

آزمایشگاه مدار منطقی



Amirkabir University of Technology
(Tehran Polytechnic)



نویسنده:

علیرضا طباطبائی (9723052)

استاد آزمایشگاه:

دکتر علیرضا

ظاهری نوید



THE PRESENT IS THEIRS, THE FUTURE, FOR



WHICH I REALLY WORK, IS MINE.

سوال 1 :

برای کد نویسی این برنامه، ابتدا یک process ساخته و درون آن یک if حساس به لبه قرار میدهیم.

سپس تابع reset با بیشترین اولویت را طراحی کرده و پس از آن، تابع pause با اولویت دوم را طراحی میکنیم (درون pause نیاز نیست چیزی قرار دهیم چون فقط نیاز است مدار از ادامه دادن بایستد).

در اولویت سوم، اصل شمارنده را قرار میدهیم بدین صورت که variable ای مثل count هر 10ns مقدار آن از 0 یکی یکی افزایش می یابد تا به 100 که رسید، متغیر number بصورت باینری یکی زیاد میشود و سپس همین روند تکرار میشود.

دوره این تکرار نیز 1us میکروثانیه می باشد.

برای تعیین اولویت اعمال بالا نظیر reset و pause و counting از تابع if/elsif استفاده میکنیم.

View: Implementation Simulation

Hierarchy

- choice - Behavioral (choice.vhd)
- comparator_if - Behavioral (compar...
- comparator - Behavioral (comparatc...
- dec2to4bymulti4to1 - Behavioral (de...
- dec4to16 - Behavioral (dec4to16.vhd)
- dec4to2 - Behavioral (dec4to2.vhd)
- multi16to1by2to1 - Behavioral (mult...
- multi4to1by2to1byselectwhen - Beh...
- priorityencoder4to2 - Behavioral (p...
- seg7case - Behavioral (seg7case.vhd)
- seg7 - Behavioral (seg7.vhd)
- test1 - Behavioral (test1.vhd)

No Processes Running

Processes: test1 - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
 - View RTL Schematic
 - View Technology Schematic
- Check Syntax
- Generate Post-Synthesis Simulati...
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

```

34 entity test1 is
35     Port ( clk : in  STD_LOGIC;
36           reset : in  STD_LOGIC;
37           pause : in  STD_LOGIC;
38           out1 : out STD_LOGIC_vector(3 downto 0));
39 end test1;
40 architecture Behavioral of test1 is
41     signal number : STD_LOGIC_VECTOR(3 downto 0) := "0000";
42 begin
43
44     process (clk,number,reset,pause)
45         variable count : integer := 0;
46     begin
47         if rising_edge(clk) then
48             if reset = '1' then
49                 count := 0;
50                 number <= "0000";
51             elsif pause = '1' then
52                 --
53             else
54                 count := count + 1;
55                 if number = "1001" then
56                     if count = 100 then
57                         number <= "0000";
58                         count := 0;
59                     end if;
60                 elsif count = 100 then
61                     number <= number + '1';
62                     count := 0;
63                 end if;
64             end if;
65             out1 <= number;
66         end if;
67     end process;

```

dec4 - Behavior (dec4.vhd)

