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

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018

Εργαστήριο 2

ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΗ ΓΛΩΣΣΑ ΠΕΡΙΓΡΑΦΗΣ ΥΛΙΚΟΥ VHDL ΚΑΙ ΤΗΝ ΙΕΡΑΡΧΙΚΗ ΣΧΕΔΙΑΣΗ

ΕΚΠΟΝΗΣΗ: Καθ. Α. Δόλλας, Δρ. Κ. Παπαδημητρίου,
Δρ. Ε. Σωτηριάδης, Μ. Κιμιωνής

ΒΟΗΘΟΙ: (θα ανακοινωθούν)

ΕΚΔΟΣΗ : 9.1
Χανιά

Μέρος Α

Σκοπός

Είναι η περαιτέρω εξοικείωση με τη γλώσσα VHDL και την ιεραρχική σχεδίαση με πολλαπλά αρχεία. Θα σχεδιάσετε και θα υλοποιήσετε έναν full adder 4-bit τύπου Carry Look Ahead, με ιεραρχική σχεδίαση.

Προετοιμασία

Πριν την προσέλευση σας στο εργαστήριο **να έχετε υλοποιήσει σε γλώσσα VHDL το παρακάτω κύκλωμα. Επίσης θα δείξετε συνοπτικά τη διαδικασία σχεδίασης του κυκλώματος.**

Εξισώσεις που θα υλοποιηθούν

$$P_i = A_i \oplus B_i$$
$$G_i = A_i \cdot B_i$$

$$C_0 = G_0 + P_0 \cdot C_{in}$$
$$C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{in}$$
$$C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{in}$$
$$C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_{in}$$

$$S_0 = A_0 \oplus B_0 \oplus \mathrm{C_{in}}$$
$$S_i = A_i \oplus B_i \oplus \mathrm{C_{i-1}}, \mathbf{for\ i} > 0$$

Ζητούμενα

Να σχεδιάσετε και να υλοποιήσετε ένα κύκλωμα που έχει εισόδους και εξόδους όπως στον Πίνακα 1.

Όνομα	in/out	Πλάτος σε bit	Αντιστοίχιση στο Board
A	είσοδος	4	SW[3:0]
B	είσοδος	4	SW[7:4]
Cin	είσοδος	1	BTN0
S	έξοδος	4	LD[3:0]
C_3	έξοδος	1	LD5

Πίνακας 1: Είσοδοι – έξοδοι κυκλώματος

Το κύκλωμα λειτουργεί ως εξής:

Τα A,B είναι οι αριθμητικές είσοδοι του κυκλώματος και το Cin είναι το κρατούμενο εισόδου. Ο αθροιστής είναι τύπου 4-bit Carry Look Ahead. Το αποτέλεσμα της πράξης φαίνεται στο S και το κρατούμενο εξόδου στο C_3.

