Vincent Li

Assignment 4

Carry bypass adder:

```
// cbAdder.v
// Vincent Li
// 3/19/2019
`timescale 1ns/1ps
module cbAdder_unit ( // for 1 bit and without mux
    input wire A, B,
    input wire Cin,
    output wire P,
    output wire S,
    output wire Cout
    );
    wire G;
    // propogate and generate
    assign P = A ^ B;
    assign G = A * B;
    // carry out and sum
    assign Cout = G + P * Cin;
    assign S = A ^ B ^ Cin;

endmodule

// connecting for cbAdder_unit's with a mux
module cbAdder_4 (
    input wire [3 : 0] A, B,
    input wire Cin,
    output wire [3 : 0] S,
    output wire Cout
    );
    // internal wires
    wire [3 : 0] cin;
    wire [3 : 0] cout;
    wire [3 : 0] P;

    // instantiating 4 units
    cbAdder_unit four[3 : 0] (.A(A), .B(B), .Cin(cin), .P(P), .Cout(cout), .S(S));
    assign cin[0] = Cin;
    assign cin[3 : 1] = cout[2 : 0];
    assign Cout = (P[0] * P[1] * P[2] * P[3] == 1) ? Cin : cout[3];

endmodule

// connecting 16 cbAdder-4's
module cbAdder #(parameter integer WIDTH = 64) ( // WIDTH 4 for 4 bits, 16 for 16 bits
```

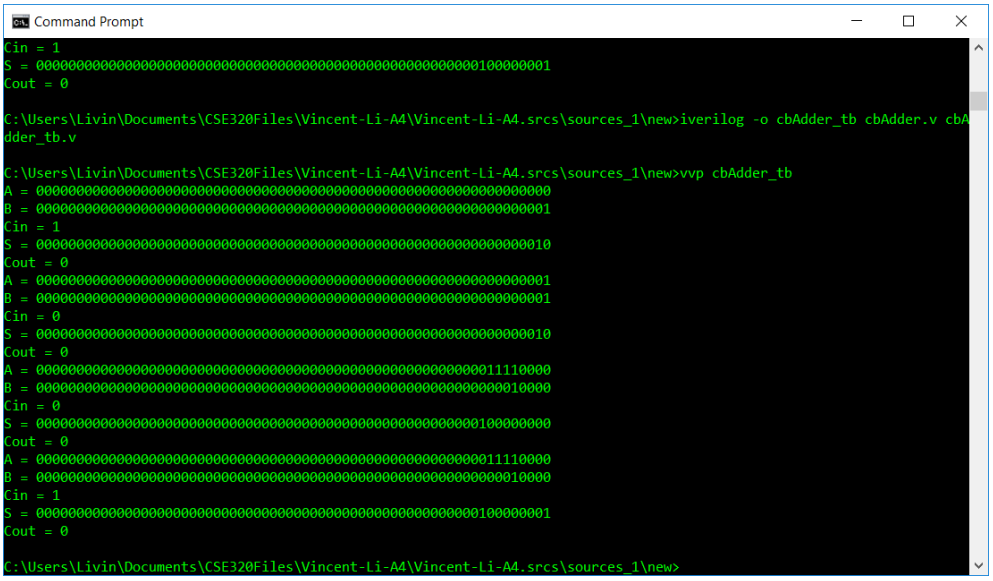
```
    input wire [WIDTH - 1 : 0] A, B,
    input wire Cin,
    output wire [WIDTH - 1 : 0] S,
    output wire Cout
    );

    wire [WIDTH / 4 - 1 : 0] cin, cout;

    genvar i;
    generate
        for(i = 0; i < WIDTH / 4; i = i + 1) begin: build
            cbAdder_4 cba (.A(A[(i+1)*4-1 : (i+1)*4-4]), .B(B[(i+1)*4-1 : (i+1)*4-4]), .Cin(cin[i]), .S(S[(i+1)*4-1 :
(i+1)*4-4]), .Cout(cout[i]));
        end
    endgenerate

    assign cin[0] = Cin;
    assign Cout = cout[WIDTH / 4 - 1];
    assign cin[WIDTH / 4 - 1 : 1] = cout[WIDTH / 4 - 2 : 0];

endmodule
```

TB:

