Ψηφιακά Συστήματα VHDL (TestBench)

Επ. Καθηγητής Κων/νος Σιώζιος

Τμήμα Φυσικής

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Overview

- Η VHDL εκτός από την χρήση της για την δημιουργία συνθέσιμων κυκλωμάτων χρησιμοποιείται και ως test language

- Απαιτείται εξονυχιστική επαλήθευση της λειτουργίας των συστημάτων που σχεδιάζονται

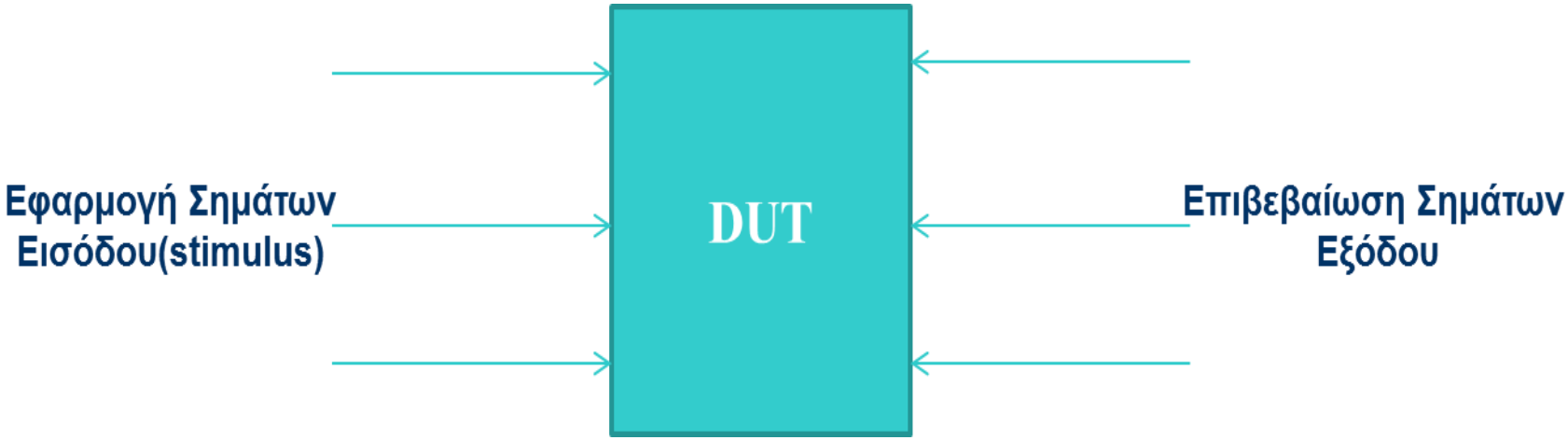
- Για την εξομοίωση των κυκλωμάτων πρέπει να παραχθεί έναν ακόμα entity και architecture.

- Το επιπρόσθετο design που θα παραχθεί για την επαλήθευση του τελικού συστήματος ονομάζεται testbench.

- Τα testbenches δεν είναι συνθέσιμα

Testbench

- Χρησιμοποιούνται για την επιβεβαίωση της σωστής λειτουργίας του κυκλώματος που έχουμε σχεδιάσει
- Οι κύριες λειτουργίες είναι οι εξής:
  - Παραγωγή σημάτων εισόδου/διέγερσης (stimulus)
  - Εφαρμογή των παραγόμενων σημάτων εισόδου στο σύστημα που βρίσκεται υπό εξομοίωση (Design Under Test)
  - Συλλογή των εξόδων από το DUT και σύγκριση με αναμενόμενες τιμές



Testbench

- Το testbench αποτελείται από:
  - ✓ **Entity**
    - Δεν έχει θύρες εισόδου και εξόδου
  - **Architecture**
    - ✓ Ορισμός των σημάτων εισόδου (stimulus), εξόδου και των ενδιάμεσων σημάτων και του DUT
    - ✓ Δημιουργία των σημάτων εισόδου/εξόδου
    - ✓ Δημιουργία του σήματος ρολογιού αν απαιτείται
    - ✓ Σύνδεση όλων των σημάτων με το DUT
    - ✓ Σύγκριση της εξόδου και επιβεβαίωση ορθής λειτουργίας

Clock Generation

Απαιτείται η παραγωγή του σήματος ρολογιού εάν το κύκλωμα που βρίσκεται υπό εξομοίωση χρησιμοποιεί ακολουθιακά στοιχεία

1ος Τρόπος

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is

    constant ClockPeriod : TIME := 50 ns;
    signal clock : std_logic := '0';

    begin

Clock_Generate: process
                begin
                    clock <= not clock after ClockPeriod/2;
                end process;

end Behavioral;
```

2ος Τρόπος

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is

    constant ClockPeriod : TIME := 50 ns;
    signal clock : std_logic := '0';

    begin

Clock_Generate: process
                begin
                   wait for ClockPeriod/2;
                   clock <= '1';
                   wait for ClockPeriod/2;
                   clock <= '0';
                end process;

end Behavioral;
```

Παραγωγή Διεγέρσεων Εισόδου

- Απαιτείται η δημιουργία των σημάτων εισόδου και η εφαρμογή τους στο DUT.