Παρατηρήσεις/Σημειώσεις

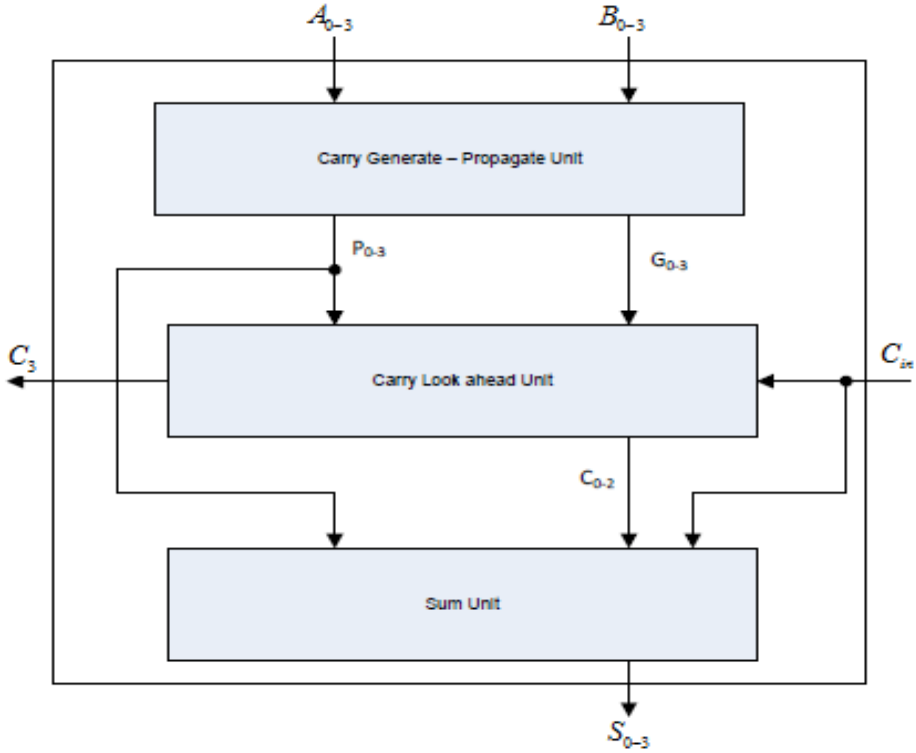
Για την ιεραρχική σχεδίαση σε VHDL, χρειάζεται να κάνουμε ανάλυση του κυκλώματος από πάνω προς τα κάτω (top down) και υλοποίηση από κάτω προς τα πάνω(bottom up). Συνεπώς, για τη συγκεκριμένη άσκηση η ανάλυση γίνεται ως εξής: ο adder των τεσσάρων bit αποτελείται από τρεις διαφορετικές μονάδες που συνδέονται κατάλληλα μεταξύ τους. Αυτές φαίνονται στο Σχήμα 1. Συγκεκριμένα:

1)Carry Generate/Propagate Unit: υπολογίζει τα τέσσερα (4) σήματα propagate και τα τέσσερα (4) σήματα carry generate.

2)Carry Look Ahead Unit: υπολογίζει τα τρία(3) σήματα Carry Look Ahead και το Carry Out.

3)Sum Unit: υπολογίζει τα τέσσερα(4) σήματα του αθροίσματος(sum).

Στο στάδιο της υλοποίησης υλοποιούμε κάθε μονάδα ξεχωριστά, και μετά ενώνουμε τις τρεις(3) μονάδες μεταξύ τους. Συνολικά πρέπει να υλοποιήσετε τέσσερα(4) διαφορετικά modules σε τέσσερα(4) διαφορετικά αρχεία.



Σχήμα 1: 4-bit Carry Look Ahead Adder block diagram

Μέρος Β

Σκοπός

Υλοποίηση μηχανής πεπερασμένων καταστάσεων(FSM) σε VHDL.

Προετοιμασία

Πριν την προσέλευση σας στο εργαστήριο **να έχετε υλοποιήσει σε γλώσσα VHDL το παρακάτω κύκλωμα. Επίσης θα δείξετε συνοπτικά τη διαδικασία σχεδίασης του κυκλώματος, π.χ. πίνακας καταστάσεων.**

Ζητούμενα

Να σχεδιάσετε και να υλοποιήσετε ένα κύκλωμα με εισόδους και εξόδους όπως στον Πίνακα 2.

Όνομα	in/out	Πλάτος σε bit	Αντιστοίχιση στο Board
RST	in	1	BTN3
IN0	in	1	BTN0
IN1	in	1	BTN1
IN2	in	1	BTN2
LED	Out	8	LD[7:0]

Πίνακας 2: Είσοδοι - έξοδοι του κυκλώματος

Το κύκλωμα λειτουργεί ως FSM που φαίνεται στο Σχήμα 2, ως εξής:

Τροφοδοσία/Reset (πατάμε BTN3) κυκλώματος:

- Μετάβαση στην κατάσταση A

Κάθε φορά που πατάμε 1 φορά το button 0 (BTN0):

- αν είσαι στην κατάσταση A -> B
- αν είσαι στην κατάσταση B -> C
- αν είσαι στην κατάσταση C -> A

