Bryan Parada

861027456

Lab #4

Deliverables

Week 0-

QUESTIONS

1-25 pages

Which one of the following Verilog structures may occur outside of other structures?

a. (•) **modules**

b. ( ) ports

c. ( ) parameters

d. ( ) instances

Correct. Top level modules occur outside of other modules.

To review content related to this question see Modules: Top Level.

What is wrong with the following module instantiation?

```
module modA;
     ...
     modB #(1,2) (p1, p2,, p4);
     ...
endmodule
```

---

a. ( ) missing ports are not allowed in instantiations

b. (•) **the module instance name is missing**

c. ( ) the module instance parameter values must be after the port list

d. ( ) nothing

That's right. The module instance name is missing.

To review content related to this question see Ports: Port List.

Given these module definitions and instantiations:

```
module modA;
        ...
        modB #(4,5,6) mb1 (p1, p2, p3, p4);
        modB #(7,8) mb2 (p4, p2, p3, q4);
        ...
endmodule

module modB (port1, port2, port3, port4);
        input port1, port2;
        inout port3;
        output port4;
        ...
        parameter par1 = 1,
                  par2 = 2,
                  par3 = par1+par2;
        ...
endmodule
```

---

a. ○ True ◉ **False** In instance modA.mb2, par3 will have the value 3.

b. ○ True ◉ **False** In instance modA.mb2, par3 will have the value 6.

c. ◉ **True** ○ False In instance modA.mb2, par3 will have the value 15.

d. ○ True ◉ **False** In the definition of modB, port4 must be declared as a reg.

Good work.

To review content related to this question see Modelling Structures.

Given the following module:

```
module modB (port1, port2, port3, port4);
        input port1, port2;
        inout port3;
        output port4;
        ...
        parameter par1 = 1,
                  par2 = 2,
                  par3 = par1+par2;
        ...
endmodule
```

Indicate whether the following instances of modB above are legal or illegal.

a. (•) **Legal** ( ) Illegal
```
modB mb2 (p1, , p3, p4);
```

b. ( ) Legal (•) **Illegal**
```
modB #(7,8,9,10) mb2 (p1, p2, p3, p4);
```

c. (•) **Legal** ( ) Illegal
```
modB mb2 (p1, p2, p3);
```

d. (•) **Legal** ( ) Illegal
```
modB mb2 (p1, p2, p3, );
```

That's right.

To review content related to this question see Ports: Assignment: Errors and Positional.

Are these two ways of redefining the parameter par2 equivalent?

a. ( ) Yes (•) **No**

```
modB #(,5,) mb1 (p1, p2, p3, p4);

and

modB mb1 (p1, p2, p3, p4);
defparam mb1.par2 = 5;
```

That's right. You can't do what this is attempting by means of a module instance parameter assignment. The defparam is the only way to change the value of only the second parameter in a module.

To review content related to this question see Modelling Structures: Parameters: Redefinition.

Are the following statements syntactically correct?

a. ○ Yes ◉ **No** reg [5] x;

b. ○ Yes ◉ **No** reg #10 x;

That's right. They're both incorrect. Here are examples of correct statements:
a. `reg[5:1]x;`
b. `reg x;// can't attach a delay to reg definition`

To review content related to this question see Modelling Structures: Registers.

Are the following statements syntactically correct?

a. ◉ **Yes** ○ No wire w1; wire [1:0] w2;

b. ◉ **Yes** ○ No wire #10 x;

Yes. Both statements are correct.

To review content related to this question see Modelling Structures: Nets.

Are the following statements syntactically correct?

a. ◉ **Yes** ○ No and #1 a1 (a,b,c), a2 (d,e,f);

b. ◉ **Yes** ○ No buf #10 (o1, o2, in1);

Yes. Both statements are correct.

To review content related to this question see Modelling Structures: Primitives.

Are the following statements syntactically correct?

a. ○ Yes ◉ **No** initial x #1 = f(y);

b. ◉ **Yes** ○ No initial x = #1 f(y);

c. ◉ **Yes** ○ No initial begin x = 1; y = 0; end

You got it. The first statement is incorrect. Here are two examples of correct statements for a.:
a. `initial #1 x = f(y);`
or
b. `initial x = #1 f(y);`

To review content related to this question see Modelling Structures: Procedural Blocks.

Are the following statements syntactically correct?

a. ○ Yes ◉ **No** function f [1:0] (x);

b. ○ Yes ◉ **No** function f (x[1:0]);

Correct. Both statements are incorrect. Here are examples of correct statements:
a. `function [1:0]f; input x;`
b. `function f; input [1:0]x;`

To review content related to this question see Modelling Structures: Tasks and Functions.

Which of the following statements are legal?

- [x] **1. wire aBc;**
- [ ] 2. wire \reg\;
- [x] **3. wire ABC;**
- [x] **4. wire abc1;**
- [ ] 5. wire 1abc;
- [ ] 6. wire $abc1;

That's right. Statements 2, 5, and 6 are illegal. Statement 2 is illegal because an escaped identifier must end with space. So the ";" is part of the identifier. Statements 5 and 6 are illegal because the first character of an identifier can't be numeric or a "$".

To review content related to this question see Lexical Conventions: Tokens: Identifiers.

Which of the following statements are legal?

- [ ] 1. $time;
- [x] **2. $display("hello sparky");**
- [ ] **3. x = $time;**
- [ ] 4. y = $display("hello sparky");
- [x] **5. if (done) $finish;**
- [x] **6. always @(x) $display($time, " x -> ", x);**

Statement 1 is illegal because $time is a function, not a task. Therefore, it can't stand alone as a statement. Statement 4 is illegal because $display is a task and can't be used as a right-hand side of an expression.

Which of the following models generates a clock waveform that has the clock high 2 time units and low 2 time units, starting at 0 with the first rising edge at time 10?

A

```
module q2;
   reg clock;
   initial begin
        clock = 0;
   #8   forever begin
        #1   clock = 0;
        #2   clock = 1;
        end
   end
endmodule
```

B

```
module q2;
   reg clock;
   initial begin
        clock = 0;
   #8   forever begin
        #2   clock = 1;
        #2   clock = 0;
        end
   end
endmodule
```

C

```
module q2;
   reg clock;
   initial begin
        clock = 0;
   #8   forever begin
        #1   clock = 1;
        #2   clock = 0;
        end
   end
endmodule
```

Right. The correct answer is B.

