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

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2016

Εργαστήριο 5

ΚΥΚΛΩΜΑ ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΥ BOOTH ΜΕ ΑΝΑΓΝΩΣΗ/ΕΓΓΡΑΦΗ ΣΕ PRE-INCREMENT/POST-DECREMENT ΣΤΟΙΒΑ

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 τον **πολλαπλασιασμό Booth.** Τα δεδομένα διαβάζονται από μια στοίβα, και το αποτέλεσμα εγγράφεται πίσω στη στοίβα. Πρέπει να έχετε υλοποιημένη τη λειτουργικότητα της pre-increment/post-decrement στοίβας (εργ 4). Χρειάζονται οι δύο βασικές λειτουργίες **push/pop (εισαγωγή/εξαγωγή),** ενώ η **πρόσθεση (add)** δε χρειάζεται σαν μεμονωμένη πράξη (δε ζητείται από τον χρήστη). Όμως το κύκλωμα της πρόσθεσης χρειάζεται ως μέρος του κυκλώματος για να υλοποιηθεί ο πολλαπλασιασμός με τον αλγόριθμο Booth (addition/subtraction).

Άρα μαζί με τις λειτουργίες **push/pop (εισαγωγή/εξαγωγή),** το κύκλωμα θα εκτελεί **πολλαπλασιασμό** μεταξύ του στοιχείου που βρίσκεται στο TOS(Top Of Stack) και του προηγούμενου στοιχείου (TOS-1). Προσοχή: μόνο το αποτέλεσμα του πολλαπλασιασμού εγγράφεται στη στοίβα, οι δύο αριθμοί μεταξύ των οποίων έγινε η πράξη, δεν εγγράφονται πάλι στη στοίβα. Είτε έχουμε υπερχείλιση (overflow) στο αποτέλεσμα είτε όχι, θα εγγράφονται πάντα τα 4 LSB του αποτελέσματος του πολλαπλασιασμού στη στοίβα. Όμως αν υπάρχει υπερχείλιση στο αποτέλεσμα, η εγγραφή στη στοίβα θα συνοδεύεται με την ενεργοποίηση ενός LED (MultiplicationOvf). Άρα, ο πολλαπλασιασμός μεταξύ των δύο αριθμών εκτελείται κανονικά και το αποτέλεσμα παράγεται σωστά εσωτερικά (δεν χρειάζεται καταχωρητής για να κρατήσετε το αποτέλεσμα), στη στοίβα όμως θα γίνονται push τα 4 LSBs μόνο. Στις εξόδους LED θα φαίνεται η τελευταία τιμή που εγγράφηκε στη στοίβα, ενώ αν η στοίβα είναι άδεια θα φαίνεται το "0".

Η κύρια διαφορά του εργαστηρίου αυτού με τα προηγούμενα είναι πως σχεδόν όλα τα υποκυκλώματα, εκτός του flip-flop, θα υλοποιηθούν με σχεδίαση structural δηλ. συγκριτές, πολυπλέκτες, αποπολυπλέκτες, κωδικοποιητές, κλπ. Η FSM θα υλοποιηθεί με σχεδίαση behavioral.

Πολλαπλασιασμός Booth – Σύνοψη

- Βασίζεται σε συνδυασμό αριθμητικής ολίσθησης, πρόσθεσης και αφαίρεσης.
- Ο αλγόριθμος εκτελείται σε επαναλήψεις.
- Έστω A = M*Q. Αν M (πολλαπλασιαστέος) είναι m bits, και Q (πολλαπλασιαστής) είναι q bits, τότε το πλάτος του αποτελέσματος A είναι m+q bits.

