ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:
ΗΡΥ 203 - ΠΡΟΧΩΡΗΜΕΝΗ ΛΟΓΙΚΗ ΣΧΕΔΙΑΣΗ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2016

Εργαστήριο 4

ΔΗΜΙΟΥΡΓΙΑ PRE-INCREMENT/POST-DECREMENT ΣΤΟΙΒΑΣ – FSM ΕΛΕΓΧΟΥ

ΕΚΠΟΝΗΣΗ : Καθηγ. Α. Δόλλας,
Δρ. Κ. Παπαδημητρίου

ΕΔΙΠ: Μ. Κιμιωνής
Κ. Παπαδημητρίου

ΒΟΗΘΟΙ: Π. Μαλακωνάκης
Ι. Γαλανομμάτης
Κ. Καλαϊτζής
Μ. Κουσανάκης

ΕΚΔΟΣΗ : 9.0 (Εαρινό εξάμηνο 2016)

Χανιά 2016

Σκοπός του Εργαστηρίου

Είναι η σχεδίαση κυκλώματος που υλοποιεί πλήρως τη λειτουργικότητα στοίβας pre-increment/post-decrement. Εκτός των λειτουργιών **push/pop (εισαγωγή/εξαγωγή),** το κύκλωμα εκτελεί **πρόσθεση** μεταξύ του στοιχείου που βρίσκεται στο TOS(Top Of Stack) και του προηγούμενου στοιχείου (TOS-1). Έγκυρο αποτέλεσμα πρόσθεσης εγγράφεται στη στοίβα, ενώ αν υπάρξει overflow στο αποτέλεσμα, δε γίνεται εγγραφή. Στις εξόδους LED θα πρέπει να φαίνεται η τελευταία τιμή που εγγράφηκε στη στοίβα (ερώτηση: είναι ίδια ή διαφορετική από τη θέση που δείχνει ο δείκτης Stack Pointer??), ενώ αν η στοίβα είναι άδεια θα φαίνεται το "0".

Περιγραφή κυκλώματος και λειτουργίας

Αποτελείται από 4 βασικά υποκυκλώματα:

- Μια στοίβα υλοποιημένη με μνήμη πλάτους 4-bit, που αποθηκεύει μέχρι και 31 στοιχεία. Η μνήμη που θα δημιουργήσετε θα είναι 32 θέσεων. Η στοίβα θα έχει όλα τα σήματα εξόδου που είδατε στη θεωρία, δηλ. empty, almost-empty, full, almost-full.
- Ο stack pointer είναι ο δείκτης της στοίβας και θα τον υλοποιήσετε ως μετρητή 5-bits που μπορεί να μετράει άνω/κάτω. Η σχεδίαση του θα είναι structural.
- Μια FSM που ανάλογα με τις τιμές τριών εξωτερικών εισόδων (δίνονται με Push Button) θα προκαλεί μια από τις εξής 3 λειτουργίες: push, pop, add. Υπάρχουν όμως περιπτώσεις όπου οι **λειτουργίες αυτές δεν μπορούν να πραγματοποιηθούν,** π.χ. αν συμβαίνει stack full δεν μπορεί να γίνει push, και αν συμβαίνει stack empty δεν μπορεί να γίνει pop/add. Επιπλέον, η FSM διαχειρίζεται το σύστημα σε περίπτωση υπερχείλισης της στοίβας (StackOvf), καθώς και σε περίπτωση υπερχείλισης του αποτελέσματος της πρόσθεσης (AdditionOvf). Σε αυτές τις περιπτώσεις, πρέπει να οδηγούνται αντίστοιχα τα LEDs (περισσότερες λεπτομέρειες παρακάτω).
- Αθροιστής 4-bit. Η σχεδίαση του θα είναι structural.

