Lab 7: Logic Synthesis, Static Timing Analysis, and Automatic Place and Route

ECEN 454-503

Alexia Perez, 127008512

Purpose: This lab serves to introduce students to design vision, and its capabilities with a Verilog module. Design vision can take a Verilog module and translate it into a netlist, in which a clock signal and delays can be assigned, and maximum and minimum paths through the module can be calculated. It also introduces students to synthesized netlists and standard cell libraries to place and route circuits on a die in a program called *innovus*.

Procedure:

Part A:

1) Open design vision and adjust the settings to source the correct libraries. Analyze and elaborate the cruisecontrol.v module.
2) Save the design and the attribute settings.
3) Create a symbol view and set the drive strength on all the input ports to 0.0335. Set the load on all the output ports to 3.
4) Set the operating conditions as 'typical' within the library specified in step 1.
5) Save the design and attribute settings.
6) Set the clock constraints to have a period of 25 and a pulse width of 12.5. Set the output delay to be 5 for max rise and min rise.
7) Check the design and use the uniquify command if needed.
8) Save the design as "ccs_attributes_b4_compile.db"
9) Compile the design and set the map effort to medium. Check the "allow boundary conditions."
10) Create the report constraints and report area files.
11) Save the optimized netlist and report the register count.

Part B:

1) Close design vision and open primetime for static timing analysis. Set up the search and link path and load the design. Set the capacitance for the output ports, and the driving strength for the input ports.
2) Create the clock with a period of 10 and a pulse width of 5. Use the check_timing command to show possible timing constraints and set the max and min input delays to 4 and 0, respectively. Set the output delay to 5.
3) Generate the path-based timing reports for the max and min paths of the module.

Part C:

1) Make a directory in the root directory for this lab. Copy the synthesized netlist created in the previous lab and place it in this directory.
2) Download the appropriate files from the lab website and add them to the directory.
3) Open innovus.conf and modify it to include the top level module for the cruise control circuit.
4) Launch innovus using the specified command.

5) Import the design to innovus. Then, specify the floorplan to have an aspect ratio of 0.9 and a core utilization of 0.7. Specify the core margins to be 30 for all.
6) Place the power rails around the die and specify metal3 for the top and bottom layers and metal2 for the left and right layers. Modify all metal width to 10 and change the offset to center in channel.
7) Add power stripes with a width of 5, and a first/last shape of 70.
8) Route the power grid entirely through the die. Next, place the standard cells. Note the number of standard cells used in your design.
9) Route the required wires with the timing driver concurrent routing feature.
10) Print out a copy of the layout generated.

Results:

Part A:

1) The area report and constraints report are displayed below. The combinational area is reported to be 8543, the non-combinational to be 1920, and the total cell area to be 10463. The design also has no violated constraints.

Area:

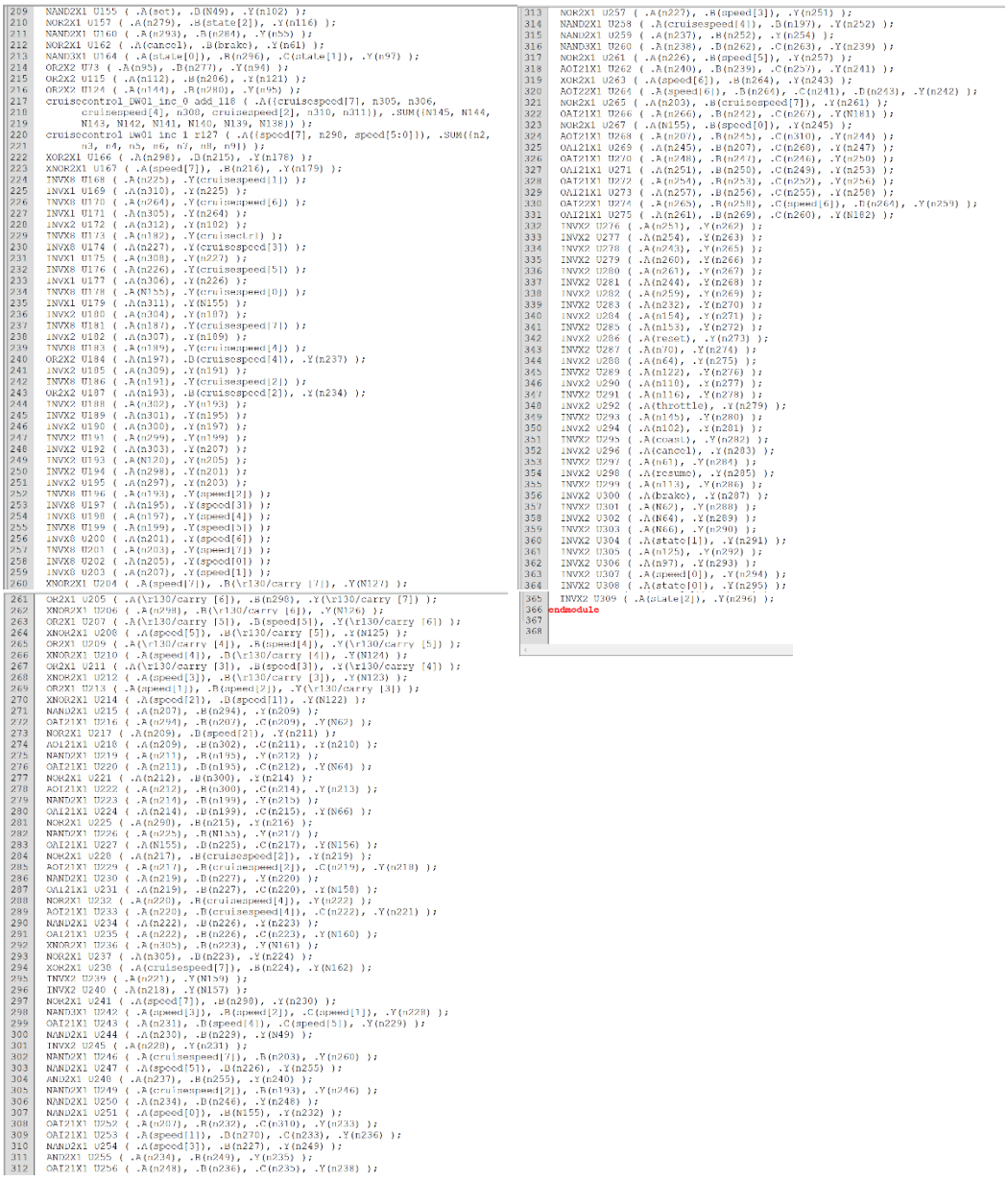
```
****************************************
Report : area
Design : cruisecontrol
Version: O-2018.06-SP3
Date   : Thu Oct 28 06:52:07 2021
****************************************

Library(s) Used:

    iit018_stdcells (File: /home/ugrads/a/alexia_perezv/synthesis/iit018_stdcells.db)

Number of ports:                          58
Number of nets:                          355
Number of cells:                         304
Number of combinational cells:           282
Number of sequential cells:               20
Number of macros/black boxes:              0
Number of buf/inv:                        73
Number of references:                     18

Combinational area:              8543.000000
Buf/Inv area:                    1576.000000
Noncombinational area:           1920.000000
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                10463.000000
Total area:                 undefined
```

