ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Ενσωματωμένα Συστήματα

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No 11

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

1. Σκοπός της άσκησης

- Ανάπτυξη ενσωματωμένων συστημάτων με τον μικρο-επεξεργαστή Picoblaze[1].
- Δημιουργία System-on-Chip με τον picoblaze.

(A) 10 ερώτηση

(C) 6 ασκήσεις/προγράμματα

2 bonus

Για το εργαστήριο αυτό απαιτείται το λογισμικό Xilinx ISE 14.7 που βρίσκεται στη διεύθυνση:

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

2. Παραδοτέα

- **Παραδοτέο C1: left_right_leds.vhd** [επίσης τα σχετιζόμενα παραδοτέα: **(a1)** left_right_leds.ucf, **(a2)** screenshot από το σχηματικό *(schematic)*, και **(a3)** από το device utilization summary].
- **Παραδοτέο C2: left_right_leds.vhd .**
- **Παραδοτέο C3: left_right_leds_embedded_system.vhd** [καθώς και τα σχετιζόμενα παραδοτέα: **(a4)** screenshot από το σχηματικό *(schematic)*, και **(a5)** από το "device utilization summary" "Resource Estimation"].
- **Παραδοτέο C4:** Κώδικας VHDL [επίσης τα αρχεία: **(a6)** ucf, **(a7)** screenshot από το σχηματικό *(schematic),* και **(a8)** από το "device utilization summary" "Resource Estimation"].
- **Παραδοτέο C5: control.vhd .**
- **Παραδοτέο C6: left_right_leds-lcd.ucf** [**(a9)** screenshot από το σχηματικό *(schematic)*, **(a10)** από το "device utilization summary" ή το "Resource Estimation"].

3. Ο επεξεργαστής Picoblaze

Είναι ένας μικρο-επεξεργαστής soft-core, δηλαδή επεξεργαστής που περιγράφεται σε γλώσσα HDL και μπορεί να ενσωματωθεί σε μια πλακέτα FPGA. Έχει αναπτυχθεί από τη Xilinx και είναι διαθέσιμος για μεταφόρτωση από τη διεύθυνση:

https://www.xilinx.com/products/intellectual-property/picoblaze.html

1Χρησιμοποιήθηκε υλικό από το εκπαιδευτικό σεμινάριο "Reconfigurable Architectures and Tools (K. Siozios & D. Diamantopoulos), NTUA".

Συνοπτικά Χαρακτηριστικά:

- ISC 8-bit Microcontroller Reference Design.
- Supports Virtex/Virtex-E, Spartan™-II/E, Virtex-II/Pro devices.
- Very small size – only 84 Virtex-II slices, 33% of XC2V40.
- Up to 66 MIPS in Virtex™-II , 100 MIPS in Virtex ™-IV.
- Everything in FPGA - no external components required.
- Highly integrated for implementing non-time critical state machine.
- Predictable fast interrupt response.
- Originally named KCPSM which stands for "Constant(K) Coded Programmable State Machine" (formerly "Ken Chapman's PSM").
- It is freely distributed under the BSD license.

4. Παραδείγματα χρήσης του επεξεργαστή Picoblaze

Παραδείγματα χρήσης του επεξεργαστή Picoblaze:

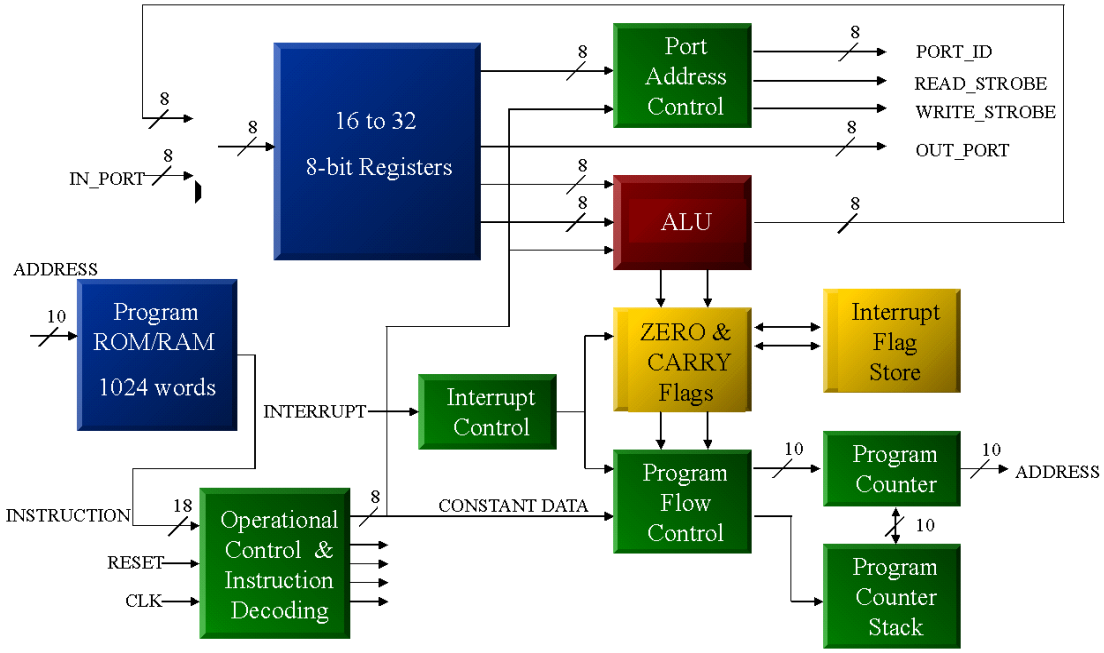
- Front panel switches and displays for Set Top Box.
- Dynamic Loop Bandwidth Multiplexer for frame to frame analysis.
- Link layer of IEEE 1394 Interface.
- Microcontroller for Compact Flash Programming engine.
- DECT Radio/Repeater.
- PCI board programming controller.
- Communications controller.
- Preprocessing for network processor.
- Motor controller.
- Programmable power supply controller.
- Part of Media Access Controller.
- Controller in broadcast video equipment.
- and many, many more…

5. Αρχιτεκτονικά χαρακτηριστικά του επεξεργαστή Picoblaze

Χαρακτηριστικά:

- 32 General Purpose 8-Bit Registers.
- Arithmetic Logic Unit (ALU):
  - Performed with operands from any of the registers.
  - Operations include logical, arithmetic, shift and rotate operations.
- Flags (Program Flow Control):
  - ZERO and CARRY flags.
- Input/Output:
  - 256 input and 256 output ports (8 bits).
  - Shift data in and out of the unit.
- Interrupt:
  - Single interrupt that can be augmented by logic.
- Predictable execution rate:
  - Two clock cycles/instruction.

