

نویسنده:

علیرضا طباطبائی (9723052)



Amirkabir University of Technology
(Tehran Polytechnic)

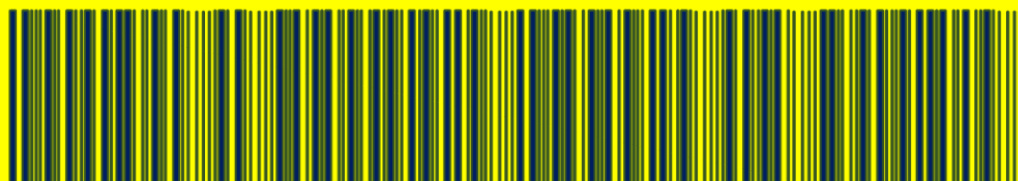
استاد آزمایشگاه:

دکتر علیرضا

ظاهری نوید



THE PRESENT IS THEIRS, THE FUTURE, FOR

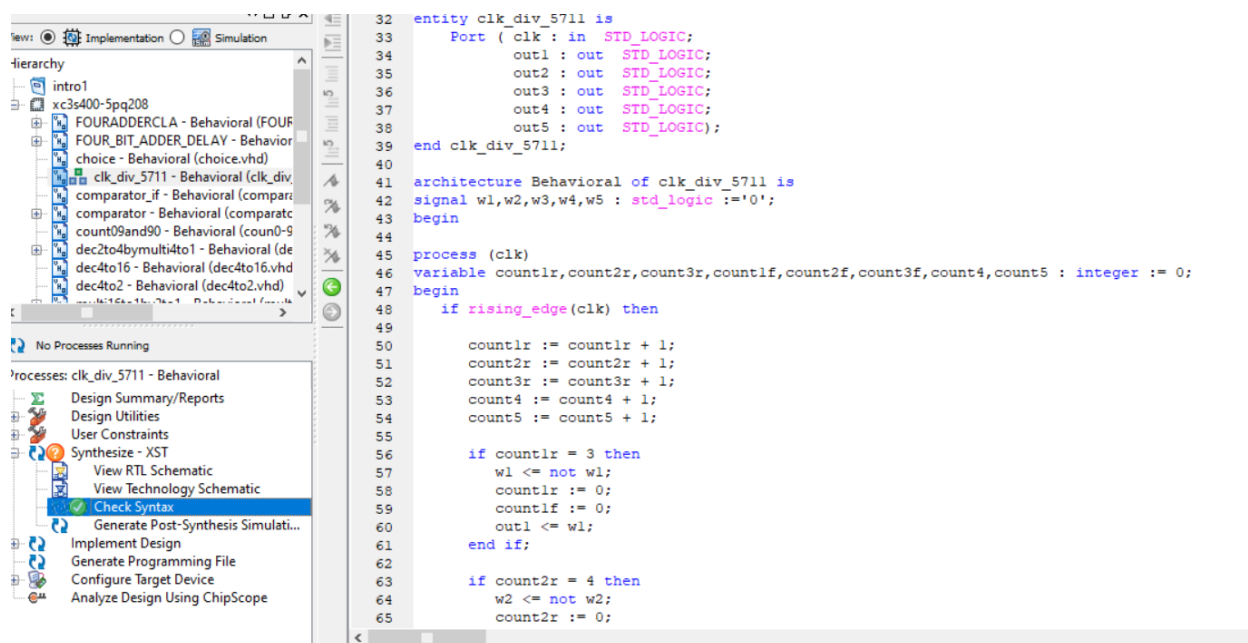


WHICH I REALLY WORK, IS MINE.

