ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

UNIVERSITY OF WEST ATTICA

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΠΡΟΗΓΜΕΝΗ ΣΥΝΘΕΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Εργασία Εξαμήνου

ΑΓΓΕΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΕΝΤΖΑΕΟΣ 21390132

Η εργασία

Αρχικά από τις επιλογές που είχαμε για την εργασία επέλεξα το FIR (Finite Impulse Response) Filter. Στην υλοποίηση του χρησιμοποίησα την γλώσσα περιγραφής υλικού VHDL. Όσο αναφορά την εργασία υλοποιήθηκαν αρχεία για τα βάρη σε packages, ένα top level αρχείο που έχει την υλοποίηση του FIR ένα testbench για την εκτέλεση του φίλτρου και ένα constrain αρχείο για το clock του φίλτρου.

fir_filter.vhd

Αρχικά στην υλοποίηση του φίλτρου αρχικοποίησα τις βιβλιοθήκες που θα χρειαστώ για την υλοποίηση. Αμέσως μετά έχω δημιουργήσει 3 αρχεία τύπου package για να αποθηκεύσω τα taps για τα 32, 64 και 128. Τα χρησιμοποιώ βάζοντας κάτω από τις βιβλιοθήκες “use work.fir1_pack.all;” και αντίστοιχα για τα άλλα δύο taps με αλλαγή μόνο στον αριθμό δηλαδή fir2_pack για τα 64 και fir3_pack για τα 128 taps. Παρακάτω δημιούργησα το entity του φίλτρου με τις μεταβλητές που χρειάζομαι όπως φαίνεται και παρακάτω :

```
entity fir_filter is
   port (
      clk       : in  std_logic;
      reset     : in  std_logic;
      enable    : in  std_logic;
      valid_in  : in  std_logic;
      x_in      : in  std_logic_vector(15 downto 0); -- Q15 input
      y_out     : out std_logic_vector(15 downto 0); -- Q15 output
      valid_out : out std_logic
   );
end fir_filter;
```

Αρχικά στο entity έχω το clk που είναι το ρολόι του φίλτρου μου το οποίο είναι τύπου std_logic δηλαδή παίρνει μόνο δυο τιμές 0 και 1. Αντίστοιχα για την τιμή reset,enable,valid_in και το valid_out. Η είσοδος που έχω x_in και η έξοδος y_out παίρνει 16 τιμές μόνο με 0 και 1 διότι δέχομαι είσοδο 16 bit αλλά και η έξοδος μου είναι 16bit

Στην συνέχεια έχουμε την αρχιτεκτονική του φίλτρου δηλαδή πώς περιγράφουμε τις εισόδους μας και τις εξόδους μας ώστε να υλοποιήσουμε το FIR. Όπως φαίνεται παρακάτω θα δούμε τα σήματα τα οποία χρησιμοποίησα για την υλοποίηση του φίλτρου.

```
architecture rtl of fir_filter is

   constant N : integer := N_TAPS;
   type state_t is (IDLE, MULT, COPY, SUM, Y_OUT1);
   type data_array_t is array (0 to N-1) of signed(15 downto 0);
   type y_array_t is array (0 to N-1) of signed(31 downto 0);

   signal x_reg      : data_array_t := (others => (others => '0'));
   signal y_acc      : signed(31 downto 0) := (others => '0');
   signal valid_out_reg : std_logic := '0';
   signal y_pipe : y_array_t := (others => (others => '0'));
   signal y_pipe1     : y_array_t := (others => (others => '0'));
   signal valid_pipe : std_logic_vector(0 to N) := (others => '0');
   signal state : state_t := IDLE;
   signal i     : integer range 0 to N := 0;
   signal acc   : signed(31 downto 0) := (others => '0');
```

Η πρώτη μεταβλητή είναι το constant N που αποθηκεύω τα taps που δέχομαι από το package. Μετά έχω 3 types το πρώτο το οποίο υλοποιεί FSM states δηλαδή κάποιες καταστάσεις όπως έγινε και στην δεύτερη άσκηση. Το type data_array_t το οποίο φτιάχτηκε για το σήμα εισόδου που έχει η αρχιτεκτονική μου. Αντίστοιχα έγινε και για την έξοδο. Αμέσως μετά έχω τα σήματα τα οποία με βοήθησαν να μειώσω το χρόνο του clock μου ώστε το κύκλωμα να τρέχει πιο γρήγορα.

