ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ — ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ — SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ — DEPARTMENT OF INFORMATICS

ΣΧΕΔΙΑΣΗ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ακαδημαϊκό Έτος 2016-2017, Εαρινό Εξάμηνο

Εργαστηριακή Άσκηση 3 (10%)

→ Ημερομηνία παράδοσης: 28 Μαΐου 2018, 11.59μμ ←

Αντικείμενο εργασίας

Στόχος της 3ης εργασίας είναι η **υλοποίηση** σε γλώσσα VHDL ενός **μικρού επεξεργαστή** που θα εκτελεί ένα συγκεκριμένο σύνολο εντολών.

*Πριν ξεκινήσετε την εκπόνηση της εργασίας, μελετήστε προσεκτικά τα σχετικά παραδείγματα σχεδίασης του βιβλίου Δομή Λεωφόρου (**7.14.1**) και Ένας απλός επεξεργαστής (**7.14.2**) καθώς και τις αντίστοιχες διαφάνειες του μαθήματος.*

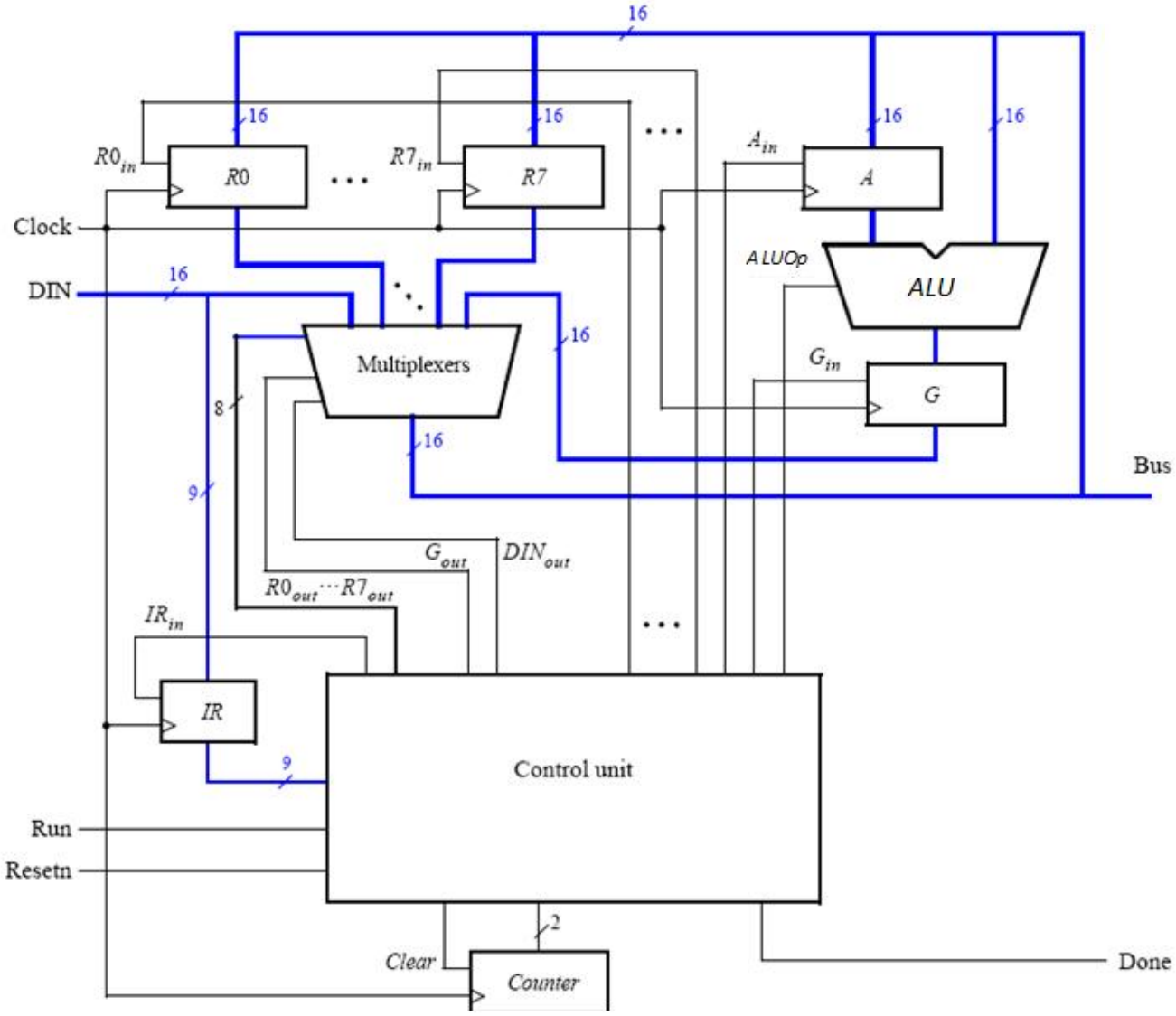
1. Σχεδιάστε και υλοποιήστε τον **επεξεργαστή**, όπως αυτός περιγράφεται στο **Σχήμα 1**, με χρήση της γλώσσας VHDL. Στο eClass παρέχεται ένας ενδεικτικός σκελετός κώδικα σε VHDL (Εργασία3_code), καθώς και μερικά υποκυκλώματα που μπορείτε να χρησιμοποιήσετε στην υλοποίησή σας. Δεν είναι απαραίτητο να κάνετε χρήση του κώδικα αυτού. Να έχετε περιεκτικά σχόλια στην VHDL, ιδιαιτέρως όταν χρειάζεται να εξηγήσετε «δύσκολα» κομμάτια κώδικα. Χρησιμοποιήστε τα συγκεκριμένα ονόματα (όπου δίνονται) για input/output των κυκλωμάτων.
2. Ελέγξτε την ορθότητα του κώδικά σας με χρήση του λογισμικού Quartus και χρησιμοποιήστε λειτουργική προσομοίωση (functional simulation) για να επιβεβαίωσετε ότι ο κώδικάς σας είναι σωστός και ο επεξεργαστής λειτουργεί όπως πρέπει. Για τον έλεγχο του κυκλώματος χρησιμοποιήστε την ακολουθία εντολών που παρατίθεται στο τέλος της εκφώνησης.

