Ψηφιακά Συστήματα HW-1

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΑΡΙΣΤΟΤΕΛΗΣ ΤΣΕΚΟΥΡΑΣ – ΔΙΠΛ. ΗΛ. ΜΗΧ. ΚΑΙ ΜΗΧ. ΥΠΟΛ.
ΕΥΑΓΓΕΛΟΣ ΤΖΟΥΒΑΡΑΣ - ΔΙΠΛ. ΗΛ. ΜΗΧ. ΚΑΙ ΜΗΧ. ΥΠΟΛ.
ΒΑΣΙΛΗΣ ΠΑΥΛΙΔΗΣ – ΑΝ. ΚΑΘΗΓΗΤΗΣ, ΤΗΜΜΥ
ΑΠΘ | ΕΡΓΑΣΤΉΡΙΟ ΗΛΕΚΤΡΟΝΙΚΗΣ

Περιεχόμενα

Άσκηση 1

Μια αριθμητική/λογική μονάδα (Arithmetic Logic Unit (ALU)) είναι υπεύθυνη για την εκτέλεση αριθμητικών λειτουργιών, όπως πρόσθεση και αφαίρεση, καθώς και λογικών λειτουργιών, όπως AND/OR. Σε αυτή την άσκηση θα δημιουργήσετε μια ALU την οποία θα χρησιμοποιήσετε σε ένα μικρό σύστημα νευρωνικού (AI accelerator) σε μεταγενέστερη άσκηση. Θα χρησιμοποιήσετε επίσης την ALU για να υλοποιήσετε μια απλή "αριθμομηχανή".

Η ALU σας θα σχεδιαστεί για να υλοποιεί τις ακόλουθες πράξεις: προσημασμένη πρόσθεση, προσημασμένη αφαίρεση, προσημασμένος πολλαπλασιασμός, λογικό AND, λογικό OR, λογικό XOR, λογικό NAND, λογικό NOR και τέσσερις διαφορετικές πράξεις ολίσθησης.

Ξεκινήστε την άσκηση δημιουργώντας ένα νέο αρχείο Verilog με όνομα **alu.v**. Ορίστε ένα νέο module με όνομα “alu” και προσθέστε τις ακόλουθες θύρες (τα ονόματα των θυρών ΠΡΕΠΕΙ να είναι ακριβή όπως ορίζονται στον πίνακα):

Όνομα θύρας	**Κατεύθυνση**	**Πλάτος [αρ. bit]**	**Σκοπός**
op1	Είσοδος	32	Τελεστής 1 σε συμπλήρωμα ως προς 2
op2	Είσοδος	32	Τελεστής 2 σε συμπλήρωμα ως προς 2
alu_op	Είσοδος	4	Δείχνει ποια λειτουργία πρέπει να εκτελεστεί
zero	Έξοδος	1	Δείχνει πότε το αποτέλεσμα της ALU είναι μηδέν
result	Έξοδος	32	Αποτέλεσμα
ovf	Έξοδος	1	Δείχνει αν το αποτέλεσμα μιας πράξης **πρόσθεσης**, **αφαίρεσης** ή **πολλαπλασιασμού** οδηγεί σε υπερχείλιση (overflow)

Η ALU που πρόκειται να δημιουργήσετε είναι μια "32-bit" ALU που σημαίνει ότι οι είσοδοι έχουν πλάτος 32-bit, και παράγει μια έξοδο πλάτους 32-bit. Κατά συνέπεια πράξεις με μεγάλους αριθμούς μπορούν να οδηγήσουν σε υπερχείλιση. Η ALU θα εκτελέσει μία από τις διαφορετικές πράξεις μεταξύ των τελεστών 'op1' και 'op2' με βάση την τιμή του σήματος εισόδου 'alu_op'.

Αυτό το κύκλωμα είναι ένα "συνδυαστικό" κύκλωμα που σημαίνει ότι το αποτέλεσμα της εξόδου του κυκλώματος εξαρτάται μόνο από τις εισόδους ('op1', 'op2' και 'alu_op') και δεν υπάρχει μνήμη για την αποθήκευση της προηγούμενης κατάστασης (δεν υπάρχουν καταχωρητές και συνεπώς δεν υπάρχει ανάγκη για είσοδο σήματος ρολογιού ή επαναφοράς). Η λειτουργία που εκτελεί η ALU βασίζεται στο σήμα εισόδου 'alu_op' τεσσάρων bit, όπως ορίζεται στον ακόλουθο πίνακα:

alu_op	**Πράξη**
1000	Λογική AND
1001	Λογική OR
1010	Λογική NOR

1011	Λογική NAND
1100	Λογική XOR
0100	Προσημασμένη Πρόσθεση
0101	Προσημασμένη Αφαίρεση
0110	Προσημασμένος Πολλαπλασιασμός
0000	Λογική ολίσθηση δεξιά κατά op2 bits
0001	Λογική ολίσθηση αριστερά κατά op2 bits
0010	Αριθμητική ολίσθηση δεξιά κατά op2 bits
0011	Αριθμητική ολίσθηση αριστερά κατά op2 bits

Πολυπλέκτης ALU

Το κύκλωμα ALU πρέπει να περιλαμβάνει έναν πολυπλέκτη για να επιλέγει ποια από τις λειτουργίες θα εκτελέσει. Μπορείτε να καθορίσετε τις σταθερές χρησιμοποιώντας την οδηγία parameter στη Verilog ως εξής:

```
parameter[3:0] ALUOP_SUB = 4'b0110;
```

Θα πρέπει να δημιουργήσετε μια σταθερά για κάθε μία από αυτές τις δώδεκα εντολές λειτουργίας της ALU. Προσθέστε όλες τις παραμέτρους μέσα στο αρχείο **alu.v**.