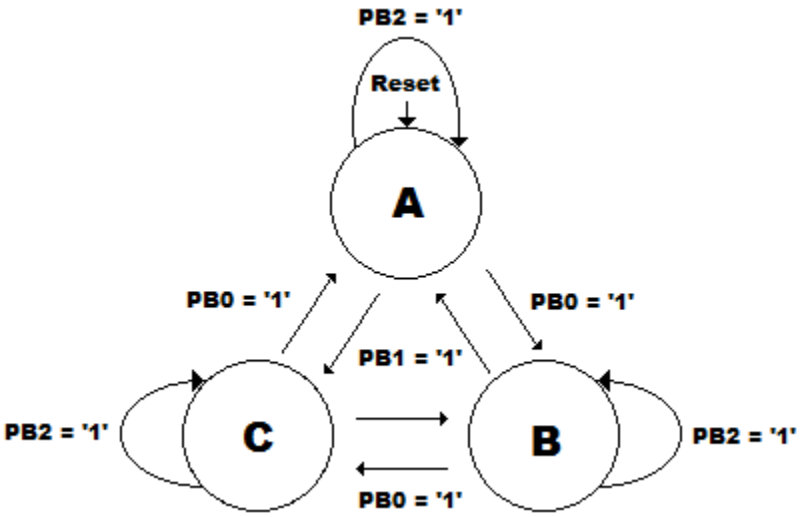
Κάθε φορά που πατάμε 1 φορά το button 1 (BTN1):

- αν είσαι στην κατάσταση A -> C
- αν είσαι στην κατάσταση C -> B
- αν είσαι στην κατάσταση B -> A

Κάθε φορά που πατάμε 1 φορά το button 2 (BTN2):

- αν είσαι στην κατάσταση A -> A
- αν είσαι στην κατάσταση C -> C
- αν είσαι στην κατάσταση B -> B

Στην **κατάσταση Α,** όλα τα LED ανάβουν.
Στην **κατάσταση Β** τα τέσσερα(4) αριστερά LEDs αναμμένα (LED 7,6,5,4), και τα υπόλοιπα τέσσερα σβηστά.
Στην **κατάσταση Β** τα τέσσερα(4) δεξιά LEDs αναμμένα (LED 3,2,1,0), και τα υπόλοιπα τέσσερα σβηστά.
Η περιγραφή αυτή



Σχήμα 2: Σχηματική παρουσίαση της FSM

Παρατηρήσεις/Σημειώσεις

(1) Κάποια υποκυκλώματα είναι συνδυαστικά, και κάποια είναι σύγχρονα, δηλ. έχουν ρολόι.
(2) Επαληθεύστε τη λειτουργία κάθε υποκυκλώματος αφού το σχεδιάσετε ξεχωριστά, με χρήση ξεχωριστού testbench.
(3) Η σύνδεση των modules γίνεται ιεραρχικά, δηλ. έχουμε modules που συνδέονται μεταξύ τους μέσα σε 1 άλλο module που βρίσκεται σε 1 παραπάνω επίπεδο. Αυτό με τη σειρά του μπορεί να συνδέεται με άλλα modules, μέσω ενός module που βρίσκεται σε 1 ακόμη παραπάνω επίπεδο κ.ο.κ. Σκεφτείτε πως θα συνδέσετε τις μονάδες για τη δημιουργία του carry look ahead. Θα χρειαστεί να δημιουργήσετε instances.
(4) Δημιουργήστε topLevel στο οποίο θα συνδέσετε όλα τα σήματα που θέλετε να «δείτε» στην προσομοίωση. Φτιάξτε το τελικό testbench, και συνδέστε το με το topLevel.
(5) Σε κάθε κύκλωμα που θα σχεδιάζετε, για να επικοινωνήσει με τον εξωτερικό κόσμο, δηλ. I/O π.χ. buttons, switches, LEDs, απαιτείται η δημιουργία του αρχείου ucf (user constraint file). Εκεί συνδέστε όσα σήματα της σχεδίασής σας χρειάζονται.
(6) Στο μέρος Β' έχουμε και ρολόι. Στο γενικό αρχείο ucf για το αναπτυξιακό Basys 2, θα βρείτε ποιο είναι το σήμα CLOCK για να οδηγήσετε το ρολόι του κυκλώματός σας: ψάξτε

