

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**Σχεδίαση και υλοποίηση διάταξης μετρήσεων με το κύκλωμα
MAXimator FPGA MAX10 Evaluation kit**

Πτυχιακή Εργασία του

Ανέστη Κολοβού (3467)

Επιβλέπων: Ι.Α. Καλόμοιρος, Καθηγητής

ΣΕΡΡΕΣ, ΣΕΠΤΕΜΒΡΙΟΣ 2020

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία μελετώνται τα χαρακτηριστικά και η λειτουργία του κυκλώματος αναλογικής/ψηφιακής μετατροπής (Analog to Digital Converter), που βρίσκεται ενσωματωμένο στα κυκλώματα FPGA της οικογένειας MAX10 και το οποίο αναπτύσσει η εταιρεία INTEL. Συγκεκριμένα, μελετάται το κύκλωμα αναλογικής ψηφιακής μετατροπής του κυκλώματος FPGA της οικογένειας MAX10 με κωδικό **10M08DAF256C8G**. Κατά την ανάπτυξη της εργασίας, μελετήθηκε η κατάλληλη διαμόρφωση των παραμέτρων του κυκλώματος αναλογικής ψηφιακής μετατροπής. Επίσης υλοποιήθηκε μια απλή εφαρμογή για τη μέτρηση του ηλεκτρικού δυναμικού που αναπτύσσεται σε συγκεκριμένο αναλογικό κανάλι εισόδου του κυκλώματος αναλογικής ψηφιακής μετατροπής, με τη βοήθεια ποτενσιομέτρου. Η τιμή της μέτρησης που προκύπτει απεικονίζεται σε οθόνη LED επτά τομέων, τεσσάρων δεκαδικών ψηφίων. Ο κώδικας περιγραφής κυκλωμάτων που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής, αναπτύχθηκε σε γλώσσα περιγραφής υλικού VHDL και δημιουργήθηκε στο περιβάλλον ανάπτυξης εφαρμογών (IDE) για κυκλώματα FPGA, Quartus Prime Lite Edition. Το υλικό πάνω στο οποίο εκτελείται η εφαρμογή είναι το αναπτυξιακό τυπωμένο κύκλωμα MAXimator της εταιρείας KAMAMI.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ	3
ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ	4
ΕΙΣΑΓΩΓΗ	6
1 ΔΟΜΗ ΤΩΝ ΚΥΚΛΩΜΑΤΩΝ FPGA	8
1.1 ΤΑ ΛΟΓΙΚΑ ΣΤΟΙΧΕΙΑ (LOGIC ELEMENTS LEs)	8
1.2 ΔΟΜΕΣ ΛΟΓΙΚΩΝ ΠΙΝΑΚΩΝ (LOGIC ARRAY BLOCKS LABs)	9
1.3 ΠΕΡΙΦΕΡΕΙΑΚΕΣ ΣΥΣΚΕΥΕΣ ΤΩΝ ΚΥΚΛΩΜΑΤΩΝ FPGA.	10
1.4 Ο ΣΥΜΒΟΛΟΚΩΔΙΚΑΣ ΠΕΡΙΓΡΑΦΗΣ ΚΥΚΛΩΜΑΤΩΝ HDL (HARDWARE DESCRIPTION LANGUAGE).....	10
1.5 ΣΥΓΚΡΙΣΗ ΤΩΝ ΚΥΚΛΩΜΑΤΩΝ FPGA ΜΕ ΤΑ ΚΥΚΛΩΜΑΤΑ ASIC (APPLICATION-SPECIFIC INTEGRATED CIRCUIT)	12
2 ΤΟ ΚΥΚΛΩΜΑ FPGA 10M08DAF256C8G ΤΗΣ INTEL	13
2.1 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ FPGA 10M08DAF256C8G	13
2.2 ΤΟ ΤΥΠΩΜΕΝΟ ΑΝΑΠΤΥΞΙΑΚΟ ΚΥΚΛΩΜΑ MAXIMATOR.	15
2.3 ΤΟ ΚΥΚΛΩΜΑ ΔΙΕΠΑΦΩΝ MAXIMATOR EXPANDER.	17
3 ΣΧΕΔΙΑΣΗ ΕΦΑΡΜΟΓΩΝ ΓΙΑ ΚΥΚΛΩΜΑΤΑ FPGA	19
3.1 ΤΑ ΣΤΑΔΙΑ ΣΧΕΔΙΑΣΗΣ ΕΦΑΡΜΟΓΩΝ ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ QUARTUS PRIME.	19
4 Ο ΕΝΣΩΜΑΤΩΜΕΝΟΣ ΑΝΑΛΟΓΙΚΟΣ/ΨΗΦΙΑΚΟΣ ΜΕΤΑΤΡΟΠΕΑΣ (ADC BLOCK) ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ FPGA 10M08DAF256C8G	22
4.1 ΟΙ ΚΑΤΗΓΟΡΙΕΣ ΑΝΑΛΟΓΙΚΩΝ/ΨΗΦΙΑΚΩΝ ΜΕΤΑΤΡΟΠΕΩΝ ΤΗΣ ΟΙΚΟΓΕΝΕΙΑΣ MAX10.	22
4.2 ΔΕΙΓΜΑΤΟΛΗΨΙΑ ΤΩΝ ΑΝΑΛΟΓΙΚΩΝ ΚΑΝΑΛΙΩΝ ΤΟΥ ΑΝΑΛΟΓΙΚΟΥ/ΨΗΦΙΑΚΟΥ ΜΕΤΑΤΡΟΠΕΑ	22
4.3 ΑΚΡΙΒΕΙΑ ΤΗΣ ΑΝΑΛΟΓΙΚΗΣ/ΨΗΦΙΑΚΗΣ ΜΕΤΑΤΡΟΠΗΣ ΚΑΙ Η ΤΑΞΗ ΑΝΑΦΟΡΑΣ.....	24
5 Ο ΕΛΕΓΧΟΣ ΤΟΥ ΑΝΑΛΟΓΙΚΟΥ/ΨΗΦΙΑΚΟΥ ΜΕΤΑΤΡΟΠΕΑ	27
5.1 5.1 Ο ΠΡΟΣΧΕΔΙΑΣΜΕΝΟΣ ΠΥΡΗΝΑΣ ΕΛΕΓΧΟΥ ALTERA MODULAR ADC IP CORE.	27
5.2 ΟΙ ΜΙΚΡΟΠΥΡΗΝΕΣ ΕΛΕΓΧΟΥ ΤΟΥ ALTERA MODULAR ADC IP CORE.	27
5.2.1 <i>ADC control core</i>	27
5.2.2 <i>Sequencer core</i>	30
5.2.3 <i>Sample Storage core</i>	32
5.2.4 <i>Threshold Detection core</i>	33
6 ΟΙ ΔΙΑΜΟΡΦΩΣΕΙΣ ΤΟΥ ALTERA MODULAR ADC IP CORE	34
6.1 ΔΙΑΔΟΧΙΚΗ ΜΕΤΑΤΡΟΠΗ ΑΝΑΛΟΓΙΚΩΝ ΚΑΝΑΛΙΩΝ ΜΕ ΕΓΓΡΑΦΗ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΣΕ ΕΝΣΩΜΑΤΩΜΕΝΗ ΜΝΗΜΗ RAM (STANDARD SEQUENCER WITH AVALON-MM SAMPLE STORAGE).....	34
6.2 ΔΙΑΔΟΧΙΚΗ ΜΕΤΑΤΡΟΠΗ ΑΝΑΛΟΓΙΚΩΝ ΚΑΝΑΛΙΩΝ ΜΕ ΕΓΓΡΑΦΗ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΣΕ ΕΝΣΩΜΑΤΩΜΕΝΗ ΜΝΗΜΗ RAM ΚΑΙ ΑΝΙΧΝΕΥΣΗ ΠΑΡΑΒΙΑΣΗΣ ΟΡΙΩΝ (STANDARD SEQUENCER WITH AVALON-MM SAMPLE STORAGE AND THRESHOLD VIOLATION DETECTION).	34
6.3 ΔΙΑΔΟΧΙΚΗ ΜΕΤΑΤΡΟΠΗ ΑΝΑΛΟΓΙΚΩΝ ΚΑΝΑΛΙΩΝ ΜΕ ΕΓΓΡΑΦΗ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΣΕ ΕΞΩΤΕΡΙΚΗ ΜΝΗΜΗ RAM (STANDARD SEQUENCER WITH EXTERNAL SAMPLE STORAGE).	35
6.4 ΧΡΗΣΗ ΜΟΝΟ ΤΟΥ ΜΙΚΡΟΠΥΡΗΝΑ ADC CONTROL CORE.....	36
7 ΔΗΜΙΟΥΡΓΙΑ ΑΠΛΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΤΟΥ ALTERA MODULAR ADC IP CORE ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ QUARTUS PRIME	37
7.1 ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ PROJECT.	37
7.2 ΔΗΜΙΟΥΡΓΙΑ ΑΝΩΤΕΡΗΣ ΟΝΤΟΤΗΤΑΣ (TOP ENTITY).....	43

7.3	ΣΧΕΔΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ	45
7.4	ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΤΟΥ ALTERA MODULAR ADC CORE	47
7.5	ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ ΧΡΟΝΙΣΜΟΥ ALTRPLL	54
7.6	ΔΗΜΙΟΥΡΓΙΑ ΣΗΜΑΤΩΝ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗΣ ΤΗΣ ΣΥΣΚΕΥΗΣ MYADC.	62
7.7	Ο ΜΕΤΑΤΡΟΠΕΑΣ ΔΥΑΔΙΚΟΥ ΑΡΙΘΜΟΥ ΣΕ ΚΩΔΙΚΟΠΟΙΗΣΗ BCD.	67
7.8	ΔΗΜΙΟΥΡΓΙΑ ΚΥΚΛΩΜΑΤΟΣ ΧΡΟΝΙΚΗΣ ΚΑΘΥΣΤΕΡΗΣΗΣ ΓΙΑ ΚΑΛΥΤΕΡΟ ΟΠΤΙΚΟ ΑΠΟΤΕΛΕΣΜΑ ΤΗΣ ΔΕΚΑΔΙΚΗΣ ΑΠΕΙΚΟΝΙΣΗΣ.	76
7.9	ΤΑ ΚΥΚΛΩΜΑΤΑ ΟΔΗΓΗΣΗΣ ΤΩΝ ΨΗΦΙΩΝ LED ΔΕΚΑΔΙΚΗΣ ΑΠΕΙΚΟΝΙΣΗΣ.	83
7.10	ΔΗΜΙΟΥΡΓΙΑ ΕΙΣΟΔΩΝ-ΕΞΟΔΩΝ ΓΙΑ ΤΗΝ ΕΦΑΡΜΟΓΗ.....	93
7.11	ΑΝΑΘΕΣΗ ΤΩΝ ΕΙΣΟΔΩΝ-ΕΞΟΔΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΣΤΟΥΣ ΑΚΡΟΔΕΚΤΕΣ ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ FPGA.	96
7.12	COMPILATION ΤΗΣ ΕΦΑΡΜΟΓΗΣ.	98
7.13	ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΔΙΑΜΟΡΦΩΣΗ ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ FPGA ΜΕ ΤΟΝ ΚΩΔΙΚΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ.	100
7.14	ΕΚΤΕΛΕΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΑΠΟ ΤΟ ΑΝΑΠΤΥΞΙΑΚΟ ΤΥΠΩΜΕΝΟ ΚΥΚΛΩΜΑ ΜΑΧΙΜΑΤΟΡ.	105
8	ΣΥΜΠΕΡΑΣΜΑΤΑ	107
9	ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ	109
9.1	ΣΧΗΜΑΤΑ	109
9.2	ΕΙΚΟΝΕΣ.....	110
9.3	ΠΙΝΑΚΕΣ	110
	ΒΙΒΛΙΟΓΡΑΦΙΑ	111
	ΠΑΡΑΡΤΗΜΑ Α	112
	ΠΑΡΑΡΤΗΜΑ Β.....	113
	ΠΑΡΑΡΤΗΜΑ Γ	118

ΕΙΣΑΓΩΓΗ

Στη σύγχρονη κοινωνική πραγματικότητα, υπάρχει έντονη η ανάγκη της απελευθέρωσης του ανθρώπου, από την ενασχόλησή του με διάφορες καθημερινές διεργασίες και δραστηριότητες. Η ανάγκη αυτή αποσκοπεί αφενός στην απόδοση πολύτιμου καθημερινού χρόνου στον άνθρωπο και αφετέρου στην εξάλειψη της πιθανότητας αστοχίας από ανθρώπινο παράγοντα, κατά τη διάρκεια εκτέλεσης κρίσιμων διαδικασιών. Επιπρόσθετα, η αλματώδης εξέλιξη της τεχνολογίας σε όλους τους τομείς της παράγει καθημερινά νέες αναγκαίες διαδικασίες, οι οποίες ως επί τω πλείστον είναι αδύνατο να διεκπεραιωθούν από τον άνθρωπο.

Την ανάγκη αυτή προσπαθούν να καλύψουν οι επιστήμες της Πληροφορικής, της Ηλεκτρονικής και της Μηχανολογίας, παρουσιάζοντας θεαματικά συνδυαστικά αποτελέσματα διάφορων εφαρμογών αυτοματισμού. Πολυάριθμα είναι τα παραδείγματα της σύγχρονης καθημερινότητας. Από τη δημιουργία ηλεκτρομηχανολογικών εφαρμογών στη βιομηχανία μέχρι τη δημιουργία των ηλεκτρικών οικιακών συσκευών καθώς και από τα συστήματα μεταφοράς ανθρώπων και εμπορευμάτων μέχρι τα συστήματα μεταφοράς πληροφοριών. Σε κάθε τομέα της σύγχρονης κοινωνικής πραγματικότητας υπάρχει μια ηλεκτρομηχανολογική υλοποίηση.

Στις υλοποιήσεις που προαναφέρθηκαν, οι επιστήμες της Ηλεκτρονικής και της Πληροφορικής παίζουν τον ουσιαστικό ρόλο του ελέγχου της συνολικής λειτουργίας. Τα ενσωματωμένα συστήματα (embedded systems) κατέχουν μεγάλο μερίδιο στον τομέα ελέγχου των διάφορων εφαρμογών αυτοματισμού, αναπτύσσονται ταχύτατα και εξελίσσονται προς την κατεύθυνση της αύξησης επεξεργαστικής ισχύος και μείωσης του όγκου και της ενεργειακής κατανάλωσης. Στην καρδιά των ενσωματωμένων συστημάτων βρίσκονται οι μονάδες επεξεργασίας, η ιστορική εξέλιξη των οποίων ακολουθεί την κατεύθυνση που προαναφέραμε. Υπάρχουν πολλά είδη μονάδων επεξεργασίας όπως οι μικροελεγκτές (microcontrollers), οι μικροεπεξεργαστές (microprocessors), τα κυκλώματα PLD (Programmable Logic Device), CPLD (Complex Programmable Logic Device), τα κυκλώματα FPGA (Field Programmable Gate Array) καθώς και πλήθος από εξειδικευμένα βάσει εφαρμογής ολοκληρωμένα κυκλώματα ASIC (Application-Specific Integrated Circuit) ή SoC (System on Chip).

Η παρούσα εργασία πραγματεύεται την χρήση των κυκλωμάτων FPGA (Field Programmable Gate Array) και πιο συγκεκριμένα την δειγματοληψία αναλογικού σήματος και την ψηφιακή μετατροπή του μέσω ενός κυκλώματος FPGA.

Στο πρώτο κεφάλαιο της εργασίας, παρουσιάζονται στοιχεία που αφορούν την δομή και την λειτουργία, γενικά, των κυκλωμάτων FPGA. Γίνεται αναφορά στις μικρότερες δομικές μονάδες των κυκλωμάτων FPGA που είναι τα λογικά στοιχεία (Logic Elements LEs), καθώς επίσης και στον τρόπο που οργανώνονται σε λογικούς πίνακες (Logic Array Blocks LABs) και συνδέονται μεταξύ του προκειμένου να υλοποιήσουν τις διάφορες εφαρμογές.

Στο δεύτερο κεφάλαιο πραγματοποιείται μια ειδικότερη ανάλυση των χαρακτηριστικών του κυκλώματος FPGA 10M08DAF256C8G που αποτελεί μέλος της οικογένειας MAX10 της εταιρείας Intel. Επίσης περιγράφονται τα χαρακτηριστικά του αναπτυξιακού τυπωμένου κυκλώματος MAXimator, το οποίο εμπεριέχει το κύκλωμα FPGA 10M08DAF256C8G και θα χρησιμοποιηθεί για την υλοποίηση της εφαρμογής.

Στο τρίτο κεφάλαιο περιγράφονται τα στάδια της σχεδίασης και υλοποίησης εφαρμογών για κυκλώματα FPGA, με την βοήθεια του λογισμικού περιβάλλοντος ανάπτυξης εφαρμογών FPGA, Quartus Prime.

Στο τέταρτο κεφάλαιο πραγματοποιείται μια αναφορά στα χαρακτηριστικά της συσκευής αναλογικής/ψηφιακής μετατροπής (ADC converter) που βρίσκεται ενσωματωμένη στο κύκλωμα FPGA 10M08DAF256C8G

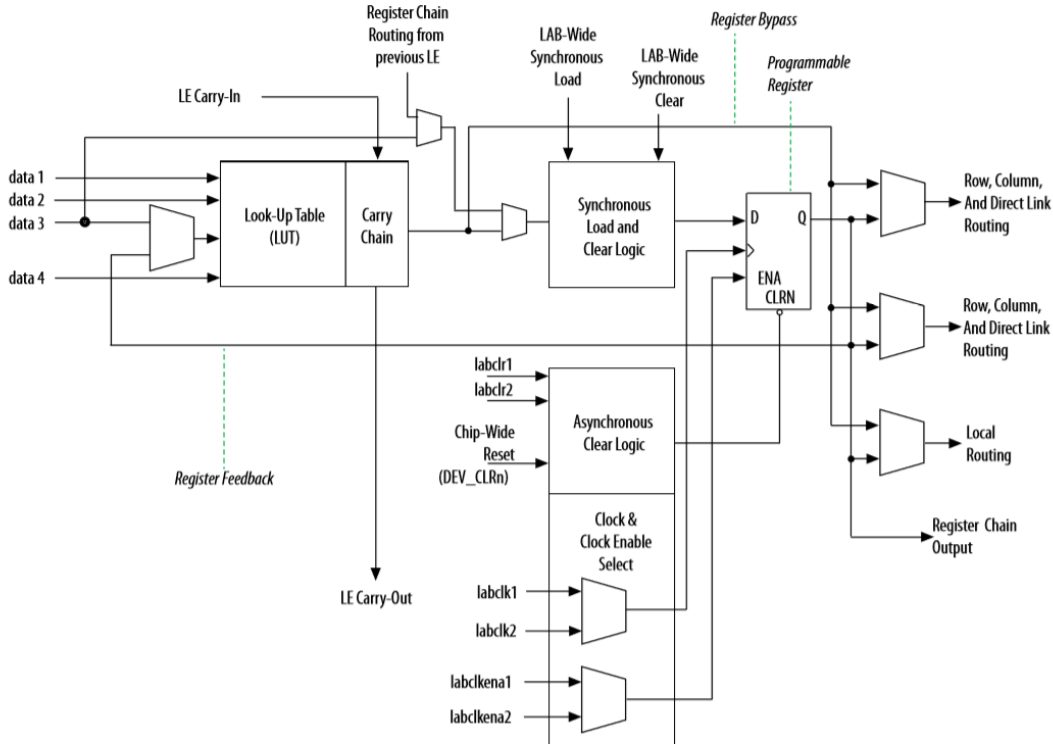
Στο πέμπτο κεφάλαιο περιγράφονται αναλυτικά οι μέθοδοι ελέγχου και παραμετροποίησης της συσκευής αναλογικής/ψηφιακής μετατροπής ενώ στο έκτο κεφάλαιο γίνεται περιγραφή των διαμορφώσεων της συσκευής αναλογικής/ψηφιακής μετατροπής, που είναι διαθέσιμες από το λογισμικό περιβάλλον ανάπτυξης Quartus Prime.

Τέλος στο έβδομο κεφάλαιο της εργασίας, πραγματοποιείται εκτενής και αναλυτική περιγραφή, της δημιουργίας και εκτέλεσης της εφαρμογής αναλογικής/ψηφιακής μετατροπής, με απεικόνιση του αποτελέσματος σε οθόνη τεσσάρων δεκαδικών ψηφίων LED επτά τομέων.

1 Δομή των κυκλωμάτων FPGA

1.1 Τα λογικά στοιχεία (Logic Elements LEs)

Τα σύγχρονα κυκλώματα FPGA αποτελούνται από λογικά στοιχεία (Logic Elements) ή αλλιώς συντομογραφικά LEs. Το λειτουργικό διάγραμμα ενός LE φαίνεται παρακάτω και είναι αρκετά πολύπλοκο. Το λογικό στοιχείο LE αποτελεί την μικρότερη λογική δομή μέσα σε ένα κύκλωμα FPGA (Σχήμα 1). Ένα λογικό στοιχείο έχει σήματα εισόδων για τα

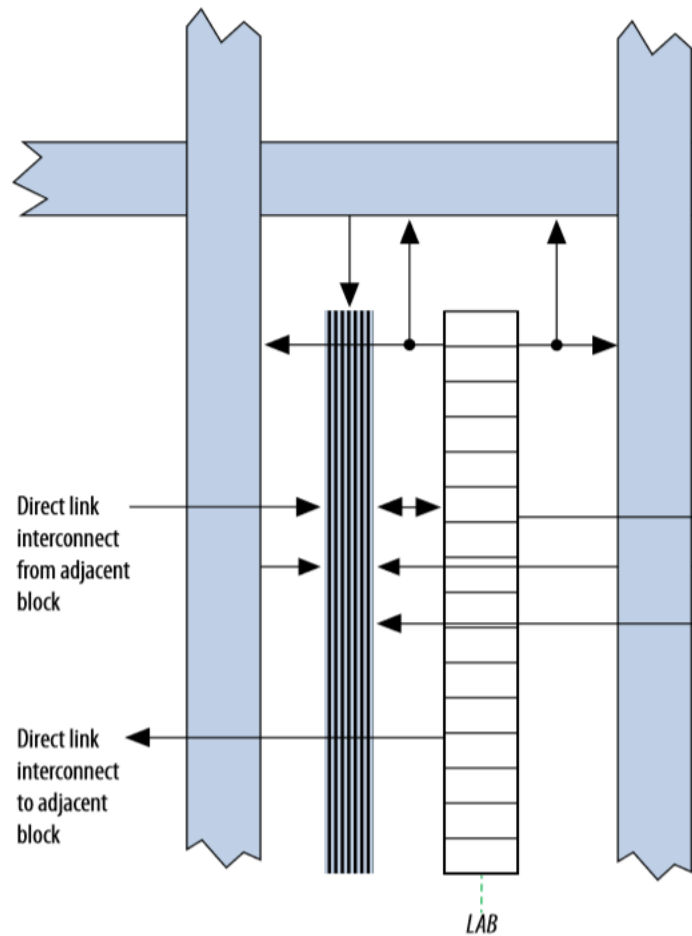


Σχήμα 1. Διαγραμματική απεικόνιση Λογικού Στοιχείου (Logic Element LE) κυκλώματος FPGA της οικογένειας MAX10.

δεδομένα προς επεξεργασία, εξόδων για τα επεξεργασμένα δεδομένα καθώς επίσης και σήματα ελέγχου της λειτουργίας του (Altera Corporation 2016).

1.2 Δομές λογικών πινάκων (Logic Array Blocks LABs)

Τα LEs διατάσσονται σε μεγαλύτερες δομές οι οποίες ονομάζονται LABs (Logic Array Blocks) και έχουν την δυνατότητα της εσωτερικής σύνδεσης μεταξύ τους καθώς και με άλλες LEs που βρίσκονται σε διαφορετικά LABs. Κάθε LAB περιέχει συγκεκριμένο αριθμό από LEs. Η σχηματική δομή ενός LAB απεικονίζεται στο Σχήμα 2.

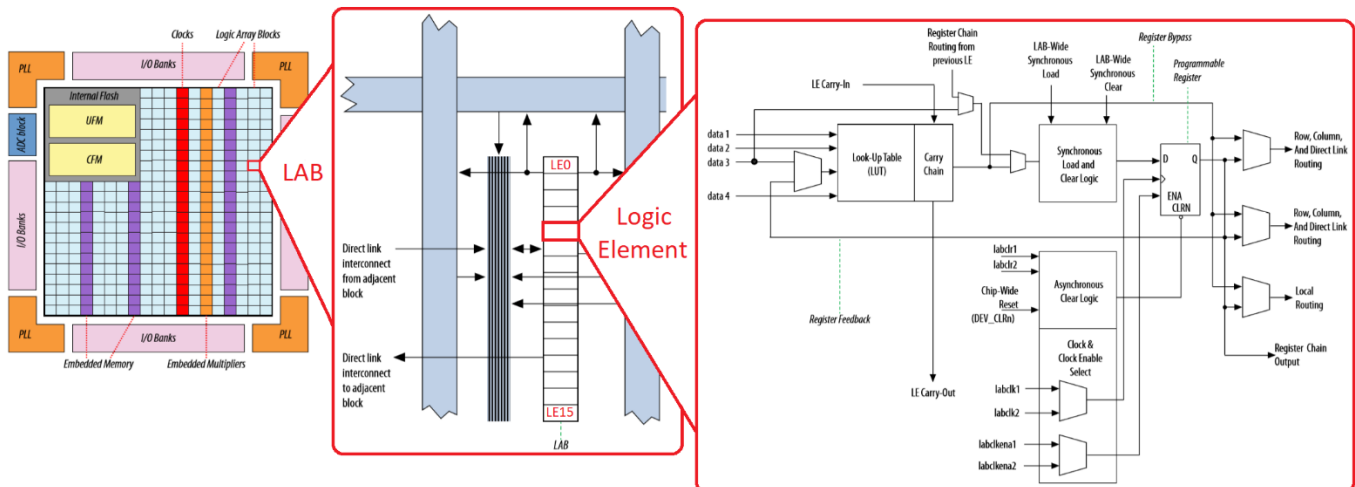


Σχήμα 2. Σχηματική απεικόνιση ενός Logic Array Block (LAB)

Στην συγκεκριμένη δομή LAB μπορούμε να παρατηρήσουμε ότι αποτελείται από 16 LEs ενώ εκατέρωθεν μπορούμε να παρατηρήσουμε σχηματικά, τους διαύλους διασύνδεσης των LEs τόσο εσωτερικά στο ίδιο LAB, όσο και εξωτερικά με LEs τα οποία βρίσκονται σε διαφορετικά LABs (Altera Corporation 2016).

1.3 Περιφερειακές συσκευές των κυκλωμάτων FPGA.

Μέσα σε ένα εξελιγμένο κύκλωμα FPGA, μπορεί να υπάρχουν και διάφορες περιφερειακές συσκευές όπως ψηφιακοί/αναλογικοί μετατροπείς (A/D converters), αναλογικοί/ψηφιακοί μετατροπείς (D/A converters), διάφορες θύρες επικοινωνίας υψηλής ταχύτητας (high speed transceivers), κυκλώματα χρονισμού (clocks, PLLs) καθώς και ενσωματωμένες μνήμες FLASH, EEPROM κ.α. Η διαδοχική κατά μέγεθος διάταξη της δομής των LEs, LABs και των διάφορων περιφερειακών σε ένα κύκλωμα FPGA φαίνεται στο Σχήμα 3.



Σχήμα 3. Σχηματική απεικόνιση της δομής ενός κυκλώματος FPGA

1.4 Η γλώσσα περιγραφής υλικού HDL (Hardware Description Language).

Το βασικό πλεονέκτημα των κυκλωμάτων FPGA σχετικά με την σχεδίαση ψηφιακών συστημάτων, είναι η δυνατότητα που παρέχουν στον σχεδιαστή ψηφιακών συστημάτων να αναδιατάσει την σύνδεση των λογικών μονάδων και να δημιουργεί κυκλωματικές υλοποιήσεις ανάλογα με τις ανάγκες σχεδίασης. Το γεγονός αυτό μάλιστα πραγματοποιείται με την χρήση συγκεκριμένης γλώσσας περιγραφής υλικού (HDL Hardware Description Language) όπως φαίνεται στο Σχήμα 4.

```

1  library ieee;
2  USE ieee.std_logic_1164.all;
3
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY binary_BCD IS
7  PORT (clk :in std_logic;
8        ds: in std_logic_vector (15 downto 0);
9        qs: OUT std_logic_vector (15 downto 0));
10 END binary_BCD;
11
12 ARCHITECTURE behaviour OF binary_BCD IS
13 BEGIN
14 PROCESS (clk)
15
16
17     variable i : integer:=0;
18     variable bcd : std_logic_vector(15 downto 0) ;
19     variable bint : std_logic_vector(15 downto 0) ;
20
21     begin
22     if rising_edge(clk) then
23     bcd:="0000000000000000";
24     bint:=ds;
25     for i in 0 to 15 loop -- repeating 16 times.
26     bcd(15 downto 1) := bcd(14 downto 0); --shifting the bits.
27     bcd(0) := bint(15);
28     bint(15 downto 1) := bint(14 downto 0);
29     bint(0) := '0';
30
31
32     if(i < 15 and bcd(3 downto 0) > "0100") then
33     bcd(3 downto 0) := bcd(3 downto 0) + "0011";
34     end if;
35
36     if(i < 15 and bcd(7 downto 4) > "0100") then
37     bcd(7 downto 4) := bcd(7 downto 4) + "0011";
38     end if;
39
40     if(i < 15 and bcd(11 downto 8) > "0100") then
41     bcd(11 downto 8) := bcd(11 downto 8) + "0011";
42     end if;
43
44     if(i < 15 and bcd(15 downto 12) > "0100") then
45     bcd(15 downto 12) := bcd(15 downto 12) + "0011";
46     end if;
47
48     end loop;
49     qs<=bcd;
50     end if;
51 END process;
52 END behaviour;
53

```

Σχήμα 4. Γλώσσα περιγραφής υλικού VHDL

Με την βοήθεια της γλώσσας περιγραφής υλικού, ο σχεδιαστής ψηφιακών συστημάτων μπορεί να περιγράψει την κυκλωματική διάταξη που επιθυμεί να υλοποιήσει. Έπειτα η συγκεκριμένη περιγραφή δημιουργεί τις απαραίτητες συνδέσεις ανάμεσα στα λογικά στοιχεία, που βρίσκονται όπως προαναφέραμε, συγκροτημένες σε ειδικές δομές, έτσι ώστε να είναι εύκολα πραγματοποιήσιμη η ημιαγωγική τους σύνδεση.

1.5 Σύγκριση των κυκλωμάτων FPGA με τα κυκλώματα ASIC (Application-Specific Integrated Circuit)

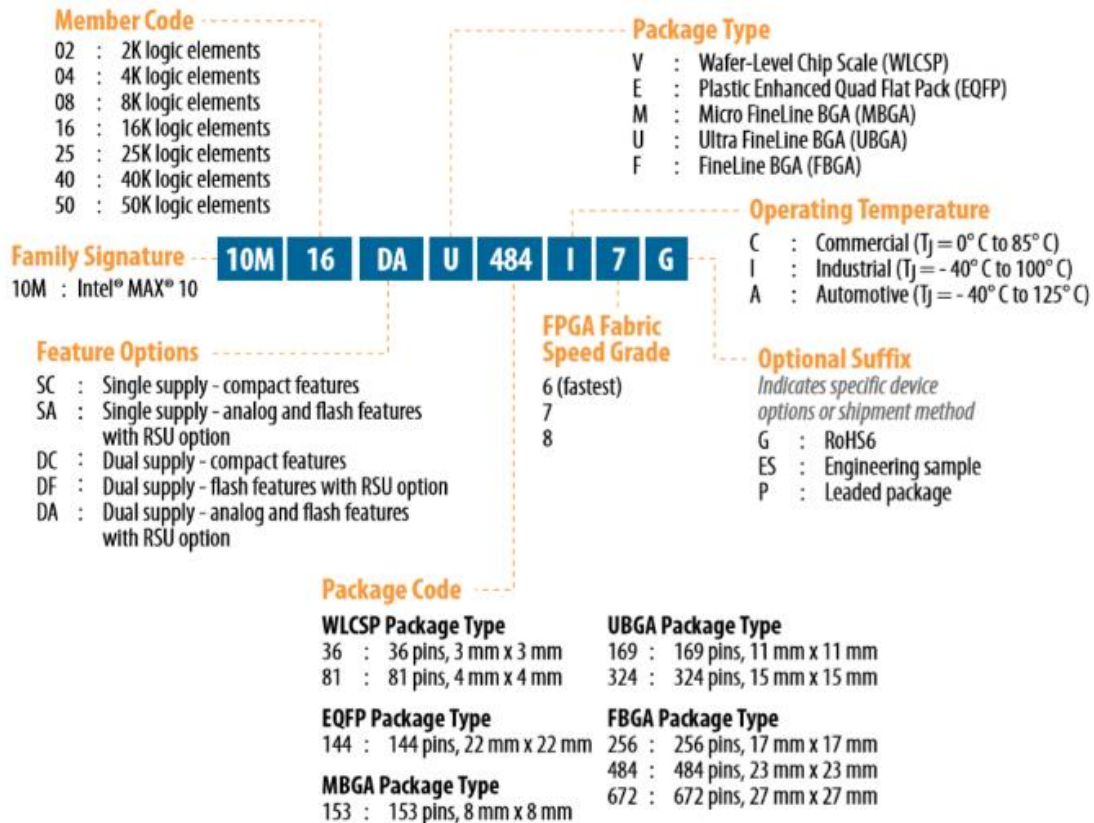
Η ιδιότητα των κυκλωμάτων FPGA να αναδιαμορφώνονται εύκολα και γρήγορα, μέσω του κώδικα περιγραφής κυκλωμάτων, τα καθιστά πολύ χρήσιμα υλικά για τον ψηφιακό σχεδιασμό στην βιομηχανία. Ο σχεδιαστής ψηφιακών συστημάτων έχει την δυνατότητα πολύ γρήγορα και ανέξοδα να υλοποιεί τον ψηφιακό σχεδιασμό του, να τον δοκιμάζει λειτουργικά και να τον επανασχεδιάζει αν αυτό είναι απαραίτητο. Με τον τρόπο αυτό επιταχύνονται οι διαδικασίες της επανασχεδίασης (redesign) και των DVT's (Design Validation Tests) οι οποίες αποτελούν και το πιο χρονοβόρο κομμάτι στην διαδικασία ανάπτυξης ενός προϊόντος.

Είναι λοιπόν αναμενόμενο, τα κυκλώματα FPGA να αντικαταστήσουν σταδιακά τα κυκλώματα ASIC, τα οποία χρησιμοποιούνται ευρύτατα στην βιομηχανία. Η ανάπτυξη και ο επανασχεδιασμός των κυκλωμάτων ASIC (Application-Specific Integrated Circuit), είναι πολύ πιο δαπανηρές διαδικασίες συγκριτικά με την ανάπτυξη ψηφιακών συστημάτων που βασίζονται πάνω στην τεχνολογία των κυκλωμάτων FPGA (Wikipedia, Field-programmable gate array).