Ανάλυση-επεξηγήσεις

Στο **Σχήμα 1** παρουσιάζεται ένα *ψηφιακό σύστημα* το οποίο περιέχει:

- οκτώ καταχωρητές 16-bit (registers): **R0**-**R7**,
- έναν καταχωρητή 16-bit **A**,
- έναν καταχωρητή 16-bit **G**,
- έναν καταχωρητή εντολών 9-bit (**IR**: instruction register),
- έναν πολυπλέκτη 16-bit (**Multiplexers**),
- μία αριθμητική λογική μονάδα **ALU**,
- έναν απαριθμητή (**Counter**), και
- μια μονάδα ελέγχου (**Control Unit**).

Τα **δεδομένα** εισάγονται στο σύστημα από την είσοδο **16-bit *DIN***. Τα δεδομένα φορτώνονται στον πολυπλέκτη 16-bit και στη συνέχεια μέσω της λεωφόρου (bus) στους διάφορους καταχωρητές *R0, …, R7* και *A*, ανάλογα με τη λειτουργία του επεξεργαστή που υλοποιείται κάθε φορά. Ο πολυπλέκτης επίσης επιτρέπει την μεταφορά δεδομένων από τον έναν καταχωρητή στον άλλον. Τα σύρματα εξόδου του πολυπλέκτη (μήκος 16-bit) υλοποιούν ουσιαστικά μία **λεωφόρο ή δίαυλο (bus)** ώστε να μεταφέρονται τα δεδομένα από ένα σημείο του συστήματος σε ένα άλλο.



