ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

10/10/2025

Σελίδα τίτλου

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΉΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΏΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΊΑΣΗ / ΥΛΟΠΟΊΗΣΗ ΥΠΟΛΟΓΙΣΤΙΚΏΝ ΜΟΝΆΔΩΝ ΣΕ ΨΗΦΙΑΚΌ ΦΊΛΤΡΟ

Καραμπάς Θωμάς

Επιβλέπων: ΦΩΤΙΟΣ ΒΑΡΤΖΙΩΤΗΣ

Επίκουρος καθηγητής

Άρτα, Οκτώβρης, 2025

DESIGN / IMPLEMENTATION OF COMPUTER UNITS IN DIGITAL FILTER

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 10/10/2025

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
ΦΩΤΙΟΣ ΒΑΡΤΖΙΩΤΗΣ,

2. Μέλος επιτροπής
Στεργίου Ελευθέριος,

3. Μέλος επιτροπής
Δουμένης Γρηγόριος,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Καραμπάς, Θωμάς

Υπογραφή

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία αφορά την σχεδίαση, υλοποίηση και εξομοίωση λειτουργίας πολλαπλασιαστών – αθροιστών και εφαρμογή των μηχανισμών για την δημιουργία ενός ψηφιακού φίλτρου γενικού σκοπού FIR. Θα χρησιμοποιηθεί γλώσσα περιγραφής υλικού (HDL) για την εξομοίωση και η υλοποίηση θα πραγματοποιηθεί σε σύστημα FPGA. Το λογισμικό που θα χρησιμοποιηθεί είναι το ALTERA QUARTUS II Lite Edition.

Τα φίλτρα πεπερασμένης παλμικής απόκρισης (FIR) χαρακτηρίζονται από μια χρονική απόκριση που εξαρτάται μόνο από έναν δεδομένο αριθμό των τελευταίων δειγμάτων του σήματος εισόδου. Το VHDL πρόγραμμα του φίλτρου θα περιέχει και έλεγχο για υπερχείλιση. Το πλήθος των bit όλων των σημάτων από τις εισόδους (x και coef) μέχρι τον πολλαπλασιαστή θα είναι m, ενώ από τις εξόδους του πολλαπλασιαστή μέχρι την έξοδο y το πλήθος θα είναι 2m. Επίσης το πλήθος των βαθμίδων (taps) είναι n.

Λέξεις-κλειδιά: Φίλτρα, FPGA, VHDL, QUARTUS, FIR.

ABSTRACT

This thesis concerns the design, implementation and simulation of operation of multipliers - adders and implementation of the mechanisms for the creation of a general purpose FIR digital filter. A hardware description language (HDL) will be used for the simulation and the implementation will be performed on an FPGA system. The software to be used is ALTERA QUARTUS II Lite Edition.

Finite impulse response (FIR) filters are characterized by a time response that depends only on a given number of the last samples of the input signal. The VHDL program of the filter will also contain a check for overflow. The number of bits of all signals from the inputs (x and coef) to the multiplier will be m, while from the outputs of the multiplier to the output y the number will be 2m. Also the number of taps is n.

Keywords: FILTERS, FPGA, VHDL, QUARTUS, FIR.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Πίνακας περιεχομένων

ΑΠΟΔΟΣΗ ΟΡΩΝ / ΓΛΩΣΣΑΡΙΟ

CPLDs (Complex Programmable Logic Devices - Σύνθετες προγραμματιζόμενες λογικές διατάξεις)

FPGAs (Field Programmable Gate Arrays - Διατάξεις πυλών προγραμματιζόμενες στο πεδίο)

FIR (Finite Impulse Response)

IIR (Infinite Impulse Response)

EEG (ηλεκτροεγκεφαλογραφήματα)

ΗΚΓ (ηλεκτροκαρδιογραφήματα)

ADC (μετατροπέας αναλογικού προς ψηφιακό)

HDL (hardware description language)

VHDL (VHSIC hardware description language)

ASICs (Application Specific Integrated Circuits)

LUT (Look-Up Tables)

CPLD (Complex Programmable Logic Devices)

ΕΙΣΑΓΩΓΗ

Η τεχνολογία των ψηφιακών φίλτρων FIR αποτελεί ένα σημαντικό πεδίο έρευνας και εφαρμογής στον τομέα της ψηφιακής σήμανσης και επεξεργασίας σημάτων. Η ικανότητα των FIR φίλτρων να προσφέρουν έναν ακριβή έλεγχο συχνοτήτων μαζί με τη γραμμική φάση τους, τα καθιστά ιδιαίτερα χρήσιμα σε πολλές εφαρμογές, όπως την εξομάλυνση σημάτων, την ανίχνευση αρμονικών και την απομάκρυνση θορύβου.

Στόχος της παρούσας πτυχιακής εργασίας είναι η συστηματική εξέταση και ανάλυση των ψηφιακών φίλτρων FIR, εστιάζοντας στις βασικές τους ιδιότητες, τη σχεδίαση και τις εφαρμογές τους. Μελετώντας τη θεωρία που διέπει τα FIR φίλτρα και εξετάζοντας πρακτικά παραδείγματα, αναδεικνύονται οι δυνατότητες και οι περιορισμοί τους σε πραγματικές εφαρμογές.

Τέλος, μέσω αυτής της εργασίας προσδιορίζονται οι προοπτικές για μελλοντικές επεκτάσεις και βελτιώσεις στον τομέα των ψηφιακών φίλτρων FIR, με έμφαση στην εφαρμογή νέων τεχνολογιών και μεθόδων για τη βελτιστοποίηση της απόδοσής τους.

Μέσα από τη συνεχή εξέλιξη και την εφαρμογή σε πραγματικά προβλήματα, τα ψηφιακά φίλτρα FIR συνεχίζουν να αποτελούν έναν θεμελιώδη πυλώνα στην πρόοδο της ψηφιακής σήμανσης, προσφέροντας εργαλεία μεγάλης σημασίας για την ανάλυση και την επεξεργασία σημάτων στην εποχή της ψηφιακής επανάστασης.

1.ΦΙΛΤΡΑ

1.1 Τι είναι ένα φίλτρο;

Είναι μερικές φορές επιθυμητό να υπάρχουν κυκλώματα ικανά να φιλτράρουν επιλεκτικά μία συχνότητα ή ένα εύρος συχνοτήτων από ένα μείγμα διαφορετικών συχνοτήτων σε ένα κύκλωμα. Ένα κύκλωμα που έχει σχεδιαστεί για να πραγματοποιεί αυτήν την επιλογή συχνότητας ονομάζεται κύκλωμα φίλτρου, ή απλώς φίλτρο. Μια συνηθισμένη ανάγκη για κυκλώματα φίλτρων εμφανίζεται σε στερεοφωνικά συστήματα υψηλής απόδοσης, όπου ορισμένα εύρη ακουστικών συχνοτήτων πρέπει να ενισχυθούν ή να κατασταλούν για την καλύτερη ποιότητα ήχου και ενεργειακή απόδοση. [13], [14], [20]

Ένα γνωστό παράδειγμα είναι οι ισοσταθμιστές (equalizers), οι οποίοι επιτρέπουν την προσαρμογή των πλατών (εντάσεων) πολλών εύρων συχνοτήτων ανάλογα με τις προτιμήσεις του ακροατή και τις ακουστικές ιδιότητες του χώρου ακρόασης. [13], [14], [20]

Ένα επίσης γνωστό παράδειγμα είναι τα δικτυώματα διαχωρισμού συχνοτήτων (crossover networks), τα οποία εμποδίζουν την πρόσβαση συγκεκριμένων εύρων συχνοτήτων στα ηχεία. Ένα tweeter (ηχείο υψηλών συχνοτήτων) είναι αναποτελεσματικό στην αναπαραγωγή χαμηλών συχνοτήτων, όπως οι χτύποι ντραμς, οπότε ένα κύκλωμα διαχωρισμού (crossover) τοποθετείται μεταξύ του tweeter και των εξόδων του στερεοφωνικού ώστε να εμποδίζει τις χαμηλές συχνότητες και να επιτρέπει μόνο τις υψηλές να φτάνουν στο ηχείο. Αυτό προσφέρει καλύτερη απόδοση στο ηχοσύστημα και συνεπώς καλύτερη λειτουργία. Τόσο οι ισοσταθμιστές όσο και τα δικτυώματα διαχωρισμού συχνοτήτων είναι παραδείγματα φίλτρων, σχεδιασμένων για να φιλτράρουν συγκεκριμένες συχνότητες. [13], [14], [20]

Υπάρχουν δύο κύριες κατηγορίες φίλτρων: τα αναλογικά και τα ψηφιακά. Τα αναλογικά φίλτρα βασίζονται σε ηλεκτρονικά κυκλώματα που λειτουργούν με συνεχή σήματα, ενώ τα ψηφιακά φίλτρα εφαρμόζουν μαθηματικούς αλγορίθμους σε διακριτά (δειγματοληπτημένα) σήματα. [13], [14], [20]

1.2 Αναλογικά Φίλτρα

Τα αναλογικά φίλτρα είναι ηλεκτρονικά κυκλώματα που επεξεργάζονται συνεχές σήματα, δηλαδή σήματα που μπορούν να έχουν οποιαδήποτε τιμή σε οποιαδήποτε χρονική στιγμή. Αυτά τα φίλτρα χρησιμοποιούνται ευρέως στις τηλεπικοινωνίες, στην επεξεργασία ήχου, και σε συστήματα ελέγχου, καθώς επιτρέπουν την απομόνωση ή την ενίσχυση συγκεκριμένων συχνοτήτων σε πραγματικό χρόνο.

Η σχεδίασή τους βασίζεται σε παθητικά και ενεργά ηλεκτρονικά στοιχεία, όπως αντιστάσεις, πυκνωτές, πηνία και τελεστικούς ενισχυτές. Τα παθητικά φίλτρα δεν περιλαμβάνουν ενίσχυση και συνήθως είναι απλούστερα, ενώ τα ενεργά φίλτρα ενσωματώνουν ενισχυτικά στάδια που παρέχουν μεγαλύτερη ευελιξία και καλύτερη απόκριση.

Παρόλο που τα αναλογικά φίλτρα είναι γενικά γρήγορα και δεν απαιτούν ψηφιακή επεξεργασία, παρουσιάζουν ορισμένα μειονεκτήματα όπως:
- Ευαισθησία σε μεταβολές της θερμοκρασίας και των παραμέτρων των στοιχείων,
- Περιορισμένη ακρίβεια στη σχεδίαση συχνοτήτων αποκοπής,
- Δυσκολία στην υλοποίηση πολύπλοκων χαρακτηριστικών χωρίς σημαντική αύξηση πολυπλοκότητας.

Η μετάβαση στα ψηφιακά φίλτρα επέτρεψε την υλοποίηση φίλτρων με πολύ ακριβείς και σταθερές προδιαγραφές, κάτι που καθιστά τα αναλογικά φίλτρα πιο κατάλληλα για εφαρμογές που απαιτούν υψηλή ταχύτητα, χαμηλή καθυστέρηση και απλότητα υλοποίησης. [13], [14], [20]

1.3 Ψηφιακά φίλτρα: Εισαγωγή

Τα ψηφιακά φίλτρα είναι αλγόριθμοι που επεξεργάζονται δειγματοληπτημένα, διακριτού χρόνου σήματα, τροποποιώντας τα φασματικά χαρακτηριστικά τους με σκοπό την απομάκρυνση ανεπιθύμητου θορύβου ή την ενίσχυση επιθυμητών στοιχείων. Λειτουργούν πάνω σε ακολουθίες αριθμών που αντιστοιχούν στα δείγματα ενός σήματος, εφαρμόζοντας μαθηματικές πράξεις όπως πολλαπλασιασμούς και αθροίσεις. [13], [14], [20]

Τα ψηφιακά φίλτρα τροποποιούν το φάσμα συχνοτήτων ενός σήματος, επιτρέποντας τη διέλευση ή την απόρριψη συγκεκριμένων συχνοτήτων, επηρεάζοντας έτσι τη μορφή του σήματος. Η λειτουργία τους βασίζεται στη συνέλιξη του εισερχόμενου σήματος με μια συνάρτηση φίλτρου (ή σύνολο συντελεστών). Αυτή η πράξη επηρεάζει άμεσα το φάσμα συχνοτήτων του σήματος. [13], [14], [20]

Τα ψηφιακά φίλτρα είναι ευέλικτα εργαλεία που βρίσκουν εφαρμογή σε πολλούς τομείς, όπως: Επεξεργασία σήματος: Αφαίρεση θορύβου, ενίσχυση συγκεκριμένων συχνοτήτων ή απομόνωση επιθυμητών στοιχείων. [13], [14], [20]

Επεξεργασία ήχου και εικόνας: Βελτίωση ποιότητας ήχου, μείωση παραμορφώσεων, ενίσχυση οπτικών στοιχείων.

Συστήματα ελέγχου: Διαμόρφωση της απόκρισης ενός συστήματος, π.χ. μέσω απόρριψης ανεπιθύμητων ταλαντώσεων.

Τα ψηφιακά φίλτρα μπορούν να διαχωριστούν σε δύο κατηγορίες:

FIR (Πεπερασμένης Απόκρισης):

Σταματούν να «απαντούν» μετά από λίγα βήματα. Είναι πολύ σταθερά και προβλέψιμα. Ιδανικά όταν θέλουμε γραμμική φάση.