2 Το κύκλωμα FPGA 10M08DAF256C8G της INTEL

2.1 Χαρακτηριστικά του κυκλώματος FPGA 10M08DAF256C8G

Στην παρούσα εργασία θα χρησιμοποιηθεί ένα κύκλωμα FPGA της INTEL και συγκεκριμένα το 10M08DAF256C8G το οποίο ανήκει στην οικογένεια MAX 10.

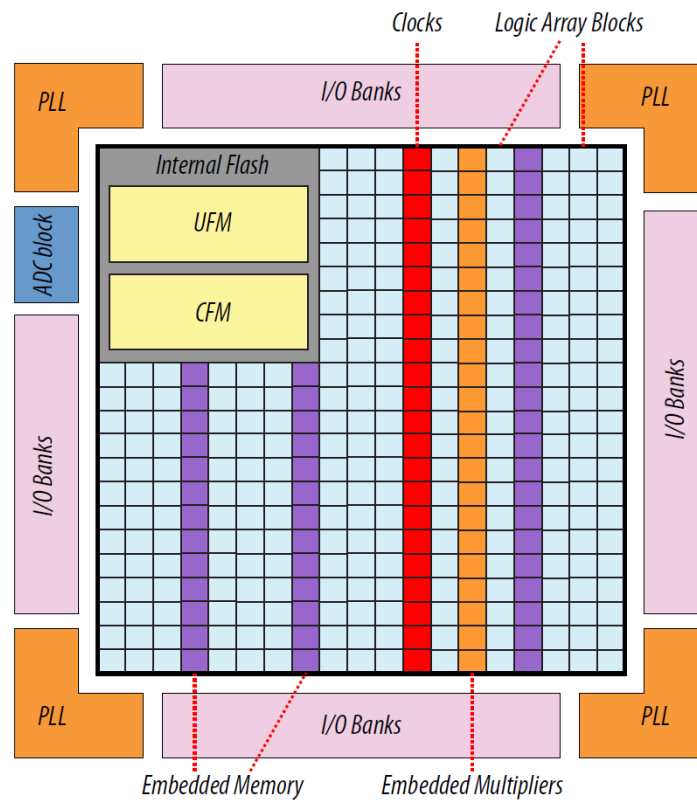


Σχήμα 5. Κωδικοποίηση ονοματολογίας των κυκλωμάτων της οικογένειας MAX 10

Σύμφωνα με Σχήμα 5 που δείχνει την κωδικοποίηση προϊόντος της INTEL, το συγκεκριμένο κύκλωμα FPGA διαθέτει τα εξής χαρακτηριστικά:

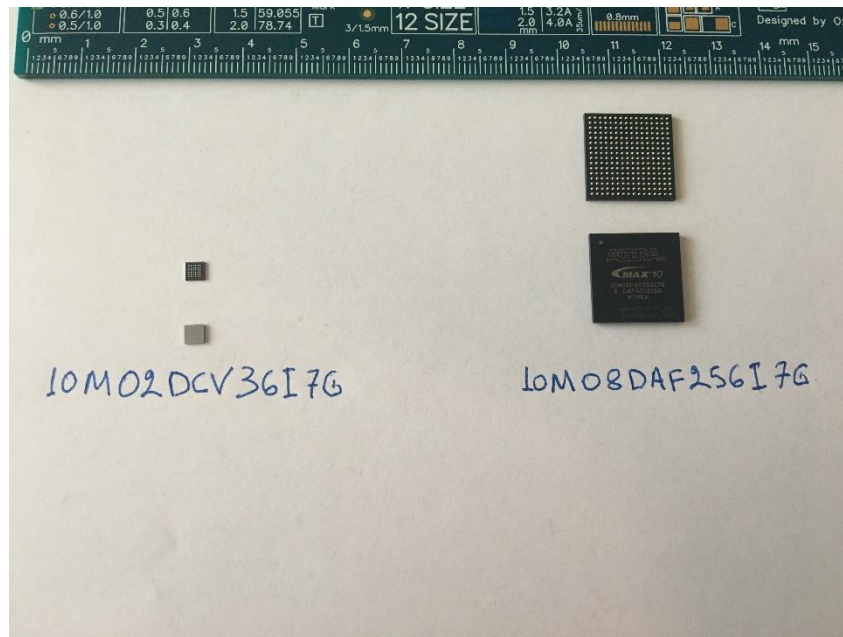
- Οικογένεια προϊόντος: **Intel MAX 10**
- Πλήθος Λογικών Στοιχείων: **8000 Logic Elements**
- Επιπλέον Χαρακτηριστικά : **A/D converter, Διπλή τροφοδοσία, Ενσωματωμένη Μνήμη FLASH**
- Πλήθος ακροδεκτών: **256 ball pins**

Το συγκεκριμένο κύκλωμα FPGA διαθέτει ενσωματωμένο αναλογικό/ψηφιακό μετατροπέα ο οποίος θα χρησιμοποιηθεί και θα μελετηθεί στην παρούσα εργασία. Όλες οι ενσωματωμένες συσκευές που εμπεριέχονται σε ένα κύκλωμα FPGA οργανώνονται σε ανεξάρτητες μονάδες υλικού (modules) και ελέγχονται μέσα από ειδικές μονάδες λογισμικού οι οποίες είναι γραμμένες σε συμβολοκώδικα περιγραφής υλικού όπως προαναφέραμε (HDL). Η Intel αναφέρει μια σειρά από προσχεδιασμένο κώδικα ελέγχου των μονάδων υλικού (modules) ως πυρήνες πνευματικής ιδιοκτησίας (intellectual property cores ή IP cores).



Σχήμα 6. Το κύκλωμα FPGA 10M08DAF256C8G

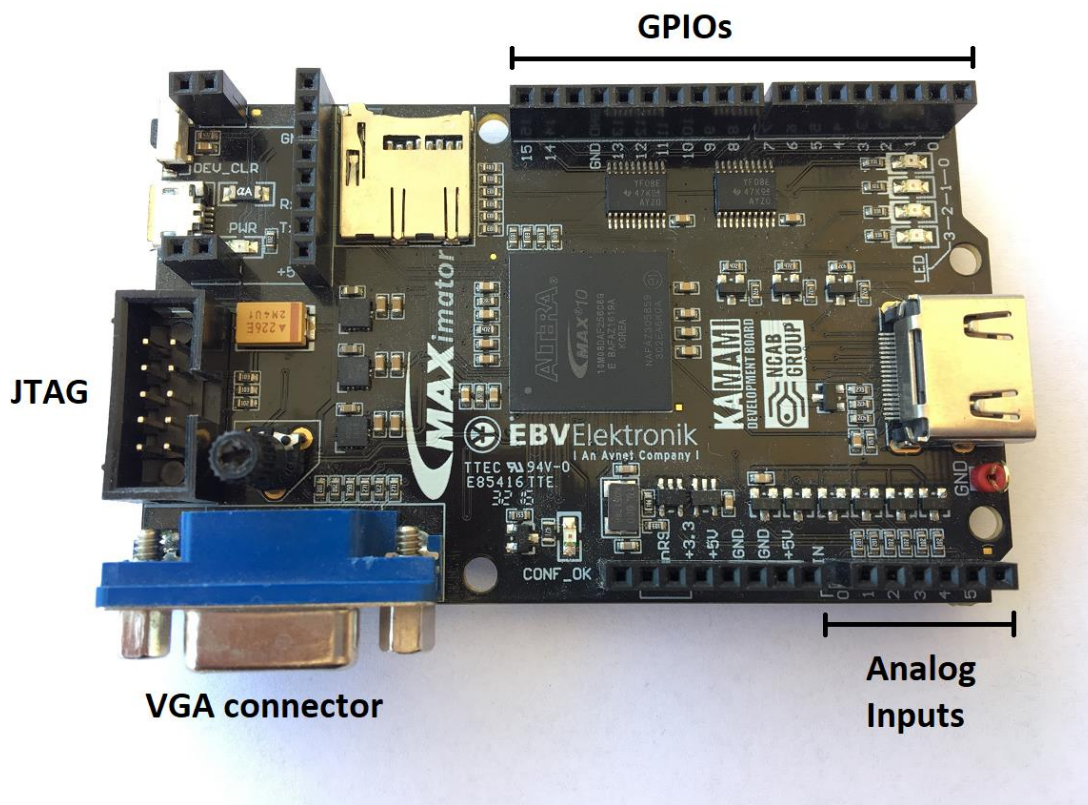
Στην Εικόνα 1 και στα δεξιά το συγκεκριμένο κύκλωμα FPGA που θα χρησιμοποιηθεί στην παρούσα εργασία σε φυσικό μέγεθος (17mm X 17mm, 256 pins). Αριστερά διακρίνεται ένα άλλο κύκλωμα FPGA της ίδιας οικογένειας MAX 10 αλλά πολύ μικρότερου μεγέθους (3mm X 4mm, 36 pins). Στην ίδια εικόνα μπορούμε να παρατηρήσουμε τους φυσικούς ακροδέκτες τις συσκευασίας των κυκλωμάτων FPGA. Ο συγκεκριμένος τύπος ακροδεκτών ονομάζεται FBGA (Fine Ball Grid Array package) (Intel, 2017α).



Εικόνα 1: Δύο μέλη της οικογένειας κυκλωμάτων FPGA MAX 10 σε φυσικές διαστάσεις

2.2 Το τυπωμένο αναπτυξιακό κύκλωμα MAXimator.

Όπως φαίνεται στην Εικόνα 1 το φυσικό μέγεθος των κυκλωμάτων FPGA είναι τόσο μικρό και αντίθετα, η πυκνότητα των ακροδεκτών τόσο μεγάλη, έτσι ώστε ο χειρισμός και η χρήση αυτών των ολοκληρωμένων κυκλωμάτων σε τυπωμένα κυκλώματα, να είναι πολύ δύσκολες διαδικασίες και να απαιτούν ακριβό και εξειδικευμένο εξοπλισμό. Για τον λόγο, αυτό στην παρούσα εργασία θα χρησιμοποιηθεί το εμπορικό αναπτυξιακό τυπωμένο κύκλωμα MAXimator της εταιρείας KAMAMI. Το συγκεκριμένο τυπωμένο κύκλωμα εμπεριέχει το κύκλωμα FPGA 10M08DAF256C8G, όπου μας ενδιαφέρει να το μελετήσουμε ως προς την ενσωματωμένη συσκευή αναλογικής/ψηφιακής μετατροπής που διαθέτει. Επίσης το συγκεκριμένο αναπτυξιακό τυπωμένο κύκλωμα, είναι προσανατολισμένο για υλοποίηση ψηφιακών εφαρμογών σήματος εικόνας VGA και HDMI, καθώς διαθέτει τους αντίστοιχους υποδοχείς (connectors), οπότε είναι μια καλή ευκαιρία να αναπτυχθεί μια συνδυαστική εφαρμογή αναλογικής δειγματοληψίας και ψηφιακής απεικόνισης σε monitor VGA. Στην παρακάτω εικόνα (Εικόνα 2) μπορούμε να παρατηρήσουμε την διάταξη των εισόδων και εξόδων που θα χρησιμοποιηθούν, πάνω στο τυπωμένο κύκλωμα.

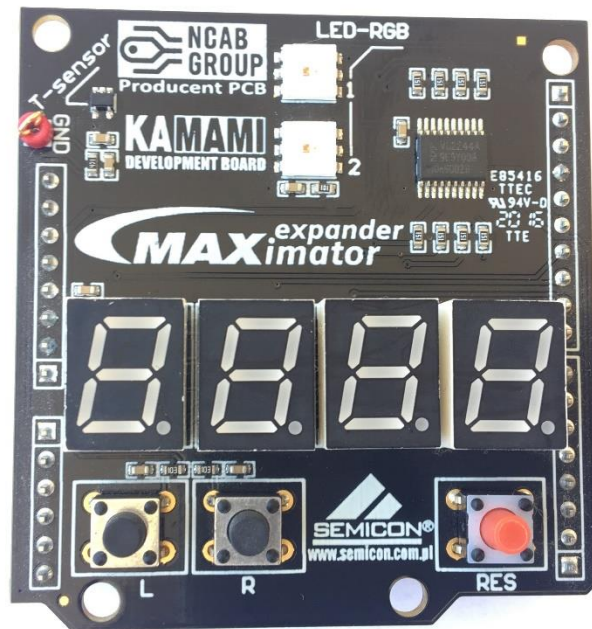


Εικόνα 2: Τυπωμένο αναπτυξιακό κύκλωμα MAXimator

Το κύκλωμα FPGA 10M08DAF256C8G διαθέτει 178 ακροδέκτες εισόδου/εξόδου (GPIO) που μπορούν να χρησιμοποιηθούν για γενική χρήση και 17 ακροδέκτες αναλογικών εισόδων που μπορούν να χρησιμοποιηθούν σε εφαρμογές αναλογικής σε ψηφιακή μετατροπή. Ο κατασκευαστής του τυπωμένου κυκλώματος MAXimator, παρέχει 16 από τους 178 ακροδέκτες για γενική είσοδο/έξοδο και 6 ακροδέκτες αναλογικών εισόδων για αναλογική/ψηφιακή μετατροπή. Επίσης υπάρχει και μία ακόμη αναλογική είσοδος πάνω στο τυπωμένο κύκλωμα, η οποία συνδέεται με το ενσωματωμένο ποτενσιόμετρο που παρέχει συνεχή τάση με εύρος 0V-2.5V, για γρήγορη ανάπτυξη εφαρμογών αναλογικής σε ψηφιακή μετατροπή.

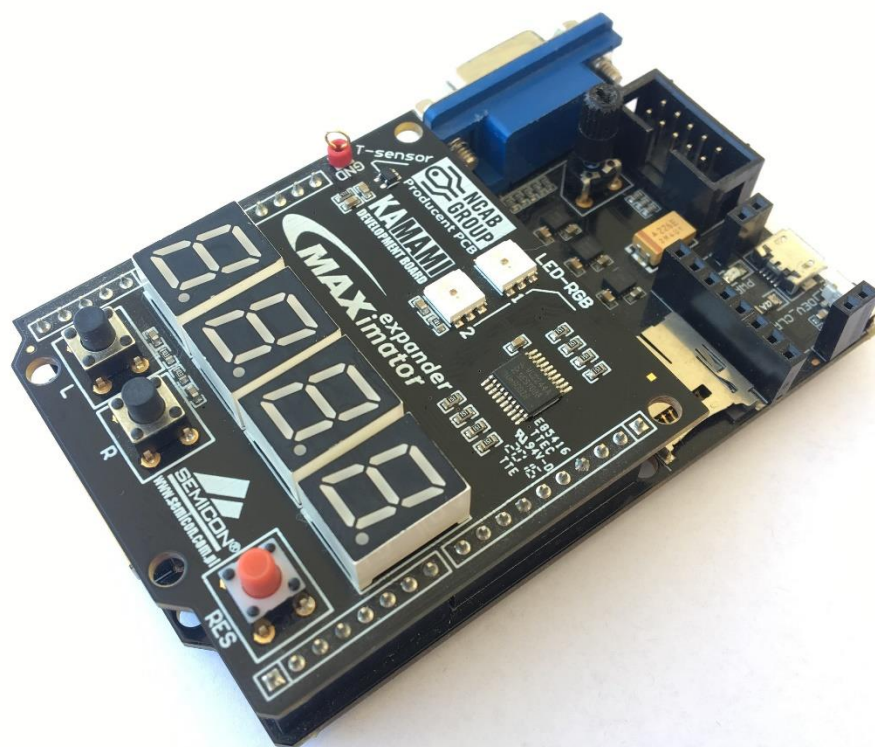
2.3 Το κύκλωμα διεπαφών MAXimator expander.

Για την αύξηση της ευελιξίας των διεπαφών του συγκεκριμένου αναπτυξιακού κυκλώματος, ο κατασκευαστής παρέχει επίσης και το τυπωμένο κύκλωμα διεπαφών (πλήκτρα, διατάξεις απεικόνισης) MAXimator expander, όπως απεικονίζεται παρακάτω (Εικόνα 3).



Εικόνα 3: Τυπωμένο κύκλωμα διεπαφών MAXimator expander

Το συγκεκριμένο κύκλωμα δίνει στον χρήστη την δυνατότητα εισόδου και εξόδου δεδομένων, από και προς το κύκλωμα FPGA του μητρικού τυπωμένου κυκλώματος. Το γεγονός αυτό επιτυγχάνεται με την σύνδεση των δύο τυπωμένων κυκλωμάτων όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 4).



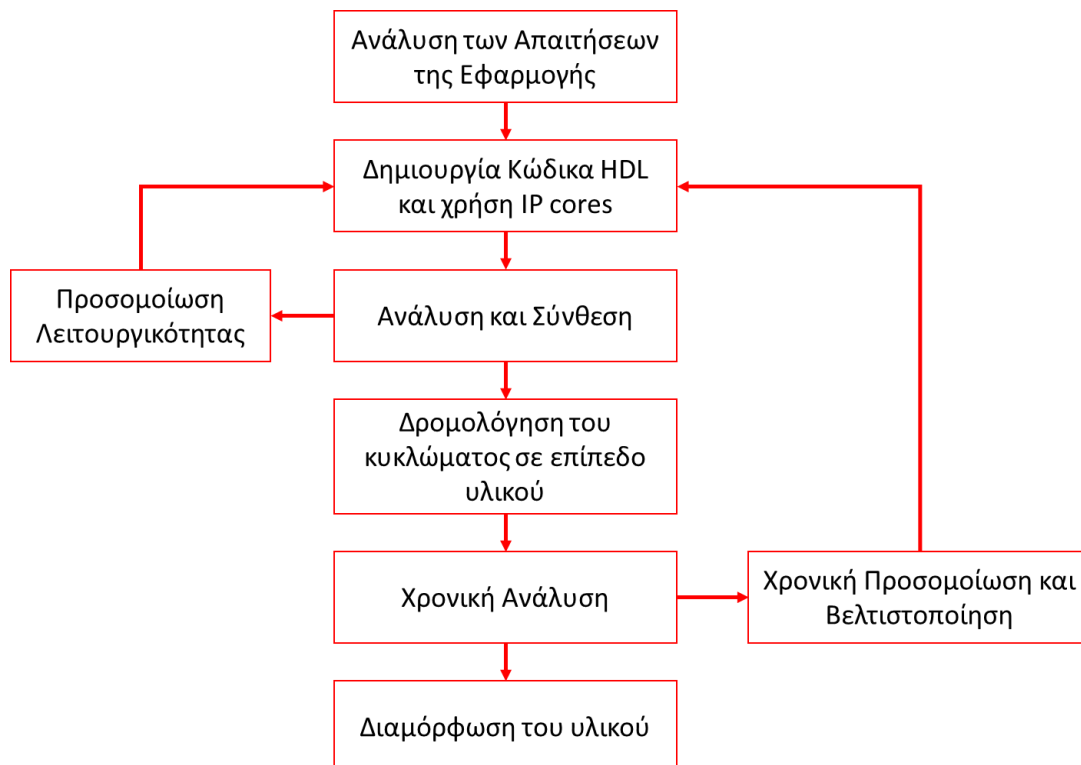
Εικόνα 4: Το συνολικό αναπτυξιακό τυπωμένο κύκλωμα.

Με την σύνδεση αυτή, ο χρήστης έχει την δυνατότητα του ελέγχου ροής, της κυκλωματικής διάταξης που θέλει να αναπτύξει στο ολοκληρωμένο κύκλωμα FPGA. Η διαμόρφωση του κυκλώματος FPGA, πραγματοποιείται μέσω της φυσικής σύνδεσης του αναπτυξιακού τυπωμένου κυκλώματος (JTAG), με ειδικό λογισμικό περιβάλλον ανάπτυξης εφαρμογών κυκλωμάτων FPGA (Quartus Prime). Το συγκεκριμένο λογισμικό περιβάλλον ανάπτυξης που θα χρησιμοποιηθεί στην παρούσα εργασία, αναφέρεται στη συνέχεια. Επίσης, τα σχηματικά διαγράμματα των δύο τυπωμένων κυκλωμάτων, παρατίθενται στο Παράρτημα Ι.

3 Σχεδίαση εφαρμογών για κυκλώματα FPGA.

3.1 Τα στάδια σχεδίασης εφαρμογών στο περιβάλλον ανάπτυξης εφαρμογών Quartus Prime.

Για τον σχεδιασμό και την ανάπτυξη εφαρμογών σε κυκλώματα FPGA της εταιρείας Intel, δημιουργήθηκε ένα ευέλικτο λογισμικό περιβάλλον, που ονομάζεται Quartus Prime. Το συγκεκριμένο λογισμικό εμπεριέχει εργαλεία για την ανάπτυξη του κώδικα περιγραφής υλικού (HDL) και για την προσομοίωση λειτουργικότητας του σχεδιασμού. Το περιβάλλον αυτό μας δίνει επίσης την δυνατότητα να χρησιμοποιήσουμε IP cores, που όπως προαναφέραμε, είναι προσχεδιασμένο λογισμικό για τον έλεγχο και την παραμετροποίηση των modules που μπορούν να εμπεριέχονται σε ένα κύκλωμα FPGA. Στο Quartus Prime έχουν ενσωματωθεί όλα τα στάδια δημιουργίας και ανάπτυξης εφαρμογών για τα κυκλώματα FPGA της εταιρείας Intel. Στο παρακάτω διάγραμμα ροής (Σχήμα 7) εμφανίζονται τα βασικά στάδια της σχεδίασης μιας εφαρμογής για κύκλωμα FPGA (Ιωάννης Α. Καλόμοιρος, 2012).



Σχήμα 7: Βασικά στάδια σχεδίασης εφαρμογής για κύκλωμα FPGA

Κατά το πρώτο στάδιο ο σχεδιαστής ψηφιακών συστημάτων, αναλύει τις απαιτήσεις και τις ανάγκες της εφαρμογής. Μελετά και οργανώνει τον τρόπο ψηφιακής υλοποίησης

με την δημιουργία διαγραμμάτων ροής για κάθε επιμέρους απαίτηση. Τα επιμέρους διαγράμματα, συνενώνονται σε μια ολοκληρωμένη ολότητα η οποία ανταποκρίνεται σε όλες τις απαιτήσεις της ψηφιακής εφαρμογής.

Στο δεύτερο στάδιο ο σχεδιαστής μετατρέπει όλες τις επιμέρους οντότητες του συνολικού διαγράμματος ροής, σε κώδικα περιγραφής κυκλωμάτων HDL, έτσι ώστε κάθε κύκλωμα που θα προκύπτει, να συμπεριφέρεται σύμφωνα με τις απαιτήσεις της εφαρμογής. Κατά το στάδιο αυτό περιγράφονται επίσης και οι συνδέσεις των επιμέρους κυκλωμάτων μεταξύ τους έτσι ώστε να δημιουργηθεί το τελικό σύστημα που θα υλοποιεί την εφαρμογή.

Η ανάλυση και η σύνθεση είναι ένα στάδιο κατά το οποίο το λογισμικό περιβάλλον ανάπτυξης των εφαρμογών (Quartus Prime) αναλύει την περιγραφή των κυκλωμάτων και προσπαθεί να δημιουργήσει την βέλτιστη κυκλωματική υλοποίηση.

Η λειτουργικότητα της κυκλωματικής υλοποίησης που θα προκύψει από την διαδικασία της ανάλυσης και σύνθεσης, ελέγχεται στο στάδιο της λειτουργικής προσομοίωσης. Υπάρχουν ενσωματωμένα εργαλεία ψηφιακής προσομοίωσης των κυκλωμάτων που δημιουργούνται έτσι ώστε ο σχεδιαστής να μελετήσει την ορθή συμπεριφορά του συστήματος και να επέμβει με διορθώσεις στον κώδικα περιγραφής κυκλωμάτων εάν αυτό είναι απαραίτητο.

Στο στάδιο της δρομολόγησης σε επίπεδο υλικού, το λογισμικό περιβάλλον ανάπτυξης των εφαρμογών (Quartus Prime) «σχεδιάζει» την κυκλωματική υλοποίηση πάνω στους διαθέσιμους υλικούς πόρους του κυκλώματος FPGA που έχει επιλεγεί για την συγκεκριμένη εφαρμογή. Δημιουργεί, με άλλα λόγια, την σύνδεση των λογικών μονάδων LEs ανάμεσα στα LABs και τις παραμετροποιεί έτσι ώστε να υλοποιηθεί η κυκλωματική περιγραφή.

Κατά το στάδιο της χρονικής ανάλυσης το λογισμικό περιβάλλον ανάπτυξης των εφαρμογών (Quartus Prime) αναλύει τον συγχρονισμό των κυκλωμάτων που περιεγράφηκαν από τον κώδικα HDL. Αναλύει, ελέγχει και συντονίζει όλες τις πιθανές πηγές χρονισμού (clocks, PLLs), καθώς και όλες τις σύγχρονες κυκλωματικές διαδικασίες, σύμφωνα με την κυκλωματική περιγραφή.

Το στάδιο της χρονικής ανάλυσης ακολουθεί η χρονική προσομοίωση. Κατά την διαδικασία αυτή ο σχεδιαστής ψηφιακών συστημάτων ελέγχει τον χρόνο εκτέλεσης και τον συγχρονισμό των κυκλωματικών διαδικασιών και επεμβαίνει με τροποποιήσεις στον κώδικα περιγραφής κυκλωμάτων, εφόσον αυτό κρίνεται απαραίτητο, έτσι ώστε να επιτύχει την βέλτιστη κυκλωματική υλοποίηση.

Έπειτα από την επιβεβαίωση της ορθής λειτουργικότητας της ψηφιακής σχεδίασης μέσα από την κυκλωματική και χρονική προσομοίωση, ο σχεδιαστής του συστήματος είναι έτοιμος για την φυσική αποτύπωση του κυκλώματος πάνω στο κύκλωμα FPGA που έχει επιλέξει. Το λογισμικό περιβάλλον ανάπτυξης των εφαρμογών (Quartus Prime) αναλαμβάνει την φυσική διαμόρφωση των λογικών μονάδων LEs του υλικού, καθώς και την διασύνδεση μεταξύ τους έτσι ώστε να υλοποιηθεί το τελικό κύκλωμα που έχει μελετηθεί.

Τα διαδικαστικά βήματα που απαιτούνται για την υλοποίηση μιας εφαρμογής σε κύκλωμα FPGA παρουσιάζονται αργότερα.

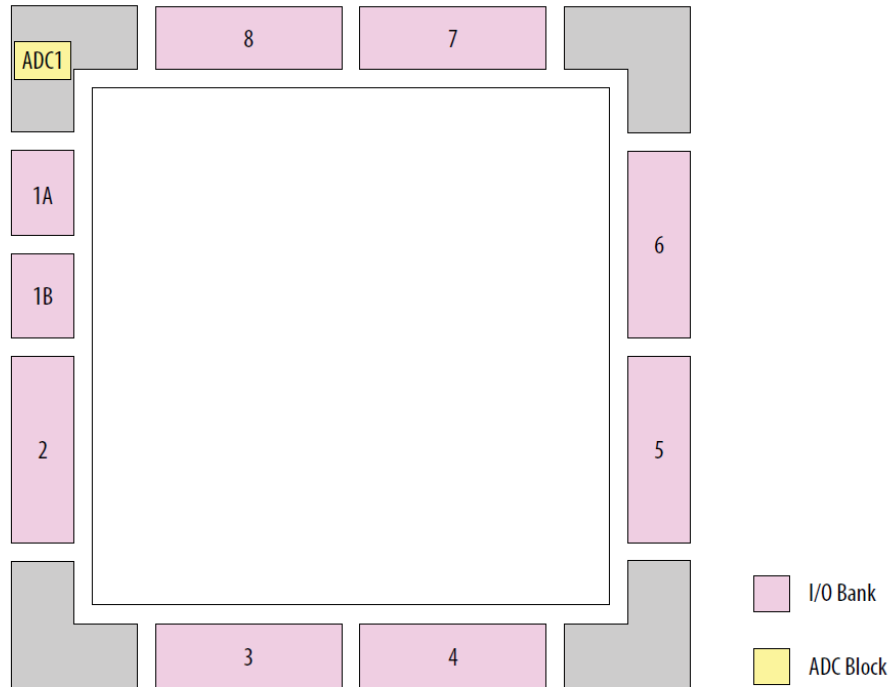
4 Ο ενσωματωμένος αναλογικός/ψηφιακός μετατροπέας (ADC block) του κυκλώματος FPGA 10M08DAF256C8G

4.1 Οι κατηγορίες αναλογικών/ψηφιακών μετατροπέων της οικογένειας MAX10.

Υπάρχουν δύο κατηγορίες κυκλωμάτων FPGA, της οικογένειας MAX 10, που διαθέτουν ενσωματωμένο αναλογικό/ψηφιακό μετατροπέα. Στην πρώτη κατηγορία τα κυκλώματα διαθέτουν δύο ενσωματωμένες μονάδες αναλογικής/ψηφιακής μετατροπής με συνολικά 18 αναλογικές εισόδους. Τα κυκλώματα FPGA της δεύτερης κατηγορίας διαθέτουν μία μονάδα αναλογικής/ψηφιακής μετατροπής με συνολικά 17 αναλογικές εισόδους. Το κύκλωμα FPGA που θα χρησιμοποιηθεί στην παρούσα εργασία 10M08DAF256C8G, ανήκει στην δεύτερη κατηγορία (Intel, 2017β).

4.2 Δειγματοληψία των αναλογικών καναλιών του αναλογικού/ψηφιακού μετατροπέα

Από σύνολο των 17 ακροδεκτών αναλογικής/ψηφιακής μετατροπής, οι 16 ακροδέκτες είναι διπλής λειτουργίας, δηλαδή μπορούν να χρησιμοποιηθούν και ως γενικές εισοδοί/έξοδοι και μόνο ο ένας ακροδέκτης μπορεί να χρησιμοποιηθεί αποκλειστικά για αναλογική/ψηφιακή μετατροπή. Επίσης υπάρχει και ένα εσωτερικό κανάλι αναλογικής/ψηφιακής μετατροπής το οποίο δεν συνδέεται με εξωτερικό ακροδέκτη αλλά με ενσωματωμένο αισθητήρα θερμοκρασίας. Στο Σχήμα 8 απεικονίζεται η θέση του αναλογικού/ψηφιακού μετατροπέα (ADC1) μέσα στο κύκλωμα FPGA.

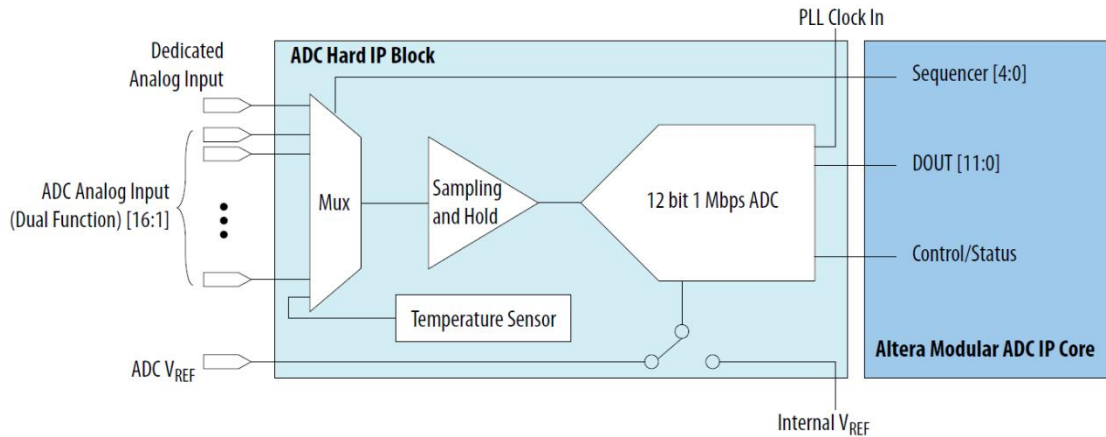


Σχήμα 8. Η θέση του αναλογικού/ψηφιακού μετατροπέα (ADC1) μέσα στο κύκλωμα FPGA

Το ADC block έχει την δυνατότητα συνολικής δειγματοληψίας και για τα 17 αναλογικά κανάλια που διαθέτει, που ανέρχεται στο 1000000 δείγματα το δευτερόλεπτο (1MSPS). Αυτό σημαίνει ότι σε περίπτωση ταυτόχρονης δειγματοληψίας 2 καναλιών, η μέγιστη δειγματοληψία ανά κανάλι θα είναι 0.5MSPS και ούτω καθεξής. Η δειγματοληψία του εσωτερικού αναλογικού καναλιού, για την μέτρηση της θερμοκρασίας από τον ενσωματωμένο αισθητήρα, ανέρχεται στα 50KSPS. Για να επιτευχθούν οι προαναφερθείσες συχνότητες δειγματοληψίας πρέπει το ADC block να τροφοδοτηθεί με τετραγωνικούς παλμούς στην είσοδο clock που διαθέτει. Η συχνότητα αυτή πρέπει να δημιουργηθεί από ανεξάρτητη εξωτερική γεννήτρια τετραγωνικών παλμών (clock), με διαμόρφωση της συχνότητας μέσω της συσκευής PLL (Phase Locked Loop), που υπάρχει ενσωματωμένη στο κύκλωμα FPGA. Η συχνότητα της γεννήτριας τετραγωνικών παλμών για το συγκεκριμένο αναπτυξιακό τυπωμένο κύκλωμα MAXimator που θα χρησιμοποιηθεί στην παρούσα εργασία είναι 10MHz. Η διαδικασία χρονοισμού του ADC block θα αναλυθεί αργότερα (Intel, 2017β).

4.3 Ακρίβεια της αναλογικής/ψηφιακής μετατροπής και η τάση αναφοράς.

Η ακρίβεια μετατροπής του ADC block είναι 12bits και η μέθοδος μετατροπής που χρησιμοποιείται είναι η SAR (Successive Approximation Register). Το διάγραμμα του ADC block απεικονίζεται στο Σχήμα 9.