Constraints:

```
****************************************
Report : constraint
         -all_violators
         -verbose
Design : cruisecontrol
Version: O-2018.06-SP3
Date   : Thu Oct 28 06:51:12 2021
****************************************

This design has no violated constraints.
```

2) Below is the synthesized netlist for the “cruisecontrol” module. It lists every gate (and, flip flop, not, etc.) or intermediate net needed to create this module as a circuit.

```
////////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : O-2018.06-SP5
// Date      : Thu Oct 28 06:53:09 2021
////////////////////////////////////////////////////////////////


module cruisecontrol_DW01_inc_0 ( A, SUM );
  input [7:0] A;
  output [7:0] SUM;

  wire   [7:2] carry;

  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  INVX1 U1 ( .A(A[0]), .Y(SUM[0]) );
  XOR2X1 U2 ( .A(carry[7]), .B(A[7]), .Y(SUM[7]) );
endmodule


module cruisecontrol_DW01_inc_1 ( A, SUM );
  input [7:0] A;
  output [7:0] SUM;

  wire   [7:2] carry;

  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  INVX2 U1 ( .A(A[0]), .Y(SUM[0]) );
  XOR2X1 U2 ( .A(carry[7]), .B(A[7]), .Y(SUM[7]) );
endmodule


module cruisecontrol ( clk, reset, throttle, set, accel, coast, cancel, resume,
        brake, speed, cruisespeed, cruisectrl );
  output [7:0] speed;
  output [7:0] cruisespeed;
  input clk, reset, throttle, set, accel, coast, cancel, resume, brake;
  output cruisectrl;
  wire   n297, n298, n299, n300, n301, n302, n303, n304, n305, n306, n307,
         n308, n309, n310, n311, n312, N49, N62, N64, N66, N120, N122, N123,
         N124, N125, N126, N127, N138, N139, N140, N141, N142, N143, N144,
         N145, N155, N156, N157, N158, N159, N160, N161, N162, N181, N182, n2,
         n3, n4, n5, n6, n7, n8, n9, n18, n19, n20, n21, n24, n25, n26, n27,
         n29, n30, n31, n32, n35, n36, n37, n38, n39, n41, n42, n44, n45, n46,
         n48, n49, n51, n52, n54, n55, n57, n61, n64, n65, n66, n67, n68, n70,
         n72, n73, n75, n77, n78, n80, n82, n83, n85, n87, n88, n90, n91, n92,
         n93, n94, n95, n97, n100, n102, n104, n105, n107, n108, n111, n112,
         n113, n114, n115, n116, n118, n119, n121, n122, n123, n124, n125,
         n126, n128, n129, n130, n131, n132, n133, n134, n135, n137, n138,
         n139, n140, n141, n142, n143, n144, n145, n146, n147, n148, n149,
         n150, n151, n152, n153, n154, n155, n156, n158, n159, n160, n161,
         n162, n163, n164, n165, n166, n167, n168, n169, n170, n171, n172,
         n173, n174, n175, n176, n177, n178, n179, n182, n187, n189, n191,
         n193, n195, n197, n199, n201, n203, n205, n207, n209, n210, n211,
         n212, n213, n214, n215, n216, n217, n218, n219, n220, n221, n222,
         n223, n224, n225, n226, n227, n228, n229, n230, n231, n232, n233,
         n234, n235, n236, n237, n238, n239, n240, n241, n242, n243, n244,
         n245, n246, n247, n248, n249, n250, n251, n252, n253, n254, n255,
         n256, n257, n258, n259, n260, n261, n262, n263, n264, n265, n266,
         n267, n268, n269, n270, n271, n272, n273, n274, n275, n276, n277,
         n278, n279, n280, n281, n282, n283, n284, n285, n286, n287, n288,
         n289, n290, n291, n292, n293, n294, n295, n296;
  wire   [2:0] state;
  wire   [8:0] \r130/carry ;

  DFFPOSX1 \cruisespeed reg[0]  ( .D(n177), .CLK(clk), .Q(n311) );
  DFFPOSX1 \cruisespeed reg[7]  ( .D(n176), .CLK(clk), .Q(n304) );
  DFFPOSX1 \cruisespeed reg[6]  ( .D(n175), .CLK(clk), .Q(n305) );
  DFFPOSX1 \speed reg[1]  ( .D(n174), .CLK(clk), .Q(n303) );
  DFFPOSX1 \state reg[1]  ( .D(n173), .CLK(clk), .Q(state[1]) );
  DFFPOSX1 \speed reg[0]  ( .D(n172), .CLK(clk), .Q(N120) );
  DFFPOSX1 \speed reg[7]  ( .D(n171), .CLK(clk), .Q(n297) );
  DFFPOSX1 \speed reg[6]  ( .D(n170), .CLK(clk), .Q(n298) );
  DFFPOSX1 \speed reg[5]  ( .D(n169), .CLK(clk), .Q(n299) );
  DFFPOSX1 \state reg[0]  ( .D(n168), .CLK(clk), .Q(state[0]) );
  DFFPOSX1 \state reg[2]  ( .D(n167), .CLK(clk), .Q(state[2]) );
  DFFPOSX1 cruisectrl reg ( .D(n166), .CLK(clk), .Q(n312) );
  DFFPOSX1 \speed reg[4]  ( .D(n165), .CLK(clk), .Q(n300) );
  DFFPOSX1 \cruisespeed reg[4]  ( .D(n164), .CLK(clk), .Q(n307) );
  DFFPOSX1 \speed reg[3]  ( .D(n163), .CLK(clk), .Q(n301) );
  DFFPOSX1 \cruisespeed reg[3]  ( .D(n162), .CLK(clk), .Q(n308) );
  DFFPOSX1 \speed reg[2]  ( .D(n161), .CLK(clk), .Q(n302) );
  DFFPOSX1 \cruisespeed reg[2]  ( .D(n160), .CLK(clk), .Q(n309) );
  DFFPOSX1 \cruisespeed reg[5]  ( .D(n159), .CLK(clk), .Q(n306) );
  DFFPOSX1 \cruisespeed reg[1]  ( .D(n158), .CLK(clk), .Q(n310) );
  NAND2X1 U3 ( .A(n18), .B(n19), .Y(n158) );
