

VHDL i

**Submitted by:**

Menna Tallah Mater Abdelhamed

**Track:**

Digital IC

**Submitted to:**

Dr.Watheq

Contents

[**1. Introduction:** 2](#_Toc59812875)

[**2. Test bench methodology:** 2](#_Toc59812876)

[**3. Examples:** 3](#_Toc59812877)

[**3.1.1 Counter:** 3](#_Toc59812878)

[a. function of Design: 3](#_Toc59812879)

[b. Test strategy: 3](#_Toc59812880)

[c. Simulation results: 4](#_Toc59812881)

[c. synthesis and schematic: 4](#_Toc59812882)

[3.1.2 Updated Counter 4](#_Toc59812883)

[**3.2 Increment:** 6](#_Toc59812884)

[a. function of Design: 6](#_Toc59812885)

[b. Test strategy: 6](#_Toc59812886)

[c. Simulation results: 8](#_Toc59812887)

[d. synthesis and schematic: 9](#_Toc59812888)

[**3.3 FSM\_moore\_2p:** 9](#_Toc59812889)

[a. function of Design: 9](#_Toc59812890)

[b. Test strategy: 9](#_Toc59812891)

[c. Simulation results: 12](#_Toc59812892)

[d. synthesis and schematic 13](#_Toc59812893)

[e. Inserting Error: 13](#_Toc59812894)

[**3.4 Bin\_2\_gray**: 16](#_Toc59812895)

[a. function of Design: 16](#_Toc59812896)

[b. Test strategy: 17](#_Toc59812897)

[c. Simulation results: 20](#_Toc59812898)

[d. synthesis and schematic 20](#_Toc59812899)

[e. Inserting Error: 21](#_Toc59812900)

[**3.5 D\_ff\_Aycr**: 22](#_Toc59812901)

[a. function of Design: 22](#_Toc59812902)

[b. Test strategy: 22](#_Toc59812903)

[c. Simulation results 25](#_Toc59812904)

[d. synthesis and schematic 25](#_Toc59812905)

