ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 1

Μετρητής BCD και απεικόνιση σε 7-segment display

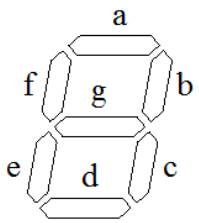
Μέρος Α: Αποκωδικοποιητής BCD σε 7-segment

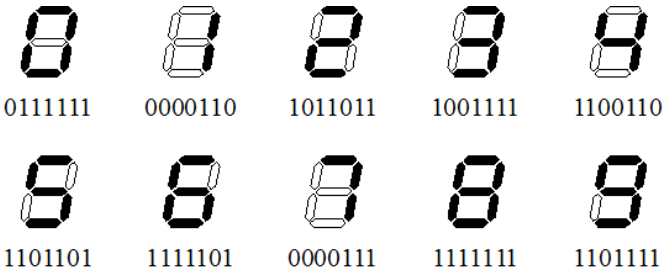
Στο πρώτο μέρος της άσκησης, θα υλοποιήσετε την αριθμητική κωδικοποίηση *δυαδικά κωδικοποιημένων δεκαδικών (binary coded decimal – BCD)*.

Εάν θεωρήσουμε ένα μόνο δεκαδικό ψηφίο, οι δέκα πιθανές τιμές είναι 0, 1, 2, 3, 4, 5, 6, 7, 8 και 9. Χρειαζόμαστε τουλάχιστον 4 bit σε ένα δυαδικό κώδικα και ο κώδικας BCD που είναι ο πιο συνηθισμένος, και έχει τις ακόλουθες κωδικές λέξεις:

0: 0000	1: 0001	2: 0010	3: 0011	4: 0100
5: 0101	6: 0110	7: 0111	8: 1000	9: 1001

Πολλά ψηφιακά συστήματα εμφανίζουν τους δεκαδικούς αριθμούς με χρήση οθονών 7 τμημάτων (7-segment displays). Κάθε ψηφίο της οθόνης αποτελείται από επτά ξεχωριστά φώτα, που τοποθετούνται όπως φαίνεται στην παρακάτω εικόνα. Αν έχουμε ένα ψηφίο κωδικοποιημένο με BCD και πρέπει να δείξουμε το ψηφίο σε μια οθόνη 7 τμημάτων, χρειαζόμαστε έναν *αποκωδικοποιητή 7 τμημάτων (7-segment decoder)*. Μιλώντας αυστηρά, θα το ονομάζαμε «μετατροπέα κώδικα 7 τμημάτων» αφού μετατρέπει μια είσοδο με κώδικα BCD σε μια έξοδο με κώδικα 7 τμημάτων. Ωστόσο, ο όρος «αποκωδικοποιητής 7 τμημάτων» χρησιμοποιείται ευρύτατα. Υποθέτοντας ότι ένα τμήμα ανάβει αν η είσοδός του είναι 1, χρειαζόμαστε έναν κώδικα των 7 bit για την αναπαράσταση των ψηφίων από 0 μέχρι 9. Η κωδική λέξη για κάθε ψηφίο έχει ένα bit 1 για κάθε τμήμα που είναι αναμμένο και ένα bit 0 για κάθε τμήμα που δεν είναι αναμμένο. Τότε, ένας αποκωδικοποιητής 7 τμημάτων μετατρέπει μεταξύ του BCD και αυτού του κώδικα των 7 bit. Ένας πιθανός κώδικας φαίνεται στην παρακάτω εικόνα, με τα bit να αντιστοιχούν από αριστερά προς τα δεξιά στα τμήματα g μέχρι a (g, f, e, d, c, b, a).



