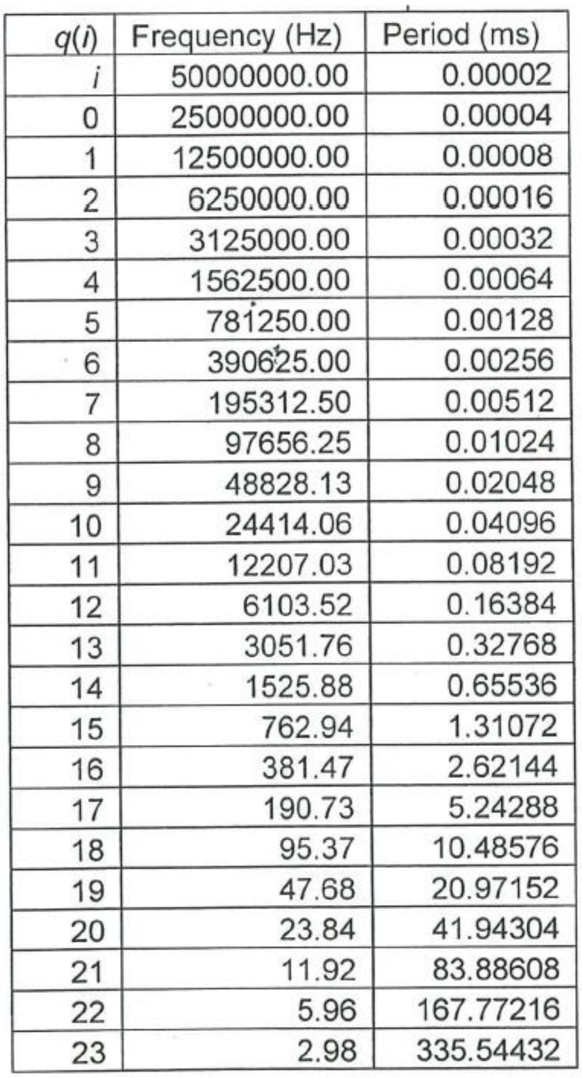
ECE 424 – INTRODUCTION TO VLSI DESIGN + LAB

LABORATORY WORK 8

In this laboratory work, we will study on clock division method, Mealy type and Moore type machines.

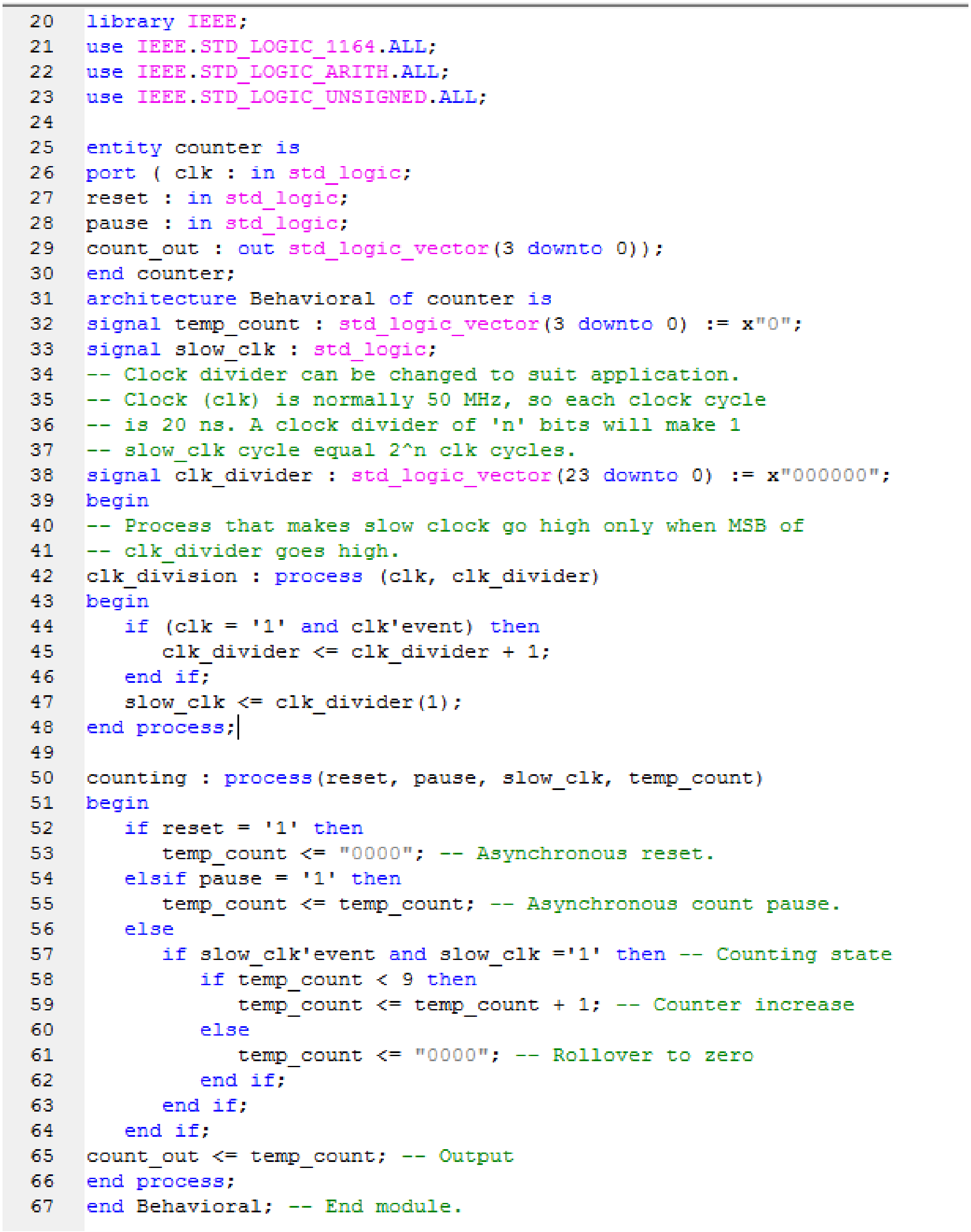
Clock Division:

When you want to show a counter on your FPGA development card, you cannot see the counting because of its velocity. It will count so fast, because the oscillator frequency of a FPGA development kit is in MHz's. So you should divide clock by using clock division method.



$q(i)$	Frequency (Hz)	Period (ms)
i	50000000.00	0.00002
0	25000000.00	0.00004
1	12500000.00	0.00008
2	6250000.00	0.00016
3	3125000.00	0.00032
4	1562500.00	0.00064
5	781250.00	0.00128
6	390625.00	0.00256
7	195312.50	0.00512
8	97656.25	0.01024
9	48828.13	0.02048
10	24414.06	0.04096
11	12207.03	0.08192
12	6103.52	0.16384
13	3051.76	0.32768
14	1525.88	0.65536
15	762.94	1.31072
16	381.47	2.62144
17	190.73	5.24288
18	95.37	10.48576
19	47.68	20.97152
20	23.84	41.94304
21	11.92	83.88608
22	5.96	167.77216
23	2.98	335.54432

Figure 1

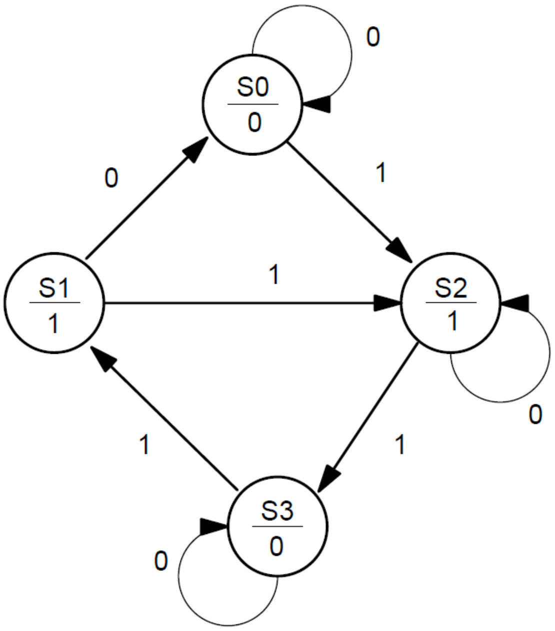


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
port ( clk : in std_logic;
reset : in std_logic;
pause : in std_logic;
count_out : out std_logic_vector(3 downto 0));
end counter;
architecture Behavioral of counter is
signal temp_count : std_logic_vector(3 downto 0) := x"0";
signal slow_clk : std_logic;
-- Clock divider can be changed to suit application.
-- Clock (clk) is normally 50 MHz, so each clock cycle
-- is 20 ns. A clock divider of 'n' bits will make 1
-- slow_clk cycle equal 2^n clk cycles.
signal clk_divider : std_logic_vector(23 downto 0) := x"000000";
begin
-- Process that makes slow clock go high only when MSB of
-- clk_divider goes high.
clk_division : process (clk, clk_divider)
begin
   if (clk = '1' and clk'event) then
      clk_divider <= clk_divider + 1;
   end if;
   slow_clk <= clk_divider(1);
end process;