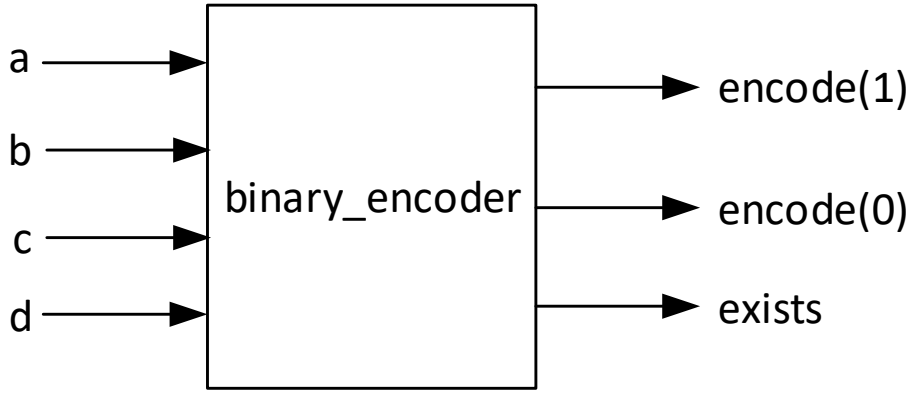
- Τα σήματα εισόδου εφαρμόζονται στο DUT με τη χρήση concurrent κώδικα.

- Οι συνηθέστεροι τρόποι εφαρμογής εισόδων είναι 2:
  - ✓ VHDL-Absolute Time
  - ✓ VHDL-Relative Time

- **Absolute Time:** Όλα τα σήματα εισόδου εφαρμόζονται με σημείο αναφοράς τον χρόνο εξομοίωσης 0.

- **Relative Time:** Εφαρμόζονται οι αρχικές τιμές των σημάτων στο DUT και στην συνέχεια με την αλλαγή κάποιων σημάτων ελέγχου αλλάζουν και οι είσοδοι.

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2



- Κάθε είσοδος αντιστοιχεί σε έναν αριθμό: (a-3, b-2, c-1, d-0). Όταν μόνο μία είσοδος είναι ενεργή, ο αντίστοιχος αριθμός εμφανίζεται στην έξοδο encode.
- Αν περισσότερες από μια εισόδους είναι ενεργές, στην έξοδο εμφανίζεται ο μεγαλύτερος αριθμός (κωδικοποίηση κατά προτεραιότητα). Σε αυτές τις περιπτώσεις η έξοδος exists είναι στο λογικό 1.
- Αν δεν υπάρχουν ενεργές είσοδοι, τότε η έξοδος encode τίθεται σε κατάσταση απομόνωσης ενώ η έξοδος exists είναι 0.

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
library IEEE;
use IEEE.std_logic_1164.all;

entity binary_encoder is
   port (
      a: in std_logic;
      b: in std_logic;
      c: in std_logic;
      d: in std_logic;
      exists: inout std_logic;
      encode: out std_logic_vector (1 downto 0));
end binary_encoder;

architecture binary_encoder of binary_encoder is
begin
      encode(1) <= a or b when exists='1' else 'Z';
      encode(0) <= a or ((not b) and c) when exists='1'
          else 'Z';
      exists <= a or b or c or d;
end binary_encoder;
```

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

1ος τρόπος

Είναι ο απλούστερος τρόπος.

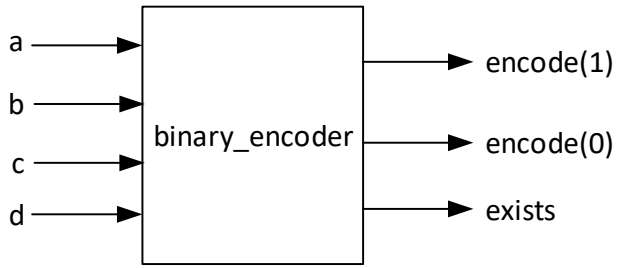
Αποτελούνται από εντολές ανάθεσης που καθορίζουν τις τιμές των σημάτων εισόδου.

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity binary_encoder_tb is
5  end binary_encoder_tb;
6
7  architecture TB_ARCHITECTURE of binary_encoder_tb is
8
9      component binary_encoder
10     port(
11         a : in std_logic;
12         b : in std_logic;
13         c : in std_logic;
14         d : in std_logic;
15         exists : inout std_logic;
16         encode : out std_logic_vector(1 downto 0));
17     end component;
18
19     signal a : std_logic:='0';
20     signal b : std_logic:='0';
21     signal c : std_logic:='0';
22     signal d : std_logic:='0';
23     signal exists : std_logic;
24     signal encode : std_logic_vector(1 downto 0);
25
26 begin
27     UUT : binary_encoder
28         port map
29             (a => a,
30             b => b,
31             c => c,
32             d => d,
33             exists => exists,
34             encode => encode );
35
36     a<=not a after 25 ns;
37     b<=not b after 50 ns;
38     c<=not c after 100 ns;
39     d<=not d after 200 ns;
40 end TB_ARCHITECTURE;
41
42 configuration TESTBENCH_FOR_binary_encoder of binary_encoder_tb is
43     for TB_ARCHITECTURE
44         for UUT : binary_encoder
45             use entity work.binary_encoder(binary_encoder);
46         end for;
47     end for;
48 end TESTBENCH_FOR_binary_encoder;
49
```

