ECE 448
Spring 2019
Final Exam (25 points)

First Name:
Last Name:

Problem 1

Task 1:

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

Interface:

```
entity quiz_mmio is
   port(
      clk     : in  std_logic;
      reset   : in  std_logic;
      cs      : in  std_logic;
      write   : in  std_logic;
      read    : in  std_logic;
      addr    : in  std_logic_vector(4 downto 0);
      rd_data : out std_logic_vector(31 downto 0);
      wr_data : in  std_logic_vector(31 downto 0)
   );
end quiz_mmio;
```

Assumptions:

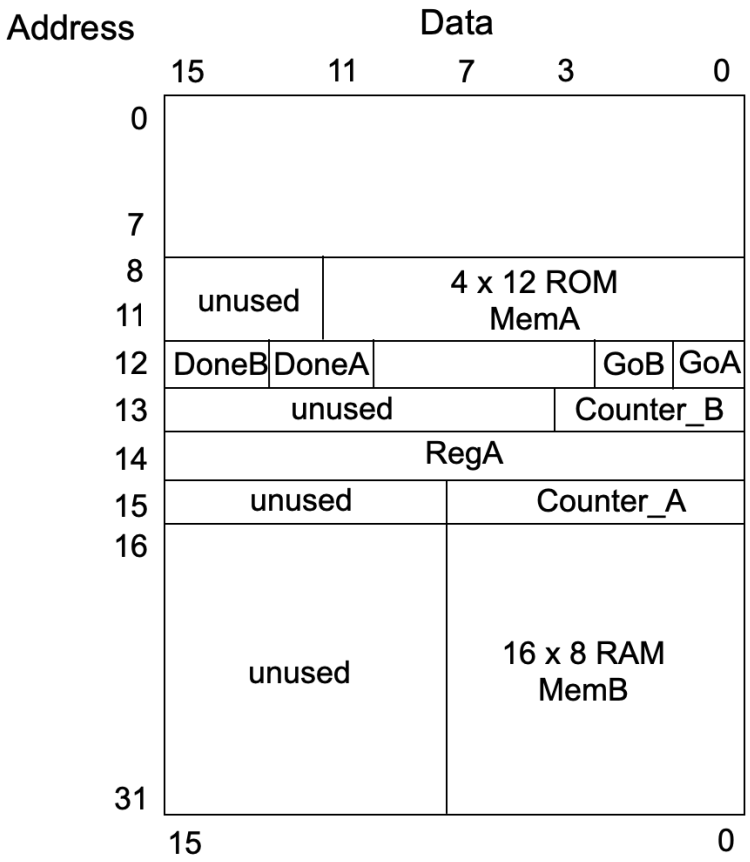
- **Registers must be implemented using actual registers, not memory. The register RegA can be written to and read from.**
- **Counter_A is an 8-bit counter, Counter_B is a 4-bit counter. Writing to a counter initializes this counter, reading from a counter, reads its current value. Assume that initializing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

Task 2: Attempt only after finishing Task 1

Extend your circuit with the following additional internal functionality. Clearly mark with * any additional components you are adding to your circuit to realize this functionality.

- **When GoA is equal to 1, the 8-bit Counter_A should count up every clock cycle. When it reaches 2^8-1 (all ones), the next value should be 0.**
- **When GoB is equal to 1, the 4-bit Counter_B should count down every clock cycle. When it reaches 0, the next value should be 2^4-1 (all ones).**
- **Initializations of counters have priority over counting.**
- **When GoB is equal to 1, the following operation should be performed: MemB[Counter_B]=Counter_A, where MemB is the 16x8 RAM shown in the memory map. This operation has a priority over an external write to MemB.**

Memory Map:



Solution to Problem 1:

Analysis of Memory Map (recommended):

Block diagram (if you need more space use the back side of this page, or ask for an extra blank page):

Problem 2

Draw a block diagram of the circuit performing the computations of the Mandelbrot Fractal algorithm, based on the pseudocode given below. Assume that you can use the MUL, ADD, and SUB units performing the multiplication, addition, and subtraction of fixed-point numbers, respectively. These units accept inputs and produce outputs in the Q4.28 representation, i.e., numbers with 4 bits in the integer part and 28 bits in the fractional part.

pixel_table is an internal RAM of the size equal to the smallest power of 2 greater than or equal to 640*480.

When Go=0, the external circuit can initialize values of internal registers cx and cy (storing numbers in the Q4.28 representation), using ports DataIn, WrInit, XYchoice. When Go=1, the circuit performs computations. When Go=0 again, the external circuit can read the contents of internal memory color_table using ports Addr, Read, PixelOut.

Please clearly mark widths of all buses in your circuit.

