# Σχεδίαση Ψηφιακών Συστημάτων - Εργασία Θεωρίας (Μέρος 5)

Χρήστος Μαργιώλης

Ιούλιος 2020

# Περιεχόμενα

# 1 Κώδικας και τεκμηρίωση

## 1.1 instrmem.vhd

Το παρακάτω κύκλωμα υλοποιεί την μνήμη εντολών του MIPS. Ο κώδικας είναι ίδιος με αυτόν της εκφώνησης, απλώς με λίγο διαφορετική μορφοποίηση. Επίσης, στην θέση μνήμης 6 έχει τοποθετηθεί και η εντολή `add $4 $5 $6`.

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

entity instrmem is port (
        addr:   in std_logic_vector(3 downto 0);
        c:      out std_logic_vector(31 downto 0)
);
end instrmem;

architecture dataflow of instrmem is

type instr_arr is array(0 to 15) of std_logic_vector (31 downto 0);
constant instr_mem: instr_arr := (
        "11111111111111111111111111111111", -- 0
        "10001010100101010101000011101111", -- 1
        "11111111111111111111111111111111", -- 2
        "00000000000000000000000000000000", -- 3
        "11111111111111111111111111111111", -- 4
        "00000000000000000000000000000000", -- 5
        "00000000101001100010000000100000", -- 6
        "11111111111111111111111111111111", -- 7
        "11111111111111111111111111111111", -- 8
        "11111111111111111111111111111111", -- 9
        "10101010101011110000101110001010", -- 10
        "11111111111111110000000000000000", -- 11
        "10001011101010111010111101010111", -- 12
        "11111111111111111111111111111111", -- 13
        "10110111000111010101010101111111", -- 14
        "11111111111111111111111111111111"  -- 15
);

begin
        c <= instr_mem(to_integer(unsigned(addr)));
end dataflow;
```

## 1.2 instrmem_tb.vhd

Testbench για την μνήμη εντολών του MIPS. Δίνουμε διάφορες τιμές θέσεων μνήμης και η έξοδος είναι τα περιεχόμενα της μνήμης στις θέσεις αυτές.

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

entity instrmem_tb is
end instrmem_tb;

architecture behav of instrmem_tb is

signal s_addr:  std_logic_vector(3 downto 0);
signal s_c:     std_logic_vector(31 downto 0);

component instrmem is port (
        addr:   in std_logic_vector(3 downto 0);
        c:      out std_logic_vector(31 downto 0)
);
end component;

begin
        uut: instrmem port map (
                addr => s_addr,
                c => s_c
        );

        process begin
                s_addr <= "0001";
                wait for 250 ns;

                s_addr <= "0010";
                wait for 250 ns;

                s_addr <= "0110";
                wait for 250 ns;

                s_addr <= "1000";
                wait for 250 ns;

                s_addr <= "1010";
                wait for 250 ns;
        end process;
end behav;
```

## 1.3 datamem.vhd

Το παρακάτω κύκλωμα υλοποιεί την μνήμη εντολών του MIPS. Ο κώδικας είναι ίδιος με αυτόν της εκφώνησης, απλώς με λίγο διαφορετική μορφοποίηση.

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

entity datamem is port (
        clk:    in std_logic;
        addr:   in std_logic_vector(5 downto 0);
        we:     in std_logic;
        re:     in std_logic;
        writed: in std_logic_vector(31 downto 0);
        readd:  out std_logic_vector(31 downto 0)
);
end datamem;

architecture behav of datamem is

type data_arr is array(0 to 63) of std_logic_vector(31 downto 0);
signal memfile: data_arr;

begin
        process (clk) begin
                if (clk'event and clk = '0') then
                        if we = '1' then
                                memfile(to_integer(unsigned(addr))) <= writed;
                        end if;
                end if;
                if re = '1' then
                        readd <= memfile(to_integer(unsigned(addr)));
                end if;
        end process;
end behav;
```

## 1.4 datamem_tb.vhd

Testbench για την μνήμη εντολών του MIPS. Οι τιμές για το testbench πάρθηκαν από την εκφώνηση της άσκησης.

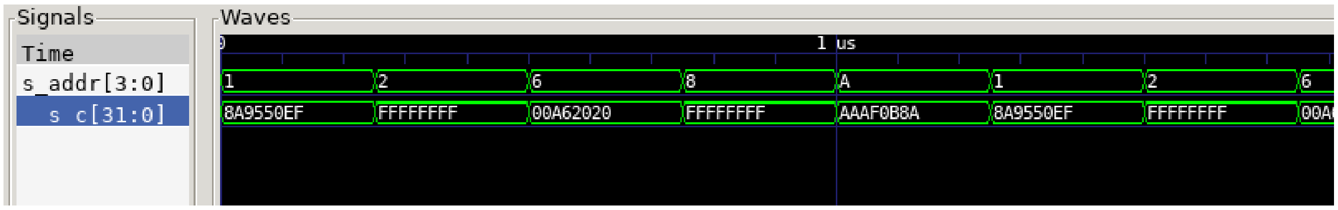
```
library ieee;
use ieee.std_logic_1164.all;