In the first field write a module which produces a 4 bit output that counts the number of rising clock edges which have occurred. The output should be produced on the falling edge of the clock. Name this module Q4. In the second field modify module Q2 so that it instantiates module Q4.

```
module Q4;

endmodule
```

```
module Q2;
   reg clock;
   initial begin
       clock = 0;
   #8   forever begin
       #2   clock = 1;
       #2   clock = 0;
       end
   end
endmodule
```

Reset

Answer

```
module Q4 (count, clock);
  input clock;
  output count;
  reg [3:0] count, count_reg;

  initial count_reg = 0;

  always @(posedge clock)
    count_reg = count_reg + 1;

  always @(negedge clock)
    count = count_reg;
endmodule
```

```
module Q2;
  reg clock;
  wire [3:0] count;
  Q4 counter (count, clock);

  initial begin
      clock = 0;
   #8 forever begin
        #2  clock = 1;
        #2  clock = 0;
      end
  end
endmodule
```

Modify module Q4 from the previous screen to use a continuous assign to drive the count output. Name your new module Q5. Hint: take advantage of the fact that the falling edge occurs 2 time units after the rising edge.

```
module Q4 (count, clock);
   input clock;
   output [3:0] count;
   reg [3:0] count, count_reg;

   initial count_reg = 0;

   always @(posedge clock)
      count_reg = count_reg + 1;

   always @(negedge clock)
      count = count_reg;
```

Reset

Done

```
module Q5 (count, clock);
   input clock;
   output [3:0] count;
   reg [3:0] count_reg;

   initial count_reg = 0;

   always @(posedge clock)
      count_reg = count_reg + 1;

   assign #2 count = count_reg;
endmodule
```

Modify module Q5 from the previous screen to use a parameter for the time delay from rising to falling edge. Name this new module Q6.

```
module Q5 (count, clock);
   input clock;
   output [3:0] count;
   reg [3:0] count_reg;

   initial count_reg = 0;

   always @(posedge clock)
      count_reg = count_reg + 1;

   assign #2 count = count_reg;
endmodule
```

Reset

Done

```
module Q6 (count, clock);
   input clock;
   output [3:0] count;
   parameter clktoq = 2;
   reg [3:0] count_reg;

   initial count_reg = 0;

   always @(posedge clock)
      count_reg = count_reg + 1;

   assign #clktoq count = count_reg;
endmodule
```

Given

```
parameter p1 = 4;
reg [4:0] x;
wire [3:2] y;
reg z;
```

Classify these expressions as "constant", "scalar", "delay", "fixed-size", or "non-fixed-size":

These are the correct answers:

1. p1-1	**constant**
2. #x	**delay**
3. x != p1	**scalar**
4. z	**scalar**
5. x*y	**non-fixed-size**
6. x & y	**fixed-size**

To review content related to this question see Expressions.

Given the following values,

```
parameter p1 = 4;
reg [4:0] x;
wire [3:2] y;
reg z;

x = 4'b1011;
y = 2'b10;
z = 1'b1;
and the current time is 5
```

These are the correct answers. Statement 1 is illegal because Stime is a function, not a task. Therefore, it can't stand alone as a statement. Statement 5 is illegal because Sdisplay is a task and can't be used as a right-hand side of an expression.

1. p1 + #(x+y)	**illegal, because a delay expression (#(x+y)) cannot be an operand of another expression.**
2. #($time+p1)	**9**
3. #($time == x)	**0**
4. ~{x, 1'bx}	**5'b0100x**
5. ~{x, 13}	**illegal, because 13 is not sized, and all components of a concatenation must be sized.**
6. x ^ |y	**4'b1010**
7. {x, p1}	**illegal, because p1 is not sized, and all components of a concatenation must be sized.**
8. z === 1'bx	**1'b0**
9. z == 1'bx	**1'bx**
10. 1'bx == 1'bx	**1'bx**

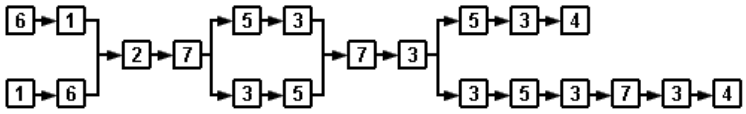
To review content related to this question see Operators.

```
module M;
     reg clock;
     integer x, i;

     initial begin
             x = 1;                  // 1
     #1      x = 2;                  // 2
             for (i=1; i<4; i=i+1)   // 2a
             @(posedge clock)        // 2b
                     if (x == 4)     // 3
                             $finish;        // 4
     end

     always @(posedge clock) begin
             x = x + 1;              // 5
     end

     initial clock = 0;              // 6
     always
     #2 clock = ~clock;              // 7
endmodule
```

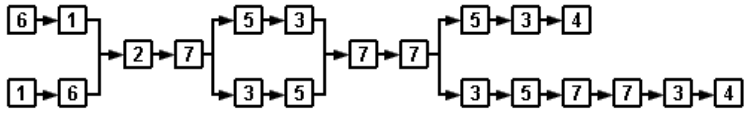
Which of the following diagrams correctly indicates the order of statement execution. Note that, in some places, statement execution order is indeterminate and can proceed down alternative paths. Click Done to check your answer.

---

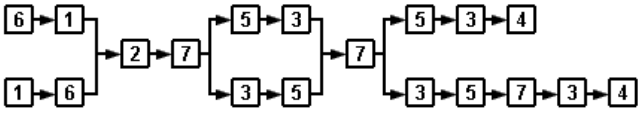
a.



b.



c.



The correct answer is B. To review content related to this question, see Simulation Mechanics.

Given the following module:

```
module M;
     reg clock;
     integer x, i;

     initial begin
             x = 1;                  // 1
     #1      x = 2;                  // 2
             for (i=1; i<4; i=i+1)   // 2a
             @(posedge clock)        // 2b
                     if (x == 4)     // 3
                             $finish;        // 4
     end

     always @(posedge clock) begin
             x = x + 1;              // 5
     end

     initial clock = 0;              // 6
     always
     #2 clock = ~clock;              // 7
endmodule
```

These are the correct answers.

1. What is the value of x at the end of execution? **4**
2. Is the value of x determinate? (Yes/No) **Yes**
3. What is the time at the end of execution? **6 or 10**
4. Is the value of time determinate? (Yes/No) **No**

Given the following module:

```
module M;
  reg clock;
  integer x, y, i;

  initial begin
    x = 1;                        // 1
    forever  @(negedge clock)     // 2
        if (x == 4)               // 3
        $finish;                  // 4
  end

  always @(posedge clock) begin
    x = x + 1;     // 5
  end

  always @(posedge clock) begin
    y = x;         // 6
  end

  initial clock = 0;     // 7
  always
  #2 clock = ~clock;     // 8
endmodule
```

The value of y at the end of execution is indeterminate. That is, it may be either 3 or 4, depending on whether statement 5 or 6 is executed first at each positive clock edge.

Suppose you needed to change only one statement so that so that when this module finishes, y has the same value as x.

There are several correct answers. Line 5 could be changed to any of:
#1 x = x + 1;
x = #1 x + 1;
x <= x + 1;
x <= #1 x + 1;

To review this topic, return to Simulation Mechanics: Concurrency and Parallelism.