Ξεκινώντας με την υλοποίηση του φίλτρου έχω ένα process με states τα οποία υπολογίζουν το γινόμενο των taps με την είσοδο και προθέτονται όλα μαζί για την τελική υλοποίηση του φίλτρου όπως φαίνεται και παρακάτω.

```
process(clk)
   begin
      if rising_edge(clk) then
         if reset = '1' then
            state <= IDLE;
            valid_out_reg <= '0';
            i <= 0;
            acc <= (others => '0');
         else
            case state is
               when IDLE =>
                  --valid_out_reg <= '0';
                  if enable = '1' and valid_in = '1' then
                     -- shift input
                     valid_out_reg <= '0';
                     for k in N-1 downto 1 loop
                        x_reg(k) <= x_reg(k-1);
                     end loop;
                     x_reg(0) <= signed(x_in);

                     i <= 0;
                     acc <= (others => '0');
                     valid_pipe <= valid_pipe(0 to N-1) & valid_in;
                     state <= MULT;
                  end if;
               when MULT =>
                  y_pipe1(i) <= resize((resize(ROM_ARRAY(i), 32) * resize(x_reg(i), 32)), 32);
                  if i = N-1 then
                     i <= 0;
                     state <= COPY;
                  else
                     i <= i + 1;
                     state <= MULT;  -- stays in MULT
                  end if;
               when COPY =>
                  y_pipe(i) <= y_pipe1(i);
                  if i = N-1 then
                     i <= 0;
                     acc <= (others => '0');
                     state <= SUM;
                  else
                     i <= i + 1;
                     state <= COPY;  -- stays in COPY
                  end if;
               when SUM =>
                  acc <= acc + y_pipe(i);
                  if i = N-1 then
                     y_acc <= acc + y_pipe(i); -- last sum
                     state <= Y_OUT1;
                  else
                     i <= i + 1;
                  end if;
               when Y_OUT1 =>
                  valid_out_reg <= '1';
                  state <= IDLE;
            end case;
         end if;
      end if;
   end process;
```

Με λίγα λόγια εάν το ρολόι είναι θετικά ακμοπυροδοτουμενο έχουμε δύο επιλογές αν είναι το reset 1 αρχικοποιούνται οι τιμές και η κατάσταση του FSM αλλιώς έχω ένα case if όπου αν είναι IDLE κάνω shift τις τιμές της εισόδου. Αν είναι MULT δηλαδή πολλαπλασιασμός, κάνω την πράξη των taps με την είσοδο. Αν φτάσω στο τέλος του πίνακα πάω να κάνω την αντιγραφή αλλιώς κάνω την πράξη για το επόμενο i. Αν είναι COPY δηλαδή αντιγραφή αντιγράφω την έξοδο του σήματος σε άλλο πίνακα κάνοντας pipeline και πάω αν έχω τελειώσει ώστε να προσθέσω όλες τις εξόδους. Αν είναι SUM δηλαδή πρόσθεση, προσθέτω τις τιμές του πίνακα εξόδου για να βγει η τελική έξοδος του πίνακα. Μόλις γίνει αυτό πάω στην κατάσταση Y_OUT1 στην οποία βάζω την τιμή valid_out_reg ως 1 ότι πήρα έξοδο.

Τέλος κρατάω μόνο τις 15 μεγαλύτερες τιμές της εξόδου όπως φαίνεται από κάτω:

```
-- Output assignments
    y_out <= std_logic_vector(y_acc(30 downto 15));
    valid_out <= valid_pipe(N);

end rtl;
```