- Πριν ξεκινήσουμε την πράξη, θεωρούμε ένα βοηθητικό bit το οποίο μπαίνει σαν ουρά στο τέλος του Q με αρχική τιμή "0", δηλ. Q_t = "0".
- Σε κάθε επανάληψη γίνεται σίγουρα ολίσθηση.
- Το αν θα γίνει αφαίρεση/πρόσθεση/τίποτα σε μια επανάληψη, εξαρτάται από τον συνδυασμό της τιμής του τελευταίου bit του Q και του Q_t, δηλ:
  - αν $Q[0]Q_t$ = "00"/"11": do nothing
  - Αν $Q[0]Q_t$ = "10" : sub
  - Αν $Q[0]Q_t$ = "01" : add
  - σημ: το Q ολισθαίνει σε κάθε επανάληψη μαζί με το A, δείτε τον αλγόριθμο
- Ο αριθμός των επαναλήψεων που χρειάζονται, είναι ίσος με τον αριθμό των bits του Q.
- Το εύρος των πράξεων είναι ίσο με m, π.χ. αν m=4 το πλάτος του adder είναι 4.
- Για την πράξη της αφαίρεσης, χρειάζεται το συμπλήρωμα ως προς 2 του πολλαπλασιαστέου.
- Η ολίσθηση είναι αριθμητική, όχι λογική.

Περιγραφή κυκλώματος και λειτουργίας

Αποτελείται από τα παρακάτω βασικά υποκυκλώματα:

- Μια στοίβα υλοποιημένη όπως την φτιάξατε από το εργ 4, με μνήμη πλάτους 4-bit, που αποθηκεύει μέχρι και 31 στοιχεία. Είναι 32 θέσεων. Η στοίβα έχει όλα τα σήματα εξόδου που είδατε στη θεωρία, δηλ. empty, almost-empty, full, almost-full.
- Ο stack pointer είναι ο δείκτης της στοίβας, υλοποιημένος ως μετρητής 5-bits που μπορεί να μετράει άνω/κάτω. Η σχεδίαση του είναι structural (εργ 4).
- Μια FSM που ανάλογα με τις τιμές τριών εξωτερικών εισόδων (δίνονται με Push Button) θα προκαλεί μια από τις εξής 3 λειτουργίες: push, pop, mul. 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/mul. Επιπλέον, η FSM διαχειρίζεται το σύστημα σε περίπτωση υπερχείλισης της στοίβας (StackOvf), καθώς και σε περίπτωση υπερχείλισης του αποτελέσματος του πολλαπλασιασμού (MultiplicationOvf). Σε αυτές τις περιπτώσεις, πρέπει να οδηγούνται αντίστοιχα τα LEDs (περισσότερες λεπτομέρειες παρακάτω).
- Αθροιστής ?-bits. Η σχεδίαση του είναι structural (εργ

2). Τι πλάτος πρέπει να έχει?

- Συγκριτής ισότητας 5-bit. Ξεκινήστε με ένα συγκριτή ισότητας που παίρνει 2 εισόδους, 1-bit κάθε μια. Σε "υψηλότερο επίπεδο", θα συνδέσετε τους συγκριτές (instances) μεταξύ τους για να φτιάξετε τον συγκριτή 5-bits. Θα χρειαστεί να δημιουργήσετε instances κάποιων πυλών (υπάρχουν και στις βιβλιοθήκες της Xilinx, 2 και 3 εισόδων). Σε ακόμη "υψηλότερο επίπεδο" δημιουργείτε όσα instances χρειάζεστε από τον συγκριτή που φτιάξατε, και κάνετε port map τις εισόδους του καθενός με τα σήματα που πρέπει. Η 1η είσοδος είναι κοινή για όλους τους συγκριτές, το σήμα SP. Η 2η είσοδος είναι διαφορετική για κάθε συγκριτή: είναι η τιμή που αντιστοιχεί σε StackFull, StackAlmostFull, StackEmpty, StackAlmostEmpty, StackOvf.
- Συγκριτές ανισότητας, υλοποιημένοι με σχεδίαση structural όπως ο συγκριτής ισότητας. Χρειάζεται για έλεγχο αν ο SP έχει τιμή μεγαλύτερη του StackAlmostEmpty/μικρότερη του StackAlmostFull. Αν αυτό ισχύει τότε θα υπάρχει τέτοια ένδειξη στα LEDs, δηλ. υποδηλώνει πως η στοίβα έχει τουλάχιστον 2 στοιχεία & δε βρίσκεται σε καμμιά από τις καταστάσεις StackFull, StackAlmostFull, StackEmpty, StackAlmostEmpty, StackOvf.
- Καταχωρητής ολίσθησης (structural). Τι μέγεθος πρέπει να έχει, πόσα flip-flops χρειάζονται? Ξεκινήστε απο το βασικό κύκλωμα, 1 flip-flop. Βάλτε σύγχρονο Reset. Έχει Din, Ld, En, Q, Clk. Σε "υψηλότερο επίπεδο", θα συνδέσετε τα flip-flops (instances) μεταξύ τους για να φτιάξετε καταχωρητή ολίσθησης **?**-bits. Η ολίσθηση, είναι προς τα δεξιά μόνο. Σε ακόμη "υψηλότερο επίπεδο", δημιουργείτε όσα instances χρειάζεστε από τον καταχωρητή ολίσθησης που φτιάξατε (1 ή 2 instances χρειάζεστε?), και κάνετε port map τις εισόδους του καθενός με τα σήματα που πρέπει. σημ. 1: ένα flip-flop χρησιμοποιείτε, αυτό που σχεδιάσατε στο εργ. 3 για τον μετρητή, βάλτε όμως σύγχρονο reset. Χρησιμοποιείστε instances του ίδιου flip-flop για να φτιάξετε τον καταχωρητή ολίσθησης. σημ. 2: **για τον καταχωρητή ολίσθησης ζητείται ξεχωριστό testbench, θα το δείξετε στη διεξαγωγή του εργαστηρίου**. σημ. 3: τη βασική λειτουργικότητα του κυκλώματος flip-flop, δηλ Q<=Din στη θετική ακμή του Clk, μπορείτε να την κάνετε σε behavioral σχεδίαση. Τη λειτουργικότητα για τον έλεγχο του flip-flop από τα σήματα Ld, En, τη σχεδιάζετε με πύλες. σημ. 4: έχει τη δυνατότητα παράλληλης φόρτισης (parallel load), άρα στον ίδιο κύκλο ρολογιού μπορούν

