Σχεδίαση Ψηφιακών Συστημάτων

Ακαδ. Έτος 2017-2018, Εαρινό Εξάμηνο

Εργαστηριακή Άσκηση 2 (10%)

→ Ημερομηνία παράδοσης: Παρασκευή 11 Μαΐου 2018, 11.59μμ ←

Ανάλυση Εργασίας

Στόχος της δεύτερης Εργαστηριακής Άσκησης είναι ο *ιεραρχικός* σχεδιασμός και η μελέτη μιας Αριθμητικής και Λογικής Μονάδας (**Arithmetic Logic Unit – ALU**).

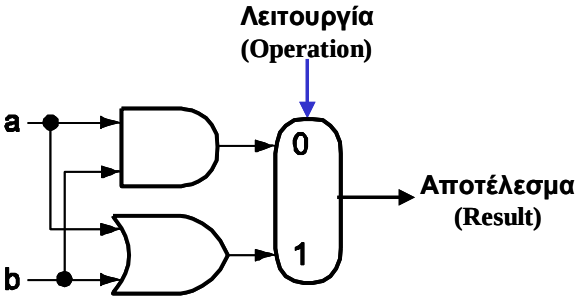
Η ALU εκτελεί τις **αριθμητικές πράξεις** όπως πρόσθεση και αφαίρεση και τις **λογικές πράξεις** όπως το AND και OR. Εμείς θα κατασκευάσουμε για αυτή την εργασία μια ALU των 16-bits βασιζόμενοι σε πέντε δομικά στοιχεία (πύλες AND, OR και XOR, NOT και πολυπλέκτες) που θα εκτελεί τις παρακάτω πράξεις:

- Αριθμητικές πράξεις (με έλεγχο υπερχείλισης):
  - Πρόσθεση (ADD)
  - Αφαίρεση (SUB)
- Λογικές πράξεις:
  - Σύζευξη (AND)
  - Διάζευξη (OR)
  - Αποκλειστική διάζευξη (XOR)
  - Αντίθετο της διάζευξης (NOR)

Η σχεδίασή μας θα γίνει **ιεραρχικά**. Τα **βήματα** που θα ακολουθήσουμε για να την κατασκευάσουμε εξηγούνται στη συνέχεια:

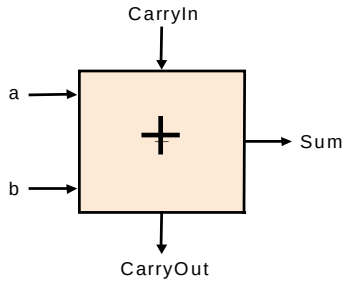
1. Υλοποίηση ενός κομματιού (slice) της ALU του 1-bit για τις πράξεις AND, OR και πρόσθεση

Λειτουργίες **AND** και **OR:** Η λογική μονάδα του 1 bit για το AND και το OR μοιάζει με αυτή της *εικόνας 1*. Ο πολυπλέκτης στα δεξιά επιλέγει το (a AND b) αν Operation=0 ή το (a OR b) αν Operation=1, όπου a, b σήματα του 1 bit.



Εικόνα 1: ***Βασική λειτουργία AND και OR με πολυπλέκτη***

Λειτουργία **πρόσθεσης:** Ένας αθροιστής πρέπει να έχει 2 εισόδους 1-bit για τους τελεστέους και μια έξοδο 1-bit για το άθροισμα. Πρέπει να υπάρχει μια δεύτερη έξοδος για να μεταβιβάζεται το κρατούμενο, που ονομάζεται CarryOut (Έξοδος κρατουμένου). Αφού το CarryOut απο τον γειτονικό αθροιστή πρέπει να συμπεριληφθεί ως είσοδος χρειαζόμαστε μια 3η είσοδο που ονομάζουμε CarryIn (Είσοδος κρατουμένου). Η *εικόνα 2* δείχνει τον αθροιστή που περιγράψαμε.