Σχήμα 9. Διάγραμμα του ADC Block

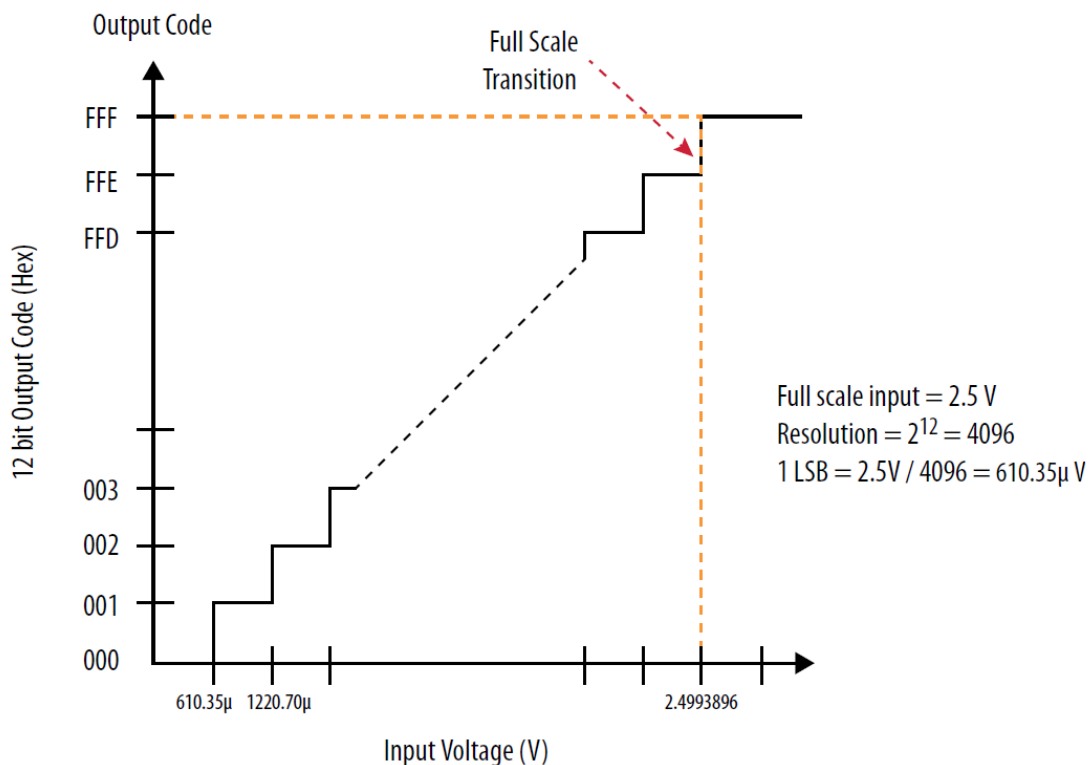
Στο διάγραμμα του ADC block μπορούμε να διακρίνουμε τον πολυπλέκτη (Mux) των 17 εξωτερικών αναλογικών καναλιών (16 dual function +1 dedicated analog), καθώς και το εσωτερικό κανάλι του ενσωματωμένου αισθητήρα θερμοκρασίας (Temperature Sensor). Επίσης στο διάγραμμα παρουσιάζεται και η διασύνδεση του υλικού ADC block, με το λογισμικό IP core που όπως έχουμε προαναφέρει χρησιμοποιείται για τον έλεγχο του υλικού ADC block. Το Altera Modular IP Core με τις παραμέτρους του, θα χρησιμοποιηθεί για τον έλεγχο και την ρύθμιση λειτουργίας του ADC block στην παρούσα εργασία.

Όπως μπορούμε επίσης να παρατηρήσουμε στο διάγραμμα, υπάρχει η τάση αναφοράς V_{ref} , η οποία χρησιμοποιείται ως αναφορά, για την αναλογική/ψηφιακή μετατροπή ενός επιπέδου τάσης V_{in} , σύμφωνα με τον παρακάτω μαθηματικό τύπο:

$$\Psi\eta\phi\iota\alpha\acute{\kappa}\eta\ T\iota\mu\acute{\eta} = \left(\frac{V_{in}}{V_{ref}} \right) \times 2^{12}$$

Η τάση αναφοράς V_{ref} για το συγκεκριμένο αναπτυξιακό τυπωμένο κύκλωμα MAXimator που θα χρησιμοποιηθεί στην παρούσα εργασία, είναι εξωτερική στην τιμή των 2.5V. Σε κάθε κύκλωμα FPGA της οικογένειας MAX 10, υπάρχει ένας ακροδέκτης ο

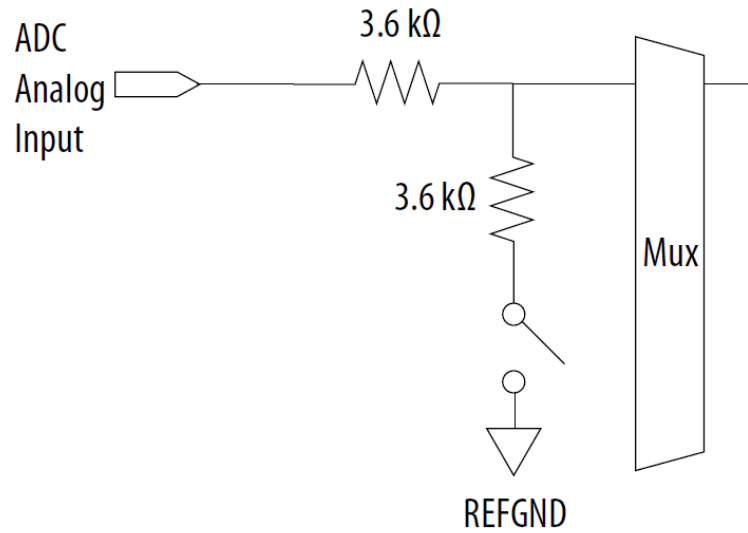
οποίος μπορεί να δεχθεί αναλογική τάση ως τάση αναφοράς V_{ref} για τον αναλογικό/ψηφιακό μετατροπέα. Το ADC block διαθέτει επίσης και εσωτερική τάση αναφοράς (Internal V_{ref}), η οποία μπορεί να επιλεγεί κατά την παραμετροποίηση του IP core. Στο παρακάτω διάγραμμα (Σχήμα 10), απεικονίζεται η μεταβολή της Ψηφιακής Τιμής σε δεκαεξαδική μορφή, σε συνάρτηση με την αναλογική τάση εισόδου V_{in} .



Σχήμα 10. Διάγραμμα του δεκαεξαδικού αποτελέσματος της αναλογικής/ψηφιακής μετατροπής σε συνάρτηση με την τάση εισόδου.

Για κάθε ADC block υπάρχει και η δυνατότητα της ενεργοποίησης ενός διαιρέτη τάσης εισόδου (prescaler) ο οποίος βρίσκεται πριν την είσοδο του σήματος στον πολυπλέκτη σημάτων που οδηγεί τον αναλογικό/ψηφιακό μετατροπέα του ADC block. Η δυνατότητα ενεργοποίησης του διαιρέτη τάσης εισόδου είναι διαθέσιμη μόνο σε συγκεκριμένους ακροδέκτες εισόδου του αναλογικού σήματος. Για το συγκεκριμένο

κύκλωμα της εφαρμογής ο διαιρέτης τάσης μπορεί να ενεργοποιηθεί για τους ακροδέκτες των αναλογικών καναλιών 8 και 16. Με την ενεργοποίηση αυτή δίνεται η δυνατότητα στον αναλογικό/ψηφιακό μετατροπέα να μετατρέπει τάσεις ως 3.6V. Στο Σχήμα 11 απεικονίζεται ο διαιρέτης τάσης εισόδου του ADC block (Intel, 2017β).

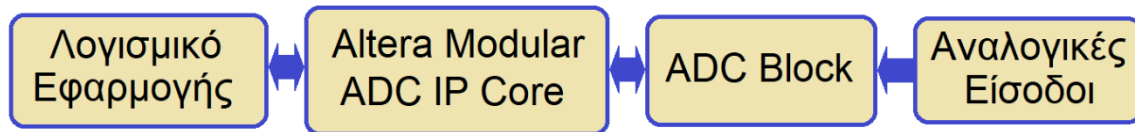


Σχήμα 11. Ενσωματωμένος διαιρέτης τάσης εισόδου

5 Ο έλεγχος του αναλογικού/ψηφιακού μετατροπέα.

5.1 5.1 Ο προσχεδιασμένος πυρήνας ελέγχου Altera Modular ADC IP Core.

Για την διευκόλυνση του λειτουργικού ελέγχου του ADC block, η Intel (Altera) έχει αναπτύξει λογισμικό σε κώδικα HDL (Altera Modular ADC IP Core) το οποίο έχει την δυνατότητα να παραμετροποιεί το ADC block σε επίπεδο υλικού (ADC hard IP core). Το συγκεκριμένο λογισμικό είναι οργανωμένο σε μια ενιαία οντότητα (modular). Η οντότητα του λογισμικού διαθέτει διεπαφή σύνδεσης με το υλικό του ADC block από την μια πλευρά και από την άλλη πλευρά διαθέτει αντίστοιχη διεπαφή με την υλοποίηση των κυκλωμάτων που θα προκύψουν από το λογισμικό της εφαρμογής (HDL κώδικας χρήστη). Μέσω της διεπαφής με το λογισμικό εφαρμογής μπορεί να παραμετροποιηθεί το ADC block για να επιτελέσει τις διάφορες λειτουργίες που απαιτούνται από τα υλοποιημένα κυκλώματα που θα προκύψουν από το λογισμικό της εφαρμογής. Ο ρόλος του Altera Modular ADC IP core περιγράφεται σχηματικά στο Σχήμα 12.



Σχήμα 12. Η διασύνδεση του λογισμικού εφαρμογής με το ADC Block μέσω του Altera Modular ADC IP core.

5.2 Οι μικροπυρήνες ελέγχου του Altera Modular ADC IP Core.

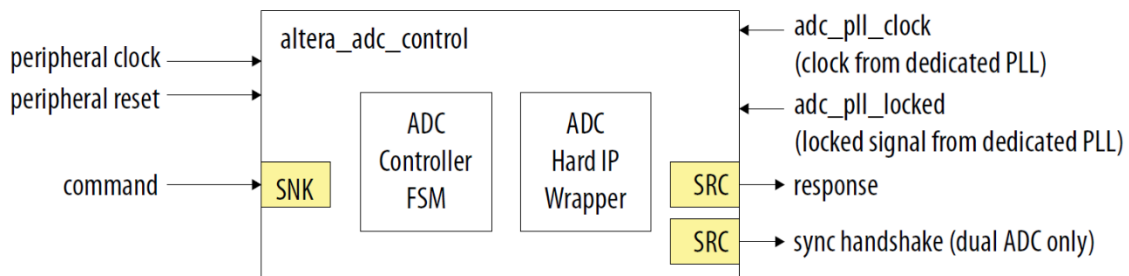
Ο πυρήνας λογισμικού (Altera Modular ADC IP Core) εμπεριέχει τέσσερις μικρότερους πυρήνες λογισμικού (micro cores):

- ADC control core
- Sequencer core
- Sample Storage core
- Threshold Detection core

5.2.1 ADC control core.

Ο συγκεκριμένος micro core αποτελεί τον βασικό μικροπυρήνα ελέγχου του ADC block σε επίπεδο υλικού. Λαμβάνει όλες τις εντολές από τα υλοποιημένα κυκλώματα της

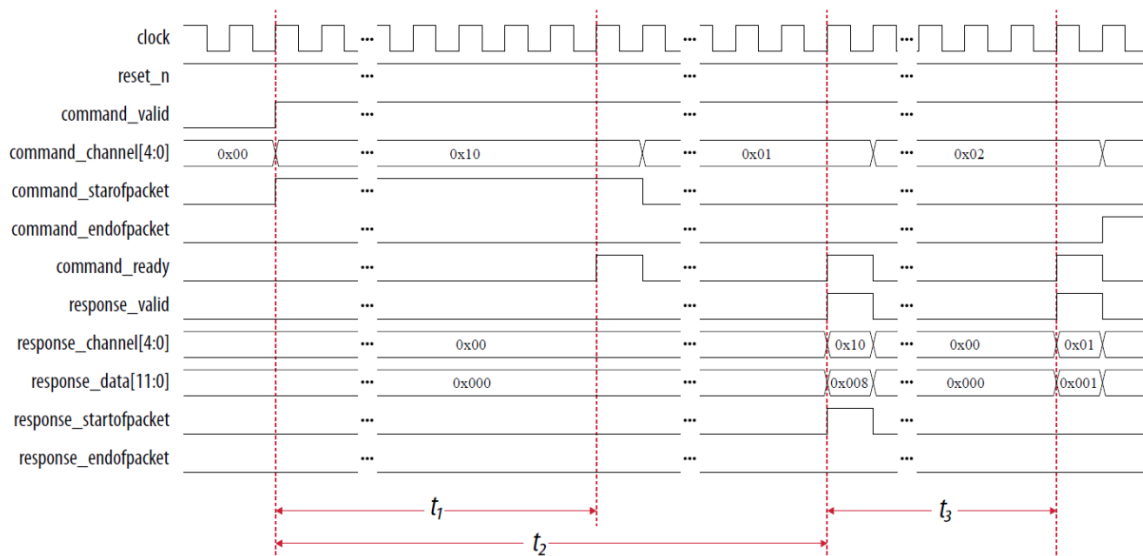
εφαρμογής και ελέγχει αναλόγως το υλικό του ADC block. Με τον τρόπο αυτό μέσω του ADC control πραγματοποιείται η επιλογή καναλιού για την ψηφιακή/αναλογική μετατροπή από τις διαθέσιμες αναλογικές εισόδους (analog pins) ή από το κανάλι του ενσωματωμένου αισθητήρα θερμοκρασίας, καθώς και η ενεργοποίηση ή η απενεργοποίηση (power up/power down) του ADC block. Επίσης μέσω του ADC control πραγματοποιείται και η λήψη των αποτελεσμάτων της αναλογικής/ψηφιακής μετατροπής. Για την μεταβίβαση των εντολών ελέγχου (commands) από την εφαρμογή προς το ADC block καθώς και για την μεταβίβαση των αποκρίσεων (responses) από το ADC block προς την εφαρμογή, έχουν αναπτυχθεί ανεξάρτητοι δίαυλοι δεδομένων. Για την εξυπηρέτηση της λειτουργικότητας του ADC control, υπάρχει η γραμμή διαύλου (data bus) που μεταφέρει δεδομένα από τα υλοποιημένα κυκλώματα της εφαρμογής προς το ADC control και ονομάζεται **command** και η γραμμή δίαυλου (data bus) που μεταφέρει τις δεδομένα των αποκρίσεων του ADC control προς τα υλοποιημένα κυκλώματα της εφαρμογής και ονομάζεται **response**. Οι δίαυλοι command και response συμβάλουν στην ταχύτατη και αξιόπιστη διασύνδεση του ADC block με εξωτερικές ψηφιακές μονάδες, καθώς έχουν την δυνατότητα να μεταφέρουν τα δεδομένα με παράλληλο τρόπο και σύγχρονα. Η σχηματική απεικόνιση του ADC control παρουσιάζεται στο Σχήμα 13 :



Σχήμα 13. Σχηματική απεικόνιση του μικροπυρήνα ADC control

Στην συγκεκριμένο διάγραμμα απεικονίζεται η είσοδος του διαύλου δεδομένων **command** που μεταφέρει τα δεδομένα ελέγχου από τα υλοποιημένα κυκλώματα εφαρμογής προς το ADC control. Το ADC control με την σειρά του θα ρυθμίσει κατάλληλα την λειτουργία του ADC block έτσι ώστε να ανταποκριθεί στις απαιτήσεις των κυκλωμάτων εφαρμογής (επιλογή καναλιού αναλογικής/ψηφιακής μετατροπής, αίτηση άμεσης αναλογικής/ψηφιακής μετατροπής κ.α.). Επίσης στο διάγραμμα διακρίνεται και η έξοδος των δεδομένων απόκρισης **response** από το ADC control προς τα υλοποιημένα

κυκλώματα της εφαρμογής. Υπάρχουν επίσης και τα σήματα χρονισμού, καθώς όπως προαναφέρθηκε η μετάδοση των δεδομένων είναι σύγχρονη και ο ρυθμός δειγματοληψίας του ADC block βασίζεται επίσης σε σήματα που προέρχονται από την έξοδο κυκλωμάτων PLL (Phase Locked Loop). Με τον τρόπο αυτό η αίτηση για αναλογική σε ψηφιακή μετατροπή ενός συγκεκριμένου αναλογικού καναλιού, η δειγματοληψία του καναλιού και η αποστολή (απόκριση) του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής είναι συγχρονισμένες με βάση το παρακάτω χρονικό διάγραμμα (Σχήμα 14).



Σχήμα 14. Διάγραμμα χρονισμού αναλογικών/ψηφιακών μετατροπών του ADC Block.

Στο διάγραμμα παρουσιάζεται η επικοινωνία των υλοποιημένων κυκλωμάτων εφαρμογής με τον πυρήνα ADC control. Στα 5 bits του διαύλου **command_channel** (command_channel[4:0]), η εφαρμογή μεταβιβάζει τον κωδικό του αναλογικού καναλιού στο οποίο θα πραγματοποιηθεί η ψηφιακή μετατροπή. Η γραμμή **command_startofpacket** ενημερώνει το ADC control για την έναρξη μεταβίβασης δεδομένων στον δίαυλο command. Με την γραμμή **command_ready** το ADC control ενημερώνει την εφαρμογή για την παραλαβή των έγκυρων δεδομένων προς επεξεργασία από τον δίαυλο command. Ο χρόνος για να πραγματοποιηθεί η ενημέρωση για την παραλαβή των έγκυρων δεδομένων από την στιγμή έναρξης της μετάδοσής τους, είναι t_1 . Από την χρονική στιγμή που το ADC control αναγνωρίσει το κανάλι στο οποίο θα πραγματοποιηθεί η αναλογική/ψηφιακή μετατροπή, ενημερώνει το ADC block να

ξεκινήσει την δειγματοληψία και την μετατροπή. Μόλις ολοκληρωθεί η διαδικασία της αναλογικής/ψηφιακής μετατροπής το ADC control μεταφέρει τον κωδικό του αναλογικού καναλιού στο οποίο πραγματοποιήθηκε η μετατροπή, στα 5 bits του διαύλου **response_channel** (response_channel[4:0]) και ταυτόχρονα μεταφέρει το αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής (12 bits) του συγκεκριμένου καναλιού, στον δίαυλο **response_data** (response_data[11:0]). Ο χρόνος που απαιτείται για να πραγματοποιηθεί η αναλογική/ψηφιακή μετατροπή από την χρονική στιγμή της μεταβίβασης του κωδικού καναλιού στον δίαυλο command_channel είναι t_2 . Η χρονική διάρκεια μεταξύ δύο διαδοχικών αποκρίσεων αναλογικής/ψηφιακής μετατροπής είναι t_3 . Οι χρονικές παράμετροι t_1 , t_2 , t_3 του διαγράμματος υπολογίζονται με βάση τον παρακάτω πίνακα (Πίνακας 1).

Πίνακας 1: Υπολογισμός των χρονικών περιόδων t_1 , t_2 και t_3 .

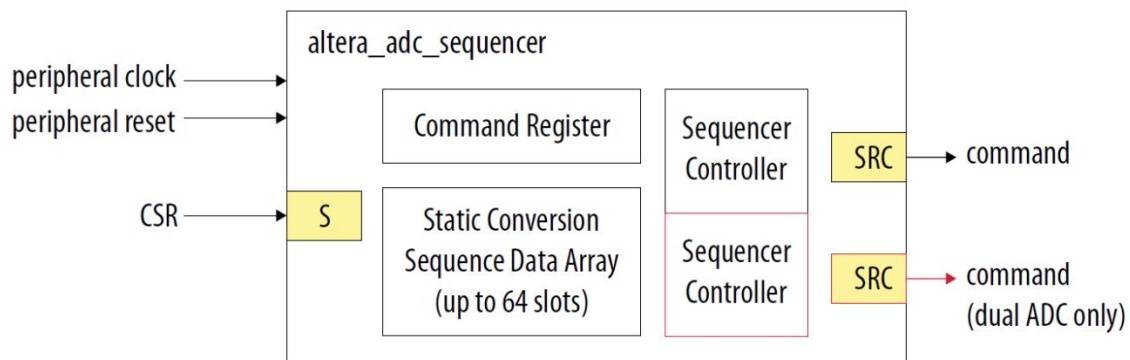
Παράμετρος	Χρονικός Υπολογισμός
t_1	$3 \times \text{ADC soft IP clock} + \frac{2}{\text{Sampling rate}}$
t_2	$3 \times \text{ADC soft IP clock} + \frac{2}{\text{Sampling rate}} + \frac{1}{\text{Sampling rate}}$
t_3	$\frac{1}{\text{Sampling rate}}$

Το ADC control αποτελεί τον σημαντικότερο μικροπυρήνα του Modular ADC IP core.

5.2.2 Sequencer core.

Το ADC block διαθέτει και λειτουργία διαδοχικής σάρωσης των αναλογικών καναλιών προς ψηφιακή μετατροπή με βάση ένα συγκεκριμένο πρόγραμμα αλληλουχίας. Το πρόγραμμα διαδοχικής μετατροπής ονομάζεται Sequencer και η παραμετροποίησή του πραγματοποιείται μέσα από τον Sequencer micro core του Modular ADC IP core. Το Sequencer διαθέτει 64 θέσεις (slots) στις οποίες μπορεί να δηλωθεί η σειρά των αναλογικών καναλιών με την οποία διαδοχικά θα πραγματοποιείται η αναλογική/ψηφιακή μετατροπή. Σε κάθε θέση του sequencer μπορεί να δηλωθεί ένα μόνο κανάλι αλλά το κάθε κανάλι μπορεί να δηλωθεί σε περισσότερες από μία θέσεις μέσα στην διαδοχική

αλληλουχία. Ο ρυθμός δειγματοληψίας του ADC block μοιράζεται στις 64 θέσεις των αναλογικών καναλιών προς ψηφιακή μετατροπή. Θέση ανάμεσα στα διαδοχικά κανάλια μπορεί να καταλάβει και το εσωτερικό κανάλι του αισθητήρα θερμοκρασίας που διαθέτει το ADC block. Όταν το ADC block έχει διαμορφωθεί με ενεργοποιημένη την λειτουργία του Sequencer, τότε το ADC block πραγματοποιεί αναλογική/ψηφιακή μετατροπή στα αναλογικά κανάλια που υπάρχουν στις 64 θέσεις του Sequencer διαδοχικά ξεκινώντας από το κανάλι που βρίσκεται στην πρώτη θέση και τελειώνοντας με το κανάλι της θέσης 64. Η σάρωση των καναλιών μπορεί να είναι συνεχής, δηλαδή έπειτα από την ολοκλήρωση των αναλογικών/ψηφιακών μετατροπών των καναλιών που βρίσκονται στις 64 θέσεις του Sequencer, ξεκινά αυτόματα ο επόμενος κύκλος σάρωσης των 64 θέσεων προς αναλογική ψηφιακή μετατροπή. Επίσης η σάρωση μπορεί να είναι και μεμονωμένη, έπειτα από την ολοκλήρωση του κύκλου σάρωσης των 64 θέσεων του Sequencer προς αναλογική/ψηφιακή μετατροπή το ADC block σταματά και περιμένει νέα εντολή σάρωσης των 64 θέσεων από την εφαρμογή. Το σχηματικό διάγραμμα του Sequencer απεικονίζεται στο Σχήμα 15.

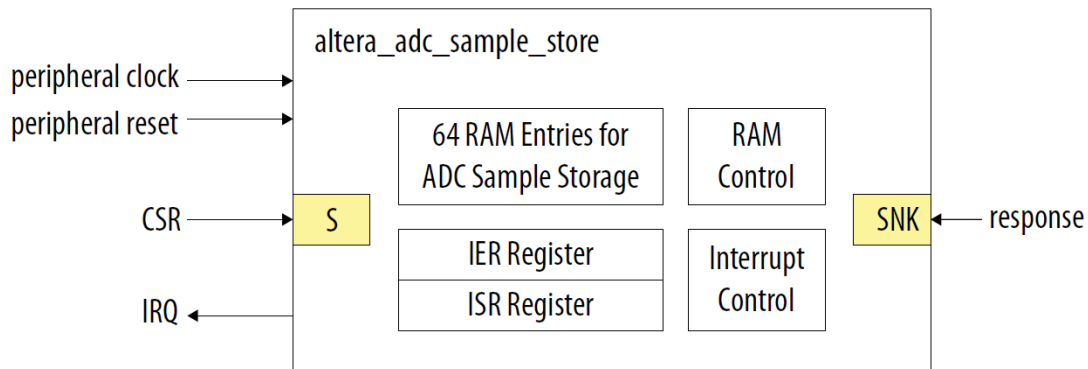


Σχήμα 15. Σχηματική απεικόνιση του ADC sequencer microcore.

Η διαμόρφωση της διάταξης των καναλιών στις 64 θέσεις του Sequencer, πραγματοποιείται κατά την διαδικασία της σχεδίασης της εφαρμογής και δεν μπορεί να αλλάξει κατά την εκτέλεση της εφαρμογής (in run-time). Ο Sequencer αποτελεί ένα χρήσιμο εργαλείο για τις εφαρμογές κατά τις οποίες είναι απαραίτητη η συνεχής παρακολούθηση (monitoring) συγκεκριμένων αναλογικών καναλιών με συγκεκριμένη περιοδικότητα.

5.2.3 Sample Storage core.

Ο μικροπυρήνας Sample Storage δημιουργεί και διαχειρίζεται σε επίπεδο υλικού, μια ενσωματωμένη μνήμη RAM (on-chip RAM), στην οποία εγγράφονται οι αναλογικές/ψηφιακές μετατροπές που προκύπτουν κατά την σάρωση των αναλογικών καναλιών, που βρίσκονται στις 64 θέσεις του Sequencer.



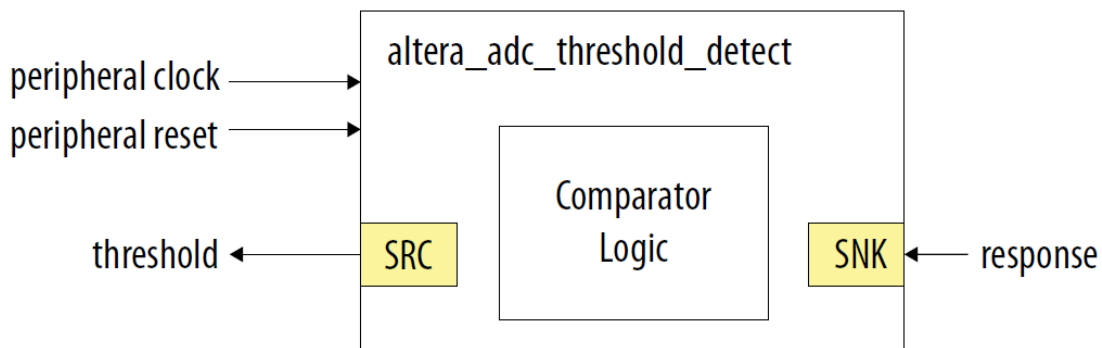
Σχήμα 16. Σχηματική απεικόνιση του ADC sample storage core

Όπως απεικονίζεται και στο Σχήμα 16, το Sample Storage διαθέτει μνήμη RAM για την αποθήκευση των 64 αναλογικών/ψηφιακών μετατροπών που προκύπτουν από την σάρωση των αντίστοιχων θέσεων του Sequencer. Το Sample Storage διαθέτει σύστημα δημιουργίας και διαχείρισης διακοπών (interrupts) σε επίπεδο υλικού, για να ενημερώνει την εφαρμογή για την αποθήκευση νέων τιμών στις θέσεις μνήμης της RAM. Η επικοινωνία του Sample Storage με τις υπόλοιπες ψηφιακές οντότητες πραγματοποιείται μέσω του διαύλου **response** που διαθέτει. Ο διάυλος response είναι μια υλοποίηση (Avalon-MM) του διαύλου μεταφοράς δεδομένων **Avalon** που έχει αναπτύξει η Intel για τα κυκλώματα FPGA. Ο διάυλος μεταφοράς δεδομένων Avalon είναι ένας πολύ ευέλικτος πυρήνας λογισμικού (IP core) ο οποίος μπορεί να παραμετροποιηθεί κατάλληλα κατά την σχεδίαση της εφαρμογής έτσι ώστε να διαμορφώσει κυκλωματικά την φυσική διασύνδεση των διάφορων ψηφιακών μονάδων μέσα σε ένα κύκλωμα FPGA. Ο διάυλος Avalon συμβάλει στην ταχύτερη και αξιόπιστη διασύνδεση των ψηφιακών οντοτήτων που μπορούν να δημιουργηθούν μέσα σε ένα κύκλωμα FPGA, καθώς έχει την δυνατότητα να μεταφέρει τα δεδομένα με παράλληλο τρόπο και σύγχρονα. Υπάρχουν διάφορες διαμορφώσεις του διαύλου Avalon, οι οποίες μπορούν να καλύψουν μεγάλο εύρος

εφαρμογών μεταφοράς δεδομένων. Η συγκεκριμένη υλοποίηση του διαύλου Avalon, για την εξυπηρέτηση της λειτουργικότητας του Sample Storage, είναι η Avalon Memory Mapped Interface (Avalon-MM), η οποία έχει αναπτυχθεί για να εξυπηρετεί εφαρμογές που απαιτούν διευθυνσιοδότηση μνήμης.

5.2.4 Threshold Detection core.

Ο συγκεκριμένος μικροπυρήνας επιτηρεί την τιμή ενός αναλογικού καναλιού και ενημερώνει την εφαρμογή όταν η συγκεκριμένη τιμή βρίσκεται εκτός των προδιαγεγραμμένων ορίων. Τα όρια (μέγιστο και ελάχιστο) προδιαγράφονται κατά την διαδικασία σχεδίασης του μικροπυρήνα και δεν υπάρχει η δυνατότητα αλλαγής τους κατά την εκτέλεση της εφαρμογής (run-time). Η ενημέρωση της εφαρμογής για την παραβίαση των προδιαγεγραμμένων ορίων πραγματοποιείται μέσω του διαύλου threshold (Avalon-ST) όπως απεικονίζεται το Σχήμα 17 (Intel, 2017β).



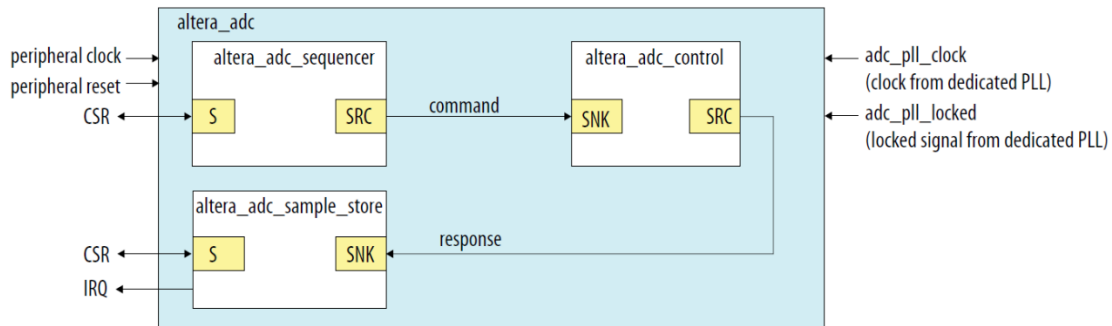
Σχήμα 17. Σχηματική απεικόνιση του ADC threshold detection core

6 Οι διαμορφώσεις του Altera Modular ADC IP core.

Ο πυρήνας Altera Modular ADC IP έχει την δυνατότητα να σχηματίζει διαφορετικές διαμορφώσεις για τον έλεγχο του ADC block, με κατάλληλο συνδυασμό των τεσσάρων μικροπυρήνων που τον αποτελούν. Υπάρχουν τέσσερις διαφορετικές διαμορφώσεις του Altera Modular ADC IP core για έλεγχο του ADC block, κάθε μία από τις οποίες ανταποκρίνεται σε συγκεκριμένες απαιτήσεις των διάφορων εφαρμογών:

6.1 Διαδοχική μετατροπή αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM (Standard Sequencer with Avalon-MM Sample Storage).

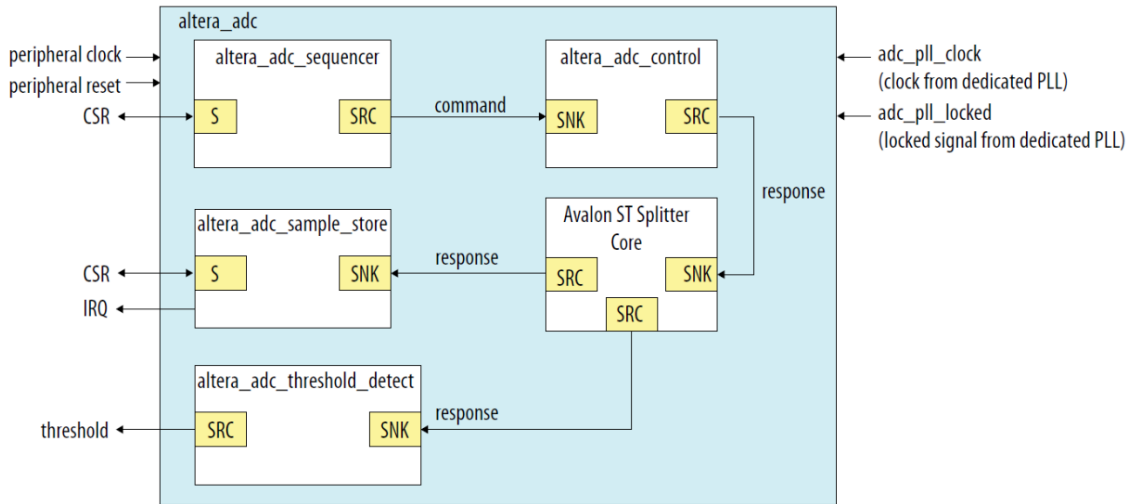
Στη διαμόρφωση αυτή συμμετέχουν το Sequencer micro core, το Sample Storage micro core και το ADC control micro core σύμφωνα με το Σχήμα 18.



Σχήμα 18. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM.

6.2 Διαδοχική μετατροπή αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM και ανίχνευση παραβίασης ορίων (Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection).

Στη διαμόρφωση αυτή συμμετέχουν όλοι οι μικροπυρήνες του Altera ADC IP core. Το Sequencer micro core, το Sample Storage micro core, το Threshold Detection micro core και το ADC control micro core σύμφωνα με το Σχήμα 19.

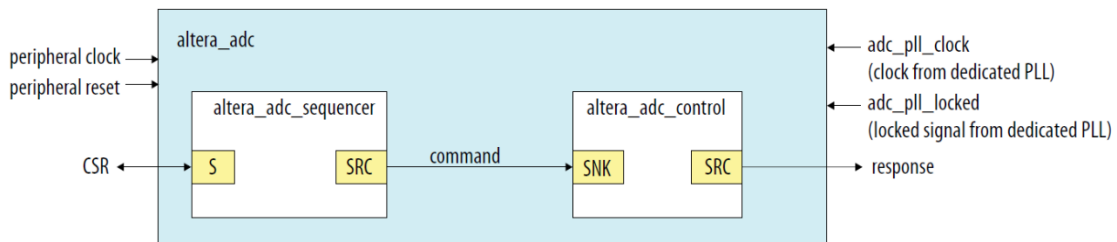


Σχήμα 19. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM και ανίχνευση παραβίασης ορίων

Στη συγκεκριμένη διαμόρφωση λειτουργεί η διαδοχική μετατροπή των προεπιλεγμένων αναλογικών καναλιών (Sequencer), με ταυτόχρονη εγγραφή των αποτελεσμάτων στην ενσωματωμένη μνήμη RAM (Sample Storage) και ανίχνευση παραβίασης προκαθορισμένων ορίων (Threshold Violation Detection). Η επικοινωνία των ψηφιακών οντοτήτων πραγματοποιείται με την υλοποίηση του δίαυλου Avalon-MM.

6.3 Διαδοχική μετατροπή αναλογικών καναλιών με εγγραφή του αποτελέσματος σε εξωτερική μνήμη RAM (Standard Sequencer with external Sample Storage).

Στη διαμόρφωση αυτή συμμετέχει το Sequencer micro core και το ADC control micro core σύμφωνα με το Σχήμα 20.

