/100 **Name:** ____Eric Hoffman & Chris Frericks____

FINAL EXAM

*Closed book except for one 8½ x 11 sheet. No calculators or electronics devices. All work is to be your own - **show your work** for maximum partial credit.*

1. (10pts)

a. [✓ **True**]/**False:** Clock Tree Synthesis is done outside of the synthesis tool.

b. **True**/[✓ **False:**] When checking min delays one would use a (P,V,T) corner of: (fast, low, cold) Want fastest corner which is highest voltage

c. [✓ **True**]/**False:** The changes the synthesis tool makes to your logic to fix hold times will make it larger and consume more power.
Adding buffers which does make it bigger and higher power

d. [✓ **True**]/**False:** wireload models use fanout and overall design size to estimate the parasitic capacitance and area of the routing.

e. **True**/[✓ **False:**] For loops that use internal timing (things like: @posedge clk) are never synthesizable. Not recommended, but they can be synthesized

f. **True**/[✓ **False:**] A SDF file contains value of the parasitic capacitances extracted from the layout. No, it contains the delays calculated from library & parasitics

g. [✓ **True**]/**False:** Architectural optimizations/choices have a bigger impact on area/speed than gate level optimizations.

h. **True**/[✓ **False:**] In modern digital design it is bad practice to use gated clocks.
You have to use gated clocks for power reasons these days.

i. **True**/[✓ **False:**] The elaboration step of synthesis performs the logic optimizations.

j. **True**/[✓ **False:**] Parasitic capacitances were a big issue in the past, but modern logic processes have solved most those issues.
Parsitics get worse with every process generation.

2. (3pts) In a few sentences explain why is the use of `localparam` is preferred instead of using `define`.

There are a few accepted answers here. Most everyone got full credit on this problem

3. (4pts) In a few sentences explain what the following synthesis commands do

a. `Uniquify`

Treats every instance of a common sub module individually, so it can be optimized to its own constraints/context.

b. `Ungroup`

Smashes (flattens) levels of hierarchy so optimizations can be made across boundaries.

4. (6pts) Rewrite the following code assuming that signal **sel** is a very late arriving signal (Assume that we are totally ok with sacrificing area for delay):

```
always@(*) begin
  if(sel)
    C = A + B;
  else
    C = A - B;

  out = C * D;
end
```

```
  always@(*) begin
    C1 = (A + B)*D;
    C2 = (A - B)*D;
    if(sel)
      out = C1;
    else
      out = C2;
 End
```

Replicate the multiplier so you have both possibilities computed and then select based on *sel*.

5. (4pts) What is one advantage and one disadvantage of FPGAs vs Standard Cells?

I think everyone got full credit here.

6. (6pts) Write a task that takes in two 12-bit numbers **A** and **B** and adds them to produce signals **cout** and **sum**. It should also subtract them to produce a signal called **diff**.

```
task the_dude_abides;

  input [11:0] A,B;
  output [11:0] sum,diff;
  output cout;

  begin
    {cout,sum} = A + B;
    diff = A - B;
  end

endtask
```

7. **(2pts)** Why could the functionality implemented in the task of problem 6 not be done with a Verilog function? **Functions can only return one value**

8. **(6pts)** There are several things wrong with the following module. Point them out.

```
module mux3to1(input [2:0] a, input [1:0] select, output reg y);
 always@ (select, a     )
   case(select)
    2'b0: y = a[0];
    2'b1: y = a[1];
    2'b2: y = a[2];
    default: y = 1'bx;
   endcase
endmodule
```

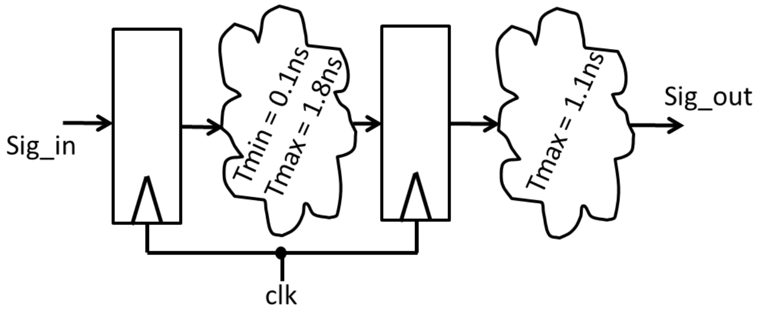
9. **(8pts)** Write a Verilog to model a ROM 16-bit wide 256 entry ROM with the interface and functionality specified in the table. **Note:** contents of ROM should be initialized from contents of a file.

