Πανεπιστήμιο Αιγαίου
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Σχεδίαση Ψηφιακών Συστημάτων
Εργασία μαθήματος

Μανόλης Καλλίγερος (kalliger@aegean.gr)

Διασύνδεση πληκτρολογίου με την πλακέτα DE2-115 και υλοποίηση ενός πολύ απλού calculator με πρόσθεση και αφαίρεση

Παραδοτέα

1) Το αρχείο με τον κώδικα Verilog που γράψατε ***πλήρως σχολιασμένο***.
2) **Αναφορά** στην οποία θα εξηγείτε τη δομή του κυκλώματος που σχεδιάσατε, το πώς υλοποιούνται τα διάφορα ζητούμενα της εργασίας καθώς και όποιο άλλο σημείο της υλοποίησής σας κρίνετε ότι πρέπει να σχολιαστεί.

Ημερομηνία παράδοσης: Κυριακή 11/2/2018
Τρόπος παράδοσης: Ηλεκτρονικός (στις ***Εργασίες*** του eClass)

Μέρος πρώτο - Διασύνδεση του πληκτρολογίου

Να διασυνδέσετε ένα πληκτρολόγιο PS/2 (PS/2 keyboard) με την πλακέτα DE2-115. Το κύκλωμα διασύνδεσης που καλείστε να σχεδιάσετε θα πρέπει να αναγνωρίζει το πάτημα ενός πλήκτρου με αριθμό (από το 0 έως το 9) και να εμφανίζει τον αντίστοιχο αριθμό σε ένα από τα 7-segment display της πλακέτας (π.χ. στο HEX7). Σημειώνεται ότι μπορείτε να αναγνωρίζετε τα πλήκτρα που βρίσκονται είτε στο βασικό τμήμα του πληκτρολογίου, είτε στο numeric keypad (ή και στα δύο αν επιθυμείτε).

Υποδείξεις

1. Για λεπτομέρειες σχετικά με τον τρόπο που επικοινωνεί το πληκτρολόγιο με κάποια host συσκευή μπορείτε να ανατρέξετε στα ακόλουθα link:
   http://retired.beyondlogic.org/keyboard/keybrd.htm
   http://www.computer-engineering.org/ps2keyboard/.
2. Το κύκλωμα που πρέπει να υλοποιήσετε θα έχει τέσσερις εισόδους (μία εκ των οποίων θα είναι και έξοδος - **inout**) και 7 εξόδους. Οι είσοδοι θα είναι οι: clk (εσωτερικό ρολόι του κυκλώματος), reset (σήμα reset), keyb_clk (το ρολόι του πληκτρολογίου με το οποίο συγχρονίζονται τα δεδομένα που στέλνονται από αυτό) και keyb_data (η γραμμή μεταφοράς των δεδομένων από το πληκτρολόγιο). Η γραμμή keyb_clk, εκτός από είσοδος, θα είναι και έξοδος (**inout**). Οι έξοδοι του κυκλώματος θα είναι οι επτά γραμμές που θα

οδηγούν τα επτά LED του 7-segment display. Σημειώνεται ότι στο module που θα γράψετε μπορείτε να χρησιμοποιήσετε διαφορετικά ονόματα για τις γραμμές εισόδου/ εξόδου από αυτά που αναφέρθηκαν παραπάνω.

