Σχεδίαση Ψηφιακών Συστημάτων - Εργασία Θεωρίας (Μέρος 6)

Χρήστος Μαργιώλης

Ιούλιος 2020

Περιεχόμενα

1 Κώδικας και τεκμηρίωση

1.1 reg.vhd

Το παρακάτω κύκλωμα υλοποιεί έναν καταχωρητή. Ο κώδικας είναι παραμετροποιημένος για να μπορεί να μετατραπεί στο επόμενο μέρος σε 32-bit χωρίς αλλαγές.

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

entity reg is
generic (
        sz:     natural := 4
);
port (
        d:      in std_logic_vector(sz-1 downto 0);
        rst:    in std_logic;
        clk:    in std_logic;
        q:      out std_logic_vector(sz-1 downto 0)
);
end reg;

architecture behav of reg is

-- We want to automatically initialize the vector no matter its size.
signal s_init:  std_logic_vector(sz-1 downto 0) := (others => '0');

begin
        process (rst, clk) begin
                if (rst = '0') then
                        q <= s_init;
```

```
                elsif (rising_edge(clk)) then
                        q <= d;
                end if;
        end process;
end behav;
```

1.2 reg_tb.vhd

Testbench για τον καταχωρητή.

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

entity reg_tb is
end reg_tb;

architecture behav of reg_tb is

signal s_sz:    natural := 4;
signal s_d:     std_logic_vector(s_sz-1 downto 0);
signal s_rst:   std_logic;
signal s_clk:   std_logic;
signal s_q:     std_logic_vector(s_sz-1 downto 0);

component reg is
generic (
        sz:     natural := 4
);
port (
        d:      in std_logic_vector(sz-1 downto 0);
        rst:    in std_logic;
        clk:    in std_logic;
        q:      out std_logic_vector(sz-1 downto 0)
);
end component;

begin
        uut: reg port map (
                d => s_d,
                rst => s_rst,
                clk => s_clk,
                q => s_q
        );

        process begin
                s_rst <= '1';
                s_clk <= '0';
                s_d <= "0010";
                wait for 250 ns;

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

```
                s_clk <= '0';
                s_d <= "1110";
                wait for 250 ns;

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

                s_clk <= '0';
                s_d <= "1010";
                wait for 250 ns;

                s_rst <= '0';
                s_clk <= '1';
                wait for 250 ns;
        end process;
end behav;
```

1.3 regfile.vhd

Το παρακάτω κύκλωμα υλοποιεί ένα register file.

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

entity regfile is
generic (
        sz:     natural := 4;
        addrw:  natural := 3
);
port (
        idata:  in std_logic_vector(sz-1 downto 0);
        addr:   in std_logic_vector(addrw-1 downto 0);
        we:     in std_logic;
        clk:    in std_logic;
        odata:  out std_logic_vector(sz-1 downto 0)
);
end regfile;

architecture behav of regfile is

signal arrsz:   natural := 4;
type regarr     is array(arrsz-1 downto 0) of std_logic_vector(sz-1 downto 0);
signal regf:    regarr;

begin
        process (clk) begin
                if (clk'event and clk = '0') then
                        if (we = '1') then
                                regf(to_integer(unsigned(addr))) <= idata;
                        end if;
                end if;
        end process;
        odata <= regf(to_integer(unsigned(addr)));
end behav;
```

1.4 regfile_tb.vhd

Testbench για το register file.

