ECE 448
Final Exam
Wednesday, May 5, 2021

You do not have to print these questions!
You can provide all solutions by writing and drawing on blank pages!

Problem 0

Specify your Day of Birth Number, *dd*, and your Month of Birth Number, *mm*.

The day of month on which you were born is your Day of Birth Number.
The month of year when you were born is your Month of Birth Number.

These numbers are as follows:
Important Note: These numbers will be used only for the purpose of differentiating problems attempted by various students. However, if you do not feel comfortable revealing these numbers due to privacy concerns, you have the right to specify randomly chosen numbers that belong to a valid range.

dd **=** **[must be a number between 1 and 31]**

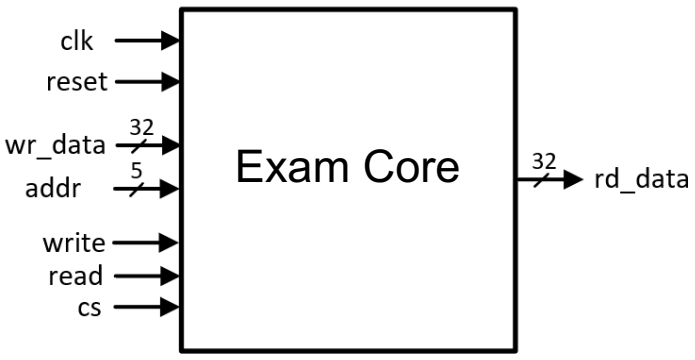
mm **=** **[must be a number between 1 and 12]**

In all following problems, please replace *dd* and *mm* by numerical values defined and specified above.

Problem 1

Analyze the memory map and draw the detailed internal block diagram of an FPro MMIO Exam Core with the following interface and memory map shown on the next page.

Interface:

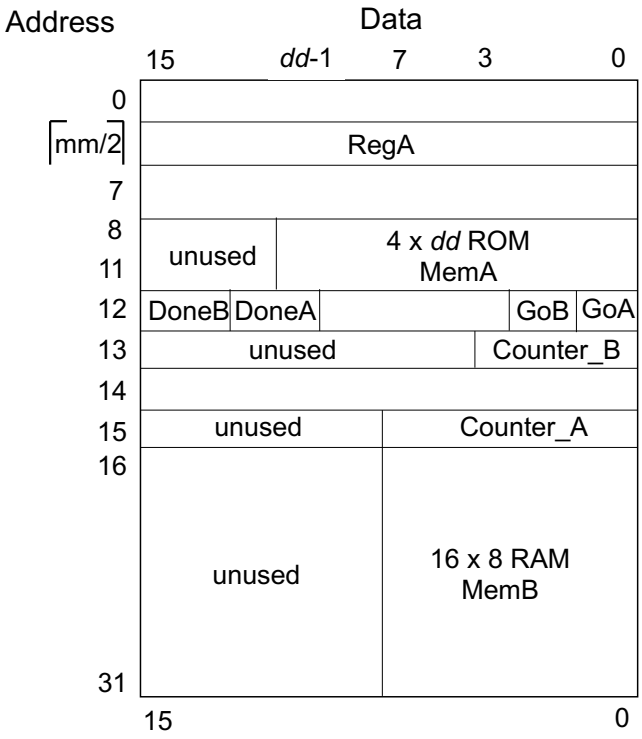


Assumptions:

- **Registers must be implemented using actual registers, not memory. The register RegA can be written to and read from.**
- **Counter_A is an 8-bit counter, Counter_B is a 4-bit counter. Writing to a counter initializes this counter, reading from a counter, reads its current value. Assume that initializing a counter requires only an active value of the input ld.**

- **GoB and GoA are 1-bit write-only flags located at the two least significant bit locations at address 12.**
- **DoneB and DoneA are 1-bit read-only flags located at the two most significant bit locations (15 and 14) at address 12.**

Memory Map:



