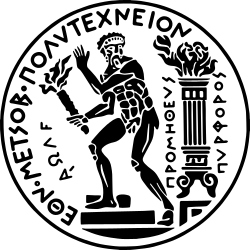
**Εθνικό Μετσόβιο Πολυτεχνείο** 

**Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών**

**Υπολογιστών**

Έτος: 2024- 2025

*Ψηφιακά VLSI*

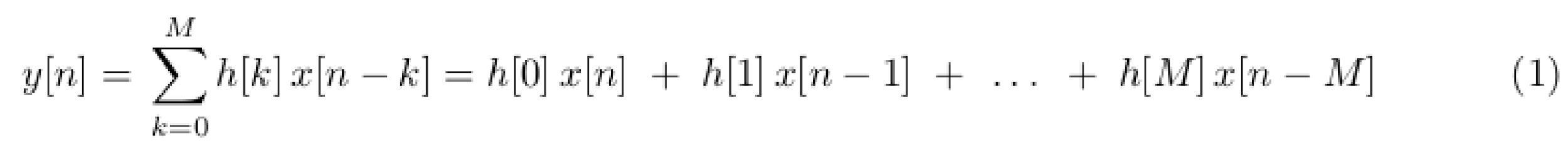
**Ομάδα 45**

*Παπαδόπουλος Χαράλαμπος* (ΑΜ: 03120199)

*Νικόλαος Παπακωνσταντόπουλος* (ΑΜ: 03120069)

# **Υλοποίηση Φίλτρου FIR**

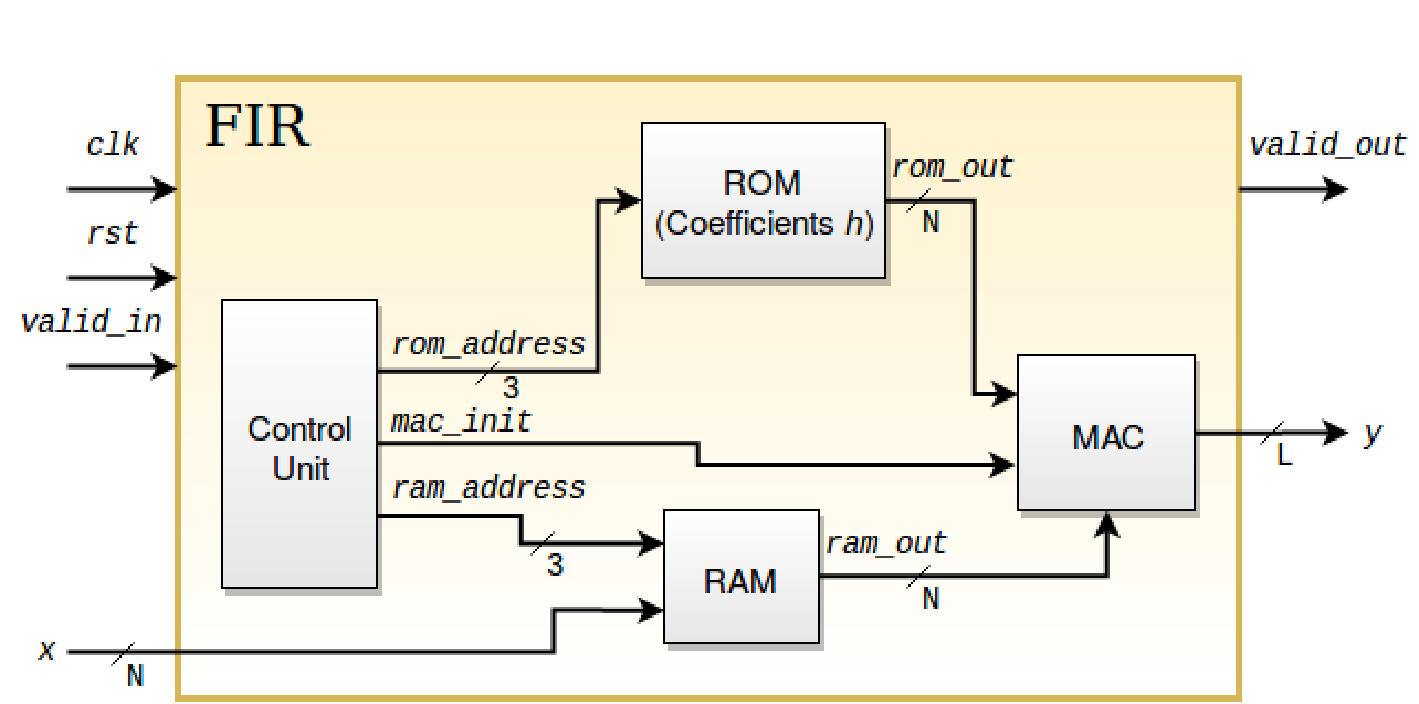
Στη γενική περίπτωση, η σχέση εισόδου - εξόδου ενός FIR φίλτρου είναι η ακόλουθη:



