Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

Ψηφιακά Συστήματα VLSI

4η Εργαστηριακή Άσκηση

Υλοποίηση FIR Φίλτρου

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

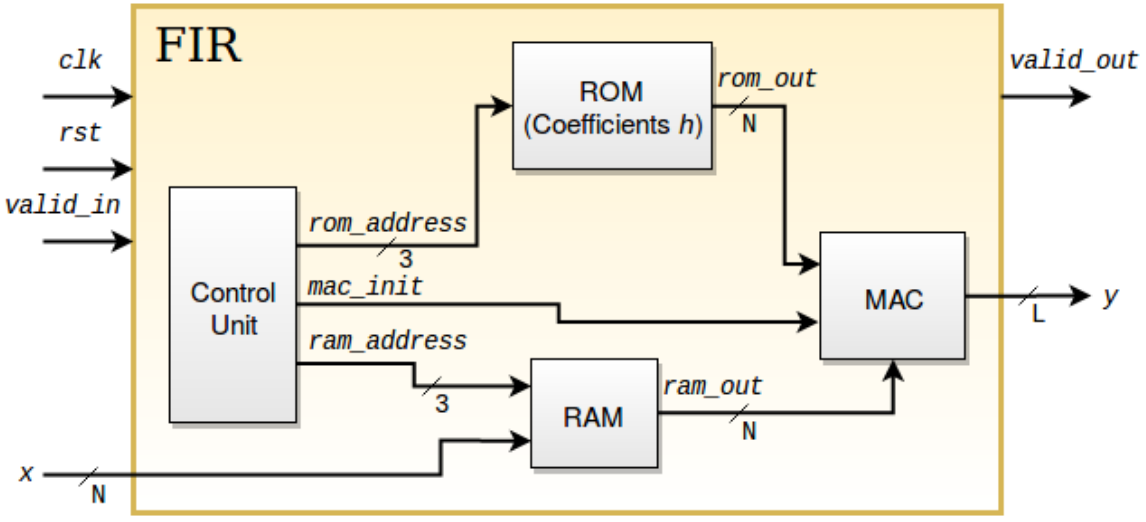
$$y[n] = \sum_{k=0}^{M} h[k]\,x[n-k] = h[0]\,x[n] \;+\; h[1]\,x[n-1] \;+\; \ldots \;+\; h[M]\,x[n-M] \qquad (1)$$

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*

Στα πλαίσια αυτής της εργαστηριακής άσκησης θα υλοποιηθούν διάφορες αρχιτεκτονικές για ένα **FIR φίλτρο**.

Ζητούμενο 1: Για την πρώτη αρχιτεκτονική θα πρέπει σχεδιάσετε ένα σύστημα που να υλοποιεί το δομικό διάγραμμα του Σχήματος 1. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιεί μια Multiply-Accumulate (MAC) μονάδα για τη υλοποίηση των πολλαπλασιασμών και των αθροίσεων. Η υλοποίηση να γίνει για ***N=8 bits*** εύρος δεδομένων x και ***M=8*** συντελεστές.



Σχήμα 1: Δομικό διάγραμμα 8-tap FIR φίλτρου 1ης υλοποίησης.

Τα βασικά modules της αρχιτεκτονικής του φίλτρου είναι τα ακόλουθα:

1. **MAC - Μονάδα Πολλαπλασιασμού με Συσσώρευση (Multiplier Accumulator Unit)**: υπολογίζει την έξοδο του φίλτρου *y*, εκτελώντας την ακόλουθη πράξη:

   $$a \leftarrow a + b \times c$$

   Δέχεται ως είσοδο τους **σταθερούς** συντελεστές του φίλτρου (*rom_out)*, τις τιμές του σήματος εισόδου (*ram_out*), καθώς και ένα σήμα αρχικοποίησης (*mac_init*) που υποδηλώνει την αρχικοποίηση της συσσώρευσης πριν τον υπολογισμό κάθε νέας τιμής του *y*.

   Η υλοποίηση αυτής της μονάδας να γίνει σε **behavioral** περιγραφή και με τη χρήση των τελεστών '+' και '*' που υποστηρίζονται από τη βιβλιοθήκη **std_logic_unsigned**.