3. Για να λάβετε ένα scan κωδικό από το πληκτρολόγιο (ένα πακέτο μήκους 11 bit δηλαδή), θα χρειαστείτε δύο shift registers. Ο ένας θα χρησιμοποιείται για τη δειγματοληψία του keyb_clk και ο άλλος για την αποθήκευση των δεδομένων που έρχονται από τη γραμμή keyb_data. Ο πρώτος (δειγματοληψία του keyb_clk) θα αρχικοποιείται με το reset στην τιμή 000...0, ενώ ο δεύτερος (αποθήκευση δεδομένων από το keyb_data) στην τιμή 111...1. Στον register δειγματοληψίας του keyb_clk θα αποθηκεύεται ένα νέο δείγμα σε κάθε κύκλο του εσωτερικού ρολογιού. Για την αποφυγή της ανίχνευσης θορύβου στη γραμμή keyb_clk σαν ακμή ρολογιού, ο register αυτός πρέπει να έχει μήκος μεγαλύτερο του 2. Για τις ανάγκες της εργασίας μπορείτε να θεωρήσετε ότι το μήκος του είναι 6. Όταν τα τρία πιο πρόσφατα δείγματα που έχουν αποθηκευθεί σε αυτόν είναι 0 ενώ τα τρία παλαιότερα είναι 1 (δηλ. όταν η τιμή του καταχωρητή είναι 000111, υποθέτοντας ολίσθηση των δειγμάτων προς τα δεξιά), τότε μία αρνητική ακμή έχει ανιχνευθεί στη γραμμή keyb_clk. Εκείνη τη χρονική στιγμή μπορούμε πλέον να δειγματοληπτήσουμε τη γραμμή keyb_data και να αποθηκεύσουμε την τιμή της στον αντίστοιχο shift register. Η λήψη ενός ολόκληρου πακέτου μήκους 11 bit από το πληκτρολόγιο ολοκληρώνεται όταν η τιμή του τελευταίου (δεξιότερου) bit του shift register αποθήκευσης δεδομένων γίνει 0.
4. Η αλλαγή της τιμής της τιμής του 11ου bit του register δεδομένων από 1 σε 0 «πυροδοτεί» τα ακόλουθα γεγονότα: α) κλείδωμα της **αποκωδικοποιημένης** τιμής του ληφθέντος scan κωδικού σε έναν register (μήκους έξι bit), και β) επαναρχικοποίηση του shift register δεδομένων στην κατάσταση 111...1 (ώστε να είναι έτοιμος για τη λήψη του επόμενου πακέτου δεδομένων από το πληκτρολόγιο). Ο shift register δειγματοληψίας του keyb_clk δεν είναι απαραίτητο να επαναρχικοποιηθεί.
5. Για την ολοκλήρωση του πρώτου μέρους της άσκησης πρέπει ακόμα να υλοποιήσετε δύο συνδυαστικά κυκλώματα αποκωδικοποίησης. Το πρώτο θα παίρνει σαν είσοδο τα 8 bit του scan κωδικού που περιλαμβάνονται σε κάθε πακέτο των 11-bit που λαμβάνεται από το πληκτρολόγιο και θα παράγει την (6-bit) αναπαράσταση του αντίστοιχου δυαδικού αριθμού. Όταν πατηθεί πλήκτρο που δεν αντιστοιχεί σε αριθμό, τότε το συγκεκριμένο κύκλωμα αποκωδικοποίησης θα επιστρέφει μία (6-bit) τιμή που θα υποδηλώνει ότι ο scan κωδικός που ελήφθη δεν αντιστοιχεί σε αριθμό. Το δεύτερο κύκλωμα αποκωδικοποίησης θα διαβάζει τον register στον οποίο αποθηκεύεται η τιμή του πρώτου κυκλώματος (βλ. υπόδειξη 4 παραπάνω) και θα οδηγεί το 7-segment display. Σημειώνεται ότι η διεπίπεδη αυτή αποκωδικοποίηση δεν είναι απαραίτητη για το πρώτο μέρος της εργασίας αλλά θα σας χρειαστεί στο δεύτερο.
6. Ο Verilog κώδικας για την **οδήγηση** του inout pin keyb_clk είναι ο εξής:

   assign keyb_clk = (!keyb_data_reg[0]) ? 1'b0 : 1'bz;

   όπου keyb_data_reg είναι ο shift register αποθήκευσης των δεδομένων από το πληκτρολόγιο, ενώ το keyb_data_reg[0] είναι το τελευταίο (δεξιότερο) bit του (υποθέτουμε και πάλι ολίσθηση προς τα δεξιά). Ο κώδικας αυτός οδηγεί τη γραμμή keyb_clk στο 0 όσο

το δεξιότερο bit του register είναι 0, ώστε το πληκτρολόγιο να μην μπορεί να στείλει δεδομένα πριν επαναρχικοποιηθεί ο register στην τιμή 111...1.