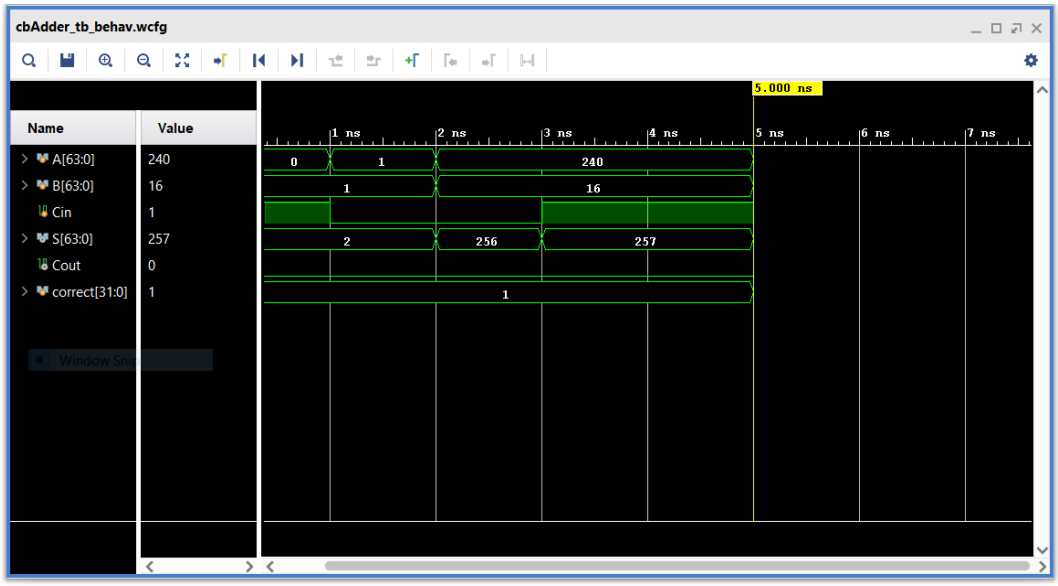
```
// cbAdder_tb.v
// Vincent Li
// 3/25/2019
module cbAdder_tb;
    parameter integer WIDTH = 64;
    reg [WIDTH - 1 : 0] A, B;
    reg Cin;
    wire [WIDTH - 1 : 0] S;
    wire Cout;

    cbAdder #(.WIDTH(WIDTH)) adder (.A(A), .B(B), .Cin(Cin), .S(S), .Cout(Cout));

    initial
        begin
            A = 0;
            B = 1;
            Cin = 1;
            #1;
            $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

            A = 1;
            B = 1;
            Cin = 0;
            #1;
            $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);
```

```
        A = 240;
        B = 16;
        Cin = 0;
        #1;
        $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

        A = 240;
        B = 16;
        Cin = 1;
        #1;
        $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

    end

endmodule
```

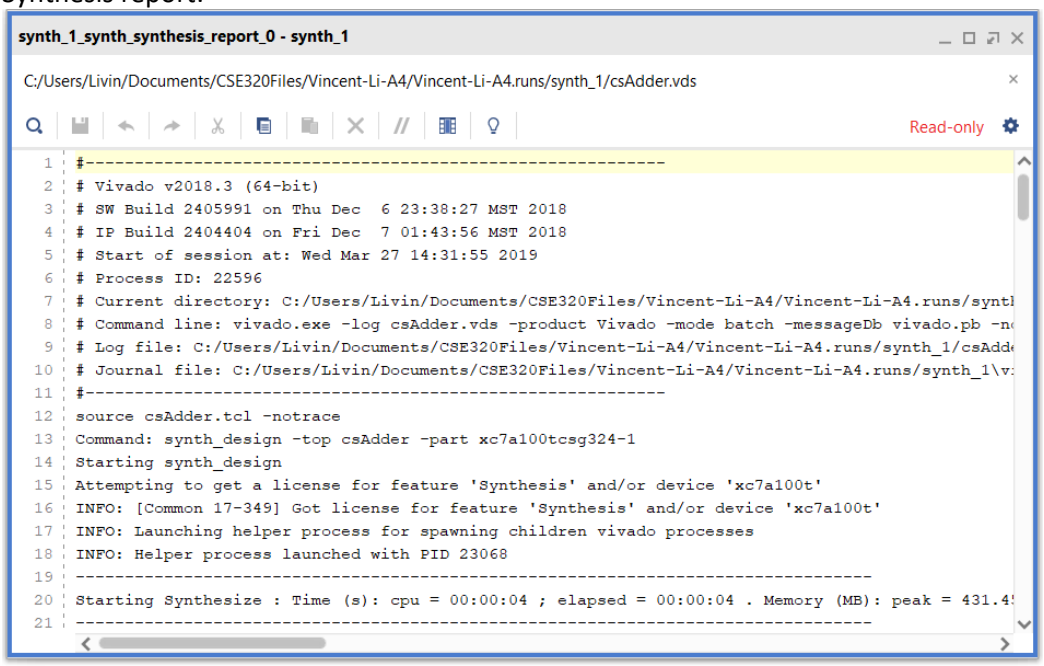
Results:



Waveform:



Synthesis report:



