testing if carryout=0 properly with carry-in = 1

sum = 54 A = 41 B = 12 carryout = 0 carryin = 1

Testing if all bits go to 1 with carry-in = 1

sum = 63 A = 38 B = 24 carryout = 0 carryin = 1

Testing if all bits go to 0 with carry-in = 1

sum = 0 A = 42 B = 21 carryout = 1 carryin = 1

Testing if carryout=0 properly with carry-in = 0

sum = 63 A = 21 B = 42 carryout = 0 carryin = 0

Testing if carryout=1 properly with carry-in = 0

sum = 0 A = 32 B = 32 carryout = 1 carryin = 0

Testing if all bits go to 1 with carry-in = 0

sum = 63 A = 38 B = 25 carryout = 0 carryin = 0

Testing if all bits go to 0 with carry-in = 0

sum = 0 A = 22 B = 42 carryout = 1 carryin = 0

$finish called from file "NonExhTB.v", line 41.

$finish at simulation time 90

V C S S i m u l a t i o n R e p o r t

Time: 90 ns

CPU Time: 0.210 seconds; Data structure size: 0.0Mb

Thu Mar 15 21:11:53 2018

**Exhaustive Log (No error)**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Mar 29 12:42 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

A = 0 B = 0 Carryin = 0 Sum = 0 Carryout = 0 ||| SumBehav = 0 CarryoutBehav = 0

A = 0 B = 1 Carryin = 0 Sum = 1 Carryout = 0 ||| SumBehav = 1 CarryoutBehav = 0

A = 0 B = 2 Carryin = 0 Sum = 2 Carryout = 0 ||| SumBehav = 2 CarryoutBehav = 0

A = 0 B = 3 Carryin = 0 Sum = 3 Carryout = 0 ||| SumBehav = 3 CarryoutBehav = 0

A = 0 B = 4 Carryin = 0 Sum = 4 Carryout = 0 ||| SumBehav = 4 CarryoutBehav = 0

A = 0 B = 5 Carryin = 0 Sum = 5 Carryout = 0 ||| SumBehav = 5 CarryoutBehav = 0

A = 0 B = 6 Carryin = 0 Sum = 6 Carryout = 0 ||| SumBehav = 6 CarryoutBehav = 0

A = 0 B = 7 Carryin = 0 Sum = 7 Carryout = 0 ||| SumBehav = 7 CarryoutBehav = 0

A = 0 B = 8 Carryin = 0 Sum = 8 Carryout = 0 ||| SumBehav = 8 CarryoutBehav = 0

A = 0 B = 9 Carryin = 0 Sum = 9 Carryout = 0 ||| SumBehav = 9 CarryoutBehav = 0

A = 0 B = 10 Carryin = 0 Sum = 10 Carryout = 0 ||| SumBehav = 10 CarryoutBehav = 0

A = 0 B = 11 Carryin = 0 Sum = 11 Carryout = 0 ||| SumBehav = 11 CarryoutBehav = 0

A = 63 B = 52 Carryin = 1 Sum = 52 Carryout = 1 ||| SumBehav = 52 CarryoutBehav = 1

A = 63 B = 53 Carryin = 1 Sum = 53 Carryout = 1 ||| SumBehav = 53 CarryoutBehav = 1

A = 63 B = 54 Carryin = 1 Sum = 54 Carryout = 1 ||| SumBehav = 54 CarryoutBehav = 1

A = 63 B = 55 Carryin = 1 Sum = 55 Carryout = 1 ||| SumBehav = 55 CarryoutBehav = 1

A = 63 B = 56 Carryin = 1 Sum = 56 Carryout = 1 ||| SumBehav = 56 CarryoutBehav = 1

A = 63 B = 57 Carryin = 1 Sum = 57 Carryout = 1 ||| SumBehav = 57 CarryoutBehav = 1

A = 63 B = 58 Carryin = 1 Sum = 58 Carryout = 1 ||| SumBehav = 58 CarryoutBehav = 1

A = 63 B = 59 Carryin = 1 Sum = 59 Carryout = 1 ||| SumBehav = 59 CarryoutBehav = 1

A = 63 B = 60 Carryin = 1 Sum = 60 Carryout = 1 ||| SumBehav = 60 CarryoutBehav = 1

A = 63 B = 61 Carryin = 1 Sum = 61 Carryout = 1 ||| SumBehav = 61 CarryoutBehav = 1

A = 63 B = 62 Carryin = 1 Sum = 62 Carryout = 1 ||| SumBehav = 62 CarryoutBehav = 1

A = 63 B = 63 Carryin = 1 Sum = 63 Carryout = 1 ||| SumBehav = 63 CarryoutBehav = 1

Exhaustive test was successful with no errors.

$finish called from file "ExhTB.v", line 65.

