

Bab 2 – Isim Simulator

Tujuan

- Mahasiswa dapat mendisain rangkaian dari persamaan Boolean.
- Mahasiswa dapat membuat RTL dari rangkaian logika.
- Mahasiswa dapat membuat simulasi testbench dari rangkaian logika.

Teori

Ketika sebuah rancangan rangkaian diimplementasikan ke dalam FPGA, baik menggunakan metode schematic atau metode HDL, hendaknya perlu diuji agar kita dapat mengetahui hasil/output dari rangkaian tersebut. Proses pengujian rancangan rangkaian ini sering disebut proses simulasi. Melalui proses simulasi, kita dapat mengetahui apakah hasil rancangan rangkaian yang sudah dibuat sudah sesuai dan memenuhi tujuan yang diinginkan atau belum. Proses ini biasanya dilakukan sebelum rancangan rangkaian diimplementasikan ke dalam FPGA. Jadi urutan sebagai berikut:

1. Pembuatan rancangan rangkaian (Metode *schematic* atau metode HDL).
2. Proses simulasi rancangan rangkaian.
3. Implementasi ke dalam FPGA.

Nantinya, proses simulasi berfungsi untuk mengetahui sekaligus menguji apakah rancangan rangkaian yang telah dibuat mampu berjalan dengan baik atau tidak. Selain itu, lewat proses simulasi dapat diketahui bagaimana output dari rancangan rangkaian tadi. Selanjutnya, proses simulasi membutuhkan suatu bentuk stimulus/pemicu. Stimulus ini akan bertindak sebagai input awal bagi rancangan rangkaian yang hendak diuji. Kemudian, setelah diberikan stimulus maka rancangan rangkaian tersebut dapat diketahui bagaimana hasil outputnya. Keseluruhan proses simulasi ini dilakukan dengan bantuan perangkat lunak (software) yang ada.

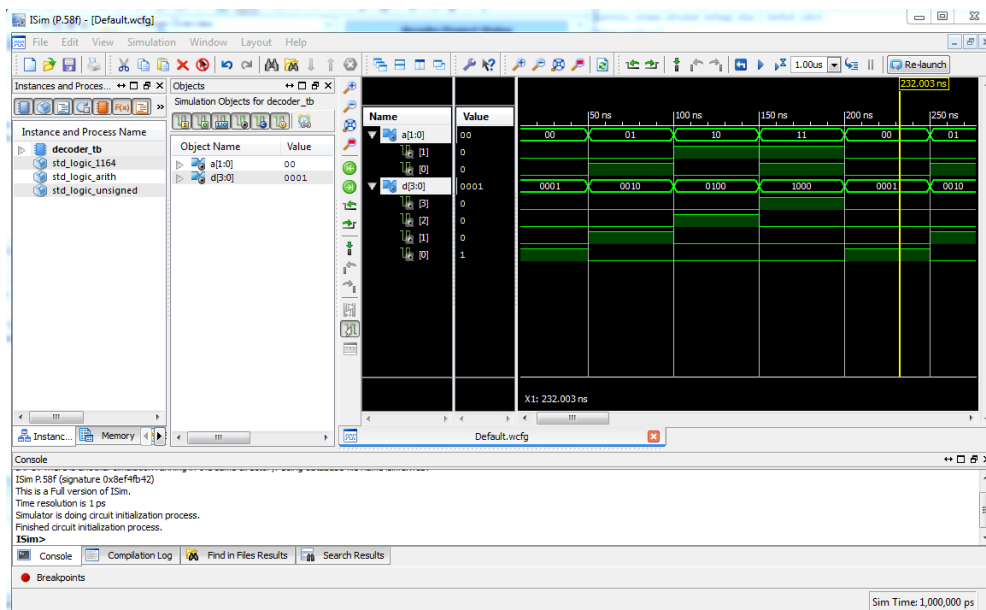
Pada umumnya, proses simulasi terbagi atas 2 bentuk yakni:

- Bentuk *testbench*.
- Bentuk *timing diagram*.

Timing Diagram

Proses simulasi dapat diketahui dengan membuat gambar timing diagram. Dengan menggunakan gambar timing diagram, dapat dilihat mengenai kondisi input, output serta hal-hal lain yang terkait. Bentuk *timing diagram* ini dibuat berdasarkan satuan waktu. Pada mulanya, untuk membuat timing diagram ini diperlukan stimulus untuk menentukan kondisi input awal. Selanjutnya, software simulator akan melakukan simulasi guna menghasilkan kondisi output sesuai dengan kondisi input tadi.