Άσκηση 2

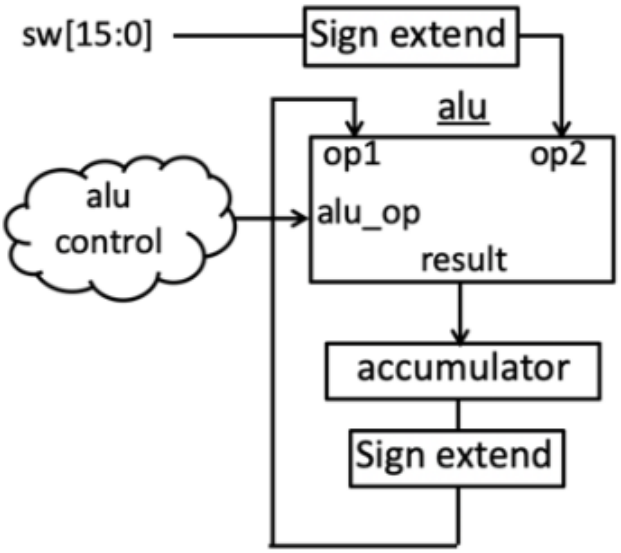
Για την άσκηση αυτή, θα σχεδιάσετε ένα κύκλωμα αριθμομηχανής που χρησιμοποιεί την ALU που δημιουργήσατε στην προηγούμενη άσκηση. Αυτό το κύκλωμα θα διατηρεί μια τρέχουσα τιμή της σε ένα συσσωρευτή 16-bit καταχωρητή και θα επιτρέπει στο χρήστη να ενημερώνει την τιμή υλοποιώντας οποιαδήποτε από τις αριθμητικές και λογικές συναρτήσεις που παρέχει η ALU σας.

Ξεκινήστε δημιουργώντας ένα νέο αρχείο Verilog με όνομα **calc.v**. Ορίστε μια νέα μονάδα με όνομα calc και προσθέστε τις ακόλουθες θύρες (τα ονόματα των θυρών ΠΡΕΠΕΙ να είναι ακριβή για να ταιριάζουν με το testbench):

Όνομα θύρας	**Κατεύθυνση**	**Πλάτος [αρ. bit]**	**Σκοπός**
clk	Είσοδος	1	Ρολόι
btnc	Είσοδος	1	Κεντρικό πλήκτρο
btnac	Είσοδος	1	Πλήκτρο εκκαθάρισης (all clear)
btnl	Είσοδος	1	Αριστερό πλήκτρο
btnr	Είσοδος	1	Δεξί πλήκτρο
btnd	Είσοδος	1	Κάτω πλήκτρο

sw	Είσοδος	16	Διακόπτες για την εισαγωγή δεδομένων
led	Έξοδος	16	LED για την έξοδο του συσσωρευτή

Τα δύο βασικά στοιχεία του κυκλώματος είναι ένας συσσωρευτής 16-bit για να κρατά την τρέχουσα τιμή της αριθμομηχανής σας και η ALU που δημιουργήσατε στην προηγούμενη άσκηση. Η σχέση της ALU και του συσσωρευτή πρέπει να σχεδιαστεί όπως φαίνεται στο ακόλουθο σχήμα.



Σχ. 1: Διάγραμμα ροής της αριθμομηχανής.

Δημιουργήστε έναν καταχωρητή 16 bit (που ονομάζεται accumulator) για να κρατάει τα περιεχόμενα της τρέχουσας τιμής της αριθμομηχανής. Σχεδιάστε τον καταχωρητή σας ως εξής:

- Ο accumulator θα πρέπει να συνδεθεί με την είσοδο του ρολογιού.
- Ο accumulator θα πρέπει να μηδενίζεται σύγχρονα με το πάτημα του **btnac.**
- Η είσοδος του accumulator θα πρέπει να είναι τα 16 χαμηλότερα bit της εξόδου αποτελέσματος 32 bit της ALU.
- Ο accumulator θα πρέπει να ενημερώνεται σύγχρονα κάθε φορά που πατιέται το "central" (**btnc**).
- Συνδέστε την τιμή του accumulator με τις εξόδους LED.

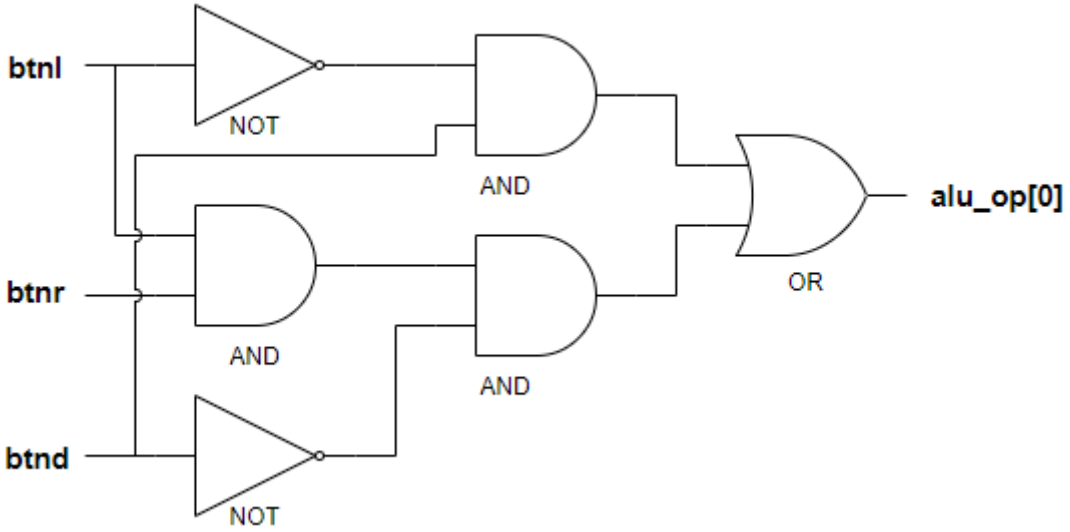
Συνδέστε τις εισόδους στην ALU ως εξής:

- Δημιουργήστε ένα σήμα 32-bit που είναι μια έκδοση με επέκταση προσήμου του accumulator 16-bit. Συνδέστε αυτό το σήμα στην είσοδο '**op1**' της ALU.
- Δημιουργήστε ένα σήμα 32-bit που είναι μια έκδοση με επέκταση προσήμου των εισόδων του διακόπτη 16-bit. Συνδέστε αυτό το σήμα στην είσοδο '**op2**' της ALU.
- Συνδέστε τα χαμηλότερα 16 bit της εξόδου "**result**" στην είσοδο του accumulator (όπως αναφέρθηκε παραπάνω).

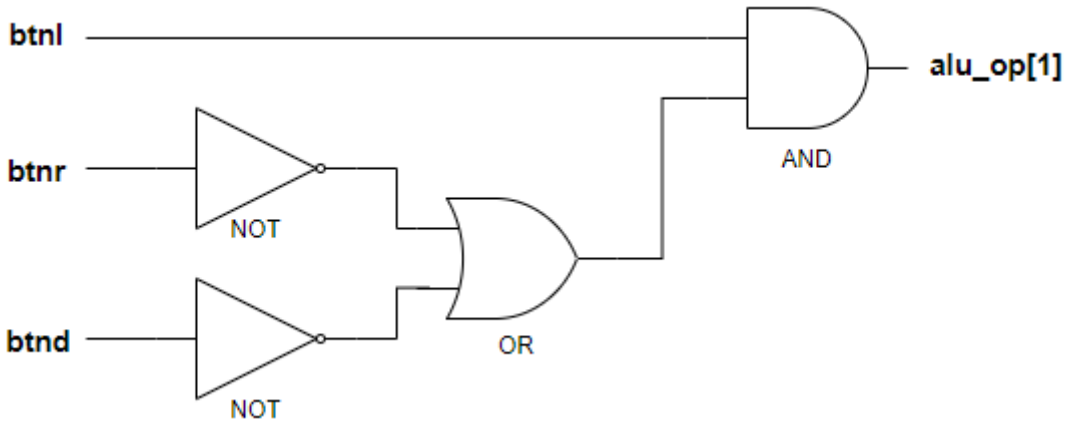
- Δημιουργήστε ένα νέο σήμα για το '**alu_op**' καθώς και τη λογική για αυτό το σήμα όπως αναλύεται παρακάτω (Σχήματα 2 – 5).

Μπορείτε να δημιουργήσετε ένα σήμα 32-bit με επέκταση προσήμου χρησιμοποιώντας τον τελεστή concatenation της Verilog (δηλ., επαναλάβετε το κορυφαίο bit του σήματος που επεκτείνετε με πρόσημο όσες φορές χρειάζεται).

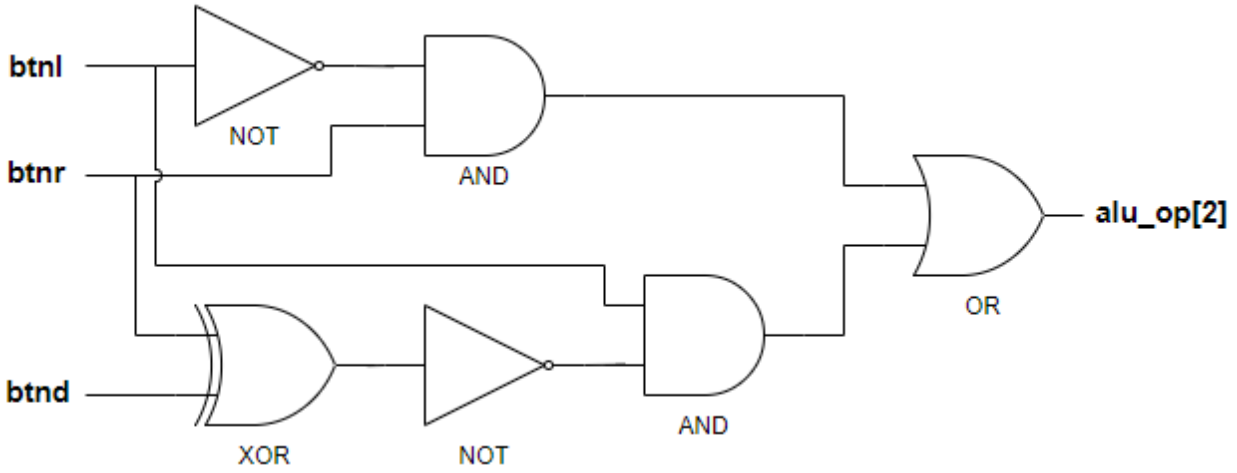
Το σήμα '**alu_op**' καθορίζει ποια πράξη της ALU θα εκτελεστεί. Θα καθορίσετε ποια λειτουργία θα εκτελέσετε με βάση την τιμή των τεσσάρων πλήκτρων: **btnl, btnr,** και **btnd**. Θα πρέπει να δημιουργήσετε το συνδυαστικό κύκλωμα των Σχημάτων 2 – 5 που να παράγει το κατάλληλο σήμα '**alu_op**' με βάση την τιμή αυτών των τριών πλήκτρων. Υλοποιήστε το κύκλωμα **calc_enc.v** σε **structural verilog** (δομική verilog) και συνδέστε το στο κύκλωμα αριθμομηχανής **calc.v**.



