Σχεδίαση Ψηφιακών Συστημάτων - Εργασία Θεωρίας (Μέρος 3)

Χρήστος Μαργιώλης

Ιούλιος 2020

Περιεχόμενα

1 Κώδικας και τεκμηρίωση

1.1 alu_ctrl.vhd

Το παρακάτω κύκλωμα υλοποιεί την μονάδα ελέγχου της ALU. Ο τρόπος υλοποιήσης της αρχικτεκτονικής προκύπτει από τους πίνακες λειτουργίας που υπάρχουνε στην εκφώνηση της άσκησης και στις διαφάνειες του μαθήματος. Το κύκλωμα θα μπορούσε να υλοποιηθεί εναλλακτικά χρησιμοποιώντας την δομή `with-select` ή την `case`.

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

entity alu_ctrl is port (
        funct:          in std_logic_vector(5 downto 0);
        alu_op:         in std_logic_vector(1 downto 0);
        op:             out std_logic_vector(3 downto 0)
);
end alu_ctrl;

architecture dataflow of alu_ctrl is
begin
        op <= "0010" when (alu_op = "00" or (alu_op = "10" and funct = "100000")) else
              "0110" when (alu_op = "01" or (alu_op = "10" and funct = "100010")) else
              "0000" when (alu_op = "10" and funct = "100100") else
              "0001" when (alu_op = "10" and funct = "100101") else
              "0111" when (alu_op = "10" and funct = "101010") else
              "1111";
end dataflow;
```

1.2 alu_ctrl_tb.vhd

Στο παρακάτω testbench δοκιμάζουμε την μονάδα ελέγχου ALU δίνοντας τις τιμές που υπάρχουνε στην εκφώνηση της άσκησης.

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

entity alu_ctrl_tb is
end alu_ctrl_tb;

architecture behav of alu_ctrl_tb is

signal s_funct:         std_logic_vector(5 downto 0);
signal s_alu_op:        std_logic_vector(1 downto 0);
signal s_op:            std_logic_vector(3 downto 0);

component alu_ctrl is port (
        funct:          in std_logic_vector(5 downto 0);
        alu_op:         in std_logic_vector(1 downto 0);
        op:             out std_logic_vector(3 downto 0)
);
end component;

begin
        uut: alu_ctrl port map (
                funct => s_funct,
                alu_op => s_alu_op,
                op => s_op
        );

        process begin
                s_alu_op <= "00";
                s_funct <= "001001";
                wait for 250 ns;

                s_alu_op <= "00";
                s_funct <= "001010";
                wait for 250 ns;

                s_alu_op <= "01";
                s_funct <= "100111";
                wait for 250 ns;

                s_alu_op <= "10";
                s_funct <= "100000";
                wait for 250 ns;

                s_alu_op <= "10";
                s_funct <= "100010";
                wait for 250 ns;

                s_alu_op <= "10";
                s_funct <= "100100";
                wait for 250 ns;

                s_alu_op <= "10";
                s_funct <= "100101";
                wait for 250 ns;

                s_alu_op <= "10";
                s_funct <= "101010";
                wait for 250 ns;
        end process;
end behav;
```

1.3 alu_ctrl_test_alu.vhd

Το παρακάτω κύκλωμα υλοποιεί ένα «δοκιμαστικό» κύκλωμα για την ALU και την μονάδα ελέγχου της. Το μόνο που χρειάζεται είναι απλώς να δηλώσουμε ως components την ALU που δημιουργήθηκε στο μέρος 1 και το προηγούμενο κύκλωμα, και να τα κάνουμε map στα κατάλληλα πεδία του entity του κυκλώματος. Αυτή τη φορά, η αρχιτεκτονική θα είναι structural.

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

entity alu_ctrl_test_alu is generic (
        t_sz:           natural := 4
);
port (
        t_funct:        in std_logic_vector(5 downto 0);
        t_alu_op:       in std_logic_vector(1 downto 0);
        t_alu_in1:      in std_logic_vector(t_sz-1 downto 0);
        t_alu_in2:      in std_logic_vector(t_sz-1 downto 0);
        t_alu_out:      out std_logic_vector(t_sz-1 downto 0);
        t_alu_zero:     out std_logic
);
end alu_ctrl_test_alu;

architecture struct of alu_ctrl_test_alu is

signal s_ctrl:          std_logic_vector(3 downto 0);

component alu_ctrl is port (
        funct:          in std_logic_vector(5 downto 0);
        alu_op:         in std_logic_vector(1 downto 0);
        op:             out std_logic_vector(3 downto 0)
);
end component;

component alu is generic (
        sz:             natural := 4
);
port (
        alu_in1:        in std_logic_vector(sz-1 downto 0);
        alu_in2:        in std_logic_vector(sz-1 downto 0);
        alu_ctrl:       in std_logic_vector(3 downto 0);
        alu_out:        out std_logic_vector(sz-1 downto 0);
        alu_zero:       out std_logic
);
end component;
```

