

به نام یکتای در کمال



دانشکده مهندسی کامپیوتر

گزارش کار ششم – آزمایشگاه معماری کامپیوتر

ماشین‌های Mealy و Moore

استاد : سرکار خانم دکتر محبتی

عرفان زارع - امیرحسین ملکوتی - محمدرضا صاحبزاده

دی ۱۴۰۱

شرح آزمایش :

در این آزمایش قصد داشتیم علاوه بر آشنایی با ماشین های Mealy و Moore آن ها را به وسیله ی زبان VHDL برای سه ماشین طراحی کنیم.

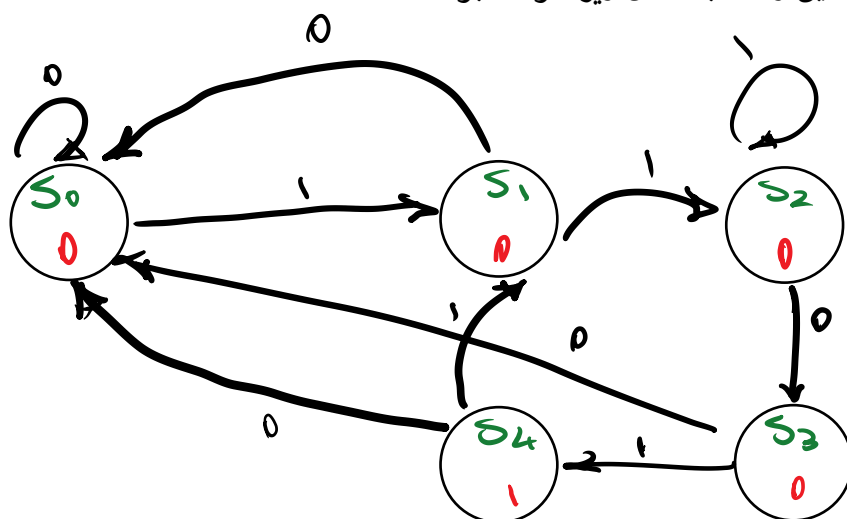
از نظر نوع ماشین ها ما در ماشین های Moore طراحی ساده تر و سنکرون با کل مدار را خواهیم داشت. همچنین در این ماشین ها خروجی توسط حالت کنونی به وجود می آید.

مزیت طراحی ماشین های Mealy در تعداد کمتر State های آن ها است. در این نوع ماشین خروجی ما به مقدار کنونی و حالت کنونی وابسته خواهد بود که همین ویژگی باعث تفاوت این دو ماشین شده است.

در این تمرین از یکی از تعاریف و مدل هایی که خواهیم داشت مدل هافمن است که قسمت های ترکیبی مدار را از قسمت های ترتیبی آن جدا می کند. قسمت های ترتیبی آن معمولا با Process ی که نسبت به سیگنال کلاک و ریست حساس است و در قسمت های ترکیبی ما بخش های حساس به حالت های مدار را خواهیم داشت.