Όπου

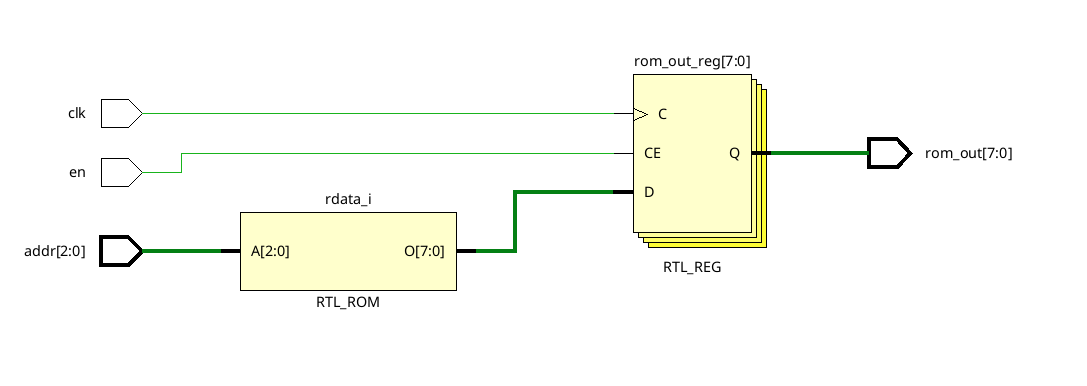
* M είναι η τάξη του φίλτρου
* y[n] είναι η έξοδος του φίλτρου τη διακριτή χρονική στιγμή n
* h[k] είναι ο k-οστός συντελεστής του φίλτρου, με k = 0, 1, 2, ..., M
* x[n] είναι η τιμή του σήματος εισόδου τη διακριτή χρονική στιγμή n

Στην συγκεκριμένη άσκηση μας ζητήθηκε η υλοποίηση ενός 8-tap FIR για N=8 bits εύρος δεδομένων Χ σύμφωνα με την ακόλουθη αρχιτεκτονική.