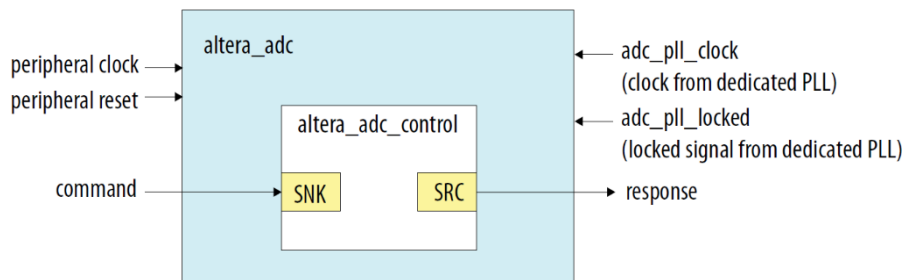


Σχήμα 20. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε εξωτερική μνήμη RAM

Στη συγκεκριμένη διαμόρφωση, λειτουργεί η διαδοχική μετατροπή των προεπιλεγμένων αναλογικών καναλιών (Sequencer) αλλά η υλοποίηση της αποθήκευσης των αποτελεσμάτων καθώς και η ανάπτυξη της ψηφιακής επικοινωνίας πραγματοποιούνται από την εφαρμογή.

6.4 Χρήση μόνο του μικροπυρήνα ADC control core.

Η διαμόρφωση αυτή αποτελεί την πιο απλή διαμόρφωση του Altera Modular ADC IP core καθώς χρησιμοποιεί μόνο το ADC control σύμφωνα με το παρακάτω σχήμα (Σχήμα 21).



Σχήμα 21. Σχηματική απεικόνιση της διαμόρφωσης με χρήση μόνο το ADC control microcore.

Η συγκεκριμένη διαμόρφωση παρέχει στην εφαρμογή την ευελιξία να χρησιμοποιήσει τα σήματα εισόδου/εξόδου του ADC control, σύμφωνα με τις εκάστοτε απαιτήσεις υλοποίησης. Με τον τρόπο αυτό μπορεί η εφαρμογή να αλλάξει τις παραμέτρους ελέγχου (επιλογή καναλιού, έναρξη αναλογικής/ψηφιακής μετατροπής) του ADC block κατά την διάρκεια της εκτέλεσης (run-time). Επίσης αυτή η διαμόρφωση προσφέρει στην εφαρμογή την ευελιξία να διαχειρίζεται και να επεξεργάζεται τα αποτελέσματα της αναλογικής/ψηφιακής μετατροπής σύμφωνα με τις εκάστοτε απαιτήσεις υλοποίησης. Η μεγάλη ευελιξία που παρέχει η συγκεκριμένη διαμόρφωση στον έλεγχο του ADC block, την καθιστά ιδανική για τις εφαρμογές που θα αναπτυχθούν στη συγκεκριμένη εργασία (Intel, 2017β).

7 Δημιουργία απλής εφαρμογής με τη χρήση του Altera Modular ADC IP Core στο περιβάλλον ανάπτυξης εφαρμογών Quartus Prime.

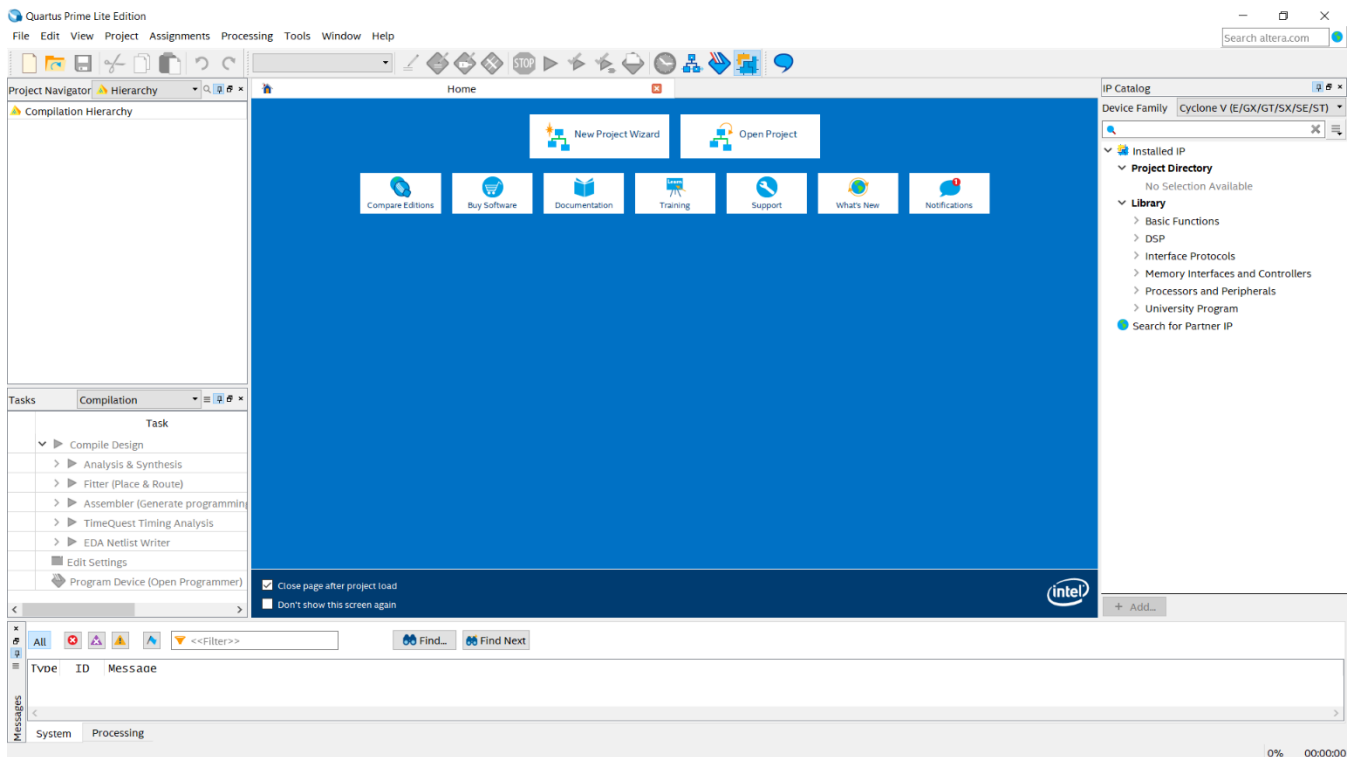
7.1 Δημιουργία νέου Project.

Όπως προαναφέρθηκε η ανάπτυξη εφαρμογών για τα κυκλώματα FPGA της εταιρείας Intel (Altera), πραγματοποιείται μέσα από το περιβάλλον ανάπτυξης εφαρμογών Quartus Prime. Το Quartus Prime έχει σχεδιαστεί ειδικά για να παρέχει τα απαραίτητα εργαλεία λογισμικού, έτσι ώστε η ανάπτυξη εφαρμογών FPGA να πραγματοποιείται γρήγορα, εύκολα και αξιόπιστα. Στην παρούσα ενότητα θα περιγραφεί βήμα προς βήμα η ανάπτυξη μιας απλής εφαρμογής η οποία θα χρησιμοποιεί το ADC block που διαθέτει το κύκλωμα FPGA 10M08DAF256C8GES.



Εικόνα 5: Λογότυπο έναρξης της εφαρμογής Intel Quartus Prime Development Suite 17.0

Με την εκτέλεση της εφαρμογής εμφανίζεται η αρχική διεπαφή του περιβάλλοντος ανάπτυξης Quartus Prime (Εικόνα 6).



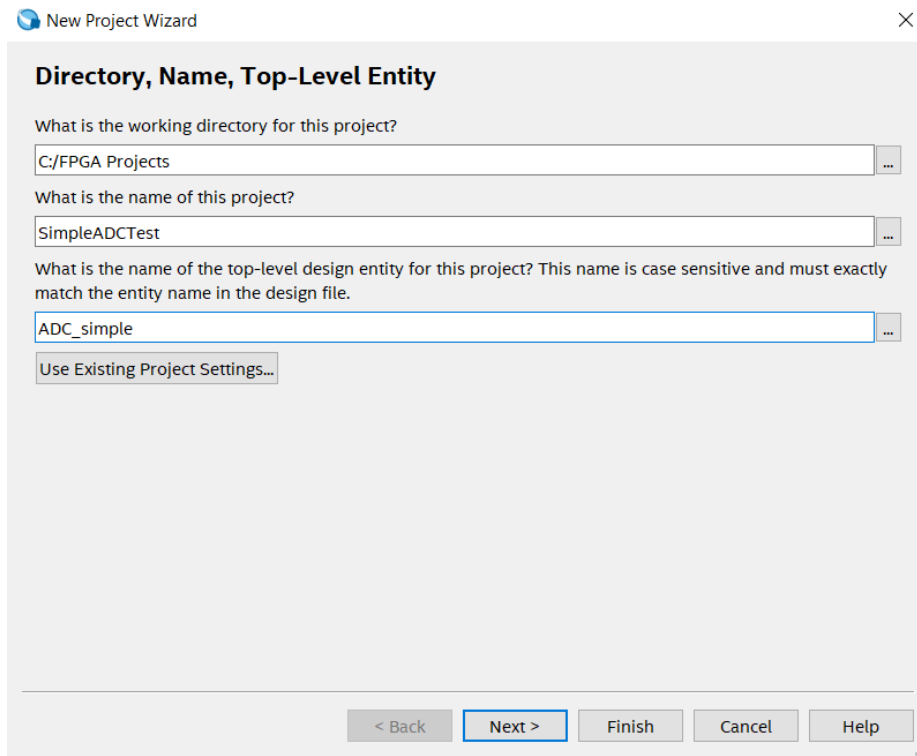
Εικόνα 6: Η αρχική διεπαφή της εφαρμογής :Quartus Prime

Για την δημιουργία νέου project επιλέγουμε από το μενού του αναπτυξιακού περιβάλλοντος:

File → New Project Wizard

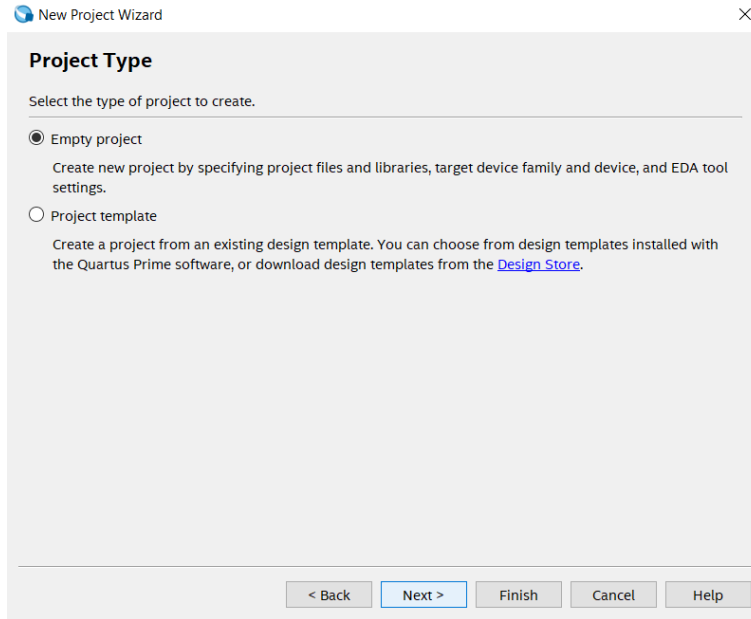
Με την επιλογή αυτή ξεκινά μια αλληλουχία διεπαφών (wizard), μέσω της οποίας πραγματοποιείται η εύκολη δημιουργία και παραμετροποίηση του νέου project. Στα πεδία της πρώτης διεπαφής που εμφανίζεται, εισάγονται ο φάκελος στον οποίο θα δημιουργηθεί η νέα εφαρμογή (project), καθώς επίσης και τα ονόματα της εφαρμογής και της οντότητας του υψηλότερου επιπέδου (top-level entity) για την συγκεκριμένη εφαρμογή. Κάθε project αποτελείται από μία ή περισσότερες οντότητες κώδικα περιγραφής κυκλωμάτων HDL οι οποίες αλληλοεπιδρούν ιεραρχικά. Κάθε οντότητα μπορεί να εμπεριέχει άλλες οντότητες οι οποίες χαρακτηρίζονται ως οντότητες χαμηλότερου επιπέδου και παράλληλα η ίδια οντότητα μπορεί να εμπεριέχεται σε μια άλλη οντότητα η οποία χαρακτηρίζεται ως οντότητα υψηλότερου επιπέδου. Η πρώτη ιεραρχικά οντότητα κώδικα, που εμπεριέχει ολόκληρη την δομή των υπόλοιπων οντοτήτων που συμμετέχουν σε κάποιο project,

ονομάζεται οντότητα υψηλού επιπέδου (top-level entity) και είναι μοναδική για κάθε project. Επιλέγουμε τα επιθυμητά ονόματα των πεδίων ως ακολούθως:

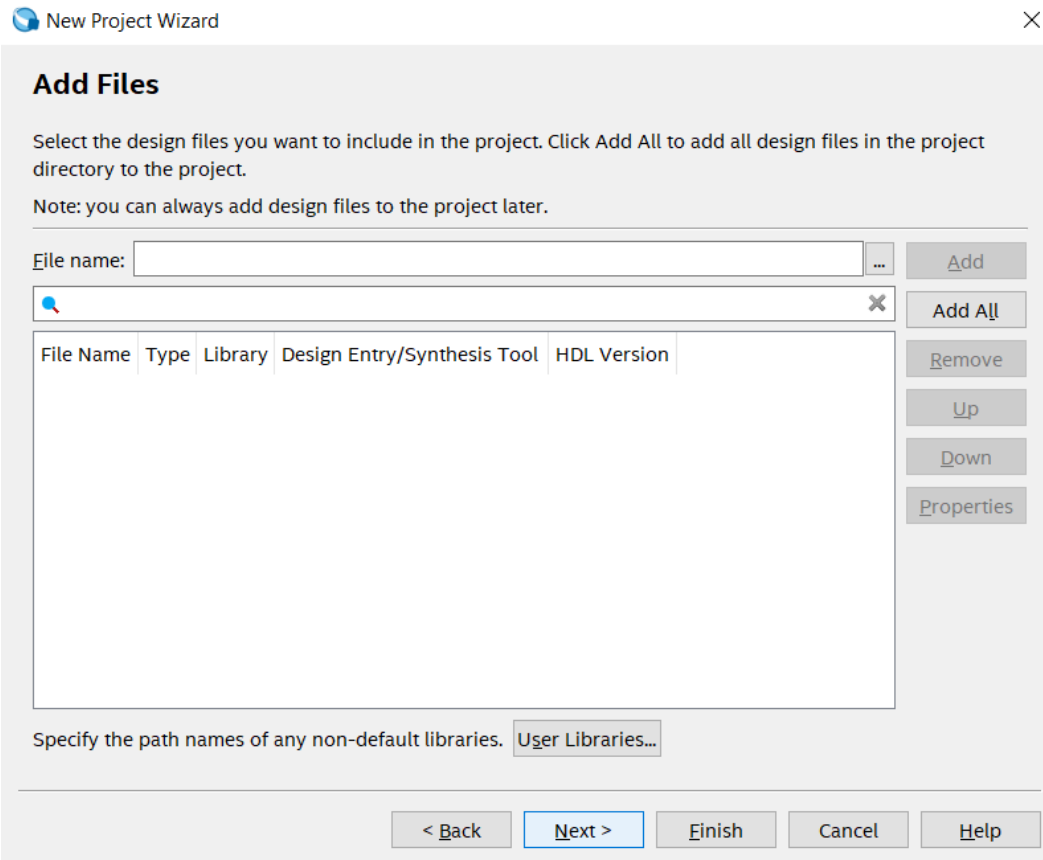


The screenshot shows a 'New Project Wizard' dialog box with the title 'Directory, Name, Top-Level Entity'. It contains three text input fields and a button. The first field is labeled 'What is the working directory for this project?' and contains 'C:/FPGA Projects'. The second field is labeled 'What is the name of this project?' and contains 'SimpleADCTest'. The third field is labeled 'What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.' and contains 'ADC_simple'. Below the fields is a button labeled 'Use Existing Project Settings...'. At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Next >' button is highlighted with a blue border.

Η επόμενη διεπαφή μας δίνει την επιλογή να επιλέξουμε ένα πρότυπο έργου (project template), στο οποίο θα αναπτυχθεί η νέα μας εφαρμογή. Η δική μας εφαρμογή δε θα βασίζεται σε κάποιο προσχεδιασμένο εκμαγείο οπότε επιλέγουμε **Empty Project** και επιλέγουμε **Next**.



Στο επόμενο βήμα μπορούμε να προσθέσουμε στο νέο project αρχεία που περιέχουν οντότητες κώδικα HDL που είναι προσχεδιασμένες, με σκοπό να τις χρησιμοποιήσουμε στη νέα σχεδίαση. Στην δική μας εφαρμογή δεν θα προστεθεί κανένα αρχείο σε αυτό το στάδιο.



Επιλέγουμε **Next** και εμφανίζεται η επόμενη διεπαφή, στην οποία πρέπει να επιλέξουμε τον τύπο του κυκλώματος FPGA που έχουμε ως στόχο (target device), για να υλοποιήσουμε την εφαρμογή.

Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.

You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: MAX 10 DA

Target device

- Auto device selected by the Fitter
 Specific device selected in 'Available devices' list
 Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 256

Core speed grade: 8

Name filter:

Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit ele
10M04DAF256C8G	1.2V	4032	178	178	193536	40
10M08DAF256C8G	1.2V	8064	178	178	387072	48
10M08DAF256C8GES	1.2V	8064	178	178	387072	48
10M16DAF256C8G	1.2V	15840	178	178	562176	90
10M25DAF256C8G	1.2V	24960	178	178	691200	110
10M40DAF256C8G	1.2V	40368	178	178	1290240	250

< Back

Next >

Finish

Cancel

Help

Επιλέγουμε τον τύπο του κυκλώματος FPGA για το οποίο έχει αναπτυχθεί η εφαρμογή. Για την συγκεκριμένη εφαρμογή ο τύπος του κυκλώματος FPGA που επιλέγουμε είναι ο 10M08DAF256C8GES. Επιλέγουμε **Next** και εμφανίζεται η διεπαφή επιλογής των εργαλείων προσομοίωσης.

EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/S...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	<None>	<None>	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back

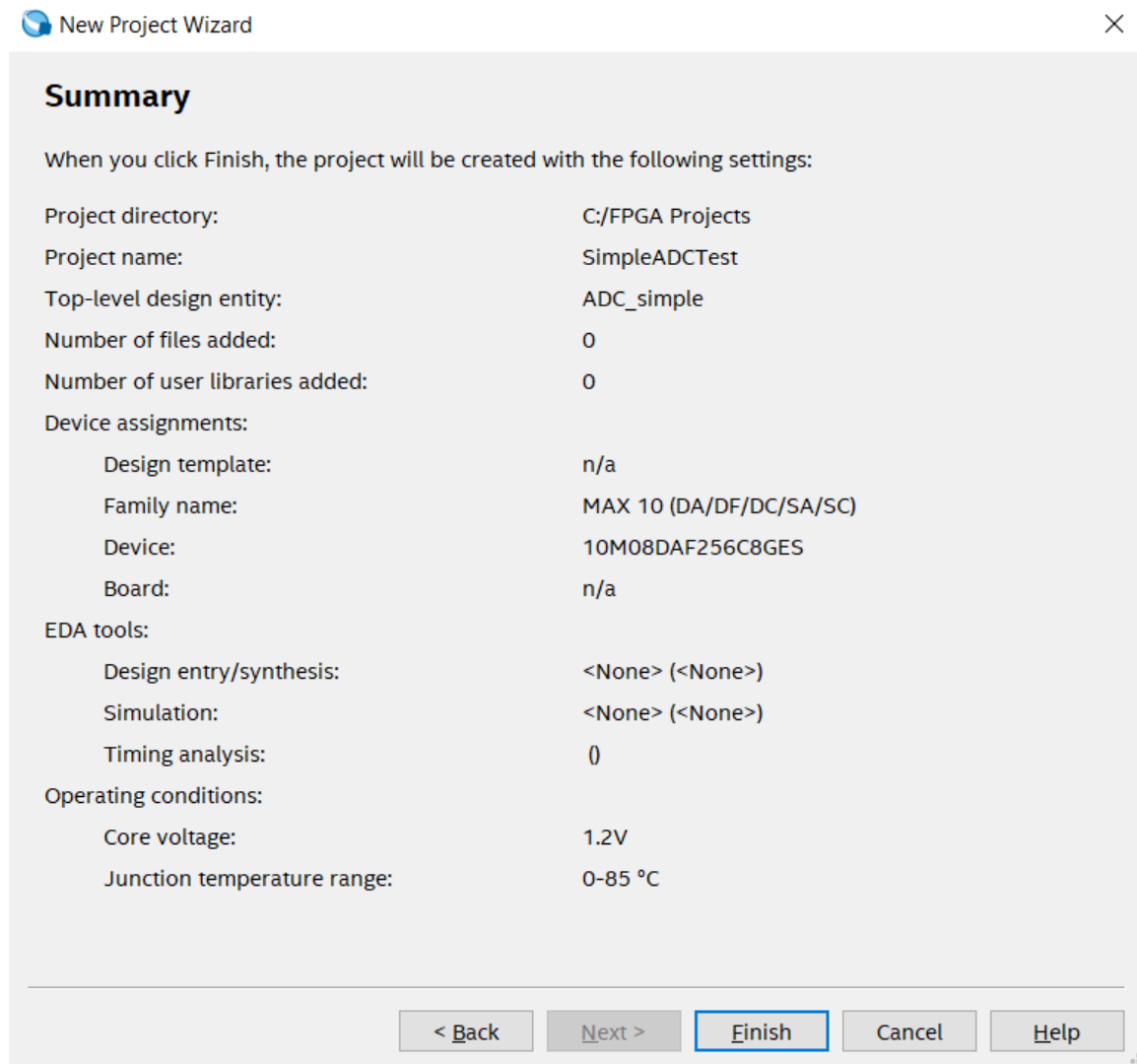
Next >

Finish

Cancel

Help

Προχωρούμε με **Next** και εμφανίζεται μια σύνοψη των επιλογών που πραγματοποιήθηκαν σε όλα τα προηγούμενα βήματα παραμετροποίησης του project.



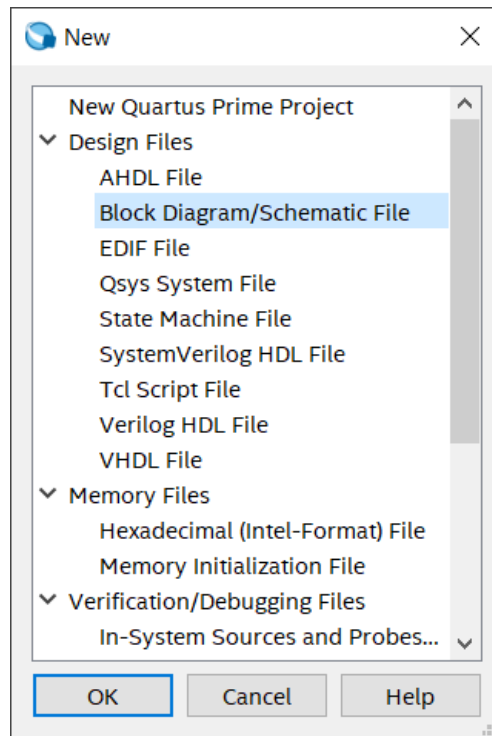
Επιλέγουμε **Finish** και οι ρυθμίσεις αρχικοποίησης του projects έχουν ολοκληρωθεί.

7.2 Δημιουργία ανώτερης οντότητας (Top Entity).

Παρά το γεγονός ότι η αρχικοποίηση του νέου project έχει ολοκληρωθεί, το project παραμένει κενό, χωρίς καμία οντότητα (entity) αναπτυγμένη. Σε αυτό το σημείο μπορεί να δημιουργηθεί η ανώτερη ιεραρχικά οντότητα (top entity) στην οποία θα αναπτυχθεί η εφαρμογή. Για τον σκοπό αυτό, επιλέγουμε:

File → New

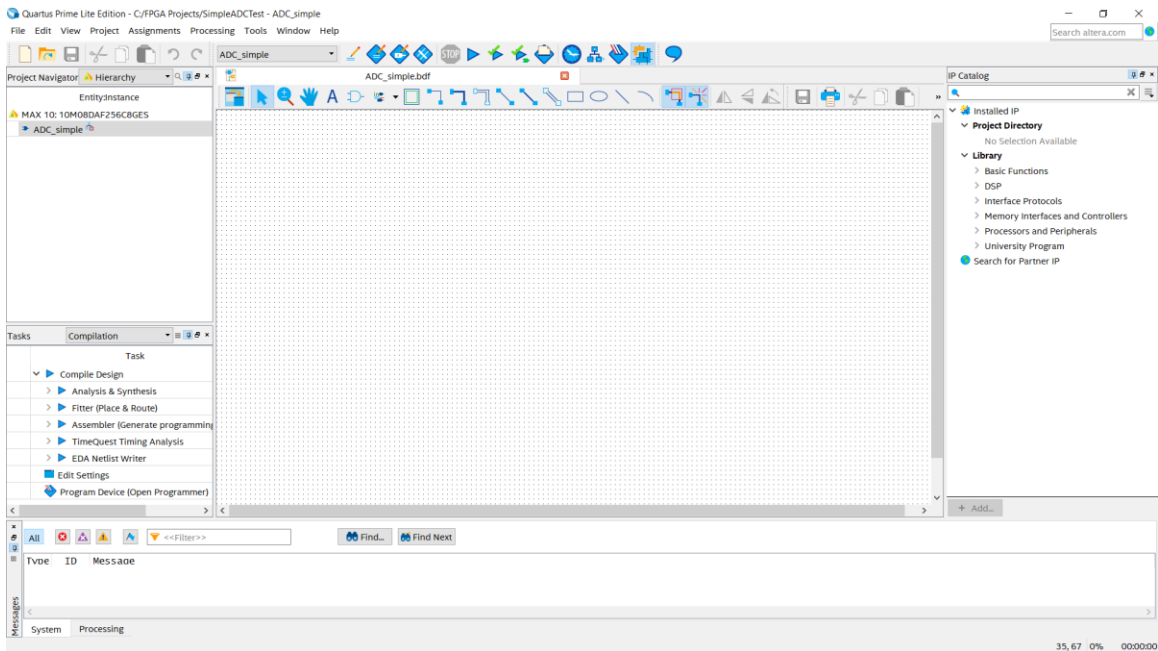
Στο μενού που εμφανίζεται επιλέγουμε **Block Diagram/Schematic File**.



Το αρχείο αυτού του είδους αποτελεί ένα γραφικό περιβάλλον στο οποίο μπορούμε να δημιουργήσουμε εύκολα και γρήγορα τις διάφορες οντότητες που θα υλοποιήσουν την εφαρμογή με σχηματικά διαγράμματα. Έπειτα επιλέγουμε:

File → Save As...

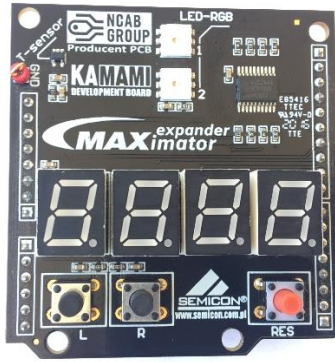
για να αποθηκεύσουμε την δημιουργία της ανώτερης οντότητας του project. Αυτόματα το αναπτυξιακό περιβάλλον προτείνει το όνομα της οντότητας να είναι ίδιο με αυτό που καταχωρίσαμε κατά την δημιουργία του project, δηλαδή το **ADC_simple**. Με τον τρόπο αυτό αποθηκεύεται ως αρχείο, η ανώτερη οντότητα, μέσα στην οποία θα αναπτυχθούν οι επιμέρους οντότητες της εφαρμογής. Το πλήρες όνομα του αρχείου είναι **ADC_simple.bdf**. Θα μπορούσαμε να επιλέξουμε διαφορετικό τύπο αρχείου ως αρχική οντότητα, όπως το αρχείο VHDL, στο οποίο οι επιμέρους οντότητες θα έπρεπε να προστεθούν περιγραφικά με κώδικα VHDL, χωρίς την βοήθεια γραφικού περιβάλλοντος.



Μέσα στην ανώτερη οντότητα (top entity) που δημιουργήθηκε θα αναπτύξουμε όλες τις απαραίτητες επιμέρους οντότητες για την υλοποίηση της εφαρμογής. Σε αυτό το σημείο είναι απαραίτητο να σχεδιασθεί και να αναλυθεί το σύνολο της λειτουργικότητας της εφαρμογής.

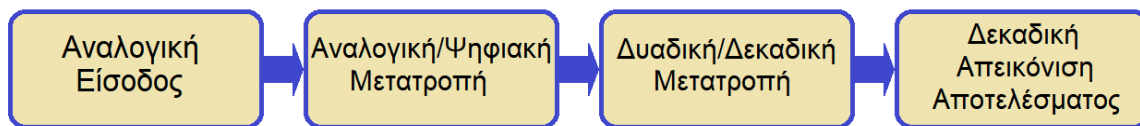
7.3 Σχεδίαση της εφαρμογής

Δεδομένου ότι σκοπός της εφαρμογής είναι να χρησιμοποιηθεί το ADC block του κυκλώματος FPGA, θα είναι απαραίτητη η χρήση του πυρήνα Altera Modular ADC IP core, στην πιο ευέλικτη διαμόρφωση, με χρήση μόνο του μικροπυρήνα ADC control. Μέσα από την παραμετροποίηση του ADC control, θα μπορεί να επιλεγεί το κανάλι προς αναλογική/ψηφιακή μετατροπή. Το αποτέλεσμα αυτής της μετατροπής θα είναι διαθέσιμο στην έξοδο του **response_data[0:11]** διαύλου. Για την απεικόνιση του αποτελέσματος θα χρησιμοποιηθεί το υλικό του αποσπώμενου κυκλώματος διεπαφών MAXimator expander, και συγκεκριμένα η οθόνη LED απεικόνισης δεκαδικών ψηφίων.



Εικόνα 7: Τυπωμένο κύκλωμα διεπαφών MAXiminator expander με την οθόνη LED τεσσάρων δεκαδικών ψηφίων.

Το αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής, πρέπει πρώτα να μετατραπεί από δυαδική σε BCD μορφή, προκειμένου να απεικονισθεί στην οθόνη LED δεκαδικών ψηφίων. Επομένως θα πρέπει να σχεδιασθεί ένα κύκλωμα μετατροπής του δυαδικού αποτελέσματος της αναλογικής/ψηφιακής μετατροπής, σε δεκαδικό αριθμό. Το συνολικό διάγραμμα της κυκλωματικής υλοποίησης της εφαρμογής που μόλις σχεδιάσαμε, απεικονίζεται παρακάτω.



Σχήμα 22. Διαγραμματική απεικόνιση της εφαρμογής αναλογικής/ψηφιακής μετατροπής.

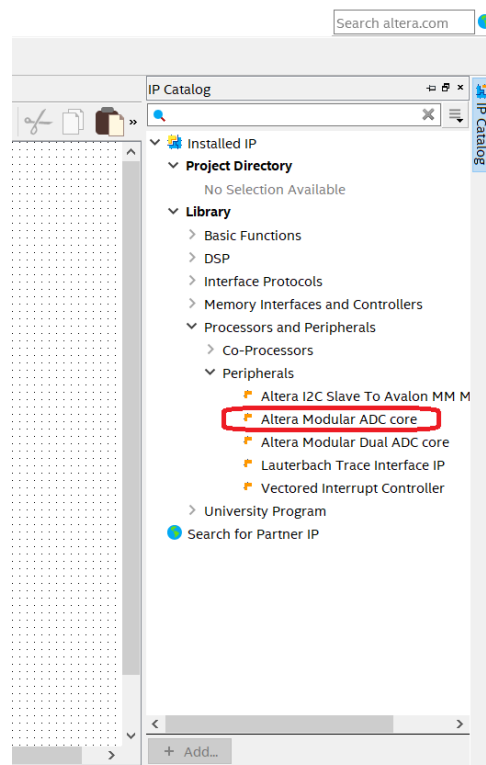
Το πιο σημαντικό στοιχείο του διαγράμματος, είναι προφανώς η αναλογική/ψηφιακή μετατροπή. Η οντότητα αυτή θα χρησιμοποιηθεί μέσα από τον έτοιμο κατάλογο των IPs (Intellectual Properties) που μας παρέχει το περιβάλλον σχεδίασης Quartus. Οι οντότητες που θα χρησιμοποιηθούν για την μετατροπή του δυαδικού αποτελέσματος της αναλογικής/ψηφιακής μετατροπής σε δεκαδικό αριθμό και της δεκαδικής απεικόνισης του αποτελέσματος, θα πρέπει να δημιουργηθούν από την αρχή. Οι δυο αυτές οντότητες θα δημιουργηθούν με την βοήθεια του κώδικα περιγραφής κυκλωμάτων VHDL.

7.4 Δημιουργία και παραμετροποίηση του Altera Modular ADC core

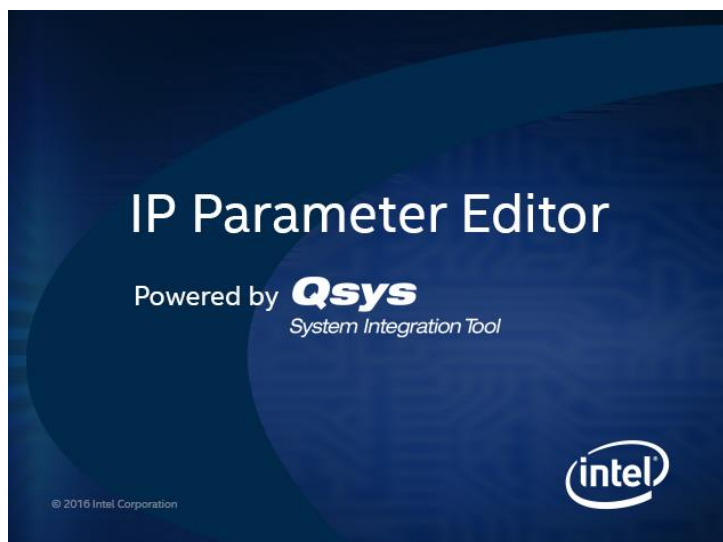
Για την δημιουργία της πιο σημαντικής οντότητας της εφαρμογής, που όπως προαναφέρθηκε είναι η IP (Intellectual Property) της αναλογικής ψηφιακής μετατροπής Altera Modular ADC core, στο περιβάλλον Quartus υπάρχει η ειδική εφαρμογή **IP Parameter Editor** η οποία διαχειρίζεται την δημιουργία και την παραμετροποίηση όλων των προσχεδιασμένων IP που παρέχει η Intel, για τον κάθε τύπο κυκλώματος FPGA που διαθέτει.

Για να εισάγουμε στο project μία από τις διαθέσιμες IP που υπάρχουν για την συσκευή στόχο, δεν έχουμε παρά να περιηγηθούμε στις επιλογές του παραθύρου **IP catalog** που βρίσκεται στην δεξιά πλευρά του περιβάλλοντος εργασίας ή στο μενού **Tools → IP Catalog**. Μέσα στον IP catalog επιλέγουμε:

Installed IP → Library → Processor and Peripherals → Peripherals → Altera Modular ADC core



Αμέσως ανοίγει η εφαρμογή IP Parameter Editor η οποία θα μας βοηθήσει να παραμετροποιήσουμε την IP που επιλέξαμε.



Εικόνα 8: Το λογότυπο έναρξης της εφαρμογής Qsys.

