ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 8

Σκοπός της εργαστηριακής άσκησης είναι να χρησιμοποιήσουμε ένα από τα πολλά interfaces τα οποία παρέχουν οι αναπτυξιακές μας κάρτες για επικοινωνία με κάποια περιφερειακή συσκευή. Συγκεκριμένα θα χρησιμοποιήσουμε ένα πληκτρολόγιο PS/2 που θα αποτελέσει συσκευή εισόδου στο σύστημά μας. Θα πρέπει να διαβάζουμε κάποιους συγκεκριμένους χαρακτήρες και να τους απεικονίζουμε στα 7 segment. Αρχικά θα μελετήσουμε το πρωτόκολλο PS/2 για επικοινωνία με το πληκτρολόγιο.

Α. Πρωτόκολλο πληκτρολογίου

Περιγράφεται παρακάτω η ανάγνωση δεδομένων από ένα πληκτρολόγιο τεχνολογίας AT, η οποία είναι ακριβώς ίδια με αυτήν του PS/2 πρωτοκόλλου. Η περιγραφή περιορίζεται μόνο στα απαραίτητα για το project στοιχεία. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει και στα :

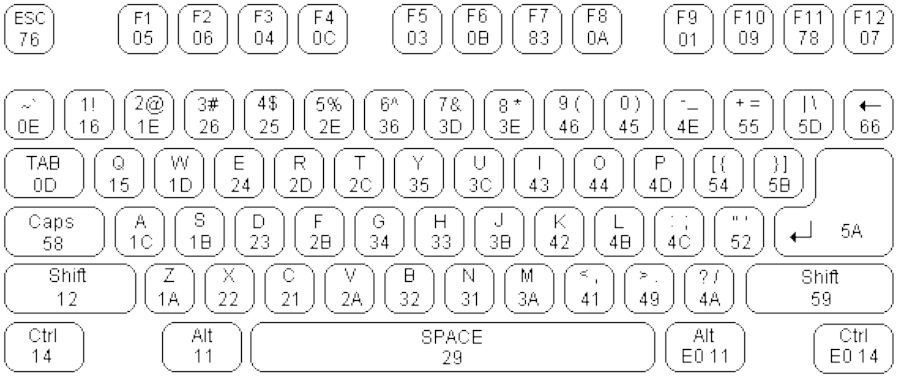
http://www.beyondlogic.org/keyboard/keybrd.htm

http://www.barcodeman.com/altek/mule/scandoc.php3

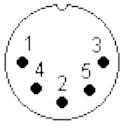
Σε κάθε πλήκτρο αντιστοιχεί ένας κωδικός 1 ή 2 ή 3 bytes που ονομάζεται κώδικας σάρωσης (scan code). Τα scan codes που μας ενδιαφέρουν στο παρόν project είναι :

Πλήκτρο	Scan Code (in HEX)
0	45
1	16
2	1E
3	26
4	25
5	2E
6	36
7	3D
8	3E
9	46

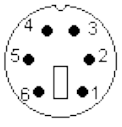
Οι κώδικες σάρωσης για ένα πληκτρολόγιο AT φαίνονται στη παρακάτω εικόνα.



Το πάτημα ενός πλήκτρου, ισοδυναμεί με τη δημιουργία και τη μεταφορά του αντίστοιχου κωδικού σάρωσης. Ο κωδικός σάρωσης θα συνεχίσει να παράγεται και να μεταδίδεται μέχρι να αφεθεί το πλήκτρο οπότε παράγεται ένας κωδικός αποτελούμενος από 2 bytes : το $F0_{16}$ και τον κωδικό σάρωσης. Συνεπώς ένα απλό πάτημα του "2", ισοδυναμεί με τη μεταφορά της πληροφορίας 1E F0 1E. Η μεταφορά των scan codes στο υπόλοιπο σύστημα γίνεται πάνω από μια σειριακή αρτηρία μέσω των γραμμών πληροφορίας (γραμμή KBD Data) όσο και ενός ρολογιού κωδικοποίησης (KBD Clock), μικρής συχνότητας (20 – 30 KHz). Τα σήματα που χρησιμοποιούνται σε 5 Pin DIN και PS/2 προσαρμογείς είναι όπως παρακάτω :