Signal	Direction	Description
clk	In	Clock
cs_n	In	Active low Chip Select (read enable)
addr[7:0]	In	Address from which to read
data[15:0]	Out	Data read from ROM. Read data becomes valid on clock low, should be 16'hffff otherwise.

```
module ROM(clk, cs_n,
addr, data);

input clk,cs_n;
input [7:0] addr;
output [15:0] reg data;

reg mem[15:0][0:255];

initial

$readmemh("filename.txt",
mem);

always @(clk)
  if (!cs_n) && (!clk))
    data = mem[addr];
  else
    data = 16'hffff;

endmodule
```

Many other valid implementations also accepted.

10. **(16 pts)** Use your knowledge of static timing analysis to answer the questions about the following circuit.



Flop Specs:	
clk2q	0.1ns
Tsetup	0.05
Thold	0.05

What would be the minimum clk period the circuit could run?

Clk2q + Tmax + Tsetup = 0.1 + 1.8 + 0.05 = 1.95ns

What is the min delay slack? (margin for worst case hold time)

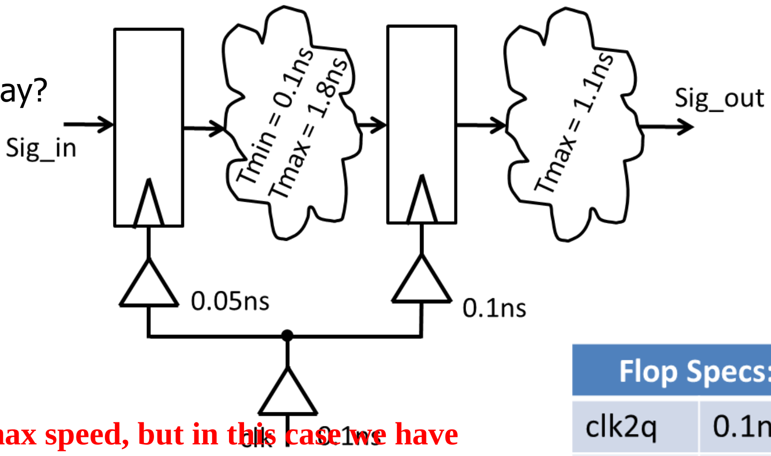
Clk2q + Tmin – Thold = 0.1 + 0.1 – 0.05 = 0.15ns

The output of the block (`sig_out`) is an input to your block. Write the input delay constraint you would use.

set_input_delay –clock clk 1.2 Sig_out

Delay to your input is Clk2q + Tmax = 0.1 + 1.1 = 1.2ns

Now a clock tree is inserted. The delay for each buffer in the tree is annotated next to the gate.



Flop Specs:	
clk2q	0.1ns
Tsetup	0.05
Thold	0.05

What is the average clock insertion delay?

0.175ns

What is the maximum clock skew?

0.05ns

Now what is the minimum period the clock could have?

In the general case clock skew reduces max speed, but in this case we have more time for flop to flop case, So the min period is 1.9ns.

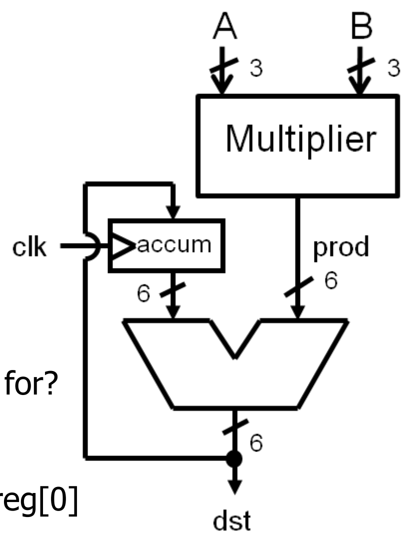
Now what is the min delay slack?

