

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Sciences

Introductory Digital Systems Lab (6.111)

Quiz #2 - Spring 2003

Prof. Anantha Chandrakasan and Prof. Don Troxel

Student Name: _____

Problem 1 (24 Points): _____

Problem 2 (30 Points): _____

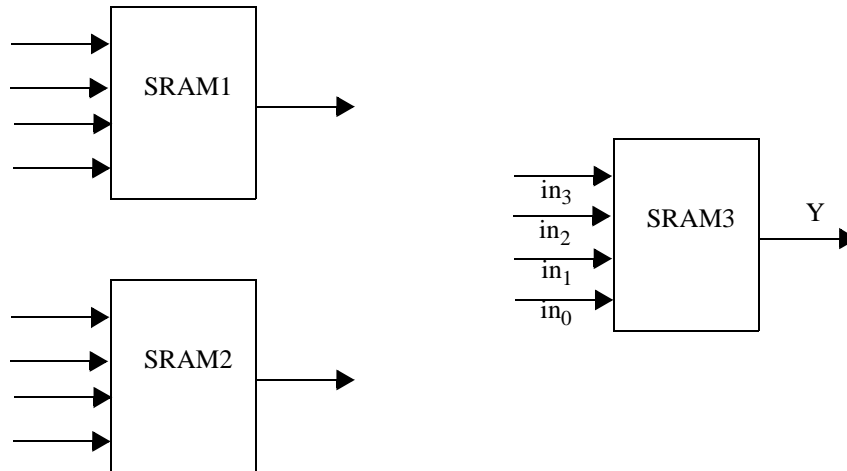
Problem 3 (18 Points): _____

Problem 4 (28 Points): _____

Total (100 Points): _____

Problem 1: FPGA

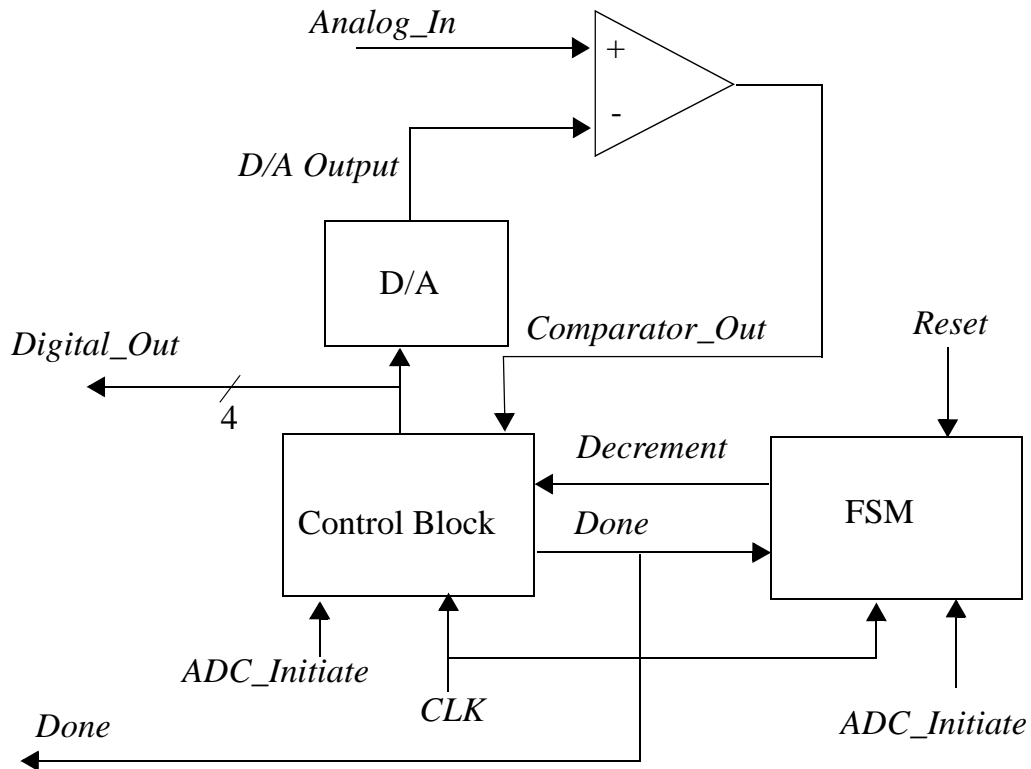
(a) Consider an FPGA architecture with three identical 4-input SRAM based lookup tables. For this problem do not worry about the circuits to perform the write operation into the SRAM. Each SRAM is shown to have 4 inputs (i.e., address bits) and one output bit. Draw the circuit diagram for using the three SRAMs shown below to implement an arbitrary function Y of 5-input variables $A_4 A_3 A_2 A_1 A_0$? Do not add any additional components. **(8 points)**