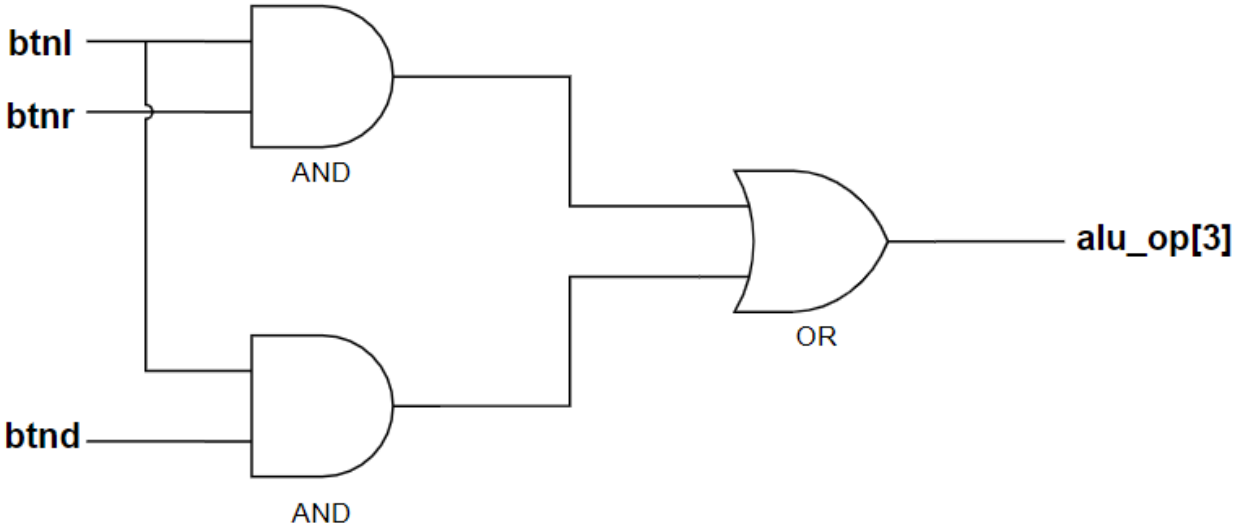
Σχ. 2: Παραγωγή του alu_op[0] μέσω των btnr, btnl, btnd.



Σχ. 3: Παραγωγή του alu_op[1] μέσω των btnr, btnl, btnd.



Σχ. 4: Παραγωγή του alu_op[2] μέσω των btnr, btnl, btnd.



Σχ. 5: Παραγωγή του alu_op[3] μέσω των btnr, btnl, btnd.

Testbench

Δημιουργήστε ένα testbench το οποίο θα ελέγχει την ορθή λειτουργία της αριθμομηχανής, καθώς και την ορθή λειτουργία της ALU. Ελέγξτε την αριθμομηχανή με τη σειρά για τις τιμές:

btnl, btnr, btnd (input)	**Previous value (acc.)**	**Switches (input)**	**Function in ALU**	**Expected Result**
(btnac for reset)	xxxx	xxxx	Reset	0x0
0,1,0	0x0	0x285a	ADD	0x285a
1,1,1	0x285a	0x04c8	XOR	0x2c92
0,0,0	0x2c92	0x0005	Logical Shift Right	0x0164

1,0,1	0x0164	0xa085	NOR	0x5e1a
1,0,0	0x5e1a	0x07fe	MULT	0x13cc
0,0,1	0x13cc	0x0004	Logical Shift Left	0x3cc0
1,1,0	0x3cc0	0xfa65	NAND	0xc7bf
0,1,1	0xc7bf	0xb2e4	SUB	0x14db

Προσοχή: Η αριθμομηχανή δεν θα υποστηρίζει τις λογικές πράξεις AND και OR, καθώς και τις αριθμητικές πράξεις ολίσθησης κατά δεξιά και κατά αριστερά.

Άσκηση 3

Το αρχείο καταχωρητών είναι άλλο ένα σημαντικό στοιχείο κάθε ψηφιακού συστήματος. Το αρχείο καταχωρητών αποθηκεύει τις τιμές των καταχωρητών που χρησιμοποιούνται από το σύστημα. Σε αυτή την άσκηση, θα δημιουργήσετε ένα αρχείο 16 καταχωρητών το οποίο θα συμπεριλάβετε στον AI accelerator. Το register file θα είναι υπεύθυνο να κρατάει τα βάρη (weights) και τις πολώσεις (biases) του νευρωνικού δικτύου. Ξεκινήστε αυτή την άσκηση δημιουργώντας ένα νέο αρχείο Verilog με όνομα **regfile.v** και προσθέστε τις ακόλουθες θύρες σε αυτή τη μονάδα:

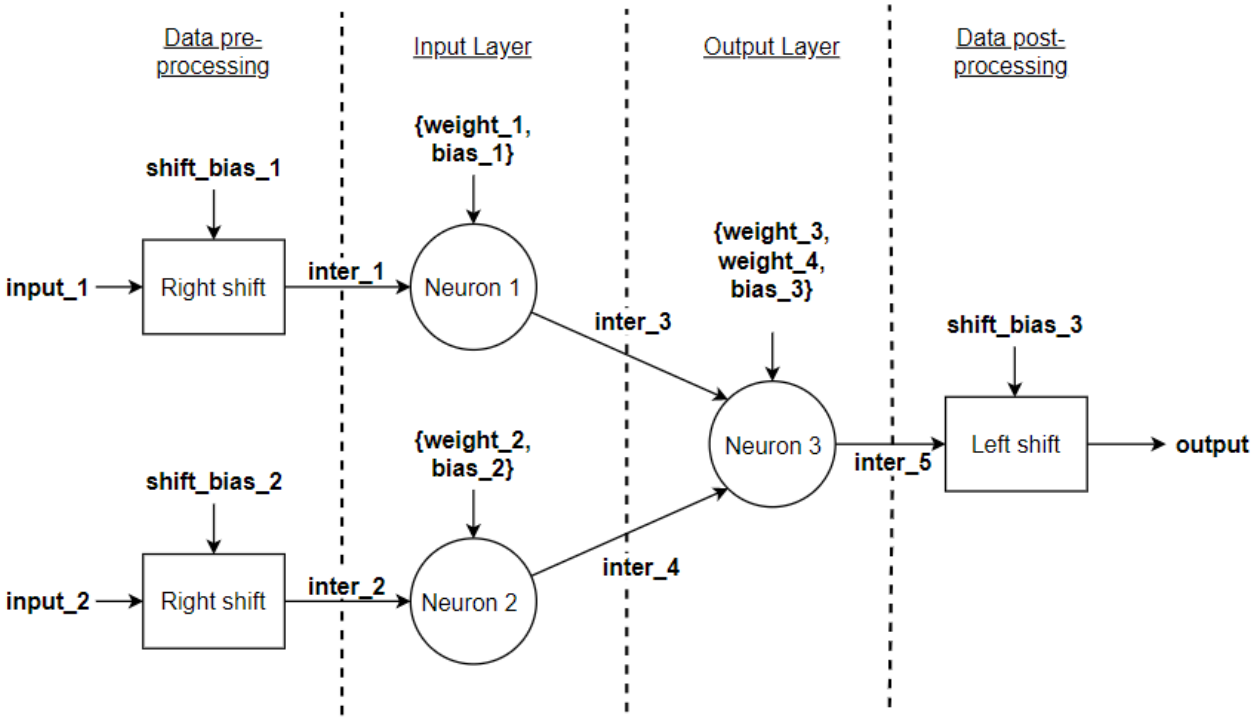
Όνομα θύρας	**Κατεύθυνση**	**Πλάτος [αρ. bit]**	**Σκοπός**
clk	Είσοδος	1	Ρολόι
resetn	Είσοδος	1	Σήμα επαναφοράς (active low)
readReg1	Είσοδος	4	Διεύθυνση για τη θύρα ανάγνωσης 1
readReg2	Είσοδος	4	Διεύθυνση για τη θύρα ανάγνωσης 2
readReg3	Είσοδος	4	Διεύθυνση για τη θύρα ανάγνωσης 3
readReg4	Είσοδος	4	Διεύθυνση για τη θύρα ανάγνωσης 4
writeReg1	Είσοδος	4	Διεύθυνση για θύρα εγγραφής 1
writeReg2	Είσοδος	4	Διεύθυνση για θύρα εγγραφής 2
writeData1	Είσοδος	DATAWIDTH	Δεδομένα προς εγγραφή στη θύρα 1
writeData2	Είσοδος	DATAWIDTH	Δεδομένα προς εγγραφή στη θύρα 2
write	Είσοδος	1	Σήμα ελέγχου που υποδεικνύει εγγραφή
readData1	Έξοδος	DATAWIDTH	Δεδομένα ανάγνωσης από τη θύρα 1
readData2	Έξοδος	DATAWIDTH	Δεδομένα ανάγνωσης από τη θύρα 2
readData3	Έξοδος	DATAWIDTH	Δεδομένα ανάγνωσης από τη θύρα 3
readData4	Έξοδος	DATAWIDTH	Δεδομένα ανάγνωσης από τη θύρα 4

Όπου ως "*DATAWIDTH*" ορίζεται παράμετρος του module "regfile", η οποία θα πρέπει να είναι ίση με 32 bits από προεπιλογή. Υλοποιήστε ένα αρχείο καταχωρητών 16×*DATAWIDTH*-bit. Το σήμα write υποδηλώνει ότι στο register file θα πραγματοποιηθεί εγγραφή δεδομένων (θεωρούμε ότι δεν μπορούμε να πραγματοποιήσουμε ταυτόχρονα και ανάγνωση και εγγραφή).

Μέσω ενός always block, θα γίνεται η εγγραφή ή ανάγνωση των καταχωρητών. Οι καταχωρητές θα πρέπει να αρχικοποιούνται στην τιμή μηδέν, ασύγχρονα με ενεργό χαμηλό σήμα επαναφοράς (**active low reset**). Ύστερα, διαβάστε τις τιμές των καταχωρητών από κάθε έξοδο (readData1, readData2, readData3 και readData4) και ανάλογα με το σήμα write εγγράψτε τα δεδομένα σε κάθε είσοδο (writeData1, writeData2) στην αντίστοιχη διεύθυνση που έχετε ως είσοδο. Σε περίπτωση εγγραφής ενός σήματος θα πρέπει να προσέξετε την περίπτωση που η διεύθυνση εγγραφής είναι ίδια με κάποια από τις διευθύνσεις ανάγνωσης. Σε αυτή την περίπτωση, δώστε προτεραιότητα στην εγγραφή του "writeData".

Άσκηση 4

Στην παρούσα άσκηση καλείστε να σχεδιάσετε το κύκλωμα του νευρωνικού του Σχήματος 6 σε Verilog.