IIR (Άπειρης Απόκρισης):

Συνεχίζουν να επηρεάζουν το σήμα για μεγάλο διάστημα. Είναι πιο αποδοτικά υπολογιστικά αλλά πιο δύσκολα στον σχεδιασμό. [13], [14], [20]

1.4 Σύγκριση αναλογικών και ψηφιακών φίλτρων

Τα φίλτρα —αναλογικά και ψηφιακά— έχουν τον ίδιο βασικό σκοπό: να διαμορφώσουν το φάσμα ενός σήματος επιτρέποντας τη διέλευση επιθυμητών συχνοτήτων και την απόρριψη ανεπιθύμητων. Ωστόσο, υπάρχουν σημαντικές διαφορές στον τρόπο λειτουργίας, την τεχνολογία και τις δυνατότητές τους: [13], [14], [20]

1. Τύπος Σήματος
    - Αναλογικά φίλτρα: Επεξεργάζονται συνεχή σήματα, δηλαδή σήματα που αλλάζουν συνεχώς με το χρόνο και δεν έχουν διακριτή δειγματοληψία. Είναι κατάλληλα για φυσικά κυκλώματα και εφαρμόζονται απευθείας σε ηλεκτρικά σήματα.
    - Ψηφιακά φίλτρα: Επεξεργάζονται διακριτά σήματα που έχουν περάσει από δειγματοληψία και μετατροπή A/D (Αναλογικό-σε-Ψηφιακό). Τα δεδομένα είναι αριθμοί που αντιπροσωπεύουν το σήμα σε συγκεκριμένες χρονικές στιγμές. [13], [14], [20]

2. Τεχνολογία Επεξεργασίας
    - Αναλογικά φίλτρα: Υλοποιούνται με παθητικά (αντιστάσεις, πυκνωτές, πηνία) ή ενεργά στοιχεία (τελεστικοί ενισχυτές). Η συμπεριφορά τους εξαρτάται από φυσικές ιδιότητες υλικών.
    - Ψηφιακά φίλτρα: Υλοποιούνται μέσω αλγορίθμων που εκτελούνται από μικροεπεξεργαστές, DSP ή FPGA. Δεν βασίζονται σε φυσικά στοιχεία αλλά σε μαθηματικούς υπολογισμούς. [13], [14], [20]
3. Γραμμική Φάση
    - Αναλογικά φίλτρα: Είναι δύσκολο να εξασφαλίσουν γραμμική φασική απόκριση, δηλαδή η φάση να μεταβάλλεται γραμμικά ως προς τη συχνότητα – κάτι απαραίτητο σε εφαρμογές όπως η επεξεργασία ήχου.
    - Ψηφιακά φίλτρα: Τα φίλτρα τύπου FIR μπορούν να σχεδιαστούν εύκολα με γραμμική φάση, κάνοντάς τα ιδανικά για περιβάλλοντα όπου η παραμόρφωση σήματος πρέπει να ελαχιστοποιείται. [13], [14], [20]
4. Ακρίβεια
    - Αναλογικά φίλτρα: Η συμπεριφορά τους επηρεάζεται από ανοχές στοιχείων, μεταβολές θερμοκρασίας και παλαιότητα εξαρτημάτων. Αυτό σημαίνει πως δεν έχουν σταθερή ακρίβεια.

- Ψηφιακά φίλτρα: Παρέχουν πολύ υψηλή ακρίβεια, καθώς οι υπολογισμοί είναι ψηφιακοί και δεν εξαρτώνται από φυσικούς παράγοντες. Το τελικό αποτέλεσμα είναι πιο προβλέψιμο και αξιόπιστο. [13], [14], [20]

5. Ευελιξία
    - Αναλογικά φίλτρα: Η παραμετροποίησή τους είναι περιορισμένη – αν θες να αλλάξεις τη συχνότητα αποκοπής ή τον τύπο φίλτρου, απαιτείται φυσική αλλαγή στο κύκλωμα.
    - Ψηφιακά φίλτρα: Είναι εξαιρετικά ευέλικτα – μπορείς να αλλάξεις τις παραμέτρους τους με απλή τροποποίηση του λογισμικού, χωρίς καμία αλλαγή στο υλικό. [13], [14], [20]
6. Αναβάθμιση
    - Αναλογικά φίλτρα: Δεν είναι δυνατή η αναβάθμιση χωρίς τροποποιήσεις στο υλικό. Απαιτείται εκ νέου σχεδίαση ή αντικατάσταση του κυκλώματος.
    - Ψηφιακά φίλτρα: Μπορούν να αναβαθμιστούν εύκολα μέσω λογισμικού ή επαναπρογραμματισμού της FPGA, παρέχοντας σημαντικό πλεονέκτημα σε εφαρμογές που εξελίσσονται με την πάροδο του χρόνου. [13], [14], [20]

Τα αναλογικά φίλτρα παραμένουν χρήσιμα για απλές, χαμηλού κόστους εφαρμογές σε συνεχή σήματα. Ωστόσο, τα ψηφιακά φίλτρα προσφέρουν σαφώς ανώτερη ακρίβεια, ευελιξία και προσαρμοστικότητα, καθιστώντας τα κατάλληλα για τις περισσότερες σύγχρονες εφαρμογές επεξεργασίας σήματος. [13], [14], [20]

1.5 Εφαρμογές φίλτρων

Τα φίλτρα, τόσο τα αναλογικά όσο και τα ψηφιακά, είναι απαραίτητα εργαλεία στην επεξεργασία σήματος. Οι κύριες εφαρμογές τους περιλαμβάνουν τα παρακάτω πεδία:

- Επεξεργασία Ήχου: Τα φίλτρα χρησιμοποιούνται για την αφαίρεση ανεπιθύμητων θορύβων, την εξομάλυνση ή ενίσχυση συγκεκριμένων συχνοτήτων, και την προσαρμογή του ηχητικού φάσματος σε συστήματα όπως equalizers, noise gates και compressors.
- Επεξεργασία Εικόνας και Βίντεο: Τα ψηφιακά φίλτρα χρησιμοποιούνται για την αφαίρεση θορύβου εικόνας, για την ενίσχυση άκρων, τον θολό μετασχηματισμό (blurring), και την απόσβεση ανεπιθύμητων στοιχείων από οπτικά δεδομένα.
- Τηλεπικοινωνίες: Χρησιμοποιούνται για διαχωρισμό συχνοτήτων (channel separation), αποθορυβοποίηση σημάτων, καθώς και για διαμόρφωση και αποδιαμόρφωση σημάτων σε ασύρματες και ενσύρματες επικοινωνίες.
- Ιατρικές Εφαρμογές: Συστήματα όπως ηλεκτροκαρδιογραφήματα (ΗΚΓ) και ηλεκτροεγκεφαλογραφήματα (EEG) χρησιμοποιούν φίλτρα για την απομάκρυνση παρεμβολών και την εξαγωγή καθαρών βιοσημάτων.
- Βιομηχανικός Έλεγχος και Αυτοματισμοί: Χρησιμοποιούνται για φιλτράρισμα σημάτων αισθητήρων, αποφυγή ψευδών μετρήσεων και εξομάλυνση εισερχόμενων δεδομένων πριν από τη λήψη αποφάσεων από μικροελεγκτές ή PLC.
- Στρατιωτικές και Διαστημικές Τεχνολογίες: Σε ραντάρ, συστήματα εντοπισμού και επικοινωνιών υψηλής πιστότητας, όπου η καθαρότητα και η μη καθυστέρηση των σημάτων είναι κρίσιμες.
- Χρηματοοικονομική Ανάλυση: Αν και λιγότερο συνηθισμένο, φίλτρα χρησιμοποιούνται σε τεχνική ανάλυση δεδομένων χρηματιστηριακών τιμών για την εξομάλυνση χρονικών σειρών και τον εντοπισμό τάσεων. [13], [14], [20]

2. ΨΗΦΙΑΚΑ ΦΙΛΤΡΑ

Ένα ψηφιακό φίλτρο είναι ένα σύστημα που εκτελεί μαθηματικές πράξεις σε ένα διακριτό και δειγματοληπτικό σήμα χρόνου, έτσι ώστε να ενισχυθούν ή να μειωθούν ορισμένες πτυχές αυτού του συγκεκριμένου σήματος. Χρησιμοποιείται σε μεγάλο βαθμό στην επεξεργασία σήματος και διαφέρει από ένα αναλογικό φίλτρο, το οποίο είναι ένα ηλεκτρονικό κύκλωμα που λειτουργεί με συνεχή σήματα. [13], [14], [20], [17]

2.1 Τι είναι ψηφιακό φίλτρο

Τα ψηφιακά φίλτρα είναι ακριβότερα σε σύγκριση με τα αναλογικά, αλλά μπορούν να μετατρέψουν πολλές πιθανές ή αδύνατες μορφές σε δυνατότητες.

Δομή και Λειτουργία:

Ένα τυπικό ψηφιακό φίλτρο περιλαμβάνει:

- Μετατροπέα αναλογικού-σε-ψηφιακό (ADC), ο οποίος μετατρέπει το συνεχές σήμα σε ακολουθία αριθμητικών δειγμάτων.
- Μικροεπεξεργαστή ή ψηφιακό επεξεργαστή σήματος (DSP), ο οποίος εφαρμόζει τον αλγόριθμο του φίλτρου.
- Μνήμη, για την αποθήκευση των δεδομένων εισόδου, των ενδιάμεσων αποτελεσμάτων και των συντελεστών φίλτρου.
- Μετατροπέα ψηφιακού-σε-αναλογικό (DAC), ο οποίος μετατρέπει το φιλτραρισμένο σήμα πίσω σε αναλογική μορφή εφόσον απαιτείται για έξοδο σε αναλογικές διατάξεις.

Το λογισμικό που εκτελείται στο DSP ή στον μικροελεγκτή εφαρμόζει μαθηματικές πράξεις, όπως πολλαπλασιασμούς και αθροίσεις, προσομοιώνοντας τη λειτουργία ενός αναλογικού φίλτρου μέσω εξισώσεων διαφοράς.

Η συμπεριφορά ενός ψηφιακού φίλτρου είναι επίσης σημαντική. Χρησιμοποιούνται διάφορες μαθηματικές προσεγγίσεις για την κατανόηση των αντιδράσεών τους. Ο απλούστερος τρόπος είναι η ανάλυση της απόκρισης όταν μια απλή είσοδος περάσει στο φίλτρο, για παράδειγμα μια ώθηση. Στη συνέχεια με βάση το αποτέλεσμα, μπορούν επίσης να αναλυθούν πολύπλοκες εισροές. [13], [14], [20], [17]

Τα ψηφιακά φίλτρα έχουν διάφορες εφαρμογές, όπως την απομάκρυνση θορύβου, την ομαλοποίηση σήματος, την ανάλυση σήματος και την ενίσχυση συγκεκριμένων συχνοτήτων. Υπάρχουν διάφοροι τύποι ψηφιακών φίλτρων, που διακρίνονται με βάση τα χαρακτηριστικά και τις λειτουργίες τους. [13], [14], [20], [17]

Τα ψηφιακά φίλτρα χωρίζονται στις εξής κατηγορίες:

- Χαμηλοπερατά φίλτρα (Low-pass filters):

  Επιτρέπουν τις χαμηλές συχνότητες να περάσουν και απορρίπτουν τις υψηλές συχνότητες. Χρήσιμα για την απομάκρυνση υψηλής συχνότητας θορύβου.

- Υψηλοπερατά φίλτρα (High-pass filters):

  Επιτρέπουν τις υψηλές συχνότητες να περάσουν και απορρίπτουν τις χαμηλές συχνότητες. Χρησιμοποιούνται για την εξάλειψη των χαμηλών συχνοτήτων θορύβου.

- Φίλτρα μεσαίων συχνοτήτων (Band-pass filters):

  Επιτρέπουν μια συγκεκριμένη περιοχή συχνοτήτων να περάσει και απορρίπτουν τις συχνότητες εκτός αυτής της περιοχής. Χρήσιμα για την ανάλυση συγκεκριμένων συχνοτήτων μέσα σε ένα σήμα.

- Φίλτρα απόρριψης ζώνης (Band-stop filters):

  Απορρίπτουν μια συγκεκριμένη περιοχή συχνοτήτων και επιτρέπουν τις υπόλοιπες συχνότητες να περάσουν. Χρησιμοποιούνται για την αφαίρεση ανεπιθύμητων συχνοτήτων από ένα σήμα.

- Φίλτρα μερικής ζώνης (Notch filters):

  Είναι ειδικά φίλτρα απόρριψης ζώνης που απορρίπτουν πολύ στενές ζώνες συχνοτήτων. Χρήσιμα για την εξάλειψη συγκεκριμένων παρεμβολών.

- Γραμμικά φίλτρα (Linear filters):

  Αλλαγές στην είσοδο του σήματος προκαλούν αντίστοιχες γραμμικές αλλαγές στην έξοδο. Εφαρμόζουν μια γραμμική συνάρτηση στο σήμα.

- Μη γραμμικά φίλτρα (Non-linear filters):

  Η έξοδος δεν είναι μια γραμμική συνάρτηση της εισόδου. Χρήσιμα σε εφαρμογές όπου τα γραμμικά φίλτρα δεν είναι αποτελεσματικά.

- Πεπερασμένης απόκρισης παλμού (FIR - Finite Impulse Response filters):

  Έχουν πεπερασμένο αριθμό συντελεστών (taps). Είναι πάντα σταθερά και μπορούν εύκολα να σχεδιαστούν με γραμμική φάση. Χρησιμοποιούν μόνο δείγματα εισόδου (όχι προηγούμενες εξόδους). Συνηθισμένα σε εφαρμογές όπου απαιτείται ακριβής χρονισμός και φασική ορθότητα (π.χ. ήχος, επικοινωνίες).

- Άπειρης απόκρισης παλμού (IIR - Infinite Impulse Response filters):

  Έχουν άπειρη διάρκεια απόκρισης στο παλμό. Μπορούν να επιτύχουν επιθυμητές αποκρίσεις με λιγότερους συντελεστές σε σύγκριση με FIR. Περιλαμβάνουν ανατροφοδότηση, δηλαδή εξαρτώνται και από προηγούμενες εξόδους. Είναι πιο αποδοτικά υπολογιστικά αλλά απαιτούν προσοχή στη σταθερότητα.

- Προσαρμοστικά φίλτρα (Adaptive filters):

  Μπορούν να προσαρμόζουν τις παραμέτρους τους αυτόματα με βάση το σήμα εισόδου. Χρησιμοποιούνται σε δυναμικά περιβάλλοντα όπου οι ιδιότητες του σήματος μπορεί να αλλάζουν με την πάροδο του χρόνου.