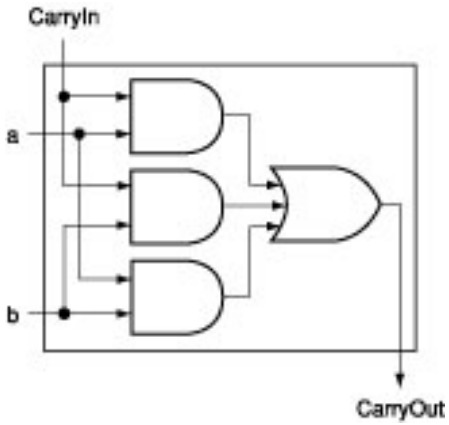


Αυτός ο αθροιστής ονομάζεται **Πλήρης Αθροιστής** (full adder). Ονομάζεται επίσης αθροιστής (3,2) επειδή έχει 3 εισόδους και 2 εξόδους.

Εικόνα 2: ***Πλήρης αθροιστής***

Οι συναρτήσεις εξόδου μπορούν να εκφραστούν ως λογικές εξισώσεις άρα και με λογικές πύλες. Όπως γνωρίζουμε, κατά την πράξη της άθροισης, προκύπτει κρατούμενο (**CarryOut**) όταν ισχύει η ακόλουθη λογική εξίσωση:

CarryOut = (b • CarryIn) + (a • CarryIn) + (a • b), την οποία μπορούμε να αναπαραστήσουμε ως εξής:

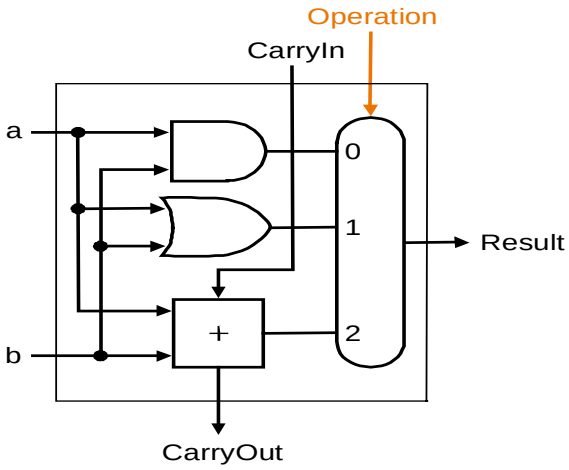


Εικόνα 3: ***Τμήμα του αθροιστή για την υλοποίηση του σήματος CarryOut.***

Η έξοδος **Sum** παίρνει την τιμή 1, όταν ακριβώς μία είσοδος έχει τιμή 1 ή όταν και οι τρεις είσοδοι έχουν τιμή 1. Δηλαδή:

$$Sum = \left(a \bullet \bar{b} \bullet \overline{\text{CarryIn}}\right) + \left(\bar{a} \bullet b \bullet \overline{\text{CarryIn}}\right) + \left(\bar{a} \bullet \bar{b} \bullet \text{CarryIn}\right) + \left(a \bullet b \bullet \text{CarryIn}\right)$$

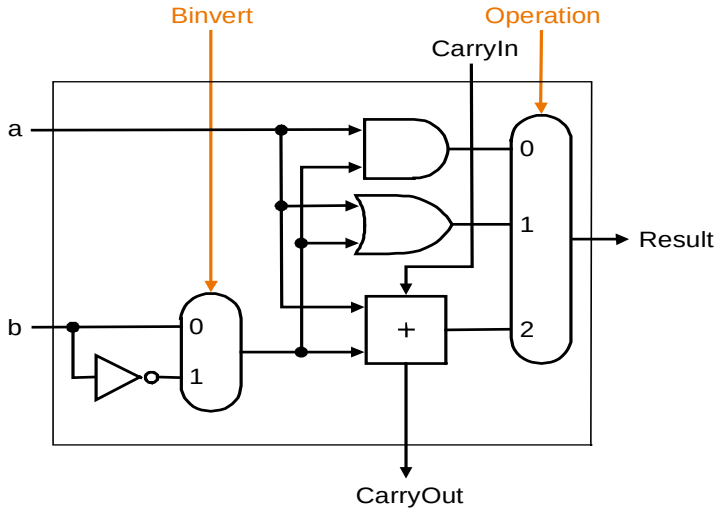
Το κομμάτι (slice) της ALU του 1 bit που έχει δημιουργηθεί από το συνδυασμό των λειτουργιών του αθροιστή και των λογικών AND και OR φαίνεται στην εικόνα 4.