Σχ. 6: Απλό νευρωνικό σύστημα με τρεις νευρώνες.

Το νευρωνικό σύστημα αποτελείται από πράξεις αριθμητικής μετατόπισης δεξιά/αριστερά, πολλαπλασιασμούς και προσθέσεις. Για την υλοποίηση των σταδίων του Σχήματος 6 θα πρέπει να γίνει χρήση της **ALU** που σχεδιάστηκε στην Άσκηση 1, η οποία θα χρησιμοποιηθεί πολλαπλές φορές για την εκτέλεση των εκάστοτε πράξεων, καθώς και το **αρχείο καταχωρητών** της Άσκησης 3 που θα χρησιμοποιηθεί για την αρχική αποθήκευση των βαρών και πολώσεων από την ROM.

Ο καθένας από τους τρεις νευρώνες του Σχήματος 6 πραγματοποιεί μια πράξη MAC (Multiply and Accumulate), δηλαδή ξεκινάει με πολλαπλασιασμό μιας εισόδου με το αντίστοιχο βάρος και συνεχίζει με πρόσθεση της εκάστοτε πόλωσης. Προκειμένου να διευκολυνθεί η διαδικασία

σχεδίασης του συγκεκριμένου νευρωνικού σας ζητείται ως πρώτο βήμα να υλοποιήσετε το συγκεκριμένο MAC module στην Verilog με την ονομασία **mac_unit.v**, το οποίο θα περιλαμβάνει δύο μονάδες ALU συνδεδεμένες σειριακά, όπου η πρώτη **θα εκτελεί πάντα** την πράξη του πολλαπλασιασμού και η δεύτερη **πάντα** την πράξη της πρόσθεσης. Το module θα πρέπει να περιλαμβάνει τις εξής θύρες:

Όνομα θύρας	**Κατεύθυνση**	**Πλάτος [αρ. bit]**	**Σκοπός**
op1	Είσοδος	32	Πρώτη είσοδος του MAC σε συμπλήρωμα ως προς 2
op2	Είσοδος	32	Δεύτερη είσοδος του MAC σε συμπλήρωμα ως προς 2
op3	Είσοδος	32	Τρίτη είσοδος του MAC σε συμπλήρωμα ως προς 2
total_result	Έξοδος	32	Αποτέλεσμα του MAC
zero_mul	Έξοδος	1	Ενεργοποίηση όταν το αποτέλεσμα του πολλαπλασιασμού είναι ίσο με μηδέν
zero_add	Έξοδος	1	Ενεργοποίηση όταν το αποτέλεσμα της πρόσθεσης είναι ίσο με μηδέν
ovf_mul	Έξοδος	1	Ενεργοποίηση όταν υπάρχει υπερχείλιση στην πράξη του πολλαπλασιασμού
ovf_add	Έξοδος	1	Ενεργοποίηση όταν υπάρχει υπερχείλιση στην πράξη της πρόσθεσης

Το νευρωνικό σύστημα του Σχ. 6 θα πρέπει περιλαμβάνει **δύο μονάδες MAC** για προσομοίωση της λειτουργίας ενός νευρώνα και **δύο ALU** για τις πράξεις μετατόπισης. Οι μονάδες αυτές χρησιμοποιούνται με στόχο την παραλληλοποίηση των λειτουργιών σε κάθε στάδιο του FSM.

Με βάση το Σχήμα 6, τα ακόλουθα στάδια πρέπει να υλοποιηθούν με την χρήση ενός FSM:

- ☐ Deactivated state: Το νευρωνικό σύστημα δεν εκτελεί πράξεις. Η εκκίνηση του νευρωνικού θα πραγματοποιηθεί μετά από ενεργοποίηση του σήματος εισόδου **enable** και εφόσον το **resetn** μεταβεί σε χαμηλή τάση. Όποτε έχουμε ξανά επαναφορά του συστήματος (reset) το κύκλωμα θα πρέπει να μεταβαίνει σε αυτή την κατάσταση.
- ☐ Loading Weights & Biases: Το νευρωνικό σύστημα ξεκινάει με την φόρτωση των βαρών και πολώσεων από την ROM στο εσωτερικό register file. Αυτό το βήμα θα πρέπει να συμβεί **μόνο μια φορά,** καθώς μετά την φόρτωση των βαρών το νευρωνικό είναι έτοιμο να εκτελέσει οποιαδήποτε πράξη μέχρις ότου **γίνει εκ νέου reset** και μηδενιστούν οι registers του register file.
- ☐ Data pre-processing Layer: Το συγκεκριμένο στάδιο περιλαμβάνει την παράλληλη προ-επεξεργασία των δεδομένων εισόδου, η οποία πραγματοποιείται μέσω **αριθμητικής ολίσθησης στα δεξιά** από τις δύο ALU. Το μέγεθος της μετατόπισης καθορίζεται από το **shift_bias_1** και το **shift_bias_2**.

- ☐ Input Layer: Σε αυτό το στάδιο οι επεξεργασμένες είσοδοι περνάνε από τους πρώτους δύο νευρώνες, οι οποίοι λειτουργούν παράλληλα. Ο κάθε νευρώνας (MAC) ορίζει μια πράξη **πολλαπλασιασμού** της εισόδου με το αντίστοιχο βάρος (**weight_1, weight_2**) και μια πράξη **πρόσθεσης** με την αντίστοιχη πόλωση (**bias_1, bias_2**), με τον εξής τρόπο:

  inter_3 = inter_1 * weight_1 + bias_1

  inter_4 = inter_2 * weight_2 + bias_2

- ☐ Output Layer: Το επόμενο στάδιο συνδυάζει τα ενδιάμεσα αποτελέσματα του προηγούμενου σταδίου με την ακόλουθη λογική:

  inter_5 = inter_3 * weight_3 + inter_4 * weight_4 + bias_3

  όπου τα **weight_3**, **weight_4** αποτελούν τα βάρη και το **bias_3** την πόλωση του αντίστοιχου νευρώνα. Υλοποιήστε την συγκεκριμένη λογική χρησιμοποιώντας τις δύο μονάδες MAC σειριακά.