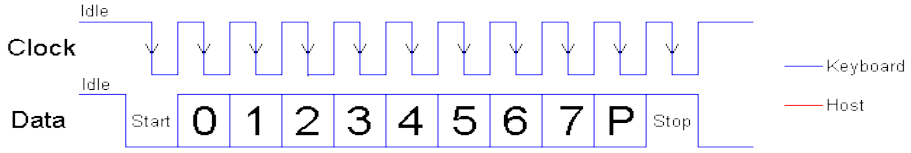
1. KBD Clock
2. KBD Data
3. N/C
4. GND
5. +5V (VCC)



1. KBD Clock
2. GND
3. KBD Data
4. N/C
5. +5V (VCC)
6. N/C

Το σειριακό πρωτόκολλο ανταλλαγής δεδομένων στην περίπτωση του πληκτρολογίου ακολουθεί τους εξής κανόνες :

- Οταν δεν ανταλλάσσονται δεδομένα η γραμμή δεδομένων (Data) είναι ανενεργή (στο λογικό 1).
- Για την αποστολή ενός byte δεδομένων απαιτείται η αποστολή ενός πακέτου 11 bits με χρονισμό που φαίνεται στην παρακάτω εικόνα.



- Το πρώτο δυαδικό ψηφίο της γραμμής δεδομένων είναι ένα ψηφίο εκκίνησης (start bit), ακολουθούμενο από τα 8 δυαδικά ψηφία του κώδικα σάρωσης, ένα ψηφίο περιττής ισοτιμίας και ένα δυαδικό ψηφίο τερματισμού (stop bit).
- Το πρώτο δυαδικό ψηφίο πληροφορίας που αποστέλλεται είναι το λιγότερο σημαντικό (τόσο για τον κώδικα σάρωσης, όσο και για το $F0_{16}$).
- Ο παραλήπτης της πληροφορίας για τη δειγματοληψία θα πρέπει να χρησιμοποιήσει την αρνητική ακμή του Clock. (Προσοχή αυτό δε σημαίνει ότι θα πρέπει στο σχεδιασμό σας να έχετε 2 ρολόγια, ένα του υπόλοιπου κυκλώματος και ένα για το πληκτρολόγιο).

Β. Υλοποίηση

Πριν τη συγγραφή του κώδικα, χρειάζεται να αναλυθούν περαιτέρω κάποια θέματα :

(1) Ο σχεδιασμός μας φαίνεται να έχει 2 ρολόγια, αυτό της αναπτυξιακής πλακέτας και αυτό του πληκτρολογίου. Ωστόσο, κάτι τέτοιο είναι απολύτως ανεπιθύμητο (μεγαλώνει το skew, χρησιμοποιούνται περισσότερα resources, είναι δύσκολο να προσδιοριστεί η συχνότητα λειτουργίας και είναι δύσκολος ο ακριβής συγχρονισμός μεταξύ υποσχεδιασμών με διαφορετικό χρονισμό). Επίσης, μιας και υπάρχει τάξεις μεγέθους διαφορά μεταξύ των συχνοτήτων αυτών των ρολογιών, υπάρχει περίπτωση ένα glitch στο αργό ρολόι να διαρκέσει δεκάδες κύκλους του γρήγορου. Για την αποφυγή όλων των παραπάνω, στον κώδικά μας πρέπει να έχουμε ένα μόνο ρολόι, αυτό της πλακέτας, το οποίο θα χρησιμοποιήσουμε και για δειγματοληψία του αργού (του πληκτρολογίου).