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

entity regfile_tb is
end regfile_tb;

architecture behav of regfile_tb is

component regfile is
generic (
        sz:     natural := 4;
        addrw:  natural := 3
);
port (
        idata:  in std_logic_vector(sz-1 downto 0);
        addr:   in std_logic_vector(addrw-1 downto 0);
        we:     in std_logic;
        clk:    in std_logic;
        odata:  out std_logic_vector(sz-1 downto 0)
);
end component;

signal s_sz:    natural := 4;
signal s_addrw: natural := 3;
signal s_idata: std_logic_vector(s_sz-1 downto 0);
signal s_addr:  std_logic_vector(s_addrw-1 downto 0);
signal s_we:    std_logic;
signal s_clk:   std_logic;
signal s_odata: std_logic_vector(s_sz-1 downto 0);

begin
        uut: regfile port map (
                idata => s_idata,
                addr => s_addr,
                we => s_we,
                clk => s_clk,
                odata => s_odata
        );

        process begin
                s_we <= '1';
                s_clk <= '0';
```

```
                wait for 250 ns;

                s_clk <= '1';
                s_addr <= "000";
                s_idata <= "0101";
                wait for 250 ns;

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

                s_clk <= '1';
                s_addr <= "001";
                s_idata <= "1101";
                wait for 250 ns;

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

                s_clk <= '1';
                s_addr <= "010";
                s_idata <= "0010";
                wait for 250 ns;

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

                s_clk <= '1';
                s_addr <= "011";
                s_idata <= "1001";
                wait for 250 ns;
        end process;
end behav;
```

1.5 regfile_ext.vhd

Το παρακάτω κύκλωμα υλοποιεί ένα register file με δύο επιπλέον θύρες.

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

entity regfile_ext is
generic (
        sz:             natural := 4;
        addrw:          natural := 3
);
port (
        idata:          in std_logic_vector(sz-1 downto 0);
        raddr1:         in std_logic_vector(addrw-1 downto 0);
        raddr2:         in std_logic_vector(addrw-1 downto 0);
        waddr:          in std_logic_vector(addrw-1 downto 0);
        we:             in std_logic;
        clk:            in std_logic;
        rst:            in std_logic;
        odata1:         out std_logic_vector(sz-1 downto 0);
        odata2:         out std_logic_vector(sz-1 downto 0)
);
end regfile_ext;

architecture behav of regfile_ext is

signal arrsz:           natural := 8;
type regarr             is array(0 to arrsz-1) of std_logic_vector(sz-1 downto 0);
-- Empty register array used for initialization when rst = 1.
signal s_init:          regarr := (others => (others => '0'));
signal regf:            regarr;

begin
        process (clk) begin
                if (rst = '1') then
                        regf <= s_init;
                elsif (clk'event and clk = '0') then
                        if (we = '1') then
                                regf(to_integer(unsigned(waddr))) <= idata;
                        end if;
                end if;
        end process;
        odata1 <= regf(to_integer(unsigned(raddr1)));
```

```
        odata2 <= regf(to_integer(unsigned(raddr2)));
end behav;
```

1.6 regfile_ext_tb.vhd

Testbench για το register file με δύο επιπλέον θύρες.

