ES 4 Exam 2 makeup

Due 23 December 2021, via Gradescope

But *please* don't put it off that long!

Name: _____

Instructions:

1. If you choose to complete this assignment, we will grade it like a normal homework assignment or exam, and will give you points for this assignment or exam 2, whichever is higher (as a percentage of the total number of possible points).
2. You may use your notes, the textbook, or online resources. You may discuss these problems with other classmates or the teaching staff, but all work you submit must be your own.
3. The Gradescope assignment is set up to use a template (like the homework), so your life will be happiest if you do the work on an electronic or physical copy of this document.
4. The VHDLweb problems are just like others you've done before: as long as you get to "TEST PASSED" there is nothing more you need to do to "submit" the problem.
5. The goal of this assignment is to give you a chance to take stock of what you did and did not understand when you took Exam 2, and to learn and practice the parts you're having trouble with. To that end, please start early and get help if you're stuck on something. We're here to help!

Question 1: Exam 2 reflection

(a) Approximately how much time did you spend studying for exam 2?

(b) What did you do you to study for exam 2?

(c) Prior to taking exam 2, how well-prepared did you feel you were?

Question 2: Sequential logic with VHDL

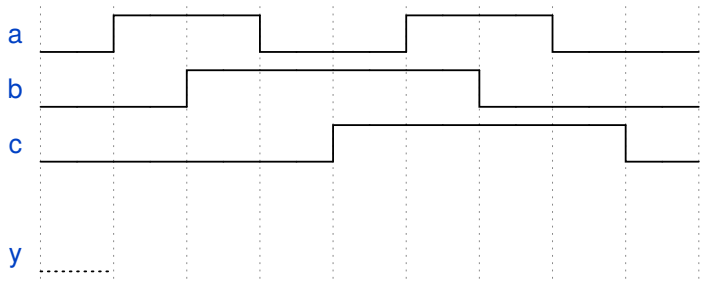
Suppose we have the following VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity makethis is
  port(
    a : in std_logic;
    b : in std_logic;
    c : in std_logic;
    y : out std_logic
  );
end;