  AOI22X1 U4 ( .A(N139), .B(n20), .C(N156), .D(n21), .Y(n19) );
  AOI22X1 U5 ( .A(n303), .B(n271), .C(n310), .D(n272), .Y(n18) );
  NAND2X1 U6 ( .A(n24), .B(n25), .Y(n159) );
  AOI22X1 U7 ( .A(N143), .B(n20), .C(N160), .D(n21), .Y(n25) );
  AOI22X1 U8 ( .A(n299), .B(n271), .C(n306), .D(n272), .Y(n24) );
  NAND2X1 U9 ( .A(n26), .B(n27), .Y(n160) );
  AOI22X1 U10 ( .A(N140), .B(n20), .C(N157), .D(n21), .Y(n27) );
  AOI22X1 U11 ( .A(n302), .B(n271), .C(cruisespeed[2]), .D(n272), .Y(n26) );
  OAI21X1 U12 ( .A(n193), .B(n29), .C(n30), .Y(n161) );
  NAND2X1 U13 ( .A(n31), .B(n32), .Y(n30) );
  OAI21X1 U14 ( .A(n276), .B(n210), .C(n35), .Y(n32) );
  AOI22X1 U15 ( .A(N122), .B(n36), .C(n7), .D(n37), .Y(n35) );
  NAND2X1 U16 ( .A(n38), .B(n39), .Y(n162) );
  AOI22X1 U17 ( .A(N141), .B(n20), .C(N158), .D(n21), .Y(n39) );
  AOI22X1 U20 ( .A(n301), .B(n271), .C(n308), .D(n272), .Y(n38) );
  OAI21X1 U21 ( .A(n29), .B(n195), .C(n41), .Y(n163) );
  NAND2X1 U22 ( .A(n31), .B(n42), .Y(n41) );
  OAI21X1 U23 ( .A(n276), .B(n289), .C(n44), .Y(n42) );
  AOI22X1 U24 ( .A(N123), .B(n36), .C(n6), .D(n37), .Y(n44) );
  NAND2X1 U30 ( .A(n45), .B(n46), .Y(n164) );
  AOI22X1 U31 ( .A(N142), .B(n20), .C(N159), .D(n21), .Y(n46) );
  AOI22X1 U32 ( .A(n300), .B(n271), .C(cruisespeed[4]), .D(n272), .Y(n45) );
  OAI21X1 U33 ( .A(n29), .B(n197), .C(n48), .Y(n165) );
  NAND2X1 U34 ( .A(n31), .B(n49), .Y(n48) );
  OAI21X1 U35 ( .A(n276), .B(n213), .C(n51), .Y(n49) );
  AOI22X1 U36 ( .A(N124), .B(n36), .C(n5), .D(n37), .Y(n51) );
  OAI21X1 U38 ( .A(n52), .B(n275), .C(n54), .Y(n166) );
  NAND3X1 U39 ( .A(n55), .B(n273), .C(cruisectrl), .Y(n54) );
  AOI22X1 U40 ( .A(n57), .B(n281), .C(resume), .D(state[2]), .Y(n52) );
  NOR2X1 U41 ( .A(n295), .B(n278), .Y(n57) );
  OAI22X1 U42 ( .A(n61), .B(n274), .C(n286), .D(n275), .Y(n167) );
  OAI21X1 U44 ( .A(brake), .B(n276), .C(n65), .Y(n168) );
  NAND2X1 U45 ( .A(n66), .B(n273), .Y(n65) );
  OAI22X1 U46 ( .A(n67), .B(n68), .C(n60), .D(n292), .Y(n66) );
  NOR2X1 U47 ( .A(throttle), .B(state[2]), .Y(n68) );
  OAI21X1 U49 ( .A(n199), .B(n29), .C(n72), .Y(n169) );
  NAND2X1 U50 ( .A(n31), .B(n73), .Y(n72) );
  OAI21X1 U51 ( .A(n276), .B(n290), .C(n75), .Y(n73) );
  AOI22X1 U52 ( .A(N125), .B(n36), .C(n4), .D(n37), .Y(n75) );
  OAI21X1 U54 ( .A(n29), .B(n201), .C(n77), .Y(n170) );
  NAND2X1 U55 ( .A(n31), .B(n78), .Y(n77) );
  OAI21X1 U56 ( .A(n276), .B(n178), .C(n80), .Y(n78) );
  AOI22X1 U57 ( .A(N126), .B(n36), .C(n3), .D(n37), .Y(n80) );
  OAI21X1 U60 ( .A(n29), .B(n203), .C(n82), .Y(n171) );
  NAND2X1 U61 ( .A(n31), .B(n83), .Y(n82) );
  OAI21X1 U62 ( .A(n276), .B(n179), .C(n85), .Y(n83) );
  AOI22X1 U63 ( .A(N127), .B(n36), .C(n2), .D(n37), .Y(n85) );
  OAI21X1 U66 ( .A(n29), .B(n296), .C(n87), .Y(n172) );
  NAND2X1 U67 ( .A(n31), .B(n88), .Y(n87) );
  OAI21X1 U68 ( .A(n276), .B(N120), .C(n90), .Y(n88) );
  AOI22X1 U69 ( .A(N120), .B(n36), .C(n9), .D(n37), .Y(n90) );
  OAI21X1 U71 ( .A(reset), .B(n91), .C(n92), .Y(n173) );
  AOI22X1 U72 ( .A(n64), .B(n93), .C(n70), .D(n94), .Y(n92) );
  NOR2X1 U74 ( .A(n97), .B(reset), .Y(n70) );
  OAI22X1 U75 ( .A(n296), .B(n285), .C(n295), .D(n100), .Y(n93) );
  OAI21X1 U76 ( .A(n281), .B(n279), .C(n296), .Y(n100) );
  NOR2X1 U78 ( .A(reset), .B(state[1]), .Y(n64) );
  OAI21X1 U79 ( .A(n207), .B(n29), .C(n104), .Y(n174) );
  NAND2X1 U80 ( .A(n31), .B(n105), .Y(n104) );
  OAI21X1 U81 ( .A(n276), .B(n288), .C(n107), .Y(n105) );
  AOI22X1 U82 ( .A(n207), .B(n36), .C(n8), .D(n37), .Y(n107) );
  NAND2X1 U83 ( .A(n108), .B(n67), .Y(n37) );
  AOI21X1 U84 ( .A(n293), .B(n280), .C(n111), .Y(n67) );
  OAI21X1 U85 ( .A(n112), .B(n113), .C(n114), .Y(n111) );
  AOI22X1 U86 ( .A(n293), .B(n115), .C(n116), .D(n291), .Y(n108) );
  OAI21X1 U87 ( .A(N181), .B(n118), .C(n119), .Y(n115) );
  OAI21X1 U88 ( .A(n97), .B(n287), .C(n121), .Y(n36) );
  NAND3X1 U92 ( .A(n123), .B(n91), .C(n124), .Y(n122) );
  AOI22X1 U93 ( .A(N125), .B(n278), .C(n293), .D(n126), .Y(n124) );
  OAI21X1 U94 ( .A(brake), .B(n283), .C(n128), .Y(n126) );
  NAND3X1 U96 ( .A(state[1]), .B(n279), .C(n129), .Y(n91) );
  AND2X1 U97 ( .A(n29), .B(n273), .Y(n31) );
  NAND3X1 U98 ( .A(n130), .D(n131), .C(n132), .Y(n29) );
  NOR2X1 U99 ( .A(n133), .B(n134), .Y(n132) );
  OAI21X1 U100 ( .A(n292), .B(n135), .C(n123), .Y(n134) );
  NAND3X1 U101 ( .A(n277), .B(n293), .C(N181), .Y(n123) );
  NAND3X1 U103 ( .A(n282), .B(n279), .C(n137), .Y(n118) );
  OAI21X1 U104 ( .A(n138), .B(n139), .C(n285), .Y(n135) );
  NAND3X1 U106 ( .A(n294), .B(n207), .C(n140), .Y(n139) );
  NOR2X1 U107 ( .A(speed[3]), .B(speed[2]), .Y(n140) );
  NAND3X1 U109 ( .A(n197), .B(n199), .C(n141), .Y(n138) );