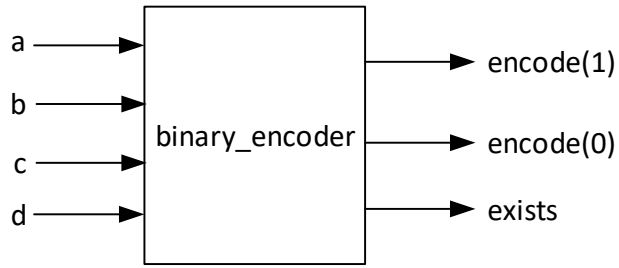
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity binary_encoder_tb is
5   end binary_encoder_tb;
6
7   architecture TB_ARCHITECTURE of binary_encoder_tb is
8
9       component binary_encoder
10      port(
11          a : in std_logic;
12          b : in std_logic;
13          c : in std_logic;
14          d : in std_logic;
15          exists : inout std_logic;
16          encode : out std_logic_vector(1 downto 0));
17      end component;
18
19      signal a : std_logic:='0';
20      signal b : std_logic:='0';
21      signal c : std_logic:='0';
22      signal d : std_logic:='0';
23      signal exists : std_logic;
24      signal encode : std_logic_vector(1 downto 0);
```

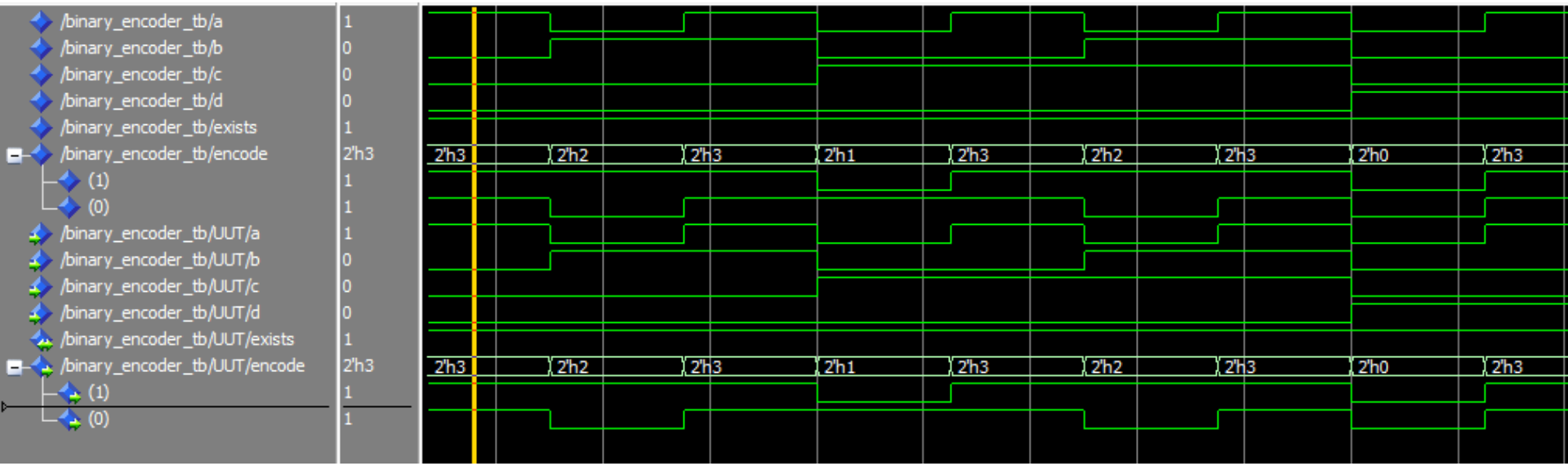


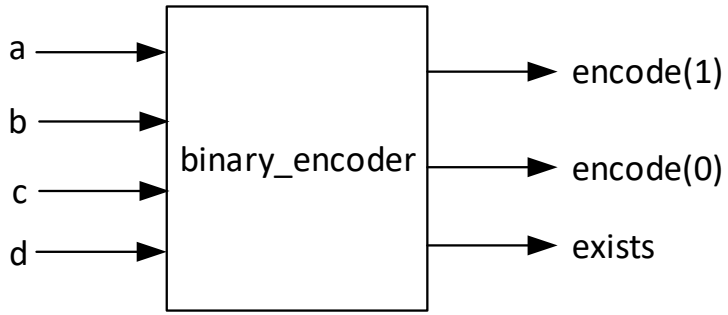
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
26  begin
27      UUT : binary_encoder
28          port map
29              (a => a,
30              b => b,
31              c => c,
32              d => d,
33              exists => exists,
34              encode => encode );
35
36      a<=not a after 25 ns;
37      b<=not b after 50 ns;
38      c<=not c after 100 ns;
39      d<=not d after 200 ns;
40  end TB_ARCHITECTURE;
41
42  configuration TESTBENCH_FOR_binary_encoder of binary_encoder_tb is
43      for TB_ARCHITECTURE
44          for UUT : binary_encoder
45              use entity work.binary_encoder(binary_encoder);
46          end for;
47      end for;
48  end TESTBENCH_FOR_binary_encoder;
```