# **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:

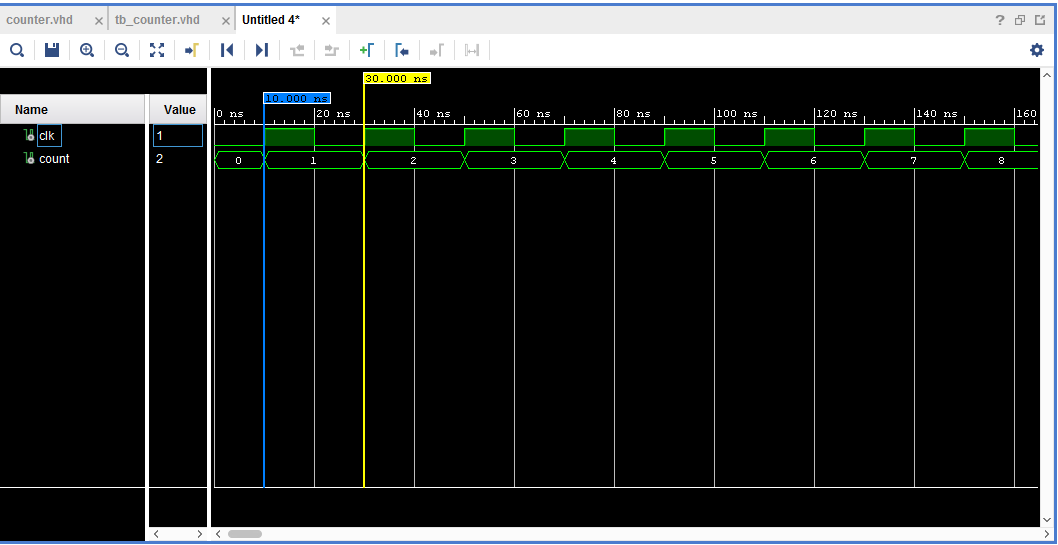


Figure :simulation of counter

### c. synthesis and schematic:

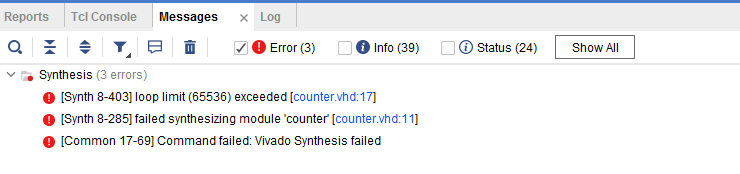
-can't synthesis this circuit because it contain loop (in time loop).

Figure :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.

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

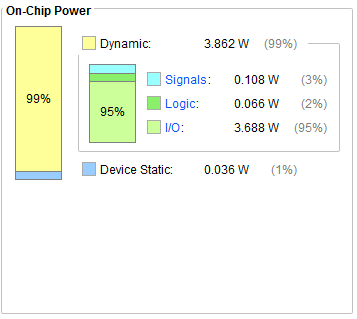
#### b. Simulation:

#### :

Figure3:simulation of updated counter

#### c. synthesis and schematic :

Figure4: schematic of updated counter

Figure5:power on device "xa7s6cpga 196-2l"

## **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:

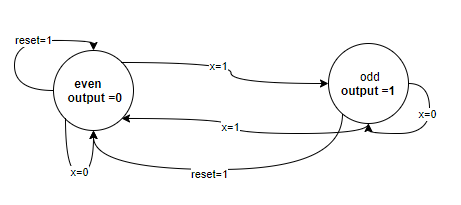
-translate from state to other according inputs.

Figure8:digram 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=1returns 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;