6. Αρχιτεκτονική του Picoblaze



7. Το πλήρες σετ εντολών (ISA) του Picoblaze

(Η ανάλυση των εντολών βρίσκεται στο εγχειρίδιο χρήσης **KCPSM3_Manual.pdf** στα έγγραφα του εργαστηρίου)

'X' and 'Y' refer to the definition of the storage registers 's' in the range 0 to F.
'kk' represents a constant value in the range 00 to FF.
'aaa' represents an address in the range 000 to 3FF.
'pp' represents a port address in the range 00 to FF.
'ss' represents an internal storage address in the range 00 to 3F.

Program Control Group

```
JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC
```

Note that call and return supports up to a stack depth of 31.

Arithmetic Group

```
ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk

ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY
```

Interrupt Group

```
RETURNI ENABLE
RETURNI DISABLE

ENABLE INTERRUPT
DISABLE INTERRUPT
```

Logical Group

```
LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk

LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY
```

Storage Group

```
STORE sX,ss
STORE sX,(sY)
FETCH sX,ss
FETCH sX,(sY)
```

Shift and Rotate Group

```
SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX

SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX
```

Input/Output Group

```
INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)
```

8. Περιγραφή της άσκησης

Σε αυτό το εργαστήριο θα γνωρίσετε τον επεξεργαστή picoblaze. Το εργαστήριο αυτό αποτελείται από 4 ασκήσεις. Οι 2 ασκήσεις εκτελούν κάποια λειτουργία χρησιμοποιώντας μόνο hardware, ενώ οι άλλες 2 εκτελούν την ίδια λειτουργία χρησιμοποιώντας hardware + software.

Αρχικά θα υλοποιήσετε μια περιστροφή LED (led rotation) χρησιμοποιώντας **(a)** μόνο VHDL χωρίς επεξεργαστή και **(b)** τον PICOBLAZE και το κατάλληλο πρόγραμμα σε assembly. Στη συνέχεια θα επαναληφθεί η άσκηση με το να τυπωθεί το Hello World στην LCD οθόνη χρησιμοποιώντας **(c)** μόνο VHDL χωρίς επεξεργαστή και **(d)** τον PICOBLAZE και το κατάλληλο πρόγραμμα σε assembly.

Σημειώστε ότι και στα (a),(c), αλλά και στα (b),(d) χρησιμοποιείται η VHDL γλώσσα. Η διαφορά των (α),(c) με τα (b),(d) είναι ότι τα πρώτα είναι μια περιγραφή καθαρά υλικού *(hardware implementation)*, ενώ τα υπόλοιπα είναι μια περιγραφή που χρησιμοποιεί υλικό *(τον επεξεργαστή)* και λογισμικό *(το πρόγραμμα σε assembly) (hardware & software implementation)*.

Θα χρησιμοποιηθεί η πλακέτα SPARTAN 3A που υπάρχει στο εργαστήριο και λειτουργεί αρμονικά με τον επεξεργαστή picoblaze.

9. Στοιχεία πλακέτας

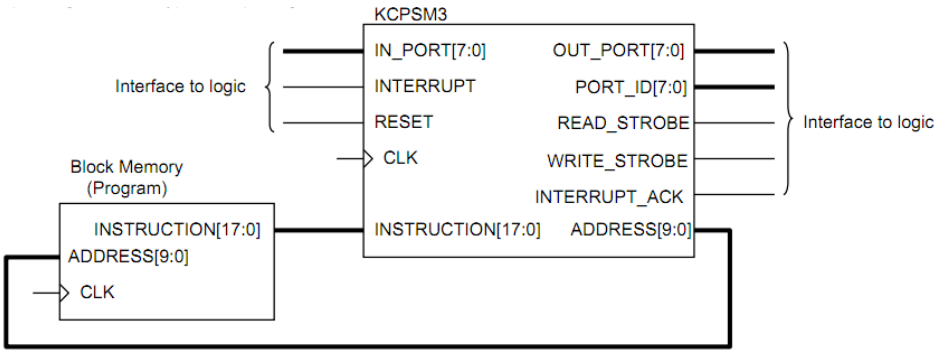
Το FPGA της πλακέτας Spartan που υπάρχει στο εργαστήριο, φέρει τον κωδικό πάνω στο ολοκληρωμένο κύκλωμα FPGA, XC3S700A και έχει τα παρακάτω τεχνικά στοιχεία (από το αρχείο ds529.pdf):

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits(1)	Block RAM bits(1)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	CLBs	Slices						
XC3S50A	50K	1,584	16	12	176	704	11K	54K	3	2	144	64
XC3S200A	200K	4,032	32	16	448	1,792	28K	288K	16	4	248	112
XC3S400A	400K	8,064	40	24	896	3,584	56K	360K	20	4	311	142
XC3S700A	700K	13,248	48	32	1,472	5,888	92K	360K	20	8	372	165
XC3S1400A	1400K	25,344	72	50	2,816	11,264	176K	576K	32	8	502	227

Η πλακέτα που έχει το συγκεκριμένο chip έχει πλήθος περιφερειακών, όπως:

- LCD 2x16 characters.
- 4 slide switches.
- 4 push buttons.
- 8 LEDs.
- 1 VGA output.
- 2 serial ports.
- 1 PS/2.
- 1 10/100 Ethernet PHY.
- A2D and D2A converters.
- Additional expansion ports.

Περισσότερες πληροφορίες μπορείτε να βρείτε στο datasheet **ds529.pdf** που βρίσκεται στα αρχεία του μαθήματος. Τέλος, παρουσιάζεται ο μικροεπεξεργαστής picoblaze *(KCPSM3)*, οι διασυνδέσεις εισόδου, εξόδου και η μνήμη εντολών *(Program Block memory)*. Αν θέλετε να συνδέσετε ένα περιφερειακό εισόδου *(π.χ. κουμπιά)* ή περιφερειακό εξόδου *(π.χ. led)* θα τα διασυνδέσετε με τον κατάλληλο κώδικα VHDL είτε στις θύρες εισόδου ή εξόδου αντίστοιχα, όπως θα δείτε στις επόμενες ασκήσεις.