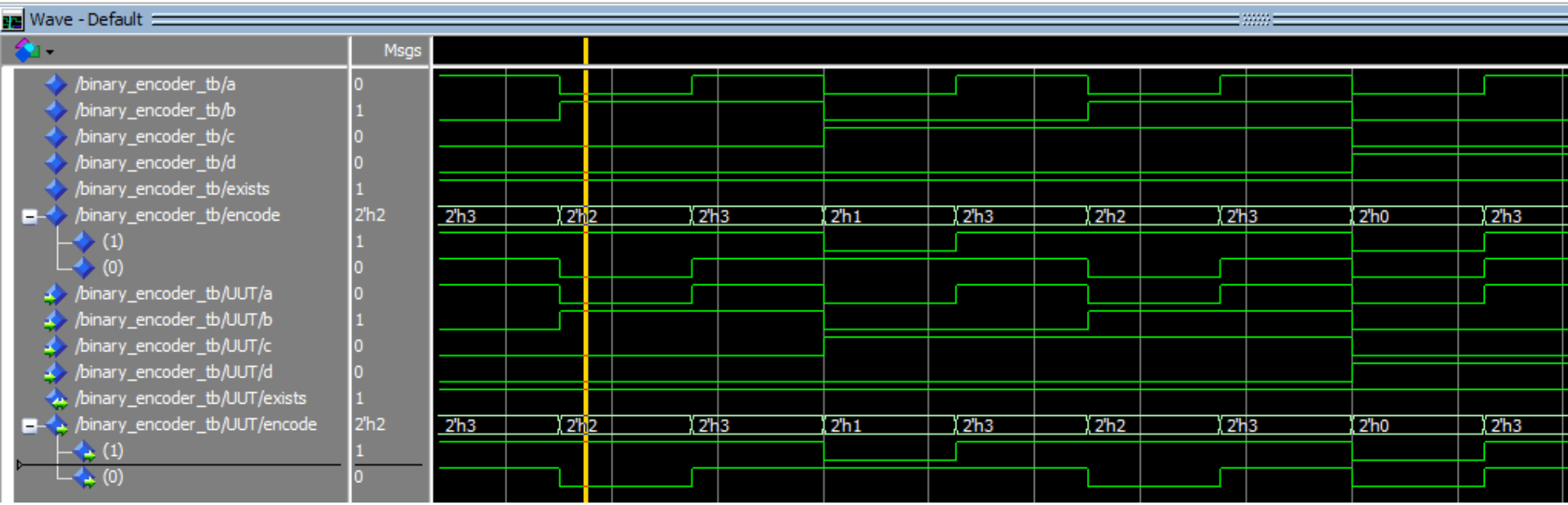
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

