Σχεδίαση Ψηφιακών Συστημάτων - Εργασία Θεωρίας (Μέρος 4)

Χρήστος Μαργιώλης

Ιούλιος 2020

Περιεχόμενα

1 Κώδικας και τεκμηρίωση

1.1 ctrl.vhd

Το παρακάτω κύκλωμα υλοποιεί την μονάδα ελέγχου ενός επεξεργαστή. Η υλοποίηση έγινε με βάση τον πίνακα από τις διαφάνειες του μαθήματος. Στο τέλος προτίμησα την δομή `with-select` για πιο ευανάγνωστο κώδικα.

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

entity ctrl is port (
        funct:          in std_logic_vector(5 downto 0);
        reg_dst:        out std_logic;
        reg_wr:         out std_logic;
        alu_src:        out std_logic;
        branch:         out std_logic;
        mem_rd:         out std_logic;
        mem_wr:         out std_logic;
        mem_toreg:      out std_logic;
        alu_op:         out std_logic_vector(1 downto 0)
);
end ctrl;

architecture dataflow of ctrl is
begin
        reg_dst         <= '1' when funct = "000000" else '0';
        reg_wr          <= '1' when funct = "000000" or funct = "100011" else '0';
        alu_src         <= '1' when funct = "100011" or funct = "101011" else '0';
        branch          <= '1' when funct = "000100" else '0';
        mem_rd          <= '1' when funct = "100011" else '0';
        mem_wr          <= '1' when funct = "101011" else '0';
```

```
        mem_toreg       <= '1' when funct = "100011" else '0';
        with funct select
                alu_op <= "10" when "000000",
                          "01" when "000100",
                          "00" when others;
end dataflow;
```

1.2 ctrl_tb.vhd

Testbench για την μονάδα ελέγχου.

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

entity ctrl_tb is
end ctrl_tb;

architecture behav of ctrl_tb is

signal s_funct:         std_logic_vector(5 downto 0);
signal s_reg_dst:       std_logic;
signal s_reg_wr:        std_logic;
signal s_alu_src:       std_logic;
signal s_branch:        std_logic;
signal s_mem_rd:        std_logic;
signal s_mem_wr:        std_logic;
signal s_mem_toreg:     std_logic;
signal s_alu_op:        std_logic_vector(1 downto 0);

component ctrl is port (
        funct:          in std_logic_vector(5 downto 0);
        reg_dst:        out std_logic;
        reg_wr:         out std_logic;
        alu_src:        out std_logic;
        branch:         out std_logic;
        mem_rd:         out std_logic;
        mem_wr:         out std_logic;
        mem_toreg:      out std_logic;
        alu_op:         out std_logic_vector(1 downto 0)
);
end component;

begin
        uut: ctrl port map (
                funct => s_funct,
                reg_dst => s_reg_dst,
                reg_wr => s_reg_wr,
                alu_src => s_alu_src,
                branch => s_branch,
                mem_rd => s_mem_rd,
                mem_wr => s_mem_wr,
                mem_toreg => s_mem_toreg,
                alu_op => s_alu_op
```

```
        );

        process begin
                s_funct <= "000000";
                wait for 250 ns;

                s_funct <= "100011";
                wait for 250 ns;

                s_funct <= "101011";
                wait for 250 ns;

                s_funct <= "000100";
                wait for 250 ns;
        end process;
end behav;
```

1.3 sign_ext.vhd

Το παρακάτω κύκλωμα υλοποιεί την μονάδα επέκτασης προσήμου. Στην αρχιτεκτονική της, ορίζουμε την έξοδο σε (`XXXX` συμβολίζει την είσοδο instruction):

- `0x0000XXXX` όταν το MSB του instruction ειναι 0.
- `0xffffXXXX` όταν το MSB είναι 1.

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

entity sign_ext is port (
        instr:          in std_logic_vector(15 downto 0);
        sign_ex:        out std_logic_vector(31 downto 0)
);
end sign_ext;

architecture dataflow of sign_ext is
begin
        -- Check the MSB to determine the sign.
        sign_ex <= x"0000" & instr when instr(15) = '0' else
                   x"ffff" & instr when instr(15) = '1';
end dataflow;
```

1.4 sign_ext_tb.vhd

Testbench για την μονάδα επέκτασης προσήμου.

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

entity sign_ext_tb is
end sign_ext_tb;

architecture behav of sign_ext_tb is

signal s_instr:         std_logic_vector(15 downto 0);
signal s_sign_ex:       std_logic_vector(31 downto 0);

component sign_ext is port (
        instr:          in std_logic_vector(15 downto 0);
        sign_ex:        out std_logic_vector(31 downto 0)
);
end component;

begin
        uut: sign_ext port map (
                instr => s_instr,
                sign_ex => s_sign_ex
        );

        process begin
                s_instr <= x"0010";
                wait for 250 ns;

                s_instr <= x"1001";
                wait for 250 ns;

                s_instr <= x"80a0";
                wait for 250 ns;
        end process;
end behav;
```