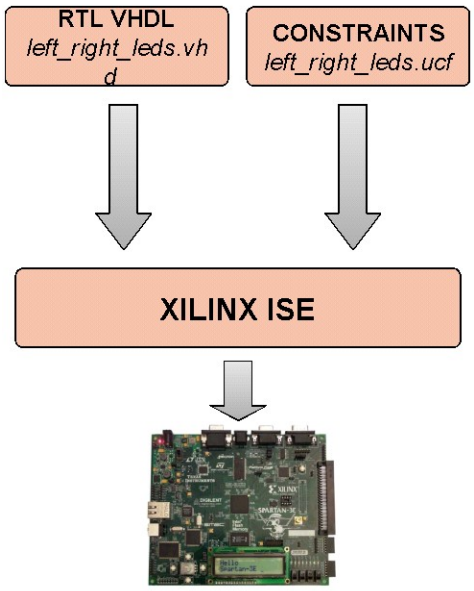
Στο εγχειρίδιο **KCPSM3_manual.pdf** στα έγγραφα του εργαστηρίου μπορείτε να διαβάσετε περισσότερες πληροφορίες για τον μικροεπεξεργαστή Picoblaze.

(BONUS1: Αν υλοποιηθεί το εργαστήριο στην πλακέτα ALTERA DE2, η άσκηση βαθμολογείται με +100%).

10. Περιστροφή LED με υλοποίηση hardware

Θα κατασκευάσετε το αρχείο **left_right_leds.vhd ,** το οποίο θα εκτελεί την περιστροφή των led, και θα το χρησιμοποιήσετε μαζί με το αρχείο των περιορισμών για την πλακέτα Spartan 3A/3E.

➢ **Using pure VHDL**



- Μεταφορτώστε το αρχείο **left_right_leds-Spartan3E.ucf** , το οποίο έχει τα constrains για την πλακέτα Spartan 3E και τροποποιήστε τα LOC, ώστε να ανταποκρίνονται στα constrains της πλακέτας Spartan 3A. Μπορείτε να χρησιμοποιήστε το σχετικό αρχείο προηγούμενου εργαστηρίου. Αποθηκεύστε το αρχείο με το όνομα **left_right_leds.ucf .**

  (αν χρησιμοποιείτε την πλακέτα Spartan3A θα πρέπει να συμβουλευτείτε το αρχείο s3astarter.ucf που φέρει όλες τις αντιστοιχήσεις ακροδεκτών ή το αρχείο ug334.pdf ενότητες Discrete LEDs, Clock Sources, Push-Button Switches. Αν εργάζεστε σε 3E δεν απαιτείται κάποια αλλαγή)

- Μεταφορτώστε το αρχείο **left_right_leds-incomplete.vhd** , μελετήστε το προσεκτικά και τροποποιήστε τις τοποθεσίες που είναι σημειωμένες με το **TODO** (Τοποθεσίες TODO1 έως TODO5), ώστε να λειτουργεί σωστά. Αποθηκεύστε το αρχείο με το όνομα **left_right_leds.vhd .**

  Σημειώσεις:

  - TODO1: Να ορίσετε τα κουμπιά btn_north, btn_west, btn_east, τα led, και το clk. Χρησιμοποιήστε τα ίδια ονόματα που έχετε ορίσει στο ucf file, ώστε να γίνει αυτόματα η αντιστοίχηση. Μπορείτε να τα δηλώσετε ως std_logic.
  - TODO2: Ορίστε ως signals τα παραπάνω κουμπιά με το επίθεμα _in
  - TODO3:Ορίστε ως signals τα παραπάνω κουμπιά με το επίθεμα _delayed
  - TODO4: Μελετήστε τις τιμές που λαμβάνει το led pattern στις άλλες 2 καταστάσεις και συμπληρώστε την τιμή που θα έπρεπε να πάρει για να γίνει η δεξιά ολίσθηση.

- TODO5: Να μεταφέρετε την τιμή του led_pattern στην κατάλληλη έξοδο της οντότητας.

- Δημιουργήστε ένα νέο project στο Xilinx IDE ISE με τις κατάλληλες ρυθμίσεις (επιβεβαιώστε κυρίως την ορθότητα στην καταχώρηση Evaluation Development Board) για την πλακέτα του εργαστηρίου. Στη συνέχεια τοποθετήστε τα 2 παραπάνω αρχεία, ορίστε , κάντε σύνθεση και δημιουργία του bitstream, προγραμματίστε την πλακέτα (Configure Target Device, το οποίο ενεργοποιεί το Impact. Εκεί Boundary Scan→Initialize chain. Assign το .bit αρχείο που είναι το bitstream στο FPGA (αν σας ερωτήσει μην το κάνετε assign στο SPI/PROM) και έπειτα Operations→Program). Επιβεβαιώστε την ορθή λειτουργία. Πατήστε τα 3 κουμπιά που έχει το board και δείτε αν αλλάζουν τα led με το σωστό τρόπο.

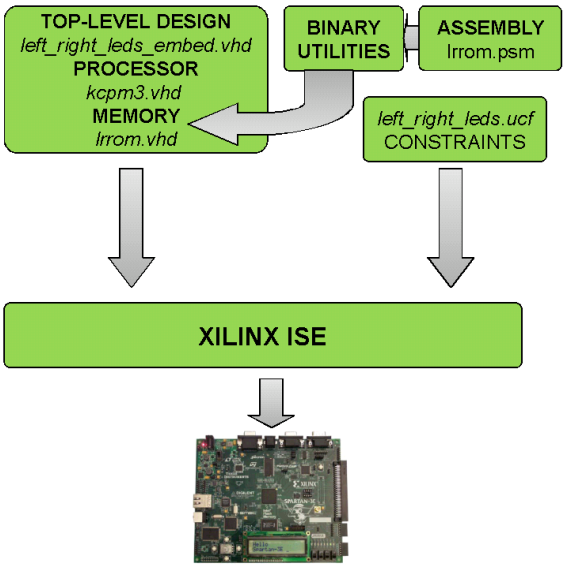
Παραδοτέα: Τα αρχεία (a1) **left_right_leds.ucf,** (c1) **left_right_leds.vhd,** (a2) screenshot από το σχηματικό *(schematic)*, και (a3) από το device utilization summary.

TIP: Μπορείτε να αποθηκεύσετε στο impact το project σας (πατήστε SAVE), να δείτε πάνω στο title bar τη διαδρομή που έχει αποθηκευτεί το αρχείο, στη συνέχεια να το κλείσετε, να μεταβείτε στο ISE, στο processes, ανοίγετε το Configure Target Device, δεξί-κλικ στο Manage Configuration, επιλέγετε Process Properties, και στο value επιλέγετε το project file, ώστε να επιλέγεται αυτόματα το συγκεκριμένο impact αρχείο.