2. **ROM**: έχει αποθηκευμένους του **σταθερούς** συντελεστές του φίλτρου *h*.
   Δέχεται ως είσοδο μία διεύθυνση (*rom_address*) και δίνει στην έξοδο τον αντίστοιχο συντελεστή που είναι αποθηκευμένος σε αυτή (*rom_out*).

   Η υλοποίηση αυτής της μονάδας (αρχικοποιημένη με τους συντελεστές) σας παρέχεται έτοιμη και πρέπει να την ενσωματώσετε κατάλληλα στη συνολική αρχιτεκτονική του φίλτρου.

3. **RAM**: αποθηκεύει την παρούσα τιμή του σήματος εισόδου *x*, καθώς και τις 7 προηγούμενες, που είναι απαραίτητες για τον υπολογισμό της εξόδου *y* σύμφωνα με την εξίσωση (1).
   Δέχεται ως είσοδο μία διεύθυνση (*ram_address*) και δίνει στην έξοδο την αντίστοιχη τιμή του σήματος εισόδου που είναι αποθηκευμένη σε αυτή (*ram_out*).
   Κατά τη διάρκεια της λειτουργίας εγγραφής, εγγράφεται η νέα τιμή του σήματος εισόδου *x* στην πρώτη θέση της μνήμης και οι ήδη αποθηκευμένες 8 τιμές ολισθαίνουν κατά μία θέση με αποτέλεσμα η παλαιότερη χρονικά να διαγράφεται.

   Για την υλοποίηση αυτής της μονάδας σας παρέχεται έτοιμος ο κώδικας μίας write-first μνήμης (τα νέα δεδομένα που εγγράφονται σε συγκεκριμένη διεύθυνση βγαίνουν και στην έξοδο της μνήμης στον ίδιο κύκλο ρολογιού). Βασιζόμενοι σε αυτή την μονάδα, θα πρέπει να προσθέσετε ένα μικρό κομμάτι κώδικα το οποίο θα υλοποιεί την μετατόπιση των δεδομένων της μνήμης κατά τη διάρκεια της εγγραφής σύμφωνα με την παραπάνω περιγραφή. Αρχικά όλες οι θέσεις της μνήμης είναι αρχικοποιημένες στο '0' μιας και δεν έχει γίνει καμία εγγραφή.

4. **Control Unit - Μονάδα Ελέγχου**: αποτελεί τη μονάδα που ελέγχει και καθορίζει τη λειτουργία του φίλτρου:

   - Παράγει το σήμα αρχικοποίησης του MAC (*mac_init*).
   - Διευθυνσιοδοτεί ταυτόχρονα τις μνήμες ROM και RAM για τη λειτουργία ανάγνωσης των αντίστοιχων τιμών μέσω των σημάτων *rom_address* και *ram_address* αντίστοιχα. Σε κάθε κύκλο ρολογιού δίνει την επόμενη διεύθυνση των μνημών για ανάγνωση.

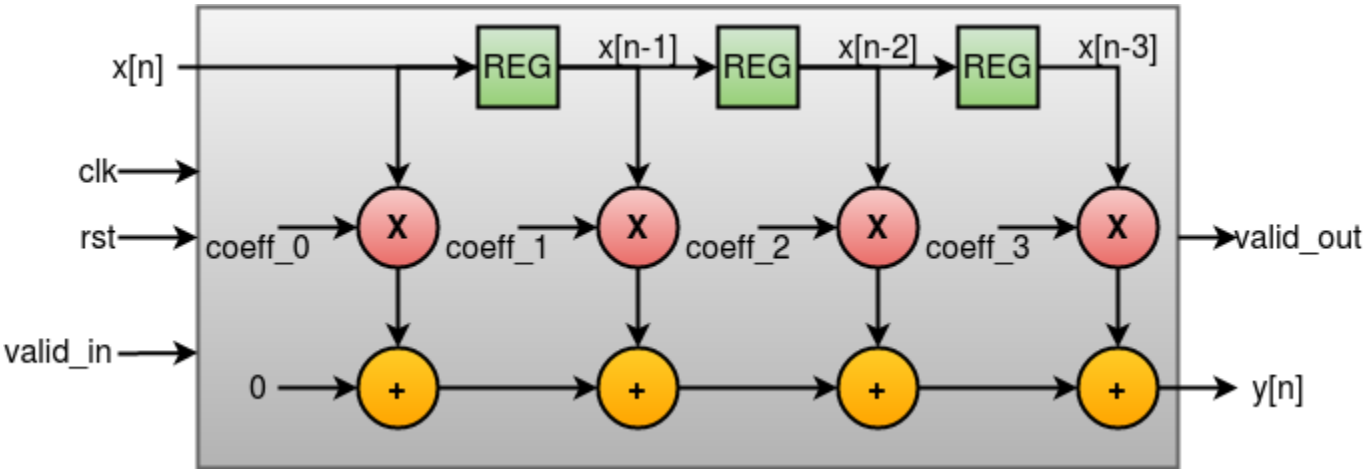
   Η υλοποίηση αυτής της μονάδας να βασιστεί σε έναν μετρητή (up-counter) ο οποίος θα χρησιμοποιηθεί για τη διευθυνσιοδότηση των μνημών ROM και RAM. Η υλοποίηση της μονάδας αυτής να γίνει σε **behavioral** περιγραφή και να χρησιμοποιηθεί η βιβλιοθήκη **std_logic_unsigned** για την υλοποίηση του μετρητή (χρησιμοποιώντας τον τελεστή '+') .