- ☐ Data post-processing Layer: Το τελευταίο στάδιο του συστήματος χρησιμοποιεί **αριθμητική ολίσθηση στα αριστερά** μέσω μιας από τις δύο ALU κατά **shift_bias_3** προκειμένου να επεξεργαστεί τα δεδομένα εξόδου (inter_5).
- ☐ Idle state: Το νευρωνικό σύστημα γίνεται ανενεργό και περιμένει για τις επόμενες εισόδους. Μόλις οι επόμενες είσοδοι είναι έτοιμοι, το σήμα enable θα πρέπει να μεταβεί σε υψηλή τάση και το νευρωνικό να ξεκινήσει ένα νέο κύκλο πράξεων, ξεκινώντας πάντα με την κατάσταση Data pre-processing Layer.

Προσοχή: Χρησιμοποιείστε 3 bits για την αναπαράσταση του εκάστοτε σταδίου, καθώς το FSM σας θα πρέπει να έχει **7 στάδια**.

Σε περίπτωση υπερχείλισης (overflow), σε οποιοδήποτε στάδιο υπολογισμού, η έξοδος του νευρωνικού κυκλώματος θα πρέπει να οδηγείται στον μέγιστο δυνατό θετικό αριθμό που μπορεί να αναπαρασταθεί με 32 bits. Όποτε συμβαίνει υπερχείλιση σε οποιοδήποτε στάδιο του FSM θα πρέπει το FSM σας να μεταβεί απευθείας στην κατάσταση IDLE και να εμφανίσει ως έξοδο απευθείας το μέγιστο δυνατό θετικό αριθμό. Κατά αυτόν τον τρόπο παρακάμπτονται οι ενδιάμεσοι περιττοί υπολογισμοί, εφόσον το overflow μας καθιστά το αποτέλεσμα μη-έγκυρο.

Σημείωση: Η επιλογή του τύπου FSM (Mealy & Moore) αφήνεται πάνω σας, ωστόσο θα πρέπει να σημειώσετε στην αναφορά σας το είδος του FSM που επιλέξατε να χρησιμοποιήσετε και γιατί. Επίσης, η επιλογή για το αν θα κρατάτε τα δεδομένα σε ενδιάμεσους registers μετά από κάθε στάδιο του FSM ή θα τα αποθηκεύετε μέσα στο register file επίσης αφήνεται πάνω σας, ωστόσο θα πρέπει να περιγράψετε με λεπτομέρεια ποιον τρόπο υλοποίησης επιλέξατε και γιατί.

Για το στάδιο “Loading Weights & Biases” οι τιμές των βαρών και πολώσεων της ROM πρέπει να αποθηκεύονται στο register file με βάση τον ακόλουθο πίνακα:

Διεύθυνση καταχωρητή	**Σήμα αποθήκευσης**	**Σκοπός**
0x0	0	reserved to zero

0x1	0	reserved to zero
0x2	shift_bias_1	Τιμή μετατόπισης εισόδου 1
0x3	shift_bias_2	Τιμή μετατόπισης εισόδου 2
0x4	weight_1	Βάρος νευρώνα 1
0x5	bias_1	Πόλωση νευρώνα 1
0x6	weight_2	Βάρος νευρώνα 2
0x7	bias_2	Πόλωση νευρώνα 2
0x8	weight_3	Βάρος νευρώνα 3 - 1ης εισόδου
0x9	weight_4	Βάρος νευρώνα 3 - 2ης εισόδου
0x10	bias_3	Πόλωση νευρώνα 3
0x11	shift_bias_3	Τιμή μετατόπισης εξόδου
0x12 - 0x15	-	reserved to zero or for intermediate FSM results

Το νευρωνικό σύστημα που θα υλοποιήσετε **nn.v,** θα πρέπει να έχει τις ακόλουθες θήρες:

Όνομα θύρας	**Κατεύθυνση**	**Πλάτος [αρ. bit]**	**Σκοπός**
input_1	Είσοδος	32	Πρώτη είσοδος του νευρωνικού σε συμπλήρωμα ως προς 2
input_2	Είσοδος	32	Δεύτερη είσοδος του νευρωνικού σε συμπλήρωμα ως προς 2
clk	Είσοδος	1	Το ρολόι του συστήματος
resetn	Είσοδος	1	Ασύγχρονο και ενεργά χαμηλό σήμα επαναφοράς
enable	Είσοδος	1	Σήμα ενεργοποίησης του συστήματος. Όταν είναι ενεργό, το σύστημα ξεκινάει και μεταβαίνει από την κατάσταση IDLE στην επόμενη κατάσταση.
final_output	Έξοδος	32	Η έξοδος του συστήματος
total_ovf	Έξοδος	1	Δείχνει αν υπάρχει υπερχείλιση σε κάποιο στάδιο του συστήματος

total_zero	Έξοδος	1	Δείχνει αν υπάρχει κάποιο μηδενικό αποτέλεσμα σε κάποιο στάδιο
ovf_fsm_stage	Έξοδος	3	Δείχνει το στάδιο στο οποίο υπάρχει υπερχείλιση. Αν δεν υπάρχει πουθενά υπερχείλιση τότε είναι ίσο με 111
zero_fsm_stage	Έξοδος	3	Δείχνει το στάδιο στο οποίο υπάρχει μηδενικό αποτέλεσμα. Αν δεν υπάρχει μηδενικό τότε είναι ίσο με 111

