

# Ψηφιακά Συστήματα ΗΥ σε Χαμηλά Επίπεδα Λογικής Ι



**Ονοματεπώνυμο:** Πελοπίδας-Νικόλαος Τσιούντσιουρας  
**ΑΕΜ:** 11085

**Υπεύθυνος Καθηγητής**  
Βασίλειος Παυλίδης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Ακαδημαϊκό Έτος 2025–2026

## **Περιεχόμενα**

<b>1</b>	<b>Εισαγωγή</b>	<b>2</b>
<b>2</b>	<b>Άσκηση 1: Υλοποίηση Arithmetic Logic Unit (ALU)</b>	<b>2</b>
<b>3</b>	<b>Άσκηση 2: Υλοποίηση Αριθμομηχανής</b>	<b>3</b>
<b>4</b>	<b>Άσκηση 3: Υλοποίηση Register File</b>	<b>3</b>
<b>5</b>	<b>Άσκηση 4: Υλοποίηση Νευρωνικού Συστήματος</b>	<b>4</b>

# 1 Εισαγωγή

Σκοπός της παρούσας αναφοράς είναι η παρουσίαση της σχεδίασης, υλοποίησης και επαλήθευσης των εργαστηριακών ασκήσεων του μαθήματος *Ψηφιακά Συστήματα HW σε Χαμηλά Επίπεδα Λογικής I*.

Η εργασία περιλαμβάνει την υλοποίηση μίας αριθμητικής/λογικής μονάδας (ALU), μίας αριθμομηχανής, ενός αρχείου καταχωρητών και ενός απλού νευρωνικού συστήματος βασισμένου σε πεπερασμένη μηχανή καταστάσεων (FSM).

Όλες οι υλοποιήσεις πραγματοποιήθηκαν σε Verilog και επαληθεύτηκαν μέσω προσομοιώσεων με χρήση κατάλληλων testbenches.

## 2 Άσκηση 1: Υλοποίηση Arithmetic Logic Unit (ALU)

Στην πρώτη άσκηση υλοποιήθηκε μία 32-bit αριθμητική/λογική μονάδα (ALU), η οποία υποστηρίζει αριθμητικές και λογικές πράξεις και πράξεις ολίσθησης, σύμφωνα με τις απαιτήσεις της εκφώνησης. Η ALU υλοποιήθηκε ως καθαρά συνδυαστικό κύκλωμα, χωρίς χρήση ρολογιού ή καταχωρητών, γεγονός που εξασφαλίζει ότι η έξοδος εξαρτάται αποκλειστικά από τις τρέχουσες τιμές των εισόδων.

Η επιλογή της εκάστοτε λειτουργίας πραγματοποιείται μέσω του σήματος ελέγχου `alu_op`, πλάτους 4 bit. Το σήμα αυτό οδηγεί έναν πολυπλέκτη επιλογής λειτουργίας, ο οποίος καθορίζει ποια πράξη θα εκτελεστεί μεταξύ των τελεστών `op1` και `op2`. Οι αριθμητικές πράξεις υλοποιούνται με προσημασμένη αριθμητική, ενώ για τις λογικές πράξεις χρησιμοποιούνται οι αντίστοιχοι bitwise τελεστές της Verilog.

Για λόγους αναγνωσιμότητας, επεκτασιμότητας και ευκολίας συντήρησης του κώδικα, οι κωδικοποιήσεις των λειτουργιών της ALU ορίστηκαν με χρήση παραμέτρων (`parameter`). Κάθε λειτουργία αντιστοιχίζεται σε μία μοναδική 4-bit τιμή του σήματος `alu_op`, επιτρέποντας την εύκολη αναγνώριση και διαχείριση των υποστηριζόμενων πράξεων. Οι παράμετροι αυτές περιλαμβάνουν λογικές πράξεις (AND, OR, XOR, NAND, NOR), αριθμητικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμός), καθώς και λογικές και αριθμητικές ολισθήσεις προς τα δεξιά και αριστερά.

Listing 1: Ορισμός παραμέτρων λειτουργιών ALU

1	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_AND</code>	<code>= 4'b1000;</code>
2	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_OR</code>	<code>= 4'b1001;</code>
3	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_NOR</code>	<code>= 4'b1010;</code>
4	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_NAND</code>	<code>= 4'b1011;</code>
5	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_XOR</code>	<code>= 4'b1100;</code>
6	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_ADD</code>	<code>= 4'b0100;</code>
7	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_SUB</code>	<code>= 4'b0101;</code>
8	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_MUL</code>	<code>= 4'b0110;</code>
9	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_LOG_SHIFT_R</code>	<code>= 4'b0000;</code>
10	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_LOG_SHIFT_L</code>	<code>= 4'b0001;</code>
11	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_ARI_SHIFT_R</code>	<code>= 4'b0010;</code>
12	<code>parameter</code>	<code>[3:0]</code>	<code>ALU_ARI_SHIFT_L</code>	<code>= 4'b0011;</code>

Το σήμα `zero` ενεργοποιείται όταν το αποτέλεσμα της ALU είναι ίσο με μηδέν, ενώ το σήμα `onf` χρησιμοποιείται για την ανίχνευση υπερχείλισης στις αριθμητικές πράξεις πρόσθεσης, αφαίρεσης και πολλαπλασιασμού. Με τον τρόπο αυτό παρέχεται επιπλέον πληροφορία κατάστασης, η οποία αξιοποιείται σε μεταγενέστερα στάδια του συστήματος.