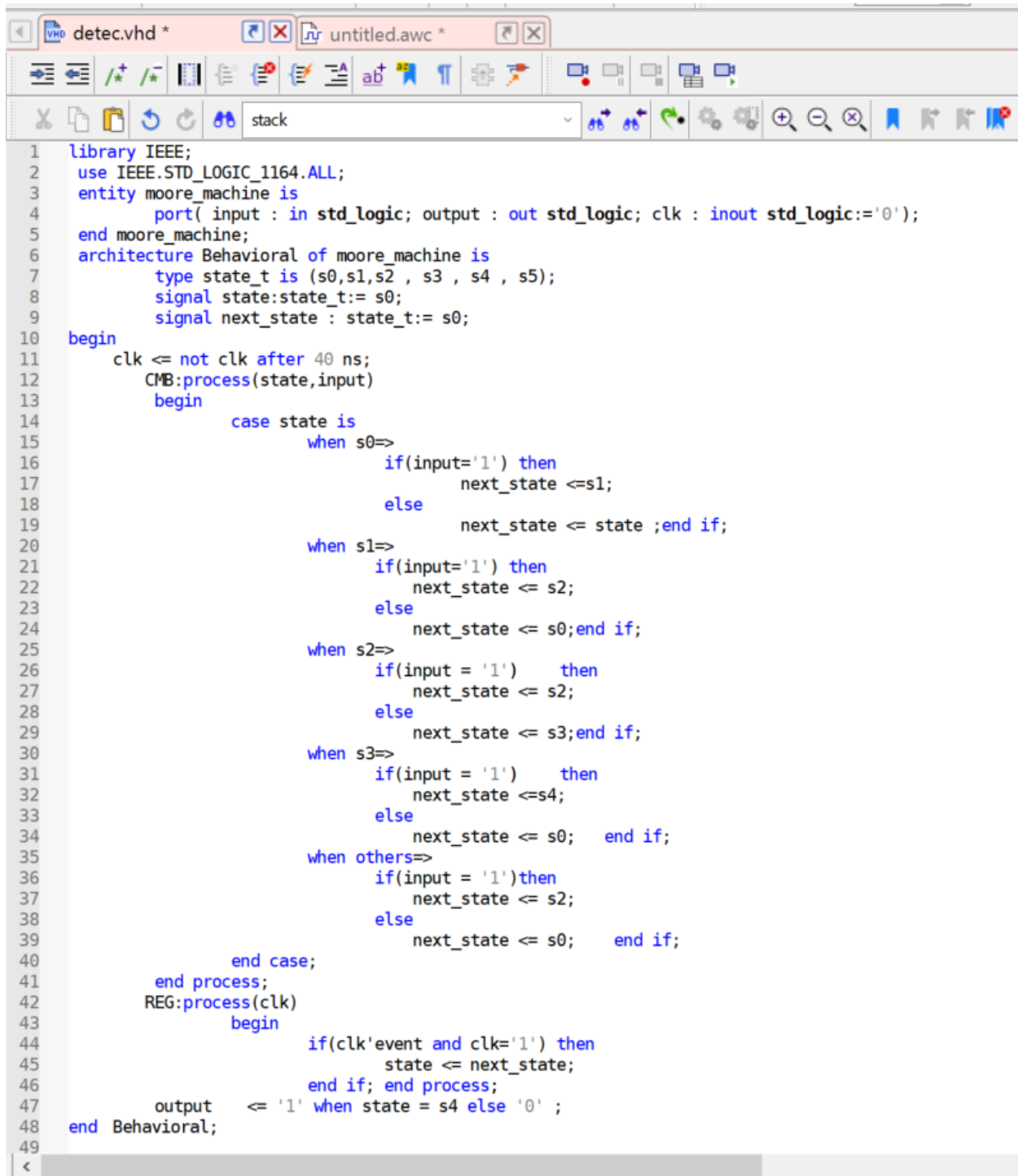
به عنوان تمرین کلاسی ما طرحی یک دیاگرام حالت را برای یک Detector Sequence خواهیم داشت که رشته ی 1011 را پردازش خواهد کرد.

شماتیک ماشین Moore برای این رشته به شکل زیر خواهد بود.



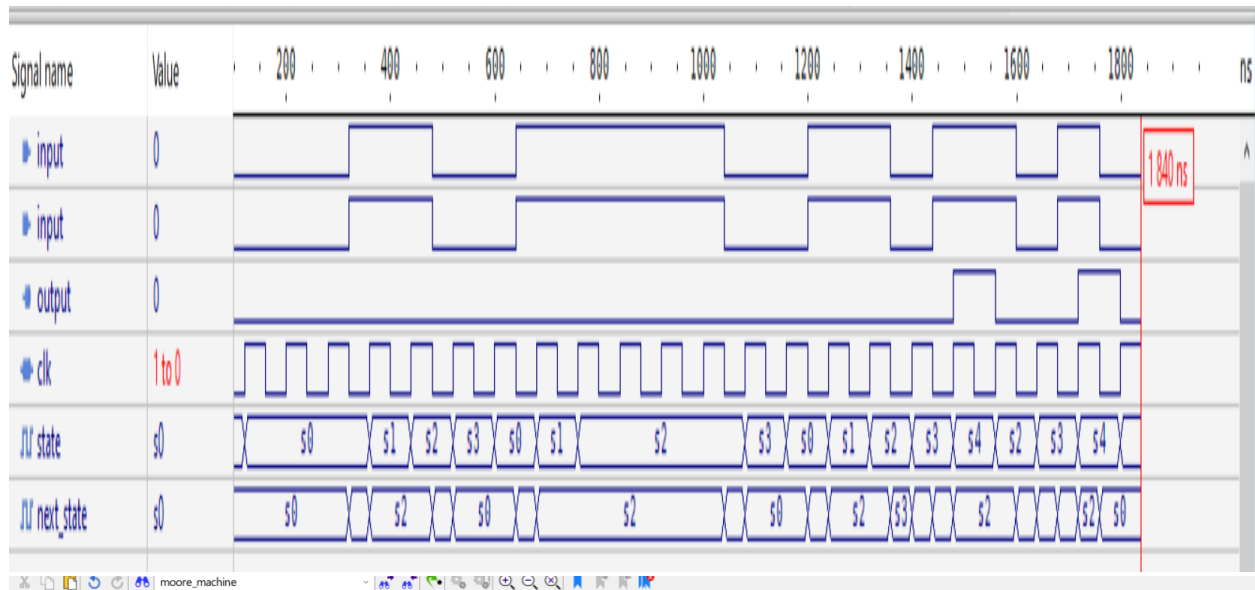
حال برای ماشین بالا کد و سیگنال های زیر را خواهیم داشت