Στην διεπαφή που εμφανίζεται επιλέγουμε το όνομα που θέλουμε να δώσουμε στην νέα οντότητα τύπου Altera Modular ADC core, καθώς επίσης και την τοποθεσία που θα αποθηκευτεί. Επιλέγουμε το όνομα MyADC για την νέα οντότητα και τον κατάλογο FPGA Projects στον οποίο έχουμε αποθηκεύσει όλα τα αρχεία του συγκεκριμένου project. Το αρχείο της οντότητας που θα δημιουργηθεί, έχει το όνομα MyADC.qsys και μπορεί να επαναχρησιμοποιηθεί και σε μελλοντικά projects.

The image shows a dialog box titled "New IP Variation" with a close button (X) in the top right corner. The dialog contains the following fields and sections:

- A message: "Your IP settings will be saved in a .qsys file."
- A section titled "Create IP Variation" with two input fields:
 - "Entity name:" with the text "MyADC" entered.
 - "Save in folder:" with the text "C:\FPGA Projects" and a browse button (...).
- A section titled "Target Device" with two dropdown menus:
 - "Family:" with "MAX 10" selected.
 - "Device:" with "10M08DAF256C8GES" selected.
- An information bar at the bottom with an info icon (i) and the text: "Info: Your IP will be saved in C:\FPGA Projects/MyADC.qsys."
- An "OK" button in the bottom right corner.

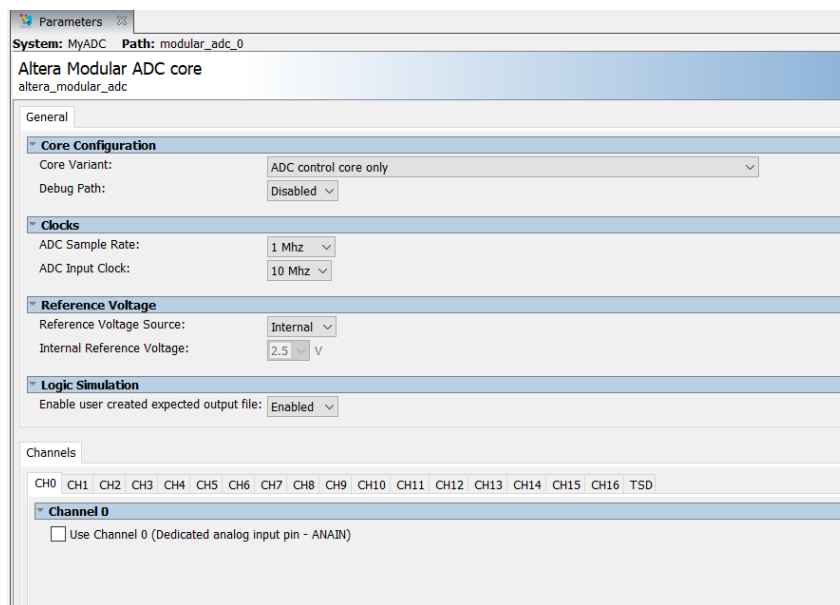
Επιλέγουμε OK και αμέσως εμφανίζεται η διεπαφή παραμετροποίησης της οντότητας που πρόκειται να δημιουργηθεί.

Στο πεδίο **Core Variant** επιλέγουμε **ADC control core only**. Με τον τρόπο αυτό επιλέγουμε την πιο απλή διαμόρφωση που διαθέτει η συγκεκριμένη IP για την λειτουργία του ADC block της συσκευής στόχου.

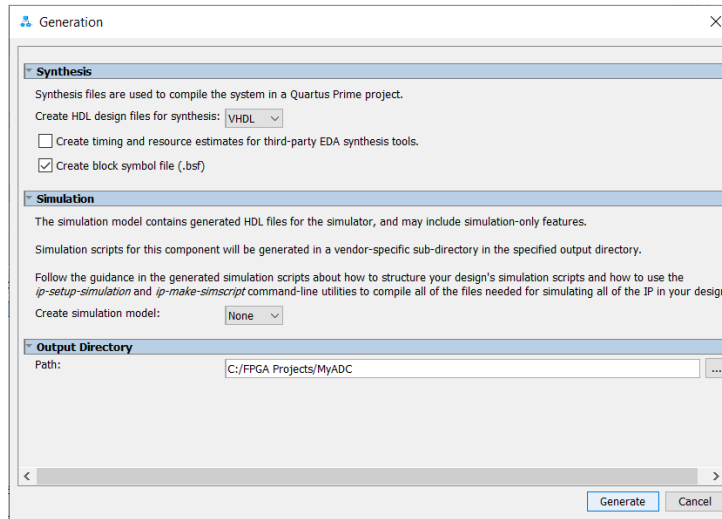
Στο πεδίο **ADC Sample Rate** επιλέγουμε την μέγιστη δειγματοληψία του 1Mhz.

Στο πεδίο **ADC Input Clock** ορίζουμε την συχνότητα λειτουργίας που θα έχει ως είσοδο το ADC block και επιλέγουμε 10Mhz.

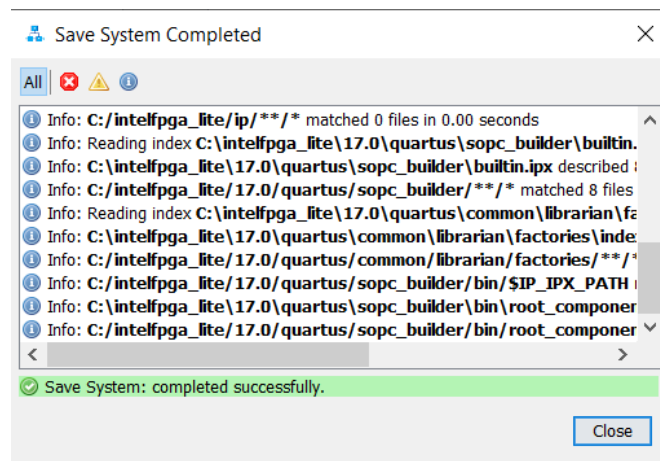
Στο πεδίο **Reference Voltage Source** επιλέγουμε την **Internal** για να ενεργοποιηθεί αυτόματα η εσωτερική τάση αναφοράς των 2.5V για το ADC block.



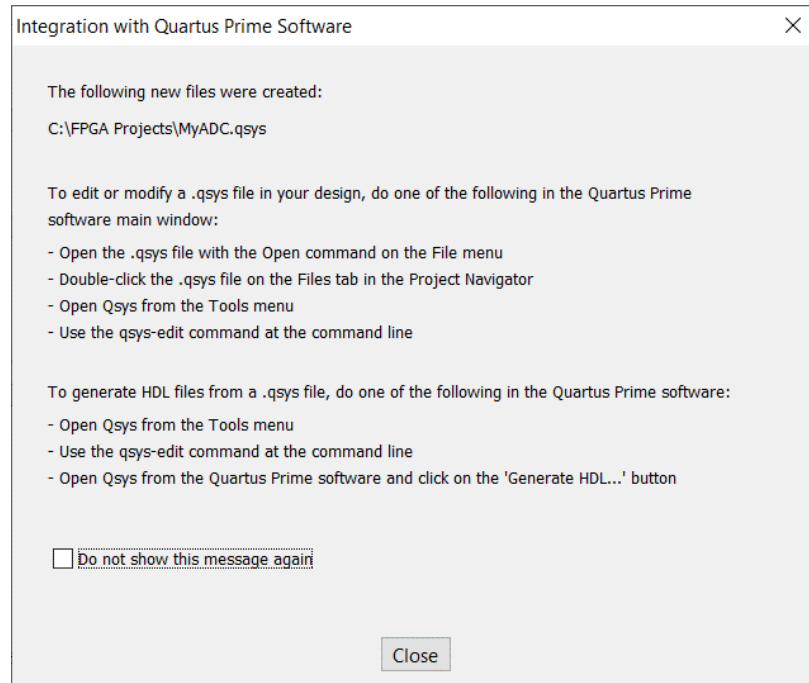
Στο σημείο αυτό έχει ολοκληρωθεί η παραμετροποίηση της οντότητας MyADC που θα ελέγχει το ADC block. Επιλέγουμε Generate HDL για να ξεκινήσει η διαδικασία δημιουργίας του κώδικα περιγραφής κυκλώματος από τον IP Parameter Editor.



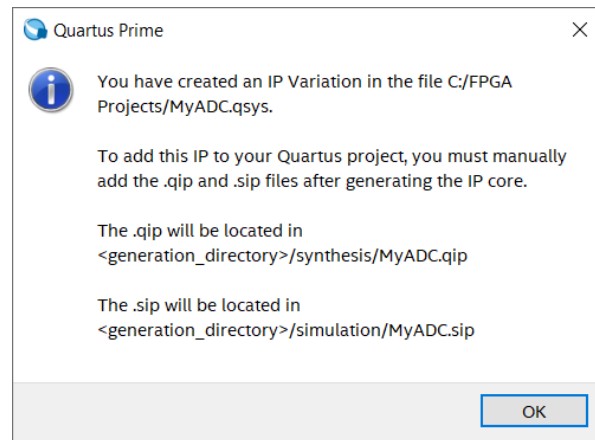
Στο πεδίο **Create HDL files for synthesis** επιλέγουμε **VHDL** ως κώδικα ανάπτυξης και έπειτα **Generate**. Ο κώδικας και τα αρχεία της νέας οντότητας δημιουργούνται και αποθηκεύονται στον φάκελο MyADC ο οποίος δημιουργείται αυτόματα από τον IP Parameter Editor. Εφόσον η διαδικασία δημιουργίας του κώδικα VHDL ήταν επιτυχής, εμφανίζεται η παρακάτω διεπαφή.



Έπειτα η εφαρμογή IP Parameter Editor ενημερώνει για τις ενέργειες που πρέπει να πραγματοποιηθούν σε περίπτωση που οι παράμετροι της οντότητας που δημιουργήσαμε χρειάζεται να τροποποιηθούν. Επίσης περιγράφονται τα βήματα που χρειάζεται να ακολουθηθούν για την δημιουργία HDL αρχείου μέσα από το αρχείο .qsys.

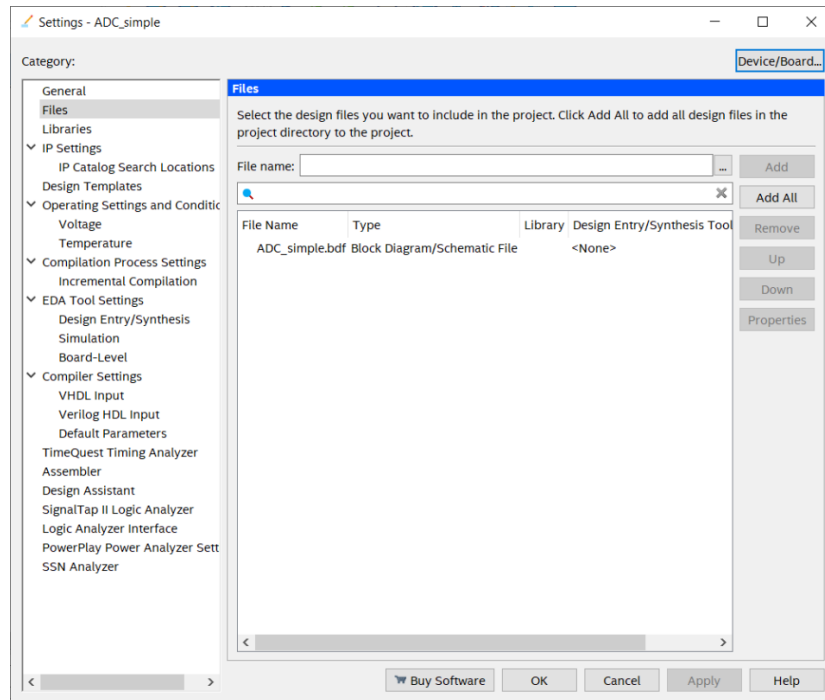


Μετά την επιτυχή ολοκλήρωση της δημιουργίας της οντότητας MyADC, η εφαρμογή παραμετροποίησης IP Parameter Editor κλείνει. Στο σημείο αυτό το Quartus ενημερώνει για την χρήση της νέας οντότητας που μόλις δημιουργήθηκε.

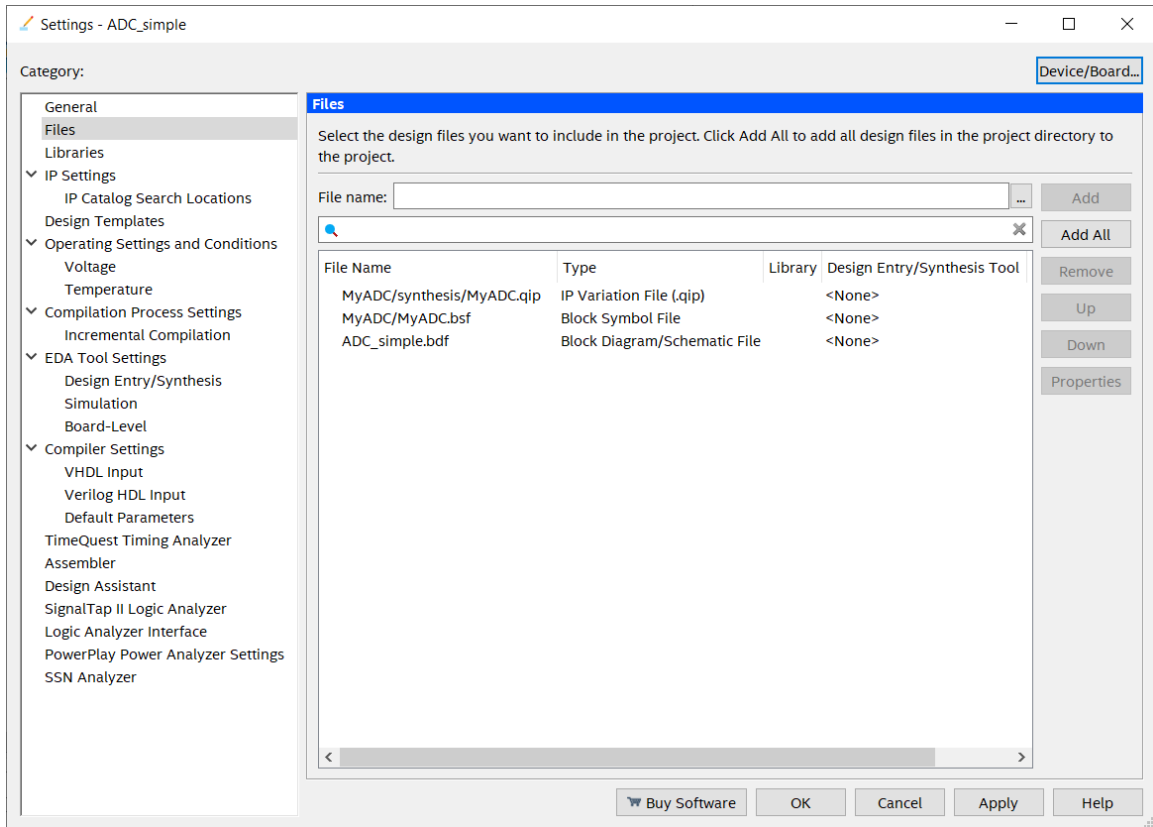


Το παραπάνω μήνυμα αναφέρει την τοποθεσία στην οποία βρίσκονται τα αρχεία που πρέπει να προστεθούν στο project, ώστε να είναι διαθέσιμη η οντότητα MyADC για την εφαρμογή. Για να χρησιμοποιηθεί η νέα οντότητα στην εφαρμογή μας, πρέπει να προσθέσουμε τα αρχεία MyADC.qip και MyADC.bsf. Το αρχείο MyADC.bsf αποτελεί την διαγραμματική απεικόνιση της οντότητας MyADC. Επιλέγουμε:

Project → Add/Remove Files in Project...

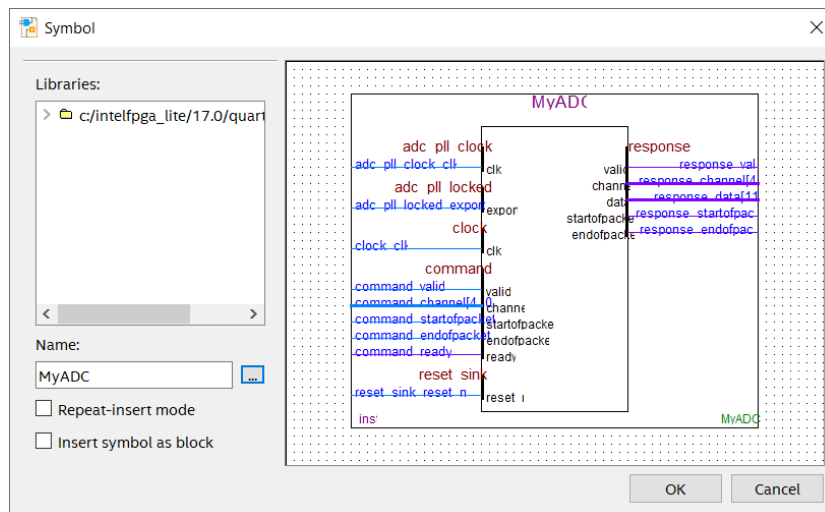


Η διεπαφή που εμφανίζεται παρουσιάζει τα αρχεία τα οποία είναι συνδεδεμένα με το project που δημιουργήσαμε. Προς το παρόν, το μόνο αρχείο που είναι προσαρτημένο στο project είναι το ADC_simple.bdf, το οποίο αποτελεί και την ανώτερη οντότητα του project. Επιλέγουμε και προσθέτουμε τα αρχεία MyADC.qip και MyADC.bsf.



Η οντότητα MyADC που ουσιαστικά αποτελεί τον πυρήνα ελέγχου του ADC block, είναι έτοιμη να χρησιμοποιηθεί στην εφαρμογή μας. Μέσα στον χώρο σχεδίασης της ανώτερης οντότητας πατάμε δεξί κλικ και επιλέγουμε:

Insert → Symbol...

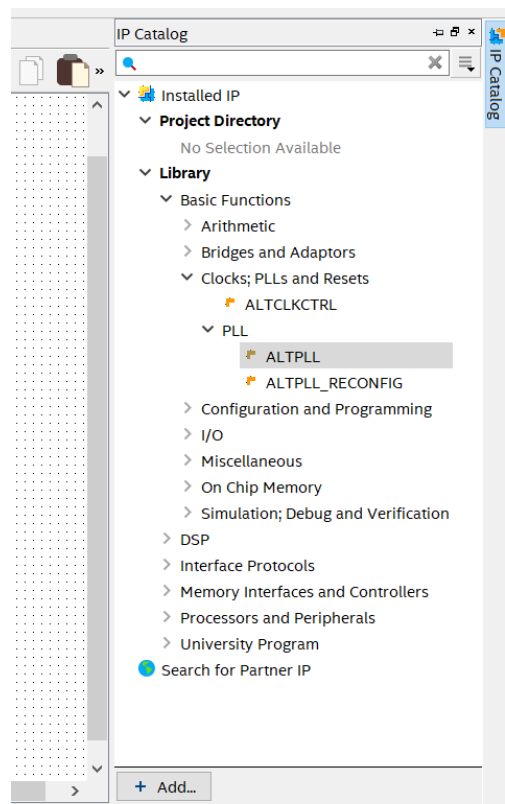


Στο πεδίο **Name** επιλέγουμε, έπειτα από αναζήτηση στην τοποθεσία του φακέλου MyADC, το αρχείο MyADC.bsf και εμφανίζεται το σχηματικό διάγραμμα του αναλογικού/ψηφιακού μετατροπέα, με όλες τις εισόδους και εξόδους της διαμόρφωσης ADC control core. Επιλέγουμε **OK** και η οντότητα MyADC έχει εισαχθεί ως σύμβολο στην ανώτερη οντότητα.

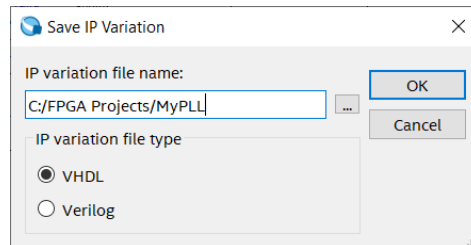
7.5 Δημιουργία και παραμετροποίηση του κυκλώματος χρονισμού ALTPLL

Για να καταστεί λειτουργική η οντότητα MyADC πρέπει πρωτίστως να συνδεθεί με το ρολόι που θα οδηγεί τα εσωτερικά κυκλώματα χρονισμού του ADC block. Για αυτό τον λόγο είναι απαραίτητο να εισαχθεί στην εφαρμογή η οντότητα χρονισμού PLL (Phase Locked Loop). Κατά την παραμετροποίηση της οντότητας ADC, επιλέχθηκε η συχνότητα 10Mhz ως συχνότητα χρονισμού. Επομένως πρέπει να δημιουργήσουμε μια νέα οντότητα PLL η οποία θα χρονίζει την οντότητα MyADC στη συχνότητα των 10Mhz. Επιλέγουμε:

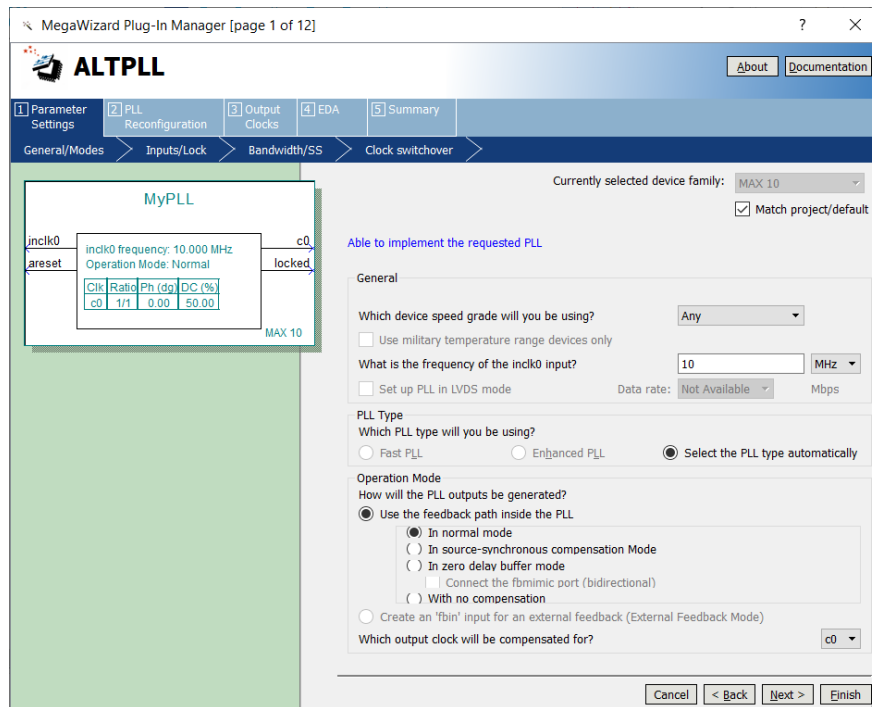
Tools → **IP Catalog** → **Installed IP** → **Library** → **Basic Functions** → **Clocks; PLLs and Resets** → **PLL** → **ALTPLL**



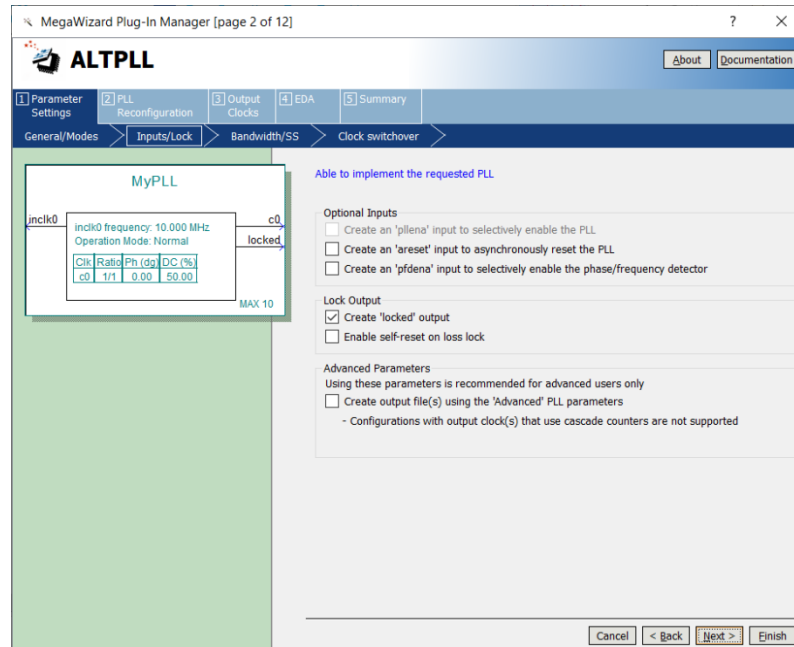
Έπειτα από την επιλογή του ALTPLL ξεκινά η δημιουργία νέας οντότητας PLL με την εμφάνιση της διεπαφής εισαγωγής των στοιχείων αποθήκευσης. Επιλέγουμε το όνομα MyPLL και αποθήκευση στον ίδιο φάκελο που βρίσκεται το project. Ως κώδικα περιγραφής υλικού επιλέγουμε VHDL.



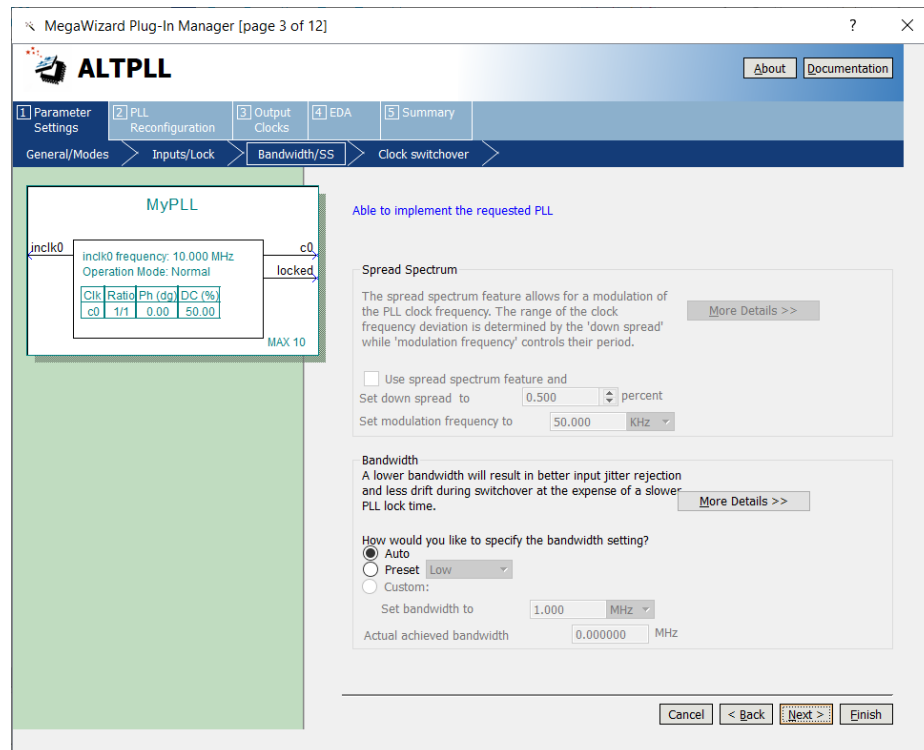
Η δημιουργία και η παραμετροποίηση του PLL πραγματοποιείται μέσα από μια εφαρμογή που ονομάζεται Mega Wizard και είναι προσαρτημένη (plug in) στο περιβάλλον Quatus. Στην πρώτη διεπαφή που εμφανίζει η Mega Wizard δηλώνεται η συχνότητα εισόδου του PLL βάσει της οποίας θα σχηματισθεί η συχνότητα εξόδου. Στα έγγραφα (datasheet) που αφορούν τη χρήση και την περιγραφή του υλικού (MAXimator) που χρησιμοποιούμε για την συγκεκριμένη εφαρμογή, δηλώνεται ως συχνότητα συστήματος η συχνότητα των 10Mhz. Η συχνότητα αυτή θα αποτελεί την βάση της οντότητας PLL της εφαρμογής.



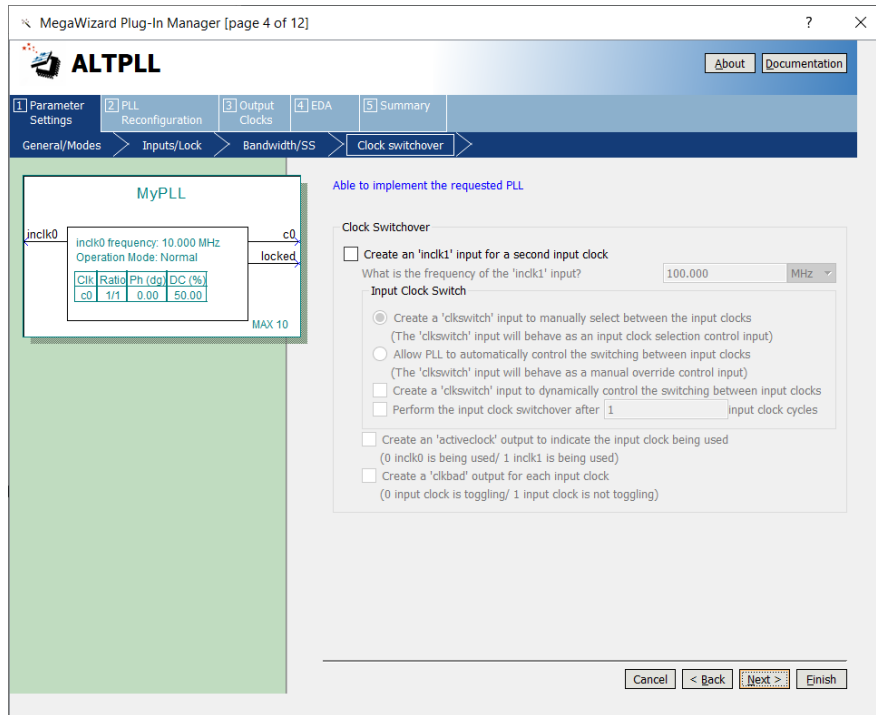
Επιλέγουμε **Next**. Στην επόμενη διεπαφή απενεργοποιούμε την επιλογή **Create an 'areset' input to asynchronously reset the PLL** και επιλέγουμε **Create 'locked' output**.



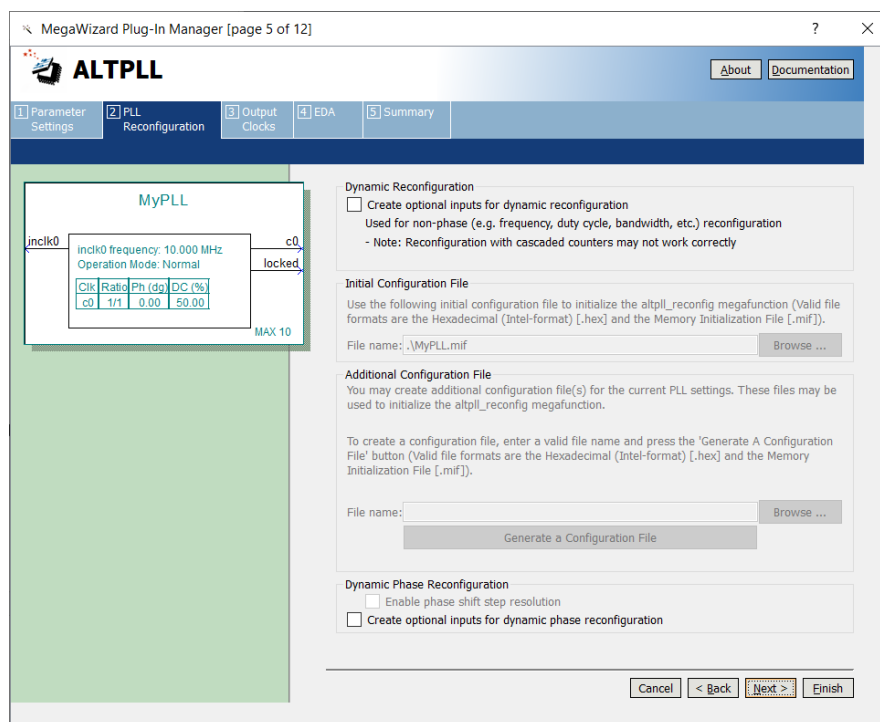
Επιλέγουμε **Next**. Στην επόμενη διεπαφή δεν πραγματοποιείται καμία τροποποίηση.



Επιλέγουμε **Next**. Στην επόμενη διεπαφή δεν πραγματοποιείται καμία τροποποίηση.

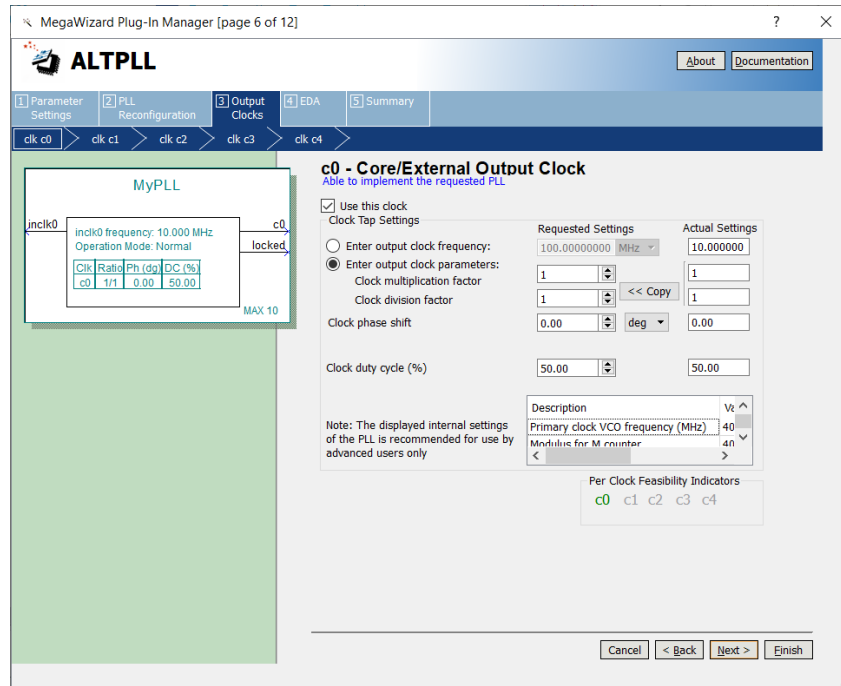


Επιλέγουμε **Next**. Στην επόμενη διεπαφή δεν πραγματοποιείται καμία τροποποίηση.