Εικόνα 4: ***Μια ALU του 1-bit που εκτελεί τις πράξεις AND (0), OR (1) και πρόσθεση (2)***

2. Προσθέτοντας στην ALU μας την αφαίρεση και την πράξη NOR

Αντί για **αφαίρεση**, προσθέτουμε το συμπλήρωμα ως προς 2 του αφαιρετέου! Για να βρούμε το συμπλήρωμα ως προς 2 ενός αριθμού, αντιστρέφουμε κάθε bit (*συμπλήρωμα ως προς 1*) και μετά να προσθέτουμε 1. Για να αντιστρέψουμε κάθε bit, προσθέτουμε ένα πολυπλέκτη 2-σε-1 που επιλέγει ανάμεσα από *b* και $\bar{b}$ όπως φαίνεται στην παρακάτω εικόνα.



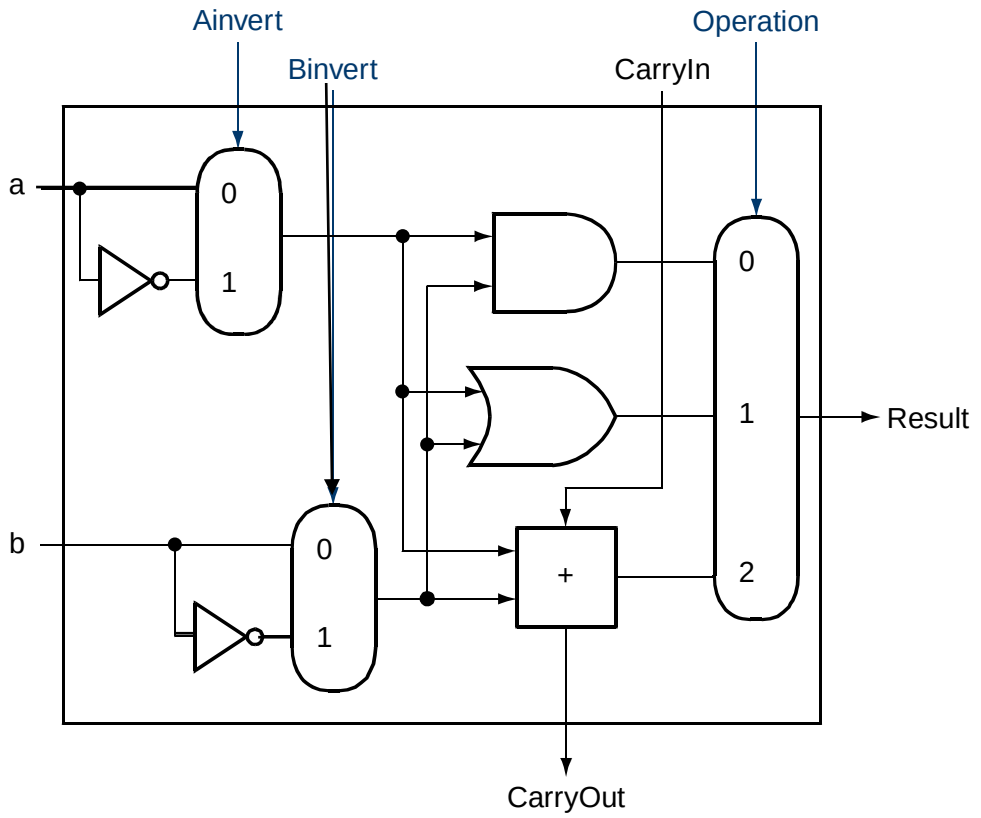
Εικόνα 5: ***Μια ALU του 1-bit που εκτελεί τις πράξεις AND, OR και πρόσθεση των a και b ή των a και*** $\bar{b}$ ***(αφαίρεση).***

Θέτοντας Binvert = 1 (δηλ. επιλέγοντας το $\bar{b}$) και CarryIn = 1, πραγματοποιείται **αφαίρεση**, προσθέτοντας το συμπλήρωμα ως προς 2 του b στο a, αλλιώς πραγματοποιείται πρόσθεση του b στο a.