11. Περιστροφή LED με υλοποίηση hardware + software

Θα χρησιμοποιήσετε την περιγραφή του picoblaze που βρίσκεται στο αρχείο **kcpm3.vhd .** Θα γράψετε την assembly για την εκτέλεση της συγκεκριμένης λειτουργίας *(led rotation)* στο αρχείο **lrrom.psm**. Όπως σε ένα πραγματικό ενσωματωμένο σύστημα, το πρόγραμμα θα πρέπει να τοποθετηθεί σε μια μνήμη ROM που θα συνδέεται με τον μικρο-επεξεργαστή για να τον τροφοδοτεί με εντολές. Θα πρέπει λοιπόν κάθε εντολή assembly που έχετε στο αρχείο psm να τη μετατρέψετε στον αντίστοιχο κώδικα μηχανής, ο οποίος θα πρέπει να τοποθετηθεί σε ένα VHDL module που θα λειτουργεί ως μνήμη ROM. Αν και μπορείτε να κάνετε εσείς χειρωνακτικά αυτή τη διαδικασία, μπορείτε να γλυτώσετε χρόνο με το να χρησιμοποιήσετε ένα ειδικό πρόγραμμα για το picoblaze που δέχεται ως είσοδο ένα αρχείο psm και εξάγει ένα αρχείο VHDL που περιγράφει μια μνήμη ROM με τον κώδικά μας. Το αρχείο αυτό που θα εξαχθεί θα το ονομάσετε **lrrom.vhd**. Χρησιμοποιώντας **(α)** την περιγραφή του επεξεργαστή *(hardware)* **kcpm3.vhd** και **(β)** την περιγραφή της μνήμης ROM *(software)* **lrrom.vhd** θα κατασκευάσετε το αρχείο σχεδιασμού που θα χρησιμοποιεί τα 2 παραπάνω components left_right_leds_embed.vhd, και το οποίο θα εκτελεί την περιοδική ενεργοποίηση των LED.

➢ **Using PicoBlaze & Assembly**



- Μεταφορτώστε το αρχείο **kcpsm3.vhd** το οποίο περιγράφει τον επεξεργαστή picoblaze. Αυτό το αρχείο δε θα το τροποποιήσετε.
- Χρησιμοποιήστε το αρχείο των περιορισμών **left_right_leds.ucf** από την προηγούμενη άσκηση. Αυτό το αρχείο δε θα το τροποποιήσετε.
- Μεταφορτώστε το αρχείο με το όνομα **left_right_leds_embed-incomplete.vhd** το οποίο θα αποτελεί την περιγραφή του SoC, στο οποίο περιέχονται ως components, ο επεξεργαστής, η μνήμη ROM προγράμματος, η διασύνδεση με τις θύρες εισόδου, και η διασύνδεση με τις θύρες εξόδου. Τροποποιήστε το αρχείο στα σημεία που επισημαίνονται ως TODO σύμφωνα με τα σχόλια και αποθηκεύστε το αρχείο με το όνομα **left_right_leds_embedded_system.vhd .**

  Σημειώσεις:

  - TODO1: Να ορίσετε τα κουμπιά btn_north, btn_west, btn_east, τα led, και το clk. Χρησιμοποιήστε τα ίδια ονόματα που έχετε ορίσει στο ucf file, ώστε να γίνει αυτόματα η αντιστοίχηση. Μπορείτε να τα δηλώσετε ως std_logic.
  - TODO2: Από τη δήλωση του entity στο KCPSM3.vhd να μεταφέρετε τα σήματα εισόδου/εξόδου για να γίνει instantiate το component kcpsm3.

- TODO3: Από το αρχείο που θα δημιουργηθεί στο επόμενο βήμα με το πρόγραμμα, να μεταφέρετε τα 2 ονόματα του entity για να γίνει instantiate το component lrrom. Εναλλακτικά, μπορείτε να βρείτε τα ονόματα των σημάτων με προσεκτική ανάγνωση των αρχείων ROM*** που βρίσκονται στον ίδιο κατάλογο με το picoblaze.
- TODO4: Να γράψετε τον αριθμό της θύρας (από 00 έως 255) εισόδου (port_id) που θα χρησιμοποιηθεί για την ανάγνωση των κουμπιών. Θα είναι 8 bit, οπότε τα 5 πρώτα bit θα είναι 0 και στη συνέχεια θα προστεθούν τα 3 bit, των θυρών west/north/east. Θα πρέπει να γράψετε τον δυαδικό αριθμό 8bit, π.χ. 0000XXXX, γιατί είναι std_logic_vector(7 downto 0).
- TODO5: Να γράψετε τον αριθμό της θύρας (από 00 έως 255) εξόδου (port_id) που θα χρησιμοποιηθεί για την εμφάνιση των led. Θα πρέπει να γράψετε τον δυαδικό αριθμό 8bit, π.χ. 0000XXXX, γιατί είναι std_logic_vector(7 downto 0).

- Μεταφορτώστε το αρχείο **lrrom-incomplete.psm** το οποίο περιέχει την assembly που θα χρησιμοποιήσετε για τη δημιουργία του αρχείου της ROM. Τροποποιήστε τις κατάλληλες γραμμές που έχουν σημανθεί με το TODO. Αποθηκεύστε το νέο αρχείο με το όνομα **lrrom.psm .**

  Σημειώσεις:

  - TODO1: Τοποθετήστε την πόρτα εξόδου στο LED_port, ίδιο αριθμό με αυτό που έχετε χρησιμοποιήσει προηγουμένως. Ομοίως για το BTN_port, τοποθετήστε την πόρτα εισόδου που επιλέξατε προηγουμένως. Για τα κουμπιά BTN_east, BTN_north, BTN_west, διαβάστε πιο bit ενεργοποιούν όταν πατηθούν, και υπολογίστε τη δεκαεξαδική τιμή του αριθμού που έχει όλα 0 και μόνο ένα '1' στο bit που αναφέρεται. Π.χ. αν ενεργοποιείται το bit 7, τότε ο δεκαδικός αριθμός είναι 127 και ο δεκαεξαδικός, 1F.
  - TODO2: Επιλέξτε τον καταχωρητή που φέρει το pattern που θα πρέπει να γραφεί στην πόρτα των LED. Διαβάστε προσεκτικά την assembly και κυρίως τα σχόλια.
  - TODO3: Διαβάστε προσεκτικά την assembly και βρείτε τον καταχωρητή που πρέπει να τοποθετήσετε.
  - TODO4: Τοποθετήστε τη σωστή ετικέτα (που φέρει άνω/κάτω τελεία :).
  - TODO5: Τοποθετήστε τις σωστές ετικέτες.