Rewrite module abc above so that the input is a 2-bit vector s[1:0], and the output is a 4-bit vector d[3:0]. Call this module vabc. Click Answer to see a solution.

```
module abc (a, b, c, d, s1, s0);
   input s1, s0;
   output a, b, c,d;

   not  (s1_, s1), (s0_, s0);

   and (a, s1_, s0_);
   and (b, s1_, s0);
   and (c, s1, s0_);
   and (d, s1, s0);
endmodule
```

Reset

Answer

```
module vabc (d, s);
   input [1:0] s;
   output [3:0] d;

   not  (s1_, s[1]), (s0_, s[0]);

   and (d[3], s1_, s0_);
   and (d[2], s1_, s[0]);
   and (d[1], s[1], s0_);
   and (d[0], s[1], s[0]);
endmodule
```

Change module abc so that its outputs change 3 time units after the inputs change. Call this module dabc. Click Answer to see a solution.

```
module abc (a, b, c, d, s1, s0);
   input s1, s0;
   output a, b, c,d;

   not  (s1_, s1), (s0_, s0);

   and (a, s1_, s0_);
   and (b, s1_, s0);
   and (c, s1, s0_);
   and (d, s1, s0);
endmodule
```

Reset

Answer

```
module dabc (a, b, c, d, s1, s0);
   input s1, s0;
   output a, b, c,d;

   not  (s1_, s1), (s0_, s0);

   and #3 (a, s1_, s0_);
   and #3 (b, s1_, s0);
   and #3 (c, s1, s0_);
   and #3 (d, s1, s0);
endmodule
```

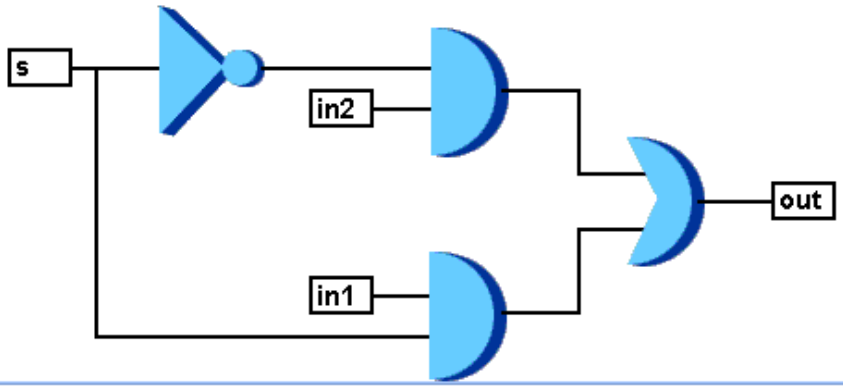
What is the output of the following model? Click Done to check your answer.

```
primitive xxx (w, x, y, z);
        output w;
        input x, y, z;
        table
//      x       y       z       :       w
        0       0       0       :       0;
        0       0       1       :       1;
        0       1       0       :       1;
        0       1       1       :       0;
        1       0       0       :       1;
        1       0       1       :       0;
        1       1       0       :       1;
        1       1       1       :       0;
        endtable

endprimitive

module t;
        reg b, c, d;
        xxx  xxx1 (a, b, c, d);

        initial begin
                b = 0; c = 0; d = 0;
        #1      $display(a);
                b = 1;
        #1      $display(a);
                c = 1;
        #1      $display(a);
                d = 1;
        #1      $display(a);
        end
endmodule
```

- (●) 1. **0 1 1 0**
- ( ) 2. 0 0 0 0
- ( ) 3. 1 0 0 1
- ( ) 4. x x x x
- ( ) 5. none of the above

That's right. To review content related to this question, see User-Defined Primitives.

Write a Verilog module named mod that corresponds to the following schematic (input is in1, in2, s and output is out). Click Done to see a sample solution.



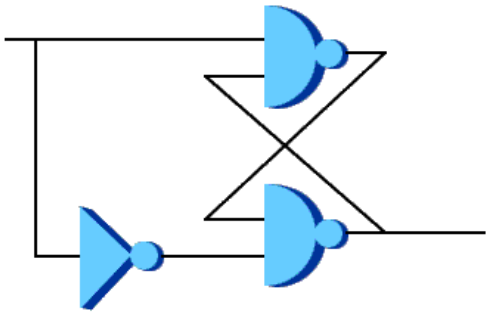

```
module mod (in1, in2, s, out);
   input in1, in2, s;
   output out;

   or (out, o1, o2);
   and (o1, in1, s);
   and (o2, in2 s_);
   not (s_, s);

endmodule
```

Answer

One possible answer is:

```
module mod (in1, in2, s, out);
   input in1, in2, s;
   output out;

   or (out, o1, o2);
   and (o1, in1, s);
   and (o2, in2 s_);
   not (s_, s);

endmodule
```

Write a Verilog module called xyz which corresponds to the following schematic (input is a, output is b). Click the Answer button to see a sample solution.



```
module xyz (a, b);
   input a;
   output b;
   not (a_, a);
   nand (t1, a, b);
   nand (b, a_, t1);
endmodule
```

Answer

```
module xyz (a, b);
   input a;
   output b;

   not (a_, a);
   nand (t1, a, b);
   nand (b, a_, t1);
endmodule
```

Identify the level of the following code fragment.

```
and n1 (a, b, c, d);
or o1 (b, e, f);
or o2 (c, e, g);
not (d, f);
```

- (•) 1. **gate**
- ( ) 2. register transfer
- ( ) 3. behavioral

You got it. This is a gate-level fragment. To review where these constructs were covered, see Chapter 4.

Identify the level of the following code fragment.

```
always #10
           clock = ~clock;
```

- ○ 1. gate
- ○ 2. register transfer
- ◉ 3. **behavioral**

You got it. This is a behavioral-level fragment. To review where this construct was covered, see Simulation Controls: Time Control.

Correct the following code fragment:

assign s #1 = a + b;

Click Done to check your answer.

Done

The correct solution is:

```
assign #1 s = a + b;
```

Correct the following code fragment:

always #10; clock = ~clock;

Click Done to check your answer.

Done

The two possible answers are:

```
always #10 clock = ~clock;
         and
always begin #10; clock = ~clock; end
```

Exercises
Question 5

Complete the following module so that it produces this output:

```
clk     x       y
0       0       0
1       0       0
0       1       0
1       1       0
0       0       1
1       0       1
0       0       0
```

Click Answer to see a sample solution.

```
module test;
        reg x, y, clk;

        always
            #10 clk = ~clk;

        initial begin
                $display("clk    x    y");
                x = 0;  y = 0;
```

Reset

Answer

```
module test;
        reg x, y, clk;

        always
            #10 clk = ~clk;

        initial begin
                $display("clk    x    y");
                x = 0;
                y = 0;
                clk = 1;
                forever
                @clk    $strobe(" %b     %b    %b", clk, x, y);
        end

        initial begin
                @(negedge clk) ;
                @(negedge clk)
                        x = 1;
                @(negedge clk)
                        x = 0;
                @(posedge clk) ;
                @(posedge clk)
                        $finish();
        end

        reg t;
        initial t = 0;
        always @(posedge clk)
                t = x;

        always @(negedge clk)
                y = t;
endmodule
```

Identify the level of the following code fragment. Click Done to check your answer.

```
initial begin
        x = 0;
#10     x = 1;
#5      x = 0;
end
```

- 1. gate
- 2. register transfer
- 3. **behavioral**

That's right.

Identify the level of the following code fragment. Click Done to check your answer.

```
always @(posedge clk or reset) begin
        if (reset)
                q = 0;
        else
                q = d;
end
```

- 1. gate
- 2. **register transfer**
- 3. **behavioral**

Right. This code fragment could be either register transfer or behavioral, though register transfer is a better answer.

Write a gate version of the following assign statement. Click Answer to see a solution.

