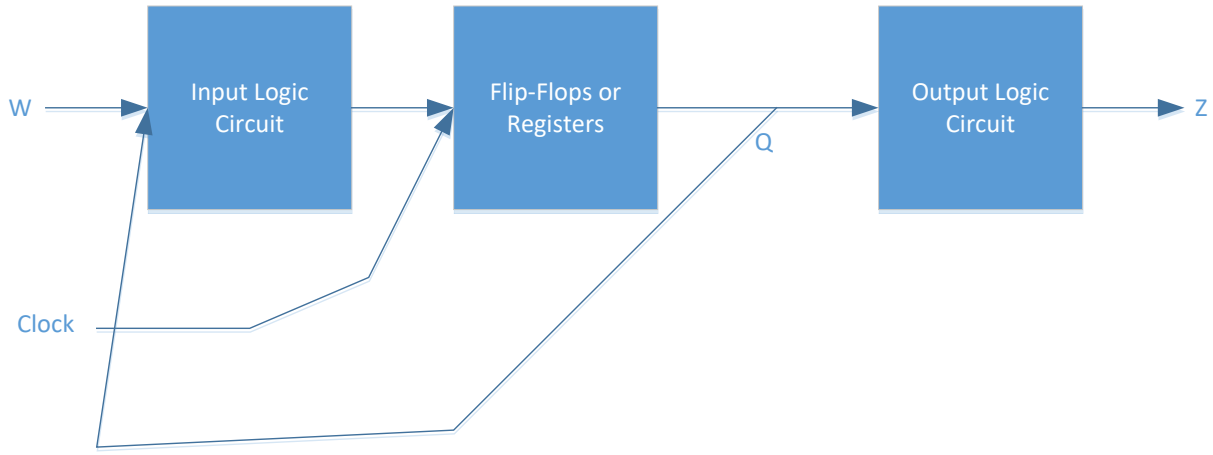
Scarlett Jones
CSCE 3730
Homework #4
Due: 11/08/2016, 11:55 PM

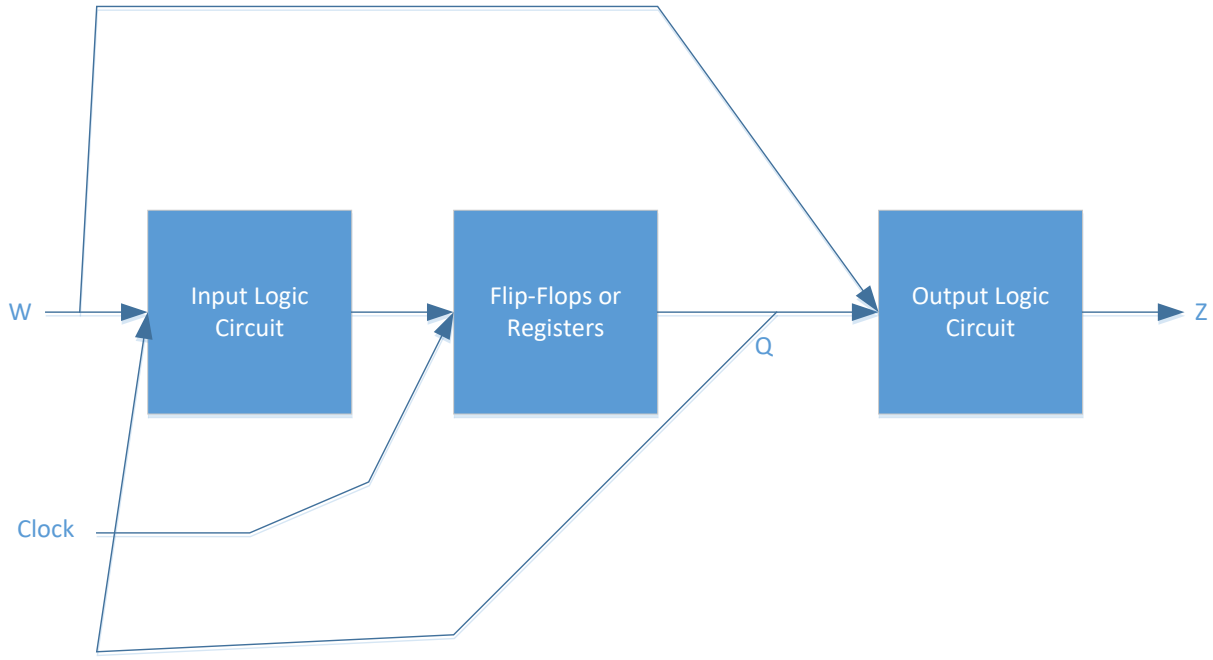
1. Explain the steps involved in a FPGA design process?
    a. Translation from HDL involves synthesis and translation.
    b. Logic Synthesis involves mapping. The mapping process maps the logic onto the resources on the board.
    c. Placement and routing involves placing and routing.
    d. Configuration generation involves program file generation.
2. What is the difference (at least two) between a combinational and sequential circuit?
    a. Combinational
        i. Output depends only on present inputs.
        ii. Doesn't depend on the clock for the design.
    b. Sequential
        i. Output depends not just on present inputs, but on past sequence of inputs.
        ii. There are synchronous and asynchronous designs.
3. What is the difference (at least two) between a latch and a flip-flop?
    a. Latch
        i. A feedback connection of 2 NOR or 2 NAND gates that can store 1 bit of information. Can be set to 1 with the S input. Can be reset to 0 with the R input.
        ii. The gated latch is a basic latch that includes a gating or a control input signal. This gating signal is typically a clock.
    b. Flip-flop
        i. Based on the gated latch.
        ii. Usually master-slave arrangement.
        iii. Can be level or edge triggered.
4. What are the applications (at least two) of a latch and flip-flop?
    a. Latch
        i. Avoids bouncing of switches.
        ii. Stores a single bit value.
    b. Flip-Flop
        i. Stores one bit at a time.
        ii. Can transfer data if you cascade several flip-flops to make a shift register.
5. What is the difference between level, positive edge, and negative edge triggered flip-flop?
    a. Level triggered flip-flop will activate when the clock input is 1.
    b. Positive edge triggered flip-flop will activate on the rising edge of the clock, just as the clock is changing from 0 to 1.
    c. Negative edge triggered flip-flop will activate on the falling edge of the clock, just as the clock is changing from 1 to 0.
6. What are the applications (at least two) of SR, D, JK, and T flip-flop?
    a. SR flip-flop
        i. When the clock input is 1, the first stage SR latch is enabled. The output from the first stage follows its inputs. The second stage is disables. Nothing changes on the output Q.