  NOR2X1 U110 ( .A(speed[7]), .B(speed[6]), .Y(n141) );
  NAND3X1 U113 ( .A(n121), .B(n273), .C(n114), .Y(n133) );
  NAND2X1 U114 ( .A(n116), .B(n295), .Y(n114) );
  NOR2X1 U117 ( .A(n296), .B(resume), .Y(n113) );
  AOI22X1 U118 ( .A(N182), .B(n142), .C(n129), .D(state[1]), .Y(n131) );
  NOR2X1 U119 ( .A(state[2]), .B(state[0]), .Y(n129) );
  OAI21X1 U120 ( .A(n296), .B(n112), .C(n97), .Y(n142) );
  NAND2X1 U121 ( .A(n295), .B(n291), .Y(n112) );
  AOI21X1 U122 ( .A(n293), .B(n95), .C(n143), .Y(n130) );
  NAND3X1 U126 ( .A(n137), .B(n282), .C(throttle), .Y(n145) );
  NAND2X1 U129 ( .A(n146), .B(n147), .Y(n175) );
  AOI22X1 U130 ( .A(N144), .B(n20), .C(N161), .D(n21), .Y(n147) );
  AOI22X1 U131 ( .A(n298), .B(n271), .C(n305), .D(n272), .Y(n146) );
  NAND2X1 U132 ( .A(n148), .B(n149), .Y(n176) );
  AOI22X1 U133 ( .A(N145), .B(n20), .C(N162), .D(n21), .Y(n149) );
  AOI22X1 U134 ( .A(n297), .B(n271), .C(cruisespeed[7]), .D(n272), .Y(n148) );
  NAND2X1 U135 ( .A(n150), .B(n151), .Y(n177) );
  AOI22X1 U136 ( .A(N138), .B(n20), .C(N155), .D(n21), .Y(n151) );
  NOR2X1 U137 ( .A(n128), .B(n152), .Y(n21) );
  NOR2X1 U138 ( .A(n152), .B(n119), .Y(n20) );
  NAND2X1 U139 ( .A(n293), .B(n273), .Y(n152) );
  AOI22X1 U140 ( .A(N120), .B(n271), .C(n311), .D(n272), .Y(n150) );
  NAND3X1 U143 ( .A(n153), .B(n273), .C(n143), .Y(n154) );
  OAI21X1 U144 ( .A(state[2]), .B(n292), .C(n155), .Y(n143) );
  NAND3X1 U146 ( .A(n156), .B(n273), .C(n155), .Y(n153) );
  AOI22X1 U147 ( .A(n156), .B(n125), .C(n293), .D(n144), .Y(n155) );
  NAND2X1 U148 ( .A(n119), .B(n128), .Y(n144) );
  NAND2X1 U149 ( .A(coast), .B(n137), .Y(n128) );
  NOR2X1 U150 ( .A(n284), .B(accel), .Y(n137) );
  NAND2X1 U151 ( .A(accel), .B(n61), .Y(n119) );
  NOR2X1 U152 ( .A(n295), .B(state[1]), .Y(n125) );
  NOR2X1 U154 ( .A(n278), .B(n102), .Y(n156) );
```



```
NAND2X1 U155 ( .A(set), .B(N49), .Y(n102) );
NOR2X1 U157 ( .A(n279), .B(state[2]), .Y(n116) );
NAND2X1 U160 ( .A(n293), .B(n284), .Y(n55) );
NOR2X1 U162 ( .A(cancel), .B(brake), .Y(n61) );
NAND3X1 U164 ( .A(state[0]), .B(n296), .C(state[1]), .Y(n97) );
OR2X2 U73 ( .A(n95), .B(n277), .Y(n94) );
OR2X2 U115 ( .A(n112), .B(n206), .Y(n121) );
OR2X2 U124 ( .A(n144), .B(n280), .Y(n95) );
cruisecontrol DW01_inc_0_add_118 ( .A({cruisespeed[7], n305, n306,
      cruisespeed[4], n308, cruisespeed[2], n310, n311}), .SUM({N145, N144,
      N143, N142, N141, N140, N139, N138}) );
cruisecontrol DW01_inc_1_r127 ( .A({speed[7], n296, speed[5:0]}), .SUM({n2,
      n3, n4, n5, n6, n7, n8, n9}) );
XOR2X1 U166 ( .A(n298), .B(n215), .Y(n178) );
XNOR2X1 U167 ( .A(speed[7]), .B(n216), .Y(n179) );
INVX8 U168 ( .A(n225), .Y(cruisespeed[1]) );
INVX1 U169 ( .A(n310), .Y(n225) );
INVX8 U170 ( .A(n264), .Y(cruisespeed[6]) );
INVX1 U171 ( .A(n305), .Y(n264) );
INVX2 U172 ( .A(n312), .Y(n102) );
INVX8 U173 ( .A(n182), .Y(cruisectrl) );
INVX8 U174 ( .A(n227), .Y(cruisespeed[3]) );
INVX1 U175 ( .A(n308), .Y(n227) );
INVX8 U176 ( .A(n226), .Y(cruisespeed[5]) );
INVX1 U177 ( .A(n306), .Y(n226) );
INVX8 U178 ( .A(N155), .Y(cruisespeed[0]) );
INVX1 U179 ( .A(n311), .Y(N155) );
INVX2 U180 ( .A(n304), .Y(n187) );
INVX8 U181 ( .A(n187), .Y(cruisespeed[7]) );
INVX2 U182 ( .A(n307), .Y(n109) );
INVX8 U183 ( .A(n189), .Y(cruisespeed[4]) );
OR2X2 U184 ( .A(n197), .B(cruisespeed[4]), .Y(n237) );
INVX2 U185 ( .A(n309), .Y(n191) );
INVX8 U186 ( .A(n191), .Y(cruisespeed[2]) );
OR2X2 U187 ( .A(n193), .B(cruisespeed[2]), .Y(n234) );
INVX2 U188 ( .A(n302), .Y(n193) );
INVX2 U189 ( .A(n301), .Y(n195) );
INVX2 U190 ( .A(n300), .Y(n197) );
INVX2 U191 ( .A(n299), .Y(n198) );
INVX2 U192 ( .A(n303), .Y(n207) );
INVX2 U193 ( .A(N120), .Y(n205) );
INVX2 U194 ( .A(n298), .Y(n201) );
INVX2 U195 ( .A(n297), .Y(n203) );
INVX8 U196 ( .A(n193), .Y(speed[2]) );
INVX8 U197 ( .A(n195), .Y(speed[3]) );
INVX8 U198 ( .A(n197), .Y(speed[4]) );
INVX8 U199 ( .A(n199), .Y(speed[5]) );
INVX8 U200 ( .A(n201), .Y(speed[6]) );
INVX8 U201 ( .A(n203), .Y(speed[7]) );
INVX8 U202 ( .A(n205), .Y(speed[0]) );
INVX8 U203 ( .A(n207), .Y(speed[1]) );
XNOR2X1 U204 ( .A(speed[7]), .B(\r130/carry [7]), .Y(N127) );
OR2X1 U205 ( .A(\r130/carry [6]), .B(n298), .Y(\r130/carry [7]) );
XNOR2X1 U206 ( .A(n298), .B(\r130/carry [6]), .Y(N126) );
OR2X1 U207 ( .A(\r130/carry [5]), .B(speed[5]), .Y(\r130/carry [6]) );
XNOR2X1 U208 ( .A(speed[5]), .B(\r130/carry [5]), .Y(N125) );
OR2X1 U209 ( .A(\r130/carry [4]), .B(speed[4]), .Y(\r130/carry [5]) );
XNOR2X1 U210 ( .A(speed[4]), .B(\r130/carry [4]), .Y(N124) );
OR2X1 U211 ( .A(\r130/carry [3]), .B(speed[3]), .Y(\r130/carry [4]) );
XNOR2X1 U212 ( .A(speed[3]), .B(\r130/carry [3]), .Y(N123) );