Να προσέξετε την επικοινωνία/αλληλεπίδραση μεταξύ της κατάστασης της στοίβας και της FSM. Ποιες είναι οι έξοδοι από τη στοίβα προς την FSM, και ποιες από την FSM προς τη στοίβα? Με πόσες καταστάσεις υλοποιήσατε την FSM? Η FSM είναι Mealy ή Moore? Μέσω της επικοινωνίας των παραπάνω υποκυκλωμάτων υποστηρίζονται οι παρακάτω λειτουργίες:

1. **Push:** εισαγωγή ενός αριθμού στη στοίβα.

2. **Pop:** εξαγωγή ενός αριθμού από τη στοίβα.
3. **Add:** πρόσθεση του αριθμού που βρίσκεται στο Top Of Stack (TOS) με τον αμέσως προηγούμενο της στοίβας, και αμέσως μετά push του αποτελέσματος στη στοίβα. Αν η στοίβα είναι άδεια (stack empty) δεν εκτελείται πρόσθεση, και αν υπάρχει μόνο ένα στοιχείο Α μέσα στη στοίβα (stack almost-empty) τότε εκτελείται η πράξη A+0=A.
4. **StackOvf:** όταν η στοίβα υπερβεί το μέγιστο αριθμό στοιχείων μετά από διαδοχικά push, το σύστημα σταματάει την κανονική λειτουργία του και δεν ανταποκρίνεται στα σήματα εισόδου, εκτός του reset.
5. **AdditionOvf:** Αν το αποτέλεσμα της πρόσθεσης υπερβεί τον αριθμό 15_{10}, δεν εγγράφεται το αποτέλεσμα στη στοίβα (δε γίνεται push). Το σύστημα θα μπορεί να συνεχίσει κανονικά η λειτουργία του.

Εξωτερικό I/O κυκλώματος & διεπαφή με το Basys2

Όνομα	Είσοδος/Εξοδος	Πλάτος σε bit	Αντιστοίχηση στο Board
Push	in	1	BTN0
Pop	in	1	BTN1
Add	in	1	BTN2
Reset	in	1	BTN3
Clock	in	1	MCLK
NumIn	in[4:0]	4	SW[3:0]
NumOut	out[3:0]	4	LED[3:0]
AdditionOvf	out	1	LED4
Empty	out[2:0]	3	LED[7:5]
AlmostEmpty	out[2:0]	3	LED[7:5]
Full	out[2:0]	3	LED[7:5]
AlmostFull	out[2:0]	3	LED[7:5]
StackOvf	out[2:0]	3	LED[7:5]

Πίνακας 1: Είσοδοι-έξοδοι του κυκλώματος

Τα LEDs 4-7 φωτοβολούν στις εξής περιπτώσεις:

1. Όταν η στοίβα αδειάσει (**Empty)** ή αρχικοποιηθεί με μηδενικά
2. Όταν η στοίβα γεμίσει (**Full)**
3. Όταν υπάρχει ένα (1) στοιχείο (**AlmostEmpty**) στη στοίβα
4. Όταν υπάρχουν τριάντα (30) στοιχεία (**AlmostFull**) στη στοίβα
5. Όταν η στοίβα υπερβεί το μέγιστο αριθμό (31) στοιχείων (**StackOvf**)
6. Όταν το αποτέλεσμα της πρόσθεσης δύο δυαδικών αριθμών υπερβεί τον αριθμό 15_{10} (**AdditionOvf)**

Για τις περιπτώσεις (1)-(5) το αποτέλεσμα θα φαίνεται στα LED[7:5] με απλή κωδικοποίηση, ενώ για την περίπτωση 6 χρησιμοποιείται το LED4 μόνο.

		LED7	LED6	LED5	LED4
1	Empty	**0**	**0**	**0**	
2	AlmostEmpty	**0**	**0**	**1**	
3	Full	**1**	**1**	**1**	
4	AlmostFull	**1**	**0**	**0**	
5	StackOvf	**1**	**0**	**1**	
6	AdditionOvf				**1**

Περιγραφή λειτουργίας σύμφωνα με τα σήματα εισόδου

Περιγράφεται παρακάτω η αντίδραση του κυκλώματος ανάλογα με την τιμή των σημάτων εισόδου. Υψηλότερη προτεραιότητα έχει το σήμα reset (ασύγχρονο).