Cara ini merupakan cara yang sederhana dan mudah dilakukan, apalagi bagi para pengguna awal FPGA, bila dibandingkan dengan bentuk testbench. Hal ini disebabkan karena tampilan timing diagram yang berupa gambar sehingga memudahkan siapa saja untuk mengamati, menganalisa, dan menjelaskan proses simulasi.

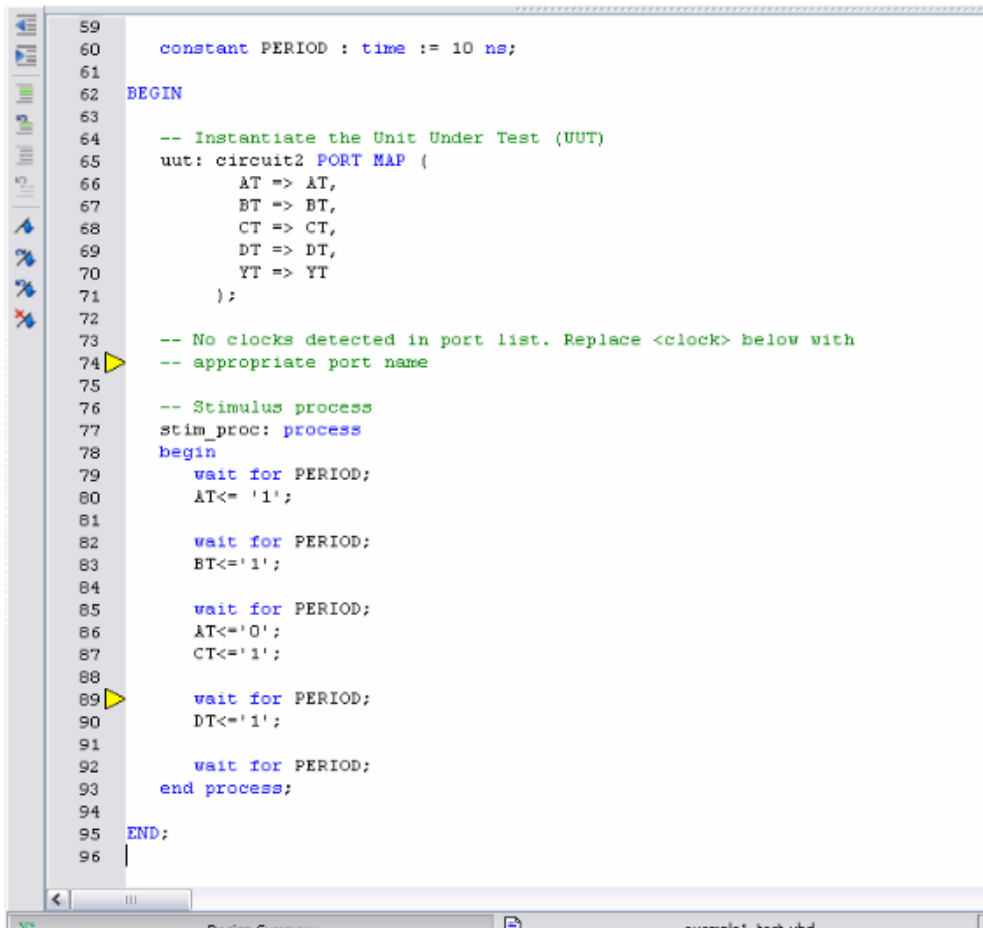


Gambar 1. 1 Isim Simulator

Testbench

Selain cara pertama, dikenal pula bentuk simulasi yang lain yaitu *testbench*. **Testbench** adalah proses pengujian suatu rancangan rangkaian. Dalam proses perancangan rangkaian, testbench akan menguji design rangkaian apakah sudah sesuai atau belum. Testbench ini dilakukan dengan menggunakan file