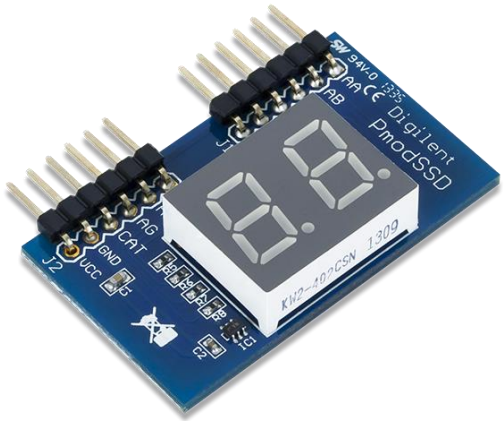


Για το πρώτο μέρος της άσκησης, θα σχεδιάσετε, θα προσομοιώσετε και θα υλοποιήσετε ένα κύκλωμα αποκωδικοποιητή BCD σε 7-segment που θα δέχεται ως είσοδο στα DIP switches SW3, SW2, SW1, SW0 (LSbit) της αναπτυξιακής κάρτας ένα δεκαδικά κωδικοποιημένο δεκαδικό αριθμό (BCD) και θα οδηγεί ένα από τα δύο ψηφία του 7-segment display (Pmod SSD κοινής καθόδου της Digilent) που συνδέεται στην αναπτυξιακή κάρτα ZedBoard μέσω των GPIO σημάτων του Bank 13 (3.3V) και των JA1 και JB1 connectors.

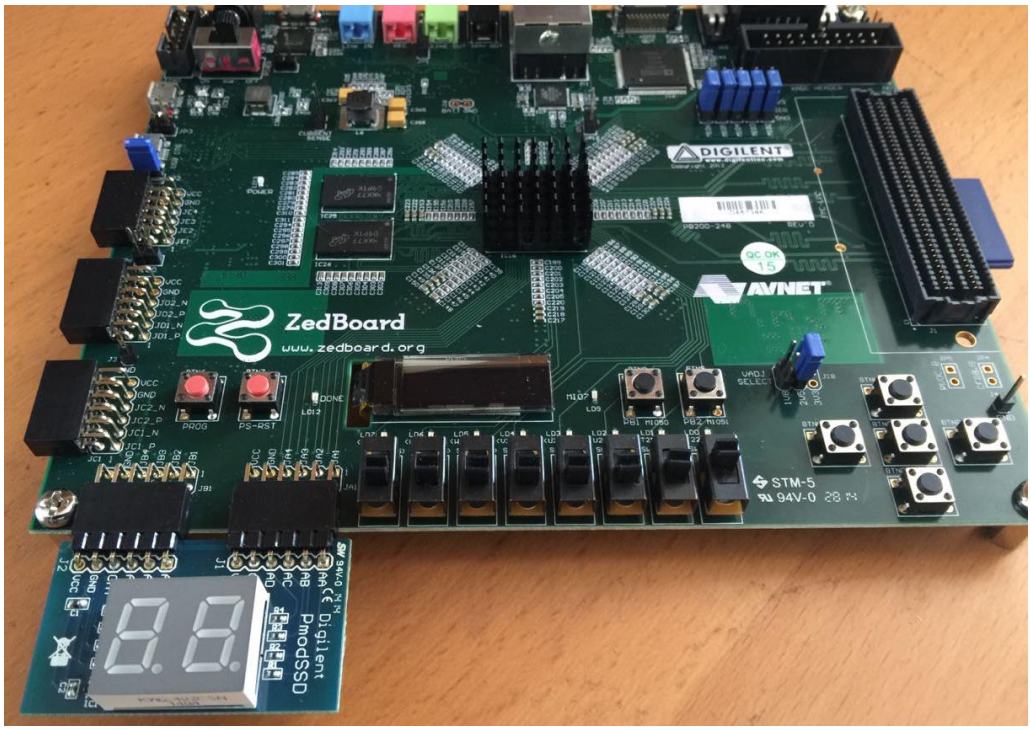
Το Pmod SSD υποστηρίζει δύο ψηφία. Τα segments ανάβουν οδηγώντας με λογικό 1 την αντίστοιχη άνοδο σε όποιο από τα δύο ψηφία έχει επιλεγεί από το χρήστη οδηγώντας αντίστοιχα το Digit Selection pin (C) του Pmod SSD σε λογικό 1 (για το αριστερό ψηφίο) ή 0 (για το δεξί ψηφίο). Χρησιμοποιείστε το δεξί ψηφίο.

Το datasheet του Pmod SSD βρίσκεται στο e-class.