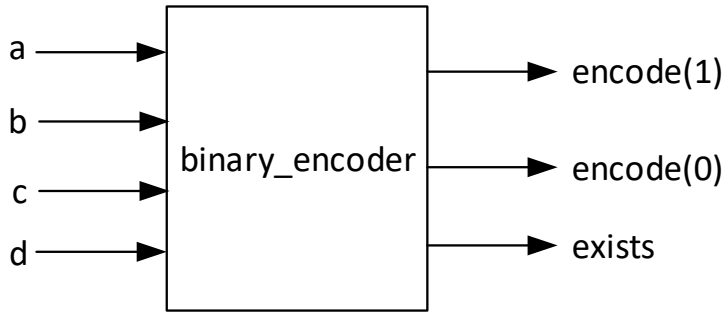




a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

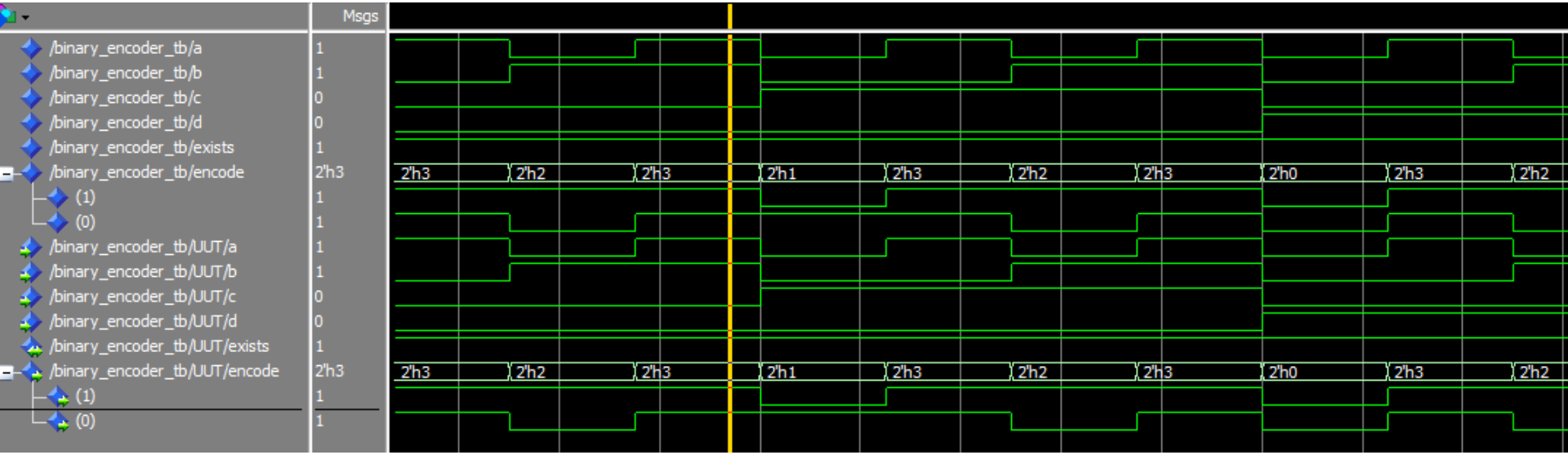
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