OR2X1 U213 ( .A(speed[1]), .B(speed[2]), .Y(\r130/carry [3]) );
XNOR2X1 U214 ( .A(speed[2]), .B(speed[1]), .Y(N122) );
NAND2X1 U215 ( .A(n207), .B(n294), .Y(n209) );
OAI21X1 U216 ( .A(n294), .B(n207), .C(n209), .Y(N62) );
NOR2X1 U217 ( .A(n209), .B(speed[2]), .Y(n211) );
AOI21X1 U218 ( .A(n209), .B(n302), .C(n211), .Y(n210) );
NAND2X1 U219 ( .A(n211), .B(n195), .Y(n212) );
OAI21X1 U220 ( .A(n211), .B(n195), .C(n212), .Y(N64) );
NOR2X1 U221 ( .A(n212), .B(n300), .Y(n214) );
AOI21X1 U222 ( .A(n212), .B(n300), .C(n214), .Y(n213) );
NAND2X1 U223 ( .A(n214), .B(n199), .Y(n215) );
OAI21X1 U224 ( .A(n214), .B(n199), .C(n215), .Y(N66) );
NOR2X1 U225 ( .A(n290), .B(n215), .Y(n216) );
NAND2X1 U226 ( .A(n225), .B(N155), .Y(n217) );
OAI21X1 U227 ( .A(N155), .B(n225), .C(n217), .Y(N156) );
NOR2X1 U228 ( .A(n217), .B(cruisespeed[2]), .Y(n219) );
AOI21X1 U229 ( .A(n217), .B(cruisespeed[2]), .C(n219), .Y(n218) );
NAND2X1 U230 ( .A(n219), .B(n227), .Y(n220) );
OAI21X1 U231 ( .A(n219), .B(n227), .C(n220), .Y(N158) );
NOR2X1 U232 ( .A(n220), .B(cruisespeed[4]), .Y(n222) );
AOI21X1 U233 ( .A(n220), .B(cruisespeed[4]), .C(n222), .Y(n221) );
NAND2X1 U234 ( .A(n222), .B(n226), .Y(n223) );
OAI21X1 U235 ( .A(n222), .B(n226), .C(n223), .Y(N160) );
XNOR2X1 U236 ( .A(n305), .B(n223), .Y(N161) );
NOR2X1 U237 ( .A(n305), .B(n223), .Y(n224) );
XOR2X1 U238 ( .A(cruisespeed[7]), .B(n224), .Y(N162) );
INVX2 U239 ( .A(n221), .Y(N159) );
INVX2 U240 ( .A(n218), .Y(N157) );
NOR2X1 U241 ( .A(speed[7]), .B(n290), .Y(n230) );
NAND3X1 U242 ( .A(speed[3]), .B(speed[2]), .C(speed[1]), .Y(n228) );
OAI21X1 U243 ( .A(n231), .B(speed[4]), .C(speed[5]), .Y(n229) );
NAND2X1 U244 ( .A(n230), .B(n229), .Y(N49) );
INVX2 U245 ( .A(n228), .Y(n231) );
NAND2X1 U246 ( .A(cruisespeed[7]), .B(n203), .Y(n260) );
NAND2X1 U247 ( .A(speed[5]), .B(n226), .Y(n255) );
AND2X1 U248 ( .A(n237), .B(n255), .Y(n240) );
NAND2X1 U249 ( .A(cruisespeed[2]), .B(n193), .Y(n246) );
NAND2X1 U250 ( .A(n234), .B(n246), .Y(n248) );
NAND2X1 U251 ( .A(speed[0]), .B(N155), .Y(n232) );
OAI21X1 U252 ( .A(n204), .B(n232), .C(n310), .Y(n233) );
OAI21X1 U253 ( .A(speed[1]), .B(n270), .C(n233), .Y(n236) );
NAND2X1 U254 ( .A(speed[3]), .B(n227), .Y(n249) );
AND2X1 U255 ( .A(n234), .B(n249), .Y(n235) );
OAI21X1 U256 ( .A(n248), .B(n236), .C(n235), .Y(n238) );
NOR2X1 U257 ( .A(n227), .B(speed[3]), .Y(n251) );
NAND2X1 U258 ( .A(cruisespeed[4]), .B(n197), .Y(n252) );
NAND2X1 U259 ( .A(n237), .B(n252), .Y(n254) );
NAND3X1 U260 ( .A(n238), .B(n262), .C(n263), .Y(n239) );
NOR2X1 U261 ( .A(n226), .B(speed[5]), .Y(n257) );
AOI21X1 U262 ( .A(n240), .B(n239), .C(n257), .Y(n241) );
XOR2X1 U263 ( .A(speed[6]), .B(n264), .Y(n243) );
AOI22X1 U264 ( .A(speed[6]), .B(n264), .C(n241), .D(n243), .Y(n242) );
NOR2X1 U265 ( .A(n203), .B(cruisespeed[7]), .Y(n261) );
OAI21X1 U266 ( .A(n266), .B(n242), .C(n267), .Y(N181) );
NOR2X1 U267 ( .A(N155), .B(speed[0]), .Y(n245) );
AOI21X1 U268 ( .A(n207), .B(n245), .C(n310), .Y(n244) );
OAI21X1 U269 ( .A(n245), .B(n207), .C(n268), .Y(n247) );
OAI21X1 U270 ( .A(n248), .B(n247), .C(n246), .Y(n250) );
OAI21X1 U271 ( .A(n251), .B(n250), .C(n249), .Y(n253) );
OAI21X1 U272 ( .A(n254), .B(n253), .C(n252), .Y(n256) );
OAI21X1 U273 ( .A(n257), .B(n256), .C(n255), .Y(n258) );
OAI22X1 U274 ( .A(n265), .B(n258), .C(speed[6]), .D(n264), .Y(n259) );
OAI21X1 U275 ( .A(n261), .B(n269), .C(n260), .Y(N182) );
INVX2 U276 ( .A(n251), .Y(n262) );
INVX2 U277 ( .A(n254), .Y(n263) );
INVX2 U278 ( .A(n243), .Y(n265) );
INVX2 U279 ( .A(n260), .Y(n266) );
INVX2 U280 ( .A(n261), .Y(n267) );
INVX2 U281 ( .A(n244), .Y(n268) );
INVX2 U282 ( .A(n259), .Y(n269) );
INVX2 U283 ( .A(n232), .Y(n270) );
INVX2 U284 ( .A(n154), .Y(n271) );
INVX2 U285 ( .A(n153), .Y(n272) );
INVX2 U286 ( .A(reset), .Y(n273) );
INVX2 U287 ( .A(n70), .Y(n274) );
INVX2 U288 ( .A(n64), .Y(n275) );
INVX2 U289 ( .A(n122), .Y(n276) );
INVX2 U290 ( .A(n110), .Y(n277) );
INVX2 U291 ( .A(n116), .Y(n278) );
INVX2 U292 ( .A(throttle), .Y(n279) );
INVX2 U293 ( .A(n145), .Y(n280) );
INVX2 U294 ( .A(n102), .Y(n281) );
INVX2 U295 ( .A(coast), .Y(n282) );
INVX2 U296 ( .A(cancel), .Y(n283) );
INVX2 U297 ( .A(n67), .Y(n284) );
INVX2 U298 ( .A(resume), .Y(n285) );
INVX2 U299 ( .A(n113), .Y(n286) );
INVX2 U300 ( .A(brake), .Y(n287) );
INVX2 U301 ( .A(N62), .Y(n288) );
INVX2 U302 ( .A(N64), .Y(n289) );
INVX2 U303 ( .A(N66), .Y(n290) );
INVX2 U304 ( .A(state[1]), .Y(n291) );
INVX2 U305 ( .A(n125), .Y(n292) );
INVX2 U306 ( .A(n97), .Y(n293) );
INVX2 U307 ( .A(speed[0]), .Y(n294) );
INVX2 U308 ( .A(state[0]), .Y(n295) );
INVX2 U309 ( .A(state[2]), .Y(n296) );
endmodule
```

