

3ο Εργαστήριο

Σχεδίαση Ψηφιακών Συστημάτων

Παράδειγμα Χρήσης Vivado

Βασιλόπουλος Διονύσης

Ε.Δι.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών - ΕΚΠΑ

VHDL – Παράδειγμα

Άσκηση

Σχεδιάστε ένα ψηφιακό κύκλωμα που θα δέχεται 2 εισόδους `data1_in` και `data2_in` των 2 bit η κάθε μία και θα έχει δύο εξόδους `data1_out` του ενός (1) bit και `data2_out` των 2 bit. Η έξοδος `data1_out` θα είναι '1' όταν το πλήθος των bit που είναι '1' και των δύο εισόδων είναι άρτιο, αλλιώς θα είναι '0'. Η έξοδος `data2_out` θα έχει το λογικό xor των δύο εισόδων. Δημιουργήστε ένα νέο project με το όνομα `DSD_Lab3`, με όνομα οντότητας σχεδίασης `DSD_Lab3`, όνομα οντότητας simulation `DSD_Lab3_tb`. Οι αντίστοιχες αρχιτεκτονικές θα λέγονται `DF` και `DF_tb`.

Σχεδιάστε και υλοποιήστε το λογικό κύκλωμα που περιγράφει το ανωτέρω πρόβλημα.

Σχεδιάστε το κύκλωμα και με αρχιτεκτονικές Behavioral και Structural

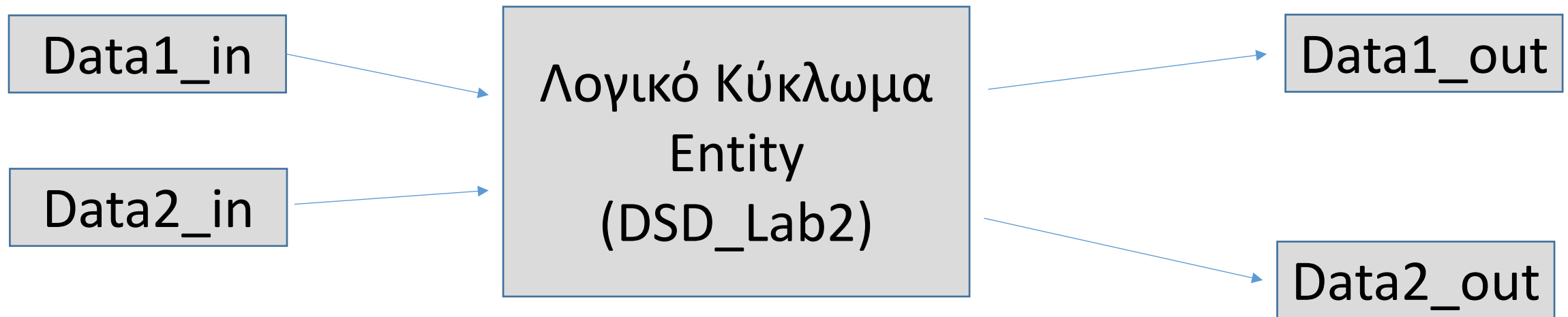
VHDL – Παράδειγμα

Πραγματικό πρόβλημα – Βήματα επίλυσης

1. Δημιουργία νέου project
2. Δημιουργία Entity – Εντοπισμός Input/Output του συστήματος
3. Εύρεση πίνακα αληθείας για κάθε έξοδο του συστήματος
4. Δημιουργία Architecture – Θα έχετε τουλάχιστον τόσες εντολές όσες είναι και οι έξοδοι του συστήματος. Κάθε μία εντολή αντιστοιχεί σε μία έξοδο.
5. Δημιουργία RTL αναπαράστασης
6. Προσομοίωση
7. Σύνθεση