Στην εργασία αυτή θα ασχοληθούμε περισσότερο με τα φίλτρα πεπερασμένης κρουστικής απόκρισης FIR, τι είναι ένα φίλτρο FIR, τα χαρακτηριστικά τους, ποιες μεθόδους σχεδίασης υπάρχουν και ποια τα πλεονεκτήματα χρήσης ενός φίλτρου FIR σε σχέση με τα άλλα ψηφιακά φίλτρα. [13], [14], [20], [17]

2.2 Ψηφιακά φίλτρα FIR

Ένα ψηφιακό φίλτρο FIR είναι ένα γραμμικό χρονικά αμετάβλητο (LTI) σύστημα που υπολογίζει την έξοδο του βασισμένο σε έναν πεπερασμένο αριθμό προηγούμενων τιμών εισόδου. Ο όρος "πεπερασμένη απόκριση κρούσης" αναφέρεται στο γεγονός ότι η απόκριση του φίλτρου σε μια κρούση (δηλαδή, ένα σήμα που είναι μηδέν σε όλες τις χρονικές στιγμές εκτός από μία όπου είναι μονάδα) είναι μηδέν μετά από ένα πεπερασμένο διάστημα χρόνου. Κάποια βασικά χαρακτηριστικά των φίλτρων fir είναι η Πεπερασμένη Διάρκεια Απόκρισης, η Γραμμική Φάση, η Σταθερότητα και η Μηδενική Ανατροφοδότηση. [13], [14], [20], [17]

Η απόκριση παλμού του φίλτρου FIR είναι πεπερασμένη, δηλαδή το φίλτρο αντιδρά σε μια είσοδο παλμού για έναν συγκεκριμένο αριθμό δειγμάτων πριν γίνει μηδενική. Αυτό σημαίνει ότι αν το φίλτρο λάβει ως είσοδο ένα σήμα που περιέχει ένα μόνο μηδενικό στοιχείο ακολουθούμενο από μηδενικά (παλμός), η έξοδος του φίλτρου θα είναι μη μηδενική μόνο για μια πεπερασμένη σειρά δειγμάτων και στη συνέχεια θα γίνει μηδενική. [13], [14], [20], [17]

Ένα από τα σημαντικότερα πλεονεκτήματα των φίλτρων FIR είναι η δυνατότητά τους να έχουν γραμμική φάση. Αυτό σημαίνει ότι το φίλτρο μπορεί να καθυστερεί όλα τα συστατικά

συχνότητας ενός σήματος κατά την ίδια χρονική διάρκεια, αποφεύγοντας τη φασική παραμόρφωση. Η γραμμική φάση είναι ιδιαίτερα σημαντική σε εφαρμογές όπως η επεξεργασία ήχου και εικόνας, όπου η διατήρηση των φασικών σχέσεων μεταξύ των συχνοτήτων είναι κρίσιμη. [13], [14], [20], [17]

Τα φίλτρα FIR είναι εγγενώς σταθερά. Δεδομένου ότι η έξοδος ενός φίλτρου FIR εξαρτάται μόνο από τα δείγματα εισόδου και όχι από προηγούμενα δείγματα εξόδου, δεν υπάρχει ανατροφοδότηση που θα μπορούσε να οδηγήσει σε ασταθή συμπεριφορά. Αυτό εξασφαλίζει ότι οποιαδήποτε πεπερασμένη είσοδος θα οδηγήσει σε πεπερασμένη έξοδο. [13], [14], [20], [17]

Τα φίλτρα FIR υπολογίζουν την τρέχουσα έξοδο χρησιμοποιώντας μόνο τα τρέχοντα και προηγούμενα δείγματα του σήματος εισόδου. Αυτό τα καθιστά διαφορετικά από τα φίλτρα IIR (Infinite Impulse Response), τα οποία χρησιμοποιούν επίσης προηγούμενα δείγματα της εξόδου.

Η έξοδος y[n] ενός ψηφιακού φίλτρου FIR υπολογίζεται ως ο γραμμικός συνδυασμός των πιο πρόσφατων τιμών της εισόδου x[n] :

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \\ &= \sum_{i=0}^{N} b_i \cdot x[n-i], \end{aligned}$$

όπου:

y[n] είναι η έξοδος στην χρονική στιγμή n,

x[n] είναι η είσοδος στην χρονική στιγμή n,

bi είναι οι συντελεστές του φίλτρου (ή βάρη) που καθορίζουν την απόκριση του φίλτρου,

N είναι η τάξη του φίλτρου, η οποία είναι ίση με τον αριθμό των συντελεστών.

Ένα φίλτρο FIR έχει μια σειρά από χρήσιμες ιδιότητες που μερικές φορές το καθιστούν προτιμότερο από ένα φίλτρο άπειρης απόκρισης ώθησης (IIR).

Δεν απαιτούνται σχόλια. Αυτό σημαίνει ότι τυχόν σφάλματα στρογγυλοποίησης δεν επιδεινώνονται από αθροιστικές επαναλήψεις. Το ίδιο σχετικό σφάλμα εμφανίζεται σε κάθε υπολογισμό. Αυτό κάνει επίσης την εφαρμογή πιο απλή. [13], [14], [20], [17]

Είναι εγγενώς σταθερές, καθώς η έξοδος είναι ένα άθροισμα ενός πεπερασμένου αριθμού πεπερασμένων πολλαπλασίων των τιμών εισόδου, επομένως δεν μπορεί να είναι μεγαλύτερο από Σ|bi| φορές τη μεγαλύτερη τιμή που εμφανίζεται στην είσοδο. [13], [14], [20], [17]

Μπορεί εύκολα να σχεδιαστεί ώστε να βρίσκεται σε γραμμική φάση κάνοντας την ακολουθία συντελεστών συμμετρική. Αυτή η ιδιότητα μερικές φορές είναι επιθυμητή για εφαρμογές ευαίσθητες στη φάση, για παράδειγμα επικοινωνίες δεδομένων, σεισμολογία, φίλτρα διασταύρωσης και mastering. [13], [14], [20], [17]

Το κύριο μειονέκτημα των φίλτρων FIR είναι ότι απαιτείται πολύ μεγαλύτερη υπολογιστική ισχύς σε έναν επεξεργαστή γενικής χρήσης σε σύγκριση με ένα φίλτρο IIR με παρόμοια ευκρίνεια ή επιλεκτικότητα, ειδικά όταν απαιτούνται αποκοπές χαμηλής συχνότητας (σε σχέση με τον ρυθμό δειγματοληψίας). Ωστόσο, πολλοί επεξεργαστές ψηφιακού σήματος παρέχουν εξειδικευμένες δυνατότητες υλικού για να κάνουν τα φίλτρα FIR περίπου εξίσου αποτελεσματικά με τα IIR για πολλές εφαρμογές. [13], [14], [20], [17]

Τα φίλτρα FIR χρησιμοποιούνται σε πληθώρα εφαρμογών, λόγω της σταθερότητας και της ικανότητάς τους να παρέχουν γραμμική φάση. Μερικές από τις κύριες εφαρμογές περιλαμβάνουν την απομάκρυνση θορύβου, την εξομάλυνση του ήχου, και την εξαγωγή χαρακτηριστικών από ηχητικά σήματα στην επεξεργασία ήχου, την θόλωση (blurring), ενίσχυση άκρων (edge enhancement), και ανίχνευση χαρακτηριστικών (feature detection) στην επεξεργασία εικόνας, την απομάκρυνση παρεμβολών, την εξομάλυνση σημάτων, τη διαμόρφωση σημάτων στις τηλεπικοινωνίες και το φιλτράρισμα βιοϊατρικών σημάτων όπως τα σήματα ΗΚΓ (Ηλεκτροκαρδιογράφημα) και EEG (Ηλεκτροεγκεφαλογράφημα) στην βιοιατρική. [13], [14], [20], [17]

2.3 Hardware Description Language / HDL

Στη μηχανική υπολογιστών, μια γλώσσα περιγραφής υλικού (hardware description language ή HDL) είναι μια γλώσσα που ανήκει σε μια κλάση γλωσσών προγραμματισμού, γλωσσών προδιαγραφών ή γλωσσών μοντελοποίησης για την τυπική περιγραφή και σχεδίαση ηλεκτρονικών κυκλωμάτων, και συνηθέστερα, ψηφιακής λογικής. Μπορεί να περιγράψει τη λειτουργία, τη σχεδίαση και την οργάνωση του κυκλώματος, μαζί με δοκιμές που επιβεβαιώνουν τη λειτουργία του μέσω προσομοίωσης. [7],[9],[10], [8], [20]

Οι HDL είναι κλασικές εκφράσεις σε κείμενο της χωρικής και χρονικής δομής και συμπεριφοράς των ηλεκτρονικών συστημάτων. Όπως και οι γλώσσες ταυτόχρονου προγραμματισμού, η σύνταξη και η σημασιολογία μιας HDL περιλαμβάνουν τον χρόνο, που είναι βασική ιδιότητα του υλικού. [7],[9],[10], [8], [20]

Οι γλώσσες περιγραφής υλικού χρησιμοποιούνται για τη συγγραφή εκτελέσιμων προδιαγραφών για κάποιο συγκεκριμένο υλικό. Ένα πρόγραμμα προσομοίωσης, που έχει σχεδιαστεί να υλοποιεί τη σημασιολογία των εντολών της γλώσσας μαζί με την προσομοίωση του χρόνου που περνά, δίνει στον σχεδιαστή του υλικού τη δυνατότητα να μοντελοποιήσει μια συσκευή υλικού πριν αυτή κατασκευαστεί. Αυτή η δυνατότητα εκτέλεσης δίνει στις HDL την εμφάνιση γλωσσών προγραμματισμού, ενώ πιο σωστά αυτές κατατάσσονται στις γλώσσες προδιαγραφών ή στις γλώσσες μοντελοποίησης. Υπάρχουν προσομοιωτές που μπορούν να μοντελοποιήσουν συστήματα διακριτών συμβάντων (ψηφιακά) και συνεχούς χρόνου (αναλογικά), με τις κατάλληλες HDL αντίστοιχα. [7],[9],[10], [8], [20]

2.4 VHSIC hardware description language / VHDL

Η VHDL είναι μία γλώσσα περιγραφής υλικού, που χρησιμοποιείται για την ανάπτυξη ολοκληρωμένων ψηφιακών κυκλωμάτων και συστημάτων. Αρχικά η δημιουργία της αποσκοπούσε στη μοντελοποίηση και στην προσομοίωση κυκλωμάτων, γι' αυτό πολλά χαρακτηριστικά της γλώσσας έχουν σκοπό την προσομοίωση των λειτουργιών. Αργότερα, η γλώσσα χρησιμοποιήθηκε και ως εργαλείο σύνθεσης. Κατά τη σύνθεση, ο μεταγλωττιστής συνθέτει ένα πραγματικό κύκλωμα που ανταποκρίνεται με ακρίβεια στη λογική και χρονική περιγραφή, την οποία μοντελοποιεί ο κώδικας. [7],[9],[10], [8], [20]

Η γλώσσα VHDL χρησιμοποιείται ευρύτατα για την περιγραφή και υλοποίηση ψηφιακών συστημάτων σε προγραμματιζόμενες λογικές διατάξεις, τύπου CPLDs (Complex Programmable Logic Devices-Σύνθετες προγραμματιζόμενες λογικές διατάξεις) και FPGAs (Field Programmable Gate Arrays). Επίσης, έχει καθιερωθεί σαν ένα πρότυπο (standard) στη σχεδίαση ηλεκτρονικών κυκλωμάτων ASICs (Application Specific Integrated Circuits). [7],[9],[10], [8], [20]

Η VHDL υπακούει στις αρχές των παράλληλων γλωσσών και δεν είναι τυχαίο ότι έχει τις ρίζες της στην Ada, που χρησιμοποιείται για να προγραμματίσει παράλληλες διεργασίες. Ταυτόχρονα, υπακούει στις αρχές του δομημένου προγραμματισμού, που δίνει τη δυνατότητα της σχεδίασης ιεραρχικών κυκλωμάτων. Τέλος, περιέχει τις αρχές του συγχρονισμού και του χρονισμού, που είναι εγγενείς στα κυκλώματα. Ένα από τα χαρακτηριστικά της VHDL είναι ότι μοντελοποιεί με ακρίβεια τόσο τις λειτουργίες του κυκλώματος, όσο και τους χρόνους κατά τους οποίους οι λειτουργίες παράγουν αποτελέσματα. [7],[9],[10], [8], [20]

Η VHDL διαφέρει από τις συμβατικές γλώσσες κατά το ότι δεν προορίζεται να περιγράψει λειτουργίες που εκτελούνται σειριακά, η μία μετά την άλλη. Κάθε πρόταση ή τμήμα κώδικα περιγράφει λειτουργίες, οι οποίες παράγουν αποτελέσματα σε συγχρονισμό με άλλες λειτουργίες,. Τα αποτελέσματα της προσομοίωσης παράγονται με βάση αυστηρές χρονικές προδιαγραφές σε διάφορα σημεία του κυκλώματος και εν τέλει στις εξόδους. Με την έννοια αυτή, είναι δυνατό ένα τμήμα κώδικα να μπορεί να γραφεί σε οποιοδήποτε σημείο του προγράμματος, αφού παράγει αποτελέσματα ανεξαρτήτως της σειράς του. Στο τέλος, ο compiler θα συνθέσει κυκλώματα που θα υλοποιούν στην πράξη τις σύγχρονες λειτουργίες που περιγράφει ο κώδικας. [7],[9],[10], [8], [20]

Η VHDL μπορεί να περιγράψει τόσο σύγχρονες όσο και ακολουθιακές λογικές λειτουργίες. Οι διεργασίες που περιγράφονται στη VHDL παράγουν αποτέλεσμα είτε ταυτόχρονα με την εφαρμογή των εισόδων, όπως συμβαίνει στα συνδυαστικά κυκλώματα, είτε σε συγχρονισμό με συμβάντα (π.χ. παλμούς ρολογιού), όπως συμβαίνει στα ακολουθιακά κυκλώματα. Για το λόγο αυτό η σύνταξη του κώδικα μπορεί να γίνει με σύγχρονες δομές (concurrent code) ή και με ακολουθιακές δομές (sequential code). [7],[9],[10], [8], [20]

Το βασικό πλεονέκτημα της VHDL, όταν αυτή χρησιμοποιείται για σχεδίαση συστημάτων, είναι ότι επιτρέπει την περιγραφή (μοντελοποίηση) και την επαλήθευση (προσομοίωση) του επιθυμητού συστήματος, πριν τα εργαλεία σύνθεσης μεταφράσουν τη σχεδίαση σε πραγματικό υλικό (πύλες και γραμμές). [7],[9],[10], [8], [20]

Ένα άλλο όφελος της VHDL είναι ότι επιτρέπει τον ορισμό ταυτόχρονων συστημάτων (concurrent systems). Η VHDL είναι γλώσσα ροής δεδομένων, σε αντίθεση με τις διαδικαστικές γλώσσες προγραμματισμού όπως η BASIC, η C και η συμβολική γλώσσα, οι οποίες εκτελούνται ακολουθιακά, με κάθε εντολή να ακολουθεί την προηγούμενη. [7],[9],[10], [8], [20]

Ένα έργο σε VHDL έχει πολλές εφαρμογές. Ένα μπλοκ υπολογισμού (calculation block) δημιουργείται μια φορά αλλά μπορεί να χρησιμοποιηθεί σε άλλα έργα. Μπορούν επίσης να ρυθμιστούν διάφορες παράμετροι διαμόρφωσης και λειτουργίας του μπλοκ (παράμετροι χωρητικότητας, το μέγεθος της μνήμης, η βάση των στοιχείων (element base), η σύνθεση μπλοκ και η δομή διασύνδεσης). [7],[9],[10], [8], [20]

Ένα έργο σε VHDL είναι επίσης μεταφέρσιμο. Αν έχει δημιουργηθεί για μια βάση στοιχείων, μπορεί να μεταφερθεί σε μια άλλη βάση, για παράδειγμα σε ένα VLSI με διάφορες τεχνολογίες. [7],[9],[10], [8], [20]