- Μεταφορτώστε τα αρχεία KCPSM3.EXE και ROM_form.vhd τα οποία θα βοηθήσουν στη δημιουργία του αρχείου lrrom.vhd. Τοποθετήστε αυτά τα 2 αρχεία, όπως και το αρχείο που lrrom.psm, σε έναν κατάλογο που μπορείτε να μεταβείτε εύκολα από τη γραμμή εντολών (παράθυρο cmd). Από τη γραμμή εντολών (**cmd.exe**) μεταβείτε στον κατάλογο που έχετε αποθηκεύσει τα αρχεία με την εντολή cd και δώστε την εντολή **KCPSM3.EXE lrrom.psm**. Δεν μπορείτε να χρησιμοποιήσετε άλλη έκδοση assembler (π.χ. **KCPSM6.EXE**) γιατί υπάρχει ασυμβατότητα στα opcodes και μπορεί να φαίνεται ότι λειτουργεί, αλλά το τελικό machine code είναι ασύμβατο με το picoblaze3 (το picoblaze3 είναι το μόνο που υποστηρίζει spartan 3). Σε περίπτωση που σας αναφέρει το ΛΣ ότι δε μπορεί να εκτελεστεί σε 64bit windows, χρησιμοποιήστε το πρόγραμμα dosbox portable (θα σας φανούν χρήσιμες οι εντολές mount και cd), ενώ προτείνετε να τοποθετήσετε τα αρχεία του picoblaze σε ένα εύκολα προσβάσιμο κατάλογο, όπως το c:\temp.

  Αν εκτελεστεί χωρίς σφάλμα *(παρατηρήστε προσεκτικά την έξοδο του προγράμματος και να δείτε αν αναφέρει 0 errors)* θα δείτε στο τέλος ότι έχει δημιουργηθεί με επιτυχία το αρχείο **lrrom.vhd** *(όπως και άλλα αρχεία)*.

- Δημιουργήστε ένα νέο project στο Xilinx IDE με τις κατάλληλες ρυθμίσεις για την πλακέτα του εργαστηρίου, κάντε σύνθεση και δημιουργία του bitstream, προγραμματίστε την πλακέτα και επιβεβαιώστε την ορθή λειτουργία.

Παραδοτέα: Τα αρχεία (c2) **left_right_leds.vhd**, (c3) **left_right_leds_embedded_system.vhd**, (a4) screenshot από το σχηματικό (schematic), και (a5) από το “device utilization summary” “Resource Estimation”.

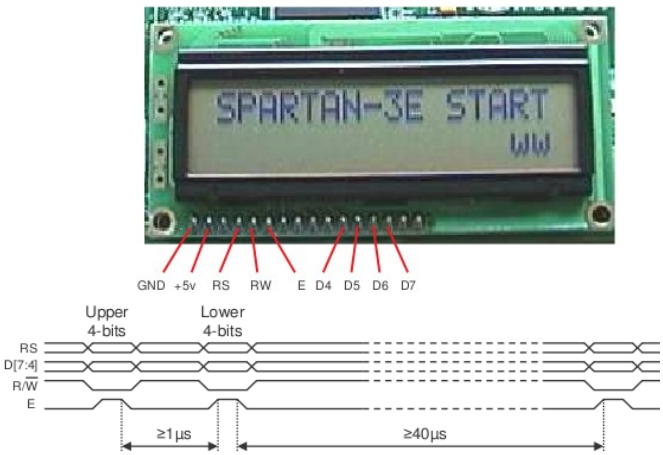
12. Εμφάνιση μηνύματος hello world στην LCD οθόνη με hardware υλοποίηση

Σε αυτό το εργαστήριο θα χρησιμοποιήσουμε το περιφερειακό LCD 2x16 της πλακέτας Spartan.

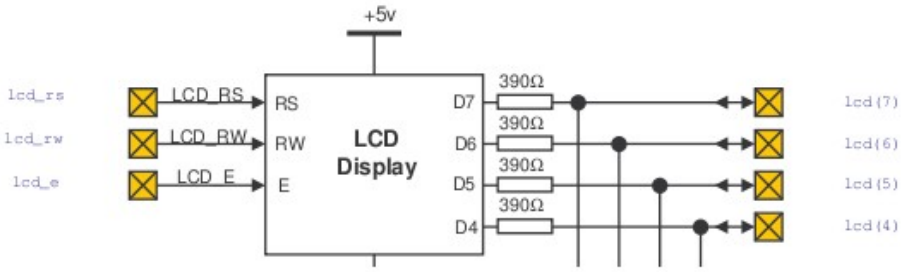
12.1 Οδηγίες Χρήσης LCD

Η LCD της πλακέτας χρησιμοποιείται ως εξής:

- The board is set up to use the 4-wire/8-wire (both supported) data interface to the LCD character module.
- After initial display communication is established, all data transfers are 8-bit ASCII character codes, data bytes or 8-bit addresses.
- Each 8-bit transfer obviously has to be decomposed into two 4-bit or one 8-bit transfer which must be spaced by at least 1µs.
- Following an 8-bit write operation, there must be an interval of at least 40µs before the next communication.



Υλοποίηση με μεταφορές των 4bit/8bit (για 4bit μεταφορές τα LCD_DB<3:0> βρίσκονται στο HIGH:



Caution! When using four-bit mode, the FPGA must drive the LCD_DB<3:0> signals High.