تمرین ۱:

برای تقسیم کلاک بصورت نرم افزاری باید یک شمارنده حساس به لبه بالا ساخت که هر n کلاک یکبار مقدار ۰ و ۱ را به هم تبدیل کند (توسط گیت not)

این روش برای تقسیم به اعداد زوج به راحتی قابل استفاده است اما برای اعداد فرد باید از یک ترفند استفاده کرد که این ترفند نیازمند استفاده از لبه پایین رونده کلاک نیز میباشد.



```
63     if count2r = 4 then
64         w2 <= not w2;
65         count2r := 0;
66         count2f := 0;
67         out2 <= w2;
68     end if;
69
70     if count3r = 6 then
71         w3 <= not w3;
72         count3r := 0;
73         count3f := 0;
74         out3 <= w3;
75     end if;
76
77     if count4 = 8 then
78         w4 <= not w4;
79         count4 := 0;
80         out4 <= w4;
81     end if;
82
83     if count5 = 13 then
84         w5 <= not w5;
85         count5 := 0;
86         out5 <= w5;
87     end if;
88
89     elsif falling_edge(clk) then
90
91         count1f := count1f + 1;
92         count2f := count2f + 1;
93         count3f := count3f + 1;
94
95         if count1f = 3 then
96             w1 <= not w1;
```

```

88
89     elsif falling_edge(clk) then
90
91         count1f := count1f + 1;
92         count2f := count2f + 1;
93         count3f := count3f + 1;
94
95         if count1f = 3 then
96             w1 <= not w1;
97             count1f := 0;
98             count1r := 0;
99             out1 <= w1;
100         end if;
101
102         if count2f = 4 then
103             w2 <= not w2;
104             count2f := 0;
105             count2r := 0;
106             out2 <= w2;
107         end if;
108
109         if count3f = 6 then
110             w3 <= not w3;
111             count3f := 0;
112             count3r := 0;
113             out3 <= w3;
114         end if;
115
116     end if;
117 end process;
118
119 end Behavioral;
120
121

```

تست پنج (که البته چیز خاصی در آن نیاز نبود بنویسیم):

View: ☐ Implementation ☒ Simulation

behavioral

Hierarchy

- add_sub_test - behavior (add_sub_te
- choice - Behavioral (choice.vhd)
- clk_div1_test - behavior (clk_div1_tes
- clk_div2_test - behavior (clk_div2_tes
- clk_div3_test - behavior (clk_div3_tes
- comparator_if_test - behavior (comp
- comparator_test - behavior (compar
- count09and90_test - behavior (coun
- counter4_test - behavior (counter4_t
- dec2_test - behavior (dec2_test.vhd)
- dec3e_test - behavior (dec3e_test.vh

