



VHDL I

Submitted by:

Menna Tallah Mater Abdelhamed

Track:

Digital IC

Submitted to:

Dr. Watheq

Contents

| | |
|-----------------------------------|----|
| 1. Introduction: | 2 |
| 2. Test bench methodology: | 2 |
| 3. Examples: | 3 |
| 3.1.1 Counter: | 3 |
| a. function of Design: | 3 |
| b. Test strategy: | 3 |
| c. Simulation results: | 4 |
| c. synthesis and schematic: | 4 |
| 3.1.2 Updated Counter: | 4 |
| 3.2 Increment: | 6 |
| a. function of Design: | 6 |
| b. Test strategy: | 6 |
| c. Simulation results: | 8 |
| d. synthesis and schematic: | 9 |
| 3.3 FSM_moore_2p: | 9 |
| a. function of Design: | 9 |
| b. Test strategy: | 9 |
| c. Simulation results: | 12 |
| d. synthesis and schematic: | 13 |
| e. Inserting Error: | 13 |
| 3.4 Bin_2_gray: | 16 |
| a. function of Design: | 16 |
| b. Test strategy: | 17 |
| c. Simulation results: | 20 |
| d. synthesis and schematic: | 20 |
| e. Inserting Error: | 21 |
| 3.5 D_ff_Ayncr: | 22 |
| a. function of Design: | 22 |
| b. Test strategy: | 22 |
| c. Simulation results | 25 |
| d. synthesis and schematic: | 25 |

1. Introduction:

Test bench is powerful instrument to insure the work of functionality (behavior) of any circuit in ideal cases which doesn't care about delay from wires or the around environment at pre_synthesis stage .it gives an early indication of the performance that components and subsystems will do its work when be in real system or not. The quality of test bench depends on how it covers all possible cases of the tested design.

2. Test bench methodology:

- ✓ It has empty entity.
- ✓ It includes a component declaration section which has input and output declaration.
- ✓ It has signals with connect to the ports of the design to test it which acts as box around the design to test it.
- ✓ It includes the test component instantiation which called Unit Under Test (UUT) and mapped it to the signal in previous step.
- ✓ It includes process to handle the change in inputs and how it effect on output.
- ✓ If there are more than one input effect in output, it put in different separated process to cover all cases.
- ✓ All that processes work concurrent.
- ✓ The strength of test bench appears in how it covers all possible cases.
- ✓ Design can be tested use many methods as using input – output text file ,assert –report and ordinary test bench.

3. Examples:

3.1.1 Counter:

a. function of Design:

-count from 0 to 15

b. Test strategy:

- **Steps:**

- know the functionality of circuit to detect the output according to change in the input

- know the type of circuit if it synchronous, asynchronous or combinational, in that design is combinational circuit with only one input.

-from the entity of the circuit ,the only input is the "clk" which represent in one bit from type bit so would take value "0" or "1" ,so it should be generated wave of clk changing with time with duty cycle 50% in process includes wait statement and detect the changing in output according to change in input.

| Input(clk) | Output(count) |
|------------|---------------|
| 0 | Don't change |
| 1 | Count+1 |

- **Testbench:**

```
-----  
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
entity tb_counter is  
-- Port ( );  
end tb_counter;  
  
architecture Behavioral_tb_counter of tb_counter is  
component counter  
    PORT( clk: IN bit;  
          count: OUT natural RANGE 0 TO 15);  
END component;  
signal clk : bit := '0';  
signal count : natural RANGE 0 TO 15;  
begin  
    uut_counter : counter port map (clk=>clk,count=>count );  
    create_clk_process : process  
    --clk_oeriod 20 ns  
    begin  
        clk <= '0';  
        wait for 10 ns;  
        clk <= '1';
```

```

        wait for 10 ns;
    end process;
end Behavioral_tb_counter;

```

c. Simulation results:

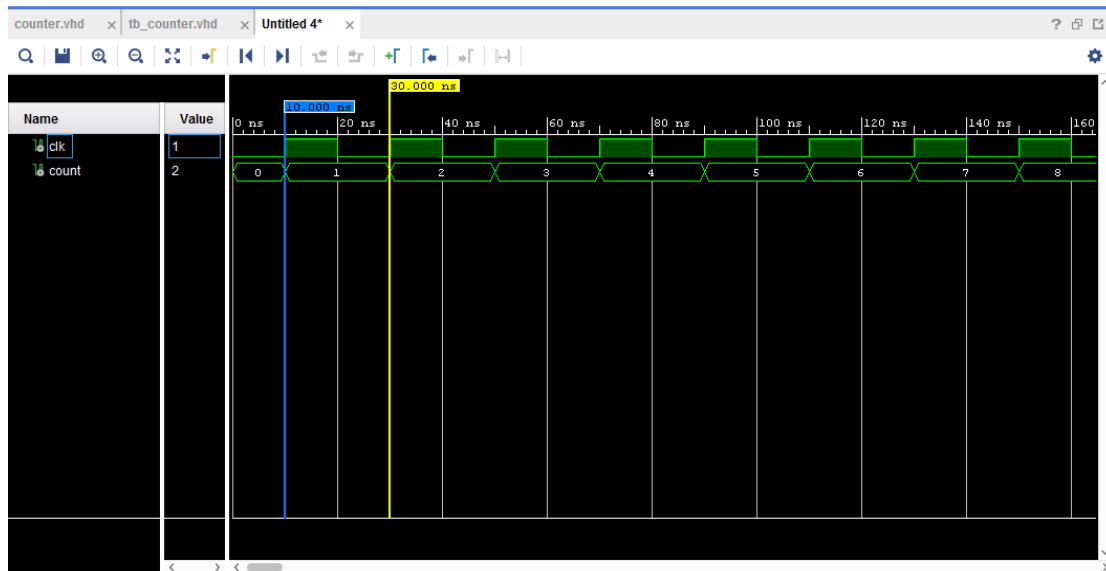


Figure 1:simulation of counter

c. synthesis and schematic:

-can't synthesis this circuit because it contain loop (in time loop).

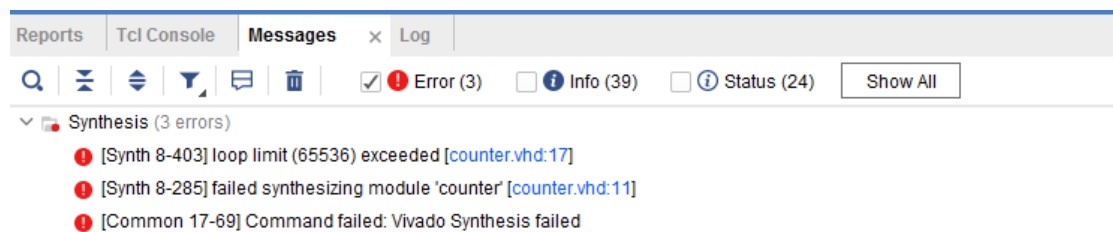


Figure 2:Error message,synthesis failed

3.1.2 Updated Counter

-added input rst to reset the counter and make it count from zero.