VHDL – Παράδειγμα

Απλοποιημένη μορφή κυκλώματος – Είσοδοι/Εξοδοι



VHDL – Παράδειγμα

Βήμα 2: Περιγραφή Οντότητας

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

entity DSD_Lab3 is
  Port ( data1_in : in STD_LOGIC_VECTOR (1 downto 0);
        data2_in : in STD_LOGIC_VECTOR (1 downto 0);
        data1_out : out STD_LOGIC;
        data2_out : out STD_LOGIC_VECTOR (1 downto 0)
        );
end entity DSD_Lab3;
```

VHDL – Παράδειγμα

Βήμα 3: Πίνακας αληθείας κυκλώματος για έξοδο Data1_out

Data1_in(1)	Data1_in(0)	Data2_in(1)	Data2_in(0)	Data1_out
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

VHDL – Παράδειγμα

Βήμα 3: Πίνακας Karnaugh για εύρεση και απλοποίηση συνάρτησης εξόδου

Data1_in\Data2_in	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

Data1_out=

VHDL – Παράδειγμα

Βήμα 4a: Περιγραφή Αρχιτεκτονικής - Dataflow

architecture DF of DSD_Lab3 is

begin

```
data1_out<= not( data1_in(0) xor data1_in(1) xor data2_in(0) xor data2_in(1));    --xnor
```

```
data2_out<= data1_in xor data2_in;
```

end architecture DF;

VHDL – Παράδειγμα

Βήμα 4b: Περιγραφή Αρχιτεκτονικής - Behavioral

```
architecture Beh of DSD_Lab3 is
```

```
begin
```

```
  beh_arc: process (data1_in, data2_in) is
```

```
  begin
```

```
    data1_out<= not( data1_in(0) xor data1_in(1) xor data2_in(0) xor data2_in(1));    --xnor
```

```
    data2_out<=data1_in xor data2_in;
```

```
  end process beh_arc;
```

```
end architecture Beh;
```

VHDL – Παράδειγμα

Βήμα 4c: Περιγραφή Αρχιτεκτονικής - Structural

```
architecture Structural of DSD_Lab3 is
```

```
component Xor_Gate is  
  Port ( IO : in STD_LOGIC;  
        I1 : in STD_LOGIC;  
        O  : out STD_LOGIC);  
end component Xor_Gate;
```

```
component Not_Gate is  
  Port ( IO : in STD_LOGIC;  
        O  : out STD_LOGIC);  
end component Not_Gate;
```

```
signal temp_xor1 : STD_LOGIC;  
signal temp_xor2 : STD_LOGIC;  
signal temp_xor3 : STD_LOGIC;
```