(2) Όπως αναλύθηκε παραπάνω, το γρήγορο πάτημα ενός χαρακτήρα, δημιουργεί την ακολουθία <scancode> <F0> <scancode>, ενώ το παρατεταμένο πάτημά του την ακολουθία <scancode><scancode> ... <scancode> <F0> <scancode>. Μια μεθοδολογία που μπορούμε να ακολουθήσουμε στο σχεδιασμό μας είναι να ανιχνεύουμε το <scancode> μόνο αμέσως το <F0>, έτσι ώστε να απεμπλακούμε από το φόρτο να κοιτάμε για πόσο πατήθηκε ένας χαρακτήρας ή να απορρίπτουμε τα 2 τελευταία σειριακά πακέτα κάθε πληκτρολόγησης.

(3) Το τελευταίο θέμα είναι τι γίνεται με τα υπόλοιπα πλήκτρα. Παρακάτω υποθέτουμε ότι γι' αυτά στην οθόνη μας όλα τα leds του 7 segment σβήνουν.

Μετά τα παραπάνω, ο κώδικας που χρειάζεται αρχικά να δοκιμάσουμε είναι ο :

(αρχείο 8/eight.v)

```
module kbd_protocol (reset, clk, ps2clk, ps2data, scancode);
 input       reset, clk, ps2clk, ps2data;
 output [7:0] scancode;
 reg    [7:0] scancode;
 // Synchronize ps2clk to local clock and check for falling edge;
 reg    [7:0] ps2clksamples; // Stores last 8 ps2clk samples
 always @(posedge clk or posedge reset)
  if (reset) ps2clksamples <= 8'd0;
   else ps2clksamples <= {ps2clksamples[7:0], ps2clk};
 wire fall_edge; // indicates a falling_edge at ps2clk
 assign fall_edge = (ps2clksamples[7:4] == 4'hF) & (ps2clksamples[3:0] == 4'h0);
 reg    [9:0] shift;   // Stores a serial package, excluding the stop bit;
 reg    [3:0] cnt;     // Used to count the ps2data samples stored so far
 reg          f0;      // Used to indicate that f0 was encountered earlier
/* A simple FSM is implemented here. Grab a whole package, check its parity validity and output it in
the scancode only if the previous read value of the package was F0 that is, we only trace when a
button is released, NOT when it is pressed. */
 always @(posedge clk or posedge reset)
  if (reset)
   begin
    cnt    <= 4'd0; scancode <= 8'd0; shift    <= 10'd0; f0 <= 1'b0;
   end
  else if (fall_edge)
     begin
```

```
            if (cnt == 4'd10) // we just received what should be the stop bit
             begin
              cnt <= 0;
              if ((shift[0] == 0) && (ps2data == 1) && (^shift[9:1]==1)) // A well received serial packet
               begin
                if (f0) // following a scancode of f0. So a key is released !
                 begin
                  scancode <= shift[8:1]; f0 <= 0;
                 end
                else if (shift[8:1] == 8'hF0) f0 <= 1'b1;
               end // All other packets have to do with key presses and are ignored
             end
            else
             begin
              shift <= {ps2data, shift[9:1]}; // Shift right since LSB first is transmitted
              cnt <= cnt+1;
             end
           end
endmodule

            module scan_2_7seg (scan, ss);
             input  [7:0] scan;
             output [7:0] ss;

             assign ss =     (scan == 8'h45) ? 8'b01111110 :
                             (scan == 8'h16) ? 8'b00110000 :
                             (scan == 8'h1E) ? 8'b01101101 :
                             (scan == 8'h26) ? 8'b01111001 :
                             (scan == 8'h25) ? 8'b00110011 :
                             (scan == 8'h2E) ? 8'b01011011 :
                             (scan == 8'h36) ? 8'b01011111 :
                             (scan == 8'h3D) ? 8'b01110010 :
                             (scan == 8'h3E) ? 8'b01111111 :
                             (scan == 8'h46) ? 8'b01111011 : 8'b10000000 ;
            endmodule

            module eight (reset, clk, ps2clk, ps2data, left);
             input      reset, clk;
             input      ps2clk, ps2data;
             output [7:0] left;
             wire   [7:0] scan;
             kbd_protocol kbd (reset, clk, ps2clk, ps2data, scan);
             scan_2_7seg  lft (scan, left);
            endmodule
```

Μερικές επιπλέον διευκρινίσεις για τον παραπάνω κώδικα. Ο σχεδιασμός μας αποτελείται από 2 υποσχεδιασμούς. Ο δεύτερος (scan_2_7seg) απλά διαβάζει ένα scancode και το μετατρέπει σε πληροφορία απεικόνισης σε 7 segment (θα χρησιμοποιήσουμε το αριστερό 7 segment της XST).