Η λειτουργία του φίλτρου θα είναι η ακόλουθη:

- Δέχεται μία νέα μη προσημασμένη (unsigned) τιμή εισόδου **x κάθε 8 κύκλους ρολογιού**.
- Κάθε νέα τιμή της εξόδου *y* παράγεται με 8 κύκλους ρολογιού διαφορά από την προηγούμενη.

- Για κάθε νέα είσοδο *x* δέχεται ένα σήμα εγκυρότητας εισόδου (`valid_in`) το οποίο υποδηλώνει την εγκυρότητα της συγκεκριμένης εισόδου.
- Για κάθε έγκυρη τιμή της εξόδου *y* ενεργοποιείται ένα σήμα εγκυρότητας εξόδου (`valid_out`).
- Περιλαμβάνει ασύγχρονο reset το οποίο αρχικοποιεί τη μονάδα Control Unit, τη μνήμη RAM με '0' καθώς και το σήμα εγκυρότητας εξόδου `valid_out`.
- Υπολογίζει το πλήθος των bits εξόδου *L* λαμβάνοντας υπόψη όλα τα απαραίτητα bits για την αναπαράσταση του ορθού αποτελέσματος, συμπεριλαμβανομένων και των περιπτώσεων υπερχείλισης.

Ζητούμενο 2: Στην δεύτερη αρχιτεκτονική ζητείται (α) να "ξεδιπλώσετε" το φίλτρο που υλοποιείσατε προηγουμένως και (β) να το μετατρέψετε σε πλήρως pipeline. Στο Σχήμα 2 δίνεται μια προτεινόμενη αρχιτεκτονική για ένα 4-tap FIR φίλτρο στην οποία μπορεί να βασιστεί η σχεδίαση σας. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιεί πολλαπλές μονάδες πολλαπλασιαστών και αθροιστών. Η υλοποίηση να γίνει για ***N=8 bits*** εύρος δεδομένων x και ***M=8*** συντελεστές.



Σχήμα 2: Direct μορφή ενός 4-tap FIR φίλτρου.

Η λειτουργία του φίλτρου θα είναι η ακόλουθη:

- Δέχεται μία νέα μη προσημασμένη (unsigned) τιμή εισόδου **κάθε κύκλο ρολογιού**.
- Μετά από κάποιο αρχικό $T_{latency}$ παράγει μία νέα τιμή της εξόδου *y* σε κάθε κύκλο ρολογιού.
- Για κάθε νέα είσοδο *x* δέχεται ένα σήμα εγκυρότητας εισόδου (`valid_in`) το οποίο υποδηλώνει την εγκυρότητα της συγκεκριμένης εισόδου.
- Για κάθε έγκυρη τιμή της εξόδου *y* ενεργοποιείται ένα σήμα εγκυρότητας εξόδου (`valid_out`).
- Περιλαμβάνει ασύγχρονο reset το οποίο αρχικοποιεί όλες τις μονάδες με '0' καθώς και το σήμα εγκυρότητας εξόδου `valid_out`.
- Υπολογίζει το πλήθος των bits εξόδου *L* λαμβάνοντας υπόψη όλα τα απαραίτητα bits για την αναπαράσταση του ορθού αποτελέσματος, συμπεριλαμβανομένων και των περιπτώσεων υπερχείλισης.