HDL (berbentuk kode, baik VHDL maupun Verilog). Sehingga berbeda dengan bentuk diagram yang menggunakan gambar sebagai tampilannya, testbench menggunakan kode-kode tulisan sebagai tampilannya. Berikut contoh testbench :



```
59
60     constant PERIOD : time := 10 ns;
61
62 BEGIN
63
64     -- Instantiate the Unit Under Test (UUT)
65     uut: circuit2 PORT MAP (
66         AT => AT,
67         BT => BT,
68         CT => CT,
69         DT => DT,
70         YT => YT
71     );
72
73     -- No clocks detected in port list. Replace <clock> below with
74     -- appropriate port name
75
76     -- Stimulus process
77     stim_proc: process
78     begin
79         wait for PERIOD;
80         AT<= '1';
81
82         wait for PERIOD;
83         BT<='1';
84
85         wait for PERIOD;
86         AT<='0';
87         CT<='1';
88
89         wait for PERIOD;
90         DT<='1';
91
92         wait for PERIOD;
93     end process;
94
95 END;
```

Gambar 1. 2 Test Bench

Alat dan Bahan

1. PC yang sudah terinstal ISE 13.1
2. Xilinx Spartan 3
3. Downloader JTAG USB

Percobaan

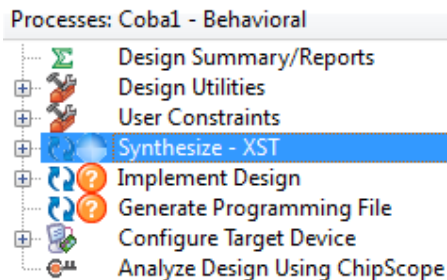
A. RTL dan Simulasi Test Bench

1. Sebelum melakukan test bench, buatlah sebuah program .vhdl misalnya membuat rangkaian logika AND.

```
Library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
  
entity Cobal is  
    port(  
        a :in STD_LOGIC;  
        b :in STD_LOGIC;  
        y :out STD_LOGIC  
    );  
end Cobal;  
  
architecture Behavioral of Cobal is  
begin  
    y <= a and b;  
end Behavioral;
```

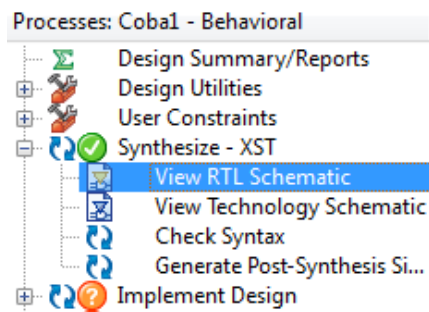
Script 2. 1 Rangkaian Logika AND

2. Kemudian lakukan Synthesize-XST.



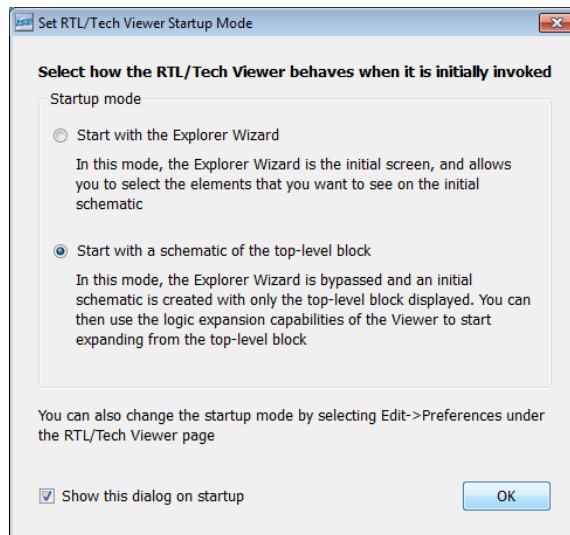
Gambar 2. 1 Proses Synthesize-XST

3. Klik 2x pada View RTL Simulation,



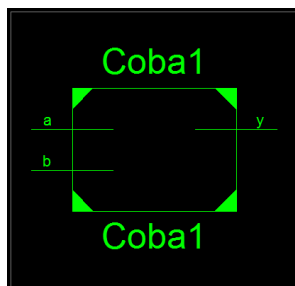
Gambar 2. 2 View RTL Simulation

4. kemudian pilih *Start with a schematic of the top-level block* pada window Set RTL/Tech Viewer Startup Mode kemudian OK.