Μέρος δεύτερο - Ενσωμάτωση των βασικών συνδυαστικών κυκλωμάτων και καταχωρητών

Για να γίνει κατανοητό τι είδους συνδυαστικά κυκλώματα και καταχωρητές χρειάζονται για την υλοποίηση της εργασίας, θα πρέπει να εξηγηθεί αναλυτικά η λειτουργία του τελικού κυκλώματος. Σημειώνεται ότι για την αναπαράσταση των εισόδων και του αποτελέσματος θα χρησιμοποιηθούν συνολικά **έξι** (6) 7-segment display της πλακέτας.

Η λειτουργία του κυκλώματος λοιπόν θα είναι η ακόλουθη: αρχικά όλα τα display θα είναι σβηστά και θα αναμένεται το πάτημα ενός πλήκτρου με αριθμό στο πληκτρολόγιο. Όταν αυτό συμβεί ο αντίστοιχος αριθμός θα εμφανίζεται στο **πρώτο** 7-segment display. Ακολούθως θα αναμένεται το πάτημα του '+' ή του '–' από το numeric keypad και μετά το πάτημα ενός από τα πλήκτρα αυτά, θα εμφανίζεται στο **δεύτερο** 7-segment display ένα σύμβολο που θα αντιστοιχεί σε καθεμία από τις προαναφερθείσες πράξεις (π.χ. για την αφαίρεση μπορεί να ανάβει το μεσαίο οριζόντιο LED του display). Στη συνέχεια θα πρέπει να πατηθεί για δεύτερη φορά ένα πλήκτρο με αριθμό (το δεύτερο τελούμενο της πράξης), ο οποίος θα εμφανίζεται στο **τρίτο** 7-segment display, και στο τέλος θα πρέπει να πατηθεί από το χρήστη το πλήκτρο με το '=' ώστε να εμφανιστεί το τελικό αποτέλεσμα (στα δύο τελευταία display αν είναι διψήφιο ή αρνητικό, ή μόνο στο τελευταίο αν είναι μονοψήφιο). Τα δυνατά αποτελέσματα που μπορεί να προκύψουν είναι όλοι οι ακέραιοι αριθμοί στο διάστημα [-9, 18]. Σημειώνεται ότι στο **τέταρτο** display εμφανίζεται το '=', μετά φυσικά από το πάτημα του σχετικού πλήκτρου.

Μετά τον υπολογισμό ενός αθροίσματος ή μιας διαφοράς, το κύκλωμα θα περιμένει από το χρήστη την εισαγωγή του πρώτου τελούμενου της επόμενης πράξης (κρατώντας στα 7-segment display την προηγούμενη). Όταν αυτό συμβεί, θα πρέπει να **σβήνουν** όλα τα display εκτός φυσικά από το πρώτο, στο οποίο θα εμφανίζεται το τελούμενο της νέας πράξης. Σε περίπτωση πατήματος λάθος πλήκτρου (π.χ. του '–' αμέσως μετά το '+' ή κάποιου χαρακτήρα του πληκτρολογίου) θα πρέπει να εμφανίζεται μία ένδειξη λάθους στο αντίστοιχο display (π.χ. το γράμμα 'e') έως ότου πατηθεί ένα σωστό πλήκτρο.

Με βάση τα παραπάνω προαπαιτούμενα θα πρέπει να υλοποιηθούν τα ακόλουθα:

1. Έξι καταχωρητές εύρους 7 bit ο καθένας, οι οποίοι θα οδηγούν τα 7-segment display. Η διαχείριση των καταχωρητών αυτών θα γίνεται από το state machine που θα ελέγχει τη λειτουργία όλου του κυκλώματος και θα υλοποιηθεί στο τρίτο μέρος της εργασίας. Σημειώνεται ότι, προφανώς, στο συνολικό κύκλωμα θα χρειαστούν και άλλοι καταχωρητές, οι οποίοι θα αναλυθούν παρακάτω.
2. Ο αποκωδικοποιητής πρώτου επιπέδου (αυτός που χρησιμοποιείται για την αποκωδικοποίηση των scan κωδικών του πληκτρολογίου) θα πρέπει να υλοποιηθεί αποκλειστικά σαν συνδυαστικό κύκλωμα και **όχι μαζί** με τον καταχωρητή που αναφέρεται στο πρώτο μέρος. Ο λόγος είναι ότι η έξοδός του στο τελικό κύκλωμα θα οδηγείται σε περισσότερα του ενός διαφορετικά σημεία (σε καταχωρητές αλλά και σε συνδυαστικό κύκλωμα).