2.5 FPGA

Τα FPGAs (Field-Programmable Gate Arrays), αποτελούν μία από τις πιο σημαντικές τεχνολογικές καινοτομίες στον χώρο των ψηφιακών ηλεκτρονικών και των επαναπρογραμματιζόμενων υπολογιστικών συστημάτων. Πρόκειται για ολοκληρωμένα κυκλώματα που έχουν τη δυνατότητα να επαναπρογραμματιστούν από τον χρήστη μετά την κατασκευή τους, γεγονός που τα καθιστά εξαιρετικά ευέλικτα και ιδανικά για πληθώρα εφαρμογών. [12], [6], [11], [5], [20]

Ουσιαστικά, τα FPGAs προσφέρουν μια πλατφόρμα ενδιάμεση μεταξύ της σταθερότητας και αποδοτικότητας των εξειδικευμένων ολοκληρωμένων κυκλωμάτων (ASICs) και της ευελιξίας που προσφέρουν τα προγραμματιζόμενα συστήματα γενικής χρήσης όπως οι μικροεπεξεργαστές. Αυτό επιτυγχάνεται μέσω της αρχιτεκτονικής τους, η οποία επιτρέπει στον προγραμματιστή να διαμορφώσει την εσωτερική λογική του κυκλώματος ώστε να υλοποιεί πολύπλοκες ψηφιακές λειτουργίες. [12], [6], [11], [5], [20]

2.5.1 Χαρακτηριστικά

Τα FPGAs αποτελούνται από τρεις βασικές συνιστώσες: τις λογικές μονάδες (logic blocks), τους προγραμματιζόμενους διαύλους διασύνδεσης (programmable interconnects), και τις μονάδες εισόδου/εξόδου (I/O blocks). Οι λογικές μονάδες είναι υπεύθυνες για την υλοποίηση των λογικών συναρτήσεων και αποτελούνται συνήθως από πίνακες LUT (Look-Up Tables), μανταλωτές (flip-flops) και πολυπλέκτες. Μέσω των LUT, κάθε λογική μονάδα μπορεί να προγραμματιστεί ώστε να εκτελεί μια συγκεκριμένη λογική συνάρτηση. [12], [6], [11], [5], [20]

Οι δίαυλοι διασύνδεσης επιτρέπουν τη σύνδεση των λογικών μονάδων μεταξύ τους, διαμορφώνοντας μια επιθυμητή ροή δεδομένων. Το μεγάλο πλεονέκτημα έγκειται στο ότι όλες αυτές οι συνδέσεις μπορούν να τροποποιηθούν δυναμικά, επιτρέποντας την υλοποίηση διαφορετικών κυκλωμάτων στο ίδιο φυσικό ολοκληρωμένο κύκλωμα. Τα μπλοκ εισόδου/εξόδου παρέχουν τη δυνατότητα διασύνδεσης του FPGA με το εξωτερικό περιβάλλον (π.χ. άλλες συσκευές, αισθητήρες, ή υπολογιστές). [12], [6], [11], [5], [20]

Ένα επιπλέον χαρακτηριστικό που καθιστά τα FPGA ισχυρά εργαλεία είναι η δυνατότητα ενσωμάτωσης επιπρόσθετων στοιχείων, όπως μνήμες SRAM, πολυπύρηνοι επεξεργαστές

(soft/hard processors), συστήματα χρονισμού (PLL), ακόμα και μονάδες DSP (Digital Signal Processing), επιτρέποντας τη χρήση τους σε πολύ απαιτητικές εφαρμογές. [12], [6], [11], [5], [20]

2.5.2 Πλεονεκτήματα των FPGA

Τα FPGAs χρησιμοποιούνται ευρέως στον σχεδιασμό ψηφιακών συστημάτων και έρχονται να συμπληρώσουν τον ρόλο των μικροεπεξεργαστών. Αν και οι μικροεπεξεργαστές είναι ευέλικτοι και μπορούν να χρησιμοποιηθούν σε πολλά διαφορετικά συστήματα, βασίζονται στο λογισμικό για να κάνουν τις λειτουργίες τους, πράγμα που τους κάνει πιο αργούς και πιο ενεργοβόρους σε σχέση με άλλα πιο εξειδικευμένα κυκλώματα. Από την άλλη, και τα FPGAs δεν είναι 100% εξειδικευμένα, οπότε δεν φτάνουν την απόδοση ενός κυκλώματος που έχει σχεδιαστεί αποκλειστικά για μια συγκεκριμένη δουλειά. Αυτό σημαίνει ότι τα FPGAs είναι γενικά πιο αργά, καταναλώνουν περισσότερη ενέργεια και συνήθως κοστίζουν περισσότερο από ένα εξειδικευμένο ολοκληρωμένο κύκλωμα.

Επίσης τα FPGAs έχουν καθιερωθεί ως ένα από τα πιο πολύτιμα εργαλεία στον τομέα της πληροφορικής και του προγραμματισμού, κυρίως λόγω της ικανότητάς τους να υλοποιούν εξειδικευμένες λογικές συναρτήσεις με υψηλές επιδόσεις και χαμηλή καθυστέρηση. Εν τούτοις, παρουσιάζουν σημαντικά πλεονεκτήματα κυρίως εξαιτίας του γεγονότος ότι αποτελούν τυποποιημένα κυκλώματα. [12], [6], [11], [5], [20]

- Δεν υπάρχει ανάγκη για αναμονή από τη στιγμή του σχεδιασμού του κυκλώματος μέχρι τη στιγμή του ελέγχου του ολοκληρωμένου. Το κύκλωμα μπορεί να προγραμματιστεί στο FPGA και να ελεγχθεί άμεσα.
- Τα FPGAs είναι τέλειο όχημα για τη πρωτοτυποποίηση. Εάν χρησιμοποιηθεί στο τελικό σχεδιασμό η μετάβαση από το πρωτότυπο σχέδιο στο τελικό προϊόν είναι βραχύχρονη και εύκολη διαδικασία.
- Το ίδιο FPGA μπορεί να χρησιμοποιηθεί στο σχεδιασμό πολλών κυκλωμάτων μειώνοντας αρκετά το κόστος.
- Ψηφιακή επεξεργασία σήματος (DSP).

  Σε εφαρμογές όπως η επεξεργασία ήχου, εικόνας ή βίντεο, η χρήση FPGA επιτρέπει την υλοποίηση πολύπλοκων αλγορίθμων σε πραγματικό χρόνο, αξιοποιώντας την παράλληλη επεξεργασία που παρέχουν. Επιπλέον, οι ειδικές μονάδες DSP που ενσωματώνονται σε

σύγχρονα FPGA καθιστούν δυνατή την υλοποίηση φίλτρων, μετασχηματισμών Fourier και άλλων μαθηματικών πράξεων με εξαιρετική ταχύτητα.

- Κρυπτογράφηση και ασφάλεια.

Τα FPGAs χρησιμοποιούνται συχνά στην υλοποίηση συστημάτων ασφαλείας, καθώς επιτρέπουν την παραμετροποίηση και γρήγορη ενσωμάτωση αλγορίθμων κρυπτογράφησης, όπως AES, RSA και SHA, προσφέροντας υψηλό επίπεδο προστασίας και παράλληλη επεξεργασία δεδομένων.

- Τεχνητή νοημοσύνη και μηχανική μάθηση.

Πρόσφατα, τα FPGAs αξιοποιούνται ολοένα και περισσότερο στην υλοποίηση νευρωνικών δικτύων και συστημάτων μηχανικής μάθησης. Η δυνατότητα παραμετροποίησης της αρχιτεκτονικής τους επιτρέπει τη δημιουργία βελτιστοποιημένων επιταχυντών για συγκεκριμένα μοντέλα AI, μειώνοντας δραστικά την κατανάλωση ενέργειας και βελτιώνοντας την απόδοση σε σχέση με λύσεις που βασίζονται αποκλειστικά σε CPU ή GPU.

- Εκπαίδευση και έρευνα.

Λόγω της ευκολίας στον σχεδιασμό, της ευελιξίας και της αμεσότητας στην προσομοίωση και υλοποίηση ψηφιακών κυκλωμάτων, τα FPGAs αποτελούν βασικό εργαλείο σε πανεπιστημιακά μαθήματα ψηφιακής σχεδίασης και αρχιτεκτονικής υπολογιστών, προσφέροντας στους φοιτητές την ευκαιρία να αποκτήσουν πρακτική εμπειρία με σύγχρονες τεχνολογίες υλικού.

Για πάρα πολλά χρόνια τα FPGAs χρησιμοποιούνταν κυρίως ως συνδετική λογική (glue-logic) και για τη προτυποποίηση τελικών κυκλωμάτων. Στη σύγχρονη εποχή χρησιμοποιούνται σε όλα τα είδη ψηφιακών συστημάτων και έχουν πλέον δεσπόζουσα θέση για υλοποιήσεις ψηφιακών συστημάτων. [12], [6], [11], [5], [20]

2.5.3 Ιστορική εξέλιξη των FPGA

Η ιδέα των επαναπρογραμματιζόμενων λογικών κυκλωμάτων ξεκίνησε τη δεκαετία του 1980, όταν η εταιρεία Xilinx παρουσίασε το πρώτο εμπορικό FPGA το 1985. Η πρόταση αυτή αποτέλεσε επανάσταση στον χώρο των ψηφιακών ηλεκτρονικών, δίνοντας στους μηχανικούς τη δυνατότητα να σχεδιάσουν και να υλοποιήσουν λογικές συναρτήσεις χωρίς την ανάγκη δημιουργίας ειδικών κυκλωμάτων κάθε φορά.

Αρχικά, τα FPGAs χρησιμοποιήθηκαν κυρίως σε εφαρμογές όπου απαιτούνταν γρήγορη πρωτοτυποποίηση (prototyping) ή όπου τα κόστη παραγωγής ειδικών ολοκληρωμένων κυκλωμάτων ήταν απαγορευτικά. Καθώς όμως εξελίχθηκε η τεχνολογία κατασκευής ολοκληρωμένων κυκλωμάτων, τα FPGA απέκτησαν μεγαλύτερη πυκνότητα στοιχείων, υψηλότερες ταχύτητες και μικρότερη κατανάλωση ενέργειας.

Σήμερα, οι μεγαλύτεροι κατασκευαστές FPGA περιλαμβάνουν τις Xilinx (που εξαγοράστηκε από την AMD), Intel (μέσω της εξαγοράς της Altera), και Lattice Semiconductor. Η χρήση των FPGA έχει επεκταθεί σε κρίσιμους τομείς όπως η αεροδιαστημική, οι τηλεπικοινωνίες, η ασφάλεια πληροφοριών και η τεχνητή νοημοσύνη. [12], [6], [11], [5], [20]

3. QUARTUS PRIME & ALTERA

Το Quartus Prime αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τον σχεδιασμό ψηφιακών κυκλωμάτων σε προγραμματιζόμενες λογικές συσκευές, όπως τα FPGA (Field-Programmable Gate Arrays) και τα CPLD (Complex Programmable Logic Devices). Αρχικά αναπτύχθηκε από την εταιρεία Altera και στη συνέχεια εξελίχθηκε υπό την αιγίδα(προστασία) της Intel μετά την εξαγορά της Altera το 2015. Το λογισμικό αυτό παρέχει εργαλεία για την ανάλυση, σύνθεση, προσομοίωση και προγραμματισμό ψηφιακών σχεδίων, υποστηρίζοντας γλώσσες περιγραφής υλικού όπως η VHDL και η Verilog.

Η Altera, ιδρυθείσα το 1983, υπήρξε πρωτοπόρος στην ανάπτυξη προγραμματιζόμενων λογικών συσκευών, προσφέροντας λύσεις που επέτρεψαν την ευελιξία και την προσαρμοστικότητα στον σχεδιασμό ψηφιακών συστημάτων. Με την πάροδο των ετών, η εταιρεία ανέπτυξε μια σειρά προϊόντων, όπως οι οικογένειες FPGA Cyclone, Arria και Stratix, καθώς και το λογισμικό Quartus, το οποίο αποτέλεσε βασικό εργαλείο για την αξιοποίηση των δυνατοτήτων αυτών των συσκευών. [1], [2], [3], [4], [20]

3.1 Ιστορική Αναδρομή της Altera και του Quartus Prime

Η Altera ιδρύθηκε το 1983 από τους Rodney Smith, Robert Hartmann, James Sansbury και Paul Newhagen, με στόχο την ανάπτυξη προγραμματιζόμενων λογικών συσκευών που θα προσέφεραν ευελιξία στον σχεδιασμό ψηφιακών κυκλωμάτων. Το 1988, η εταιρεία εισήχθη στο χρηματιστήριο, ενισχύοντας τη θέση της στην αγορά των ημιαγωγών. Κατά τη διάρκεια των επόμενων δεκαετιών, η Altera παρουσίασε καινοτόμα προϊόντα, όπως οι οικογένειες FPGA Cyclone, Arria και Stratix, καθώς και το λογισμικό σχεδίασης Quartus.

Το Quartus, αρχικά γνωστό ως MAX+PLUS II, εξελίχθηκε σε Quartus II και αργότερα σε Quartus Prime, προσφέροντας ολοκληρωμένες λύσεις για τον σχεδιασμό και την υλοποίηση ψηφιακών συστημάτων. Το 2015, η Intel εξαγόρασε την Altera, ενσωματώνοντας το χαρτοφυλάκιό της στις δικές της λύσεις και συνεχίζοντας την ανάπτυξη του Quartus Prime υπό τη δική της αιγίδα. [1], [2], [3], [4], [20]

3.2 Χρήση και Δυνατότητες του Quartus Prime

Το Quartus Prime είναι ένα ισχυρό εργαλείο σχεδίασης που υποστηρίζει όλες τις φάσεις ανάπτυξης ψηφιακών κυκλωμάτων σε FPGA και CPLD. Οι βασικές δυνατότητές του περιλαμβάνουν:

- Εισαγωγή Σχεδίου: Υποστήριξη για γλώσσες περιγραφής υλικού όπως VHDL και Verilog, καθώς και γραφικά εργαλεία για τη σχεδίαση κυκλωμάτων.
- Σύνθεση και Βελτιστοποίηση: Μετατροπή του σχεδίου σε βελτιστοποιημένο κύκλωμα για την επιλεγμένη συσκευή.
- Προσομοίωση και Επαλήθευση: Ενσωματωμένα εργαλεία για την προσομοίωση της λειτουργίας του κυκλώματος και την επαλήθευση της σωστής λειτουργίας του.
- Προγραμματισμός Συσκευής: Δυνατότητα προγραμματισμού της τελικής συσκευής με το σχεδιασμένο κύκλωμα.