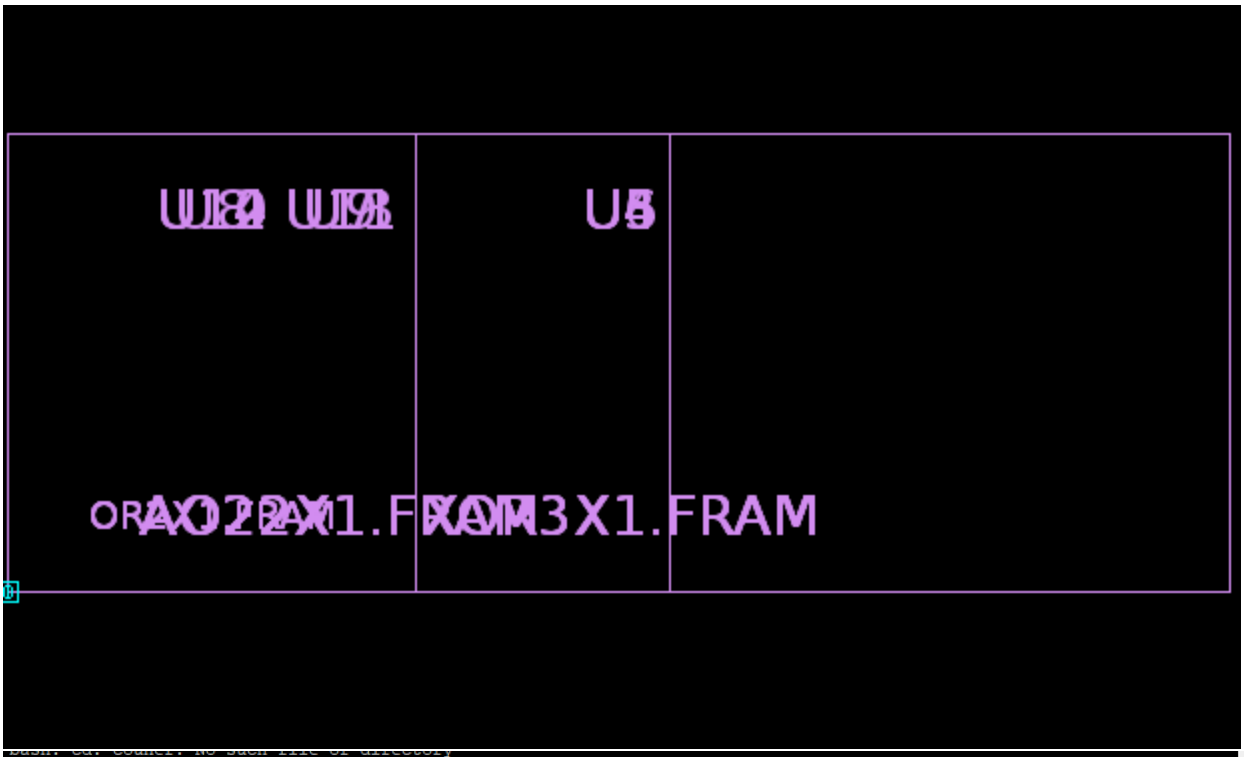
```
wire [1:0] s;
wire a, x, y, z;
assign #1 a = s==1 ? x : s==2 ? y : z;
```

```
or  #1 mux (a, t1, t2, t3);
       not (s0_, s[0]);
       not (s1_, s[1]);
       and (sel1, s1_, s[0]);
       and (sel2, s[1], s0_);
       nor (sel3, sel1, sel2);
       and (t1, sel1, x);
       and (t2, sel2, y);
       and (t3, sel3, z);
```

Answer

One possible solution is:

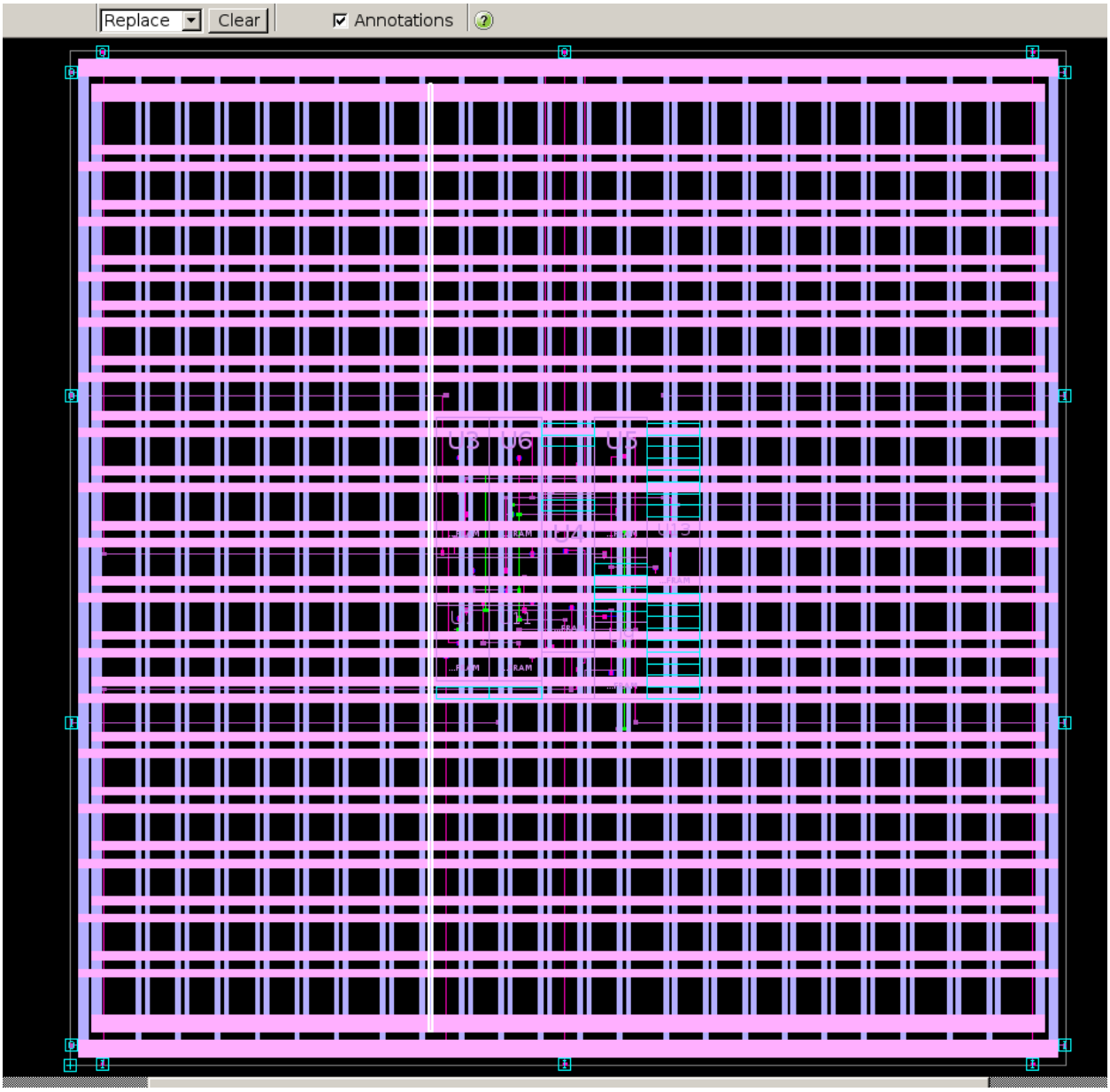
```
or  #1 mux (a, t1, t2, t3);
not (s0_, s[0]);
not (s1_, s[1]);
and (sel1, s1_, s[0]);
and (sel2, s[1], s0_);
nor (sel3, sel1, sel2);
and (t1, sel1, x);
and (t2, sel2, y);
and (t3, sel3, z);
```