Testbench

Για τον έλεγχο της σωστής λειτουργίας του κυκλώματος νευρωνικού δικτύου θα δημιουργήσετε ένα αρχείο testbench **tb_nn.v**. Το testbench θα συγκρίνει το αποτέλεσμα της εξόδου του νευρωνικού κυκλώματος με την έξοδο αναφοράς (reference output), της συνάρτησης **nn_model** που θα βρείτε στο αντίστοιχο αρχείο. Η συνάρτηση αναφοράς λαμβάνει ως είσοδο δύο 32-bit προσημασμένους αριθμούς (input_1 και input_2) και επιστρέφει ως έξοδο το αποτέλεσμα αναφοράς, που είναι η 32-bit προσημασμένη έξοδος (output).

Το testbench θα πρέπει να δημιουργεί ένα σήμα ρολογιού με περίοδο **10 ns** και duty cycle 50%, και θα περιέχει και ένα instance του νευρωνικού κυκλώματος. Σε περίπτωση που η έξοδος του νευρωνικού δικτύου δεν συμβαδίζει με την έξοδο του μοντέλου θα πρέπει να τυπώνετε τη χρονική στιγμή προσομοίωσης που γίνεται το λάθος, τις 2 εισόδους του κυκλώματος, την έξοδο του κυκλώματος και την αντίστοιχη έξοδο της συνάρτησης αναφοράς. Τέλος, θα πρέπει να μετράτε τον αριθμό των σωστών συγκρίσεων και να τυπώνετε μήνυμα του αριθμού σωστών συγκρίσεων στις συνολικές δοκιμές που έγιναν (number of PASS/ number of test cases).

Θα πρέπει να να ελέγξετε το κύκλωμα για 100 επαναλήψεις, όπου κάθε επανάληψη θα περιέχει τα εξής τεστ, που θα εκτελούνται διαδοχικά:

- ☐ τυχαία παραγόμενο ζεύγος εισόδου με εύρος τιμών [-4096, 4095].
- ☐ τυχαία παραγόμενο ζεύγος εισόδου με εύρος τιμών [max_positive_number / 2 - max_positive_number], για την κάλυψη ελέγχου της περίπτωσης υπερχείλισης με θετικούς αριθμούς.
- ☐ τυχαία παραγόμενο ζεύγος εισόδου με εύρος τιμών [max_negative_number - max_negative_number / 2], για την κάλυψη ελέγχου της περίπτωσης υπερχείλισης με αρνητικούς αριθμούς.

Για την παραγωγή των τυχαίων τιμών σε δεδομένο εύρος, χρησιμοποιήστε την συνάρτηση ***$urandom_range().***

Σημείωση: Το μοντέλο αναφοράς είναι μια απλή συνάρτηση verilog, οπότε και παράγει το αποτέλεσμα απευθείας τη στιγμή που εισάγονται οι είσοδοι. Λάβετε υπόψη την καθυστέρηση του κυκλώματος του νευρωνικού δικτύου, ώστε να καθυστερείτε τον κατάλληλο χρόνο το αποτέλεσμα αναφοράς, για να γίνεται σωστά η σύγκριση μεταξύ των 2 αποτελεσμάτων.

Παραδοτέα Ασκήσεων 1-4

Για τις παραπάνω ασκήσεις θα πρέπει να υποβάλετε μία αναφορά η οποία να περιέχει τα ακόλουθα και με την εξής σειρά:

1) Το αρχείο **alu.v** της άσκησης 1.
2) Τα αρχεία **calc.v, calc_enc.v, calc_tb.v** της άσκησης 2.
3) Το αρχείο **regfile.v** της άσκησης 3.
4) Τα αρχεία **mac_unit.v, nn.v, tb_nn.v** της άσκησης 4.

Στην αναφορά σας συμπεριλάβετε **σχηματικό διάγραμμα από το FSM** της Άσκησης 4 καθώς και **κυματομορφές προσομοίωσης** από τις Ασκήσεις 2 και 4. Επίσης, προσθέστε σύντομες αναφορές στη διαδικασία που ακολουθήσατε για σχεδίαση και υλοποίηση κάθε module από κάθε άσκηση.

Η αναφορά πρέπει να υποβληθεί ως 1 αρχείο pdf όπου θα αναφέρεται το όνομά σας και το ΑΕΜ σας. Σε διαφορετική περίπτωση δε θα δίνεται κανένας βαθμός στην εργασία.

Βαθμολόγηση: Η εργασία είναι ατομική και υποχρεωτική, βαθμολογείται με άριστα το 30 και ισοδυναμεί με το 30% του συνολικού βαθμού του μαθήματος (δηλ. μέχρι 3 βαθμούς). Μη υποβολή ή εκπρόθεσμη υποβολή της εργασίας ισοδυναμεί με απώλεια τριών βαθμών στο συνολικό βαθμό. Επίσης, η βαθμολόγηση γίνεται με βάση τη συνολική εικόνα της αναφοράς.

Η προθεσμία υποβολής της εργασίας είναι: **21 Ιανουαρίου 2026 στις 23:59**, μόνο μέσω της πλατφόρμας elearning.

Αναφορές

ECEN 323: Computer Organization Home • ECEn 323: Computer Organization. Available at: https://ecen323wiki.groups.et.byu.net/ (Accessed: 12 November 2023).