Επιπλέον, το Quartus Prime προσφέρει εργαλεία όπως το Platform Designer για την ενσωμάτωση IP πυρήνων, το TimeQuest Timing Analyzer για την ανάλυση χρονισμού, και το DSP Builder για τη σχεδίαση ψηφιακών συστημάτων επεξεργασίας σήματος. [1], [2], [3], [4], [20]

3.3 Πλεονεκτήματα και Μειονεκτήματα του Quartus Prime

Πλεονεκτήματα του Quartus

- Ολοκληρωμένο Περιβάλλον Ανάπτυξης (All-in-One IDE)

Το Quartus Prime προσφέρει ένα πλήρες και ενοποιημένο περιβάλλον σχεδίασης ψηφιακών συστημάτων, που περιλαμβάνει:

1. Εργαλεία σύνθεσης (synthesis)
2. Ανάλυση χρονισμού (timing analysis)
3. Προσομοίωση (simulation)
4. Τοποθέτηση και δρομολόγηση (place and route)
5. Δημιουργία bitstream και προγραμματισμό της FPGA

Αυτό μειώνει την ανάγκη για εξωτερικά ή τρίτα εργαλεία, ενισχύοντας την παραγωγικότητα και την εστίαση του σχεδιαστή στο κύριο έργο του.

- Υποστήριξη Πολλαπλών Γλωσσών Περιγραφής Υλικού (HDL)

Το Quartus υποστηρίζει τόσο VHDL όσο και Verilog, επιτρέποντας στον χρήστη να εργαστεί στη γλώσσα με την οποία είναι περισσότερο εξοικειωμένος ή να συνδυάσει και τις δύο. Αυτό προσφέρει ευελιξία και δια λειτουργικότητα με υπάρχοντα σχέδια.

- Δυνατότητα Ενσωμάτωσης IP Πυρήνων (Intellectual Property Cores)

Μέσω εργαλείων όπως ο Platform Designer (παλαιότερα Qsys), το Quartus Prime διευκολύνει την ένθεση προκατασκευασμένων IP blocks όπως ALUs, DSP blocks, controllers, memory interfaces κ.ά.

Αυτά τα IP είναι συνήθως βελτιστοποιημένα για τις Altera/Intel FPGAs και επιταχύνουν σημαντικά τον χρόνο ανάπτυξης.

- Εξειδικευμένα Εργαλεία για Χρονισμό (TimeQuest Timing Analyzer)

Το TimeQuest προσφέρει δυνατότητες στατικού χρονισμού με ακρίβεια και λεπτομέρεια, κάτι απαραίτητο για σύνθετα σχέδια υψηλής συχνότητας. Υποστηρίζει constraints μέσω SDC (Synopsys Design Constraints), όπως σε εργαλεία ASIC.

- Υποστήριξη Σύγχρονων FPGA Οικογενειών

Υποστηρίζει προηγμένες FPGA οικογένειες όπως τις Stratix, Arria και Cyclone, με υποστήριξη για νέες δυνατότητες όπως transceivers, hard processors (SoC), και embedded memory blocks.

- Καλή Τεκμηρίωση και Υποστήριξη

Παρέχεται πλήρης τεκμηρίωση για όλα τα εργαλεία του λογισμικού και ευρεία κοινότητα υποστήριξης. Η Intel διαθέτει online guides, application notes, και video tutorials για την βοήθεια και επίλυση όλων των προβλημάτων που μπορεί να αντιμετωπίσουν οι χρήστες. [1], [2], [3], [4], [20]

Μειονεκτήματα του Quartus

- Περιορισμένη Υποστήριξη για FPGA άλλων Κατασκευαστών.

Το Quartus Prime υποστηρίζει μόνο FPGA της Intel (Altera). Δεν μπορεί να χρησιμοποιηθεί για FPGA της Xilinx, Lattice ή άλλων εταιρειών. Αυτό περιορίζει τη δυνατότητα επαναχρησιμοποίησης σχεδίων ή της τεχνογνωσίας του χρήστη σε άλλες πλατφόρμες.

- Υψηλές Απαιτήσεις Υπολογιστικών Πόρων.

Η σύνθεση, η ανάλυση χρονισμού και η δρομολόγηση (P&R) μεγάλων σχεδίων απαιτούν ισχυρό υπολογιστικό σύστημα. Ειδικά σε μεγάλα FPGA (π.χ. Stratix), η διαδικασία μπορεί να διαρκεί ώρες, ακόμα και σε υπολογιστές με πολυπύρηνους επεξεργαστές και μεγάλες ποσότητες RAM.

- Σχετικά Απότομη Καμπύλη Μάθησης.

Το πλήθος των εργαλείων, των παραθύρων, των ρυθμίσεων και των διαδικασιών μπορεί να είναι αποθαρρυντικό για αρχάριους χρήστες. Η πλήρης κατανόηση του Project Navigator, του TimeQuest, του Platform Designer, και του Pin Planner απαιτεί αρκετό χρόνο εξοικείωσης.

- Περιορισμένη Ενσωμάτωση με Εξωτερικά Εργαλεία Simulators.

Αν και διαθέτει εσωτερικό προσομοιωτή (ModelSim-Altera Edition), η χρήση τρίτων εργαλείων όπως το full ModelSim, Vivado Simulator ή άλλων commercial simulators μπορεί να απαιτεί χειροκίνητη ρύθμιση, προκαλώντας καθυστερήσεις.

- Λογισμικές Εκδόσεις με Διαφορές.

Το Quartus Prime διατίθεται σε διαφορετικές εκδόσεις:

1. Lite Edition: Δωρεάν αλλά με περιορισμένες δυνατότητες.
2. Standard Edition: Για τις mainstream συσκευές.
3. Pro Edition: Για τις high-end FPGA (π.χ. Stratix 10).

Οι λειτουργίες που είναι διαθέσιμες σε κάθε έκδοση διαφέρουν, και αυτό μπορεί να δημιουργήσει προβλήματα ασυμβατότητας μεταξύ ομάδων ή έργων. [1], [2], [3], [4], [20]

Συνοψίζοντας, το Quartus Prime αποτελεί ένα ισχυρό και ολοκληρωμένο εργαλείο για τον σχεδιασμό ψηφιακών κυκλωμάτων σε FPGA και CPLD, προσφέροντας μια σειρά από δυνατότητες που καλύπτουν όλες τις φάσεις ανάπτυξης. Η ιστορία και η εξέλιξή του αντικατοπτρίζουν την πρόοδο της Altera και αργότερα της Intel στον τομέα των προγραμματιζόμενων λογικών συσκευών. Παρά τα μειονεκτήματά του, το Quartus Prime παραμένει ένα από τα βασικά εργαλεία για μηχανικούς και σχεδιαστές ψηφιακών συστημάτων. [1], [2], [3], [4], [20]

4. Ανάλυση κώδικα φίλτρου FIR ειδικού σκοπού

4.1 Φίλτρο FIR main κώδικας

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
```

Δηλώνει τη χρήση της τυπικής βιβλιοθήκης ieee, η οποία περιέχει τα περισσότερα χρήσιμα πακέτα για τον χειρισμό σημάτων και τύπων. Το std_logic_1164 είναι απαραίτητο για τη χρήση των τύπων std_logic και std_logic_vector, που αποτελούν τη βάση για την περιγραφή ψηφιακών κυκλωμάτων. Το std_logic_arith χρησιμοποιείται για αριθμητικές πράξεις σε std_logic_vector, αν και σε σύγχρονο VHDL συνιστάται περισσότερο το numeric_std.

```
ENTITY FIR1 IS
   GENERIC (n: INTEGER := 4; m: INTEGER :=4);
   --n = # του coef., m = # bit των input και coef. εκτός από τα n και m χρειάζεται και ρύθμιση του CONSTANT.
   PORT ( x:IN SIGNED(m-1 DOWNTO 0);
      clk, rst: IN STD_LOGIC;
      y: OUT SIGNED(2*m-1 DOWNTO 0));
END FIR1;
```

Δήλωση της «μαύρης γραμμής» του FIR.

n → πλήθος συντελεστών (tap order).

m → bit-width εισόδου & coefficients.

x → τρέχον δείγμα,

y → έξοδος με διπλάσιο εύρος (αποφυγή overflow).

```
ARCHITECTURE rtl OF FIR1 IS
   TYPE registers IS ARRAY (n-2 DOWNTO 0) OF
      SIGNED(m-1 DOWNTO 0);
   TYPE coefficients is ARRAY (n-1 DOWNTO 0) OF
      SIGNED(m-1 DOWNTO 0);
   SIGNAL reg: registers;
   CONSTANT coef: coefficients := ("0101", "1010", "1111", "0100");
```

registers: πίνακας με τα n-1 προηγούμενα δείγματα (shift register).

coefficients: πίνακας σταθερών συντελεστών.

coef: συγκεκριμένες τιμές (δείγμα).

reg: ίδιος τύπος με το shift-register.

```
BEGIN
   PROCESS(clk, rst)
      VARIABLE acc, prod:
      SIGNED(2*m-1 DOWNTO 0) := (OTHERS=>'0');
      VARIABLE sign: STD_LOGIC;
   BEGIN
-- Ασύγχρονη επαναφορά (reset)
      IF (rst='1') THEN
      FOR i IN n-2 DOWNTO 0 LOOP
         FOR j IN m-1 DOWNTO 0 LOOP
            reg(i) (j) <= '0';
         END LOOP;
      END LOOP;
```

1. Ορισμός ενδιάμεσων μεταβλητών
   - acc και prod : Signed μεταβλητές πλάτους 2*m bit, αρχικοποιημένες στο 0. Θα φιλοξενήσουν το άθροισμα και το γινόμενο κάποιας υπο-μονάδας (π.χ. MAC).
   - sign : ένα bit τύπου STD_LOGIC, μάλλον για το πρόσημο.
2. Ασύγχρονη επαναφορά (reset)
   - Όταν το σήμα rst γίνει '1', εκκαθαρίζεται ένας δισδιάστατος πίνακας καταχωρητών reg(i)(j) σε μηδενικά.
   - Ο εξωτερικός βρόχος “σκανάρει” τις n-1 γραμμές καταχωρητών.
   - Ο εσωτερικός βρόχος μηδενίζει τα m bit κάθε γραμμής.

```
-- Δημιουργία καταχωρητή + MAC
      ELSIF (clk'EVENT AND clk='1') THEN
         acc := coef(0)*x;
         FOR i IN 1 TO n-1 LOOP
            sign := acc(2*m-1);
            prod := coef(i)*reg(n-1-i);
            acc := acc + prod;
```

Υλοποίηση του βασικού βρόχου ενός FIR φίλτρου με MAC, όπου η νέα τιμή εξόδου προκύπτει από το άθροισμα πολλαπλασιασμών εισόδου επί τους συντελεστές (tap weights), βασισμένο σε χρονικές καθυστερήσεις της εισόδου.

Πρόκειται δηλαδή για το υπολογιστικό τμήμα ενός φίλτρου FIR.

```
-- Δημιουργία καταχωρητή + MAC
      ELSIF (clk'EVENT AND clk='1') THEN
         acc := coef(0)*x;
         FOR i IN 1 TO n-1 LOOP
            sign := acc(2*m-1);
            prod := coef(i)*reg(n-1-i);
            acc := acc + prod;
```

Υλοποίηση βασικού βρόχου υπολογιστικού τμήματος φίλτρου FIR με MAC, όπου η έξοδος προκύπτει από το άθροισμα γινομένων συντελεστών με καθυστερημένες τιμές εισόδου.

```
-- Έλεγχος υπερχείλισης
            IF (sign=prod(prod'left)) AND
               (acc(acc'left) /= sign)
            THEN
            acc := (acc'LEFT => sign,
            OTHERS => NOT sign);
            END IF;
         END LOOP;
         reg <= x & reg(m-2 DOWNTO 1);
      END IF;
      y <= acc;
   END PROCESS;
END rtl;
```

Έλεγχος υπερχείλισης αποτελέσματος MAC, με διόρθωση πρόσημου σε περίπτωση ασυμφωνίας μεταξύ πρόσημου του γινομένου και του συσσωρευτή.
Ενσωματώνεται επίσης η λειτουργία μετατόπισης καταχωρητών και ενημέρωση της εξόδου.

4.2 Test Bench φίλτρου FIR

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY tb_FIR1 IS
END tb_FIR1;

ARCHITECTURE behavior OF tb_FIR1 IS

    -- Constants for generics
    CONSTANT n : INTEGER := 4;
    CONSTANT m : INTEGER := 4;

    -- Signals to connect to the FIR1 module
    SIGNAL clk  : STD_LOGIC := '0';
    SIGNAL rst  : STD_LOGIC := '0';
    SIGNAL x    : SIGNED(m-1 DOWNTO 0) := (OTHERS => '0');
    SIGNAL y    : SIGNED(2*m-1 DOWNTO 0);

    -- Clock period definition
    CONSTANT clk_period : TIME := 10 ns;
```

Το τμήμα αυτό αποτελεί το αρχικό στάδιο της αρχιτεκτονικής του testbench. Πιο συγκεκριμένα:

- Δηλώνει τη χρήση της τυπικής βιβλιοθήκης ieee, η οποία περιέχει τα περισσότερα χρήσιμα πακέτα για τον χειρισμό σημάτων και τύπων. Το std_logic_1164 είναι απαραίτητο για τη χρήση των τύπων std_logic και std_logic_vector, που αποτελούν τη βάση για την περιγραφή ψηφιακών κυκλωμάτων. Το std_logic_arith χρησιμοποιείται για αριθμητικές πράξεις σε std_logic_vector, αν και σε σύγχρονο VHDL συνιστάται περισσότερο το numeric_std.
- Ορίζονται δύο ακέραιες σταθερές n και m, οι οποίες καθορίζουν το πλήθος των δειγμάτων και το εύρος των δεδομένων εισόδου αντίστοιχα.
- Δημιουργούνται τα απαραίτητα σήματα για τη σύνδεση με το φίλτρο FIR (FIR1), όπως το ρολόι (clk), η επαναφορά (rst), η είσοδος (x) και η έξοδος (y).
- Ορίζεται και η χρονική περίοδος του ρολογιού (clk_period) ίση με 10 νανοδευτερόλεπτα.