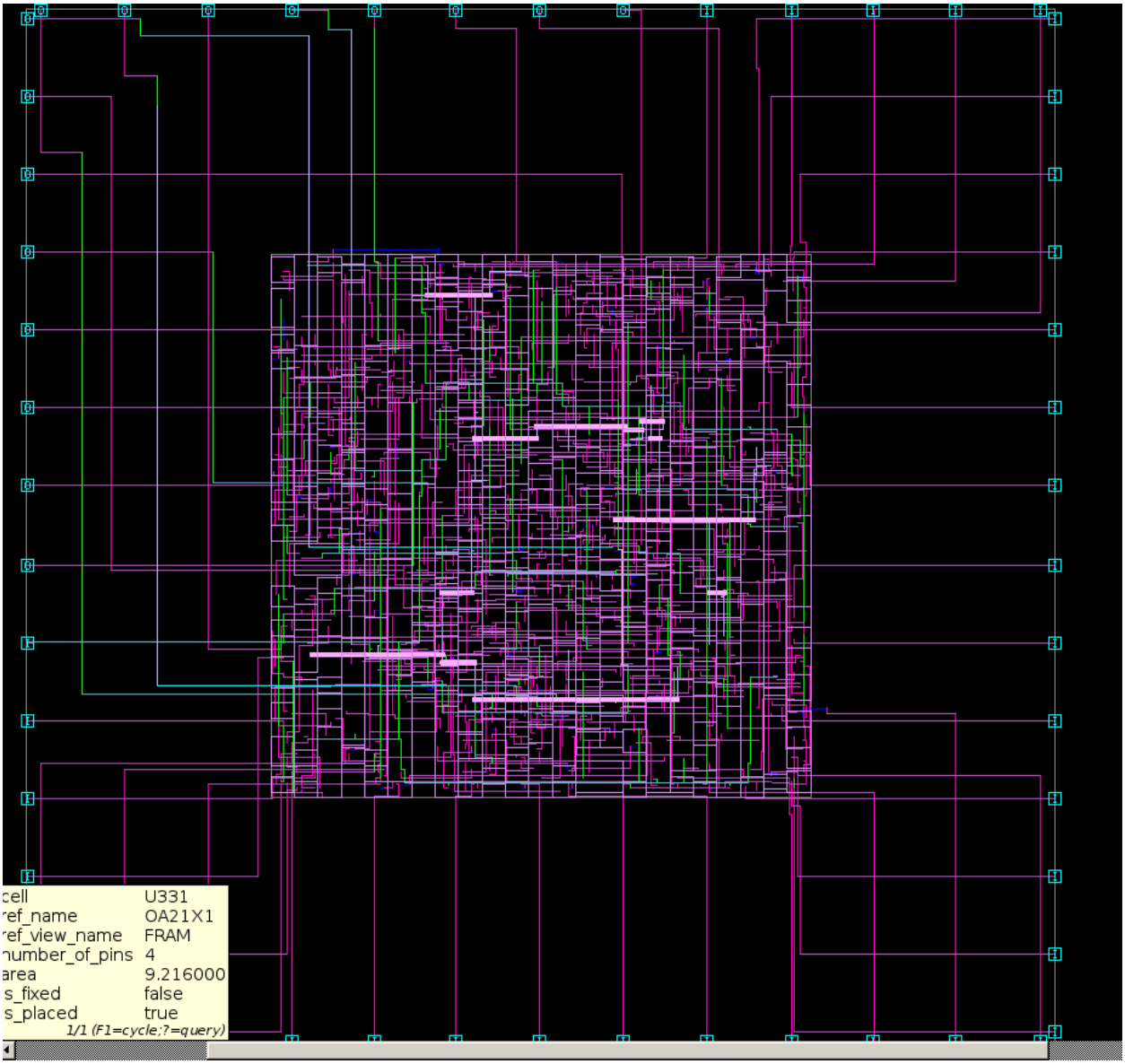

```
-bash-4.2$ cd counter/
-bash-4.2$ ls
counter.dump  counter_tb.v  counter.v  csrc  simv  simv.daidir  ucli.key
-bash-4.2$ vcs counter_tb.v counter.v +v2k

Warning-[LNX_OS_VERUN] Unsupported Linux version
  Linux version 'CentOS Linux release 7.4.1708 (Core) ' is not supported on
  'x86_64' officially, assuming linux compatibility by default. Set
  VCS_ARCH_OVERRIDE to linux or suse32 to override.
  Please refer to release notes for information on supported platforms.

                         Chronologic VCS (TM)
          Version K-2015.09-SP1-1 -- Sun Mar 18 17:38:12 2018
               Copyright (c) 1991-2015 by Synopsys Inc.
                         ALL RIGHTS RESERVED

This program is proprietary and confidential information of Synopsys Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

Parsing design file 'counter_tb.v'
Parsing design file 'counter.v'
Top Level Modules:
       timeunit
       counter_testbench
TimeScale is 1 ns / 10 ps
Starting vcs inline pass...
2 modules and 0 UDP read.
       However, due to incremental compilation, no re-compilation is necessary.
rm -f _csrc*.so linux_scvhdl_*.so pre_vcsobj_*.so share_vcsobj_*.so
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv   -Wl,-rpath-link=./ -Wl,-rpath='$ORIGIN'/simv.daidir/ -Wl,-rpath=./simv.daidir/ -Wl,-rpath='$ORIGIN'/si
mv.daidir//scsim.db.dir  -m32 -m32 -rdynamic   amcQwB.o objs/amcQw_d.o   _prev_archive_1.so  SIM_1.o       rmapats_mop.o
 rmapats.o rmar.o  rmar_llvm_0_1.o rmar_llvm_0_0.o          /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libzerosof
t_rt_stubs.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libvirsim.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/lin
ux/lib/liberrorinf.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libsnpsmalloc.so    /usr/local/synopsys/vcs/K-20
15.09-SP1-1/linux/lib/libvcsnew.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libsimprofile.so /usr/local/synopsy
s/vcs/K-2015.09-SP1-1/linux/lib/libuclinative.so   -Wl,-whole-archive /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/
libvcsucli.so -Wl,-no-whole-archive          /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/vcs_save_restore_new.o /u
sr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/ctype-stubs_32.a -ldl  -lc -lm -lpthread -ldl
../simv up to date
CPU time: .091 seconds to compile + .024 seconds to elab + .114 seconds to link
-bash-4.2$
```

```
Parsing design file 'counter_tb.v'
Parsing design file 'counter.v'
Top Level Modules:
       timeunit
       counter_testbench
TimeScale is 1 ns / 10 ps
Starting vcs inline pass...
2 modules and 0 UDP read.
       However, due to incremental compilation, no re-compilation is necessary.
rm -f _csrc*.so linux_scvhdl_*.so pre_vcsobj_*.so share_vcsobj_*.so
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv   -Wl,-rpath-link=./ -Wl,-rpath='$ORIGIN'/simv.daidir/ -Wl,-rpath=./simv.daidir/ -Wl,-rpath='$ORIGIN'/si
mv.daidir//scsim.db.dir  -m32 -m32 -rdynamic   amcQwB.o objs/amcQw_d.o   _prev_archive_1.so  SIM_1.o       rmapats_mop.o
 rmapats.o rmar.o  rmar_llvm_0_1.o rmar_llvm_0_0.o          /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libzerosof
t_rt_stubs.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libvirsim.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/lin
ux/lib/liberrorinf.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libsnpsmalloc.so    /usr/local/synopsys/vcs/K-20
15.09-SP1-1/linux/lib/libvcsnew.so /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/libsimprofile.so /usr/local/synopsy
s/vcs/K-2015.09-SP1-1/linux/lib/libuclinative.so   -Wl,-whole-archive /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/
libvcsucli.so -Wl,-no-whole-archive          /usr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/vcs_save_restore_new.o /u
sr/local/synopsys/vcs/K-2015.09-SP1-1/linux/lib/ctype-stubs_32.a -ldl  -lc -lm -lpthread -ldl
../simv up to date
CPU time: .091 seconds to compile + .024 seconds to elab + .114 seconds to link
-bash-4.2$ ./simv
Chronologic VCS simulator copyright 1991-2015
Contains Synopsys proprietary information.
Compiler version K-2015.09-SP1-1; Runtime version K-2015.09-SP1-1;  Mar 18 17:38 2018
time=    0 ns, clk=0, reset=0, out=xxxx
time=   10 ns, clk=1, reset=0, out=xxxx
time=   11 ns, clk=1, reset=1, out=xxxx
time=   20 ns, clk=0, reset=1, out=xxxx
time=   30 ns, clk=1, reset=1, out=xxxx
time=   31 ns, clk=1, reset=0, out=0000
time=   40 ns, clk=0, reset=0, out=0000
time=   50 ns, clk=1, reset=0, out=0000
time=   51 ns, clk=1, reset=0, out=0001
time=   60 ns, clk=0, reset=0, out=0001
time=   70 ns, clk=1, reset=0, out=0001
time=   71 ns, clk=1, reset=0, out=0010
time=   80 ns, clk=0, reset=0, out=0010
time=   90 ns, clk=1, reset=0, out=0010
time=   91 ns, clk=1, reset=0, out=0011
time=  100 ns, clk=0, reset=0, out=0011
time=  110 ns, clk=1, reset=0, out=0011
time=  111 ns, clk=1, reset=0, out=0100
time=  120 ns, clk=0, reset=0, out=0100
time=  130 ns, clk=1, reset=0, out=0100
time=  131 ns, clk=1, reset=0, out=0101
time=  140 ns, clk=0, reset=0, out=0101
time=  150 ns, clk=1, reset=0, out=0101
time=  151 ns, clk=1, reset=0, out=0110
time=  160 ns, clk=0, reset=0, out=0110
time=  170 ns, clk=1, reset=0, out=0110
All tests completed sucessfully


$finish called from file "counter_tb.v", line 55.
$finish at simulation time  171.0 ns
           V C S   S i m u l a t i o n   R e p o r t
Time: 171000 ps
CPU Time:      0.310 seconds;       Data structure size:   0.0Mb
Sun Mar 18 17:38:32 2018
-bash-4.2$
```