-make it responses with the positive edge of clk.

-added flip flop when completed the counting to 15, its output will be one for one cycle.

- used flip flop to store the value and try how instantiation component in the design.

a. Test strategy:

-generated wave of clk with duty cycle 50% in process and changing the input rst from 0 to 1 in other process to detect the output.

| clk | Rst | Count |
|-----------------|-----|-----------|
| Rising edge | 1 | 0000 |
| Not rising edge | 1 | N0 change |
| Rising edge | 0 | Count+1 |
| Not rising edge | 0 | N0 change |

b. Simulation:

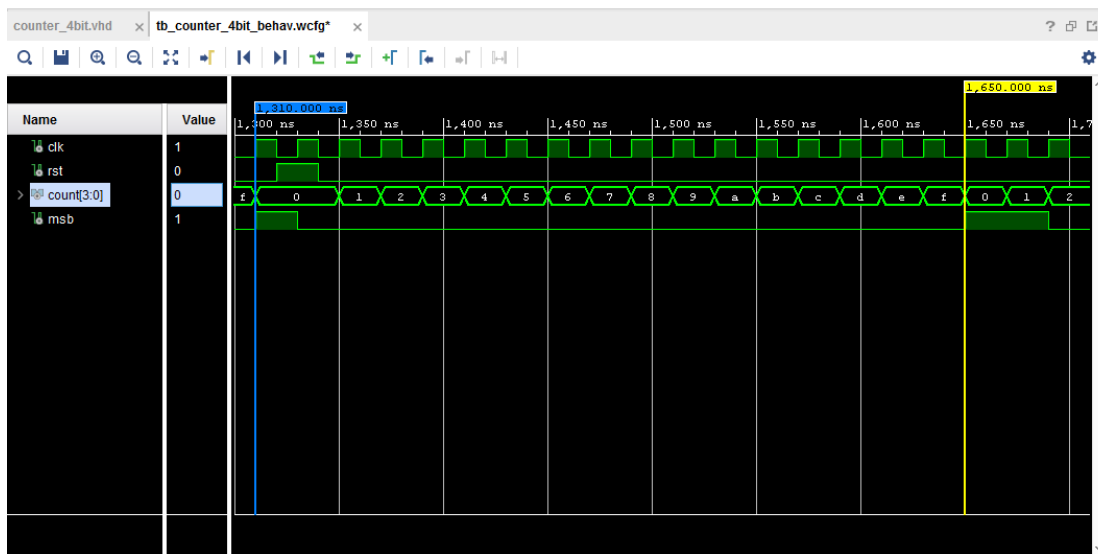


Figure3:simulation of updated counter

c. synthesis and schematic :

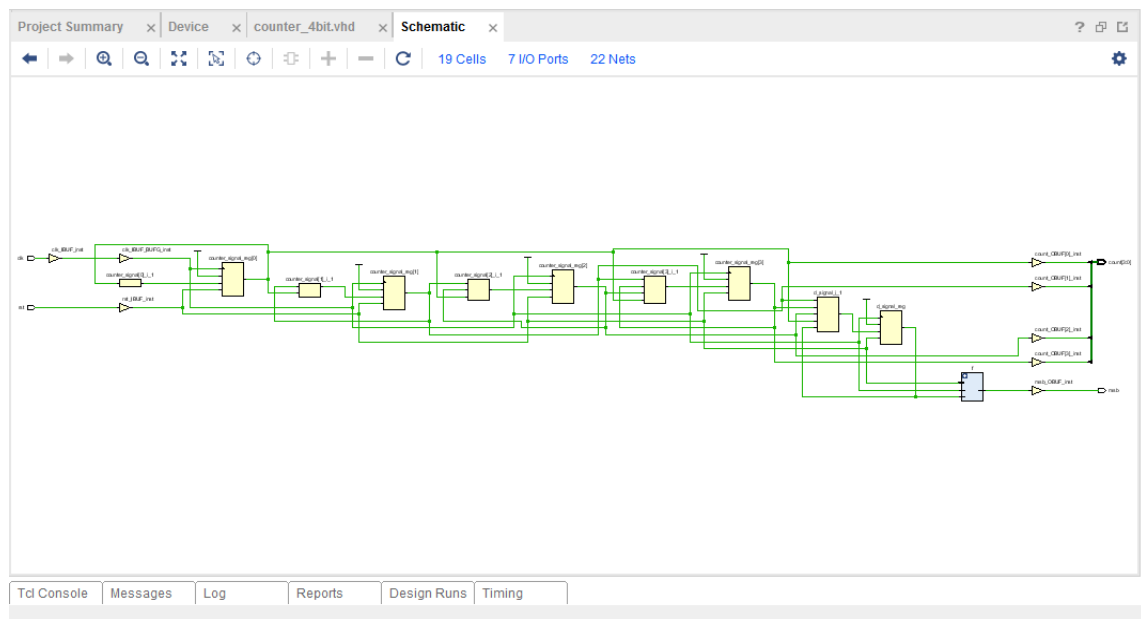


Figure4: schematic of updated counter

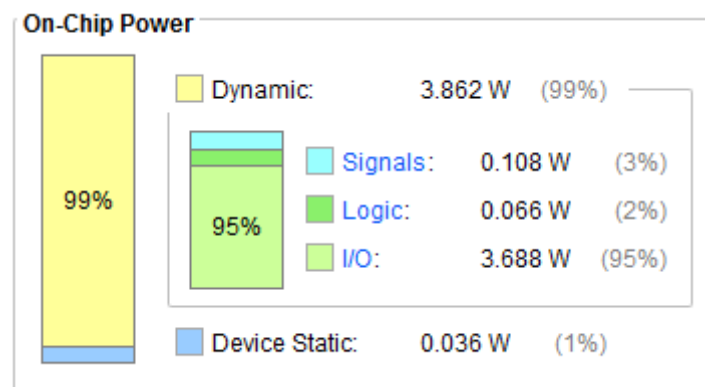


Figure5:power on device "xa7s6cpgg 196-2I"

3.2 Increment:

a. function of Design:

-increment the output when input "inc"=1.

b. Test strategy:

- **Steps:**

- know the functionality of circuit to detect the output according to change in the input
- know the type of circuit if it synchronous, asynchronous or combinational, in that design is combinational circuit with only one input.

-set inc signal to be one to detect the incrementing in output ,it is one bit so it has only two cases being one or zero

-to detect the output according changing in input, create process changing the input from zero to one each 50 ns.

| Input(inc) | Output(count) |
|------------|---------------|
| 0 | Don't change |
| 1 | Count+1 |

- **Notes:**

-output z is unsigned value from 2 bit so it can only store up to "2" and any increment in it gets from zero to two again .