Within the synthesized net list above, there are no DFFSR registers, but there are 4 DFFPOSX1 registers (the state register, speed register, cruisespeed register, and cruisectrl register).

Part B:

1) Below is the max paths report for a slack lesser than 5. This shows the least possible slack lesser than 5 considering only the setup paths. According to the report, the lowest slack achievable is 3.79.

```
****************************************
Report : timing
        -path_type full
        -delay_type max
        -slack_lesser_than 5.00
        -max_paths 3
        -sort_by slack
Design : cruisecontrol
Version: O-2018.06-SP3
Date   : Thu Oct 28 07:31:20 2021
****************************************


  Startpoint: cruisespeed_reg[6]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: cruisespeed[6]
              (output port clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                   Incr       Path
  -----------------------------------------------------------------------------
  clock clk (rise edge)                                   0.00       0.00
  clock network delay (ideal)                             0.00       0.00
  cruisespeed_reg[6]/CLK (DFFPOSX1)                       0.00       0.00 r
  cruisespeed_reg[6]/Q (DFFPOSX1)                         0.18       0.18 r
  U171/Y (INVX1)                                          0.27       0.44 f
  U170/Y (INVX8)                                          0.77       1.21 r
  cruisespeed[6] (out)                                    0.00       1.21 r
  data arrival time                                                  1.21

  clock clk (rise edge)                                  10.00      10.00
  clock network delay (ideal)                             0.00      10.00
  clock reconvergence pessimism                           0.00      10.00
  output external delay                                  -5.00       5.00
  data required time                                                 5.00
  -----------------------------------------------------------------------------
  data required time                                                 5.00
  data arrival time                                                 -1.21
  -----------------------------------------------------------------------------
  slack (MET)                                                        3.79


  Startpoint: cruisespeed_reg[0]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: cruisespeed[0]
              (output port clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                   Incr       Path
  -----------------------------------------------------------------------------
  clock clk (rise edge)                                   0.00       0.00
  clock network delay (ideal)                             0.00       0.00
  cruisespeed_reg[0]/CLK (DFFPOSX1)                       0.00       0.00 r
  cruisespeed_reg[0]/Q (DFFPOSX1)                         0.14       0.14 r
  U179/Y (INVX1)                                          0.26       0.40 f
  U178/Y (INVX8)                                          0.77       1.17 r
  cruisespeed[0] (out)                                    0.00       1.17 r
  data arrival time                                                  1.17

  clock clk (rise edge)                                  10.00      10.00
  clock network delay (ideal)                             0.00      10.00
  clock reconvergence pessimism                           0.00      10.00
  output external delay                                  -5.00       5.00
  data required time                                                 5.00
  -----------------------------------------------------------------------------
  data required time                                                 5.00
  data arrival time                                                 -1.17
  -----------------------------------------------------------------------------
  slack (MET)                                                        3.83


  Startpoint: cruisespeed_reg[5]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: cruisespeed[5]
              (output port clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                   Incr       Path
  -----------------------------------------------------------------------------
  clock clk (rise edge)                                   0.00       0.00
  clock network delay (ideal)                             0.00       0.00
  cruisespeed_reg[5]/CLK (DFFPOSX1)                       0.00       0.00 r
  cruisespeed_reg[5]/Q (DFFPOSX1)                         0.14       0.14 r
  U177/Y (INVX1)                                          0.23       0.37 f
  U176/Y (INVX8)                                          0.76       1.12 r
  cruisespeed[5] (out)                                    0.00       1.12 r
  data arrival time                                                  1.12

  clock clk (rise edge)                                  10.00      10.00
  clock network delay (ideal)                             0.00      10.00
  clock reconvergence pessimism                           0.00      10.00
  output external delay                                  -5.00       5.00
  data required time                                                 5.00
  -----------------------------------------------------------------------------
  data required time                                                 5.00
  data arrival time                                                 -1.12
  -----------------------------------------------------------------------------
  slack (MET)                                                        3.88
```