Pseudocode:

```
begin:
wait for Go=1
cy0 = -1
cx0 = -2
for i = 40 to 439 do
  for j = 20 to 619 do
     zx = zx0
     zy = zy0
     iteration = 0
     limit = 0
     while( (limit < 4) and (iteration < 500)) do
     {
      zxtemp = zx*zx - zy*zy + zx + cx
      zytemp = 2*zx*zy + zy + cy
      limit = zx*zx + zy*zy
      zx = zxtemp
      zy = zytemp
      iteration++
     }
     if (limit < 4)
       color = 1
     else
       color = 0
     endif
     pixel_table[600*i+j] = color
     cx0 = cx0 + 1/200
  endfor
  cy0 = cy0 + 1/200
endfor
Done=1
wait for Go=0
go to begin
```

Interface:

Assume the following interface to your circuit:

Port	Width	Meaning
Clk	1	System clock.
Reset	1	System reset – clears all internal registers and counters. Active high.
Go	1	Operating mode: 0 = waiting for data/reading results, 1 = processing.
DataIn	32	Input data bus.
WrInit	1	Synchronous write control signal
XYchoice	1	Selection between initializing zx0 and zy0. 0 – initializing zx0, 1 – initializing zy0.
Addr	Please determine by yourself	Address of a memory location in pixel_table.
Read	1	Read enable. 0 = high impedance on the output bus PixelOut, 1 = valid output on the output bus PixelOut.
PixelOut	1	Output data bus used to read results. If Read = 1, PixelOut = pixel_table[Addr], otherwise PixelOut = 'Z' (high-impedance).
Done	1	Asserted when Go=1 and all computations are completed, zero otherwise.

Solution to Problem 2:

Analysis of variables (recommended):

Block diagram (if you need more space use the back side of this page, or ask for an extra blank page):

Problem 3

Perform the following three tasks:

A. **Draw in interface of Mandelbrot Fractal unit, designed as a part of Problem 2, with the division into the Datapath and Controller.**
B. **Draw an ASM chart describing the Controller of the Mandelbrot Fractal unit. Use actions and expressions corresponding (as much as practical) to the operations and conditions of the pseudocode.**
C. **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.**

<u>Solutions to Problem 3:</u>

Part A: Interface with the division into the Datapath and Controller:

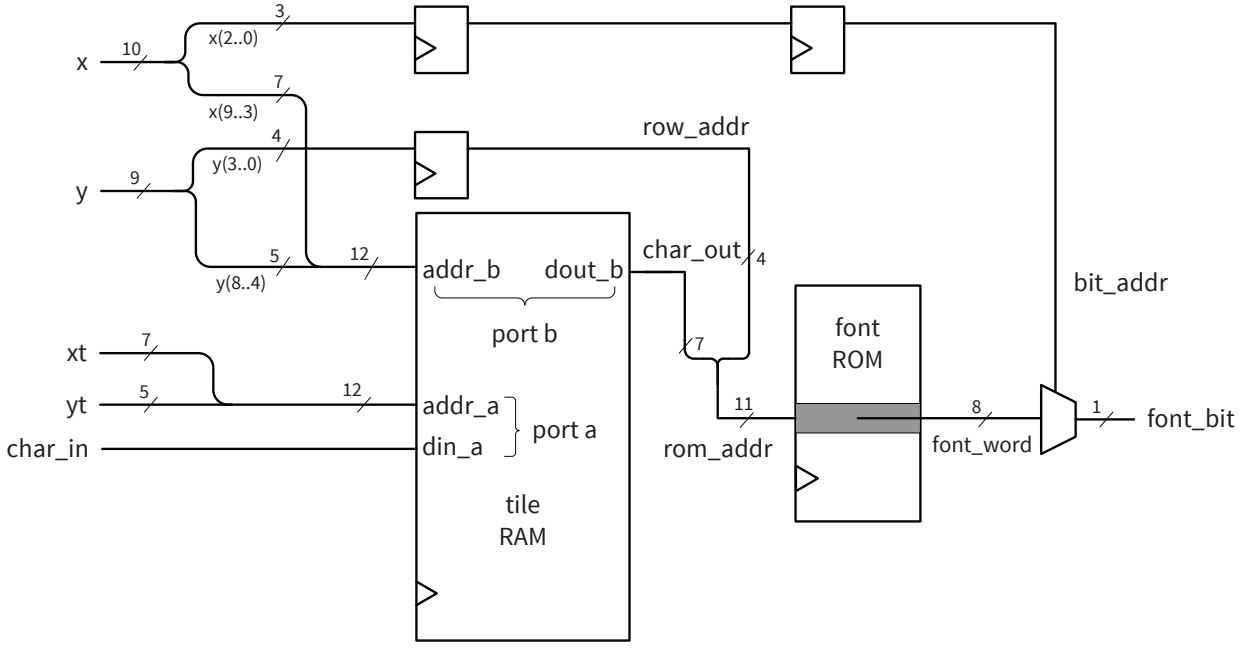
Part B: ASM chart based on the pseudocode

