Mod-10 Counter state table

* The first line tells us that only a positive clock edge will result in an increment. Any other condition results in the next state of Q (denoted Q+) being unchanged (equal to Q).
* The second line defines the behavior when the reset is active low (to agree with the active-low reset on the Nexys Video board).
* The third row tells us that the counter will hold when the 2-bit control is equal to 00.
* The fourth row tells us that the counter will count up when the 2-bit control is equal to 01. The "mod 10" means that the counter is supposed to count from 0 to 9 and then, on the next count, roll back to 0.
* The fifth row tells us that the counter will load in the value 'D' when the 2-bit control is equal to 10.
* The sixth row tells us that the counter will synchronously reset when the 2-bit control is equal to 11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| clk | reset | ctrl | D | Q+ |
| 0,1,falling | x | xx | x | Q |
| rising | 0 | xx | x | 0 |
| rising | 1 | 00 | x | Q |
| rising | 1 | 01 | x | Q+1 mod 10 |
| rising | 1 | 10 | D | D |
| rising | 1 | 11 | x | 0 |

Complete the Q trace in the following timing diagram based on the state table for the mod-10 counter.



Using only comparators, a 4-bit register, and an adder construct the logic for the mod-10 counter.



--------------------------------------------------------------------

-- Name: Chris Coulston

-- Date: Jan 13, 2015

-- File: lec04.vhdl

-- HW: Lecture 4

-- Crs: ECE 383

-- Purp: Demo the use of processes for a mod 10 counter

-- Documentation: I pulled some information from chapter 8.

-- Academic Integrity Statement: I certify that, while others may have

-- assisted me in brain storming, debugging and validating this program,

-- the program itself is my own work. I understand that submitting code

-- which is the work of other individuals is a violation of the honor

-- code. I also understand that if I knowingly give my original work to

-- another individual is also a violation of the honor code.

-------------------------------------------------------------------------

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.NUMERIC\_STD.ALL;

entity lec4 is

 Port( clk: in STD\_LOGIC;

 reset : in STD\_LOGIC;

 ctrl: in std\_logic\_vector(1 downto 0);

 D: in unsigned (3 downto 0);

 Q: out unsigned (3 downto 0));

end lec4;

architecture behavior of lec4 is

 signal rollSynch, rollCombo: STD\_LOGIC;

 signal processQ: unsigned (3 downto 0);

begin

 -------------------------------------

 -- ctrl behavior

 -- 00 hold

 -- 01 count up mod 10

 -- 10 load D

 -- 11 synch reset

 -------------------------------------

 process(clk)

 begin

 if (rising\_edge(clk)) then

 if (reset = '0') then

 processQ <= (others => '0');

 rollSynch <= '0';

 elsif ((processQ < 9) and (ctrl = "01")) then

 processQ <= processQ + 1;

 rollSynch <= '0';

 elsif ((processQ = 9) and (ctrl = "01")) then

 processQ <= (others => '0');

 rollSynch <= '1';

 elsif (ctrl = "10") then

 processQ <= unsigned(D);

 elsif (ctrl = "11") then

 processQ <= (others => '0');

 end if;

 end if;

 end process;

 rollCombo <= '1' when (processQ = 9) else '0';

 Q <= processQ;

end behavior;