Pre-Lab 6: Finite State Machines and Controllers

EGCP 281: Designing with VHDL
Due Date: Tuesday, December 3, 2019 by 2:00 PM
11 points possible

Luke Bachman

Please submit your completed assignment via Titanium prior to the due date

You may discuss this assignment with others, but this assignment must be completed individually. *Please recall that academic dishonesty will not be tolerated. By submitting this assignment, you understand penalties will be assessed if you submit work for credit that is not your own.*

Solve the following problems. For each problem, cite all references used and students consulted.

1. [5 pts] Review the *Finite State Machine Tutorial* slides on Titanium and answer the following questions:
   - ✗ a What is an unconditional state transition? In the example state transition diagram solution, what state uses an unconditional state transition and why?
   - ✗ b What is the difference between a Mealy output and a Moore output? How are these written differently on a state transition diagram? How do these impact the system's circuitry?
   - + c What is the difference between implementing a finite state machine using structural VHDL vs. behavioral VHDL?
   - ✗ d What is the type statement, and why might this be useful to use in a behavioral VHDL implementation of a finite state machine?
   - X e How many process statements are required in a behavioral VHDL implementation of a finite state machine? What is the purpose for each of these process statements?

X 2. [3 pts] Create a state transition diagram for the single-digit stopwatch controller of **Lab Task 1**. Carefully refer to **Problem 3 of Digilent Real Digital Project 10**. There are three inputs into the system (**Start**, **Stop**, and **Increment**) as well as a **clock** and **asynchronous reset**. The controller will produce only one output, **Run**, which will control the clock enable input to a 4-bit counter. Note that the controller will not keep track of the stopwatch's value (the counter will do this). Instead the controller will determine if the counter should, or should not, count. As you design your state transition diagram, ensure that the system will work properly and ensure that the sum and exclusion rules are obeyed.

✗ 3. [3 pts] Write the VHDL code to implement the finite state machine from **Problem 4 Top (with outputs RED and GRN) of Digilent Real Digital Exercise 10**.

Luke
Bachman
12/10/19

Pre-Lab 6: Finite State Machines & Controllers

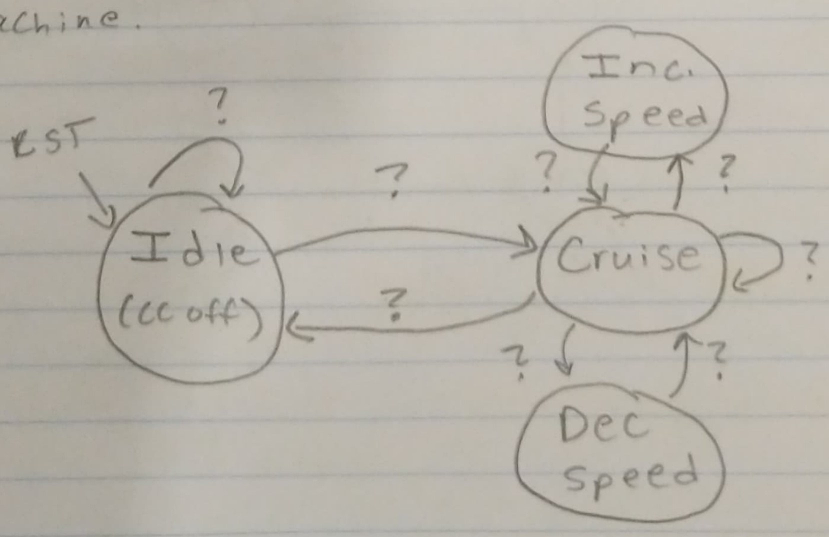
1) a) An unconditional state transition is a transition that can take place immediately after the actions of the presently active state have been accomplished. Such unconditional transitions are named default transitions. The transition condition of an unconditional (default) transition has a permanent value of true and can be expressed as:

T_3_4: Default

or T_3_4: TRUE

In the example state transition diagram solution of a simple cruise control system for a car it contains unconditional state transitions because it just hops from one state to the next until it resets at the starting state.

