# Σχεδίαση Ψηφιακών Συστημάτων - Εργασία Θεωρίας (Μέρος 2)

Χρήστος Μαργιώλης

Ιούλιος 2020

# Περιεχόμενα



# 1 Κώδικας και τεκμηρίωση

## 1.1 alu.vhd

Το παρακάτω κύκλωμα υλοποιεί την αριθμητική και λογική μονάδα ενός επεξεργαστή. Οι πράξεις που μπορεί να εκτελέσει το κύκλωμα ειναι τα λογικά AND και OR, και η αριθμητική πρόσθεση και αφαίρεση. Στην αρχιτεκτονική ορίζουμε ένα process το οποίο είναι «ευαίσθητο» στο σήμα της μονάδας ελέγχου ALU. 'Οταν εισέλθουμε στο process, εκτελούμε την κατάλληλη πράξη ανάλογα με το σήμα που δώθηκε. Για την αποθήκευση του αποτελέσματος χρησιμοποιούμε ένα ενδιάμεσο σήμα `sig`. Τέλος, ορίζουμε στο `alu_zero` την τιμή 1 όταν το αποτέλεσμα είναι 0000, ειδάλλως 0, και επίσης θέτουμε το απότελεσμα στην έξοδο `alu_out`.

Μία αλλαγή που έκανα στο κύκλωμα είναι να θέσω μία generic τιμή για να μπορεί η ALU να μετατραπεί εύκολα από 4 bit σε 32 bit, ωστέ να αποφευχθεί η αντιγραφή κώδικα.

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

entity 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 alu;

architecture behav of alu is
signal sig:             std_logic_vector(sz-1 downto 0);

begin
        process (alu_ctrl) begin
                case alu_ctrl is
                when "0000" => sig <= alu_in1 and alu_in2;
                when "0001" => sig <= alu_in1 or alu_in2;
                when "0010" =>
```

```
                sig <= std_logic_vector(signed(alu_in1) + signed(alu_in2));
            when "0110" =>
                sig <= std_logic_vector(signed(alu_in1) - signed(alu_in2));
            when others => sig <= (others => 'X');
            end case;
        end process;
        alu_zero <= '1' when sig = "0000" else '0';
        alu_out <= sig;
end behav;
```

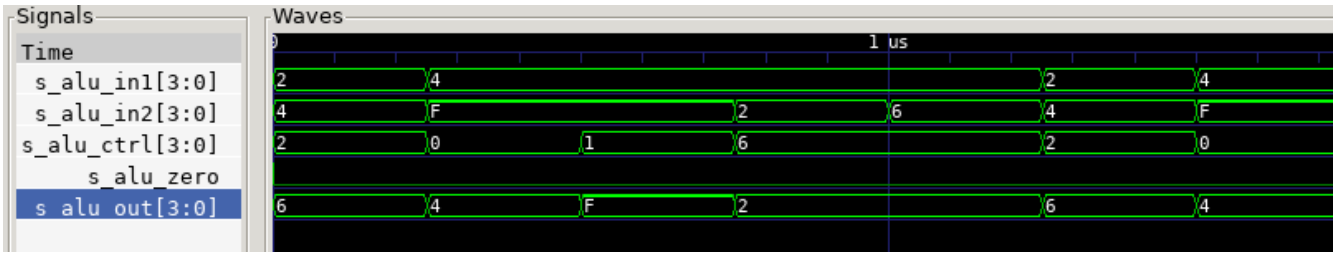
## 1.2 alu_tb.vhd

Στο παρακάτω testbench δοκιμάζουμε την λειτουργία της ALU. Η λογική για την δημιουργία του testbench είναι ίδια με αυτή που εξηγήθηκε στο μέρος 0. Οι τιμές για την δοκιμή δώθηκαν από την εκφώνηση της άσκησης.

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

entity alu_tb is
end alu_tb;

architecture behav of alu_tb is

signal s_sz:            natural := 4;
signal s_alu_in1:       std_logic_vector(s_sz-1 downto 0);
signal s_alu_in2:       std_logic_vector(s_sz-1 downto 0);
signal s_alu_ctrl:      std_logic_vector(3 downto 0);
signal s_alu_out:       std_logic_vector(s_sz-1 downto 0);
signal s_alu_zero:      std_logic;

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 port map (
                alu_in1 => s_alu_in1,
                alu_in2 => s_alu_in2,
                alu_ctrl => s_alu_ctrl,
                alu_out => s_alu_out,
                alu_zero => s_alu_zero
        );

        process begin
                s_alu_in1 <= "0010";
                s_alu_in2 <= "0100";
                s_alu_ctrl <= "0010";
```

```
                wait for 250 ns;

                s_alu_in1 <= "0100";
                s_alu_in2 <= "1111";
                s_alu_ctrl <= "0000";
                wait for 250 ns;

                s_alu_in1 <= "0100";
                s_alu_in2 <= "1111";
                s_alu_ctrl <= "0001";
                wait for 250 ns;

                s_alu_in1 <= "0100";
                s_alu_in2 <= "0010";
                s_alu_ctrl <= "0110";
                wait for 250 ns;

                s_alu_in1 <= "0100";
                s_alu_in2 <= "0110";
                s_alu_ctrl <= "0110";
                wait for 250 ns;
        end process;
end behav;
```

## 2 Εκτέλεση


Signals
Waves
Time
s_alu_in1[3:0]
s_alu_in2[3:0]
s_alu_ctrl[3:0]
s_alu_zero
s_alu_out[3:0]
1 us
2 4 2 4
4 F 2 6 4 F
2 0 1 6 2 0
6 4 F 2 6 4