3. Ο αποκωδικοποιητής δευτέρου επιπέδου θα πρέπει να επεκταθεί ώστε να αποκωδικοποιεί όλες τις πιθανές αριθμητικές τιμές που μπορεί να προκύψουν στο κύκλωμα ([-9, 18]), καθώς και ένα συνδυασμό εισόδου που αντιστοιχεί σε σφάλμα (για την εμφάνιση του χαρακτήρα 'e' σε κάποιο 7-segment display).
4. Το κύκλωμα πρόσθεσης/αφαίρεσης μπορεί να υλοποιηθεί με όποιον τρόπο επιθυμείτε, με προτιμότερο αυτόν της dataflow μοντελοποίησης (δηλαδή με assign). Υπενθυμίζεται ότι για την εκτέλεση της αφαίρεσης δύο αριθμών A, B από ένα κύκλωμα άθροισης θα πρέπει να προστεθεί στον A το συμπλήρωμα ως προς 2 του B (= συμπλήρωμα ως προς 1 του B + 1). Τα A, B θα πρέπει να είναι δύο 6-bit καταχωρητές, στους οποίους θα αποθηκεύονται τα τελούμενα της πράξης από το state machine. Θα σας χρειαστεί επίσης και ένας καταχωρητής του 1 bit, του οποίου η τιμή θα καθορίζει την πράξη (πρόσθεση ή αφαίρεση).
5. Τέλος, θα χρειαστεί να υλοποιήσετε και έναν πολυπλέκτη 2-σε-1 ποσοτήτων εύρους 6-bit, ο οποίος θα οδηγεί στον αποκωδικοποιητή δεύτερου επιπέδου την έξοδο του αποκωδικοποιητή πρώτου επιπέδου ή του αθροιστή (για την αποκωδικοποίηση των τελουμένων ή του αποτελέσματος αντίστοιχα). Η γραμμή επιλογής του συγκεκριμένου πολυπλέκτη θα εξαρτάται από την κατάσταση του state machine. Για το λόγο αυτό μπορεί να υλοποιηθεί και στο τρίτο μέρος της εργασίας. Και πάλι το επιθυμητό επίπεδο μοντελοποίησης για τον πολυπλέκτη είναι το dataflow.

Μέρος τρίτο - υλοποίηση του state machine

Στο τρίτο και τελευταίο μέρος της εργασίας καλείστε να υλοποιήσετε το finite state machine (FSM) –(ακολουθιακό) κύκλωμα πεπερασμένων καταστάσεων– που θα ελέγχει τόσο την ακολουθία των scan κωδικών που αποστέλλονται από το πληκτρολόγιο, όσο και τη συνολική λειτουργία του συστήματος. Οι δύο αυτές απαιτήσεις μπορούν εύκολα να ενσωματωθούν στο ίδιο state machine. Στη συνέχεια θα δοθούν οδηγίες σχετικά με την υλοποίηση ενός τέτοιου FSM. Σημειώνεται ότι οι οδηγίες αυτές είναι εντελώς ενδεικτικές και ότι οποιαδήποτε σωστή υλοποίηση είναι αποδεκτή, άσχετα με το αν ακολουθεί τις συγκεκριμένες οδηγίες.

Καταστάσεις του FSM

Το FSM θα χρειαστεί ένα πλήθος καταστάσεων για τον έλεγχο της **λειτουργίας του calculator**. Αυτές θα μπορούσαν να είναι οι εξής:

- *Αναμονή για την είσοδο του πρώτου τελούμενου από το πληκτρολόγιο (WAIT_OPERAND1).*
- *Αναμονή για το πάτημα του τελεστή της πράξης ('+' ή '–') στο πληκτρολόγιο (WAIT_OPERATOR).*
- *Αναμονή για την είσοδο του δεύτερου τελούμενου από το πληκτρολόγιο (WAIT_OPERAND2).*
- *Αναμονή για το πάτημα του συμβόλου της ισότητας ('=') στο πληκτρολόγιο (WAIT_EQ).*