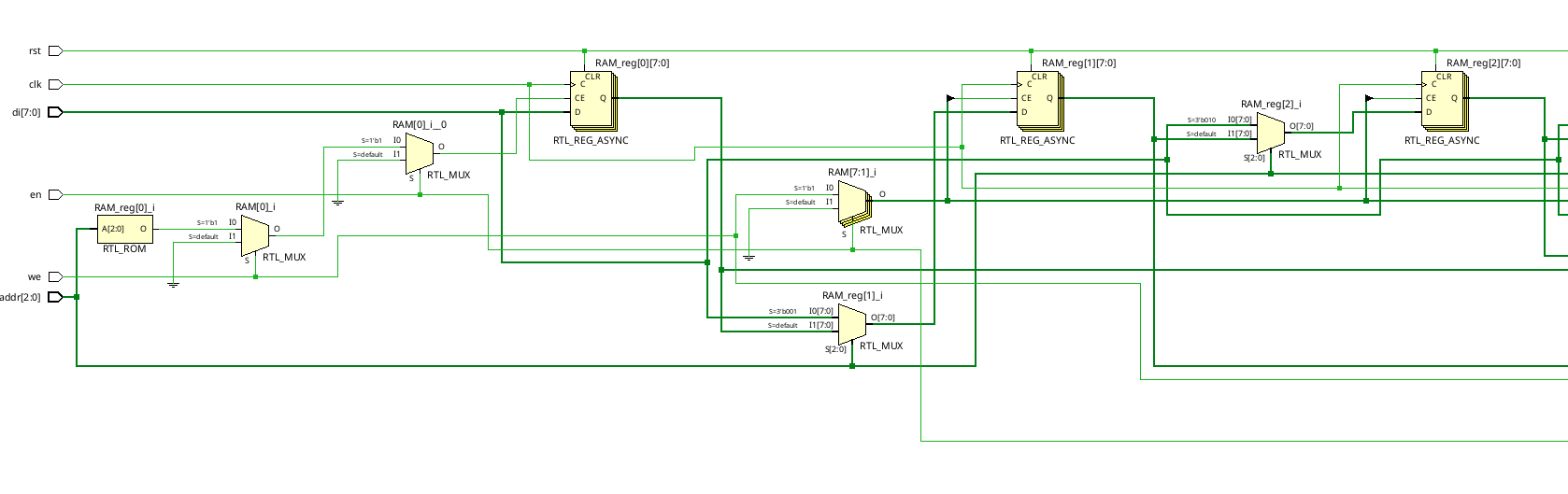


Ακολουθούν οι κώδικες μαζί με τα RTL της κάθε δομικής μονάδας.

| ROM |
| --- |
| **library** **IEEE**;  **use** **IEEE.STD\_LOGIC\_1164.ALL**;  **use** **ieee.std\_logic\_unsigned.all**;  **entity** **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** **rom**;  **architecture** **Behavioral** **of** **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**; |

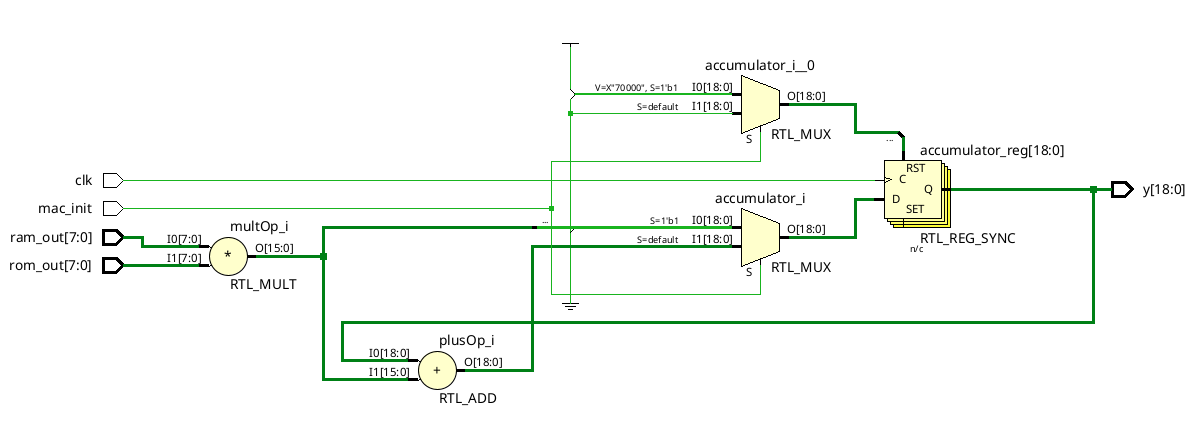


| RAM |
| --- |
| **library** **ieee**;  **use** **ieee.std\_logic\_1164.all**;  **use** **ieee.std\_logic\_unsigned.all**;  **entity** **ram** **is**  **generic** (  data\_width : **integer** :=**8** --- width of data (bits)  );  **port** (clk : **in** **std\_logic**;  rst : **in** **std\_logic**;  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** **ram**;  **architecture** **Behavioral** **of** **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')); -- initialize at 0  **begin**  **process** (clk, rst)  **begin**  **if** rst = '1' **then**  RAM <= (**others** => (**others** => '0')); -- reset to 0  **else**  **if** clk'event **and** clk = '1' **then**  **if** en = '1' **then**  **if** we = '1' **then** -- write operation  RAM(**7** **downto** **1**) <= RAM(**6** **downto** **0**);  RAM(conv\_integer(addr)) <= di;  do <= di;  **else** -- read operation  do <= RAM(conv\_integer(addr));  **end** **if**;  **end** **if**;  **end** **if**;  **end** **if**;  **end** **process**;  **end** **Behavioral**; |