Min delay slack is reduced because the clock to the 2nd flop is later. More likely to have shoot through, so lower margin . 0.15ns – 0.05 = 0.1ns

Why are clock buffering trees used?

If you were to buffer using one giant buffer, and one long route, the RC of the route would kill you. Or if you made the route really wide the area and capacitance of the route would kill you.

11. **(12 pts)** Below two timing reports are given (max_delay & min_delay) for the datapath circuit shown. Answer the following questions:



a. Is the max_delay path a path through a sequential element or is it purely a combinational delay path?

Pure combinational

b. Had an "ungroup –all –flatten" command been issued?

No, you can still see hierarchy in timing paths

c. What do you think the first 2 letters of FA1AM1P stand for?

Full Adder

d. What is the best case (fastest) clk2q timing of accum_reg[0]

0.11ns

e. What is the minimum delay shown through the adder (time units are ns)

0.04ns

f. When the clock uncertainty was increased the min delay slack reported for this path became worse. Is this correct, or is Synopsys being stupid?

This min delay path originates and terminates at the exact same flop, so in this instance Synopsys is being stupid because it is the same flop. Skew occurs between branches of the clock. This is all the same branch.

max_delay

```
Point                                                Incr       Path
------------------------------------------------------------------------
clock clk (rise edge)                                0.00       0.00
clock network delay (ideal)                          0.00       0.00
input external delay                                 0.45       0.45
A[1] (in)                                            0.00       0.45
mult_17/a[1] (mac_DW_mult_uns_0)                     0.00       0.45
mult_17/U33/Z (N1M1P)                                0.03       0.48
mult_17/U31/Z (NR2M1P)                               0.03       0.51
mult_17/U7/S (HA1M1P)                                0.14       0.65
mult_17/U4/S (FA1AM1P)                               0.14       0.79
mult_17/product[2] (mac_DW_mult_uns_0)               0.00       0.79
add_18/A[2] (mac_DW01_add_0)                         0.00       0.79
add_18/U1_2/CO (FA1AM1P)                             0.11       0.89
add_18/U1_3/CO (FA1AM1P)                             0.10       0.99
add_18/U1_4/CO (FA1AM1P)                             0.10       1.08
add_18/U1_5/S (FA1AM1P)                              0.13       1.21
add_18/SUM[5] (mac_DW01_add_0)                       0.00       1.21
result[5] (out)                                      0.00       1.21
data arrival time                                               1.21

clock clk (rise edge)                                2.00       2.00
clock network delay (ideal)                          0.00       2.00
clock uncertainty                                   -0.01       1.99
output external delay                               -0.30       1.69
data required time                                              1.69
------------------------------------------------------------------------
data required time                                              1.69
data arrival time                                              -1.21
------------------------------------------------------------------------
slack (MET)                                                     0.48
```

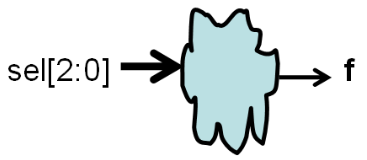
```
Startpoint: accum_reg[0]
            (rising edge-triggered flip-flop clocked by clk
Endpoint: accum_reg[0]
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min
```

min_delay

```
Des/Clust/Port      Wire Load Model       Library
------------------------------------------------
mac                 B0.1X0.1              gflxp

Point                                    Incr       Path
----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
accum_reg[0]/CP (FD2QM1P)                0.00       0.00 r
accum_reg[0]/Q (FD2QM1P)                 0.11       0.11 f
add_18/B[0] (mac_DW01_add_0)             0.00       0.11 f
add_18/U2/Z (EOFM1P)                     0.04       0.14 r
add_18/SUM[0] (mac_DW01_add_0)           0.00       0.14 r
accum_reg[0]/D (FD2QM1P)                 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 uncertainty                        0.01       0.01
accum_reg[0]/CP (FD2QM1P)                0.00       0.01 r
library hold time                        0.10       0.11
data required time                                  0.11
----------------------------------------------------------
data required time                                  0.11
data arrival time                                  -0.14
----------------------------------------------------------
slack (MET)                                         0.03
```

12. **(6pts)** We need to generate signal ***f*** which is a function of a 3-bit input ***sel[2:0]***. Write a verilog **case** statement to implement ***f***. This is DUT code and will be synthesized.