ΣΧΗΜΑ 1: ένας απλός επεξεργαστής

Το σύστημα μπορεί να εκτελέσει διάφορες **λειτουργίες** σε κάθε κύκλο ρολογιού, όπως ορίζει η μονάδα ελέγχου *Control Unit.* Η μονάδα αυτή καθορίζει πότε τα συγκεκριμένα δεδομένα μεταφέρονται στα σύρματα του bus και ελέγχει σε ποιόν από τους καταχωρητές θα φορτωθούν. Για παράδειγμα, αν η μονάδα ελέγχου στείλει τα σήματα *R0out* και *Ain*, ο πολυπλέκτης θα τοποθετήσει τα δεδομένα του καταχωρητή *R0* στο bus και αυτά τα δεδομένα θα φορτωθούν στον επόμενο κύκλο ρολογιού στον καταχωρητή *A. Για να καταλάβετε τη λειτουργία και υλοποίηση της λεωφόρου, μελετήστε το αντίστοιχο παράδειγμα από το βιβλίο (κεφ.* ***7.14.1****) και τις διαφάνειες του μαθήματος.*

Ένα σύστημα όπως αυτό το ονομάζουμε συνήθως **επεξεργαστή (processor).** Εκτελεί λειτουργίες με τη μορφή εντολών (instructions). Στο μάθημα *Οργάνωση Η/Υ* θα μάθετε μία γλώσσα assembly που περιέχει μια πιο πολύπλοκη συλλογή εντολών και στο μάθημα *Αρχιτεκτονική Η/Υ* θα κατασκευάσετε πιο πολύπλοκους επεξεργαστές για τη γλώσσα assembly που θα μάθετε στο δεύτερο έτος. Στο μάθημα αυτό θα δούμε μια πολύ απλή γλώσσα assembly, με λίγες εντολές επεξεργασίας δεδομένων.

Ο **Πίνακας 1** παρουσιάζει τις **εντολές** που υποστηρίζει ο συγκεκριμένος επεξεργαστής. Η στήλη "*Opcode*" δείχνει την κωδικοποίηση που θα χρησιμοποιηθεί για την κάθε εντολή. Η στήλη "*Operation*" δείχνει το όνομα της εντολής και τους τελεστέους που τη συνοδεύουν. Η στήλη "*Function Performed*" περιγράφει τη λειτουργία που επιτελεί η κάθε εντολή. Η σύνταξη RX ← [RY] σημαίνει ότι τα περιεχόμενα του καταχωρητή RY φορτώνονται στον καταχωρητή RX. Η εντολή **mv** (move) επιτρέπει στα δεδομένα να αντιγραφούν από τον ένα καταχωρητή στον άλλο. Για την εντολή **mvi** (move immediate), η διατύπωση RX ← D σημαίνει ότι η σταθερά 16-bit D πρέπει να φορτωθεί στον καταχωρητή RX.

Opcode	Operation	Function Performed
000	**mv** *Rx,Ry*	Rx ←[Ry]
001	**mvi** *Rx, #D*	Rx ←D
010	**and** *Rx, Ry*	Rx ←[Rx] AND [Ry]
011	**or** *Rx, Ry*	Rx ←[Rx] OR [Ry]
100	**add** *Rx, Ry*	Rx ←[Rx]+[Ry]
101	**sub** *Rx, Ry*	Rx ←[Rx]-[Ry]
110	**xor** *Rx, Ry*	Rx ←[Rx] XOR [Ry]
111	**not** *Rx, Ry*	Rx ←NOT [Ry]

Πίνακας 1: Εντολές του επεξεργαστή

Κάθε εντολή που λαμβάνεται από τον επεξεργαστή μέσω του σήματος 16-bit **DIN**, μπορεί να κωδικοποιηθεί και να αποθηκευτεί στον καταχωρητή ***IR*** χρησιμοποιώντας το 9-bit format IIIXXXYYY, όπου III αναπαριστά τον κωδικό της εντολής (opcode), XXX τον καταχωρητή RX και YYY τον καταχωρητή RY. Η **εντολή** περιέχεται στα πρώτα 9 bits από αριστερά (τα πιο σημαντικά) του σήματος εισόδου 16-bit DIN, ενώ τα υπόλοιπα 7 bits του DIN θα έχουν την τιμή 0 και θα αγνοούνται.
Μέσω του σήματος εισόδου 16-bit **DIN**, λαμβάνονται από τον επεξεργαστή και οι **δυαδικοί αριθμοί 16-bit προς επεξεργασία**. Προκειμένου να εκτελεστεί οποιαδήποτε εντολή επεξεργασίας από τον επεξεργαστή, θα πρέπει να έχει προηγηθεί η φόρτωση των αντίστοιχων δυαδικών αριθμών 16-bit σε κάποιον/κάποιους καταχωρητές μέσω της εντολής **mvi**.
Ειδικά για την εντολή **mvi**, το πεδίο YYY δεν έχει νόημα (δεν εμπλέκεται δεύτερος καταχωρητής), ενώ η τιμή D που θα φορτωθεί στον άλλο καταχωρητή είναι η αμέσως επόμενη τιμή που θα σταλεί στο 16-bit *DIN* input, αμέσως αφού η εντολή **mvi** έχει αποθηκευτεί στο *IR*. Επομένως η **mvi** εντολή θα έχει μήκος 6 αντί για 9 bits, και αμέσως μετά ο επεξεργαστής θα περιμένει να λάβει από την είσοδο DIN τον αριθμό 16-bit που θα τοποθετήσει στον αντίστοιχο καταχωρητή που έχει οριστεί από την mvi.