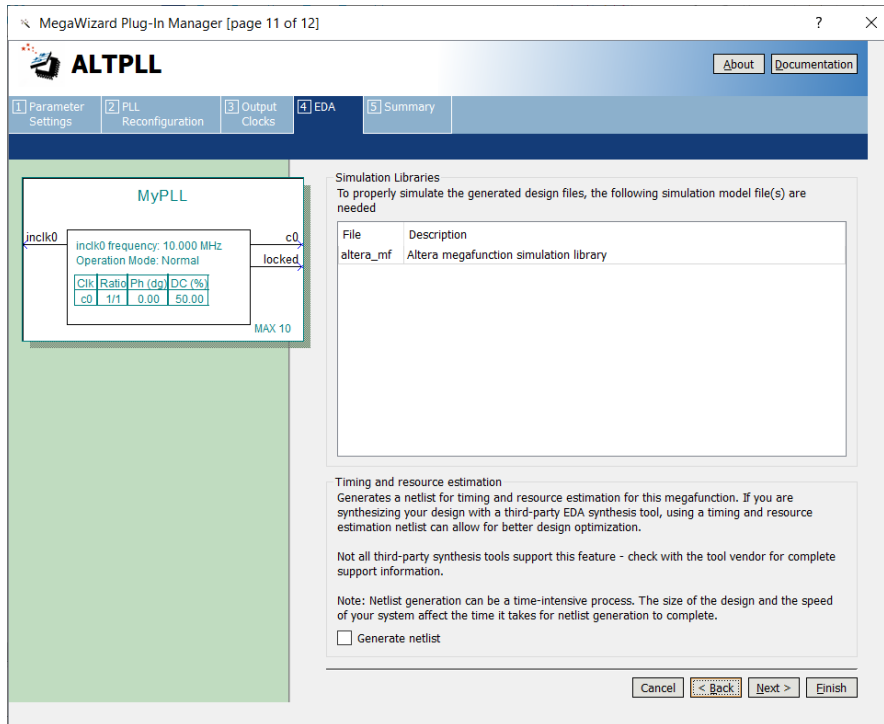


Επιλέγουμε **Next**. Στην επόμενη διεπαφή, πραγματοποιούνται οι ρυθμίσεις για το clock c0 το οποίο έχουμε επιλέξει να λειτουργεί ως μετατροπέας της συχνότητας εισόδου, για τη συγκεκριμένη εφαρμογή. Ο παράγοντας πολλαπλασιασμού της συχνότητας εισόδου

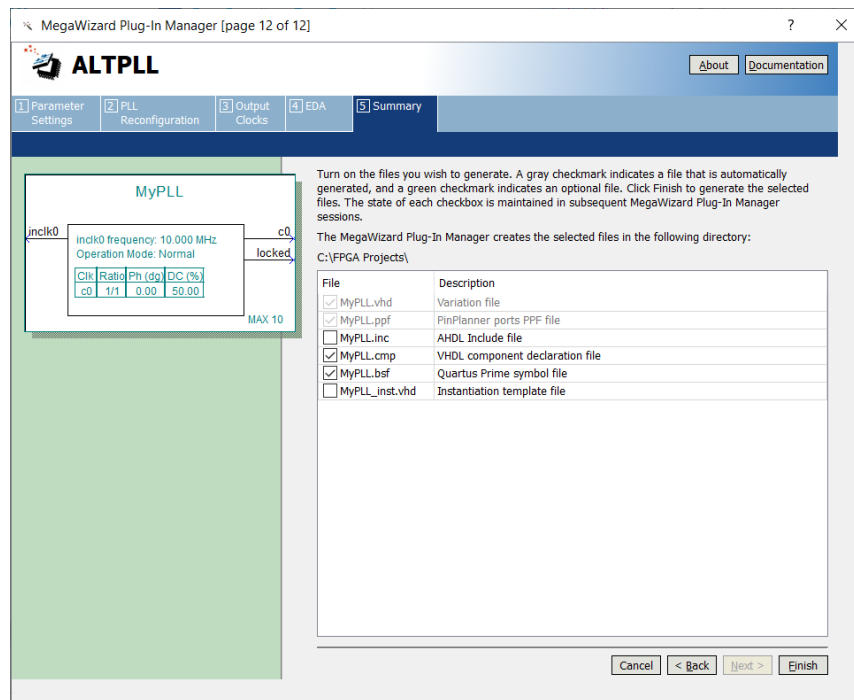
είναι 1, όπως επίσης και ο παράγοντας διαίρεσης της συχνότητας εισόδου, οπότε στην έξοδο του PLL θα εμφανίζεται η ίδια συχνότητα με αυτή της εισόδου, που έχει επιλεγεί στα 10Mhz.



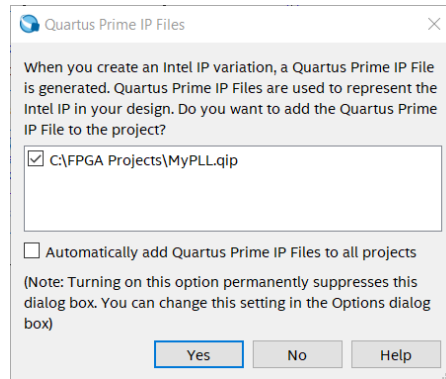
Επιλέγουμε **Next**. Ακολουθούν τέσσερις διαδοχικές διεπαφές που αφορούν τις ρυθμίσεις των clock c1,c2,c3,c4 τα οποία δε χρησιμοποιούνται στην συγκεκριμένη εφαρμογή. Επιλέγουμε απλά **Next** σε κάθε περίπτωση χωρίς να πραγματοποιήσουμε καμία αλλαγή. Στην διεπαφή που ακολουθεί επιλέγουμε επίσης **Next** χωρίς καμία αλλαγή.



Στην επόμενη και τελευταία διεπαφή επιλέγουμε να εξαχθεί και το αρχείο συμβόλου MyPLL.bsf για την οντότητα MyPLL που μόλις δημιουργήθηκε και επιλέγουμε **Finish**.



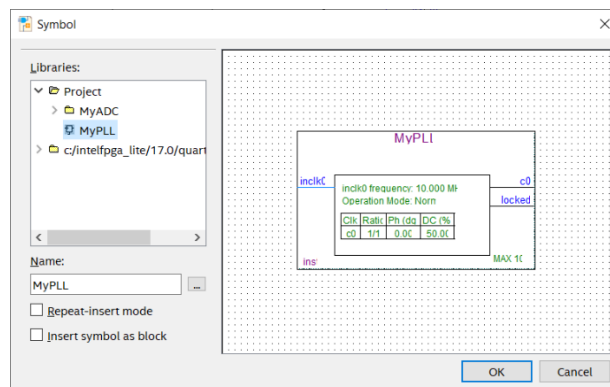
Με το κλείσιμο του Mega Wizard, το περιβάλλον Quartus ρωτά για την εισαγωγή της οντότητας MyPLL στο project. Επιλέγουμε **Yes** και η οντότητα έχει προστεθεί στο project και είναι διαθέσιμη για να χρησιμοποιηθεί στην εφαρμογή.

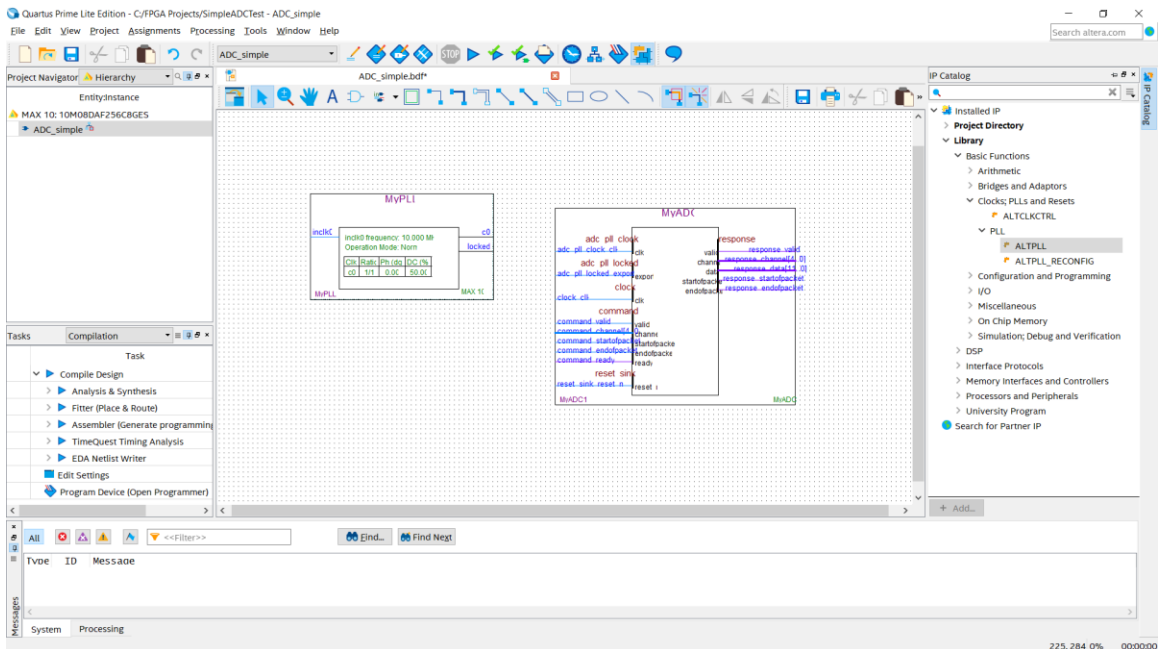


Για να χρησιμοποιηθεί η οντότητα MyPLL στην εφαρμογή πατάμε δεξί κλικ στο χώρο σχεδίασης της ανώτερης οντότητας και στο μενού που εμφανίζεται επιλέγουμε:

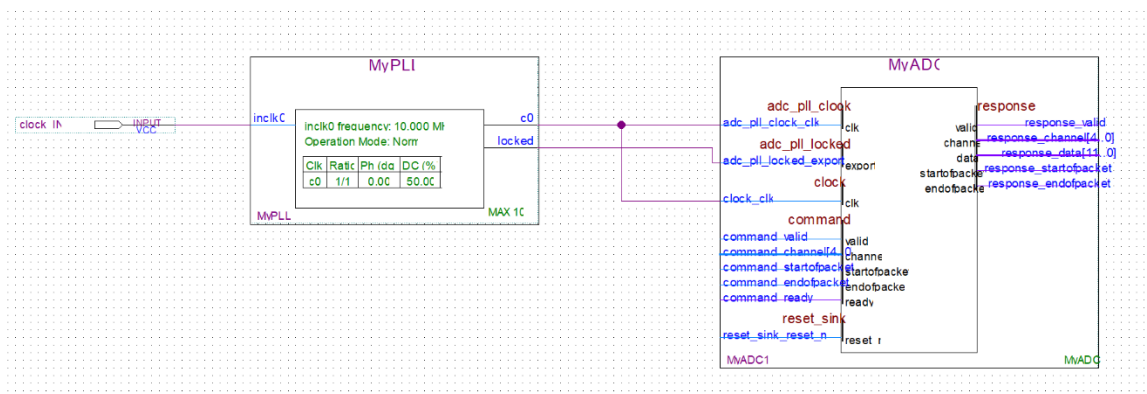
Insert → Symbol...

Στην διεπαφή που εμφανίζεται, επιλέγουμε το σύμβολο της οντότητας MyPLL και αμέσως αυτή προστίθεται στο χώρο σχεδίασης της ανώτερης οντότητας ADC_simple.





Στο σημείο αυτό υπάρχουν στον χώρο σχεδίασης το σύμβολο MyPLL που αναπαριστά τη συσκευή PLL η οποία θα χρησιμοποιηθεί για τον χρονισμό της συσκευής ADC block και το σύμβολο MyADC που αποτελεί τον ουσιαστικό έλεγχο του ADC block. Χρησιμοποιώντας το **pin tool** από την μπάρα εργαλείων, εισάγουμε ένα pin εισόδου στον σχεδιασμό το οποίο θα χρησιμοποιηθεί ως είσοδος για την συχνότητα εισόδου του MyPLL. Το pin εισόδου το ονομάζουμε **clock_IN**, πατώντας δεξί κλικ και επιλέγοντας **properties**. Επίσης με το εργαλείο **orthogonal node tool** δημιουργούμε τις απαραίτητες συνδέσεις μεταξύ του συμβόλου MyPLL του pin clock_IN και του συμβόλου MyADC (Σχήμα 23).



Σχήμα 23. Η διασύνδεση του κυκλώματος χρονισμού (MyPLL) με τον αναλογικό/ψηφιακό μετατροπέα MyADC.

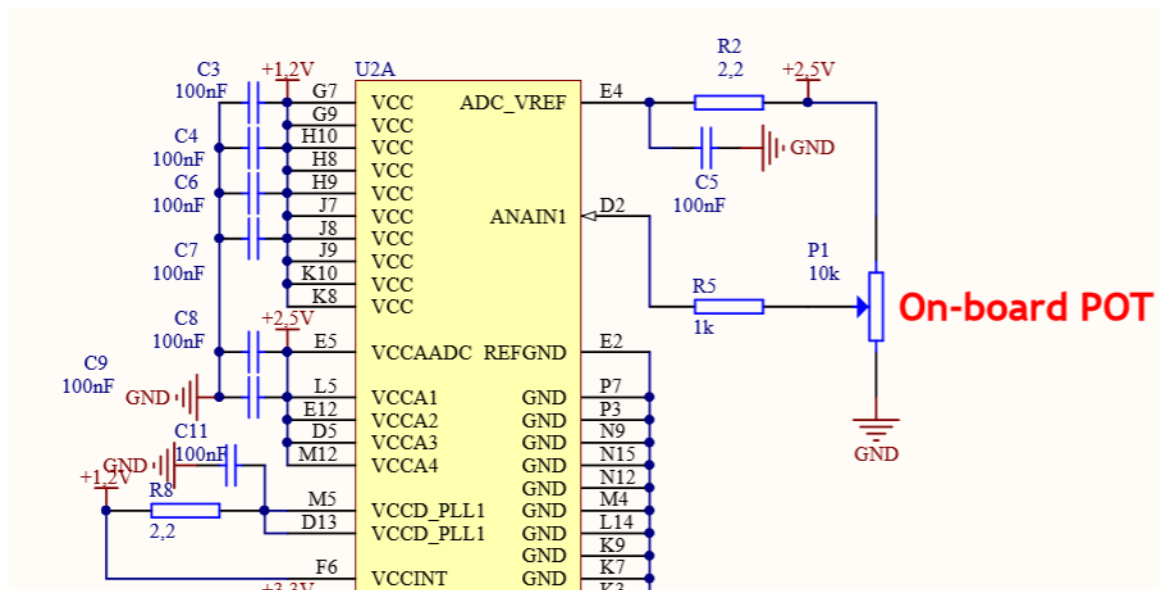
7.6 Δημιουργία σημάτων παραμετροποίησης της συσκευής MyADC.

Στη συνέχεια της παραμετροποίησης της συσκευής MyADC, θα πρέπει να καθορίσουμε την αναλογική είσοδο που θα χρησιμοποιεί η οντότητα MyADC για να πραγματοποιεί την ψηφιακή μετατροπή. Αυτό επιτυγχάνεται με την ανάθεση της κατάλληλης δυαδικής τιμής στον δίαυλο **command channel[4..0]**. Η τιμή αυτή αντιστοιχεί στο αναλογικό κανάλι εισόδου σύμφωνα με τον παρακάτω πίνακα (Πίνακας 2).

Πίνακας 2. Η δυαδική κωδικοποίηση των αναλογικών καναλιών.

Αναλογικό κανάλι	Δυαδική Κωδικοποίηση	Αναλογικό κανάλι	Δυαδική Κωδικοποίηση
CH0	00000	CH9	01001
CH1	00001	CH10	01010
CH2	00010	CH11	01011
CH3	00011	CH12	01100
CH4	00100	CH13	01101
CH5	00101	CH14	01110
CH6	00110	CH15	01111
CH7	00111	CH16	10000
CH8	01000	TSD	10001

Στην συγκεκριμένη εφαρμογή θα χρησιμοποιηθεί το αναλογικό σήμα που προκύπτει από το ποτενσιόμετρο που διαθέτει το κύκλωμα MAXimator. Σύμφωνα με το σχηματικό διάγραμμα του κυκλώματος, το ποτενσιόμετρο P1 (10k) είναι συνδεδεμένο με τον ακροδέκτη ANAIN1 (D2) (Σχήμα 24).

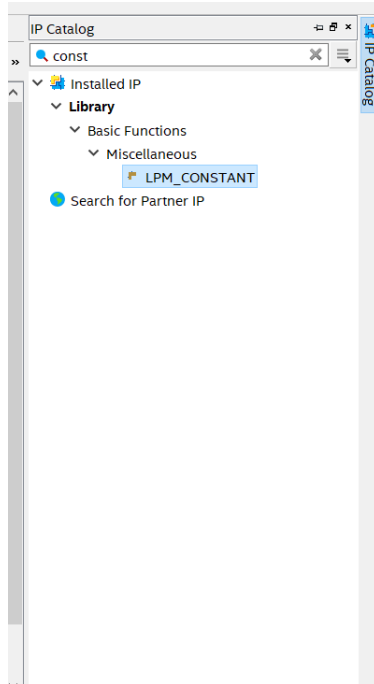


Σχήμα 24. Σχηματικό διάγραμμα του κυκλώματος του ποτενσιομέτρου P1 (10kΩ) (MAXimator, Schematics).

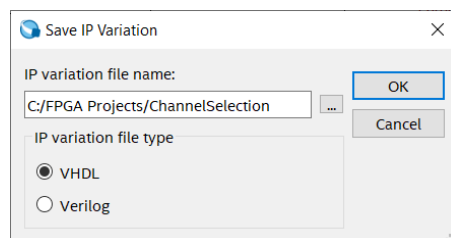
Στο έγγραφο τεχνικής περιγραφής (datasheet) του ADC block, αναφέρεται ότι η είσοδος ANAIN1 αντιστοιχεί στο αναλογικό κανάλι 0 (CH0). Η δυαδική κωδικοποίηση του συγκεκριμένου καναλιού είναι η 00000 όπως περιγράφεται στον παραπάνω πίνακα. Η τιμή που πρέπει επομένως να ανατεθεί στον διάλο **command channel[4..0]** είναι η τιμή 00000. Με την συγκεκριμένη ανάθεση τιμής, το ADC block θα μετατρέπει συνεχώς την αναλογική είσοδο CH0 σε ψηφιακό δυαδικό αριθμό. Το δυαδικό αποτέλεσμα αυτής της μετατροπής έχει εύρος 12 bits και εμφανίζεται στον διάλο **response data[11..0]**.

Η ανάθεση ορισμένης δυαδικής τιμής σε κάποιο διάλο εισόδου, πραγματοποιείται με την χρήση της IP **LPM_CONSTANT** που βρίσκεται στο μενού:

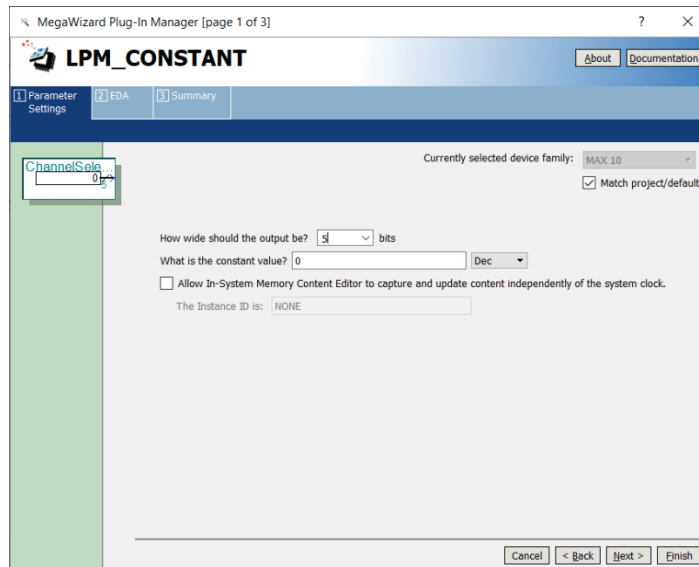
Tools → **IP Catalog** → **Installed IP** → **Library** → **Basic Functions** → **Miscellaneous** → **LPM_CONSTANT**



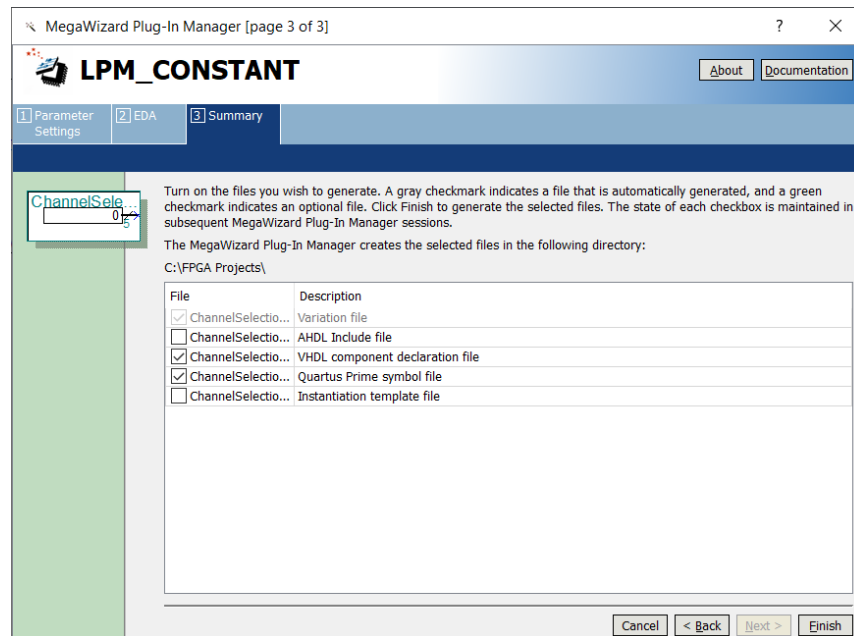
Με την επιλογή της IP LPM_CONSTANT, εμφανίζεται η διεπαφή της επιλογής ονόματος και επιλογής κώδικα περιγραφής κυκλωμάτων.



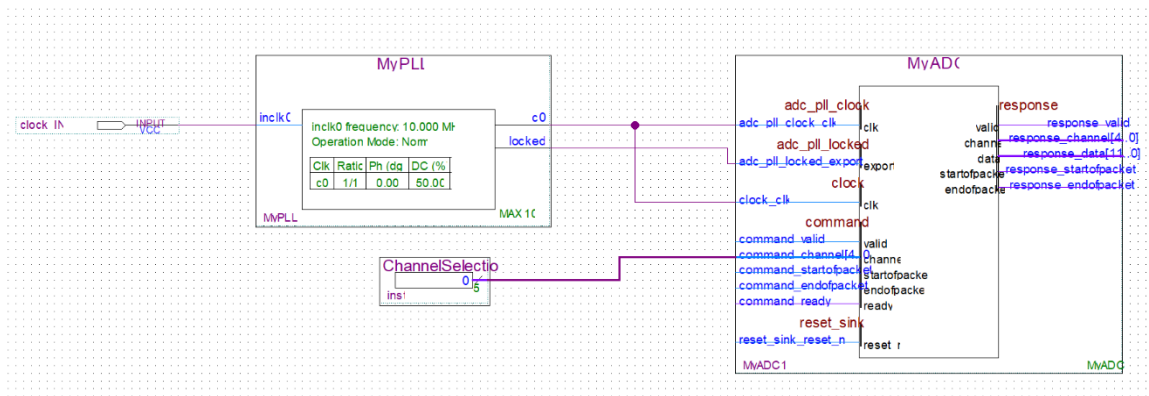
Δίνουμε το όνομα **ChannelSelection** και επιλέγουμε τον κώδικα περιγραφής κυκλωμάτων VHDL. Έπειτα εμφανίζεται η εφαρμογή Mega Wizard για την παραμετροποίηση της IP LPM_CONSTANT. Στην πρώτη διεπαφή επιλέγουμε το εύρος του δυαδικού αριθμού που πρόκειται να παρέχει η LPM_CONSTANT και στην προκειμένη περίπτωση είναι 5 bits. Επίσης επιλέγουμε την τιμή του σταθερού αριθμού που αντιστοιχεί στο αναλογικό κανάλι CH0 (0) και το αριθμητικό σύστημα αναφοράς (δεκαδικό Dec).



Επιλέγουμε δυο φορές **Next** και στην τελευταία διεπαφή ενεργοποιούμε στην εξαγωγή αρχείων και την επιλογή της δημιουργίας αρχείου συμβόλου.

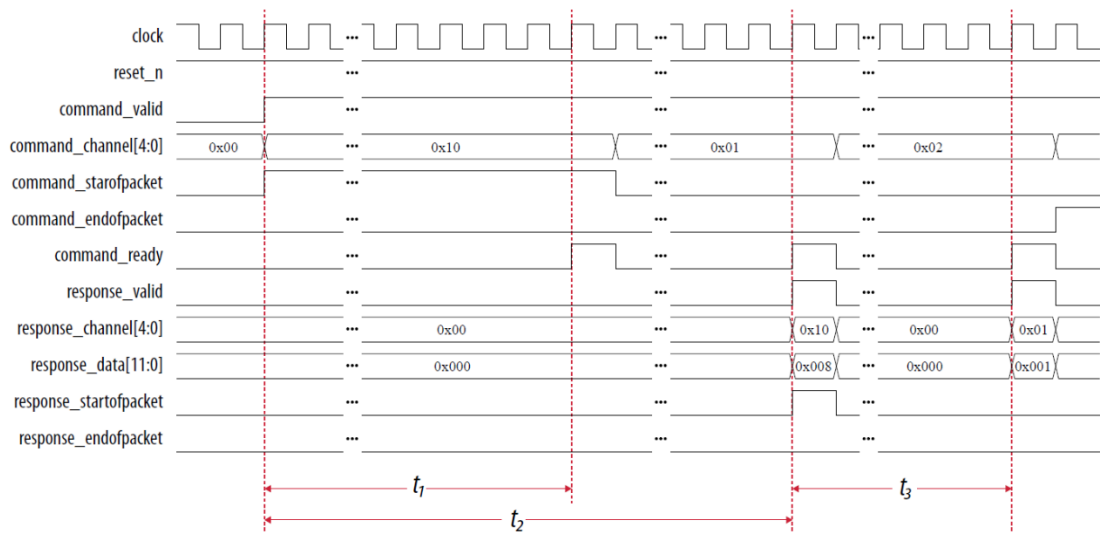


Επιλέγουμε **Finish** και εισάγουμε το σύμβολο ChannelSelection που δημιουργείται, στην ανώτερη οντότητα της εφαρμογής. Συνδέουμε έπειτα την τιμή που εξάγει η ChannelSelection, με τον διάυλο **command channel[4..0]** (Σχήμα 25).



Σχήμα 25. Η σύνδεση της επιλογής αναλογικού καναλιού *ChannelSelection* με τον διάυλο *command channel[4..0]*.

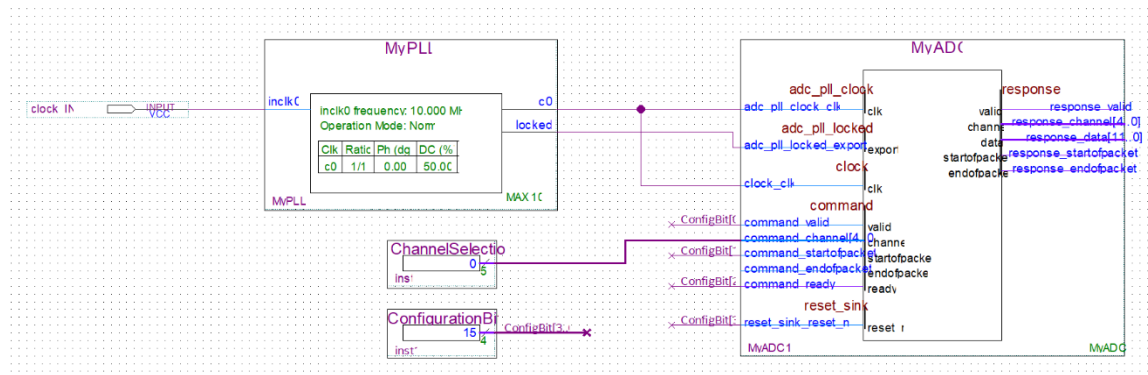
Για την συνεχή και αδιάληπτη αναλογική/ψηφιακή μετατροπή του αναλογικού καναλιού CH0, πρέπει να ενεργοποιηθούν και τα σήματα εισόδου **command_valid**, **command_startofpacket**, **command_ready** και **reset_sink_reset_n**. Σύμφωνα με το παρακάτω διάγραμμα χρονισμού (Σχήμα 24) του ADC block τα σήματα αυτά πρέπει να είναι σε κατάσταση **High** για να λειτουργεί η συνεχής αναλογική/ψηφιακή μετατροπή του αναλογικού καναλιού που έχει επιλεγεί (free running mode).



Σχήμα 26. Χρονικό διάγραμμα αναλογικής/ψηφιακής μετατροπής.

Για την ενεργοποίηση των παραπάνω σημάτων δημιουργούμε ακόμη μία IP LPM_CONSTANT με όνομα **ConfigurationBits** με εύρος 4-bits και δυαδική τιμή 1111

(15 Decimal). Θα μπορούσαμε για τον ίδιο σκοπό να διευρύνουμε την **ChannelSelection** σε 9-bits αλλά δε θα ήταν εύκολος ο διαχωρισμός της λειτουργίας εκάστου bit. Έτσι το design αποκτά την μορφή που απεικονίζεται στο Σχήμα 27.



Σχήμα 27. Η σύνδεση των ConfigurationBits στα αντίστοιχα σήματα του MyADC.

Μέχρι στιγμής, στο σχεδιασμό της εφαρμογής, έχουν παραμετροποιηθεί, ενσωματωθεί και συνδεθεί οι δύο βασικές IP, το ALTPLL και Altera Modular ADC core. Επίσης όλες οι απαραίτητες εισοδοί για την λειτουργία της συνεχούς αναλογικής/ψηφιακής μετατροπής του ADC block, έχουν παραμετροποιηθεί κατάλληλα. Το επόμενο βήμα είναι η διαχείριση της εξόδου του ADC block, που είναι το δυαδικό (12-bits) αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής.

7.7 Ο μετατροπέας δυαδικού αριθμού σε κωδικοποίηση BCD.

Στον δυαδικό κόσμο των ψηφιακών κυκλωμάτων, οι δεκαδικοί αριθμοί δεν έχουν καμία υπόσταση. Έτσι ακόμη και η απεικόνιση των δεκαδικών αριθμών θα πρέπει να πραγματοποιηθεί με την βοήθεια του δυαδικού συστήματος. Το αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής, είναι ένας δυαδικός αριθμός που έχει εύρος 12-bits. Η τελική απεικόνιση θα πραγματοποιηθεί μέσω της οθόνης των τεσσάρων δεκαδικών ψηφίων LED. Το κάθε ψηφίο του δεκαδικού αριθμού που θα προκύψει από την δυαδική/δεκαδική μετατροπή, θα πρέπει να κωδικοποιηθεί με την μορφή δυαδικού αριθμού για να αναπαρασταθεί στην οθόνη των δεκαδικών ψηφίων LED. Για τον σκοπό αυτό υπάρχει η κωδικοποίηση BCD (Binary Coded Decimal). Με την κωδικοποίηση BCD, κάθε δεκαδικό ψηφίο αναπαρίσταται με έναν μοναδικό συνδυασμό δυαδικών ψηφίων όπως φαίνεται στον παρακάτω πίνακα (Πίνακας 3).

Πίνακας 3.Κωδικοποίηση BCD.

Δεκαδικό Ψηφίο	Δυαδική Κωδικοποίηση (BCD)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Όπως έχει προαναφερθεί, το αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής του ADC block, έχει ακρίβεια 12-bits. Κατά συνέπεια, ο μέγιστος δεκαδικός αριθμός που μπορεί να προκύψει από μια αναλογική/ψηφιακή μετατροπή, είναι το 4095 ($2^{12}-1$), το οποίο αποτελείται από 4 ψηφία του δεκαδικού συστήματος. Επομένως το ψηφιακό κύκλωμα που θα πραγματοποιεί την μετατροπή του δυαδικού αποτελέσματος της αναλογικής/ψηφιακής μετατροπής, θα πρέπει να έχει ως είσοδο τα 12 bits του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής και να εξάγει συνολικά 4 δεκαδικά ψηφία με κωδικοποίηση BCD. Ένα τέτοιο κύκλωμα χαρακτηρίζεται ως Binary to BCD converter. Μια δημοφιλής υλοποίηση της μετατροπής αυτής, είναι η «ολίσθηση και πρόσθεση με 3» (shifting and add 3 or double dabble) (Wikipedia, Double dabble algorithm). Κατά την υλοποίηση αυτή, κατασκευάζεται πίνακας ο οποίος περιέχει τόσες στήλες όσα και τα δυαδικά ψηφία που πρόκειται να αναπαραστήσουν τον δεκαδικό αριθμό με κωδικοποίηση BCD. Οι στήλες διατάσσονται από αριστερά προς τα δεξιά με τις στήλες των μεγαλύτερων δεκαδικών ψηφίων να προηγούνται (χιλιάδες, εκατοντάδες, δεκάδες, μονάδες). Στην συνέχεια των στηλών αναπαράστασης των δεκαδικών ψηφίων με μορφή

BCD, αναπτύσσονται τα ψηφία του δυαδικού αριθμού που πρόκειται να μετατραπεί σε μορφή BCD. Οι στήλες διατάσσονται από αριστερά προς τα δεξιά με τις στήλες των σημαντικότερων δυαδικών ψηφίων (MSB) να προηγούνται. Σε κάθε βήμα της διαδικασίας μετατροπής, πραγματοποιείται ολίσθηση των ψηφίων του δυαδικού αριθμού αριστερά κατά μία θέση, έτσι ώστε να συμπληρώνεται κάθε φορά μία επιπλέον θέση στις στήλες των δυαδικών ψηφίων BCD. Έπειτα από κάθε ολίσθηση μίας θέσης προς τα αριστερά, ελέγχονται όλες οι τετράδες των δυαδικών ψηφίων BCD εάν είναι μεγαλύτερες του 4. Σε όποια τετράδα BCD η συνθήκη αυτή είναι αληθείς, προστίθεται ο αριθμός 3. Ακολούθως πραγματοποιείται το επόμενο βήμα ολίσθησης κατά μία θέση προς τα αριστερά και ο έλεγχος των τετράδων των δυαδικών ψηφίων BCD με το 4. Η επανάληψη αυτή πραγματοποιείται μέχρι να ολισθήσουν όλα τα ψηφία του δυαδικού αριθμού που πρόκειται να μετατραπεί σε BCD.

Στο παρακάτω παράδειγμα (Πίνακας 4), πραγματοποιείται η μετατροπή του δυαδικού αριθμού 1101 1111 0000 (12-bits), σε δεκαδικό (3568) με κωδικοποίηση BCD (0011 0101 0110 100).

Πίνακας 4. Παράδειγμα μετατροπής του δυαδικού αριθμού 12-bits 1101111000 σε δεκαδικό 3586.