Carry select adder:

```
// csAdder.v
// Vincent Li
// 3/19/2019
`timescale 1ns / 1ps

module rca_1 (
   // modified from assignment 1
   input wire a, b, c_in,
   output wire sum, c_out
```

```
    );
    assign sum = a ^ b ^ c_in;
    assign c_out = a * b + c_in * (a ^ b);

endmodule

// 4-bit rca
module rca_4 #(parameter integer WIDTH = 4) (
    // from assigment 1
    output wire [WIDTH - 1 : 0] sum,
    output wire out,
    input wire [WIDTH - 1 : 0] a, b,
    input wire in
    );

    wire [WIDTH - 1 : 0] c_in, c_out;
    wire [WIDTH - 1 : 0] s;

    rca_1 M[WIDTH - 1 : 0] (.a(a), .b(b), .c_in(c_in), .sum(s), .c_out(c_out));
    // connecting wires
    // for c_in and c_out
    assign c_in[0] = in;
    assign out = c_out[WIDTH - 1];
    assign c_in[WIDTH - 1 : 1] = c_out[WIDTH - 2 : 0];
    // for sum
    assign sum[WIDTH - 1 : 0] = s[WIDTH - 1 : 0];

endmodule

// csAdder for 4 bits
module csAdder_4 (
    input wire [3 : 0] A, B,
    input wire Cin,
    output wire [3 : 0] S,
    output wire Cout
    );
    // internal wires
    wire carry0out, carry1out;
    wire [3 : 0] carry0S, carry1S;
    // instantiating both rca's
    rca_4 carry0[3 : 0] (.a(A), .b(B), .in(1'b0), .sum(carry0S), .out(carry0out));
    rca_4 carry1[3 : 0] (.a(A), .b(B), .in(1'b1), .sum(carry1S), .out(carry1out));
    // mux for carry out
    assign Cout = Cin ? carry1out : carry0out;
    // mux for sums
    assign S = Cin ? carry1S : carry0S;

endmodule
```

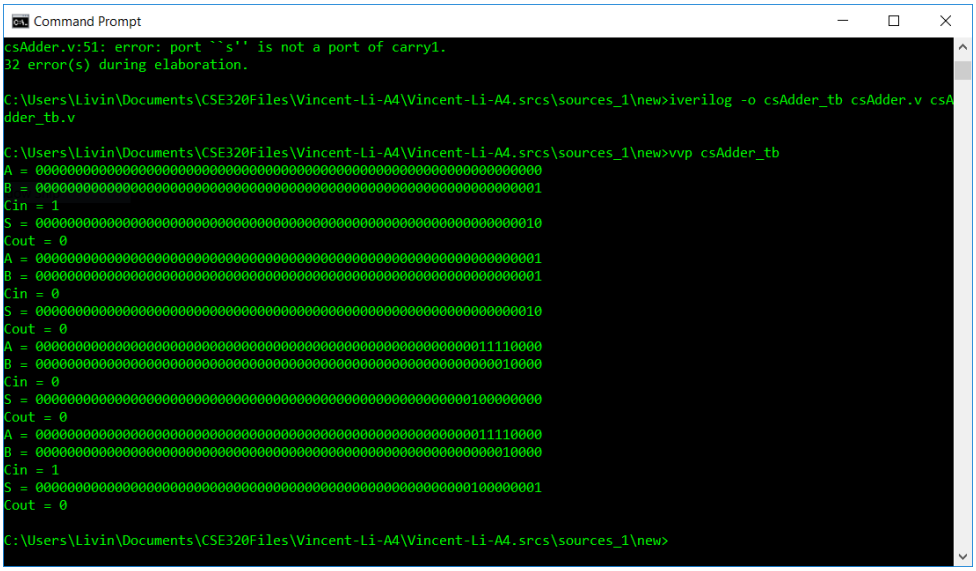
```
module csAdder #(parameter integer WIDTH = 64) (
   input wire [WIDTH - 1 : 0] A, B,
   input wire Cin,
   output wire [WIDTH - 1 : 0] S,
   output wire Cout
   );
   // internal wires
   wire [WIDTH / 4 - 1 : 0] cin, cout;
   // generate
   genvar i;
   generate
      for(i = 0; i < WIDTH / 4; i = i + 1) begin: build
         csAdder_4 csa(.A(A[(i+1)*4-1 : (i+1)*4-4]), .B(B[(i+1)*4-1 : (i+1)*4-4]), .Cin(cin[i]), .S(S[(i+1)*4-1 :
(i+1)*4-4]), .Cout(cout[i]));
      end
   endgenerate
   // connections
   assign cin[0] = Cin;
   assign Cout = cout[WIDTH / 4 - 1 : 0];
   assign cin[WIDTH / 4 - 1 : 1] = cout[WIDTH / 4 - 2 : 0];

endmodule
```

TB:

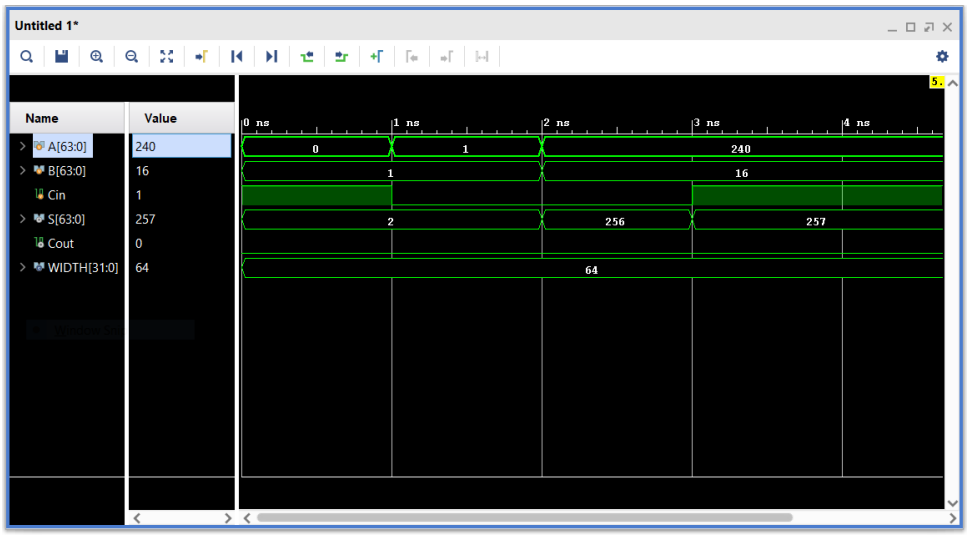
```
`timescale 1ns / 1ps
// csAdder_tb.v
// Vincent Li
// 3/26/2019

module csAdder_tb;
   parameter integer WIDTH = 64;
   reg [WIDTH - 1 : 0] A, B;
   reg Cin;
   wire [WIDTH - 1 : 0] S;
   wire Cout;

   csAdder #(.WIDTH(WIDTH)) adder(.A(A), .B(B), .Cin(Cin), .S(S), .Cout(Cout));

   initial
      begin
         A = 0;
         B = 1;
         Cin = 1;
         #1;
         $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);
```

```
        A = 1;
        B = 1;
        Cin = 0;
        #1;
        $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

        A = 240;
        B = 16;
        Cin = 0;
        #1;
        $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

        A = 240;
        B = 16;
        Cin = 1;
        #1;
        $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b", A, B, Cin, S, Cout);

    end

endmodule
```

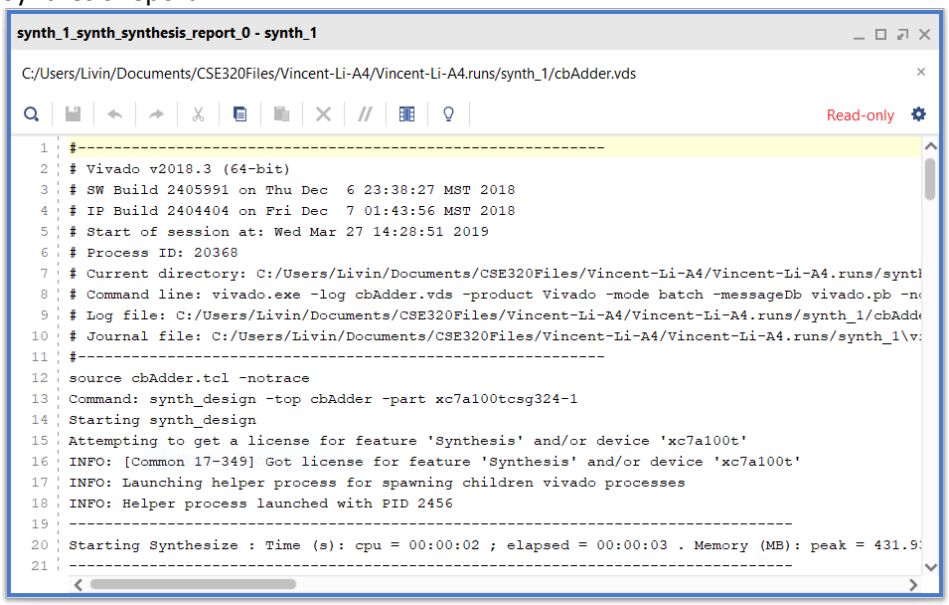
Results:



Waveform:



Synthesis report:



Square root linear carry select adder:

Code:

```
// srclsAdder.v
// Vincent Li
// 3/19/2019
`timescale 1ns / 1ps
module rca_1 (
   // modified from assignment 1
   input wire a, b, c_in,
   output wire sum, c_out
   );
   assign sum = a ^ b ^ c_in;
   assign c_out = a * b + c_in * (a ^ b);
```

```
endmodule

// 4-bit rca
module rca_X #(parameter integer X = 2) (
    // from assigment 1
    output wire [X - 1 : 0] sum,
    output wire out,
    input wire [X - 1 : 0] a, b,
    input wire in
    );

    wire [X - 1 : 0] c_in, c_out;
    wire [X - 1 : 0] s;

    rca_1 M[X - 1 : 0] (.a(a), .b(b), .c_in(c_in), .sum(s), .c_out(c_out));
    // connecting wires
    // for c_in and c_out
    assign c_in[0] = in;
    assign out = c_out[X - 1];
    assign c_in[X - 1 : 1] = c_out[X - 2 : 0];
    // for sum
    assign sum[X - 1 : 0] = s[X - 1 : 0];

endmodule

// csAdder for 4 bits
module csAdder_X #(parameter integer X = 2) (
    input wire [X - 1 : 0] A, B,
    input wire Cin,
    output wire [X - 1 : 0] S,
    output wire Cout
    );
    // internal wires
    wire carry0out, carry1out;
    wire [X - 1 : 0] carry0S, carry1S;
    // instantiating both rca's
    rca_X #(.X(X)) carry0 (.a(A), .b(B), .in(1'b0), .sum(carry0S), .out(carry0out));
    rca_X #(.X(X)) carry1 (.a(A), .b(B), .in(1'b1), .sum(carry1S), .out(carry1out));
    // mux for carry out
    assign Cout = Cin ? carry1out : carry0out;
    // mux for sums
    assign S = Cin ? carry1S : carry0S;

endmodule

// 2 bits to 11 bits. 10 modules
module srlcsAdder #(parameter integer WIDTH = 64) (
```

```
    input wire [WIDTH - 1 : 0] A, B,
    input wire Cin,
    output wire [WIDTH - 1 : 0] S,
    output wire Cout
    );

    // internal wires
    wire [10 : 0] cin, cout;
    // generate
    genvar i, j;
    generate
        for(i = 0; i < 10; i = i + 1) begin
            //
            for(j = i; j < i + 2; j = j + 1) begin: build
                // generate rca's and link to a mux

            end

        end
    endgenerate
    // connections
    assign cin[0] = Cin;
    assign Cout = cout[WIDTH / 4 - 1 : 0];
    assign cin[WIDTH / 4 - 1 : 1] = cout[WIDTH / 4 - 2 : 0];

endmodule
```

Carry look-ahead adder:
Code:

```
// claAdder.v
// Vincent Li
// 3/19/2019
`timescale 1ns / 1ps

module claAdder_4 (
    input wire [3 : 0] A, B,
    input wire Cin,
    output wire [3 : 0] S,
    output wire Cout
    );
    wire [3 : 0] cin; // carries
    wire [3 : 0] P, G;

    // propogate and generate
    assign P[0] = A[0] ^ B[0];
    assign P[1] = A[1] ^ B[1];
    assign P[2] = A[2] ^ B[2];
    assign P[3] = A[3] ^ B[3];
```

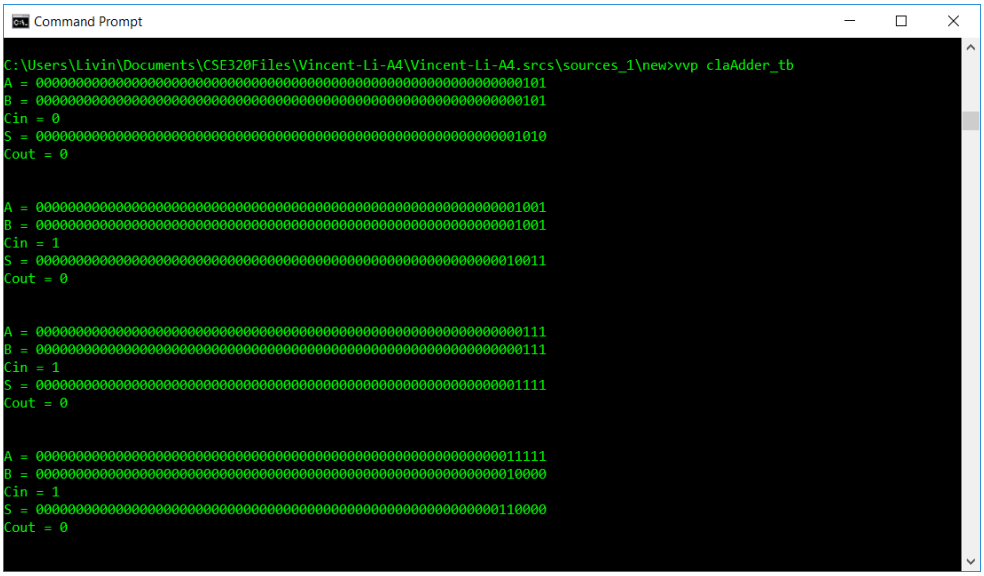
```
    assign G[0] = A[0] * B[0];
    assign G[1] = A[1] * B[1];
    assign G[2] = A[2] * B[2];
    assign G[3] = A[3] * B[3];
    // sums
    assign S[0] = P[0] ^ cin[0];
    assign S[1] = P[1] ^ cin[1];
    assign S[2] = P[2] ^ cin[2];
    assign S[3] = P[3] ^ cin[3];
    // carries
    assign cin[0] = Cin;
    assign cin[1] = G[0] + P[0]*cin[0];
    assign cin[2] = G[1] + P[1]*cin[1];
    assign cin[3] = G[2] + P[2]*cin[2];
    assign Cout = G[3] + P[3]*cin[3];

endmodule

module claAdder #(parameter integer WIDTH = 64) (
    input wire [WIDTH - 1 : 0] A, B,
    input wire Cin,
    output wire [WIDTH - 1 : 0] S,
    output wire Cout
    );
    // connecting wires
    wire [WIDTH / 4 - 1 : 0] cin, cout;
    // generating modules
    genvar i;
    generate
        for(i = 0; i < WIDTH / 4; i = i + 1) begin: build
            claAdder_4 claa (.A(A[(i+1)*4-1 : (i+1)*4-4]), .B(B[(i+1)*4-1 : (i+1)*4-4]), .Cin(cin[i]), .S(S[(i+1)*4-1 :
(i+1)*4-4]), .Cout(cout[i]));
        end
    endgenerate
    // connections
    assign cin[0] = Cin;
    assign Cout = cout[WIDTH / 4 - 1];
    assign cin[WIDTH / 4 - 1 : 1] = cout[WIDTH / 4 - 2 : 0];

endmodule
```