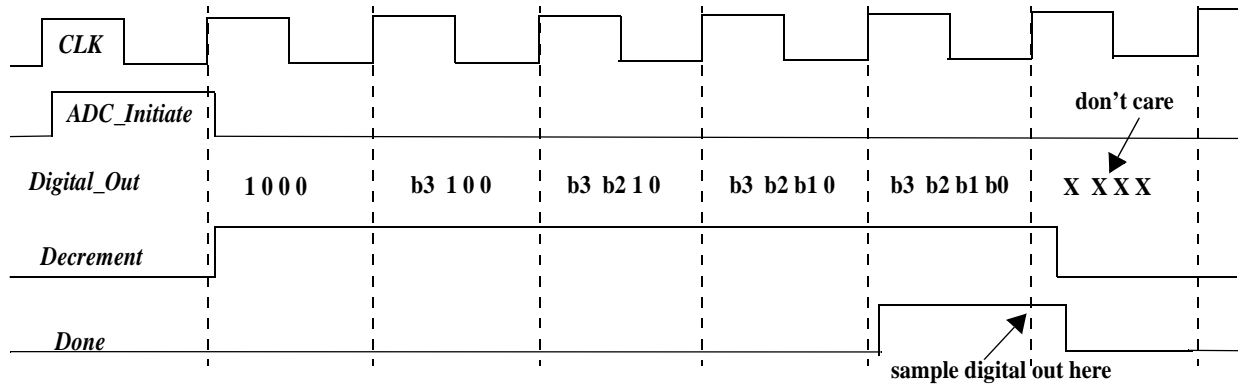
Problem 2: Finite State Machine Design for a Successive Approximation ADC

The block diagram below shows a 4-bit A/D converter using the successive approximation algorithm. The ADC takes in an analog input (*Analog_In*) and converts it to a 4-bit digital output (*Digital_Out*). The signal *Done* is valid for one clock cycle indicating when the *Digital_Out* signal is valid. The *ADC_Initiate* is a one cycle pulse that initiates an A/D conversion cycle. *Reset* puts the FSM in a known IDLE state. *Comparator_Out* is a digital signal that is 1 if $Analog_In > D/A\ output$, else 0.

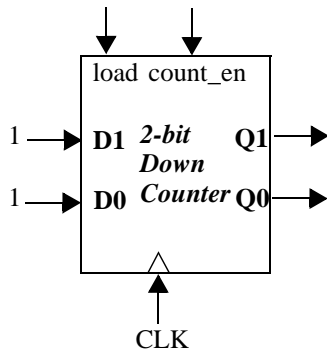
To review, a successive approximation A/D converter is a bit-serial data conversion technique that checks if the analog voltage is in the top or bottom half of a voltage range. Initially, the MSB of the *Digital_Out* is set to 1 with all the lower order bits reset to 0. If the comparator output (which compares the *Analog_In* with the analog representation of the *Digital_Out* signal) is a one then the analog signal is in the top half, else it is in the bottom half. The MSB is reset if it is in the bottom half, otherwise it is left as a one. In the same clock cycle, the next lower bit is set to one to test whether or not the analog voltage is in the top or bottom half of the resulting range. This continues until the last bit has been appropriately determined.