BCD 4-digits												Binary 12-bits								STEP							
1000'				100'				10'				1'				1	1	0	1		1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	2	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	3		
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	4		
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	5			
0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	6				
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	1	0	0	0	0	7				
0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	1	0	0	0	0	8					
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	9					
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	10						
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	11							
0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	12								


```
library ieee;
USE ieee.std_logic_1164.all;

USE ieee.std_logic_unsigned.all;

ENTITY BinarytoBCD IS
PORT (clk_1 :in std_logic;
      valid_data:in std_logic;
      ds: in std_logic_vector (15 downto 0);
      qs: OUT std_logic_vector (15 downto 0));
END BinarytoBCD;
```

ARCHITECTURE behaviour OF BinarytoBCD IS

BEGIN

PROCESS (clk_1)

variable i : integer:=0;

variable bcd : std_logic_vector(15 downto 0) ;

variable bint : std_logic_vector(15 downto 0) ;

begin

if (clk_1'event and clk_1 = '1' and valid_data = '1') then

 bcd:="0000000000000000";

 bint:=ds;

 for i in 0 to 15 loop -- repeating 16 times.

 bcd(15 downto 1) := bcd(14 downto 0); --shifting the bits.

 bcd(0) := bint(15);

 bint(15 downto 1) := bint(14 downto 0);

 bint(0) :='0';

 if(i < 15 and bcd(3 downto 0) > "0100") then

 bcd(3 downto 0) := bcd(3 downto 0) + "0011";

 end if;

 if(i < 15 and bcd(7 downto 4) > "0100") then

 bcd(7 downto 4) := bcd(7 downto 4) + "0011";

 end if;

 if(i < 15 and bcd(11 downto 8) > "0100") then

 bcd(11 downto 8) := bcd(11 downto 8) + "0011";

 end if;

 if(i < 15 and bcd(15 downto 12) > "0100") then

 bcd(15 downto 12) := bcd(15 downto 12) + "0011";

 end if;

 end loop;

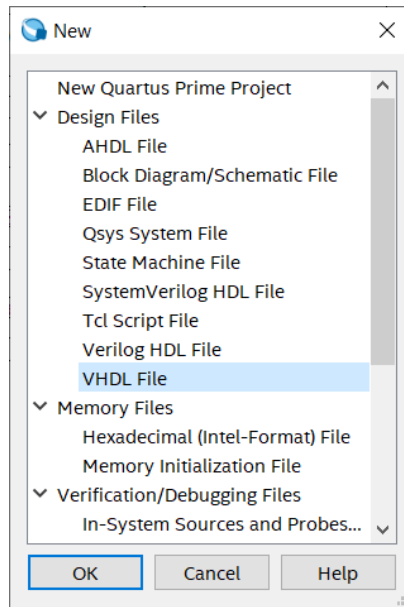
qs<=bcd;


```
        end if;  
END process;  
END behaviour;
```

Ο παραπάνω κώδικας εκτελείται σε μόλις έναν κύκλο ρολογιού, κάθε φορά που η είσοδος **clk_1** βρίσκεται στο μέτωπο ανόδου (rising edge) του παλμού που δέχεται και η είσοδος **valid_data** βρίσκεται σε κατάσταση HIGH. Αυτό σημαίνει ότι σε κάθε παλμό στην είσοδο **clk_1**, ο δυαδικός αριθμός που βρίσκεται στην είσοδο **ds** με εύρος 16 bits, θα μετατρέπεται αυτόματα σε αριθμό με κωδικοποίηση BCD τεσσάρων ψηφίων (16 bits). Την δυνατότητα αυτή την παρέχει η διαδικασία *PROCESS*(clk_1). Οι εντολές περιγραφής κυκλωμάτων που εμπεριέχονται μέσα στην διαδικασία *PROCESS*, υλοποιούνται κυκλωματικά με τέτοιο τρόπο ώστε να εκτελούνται ταυτόχρονα, κάθε φορά που θα εμφανίζεται στην είσοδο **clk_1** το μέτωπο ανόδου ενός παλμού. Στο παράδειγμα του συγκεκριμένου κώδικα, υπάρχει μέσα στην διαδικασία *PROCESS*, ένας βρόγχος δώδεκα επαναλήψεων ο οποίος πραγματοποιεί την ολίσθηση των δυαδικών ψηφίων, σύμφωνα με τον αλγόριθμο που προαναφέρθηκε. Η κυκλωματική υλοποίηση του συγκεκριμένου βρόχου είναι κατάλληλη για να εκτελεστούν όλες η επαναλήψεις σε έναν κύκλο ρολογιού. Η δυνατότητα αυτή της διαδικασίας *PROCESS* είναι πολύ χρήσιμη για την κυκλωματική υλοποίηση κώδικα που απαιτεί εκτέλεση υψηλής ταχύτητας. Το γεγονός αυτό, έχει και το αντίστοιχο κόστος σε επίπεδο διαθέσιμου υλικού μέσα στο κύκλωμα FPGA, καθώς ο αριθμός των λογικών στοιχείων που απαιτούνται είναι μεγάλος.

Για την εισαγωγή του μετατροπέα binary to BCD στο project, πρέπει να δημιουργηθεί ένα διάγραμμα οντότητας (block) και να συνδεθεί στην υπάρχουσα κυκλωματική διάταξη. Το συγκεκριμένο block θα απεικονίζει διαγραμματικά τον κώδικα VHDL για την υλοποίηση του μετατροπέα binary to BCD. Για να εισάγουμε τον κώδικα επιλέγουμε:

File→ **New** → **VHDL File**



Άμεσα εμφανίζεται ο κειμενογράφος για την εισαγωγή του κώδικα VHDL. Εισάγουμε τον παραπάνω κώδικα για την υλοποίηση του μετατροπέα binary to BCD και αποθηκεύουμε το αρχείο με όνομα **BinarytoBCD.vhd**, στον φάκελο του project.

```

1  library ieee;
2  USE ieee.std_logic_1164.all;
3
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY BinarytoBCD IS
7  PORT (clk_1 : in std_logic;
8        valid_data: in std_logic;
9        ds: in std_logic_vector (15 downto 0);
10       qs: OUT std_logic_vector (15 downto 0));
11  END BinarytoBCD;
12
13  ARCHITECTURE behaviour OF BinarytoBCD IS
14  BEGIN
15  PROCESS (clk_1)
16
17
18     variable i : integer:=0;
19     variable bcd : std_logic_vector(15 downto 0) ;
20     variable bint : std_logic_vector(15 downto 0) ;
21
22     begin
23     if ( clk_1'event and clk_1 = '1' and valid_data = '1' ) then
24     bcd:="0000000000000000";
25     bint:=ds;
26     for i in 0 to 15 loop -- repeating 16 times.
27     bcd(15 downto 1) := bcd(14 downto 0); --shifting the bits.
28     bcd(0) := bint(15);
29     bint(15 downto 1) := bint(14 downto 0);
30     bint(0) :='0';
31
32
33     if(i < 15 and bcd(3 downto 0) > "0100") then
34     bcd(3 downto 0) := bcd(3 downto 0) + "0011";
35     end if;
36
37     if(i < 15 and bcd(7 downto 4) > "0100") then
38     bcd(7 downto 4) := bcd(7 downto 4) + "0011";
39     end if;
40
41     if(i < 15 and bcd(11 downto 8) > "0100") then
42     bcd(11 downto 8) := bcd(11 downto 8) + "0011";
43     end if;
44
45     if(i < 15 and bcd(15 downto 12) > "0100") then
46     bcd(15 downto 12) := bcd(15 downto 12) + "0011";
47     end if;
48
49     end loop;
50     qs<=bcd;
51     end if;
52   END process;
53   END behaviour;
54

```

Έπειτα επιλέγουμε:

Processing → Analyze Current File

Με αυτή την διαδικασία πραγματοποιείται ο έλεγχος για την ορθότητα του VHDL κώδικα που εμπεριέχεται στο αρχείο. Εφόσον δεν υπάρχουν συντακτικά λάθη στον κώδικα, εμφανίζεται το παρακάτω μήνυμα:

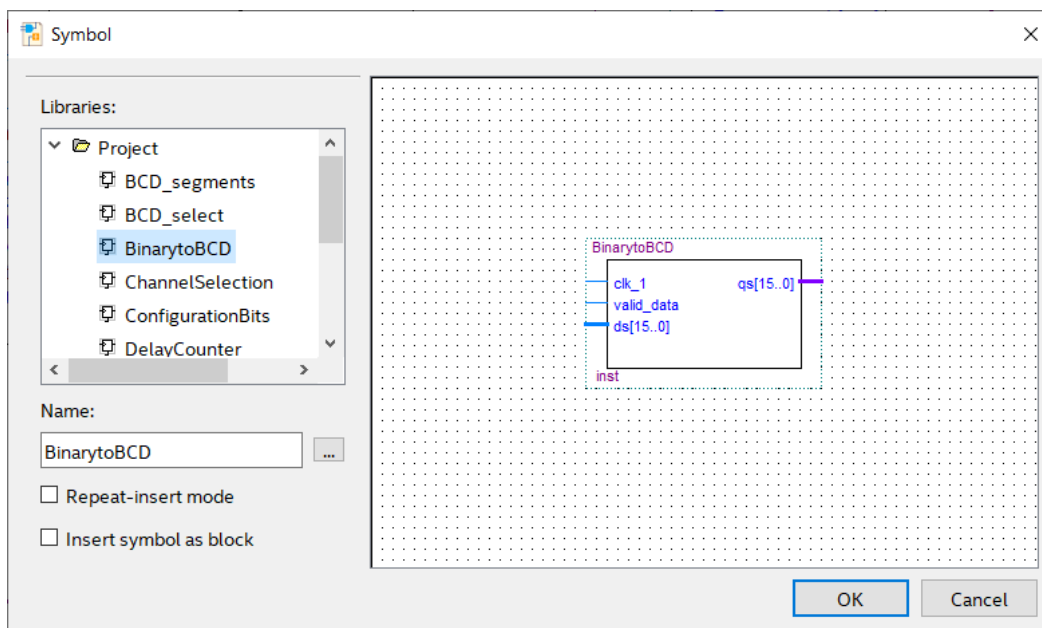
Type	ID	Message

		Running Quartus Prime Analyze Current File
		Command: quartus_map --read_settings_files=on --write_settings_files=off SimpleADCTest -c ADC
		18236 Number of processors has not been specified which may cause overloading on shared machines.
		20030 Parallel compilation is enabled and will use 2 of the 2 processors detected
		Quartus Prime Analyze Current File was successful. 0 errors, 1 warning

Για την δημιουργία του διαγράμματος οντότητας για τον μετατροπέα binary to BCD, επιλέγουμε:

File → Create/Update → Create Symbol Files for Current File

Η επιλογή αυτή, δημιουργεί αρχείο συμβόλου για τον VHDL κώδικα που απεικονίζει διαγραμματικά την οντότητα με τις εισόδους και εξόδους που έχουν δηλωθεί. Έπειτα επιλέγουμε την εισαγωγή συμβόλου στο project και βλέπουμε ότι στην βιβλιοθήκη συμβόλων του project, έχει προστεθεί και το σύμβολο της οντότητας BinarytoBCD που μόλις δημιουργήσαμε. Το σύμβολο απεικονίζει διαγραμματικά όλες τις εισόδους και εξόδους που έχουν δηλωθεί στον κώδικα VHDL.

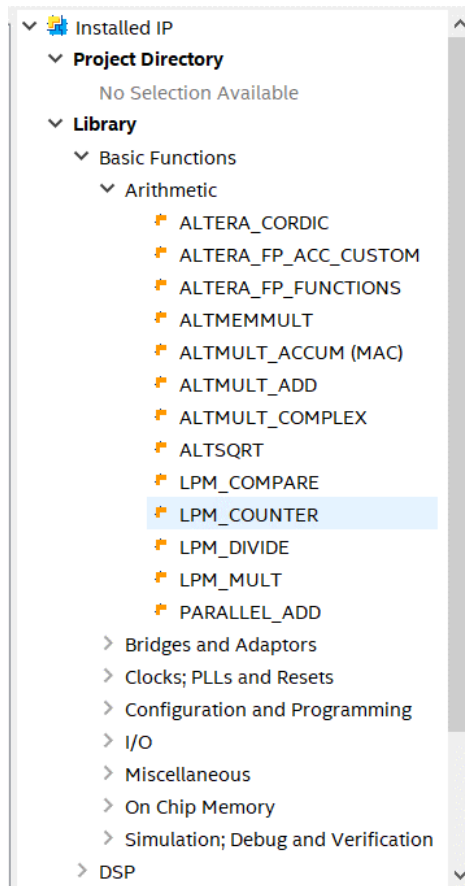


7.8 Δημιουργία κυκλώματος χρονικής καθυστέρησης για καλύτερο οπτικό αποτέλεσμα της δεκαδικής απεικόνισης.

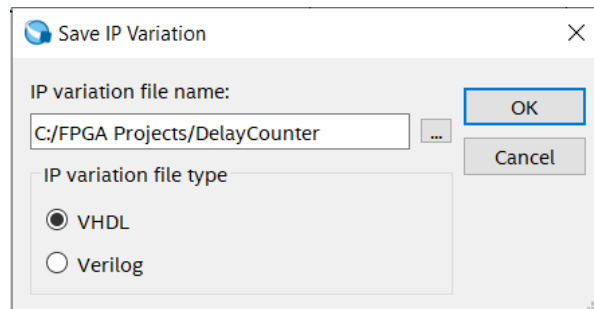
Με την εισαγωγή του συμβόλου στο project, πραγματοποιούμε και τις κατάλληλες συνδέσεις των εισόδων και των εξόδων (Σχήμα 28). Για τον σκοπό αυτό, ο διάυλος εισόδου **ds[15..0]** συνδέεται με την έξοδο του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής **response_data[11..0]** της οντότητας MyADC, μέσω του διαύλου **Data[15..0]**. Για τον χρονισμό της μετατροπής του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής, χρησιμοποιείται η είσοδος **clk_1** σε συνδυασμό με το σήμα εξόδου **response_valid** της οντότητας MyADC. Η έξοδος αυτή ενεργοποιείται κάθε φορά που υπάρχουν στην έξοδο αποτελέσματος **response_data[11..0]** διαθέσιμα έγκυρα δεδομένα της αναλογικής/ψηφιακής μετατροπής. Με τον τρόπο αυτό, κάθε φορά που θα υπάρχει ένα έγκυρο αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής, η είσοδος **ds[15..0]** της οντότητας BinarytoBCD θα εισάγει τα δεδομένα προς μετατροπή binary to BCD και αυτομάτως θα εξάγεται το αποτέλεσμα στην έξοδο της οντότητας **qs[11..0]**. Στην περίπτωση που η συχνότητα του ρολογιού **clk_1** συνδεθεί με την έξοδο **c0**, η μετατροπή του δυαδικού αριθμού σε BCD, θα πραγματοποιείται με υψηλή ταχύτητα λόγω της υψηλής ταχύτητας της πηγής χρονισμού **c0** της οντότητας MyPLL. Το γεγονός αυτό δεν είναι θεμιτό για την εφαρμογή που αναπτύσσουμε, γιατί ο σκοπός της εφαρμογής είναι η οπτική απεικόνιση του αποτελέσματος την αναλογικής/ψηφιακής μετατροπής, στην οθόνη των τεσσάρων δεκαδικών ψηφίων LED επτά τμημάτων. Μέσα στο αναλογικό σήμα που προκύπτει από την έξοδο του ποτενσιομέτρου, υπάρχει ηλεκτρονικός θόρυβος της τάξης μερικών δεκάδων mV. Το γεγονός αυτό έχει ως αποτέλεσμα την ανίχνευση και καταγραφή του ηλεκτρονικού θορύβου στο δυαδικό αποτέλεσμα της αναλογικής/ψηφιακής μετατροπής. Έτσι η άμεση απόκριση του συστήματος στην παρουσία έγκυρων δεδομένων αναλογικής/ψηφιακής μετατροπής, θα προκαλούσε οπτική ασάφεια στο πρώτο δεκαδικό ψηφίο της κωδικοποίησης BCD. Η χρήση της πηγής χρονισμού **c0** για την εφαρμογή μας δεν είναι η κατάλληλη. Για τον λόγο αυτό πρέπει να δημιουργηθεί μια καθυστέρηση στην ανάγνωση του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής. Η καταλληλότερη μέθοδος για την δημιουργία καθυστέρησης στην ανάγνωση και μετατροπή του αναλογικού/ψηφιακού αποτελέσματος σε BCD, είναι η χρήση ενός διαιρέτη για την συχνότητα των παλμών του ρολογιού **c0**, έτσι ώστε η συχνότητα μετατροπής του αναλογικού/ψηφιακού αποτελέσματος σε BCD και η απεικόνισή του στην οθόνη, να μην

είναι τόσο γρήγορη για το ανθρώπινο μάτι και να μη δημιουργεί οπτική ασάφεια. Η υλοποίηση του διαιρέτη των παλμών του ρολογιού είναι ένας δυαδικός απαριθμητής (counter), που σαν είσοδο χρονισμού θα έχει το σήμα των παλμών του ρολογιού c0 και σαν έξοδο για την διαίρεση της συχνότητας θα παίρνουμε ένα από τα δυαδικά ψηφία εξόδου της μέτρησης των παλμών εισόδου. Με αυτό τον τρόπο, η διαίρεση πραγματοποιείται με διαιρέτη που εξαρτάται από την τάξη του δυαδικού ψηφίου της εξόδου μέτρησης που θα επιλέξουμε και είναι πάντα κάποια δύναμη του αριθμού 2. Η μέτρηση των παλμών εισόδου σε έναν δυαδικό απαριθμητή πραγματοποιείται είτε κατά το ανοδικό μέτωπο του παλμού εισόδου, είτε κατά το καθοδικό μέτωπο του παλμού εισόδου. Το πρώτο δυαδικό ψηφίο (2^0) του δυαδικού αποτελέσματος μέτρησης των παλμών, εναλλάσσεται σε κάθε δεύτερο παλμό εισόδου, με αποτέλεσμα να διαιρεί τη συχνότητα των παλμών εισόδου με διαιρέτη το 2. Αντίστοιχα το δεύτερο δυαδικό ψηφίο (2^1) του δυαδικού αποτελέσματος μέτρησης των παλμών, διαιρεί την συχνότητα των παλμών εισόδου με διαιρέτη το 4, το τρίτο δυαδικό ψηφίο (2^2) του δυαδικού αποτελέσματος μέτρησης των παλμών, διαιρεί την συχνότητα των παλμών εισόδου με διαιρέτη το 8 κ.ο.κ. Για την χρήση του δυαδικού απαριθμητή από τον κατάλογο των IP που παρέχει η Intel επιλέγουμε:

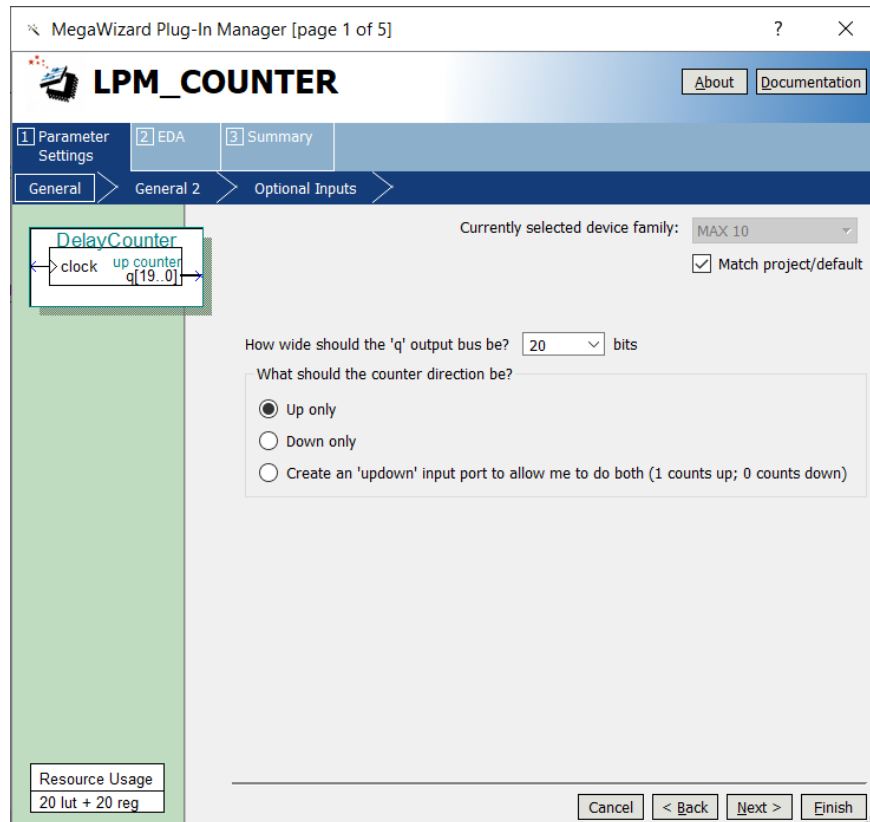
Tools → **IP Catalog** → **Installed IP** → **Library** → **Basic Functions** → **Arithmetic** → **LPM_COUNTER**



Επιλέγουμε το όνομα της οντότητας του απαριθμητή **DelayCounter** και ως τύπο κώδικα τον VHDL.



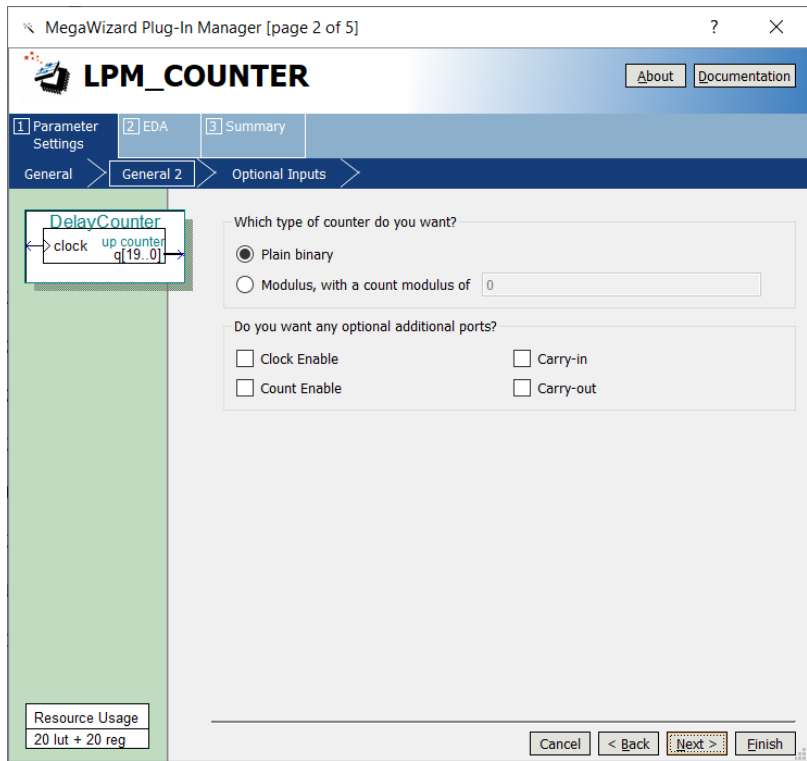
Αμέσως μετά, εκτελείται αυτόματα η διεπαφή MegaWizard για την παραμετροποίηση του απαριθμητή.



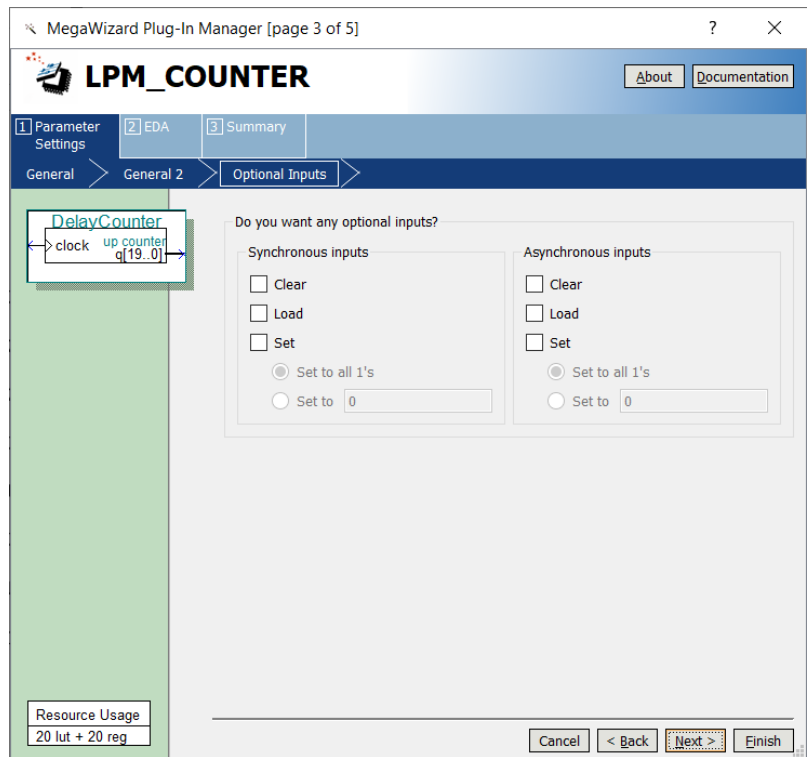
Επιλέγουμε το εύρος του απαριθμητή σε 20 bits για να πραγματοποιήσουμε διαίρεση της συχνότητας των 10MHz του ρολογιού c0, με διαιρέτη το 2^{16} που αντιστοιχεί σε μια συχνότητα περίπου 152Hz. Η συχνότητα αυτή είναι φυσιολογική, ως ρυθμός ανανέωσης του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής για την οθόνη LED τεσσάρων δεκαδικών ψηφίων και δεν προκαλεί οπτική ασάφεια για το ανθρώπινο μάτι.

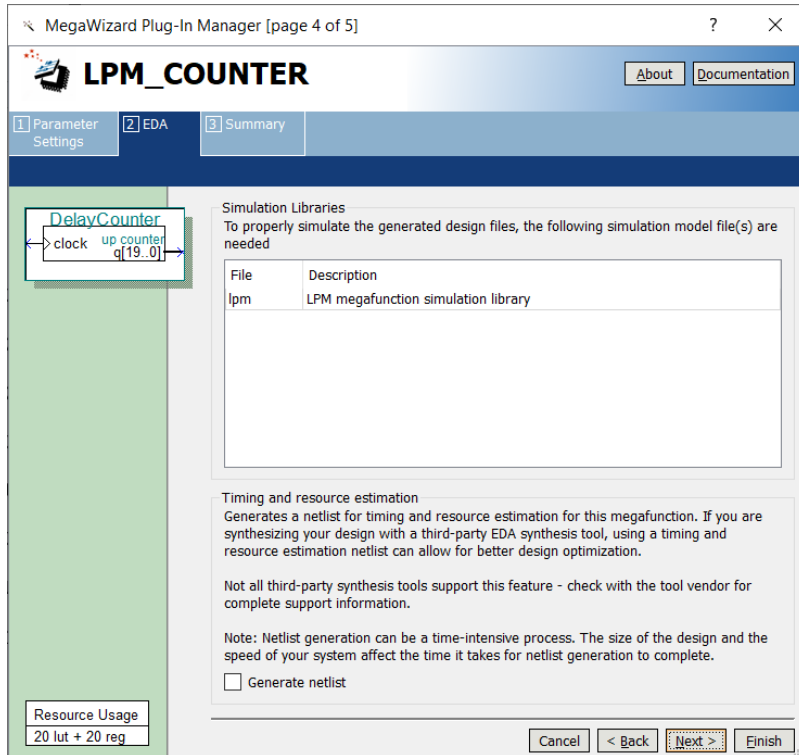
Επιλέγουμε **Next**.

Στην νέα διεπαφή επιλέγουμε την επιλογή **Plain binary** και έπειτα επιλέγουμε **Next**

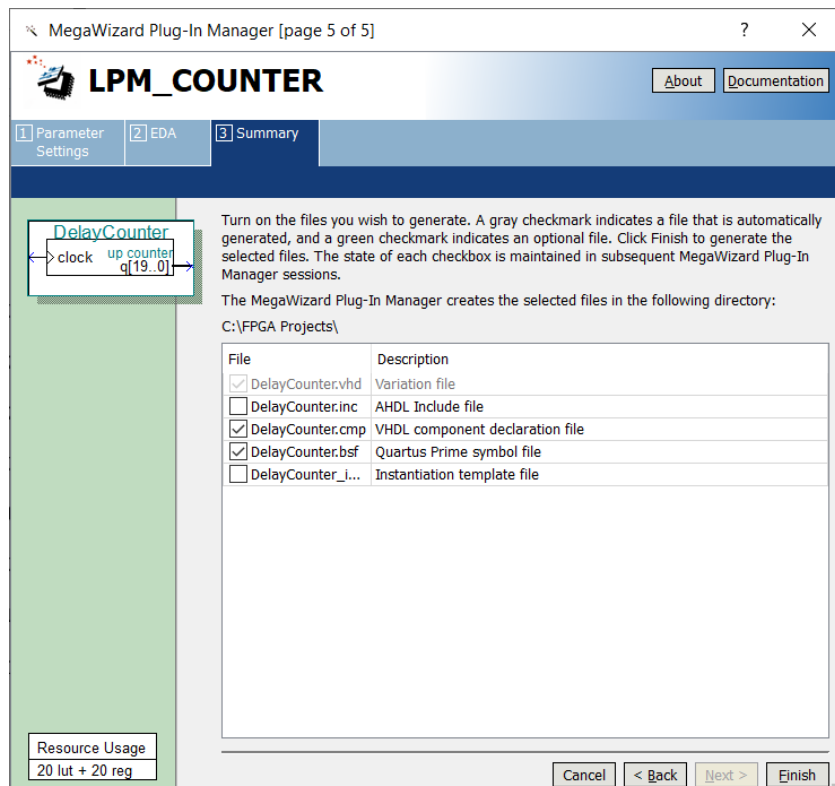


Στις επόμενες δύο διεπαφές που εμφανίζονται επιλέγουμε απλά **Next**

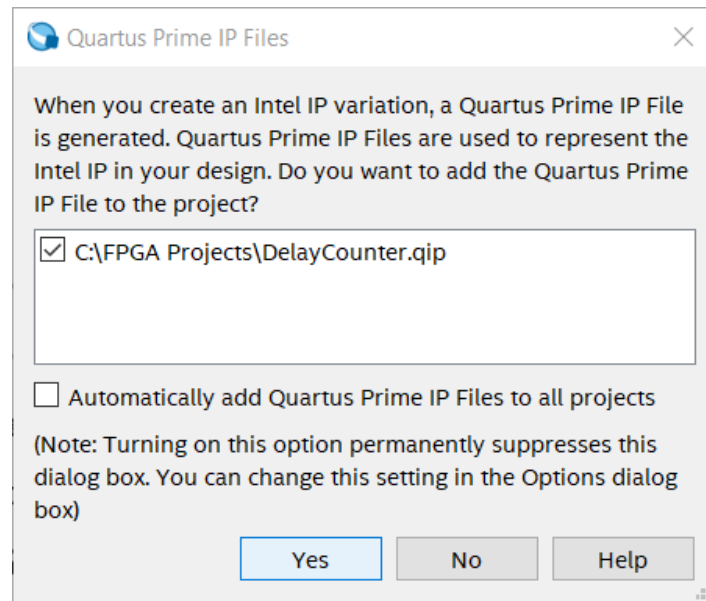




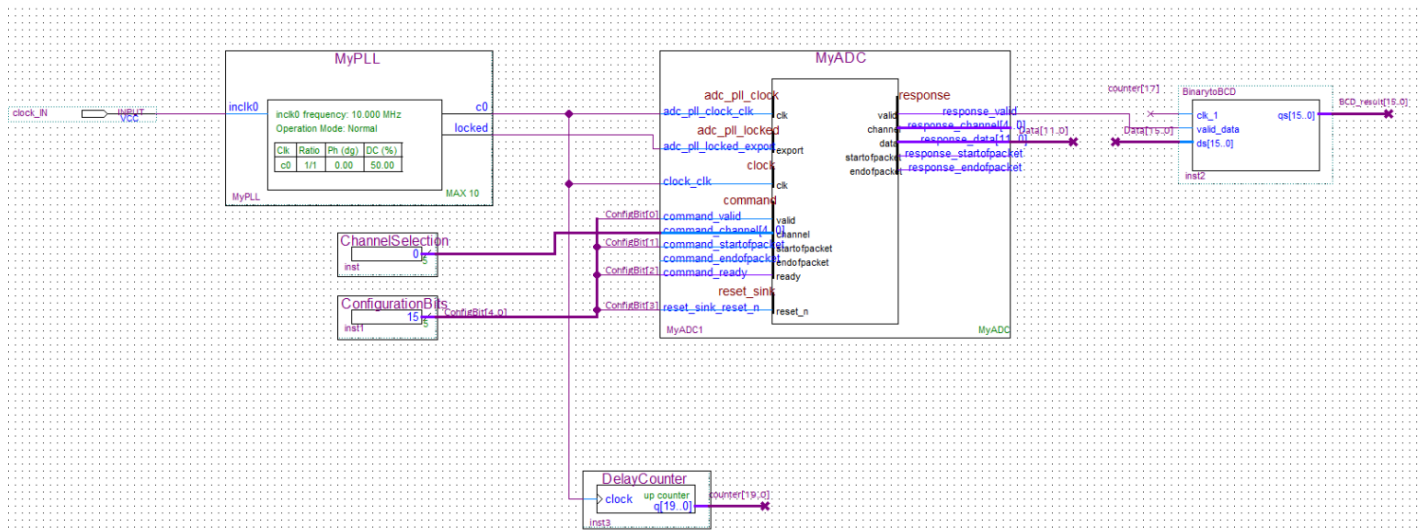
Έπειτα επιλέγουμε την δημιουργία συμβόλου για τον απαριθμητή που παραμετροποιήσαμε, με την επιλογή **Quartus Prime symbol file** και επιλέγουμε **Finish**.



Αμέσως το Quartus Prime προτρέπει για την χρήση της οντότητας του απαριθμητή 20 bits που μόλις δημιουργήθηκε, στο συγκεκριμένο project.



Επιβεβαιώνουμε την χρήση του απαριθμητή στο συγκεκριμένο project και τον εισάγουμε με την γνωστή διαδικασία εισαγωγής συμβόλου, καθώς το σύμβολο έχει ενσωματωθεί με την παραπάνω διαδικασία στην βιβλιοθήκη συμβόλων του project. Η σύνδεση του συμβόλου του απαριθμητή πραγματοποιείται όπως απεικονίζεται στο παρακάτω διάγραμμα.

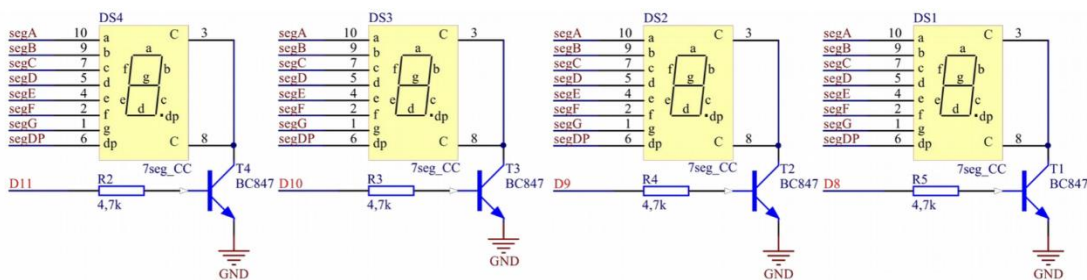


Σχήμα 28. Η σύνδεση του απαριθμητή DelayCounter και του μετατροπέα BinarytoBCD στα κυκλώματα της εφαρμογής.

Όπως φαίνεται στο παραπάνω διάγραμμα, η είσοδος του απαριθμητή είναι η πηγή χρονισμού **c0** και από την έξοδο δυαδικού αποτελέσματος απαρίθμησης **counter[19..0]**, έχει επιλεγεί το σήμα **counter[17]** για να αποτελέσει την είσοδο χρονισμού του binary to BCD μετατροπέα, με συχνότητα περίπου 152Hz.

7.9 Τα κυκλώματα οδήγησης των ψηφίων LED δεκαδικής απεικόνισης.