-it can fixed that by adding flag when the value reaches to "2", sets that flag by one.

- **Testbench :**

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
USE ieee.numeric_std.ALL;
```

```
entity tb_increment is
```

```
-- Port ( );
```

```
end tb_increment;
```

```
architecture Behavioral_tb_increment of tb_increment is
```

```
component increment
```

```
    PORT( inc: IN bit;
```

```
          z: OUT unsigned (0 TO 1));
```

```
END component;
```

```
signal inc : bit := '0';
```

```
signal z: unsigned (0 TO 1) := (OTHERS => '0');
```

```
begin
```

```
increment_1:increment port map (inc=>inc,z=>z);
```

```
process
```

```
begin
```

```
wait for 50 ns;
```



```

inc <='0';

wait for 50 ns;

inc <= '1';

wait for 50 ns;

end process;

end Behavioral_tb_increment;

```

c. Simulation results:

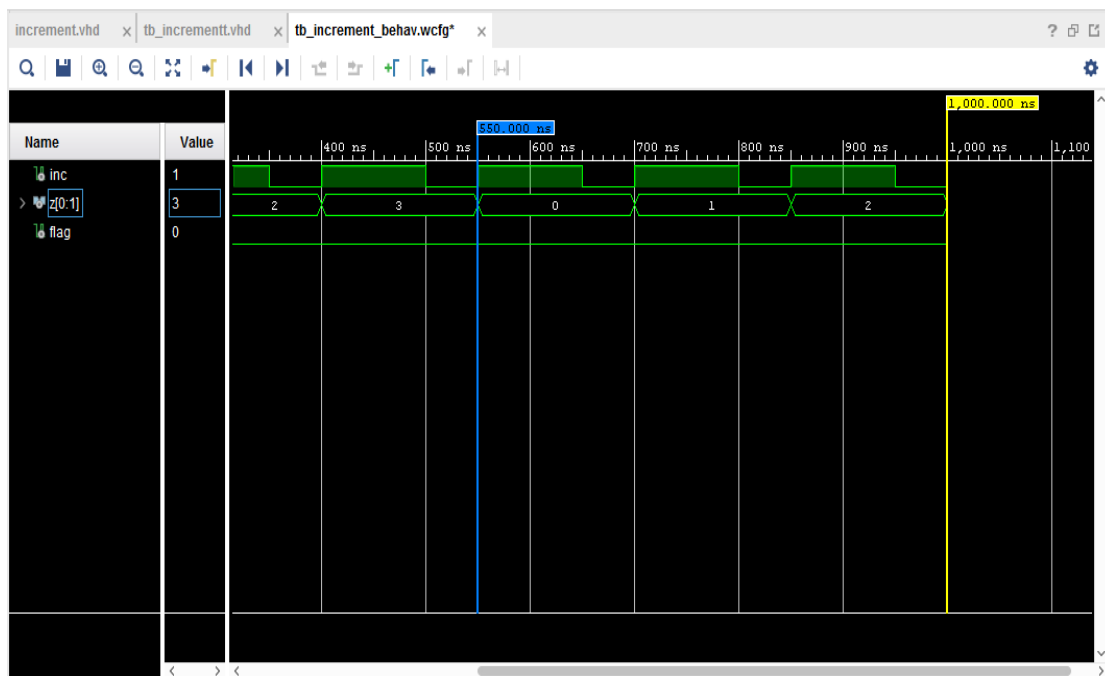


Figure6:simulation of increment

Marks here verified that output is only incremented when input is one and if it is zero still at same value.

d. synthesis and schematic:

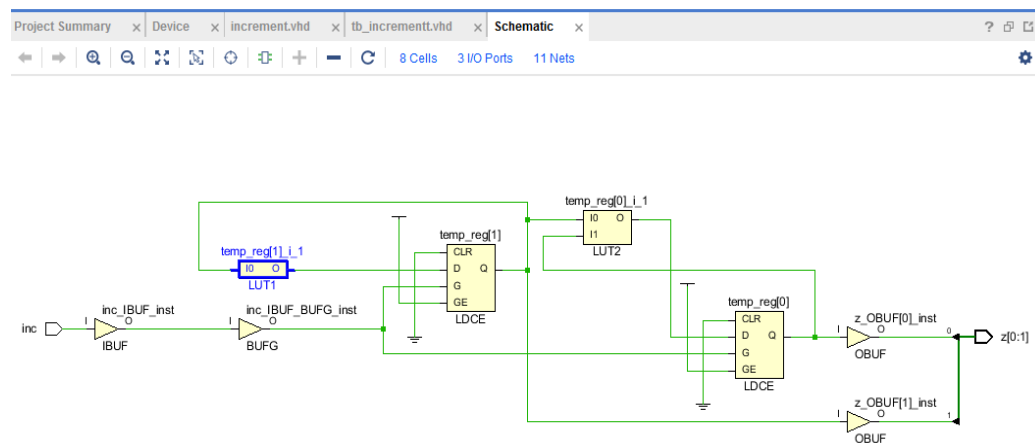


Figure7:schematic of increment

3.3 FSM_moore_2p:

a. function of Design:

-translate from state to other according inputs.

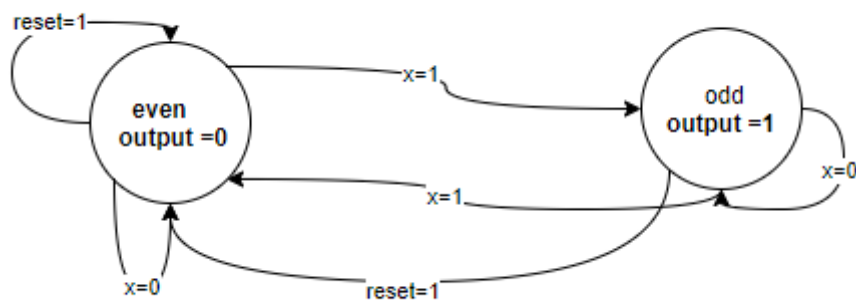


Figure8:diagram of FSM

-output is function in current state.

-if reset =1, state =even.

-if reset =0, x=0 => state doesn't change.

-if reset =0, x=1 => state changes.

So this circuit detects if number even or odd and when reset=1 returns to even state.

b. Test strategy:

- **Steps:**

-use assert and report warning if the output of that state doesn't be the expected output.

-test cases is chosen according to previous diagram of FSM in figure (8)

To cover all possible cases.

- **Testbench:**

```
-----  
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
  
entity tb_fsm_moore_2p_using_assert is  
-- Port ( );  
end tb_fsm_moore_2p_using_assert;  
  
architecture Behavioral_using_assert of tb_fsm_moore_2p_using_assert is  
component fsm  
  PORT( clk, reset: IN std_logic;  
        x: IN std_logic;  
        y: OUT std_logic);  
END component;  
SIGNAL clk, reset, y: std_logic;  
SIGNAL x: std_logic:= '0';  
  
begin  
  fsm_used_assert:fsm PORT MAP (clk, reset, x, y);  
  clk_process:process  
  begin  
    clk<='1';  
    wait for 20 ns;  
    clk<='1';  
    wait for 20 ns;  
  end process;  
  