Part C: ASM chart based on active values of control and status signals

Problem 4

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

A. **Type of memory (single-port or dual-port): …….**
B. **Total number of words (number of locations in memory): …….**
C. **Size of each word (a single location in memory): …….**
D. **Total size of memory in kbits: …….**
E. **Number of words used to describe a font pattern of a single character: …….**
F. **Total number of font patterns stored in this memory: …….**
G. **Range of locations in this memory used to represent a font pattern of a character with an ASCII code 80 decimal = 50 hex (this character is P): …….**



How would your answers to questions A.-G. changed if an additional control bit, used to conditionally reverse the foreground and background color of every character of the VGA screen is stored in the tile RAM? Which answers would remain the same, and which would change?

Problem 5

Fill in the blanks in the code of the FPro VGA slot mouse core, described by the following memory map:

Memory Map:

```
0x xxaa aaaa aaaa:  2^10 x 12 Sprite RAM
1x xxxx xxxx xx00:  bit 0: bypass bit
1x xxxx xxxx xx01:  bits 9 downto 0: x0 register
1x xxxx xxxx xx10:  bits 9 downto 0: y0 register
```

Code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity chu_vga_slot_mouse_core is
   generic(
      CD         : integer                       := 12;
      ADDR_WIDTH : integer                       := 10;
      KEY_COLOR  : std_logic_vector(11 downto 0) := (others => '0')
   );

   port(
      clk     : in  std_logic;
      reset   : in  std_logic;
      -- frame counter
      x       : in  std_logic_vector(10 downto 0);
      y       : in  std_logic_vector(10 downto 0);
      -- video slot interface
      cs      : in  std_logic;
      write   : in  std_logic;
      addr    : in  std_logic_vector(..... downto 0);
      wr_data : in  std_logic_vector(CD - 1 downto 0);
      -- stream interface
      si_rgb  : in  std_logic_vector(CD - 1 downto 0);
      so_rgb  : out std_logic_vector(CD - 1 downto 0)
   );
end chu_vga_slot_mouse_core;

architecture arch of chu_vga_slot_mouse_core is
```

```
   signal wr_en      : std_logic;
   signal wr_ram     : std_logic;
   signal wr_reg     : std_logic;
   signal wr_bypass  : std_logic;
   signal wr_x0      : std_logic;
   signal wr_y0      : std_logic;
   signal x0_reg     : std_logic_vector(10 downto 0);
   signal y0_reg     : std_logic_vector(10 downto 0);
   signal bypass_reg : std_logic;
   signal mouse_rgb  : std_logic_vector(CD - 1 downto 0);
   signal chrom_rgb  : std_logic_vector(CD - 1 downto 0);
begin
   -- instantiate sprite generator
   slot_unit : entity work.mouse_src
      generic map(
         CD        => 12,
         KEY_COLOR => (others => '0')
      )
      port map(
         clk       => clk,
         x         => x,
         y         => y,
         x0        => x0_reg,
         y0        => y0_reg,
         we        => wr_ram,
         addr_w    => addr(…… downto 0),
         pixel_in  => wr_data(…… downto 0),
         mouse_rgb => mouse_rgb
      );
   -- registers and decoding
   process(clk, reset)
   begin
      if reset = '1' then
         x0_reg <= ……;
         y0_reg <= ……
      elsif (clk'event and clk = '1') then
         if wr_x0 = '1' then
            x0_reg <= ……;
```

```
         end if;
         if wr_y0 = '1' then
            y0_reg <= ……;
         end if;
         if wr_bypass = '1' then
            bypass_reg <= wr_data(0);
         end if;
      end if;
   end process;
   wr_en     <= '1' when write = '1' and cs = '1' else '0';
   wr_ram    <= '1' when …… and wr_en = '1' else '0';
   wr_reg    <= '1' when …… and wr_en = '1' else '0';
   wr_bypass <= '1' when …… and …… else '0';
   wr_x0     <= '1' when …… and …… else '0';
   wr_y0     <= '1' when …… and …… else '0';
   -- chroma-key blending and multiplexing
   chrom_rgb <= mouse_rgb when mouse_rgb /= KEY_COLOR else si_rgb;
   so_rgb    <= si_rgb when bypass_reg = '1' else chrom_rgb;
end arch;
```