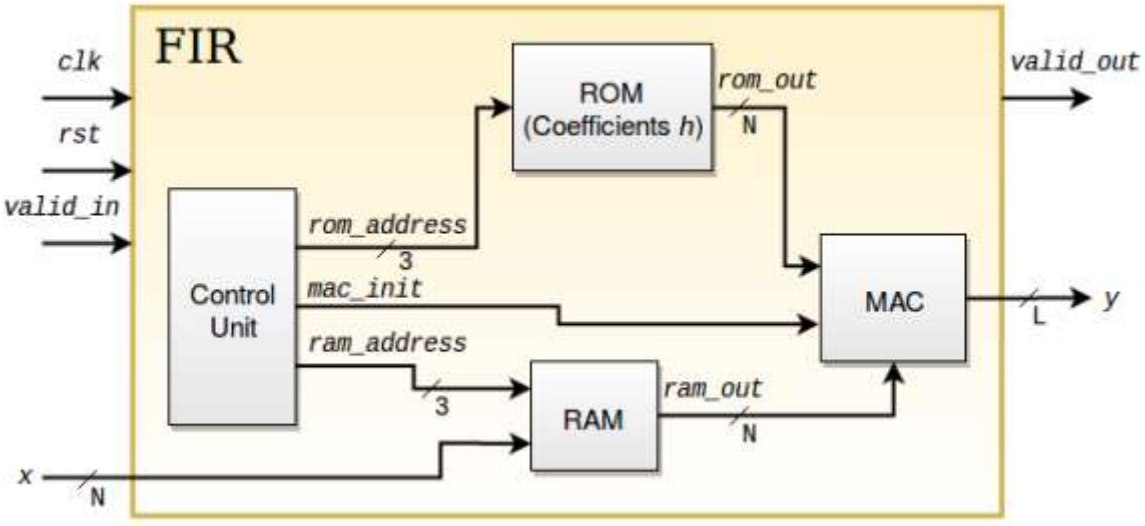
4η εργαστηριακή άσκηση

Φωτεινή Κυριακοπούλου 03120182
Ιωάννης Μπουφίδης 03120162

FIR Φίλτρο

Για την περιγραφή του φίλτρου βασιστήκαμε στο παρακάτω σχήμα:



Προκειμένου να περιγράψουμε το παραπάνω κύκλωμα με Structural αρχιτεκτονική, δημιουργήσαμε τις εξής οντότητες:

- **D Flip-Flop** (καθυστέρησης 1 κύκλου)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity d1 is
    Port(
          Clk, reset, D: in std_logic;
          Q: out std_logic
         );
end d1;

architecture Behavioral of d1 is
    signal rst: std_logic;
begin
    process(Clk, reset)
    begin
        if reset='1' then rst <= '1';
        elsif (Clk'event and Clk='1') then
            if rst='1' then Q <= '0'; rst <= '0';
            else Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

4η εργαστηριακή άσκηση

- **Control Unit - Μονάδα Ελέγχου**

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

entity control_unit is
    port (
        clk : in std_logic;
        reset : in std_logic;
        valid_in: in std_logic;
        mac_init : out std_logic;
        we, en: out std_logic;
        rom_address : out std_logic_vector(2 downto 0);
        ram_address : out std_logic_vector(2 downto 0)
    );
end entity control_unit;

architecture control_unit_arch of control_unit is
    signal counter : std_logic_vector(2 downto 0):="000";  -- Counter for ROM and RAM
begin
    process (clk, reset)
    begin
        if reset = '1' then
            counter <= "000"; -- Reset counter
        elsif rising_edge(clk) then
            if counter="000" then
                if valid_in='1' then
                    mac_init <= '1'; -- Initialize mac unit
                    we <= '1'; en <= '1'; -- Write x to RAM[0]
                    rom_address <= std_logic_vector(counter);
                    ram_address <= std_logic_vector(counter);
                    counter <= counter + 1;
                else
                    we <= '0'; en <= '0'; -- Input is invalid, disable access to RAM
                    mac_init <= '1';
                end if;
             else
                we <= '0'; en <= '1';  -- Read RAM[1-7]
                mac_init <= '0';
                rom_address <= std_logic_vector(counter);
                ram_address <= std_logic_vector(counter);
                counter <= counter + 1;
            end if;
        end if;
    end process;
end architecture control_unit_arch;
```

4η εργαστηριακή άσκηση

- **Μνήμη ROM**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;


entity mlab_rom is
        generic (
              coeff_width : integer :=8      --- width of coefficients (bits)
        );
    Port ( clk : in  STD_LOGIC;
           en : in  STD_LOGIC;                                    --- operation enable
           addr : in  STD_LOGIC_VECTOR (2 downto 0);              -- memory address
           rom_out : out  STD_LOGIC_VECTOR (coeff_width-1 downto 0)); -- output data
end mlab_rom;

architecture Behavioral of mlab_rom is

    type rom_type is array (7 downto 0) of std_logic_vector (coeff_width-1 downto 0);
    signal rom : rom_type:= ("00001000", "00000111", "00000110", "00000101",
                             "00000100", "00000011", "00000010", "00000001");
      -- initialization of rom with user data

    signal rdata : std_logic_vector(coeff_width-1 downto 0) := (others => '0');
begin

    rdata <= rom(conv_integer(addr));

    process (clk)
    begin
        if (clk'event and clk = '1') then
            if (en = '1') then
                rom_out <= rdata;
            end if;
        end if;
    end process;


end Behavioral;
```