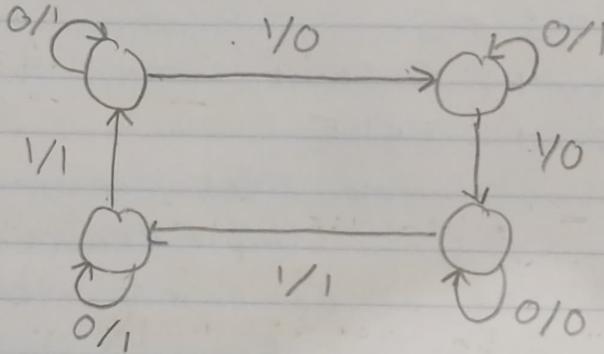
* Example Moore machine solution
- This solution (w/ transition conditions not shown) is a Moore machine.



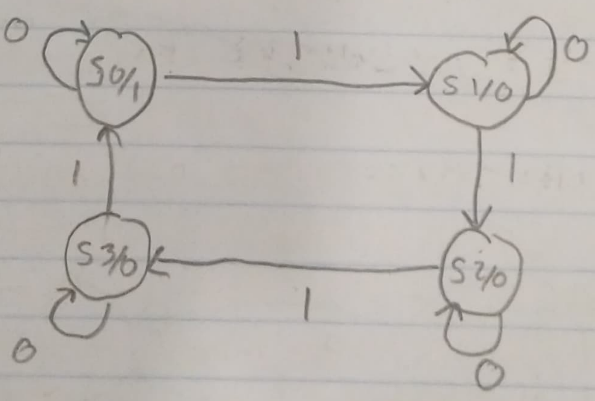
ⓑ Difference between Mealy machine and Moore machine.

Mealy Machine - A mealy machine is defined as a machine in theory of computation whose output values are determined by both its current state and current inputs. In this machine atmost one transition is possible.

Diagram -



Moore Machine - A moore machine is defined as a machine in theory of computation whose output values are determined only by its current state.



Luke
Bachman
12/11/19

Moore machine

① Output depends only upon present state.
② If input changes, output does not change.
③ More number of states are required.
④ There is more hardware requirement.
⑤ They react slower to inputs (one clock cycle later)
⑥ Synchronous output and state generation.
⑦ Output is placed on states.
⑧ Easy to design.

Mealy machine

① Output depends on present state as well as present input
② If input changes, output also changes.
③ Less number of states are required.
④ There is less hardware requirement
⑤ They react faster to inputs
⑥ Asynchronous output generation.
⑦ Output is placed on transitions.
⑧ It is difficult to design.

ⓒ Difference b/w implementing a finite state machine using "Structural" VHDL vs. "Behavioral" VHDL.

Structural Design

- This is a functional solution, but tough to read and modify.
- This approach is not typically used in industry
  - There is a more concise approach
- A more readable and concise method is to design the state register and next state logic from the state diagram instead of

the System Circuitry

Complete VHDL Solution: "Structural Design" of Entire Finite State Machine.

```
process (CLK, RESET)
begin
   if (RESET ='1') then
         presentState <= "00";
   elsif(rising_edge(CLK)) then
         present State <= nextState;
   end if;
end process;
nextState(1) <= Accel and CC and presentState(0)
                  and (not presentState(1));
nextState(0) <= CC or presentState(1);
INC <= presentState(1);
HOLD <= presentState(0);
DEC <= (not presentState(1)) and presentState(0)
         and Coast;
```

Behavioral VHDL

- This is the highest level of "abstraction" and when writing behavioral code, we simplify the need to define the relationships b/w input and output w/o specifying anything about how those relationships will be implemented.

Best (Industry-Grade) Behavioral VHDL Design of Entire Finite State Machine

```
process(CLK, RESET)
begin
  if(RESET = '1') then
    State <= st_Idle;
    HOLD <= '0';
    INC <= '0';
    DEC <= '0';
  elsif(rising-edge(CLK)) then
    Case state is
        when st_Idle =>
            if(CC='1') then
                state <= St_cruise;
            else
                State <= St_Idle;
            endif;
            HOLD  <= '0';
            INC   <= '0';
            DEC   <= '0';
      when St_Inc =>
          State <= St Cruise;
          HOLD <= '1';
          INC <= '1';
          DEC <= '0';
    when   St_Cruise =>
          if (CC = '0') then
              State <= St_Idle;
          elsif (Accel = '1' and CC = '1') then
              State <= st_Inc;
          else --Accel = '0' and CC = '1'
```

```
            State <= st_Cruise;
        end if;
        HOLD <='1'
        INC  <='0'
        DEC  <= coast;
    When others=>
      -- Shouldn't get here
        State <= st_Idle;
        HOLD<='0';
        INC<='0';
        DEC <='0';
    end case;
    end if; -- rising-edge of the CLK
    end process;
```