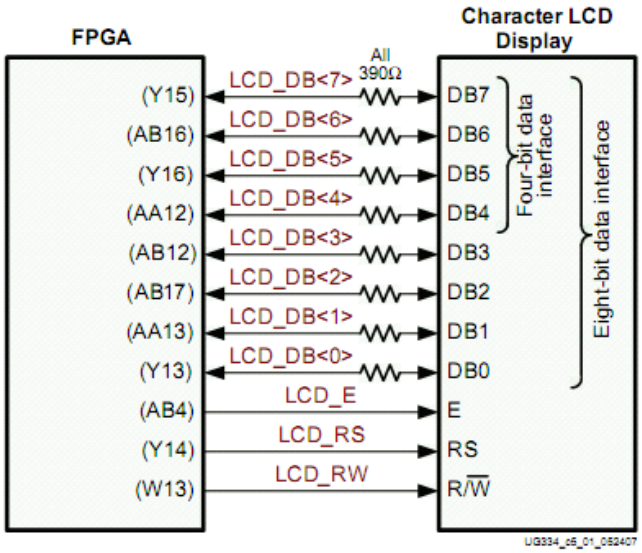


Figure 5-1: **Character LCD Interface**

Περισσότερες πληροφορίες μπορείτε να δείτε στο datasheet **ug334.pdf** για το Spartan 3 στα έγγραφα του εργαστηρίου, στο κεφάλαιο “Character LCD Screen / Chapter 5”. Συγκεκριμένα, στον πίνακα 5.1: Character LCD Interface, εμφανίζονται τα FPGA pins που συνδέονται στο LCD module και πως πρέπει να χρησιμοποιηθούν.

Το LCD της πλακέτας είναι πάρα πολύ σημαντικό περιφερειακό, γιατί μπορεί να εμφανίσει πλήθος ASCII πληροφοριών κατά την εκτέλεση.

Σε αυτή την ενότητα, θα ελέγξουμε το περιφερειακό χρησιμοποιώντας **MONO** VHDL (hardware λογική).

- Να δημιουργήσετε μια hardwire VHDL υλοποίηση *(μπορείτε να χρησιμοποιήσετε το υλικό που έχετε αναπτύξει σε προηγούμενα εργαστήρια)*, η οποία θα κάνει τα εξής:
    - Θα εμφανίζει και θα κάνει SCROLL στην οθόνη LCD το μήνυμα SPARTAN-3A STARTER KIT " και " www.xilinx.com " .
    - SW0 turns on/off LD0.
    - SW1 turns on/off LD1.
    - SW2 turns on/off LD2.
    - SW3 turns on/off LD3.
    - BTN East turns on/off LD4.

- BTN South turns on/off LD5.
- BTN North turns on/off LD6.
- BTN West turns on/off LD7.
- Single LED is moved left or right by rotation of control
- Pressing centre button of rotary encoder toggle mode of LEDs.

12.2 Βήματα Υλοποίησης

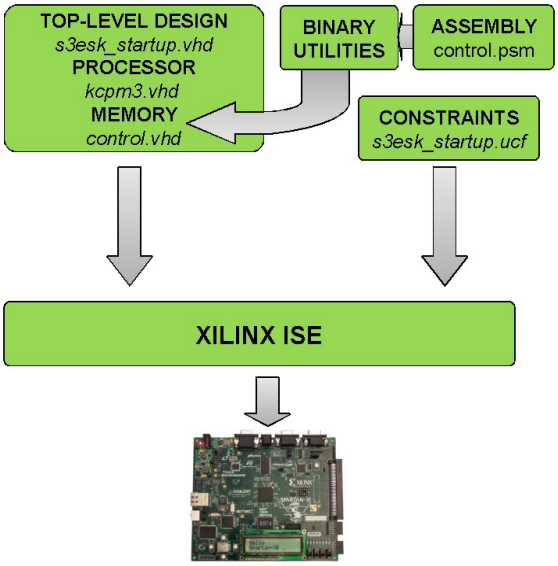
- Δημιουργήστε ένα νέο project που στοχεύει στην πλακέτα που έχετε.
- Δημιουργήστε ή τροποποιήστε κατάλληλα το left_right_leds-lcd-Spartan3E.ucf το οποίο είναι το αρχείο των περιορισμών και τον συνδέσεων των θυρών των VHDL αρχείων μας, με τα pin του FPGA, και προσθέστε το στο project. Σε αυτό το αρχείο θα πρέπει να ορίζονται:
  - "clk"
  - "lcd_e", "lcd_rs", "lcd_rw", "lcd_dbXX" (για το LCD)
  - "btn_east", "btn_north", "btn_south", "btn_west" (για τα κουμπιά)
  - "rot_a", "rot_b", "rot_center" (για το rotary encoder)
  - "sw<0> … sw<3>" (για τους 4 μηχανικούς διακόπτες)
  - led (για τα 8 led της πλακέτας)
- Δημιουργήστε ένα αρχείο VHDL (δεξί κλικ στο project Hierarcy→ New Source→ VHDL Module) που έχει στο entity δηλωμένα όλα τα παραπάνω σήματα. Αν τα σήματα είναι μέρος συνόλου, τότε θα δηλώσετε μια φορά το σήμα ως std_logic_vector. Π.χ. τα sw<0>..sw<3> θα είναι
  ```
  sw: in std_logic_vector(3 downto 0);
  ```
- Πατήστε διπλό κλικ στο synthesize για έλεγχο ορθότητας, το οποίο δε θα πρέπει να εμφανίσει Error (μόνο warnings).
- Στην περιγραφή της αρχιτεκτονικής θα κάνετε τα εξής:
  - Θα κάνετε instantiate το component που σας δίνεται στο lcd_spartan.vhd . Μπορείτε να μελετήσετε το datasheet της πλακέτας για να δείτε πως πρέπει να αρχικοποιείται η LCD, τους χαρακτήρες ελέγχου που πρέπει να στείλετε, και το character map. Σε αυτό το αρχείο να τροποποιήσετε την ακολουθία των Byte, ώστε να τυπώνει είτε το κείμενο που ζητήθηκε αρχικά, είτε το ονοματεπώνυμό σας.
  - Θα κάνετε instantiate το component rottary_encoder.vhd το οποίο έχει ως έξοδο πότε έχει συμβεί κάποιο rotary event (detent) και την κατεύθυνση (dir). Μπορείτε να δημιουργήσετε και μόνοι σας τη λογική για αυτό το module, αλλά θα διαπιστώσετε ότι λόγω μηχανικών glitches, δε θα ανιχνέυεται σωστά η φορά. Το module που σας δίνεται έχει υλοποιήσει τις επίσημες οδηγίες της xilinx για τους rotary encoders που βρίσκεται στο κείμενο s3esk_rotary_encoder_interface.pdf .
  - Αντιγράψτε από την προηγούμενη άσκηση τον κώδικα της αρχιτεκτονικής. Θα τον τροποποιήσετε με τις παρακάτω οδηγίες:
    - θα τοποθετήσετε τα port map για τα 2 components.
    - Θα ορίσετε τα σήματα με τα ίδια ονόματα στο entity με τις καταλήξεις _in και _delayed.
    - Θα τοποθετείτε τις τιμές από τις εισόδους σε αυτά τα σήματα (μέσα στο button_ctrl_process).
    - Θα τροποποιήσετε τη γραμμή 'btn_ctrl <= ' ώστε να υπάρχει η μετακίνηση όταν ανιχνεύεται event από το rotary encoder, ώστε μαζί με το σήμα της κατεύθυνσης θα θέτει 10 ή 01 ή αν πατηθεί το κεντρικό κουμπί του rotary encoder (το χειρίζεστε ως απλό κουμπί) θα θέτει 11, διαφορετικά 00.