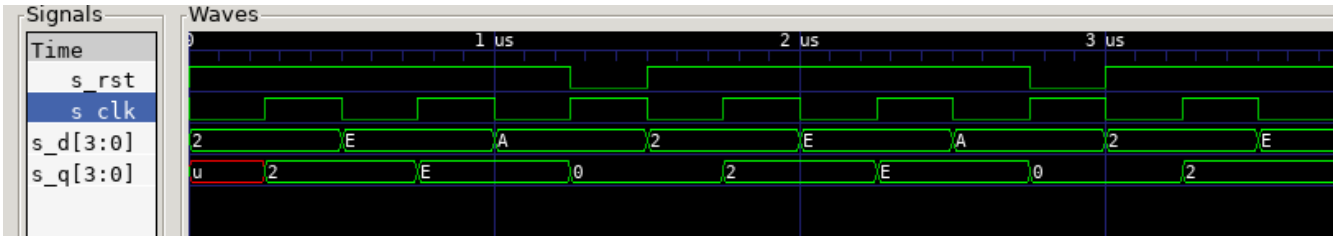
```
library ieee;
use ieee.std_logic_1164.all;

entity regfile_ext_tb is
end regfile_ext_tb;

architecture behav of regfile_ext_tb is

component regfile_ext is
generic (
        sz:             natural := 4;
        addrw:          natural := 2
);
port (
        idata:          in std_logic_vector(sz-1 downto 0);
        raddr1:         in std_logic_vector(addrw-1 downto 0);
        raddr2:         in std_logic_vector(addrw-1 downto 0);
        waddr:          in std_logic_vector(addrw-1 downto 0);
        we:             in std_logic;
        clk:            in std_logic;
        rst:            in std_logic;
        odata1:         out std_logic_vector(sz-1 downto 0);
        odata2:         out std_logic_vector(sz-1 downto 0)
);
end component;

signal s_sz:            natural := 4;
signal s_addrw:         natural := 2;
signal s_idata:         std_logic_vector(s_sz-1 downto 0);
signal s_raddr1:        std_logic_vector(s_addrw-1 downto 0);
signal s_raddr2:        std_logic_vector(s_addrw-1 downto 0);
signal s_waddr:         std_logic_vector(s_addrw-1 downto 0);
signal s_we:            std_logic;
signal s_clk:           std_logic;
signal s_rst:           std_logic;
signal s_odata1:        std_logic_vector(s_sz-1 downto 0);
signal s_odata2:        std_logic_vector(s_sz-1 downto 0);

begin
        uut: regfile_ext port map (
                idata => s_idata,
                raddr1 => s_raddr1,
```

```
        raddr2 => s_raddr2,
        waddr => s_waddr,
        we => s_we,
        clk => s_clk,
        rst => s_rst,
        odata1 => s_odata1,
        odata2 => s_odata2
);

process begin
        s_we <= '1';
        s_clk <= '0';
        s_rst <= '1';
        wait for 250 ns;

        s_clk <= '1';
        s_rst <= '0';
        s_waddr <= "00";
        s_idata <= "0101";
        wait for 250 ns;

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

        s_clk <= '1';
        s_waddr <= "01";
        s_idata <= "1101";
        wait for 250 ns;

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

        s_clk <= '1';
        s_rst <= '0';
        s_waddr <= "10";
        s_idata <= "0010";
        wait for 250 ns;

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

        s_clk <= '1';
        s_waddr <= "11";
        s_idata <= "1001";
        wait for 250 ns;
```

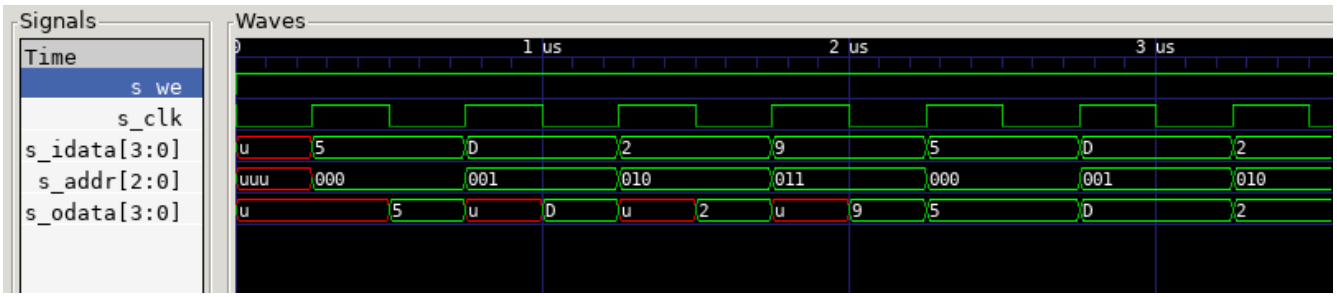
```
                s_clk <= '0';
                wait for 250 ns;

                s_we <= '0';
                s_clk <= '1';
                s_raddr1 <= "00";
                s_raddr2 <= "10";
                wait for 250 ns;

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

                s_we <= '0';
                s_clk <= '1';
                s_raddr1 <= "01";
                s_raddr2 <= "10";
                wait for 250 ns;
        end process;
end behav;
```

2 Εκτέλεση

2.1 reg_tb



2.2 regfile_tb



2.3 regfile_ext_tb