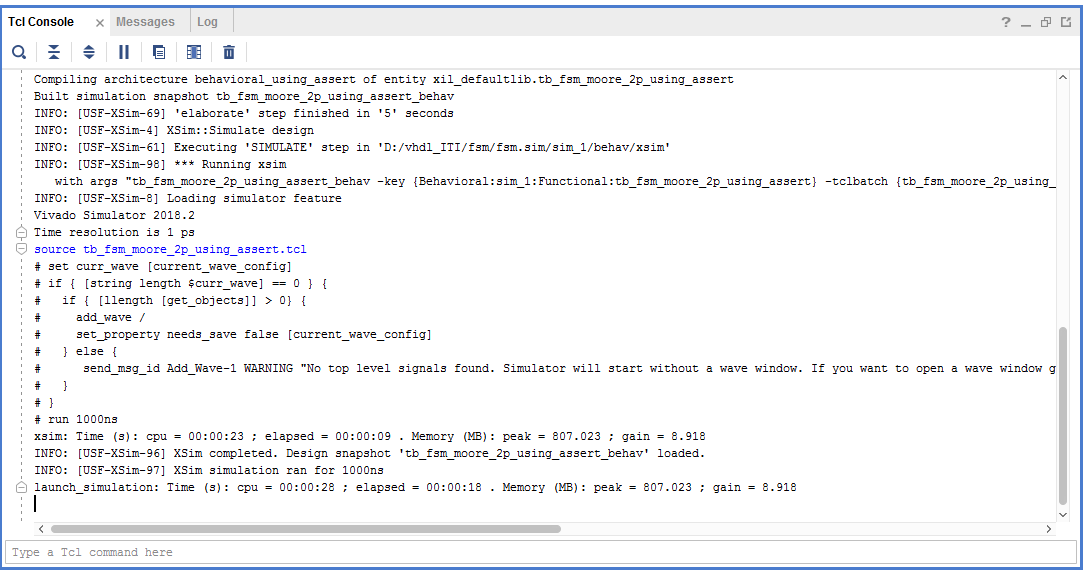


Figure9:Tcl console

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

### c. Simulation results:

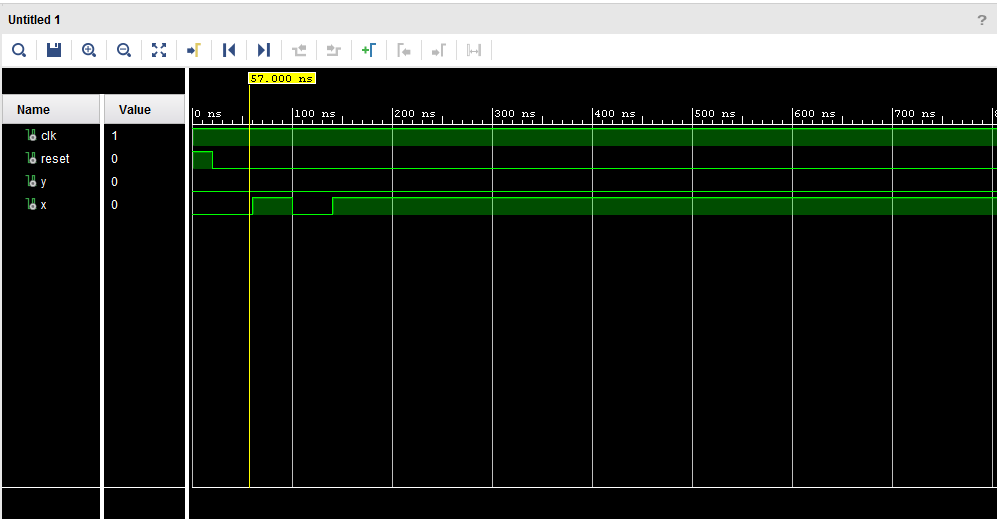
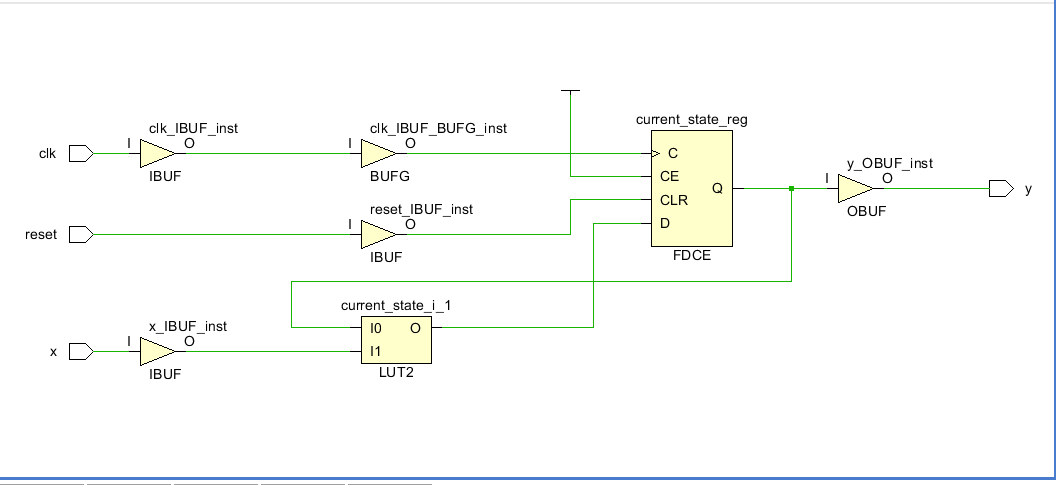


Figure10:Simulation of FSM

d. synthesis and schematic:

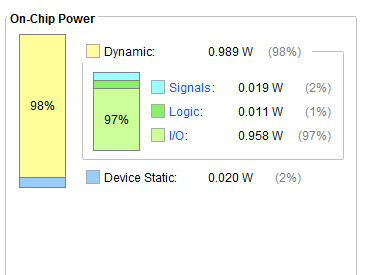
 Figure11:schematic of FSM ,it repesent in register and combinational logic