να φορτωθούν όλα τα flip-flops με μια τιμή εισόδου από τον εξωτερικό κόσμο (χρειάζεται πολυπλέκτης στην είσοδο των flip-flops).

- Κωδικοποιητής (structural). Για την κωδικοποίηση των εξόδων των συγκριτών στα LEDs. 3 LEDs θα "δείχνουν" το StackFull, StackAlmostFull, StackEmpty, StackAlmostEmpty, StackOvf, Stack2atLeast με κωδικοποίηση "000", "001", "100", "111", "101", "010" αντίστοιχα.
- Πολυπλέκτης (structural). Το υποκύκλωμα αυτό μπορεί να το χρειαστείτε σε διάφορα σημεία. Σίγουρα πριν την είσοδο DataIn της στοίβας, για να επιλέγετε αν θα γίνει push τιμή από τα switches, ή αν θα γίνει push το αποτέλεσμα της πράξης. Σε χαμηλό επίπεδο φτιάξτε έναν πολυπλέκτη 2:1 μόνο από πύλες. Αν χρειαστείτε μεγαλύτερο πολυπλέκτη, π.χ. 4:1 mux, τότε σε υψηλότερο επίπεδο δημιουργήστε 3 instances του 2:1 mux και συνδέστε τα κατάλληλα.

Τα παραπάνω είναι τα βασικά υποκυκλώματα, όμως εσείς μπορείτε να υλοποιήσετε οτιδήποτε άλλα υποκυκλώματα χρειαστείτε για να φτάσετε στην τελική υλοποίηση.

Προσέξτε στην 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. **Mul:** πολλαπλασιασμός 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 πολλαπλασιασμός, και αν υπάρχει μόνο ένα στοιχείο Α μέσα στη στοίβα (stack almost-empty) τότε εκτελείται η πράξη Α*0.
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. **MultiplicationOvf:** Αν το αποτέλεσμα του πολλαπλασιασμού υπερβεί τον αριθμό 15_{10}, το αποτέλεσμα (LSB) εγγράφεται στη στοίβα (γίνεται push), και ενεργοποιείται το LED που αντιστοιχεί στο MultiplicationOvf. Το σύστημα θα μπορεί να συνεχίσει

κανονικά η λειτουργία του, δηλαδή θα αντιδράει στα σήματα εισόδου.