Το συγκεκριμένο κομμάτι αποτελεί τη βασική προετοιμασία για τη δημιουργία ενός testbench που θα χρησιμοποιηθεί για την επαλήθευση του κυκλώματος FIR.

```
BEGIN

    -- Instantiate the FIR1 Unit Under Test (UUT)
    uut: ENTITY work.FIR1
        GENERIC MAP (n => n, m => m)
        PORT MAP (
            x   => x,
            clk => clk,
            rst => rst,
            y   => y
        );

    -- Clock process
    clk_process : PROCESS
```

Αυτό το κομμάτι κώδικα είναι υπεύθυνο για την ενσωμάτωση της μονάδας προς δοκιμή (Unit Under Test - UUT) στο testbench. Αναλυτικά:

- Η μονάδα FIR1 καλείται ως οντότητα από το τρέχον work library.
- Με χρήση του GENERIC MAP αντιστοιχίζονται οι γενικές παράμετροι n και m που καθορίστηκαν προηγουμένως.
- Η PORT MAP πραγματοποιεί τη σύνδεση των θυρών εισόδου και εξόδου της μονάδας FIR1 με τα αντίστοιχα σήματα του testbench.

Αυτό το κομμάτι επιτρέπει τη σύνδεση και αλληλεπίδραση μεταξύ του testbench και της υπό δοκιμή σχεδίασης.

```
    BEGIN
        WHILE TRUE LOOP
            clk <= '0';
            WAIT FOR clk_period / 2;
            clk <= '1';
            WAIT FOR clk_period / 2;
        END LOOP;
    END PROCESS;

    -- Stimulus process
    stim_proc: PROCESS
```

Η διαδικασία αυτή αναπαριστά τον παλμογεννήτρια του σήματος ρολογιού (clock generator). Αναλυτικά:

- Η διαδικασία clk_process εκτελείται επ’ άπειρον (με χρήση του WHILE TRUE LOOP).
- Το σήμα clk εναλλάσσεται περιοδικά μεταξύ '0' και '1', δημιουργώντας παλμούς ρολογιού με περίοδο που ορίζεται από τη σταθερά clk_period.
- Με αυτόν τον τρόπο εξασφαλίζεται η χρονική βάση για τη λειτουργία της μονάδας FIR.

Αυτό το υποσύστημα είναι απαραίτητο για οποιαδήποτε σύγχρονη ψηφιακή σχεδίαση, καθώς η μονάδα FIR λειτουργεί με βάση το ρολόι.

```
    BEGIN
        -- Reset
        rst <= '1';
        WAIT FOR 20 ns;
        rst <= '0';

        -- Apply input samples (example values)
        x <= "0101"; -- 1
        WAIT FOR clk_period;
        x <= "1010"; -- 2
        WAIT FOR clk_period;
        x <= "1111"; -- 3
        WAIT FOR clk_period;
        x <= "0100"; -- 4
        WAIT FOR clk_period;
        x <= "0000"; -- 0 (optional test case)
        WAIT FOR 4 * clk_period;

        -- Stop simulation
        WAIT;
    END PROCESS;

END behavior;
```

Το συγκεκριμένο κομμάτι κώδικα αποτελεί τμήμα διαδικασίας (PROCESS) σε VHDL testbench και χρησιμοποιείται για την προσομοίωση ενός ψηφιακού κυκλώματος. Αρχικά εφαρμόζεται σήμα reset για 20 ns, ώστε να αρχικοποιηθεί το κύκλωμα σε γνωστή κατάσταση. Στη συνέχεια δίνονται διαφορετικές δυαδικές τιμές στο σήμα εισόδου x με χρονικές καθυστερήσεις ίσες με την περίοδο ρολογιού, επιτρέποντας τον έλεγχο της λειτουργίας του κυκλώματος σε διαφορετικές συνθήκες εισόδου. Τέλος, η προσομοίωση σταματά με εντολή αναμονής. Η δομή αυτή είναι χαρακτηριστική για testbenches, επιτρέποντας τον έλεγχο και την επαλήθευση της σωστής συμπεριφοράς του σχεδίου πριν την υλοποίηση.

4.3 Επεξήγηση Μεθοδολογίας

Στη σχεδίαση και προσομοίωση του FIR φίλτρου χρησιμοποιήθηκε η γλώσσα περιγραφής υλικού VHDL. Η μεθοδολογία που ακολουθήθηκε περιλαμβάνει τα εξής στάδια:

- Σχεδίαση του κυκλώματος (FIR1):

Το φίλτρο FIR υλοποιείται ως αρχιτεκτονική «rtl», όπου χρησιμοποιείται μια δομή μετατόπισης (shift register) για την αποθήκευση προηγούμενων δειγμάτων εισόδου και ένας MAC (Multiply-Accumulate) βρόχος για τον υπολογισμό της εξόδου. Τα συντελεστές (coefficients) ορίζονται ως σταθερός πίνακας:

coef := ("0001", "0010", "0011", "0100");

Αντιστοιχούν σε ακέραιες τιμές 1, 2, 3, 4.

- Χρήση ασύγχρονου reset:

Το σήμα «rst» καθαρίζει τα εσωτερικά register στην αρχή της προσομοίωσης, διασφαλίζοντας γνωστή αρχική κατάσταση.

- Σχεδίαση testbench (tb_FIR1):

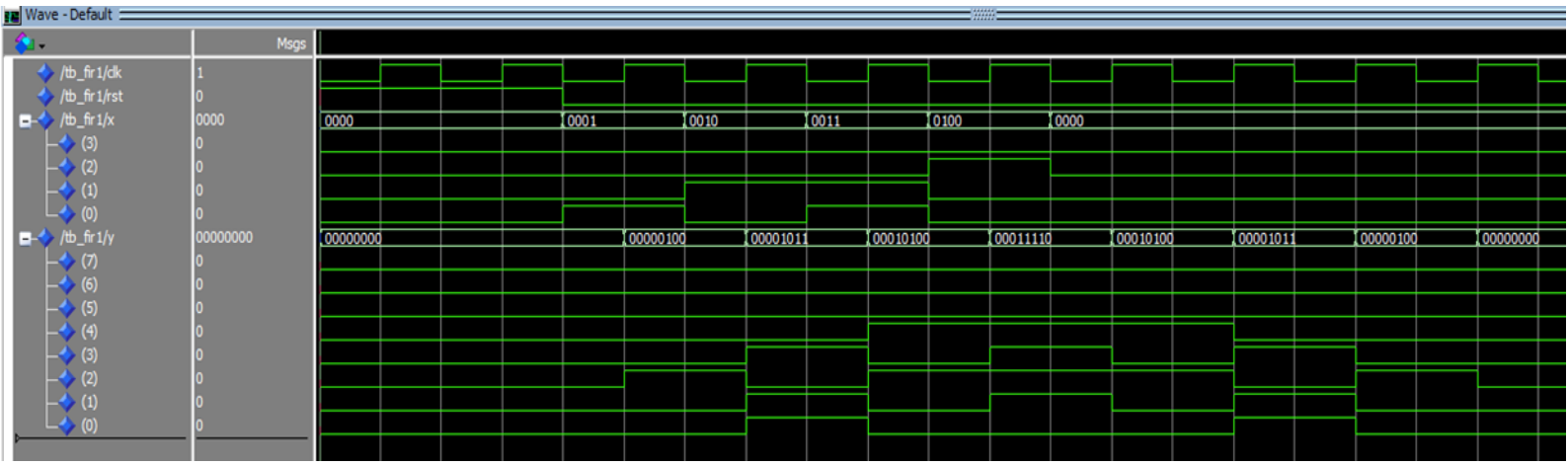
Περιλαμβάνει:

  - Ρολογιακή διεργασία (10 ns period).
  - Stimulus διεργασία που εφαρμόζει διαδοχικά δείγματα στο x: 1, 2, 3, 4, 0.
  - Αναμονή μεταξύ των δειγμάτων ώστε να συγχρονιστούν με το ρολόι.

- Συλλογή και ανάλυση κυματομορφών:

Η προσομοίωση καταγράφει σήματα όπως clk, rst, x και y. Η ανάλυση των waveforms επιβεβαιώνει τη λειτουργία του φίλτρου.

4.4 Ανάλυση Αποτελεσμάτων



Στην εικόνα εμφανίζονται τα σήματα:

- /tb_fir1/clk: Σήμα ρολογιού. Σταθερή συχνότητα (10 ns period).
- /tb_fir1/rst: Υψηλό μόνο στην αρχή (reset).
- /tb_fir1/x: Είσοδος φίλτρου (ακολουθία 0001, 0010, 0011, 0100, 0000).
- /tb_fir1/y: Έξοδος φίλτρου (αποτελέσματα MAC).

Η είσοδος x μεταβάλλεται ανά ακμή ρολογιού μετά το reset. Οι καταχωρητές στο φίλτρο μετακινούν τις προηγούμενες τιμές εισόδου και κάθε νέα έξοδος y υπολογίζεται ως:

$$y = coef(0)*x(n) + coef(1)*x(n-1) + coef(2)*x(n-2) + coef(3)*x(n-3)$$

με coefficients = (1, 2, 3, 4).

Στην εικόνα παρατηρούμε:

- Στην αρχή, η έξοδος y είναι 0 λόγω του reset.
- Μετά από τις πρώτες εισόδους (4, 3, 2, 1), η έξοδος αυξάνεται καθώς το φίλτρο "γεμίζει" με δείγματα.
- Οι έξοδοι είναι πολυψήφιοι δυαδικοί αριθμοί (8-bit λόγω 2*m), αντιπροσωπεύοντας το άθροισμα των γινομένων των συντελεστών με τις αντίστοιχες καθυστερημένες εισόδους.
- Όταν το x επιστρέφει σε 0000, η έξοδος μειώνεται σταδιακά καθώς το shift register "ξεπλένεται" από τις προηγούμενες τιμές.

Η μορφή των κυματομορφών δείχνει καθαρά το φαινόμενο της «απόκρισης» του φίλτρου σε μια ακολουθία παλμών εισόδου και επιβεβαιώνει ότι η συνάρτηση μεταφοράς έχει υλοποιηθεί σωστά.

5 Ανάλυση κώδικα φίλτρου FIR γενικού σκοπού

5.1 Φίλτρο FIR main κώδικας

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
```

Δηλώνει τη χρήση της τυπικής βιβλιοθήκης ieee, η οποία περιέχει τα περισσότερα χρήσιμα πακέτα για τον χειρισμό σημάτων και τύπων. Το std_logic_1164 είναι απαραίτητο για τη χρήση των τύπων std_logic και std_logic_vector, που αποτελούν τη βάση για την περιγραφή ψηφιακών κυκλωμάτων. Το std_logic_arith χρησιμοποιείται για αριθμητικές πράξεις σε std_logic_vector, αν και σε σύγχρονο VHDL συνιστάται περισσότερο το numeric_std.

```
ENTITY FIR2 IS
  GENERIC (
    n : INTEGER := 4; -- taps
    m : INTEGER := 4  -- bit-width
  );
  PORT (
    x       : IN  SIGNED(m-1 DOWNTO 0);
    coef_in : IN  SIGNED(m-1 DOWNTO 0); -- σειριακή είσοδος coefficient
    load    : IN  STD_LOGIC;            -- φόρτωση coefficient
    clk     : IN  STD_LOGIC;
    rst     : IN  STD_LOGIC;
    y       : OUT SIGNED(2*m-1 DOWNTO 0)
  );
END FIR2;
```

Αυτό το entity περιγράφει τη διεπαφή ενός παραμετρικού FIR φίλτρου (με αριθμό taps και bit-width που μπορεί να αλλάξει), όπου οι συντελεστές φορτώνονται σειριακά μέσω της coef_in, και η έξοδος είναι το φιλτραρισμένο σήμα.

Generic παράμετροι:

- n = taps (πλήθος συντελεστών / βαθμός φίλτρου).
- m = bit-width (αριθμός bits εισόδου και coefficients).

Port (θύρες εισόδου/εξόδου):

- x → είσοδος δεδομένων (m bits signed).

- coef_in → σειριακή είσοδος για τους coefficients (m bits signed).
- load → σήμα ελέγχου για να φορτώνει coefficient.
- clk → ρολόι.
- rst → reset.
- y → έξοδος του φίλτρου (διπλάσιο εύρος bits: 2*m, γιατί προκύπτει από πολλαπλασιασμούς και αθροίσεις).

```
ARCHITECTURE rtl OF FIR2 IS
  TYPE reg_array IS ARRAY (0 TO n-2) OF SIGNED(m-1 DOWNTO 0);
  TYPE coef_array IS ARRAY (0 TO n-1) OF SIGNED(m-1 DOWNTO 0);

  SIGNAL reg  : reg_array := (OTHERS => (OTHERS => '0'));
  SIGNAL coef : coef_array := (OTHERS => (OTHERS => '0'));
  SIGNAL coef_index : INTEGER RANGE 0 TO n-1 := 0;
```

Εδώ στήνονται οι εσωτερικές δομές του φίλτρου FIR — η μνήμη για τα προηγούμενα δείγματα, η μνήμη για τους coefficients, και ένας δείκτης για τη διαδικασία φόρτωσής τους.

- TYPE reg_array → πίνακας από registers για τις προηγούμενες τιμές της εισόδου (shift register). Έχει μέγεθος n-1 (γιατί το τρέχον δείγμα είναι το x).
- TYPE coef_array → πίνακας που κρατάει τους συντελεστές (coefficients) του FIR φίλτρου, πλήθους n.
- SIGNAL reg → το shift register, αρχικοποιημένο με μηδενικά.
- SIGNAL coef → ο πίνακας coefficients, επίσης αρχικοποιημένος με μηδενικά.
- SIGNAL coef_index → δείκτης που δείχνει σε ποιον συντελεστή θα γραφτεί το επόμενο coef_in όταν φορτώνονται coefficients.

```
BEGIN
  PROCESS(clk, rst)
    VARIABLE acc  : SIGNED(2*m-1 DOWNTO 0) := (OTHERS => '0');
    VARIABLE prod : SIGNED(2*m-1 DOWNTO 0);
    VARIABLE sign : STD_LOGIC;
```

Εδώ ξεκινάει ο κύριος process που συγχρονίζεται με το clk και το rst, και ορίζει τις ενδιάμεσες μεταβλητές για την υλοποίηση του MAC **(**Multiply-Accumulate**)**, που είναι η βασική πράξη ενός FIR φίλτρου.

- acc (accumulator) → signed μεταβλητή με εύρος 2*m bits, αρχικοποιημένη στο 0. Χρησιμοποιείται για να μαζεύει (αθροίζει) τα γινόμενα δείγμα × coefficient.
- prod (product) → signed μεταβλητή 2*m bits. Χρησιμοποιείται για το προσωρινό αποτέλεσμα του πολλαπλασιασμού δείγμα × coefficient.
- sign → μεταβλητή τύπου STD_LOGIC. Χρησιμοποιείται πιθανόν για έλεγχο πρόσημου ή για εσωτερικό flag.