1.5 shl2.vhd

Το παρακάτω κύκλωμα υλοποιεί μία μονάδα αριστερής ολίσθησης κατά 2 bit. Η αρχιτεκτονική της είναι πολύ απλή: παίρνουμε τα 30 τελευταία bit της εισόδου και τους «κολλάμε» (concatenate) 2 μηδενικά bit στο τέλος.

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

entity shl2 is port (
        in1:    in std_logic_vector(31 downto 0);
        d:      out std_logic_vector(31 downto 0)
);
end shl2;

architecture dataflow of shl2 is
begin
        d <= in1(29 downto 0) & "00";
end dataflow;
```

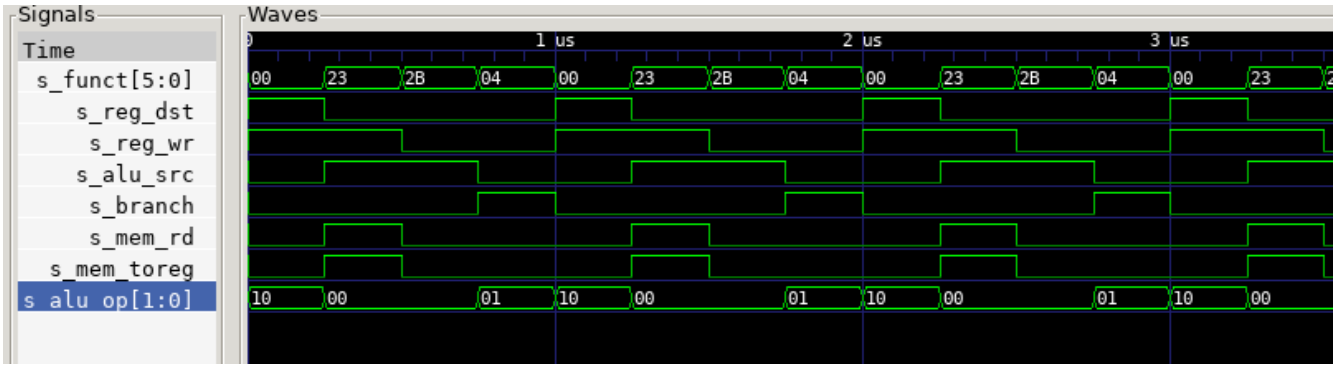
1.6 shl2_tb.vhd

Testbench για την μονάδα αριστερής ολίσθησης κατά 2 bit.

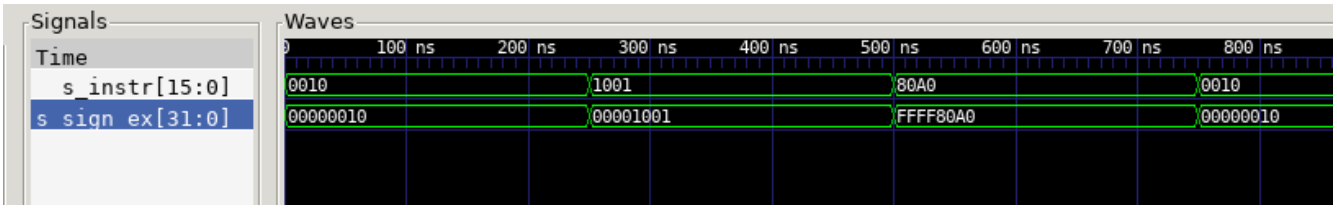
```
library ieee;
use ieee.std_logic_1164.all;

entity shl2_tb is
end shl2_tb;

architecture behav of shl2_tb is

signal s_in1:   std_logic_vector(31 downto 0);
signal s_d:     std_logic_vector(31 downto 0);

component shl2 is port (
        in1:    in std_logic_vector(31 downto 0);
        d:      out std_logic_vector(31 downto 0)
);
end component;

begin
        uut: shl2 port map (
                in1 => s_in1,
                d => s_d
        );

        process begin
                s_in1 <= x"0000aaaf";
                wait for 250 ns;

                s_in1 <= x"ffffaaaf";
                wait for 250 ns;
        end process;
end behav;
```

2 Εκτέλεση

2.1 ctrl_tb



2.2 sign_ext_tb



2.3 shl2_tb