```
begin
```

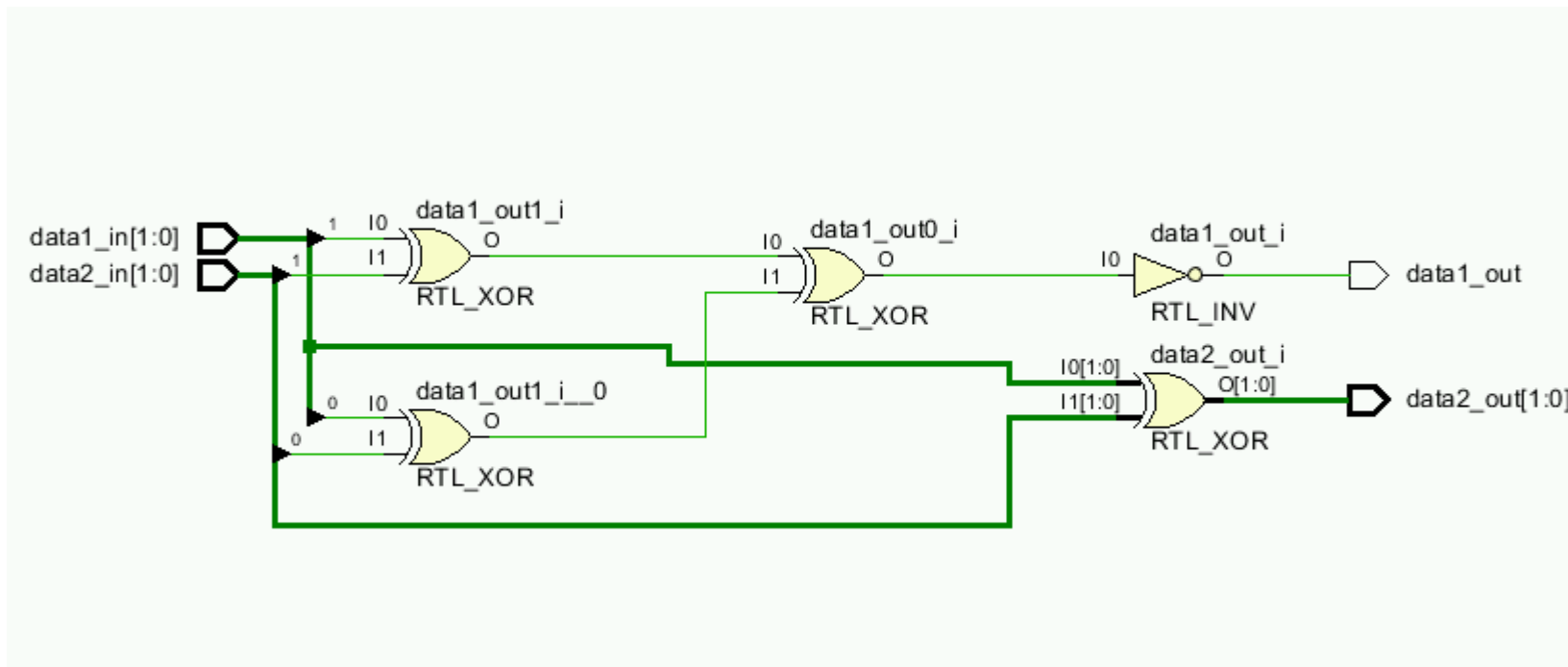
```
xor1: Xor_Gate port map(data1_in(0),data1_in(1),  
temp_xor1);  
xor2: Xor_Gate port map(data2_in(0),data2_in(1),  
temp_xor2);  
xor3: Xor_Gate port map(temp_xor1,temp_xor2, temp_xor3);  
not1: Not_gate port map(temp_xor3, data1_out);
```

```
xor4: Xor_Gate port map(data1_in(0),data2_in(0),  
data2_out(0));  
xor5: Xor_Gate port map(data1_in(1),data2_in(1),  
data2_out(1));
```