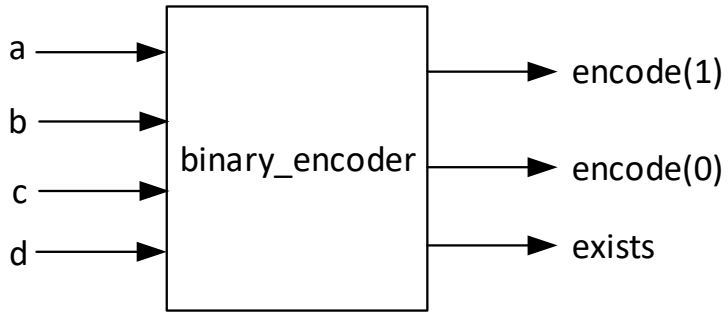




a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

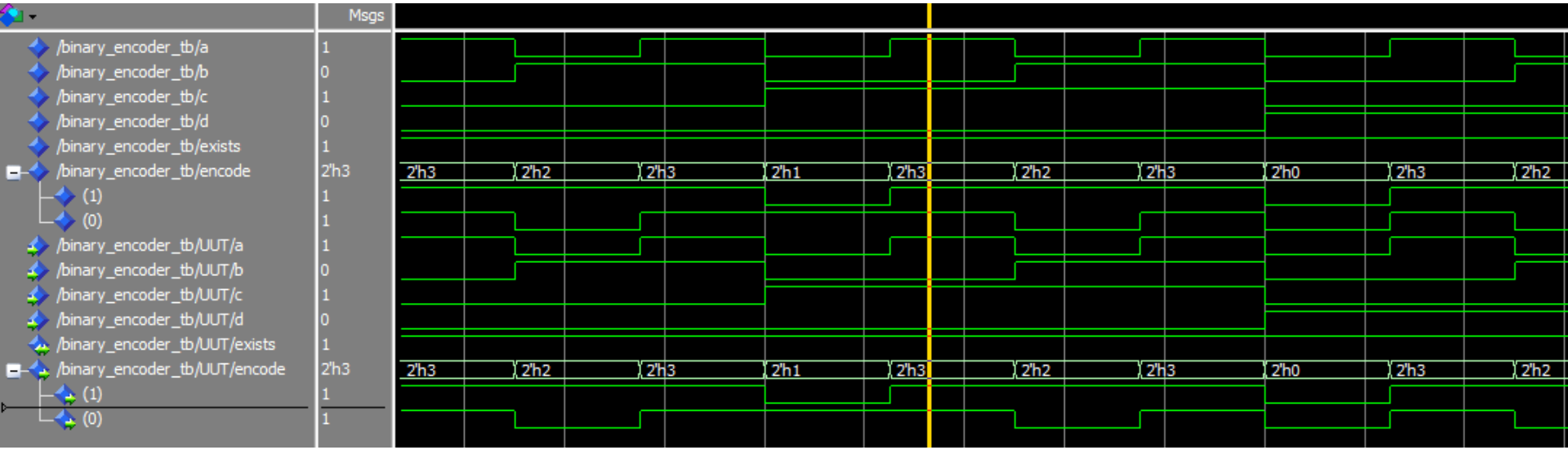
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

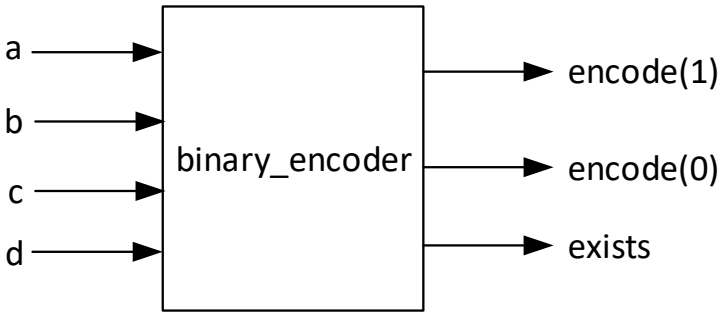




a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2





a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

2ος τρόπος

Βασίζεται στις συναρτήσεις χειρισμού αρχείων.

Υπάρχει ένας βρόχος που διαβάζει συνεχώς εισόδους, τις αναθέτει στα σήματα εισόδου της μονάδας UUT και αποθηκεύει ή συγκρίνει τις τιμές των εξόδων που προκύπτουν.

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
library ieee;
use ieee.std_logic_1164.all;
use STD.TEXTIO.all;
-- Add your library and packages declaration here ...

entity binary_encoder_tb is
end binary_encoder_tb;

architecture TB_ARCHITECTURE of binary_encoder_tb is
    file IN_VECTORS: TEXT open READ_MODE is "inputs.txt";
    file OUT_VECTORS: TEXT open WRITE_MODE is "outputs.txt";
    -- Component declaration of the tested unit
    component binary_encoder
    port(
        a : in std_logic;
        b : in std_logic;
        c : in std_logic;
        d : in std_logic;
        exists : inout std_logic;
        encode : out std_logic_vector(1 downto 0) );
end component;

    -- Stimulus signals
    -- signals mapped to the I/O ports of tested entity
    signal a : std_logic;
    signal b : std_logic;
    signal c : std_logic;
    signal d : std_logic;
    -- Observed signals
    --signals mapped to the output ports of tested entity
    signal exists : std_logic;
    signal encode : std_logic_vector(1 downto 0);
    -- Add your code here ...

begin
    -- Unit Under Test port map
    UUT : binary_encoder
        port map
            (a => a,
            b => b,
            c => c,
            d => d,
            exists => exists,
            encode => encode );
    -- Add your stimulus here ...
    -- Add your stimulus here ...
    process
        variable IN_BUF: LINE;
        variable OUT_BUF: LINE;
        variable a_var,b_var,c_var,d_var : bit;
    begin
        while not ENDFILE(IN_VECTORS) loop
            READLINE(IN_VECTORS,IN_BUF);
            READ(IN_BUF,a_var);
            READ(IN_BUF,b_var);
            READ(IN_BUF,c_var);
            READ(IN_BUF,d_var);
            a<=to_stdulogic(a_var);
            b<=to_stdulogic(b_var);
            c<=to_stdulogic(c_var);
            d<=to_stdulogic(d_var);
            wait for 5 ns;
        WRITE(OUT_BUF,STRING'("Exists= "));
        WRITE(OUT_BUF,to_bit(exists));
        WRITE(OUT_BUF,STRING'(", Encode= "));
        WRITE(OUT_BUF,to_bitvector(encode));
        WRITELINE(OUT_VECTORS,OUT_BUF);
    end loop;
    wait;
end process;
end TB_ARCHITECTURE;