architecture synth of makethis is
begin
  process(a,b,c) begin
    if(c = '0') then
      y <= a;
    else
      y <= b;
    end if;
  end process;
end;
```

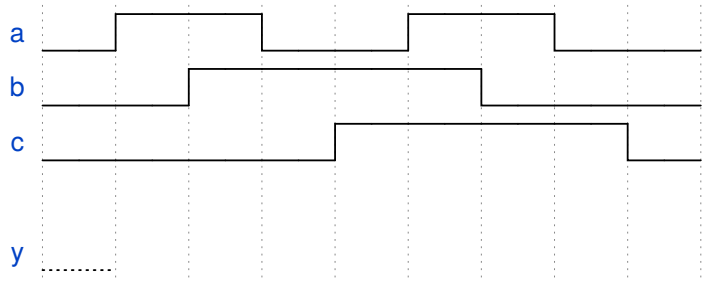
(a) [4 pts] Complete the timing diagram for this design to show what the output `y` will be as a function of `a`, `b`, and `c`.



(b) [2 pts] Can this behavior be implemented with the digital logic components we have studied in this course? If so, draw a logic diagram for a circuit that would implement this behavior. If not, explain why.

Suppose we remove `a` and `b` from the process sensitivity list, leaving only `process(c)`.

(c) [4 pts] Complete the timing diagram for this design to show what the output `y` will be as a function of `a`, `b`, and `c` with the modified sensitivity list.



(d) [2 pts] Can this behavior be implemented with the digital logic components we have studied in this course? If so, draw a logic diagram for a circuit that would implement this behavior. If not, explain why.

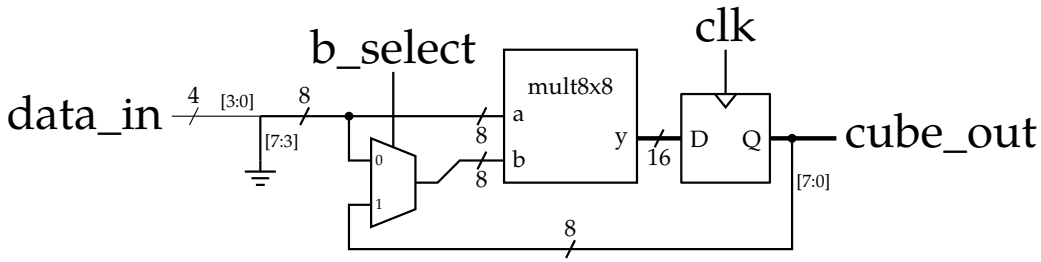
(e) [4 pts] **VHDLweb problem:** Write a VHDL architecture to describe an SR latch.

(f) [2 pts] Is it possible to implement an SR latch on our FPGA (which has only D flip-flops and lookup tables)? Why or why not? *You can try it in Radiant if you're not sure.*

Question 3: Cubing a number

(a) [6 pts] It is common for FPGAs to include a small number of "baked-in" multipliers, because they run faster and use less silicon area than an equivalent multiplier built with fully-reconfigurable LUTs. But because these are a limited resource (the iCE40UP5K has a grand total of 8), it's often important to reuse them for multiple steps of a computation.

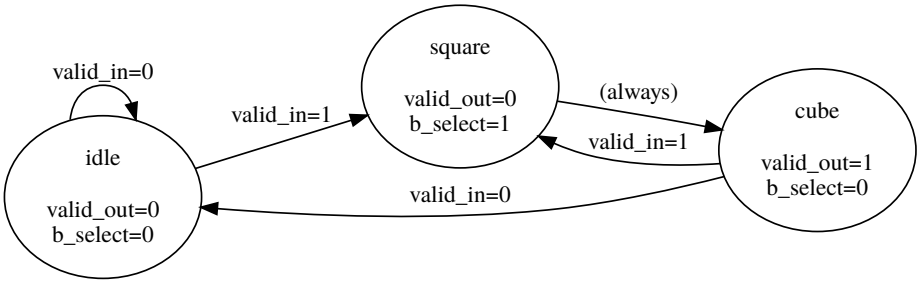
The circuit below is a module which takes an input number and can store the result of a multiply and feed it back in order to compute the cube of an input value. Write VHDL code to create the circuit as shown. The architecture has been started for you.




```
entity cubeit is
  port (
    clk : in std_logic;
    b_select : in std_logic;
    data_in  : in  std_logic_vector(3 downto 0);
    cube_out : out std_logic_vector(15 downto 0)
  );
end;