- Θα τροποποιήσετε το btn_event ώστε να ενεργοποιείται σε κάθε περίπτωση που υπάρχει είσοδος. Για παράδειγμα θα έχετε είσοδο αν υπάρχει rotary event ή αν το κουμπί του rotary έχει άλλη τιμή (rot_center_in /= rot_center_delayed) ή οι διακόπες έχουν άλλη τιμή (sw_in /= sw_delayed) κ.ο.κ.
- Μέσα στο led display θα προσθέσετε με παρόμοιο τρόπο τα elsif των καινούργιων καταστάσεων. Π.χ. αν θέλετε στο btn_east να ανάψει ή να σβήσει το αντίστοιχο led, μέσα στο if θα δώστε: led_pattern(4)<=led_pattern(4) xor '1';
- Σε κάθε elsif θα πρέπει να έχετε πάντα την αρχικοποίηση του counter (όπως ακριβώς στον αρχικό κώδικα).
- Τα sw είναι διακόπτες που βρίσκονται σε μορφή πίνακα, οπότε αν αλλάξει κάποιο από αυτά, μπορείτε να κάνετε led_pattern(3 downto 0)<=sw(3 downto 0);

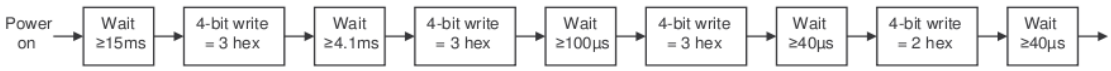
- Αφού ολοκληρώσετε όλες τις καταστάσεις, το αποθηκεύετε, το κάνετε compile/synthesis και προγραμματίζετε το board για την επιβεβαίωση της ορθής λειτουργίας.

Παραδοτέα: Τα αρχεία (a6) **ucf**, (c4) **κώδικας vhdl**, (a7) screenshot από το σχηματικό *(schematic)*, και (a8) από το “device utilization summary” “Resource Estimation”.

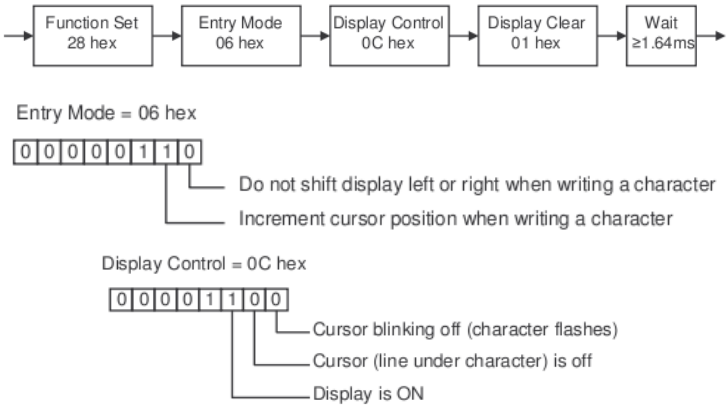
13. Εμφάνιση μηνύματος hello world στην LCD οθόνη με hardware και software υλοποίηση



Πριν χρησιμοποιηθεί το LCD display από τον επεξεργαστή, σύμφωνα με το datasheet, θα πρέπει να αρχικοποιηθεί με μια συγκεκριμένη ακολουθία. Σε αυτό το εργαστήριο, η ακολουθία έχει δημιουργηθεί σε λογισμικό (στην προηγούμενη άσκηση είχε δημιουργηθεί με VHDL σε hardware). Θα χρησιμοποιηθεί η ρουτίνα LCD_reset η οποία εκτελεί την παρακάτω ακολουθία αρχικοποίησης:



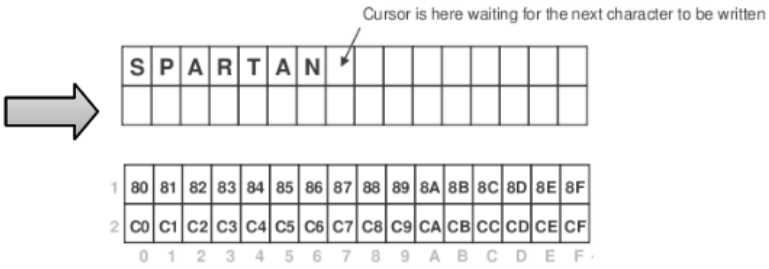
Το επόμενο τμήμα της ακολουθίας χρησιμοποιείται για να εγκαθιδρύσει τη λειτουργία της οθόνης:



Μόλις ολοκληρωθεί η αρχικοποίηση, η εγγραφή χαρακτήρων στο LCD έχει ως συνέπεια να τοποθετηθούν στην άνω αριστερή γωνία και να τοποθετούνται σειριακά προς τα δεξιά και στη συνέχεια, μόλις γεμίσει η γραμμή, στην κάτω γραμμή.

Όταν χρησιμοποιήσετε picoblaze, θα γράφετε στην οθόνη με εντολές assembly, όπως παρακάτω:

```
LOAD s5, character_S
CALL LCD_write_data
LOAD s5, character_P
CALL LCD_write_data
LOAD s5, character_A
CALL LCD_write_data
LOAD s5, character_R
CALL LCD_write_data
LOAD s5, character_T
CALL LCD_write_data
LOAD s5, character_A
CALL LCD_write_data
LOAD s5, character_N
```



Στον παραπάνω κώδικα, κάθε φορά τοποθετούμε στον καταχωρητή **s5** τη δεκαεξαδική τιμή του χαρακτήρα σύμφωνα με τον κώδικα ASCII και καλούμε τη συνάρτηση LCD_write_data που θα κατασκευάσουμε, η οποία γράφει αυτόν τον χαρακτήρα στο LCD και δέχεται μια παράμετρο ως είσοδο, η οποία παράμετρος είναι η τιμή του καταχωρητή s5.

