The University of Alabama in Huntsville
Electrical and Computer Engineering
CPE/EE 422/522
Spring 2005
Homework #4 Solution

2.3 In the following VHDL process A, B, C, and D are all integers that have a value of 0 at time = 10 ns. If E changes from '0' to '1' at time = 20 ns, specify the time(s) at which each signal will change and the value to which it will change. List these changes in chronological order (20, 20 + Δ, 20 + 2Δ, etc.)

```
p1:process
  begin
    A <= 0 after 10 ns;
    B <= 0 after 10 ns;
    C <= 0 after 10 ns;
    D <= 0 after 10 ns;
    wait on E;
    A <= 1 after 5 ns;
    B <= A + 1;
    C <= B after 10 ns;
    wait for 0 ns;
    D <= B after 3 ns;
    A <= A + 5 after 15 ns;
    B <= B + 7;
    wait;
  end process p1;
```

ns	delta	A	B	C	D	E
0	+0	0	0	0	0	'0'
20	+0	0	0	0	0	'1'
20	+1	0	1	0	0	'1'
20	+2	0	8	0	0	'1'
23	+0	0	8	0	1	'1'
25	+0	1	8	0	1	'1'
28	+0	5	8	0	1	'1'

2.8 (modified) An inhibited toggle flip-flop has inputs I0, T, and Reset, and outputs Q and QN. Reset is active high and overrides the action of the other inputs. The flip-flop works as follows. If I0 = '1', the flip-flop changes state on the rising edge of T, if I0 = '0', no state change occurs (except on reset). Assume the propagation delay from T to output is 8 ns and from reset output is 5 ns.

```
entity INHIBITED_T_FF is
  port (I0, T, RESET, CLK : in bit;
        Q, QN : out bit);
end INHIBITED_T_FF;

architecture BEHAVE of INHIBITED_T_FF is
  signal QTEMP : bit;
begin
  process(RESET, CLK)
  begin
    if (RESET = '1') then
```

```
      QTEMP <= '0' after 5 ns;
    elsif (CLK'event and CLK = '1') then
      if (I0 = '1') then
        QTEMP <= not QTEMP after 8 ns;
      end if;
    end if;
  end process;
  Q <= QTEMP;
  QN <= not QTEMP;
end BEHAVE;
```

2.9 (using 1.8 instead of 1.11) (a) only, using the test sequence from 1.8. Write a behavioral VHDL description of the state machine that you designed in Problem 1.8. Assume that state changes occur on the falling edge of the clock pulse. Use a case statement together with if-then-else statements to represent the state table. Compile and simulate your code using the test sequence from 1.8.

```
entity SEQUENCE_DETECTOR is
  port(X, CLK : in bit;
       Z : out bit);
end SEQUENCE_DETECTOR;

architecture IF_CASE of SEQUENCE_DETECTOR is
  type STATE_TYPE is (S0, S1, S2, S3, S4, S5);
  signal STATE, NEXT_STATE : STATE_TYPE;
begin
  process(STATE, X)
  begin
    case STATE is
      when S0 => if(X = '0') then
                   NEXT_STATE <= S1;
                   Z <= '0';
                 else
                   NEXT_STATE <= S0;
                   Z <= '0';
                 end if;
      when S1 => if(X = '0') then
                   NEXT_STATE <= S1;
                   Z <= '0';
                 else
                   NEXT_STATE <= S2;
                   Z <= '0';
                 end if;
      when S2 => if(X = '0') then
                   NEXT_STATE <= S3;
                   Z <= '0';
                 else
                   NEXT_STATE <= S4;
                   Z <= '0';
                 end if;
      when S3 => if(X = '0') then
                   NEXT_STATE <= S1;
                   Z <= '0';
                 else
                   NEXT_STATE <= S5;
                   Z <= '1';
```

```
                end if;
      when S4 => if(X = '0') then
                   NEXT_STATE <= S5;
                   Z <= '1';
                 else
                   NEXT_STATE <= S0;
                   Z <= '0';
                 end if;
      when S5 => if(X = '0') then
                   NEXT_STATE <= S0;
                   Z <= '1';
                 else
                   NEXT_STATE <= S0;
                   Z <= '1';
                 end if;
    end case;
  end process;
  process (CLK)
  begin
    if (CLK'event and CLK = '0') then
      STATE <= NEXT_STATE;
    end if;
  end process;
end IF_CASE;

entity TEST_SEQ_DET is
end TEST_SEQ_DET;

architecture BEHAVE of TEST_SEQ_DET is
  component SEQUENCE_DETECTOR
    port (X, CLK : in bit;
          Z : out bit);
  end component;
  signal CLOCK, SEQ_IN, SEQ_OUT : bit := '0';
begin
  U1 : SEQUENCE_DETECTOR port map(SEQ_IN, CLOCK, SEQ_OUT);
  CLOCK <= not CLOCK after 50 ns;
  process
  begin
    SEQ_IN <= '0' after 25 ns, '1' after 125 ns, '0' after 225 ns,
              '1' after 425 ns, '0' after 725 ns, '1' after 825 ns,
              '0' after 925 ns, '1' after 1025 ns, '0' after 1125 ns,
              '1' after 1225 ns, '0' after 1325 ns, '1' after 1425 ns,
              '0' after 1625 ns, '1' after 1725 ns;
    wait;
  end process;
end BEHAVE;
```