decby_test - behavior (decby_test.vh

No Processes Running

Processes: counter4_test - behavior

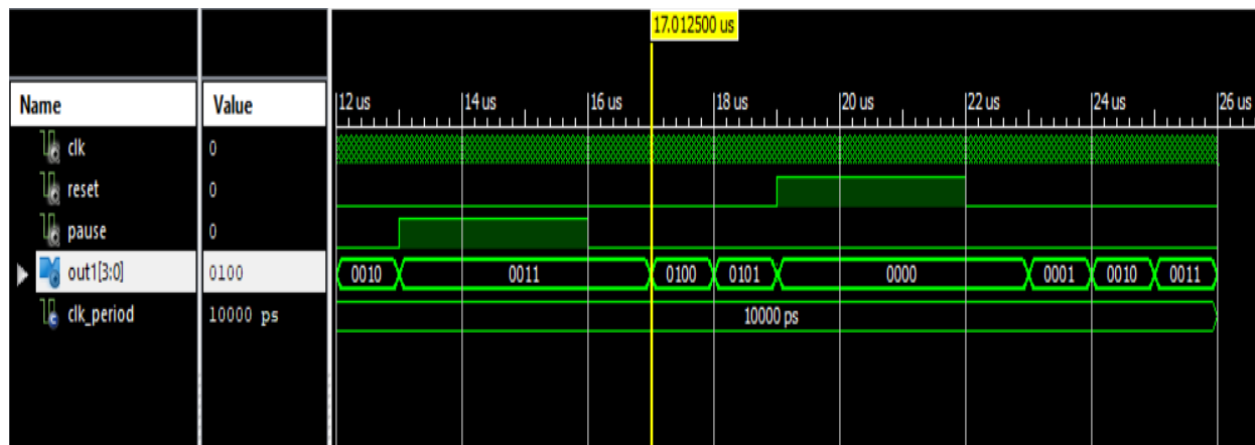
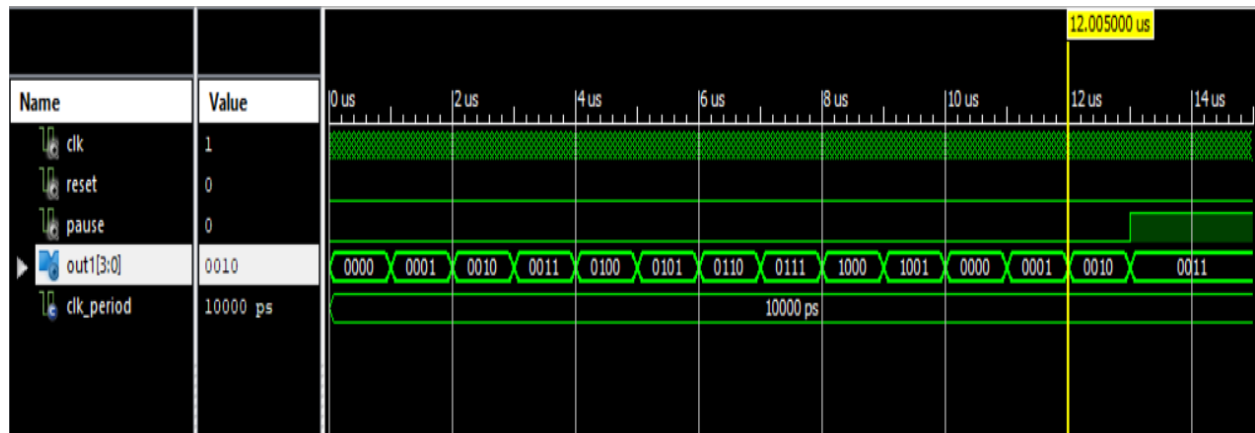
- ISim Simulator
- Behavioral Check Syntax
- Simulate Behavioral Model

```

81
82
83 -- Stimulus process
84 stim_proc: process
85 begin
86     -- hold reset state for 100 ns.
87     wait for 13 us;
88     pause <= '1';
89     wait for 3 us;
90     pause <= '0';
91     wait for 3 us;
92     reset <= '1';
93     wait for 3 us;
94     reset <= '0';
95     wait for 3 us;
96
97     wait for clk_period*10;
98
99     -- insert stimulus here
100