   ii. When the clock input is 0, the second stage SR latch is enabled. The first stage is disables. The inputs to stage 2 come from the output of stage 1. The inputs to stage 2 propagate to the output Q.
   iii. Offers a set and reset option. Set makes the output 1 and reset makes the output 0.
   iv. Uses 2 SR latches.
- b. D flip-flop
   i. The current state doesn't matter when determining the next state.
   ii. The next state follows whatever the D input is.
   iii. Prevents the illegal state of the SR by sending the D input to the upper NAND gate and the opposite of the D input to the lower NAND gate.
   iv. Uses 2 SR latches.
- c. JK flip-flop
   i. When the current state is 0, the next state will follow the J input.
   ii. When the current state is 1, the next state will be the opposite of the K input.
   iii. Uses 2 SR latches.
- d. T flip-flop
   i. If the input T is high and a clock pulse occurs, the output Q will toggle.
   ii. If the input T is low and a clock pulse occurs, the output Q will not change states.
   iii. The output doesn't change until a clock pulse occurs.

7. What is a digital clock signal? Define clock period, clock cycle, and clock frequency.
   a. A digital clock signal is a pulsing signal used to enable latches/flip-flops.
   b. Clock period is the time interval between pulses.
   c. Clock cycle is the amount of clock periods that go by.
   d. Clock frequency is 1/clock period.
8. What is a FSM (Finite State Machine)? What is the difference between Moore and Mealy FSMs (explain with block diagrams)?
   a. A finite state machine is made of synchronous sequential circuits. An FSM uses combinational logic and one or more flip-flops. The flip-flops are sometimes referred to as registers. The output of the flip-flops is referred to as the state of the circuit. Flip-flops change their state based on the input logic. The input logic circuit acquires inputs from the primary inputs and the register outputs. Changes in the state of the FSM depends on the present state and the primary inputs.
   b. Moore FSMs acquire inputs for the output logic circuit from register outputs only.



c. Mealy FSMs acquire inputs for the output logic circuit from register outputs and the primary inputs.