  -----  
  assert_process:process  
  begin  
    reset <= '1'; --current_state=even ,output=0,x=0 :next_current:even,x=1  
    :next_current:odd  
    WAIT FOR 20 ns;  
    ASSERT y = '0'  
    REPORT "Error:current_state=even => output=0 "  
    SEVERITY warning;  
  
    -----  
    reset <= '0'; --current_state=even  
    x <= '0'; --  
    WAIT FOR 20 ns;  
    ASSERT y = '0'  
    REPORT "Error:current_state=even => output=0 "
```

```
SEVERITY warning;
-----
WAIT FOR 20 ns;
x <= '1';  --
WAIT FOR 20 ns;
ASSERT y = '0'
  REPORT "Error: current_state=even => y = 0"
  SEVERITY warning;
WAIT FOR 20 ns;
x <= '0';
WAIT FOR 20 ns;
ASSERT y = '0'
  REPORT "Error: current_state=even => y = 0"
  SEVERITY warning;
WAIT FOR 20 ns;
x <= '1';
WAIT FOR 20 ns;
ASSERT y = '0' --change
  REPORT "Error: current_state=even => y = 0"
  SEVERITY warning;
WAIT;
end process;
end Behavioral_using_assert;
```

```

Tcl Console x Messages Log
[Icons: Search, Run, Break, Stop, Copy, Paste, Delete]

Compiling architecture behavioral_using_assert of entity xil_defaultlib.tb_fsm_moore_2p_using_assert
Built simulation snapshot tb_fsm_moore_2p_using_assert_behav
INFO: [USF-XSim-69] 'elaborate' step finished in '5' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'D:/vhdl_ITI/fsm/fsm.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
    with args "tb_fsm_moore_2p_using_assert_behav -key {Behavioral:sim_1:Functional:tb_fsm_moore_2p_using_assert} -tclbatch {tb_fsm_moore_2p_using_
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_fsm_moore_2p_using_assert.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#     if { [llength [get_objects]] > 0 } {
#         add_wave /
#         set_property needs_save false [current_wave_config]
#     } else {
#         send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window g
#     }
# }
# run 1000ns
xsim: Time (s): cpu = 00:00:23 ; elapsed = 00:00:09 . Memory (MB): peak = 807.023 ; gain = 8.918
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_fsm_moore_2p_using_assert_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:28 ; elapsed = 00:00:18 . Memory (MB): peak = 807.023 ; gain = 8.918
|
Type a Tcl command here

```

Figure9:Tcl console

- **Notes:** No error because the output of each stage is the expected output in test bench.

c. Simulation results:

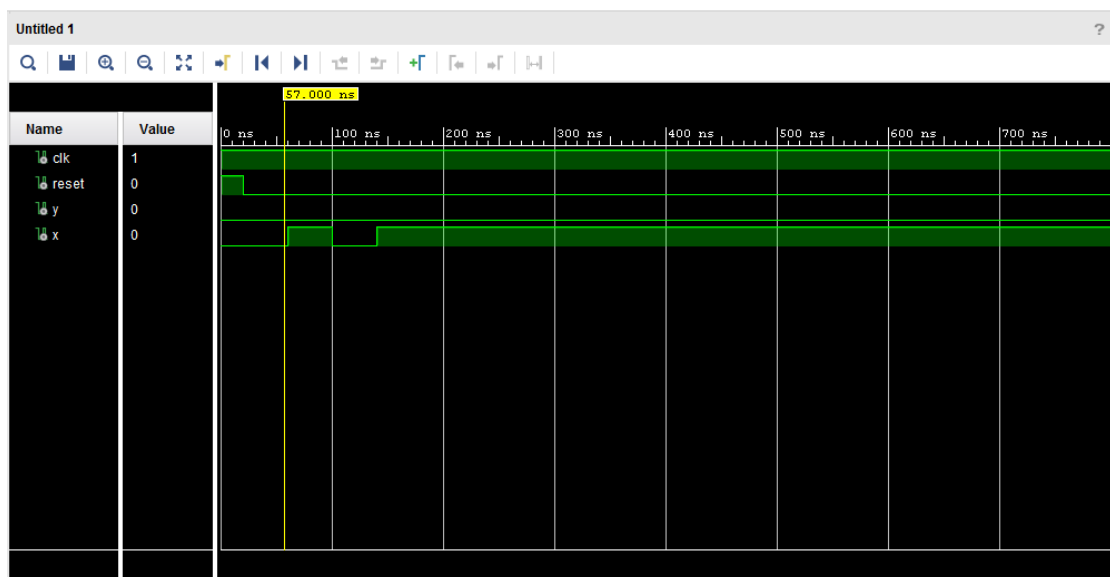


Figure10:Simulation of FSM

d. synthesis and schematic:

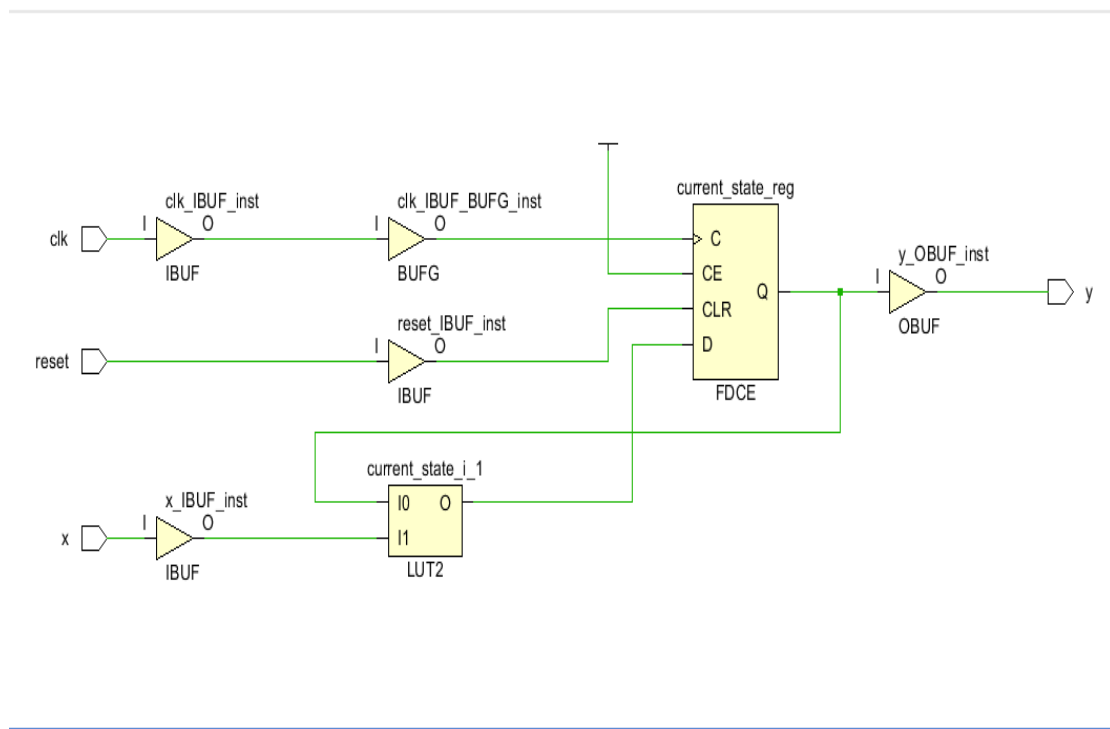


Figure11:schematic of FSM ,it represent in register and combinational logic

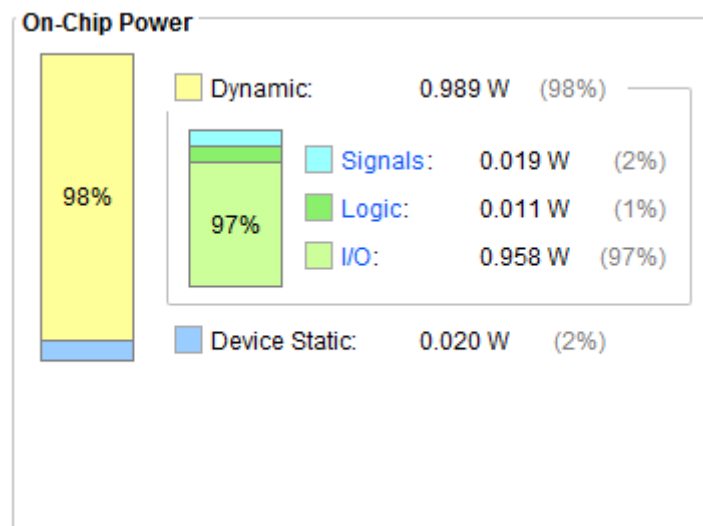


Figure 12: power of FSM on device "xa7s6cpga 196-2I"

e. Inserting Error:

- **Testbench:**

library IEEE;