\\



The image shows a screenshot of a VHDL code editor. The top window is titled 'detec.vhd \*' and the bottom window is titled 'untitled.awc \*'. The editor has a toolbar with various icons for editing and simulation. The code is written in VHDL and defines a Moore machine named 'moore\_machine'. The code includes a library declaration for IEEE, a use clause for IEEE.STD\_LOGIC\_1164.ALL, and an entity declaration for 'moore\_machine' with ports 'input', 'output', and 'clk'. The architecture is 'Behavioral' and defines a state type 'state\_t' with states 's0', 's1', 's2', 's3', 's4', and 's5'. It also defines a signal 'state' of type 'state\_t' and a signal 'next\_state' of type 'state\_t'. The code includes a 'begin' block with a clock signal 'clk' and a process 'CMB' that handles the state transitions. The process 'CMB' is a 'process' of 'state' and 'input'. It contains a 'case' statement for 'state' with branches for 's0', 's1', 's2', 's3', and 'others'. Each branch contains an 'if' statement for 'input' and a 'next\_state' assignment. The 'others' branch also contains an 'if' statement for 'input' and a 'next\_state' assignment. The process 'CMB' ends with 'end case;', 'end process;', and 'REG:process(clk)'. The 'REG' process is a 'process' of 'clk' and contains an 'if' statement for 'clk' event and 'clk' value. It assigns 'state' to 'next\_state' and ends with 'end if; end process;'. The code also includes an 'output' assignment and an 'end Behavioral;' statement.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 entity moore_machine is
4     port( input : in std_logic; output : out std_logic; clk : inout std_logic:= '0');
5 end moore_machine;
6 architecture Behavioral of moore_machine is
7     type state_t is (s0,s1,s2 , s3 , s4 , s5);
8     signal state:state_t:= s0;
9     signal next_state : state_t:= s0;
10 begin
11     clk <= not clk after 40 ns;
12     CMB:process(state,input)
13     begin
14         case state is
15             when s0=>
16                 if(input='1') then
17                     next_state <=s1;
18                 else
19                     next_state <= state ;end if;
20             when s1=>
21                 if(input='1') then
22                     next_state <= s2;
23                 else
24                     next_state <= s0;end if;
25             when s2=>
26                 if(input = '1') then
27                     next_state <= s2;
28                 else
29                     next_state <= s3;end if;
30             when s3=>
31                 if(input = '1') then
32                     next_state <=s4;
33                 else
34                     next_state <= s0; end if;
35             when others=>
36                 if(input = '1')then
37                     next_state <= s2;
38                 else
39                     next_state <= s0; end if;
40         end case;
41     end process;
42     REG:process(clk)
43     begin
44         if(clk'event and clk='1') then
45             state <= next_state;
46         end if; end process;
47     output <= '1' when state = s4 else '0' ;
48 end Behavioral;
```

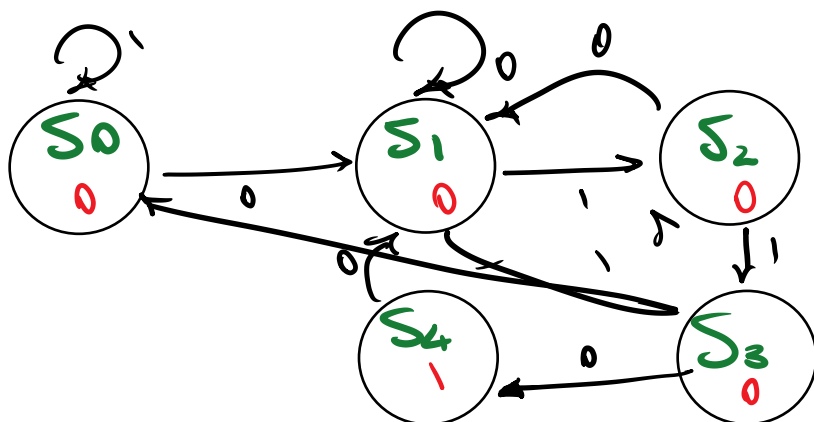