4η εργαστηριακή άσκηση

- **Μνήμη RAM**

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

entity mlab_ram is
     generic (
        data_width : integer :=8                    --- width of data (bits)
     );
    port (clk  : in std_logic;
          reset: in std_logic;
          we   : in std_logic;                          --- memory write enable
          en   : in std_logic;                          --- operation enable
          addr : in std_logic_vector(2 downto 0);            -- memory address
          di   : in std_logic_vector(data_width-1 downto 0);        -- input data
          do   : out std_logic_vector(data_width-1 downto 0));        -- output data
end mlab_ram;

architecture Behavioral of mlab_ram is

    type ram_type is array (7 downto 0) of std_logic_vector (data_width-1 downto 0);
    signal RAM : ram_type := (others => (others => '0'));

begin


    process (clk, reset)
    begin
        if reset='1' then – Initialize RAM with all zeroes
            for i in 7 downto 0 loop
                RAM(i) <= "00000000";
            end loop;
        elsif clk'event and clk = '1' then
            if en = '1' then
                if we = '1' then                -- write operation
                  for i in 6 downto 0 loop
                        RAM(i+1) <= RAM(i);      -- shift previous RAM entries
                  end loop;
                  -- Write new data to RAM at address 0
                  RAM(0) <= di;
                  do <= di;
                else                          -- read operation
                    do <= RAM(conv_integer(addr));
                end if;
            end if;
        end if;
    end process;

end Behavioral;
```

4η εργαστηριακή άσκηση

- **MAC - Μονάδα Πολλαπλασιασμού με Συσσώρευση**

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

entity mac_unit is
    port (
        clk : in std_logic;
        rom_out : in std_logic_vector(7 downto 0); -- Coefficients from ROM
        ram_out : in std_logic_vector(7 downto 0); -- Input data from RAM
        mac_init : in std_logic;                   -- Initialization signal
        valid_out: out std_logic := '0';
        y_out : inout std_logic_vector(18 downto 0) -- Output
    );
end entity mac_unit;

architecture mac_unit_arch of mac_unit is
 signal sum: std_logic_vector(18 downto 0);
begin
    process (clk)
    begin
        if rising_edge(clk) then
            if mac_init = '1' then
                sum <= "000" & rom_out*ram_out;
                y_out <= sum;
                if sum/="U" then valid_out <= '1'; end if;

            elsif mac_init = '0' then
                sum <= sum + ram_out*rom_out; -- Add to partial sum computed product
                valid_out <= '0';

            end if;
        end if;
    end process;
end mac_unit_arch;
```

4η εργαστηριακή άσκηση

Χρησιμοποιώντας τις παραπάνω οντότητες και προσθέτοντας τις κατάλληλες καθυστέρησεις όπου χρειαζόταν (πχ mac_init από το Control Unit στη MAC), προκύπτει η εξής περιγραφή του FIR φίλτρου:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fir_filter is
  port (
        clk, reset, valid_in: in std_logic;
        x: in std_logic_vector(7 downto 0);      -- Filter input
        valid_out: out std_logic;
        y: inout std_logic_vector(18 downto 0)  -- Filter output
        );
end fir_filter;

architecture fir_filter_arch of fir_filter is
    component mac_unit is
    port (
        clk : in std_logic;
        rom_out : in std_logic_vector(7 downto 0);
        ram_out : in std_logic_vector(7 downto 0);
        mac_init : in std_logic;
        valid_out: out std_logic;
        y_out : inout std_logic_vector(18 downto 0)
    );
    end component;

    component control_unit is
    port (
        clk : in std_logic;
        reset : in std_logic;
        valid_in: in std_logic;
        mac_init : out std_logic;
        we, en: out std_logic;
        rom_address : out std_logic_vector(2 downto 0);
        ram_address : out std_logic_vector(2 downto 0)
    );
    end component;

    component mlab_ram is
    generic (
        data_width : integer :=8
    );
    port (clk  : in std_logic;
          reset: in std_logic;
          we   : in std_logic;
          en   : in std_logic;
          addr : in std_logic_vector(2 downto 0);
          di   : in std_logic_vector(data_width-1 downto 0);
```