```
  BEGIN
    IF rst = '1' THEN
      reg  <= (OTHERS => (OTHERS => '0'));
      coef <= (OTHERS => (OTHERS => '0'));
      coef_index <= 0;
      y <= (OTHERS => '0');

    ELSIF rising_edge(clk) THEN
      -- Σειριακή φόρτωση συντελεστών
      IF load = '1' THEN
      -- Shift προς τα δεξιά (από μικρό προς μεγαλύτερο index)
         FOR i IN n-1 DOWNTO 1 LOOP
            coef(i) <= coef(i-1);
         END LOOP;
         coef(0) <= coef_in;


      -- Κανονική λειτουργία φίλτρου
      ELSE
        acc := coef(0) * x;

        FOR i IN 1 TO n-1 LOOP
          prod := coef(i) * reg(i-1);
          sign := acc(2*m-1);
          acc := acc + prod;

          -- Έλεγχος υπερχείλισης
          IF (sign = prod(prod'LEFT)) AND (acc(acc'LEFT) /= sign) THEN
            acc := (acc'LEFT => sign, OTHERS => NOT sign);
          END IF;
        END LOOP;

        -- Shift register
      FOR i IN n-2 DOWNTO 0 LOOP
         IF i = 0 THEN
            reg(0) <= x;
         ELSE
            reg(i) <= reg(i-1);
         END IF;
      END LOOP;

        y <= acc;
      END IF;
    END IF;
  END PROCESS;
END rtl;
```

Αυτό είναι το κύριο σώμα του φίλτρου FIR, δηλαδή η πραγματική λειτουργεία του φίλτρου, δηλαδή εδώ γίνεται

- Η φόρτωση των coefficients όταν χρειάζεται,
- Οι πολλαπλασιασμοί και τα αθροίσματα (MAC),
- Κρατάει τις προηγούμενες τιμές με shift register και
- Δίνει την έξοδο y.

5.2 Test Bench φίλτρου FIR

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY tb_FIR2 IS
END tb_FIR2;
```

Δηλώνει τη χρήση της τυπικής βιβλιοθήκης ieee, η οποία περιέχει τα περισσότερα χρήσιμα πακέτα για τον χειρισμό σημάτων και τύπων. Το std_logic_1164 είναι απαραίτητο για τη χρήση των τύπων std_logic και std_logic_vector, που αποτελούν τη βάση για την περιγραφή ψηφιακών κυκλωμάτων. Το std_logic_arith χρησιμοποιείται για αριθμητικές πράξεις σε std_logic_vector, αν και σε σύγχρονο VHDL συνιστάται περισσότερο το numeric_std.

Ορίζεται η οντότητα tb_FIR2. Επειδή είναι testbench, δεν έχει θύρες εισόδου/εξόδου (ports), αφού χρησιμοποιείται μόνο για προσομοίωση και δεν θα συντεθεί σε υλικό.

```
ARCHITECTURE behavior OF tb_FIR2 IS
  CONSTANT n : INTEGER := 4;
  CONSTANT m : INTEGER := 4;

  SIGNAL x         : SIGNED(m-1 DOWNTO 0) := (OTHERS => '0');
  SIGNAL coef_in   : SIGNED(m-1 DOWNTO 0) := (OTHERS => '0');
  SIGNAL load      : STD_LOGIC := '0';
  SIGNAL clk       : STD_LOGIC := '0';
  SIGNAL rst       : STD_LOGIC := '1';
  SIGNAL y         : SIGNED(2*m-1 DOWNTO 0);
  CONSTANT clk_period : TIME := 20 ns;
```

Αρχή της αρχιτεκτονικής του testbench με όνομα behavior. Εδώ δηλώνονται οι σταθερές και τα σήματα που θα χρησιμοποιηθούν για την προσομοίωση.

Αυτό το κομμάτι κώδικα ορίζει τα σήματα και τις σταθερές που θα χρησιμοποιηθούν στο testbench tb_FIR2. Δημιουργεί τα δεδομένα εισόδου, τους συντελεστές, τα σήματα ελέγχου (load, reset, clock) και την έξοδο, καθώς και την περίοδο ρολογιού. Είναι το "περιβάλλον" μέσα στο οποίο θα τρέξει το FIR φίλτρο κατά την προσομοίωση.

```
COMPONENT FIR2
  GENERIC (
    n : INTEGER := 4;
    m : INTEGER := 4
  );
  PORT (
    x       : IN  SIGNED(m-1 DOWNTO 0);
    coef_in : IN  SIGNED(m-1 DOWNTO 0);
    load    : IN  STD_LOGIC;
    clk     : IN  STD_LOGIC;
    rst     : IN  STD_LOGIC;
    y       : OUT SIGNED(2*m-1 DOWNTO 0)
  );
END COMPONENT;
```

Αυτό το κομμάτι δηλώνει τον component FIR2, δηλαδή το ίδιο το FIR φίλτρο που θα καλεστεί μέσα στο testbench. Ορίζει παραμέτρους μεγέθους (bits) για τα δεδομένα και τους συντελεστές.

- x (IN): είσοδος δεδομένων (m-bit signed).
- coef_in (IN): είσοδος συντελεστών (m-bit signed).
- load (IN): σήμα φόρτωσης συντελεστών.
- clk (IN): ρολόι.
- rst (IN): reset.
- y (OUT): έξοδος αποτελέσματος (διπλάσιο εύρος, 2m-bit signed).

```
BEGIN
  -- Συνδέσεις
  uut: ENTITY work.FIR2
    GENERIC MAP (
      n => n,
      m => m
    )
    PORT MAP (
      x       => x,
      coef_in => coef_in,
      load    => load,
      clk     => clk,
      rst     => rst,
      y       => y
    );

  -- Ρολογάκι
  clk_process : PROCESS
  BEGIN
   WHILE TRUE LOOP
    clk <= '0';
    WAIT FOR clk_period / 2;
    clk <= '1';
    WAIT FOR clk_period / 2;
   END LOOP;
  END PROCESS;
```

Αρχικά πραγματοποιείται η δήλωση και η σύνδεση της υπό δοκιμή οντότητας FIR2, η οποία προέρχεται από τη βιβλιοθήκη work. Η μονάδα αυτή ορίζεται με γενικές παραμέτρους (n, m) που πιθανότατα καθορίζουν το μήκος του φίλτρου ή το πλάτος των δεδομένων. Με αυτόν τον τρόπο, η UUT συνδέεται με τα αντίστοιχα σήματα του testbench, ώστε να μπορεί να ελεγχθεί η συμπεριφορά της κατά τη διάρκεια της προσομοίωσης.

Το δεύτερο τμήμα του κώδικα αφορά στη δημιουργία του σήματος ρολογιού (clk), το οποίο είναι απαραίτητο για τον συγχρονισμό της λειτουργίας του ψηφιακού συστήματος.

Η διαδικασία clk_process υλοποιεί έναν ατέρμονο βρόχο (WHILE TRUE LOOP) που εναλλάσσει την τιμή του clk μεταξύ '0' και '1' με χρονική καθυστέρηση ίση με το μισό της περιόδου (clk_period / 2). Έτσι παράγεται ένα τετραγωνικό περιοδικό σήμα που προσομοιώνει το πραγματικό ρολόι υλικού.

```
  -- Κύριο σενάριο δοκιμής
  stim_proc: PROCESS
  BEGIN
    -- Reset αρχικά
    rst <= '1';
    WAIT FOR 20 ns;
    rst <= '0';
    WAIT FOR 20 ns;

    -- Σειριακή φόρτωση συντελεστών: coef = (1, 2, 3, 4)
    load <= '1';
    coef_in <= "0001";  -- 1
    WAIT FOR 20 ns;
    coef_in <= "0010";  -- 2
    WAIT FOR 20 ns;
    coef_in <= "0011";  -- 3
    WAIT FOR 20 ns;
    coef_in <= "0100";  -- 4
    WAIT FOR 20 ns;
    load <= '0';

    -- Τώρα στέλνουμε δείγματα x = 1, 2, 3, 4, 0, 0...
    x <= "0001"; -- 1
    WAIT FOR 20 ns;
    x <= "0010"; -- 2
    WAIT FOR 20 ns;
    x <= "0011"; -- 3
    WAIT FOR 20 ns;
    x <= "0100"; -- 4
    WAIT FOR 20 ns;
    x <= "0000"; -- 0
    WAIT FOR 20 ns;
    x <= "0000"; -- 0
    WAIT FOR 20 ns;

    -- Τέλος προσομοίωσης
    WAIT;
  END PROCESS;
END behavior;
```

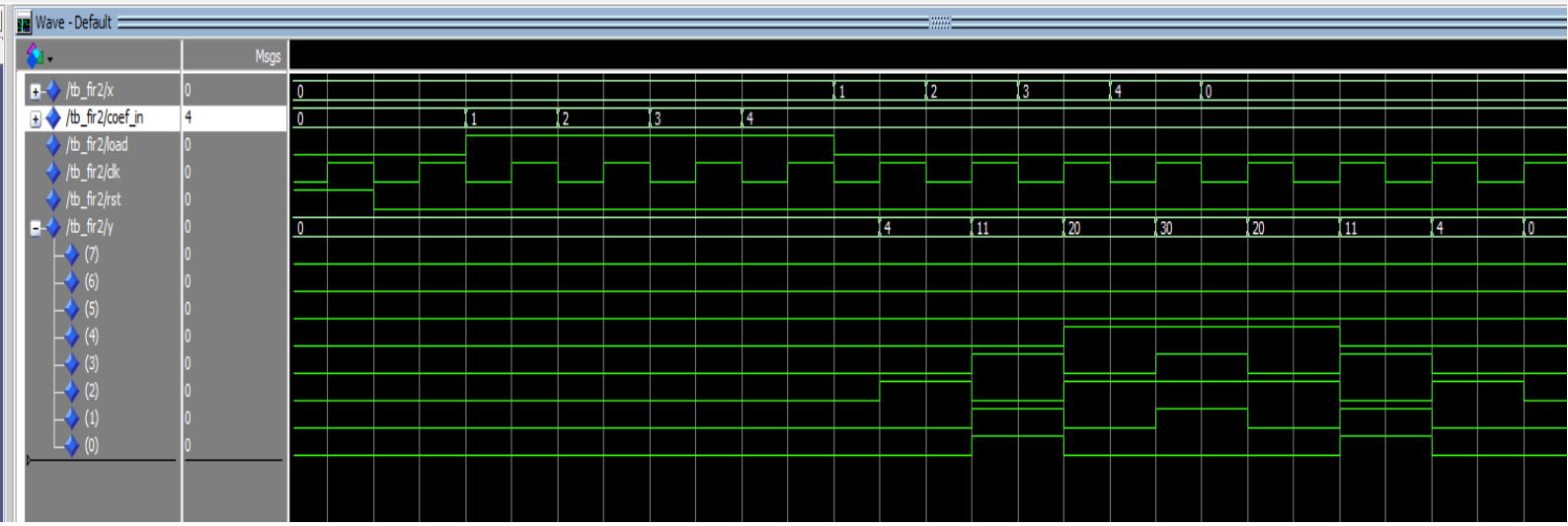
Τέλος, σε αυτό το κομμάτι κώδικα έχουμε τις εξής λειτουργίες. Αρχικά πραγματοποιείται η σύνδεση της υπό δοκιμή οντότητας FIR2, στην οποία αντιστοιχίζονται οι γενικές παράμετροι και χαρτογραφούνται οι θύρες εισόδου και εξόδου με τα αντίστοιχα σήματα του testbench.

Στη συνέχεια δημιουργείται το σήμα ρολογιού μέσω μιας διαδικασίας που εναλλάσσει περιοδικά την τιμή του clk με χρονική καθυστέρηση ίση με το μισό της περιόδου, ώστε να υπάρχει συγχρονισμός όπως σε ένα πραγματικό ψηφιακό κύκλωμα. Το κύριο σενάριο δοκιμής ξεκινά με την ενεργοποίηση του reset ώστε το κύκλωμα να αρχικοποιηθεί σε γνωστή κατάσταση. Έπειτα φορτώνονται σειριακά οι συντελεστές του φίλτρου μέσω του σήματος coef_in με ενεργό το σήμα load, και στη συνέχεια το load μηδενίζεται ώστε να δηλωθεί το τέλος της φόρτωσης.

Μετά την αποθήκευση των συντελεστών εφαρμόζονται στην είσοδο x διαδοχικά δείγματα τιμών (1, 2, 3, 4, 0, 0) σε συγκεκριμένα χρονικά διαστήματα, ώστε να παρατηρηθεί η απόκριση του φίλτρου στην πράξη.

Τέλος, η διαδικασία σταματά με την εντολή WAIT, σηματοδοτώντας το πέρας της προσομοίωσης. Συνολικά, ο κώδικας δημιουργεί το απαραίτητο περιβάλλον προσομοίωσης για το FIR φίλτρο, εξασφαλίζοντας την αρχικοποίηση του συστήματος, τη σωστή φόρτωση παραμέτρων και την εφαρμογή δειγμάτων εισόδου, με σκοπό την αξιολόγηση της λειτουργίας και της απόκρισης του κυκλώματος.

5.3[Ανάλυση Αποτελεσμάτων]



1. Φόρτωση συντελεστών
    - Στην αρχή, με ενεργό το σήμα load, οι συντελεστές (coef_in) εισάγονται σειριακά στο φίλτρο.
    - Σε κάθε κύκλο ρολογιού, η νέα τιμή που δίνεται στο coef_in αποθηκεύεται στη θέση coef(0), ενώ οι προηγούμενες τιμές μετακινούνται μία θέση προς τα δεξιά (shift).
    - Έτσι, μετά από 4 κύκλους, το φίλτρο έχει αποθηκεύσει τους συντελεστές:

    $coef$=(4, 3, 2, 1)
    - Αυτό σημαίνει ότι η συνάρτηση μεταφοράς του φίλτρου είναι:

    y[n]=1·x[n]+2·x[n−1]+3·x[n−2]+4·x[n−3]

2. Επεξεργασία εισόδων
    - Αφού φορτωθούν οι συντελεστές, το load γίνεται '0' και το φίλτρο περνάει στη κανονική λειτουργία.
    - Στην είσοδο x δίνονται διαδοχικά τα δείγματα:

    1,2,3,4,0,0
    - Σε κάθε θετική ακμή του ρολογιού, η νέα είσοδος εισάγεται στο shift register reg, το οποίο διατηρεί τα προηγούμενα δείγματα. Έτσι, το φίλτρο έχει πάντα πρόσβαση στο τρέχον και στα 3 προηγούμενα δείγματα.
3. Υπολογισμός της εξόδου

- Στο εσωτερικό του φίλτρου, κάθε συντελεστής πολλαπλασιάζεται με το αντίστοιχο δείγμα εισόδου (τρέχον ή προηγούμενο).
- Τα αποτελέσματα αυτών των γινομένων αθροίζονται στον συσσωρευτή acc.
- Έτσι, σε κάθε νέο δείγμα εισόδου, υπολογίζεται η έξοδος σύμφωνα με τον τύπο FIR.
- Αν υπάρξει υπερχείλιση (δηλαδή αν το αποτέλεσμα δεν χωράει στα διαθέσιμα bits), γίνεται έλεγχος και κορεσμός (saturation logic).