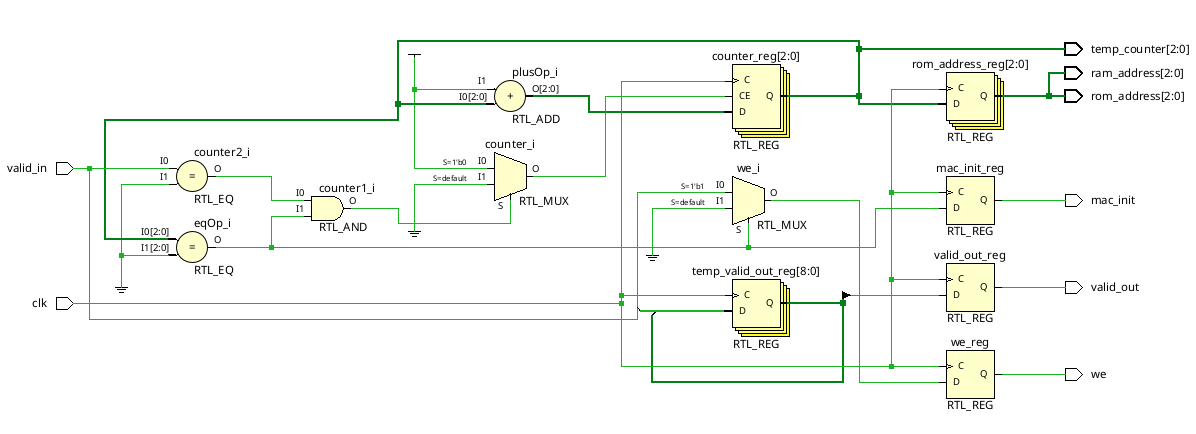


Περικομμένο RTL. Κανονικά έχει κι άλλα (πανομοιότυπα) στάδια

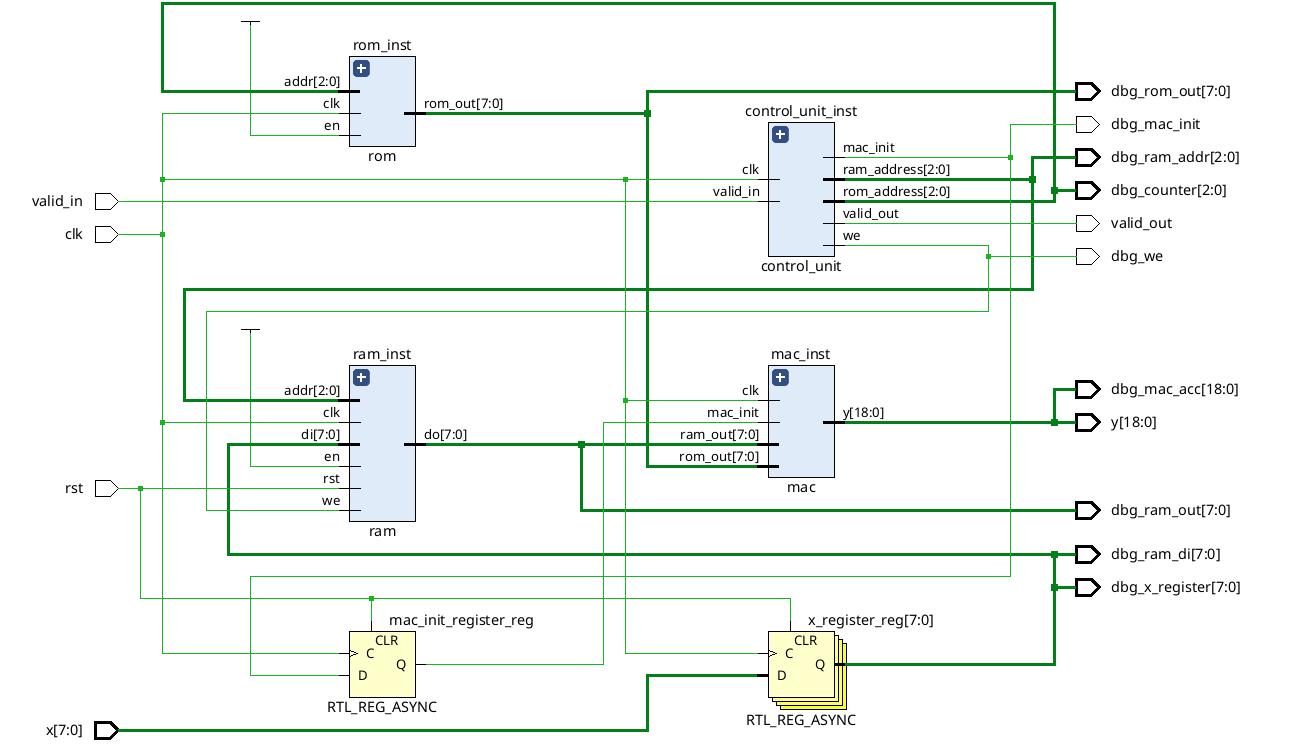
| MAC |
| --- |
| **library** **ieee**;  **use** **ieee.std\_logic\_1164.all**;  **use** **ieee.std\_logic\_unsigned.all**;  **entity** **mac** **is**  **Port** (  clk : **in** **std\_logic**;  mac\_init : **in** **std\_logic**;  rom\_out : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  ram\_out : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  y : **out** **std\_logic\_vector**(**19**-**1** **downto** **0**)  );  **end** **mac**;  **architecture** **behavioral** **of** **mac** **is**  **signal** accumulator : **std\_logic\_vector**(**19**-**1** **downto** **0**) := (**others** => '0');  **begin**  **process**(clk)  **begin**  **if** rising\_edge(clk) **then**  **if** mac\_init = '1' **then**  accumulator <= "000" & (ram\_out \* rom\_out);  **else**  accumulator <= accumulator + (ram\_out \* rom\_out);  **end** **if**;  **end** **if**;  **end** **process**;  y <= accumulator;  **end** **behavioral**; |