fir(1-3)_pack.vhd

Στα συγκεκριμένα 3 αρχεία έχω δημιουργήσει packages για να έχω έτοιμα τα taps για 32, 64 και 128. Όπως φαίνεται και παρακάτω θα δείξω για τα 32 taps πως υλοποιήθηκε το αρχείο (αντίστοιχα έγινε και για τα άλλα 2).

```
package fir1_pack is

    constant N_TAPS : integer := 32;
    subtype coeff_t is signed(15 downto 0);  -- Q15 format

    type rom_array_t is array (0 to N_TAPS-1) of coeff_t;

    constant ROM_ARRAY : rom_array_t := (
      to_signed(-185, 16),
       to_signed(-265, 16),
       to_signed(-327, 16),
       to_signed(-350, 16),
       to_signed(-314, 16),
       to_signed(-199, 16),
       to_signed(5, 16),
       to_signed(302, 16),
       to_signed(686, 16),
       to_signed(1137, 16),
       to_signed(1629, 16),
       to_signed(2126, 16),
       to_signed(2589, 16),
       to_signed(2977, 16),
       to_signed(3258, 16),
       to_signed(3405, 16),
       to_signed(3405, 16),
       to_signed(3258, 16),
       to_signed(2977, 16),
       to_signed(2589, 16),
       to_signed(2126, 16),
       to_signed(1629, 16),
       to_signed(1137, 16),
       to_signed(686, 16),
       to_signed(302, 16),
       to_signed(5, 16),
       to_signed(-199, 16),
       to_signed(-314, 16),
       to_signed(-350, 16),
       to_signed(-327, 16),
       to_signed(-265, 16),
       to_signed(-185, 16)
    );
end package;
```

Έχω μετατρέψει από το αρχείο που μας δόθηκε με κώδικα MATLAB για να μετατρέψω τα taps σε to_signed αριθμούς.

const.xdc

Για να ελέγξω το ρολόι δημιούργησα αρχείο constrains για να ορίσω το clock στα 10 ns με την εντολή "create_clock -period 10.000 [get_ports clk]"

fir_filter_tb.vhd

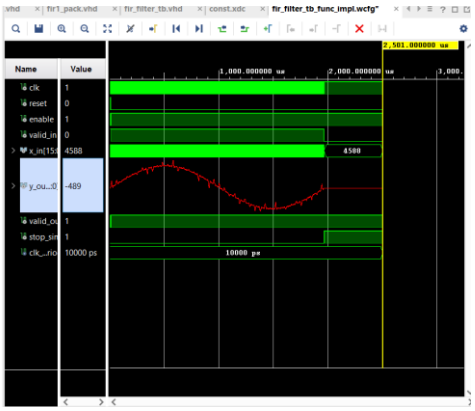
Για να δω αποτελέσματα δημιούργησα ένα αρχείο testbench στο οποίο με κάποια signals και άνοιγμα αρχείων ένα για ανάγνωση και ένα για εγγραφή. Έτσι έχω 3 process ένα για το clk να γίνεται από 0 σε 1 κάθε 5ns, ένα για να διαβάζω από το αρχείο εισόδου τις τιμές x και να παίρνω την έξοδο μετά από N taps * 3(διότι θέλω 3 κύκλους για να βγάλω 1 τιμή) και ένα για να γράφω την έξοδο στο αρχείο εξόδου. Από κάτω φαίνονται τα process για την ανάγνωση, εγγραφή και clock.

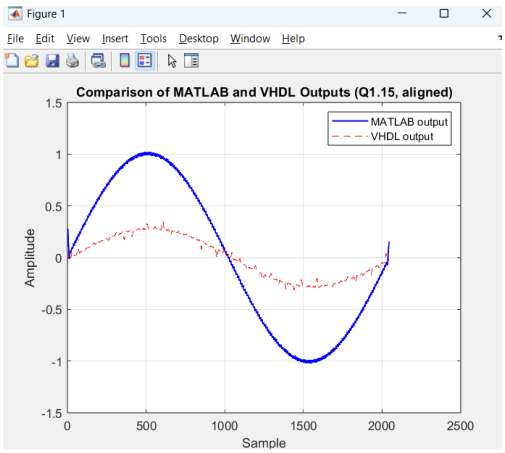
```
clk_process : process
    begin
        while stop_sim = '0' loop
            clk <= '0';
            wait for clk_period / 2;
            clk <= '1';
            wait for clk_period / 2;
        end loop;
        wait;
    end process;

    stim_proc: process
        variable inline  : line;
        variable data_int : integer;
    begin
    -- Reset
        wait for 20 ns;
        reset <= '0';
        enable <= '1';

        -- Read each sample
        while not endfile(input_file) loop
            -- Read next input value
            readline(input_file, inline);
            read(inline, data_int);
            x_in <= std_logic_vector(to_signed(data_int, 16));
            valid_in <= '1';
            wait for clk_period * 3 * N_TAPS;
        end loop;

        -- Wait extra
        valid_in <= '0';
        wait for clk_period * N_TAPS;
        stop_sim <= '1';
        wait;
    end process;

    -- Output logging process
    process(clk)
    variable outline : line;
    begin
        if rising_edge(clk) then
            if valid_out = '1' then
                write(outline, integer'image(to_integer(signed(y_out))));
                writeline(output_file, outline);
            end if;
        end if;
    end process;
```