```
            do   : out std_logic_vector(data_width-1 downto 0));
    end component;

    component mlab_rom is
    generic (
        coeff_width : integer :=8
     );
    port ( clk : in  STD_LOGIC;
           en : in  STD_LOGIC;
           addr : in  STD_LOGIC_VECTOR (2 downto 0);
           rom_out : out  STD_LOGIC_VECTOR (coeff_width-1 downto 0));
    end component;

    component D1 is
    port (
          reset, Clk, D: in std_logic;
          Q: out std_logic
         );
    end component;

    signal rom_out, ram_out: std_logic_vector(7 downto 0);
    signal rom_ad, ram_ad: std_logic_vector(2 downto 0);
    signal mac_i, mac_ii, we, en: std_logic;
begin
    control: control_unit port map(clk => clk, reset => reset, valid_in => valid_in,
                                   mac_init => mac_i, rom_address => rom_ad,
                                   ram_address => ram_ad, we => we, en => en);
    mac_in_delay: D1 port map (clk => clk, D => mac_i, Q => mac_ii, reset => '0');
    -- Delay mac_init signal one cycle to synchronize operations
    ram: mlab_ram generic map(data_width => 8)
         port map(clk => clk, reset => reset,
                  we => we, en => en, addr => ram_ad, di => x, do => ram_out);
    rom: mlab_rom generic map(coeff_width => 8)
         port map(clk => clk, en => en, addr => rom_ad, rom_out => rom_out);
    mac: mac_unit port map (clk => clk, rom_out => rom_out, ram_out => ram_out,
                            valid_out => valid_out, mac_init => mac_ii, y_out => y);
end fir_filter_arch;
```

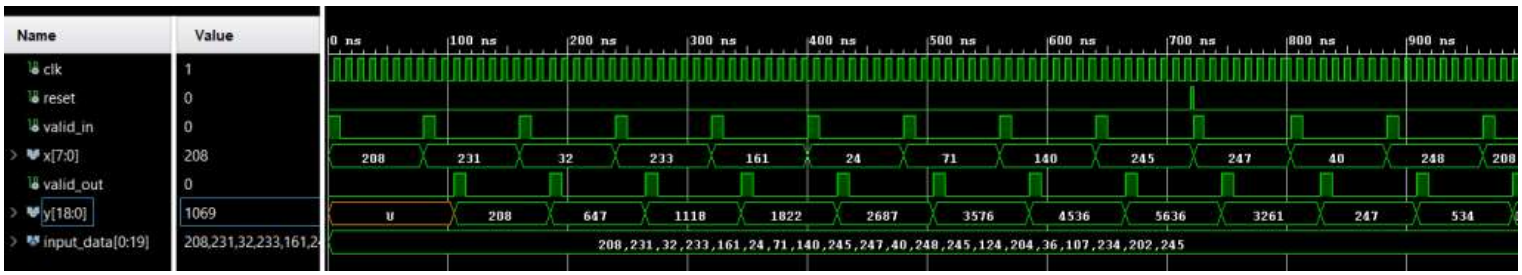
4η εργαστηριακή άσκηση

Προκειμένου να ελέγξουμε την σωστή λειτουργία του κυκλώματος, γράφουμε το εξής testbench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity fir_filter_tb is
end fir_filter_tb;

architecture tb_arch of fir_filter_tb is
    signal clk, reset, valid_in : std_logic;
    signal x : std_logic_vector(7 downto 0);
    signal valid_out : std_logic;
    signal y : std_logic_vector(18 downto 0);

    -- Component declaration for the DUT
    component fir_filter is
        port (
            clk, reset, valid_in: in std_logic;
            x: in std_logic_vector(7 downto 0);
            valid_out: out std_logic;
            y: inout std_logic_vector(18 downto 0)
        );
    end component;

    type input_array is array (0 to 19) of integer range 0 to 255;
    constant input_data : input_array := (208, 231, 32, 233, 161, 24, 71, 140, 245,
                                          247, 40, 248, 245, 124, 204, 36, 107, 234,
                                          202, 245);
begin
    -- DUT instantiation
    DUT: fir_filter port map (
        clk => clk,
        reset => reset,
        valid_in => valid_in,
        x => x,
        valid_out => valid_out,
        y => y
    );
    check: process
    begin

    for i in 0 to 8 loop
        x <= std_logic_vector(to_unsigned(input_data(i), x'length));
        valid_in <= '1';
        clk <= '0';  wait for 5 ns; clk <= '1';  wait for 5 ns;
        valid_in <= '0';
        for i in 6 downto 0 loop
            clk <= '0';  wait for 5 ns;
            clk <= '1';  wait for 5 ns;
        end loop;
    end loop;
```