### 3 Άσκηση 2: Υλοποίηση Αριθμομηχανής

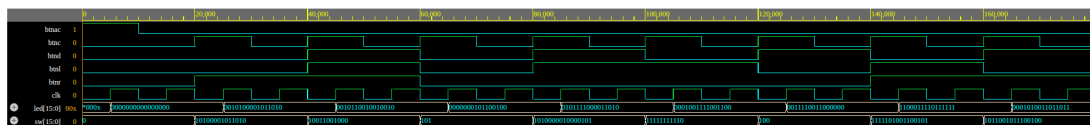
Στη δεύτερη άσκηση υλοποιήθηκε μία απλή αριθμομηχανή, η οποία βασίζεται στη μονάδα ALU που σχεδιάστηκε στην Άσκηση 1. Το κύκλωμα διατηρεί την τρέχουσα τιμή της αριθμομηχανής σε έναν συσσωρευτή (accumulator) πλάτους 16 bit και επιτρέπει την εκτέλεση αριθμητικών και λογικών πράξεων μέσω της ALU, σύμφωνα με τα σήματα ελέγχου που προκύπτουν από τα πλήκτρα εισόδου.

Ο συσσωρευτής υλοποιήθηκε ως σύγχρονος καταχωρητής, ο οποίος ενημερώνεται στην ανερχόμενη ακμή του ρολογιού όταν ενεργοποιείται το κεντρικό πλήκτρο (`btnc`), ενώ μηδενίζεται σύγχρονα μέσω του πλήκτρου εκκαθάρισης (`btncac`). Η τιμή του συσσωρευτή συνδέεται απευθείας στις εξόδους LED, παρέχοντας άμεση οπτική ένδειξη της τρέχουσας κατάστασης της αριθμομηχανής.

Για τη σύνδεση του συσσωρευτή με την ALU χρησιμοποιήθηκε επέκταση προσήμου (sign extension), όπως ορίζεται στην εκφώνηση. Συγκεκριμένα, τόσο η τιμή του συσσωρευτή όσο και οι τιμές των διακοπών εισόδου (`sw`) επεκτείνονται από 16 bit σε 32 bit και οδηγούν τις εισόδους `op1` και `op2` της ALU αντίστοιχα. Με τον τρόπο αυτό εξασφαλίζεται η σωστή εκτέλεση προσημασμένων πράξεων, ενώ το αποτέλεσμα της ALU αποθηκεύεται εκ νέου στον συσσωρευτή μέσω των 16 λιγότερο σημαντικών bit της εξόδου.

Η παραγωγή του σήματος ελέγχου `alu_op` υλοποιήθηκε σε ξεχωριστό module (`calc_enc`), με χρήση δομικής Verilog. Το κύκλωμα αυτό παράγει τα τέσσερα bit του `alu_op` αποκλειστικά μέσω συνδυαστικής λογικής και βασίζεται στις καταστάσεις των πλήκτρων `btnl`, `btnr` και `btnd`, ακολουθώντας πιστά τα λογικά διαγράμματα που δίνονται στην εκφώνηση. Ο διαχωρισμός αυτός βελτιώνει τη δομημένη σχεδίαση και διευκολύνει την επαναχρησιμοποίηση και τον έλεγχο του κυκλώματος.

Η λειτουργία της αριθμομηχανής επαληθεύτηκε μέσω κατάλληλου testbench, στο οποίο εφαρμόστηκαν διαδοχικές πράξεις σύμφωνα με τα σενάρια ελέγχου της εκφώνησης. Στην κυματομορφή εξόδου παρατηρείται η σωστή ενημέρωση του συσσωρευτή μετά από κάθε ενεργοποίηση του `btnc`, καθώς και η ορθή εκτέλεση των αντίστοιχων λειτουργιών της ALU. Ειδικότερα, επιβεβαιώνεται ότι η επαναφορά μέσω του `btncac` μηδενίζει τον συσσωρευτή, ενώ οι αριθμητικές και λογικές πράξεις παράγουν τα αναμενόμενα αποτελέσματα, τα οποία εμφανίζονται άμεσα στα LED.