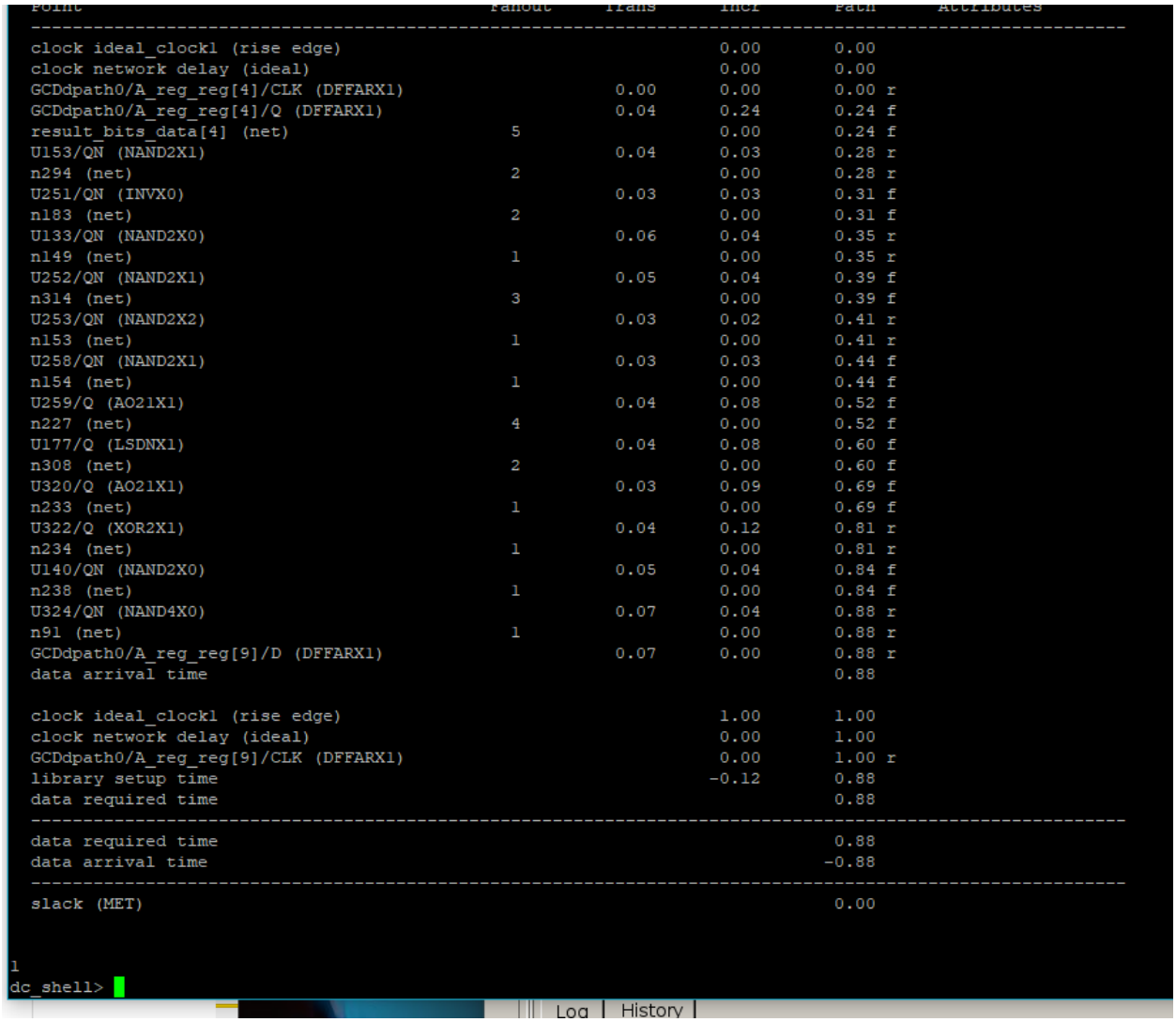
Week 1-



Week 2-


cell U331
ref_name OA21X1
ref_view_name FRAM
number_of_pins 4
area 9.216000
s_fixed false
s_placed true
1/1 (F1=cycle;?=query)