Ζητούμενο 3: Στην τρίτη αρχιτεκτονική ζητείται να μετατρέψετε το φίλτρο που υλοποιήσατε στο Ζητούμενο 2 σε παράλληλο. Συγκεκριμένα, θα πρέπει να σχεδιάσετε ένα φίλτρο το οποίο μπορεί να δεχτεί δύο δεδομένα εισόδου παράλληλα και να παράξει τα αντίστοιχα αποτελέσματα συγχρόνως. Η υλοποίηση να γίνει για ***N=8 bits*** εύρος δεδομένων x και ***M=8*** συντελεστές. Η μετατροπή ενός FIR φίλτρου (π.χ. 4-tap) σε παράλληλο με βαθμό παραλληλίας 2 γίνεται ως εξής:

- Η αρχική εξίσωση του φίλτρου είναι:

$$y[n] = x[n]*h[0] + x[n-1]*h[1] + x[n-2]*h[2] + x[n-3]*h[3]$$

- Αφού θέλουμε να επεξεργαζόμαστε 2 σήματα εισόδου παράλληλα, αντικαθιστώντας στην αρχική εξίσωση το n με 2k και 2k+1, έχουμε:

$$y[2k] = x[2k]*h[0] + x[2k-1]*h[1] + x[2k-2]*h[2] + x[2k-3]*h[3]$$

$$y[2k+1] = x[2k+1]*h[0] + x[2k+1-1]*h[1] + x[2k+1-2]*h[2] + x[2k+1-3]*h[3]$$

  Οι δύο αυτές εξισώσεις δίνουν τα αποτελέσματα για την επεξεργασία των δύο εισόδων παράλληλα.

Η λειτουργία του φίλτρου θα είναι η ακόλουθη:

- Δέχεται δύο νέες μη προσημασμένη (unsigned) τιμές εισόδου **<u>κ κάθε κύκλο ρολογιού</u>**.
- Μετά από κάποιο αρχικό **$T_{latency}$** παράγει δύο νέες τιμές εξόδου *y* σε κάθε κύκλο ρολογιού.
- Για κάθε ζεύγος εισόδων *x* δέχεται ένα σήμα εγκυρότητας εισόδου (`valid_in`) το οποίο υποδηλώνει την εγκυρότητα του ζεύγους εισόδων.
- Για κάθε ζεύγος έγκυρων τιμών της εξόδου *y* ενεργοποιείται ένα σήμα εγκυρότητας εξόδου (`valid_out`).
- Περιλαμβάνει ασύγχρονο reset το οποίο αρχικοποιεί όλες τις μονάδες με ‘0’ καθώς και το σήμα εγκυρότητας εξόδου `valid_out`.
- Υπολογίζει το πλήθος των bits εξόδου *L* λαμβάνοντας υπόψη όλα τα απαραίτητα bits για την αναπαράσταση του ορθού αποτελέσματος, συμπεριλαμβανομένων και των περιπτώσεων υπερχείλισης.

<u>Προσοχή: Κατά την ενσωμάτωση όλων των ξεχωριστών μονάδων για τη δημιουργία της συνολικής αρχιτεκτονικής του φίλτρου για κάθε ζητούμενη υλοποίηση θα πρέπει να ληφθεί υπόψη ο κατάλληλος συγχρονισμός μεταξύ τους.</u>

<u>ΖΗΤΟΥΜΕΝΑ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ</u>

1) Να υλοποιήσετε τα παραπάνω φίλτρα λαμβάνοντας υπόψη όλες τις ζητούμενες λειτουργίες τους.
2) Να δημιουργήσετε testbench για τον έλεγχο της ορθής λειτουργίας όλων των φίλτρων για *N=8*. Το testbench να περιλαμβάνει τον έλεγχο όλων των λειτουργιών.
3) Να καταγράψετε και να συγκρίνετε τους πόρους (resources) που χρησιμοποιήθηκαν για την υλοποίηση του κάθε φίλτρου καθώς και τη συχνότητα λειτουργίας του FPGA ελέγχοντας το κρίσιμο μονοπάτι.