```

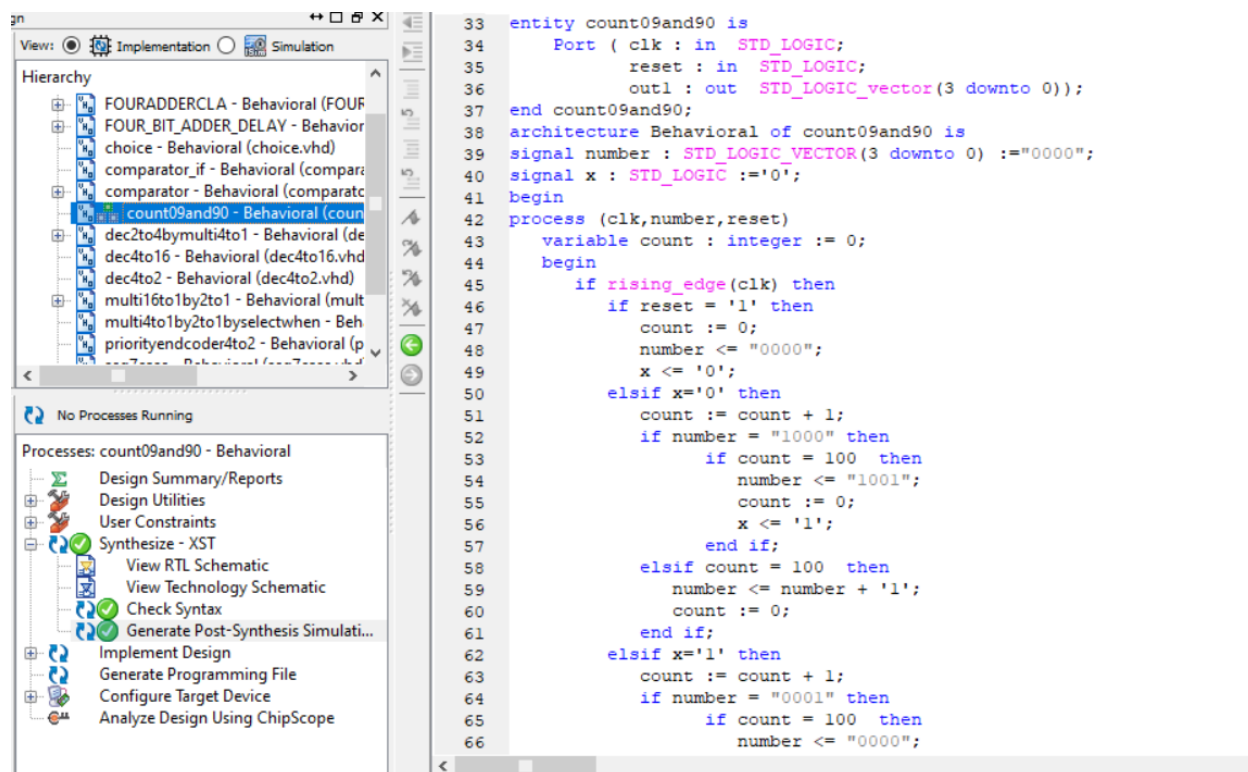
دو عکس زیر، ادامه یکدیگر می باشند:



سوال دوم:

برای ساختن شمارنده متغیری واسطه به نام X تعریف میکنیم که هر گاه 0 باشد شمارنده صعودی و هر گاه 1 باشد شمارنده بصورت نزولی عمل میکند. اکنون در پایان هر شمارش از 0 تا 9، مقدار x را از 0 به 1 و پایان هر شمارش از 9 به 0 مقدار x را از 1 به 0 تغییر میدهیم و این چرخه تکرار میشود.

دو تصویر زیر مربوط به کد میباشد:

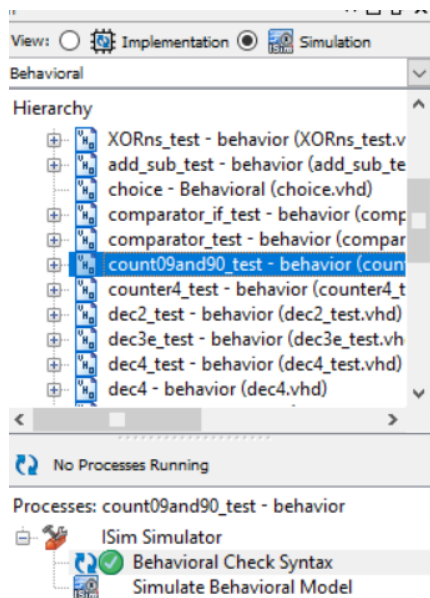


```
33 entity count09and90 is
34     Port ( clk : in  STD_LOGIC;
35           reset : in  STD_LOGIC;
36           out1 : out  STD_LOGIC_vector(3 downto 0));
37 end count09and90;
38 architecture Behavioral of count09and90 is
39     signal number : STD_LOGIC_VECTOR(3 downto 0) := "0000";
40     signal x : STD_LOGIC := '0';
41 begin
42     process (clk,number,reset)
43         variable count : integer := 0;
44     begin
45         if rising_edge(clk) then
46             if reset = '1' then
47                 count := 0;
48                 number <= "0000";
49                 x <= '0';
50             elsif x='0' then
51                 count := count + 1;
52                 if number = "1000" then
53                     if count = 100 then
54                         number <= "1001";
55                         count := 0;
56                         x <= '1';
57                     end if;
58                 elsif count = 100 then
59                     number <= number + '1';
60                     count := 0;
61                 end if;
62             elsif x='1' then
63                 count := count + 1;
64                 if number = "0001" then
65                     if count = 100 then
66                         number <= "0000";
```

```

65         if count = 100 then
66             number <= "0000";
67             count := 0;
68             x <= '0';
69         end if;
70         elsif count = 100 then
71             number <= number - '1';
72             count := 0;
73         end if;
74     end if;
75     out1 <= number;
76 end if;
77 end process;
78
79 end Behavioral;
80
81