```

use IEEE.STD_LOGIC_1164.ALL;

entity tb_FSM_insert_ERRORS is
-- Port ( );
end tb_FSM_insert_ERRORS;

architecture Behavioral_tb_FSM_insert_ERRORS of tb_FSM_insert_ERRORS is
component fsm
  PORT( clk, reset: IN std_logic;
        x: IN std_logic;
        y: OUT std_logic);
END component;
SIGNAL clk, reset, y: std_logic;
SIGNAL x: std_logic:='0';

begin

fsm_used_assert:fsm PORT MAP (clk, reset, x, y);
clk_process:process

begin
clk<='1';
wait for 20 ns;
clk<='1';
wait for 20 ns;
end process;

-----

assert_process:process
begin
-----

  reset <= '1'; --current_state=even ,output=0,x=0 :next_current:even,x=1
:next_current:odd

```

```

WAIT FOR 20 ns;
ASSERT y = '0'
    REPORT "Error:current_state=even => output=0 "
    SEVERITY warning;

-----

reset <= '0'; --current_state=even
x <= '1';    --
WAIT FOR 20 ns;
ASSERT y = '0'
    REPORT "Error:current_state=even => output=0 "
    SEVERITY warning;

-----

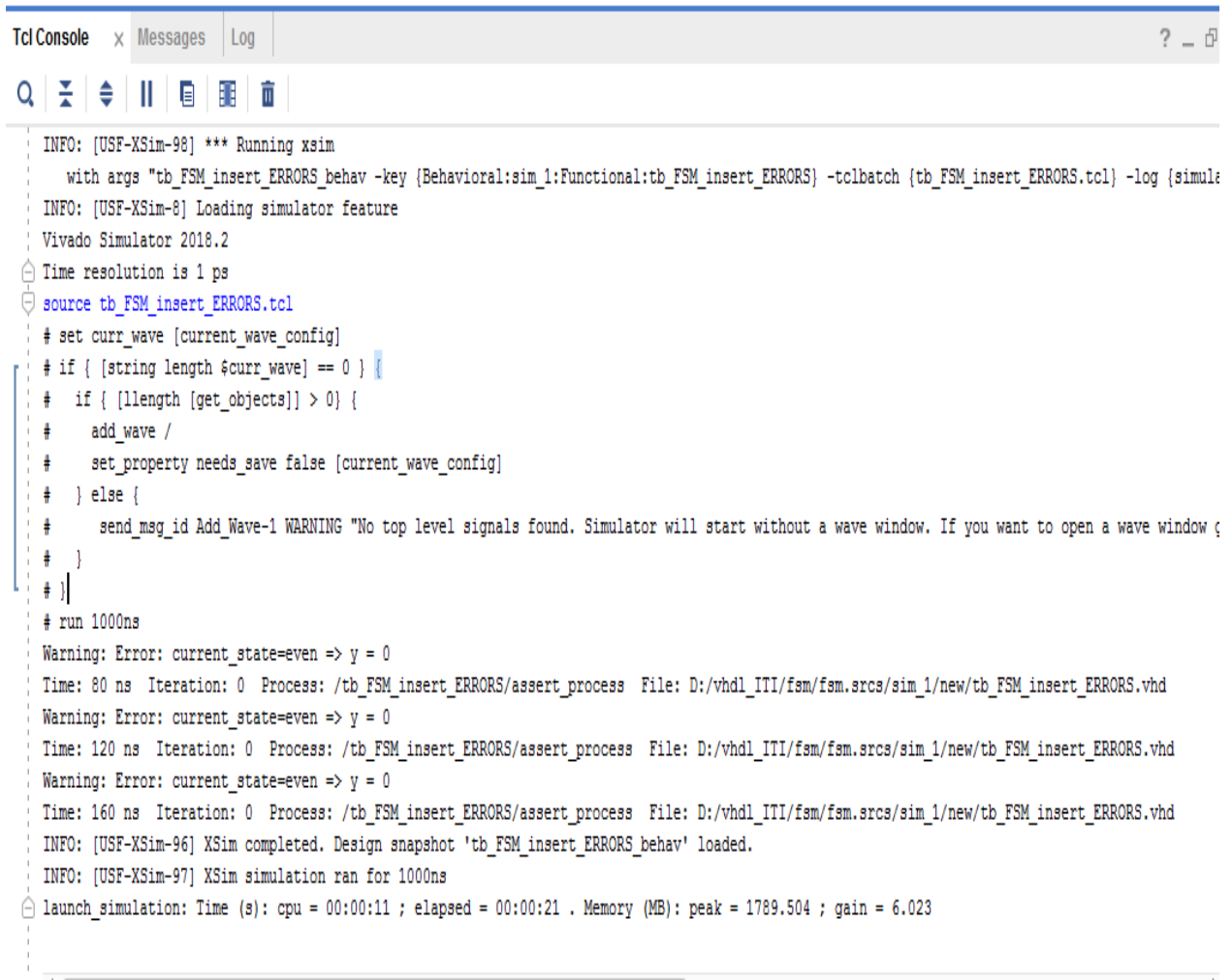
WAIT FOR 20 ns;
x <= '1';    --
WAIT FOR 20 ns;
ASSERT y = '1'
    REPORT "Error: current_state=even => y = 0"
    SEVERITY warning;
WAIT FOR 20 ns;
x <= '0';
WAIT FOR 20 ns;
ASSERT y = '1'
    REPORT "Error: current_state=even => y = 0"
    SEVERITY warning;
WAIT FOR 20 ns;
x <= '1';
WAIT FOR 20 ns;
ASSERT y = '1' --change
    REPORT "Error: current_state=even => y = 0"
    SEVERITY warning;

```


WAIT;

end process;

end Behavioral_tb_FSM_insert_ERRORS;



The screenshot shows the Vivado Tcl Console window. The title bar includes 'Tcl Console', 'Messages', and 'Log'. The console displays the following text:

```
INFO: [USF-XSim-98] *** Running xsim
  with args "tb_FSM_insert_ERRORS_behav -key {Behavioral:sim_1:Functional:tb_FSM_insert_ERRORS} -tclbatch {tb_FSM_insert_ERRORS.tcl} -log {simulatio
INFO: [USF-XSim-8] Loading simulator feature
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_FSM_insert_ERRORS.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0 } {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#     send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window c
#   }
# }
# run 1000ns
Warning: Error: current_state=even => y = 0
Time: 80 ns Iteration: 0 Process: /tb_FSM_insert_ERRORS/assert_process File: D:/vhdl_ITI/fsm/fsm.srca/sim_1/new/tb_FSM_insert_ERRORS.vhd
Warning: Error: current_state=even => y = 0
Time: 120 ns Iteration: 0 Process: /tb_FSM_insert_ERRORS/assert_process File: D:/vhdl_ITI/fsm/fsm.srca/sim_1/new/tb_FSM_insert_ERRORS.vhd
Warning: Error: current_state=even => y = 0
Time: 160 ns Iteration: 0 Process: /tb_FSM_insert_ERRORS/assert_process File: D:/vhdl_ITI/fsm/fsm.srca/sim_1/new/tb_FSM_insert_ERRORS.vhd
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_FSM_insert_ERRORS_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:11 ; elapsed = 00:00:21 . Memory (MB): peak = 1789.504 ; gain = 6.023
```

Figure 13: console of FSM with Error message

- **Note:** warning messages appear in console which means test bench managed to detected errors.

3.4 Bin_2_gray:

a. function of Design:

-convert binary with width 4 bits to gray code.

-first the msb is the same then xor each bit with its previous bit.

As :-

- 0000 => 0000

- 0001 => 0001
- 0010 => 0011

b. Test strategy:

- **Steps:**

-using text file to read input and detected output and compare it with expected output of design if they are equal it will display "succeed message" if not it will display "Error message".

- **Cases :** the input is four bits so the possible inputs is 16 variable

| INPUT | OUTPUT |
|-------|--------|
| 0000 | 0000 |
| 0001 | 0001 |
| 0010 | 0011 |
| 0011 | 0010 |
| 0100 | 0110 |
| 0101 | 0111 |
| 0110 | 0101 |
| 0111 | 0100 |
| 1000 | 1100 |
| 1001 | 1101 |
| 1010 | 1101 |
| 1011 | 1110 |
| 1100 | 1010 |
| 1101 | 1011 |
| 1110 | 1001 |
| 1111 | 1000 |

- **File input:**

0000 20 ns 0000 succeed output

0001 20 ns 0001 succeed output

0010 20 ns 0011 succeed output

0011 20 ns 0010 succeed output

0100 20 ns 0110 succeed output

0101 20 ns 0111 succeed output

0110 20 ns 0101 succeed output

0111 20 ns 0100 succeed output

1000 20 ns 1100 succeed output

1001 20 ns 1101 succeed output

1010 20 ns 1111 succeed output

1011 20 ns 1110 succeed output

1100 20 ns 1010 succeed output

1101 20 ns 1011 succeed output

1110 20 ns 1001 succeed output

1111 20 ns 1000 succeed output

- **File output:**

At time 40 ns , input is : 0000, output is : 0000, the expected output is : 0000

At time 60 ns , input is : 0001, output is : 0001, the expected output is : 0001

At time 80 ns , input is : 0010, output is : 0011, the expected output is : 0011

At time 100 ns , input is : 0011, output is : 0010, the expected output is : 0010

At time 120 ns , input is : 0100, output is : 0110, the expected output is : 0110

At time 140 ns , input is : 0101, output is : 0111, the expected output is : 0111

At time 160 ns , input is : 0110, output is : 0101, the expected output is : 0101

At time 180 ns , input is : 0111, output is : 0100, the expected output is : 0100

At time 200 ns , input is : 1000, output is : 1100, the expected output is : 1100

At time 220 ns , input is : 1001, output is : 1101, the expected output is : 1101

At time 240 ns , input is : 1010, output is : 1111, the expected output is : 1111

At time 260 ns , input is : 1011, output is : 1110, the expected output is : 1110

At time 280 ns , input is : 1100, output is : 1010, the expected output is : 1010

At time 300 ns , input is : 1101, output is : 1011, the expected output is : 1011

At time 320 ns , input is : 1110, output is : 1001, the expected output is : 1001

At time 340 ns , input is : 1111, output is : 1000, the expected output is : 1000

- **Testbench:**

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE std.textio.all;

ENTITY tb_bin_2_gray_io_file IS
END tb_bin_2_gray_io_file;

ARCHITECTURE behavior OF tb_bin_2_gray_io_file IS

COMPONENT bin2gray

```

PORT(
    input : IN bit_vector(3 downto 0);
    output : OUT bit_vector(3 downto 0)
);