```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  ENTITY moore_tb IS
4  END moore_tb;
5
6  ARCHITECTURE behavior OF moore_tb IS
7      signal input, clk : std_logic := '0';
8      signal output_moore : std_logic := '0';
9      constant clk_period : time := 80 ns;
10 BEGIN
11
12     UUT1 : ENTITY WORK.moore_1011 (Behavioral) PORT MAP (input,output_moore,clk);
13     clk_process : process
14     begin
15         clk <= '0';
16         wait for clk_period/2;
17         clk <= '1';
18         wait for clk_period/2;
19     end process;
20
21 END;
22

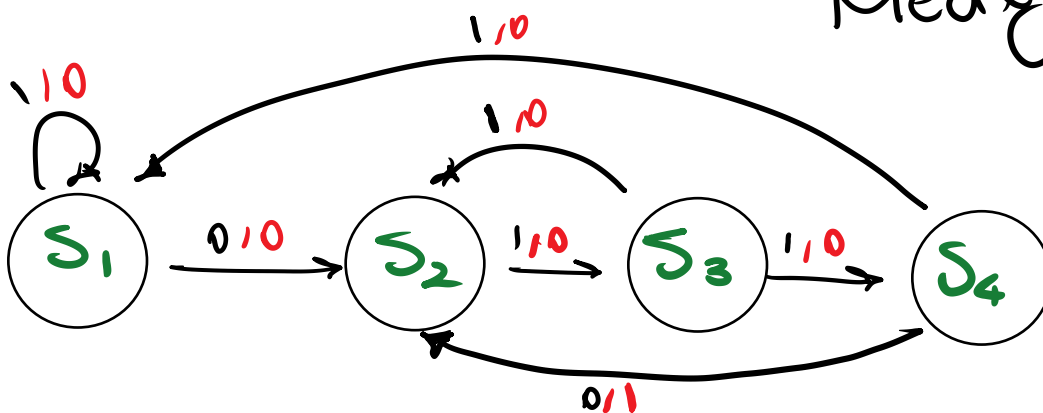
```

حال در مرحله ی بعد یک مدار آشکار ساز با استفاده از ماشین های میلی و مور را برای دنباله ی ۱۱۰ خواهیم داشت.



Moore Machine

Mealy Machine



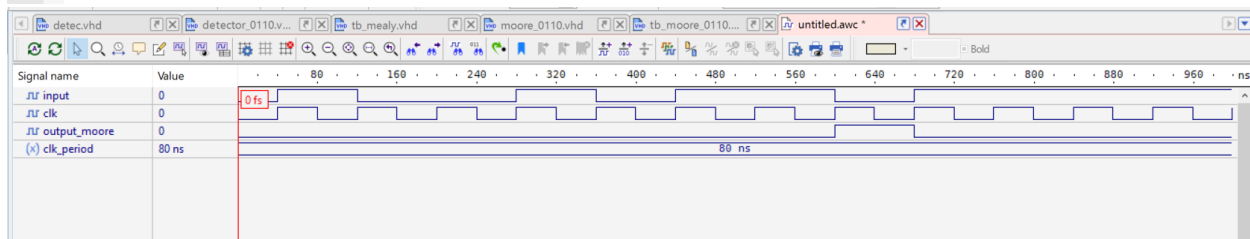
حال برای ماشین های بالا کدهای زیر را خواهیم داشت.

Moore Machine

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  ENTITY moore_tb IS
4  END moore_tb;
5
6  ARCHITECTURE behavior OF moore_tb IS
7      signal input, clk : std_logic := '0';
8      signal output_moore : std_logic := '0';
9      constant clk_period : time := 80 ns;
10 BEGIN
11
12     UUT1 : ENTITY WORK.moore_machine(Behavioral) PORT MAP (input,output_moore,clk);
13     clk_process : process
14     begin
15         clk <= '0';
16         wait for clk_period/2;
17         clk <= '1';
18         wait for clk_period/2;
19     end process;
20
21     input <= '1' after 40 ns, '0' after 120 ns, '0' after 200 ns,
22           '1' after 280 ns, '0' after 360 ns, '1' after 440 ns,
23           '1' after 520 ns, '0' after 600 ns, '1' after 680 ns;
24     --reset <= '1' after 140 ns;
25 END;
26