Αποτελέσματα και Σύγκριση τιμών

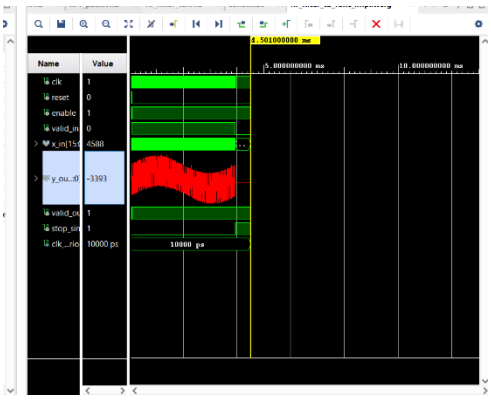
Τα αποτελέσματα που βγάζω είναι τα παρακάτω για τα 3 διαφορετικά taps και η σύγκριση με το MATLAB

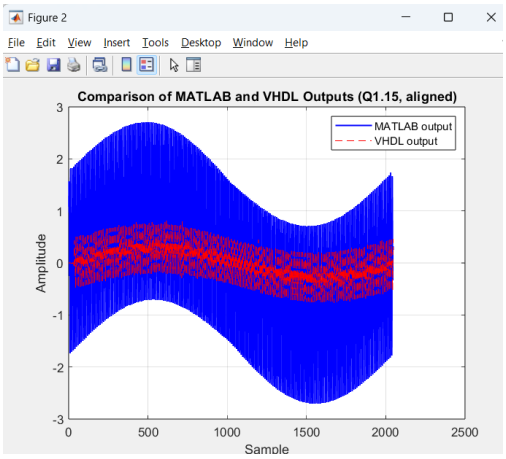
Για τα 32 taps



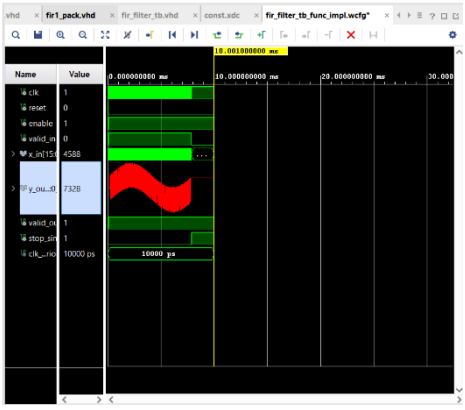


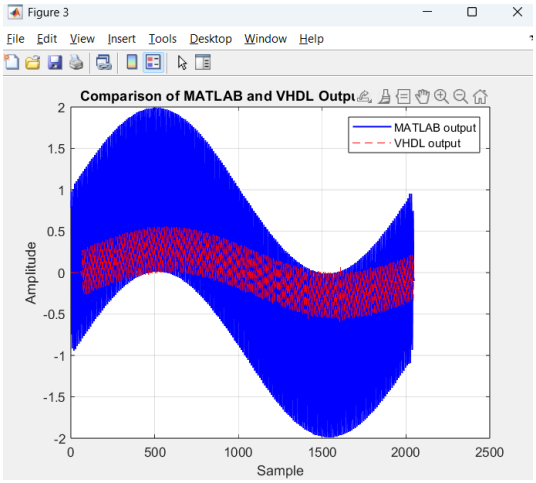
Για 64 taps





Για 128 taps





Παρατηρήσεις

Βλέπουμε ότι ακολουθεί περίπου όπως τα σήματα του MATLAB αλλά κατά την δημιουργία της εισόδου σε τιμές Q1,15 οι τιμές διαιρέθηκαν με το 4 για να μπορούν να είναι σε Q1,15 έγινε σαν μια συμπίεση. Για να κάνω οπτικοποίηση στο MATLAB επειδή το σήμα έχει κάποια καθυστέρηση το έφερα στους ίδιους χρόνους Τέλος με αυτό το φίλτρο μπορώ να έχω και λιγότερο clock από 10ns ακόμα και αν είναι 128 τα taps.