```

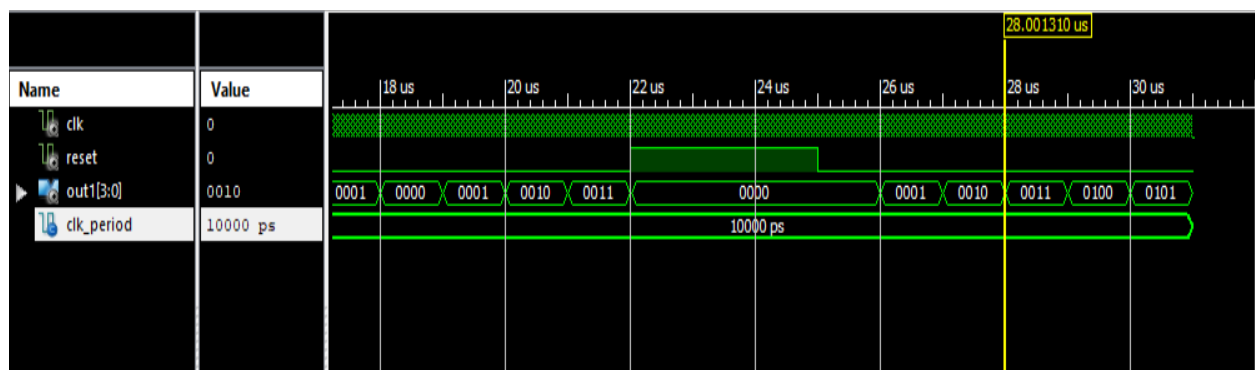
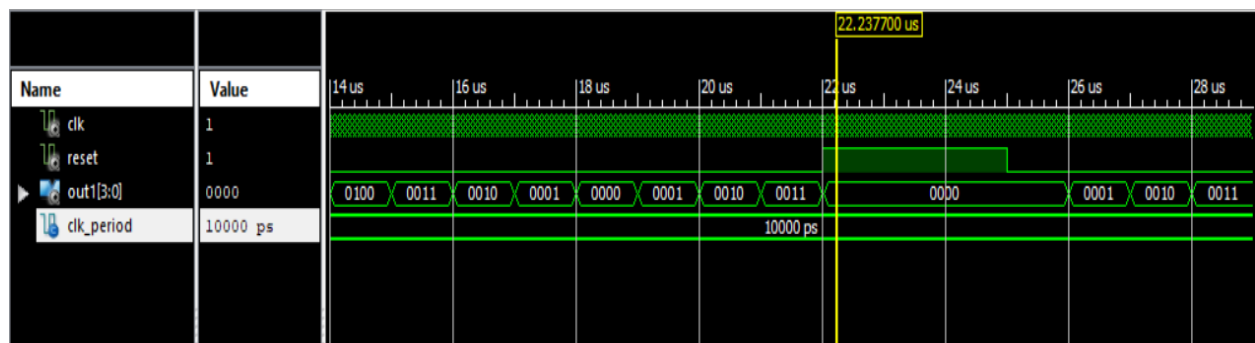
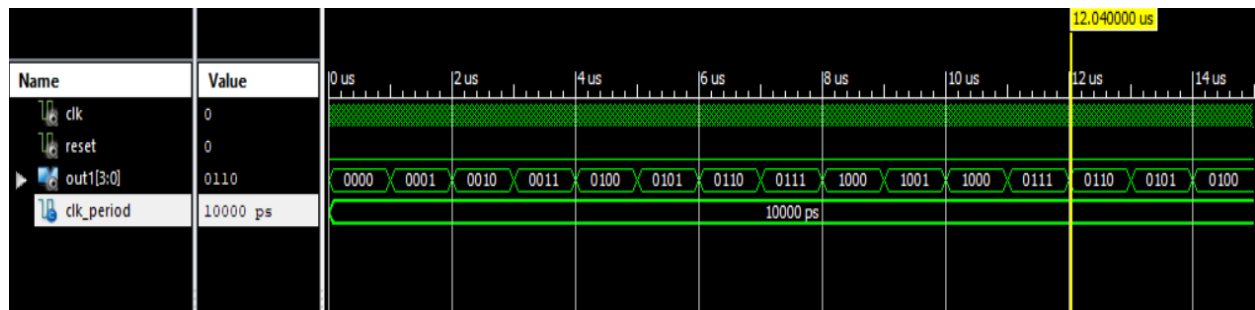


```

65         clk => clk,
66         reset => reset,
67         out1 => out1
68     );
69
70     -- Clock process definitions
71     clk_process : process
72     begin
73         clk <= '0';
74         wait for clk_period/2;
75         clk <= '1';
76         wait for clk_period/2;
77     end process;
78
79     -- Stimulus process
80     stim_proc: process
81     begin
82         -- hold reset state for 100 ns.
83         wait for 22 us;
84         reset <= '1';
85         wait for 3 us;
86         reset <= '0';
87
88
89         wait for clk_period*10;
90
91         -- insert stimulus here
92
93         wait;
94     end process;
95
96 END;
97
98