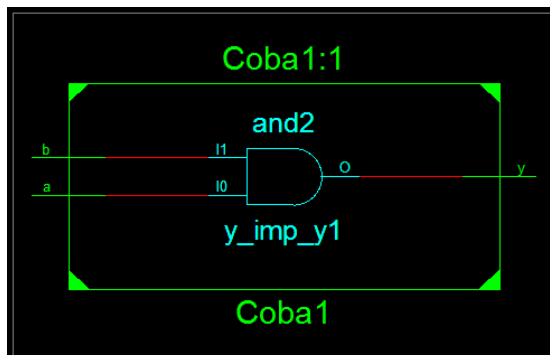
Gambar 2. 3 Set RTL/Tech Viewer Startup Mode

5. Akan muncul RTL skematik yaitu rangkaian yang sudah kita buat.



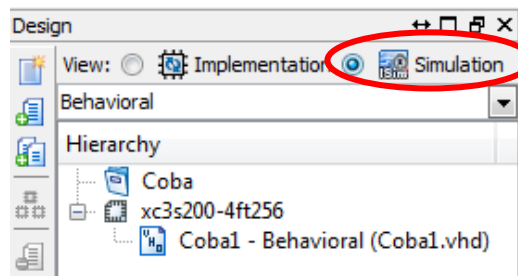
Gambar 2. 4 2.6 RTL Skematik

6. Klik 2x di dalam diagram block, maka akan muncul rangkaian logikanya



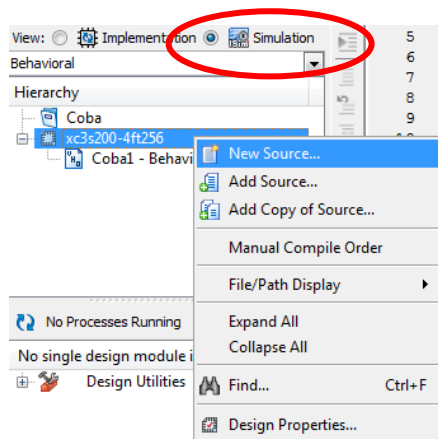
Gambar 2. 5 Rangkaian Logika AND

7. Kembali ke kode program, jika tidak terdapat error, pilih **Simulation** pada **View** di window kolom **Design**



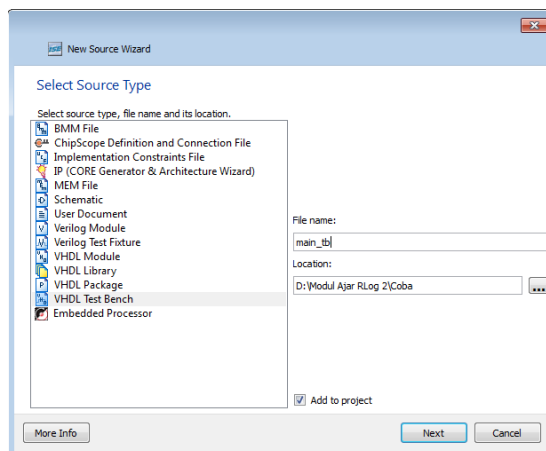
Gambar 2. 6 Simulation View

8. Kemudian buat project baru dengan memilih- New Source



Gambar 2. 7 Membuat New Source

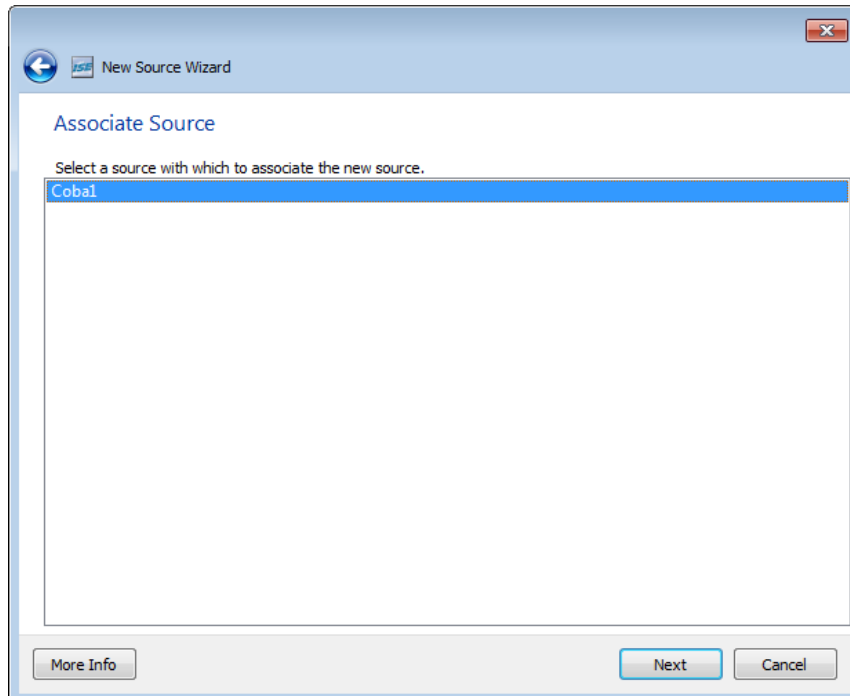
9. Pilih **VHDL Test bench** dan isi **file name**-nya dengan nama **main_tb**, kemudian klik Next.