2) Below is the min paths report for a slack lesser than 3. This shows the least possible slack lesser than 3 considering only the hold time paths. According to the report, the lowest slack achievable is 0.13.

```
****************************************
Report : timing
        -path_type full
        -delay_type min
        -slack_lesser_than 3.00
        -max_paths 3
        -sort_by slack
Design : cruisecontrol
Version: O-2018.06-SP3
Date   : Thu Oct 28 07:31:54 2021
****************************************


  Startpoint: reset (input port clocked by clk)
  Endpoint: state_reg[1]
              (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  input external delay                     0.00       0.00 f
  reset (in)                               0.06       0.06 f
  U71/Y (OAI21X1)                          0.08       0.13 r
  state_reg[1]/D (DFFPOSX1)                0.00       0.13 r
  data arrival time                                   0.13

  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clock reconvergence pessimism            0.00       0.00
  state_reg[1]/CLK (DFFPOSX1)                         0.00 r
  library hold time                        0.00       0.00
  data required time                                  0.00
  -----------------------------------------------------------
  data required time                                  0.00
  data arrival time                                  -0.13
  -----------------------------------------------------------
  slack (MET)                                         0.13


  Startpoint: brake (input port clocked by clk)
  Endpoint: state_reg[0]
              (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  input external delay                     0.00       0.00 f
  brake (in)                               0.06       0.06 f
  U44/Y (OAI21X1)                          0.08       0.14 r
  state_reg[0]/D (DFFPOSX1)                0.00       0.14 r
  data arrival time                                   0.14

  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clock reconvergence pessimism            0.00       0.00
  state_reg[0]/CLK (DFFPOSX1)                         0.00 r
  library hold time                        0.00       0.00
  data required time                                  0.00
  -----------------------------------------------------------
  data required time                                  0.00
  data arrival time                                  -0.14
  -----------------------------------------------------------
  slack (MET)                                         0.13


  Startpoint: resume (input port clocked by clk)
  Endpoint: cruisectrl_reg
              (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  input external delay                     0.00       0.00 r
  resume (in)                              0.05       0.05 r
  U40/Y (AOI22X1)                          0.06       0.11 f
  U38/Y (OAI21X1)                          0.07       0.18 r
  cruisectrl_reg/D (DFFPOSX1)              0.00       0.18 r
  data arrival time                                   0.18

  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clock reconvergence pessimism            0.00       0.00
  cruisectrl_reg/CLK (DFFPOSX1)                       0.00 r
  library hold time                        0.00       0.00
  data required time                                  0.00
  -----------------------------------------------------------
  data required time                                  0.00
  data arrival time                                  -0.18
  -----------------------------------------------------------
  slack (MET)                                         0.18
```