```

سه تصویر زیر مربوط به نتایج شبیه سازی میباشد:



سوال 3:

برای قسمت (ب) توضیحات در پاورقی نوشته شده است (وصل کردن تابع کلاک به متغیر x برای آن است که متغیر x نیاز به تغییر دستی نداشته باشد و عدد 18 برای آن است که یک چرخه کامل 18 بار کلاک نیاز دارد که نیمی از آن x برابر 0 و نیمی از آن x برابر 1 است).

۳- الف)

	PS	NS	D-FF
0000	0000	0001	0100
0001	0001	0010	0101
0010	0010	0011	0110
0011	0011	0100	0111
0100	0100	0101	1000
0101	0101	0110	1001
0110	0110	0111	1010
0111	0111	1000	1011
1000	1000	1001	1100
1001	1001	0000	1101

A B C D

O₁ O₂ O₃ O₄

input D-FF

$O_1 = A'D + BCD$

$O_2 = BC' + BD' + B'CD$

$O_3 = CD' + A'CD$

$O_4 = D'$

				NS	
PS		$X=0$	$X=1$		
0000	0000	0001	1001		
0001	0001	0010	0000		
0010	0010	0011	0001		
0011	0011	0100	0010		
0100	0100	0101	0011		
0101	0101	0110	0100		
0110	0110	0111	0101		
0111	0111	1000	0110		
1000	1000	1001	0111		
1001	1001	0000	1000		
A B C D		0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0'		0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0' 0'	

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	1	X	X

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	0	X	X

CD \ AB	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

CD \ AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$$O_1'' = AD + A'B'C'D' \quad O_2'' = BD + BC + AD' \quad O_3'' = CD + AD' + BC'D' \quad O_4'' = D'$$

$$O_i = O_i'X' + O_i''X = \begin{cases} (AD' + BC'D)X' + (AD + A'B'C'D')X & : O_1 \\ (BC' + BD' + BC'D)X' + (BD + BC + AD')X & : O_2 \\ (CD' + A'C'D)X' + (CD + AD' + BC'D')X & : O_3 \\ D'X' + D'X = D' & : O_4 \end{cases}$$

اگر t به Clock را بصورت $CLK(t)$ فرض کنیم، می توان $CLK(\frac{t}{1N})$

را به X متصل نمود.