4. Παρατήρηση στα κύματα (waveform)

Στο διάγραμμα που φαίνεται:

- Η είσοδος x στέλνει τα δείγματα 1, 2, 3, 4, 0.
- Οι συντελεστές έχουν φορτωθεί ως 4, 3, 2, 1
- Η έξοδος y δείχνει την απόκριση του FIR φίλτρου για κάθε χρονική στιγμή:
- Για x=0:
  y0=4·0+3·0+2·0+1·0=0
- Για x=1(reg = [0,0,0]):
  y1=4·1+3·0+2·0+1·0=4
- Για x=2(reg=[1,0,0](από προηγ. κύκλο)):
  y2=4·2+3·1+2·0+1·0=8+3=11
- Για x=3(reg = [2,1,0]):
  y3=4·3+3·2+2·1+1·0=12+6+2=20
- Για x=4(reg = [3,2,1]):
  y4=4·4+3·3+2·2+1·1=16+9+4+1=30
- Για x=0(reg = [4,3,2]):
  y5=4·0+3·4+2·3+1·2=0+12+6+2=20
- Για x=0(reg = [0,4,3] (μετά το shift)):
  y6=4·0+3·0+2·4+1·3=0+0+8+3=11
- Για x=0(reg = [0,0,4]):
  y7=4·0+3·0+2·0+1·4=4
- Για x=0(reg = [0,0,0]):
  y8=0

Όπως φαίνεται στο waveform, οι έξοδοι που προκύπτουν είναι ακριβώς αυτές οι τιμές (0, 4, 11, 20, 30, 20, 11, 4, 0), κάτι που επιβεβαιώνει τη σωστή λειτουργία του φίλτρου.

5. Συμπέρασμα

Η μεθοδολογία παραγωγής των αποτελεσμάτων βασίζεται σε τρία στάδια: φόρτωση συντελεστών, αποθήκευση και μετακίνηση δειγμάτων εισόδου μέσω shift register και υπολογισμό της εξόδου με βάση τον πολλαπλασιασμό και άθροιση συντελεστών-δειγμάτων. Με αυτόν τον τρόπο, η προσομοίωση αποδεικνύει ότι το FIR φίλτρο λειτουργεί σύμφωνα με τη θεωρητική του εξίσωση, παράγοντας τα αναμενόμενα αποτελέσματα σε κάθε χρονική στιγμή.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα πτυχιακή εργασία είχε ως κύριο στόχο τη μελέτη, σχεδίαση και υλοποίηση ενός ψηφιακού φίλτρου FIR μέσω της χρήσης γλώσσας περιγραφής υλικού (VHDL) και της υλοποίησής του σε πλατφόρμα FPGA, με χρήση του λογισμικού Quartus Prime. Μέσα από τη διαδικασία αυτή, επιβεβαιώθηκε η ισχυρή συσχέτιση της θεωρητικής γνώσης με την πρακτική εφαρμογή στον τομέα της ψηφιακής επεξεργασίας σήματος.

Η εργασία ανέδειξε τη σημασία των φίλτρων FIR για εφαρμογές που απαιτούν γραμμική φάση, σταθερότητα και ακρίβεια. Ιδιαίτερη έμφαση δόθηκε στη δυνατότητα των ψηφιακών φίλτρων να προσαρμόζουν την έξοδο ενός σήματος, απορρίπτοντας ανεπιθύμητες συχνότητες και διατηρώντας τα χρήσιμα χαρακτηριστικά του αρχικού σήματος. Επιπλέον, παρουσιάστηκαν και συγκρίθηκαν οι διαφορετικοί τύποι φίλτρων, με σκοπό την καλύτερη κατανόηση των πλεονεκτημάτων και των περιορισμών της κάθε προσέγγισης.

Μέσα από την υλοποίηση του φίλτρου FIR με χρήση VHDL, κατέστη σαφές πως η χρήση περιγραφικών γλωσσών υλικού επιτρέπει τη σχεδίαση ψηφιακών κυκλωμάτων με υψηλό επίπεδο ακρίβειας, επεκτασιμότητας και ελέγχου. Η χρήση του FPGA ως πλατφόρμα υλοποίησης έδωσε τη δυνατότητα άμεσης προσομοίωσης και ελέγχου του φίλτρου, επιβεβαιώνοντας τη σωστή του λειτουργία, αλλά και την προσαρμοστικότητα της τεχνολογίας στις ανάγκες της εφαρμογής.

Η μεθοδολογία που ακολουθήθηκε αποδείχθηκε αποτελεσματική τόσο για την ανάπτυξη του συστήματος όσο και για τη δοκιμή και αξιολόγηση της λειτουργίας του. Η χρήση του testbench ανέδειξε την κρισιμότητα της εξομοίωσης κατά τη φάση σχεδίασης, εφόσον εξασφαλίζει την επαλήθευση της λογικής συμπεριφοράς του κυκλώματος πριν αυτό υλοποιηθεί στο υλικό.

Συμπερασματικά, η εργασία αποδεικνύει την αξία και τη δυναμική της τεχνολογίας FPGA και της γλώσσας VHDL στον τομέα της ψηφιακής επεξεργασίας σήματος. Μέσα από την εφαρμογή ενός φίλτρου FIR, αναδεικνύεται η δυνατότητα υλοποίησης πολύπλοκων υπολογιστικών μονάδων με ακρίβεια, ταχύτητα και αξιοπιστία. Η εμπειρία που αποκτήθηκε αποτελεί πολύτιμο εφόδιο για μελλοντικές ερευνητικές ή επαγγελματικές δραστηριότητες στους τομείς της μικροηλεκτρονικής, της σχεδίασης υλικού και των ενσωματωμένων συστημάτων.

ΠΑΡΑΡΤΗΜΑ

- **ΜΑΙΝ ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ**

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

USE ieee.std_logic_arith.ALL;



ENTITY FIR2 IS

 GENERIC (

   n : INTEGER := 4; -- taps

   m : INTEGER := 4  -- bit-width

 );

 PORT (

   x      : IN  SIGNED(m-1 DOWNTO 0);

   coef_in : IN  SIGNED(m-1 DOWNTO 0); -- σειριακή είσοδος coefficient

   load    : IN  STD_LOGIC;          -- φόρτωση coefficient

   clk     : IN  STD_LOGIC;

   rst     : IN  STD_LOGIC;

   y      : OUT SIGNED(2*m-1 DOWNTO 0)

 );

END FIR2;
```

```
ARCHITECTURE rtl OF FIR2 IS

  TYPE reg_array IS ARRAY (0 TO n-2) OF SIGNED(m-1 DOWNTO 0);

  TYPE coef_array IS ARRAY (0 TO n-1) OF SIGNED(m-1 DOWNTO 0);


  SIGNAL reg  : reg_array := (OTHERS => (OTHERS => '0'));

  SIGNAL coef : coef_array := (OTHERS => (OTHERS => '0'));

  SIGNAL coef_index : INTEGER RANGE 0 TO n-1 := 0;


BEGIN

  PROCESS(clk, rst)

    VARIABLE acc  : SIGNED(2*m-1 DOWNTO 0) := (OTHERS => '0');

    VARIABLE prod : SIGNED(2*m-1 DOWNTO 0);

    VARIABLE sign : STD_LOGIC;

  BEGIN

    IF rst = '1' THEN

      reg  <= (OTHERS => (OTHERS => '0'));

      coef <= (OTHERS => (OTHERS => '0'));

      coef_index <= 0;

      y <= (OTHERS => '0');


    ELSIF rising_edge(clk) THEN

      -- Σειριακή φόρτωση συντελεστών
```

```
                IF load = '1' THEN
                -- Shift προς τα δεξιά (από μικρό προς μεγαλύτερο index)
                        FOR i IN n-1 DOWNTO 1 LOOP
                                coef(i) <= coef(i-1);
                        END LOOP;
                        coef(0) <= coef_in;



-- Κανονική λειτουργία φίλτρου
ELSE
  acc := coef(0) * x;

  FOR i IN 1 TO n-1 LOOP
    prod := coef(i) * reg(i-1);
    sign := acc(2*m-1);
    acc := acc + prod;

    -- Έλεγχος υπερχείλισης
    IF (sign = prod(prod'LEFT)) AND (acc(acc'LEFT) /= sign) THEN
      acc := (acc'LEFT => sign, OTHERS => NOT sign);
    END IF;
  END LOOP;
```

```
        -- Shift register
                FOR i IN n-2 DOWNTO 0 LOOP
                        IF i = 0 THEN
                                reg(0) <= x;
                        ELSE
                                reg(i) <= reg(i-1);
                        END IF;
                END LOOP;


        y <= acc;
      END IF;
    END IF;
  END PROCESS;
END rtl;
```

- **ΚΩΔΙΚΑΣ TEST BENCH**

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

USE ieee.std_logic_arith.ALL;


ENTITY tb_FIR2 IS

END tb_FIR2;


ARCHITECTURE behavior OF tb_FIR2 IS

  CONSTANT n : INTEGER := 4;

  CONSTANT m : INTEGER := 4;


  SIGNAL x       : SIGNED(m-1 DOWNTO 0) := (OTHERS => '0');

  SIGNAL coef_in  : SIGNED(m-1 DOWNTO 0) := (OTHERS => '0');

  SIGNAL load     : STD_LOGIC := '0';

  SIGNAL clk      : STD_LOGIC := '0';

  SIGNAL rst      : STD_LOGIC := '1';

  SIGNAL y       : SIGNED(2*m-1 DOWNTO 0);

  CONSTANT clk_period : TIME := 20 ns;


  COMPONENT FIR2

    GENERIC (
```

```
    n : INTEGER := 4;
    m : INTEGER := 4
  );
  PORT (
    x      : IN  SIGNED(m-1 DOWNTO 0);
    coef_in : IN  SIGNED(m-1 DOWNTO 0);
    load    : IN  STD_LOGIC;
    clk     : IN  STD_LOGIC;
    rst     : IN  STD_LOGIC;
    y      : OUT SIGNED(2*m-1 DOWNTO 0)
  );
 END COMPONENT;

BEGIN
 -- Συνδέσεις
 DUT: FIR2
  GENERIC MAP (
    n => n,
    m => m
  )
  PORT MAP (
    x      => x,
```

```
   coef_in => coef_in,
   load    => load,
   clk     => clk,
   rst     => rst,
   y       => y
  );

-- Ρολογάκι
clk_process : PROCESS
BEGIN
        WHILE TRUE LOOP
  clk <= '0';
  WAIT FOR clk_period / 2;
  clk <= '1';
  WAIT FOR clk_period / 2;
        END LOOP;
END PROCESS;

-- Κύριο σενάριο δοκιμής
stim_proc: PROCESS
BEGIN
  -- Reset αρχικά
```

```
rst <= '1';
WAIT FOR 20 ns;
rst <= '0';
        WAIT FOR 20 ns;

-- Σειριακή φόρτωση συντελεστών: coef = (1, 2, 3, 4)
load <= '1';
coef_in <= "0001";  -- 1
WAIT FOR 20 ns;
coef_in <= "0010";  -- 2
WAIT FOR 20 ns;
coef_in <= "0011";  -- 3
WAIT FOR 20 ns;
coef_in <= "0100";  -- 4
WAIT FOR 20 ns;
load <= '0';

-- Τώρα στέλνουμε δείγματα x = 1, 2, 3, 4, 0, 0...
x <= "0001"; -- 1
WAIT FOR 20 ns;
x <= "0010"; -- 2
WAIT FOR 20 ns;
```

```
    x <= "0011"; -- 3
    WAIT FOR 20 ns;
    x <= "0100"; -- 4
    WAIT FOR 20 ns;
    x <= "0000"; -- 0
    WAIT FOR 20 ns;
    x <= "0000"; -- 0
    WAIT FOR 20 ns;


    -- Τέλος προσομοίωσης
    WAIT;
  END PROCESS;
END behavior;
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Intel.(2023).Quartus Prime Software Brochure. https://www.intel.com/content/dam/www/central-libraries/us/en/documents/quartus-prime-software-brochure.pdf

[2] HardwareBee. (2023). Brief History of Altera. https://hardwarebee.com/brief-history-altera/

[3] Wikipedia. (n.d.). Altera. https://en.wikipedia.org/wiki/Altera

[4] RayPCB. (n.d.). Altera Quartus. https://www.raypcb.com/altera-quartus/

[5] Xilinx. (n.d.). What is an FPGA. https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[6] Wikipedia. (n.d.). FPGA. https://en.wikipedia.org/wiki/Field-programmable_gate_array

[7] Wikipedia. (n.d.). VHDL. https://el.wikipedia.org/wiki/VHDL

[8] IHU. (n.d.). Εισαγωγή στη VHDL. http://teachers.cm.ihu.gr/kalomiros/Proigmena_Psifiaka_Theoria/didaktiko_yliko/Intro_VHDL_Kalomiros_teuxos2.pdf

[9] Wikipedia. (n.d.). Hardware description language. https://en.wikipedia.org/wiki/Hardware_description_language

[10] Wikipedia. (n.d.). Γλώσσα περιγραφής υλικού. https://el.wikipedia.org/wiki/Γλώσσα_περιγραφής_υλικού

[11] Porlidas. (n.d.). Εισαγωγή στα FPGA. https://www.porlidas.gr/MSC/FPGA.pdf

[12] CSD UOC. (n.d.). FPGA Lecture Slides. https://www.csd.uoc.gr/~hy370/w18/Slides/Lec26.pdf

[13] TheAstrologyPage. (n.d.). Τι είναι τα ψηφιακά φίλτρα. https://el.theastrologypage.com/digital-filter

[14] Embedded.com. (n.d.). Introduction to Digital Filters. https://www.embedded.com/introduction-to-digital-filters/

[15] Dewetron. (2023). FIR Filter Whitepaper. https://www.dewetron.com/2023/07/fir-filter-whitepaper/

[16] TEI West. (n.d.). DSP Παρασκευάς. http://opencourses.teiwest.gr/modules/document/file.php/CIED105/DSP_Paraskevas_Lecture_12.pdf

[17] Σχεδίαση κυκλωμάτων με την VHDL Volnei A. Pedroni

[18] https://en.wikipedia.org/wiki/Digital_filter

[19] https://www.allaboutcircuits.com/textbook/alternating-current/chpt-8/what-is-a-filter/

[20]https://chatgpt.com

[Οπισθόφυλλο. Κενή σελίδα]