```
end architecture Structural;
```

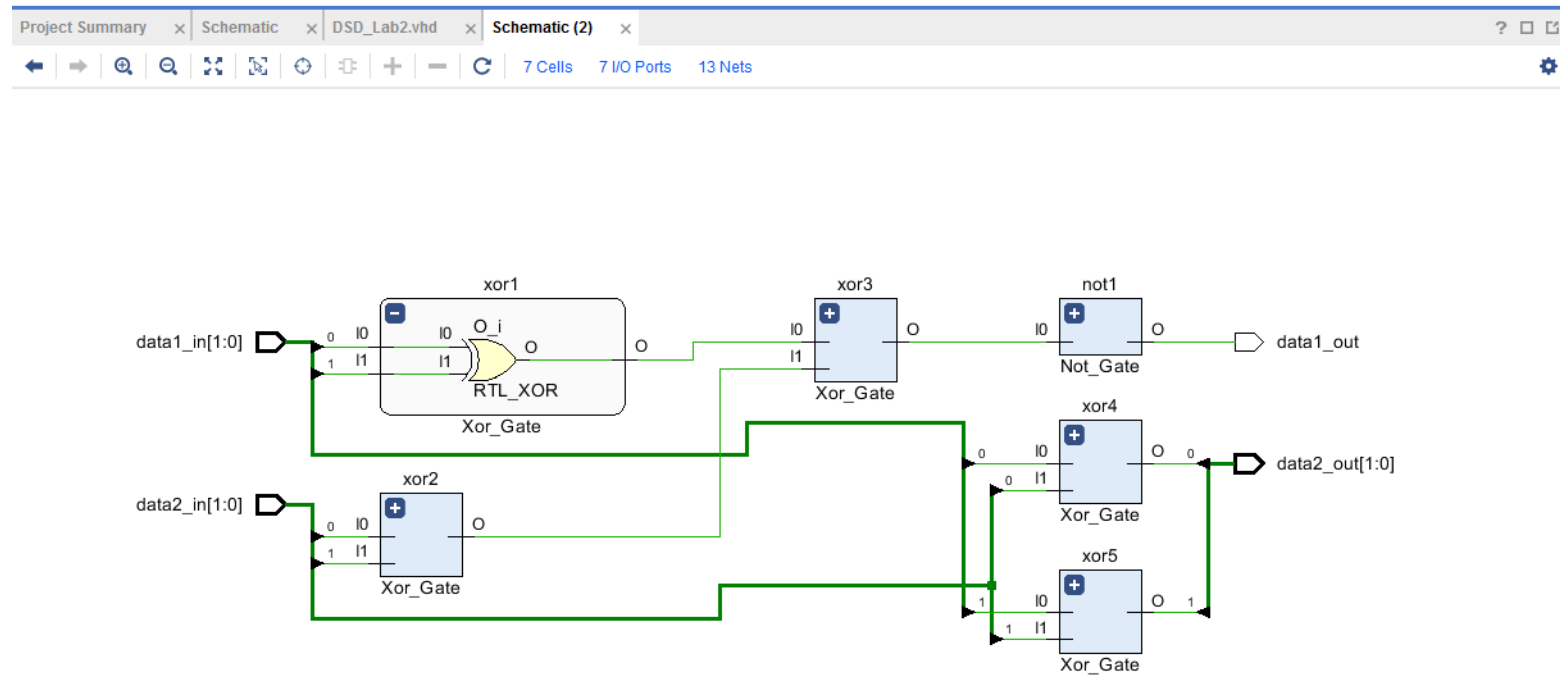
VHDL – Παράδειγμα

Βήμα 5α: RTL Αναπαράσταση – Dataflow/Behavioral



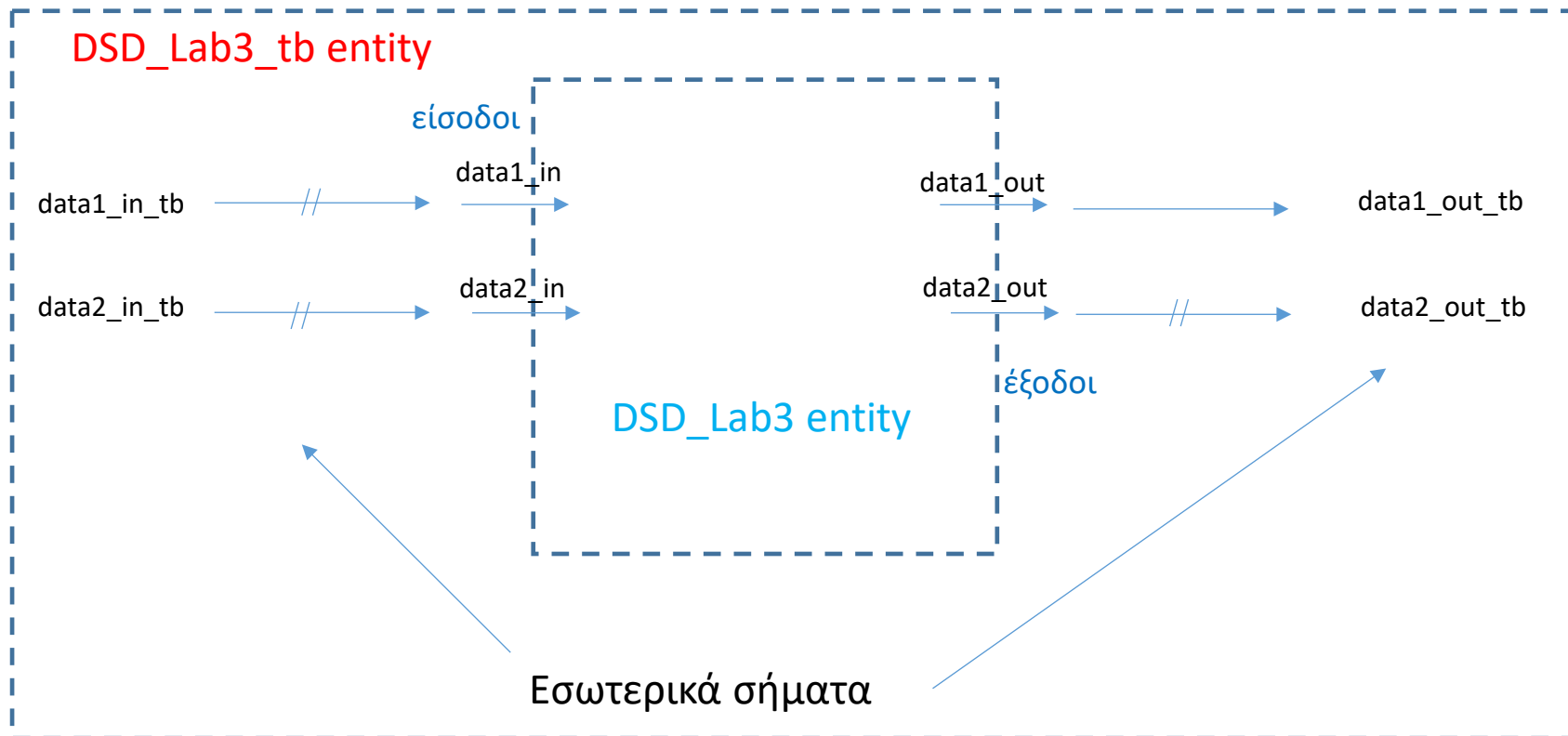
VHDL – Παράδειγμα

Βήμα 5b: RTL Αναπαράσταση – Structural



VHDL – Παράδειγμα

Βήμα 6: Simulation



VHDL – Παράδειγμα

Βήμα 6α: Simulation

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
entity DSD_Lab3_tb is
end DSD_Lab3_tb;

architecture DF_tb of DSD_Lab3_tb is

component DSD_Lab3 is
  Port ( data1_in : in STD_LOGIC_VECTOR (1 downto 0);
        data2_in : in STD_LOGIC_VECTOR (1 downto 0);
        data1_out : out STD_LOGIC;
        data2_out : out STD_LOGIC_VECTOR (1 downto 0)