Switches [3:0] (NumIn)
Σχηματίζουμε τον αριθμό που θα δοθεί για εγγραφή στη στοίβα, π.χ. αν SW3=up, SW2=down, SW1=down, SW0=up, αυτό αντιστοιχεί στην τιμή "1001".

Πάτημα BTN0 (push)
Αν πατηθεί μια φορά γίνεται εισαγωγή του αριθμού NumIn που έχει σχηματιστεί στα Switches [3:0], δεδομένου ότι η στοίβα δεν είναι γεμάτη.

Πάτημα BTN1 (pop)
Αν πατηθεί μια φορά γίνεται εξαγωγή του αριθμού που βρίσκεται στην κορυφή της στοίβας, δεδομένου ότι η στοίβα δεν είναι κενή.

Πάτημα BTN2 (add)
Αν πατηθεί μια φορά εκτελείται συνδυασμός των λειτουργιών pop, push και add. Συγκεκριμένα: θα γίνουν διαδοχικά 2 pop, 1 add, και αν το αποτέλεσμα είναι έγκυρο, θα γίνει 1 push του αποτελέσματος στη στοίβα. Επίσης, αν η στοίβα είναι άδεια (stack empty) δεν εκτελείται πρόσθεση, και αν υπάρχει ακριβώς ένα στοιχείο στη στοίβα A (stack almost-empty), τότε εκτελείται η πράξη A+0=A.

Πάτημα BTN3(reset)
Για όση ώρα κρατάμε πατημένο το reset (ενεργό στο '1'), το κύκλωμα οδηγείται σε μια αρχική κατάσταση.

Οδηγίες σχεδίασης που πρέπει να ακολουθήσετε

Κάθε υποκύκλωμα μπορεί να αποτελείται από ξεχωριστά μικρότερα υποκυκλώματα. Π.χ ο 5-bit μετρητής που θα σχεδιάσετε **πρέπει να αποτελείται από 5 flip-flops**, καθένα έχει είσοδο clock, reset, Din, ld, en, updown. Δείτε το Κεφ 3 των σημειώσεων. Η έξοδος Q κάθε flip-flop, θα συνδεθεί στην είσοδο του επόμενου για να δημιουργηθεί ο μετρητής 5-bit, με σχεδίαση structural, όχι behavioral. Τη σύνδεση μεταξύ των flip-flops, θα την κάνετε με instances (component και port map) σε ένα module που θα ονομάσετε counter. **Αυτό το έχετε κάνει ήδη στο εργαστήριο #3.** Για τον adder χρησιμοποιήστε πάλι structural σχεδίαση, τον carry-look-ahead (CLA) adder, **από το εργαστήριο #2.** Η FSM θα υλοποιηθεί ως ξεχωριστό module. Η στοίβα θα υλοποιηθεί ως μνήμη 32 θέσεων, πλάτους 4-bit με το εργαλείο Xilinx Core Generator.
Τα παραπάνω υποκυκλώματα και ότι άλλο υποκύκλωμα χρειαστεί να σχεδιάσετε, θα τα συνδέσετε στο topLevel. Τα κουμπιά push, pop και add θα συνδεθούν το καθένα με ένα ξεχωριστό debouncebutton.

Για την προσομοίωση, θα "καλέσετε" μόνο το toplevel ως instance στο testbench.

Προετοιμασία

Πριν την προσέλευση σας στο εργαστήριο:

α) Θα έχετε σχεδιάσει το block diagram του κυκλώματος (και της στοίβας). Στο block diagram να φαίνονται αναλυτικά οι είσοδοι/έξοδοι και εσωτερικές συνδέσεις μεταξύ των υποκυκλωμάτων.