Για την υλοποίηση της πράξης NOR: Σύμφωνα με το θεώρημα De Morgan $\left(\overline{a+b}\right)=\bar{a}\bullet\bar{b}$

Άρα αφού έχουμε το AND και το NOT b χρειάζεται να προσθέσουμε μόνο το NOT a, και έτσι έχουμε το παρακάτω κύκλωμα που εκτελεί τις πράξεις AND, OR, και πρόσθεση των a και b, όπως και των $\bar{a}$ και $\bar{b}$.

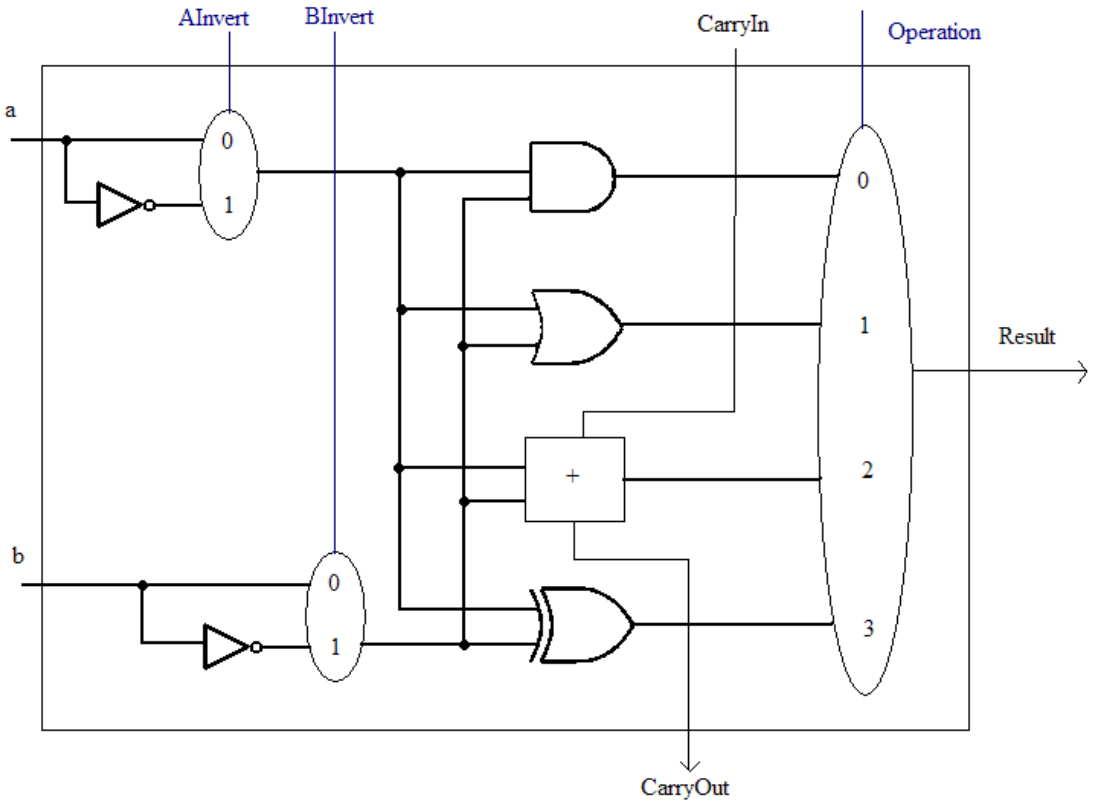
Επιλέγοντας το $\bar{a}$ (Ainvert= 1) και το $\bar{b}$ (Binvert = 1), παίρνουμε το **a NOR b** αντί για a AND b.



Εικόνα 6: ***Μια ALU του 1-bit που εκτελεί τις πράξεις AND, OR, πρόσθεση/αφαίρεση και NOR.***