        );
end component DSD_Lab3 ;

signal data1_in_tb : STD_LOGIC_VECTOR (1 downto 0);
signal data2_in_tb : STD_LOGIC_VECTOR (1 downto 0);
signal data1_out_tb : STD_LOGIC;
signal data2_out_tb : STD_LOGIC_VECTOR (1 downto 0)

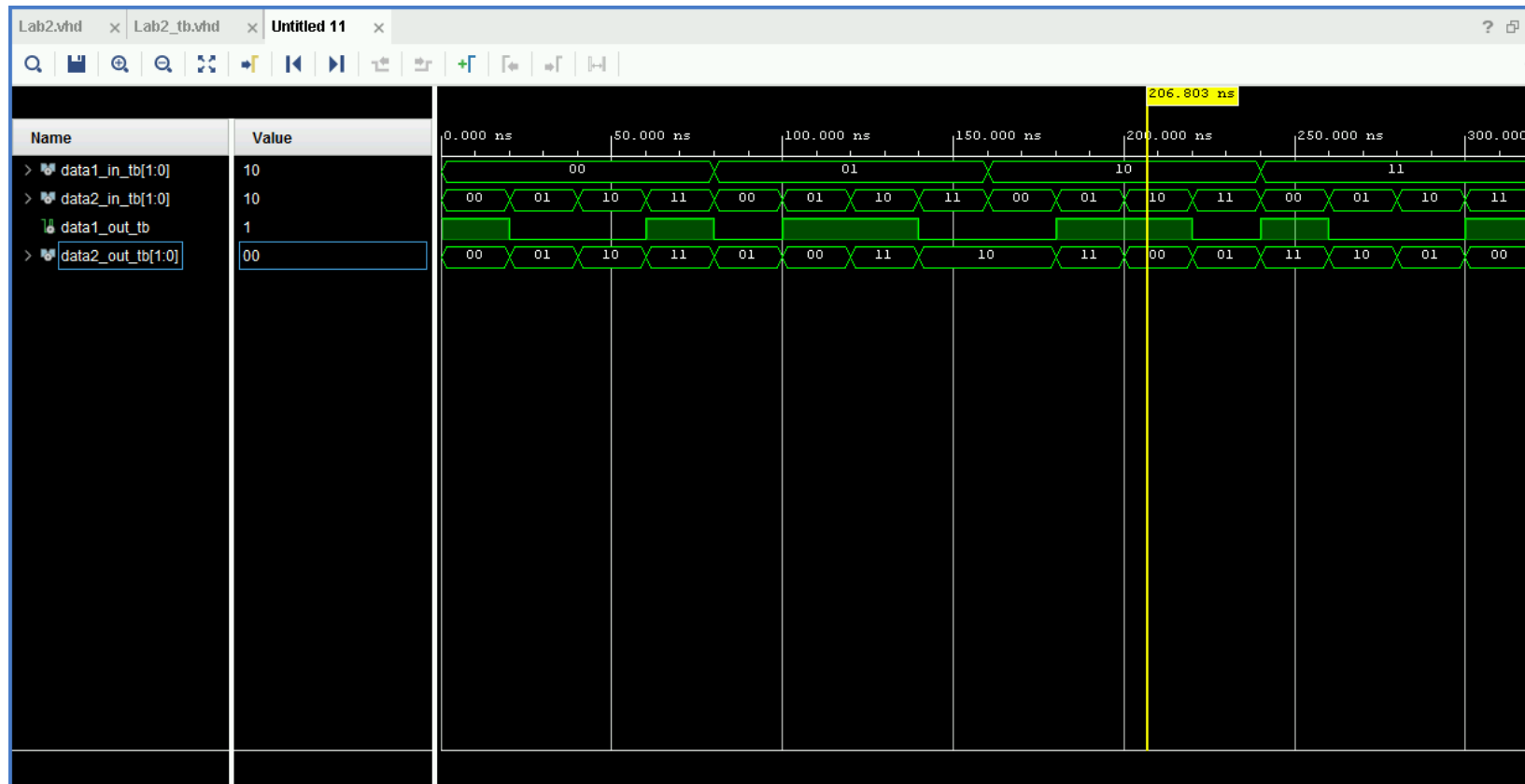
Begin
 uut: DSD_Lab3 port map (data1_in=>data1_in_tb,data2_in=>data2_in_tb,
 data1_out=>data1_out_tb,data2_out=>data2_out_tb);
```

```
test_DF: process is
begin
 data1_in_tb<="00";data2_in_tb<="00";wait for 20 ns;
 data1_in_tb<="00";data2_in_tb<="01";wait for 20 ns;
 data1_in_tb<="00";data2_in_tb<="10";wait for 20 ns;
 data1_in_tb<="00";data2_in_tb<="11";wait for 20 ns;
 data1_in_tb<="01";data2_in_tb<="00";wait for 20 ns;
 data1_in_tb<="01";data2_in_tb<="01";wait for 20 ns;
 data1_in_tb<="01";data2_in_tb<="10";wait for 20 ns;
 data1_in_tb<="01";data2_in_tb<="11";wait for 20 ns;
 data1_in_tb<="10";data2_in_tb<="00";wait for 20 ns;
 data1_in_tb<="10";data2_in_tb<="01";wait for 20 ns;
 data1_in_tb<="10";data2_in_tb<="10";wait for 20 ns;
 data1_in_tb<="10";data2_in_tb<="11";wait for 20 ns;
 data1_in_tb<="11";data2_in_tb<="00";wait for 20 ns;
 data1_in_tb<="11";data2_in_tb<="01";wait for 20 ns;
 data1_in_tb<="11";data2_in_tb<="10";wait for 20 ns;
 data1_in_tb<="11";data2_in_tb<="11";wait for 20 ns;

end process test_DF;
end DF_tb;
```