```
begin
        uut_alu_ctrl: alu_ctrl port map (
                funct => t_funct,
                alu_op => t_alu_op,
                op => s_ctrl
        );

        uut_alu: alu port map (
                alu_in1 => t_alu_in1,
                alu_in2 => t_alu_in2,
                alu_ctrl => s_ctrl,
                alu_out => t_alu_out,
                alu_zero => t_alu_zero
        );
end struct;
```

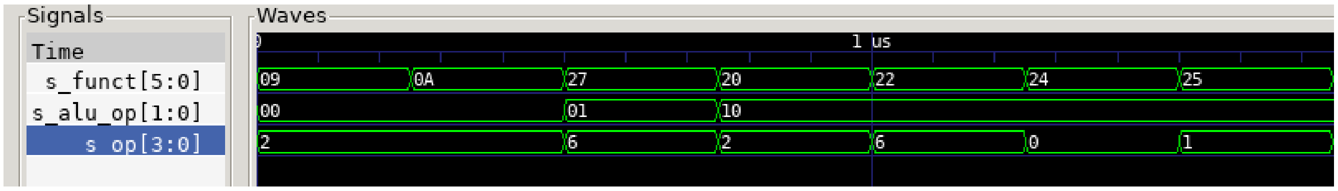
1.4 alu_ctrl_test_alu_tb.vhd

Testbench για το παραπάνω κύκλωμα.

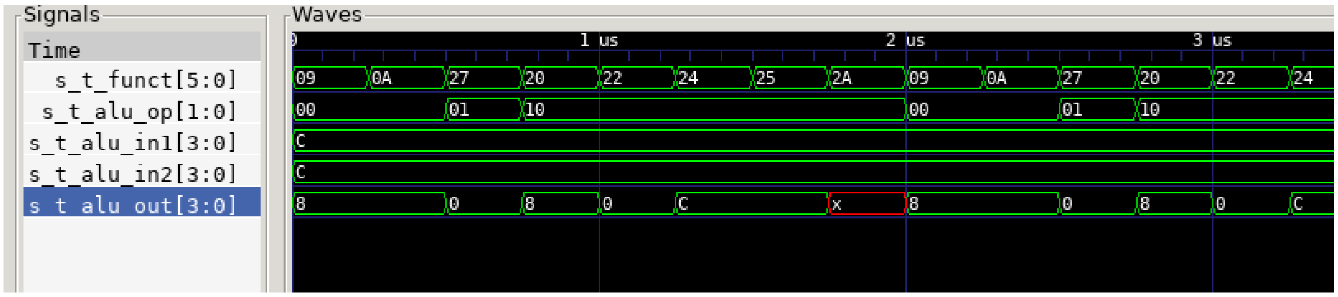
```
library ieee;
use ieee.std_logic_1164.all;

entity alu_ctrl_test_alu_tb is
end alu_ctrl_test_alu_tb;

architecture behav of alu_ctrl_test_alu_tb is

signal s_sz:            natural := 4;
signal s_t_funct:       std_logic_vector(5 downto 0);
signal s_t_alu_op:      std_logic_vector(1 downto 0);
signal s_t_alu_in1:     std_logic_vector(s_sz-1 downto 0);
signal s_t_alu_in2:     std_logic_vector(s_sz-1 downto 0);
signal s_t_alu_out:     std_logic_vector(s_sz-1 downto 0);
signal s_t_alu_zero:    std_logic;

component alu_ctrl_test_alu is generic (
        t_sz:           natural := 4
);
port (
        t_funct:        in std_logic_vector(5 downto 0);
        t_alu_op:       in std_logic_vector(1 downto 0);
        t_alu_in1:      in std_logic_vector(t_sz-1 downto 0);
        t_alu_in2:      in std_logic_vector(t_sz-1 downto 0);
        t_alu_out:      out std_logic_vector(t_sz-1 downto 0);
        t_alu_zero:     out std_logic
);
end component;

begin
        uut: alu_ctrl_test_alu port map (
                t_funct => s_t_funct,
                t_alu_op => s_t_alu_op,
                t_alu_in1 => s_t_alu_in1,
                t_alu_in2 => s_t_alu_in2,
                t_alu_out => s_t_alu_out,
                t_alu_zero => s_t_alu_zero
        );

        process begin
                s_t_alu_in1 <= "1100";
                s_t_alu_in2 <= "1100";
                s_t_alu_op <= "00";
                s_t_funct <= "001001";
                wait for 250 ns;

                s_t_alu_op <= "00";
                s_t_funct <= "001010";
                wait for 250 ns;

                s_t_alu_op <= "01";
                s_t_funct <= "100111";
                wait for 250 ns;

                s_t_alu_op <= "10";
                s_t_funct <= "100000";
                wait for 250 ns;

                s_t_alu_op <= "10";
                s_t_funct <= "100010";
                wait for 250 ns;

                s_t_alu_op <= "10";
                s_t_funct <= "100100";
                wait for 250 ns;

                s_t_alu_op <= "10";
                s_t_funct <= "100101";
                wait for 250 ns;

                s_t_alu_op <= "10";
                s_t_funct <= "101010";
                wait for 250 ns;
        end process;
end behav;
```

2 Εκτέλεση

2.1 alu_ctrl_tb



2.2 alu_ctrl_test_alu_tb


Signals
Waves
Time
s_t_funct[5:0]
s_t_alu_op[1:0]
s_t_alu_in1[3:0]
s_t_alu_in2[3:0]
s_t_alu_out[3:0]
1 us
2 us
3 us
09 0A 27 20 22 24 25 2A 09 0A 27 20 22 24
00 01 10 00 01 10
C
C
8 0 8 0 C x 8 0 8 0 C