(a) Complete the schematic for the Control Block. The D Flip-Flops copy the D input to the Q output on the rising edge of the clock when EN is high. Set (S) and Reset (R) have priority over Enable (EN). When $S=R=EN=0$, the FF holds state. The inputs to the control block are: CLK , $ADC_Initiate$, $Decrement$ and $Comparator_Out$. The outputs of the control block are: $Done$ and a 4-bit $Digital_Out$. (20 points)

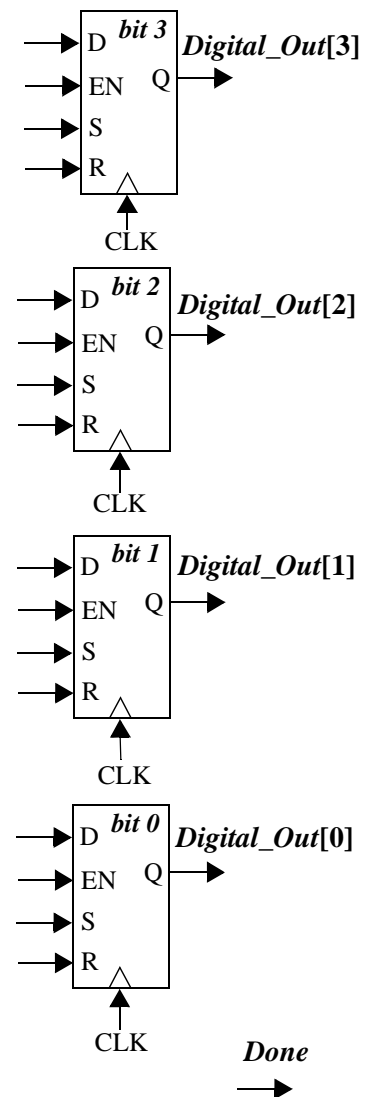


Decrement →



ADC_Initiate →

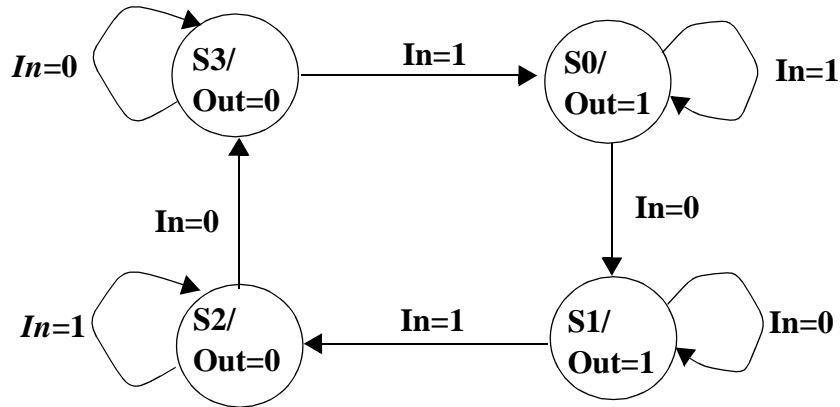
Comparator_Out →



(b) Implement (draw the state transition diagram) the FSM block in the block diagram. Clearly label inputs on arcs and outputs in the states. Assume that the reset input drives the FSM to an IDLE state. The FSM inputs are: *CLK*, *Reset*, *ADC_Initiate*, and *Done*. The FSM output is: *Decrement*. **(10 points)**

Problem 3: FSM/VHDL

Consider the following FSM. OUT should be the output of a flip-flop so it is glitch free.