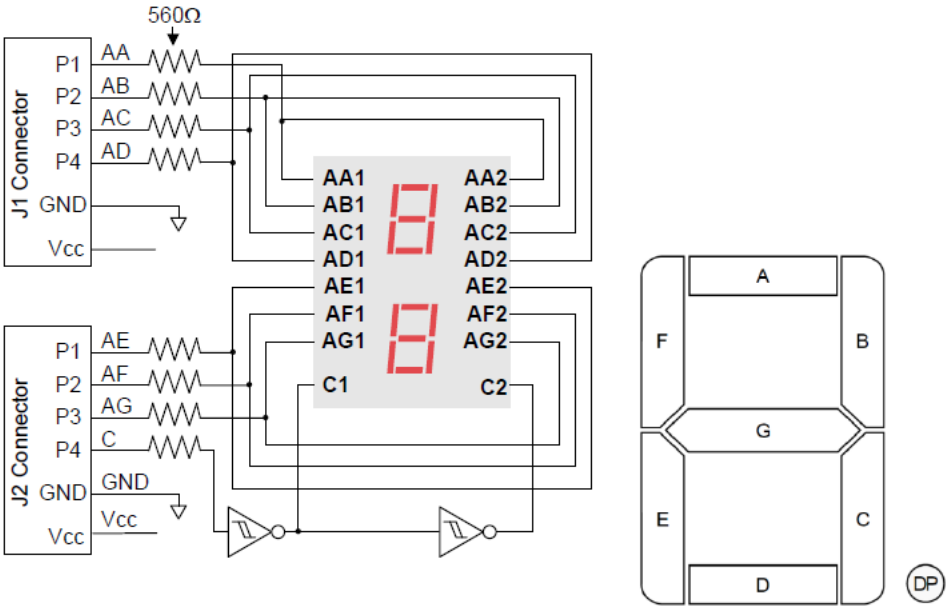
Το FPGA της κάρτας Zedboard είναι το xc7z020clg484-1.



Εικόνα 1. Pmod SSD της Digilent



Εικόνα 2. Σύνδεση του Pmod SSD στα JA1 και JB1 Pmod connectors της ZedbBoard



Εικόνα 3. Σχηματικό διάγραμμα του Pmod SSD της Digilent

Στα πλαίσια της άσκησης σας δίδεται το αρχείο .xdc με τα constraints για τα pin assignments που έχουν προκύψει λαμβάνοντας υπόψιν α) το σχηματικό διάγραμμα του Pmod SSD (Εικόνα 3), β) τον Πίνακα 1 που περιγράφει τις συνδέσεις των DIP switches με τα pins του Zynq FPGA και γ) τον Πίνακα 2 που περιγράφει τις συνδέσεις των Pmod JA1 και Pmod JA2 με τα pins του Zynq FPGA.

Signal Name	Zynq pin
SW0	F22
SW1	G22
SW2	H22
SW3	F21
SW4	H19
SW5	H18
SW6	H17
SW7	M15

Πίνακας 1. Συνδέσεις των DIP switches στα pins του Zynq FPGA

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

Πίνακας 2. Συνδέσεις των JA1 και JA2 Pmods στα pins του Zynq FPGA

Στο e-class θα σας δοθεί ένα template πηγαίου κώδικα με την περιγραφή του entity καθώς και το αντίστοιχο testbench σε VHDL.

- Συμπληρώστε το architecture του entity
- Ελέγξτε τη λειτουργία του κυκλώματος τόσο στη προσομοίωση όσο και στο υλικό προγραμματίζοντας το FPGA
- Συμπληρώστε τον παρακάτω πίνακα κοιτάζοντας το report μετά το implementation:

FF:	_____
LUT:	_____
I/O:	_____

Σημείωση: Στη σχεδίαση του αποκωδικοποιητή να χρησιμοποιήσετε δομή case και όχι selected signal assignment (with-select) ή conditional signal assignment (when-else). Επίσης, εάν η BCD είσοδος είναι μεγαλύτερη του εννέα (9) η οθόνη δεν θα πρέπει να εμφανίζει τίποτα.