3. Προσθέτοντας στην ALU μας την πράξη XOR

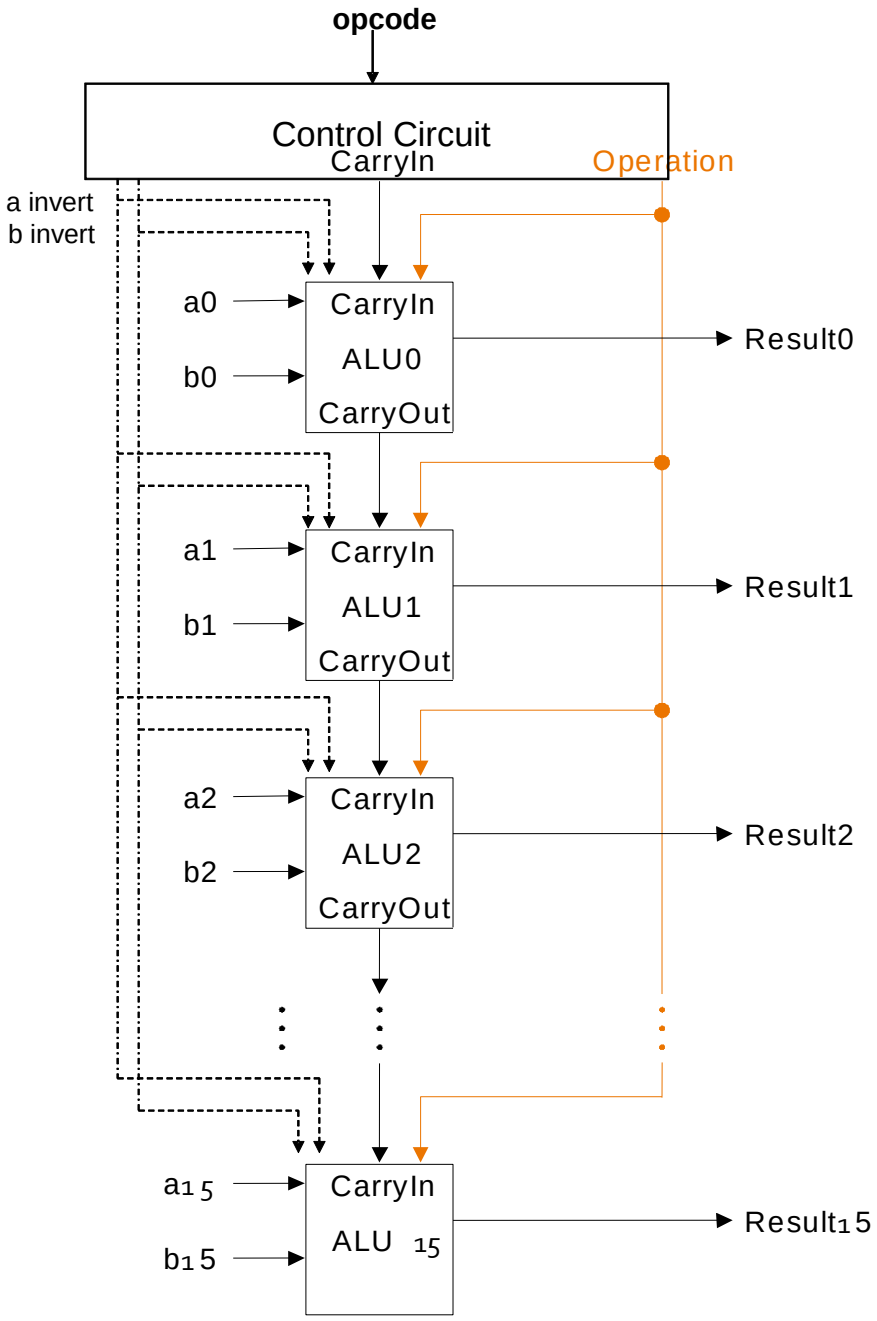
Τέλος προσθέτουμε μια επιπλέον πύλη XOR στην 1-bit ALU μας και επεκτείνουμε τον πολυπλέκτη που παράγει το τελικό αποτέλεσμα για την υλοποίηση και της **πράξη XOR** (Εικόνα 7).