Σχήμα 1: Κυματομορφές προσομοίωσης αριθμομηχανής

### 4 Άσκηση 3: Υλοποίηση Register File

Στην τρίτη άσκηση υλοποιήθηκε ένα αρχείο καταχωρητών (register file) 16×32-bit, το οποίο διαθέτει τέσσερις θύρες ανάγνωσης και δύο θύρες εγγραφής, σύμφωνα με τις απαιτήσεις της εκφώνησης. Το register file σχεδιάστηκε ώστε να χρησιμοποιηθεί στα επόμενα στάδια του συστήματος, και συγκεκριμένα για την αποθήκευση βαρών και πολώσεων του νευρωνικού δικτύου της Άσκησης 4.

Η λειτουργία του κυκλώματος διαχωρίστηκε σε δύο ανεξάρτητα `always` blocks με διακριτό ρόλο. Το πρώτο είναι συγχρονισμένο με το ρολόι και υλοποιεί τόσο την εγγραφή δεδομένων όσο και την ασύγχρονη επαναφορά ενεργή σε χαμηλό επίπεδο (`resetn`). Κατά την ενεργοποίηση

του `resetn`, όλοι οι καταχωρητές αρχικοποιούνται στην τιμή μηδέν, εξασφαλίζοντας γνωστή αρχική κατάσταση του συστήματος. Όταν το σήμα `write` είναι ενεργό, πραγματοποιείται εγγραφή δεδομένων κατά την ανερχόμενη ακμή του ρολογιού, με υποστήριξη δύο ταυτόχρονων εγγραφών μέσω των σημάτων `writeReg1/writeData1` και `writeReg2/writeData2`.

Το δεύτερο `always` block είναι καθαρά συνδυαστικό και υλοποιεί τη λειτουργία ανάγνωσης. Οι τιμές των καταχωρητών που υποδεικνύονται από τα σήματα `readReg1` έως `readReg4` ανατίθενται άμεσα στις αντίστοιχες εξόδους `readData1` έως `readData4`, χωρίς τη χρήση ρολογιού. Με τον τρόπο αυτό εξασφαλίζεται άμεση πρόσβαση στα αποθηκευμένα δεδομένα, κάτι ιδιαίτερα σημαντικό για την παράλληλη λειτουργία των επόμενων σταδίων του νευρωνικού συστήματος.

Ιδιαίτερη μέριμνα δόθηκε στη διαχείριση περιπτώσεων ταυτόχρονης εγγραφής και ανάγνωσης του ίδιου καταχωρητή. Στις περιπτώσεις αυτές υλοποιήθηκε μηχανισμός `write forwarding`, ο οποίος δίνει προτεραιότητα στα δεδομένα εγγραφής έναντι της παλαιότερης αποθηκευμένης τιμής. Έτσι διασφαλίζεται ότι η ανάγνωση επιστρέφει πάντα τη πιο πρόσφατη και έγκυρη τιμή, ακόμη και όταν η εγγραφή και η ανάγνωση αφορούν τον ίδιο κύκλο λειτουργίας.

Η συγκεκριμένη υλοποίηση καθιστά το register file αξιόπιστο και κατάλληλο για χρήση ως κεντρικό στοιχείο αποθήκευσης στο νευρωνικό σύστημα της Άσκησης 4, υποστηρίζοντας πολλαπλές ταυτόχρονες προσβάσεις και αποφεύγοντας ασυνέπειες στα δεδομένα.

## 5 Άσκηση 4: Υλοποίηση Νευρωνικού Συστήματος

Η τέταρτη άσκηση αφορά την υλοποίηση ενός απλού νευρωνικού συστήματος, το οποίο ελέγχεται μέσω πεπερασμένης μηχανής καταστάσεων (FSM) τύπου Moore. Η επιλογή FSM τύπου Moore κρίθηκε κατάλληλη, καθώς οι έξοδοι του συστήματος εξαρτώνται αποκλειστικά από την τρέχουσα κατάσταση και όχι άμεσα από τις εισόδους, εξασφαλίζοντας σταθερή και προβλέψιμη συμπεριφορά κατά τη μετάβαση μεταξύ των διαδοχικών σταδίων υπολογισμού.

Το νευρωνικό σύστημα αξιοποιεί τη μονάδα ALU της Άσκησης 1 και το αρχείο καταχωρητών της Άσκησης 3 για την εκτέλεση των αριθμητικών πράξεων, μετατοπίσεων και την αποθήκευση των βαρών και των πολώσεων. Η συνολική λειτουργία οργανώνεται σε διακριτά στάδια, σύμφωνα με τη δομή που ορίζεται στην εκφώνηση, και περιλαμβάνει στάδια φόρτωσης βαρών, προεπεξεργασίας εισόδων, υπολογισμού των νευρώνων εισόδου, υπολογισμού του νευρώνα εξόδου και τελικής μεταεπεξεργασίας.