architecture synth of cubeit is
  component mult8x8 is
    port (
      a : in std_logic_vector(7 downto 0);
      b : in std_logic_vector(7 downto 0);
      y : out std_logic_vector(15 downto 0));
  end component;
```

Below is the state machine which controls the `cubeit` circuit. It waits for an input signal (`valid_in`) indicating that `data_in` is valid and should be cubed, performs the operation (by controlling `b_select`), and asserts an output (`valid_out`) when the cubed result is ready.



(b) [1 pt] Choose a state encoding. A clever choice here will simplify the rest of the problem, so don't be afraid to revisit your choice if the rest is getting ugly.

(c) [4 pts] Write the truth table(s) for the state and the `valid_out` and `b_select` outputs as a function of the current state and the `valid_in` input.

(d) [5 pts] Draw a circuit which implements this state machine.

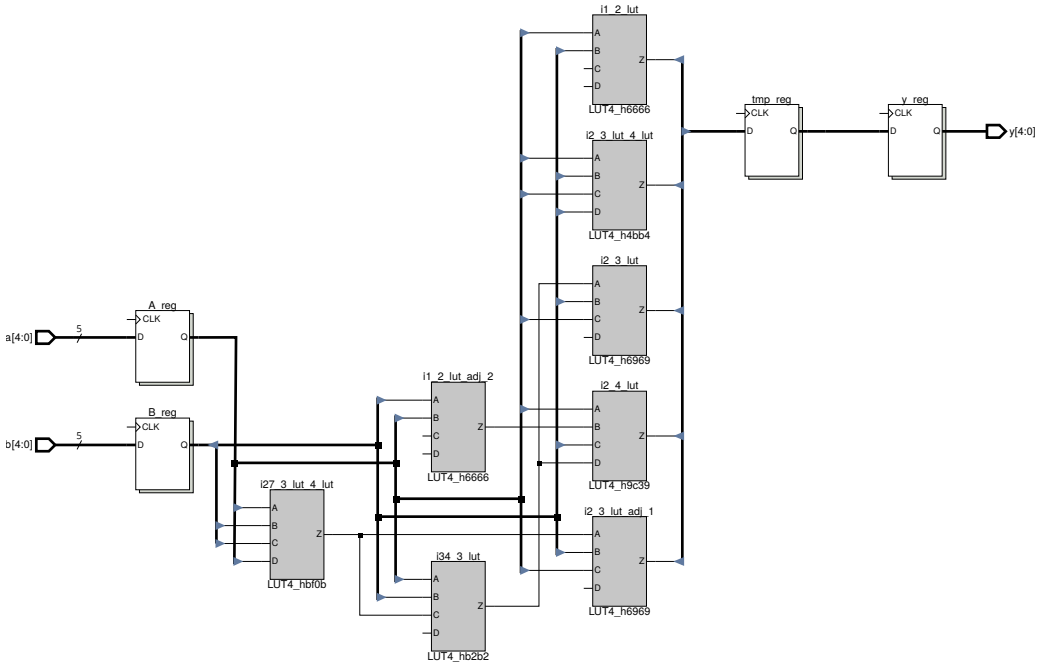
(e) [5 pts] **VHDLweb problem:** Write VHDL code to implement this state machine. You can implement it based on your circuit design, or you can use a behavioral implementation (i.e., creating a state type and using a `case` statement).

Question 4: Timing

The iCE40UP5K FPGA has the following timing characteristics[1]:

Flip-flop clk-to-Q t_{pd}	1.0 ns
Flip-flop setup time	0.2 ns
Flip-flop hold time	0 ns
LUT input-to-output (A/B/C/D to Z) t_{pd}	0.5 ns

The circuit below is an actual FPGA implementation of a 5-bit adder.



(a) [5 pts] What is the maximum frequency this circuit could be clocked at?

[1] These numbers are taken from the Radiant P&R report and rounded for simplicity.

(b) [2 pts] How could you modify the circuit so that it can run at a higher clock frequency? Assume that you can't get a faster FPGA or do anything that would alter the functionality of the circuit. You can also assume that Radiant has done a pretty good job optimizing the combinational logic.[2]

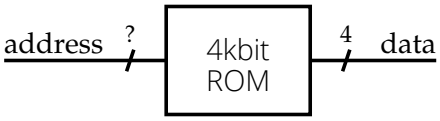
(c) [3 pts] Suppose you are working with some flip-flops with a very long **setup** time. What do you have to be careful of? (i.e., what problems are you likely to run into?)

(d) [3 pts] Suppose you are working with some flip-flops with a very long **hold** time. What do you have to be careful of? Be specific, and make it clear whether/how this is different from part (c).

[2] If you stare at the diagram long enough, you might notice that Radiant has created something like a carry-lookahead adder.

Question 5: Memory

(a) [2 pts] I have a bunch of 4-kilobit (4096-bit) memory blocks, which I've configured to have 4-bit words.[3] How many bits wide is the address input?



(b) [4 pts] Suppose I wanted to turn this into a bit-addressable memory, where each bit has a unique address (or said another way, where a word is just a single bit). Draw a logic diagram showing how you could do this with the memory block above plus some other logic (logic gates, multiplexers, etc.). Make sure to label the address width and make it clear which bits of the bus are going where.

[3] Once again, not hypothetical — the iCE40UP5K FPGA has 30 of these, and the word size is reconfigurable like this.