Οι παραπάνω καταστάσεις θα εναλλάσσονται με το πάτημα των πλήκτρων. Επειδή το πάτημα ενός πλήκτρου σε ένα πληκτρολόγιο PS/2 αντιστοιχεί στην αποστολή πολλαπλών scan κωδικών (τουλάχιστον τριών), θα χρειαστούν κάποιες επιπλέον καταστάσεις για τον έλεγχο κάθε πατήματος. Αν παρατηρήσετε προσεκτικά την αλληλουχία των scan κωδικών που

αποστέλλονται για κάθε πλήκτρο θα διαπιστώσετε ότι πάντα ο προτελευταίος κωδικός είναι ο F0, ακολουθούμενος από έναν ακόμα, ο οποίος, τις περισσότερες φορές, είναι ο κωδικός του πλήκτρου που πατήθηκε. Αφού είναι επιθυμητό κάθε πλήκτρο να γίνεται αντιληπτό από το πρώτο πάτημά του, οι επιπλέον καταστάσεις του FSM θα ελέγχουν **την απελευθέρωση** («άφημα») **του πλήκτρου**. Συγκεκριμένα, θα είναι οι εξής:

- *Κατάσταση αναμονής του κωδικού F0 (WAIT_F0).*
- *Κατάσταση αναμονής του κωδικού που ακολουθεί τον F0 (AFTER_F0).*

Άρα, μετά από καθεμία από τις καταστάσεις WAIT_OPERAND1, WAIT_OPERATOR, WAIT_OPERAND2 και WAIT_EQ, οι οποίες θα ανιχνεύουν το πρώτο πάτημα ενός πλήκτρου και θα πραγματοποιούν τις κατάλληλες ενέργειες, θα ακολουθούν **πάντα** οι καταστάσεις WAIT_F0 και AFTER_F0. ***Στην κατάσταση AFTER_F0 θα αποφασίζεται σε ποια από τις τέσσερις καταστάσεις λειτουργίας του calculator (WAIT_OPERAND1, WAIT_OPERATOR, WAIT_OPERAND2 και WAIT_EQ) θα μεταβεί το FSM μετά την απελευθέρωση του τρέχοντος πλήκτρου***. Για παράδειγμα, μία πιθανή αλληλουχία καταστάσεων είναι η: WAIT_OPERAND1 → WAIT_F0 → AFTER_F0 → WAIT_OPERATOR → WAIT_F0 → AFTER_F0 → WAIT_OPERAND2 → ...

Σημειώνεται ότι εκτός από τις προαναφερθείσες 6 καταστάσεις, στην εντολή case που θα χρησιμοποιηθεί για την υλοποίηση του FSM, θα πρέπει να συμπεριληφθεί και μία περίπτωση default, στην οποία θα αρχικοποιούνται όλοι οι καταχωρητές που διαχειρίζεται το FSM με τον ίδιο ακριβώς τρόπο που αρχικοποιούνται και στην περίπτωση που ενεργοποιηθεί η είσοδος reset του συστήματος.

Καταχωρητές του FSM και καταχωρητές που διαχειρίζεται το FSM

Οι καταχωρητές που αφορούν τη λειτουργία του FSM είναι οι εξής:

- *Ένας καταχωρητής για την αποθήκευση της κατάστασης του FSM.*
- *Ένας καταχωρητής για την αποθήκευση της τρέχουσας* ***κατάστασης λειτουργίας του calculator*** *(μία εκ των WAIT_OPERAND1, WAIT_OPERATOR, WAIT_OPERAND2 και WAIT_EQ).* ***Η τιμή του καταχωρητή αυτού θα είναι χρήσιμη μετά την απελευθέρωση ενός πλήκτρου (στην κατάσταση AFTER_F0), για τη μετάβαση στην επόμενη κατάσταση λειτουργίας του calculator ή στην ίδια σε περίπτωση που πατήθηκε λάθος πλήκτρο.***

Σημαντικοί για τη λειτουργία του συστήματος (και ελεγχόμενοι από το FSM) είναι, επίσης, οι ακόλουθοι καταχωρητές:

- *Δύο καταχωρητές για την αποθήκευση των τελούμενων της πράξης.*
- *Ένας καταχωρητής* αποθήκευσης του είδους της πράξης (πρόσθεση ή αφαίρεση) του ενός bit.
- *Ένας καταχωρητής του ενός bit, του οποίου η τιμή θα δηλώνει αν έχει πατηθεί λάθος πλήκτρο ή όχι.*

Τέλος, το FSM θα διαχειρίζεται και τους καταχωρήτες που οδηγούν τα 7-segment display, οι οποίοι αναφέρθηκαν στο προηγούμενο μέρος της εκφώνησης.