$finish at simulation time 83210

V C S S i m u l a t i o n R e p o r t

Time: 83210 ns

CPU Time: 0.220 seconds; Data structure size: 0.0Mb

Thu Mar 29 12:42:02 2018

**Exhaustive Log (Forced error)**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Mar 29 13:04 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

A = 0 B = 0 Carryin = 0 Sum = 0 Carryout = 0 ||| SumBehav = 0 CarryoutBehav = 0

A = 0 B = 1 Carryin = 0 Sum = 1 Carryout = 0 ||| SumBehav = 1 CarryoutBehav = 0

A = 0 B = 2 Carryin = 0 Sum = 2 Carryout = 0 ||| SumBehav = 2 CarryoutBehav = 0

A = 0 B = 3 Carryin = 0 Sum = 3 Carryout = 0 ||| SumBehav = 3 CarryoutBehav = 0

A = 0 B = 4 Carryin = 0 Sum = 4 Carryout = 0 ||| SumBehav = 4 CarryoutBehav = 0

A = 0 B = 5 Carryin = 0 Sum = 5 Carryout = 0 ||| SumBehav = 5 CarryoutBehav = 0

A = 0 B = 6 Carryin = 0 Sum = 6 Carryout = 0 ||| SumBehav = 6 CarryoutBehav = 0

A = 0 B = 7 Carryin = 0 Sum = 7 Carryout = 0 ||| SumBehav = 7 CarryoutBehav = 0

A = 0 B = 8 Carryin = 0 Sum = 8 Carryout = 0 ||| SumBehav = 8 CarryoutBehav = 0

A = 0 B = 9 Carryin = 0 Sum = 9 Carryout = 0 ||| SumBehav = 9 CarryoutBehav = 0

A = 0 B = 10 Carryin = 0 Sum = 10 Carryout = 0 ||| SumBehav = 10 CarryoutBehav = 0

A = 0 B = 11 Carryin = 0 Sum = 11 Carryout = 0 ||| SumBehav = 11 CarryoutBehav = 0

A = 47 B = 19 Carryin = 1 Sum = 3 Carryout = 1 ||| SumBehav = 3 CarryoutBehav = 1

A = 47 B = 20 Carryin = 1 Sum = 4 Carryout = 1 ||| SumBehav = 4 CarryoutBehav = 1

A = 47 B = 21 Carryin = 1 Sum = 5 Carryout = 1 ||| SumBehav = 5 CarryoutBehav = 1

A = 47 B = 22 Carryin = 1 Sum = 6 Carryout = 1 ||| SumBehav = 6 CarryoutBehav = 1

A = 47 B = 23 Carryin = 1 Sum = 7 Carryout = 1 ||| SumBehav = 7 CarryoutBehav = 1

A = 47 B = 24 Carryin = 1 Sum = 8 Carryout = 1 ||| SumBehav = 8 CarryoutBehav = 1

A = 47 B = 25 Carryin = 1 Sum = 9 Carryout = 1 ||| SumBehav = 9 CarryoutBehav = 1

A = 47 B = 26 Carryin = 1 Sum = 10 Carryout = 1 ||| SumBehav = 10 CarryoutBehav = 1

A = 47 B = 27 Carryin = 1 Sum = 11 Carryout = 1 ||| SumBehav = 11 CarryoutBehav = 1

A = 47 B = 28 Carryin = 1 Sum = 12 Carryout = 1 ||| SumBehav = 12 CarryoutBehav = 1

A = 47 B = 29 Carryin = 1 Sum = 13 Carryout = 1 ||| SumBehav = 13 CarryoutBehav = 1

A = 47 B = 30 Carryin = 1 Sum = 14 Carryout = 1 ||| SumBehav = 14 CarryoutBehav = 1

A = 47 B = 31 Carryin = 1 Sum = 15 Carryout = 1 ||| SumBehav = 10 CarryoutBehav = 1

ERROR

A = 47 B = 32 Carryin = 1 Sum = 15 Carryout = 1 SumBehav = 10 CarryoutBehav = 1

$finish called from file "ExhTB.v", line 47.

$finish at simulation time 72550

V C S S i m u l a t i o n R e p o r t

Time: 72550 ns

CPU Time: 0.230 seconds; Data structure size: 0.0Mb

Thu Mar 29 13:04:32 2018

**Conclusion:**

Everything in this lab worked as expected. Some issues were encountered when trying to use assign to concatenate a 6 bit output into 2 separate outputs, resulting in the MSB being output as Z. This was solved by using a temporary variable to first be set to the 6 bit output, and then using that temporary variable to assign the proper values to the 2 outputs. The biggest takeaway from this lab was the use of loops and error checking. When designing something like this that would need thousands of vectors to be exhaustively tested, it is much more efficient to use loops and then check the output against known correct output. The use of `ifdef makes it very easy to enable and disable a force statement to make sure the error checking is working properly.