Week 3-

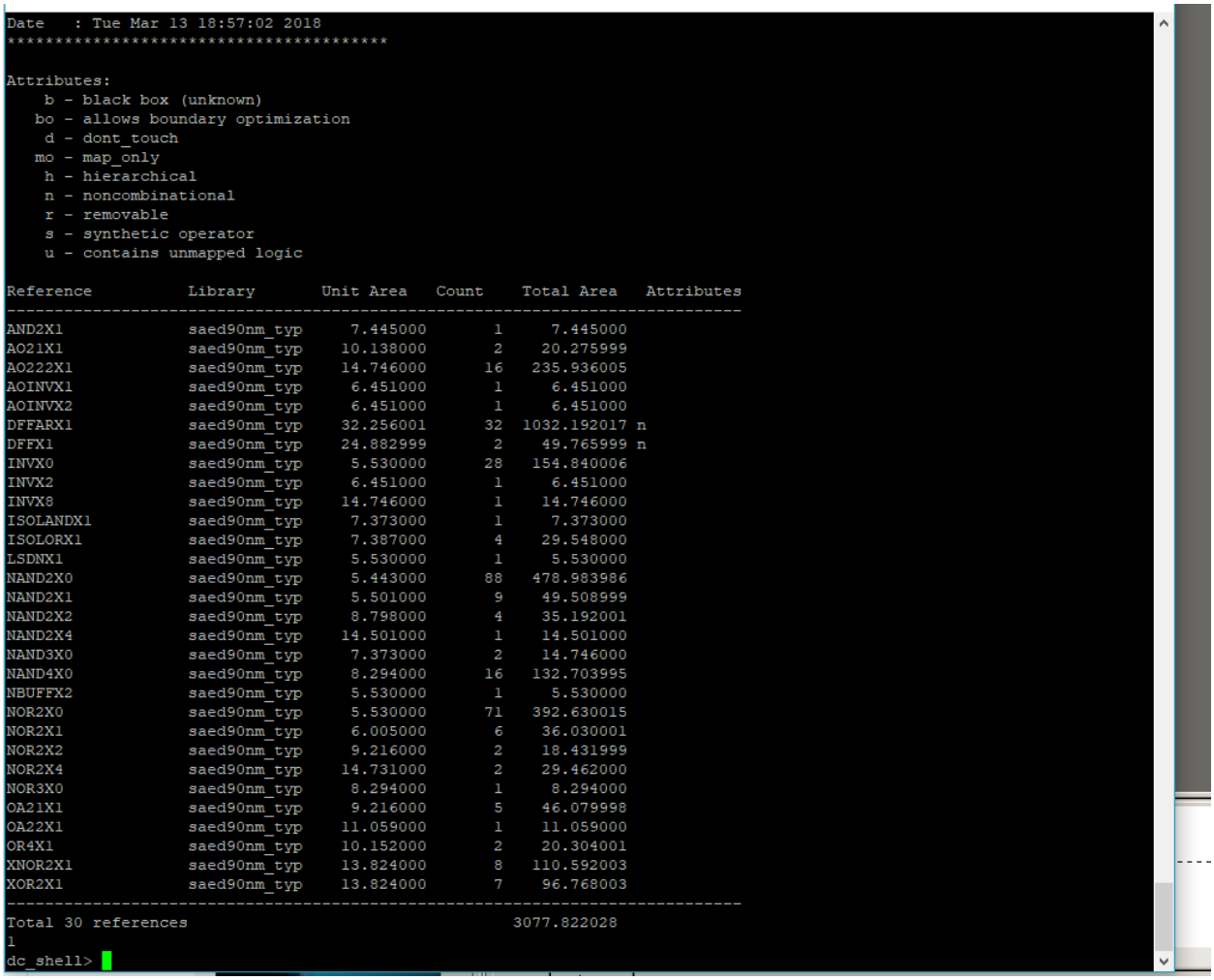


```
  Point                                    Fanout     Trans      Incr       Path      Attributes
  ----------------------------------------------------------------------------------------------
  clock ideal_clock1 (rise edge)                                0.00       0.00
  clock network delay (ideal)                                   0.00       0.00
  GCDdpath0/A_reg_reg[4]/CLK (DFFARX1)                  0.00    0.00       0.00 r
  GCDdpath0/A_reg_reg[4]/Q (DFFARX1)                    0.04    0.24       0.24 f
  result_bits_data[4] (net)                  5                  0.00       0.24 f
  U153/QN (NAND2X1)                                     0.04    0.03       0.28 r
  n294 (net)                                 2                  0.00       0.28 r
  U251/QN (INVX0)                                       0.03    0.03       0.31 f
  n183 (net)                                 2                  0.00       0.31 f
  U133/QN (NAND2X0)                                     0.06    0.04       0.35 r
  n149 (net)                                 1                  0.00       0.35 r
  U252/QN (NAND2X1)                                     0.05    0.04       0.39 f
  n314 (net)                                 3                  0.00       0.39 f
  U253/QN (NAND2X2)                                     0.03    0.02       0.41 r
  n153 (net)                                 1                  0.00       0.41 r
  U258/QN (NAND2X1)                                     0.03    0.03       0.44 f
  n154 (net)                                 1                  0.00       0.44 f
  U259/Q (AO21X1)                                       0.04    0.08       0.52 f
  n227 (net)                                 4                  0.00       0.52 f
  U177/Q (LSDNX1)                                       0.04    0.08       0.60 f
  n308 (net)                                 2                  0.00       0.60 f
  U320/Q (AO21X1)                                       0.03    0.09       0.69 f
  n233 (net)                                 1                  0.00       0.69 f
  U322/Q (XOR2X1)                                       0.04    0.12       0.81 r
  n234 (net)                                 1                  0.00       0.81 r
  U140/QN (NAND2X0)                                     0.05    0.04       0.84 f
  n238 (net)                                 1                  0.00       0.84 f
  U324/QN (NAND4X0)                                     0.07    0.04       0.88 r
  n91 (net)                                  1                  0.00       0.88 r
  GCDdpath0/A_reg_reg[9]/D (DFFARX1)                    0.07    0.00       0.88 r
  data arrival time                                                        0.88

  clock ideal_clock1 (rise edge)                                1.00       1.00
  clock network delay (ideal)                                   0.00       1.00
  GCDdpath0/A_reg_reg[9]/CLK (DFFARX1)                          0.00       1.00 r
  library setup time                                           -0.12       0.88
  data required time                                                       0.88
  ----------------------------------------------------------------------------------------------
  data required time                                                       0.88
  data arrival time                                                       -0.88
  ----------------------------------------------------------------------------------------------
  slack (MET)                                                              0.00


1
dc_shell>
```

```
  slack (MET)                                                                0.00


1
dc_shell> report_area -nosplit -hierarchy

****************************************
Report : area
Design : gcdGCDUnit_rtl
Version: K-2015.06-SP4
Date   : Tue Mar 13 18:55:55 2018
****************************************

Library(s) Used:

    saed90nm_typ (File: /usr/local/synopsys/pdk/SAED90_EDK/SAED_EDK90nm_REF/references/ChipTop/ref/saed90nm_fr/LM/saed90
nm_typ.db)

Number of ports:                          54
Number of nets:                          384
Number of cells:                         317
Number of combinational cells:           283
Number of sequential cells:               34
Number of macros/black boxes:              0
Number of buf/inv:                        34
Number of references:                     30

Combinational area:              1995.864012
Buf/Inv area:                     199.999007
Noncombinational area:           1081.958015
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 3077.822028
Total area:                 undefined

Hierarchical area distribution
------------------------------

                                Global cell area          Local cell area
                                ------------------  ----------------------------
Hierarchical cell               Absolute   Percent  Combi-     Noncombi-  Black-
                                Total      Total    national   national   boxes   Design
------------------------------  ---------  -------  ---------  ---------  ------  ---------
gcdGCDUnit_rtl                  3077.8220    100.0  1995.8640  1081.9580  0.0000  gcdGCDUnit_rtl
------------------------------  ---------  -------  ---------  ---------  ------  ---------
Total                                                1995.8640  1081.9580  0.0000

1
dc_shell>
```