entity datamem_tb is
end datamem_tb;

architecture behav of datamem_tb is

signal s_clk:           std_logic;
signal s_addr:          std_logic_vector(5 downto 0);
signal s_we:            std_logic;
signal s_re:            std_logic;
signal s_writed:        std_logic_vector(31 downto 0);
signal s_readd:         std_logic_vector(31 downto 0);

component datamem is port (
        clk:            in std_logic;
        addr:           in std_logic_vector(5 downto 0);
        we:             in std_logic;
        re:             in std_logic;
        writed:         in std_logic_vector(31 downto 0);
        readd:          out std_logic_vector(31 downto 0)
);
end component;

begin
        uut: datamem port map (
                clk => s_clk,
                addr => s_addr,
                we => s_we,
                re => s_re,
                writed => s_writed,
                readd => s_readd
        );

        process begin
                s_clk <= '0';
                wait for 250 ns;

                s_clk <= '1';
                wait for 250 ns;
```

```
s_we <= ’1’;
s_re <= ’0’;
s_addr <= "000000";
s_writed <= "01010101010101010101010101010101";
wait for 250 ns;

s_clk <= ’0’;
wait for 250 ns;

s_clk <= ’1’;
wait for 250 ns;

s_addr <= "000001";
s_writed <= "11011101110111011101110111011101";
wait for 250 ns;

s_clk <= ’0’;
wait for 250 ns;

s_clk <= ’1’;
wait for 250 ns;

s_addr <= "000010";
s_writed <= "00100010001000100010001000100010";
wait for 250 ns;

s_clk <= ’0’;
wait for 250 ns;

s_clk <= ’1’;
wait for 250 ns;

s_addr <= "000011";
s_writed <= "10011001100110011001100110011001";
wait for 250 ns;

s_clk <= ’0’;
wait for 250 ns;

s_clk <= ’1’;
wait for 250 ns;

s_we <= ’0’;
s_re <= ’1’;
s_addr <= "000000";
wait for 250 ns;
```

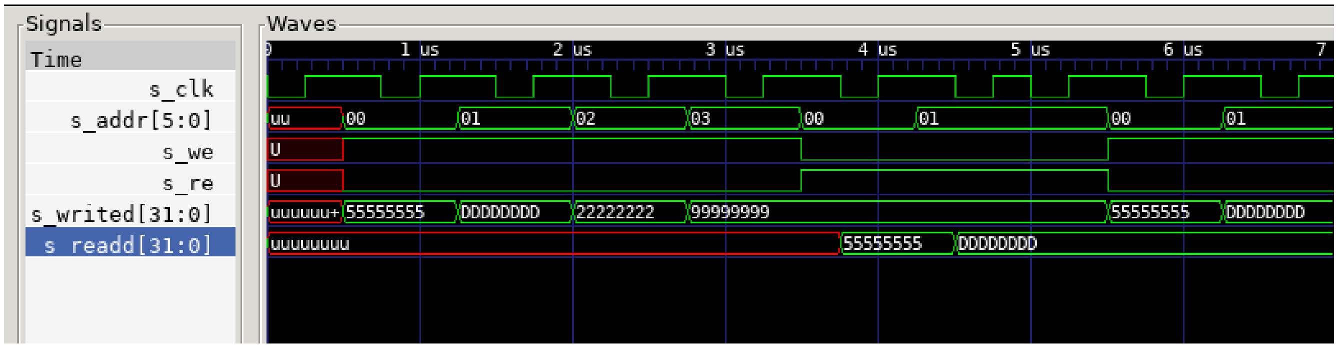
```
                s_clk <= ’0’;
                wait for 250 ns;

                s_clk <= ’1’;
                wait for 250 ns;

                s_addr <= "000001";
                wait for 250 ns;

                s_clk <= ’0’;
                wait for 250 ns;

                s_clk <= ’1’;
                wait for 250 ns;
        end process;
end behav;
```

## 2 Εκτέλεση

### 2.1 instrmem_tb



### 2.2 datamem_tb


Signals
Time
s_clk
s_addr[5:0]
s_we
s_re
s_writed[31:0]
s_readd[31:0]
Waves
0
1 us
2 us
3 us
4 us
5 us
6 us
7
uu
00
01
02
03
00
01
00
01
U
U
uuuuuu+
55555555
DDDDDDDD
22222222
99999999
55555555
DDDDDDDD
uuuuuuuu
55555555
DDDDDDDD