β) Διάγραμμα της FSM στο οποίο θα φαίνονται οι είσοδοι/έξοδοι, αναλυτικά οι καταστάσεις/μεταβάσεις, και να μπορείτε να περιγράψετε πως η FSM αλληλεπιδρά με τη στοίβα.

γ) Να έχετε υλοποιήσει το κύκλωμα σε κώδικα VHDL. Θα πρέπει να υπάρχει "συμφωνία" μεταξύ των υποκυκλωμάτων του block diagram και των αρχείων VHDL.

δ) Να έχετε κάνει προσομοίωση, και να έχετε ετοιμάσει το UCF σύμφωνα με τον Πίνακα 1. Το κύκλωμα σας να έχει περάσει και το στάδιο "Implement" δηλ. "Place & Route".

Παρατηρήσεις/Σημειώσεις

α) Κάποια υποκυκλώματα είναι σύγχρονα, έχουν ρολόϊ, κάποια είναι συνδυαστικά.
β) Το κύκλωμα "debouncebutton.vhd" βρίσκεται στο courses.
γ) Υλοποιήστε τη στοίβα με τη δημιουργία μνήμης BlockRam των τριάντα δύο (32) θέσεων, πλάτους 4 bits. Για να το κάνετε αυτό Θα χρησιμοποιήσετε το εργαλείο Xilinx Core Generator. Οδηγίες για τη δημιουργία μνήμης με το εργαλείο Core Generator υπάρχουν στο courses.
δ) Επαληθεύστε τη λειτουργία κάθε υποκυκλώματος, βήμα-προς-βήμα καθώς τα σχεδιάζετε, δηλ: για κάθε flip-flop φτιάξτε ένα testbench και προσομοιώστε το, στη συνέχεια κάντε το ίδιο για το counter module (τα έχετε ήδη κάνει από το εργαστήριο #3). Επίσης, ελέγξτε τη σύνδεση με το debouncebutton. Ξεχωριστό testbench να κάνετε και για το fsm module, και ξεχωριστό testbench για τον adder (το έχετε ήδη από το εργαστήριο #2). Θα πρέπει επίσης να ελέγξετε τη λειτουργία της στοίβας με δημιουργία ξεχωριστού testbench. Προτιμήστε να συνδέσετε τη στοίβα με το counter module και να γράψετε ένα testbench για να ελέγξετε stack+counter μαζί. Ο counter είναι ο Stack Pointer! Ενώστε τα όλα (στοίβα, counter, adder, FSM κλπ) με τρόπο που να σας βολεύει στο topLevel και φτιάξτε το τελικό testbench.
ε) Λεπτομέρειες για το αναπτυξιακό Basys2 θα βρείτε στο παρακάτω link:
https://reference.digilentinc.com/ media/basys2:basys2_rm.pdf

Παραδοτέα: Πηγαίος κώδικας VHDL, κυματομορφές προσομοίωσης, παρουσίαση κυκλώματος.

Βαθμολογία:

Διεξαγωγή εργαστηρίου	**70%**
	Προετοιμασία 20%
	Προσομοίωση 30%
	Σωστή λειτουργία του κυκλώματος στο Board 20%
Αναφορές	**30%**

ΠΡΟΣΟΧΗ!

1) Η έλλειψη προετοιμασίας οδηγεί στην απόρριψη στη συγκεκριμένη εργαστηριακή άσκηση.

2) Η διαπίστωση αντιγραφής σε οποιοδήποτε σκέλος της άσκησης οδηγεί στην άμεση απόρριψη από το σύνολο των

εργαστηριακών ασκήσεων.

3) Ο βαθμός της αναφοράς μετράει στον τελικό βαθμό του εργαστηρίου μόνο αν ο βαθμός της διεξαγωγής του εργαστηρίου είναι (35/70)%.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ! ☺