Παράδειγμα ακολουθίας εντολών:

Εντολές	**Επεξήγηση**
001000	mvi R0
1000101000111010	Ο 16-bit αριθμός που θα τοποθετηθεί στον R0
001001	mvi R1
1000101000110011	Ο 16-bit αριθμός που θα τοποθετηθεί στον R1
110000001	xor R0 R1

Ο επεξεργαστής εκτελεί τις αριθμητικές εντολές χρησιμοποιώντας παρόμοια **ALU** με αυτή που υλοποιήσατε στην 2η εργασία. Το σύνολο των εντολών που εκτελεί η ALU είναι οι εξής:

- Αριθμητικές πράξεις (με έλεγχο υπερχείλισης):
  - Πρόσθεση (ADD)
  - Αφαίρεση (SUB)
- Λογικές πράξεις:
  - Σύζευξη (AND)
  - Διάζευξη (OR)
  - Αποκλειστική διάζευξη (XOR)
  - Εύρεση συμπληρώματος (NOT)

Προκειμένου να εκτελεστούν οι πράξεις/εντολές της ALU, ο *πολυπλέκτης* τοποθετεί πρώτα έναν αριθμό 16-bit (από κάποιον καταχωρητή R) μέσω του bus στον καταχωρητή **A**. Στη συνέχεια ένας δεύτερος αριθμός 16-bit μεταφέρεται στο **bus**, η μονάδα ALU κάνει την πράξη ανάμεσα στους δύο αριθμούς (από καταχωρητή A και από bus) και τοποθετεί το αποτέλεσμα στον καταχωρητή **G**. Τα δεδομένα από το G μπορούν μετά να μεταφερθούν αν χρειάζεται σε κάποιον άλλο καταχωρητή. *Για την υλοποίηση της λειτουργίας της ALU μπορείτε να χρησιμοποιήσετε κώδικα από την προηγούμενη εργασία ή να λάβετε υπόψη σας τον ενδεικτικό κώδικα που δίνεται στο eClass.*

Μερικές εντολές, όπως η προσθαφαίρεση, μπορεί να διαρκέσουν παραπάνω από ένα κύκλο ρολογιού (T), γιατί πρέπει να γίνουν πολλαπλές μεταφορές στο bus. Η μονάδα ελέγχου χρησιμοποιεί τον **2-bit απαριθμητή (counter)**, που φαίνεται στο Σχήμα 1, για να μπορέσει να εκτελέσει βήμα-βήμα τέτοιες εντολές.
Ο επεξεργαστής αρχίζει να εκτελεί την εντολή στο *DIN* input όταν το σήμα ***Run*** έχει σταλεί και τελειώνει στέλνοντας το σήμα εξόδου ***Done***. Ο **Πίνακας 2** δείχνει τα σήματα ελέγχου που στέλνονται σε κάθε βήμα (χρονική στιγμή T) για να υλοποιηθούν οι εντολές του Πίνακα 1.