END COMPONENT;

    signal input_s: bit_vector (3 downto 0);
    signal output_s: bit_vector (3 downto 0);
begin
uut:bin2gray port map(input=>input_s, output=>output_s);

read_from_file:process

file input_bit_gray : text open read_mode is
("D:\vhdl_ITI\binary_2_gray\input_bit_2_gray_insert_errors.txt");
file output_bit_gray : text open write_mode is
("D:\vhdl_ITI\binary_2_gray\output_bit_2_gray__insert_errors.txt");
variable in_1 ,out_1 : line;
variable input_sim : bit_vector (3 downto 0);
variable pause:time;
variable output_sim : bit_vector (3 downto 0);
variable message : string (1 TO 45);

begin
input_s <= "0000";
WAIT FOR 20 ns;

while not endfile (input_bit_gray) loop
readline(input_bit_gray,in_1);
read(in_1,input_sim);
read(in_1,pause);
read(in_1,output_sim);
read(in_1,message);
input_s<=input_sim;
--output_s<=output_sim;
wait for pause;
write (out_1,string' ("At time "));
write(out_1,NOW);
write(out_1,string' (" , input is : "));
write (out_1,input_s);
write(out_1,string' (" , output is : "));
write (out_1,output_sim);
write(out_1,string' (" , the expected output is : "));
write (out_1,output_s);
if (output_s /= output_sim)then
write(out_1,string' (" , ERROR :unexpected output "));

```

```

else
write(out_l,message);
end if;
WRTIELINE (output_bit_gray, out_l);
END loop;
WAIT;

end process;
end ARCHITECTURE behavior;

```

c. Simulation results:



Figure14: simulation of binary to gray circuit

d. synthesis and schematic:

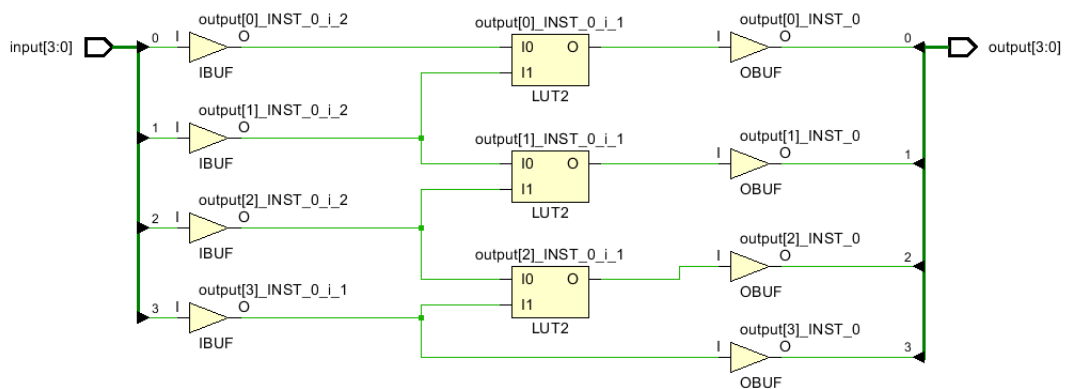


Figure 15: schematic of binary to gray

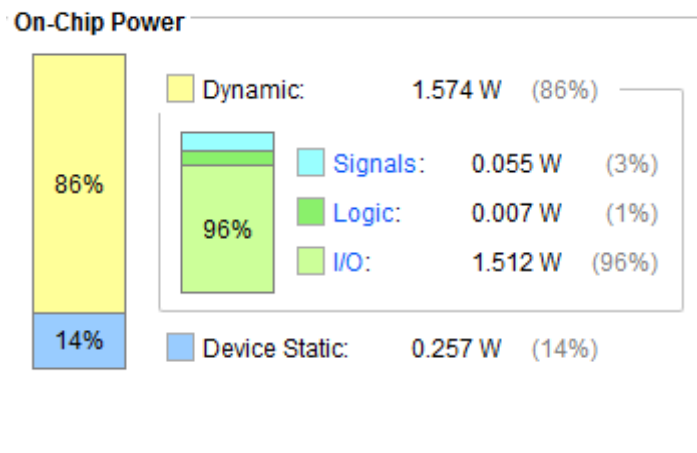


Figure16: power of binary to gray circuit on device "xa7s6cpga 196-2I"

e. Inserting Error:

- **File input:**

0000 20 ns 0000 succeed output
0001 20 ns 0001 succeed output
0010 20 ns 0011 succeed output
1011 20 ns 0010 succeed output
0100 20 ns 0110 succeed output
0101 20 ns 0111 succeed output
0111 20 ns 0101 succeed output
0111 20 ns 0100 succeed output
1001 20 ns 1100 succeed output
1001 20 ns 1101 succeed output
1010 20 ns 1111 succeed output
1011 20 ns 1110 succeed output
1100 20 ns 1010 succeed output
1101 20 ns 1011 succeed output
1110 20 ns 1001 succeed output
1111 20 ns 1000 succeed output

- **File output:**

At time 40 ns , input is : 0000, output is : 0000, the expected output is : 0000

At time 60 ns , input is : 0001, output is : 0001, the expected output is : 0001

At time 80 ns , input is : 0010, output is : 0011, the expected output is : 0011

At time 100 ns , input is : 1011, output is : 0010, the expected output is : 1110 ,
ERROR :unexpected output

At time 120 ns , input is : 0100, output is : 0110, the expected output is : 0110

At time 140 ns , input is : 0101, output is : 0111, the expected output is : 0111

At time 160 ns , input is : 0111, output is : 0101, the expected output is : 0100 ,
ERROR :unexpected output

At time 180 ns , input is : 0111, output is : 0100, the expected output is : 0100

At time 200 ns , input is : 1001, output is : 1100, the expected output is : 1101 ,
ERROR :unexpected output

At time 220 ns , input is : 1001, output is : 1101, the expected output is : 1101

At time 240 ns , input is : 1010, output is : 1111, the expected output is : 1111

At time 260 ns , input is : 1011, output is : 1110, the expected output is : 1110

At time 280 ns , input is : 1100, output is : 1010, the expected output is : 1010

At time 300 ns , input is : 1101, output is : 1011, the expected output is : 1011

At time 320 ns , input is : 1110, output is : 1001, the expected output is : 1001

At time 340 ns , input is : 1111, output is : 1000, the expected output is : 1000

- **Note:**

-when insert wrong inputs and mapped it to output in input file appears in output file the error message.

3.5 D_ff_Aycr:

a. function of Design:

-asynchronous register works with rising edge of clock but when set signal =1 doesn't respect the clock and output =1 then when clear signal =1 doesn't respect the clock and output =0 but clear signal has less priority than set.

b. Test strategy:

- **Steps:**

- inputs are clear,set and d_in so it should be covered all cases
- First, create process to generate clock with duty cycle 50%.
- Then, set all possible cases for input to trace output by create processes of clear ,set and d_in ,making them changes with time to detect the output.

| set | clear | D_in | D_out | comments |
|-----|-------|------|-------|------------------------------------------------------------------------------------------------------------|
| 1 | x | x | 1 | When set =1, it has the highest priority so the other inputs are don't care(d_out=1). |
| 0 | 1 | x | 0 | When set =0 and clear =1, it has the second highest priority so the other inputs are don't care(d_out=0). |
| 0 | 0 | 1 | 1 | When set =0 and clear =0,the d_out=d_in at rising edge of clk and don't change at the failing edge |
| 0 | 0 | 0 | 0 | When set =0 and clear =0,the d_out=d_in at rising edge of clk and don't change at the failing edge |
| 0 | 0 | 1 | 0 | When set =0 and clear =0,the d_out=d_in at rising edge of clk and don't change at the failing edge |
| 0 | 1 | 1 | 0 | When set =0 and clear =1, it has the second highest priority so the other inputs are don't care (d_out=0). |
| 1 | 0 | 0 | 1 | When set =1, it has the highest priority so the other inputs are don't care(d_out=1). |
| 1 | 1 | 0 | 1 | When set =1, it has the highest priority so the other inputs are don't care(d_out=1). |

- **Testbench:**

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity tb_ff_async is

-- Port ();

end tb_ff_async;

architecture Behavioral_tb_ff_async of tb_ff_async is

component d_ff_async

PORT(clk, set, clear: IN std_logic;

d_in : IN std_logic;

d_out: OUT std_logic);

END component;

signal clk, set, clear:std_logic;

signal d_in : std_logic;

signal d_out: std_logic;

begin

ff1:d_ff_async port map (clk=>clk, set=>set, clear=>clear, d_in => d_in , d_out => d_out);

process_clk:process