για το NET "mclk". Το ucf που χρειάζεται να φτιάξετε για τη 2η εργαστηριακή άσκηση, διαφέρει απ' αυτό της 1ης.
(7) Θα σας δώσουμε κύκλωμα που δημιουργεί παλμό διάρκειας 1 κύκλου ρολογιού, είναι το αρχείο "singlepulsegen.vhd". Σας χρειάζεται για το μέρος Β'. Ελέγξτε το μόνο του πρώτα, υπάρχει testbench για να το ελέγξετε. Το κύκλωμα "singlepulsegen.vhd" το ενσωματώνετε στη σχεδίαση σας με την επιλογή "add copy of source". Όταν θα φτιάξετε το αρχείο "topLevel.vhd", μέσα σε αυτό δημιουργήστε 1 instance της FSM, και 3 για το "singlepulsegen". Με αυτό τον τρόπο δημιουργείτε 3 διαφορετικές γεννήτριες παλμών διάρκειας 1 κύκλου ρολογιού. Για κάθε μια απ' αυτές τις γεννήτριες, η είσοδός της θα συνδεθεί με ξεχωριστό push button (BTN), και η έξοδός της με ξεχωριστή είσοδο της FSM σας. Συγκεκριμένα: το "singlepulsegen.vhd" έχει 1 είσοδο και 1 έξοδο (εκτός των clock/reset). Για το κάθε instance η είσοδός του πρέπει να συνδεθεί με διαφορετικό push button (έχουμε 3 από αυτά, όχι το reset όμως), και η έξοδός του με διαφορετική είσοδο της FSM (έχει 3 εισόδους).
(8) Το testbench που συνοδεύεται με το "singlepulsegen.vhd" είναι μόνο για τον έλεγχο του. Μην το προσθέσετε στη σχεδίαση σας! Πάρτε όμως ιδέες για το πως φτιάχνουμε ένα testbench, και φτιάξτε το δικό σας.
(9) Instances φτιάχνουμε με "component" και "port map", π.χ. για το "singlepulsegen", θα το "καλέσετε" 1 φορά με χρήση της εντολής "component", και στη συνέχεια κάνετε 3 φορές "port map".
(10) Λεπτομέρειες για το αναπτυξιακό Basys2 υπάρχουν στο link:
https://reference.digilentinc.com/ media/basys2:basys2_rm.pdf
(11) Θεωρούμε την περίπτωση να πατηθούν και τα δύο buttons ταυτόχρονα ως αδύνατη κατάσταση, οπότε την αγνοούμε.

Παραδοτέα:
Πηγαίος κώδικας VHDL, κυματομορφές προσομοίωσης, παρουσίαση κυκλώματος, επίσης να δείξετε τη διαδικασία λύσης, τι χρησιμοποιήσατε από τη θεωρία και πως, π.χ. πίνακας καταστάσεων.

Βαθμολογία: Τα κυκλώματα είναι ισοδύναμα βαθμολογικά

Διεξαγωγή εργαστηρίου	**Σύνολο: 70%**
	Προετοιμασία: 20%
	Προσομοίωση: 30%
	Ορθή λειτουργία κυκλώματος στο Board: 20%
Αναφορές	**30%**

ΠΡΟΣΟΧΗ!

1) Η προεργασία να είναι σε ηλεκτρονική μορφή, σε αρχείο pdf.

2) Η έλλειψη προετοιμασίας οδηγεί σε απόρριψη.

3) Η διαπίστωση αντιγραφής σε οποιοδήποτε σκέλος της άσκησης οδηγεί στην απόρριψη από το σύνολο των εργαστηριακών ασκήσεων. Αυτό γίνεται οποιαδήποτε στιγμή στη διάρκεια του εξαμήνου.

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

ΚΑΛΗ ΕΠΙΤΥΧΙΑ! ☺