	T_0	T_1	T_2	T_3
(mv): I_0	*IRin*	*RYout, Rxin Done*		
(mvi): I_1	*IRin*	*DINout, RXin, Done*		
(add): I_2	*IRin*	*RXout, Ain*	*RYout, Gin ALUop*	*Gout, RXin Done*
(sub): I_3	*IRin*	*RXout, Ain*	*RYout, Gin ALUop*	*Gout, RXin Done*
(and): I_4	*IRin*	*RXout, Ain*	*RYout, Gin, ALUop*	*Gout, RXin Done*
(or): I_5	*IRin*	*RXout, Ain*	*RYout, Gin ALUop*	*Gout, RXin Done*
(not): I_6	*IRin*	*RXout, Ain*	*RYout, Gin ALUop*	*Gout, RXin Done*
(xor): I_7	*IRin*	*RXout, Ain*	*RYout, Gin ALUop*	*Gout, RXin Done*

Πίνακας 2: Σήματα ελέγχου από το Control Unit για κάθε εντολή και χρονική στιγμή (Ti)

Σε περίπτωση που συμβεί υπερχείλιση, ο επεξεργαστής θα πρέπει να επανεκκινήσει τη λειτουργία του (χρήση του σήματος Reset).

Για παράδειγμα, αν οι καταχωρητές R1 = 0000000000000001, R2 = 0000000000000011, τότε η εντολή mv R1, R2 (opcode 000) σημαίνει R1 ←[R2], δηλαδή ότι καταχωρούμε στο R1 το 0000000000000011, ενώ εάν η εντολή είναι η not (opcode 111), σημαίνει οτι το R1 ←NOT[R2], δηλαδή ότι καταχωρούμε στο R1 το 1111111111111100.

Κατά την Λειτουργική Προσομοίωση, ελέγξτε την ορθότητα του κυκλώματός σας για τις ακόλουθες εντολές:

1. Φόρτωση (mvi) του αριθμού 0000011100001111 στον καταχωρητή R0.
2. Φόρτωση (mvi) του αριθμού 0001100100111111 στον καταχωρητή R1.
3. Πρόσθεση (add) των αριθμών που βρίσκονται στους καταχωρητές R0 και R1.
4. Φόρτωση (mvi) του αριθμού 0011000000001111 στον καταχωρητή R2.
5. Εκτέλεση πράξης AND των αριθμών που βρίσκονται στους καταχωρητές R1 και R2.
6. Φόρτωση (mvi) του αριθμού 0001000101010011 στον καταχωρητή R3.
7. Εκτέλεση πράξης NOT στον αριθμό που βρίσκεται στον καταχωρητή R3 και αποθήκευση του αποτελέσματος στον καταχωρητή R4.
8. Εκτέλεση πράξης XOR των αριθμών που βρίσκονται στους καταχωρητές R1 και R4.
9. Εκτέλεση πράξης OR των αριθμών που βρίσκονται στους καταχωρητές R2 και R3.
10. Αφαίρεση (sub) των αριθμών που βρίσκονται στους καταχωρητές R2 και R1.
11. Αντιγραφή των περιεχομένων (mv) του καταχωρητή R4 στον καταχωρητή R5.