Gambar 2. 8 VHDL Test Bench

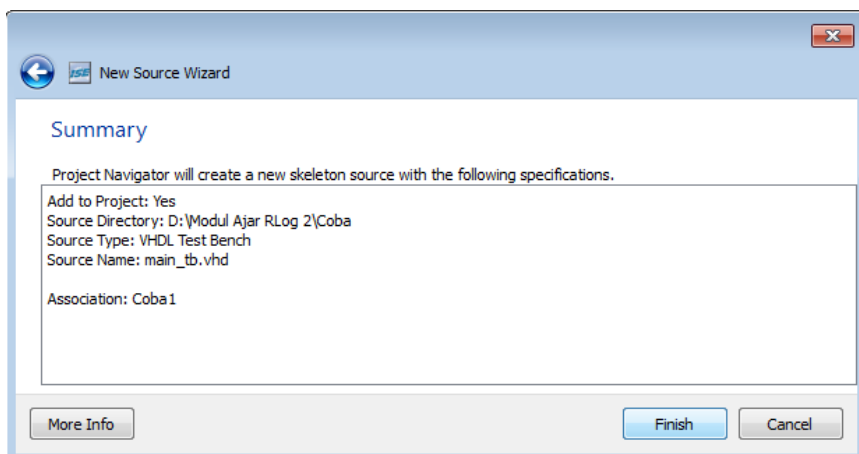
10. Bagian ini menunjukkan tentang file yang akan disimulasikan. Kemudian pilih Next.

(Pada gambar 2.10 file yang akan disimulasikan adalah program untuk membuat rangkaian logika “AND” dan nama filenya adalah Coba1)



Gambar 2. 9 VHDL Test Bench

11. Kemudian pilih finish.



Gambar 2. 10 Finish Wizard

12. Setelah itu akan muncul kode dari main_tb.

```
35 ENTITY main_tb IS
36 END main_tb;
37
38 ARCHITECTURE behavior OF main_tb IS
39
40     -- Component Declaration for the Unit Under Test (UUT)
41
42     COMPONENT cobal
43     PORT(
44         a : IN  std_logic;
45         b : IN  std_logic;
46         y : OUT std_logic
47     );
48     END COMPONENT;
49
50
51     --Inputs
52     signal a : std_logic := '0';
53     signal b : std_logic := '0';
54
55     --Outputs
56     signal y : std_logic;
57     -- No clocks detected in port list. Replace <clock> below with
58     -- appropriate port name
59
60     constant <clock>_period : time := 10 ns;
```

```
61
62 BEGIN
63
64     -- Instantiate the Unit Under Test (UUT)
65     uut: cobal PORT MAP (
66         a => a,
67         b => b,
68         y => y
69     );
70
71     -- Clock process definitions
72     <clock>_process :process
73     begin
74         <clock> <= '0';
75         wait for <clock>_period/2;
76         <clock> <= '1';
77         wait for <clock>_period/2;
78     end process;
79
```



```

79
80
81     -- Stimulus process
82     stim_proc: process
83     begin
84         -- hold reset state for 100 ns.
85         | wait for 100 ns;
86
87     wait for <clock>_period*10;
88
89         -- insert stimulus here
90
91         wait;
92     end process;
93
94 END;