No Processes Running

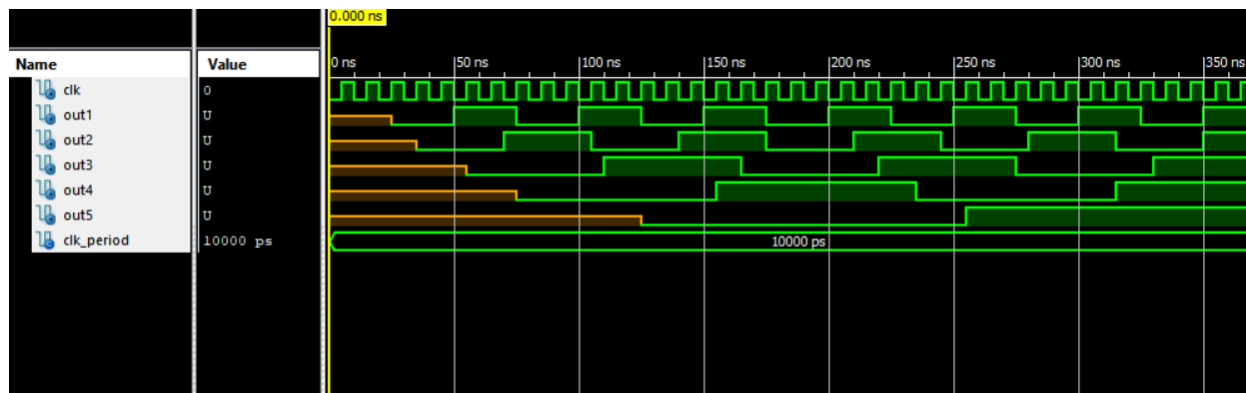
Processes: clk_div3_test - behavior

- ISim Simulator
- ☒ Behavioral Check Syntax
- ☐ Simulate Behavioral Model

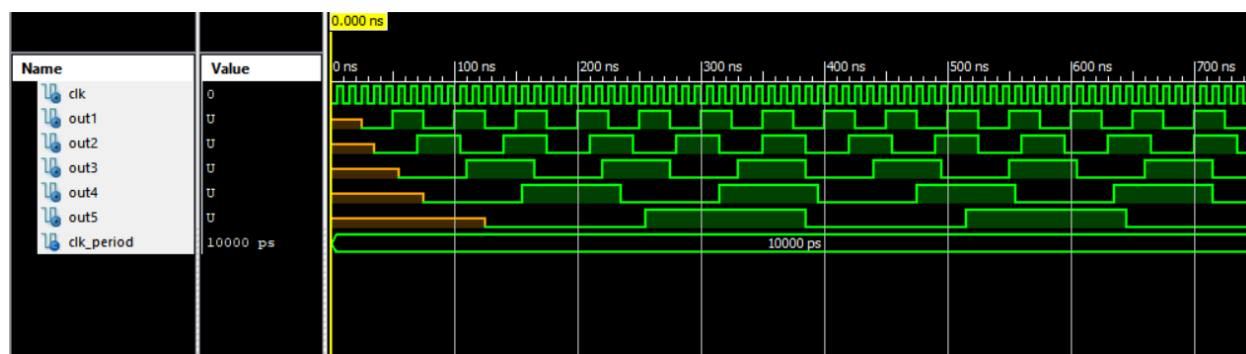
```
70 uut: clk_div_5711 PORT MAP (  
71     clk => clk,  
72     out1 => out1,  
73     out2 => out2,  
74     out3 => out3,  
75     out4 => out4,  
76     out5 => out5  
77 );  
78  
79 -- Clock process definitions  
80 clk_process :process  
81 begin  
82     clk <= '0';  
83     wait for clk_period/2;  
84     clk <= '1';  
85     wait for clk_period/2;  
86 end process;  
87  
88  
89 -- Stimulus process  
90 stim_proc: process  
91 begin  
92     -- hold reset state for 100 ns.  
93     wait for 100 ns;  
94  
95     wait for clk_period*10;  
96  
97     -- insert stimulus here  
98  
99     wait;  
100 end process;  
101  
102 END;  
103
```

clk_div_5711.vhd

نتایج:

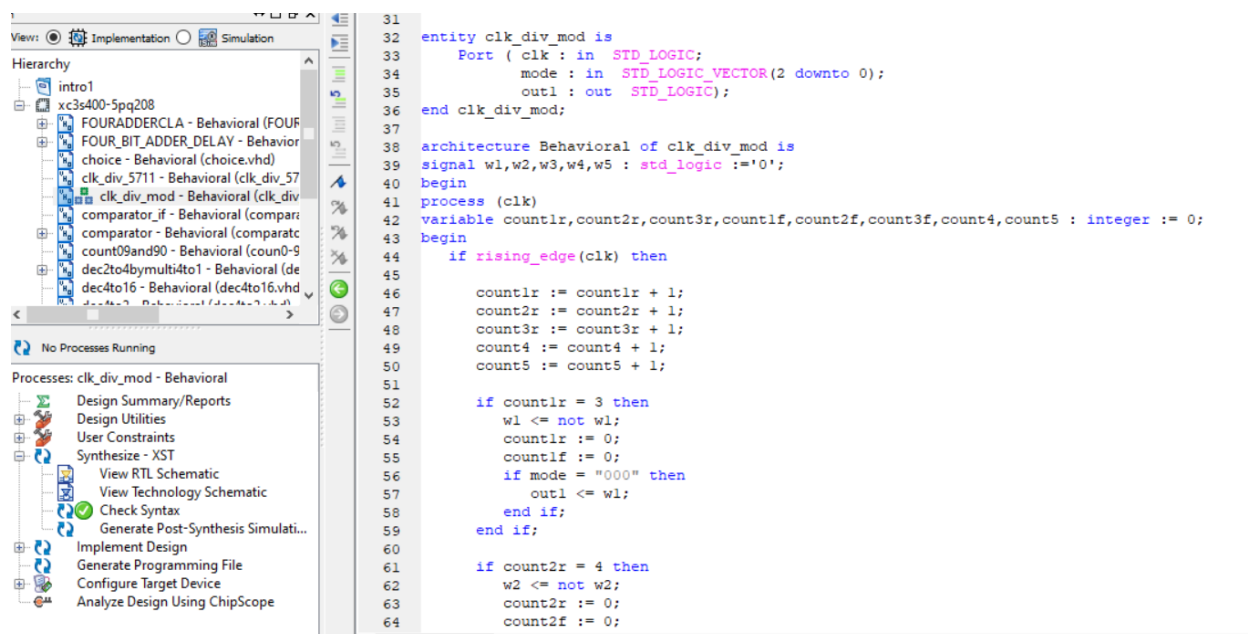


با کمی زوم بک:



تمرین ۲:

برای حالت mode دار از همان کد قبلی استفاده میکنیم با این تفاوت که فقط یک خروجی داریم و قبل از هر اتصال sig به خروجی یک if قرار میدهیم.



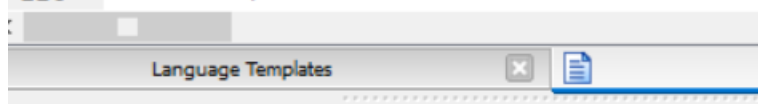
```
60
61     if count2r = 4 then
62         w2 <= not w2;
63         count2r := 0;
64         count2f := 0;
65         if mode = "001" then
66             out1 <= w2;
67         end if;
68     end if;
69
70     if count3r = 6 then
71         w3 <= not w3;
72         count3r := 0;
73         count3f := 0;
74         if mode = "010" then
75             out1 <= w3;
76         end if;
77     end if;
78
79     if count4 = 8 then
80         w4 <= not w4;
81         count4 := 0;
82         if mode = "011" then
83             out1 <= w4;
84         end if;
85     end if;
86
87     if count5 = 13 then
88         w5 <= not w5;
89         count5 := 0;
90         if mode = "100" then
91             out1 <= w5;
92         end if;
93     end if;
```



```

95     elsif falling_edge(clk) then
96
97         count1f := count1f + 1;
98         count2f := count2f + 1;
99         count3f := count3f + 1;
100
101         if count1f = 3 then
102             w1 <= not w1;
103             count1f := 0;
104             count1r := 0;
105             if mode = "000" then
106                 out1 <= w1;
107             end if;
108         end if;
109
110         if count2f = 4 then
111             w2 <= not w2;
112             count2f := 0;
113             count2r := 0;
114             if mode = "001" then
115                 out1 <= w2;
116             end if;
117         end if;
118
119         if count3f = 6 then
120             w3 <= not w3;
121             count3f := 0;
122             count3r := 0;
123             if mode = "010" then
124                 out1 <= w3;
125             end if;
126         end if;
127
128     end if;