a) Propose a state assignment such that a minimum number of flip-flops are use to implement the FSM including the output. (4 points)

b) Write the VHDL code for this FSM (including the output) on the next page using the template below. (14 points)

```

library ieee;
use ieee.std_logic_1164.all;
entity fsm is port (
  IN, clk : in std_logic;
  OUT : out std_logic);
end fsm;
architecture state_machine of fsm is
-- the following constants are to assign the state
constant numff : integer := ?;
signal p_s, n_s : std_logic_vector(numff - 1 downto 0);
constant s0 : std_logic_vector(numff - 1 downto 0) := ?;
constant s1 : std_logic_vector (numff -1 downto 0) := ?;
constant s2 : std_logic_vector (numff -1 downto 0) := ?;
constant s3 : std_logic_vector(numff -1 downto 0) := ?;
begin
  OUT <= ?;

  state_clocked:process(clk)
  begin
    .....
  end process state_clocked;

  fsm:process(p_s, IN) -- combinational
  begin -- case
    case p_s is
      when s0 =>
        ??????
        ??????
      end case;
    end process fsm;
  end architecture state_machine;
  
```

Complete the VHDL code below (based on the template provided in the previous page)

```
constant numff : integer :=      ;
signal p_s, n_s : std_logic_vector(numff - 1 downto 0);
constant s0 : std_logic_vector(numff - 1 downto 0) :=      ;
constant s1 : std_logic_vector (numff -1 downto 0) :=      ;
constant s2 : std_logic_vector (numff -1 downto 0) :=      ;
constant s3 : std_logic_vector(numff -1 downto 0) :=      ;
begin
```

```
    OUT <=      ;
```

```
    state_clocked:process(clk)
    begin -- case
```

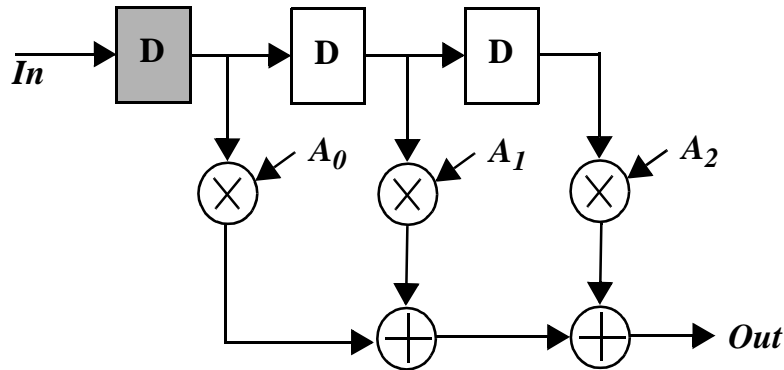
```
end process state_clocked;
```

```
fsm:process(p_s, IN) -- combinational
begin
    case p_s is
        when s0 =>
```

```
        end case;
    end process fsm;
```

Problem 4: Arithmetic Structures and Transformations

- (a) Consider the following Finite Impulse Response filter (similar to the one shown in class). D represents a delay elements (i.e., register). Assume that the delay is ideal with zero CLK to Q delay. A_0 , A_1 , and A_2 are fixed coefficients. Assume that the delay through a multiplier is 10ns and the delay through an adder is 5ns. Assume that the delay through a cascade of logic stages is the sum of the delays through the individual stages. What is the critical path (i.e., what is the minimum clock period) for this filter structure? Highlight the critical path. (4 points)



- (b) Apply the associative transformation to the above structure to re-organize the additions. Draw the new block diagram. (6 points)

(c) Retime the structure in part (b) with the goal of minimizing the clock period. Do not retime the shaded delay element. What is the new critical path after retiming? **(8 points)**

Parts (d) and (e) don't relate to parts (a)-(c).

(d) Consider multiplication between a 16-bit input (In) represented in two's complement and an 8-bit constant number (in magnitude form). Let the constant be 01110110. If the multiplication were to be done using hardwired shifts and adds, how many additions will be required? **(2 points)**

(e) For the constant coefficient given in part (d), transform it using the Canonical Signed Digit (CSD) transform. Assuming that subtractions and additions have the same cost, what is the minimum number of additions/subtractions needed? Show a block diagram of the multiplication with shifters, adders, and subtractors. Assume that appropriate sign-extension have been performed. **(8 points)**