Part C:

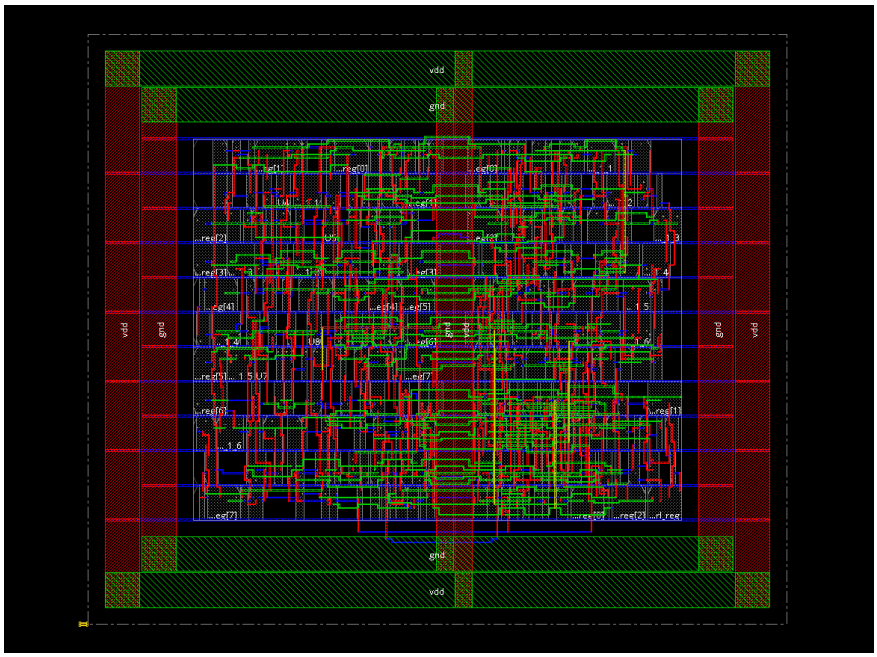
1) For this lab, the total wire length was 9027 µm, and the number of vias was 1436. The screenshot of this information is shown below:

```
#Start Detail Routing..
#start initial detail routing ...
#   number of violations = 3
#
#    By Layer and Type :
#                MetSpc   Totals
#       metal1        3        3
#       Totals        3        3
#cpu time = 00:00:01, elapsed time = 00:00:01, memory = 795.24 (MB), peak = 795.43 (MB)
#start 1st optimization iteration ...
#   number of violations = 0
#cpu time = 00:00:00, elapsed time = 00:00:00, memory = 778.05 (MB), peak = 795.44 (MB)
#Complete Detail Routing.
#Total wire length = 9027 um.
#Total half perimeter of net bounding box = 10143 um.
#Total wire length on LAYER metal1 = 896 um.
#Total wire length on LAYER metal2 = 4274 um.
#Total wire length on LAYER metal3 = 3669 um.
#Total wire length on LAYER metal4 = 157 um.
#Total wire length on LAYER metal5 = 32 um.
#Total wire length on LAYER metal6 = 0 um.
#Total number of vias = 1436
#Up-Via Summary (total 1436):
#
#-----------------------
# metal1            907
# metal2            513
# metal3             14
# metal4              2
#-----------------------
#                  1436
#
#Total number of DRC violations = 0
#Cpu time = 00:00:01
#Elapsed time = 00:00:01
#Increased memory = 4.54 (MB)
#Total memory = 752.03 (MB)
#Peak memory = 795.44 (MB)
```

The number of standard cells reported was 264. The screenshot of this information is shown below:

```
Building hierarchical netlist for Cell cruisecontrol ...
*** Netlist is unique.
** info: there are 36 modules.
** info: there are 302 stdCell insts.
```

2) Lastly, the full layout generated by the program is displayed below.



Conclusion: This lab helped me better understand how to use Design Vision, the graphical interface to the Synopsys family of logic synthesis tools. I became familiar with the basics of synthesis using Design Vision through the simple “cruise control” design example. After completing the synthesis portion, I performed pre-layout static timing analysis of my synthesized design, and the defined constraints for it and check the timing of all the paths in it. In the final part of this lab, I learned how to use the synthesized Verilog netlist that I previously generated to "Place and Route" the cruise control logic circuit on a die.