TB:

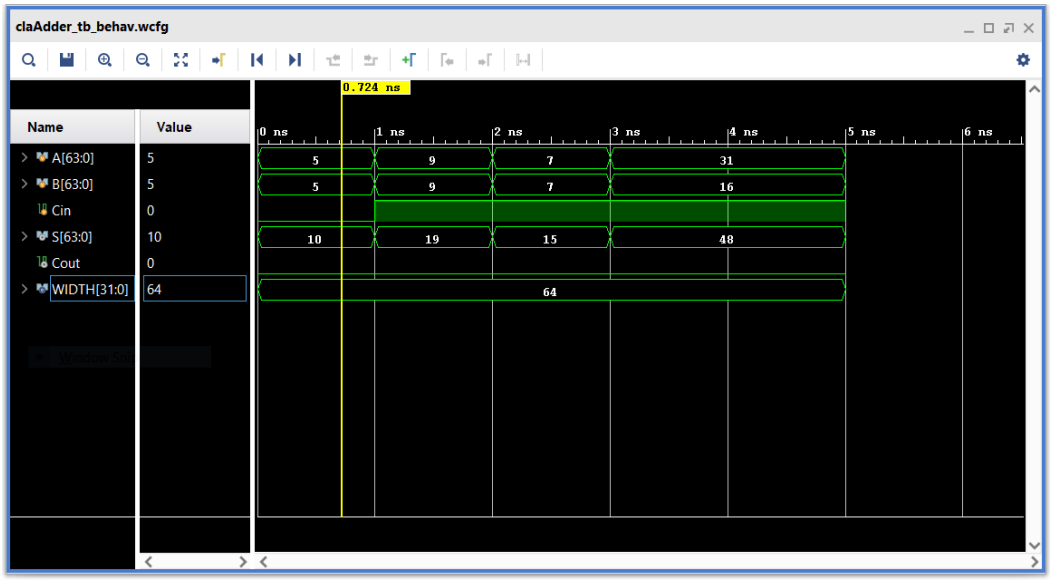
```
// claAdder_tb.v
// Vincent Li
// 3/27/2019
`timescale 1ns / 1ps

module claAdder_tb;
```

```
  parameter integer WIDTH = 64;
  reg [WIDTH - 1 : 0] A, B;
  reg Cin;
  wire [WIDTH - 1 : 0] S;
  wire Cout;

  claAdder #(.WIDTH(WIDTH)) M(.A(A), .B(B), .Cin(Cin), .S(S), .Cout(Cout));

  initial
    begin
      // wrong
      A = 64'b101;
      B = 64'b101;
      Cin = 0;
      #1;
      $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b\n\n", A, B, Cin, S, Cout);
      // wrong
      A = 64'b1001;
      B = 64'b1001;
      Cin = 1;
      #1;
      $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b\n\n", A, B, Cin, S, Cout);

      A = 64'b111;
      B = 64'b111;
      Cin = 1;
      #1;
      $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b\n\n", A, B, Cin, S, Cout);

      A = 64'b11111;
      B = 64'b10000;
      Cin = 1;
      #1;
      $display("A = %b\nB = %b\nCin = %b\nS = %b\nCout = %b\n\n", A, B, Cin, S, Cout);

    end

endmodule
```