Problem 2

The circuit Exam is defined as follows:
MEM_A and MEM_B are two single-port memories with synchronous read of the size of 2^{mm} x 32 each. These memories are initialized using a burst mode. The beginnings of the bursts are indicated with active values of inputs `burst_A` and `burst_B`, respectively. The start of calculations is indicated with the active value of the input `s`, which stays active for the entire duration of computations. The results are written to an output FIFO, as soon as this FIFO is not full.
In this code `| S |` denotes an absolute value of `S`.

```
begin:
done = 0
wait for s=1

wait for burst_A=1
for i = 0 to  2^mm-1 do
   MEM_A[i] = dina
end for

wait for burst_B=1
for i = 0 to 2^mm-1 do
   MEM_B[i] = dinb;
end for
```

```
SUM_DIFF = 0

for i = 0 to 2^mm-1 do
     ABS_DIFF = | MEM_A[i] - MEM_B[i] |
     SUM_DIFF = SUM_DIFF + ABS_DIFF
     if ( ABS_DIFF > dd) then
         MEM_B[i] = MEM_A[i]
         MEM_A[i] = ABS_DIFF - dd
    end if
end for

AVR_DIFF = SUM_DIFF / 2^mm

for i = 0 to 2^mm-1  do
   wait for fifo_full = 0
   write MEM_A[i] to fifo
end for

for i = 0 to 2^mm-1 do
   wait for fifo_full = 0
   write MEM_B[i] to fifo
end for

done = 1
wait for s=0
go to begin
```

Assume the following interface to your circuit:

Port	Dir	Width	Meaning
clk	In	1	System clock.
reset	In	1	System reset – clears all internal registers and counters. Active high.
s	In	1	Operating mode: 0 = initialization/reading results, 1 = processing.
dina	In	32	Input data bus for MEM_A
dinb	In	32	Input data bus for MEM_B
burst_A	In	1	Start of the burst at input dina
burst_B	In	1	Start of the burst at input dinb
dout_fifo	Out	32	Output data bus connected to an external FIFO
wr_fifo	Out	1	Writing data to an external fifo
fifo_full	In	1	External FIFO full
dout	Out	32	Output data bus
rd	In	1	External read enable for the output dout. 0 = high impedance on the output bus dout, 1 = AVR_DIFF at the output dout
done	Out	1	Asserted when all results are ready, zero otherwise

Perform the following tasks:

Task 1

Draw a block diagram of the datapath unit of the Exam circuit.
Optimize this circuit for the minimum execution time!
Please clearly mark the widths of all buses in your circuit.

Task 2

Draw an interface of the Exam circuit, with the division into the Datapath and Controller.

Task 3

Draw an ASM chart describing the Controller of the Exam circuit using actions and expressions corresponding (as much as practical) to the operations and conditions of the pseudocode.

Task 4

Draw a second version of the same ASM chart, expressing

a. operations in terms of active values of control signals generated by the Controller.
b. conditions in terms of values of status signals generated by the Datapath.

Task 5

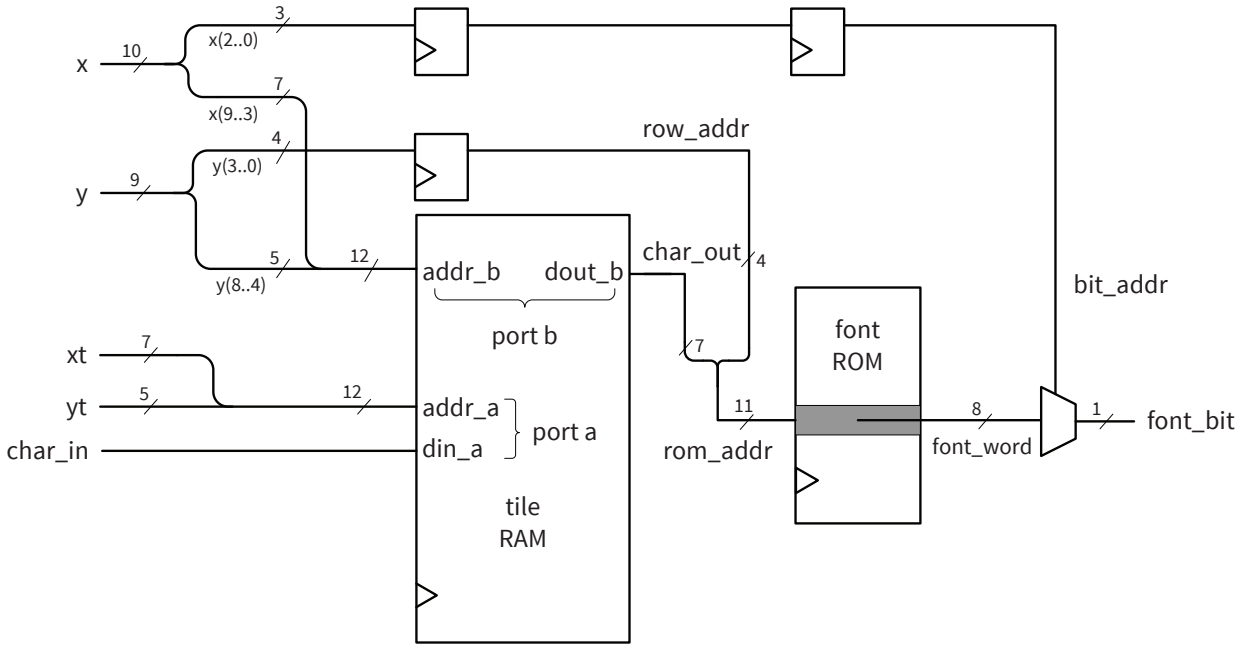
Determine the total number of clock cycles between the external circuit asserting `s` and the Exam circuit asserting `done`. Assume that `burst_A` and `burst_B` are asserted at the same time as `s`, and `fifo_full` is always 0. This number of clock cycles should match your block diagram and ASM chart.