configuration TESTBENCH_FOR_binary_encoder of binary_encoder_tb is
    for TB_ARCHITECTURE
        for UUT : binary_encoder
            use entity work.binary_encoder(binary_encoder);
        end for;
    end for;
end TESTBENCH_FOR_binary_encoder;
```

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

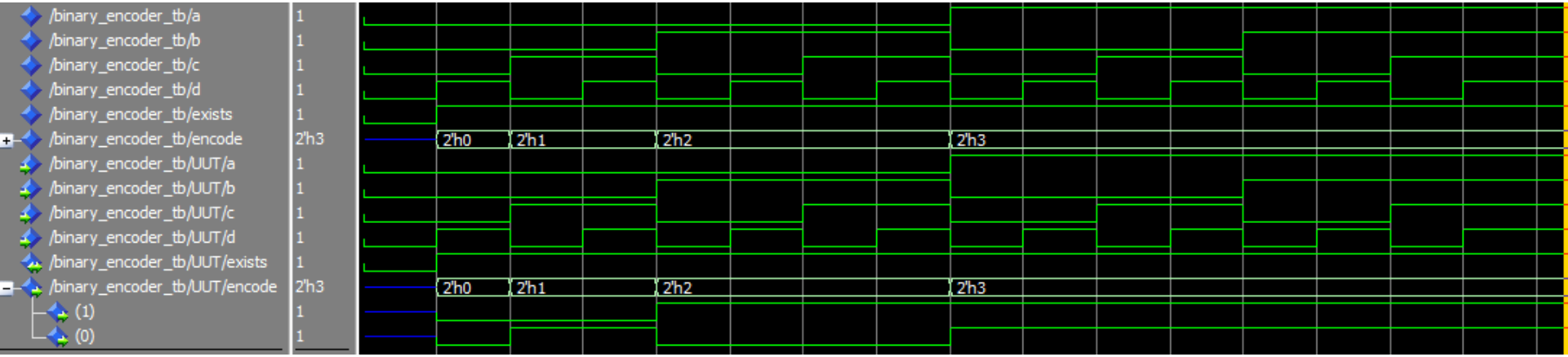
inputs.txt

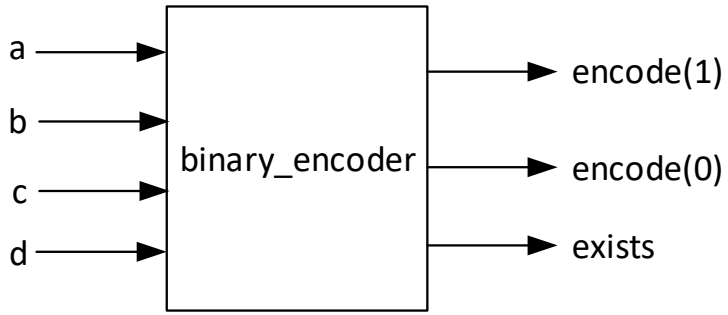
```
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

outputs.txt

```
Exists= 0, Encode= 00
Exists= 1, Encode= 00
Exists= 1, Encode= 01
Exists= 1, Encode= 01
Exists= 1, Encode= 10
Exists= 1, Encode= 10
Exists= 1, Encode= 10
Exists= 1, Encode= 10
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
Exists= 1, Encode= 11
```

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2





a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

3ος τρόπος

Περιγράφουμε αλγοριθμικά την λειτουργία του κυκλώματος UUT και συγκρίνονται τα αποτελέσματα.

Τυχόν ασυμφωνίες ελέγχονται και αναφέρονται με την εντολή assert.

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
library ieee;
use ieee.std_logic_1164.all;
    -- Add your library and packages declaration here ...

entity binary_encoder_tb is
end binary_encoder_tb;

architecture TB_ARCHITECTURE of binary_encoder_tb is
    -- Component declaration of the tested unit
    component binary_encoder
    port(
        a, b, c, d : in std_logic;
        exists : inout std_logic;
        encode : out std_logic_vector(1 downto 0) );
end component;

-- Stimulus signals - signals mapped to the input and inout ports of tested entity
    signal a, b, c, d : std_logic;
-- Observed signals -signals mapped to the output ports of tested entity
    signal exists : std_logic;
    signal encode : std_logic_vector(1 downto 0);
    -- Add your code here ...

begin
    -- Unit Under Test port map
    UUT : binary_encoder
        port map
            (a => a,
            b => b,
            c => c,
            d => d,
            exists => exists,
            encode => encode );
    -- Add your stimulus here ...