Επίσης, θα χρησιμοποιήσετε και άλλες συναρτήσεις σε assembly για το χειρισμό του LCD, όπως τοποθέτηση του δρομέα σε συγκεκριμένο σημείο, καθαρισμό του display κ.α.Τα βήματα που θα ακολουθήσετε είναι ίδια με τα βήματα της άσκησης (b) για την harware + software υλοποίηση του LED Rotation.

- Μεταφορτώστε το αρχείο **kcpsm3.vhd**, το οποίο φέρει την HDL περιγραφή για το softcore επεξεργαστή picoblaze.
- Μεταφορτώστε το αρχείο **left_right_leds-lcd-Spartan3E.ucf** , το οποίο έχει τα constrains για την πλακέτα Spartan 3E και τροποποιήστε τα LOC, ώστε να ανταποκρίνονται στα constrains της πλακέτας Spartan 3A. Μπορείτε να χρησιμοποιήστε σχετικό αρχείο προηγούμενου εργαστηρίου (αν το έχετε ολοκληρώσει και επιβεβαιώσει την ορθή του λειτουργία). Αποθηκεύστε το αρχείο με το όνομα **left_right_leds-lcd.ucf .**
- Μεταφορτώστε το αρχείο **s3A-led-lcd-interrupts.vhd** και παρατηρήστε:
    - τις συνδέσεις εισόδου *(κουμπιά* btn_XXXX και *rotary knob* rotary_XXX*)*
    - τις συνδέσεις οθόνης *(*lcd_XXX*)*.
    - τον κώδικα χειρισμού interrupt (διεργασία interrupt_control), που ενεργοποιείται μόλις συμβεί κάποια αλλαγή στο rotary knob και που στέλνει το κατάλληλο σήμα interrupt στον επεξεργαστή.

  Σε αυτό το αρχείο, ενδέχεται να χρειαστεί να απομακρύνετε τις γραμμές που αναφέρουν strataflash, γιατί αναφέρονται στην 3E πλακέτα και αν έχετε 3A πλακέτα δεν υπάρχει.
- Μεταφορτώστε το αρχείο **control-incomplete.psm** , το οποίο περιέχει την assembly που θα χρησιμοποιήσετε για τη δημιουργία του αρχείου της ROM. Τροποποιήστε τις κατάλληλες γραμμές που έχουν σημανθεί με το TODO. Τις τιμές μπορείτε να τις βρείτε με τη μελέτη του προηγούμενου αρχείου μαζί με τη γνώση των εντολών assembly από το εγχειρίδιο χρήσης του picoblaze. Αποθηκεύστε το νέο αρχείο με το όνομα **control.psm .**

  Σημειώσεις:
    - **TODO1:** Να τοποθετήσετε τη δεκαεξαδική τιμή (χωρίς το h) που υποδηλώνει την LCD θύρα εξόδου, δηλαδή την output port που έχει συνδεθεί η LCD (μελετήστε το προηγούμενο vhd αρχείο).
    - **TODO2:** Να γραφεί σε ASSEMBLY η κλήση της συνάρτησης που εμφανίζει το μήνυμα HELLO UOWM (όχι άλλων μηνυμάτων π.χ. hello spartan).
    - **TODO3**: Να προσθέσετε γραμμές assembly, παρόμοιες με τη συνάρτηση που εμφανίζει το μήνυμα www.xilinx.com, ώστε να εμφανίζεται το μήνυμα HELLO UOWM. Μην ξεχάσετε το RETURN στο τέλος των εντολών assembly που θα γράψετε.

- **TODO4:** Απαιτείται κάποια εντολή ολίσθησης. Μπορείτε να βρείτε τις εντολές από το συγκεντρωτικό πίνακα KCPSM3 Instruction Set στο “Shift and Rotate Group”.
- **TODO5:** Απαιτείται κάποια εντολή εξόδου, ώστε να γραφείο καταχωρητής s4 στη θύρα εξόδου LCD, η οποία έχει οριστεί ως CONSTANT σε κάποιο σημείο αυτού του αρχείου. Μπορείτε να βρείτε τις εντολές από το συγκεντρωτικό πίνακα KCPSM3 Instruction Set στο “Input/Output Group”.
- **TODO6**: Ίδιο με TODO5
- **TODO7**: Ίδιο με TODO5
- **TODO8**: Ίδιο με TODO5
- **TODO9**: Απαιτείται κάποια εντολή ολίσθησης. Μπορείτε να βρείτε τις εντολές από το συγκεντρωτικό πίνακα KCPSM3 Instruction Set στο “Shift and Rotate Group”.
- **TODO10**: Απαιτείται εντολή επιστροφής από Interrupt. Μπορείτε να βρείτε την εντολή από το συγκεντρωτικό πίνακα KCPSM3 Instruction Set στο “Interrupt Group”
- **TODO11**: Βρείτε την κατάλληλη ετικέτα ώστε να τοποθετηθεί η κατάλληλη διεύθυνση στο interrupt Vector, ώστε όταν συμβεί κάποιο interrupt, να εκτελεστεί η ρουτίνα εξυπηρέτησης διακοπής.

- Μεταφορτώστε τα αρχεία KCPSM3.EXE και ROM_form.vhd , τα οποία θα βοηθήσουν στη δημιουργία του αρχείου lrrom.vhd. Τοποθετήστε αυτά τα 2 αρχεία, όπως και το αρχείο που control.psm σε έναν κατάλογο που μπορείτε να μεταβείτε εύκολα από τη γραμμή εντολών. Από τη γραμμή εντολών μεταβείτε στον κατάλογο που έχετε αποθηκεύσει τα αρχεία και δώστε την εντολή **KCPSM3.EXE control.psm**. Αν εκτελεστεί χωρίς σφάλμα *(παρατηρήστε προσεκτικά την έξοδο του προγράμματος),* θα δείτε στο τέλος ότι έχει δημιουργηθεί με επιτυχία το αρχείο **control.vhd** *(όπως και άλλα αρχεία).*
- Δημιουργήστε ένα νέο project στο Xilinx IDE με τις κατάλληλες ρυθμίσεις για την πλακέτα του εργαστηρίου. Τοποθετήστε τα αρχεία: **control.vhd, s3A-led-lcd-interrupts.vhd, kcpsm3.vhd** κάντε σύνθεση και δημιουργία του bitstream, προγραμματίστε την πλακέτα και επιβεβαιώστε την ορθή λειτουργία.

Παραδοτέα: Τα αρχεία (c5) **control.vhd,** (c6) **left_right_leds-lcd.ucf,** (a9) screenshot από το σχηματικό *(schematic)*, και (a10) από το “device utilization summary” ή το “Resource Estimation”.