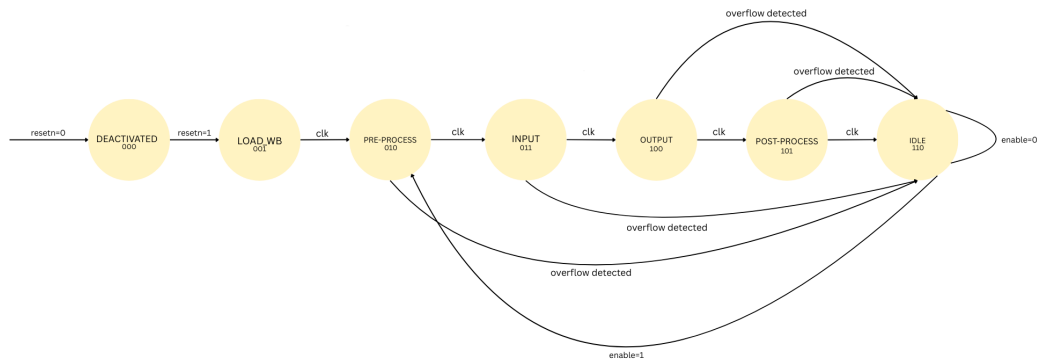
Η FSM περιλαμβάνει επτά καταστάσεις και αρχικοποιείται στην κατάσταση `DEACTIVATED`. Το σήμα επαναφοράς `resetn` θεωρείται ενεργό σε χαμηλό επίπεδο (`active-low`), και συνεπώς η εκκίνηση του νευρωνικού συστήματος πραγματοποιείται μόνο όταν το `resetn` μεταβεί σε υψηλή στάθμη και το σήμα `enable` ενεργοποιηθεί. Η επιλογή αυτή διασφαλίζει γνωστή αρχική κατάσταση και ορθή συγχρονισμένη εκκίνηση του συστήματος.

Κάθε νευρώνας υλοποιεί μία πράξη MAC (`Multiply and Accumulate`) μέσω ξεχωριστού module, το οποίο αποτελείται από δύο μονάδες ALU συνδεδεμένες σειριακά: η πρώτη εκτελεί τον πολλαπλασιασμό και η δεύτερη την πρόσθεση με την αντίστοιχη πόλωση. Με τον τρόπο αυτό επιτυγχάνεται καθαρός διαχωρισμός λειτουργιών και πιστή αντιστοίχιση με το θεωρητικό μοντέλο MAC. Παράλληλα, παρέχονται ξεχωριστά σήματα ένδειξης μηδενικού αποτελέσματος και υπερχειλίσας για κάθε στάδιο υπολογισμού.

Ιδιαίτερη έμφαση δόθηκε στη διαχείριση υπερχειλίσας, σύμφωνα με τις διευκρινίσεις της άσκησης και το μοντέλο αναφοράς. Σε περίπτωση που ανιχνευθεί `overflow` σε οποιοδήποτε στάδιο υπολογισμού, το FSM μεταβαίνει άμεσα στην κατάσταση `IDLE`, παρακάμπτοντας τα επόμενα στάδια, και η έξοδος του συστήματος οδηγείται στην προσημασμένη 32-bit τιμή που

αντιστοιχεί στο  $-1$ . Η προσέγγιση αυτή εξασφαλίζει συμφωνία με το μοντέλο αναφοράς και αποτρέπει την παραγωγή μη έγκυρων αποτελεσμάτων.

Το διάγραμμα της FSM παρουσιάζεται στο Σχήμα 2, όπου απεικονίζονται οι μεταβάσεις μεταξύ των καταστάσεων του συστήματος και η αντιστοίχισή τους στα επιμέρους στάδια επεξεργασίας.



Σχήμα 2: Διάγραμμα FSM του νευρωνικού συστήματος

Η λειτουργία του νευρωνικού συστήματος επαληθεύτηκε μέσω εκτενούς testbench, το οποίο συγκρίνει την έξοδο του κυκλώματος με την έξοδο αναφοράς του μοντέλου. Στην ενιαία κυματομορφή προσομοίωσης παρατηρείται η σωστή ακολουθία των καταστάσεων της FSM, η ορθή φόρτωση των βαρών και πολώσεων στο register file, καθώς και η σταδιακή παραγωγή των ενδιάμεσων και τελικών αποτελεσμάτων.

Επιπλέον, επιβεβαιώνεται ότι σε περιπτώσεις υπερχείλισης το σύστημα μεταβαίνει άμεσα στην κατάσταση IDLE και η έξοδος λαμβάνει την προβλεπόμενη τιμή  $-1$ , γεγονός που αποδεικνύει τη σωστή ενσωμάτωση της λογικής ελέγχου σφαλμάτων και τη συμφωνία της υλοποίησης με το μοντέλο αναφοράς.