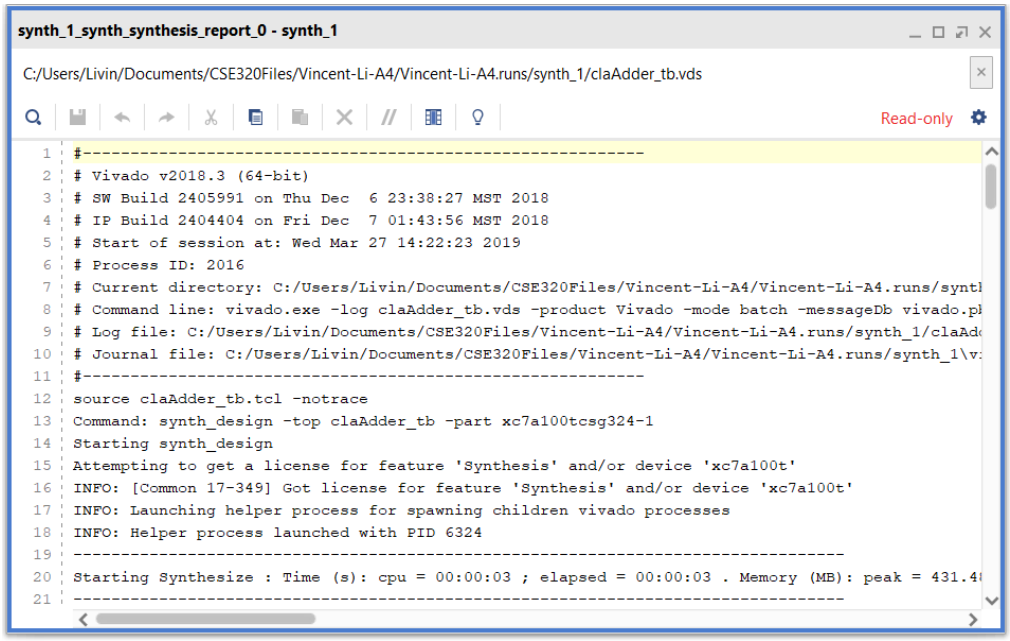
Results:



Waveform:



Synthesis report:

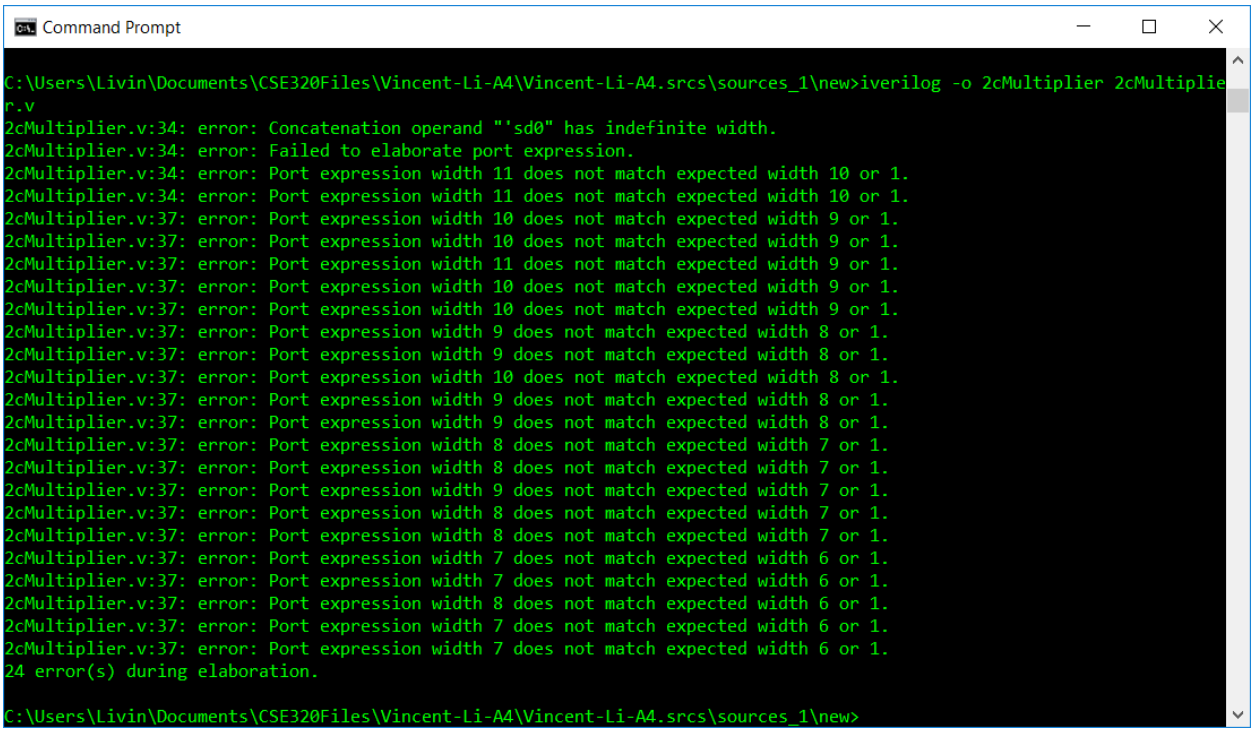


2C Multiplier:
Code: (not completed)

```
// 2cMultiplier.v
// Vincent Li
// 3/28/2019
`timescale 1ns / 1ps

module fullAdder (
   input A, B, Cin,
   output S, Cout
   );
   assign S = A ^ B ^ Cin;
   assign Cout = A * B + Cin * (A ^ B);

endmodule

module twosCMultiplier #(parameter integer W = 6) (
   input wire [W - 1 : 0] X, Y,
   output wire [W * 2 - 1 : 0] Z
   );

   wire [2 * W - 1 : 0] interm [W - 1 : 0];  // 2W bits each wire, W wires
   wire [2 * W - 2 : 0] S [W - 1 : 0];
   wire [2 * W - 2 : 0] carries [W - 2 : 0];

   genvar i;
   generate
      // and gates and sign extensions
      for(i = 0; i < W; i = i + 1) begin: ands
```

```
        and a[W - 1 : 0] (Y, {W{X[i]}}, interm[i][W - 1 + i : i]);
        assign interm[i][2 * W - 1 : W + i] = {(W - i){interm[i][W - 1 + i]}};
      end
      // adders
      for(i = 0; i < W - 1; i = i + 1) begin: genAdders
        if(i == 0) begin
          fullAdder fa [2 * W - 3 - i : 0] (.A(interm[0][2 * W - 2 : i + 1]), .B(interm[1][2 * W - 2 : i + 1]),
.Cin({(2 * W - 1){0}}), .S(S[i][2 * W - 2 : i]), .Cout(carries[i][2 * W - 2 : i]));
        end
        else begin
          fullAdder fa [2 * W - 3 - i : 0] (.A(S[i - 1][2 * W - 2 : i]), .B(interm[i + 1][2 * W - 1 : i + 1]),
.Cin(carries[i - 1][2 * W - 2 : i - 1]), .S(S[i][2 * W - 2 : i]), .Cout(carries[i][2 * W - 2 : i]));
        end
      end
      // assigning Z's
      // first W bits
      for(i = 0; i < 2 * W; i = i + 1) begin
        // first bit of Z
        if(i == 0) begin
          assign Z[i] = interm[i][i];
        end
        // second to W - 1 bit
        else if(i < W - 1) begin
          assign Z[i] = S[i][i];
        end
        // the rest
        else begin
          assign Z[i] = S[W - 1][i];
        end
      end
    endgenerate
```

Results:


Command Prompt
C:\Users\Livin\Documents\CSE320Files\Vincent-Li-A4\Vincent-Li-A4.srcs\sources_1\new>iverilog -o 2cMultiplier 2cMultiplie
r.v
2cMultiplier.v:34: error: Concatenation operand "'sd0" has indefinite width.
2cMultiplier.v:34: error: Failed to elaborate port expression.
2cMultiplier.v:34: error: Port expression width 11 does not match expected width 10 or 1.
2cMultiplier.v:34: error: Port expression width 11 does not match expected width 10 or 1.
2cMultiplier.v:37: error: Port expression width 10 does not match expected width 9 or 1.
2cMultiplier.v:37: error: Port expression width 10 does not match expected width 9 or 1.
2cMultiplier.v:37: error: Port expression width 11 does not match expected width 9 or 1.
2cMultiplier.v:37: error: Port expression width 10 does not match expected width 9 or 1.
2cMultiplier.v:37: error: Port expression width 10 does not match expected width 9 or 1.
2cMultiplier.v:37: error: Port expression width 9 does not match expected width 8 or 1.
2cMultiplier.v:37: error: Port expression width 9 does not match expected width 8 or 1.
2cMultiplier.v:37: error: Port expression width 10 does not match expected width 8 or 1.
2cMultiplier.v:37: error: Port expression width 9 does not match expected width 8 or 1.
2cMultiplier.v:37: error: Port expression width 9 does not match expected width 8 or 1.
2cMultiplier.v:37: error: Port expression width 8 does not match expected width 7 or 1.
2cMultiplier.v:37: error: Port expression width 8 does not match expected width 7 or 1.
2cMultiplier.v:37: error: Port expression width 9 does not match expected width 7 or 1.
2cMultiplier.v:37: error: Port expression width 8 does not match expected width 7 or 1.
2cMultiplier.v:37: error: Port expression width 8 does not match expected width 7 or 1.
2cMultiplier.v:37: error: Port expression width 7 does not match expected width 6 or 1.
2cMultiplier.v:37: error: Port expression width 7 does not match expected width 6 or 1.
2cMultiplier.v:37: error: Port expression width 8 does not match expected width 6 or 1.
2cMultiplier.v:37: error: Port expression width 7 does not match expected width 6 or 1.
2cMultiplier.v:37: error: Port expression width 7 does not match expected width 6 or 1.
24 error(s) during elaboration.
C:\Users\Livin\Documents\CSE320Files\Vincent-Li-A4\Vincent-Li-A4.srcs\sources_1\new>