Εικόνα 7. ***Η τελική μορφή της 1-bit ALU που υλοποιεί όλες τις απαιτούμενες πράξεις.***

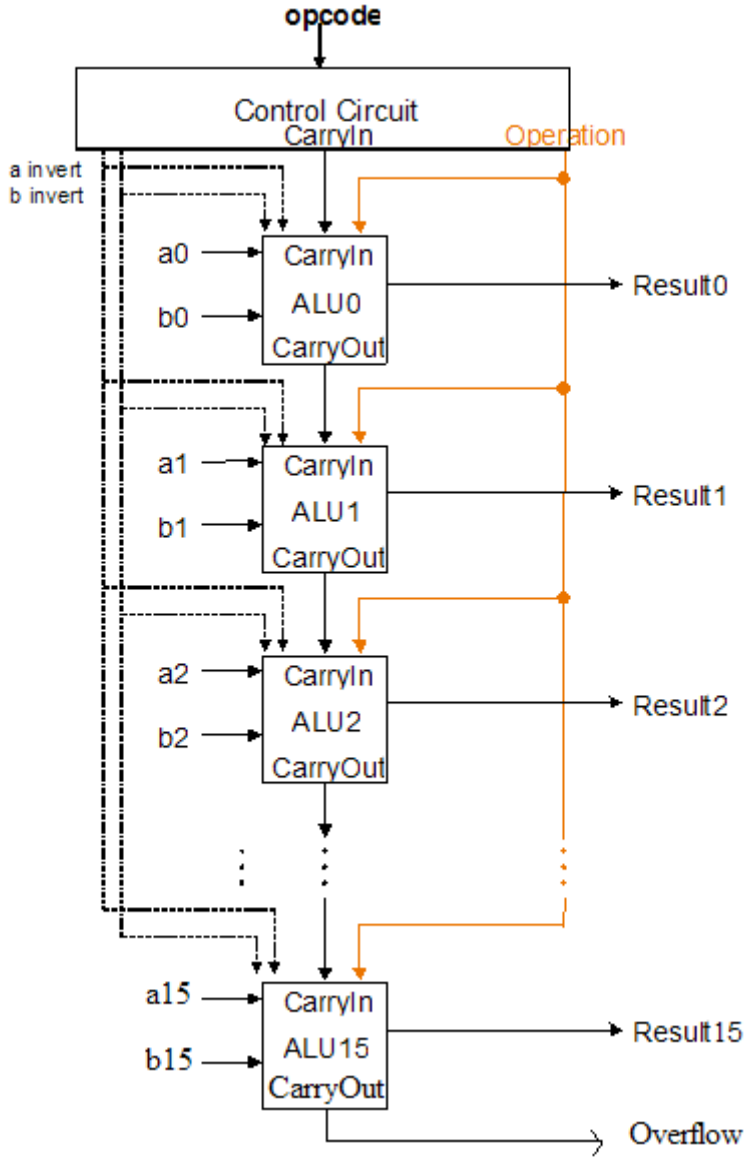
4. Υλοποίηση ιεραρχικής ALU των 16-bits με slices της 1-bit ALU

Στην **Εικόνα 8** φαίνεται μια ALU των 16-bits βασιζόμενη στη διαδοχική σύνδεση 16 slices της 1-bit ALU. Στην 16-bits ALU το CarryOut του κάθε slice γίνεται CarryIn για το επόμενο slice (π.χ. το CarryOut της ALU0 γίνεται CarryIn για την ALU1, το CarryOut της ALU1 γίνεται CarryIn για την ALU2, κ.ο.κ), δηλαδή για την πράξη της άθροισης/αφαίρεσης υλοποιείται *αθροιστής διάδοσης κρατουμένου (ripple carry adder)*. Το αποτέλεσμα της εκτελούμενης κάθε φορά πράξης εξάγεται στο Result0..Result15.



Εικόνα 8: ***Μία 16-bit ALU σε ιεραρχική σύνδεση με ripple carry***

Πρέπει όμως κάπως η ALU να ειδοποιήσει το σύστημα για πιθανή **υπερχείλιση**. Υπερχείλιση συμβαίνει όταν το αποτέλεσμα της πράξης που εκτελεί η ALU είναι μεγαλύτερο από τους τελεστέους (στην περίπτωσή μας μεγαλύτερο από 16 bits). Ένας εύκολος τρόπος να διαπιστώσουμε αν συνέβη υπερχείλιση είναι να ελέγξουμε αν παράγεται κρατούμενο στο τελευταίο slice της ALU (ALU15). Έτσι το σχήμα για την ALU με **έλεγχο για υπερχείλιση** είναι το εξής:



Εικόνα 9: ***Μία 16-bits ALU σε ιεραρχική σύνδεση με ripple carry που υποστηρίζει έλεγχο υπερχείλισης***

Το **κύκλωμα ελέγχου (Control Circuit)** ορίζει τις πράξεις που θα γίνουν από την ALU με βάση τους παρακάτω *λειτουργικούς κωδικούς (opcode)*. Ανάλογα με την πράξη που θέλουμε να κάνουμε το Control Unit δίνει τις ανάλογες τιμές στα Ainvert (1-bit), Binvert (1-bit), CarryIn (1-bit), Operation (2-bits), όπως φαίνεται παρακάτω:

OPCODE	ΠΡΑΞΗ	Operation	Ainvert	Binvert	CarryIn
000	AND	00	0	0	0
001	OR	01	0	0	0
011	ADD	11	0	0	0
010	SUB	11	0	1	1
101	NOR	00	1	1	0
100	XOR	10	0	0	0
110	*Αδιάφορες τιμές*				
111					

Πίνακας 1: ***Η λειτουργία του Control Unit***

Μπορείτε να περιγράψετε τη λειτουργία του Control Circuit με τη μορφή λογικών συναρτήσεων για τις διάφορες εξόδους του κυκλώματος.

Ζητούμενα

1. Χρησιμοποιήστε το λογισμικό Quartus II, για να γράψετε πηγαίο κώδικα (πρόγραμμα) στη γλώσσα VHDL, ο οποίος να υλοποιεί το κύκλωμα της **εικόνας 7** (1 slice της 1-bit ALU) με **structural τρόπο**. Χρησιμοποιήστε την προσομοίωση λειτουργίας του λογισμικού Quartus II για να αποδείξετε την ορθότητα της υλοποίησής σας για όλες τις πιθανές τιμές των a, b και τις πράξεις που υποστηρίζει (μπορείτε να φτιάξετε για λόγους ευκρίνειας μία κυματομορφή για κάθε πράξη).

2. Χρησιμοποιήστε το λογισμικό Quartus II, για να γράψετε πηγαίο κώδικα (πρόγραμμα) στη γλώσσα VHDL, ο οποίος να υλοποιεί το κύκλωμα της **εικόνας 9** (16-bit ALU) με structural τρόπο (***Υπόδειξη***: *χρησιμοποιήστε το κύκλωμα που υλοποιήσατε για το ζητούμενο 1 ως component*). Χρησιμοποιήστε την προσομοίωση λειτουργίας του λογισμικού Quartus II για να αποδείξετε την ορθότητα της υλοποίησής σας για τις τιμές των a, b που δίνονται στη συνέχεια και τις πράξεις που υποστηρίζει (μία κυματομορφή για κάθε πράξη).

   a =0000111100001111

   b =0101010101010101

Οδηγίες

Οι ομάδες θα απαρτίζονται από τα **ίδια άτομα (3-4)** όπως και στην πρώτη εργασία. Εκπρόθεσμες εργασίες **δε θα γίνουν δεκτές**.

Θα παραδώσετε μέσω του eClass ένα αρχείο zip/rar, που θα ονομάσετε με τους αρ. μητρώου των μελών της ομάδας (π.χ. 3170415_3170400_3170452.rar), το οποίο θα περιλαμβάνει:

- ένα **project** του Quartus για κάθε ένα από τα ζητούμενα 1 και 2.
- ένα **pdf αρχείο** το οποίο θα περιέχει τα στοιχεία (ονοματεπώνυμο, αριθμό μητρώου και email) των μελών της ομάδας και ό,τι άλλο θεωρείτε απαραίτητο σχετικά με την υλοποίηση των προβλημάτων. Συγκεκριμένα και για τα δύο ζητούμενα, το PDF θα πρέπει κατ' ελάχιστον να περιέχει:
  - τις κυματομορφές που προκύπτουν από τη λειτουργική προσομοίωση του κυκλώματος που υλοποιήσατε για όλες τις πράξεις που εκτελεί η ALU.
  - το αντίστοιχο RTL διάγραμμα.
- Προσοχή! Ο κώδικας VHDL θα πρέπει να συνοδεύεται από επεξηγηματικά σχόλια.