bender.engr.ucr.edu - PuTTY

```
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


--------------------------------------------------------------------------------
                                  Switch   Int      Leak     Total
Hierarchy                         Power    Power    Power    Power    %
--------------------------------------------------------------------------------
gcdGCDUnit_rtl                       0.128    1.124 9.51e+06    1.262 100.0
1
dc_shell> report_power -nosplit -hierarchy

****************************************
Report : power
        -hier
        -analysis_effort low
Design : gcdGCDUnit_rtl
Version: K-2015.06-SP4
Date   : Tue Mar 13 18:56:42 2018
****************************************


Library(s) Used:

    saed90nm_typ (File: /usr/local/synopsys/pdk/SAED90_EDK/SAED_EDK90nm_REF/references/ChipTop/ref/saed90nm_fr/LM/saed90
nm_typ.db)


Operating Conditions: TYPICAL   Library: saed90nm_typ
Wire Load Model Mode: top


Global Operating Voltage = 1.2
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


--------------------------------------------------------------------------------
                                  Switch   Int      Leak     Total
Hierarchy                         Power    Power    Power    Power    %
--------------------------------------------------------------------------------
gcdGCDUnit_rtl                       0.128    1.124 9.51e+06    1.262 100.0
1
dc_shell>
```



```
Date   : Tue Mar 13 18:57:02 2018
****************************************

Attributes:
    b - black box (unknown)
   bo - allows boundary optimization
    d - dont_touch
   mo - map_only
    h - hierarchical
    n - noncombinational
    r - removable
    s - synthetic operator
    u - contains unmapped logic

Reference          Library       Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
AND2X1             saed90nm_typ      7.445000       1      7.445000
AO21X1             saed90nm_typ     10.138000       2     20.275999
AO222X1            saed90nm_typ     14.746000      16    235.936005
AOINVX1            saed90nm_typ      6.451000       1      6.451000
AOINVX2            saed90nm_typ      6.451000       1      6.451000
DFFARX1            saed90nm_typ     32.256001      32   1032.192017 n
DFFX1              saed90nm_typ     24.882999       2     49.765999 n
INVX0              saed90nm_typ      5.530000      28    154.840006
INVX2              saed90nm_typ      6.451000       1      6.451000
INVX8              saed90nm_typ     14.746000       1     14.746000
ISOLANDX1          saed90nm_typ      7.373000       1      7.373000
ISOLORX1           saed90nm_typ      7.387000       4     29.548000
LSDNX1             saed90nm_typ      5.530000       1      5.530000
NAND2X0            saed90nm_typ      5.443000      88    478.983986
NAND2X1            saed90nm_typ      5.501000       9     49.508999
NAND2X2            saed90nm_typ      8.798000       4     35.192001
NAND2X4            saed90nm_typ     14.501000       1     14.501000
NAND3X0            saed90nm_typ      7.373000       2     14.746000
NAND4X0            saed90nm_typ      8.294000      16    132.703995
NBUFFX2            saed90nm_typ      5.530000       1      5.530000
NOR2X0             saed90nm_typ      5.530000      71    392.630015
NOR2X1             saed90nm_typ      6.005000       6     36.030001
NOR2X2             saed90nm_typ      9.216000       2     18.431999
NOR2X4             saed90nm_typ     14.731000       2     29.462000
NOR3X0             saed90nm_typ      8.294000       1      8.294000
OA21X1             saed90nm_typ      9.216000       5     46.079998
OA22X1             saed90nm_typ     11.059000       1     11.059000
OR4X1              saed90nm_typ     10.152000       2     20.304001
XNOR2X1            saed90nm_typ     13.824000       8    110.592003
XOR2X1             saed90nm_typ     13.824000       7     96.768003
--------------------------------------------------------------------------------
Total 30 references                                    3077.822028
1
dc_shell>
```

```
NAND2X0            saed90nm_typ     5.443000       88   478.983986
NAND2X1            saed90nm_typ     5.501000        9    49.508999
NAND2X2            saed90nm_typ     8.798000        4    35.192001
NAND2X4            saed90nm_typ    14.501000        1    14.501000
NAND3X0            saed90nm_typ     7.373000        2    14.746000
NAND4X0            saed90nm_typ     8.294000       16   132.703995
NBUFFX2            saed90nm_typ     5.530000        1     5.530000
NOR2X0             saed90nm_typ     5.530000       71   392.630015
NOR2X1             saed90nm_typ     6.005000        6    36.030001
NOR2X2             saed90nm_typ     9.216000        2    18.431999
NOR2X4             saed90nm_typ    14.731000        2    29.462000
NOR3X0             saed90nm_typ     8.294000        1     8.294000
OA21X1             saed90nm_typ     9.216000        5    46.079998
OA22X1             saed90nm_typ    11.059000        1    11.059000
OR4X1              saed90nm_typ    10.152000        2    20.304001
XNOR2X1            saed90nm_typ    13.824000        8   110.592003
XOR2X1             saed90nm_typ    13.824000        7    96.768003
------------------------------------------------------------------------------
Total 30 references                                  3077.822028
1
dc_shell> report_resources -nosplit -hierarchy

****************************************
Report : resources
Design : gcdGCDUnit_rtl
Version: K-2015.06-SP4
Date   : Tue Mar 13 18:57:15 2018
****************************************


Resource Report for this hierarchy in file ./gcd_dpath.v
===============================================================================
| Cell           | Module         | Parameters | Contained Operations       |
===============================================================================
| sub_x_2        | DW01_sub       | width=16   | GCDdpath0/sub_45 (gcd_dpath.v:45) |
| lt_x_3         | DW_cmp         | width=16   | GCDdpath0/lt_51 (gcd_dpath.v:51) |
===============================================================================


Implementation Report
===============================================================================
|                  |                  | Current           | Set            |
| Cell             | Module           | Implementation    | Implementation |
===============================================================================
| sub_x_2          | DW01_sub         | pparch (area,speed)                |
| lt_x_3           | DW_cmp           | pparch (area,speed)                |
===============================================================================

1
dc_shell>
```

```
Library(s) Used:

    saed90nm_typ (File: /usr/local/synopsys/pdk/SAED90_EDK/SAED_EDK90nm_REF/references/ChipTop/ref/saed90nm_fr/LM/saed9
nm_typ.db)


Operating Conditions: TYPICAL   Library: saed90nm_typ
Wire Load Model Mode: top

<no wire load model is set>

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W



                                  Int      Switch   Leak     Total
Hierarchy                         Power    Power    Power    Power    %
------------------------------------------------------------------------------
gcdGCDUnit_rtl                    1.68e-04 4.76e-05 1.18e-05  2.27e-04 100.0
1
pt_shell>
```

Issues- Week 2 diagram I started from nothing 3 times and could never make it look the same as the one provided in the lab manual. Only issue.