process
        function exists_behavior (a,b,c,d: std_logic)
        return std_logic is
        begin
            if (a='1') or (b='1') or (c='1') or (d='1') then
                return '1';
            else
                return '0';
            end if;
        end exists_behavior;
```

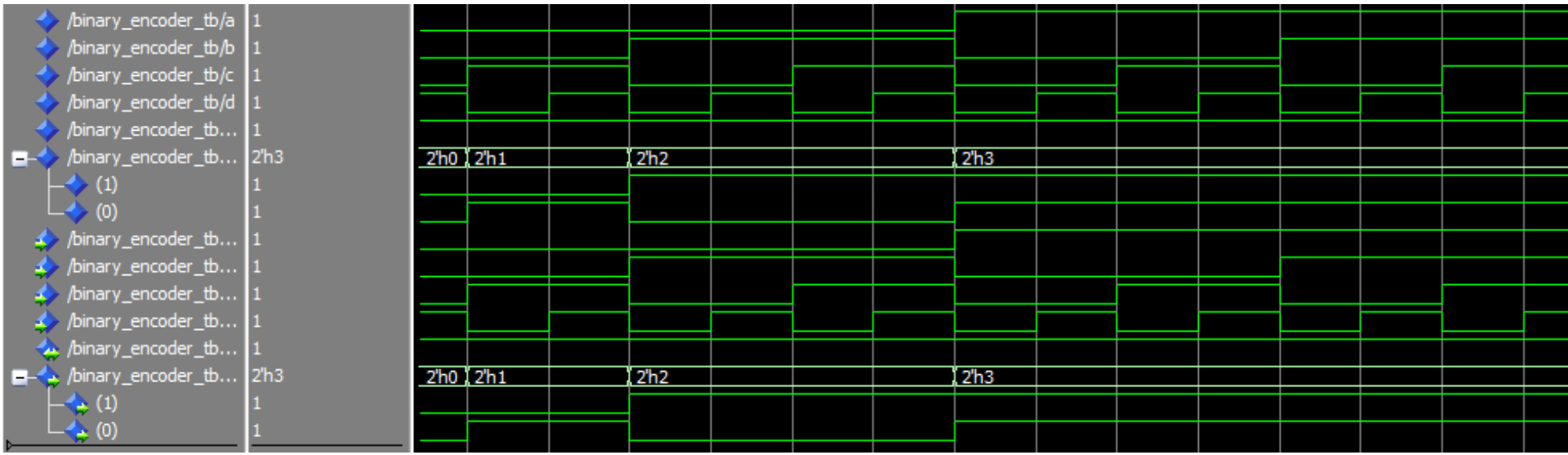
continue...

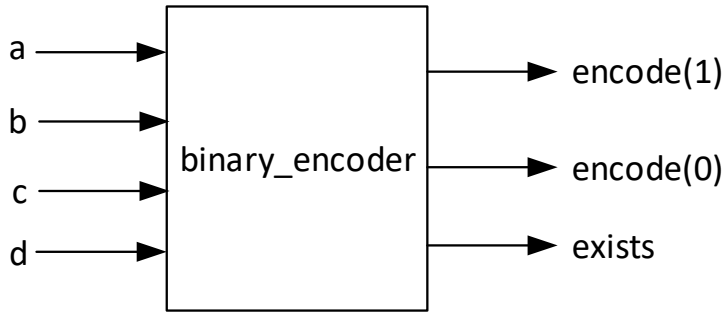
Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2

```
        function encode_behavior (a,b,c,d: std_logic)
            return std_logic_vector is
        begin
            if (a='1') then
                return "11";
            elsif (b='1') then
                return "10";
            elsif (c='1') then
                return "01";
            else
                return "00";
            end if;
        end encode_behavior;
    begin
        for a1 in std_logic'('0') to std_logic'('1') loop
            for b1 in std_logic'('0') to std_logic'('1') loop
                for c1 in std_logic'('0') to std_logic'('1') loop
                    for d1 in std_logic'('0') to std_logic'('1') loop
                        a<=a1;
                        b<=b1;
                        c<=c1;
                        d<=d1;
                        wait for 5 ns;
                        assert(exists=exists_behavior(a,b,c,d))
                        report "Error on signal exists!";
                        assert(encode=encode_behavior(a,b,c,d))
                        report "Error on signal encode!";
                    end loop;
                end loop;
            end loop;
        end loop;
    wait;
end process;
end TB_ARCHITECTURE;

configuration TESTBENCH_FOR_binary_encoder of binary_encoder_tb is
    for TB_ARCHITECTURE
        for UUT : binary_encoder
            use entity work.binary_encoder(binary_encoder);
        end for;
    end for;
end TESTBENCH_FOR_binary_encoder;
```

Δυαδικός κωδικοποιητής προτεραιότητας 4 σε 2





a	b	c	d	enc(0)	enc(1)	exist
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1