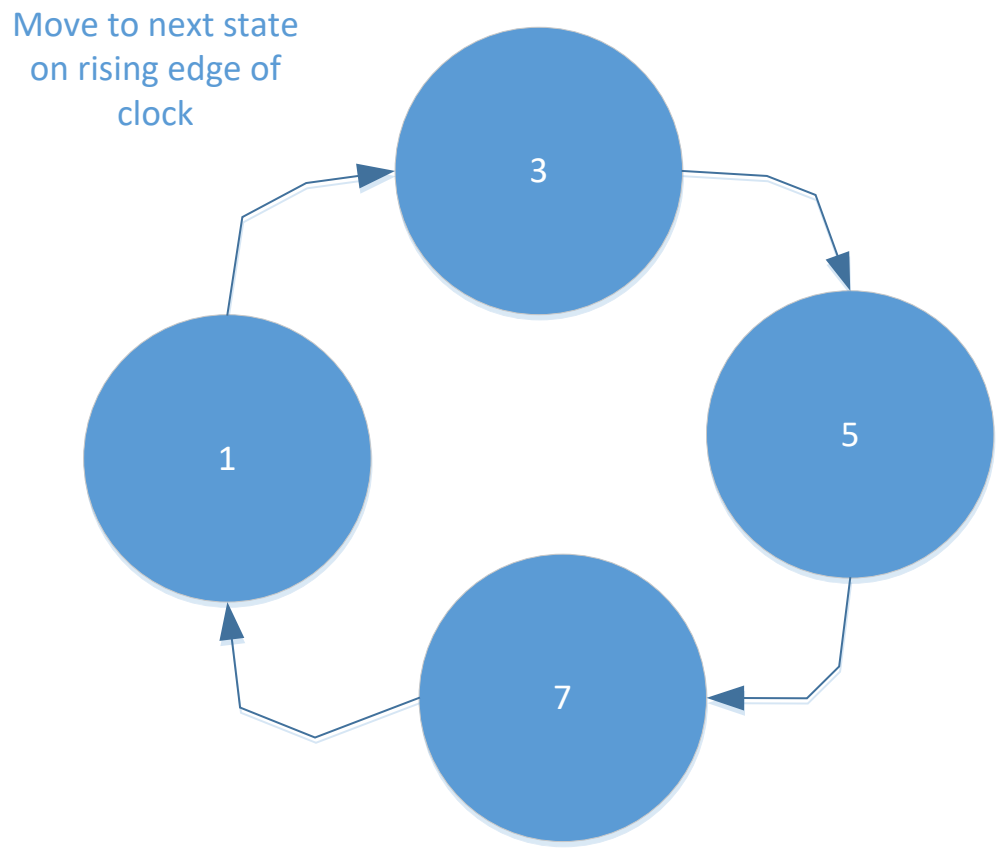
9. List the steps involved in a FSM design process?
    a. Obtain the specification of the desired circuit.
    b. Derive the states of the machine and develop a state diagram.
        i. Should show all possible states.
        ii. Provide the conditions for which the circuit moves from one state to the next.
    c. Develop the state table from the state diagram.
    d. Decide on the number of state variable needed to represent all states.
        i. Minimize the number of states if possible.
    e. Choose the type of flip-flops that will be used in the implementation.

i. Derive the next-state logic expressions to develop the Input Logic Circuit.
ii. Derive the logic expressions for the Output Logic Circuit.

f. Implement the design.

10. Design a FSM (only state diagram and state table) for a 3-bit counter that counts through odd numbers upwards. Write a behavioral VHDL code that implements the FSM. Write a VHDL test bench to test the FSM.

a. State diagram



b. State Table

State	Current State	Next State	D_A	D_B	D_C
	$Q_AQ_BQ_C$	$q_Aq_Bq_C$			
0	000	XXX	X	X	X
1	001	011	0	1	1
2	010	XXX	X	X	X
3	011	101	1	0	1
4	100	XXX	X	X	X
5	101	111	1	1	1
6	110	XXX	X	X	X
7	111	001	0	0	1

c. VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity CntrOdd is
        Port ( Clk, reset : in std_logic;
                Q : out std_logic_vector (2 DOWNTO 0) );
End CntrOdd;
Architecture Behavioral of CntrOdd is
        Signal Count: std_logic_vector (2 DOWNTO 0);
begin
        Process (Clk, reset)
        Begin
                IF Reset = '0' THEN
                        Count <= "001";
                ELSIF (Clk'EVENT AND Clk = '1') THEN
                        If Count = "001" then
                                Count <= "011";
                        Elsif Count = "011" then
                                Count <= "101";
                        Elsif Count = "101" then
                                Count <= "111";
                        Else Count <= "001";
                        End If;
                End If;
        End Process;
Q <= Count;
```

```
End Behavioral;
```

d. VHDL test bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY CntrOdd _tb IS
END CntrOdd _tb;
ARCHITECTURE behavioral OF CntrOdd _tb IS
        COMPONENT CntrOdd
        Port ( Clk, reset : in std_logic;
               Q : inout std_logic_vector (2 DOWNTO 0) );
        END COMPONENT;
        SIGNAL reset : STD_LOGIC;
        SIGNAL Q: STD_LOGIC_VECTOR(2 DOWNTO 0);
        constant Clk_period : time := 10 ns;
BEGIN
        UUT: CntrOdd PORT MAP(
                                Clk=>Clk,
                                reset => reset,
                                Q(0) => Q(0),
                                Q(1) => Q(1)
                                Q(2) => Q(2));
         -- Clock process definitions
        Clk_process :process
         begin
                        Clk <= '0';
```

```
        wait for Clk_period/2;
        Clk <= '1';
        wait for Clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
 -- hold reset state for 100 ns.
wait for 100 ns;

wait for Clk_period*10;

    -- insert stimulus here
Q(2)<='0'; Q(1)='0'; Q(0)='0';
wait for 10 ms;
Q(2)<='0'; Q(1)<='0'; Q(0)<='1';
wait for 10 ms;
Q(2)<='0'; Q(1)<='1'; Q(0)<='0';
wait for 10 ms;
Q(2)<='0'; Q(1)<='1'; Q(0)<='1';
wait for 10 ms;
Q(2)<='1'; Q(1)<='0'; Q(0)<='0';
wait for 10 ms;
Q(2)<='1'; Q(1)<='0'; Q(0)<='1';
wait for 10 ms;
Q(2)<='1'; Q(1)<='1'; Q(0)<='0';
```

```
        wait for 10 ms;
        Q(2)<='1'; Q(1)<='1'; Q(0)<='1';
        wait for 10 ms;
WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END behavioral;
```