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

### 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;

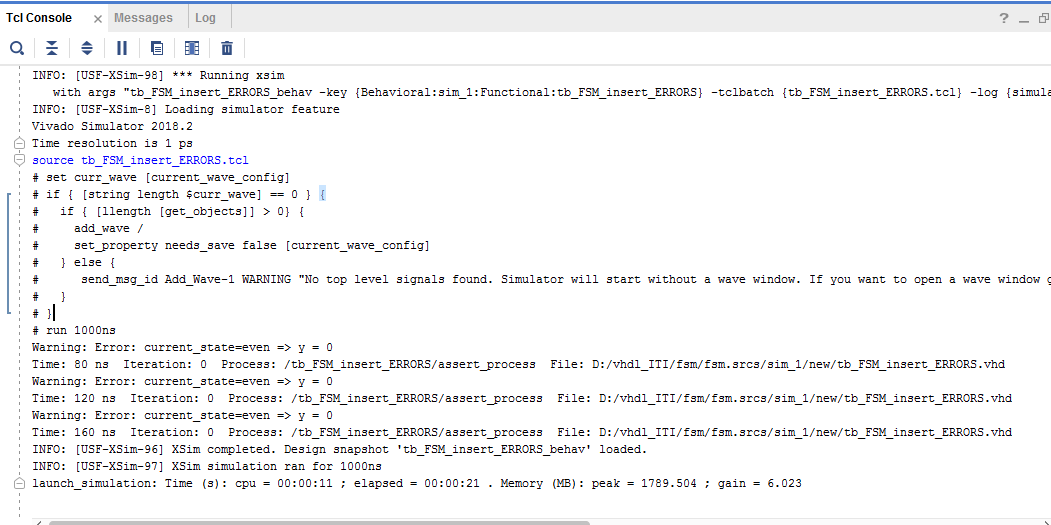


Figure 13: console of FSM wih Errpr message

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

## 3.4 Bin\_2\_gray:

### a. function of Design:

-convert binay 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\_l ,out\_l : 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\_l);

read(in\_l,input\_sim);

read(in\_l,pause);

read(in\_l,output\_sim);

read(in\_l,message);

input\_s<=input\_sim;

--output\_s<=output\_sim;

wait for pause;

write (out\_l,string' ("At time "));

write(out\_l,NOW);

write(out\_l,string' (" , input is : "));

write (out\_l,input\_s);

write(out\_l,string' (", output is : "));

write (out\_l,output\_sim);

write(out\_l,string' (", the expected output is : "));

write (out\_l,output\_s);

if (output\_s /= output\_sim)then

write(out\_l,string' (" , ERROR :unexpected output "));

else

write(out\_l,message);

end if;

WRITELINE (output\_bit\_gray, out\_l);

END loop;

WAIT;

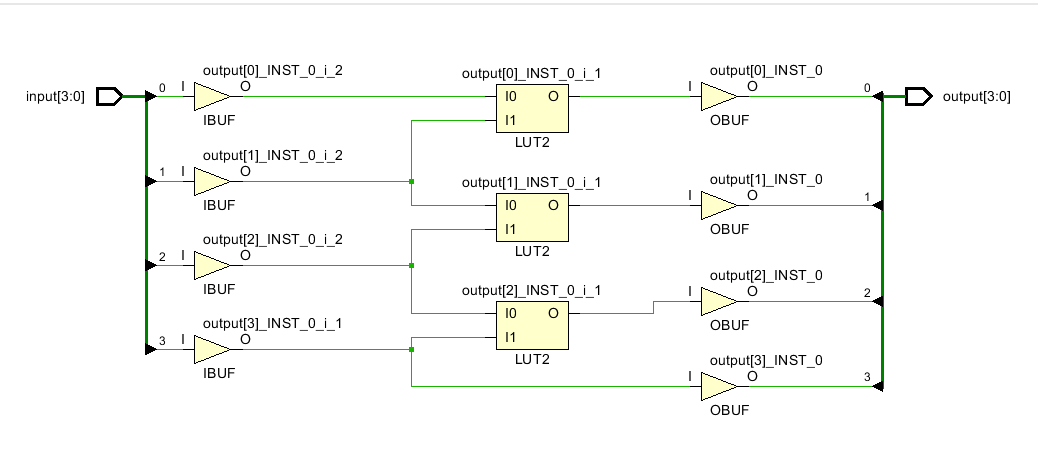
end process;

end ARCHITECTURE behavior;