Οδηγίες

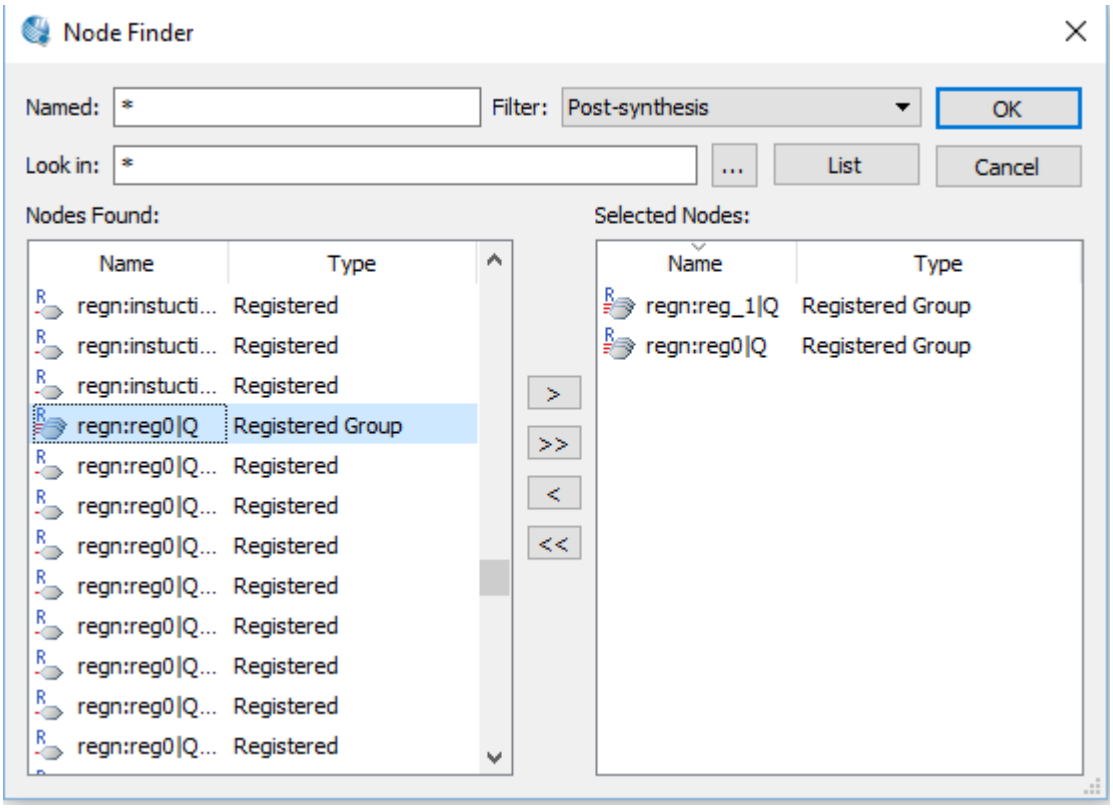
Οι ομάδες θα απαρτίζονται από τα ίδια άτομα όπως και στις προηγούμενες εργασίες.
Θα παραδώσετε μέσω του eClass, ένα αρχείο zip/rar, που θα το ονομάσετε με τους αρ. μητρώου των μελών της ομάδας (π.χ. 3170415_3170400_3170452.rar), το οποίο θα περιλαμβάνει:

- ένα project του Quartus (συμπιέστε ολόκληρο τον αντίστοιχο φάκελο του project) με την απάντηση στα ζητούμενα της άσκησης.
- ένα αρχείο σε μορφή PDF στο οποίο θα περιέχονται τα στοιχεία (ονοματεπώνυμο, αρ. μητρώου και e-mail) των μελών της ομάδας, οι κυματομορφές που προκύπτουν κατά την λειτουργική προσομοίωση (για τις παραπάνω εντολές), καθώς επίσης και ο,τιδήποτε θεωρείτε απαραίτητο να αναφέρετε σχετικά με την υλοποίηση της άσκησης (σχολιασμός, επεξήγηση, screenshots, κ.λπ.).
- Προσοχή! Ο κώδικας VHDL θα πρέπει να συνοδεύεται από επεξηγηματικά σχόλια.

Βοηθητικές Παρατηρήσεις

1. Για να συμπεριλάβετε στο functional simulation (μέσω waveforms) τα **internal signals** (π.χ. σήματα εξόδου καταχωρητών), θα πρέπει στο Simulation Waveform Editor, εκεί που εισάγετε τα nodes του κυκλώματος, να επιλέξετε στο node finder ως filter “Post-synthesis”, έτσι ώστε να σας εμφανίσει όλα τα σήματα που προκύπτουν αφού έχει γίνει η διαδικασία της σύνθεσης και να επιλέξετε τα internal signals που θέλετε να εμφανίζονται στο waveform σας. Στην περίπτωση αυτή, τα signals δεν θα εμφανίζονται ως IN ή OUT αλλά ως register (όταν πρόκειται για παράδειγμα για σήμα από register) και θα πάρουν τιμές όταν τρέξετε την προσομοίωση, όπως συμβαίνει με τα σήματα εξόδου (αρχικά θα είναι undefined).

Ενδεικτικό παράδειγμα:



2. Ο κώδικας που δίνεται ως βοηθητικός για την ALU είναι ενδεικτικός και μπορεί να χρειάζεται αλλαγές για να χρησιμοποιηθεί και να λειτουργεί ακριβώς όπως απαιτείται από την εκφώνηση.