To Recap: Behavioral VHDL reduces the amount of work you have to do and reduces the amount of errors that occur. Behavioral is a more readabe and concise method and can be defined from the state diagram instead of system circuitry

(d) The type Statement is Similar to an enum. in Java or C type

```
type my States_t is
        st_Idle, -- st Idle is the name
                    of a State
        St_Cruise,
        St_Inc
);
```

- You could use other names for the type

Luke
Bachman
12/15/19

and States, but the point is to improve readability, using state names instead of state numbers.

ⓔ Two process Statements are required in behavioral model. * The first process is used to update state register at rising edge of every clock pulse.
* The second process block is used to implement next state expression and output for all possible state. However it is possible to model using single process block. Moore Machines are modeled using three process blocks.

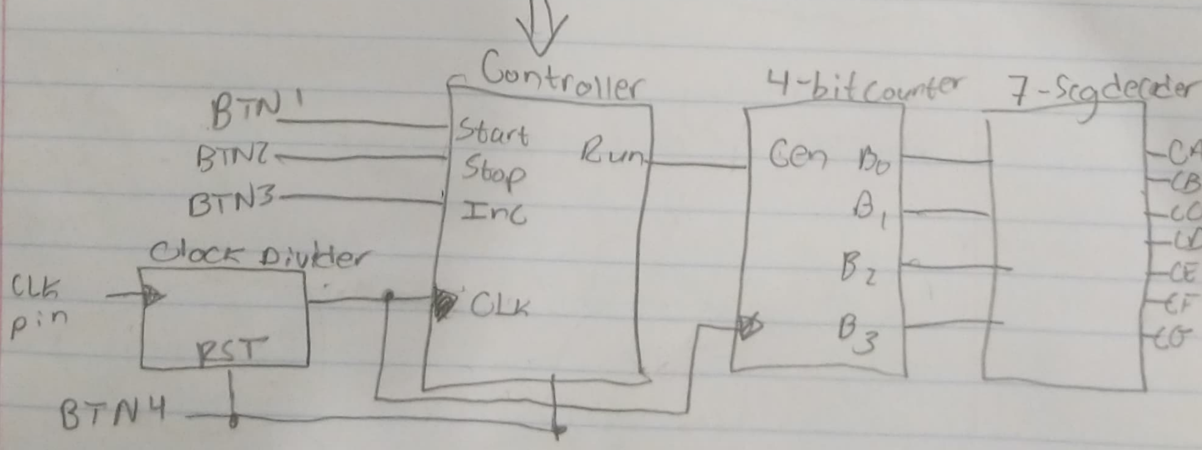
②

<u>Inputs:</u>

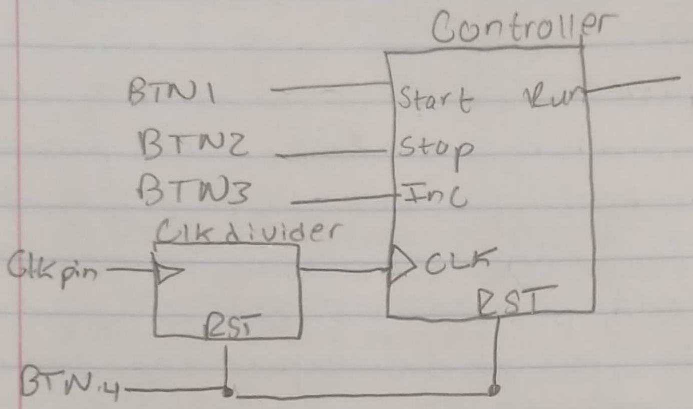
- Start
- Stop
- increment
- CLK
- Asynchronous RST

<u>Output:</u>

- Run



Single-digit Stopwatch Controller.