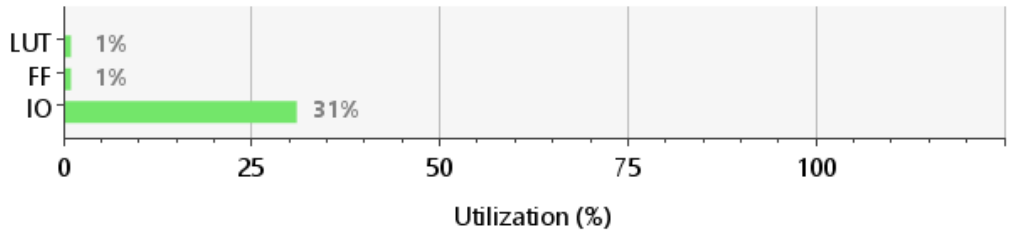
```
    clk <= '0'; reset <= '1'; wait for 2 ns; reset <= '0';  -- Reset filter
    x <= std_logic_vector(to_unsigned(input_data(9), x'length));
    valid_in <= '1';
    wait for 3 ns; clk <= '1';  wait for 5 ns;
    valid_in <= '0';
    for i in 6 downto 0 loop
        clk <= '0';  wait for 5 ns;
        clk <= '1';  wait for 5 ns;
    end loop; wait for 3 ns;

    for i in 10 to 11 loop
        x <= std_logic_vector(to_unsigned(input_data(i), x'length));
        valid_in <= '1';
        clk <= '0';  wait for 5 ns; clk <= '1';  wait for 5 ns;
        valid_in <= '0';
        for i in 6 downto 0 loop
            clk <= '0';  wait for 5 ns;
            clk <= '1';  wait for 5 ns;
        end loop;
    end loop;

end process;
end tb_arch;
```

→ Το αποτέλεσμα της προσομοίωσης φαίνεται στο παρακάτω σχήμα:



→ Οι πόροι που χρησιμοποιήθηκαν και η ισχύς που καταναλώνεται:

Resource	Utilization	Available	Utilization %
LUT	99	17600	0.56
FF	124	35200	0.35
IO	31	100	31.00

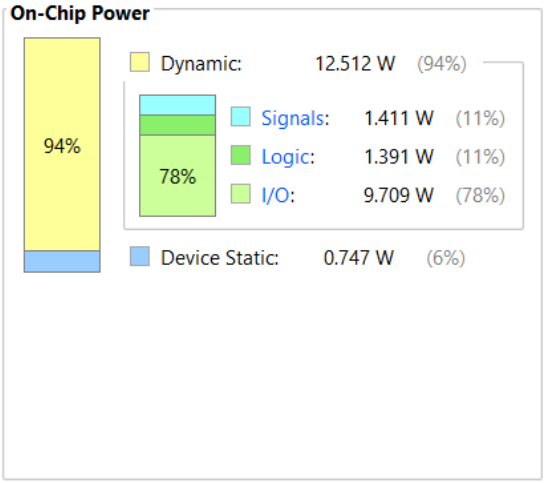


4η εργαστηριακή άσκηση

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**13.259 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**125,0°C**
Thermal Margin:	-92,9°C (-7,5 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



→ Το κρίσιμο μονοπάτι είναι από την έξοδο της ROM μέχρι τον καταχωρητή που αποθηκεύουμε το επιμέρους άθροισμα σε κάθε κύκλο

→ Το κρίσιμο μονοπάτι παρουσιάζει καθυστέρηση 6.234 ns, επομένως η συχνότητα λειτουργίας του FPGA είναι $\frac{1}{6.234 \times 10^{-9}}$ = 160.42 MHz