### c. Simulation results:

Figure14: simulation of binary to gray circuit

d. synthesis and schematic:



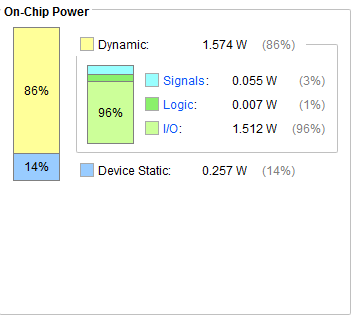
Figure 15: schematic of binary to gray

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

### 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 shoud 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\_asyc is

-- Port ( );

end tb\_ff\_asyc;

architecture Behavioral\_tb\_ff\_asyc of tb\_ff\_asyc 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;

end Behavioral\_tb\_ff\_asyc**;**

### c. Simulation results

Figure17: sinulation of d\_ff\_aycr

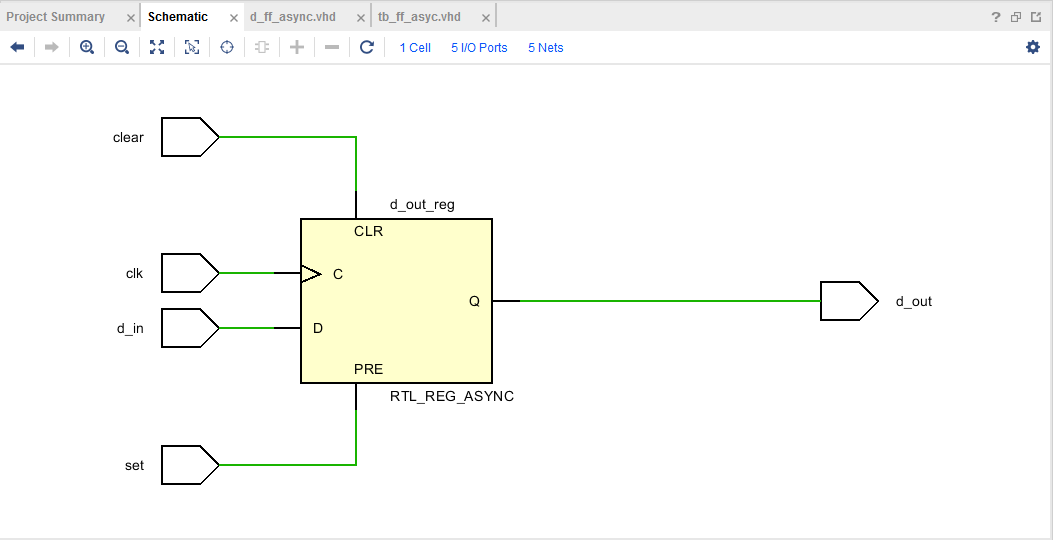
d. synthesis and schematic:

Figure18: schematic of d\_ff\_aycr

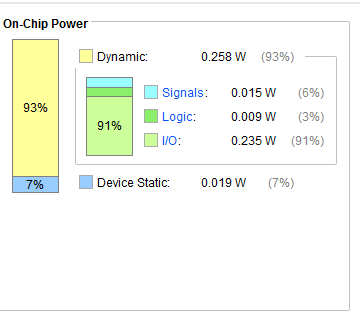


Figure 19:power of d\_ff\_aycr on device "xa7s6cpga 196-2l"