```



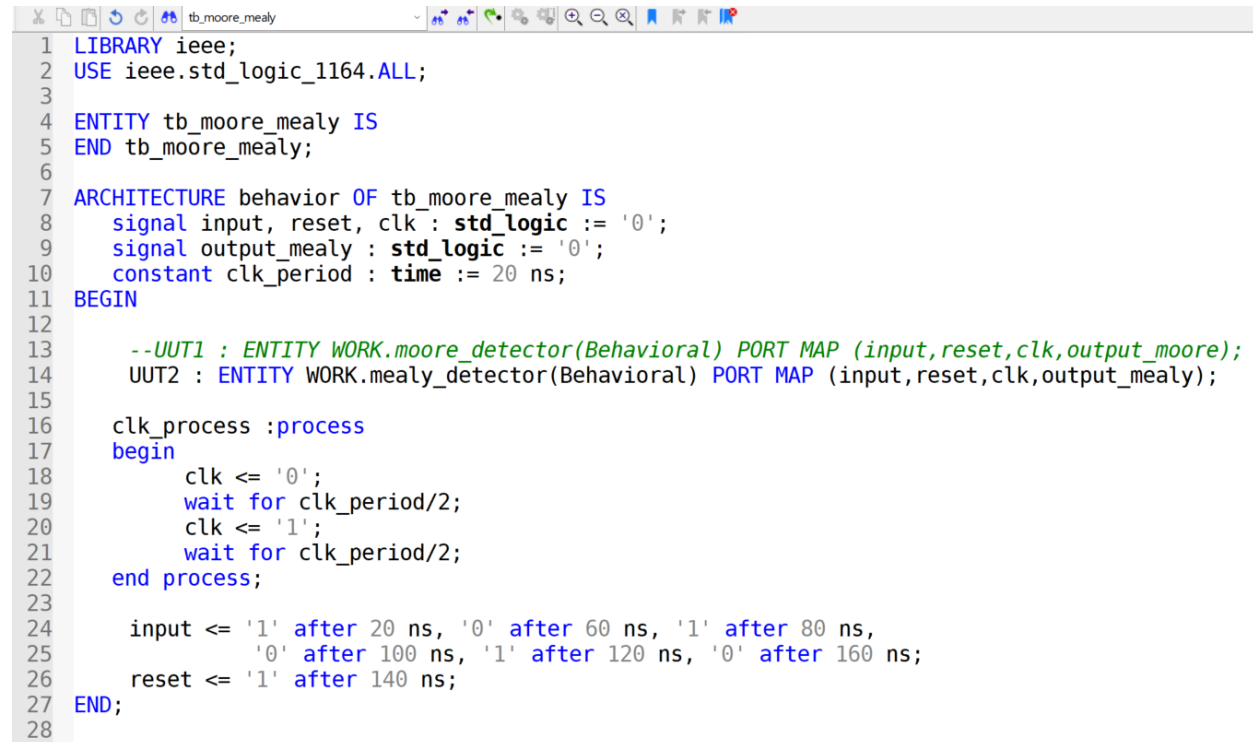
```

3  entity moore_machine is
4      port( input : in std_logic; output : out std_logic; clk : in std_logic:= '0' );
5  end moore_machine;
6  architecture Behavioral of moore_machine is
7      type state_t is (s0,s1,s2 , s3 , s4);
8      signal state:state_t:= s0;
9      signal next_state : state_t:= s0;
10 begin
11     CMB:process(state,input)
12     begin
13         case state is
14             when s0=>
15                 if(input='1') then
16                     next_state <=s0;
17                 else
18                     next_state <= s1 ;end if;
19             when s1=>
20                 if(input='1') then
21                     next_state <= s2;
22                 else
23                     next_state <= s1;end if;
24             when s2=>
25                 if(input = '1') then
26                     next_state <= s3;
27                 else
28                     next_state <= s1;end if;
29             when s3=>
30                 if(input = '1') then
31                     next_state <=s0;
32                 else
33                     next_state <= s4; end if;
34             when others=>
35                 if(input = '1')then
36                     next_state <= s2;
37                 else
38                     next_state <= s1; end if;
39         end case;
40     end process;
41     REG:process(clk)
42     begin
43         if(clk'event and clk='1') then
44             state <= next_state;
45         end if; end process;
46     output <= '1' when state = s4 else '0' ;
47 end Behavioral;

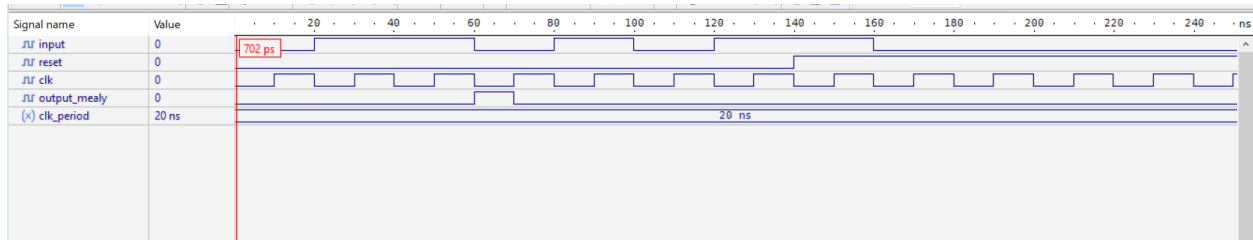
```