```

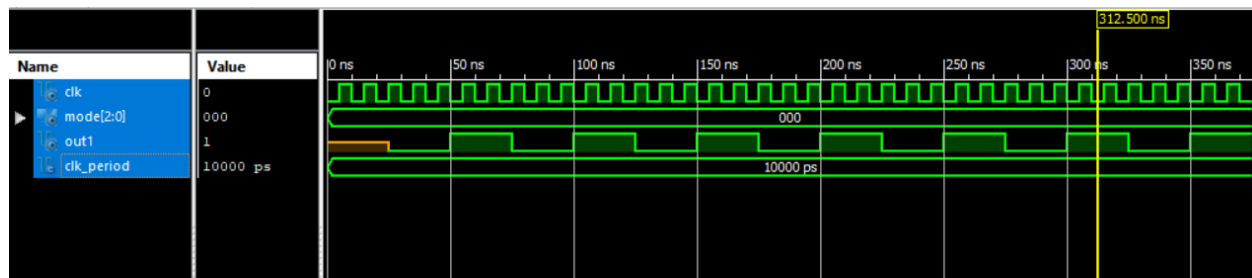


برای تست پنج ۲ حالت مختلف mode برابر ۰ و ۴ که به ترتیب کلاک را تقسیم به ۵ و ۲۶ میکنند امتحان میکنیم:

```

61 BEGIN
62
63     -- Instantiate the Unit Under Test (U
64     uut: clk_div_mod PORT MAP (
65         clk => clk,
66         mode => mode,
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         mode <= "000";
83         wait for 100 ns;
84
85         wait for clk_period*10;
86
87         -- insert stimulus here
88
89         wait;
90     end process;
91
92
93 END;

```



```

BEGIN

    -- Instantiate the Unit Under Tes
    uut: clk_div_mod PORT MAP (
        clk => clk,
        mode => mode,
        out1 => out1
    );

    -- Clock process definitions
    clk_process :process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
        mode <= "100";
        wait for 100 ns;

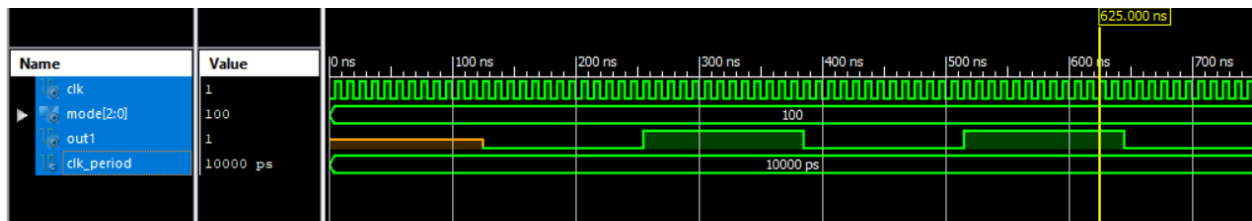
        wait for clk_period*10;

        -- insert stimulus here

        wait;
    end process;

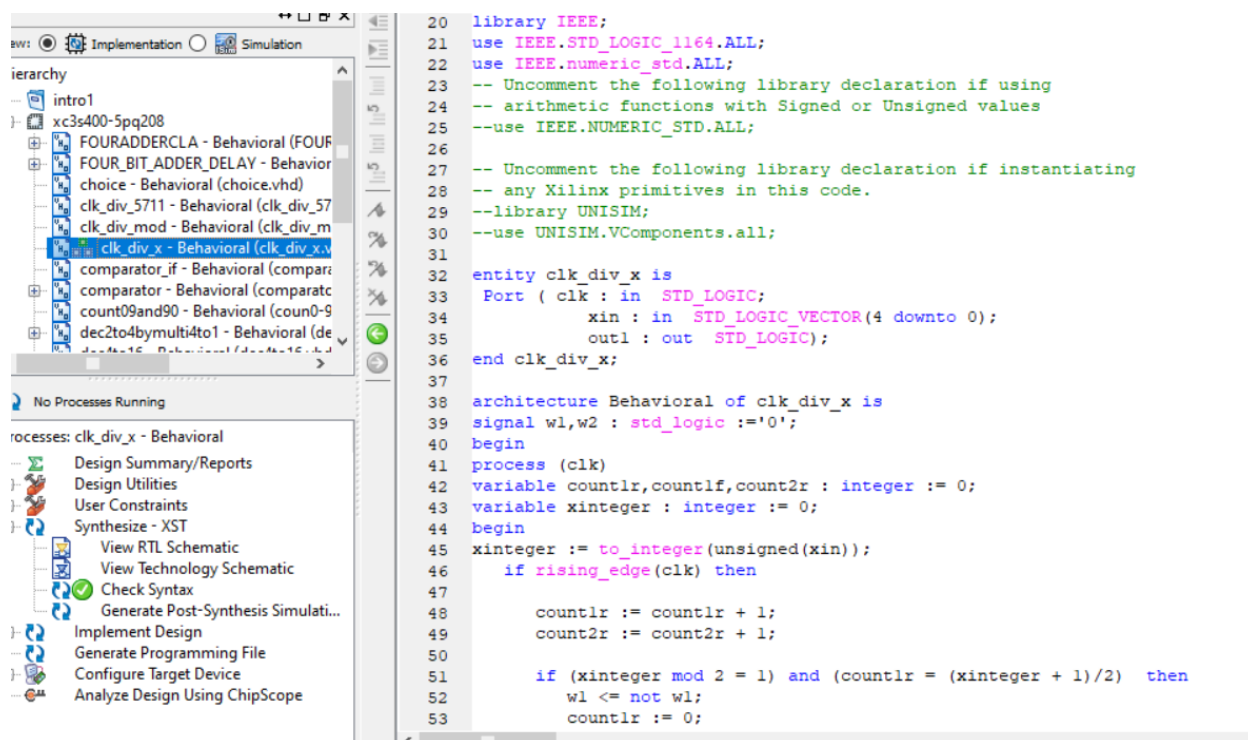
END;