Υποδείξεις σχετικά με τη λειτουργία του FSM

- Το FSM θα πρέπει να λειτουργεί κάθε φορά που έχει ληφθεί ένας **πλήρης** scan κωδικός από το πληκτρολόγιο (δηλαδή κάθε φορά που **γεμίζει** ο keyb_data_reg).
- Σε κάθε κατάσταση θα πρέπει να ανατίθενται κατάλληλα οι σχετικοί καταχωρητές. Για παράδειγμα, στην κατάσταση WAIT_OPERAND1 θα πρέπει να γίνονται τα εξής:
    - Οι καταχωρητές όλων των 7-segment display, πλην του πρώτου, θα πρέπει να τεθούν έτσι ώστε να έχουν σε όλα τους τα bit την τιμή 1. Με αυτόν τον τρόπο θα σβήσουν όλα τα LED των display.
    - Ο καταχωρητής του πρώτου display θα πρέπει να τεθεί ίσος με την έξοδο του αποκωδικοποιητή **δευτέρου** επιπέδου.
    - Ο καταχωρητής του πρώτου τελούμενου θα πρέπει να τεθεί ίσος με την έξοδο του αποκωδικοποιητή **πρώτου** επιπέδου.
    - Θα πρέπει να ελεγχθεί αν έχει πατηθεί λάθος πλήκτρο και, αν ναι, να τεθεί κατάλληλα ο καταχωρητής λάθους. *Ο έλεγχος για το πάτημα λάθος πλήκτρου μπορεί να γίνει πολύ εύκολα στην έξοδο του αποκωδικοποιητή πρώτου επιπέδου, αν, σε περίπτωση μη πατήματος πλήκτρου με αριθμό, επιστρέφεται από τον αποκωδικοποιητή (σαν κωδικός λάθους) μία **αρνητική** τιμή που δεν πρόκειται να εμφανιστεί στο ζητούμενο σύστημα (π.χ. η τιμή 100000 = -32). Κατά αυτόν τον τρόπο θα μπορούσε να ελεγχθεί μόνο το **περισσότερο σημαντικό ψηφίο** της εξόδου του αποκωδικοποιητή ώστε να διαπιστωθεί τυχόν πάτημα λάθος πλήκτρου.*
    - Ο καταχωρητής τρέχουσας κατάστασης λειτουργίας του calculator πρέπει να τεθεί ίσος με WAIT_OPERAND1.
    - Ο καταχωρητής κατάστασης του FSM θα πρέπει να τεθεί ίσος με WAIT_F0 (έτσι γίνεται η μετάβαση στην επόμενη κατάσταση του FSM).
- Στις καταστάσεις που χρειάζεται έλεγχος για το πάτημα ενός πλήκτρου, για το οποίο δεν επιστρέφεται τιμή από τον αποκωδικοποιητή πρώτου επιπέδου (πλήκτρα '+', '–' ή '='), μπορεί να ελεγχθεί απευθείας ο κωδικός που είναι αποθηκευμένος στον keyb_data_reg.
- Οι καταχωρητές που οδηγούν τα display, στα οποία εμφανίζονται τα σύμβολα της πρόσθεσης/αφαίρεσης και της ισότητας και για τα οποία δεν παράγει έξοδο ο αποκωδικοποιητής δευτέρου επιπέδου, μπορούν να ανατεθούν κατάλληλα στις σχετικές καταστάσεις του FSM.
- Ο πολυπλέκτης που οδηγεί την είσοδο του αποκωδικοποιητή δευτέρου επιπέδου, θα περνάει σε αυτή την έξοδο του κυκλώματος πρόσθεσης/αφαίρεσης όταν η κατάσταση του FSM είναι η WAIT_EQ, και την έξοδο του αποκωδικοποιητή πρώτου επιπέδου σε οποιαδήποτε άλλη περίπτωση.