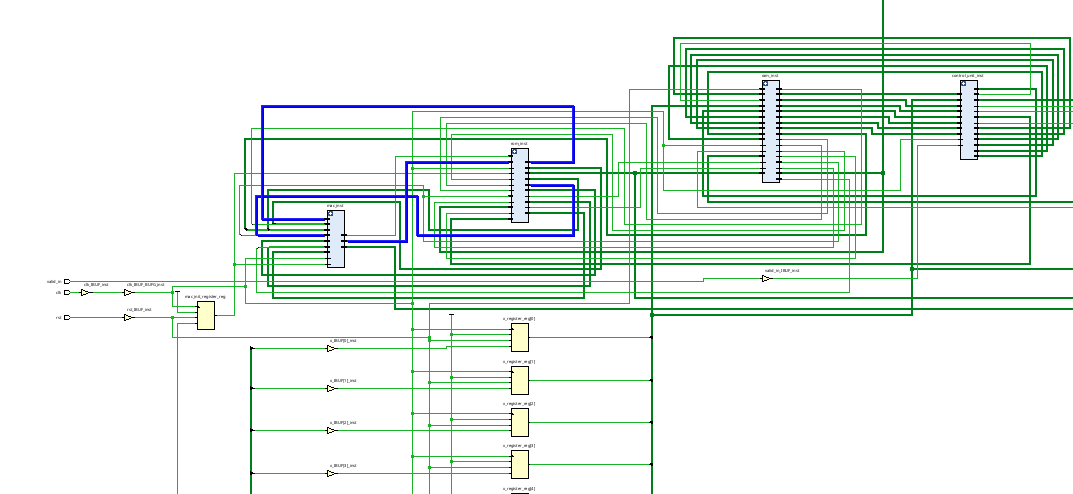
| Control Unit |
| --- |
| **library** **IEEE**;  **use** **IEEE.std\_logic\_1164.ALL**;  **use** **IEEE.std\_logic\_unsigned.ALL**;  **use** **ieee.numeric\_std.all**;  **entity** **control\_unit** **is**  **Port** (  clk : **in** **std\_logic**;  valid\_in : **in** **std\_logic**;  rom\_address : **out** **std\_logic\_vector**(**3**-**1** **downto** **0**);  ram\_address : **out** **std\_logic\_vector**(**3**-**1** **downto** **0**);  mac\_init : **out** **std\_logic**;  we : **out** **std\_logic**; -- write enable for ram  valid\_out : **out** **std\_logic**;  temp\_counter : **out** **std\_logic\_vector**(**2** **downto** **0**)  );  **end** **control\_unit**;  **architecture** **Behavioral** **of** **control\_unit** **is**  **signal** counter : **std\_logic\_vector**(**2** **downto** **0**):= (**others** => '0');  **signal** temp\_valid\_out : **std\_logic\_vector**(**9**-**1** **downto** **0**) := (**others** => '0');  **begin**  **process**(clk)  **begin**  **if** rising\_edge(clk) **then**  **if** counter = "000" **then**  we <= valid\_in;  mac\_init <= '1';  temp\_valid\_out(**0**) <= valid\_in;  **else**  temp\_valid\_out(**0**) <= valid\_in;  mac\_init <= '0';  we <= '0';  **end** **if**;  rom\_address <= counter;  ram\_address <= counter;  -- increment the counter only if valid\_in is high so we write a new value to the ram  -- or if valid\_in is low and the counter is not zero (so we can read the ram)  **if** **not** (valid\_in = '0' **and** counter = "000" ) **then**  counter <= **std\_logic\_vector**(unsigned(counter) + **1**);  **end** **if**;  valid\_out <= temp\_valid\_out(**8**);  **end** **if**;  **end** **process**;  **process**(clk)  **begin**  **if** rising\_edge(clk) **then**  temp\_valid\_out(**8** **downto** **1**) <= temp\_valid\_out(**7** **downto** **0**);  **end** **if**;  **end** **process**;  **process**(counter)  **begin**  temp\_counter <= counter;  **end** **process**;  **end** **Behavioral**; |