Ο πρώτος (kbd_protocol) υλοποιεί όλο το πρωτόκολλο ανάγνωσης από το πληκτρολόγιο. Χρησιμοποιούμε :

- τον καταχωρητή ps2clksamples για να κρατήσουμε τα τελευταία 8 δείγματα του ρολογιού του πληκτρολογίου. Παίρνουμε ένδειξη αρνητικής ακμής μόνον όταν αυτός έχει τη τιμή 11110000.
- έναν καταχωρητή ολίσθησης (shift) ο οποίος σε κάθε τέτοια αρνητική ακμή διαβάζει ένα δυαδικό ψηφίο δεδομένων.
- έναν μετρητή (cnt) για να διαπιστώσουμε πότε διαβάσαμε 11 ψηφία και εάν είναι όπως αναμενόταν (σωστά start – stop bit και σωστή ισοτιμία) τότε μόνο είναι είναι ένα σωστό σειριακό πακέτο με scancode υποψήφιο για έξοδο.

- ένα flag (f0) για να θυμόμαστε αν ο προηγούμενος scancode που ανιχνεύσαμε ήταν το f0. Μόνο σε αυτή την περίπτωση ανανεώνουμε την έξοδό μας.

Γ. Αρχείο φυσικών παραμέτρων

Μπορούμε πλέον να προχωρήσουμε στον ορισμό των pins που θα μας χρειαστούν. Προσοχή χρειάζεται στο ότι πρέπει να χρησιμοποιήσουμε το Excel αρχείο του XST+XSA. Για το reset θα επιλέξουμε το 1ο dip switch της XSA πλακέτας και για ρολόι το pin t9 που οδηγείται από το CPLD (CLKA). Έτσι το αρχείο μας διαμορφώνεται στα αναγραφόμενα στο **(αρχείο 8/eight.ucf)**

Δ. Ζητούμενα

(1) Εφαρμόστε όλη τη ροή σύνθεσης και υλοποίησης του κυκλώματος όπως αυτή αναπτύχθηκε στην εργαστηριακή άσκηση 7 ώστε να καταλήξετε σε ένα λειτουργούν πρωτότυπο. Φωνάξτε τον επιβλέποντα, για να του το δείξετε. Κρατήστε συνεχώς ένα αριθμητικό πλήκτρο πατημένο και δείτε τι συμβαίνει. Εξηγήστε το γιατί.

(2) Επαυξήστε τις δυνατότητες του σχεδιασμού σας με τα ακόλουθα :

- Λειτουργία προηγούμενου πλήκτρου : Χρησιμοποιείστε και το δεύτερο 7-segment της XSA για να απεικονίζετε το προηγούμενο πατηθέν πλήκτρο.
- Λειτουργία calculator : Δώστε τη δυνατότητα ανίχνευσης και των συμβόλων των αριθμητικών πράξεων με τους ακόλουθους scancodes (εισάγονται από το αριθμητικό πληκτρολόγιο πλην του /):

Πλήκτρο	Scan Code (in HEX)
+	79
-	7B
*	7C
/	4A

Όταν ανιχνευθεί ένας τέτοιος χαρακτήρας, εκτελείται η πράξη <πρόσφατο στοιχείο> <πράξη><προηγούμενο στοιχείο> και το αποτέλεσμα (για την πράξη * μόνο το least significant digit, για την / μόνο το ακέραιο μέρος) εμφανίζεται στο 7 segment της XSA.