```



امتیازی:

عدد xin را در ورودی دریافت کرده (میتوان بیت های بیشتری به آن اختصاص داد اما در اینجا ۵ بیت تعریف شده اما هر مقدار بیتی میتوان به آن اختصاص داد) و سپس توسط تابع to_integer آن را به integer تبدیل میکنیم و سپس زوج و فرد بودن آن را توسط تابع mod تشخیص میدهیم و سپس طبق قاعده بخش های قبلی عمل میکنیم.



The screenshot displays the Xilinx Vivado IDE interface. On the left, the 'Hierarchy' pane shows the project structure, with 'clk_div_x - Behavioral (clk_div_x.v)' selected. Below it, the 'Processes' pane lists various design tasks, with 'Synthesize - XST' and 'View RTL Schematic' visible. The main editor window shows the VHDL code for the 'clk_div_x' entity. The code includes library declarations for IEEE and UNISIM, followed by the entity definition and its behavioral architecture. The architecture uses a process to handle the clock signal, incrementing counters and outputting a signal 'w1' based on the modulo 2 result of the counter and the input 'xin'.

```
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.numeric_std.ALL;
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity clk_div_x is
33   Port ( clk : in  STD_LOGIC;
34         xin : in  STD_LOGIC_VECTOR(4 downto 0);
35         out1 : out STD_LOGIC);
36 end clk_div_x;
37
38 architecture Behavioral of clk_div_x is
39   signal w1,w2 : std_logic := '0';
40   begin
41   process (clk)
42     variable count1r,count1f,count2r : integer := 0;
43     variable xinteger : integer := 0;
44   begin
45     xinteger := to_integer(unsigned(xin));
46     if rising_edge(clk) then
47
48       count1r := count1r + 1;
49       count2r := count2r + 1;
50
51       if (xinteger mod 2 = 1) and (count1r = (xinteger + 1)/2) then
52         w1 <= not w1;
53         count1r := 0;
```

View: ☒ Implementation ☐ Simulation

Hierarchy

- intro1
- xc3s400-5pq208
- FOURADDERCLA - Behavioral (FOURADDERCLA.vhd)
- FOUR_BIT_ADDER_DELAY - Behavioral (FOUR_BIT_ADDER_DELAY.vhd)
- choice - Behavioral (choice.vhd)
- clk_div_5711 - Behavioral (clk_div_5711.vhd)
- clk_div_mod - Behavioral (clk_div_mod.vhd)
- clk_div_x - Behavioral (clk_div_x.vhd)
- comparator_if - Behavioral (comparator_if.vhd)
- comparator_test - Behavioral (comparator_test.vhd)
- count09and90 - Behavioral (count09and90.vhd)
- dec2to4bymulti4to1 - Behavioral (dec2to4bymulti4to1.vhd)
- counter4_test - Behavioral (counter4_test.vhd)