| FIR |
| --- |
| **library** **IEEE**;  **use** **IEEE.std\_logic\_1164.ALL**;  **use** **IEEE.numeric\_std.all**;  **entity** **fir** **is**  **Port** (  valid\_in : **in** **std\_logic**;  clk : **in** **std\_logic**;  rst : **in** **std\_logic**;  x : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  valid\_out : **out** **std\_logic**;  y : **out** **std\_logic\_vector**(**19**-**1** **downto** **0**);  -- Debug ports  dbg\_counter : **out** **std\_logic\_vector**(**2** **downto** **0**);  dbg\_rom\_out : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_out : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_mac\_init : **out** **std\_logic**;  dbg\_we : **out** **std\_logic**;  -- Additional debug ports  dbg\_x\_register : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_di : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_addr : **out** **std\_logic\_vector**(**2** **downto** **0**);  dbg\_mac\_acc : **out** **std\_logic\_vector**(**18** **downto** **0**)  );  **end** **fir**;  **architecture** **structural** **of** **fir** **is**  **component** **mac** **is**  **Port** (  clk : **in** **std\_logic**;  mac\_init : **in** **std\_logic**;  rom\_out : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  ram\_out : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  y : **out** **std\_logic\_vector**(**19**-**1** **downto** **0**)  );  **end** **component**;    **component** **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** **component**;      **component** **ram** **is**  **generic** (  data\_width : **integer** :=**8** --- width of data (bits)  );  **port** (clk : **in** **std\_logic**;  rst : **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** **component**;    **component** **control\_unit** **is**  **Port** (  clk : **in** **std\_logic**;  valid\_in : **in** **std\_logic**;  rom\_address : **out** **std\_logic\_vector**(**3**-**1** **downto** **0**);  ram\_address : **out** **std\_logic\_vector**(**3**-**1** **downto** **0**);  mac\_init : **out** **std\_logic**;  we : **out** **std\_logic**; -- write enable for ram  valid\_out : **out** **std\_logic**  );  **end** **component**;  **signal** rom\_out\_int : **std\_logic\_vector**(**8**-**1** **downto** **0**);  **signal** ram\_out\_int : **std\_logic\_vector**(**8**-**1** **downto** **0**);  **signal** mac\_init : **std\_logic**;  **signal** we : **std\_logic**;    **signal** rom\_address : **std\_logic\_vector**(**3**-**1** **downto** **0**);  **signal** ram\_address : **std\_logic\_vector**(**3**-**1** **downto** **0**);    **signal** mac\_init\_register : **std\_logic**;  **signal** x\_register : **std\_logic\_vector**(**8**-**1** **downto** **0**);  -- Add new internal signal for MAC output  **signal** mac\_output : **std\_logic\_vector**(**19**-**1** **downto** **0**);  **begin**  -- stage 1: control unit and register for x  control\_unit\_inst : control\_unit  **port** **map** (  clk => clk,  valid\_in => valid\_in,  rom\_address => rom\_address,  ram\_address => ram\_address,  mac\_init => mac\_init,  we => we,  valid\_out => valid\_out  );    x\_reg\_proc : **process**(clk, rst)  **begin**  **if** rst = '1' **then**  x\_register <= (**others** => '0');  **elsif** rising\_edge(clk) **then**  x\_register <= x;  **end** **if**;  **end** **process**;    -- stage 2: rom and ram and mac\_init register  rom\_inst : rom  **port** **map** (  clk => clk,  en => '1',  addr => rom\_address,  rom\_out => rom\_out\_int  );    ram\_inst : ram  **port** **map** (  clk => clk,  rst => rst,  we => we,  en => '1',  addr => ram\_address,  di => x\_register,  do => ram\_out\_int  );    mac\_init\_register\_proc : **process**(clk, rst)  **begin**  **if** rst = '1' **then**  mac\_init\_register <= '0';  **elsif** rising\_edge(clk) **then**  mac\_init\_register <= mac\_init;  **end** **if**;  **end** **process**;  -- stage 3: mac  mac\_inst : mac  **port** **map** (  clk => clk,  mac\_init => mac\_init\_register,  rom\_out => rom\_out\_int,  ram\_out => ram\_out\_int,  y => mac\_output -- Connect to internal signal  );  -- Connect output port  y <= mac\_output;  -- Debug assignments  dbg\_counter <= rom\_address;  dbg\_rom\_out <= rom\_out\_int;  dbg\_ram\_out <= ram\_out\_int;  dbg\_mac\_init <= mac\_init;  dbg\_we <= we;    -- Add new debug assignments  dbg\_x\_register <= x\_register;  dbg\_ram\_di <= x\_register; -- RAM data input  dbg\_ram\_addr <= ram\_address;  dbg\_mac\_acc <= mac\_output; -- Use internal signal instead of y  **end** **structural**; |