```

begin
clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
end process;
create_rst_process:process
begin
    clear <= '1';
    -- wait for 320 ns;
wait for 40 ns;
    clear <= '0';
    wait for 40 ns ;
end process;
process_set:process
begin
    set <= '1';
    -- wait for 320 ns;
wait for 30 ns;
    set <= '0';
    wait for 30 ns ;
end process;
process_d_in:process
begin
d_in<='0';
    wait for 25 ns ;
    d_in<='1';
    wait for 25 ns ;

end process;

```

c. Simulation results

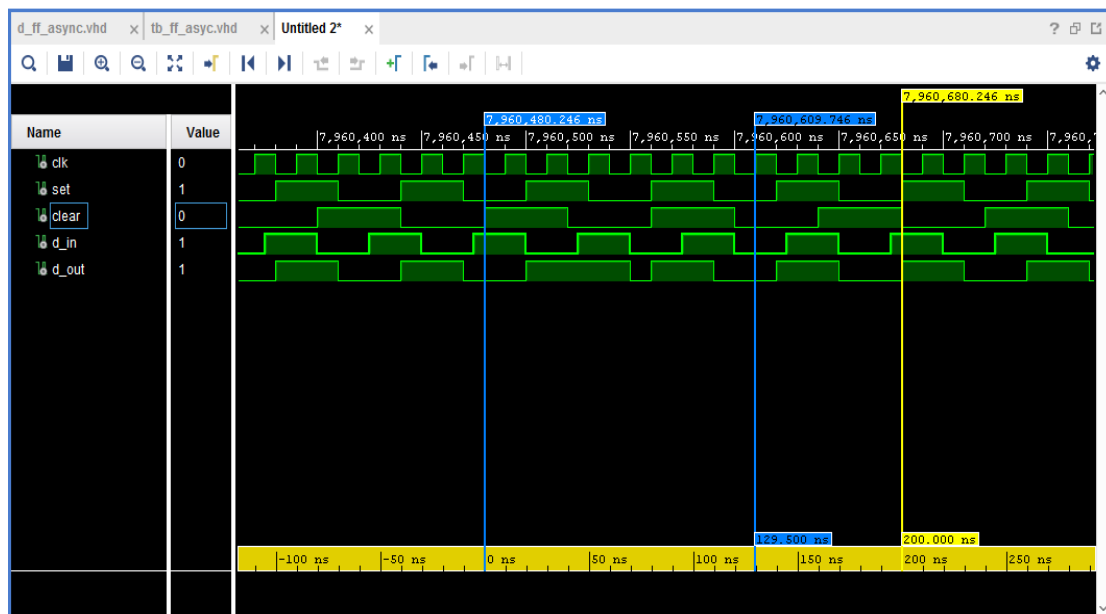


Figure17: sinulation of d_ff_aycr

d. synthesis and schematic:

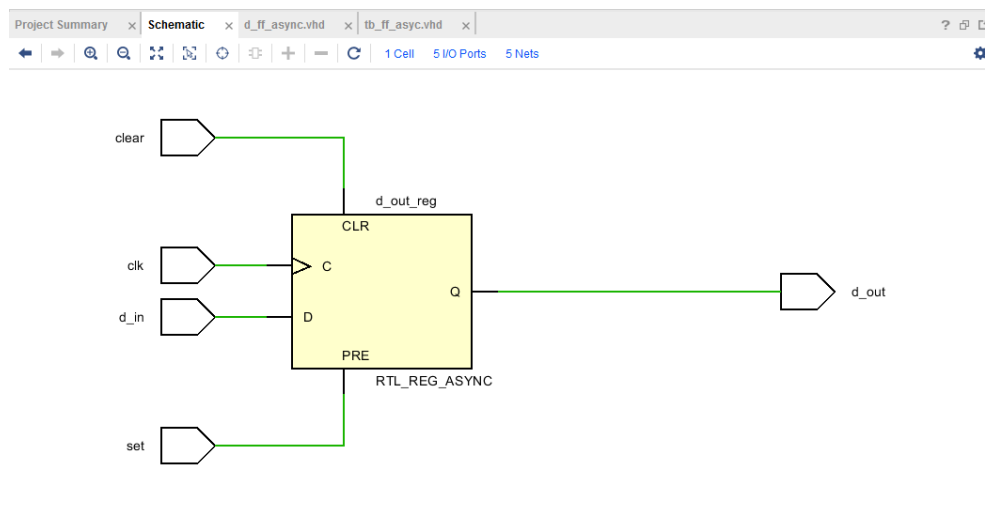


Figure18: schematic of d_ff_aycr

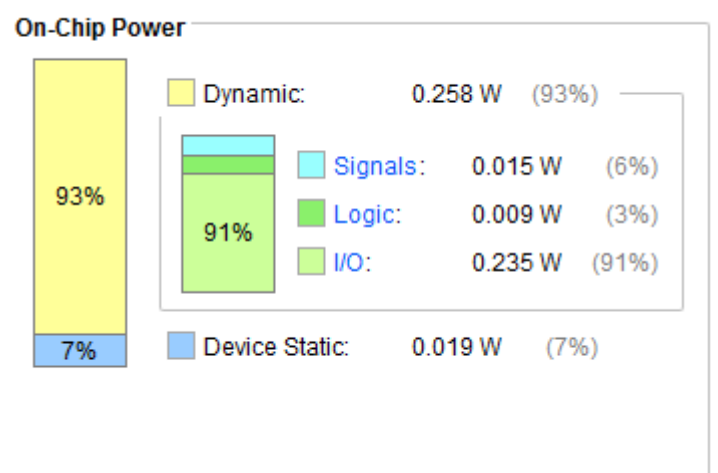


Figure 19:power of d_ff_aycr on device "xa7s6cpga 196-2I"