No Processes Running

Processes: clk_div_x - Behavioral

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

```

47
48   count1r := count1r + 1;
49   count2r := count2r + 1;
50
51   if (xinteger mod 2 = 1) and (count1r = (xinteger + 1)/2) then
52       w1 <= not w1;
53       count1r := 0;
54       count1f := 0;
55       out1 <= w1;
56   end if;
57
58   if (xinteger mod 2 = 0) and (count2r = xinteger/2) then
59       w2 <= not w2;
60       count2r := 0;
61       out1 <= w2;
62   end if;
63
64   elsif falling_edge(clk) then
65
66       count1f := count1f + 1;
67
68       if (xinteger mod 2 = 1) and (count1f = (xinteger+1)/2) then
69           w1 <= not w1;
70           count1f := 0;
71           count1r := 0;
72           out1 <= w1;
73       end if;
74   end if;
75 end process;
76 end Behavioral;
77
78
79
80

```

View: ☐ Implementation ☒ Simulation

Behavioral

Hierarchy

- add_sub_test - behavior (add_sub_test.vhd)
- choice - Behavioral (choice.vhd)
- clk_div_5711_test - behavior (clk_div_5711_test.vhd)
- clk_div_mod_test - behavior (clk_div_mod_test.vhd)
- clk_div_x1_test - behavior (clk_div_x1_test.vhd)
- clk_div_x2_test - behavior (clk_div_x2_test.vhd)
- clk_div_x_test - behavior (clk_div_x_test.vhd)
- comparator_if_test - behavior (comparator_if_test.vhd)
- comparator_test - behavior (comparator_test.vhd)
- count09and90_test - behavior (count09and90_test.vhd)
- counter4_test - behavior (counter4_test.vhd)

No Processes Running

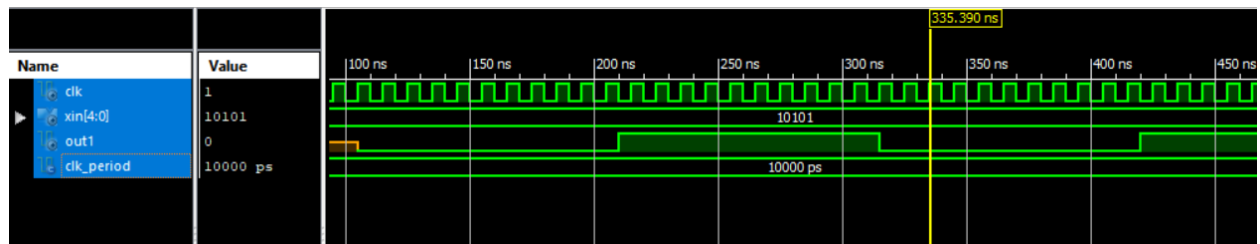
Processes: clk_div_x_test - behavior

- ISim Simulator
- Behavioral Check Syntax
- Simulate Behavioral Model

```

61 BEGIN
62
63   -- Instantiate the Unit Under Test (UUT)
64   uut: clk_div_x PORT MAP (
65       clk => clk,
66       xin => xin,
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       xin <= "10101";
83       wait for 1000 ns;
84
85       wait for clk_period*10;
86
87       -- insert stimulus here
88
89       wait;
90   end process;
91
92
93 END;
94

```



```
BEGIN
```

```
-- Instantiate the Unit Under Test (UUT)
```

```
uut: clk_div_x PORT MAP (
    clk => clk,
    xin => xin,
    out1 => out1
);
```

```
-- Clock process definitions
```

```
clk_process :process
begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
end process;
```

```
-- Stimulus process
```

```
stim_proc: process
begin
    xin <= "01100";
    wait for 1000 ns;

    wait for clk_period*10;

    -- insert stimulus here

    wait;
end process;
```

```
END;
```