Inputs:
- Start
- Stop
- Increment
- Asyncronaus RST
- CLK

Output:
- Run

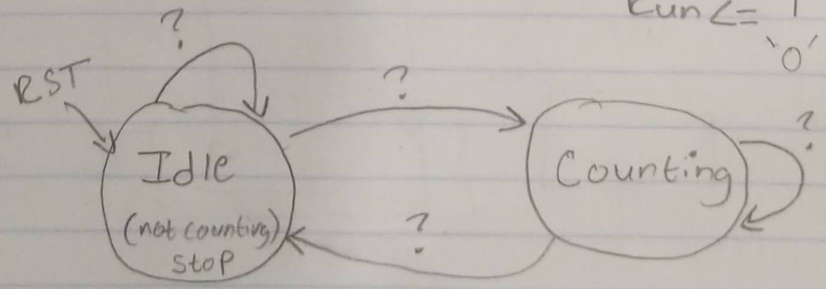
STEPS of Design

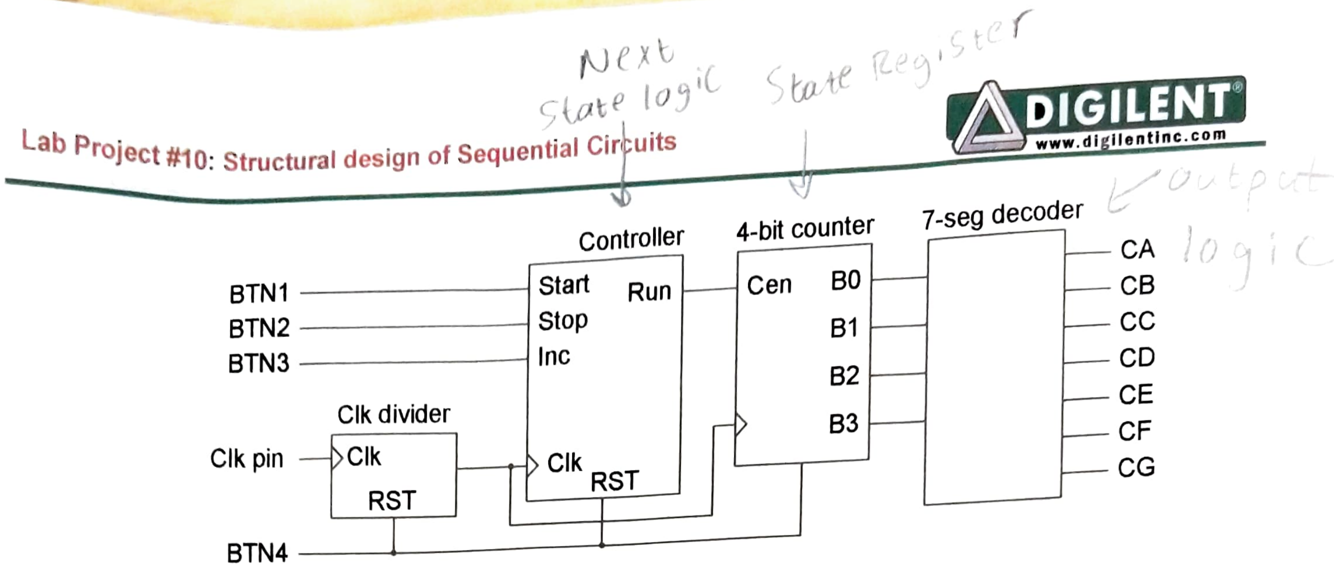
1. State Equations
2. State Table
3. State Diagram ←
4. Controller is responsible for deciding whether the Counter should or should not count.

States

- Counting
- Idle (Not Counting)

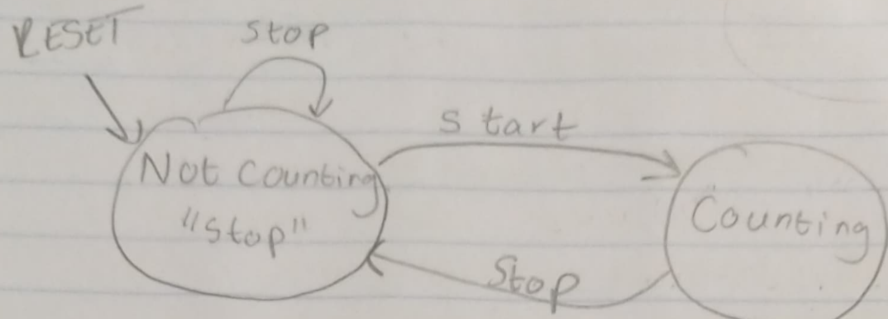
Run <= '1' if increment
'1' if Start
'0' if Stop