counting : process(reset, pause, slow_clk, temp_count)
begin
   if reset = '1' then
      temp_count <= "0000"; -- Asynchronous reset.
   elsif pause = '1' then
      temp_count <= temp_count; -- Asynchronous count pause.
   else
      if slow_clk'event and slow_clk ='1' then -- Counting state
         if temp_count < 9 then
            temp_count <= temp_count + 1; -- Counter increase
         else
            temp_count <= "0000"; -- Rollover to zero
         end if;
      end if;
   end if;
count_out <= temp_count; -- Output
end process;
end Behavioral; -- End module.
```

Figure 2

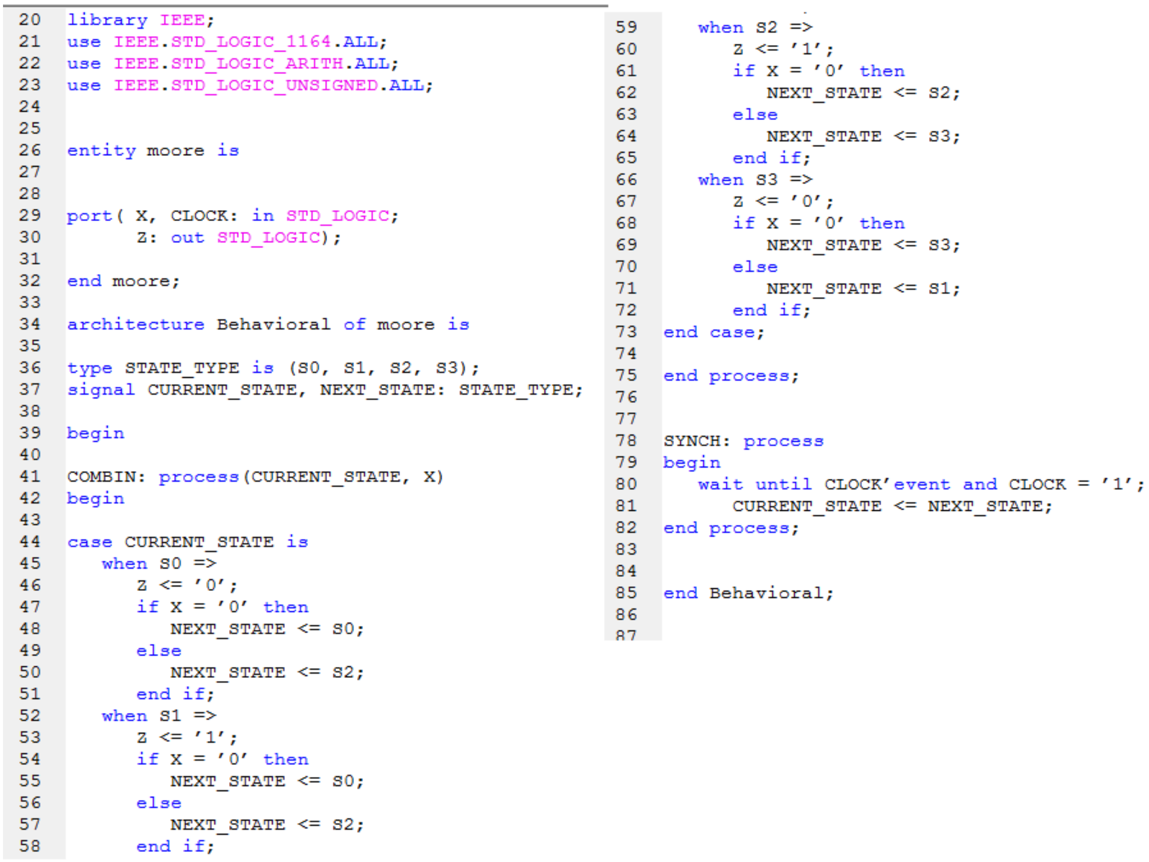
Figure 1 is explaining the main idea of clock division and figure 2 is giving the vhdl codes for doing this for modulo9 counter. Write the codes and simulate it by using test bench waveform.

Moore Machine:



Present state	Next state		Output (Z)
	X=0	X=1	X=0
S0	S0	S2	0
S1	S0	S2	1
S2	S2	S3	1
S3	S3	S1	0

Figure 3

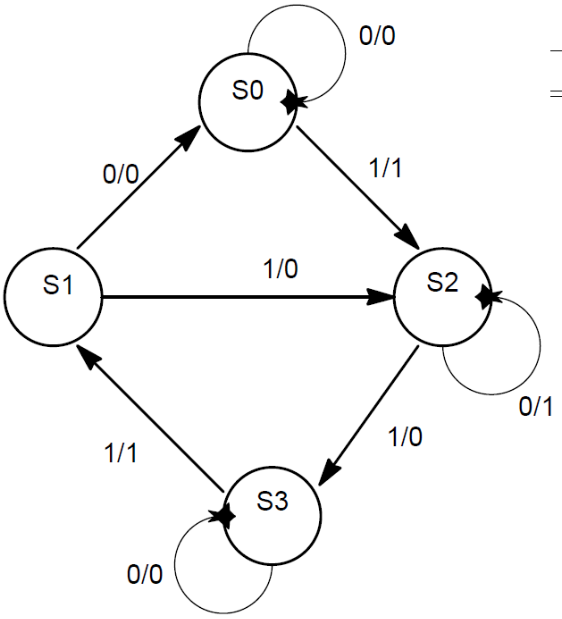


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity moore is


port( X, CLOCK: in STD_LOGIC;
      Z: out STD_LOGIC);

end moore;

architecture Behavioral of moore is

type STATE_TYPE is (S0, S1, S2, S3);
signal CURRENT_STATE, NEXT_STATE: STATE_TYPE;

begin

COMBIN: process(CURRENT_STATE, X)
begin

case CURRENT_STATE is
   when S0 =>
      Z <= ’0’;
      if X = ’0’ then
         NEXT_STATE <= S0;
      else
         NEXT_STATE <= S2;
      end if;
   when S1 =>
      Z <= ’1’;
      if X = ’0’ then
         NEXT_STATE <= S0;
      else
         NEXT_STATE <= S2;
      end if;
   when S2 =>
      Z <= ’1’;
      if X = ’0’ then
         NEXT_STATE <= S2;
      else
         NEXT_STATE <= S3;
      end if;
   when S3 =>
      Z <= ’0’;
      if X = ’0’ then
         NEXT_STATE <= S3;
      else
         NEXT_STATE <= S1;
      end if;
end case;

end process;


SYNCH: process
begin
   wait until CLOCK’event and CLOCK = ’1’;
      CURRENT_STATE <= NEXT_STATE;
end process;


end Behavioral;
```