Μέρος Β: Μετρητής BCD και απεικόνιση σε 7-segment display

Στο δεύτερο μέρος της άσκησης, θα υλοποιήσετε έναν μετρητή BCD των 4-bit σε VHDL που υποστηρίζει:

- μέτρηση είτε προς τα πάνω (είσοδος *up*=1) είτε προς τα κάτω (είσοδος *up*=0),
- δυνατότητα φόρτωσης (είσοδος *load*=1) με τα δεδομένα φόρτωσης σε είσοδο *load_data*
- ασύγχρονο μηδενισμό (είσοδος *reset*=1)

Επειδή το ρολόι (*clk*) του συστήματος παράγεται από τον εξωτερικό κρύσταλλο των 100MHz που διαθέτει η κάρτα, ο μετρητής θα δέχεται ένα σήμα *enable* με χαμηλή συχνότητα 1Hz. Το σήμα *enable* θα παράγεται από μονάδα *enable_gen*. Η μονάδα *enable*_gen είναι μια γεννήτρια παλμών (pulse generator) που δίνει 1 παλμό (enable) μετά από συγκεκριμένο αριθμό κύκλων (κάθε 100,000,000 για 1Hz).

Οι είσοδοι του μετρητή συνδέονται στα DIP switches και στα push-buttons της κάρτας ZedBoard σύμφωνα με τον παρακάτω πίνακα.

Είσοδοι	**Περιγραφή**	**DIP Switch**
rst	Ασύγχρονο reset	BTNU
load	Ενεργοποίηση φόρτωσης	SW7
up	1: Μέτρηση προς τα πάνω 0: Μέτρηση προς τα κάτω	SW6
load_data[3]	Δεδομένα φόρτωσης	SW3
load_data[2]		SW2
load_data[1]		SW1
load_data[0]		SW0

Πίνακας 2. Είσοδοι του μετρητή

Οι έξοδος του μετρητή θα απεικονιστεί σε 7-segment display (χρήση του δεξιού ψηφίου του Pmod SSD και του αποκωδικοποιητή που σχεδιάσατε στο πρώτο μέρος της Εργαστηριακής Άσκησης).

- Συμπληρώστε το architecture του entity counter
- Μελετήστε το testbench του counter που σας δίνεται
- Τροποποιήστε το testbench του μετρητή που σας δίνεται για να προσθέσετε στο τέλος το παρακάτω σενάριο δοκιμής:
    a. Φόρτωση της τιμής "0011"
    b. Μέτρηση προς τα επάνω έως την τιμή "1001"
    c. Μέτρηση προς τα κάτω έως την τιμή "0001"
- Ελέγξτε τη λειτουργία του κυκλώματος του μετρητή σας με προσομοίωση μελετώντας τα waveforms
- Μελετήστε προσεκτικά το σύστημα (system.vhd) και που κάνει instantiate το κύκλωμα enable_gen που σας παρέχεται καθώς και τα κυκλώματα του μετρητή και του αποκωδικοποιητή που σχεδιάσατε και προσομοιώσατε επιτυχώς.
- Μελετήστε και τροποποιείστε το αρχείο .xdc με τα pin constraints για το σύστημα (system) που ενσωματώνει το μετρητή και τον αποκωδικοποιητή που σχεδιάσατε μαζί με το κύκλωμα enable_gen.

  Για τον ορισμό της συχνότητας ρολογιού δώστε το παρακάτω timing constraint στο αρχείο .xdc
  create_clock -period 10.000 -name clk_100MHz -waveform {0.000 5.000} [get_ports clk_100MHz]

- Ελέγξτε τη λειτουργία του μετρητή σας μέσω του συστήματος που ενσωματώνει τον μετρητή σας και το τον αποκωδικοποιητή σας απευθείας στο υλικό προγραμματίζοντας το FPGA.
- Συμπληρώστε τα παρακάτω στοιχεία κοιτάζοντας το report μετά το implementation:

  FF: _____
  LUT: _____
  I/O: _____
  BUFG: _____

- Προσδιορίστε την ελάχιστη περίοδο (minimum period) σε ns και το πιο κρίσιμο μονοπάτι (max delay path) εκτελώντας την εντολή *report_timing_summary –datasheet*

  Min Period (ns): _____
  Max Delay Path
  Slack (ns): _____
  Source: _____
  Destination: _____
  Logic Levels: _____

Εργαστηριακή άσκηση 1 Μετρητής BCD και απεικόνιση σε 7-segment	
Ημερομηνία:	
Ονοματεπώνυμο:	
Βαθμός:	
Εξεταστής:	