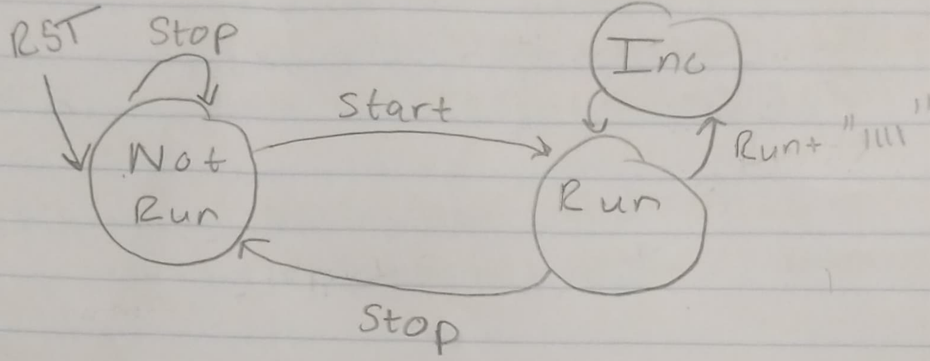




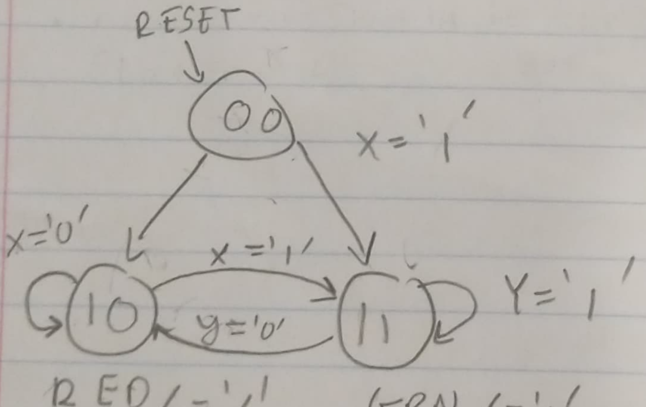
Problem 4. Modify your circuit so that all four digits on the seven-segment display are driven, and drive the least-significant digit so that it changes at a rate of once per millisecond (and so, the most-significant bit will change at a rate of once per second). This circuit requires a scanning display controller which is discussed in your board's Reference Manual. When complete, demonstrate your circuit to the lab assistant and print and submit your source files for credit. Also create and submit a detailed block diagram of your circuit showing all circuit blocks and signal connections (and make sure to appropriately label all circuit blocks and signals).

Luke
Bachman





③ Sketch circuits for the state machines below.



* Make sure that if statements are fully defined.
  - They are either if-else or if-elsif-else

* Best Industry Standard
Behavioral VHDL FSM. Type statement
was not used to redefine states

Luke
Bachman

VHDL Code for State Machine

```
library IEEE;
use IEEE.STD-Logic-1164.ALL
entity FSM is
    port ( X: in STD-LOGIC;
           Y: in STD_LOGIC;
           RED: out STD-LOGIC;
           GRN: out STD-LOGIC );
end FSM;
architecture Behavioral of FSM is
signal State: STD-LOGIC-VECTOR(1 downto 0);
begin
process(CLK, RESET)
   begin
      if(RESET = '1') then
         State <= "00";
         RED <= '0';
         GRN <= '0';
      elsif( rising-edge(CLK)) then
         case State is
            when "00" =>
               if(X='0') then
                  State <= "10";
               else
                  State <= "11";
               end if;
         RED <= '0';
         GRN <= '0';
            when "10" =>
               if(X='1') then
                  State <= "11";
               else
                  State <= "10";
```

* CLK & RESET are also inputs

```
    RED <= '1';
    GRN <= '0';
      when "11" =>
        if (y = '1') then
          State <= "11";
        else
          State <= "10";
        end if;
   RED <= '0';
   GRN <= '1';
      when others =>
      -- Shouldn't get here
      State <= "00";
      RED <= '0';
      GRN <= '0';
     end case;
    end if; -- rising edge of CLK
   end process;
  end behavioral;
```