```

Gambar 2. 11 Kode Program main_tb

13. Pada source code uut-main_src, comment baris program yang dilingkari warna merah.

```

59
60 -- constant <clock>_period : time := 10 ns;
61 BEGIN
62     -- Instantiate the Unit Under Test (UUT)
63     uut: cobal PORT MAP (
64         a => a,
65         b => b,
66         y => y
67     );
68     -- Clock process definitions
69     -- <clock>_process :process
70     begin
71         <clock> <= '0';
72         wait for <clock>_period/2;
73         <clock> <= '1';
74         wait for <clock>_period/2;
75     end process;
76
77
78     -- Stimulus process
79     stim_proc: process
80     begin
81         -- hold reset state for 100 ns.
82         wait for 100 ns;
83
84     -- wait for <clock>_period*10;
85

```

Gambar 2. 12 Comment baris program

14. Pada bagian **stim_proc : process**, ubahlah baris kodenya menjadi seperti skrip 2.2.

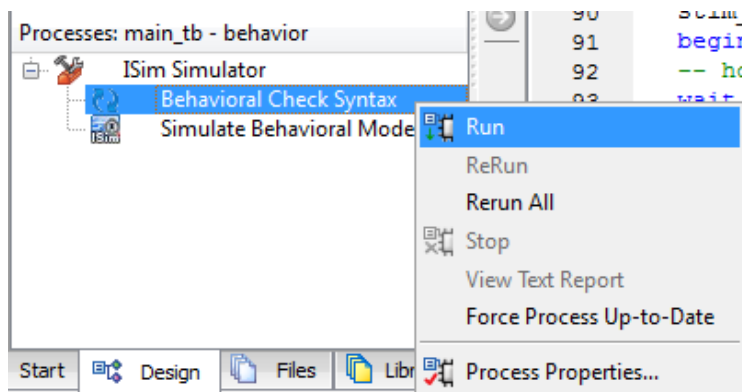
```
-- Stimulus process
stim_proc:process
begin
-- hold reset state for 100 ns.
Wait for 100 ns;
a <=not a;
end process;

-- Stimulus process
stim_procl:process
begin
-- hold reset state for 100 ns.
Wait for 50 ns;
b <=not b;
end process;
```

Script 2. 2 Membagi periode waktu

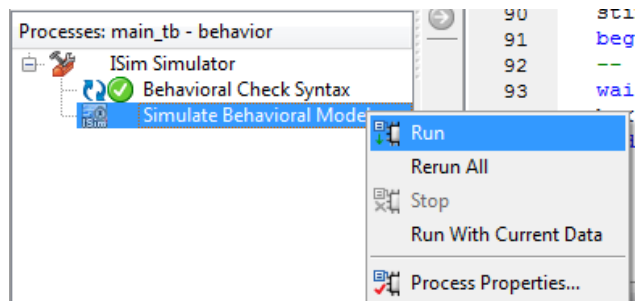
*“Catatan : Dalam kode diatas, maksudnya adalah waktu sinyal **a** akan high selama 100ns dan low selama 100ns, demikian juga pada signal **b** akan high selama 50ns dan low selama 50ns. “*

15. Kemudian Cek syntax, dengan klik kanan pada **Behavioral Check Syntax-Run**.



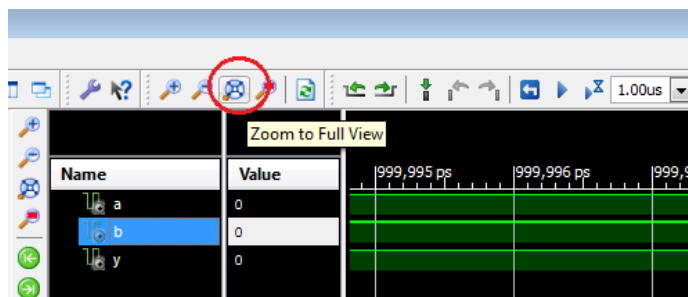
Gambar 2. 1312 Cek syntax simulasi

16. Kemudian klik kanan **Simulate Behavioral Model-Run** untuk menampilkan simulasi.



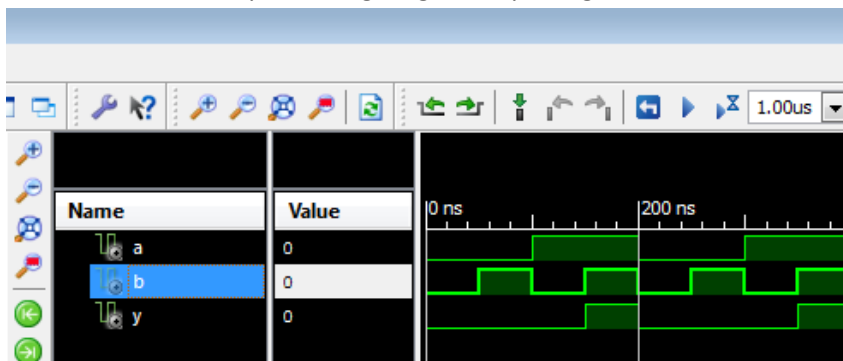
Gambar 2. 1413 Menampilkan Simulasi ISim