Figure 4

Figure 3 is a simple moore machine Figure 4 is giving vhdl codes of that machine. Write these codes and simulate it.

Mealy Machine:



Present state	Next state		Output (Z)	
	X=0	X=1	X=0	X=1
S0	S0	S2	0	1
S1	S0	S2	0	0
S2	S2	S3	1	0
S3	S3	S1	0	1

Figure 5

Figure 5 is a FSM(Finite State Machine) that is explaining a simple mealy type machine. Write your own vhdl code for this machine and simulate it to see that your code is working properly.

Homework: By using Mealy / Moore machine, design a vending machine, draw the finite state machine diagram and truth table.

Your vending machine is accepting 25Kr, 50Kr and 1 TL. There are three types of drinks in your machine. These drinks are water, ice tea and cola.

The price of cola: 1.50TL

The price of ice tea: 1TL

The price of water: 75 Kr

The customer should choose the drink after giving money. There are 4 buttons on the machine. 3 of them are for cola, ice tea and water. 1 of them is for giving back the money. Don’t forget to give back the reminder of money if there is.

Send me FSM diagram and simulated VHDL codes with whole project file.

Deadline: 07.01.2014 Tuesday, 21:00.

Prepared By: **İbrahim BOZKURT**
e-mail: ibrahimbozkurt@cankaya.edu.tr