VHDL – Παράδειγμα

Βήμα 6b: Waveform



VHDL – Improving code

Βήμα 6c: Simulation

```
use IEEE.NUMERIC_STD.ALL; -- package declaration part
```

```
test_DF: process is  
begin
```

```
for i in 0 to 3 loop
```

```
  data1_in_tb<=std_logic_vector(to_unsigned(i, data1_in_tb'length));
```

```
  for j in 0 to 3 loop
```

```
    data2_in_tb<=std_logic_vector(to_unsigned(j, data2_in_tb'length));wait for 20 ns;
```

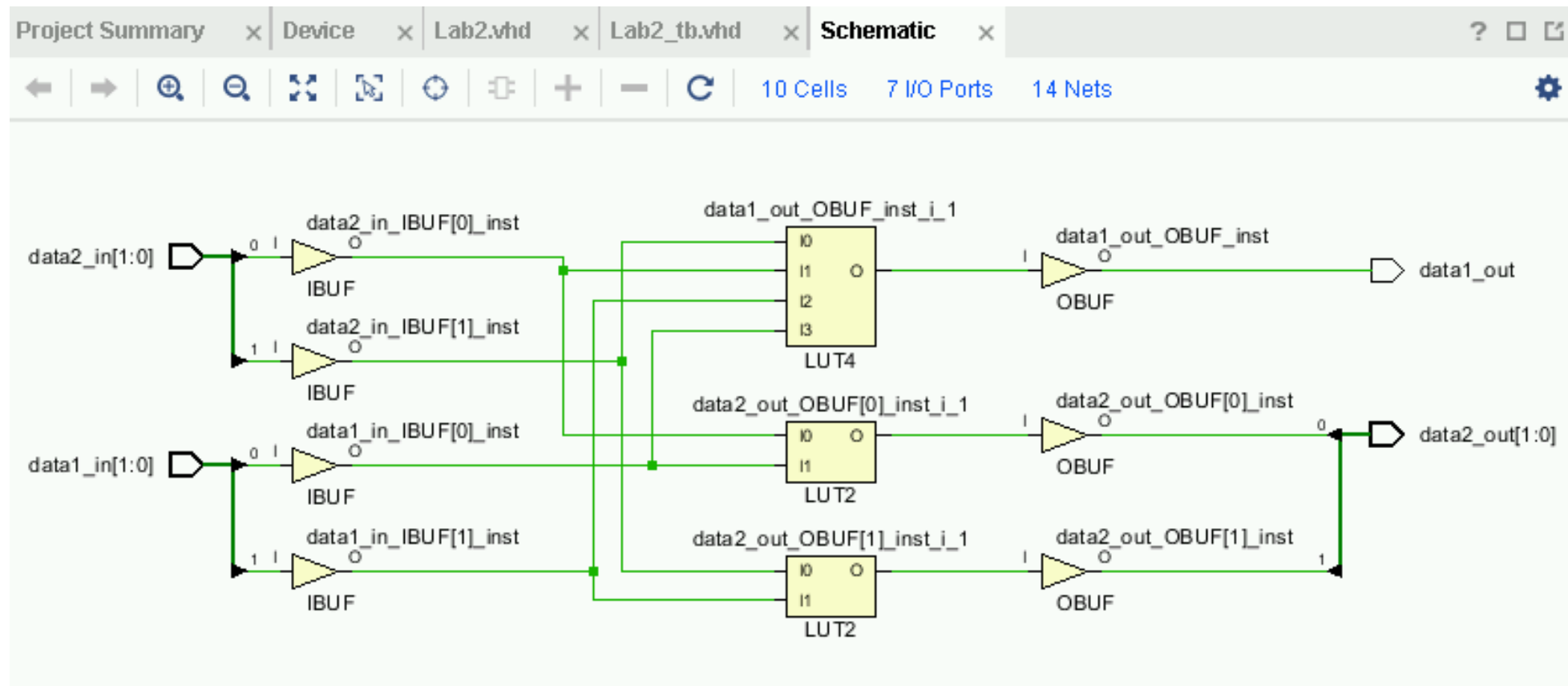
```
  end loop;
```

```
end loop;
```

```
end process test_DF;
```


VHDL – Παράδειγμα

Βήμα 7α: Σύνθεση



VHDL – Παράδειγμα

Βήμα 7b: Post Synthesis Timing Simulation