## Mealy Machine



```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY tb_moore_mealy IS
5  END tb_moore_mealy;
6
7  ARCHITECTURE behavior OF tb_moore_mealy IS
8      signal input, reset, clk : std_logic := '0';
9      signal output_mealy : std_logic := '0';
10     constant clk_period : time := 20 ns;
11 BEGIN
12
13     --UUT1 : ENTITY WORK.moore_detector(Behavioral) PORT MAP (input,reset,clk,output_moore);
14     UUT2 : ENTITY WORK.mealy_detector(Behavioral) PORT MAP (input,reset,clk,output_mealy);
15
16     clk_process :process
17     begin
18         clk <= '0';
19         wait for clk_period/2;
20         clk <= '1';
21         wait for clk_period/2;
22     end process;
23
24     input <= '1' after 20 ns, '0' after 60 ns, '1' after 80 ns,
25            '0' after 100 ns, '1' after 120 ns, '0' after 160 ns;
26     reset <= '1' after 140 ns;
27 END;
```



7

```

tb_moore_mealy
3  entity mealy_detector is
4  port (input, reset, clk : in std_logic;
5        output : out std_logic);
6  end entity mealy_detector;
7  architecture Behavioral of mealy_detector is
8      type state is (rst, s0, s1, s2);
9      signal present_state : state := rst;
10 begin
11 process (clk, reset)    --reset can be asynchronous with clk
12 begin
13     if reset = '1' then
14         present_state <= rst;
15     elsif (clk = '1' and clk'event) then
16         case present_state is
17             when rst =>
18                 if input = '1' then
19                     present_state <= rst;
20                 else
21                     present_state <= s0;
22                 end if;
23             when s0 =>
24                 if input = '1' then
25                     present_state <= s1;
26                 else
27                     present_state <= s0;
28                 end if;
29             when s1 =>
30                 if input = '1' then
31                     present_state <= s2;
32                 else
33                     present_state <= s0; --present_state <= rst;
34                 end if;
35             when s2 =>
36                 if input = '1' then
37                     present_state <= rst;
38                 else
39                     present_state <= s0;
40                 end if;
41             when others =>
42                 present_state <= rst;
43         end case;
44     end if;
45 end process;
46 output <= '1' when (present_state = s2 and input = '0') else '0';
47 end Behavioral;

```