Εξωτερικό I/O κυκλώματος & διεπαφή με το Basys2

Όνομα	Είσοδος/Έξοδος	Πλάτος σε bit	Αντιστοίχηση στο Board
Push	in	1	BTN0
Pop	in	1	BTN1
Mul	in	1	BTN2
Reset	in	1	BTN3
Clock	in	1	MCLK
NumIn	in[3:0]	4	SW[3:0]
NumOut	out[3:0]	4	LED[3:0]
MultiplicationOvf	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]
Stack2atLeast	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 η στοίβα έχει τουλάχιστον 2 στοιχεία και δε βρίσκεται σε καμιά από τις παραπάνω περιπτώσεις
7. Όταν το αποτέλεσμα του πολλαπλασιασμού δύο δυαδικών αριθμών υπερβεί τον αριθμό 15_{10} (**MultiplicationOvf)**

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

		LED7	LED6	LED5	LED4
1	Empty	**0**	**0**	**0**	**x**
2	AlmostEmpty	**0**	**0**	**1**	**x**
3	Full	**1**	**1**	**1**	**x**

4	AlmostFull	**1**	**0**	**0**	**x**
5	StackOvf	**1**	**0**	**1**	**x**
6	Stack2atLeast	**0**	**1**	**0**	**x**
7	MultiplicationOvf	**x**	**x**	**x**	**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 (mul)
Αν πατηθεί μια φορά εκτελείται συνδυασμός των λειτουργιών pop, push και mul. Συγκεκριμένα: θα γίνουν διαδοχικά 2 pop, 1 mul, και push του αποτελέσματος στη στοίβα. Επίσης, αν η στοίβα είναι άδεια (stack empty) δεν εκτελείται πολλαπλασιασμός, και αν υπάρχει ακριβώς ένα στοιχείο στη στοίβα A (stack almost-empty), τότε εκτελείται η πράξη A*0.

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.** Το flip-flop θα χρησιμοποιήσετε για να υλοποιήσετε και τον καταχωρητή ολίσθησης. Για να οδηγήσετε τις εισόδους των flip-flops θα χρειαστείτε πολυπλέκτη. Για τον adder χρησιμοποιήστε πάλι structural σχεδίαση, τον carry-look-ahead (CLA) adder, **από το εργαστήριο #2.** Η FSM θα υλοποιηθεί ως ξεχωριστό module. Τη στοίβα την έχετε υλοποιήσει **από το εργαστήριο #4**, 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 mul 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) Φτιάξτε πρώτα το κύκλωμα του πολλαπλασιασμού, χωρίς στοίβα. Εκτός από τα βασικά κυκλώματα υλοποιημένα με σχεδίαση structural, χρειάζεται να υλοποιήσετε την FSM για να υλοποιηθούν οι επαναλήψεις του αλγορίθμου. Ανάλογα στο state που βρίσκεται η FSM ενεργοποιείστε τα κατάλληλα σήματα για πρόσθεση και μετά ολίσθηση, αφαίρεση και μετά ολίσθηση, ή μόνο ολίσθηση.
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). Το ίδιο θα κάνετε για τον καταχωρητή ολίσθησης, τους συγκριτές και τον κωδικοποιητή. 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 έχετε ελέγξει τη λειτουργία της στοίβας με δημιουργία ξεχωριστού testbench από το εργαστήριο #4. Προτιμήστε να συνδέσετε τη στοίβα με το counter module και να γράψετε ένα testbench για να ελέγξετε stack+counter μαζί. Ενώστε τα όλα (στοίβα, μετρητής, αθροιστής, συγκριτές, καταχωρητής ολίσθησης, κωδικοποιητής, FSM, debouncebutton κλπ) με τρόπο που να σας βολεύει στο topLevel και φτιάξτε το τελικό testbench.
ε) Προσοχή: χρειάζεστε το συμπλήρωμα ως προς 2 του πολλαπλασιαστέου (M). Πως θα φτιάξετε αυτό το κύκλωμα? **σημ.:** το συμπλήρωμα ως προς 2 ενός αριθμού, είναι το αποτέλεσμα της πρόσθεσης μεταξύ του συμπληρώματος τους ως προς 1, και του "1" (1's complement of M + "1").
ζ) 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! ☺