Problem 3

Based on the given below block diagram of the Text Generation Circuit with Tile Memory, please provide the following information:

A. Range of locations in the font ROM used to represent a font pattern of <u>the first character of your last name treated as a capital letter</u>:

B. Address of the location holding ASCII code of the tile with the coordinates x=6*(*mm*-1)+13, y=min(*dd*-1, 29).



Problem 4

Draw a block diagram of the circuit described using the following code.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-------------------------------------------------------------------------
entity dsa is
-------------------------------------------------------------------------
   generic (
      d     : integer := mm;
      k     : integer := 4*mm
   );
   port(
      a     : in  std_logic_vector(k-1 downto 0);
      b     : in  std_logic_vector(k-1 downto 0);
      start : in  std_logic;
      c0    : in  std_logic;
      clk   : in  std_logic;
      ldin  : in  std_logic;
      shift : in  std_logic;
      cout  : out std_logic;
      sum   : out std_logic_vector(k-1 downto 0)
   );
```

```
end dsa;

-------------------------------------------------------------------------
architecture arch of dsa is
-------------------------------------------------------------------------
constant zero_digit : std_logic_vector(d-1 downto 0) := (others =>
'0');

signal   x          : std_logic_vector(d-1 downto 0);
signal   y          : std_logic_vector(d-1 downto 0);
signal   s          : unsigned(d downto 0);
signal   ucin       : unsigned(0 downto 0);
signal   areg       : std_logic_vector(k-1 downto 0);
signal   breg       : std_logic_vector(k-1 downto 0);
signal   sreg       : std_logic_vector(k-1 downto 0);
signal   ci         : std_logic;
signal   cdout      : std_logic;
signal   cdin       : std_logic;
-------------------------------------------------------------------------
begin

input_rega : process(clk)
begin
   if rising_edge(clk) then
      if ldin = '1' then
         areg <= a;
      elsif shift = '1' then
         areg <= zero_digit & areg(k-1 downto d);
      end if;
   end if;
end process;

input_regb : process(clk)
begin
   if rising_edge(clk)  then
     if ldin = '1' then
        breg <= b;
     elsif shift = '1' then
        breg <= zero_digit & breg(k-1 downto d);
     end if;
   end if;
end process;

x     <= areg(d-1 downto 0);
y     <= breg(d-1 downto 0);
cdin  <= c0 when start = '1' else ci;
ucin(0) <= cdin;
s     <= unsigned('0'& x) + unsigned(y) + ucin;
cdout <= s(d);

carry : process(clk)
begin
   if rising_edge(clk) then
      ci <= cdout;
```

```
   end if;
end process;

sumout : process(clk)
begin
   if rising_edge(clk) then
     if shift = '1' then
       sreg <= std_logic_vector(s(d-1 downto 0)) & sreg(k-1 downto d);
     end if;
   end if;
end process;

sum  <= sreg;
cout <= ci;

end arch;
```