17. Pilih Zoom to full view untuk melihat secara detail.



Gambar 2. 14 Zoom to full view

18. Kemudian akan tampak timing diagram seperti gambar 2.16



Gambar 2. 15 Timing Diagram rangkaian logika AND

“Pada timing diagram gambar 2.16 hanya ketika sinyal a dan b berlogika high, sinyal output akan mengeluarkan logika high ($C=A.B$) “

B. Project 2

1. Buatlah project baru untuk disain decoder 2 to 4. Lakukan seperti langkah sebelumnya. Berikut program decoder 2 to 4.

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

entity decoder is
    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
          d : out  STD_LOGIC_VECTOR (3 downto 0)
        );
end decoder;

architecture Behavioral of decoder is

begin
    with a select
        d(3 downto 0) <=  "0001" when "00",
                           "0010" when "01",
                           "0100" when "10",
                           "1000" when "11",
                           "0000" when others;
end Behavioral;
```

2. Berikut program testbench.

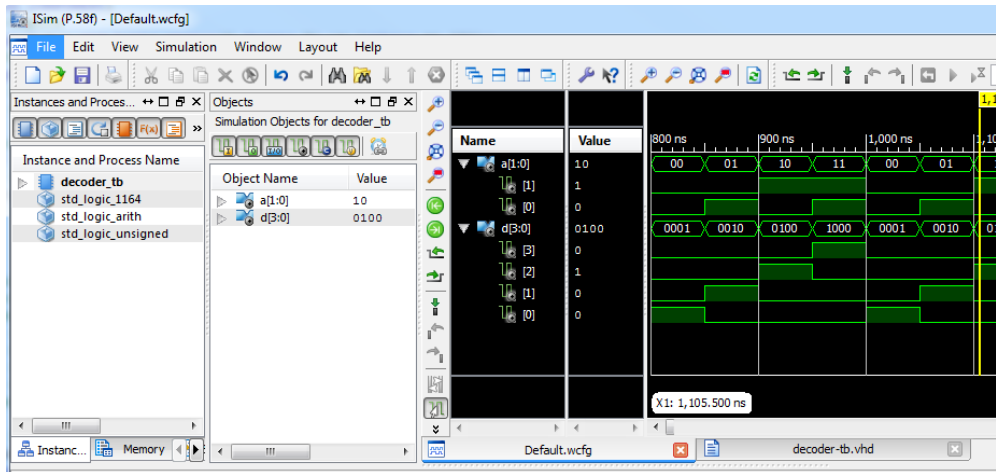
```
BEGIN
-- Instantiate the Unit Under Test (UUT)
ut: decoder PORT MAP (
    a => a,
    d => d
);

-- Clock process definitions
-- <clock>_process :process
-- begin
--     <clock> <= '0';
--     wait for <clock>_period/2;
--     <clock> <= '1';
--     wait for <clock>_period/2;
-- end process;

-- Stimulus process
stim_proc: process
begin
    wait for 50 ns;
    a(0) <= not a(0);
end process;

stim_proc1: process
begin
    wait for 100 ns;
    a(1) <= not a(1);
end process;
END;
```

3. Dan hasil dari simulasinya ditampilkan seperti gambar dibawah ini.



- ***Laporan sementara berisi screen shoot hasil simulasi kedua program diatas**

Tugas

Buatlah :

1. Tabel kebenaran
2. K-map
3. Persamaan Boolean
4. Program VHDL
5. Tampilkan RTL Schematics
6. Timing diagram

Dari rangkaian berikut ini

- a. Biner to Gray code 4 bit.
- b. Biner to code 2421

Referensi

1. Haskell, Richard and Darrin, *Learning By Example Using VHDL Advanced Digital Design With a NEXYS 2TM FPGA Board*, 2009, Oakland university Rochester, Michigan, ISBN 978-0-9801337-4-5.
2. <http://ndoware.com/simulasi-fpga.html>