A couple of different correct answers here. Looking for an optimized coding that was insightful in minimizing logic. Like use of don't cares.



When sel = 001, or 110 then f should be a 1. When sel = 100 then f should be a 0. In other cases the function f is not used by the downstream logic.

13. **(6pts)** Given module **`arith1`** synthesized to 190μ^2. How large do you think module **`arith2`** will synthesize to? **Explain your thoughts** use drawings.

Circle one 198μ^2 388μ^2 288μ^2 188μ^2

```
module arith1(a,b,c,sel,result);

input sel;
input [31:0] a,b,c;
output [31:0] result;

assign result = (sel) ? a + b : c;

endmodule
```

```
module arith2(a,b,sel,result);

input sel;
input [31:0] a,b;
output [31:0] result;

assign result = (sel) ? a + b : a - b;

endmodule
```

Depended on your reasoning. But if your reasoning didn't include that Synopsys could and probably would choose an adder/subtractor for the arith2 then you probably got 1.5 points taken off.

14. **(6pts)** Synthesis needs to implement the 3 functions shown. What is the gate effort (sum of fan ins of the gates needed) to implement each function? If Synopsys looks for common terms between the functions how much can the total gate effort be reduced?

F = ab + c
G = abe + ce
H = abe + ce + d

Gate Effort F =	**2+2 = 4**
Gate Effort G =	**3+2+2=7**
Gate Effort H =	**3+2+3=8**

G = Fe
H = G + d

Now to implement G from F you Only need a 2-input gate.

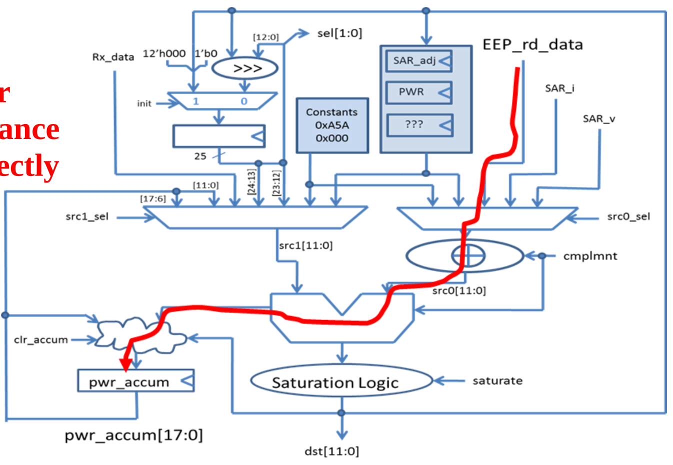
Now to implement H from G you only need a 2 input gate.

Prior cost = 19

After optimization total is 4+2+2 = 8
Save 11 in gate effort

15. **(4pts)** In the project datapath there is a logical path from the eeprom read data input through the ALU, and into the upper bits of the 18-bit power accumulator. So if this path exists why was it valid to use a **`set_false_path`** synthesis directive to ignore timing violations of this path?

The path may exist but our control would never exercise this path because we never had an instance in which data read from the EEPROM was directly involved in accumulation.



16. **(1 pt)** *(Multiple choice)* You have been looking at a chunk of Verilog for greater than 15 seconds, and you are not sure what it will synthesize to…What will be combinational and what will be flops. How many flops are inferred…?

a. The person who wrote this code is obviously a genius.

b. You are a moron and learned nothing from Hoffman's excellent instruction.

c. You are an excellent student but Hoffman is a moron so you learned nothing from ECE 551.

[✓ d.] The code should be rewritten, because if you stare at Verilog block for more than 15 seconds and you can't tell what it will synthesize to then there is something wrong with the code.