| FIR Testbench |
| --- |
| **library** **IEEE**;  **use** **IEEE.STD\_LOGIC\_1164.ALL**;  **use** **IEEE.STD\_LOGIC\_UNSIGNED.ALL**;  **use** **IEEE.NUMERIC\_STD.ALL**;  **use** **IEEE.math\_real.all**;  **entity** **fir\_tb** **is**  **end** **fir\_tb**;  **architecture** **testbench** **of** **fir\_tb** **is**  **constant** data\_width, coeff\_width : **integer** := **8**;  **constant** min\_vector\_size : **integer** := **8**;  **component** **fir** **is**  **Port** (  valid\_in : **in** **std\_logic**;  clk : **in** **std\_logic**;  rst : **in** **std\_logic**;  x : **in** **std\_logic\_vector**(**8**-**1** **downto** **0**);  valid\_out : **out** **std\_logic**;  y : **out** **std\_logic\_vector**(**19**-**1** **downto** **0**);  -- Debug ports  dbg\_counter : **out** **std\_logic\_vector**(**2** **downto** **0**);  dbg\_rom\_out : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_out : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_mac\_init : **out** **std\_logic**;  dbg\_we : **out** **std\_logic**;  dbg\_x\_register : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_di : **out** **std\_logic\_vector**(**7** **downto** **0**);  dbg\_ram\_addr : **out** **std\_logic\_vector**(**2** **downto** **0**);  dbg\_mac\_acc : **out** **std\_logic\_vector**(**18** **downto** **0**)  );  **end** **component**;  --Input Signals  **signal** clk: **std\_logic**:='0';  **signal** rst: **std\_logic**:='0';  **signal** valid\_in: **std\_logic**:='0';  **signal** x : **std\_logic\_vector**(**8**-**1** **downto** **0**):=(**others**=>'0');  **signal** valid\_out: **std\_logic**:='0';  --Output Signals  **signal** y : **std\_logic\_vector**(**19**-**1** **downto** **0**):=(**others**=>'0');  -- Debug Signals  **signal** dbg\_counter : **std\_logic\_vector**(**2** **downto** **0**);  **signal** dbg\_rom\_out : **std\_logic\_vector**(**7** **downto** **0**);  **signal** dbg\_ram\_out : **std\_logic\_vector**(**7** **downto** **0**);  **signal** dbg\_mac\_init : **std\_logic**;  **signal** dbg\_we : **std\_logic**;  **signal** dbg\_x\_register : **std\_logic\_vector**(**7** **downto** **0**);  **signal** dbg\_ram\_di : **std\_logic\_vector**(**7** **downto** **0**);  **signal** dbg\_ram\_addr : **std\_logic\_vector**(**2** **downto** **0**);  **signal** dbg\_mac\_acc : **std\_logic\_vector**(**18** **downto** **0**);  --Clock  **constant** CLK\_PERIOD : **time** := **10**ns;  **begin**  clk\_process: **process**  **begin**  clk <= '0';  **wait** **for** CLK\_PERIOD/**2**;  clk <= '1';  **wait** **for** CLK\_PERIOD/**2**;  **end** **process**;  UUT: fir **port** **map** (  clk => clk,  rst => rst,  valid\_in => valid\_in,  x => x ,  y => y,  valid\_out => valid\_out,  -- Debug connections  dbg\_counter => dbg\_counter,  dbg\_rom\_out => dbg\_rom\_out,  dbg\_ram\_out => dbg\_ram\_out,  dbg\_mac\_init => dbg\_mac\_init,  dbg\_we => dbg\_we,  dbg\_x\_register => dbg\_x\_register,  dbg\_ram\_di => dbg\_ram\_di,  dbg\_ram\_addr => dbg\_ram\_addr,  dbg\_mac\_acc => dbg\_mac\_acc  );  TEST:  **process**  **begin**  rst <= '1';  valid\_in <= '1';  x <= x"01";  **wait** **for** CLK\_PERIOD/**4**;  rst <= '0';  **wait** **for** **3**\*CLK\_PERIOD/**4**;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"02";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"03";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"04";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;  valid\_in <= '1';  x <= x"05";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;  valid\_in <= '1';  x <= x"06";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;  valid\_in <= '1';  x <= x"07";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;  valid\_in <= '1';  x <= x"08";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;  valid\_in <= '1';  x <= x"09";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **15**\*CLK\_PERIOD;      -- new input arrives delayed  valid\_in <= '1';  x <= x"10";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **10**\*CLK\_PERIOD;    -- new invalid input arrives  valid\_in <= '1';  x <= x"10";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **10**\*CLK\_PERIOD;      rst <= '1';  valid\_in <= '1';  x <= x"01";  **wait** **for** CLK\_PERIOD/**4**;  rst <= '0';  **wait** **for** **3**\*CLK\_PERIOD/**4**;  valid\_in <= '0';  rst <= '1';  valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD/**4**;  rst <= '0';  **wait** **for** **3**\*CLK\_PERIOD/**4**;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    valid\_in <= '1';  x <= x"ff";  **wait** **for** CLK\_PERIOD;  valid\_in <= '0';  **wait** **for** **7**\*CLK\_PERIOD;    **wait**;    **end** **process**;  **end** **testbench**; |



Επειδή το critical path από το implementation είναι μεγαλύτερο (7.989 ns) θα λάβουμε αυτό υπόψη για να βρούμε τη μέγιστη συχνότητα στην οποία μπορεί να λειτουργήσει το FPGA:

Και οι πόροι που χρησιμοποιήθηκαν (Post Implementation Utilization):