Η οντότητα BinarytoBCD έχει παραμετροποιηθεί και συνδεθεί κατάλληλα, για να παρέχει αποτελέσματα προς απεικόνιση για την οθόνη LED, με συχνότητα 152Hz. Το αποτέλεσμα της οντότητας BinarytoBCD, πρέπει να οδηγηθεί στην οθόνη απεικόνισης που αποτελείται από τέσσερα ψηφία δεκαδικής απεικόνισης, που εμπεριέχουν επτά φωτεινά τμήματα LED έκαστο. Στο παρακάτω σχηματικό διάγραμμα του τυπωμένου κυκλώματος που περιέχει τα ψηφία δεκαδικής απεικόνισης, διακρίνεται η σύνδεση όλων των σημάτων ανόδου των τομέων LED σε έναν διάυλο (**segA**, **segB**, **segC**, **segD**, **segE**, **segF**, **segG**, **SegDP**). Η επιλογή του ψηφίου γίνεται με πολυπλεξία, μέσω της επιλογής του σήματος καθόδου **D8**, **D9**, **D10**, **D11**, του ψηφίου που πρόκειται να ενεργοποιηθεί.



Σχήμα 29. Σχηματικό διάγραμμα του κυκλώματος οδήγησης των ψηφίων δεκαδικής απεικόνισης LED επτά τομέων (MAXimator, Schematics).

Για την οδήγηση των ψηφίων δεκαδικής απεικόνισης, θα πρέπει να δημιουργηθεί μία οντότητα, η οποία θα κωδικοποιεί κάθε τετράδα ψηφίων BCD που προκύπτει από τον binary to BCD μετατροπέα, στην αντίστοιχη απεικόνιση από τους τομείς LED στα ψηφία της οθόνης. Ο κώδικας VHDL που υλοποιεί την οντότητα οδήγησης των ψηφίων δεκαδικής απεικόνισης σύμφωνα με την αντίστοιχη τιμή της εξόδου του binary to BCD μετατροπέα, παρουσιάζεται παρακάτω.

```

library ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY BCD_segments IS
PORT (en_out:in std_logic;
      bcd_in: IN std_logic_vector (3 downto 0);
      segments: out std_logic_vector (6 downto 0));
END BCD_segments;

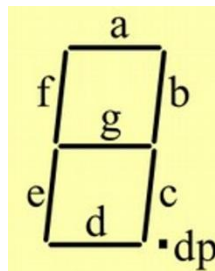
ARCHITECTURE behaviour OF BCD_segments IS
BEGIN
    segments <= "1111110" when bcd_in = "0000" and en_out = '1' else
                "1110011" when bcd_in = "1001" and en_out = '1' else
                "1111111" when bcd_in = "1000" and en_out = '1' else
                "1110000" when bcd_in = "0111" and en_out = '1' else
                "0011111" when bcd_in = "0110" and en_out = '1' else
                "1011011" when bcd_in = "0101" and en_out = '1' else
                "0110011" when bcd_in = "0100" and en_out = '1' else
                "1111001" when bcd_in = "0011" and en_out = '1' else
                "1101101" when bcd_in = "0010" and en_out = '1' else
                "0110000" when bcd_in = "0001" and en_out = '1' else
                "ZZZZZZZ";
END behaviour;

```

Σύμφωνα με τον παραπάνω κώδικα, για κάθε τιμή των τεσσάρων ψηφίων BCD που εισάγονται από την είσοδο **bcd_in[3..0]**, ενεργοποιούνται τα κατάλληλα σήματα ανόδου στην έξοδο της οντότητας **segments[6..0]**, προκειμένου να απεικονισθεί το αντίστοιχο δεκαδικό ψηφίο. Κάθε ένα από τα επτά σήματα του διαύλου **segments[6..0]**, αντιστοιχεί στην άνοδο ενός τομέα LED του ψηφίου απεικόνισης, σύμφωνα με τον παρακάτω πίνακα (Πίνακας 5).

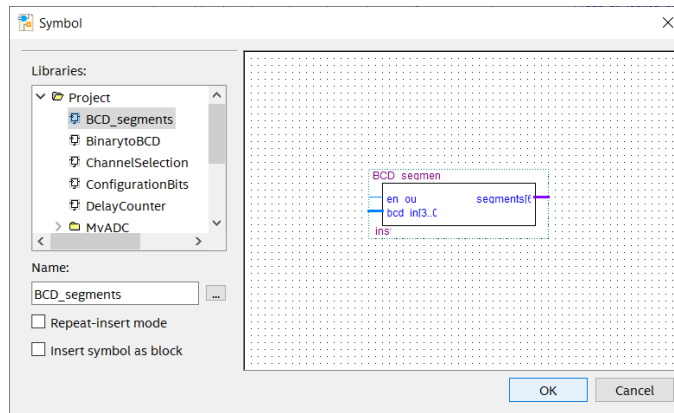
Πίνακας 5. Αντιστοιχία των σημάτων διαύλου `segment[6..0]` με τους τομείς LED ενός ψηφίου δεκαδικής απεικόνισης.

Σήμα διαύλου <code>segment[6..0]</code>	Τομέας LED
<code>segment[6]</code>	g
<code>segment[5]</code>	f
<code>segment[4]</code>	e
<code>segment[3]</code>	d
<code>segment[2]</code>	c
<code>segment[1]</code>	b
<code>segment[0]</code>	a

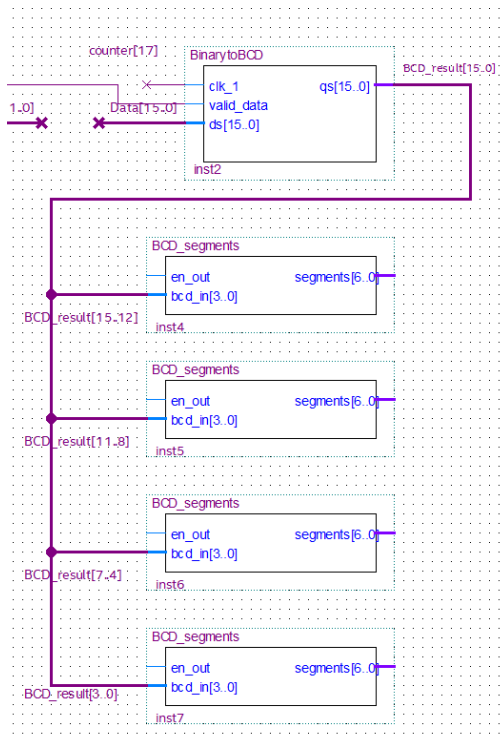


Η είσοδος ενεργοποίησης του οδηγού `en_out` αποτελεί το σήμα επιλογής του ψηφίου που πρέπει να απεικονισθεί κάθε φορά. Σε περίπτωση που το σήμα `en_out` βρίσκεται σε κατάσταση LOW, τότε η έξοδος `segments[6..0]` για την συγκεκριμένη οντότητα βρίσκεται σε κατάσταση υψηλής εμπέδησης (κατάσταση 'Z'), για να επιτρέψει την εφαρμογή των σημάτων από διαφορετική οντότητα οδηγού ψηφίου LED, στον δίαυλο απεικόνισης. Με αυτό τον τρόπο διευκολύνεται η πολυπλεξία των οδηγών ψηφίων LED, πάνω στον δίαυλο ενεργοποίησης των τομέων LED.

Για την οντότητα του οδηγού ψηφίου LED, δημιουργούμε αρχείο συμβόλου με την διαδικασία που προαναφέρθηκε και με το όνομα `BCD_segments`.



Εισάγουμε τέσσερις οντότητες BCD_segments στο project, μία για κάθε ψηφίο BCD που εξάγεται από την οντότητα BinarytoBCD. Κάθε μία από τις οντότητες BCD_segments, προορίζεται να ελέγχει τους επτά τομείς LED από ένα ψηφίο δεκαδικής απεικόνισης της οθόνης LED. Δημιουργούμε έναν δίαυλο με όνομα **BCD_result[15..0]** ο οποίος θα φέρει το αποτέλεσμα της binary to BCD μετατροπής και αποδίδουμε την κάθε τετράδα σημάτων BCD, στην αντίστοιχη οντότητα BCD_segments. Έτσι η οντότητα BCD_segments που θα απεικονίζει το ψηφίο των μονάδων, έχει ως είσοδο τα σήματα **BCD_result[3..0]**, η οντότητα BCD_segments που θα απεικονίζει το ψηφίο των δεκάδων, έχει ως είσοδο τα σήματα **BCD_result[7..4]**, η οντότητα BCD_segments που θα απεικονίζει το ψηφίο των εκατοντάδων, έχει ως είσοδο τα σήματα **BCD_result[11..8]** και τέλος, η οντότητα BCD_segments που θα απεικονίζει το ψηφίο των χιλιάδων, έχει ως είσοδο τα σήματα **BCD_result[15..12]**. Η σύνδεση των τεσσάρων συμβόλων BCD_segments με τον δίαυλο **BCD_result[15..0]**, απεικονίζεται στο Σχήμα 30.

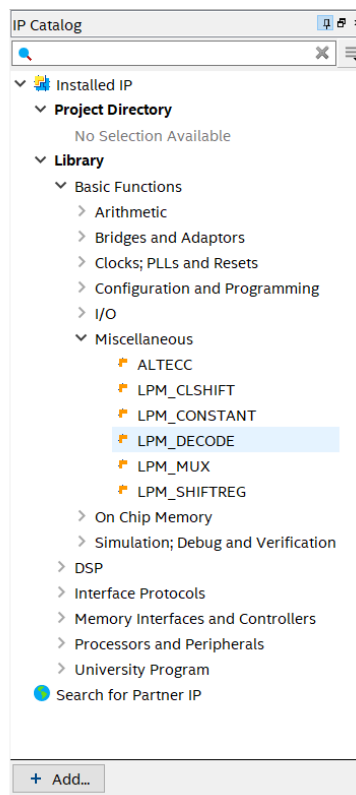


Σχήμα 30. Η σύνδεση των τεσσάρων συμβόλων *BCD_segments* με τον διάυλο *BCD_result[15..0]*.

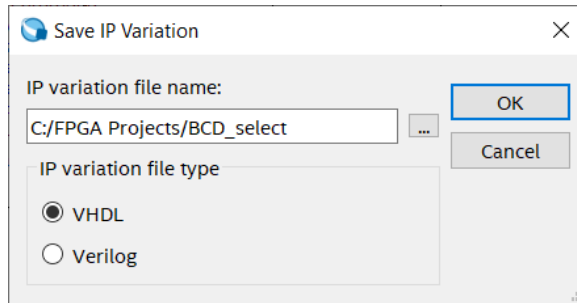
Λόγω του ότι οι έξοδοι **segment[6..0]**, θα συνδέονται πάνω στον ίδιο διάυλο προκειμένου να οδηγήσουν τα τμήματα LED των δεκαδικών ψηφίων της οθόνης, θα πρέπει να σχεδιαστεί κυκλωματική διάταξη πολυπλεξίας η οποία θα επιτρέπει κάθε φορά την εμφάνιση ενός μόνο ψηφίου πάνω στον διάυλο. Η κυκλωματική διάταξη αυτή, θα πρέπει να σαρώνει τους τέσσερεις οδηγούς των δεκαδικών ψηφίων, με συχνότητα τέτοια ώστε η σάρωση να μην είναι εμφανής στο οπτικό αποτέλεσμα (flickering). Για την πολυπλεξία (σάρωση) των οδηγών δεκαδικών ψηφίων *BCD_segments*, θα χρησιμοποιηθεί όπως προαναφέρθηκε το σήμα εισόδου **en_out**, το οποίο ενεργοποιεί την έξοδο του οδηγού όταν βρίσκεται σε λογική κατάσταση HIGH, ενώ παρουσιάζει υψηλή εμπέδηση στην έξοδο όταν η λογική κατάστασή του είναι LOW. Για τον έλεγχο του σήματος **en_out** με την κατάλληλη συχνότητα θα δημιουργηθεί ένας αποκωδικοποιητής με είσοδο δύο δυαδικών ψηφίων και έξοδο τεσσάρων σημάτων. Κάθε τιμή εκ των τεσσάρων που μπορούν να δημιουργηθούν στην δυαδική είσοδο, ενεργοποιεί μία εκ των τεσσάρων εξόδων. Με τον τρόπο αυτό, συνδέοντας κάθε έξοδο του αποκωδικοποιητή σε μία από τις εισόδους **en_out** των οδηγών *BCD_segments*, μπορεί να πραγματοποιηθεί η πολυπλεξία των ψηφίων δεκαδικής απεικόνισης LED. Η δυαδική είσοδος του αποκωδικοποιητή, θα

πρέπει ,όπως έχει αναφερθεί, να σαρώνει διαδοχικά τις δυαδικές τιμές 00, 01, 10, 11, με κατάλληλη συχνότητα. Για την διαδοχική σάρωση των παραπάνω δυαδικών τιμών, μπορούν να χρησιμοποιηθούν δύο διαδοχικά σήματα του μετρητή DelayCounter, που έχει ήδη εισαχθεί στο project. Έτσι, αν χρησιμοποιηθούν τα σήματα **counter[16..15]** ως είσοδοι για τον αποκωδικοποιητή, η συχνότητα σάρωσης που θα προκύψει από τις εναλλαγές του σήματος **counter[15]**, είναι περίπου 610Hz. Η συχνότητα αυτή είναι αρκετά υψηλή, έτσι ώστε να διασφαλίζει την ομαλή απεικόνιση του αποτελέσματος χωρίς οπτικά προβλήματα (flickering). Για την δημιουργία του αποκωδικοποιητή επιλέγουμε:

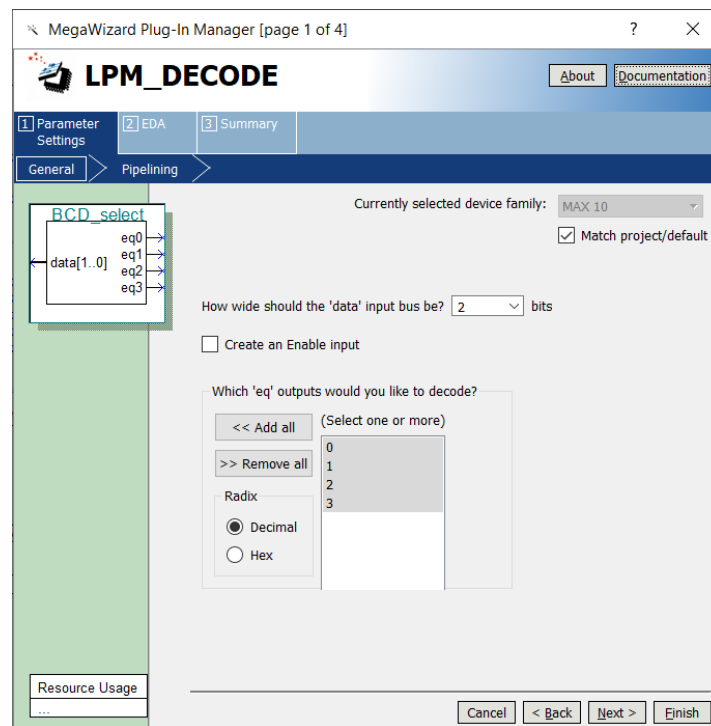
Tools→ **IP Catalog** → **Installed IP** → **Library** → **Basic Functions** → **Miscellaneous**→ **LPM_DECODE**



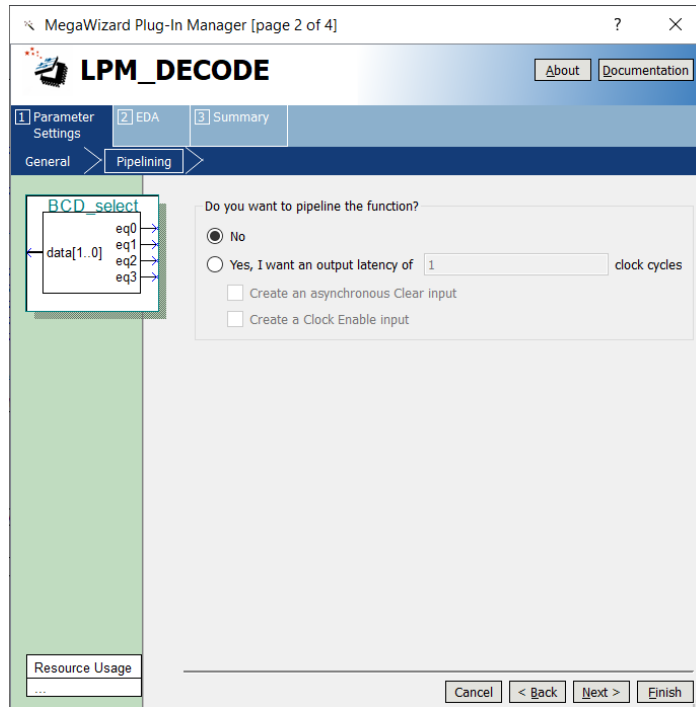
Στην διεπαφή που ακολουθεί επιλέγουμε κώδικα VHDL και για το όνομα της οντότητας του αποκωδικοποιητή, εισάγουμε το **BCD_select**.



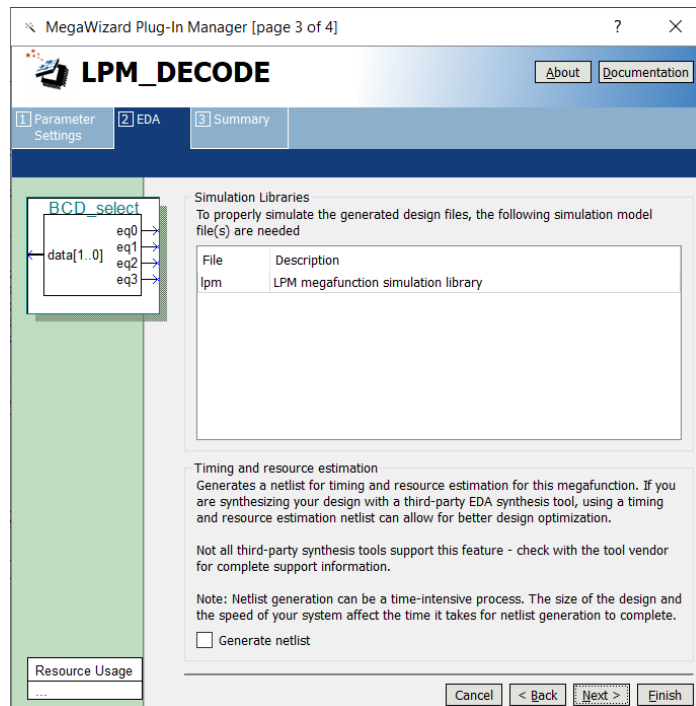
Άμεσα ξεκινά η εκτέλεση του MegaWizard για την παραμετροποίηση του αποκωδικοποιητή. Στην πρώτη διεπαφή επιλέγουμε το εύρος εισόδου του αποκωδικοποιητή στα 2bits, και ενεργοποιούμε όλα τα σήματα εξόδου που προκύπτουν με την επιλογή **Add all**. Έπειτα επιλέγουμε **Next**.



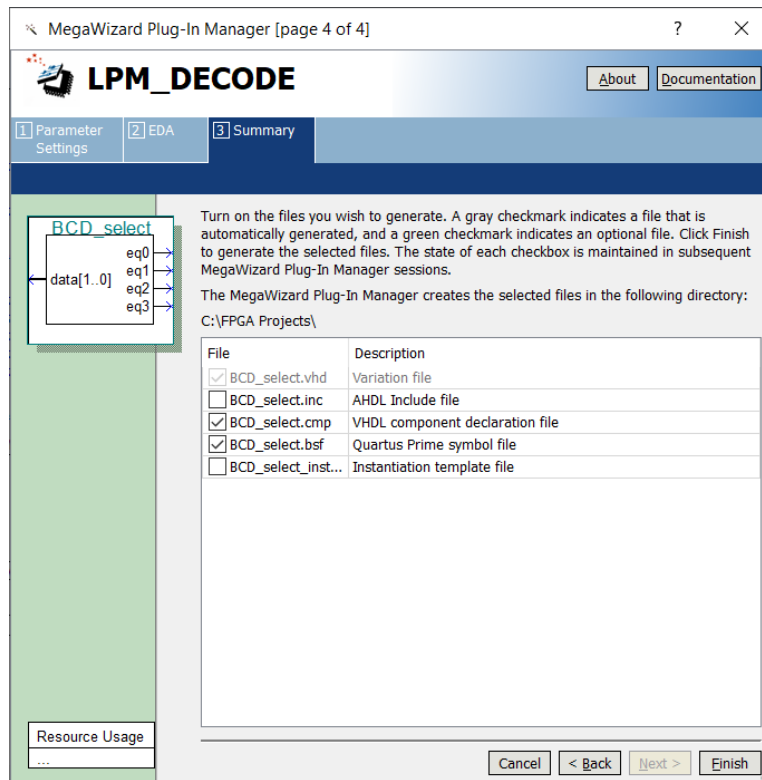
Στην επόμενη διεπαφή ενεργοποιούμε την επιλογή **No**, και επιλέγουμε **Next**.



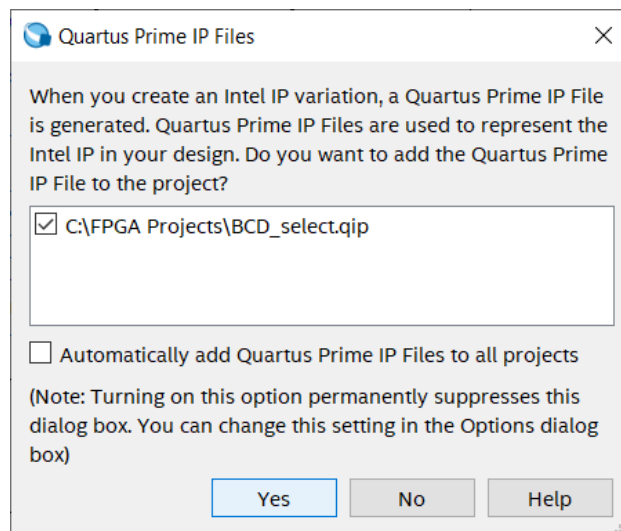
Έπειτα επιλέγουμε **Next** χωρίς να αλλάξουμε καμία από τις επιλογές της επόμενης διεπαφής.



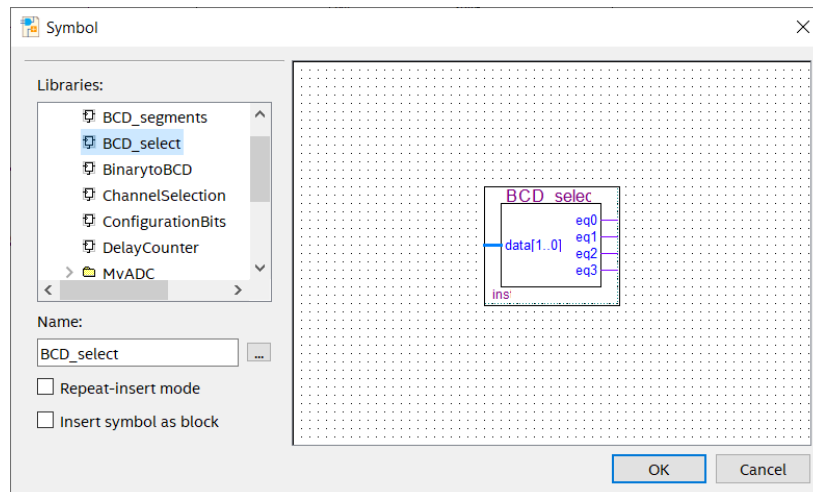
Στην τελευταία διεπαφή της εφαρμογής MegaWizard, ενεργοποιούμε την επιλογή δημιουργίας του αρχείου συμβόλου και επιλέγουμε **Finish**.



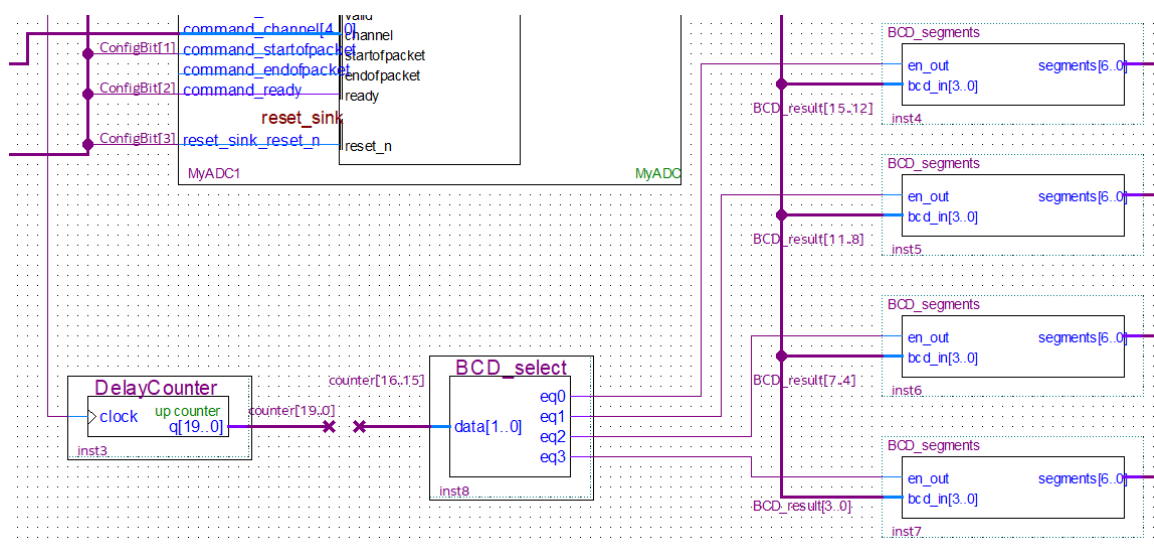
Επίσης επιλέγουμε την προσθήκη της οντότητας του αποκωδικοποιητή στον τρέχον project.



Με τον τρόπο αυτό έχει δημιουργηθεί και παραμετροποιηθεί ο αποκωδικοποιητής για την σάρωση των οδηγών ψηφίων δεκαδικής απεικόνισης. Ο αποκωδικοποιητής βρίσκεται στη βιβλιοθήκη συμβόλων του project.



Επιλέγουμε την εισαγωγή του συμβόλου στην ανώτερη οντότητα του project και πραγματοποιούμε τις συνδέσεις που προαναφέρθηκαν (Σχήμα 31).

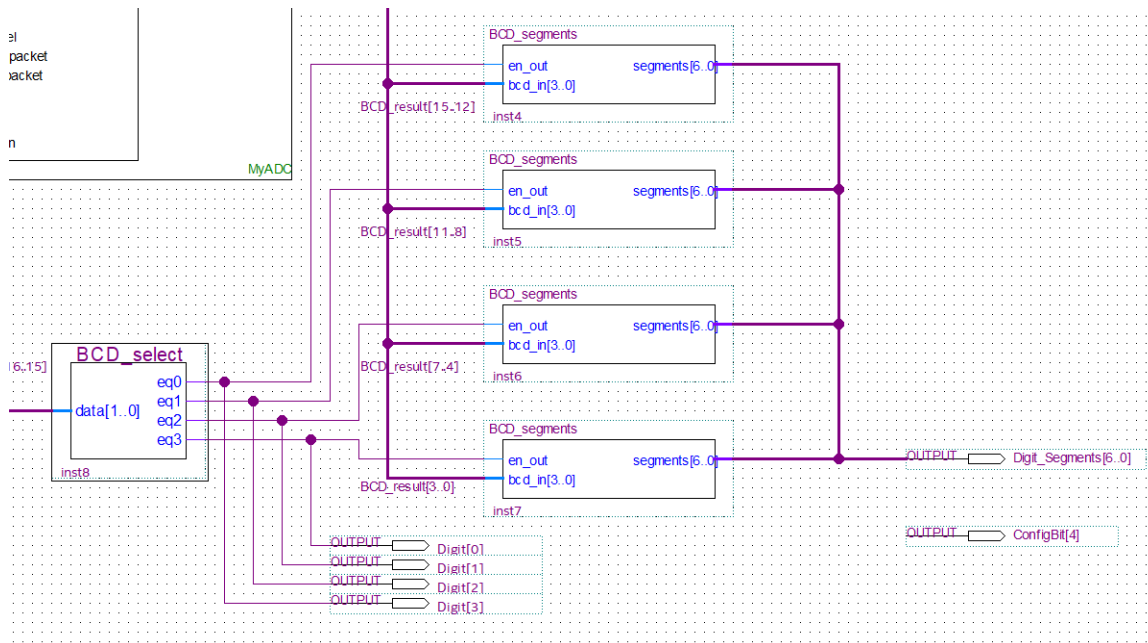


Σχήμα 31. Η σύνδεση του αποκωδικοποιητή **BCD_select** με τους οδηγούς **BCD_segments**.

Είναι εύκολα αντιληπτό, ότι ο συγκεκριμένος αποκωδικοποιητής, θα μπορούσε να δημιουργηθεί και με κώδικα VHDL, αντί να χρησιμοποιηθεί η έτοιμη οντότητα που παρέχει η βιβλιοθήκη του Quartus Prime. Με την εισαγωγή του αποκωδικοποιητή στο project, ολοκληρώθηκε και η κυκλωματική διάταξη απεικόνισης του αποτελέσματος της αναλογικής/ψηφιακής μετατροπής.

7.10 Δημιουργία εισόδων-εξόδων για την εφαρμογή.

Το σύνολο των κυκλωματικών διατάξεων που περιεγράφηκαν, θα υλοποιηθούν εσωτερικά στο κύκλωμα FPGA. Οι κυκλωματικές διατάξεις αυτές, θα πρέπει να συνδεθούν με το εξωτερικό περιβάλλον του κυκλώματος FPGA μέσω των ακροδεκτών που διαθέτει. Για τον λόγο αυτό πρέπει να δημιουργηθούν στην εφαρμογή οι κατάλληλες εισοδοι και έξοδοι. Μέχρι στιγμής, οι εισοδοι που έχουν δημιουργηθεί, αφορούν την είσοδο των παλμών από το εξωτερικό κύκλωμα του ταλαντωτή 10MHz **clock_IN** και την αναλογική είσοδο **ANIN**, στο σήμα της οποίας θα πραγματοποιηθεί η αναλογική/ψηφιακή μετατροπή. Ως έξοδοι για την οδήγηση της οθόνης LED, θα εξαχθούν τα σήματα του διαύλου **segments[6..0]** καθώς επίσης και οι έξοδοι του αποκωδικοποιητή **eq0, eq1, eq2, eq3**, τα οποία ενεργοποιούν τις καθόδους των τομέων LED, για κάθε ψηφίο δεκαδικής απεικόνισης. Επίσης, για την απενεργοποίηση της τελείας (dot) που υπάρχει σε κάθε ψηφίο δεκαδικής απεικόνισης, πρέπει να εξαχθεί και ένα σήμα με κατάσταση LOW. Για την δημιουργία αυτού του σήματος μπορεί να προστεθεί ένα επιπλέον σήμα εξόδου (bit) στην οντότητα ConfigurationBits και έτσι η έξοδος της οντότητας να μετατραπεί σε **ConfigBit[4..0]**. Από την συγκεκριμένη έξοδο, το σήμα **ConfigBit[4]** το οποίο είναι σε κατάσταση LOW, θα χρησιμοποιηθεί για την απενεργοποίηση του τομέα LED της τελείας (dot), των ψηφίων δεκαδικής απεικόνισης. Η δημιουργία των εισόδων/εξόδων, πραγματοποιείται με την χρήση του **Pin Tool**, που βρίσκεται στην μπάρα εργαλείων του Quartus Prime. Οι έξοδοι που δημιουργήθηκαν απεικονίζονται στο Σχήμα 32.



Σχήμα 32. Οι έξοδοι **Digit[3..0]** και η σύνδεσή τους με τα κυκλώματα της εφαρμογής.

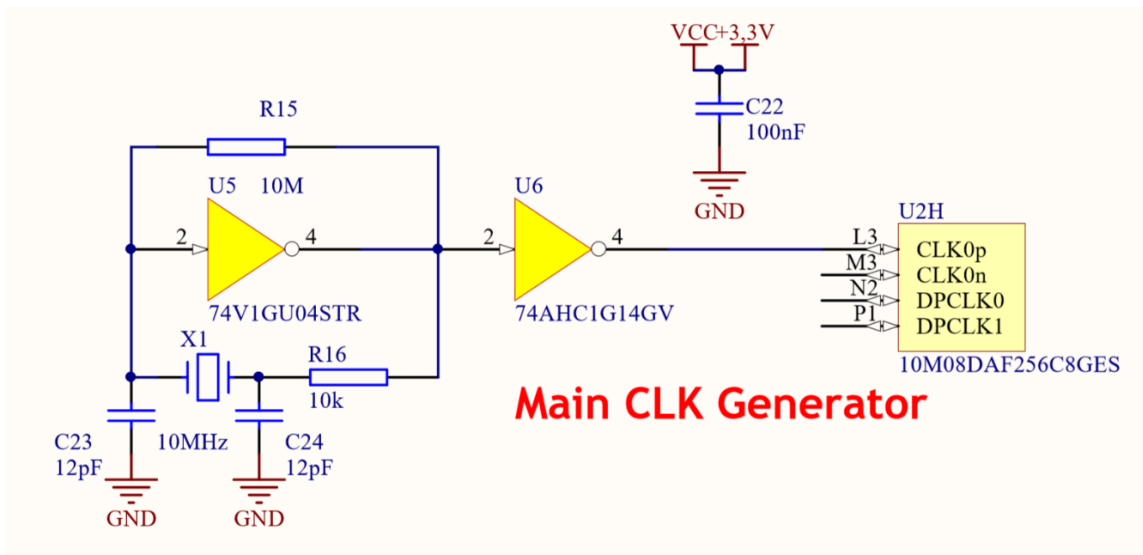
Για κάθε σήμα επιλογής δεκαδικού ψηφίου **eq0**, **eq1**, **eq2**, **eq3**, έχει δημιουργηθεί η αντίστοιχη έξοδος **Digit[0]**, **Digit[1]**, **Digit[2]**, **Digit[3]**. Επίσης για τον διάλογο ενεργοποίησης των επτά τομέων LED **segments[6..0]**, δημιουργήθηκε ο διάλογος εξόδου **Digit_Segments[6..0]** και για την απενεργοποίηση του τομέα LED της τελείας, δημιουργήθηκε η έξοδος **ConfigBit[4]**.

Σε αυτό το σημείο έχουν δημιουργηθεί όλες οι απαραίτητες εισοδοι και έξοδοι του κυκλώματος της εφαρμογής. Στο επόμενο βήμα θα πρέπει κάθε είσοδος και έξοδος να αντιστοιχηθεί με τον κατάλληλο ακροδέκτη του κυκλώματος FPGA. Η αντιστοίχιση των εισόδων/εξόδων με τους κατάλληλους ακροδέκτες του κυκλώματος FPGA, θα πραγματοποιηθεί σύμφωνα με την κατασκευαστική αντιστοίχιση των ακροδεκτών του κυκλώματος FPGA, με τα περιφερειακά κυκλώματα του αναπτυξιακού τυπωμένου κυκλώματος Maximator, που απαιτούνται για την υλοποίηση της εφαρμογής.

Τα περιφερειακά κυκλώματα που απαιτούνται για την υλοποίηση της εφαρμογής είναι τα εξής:

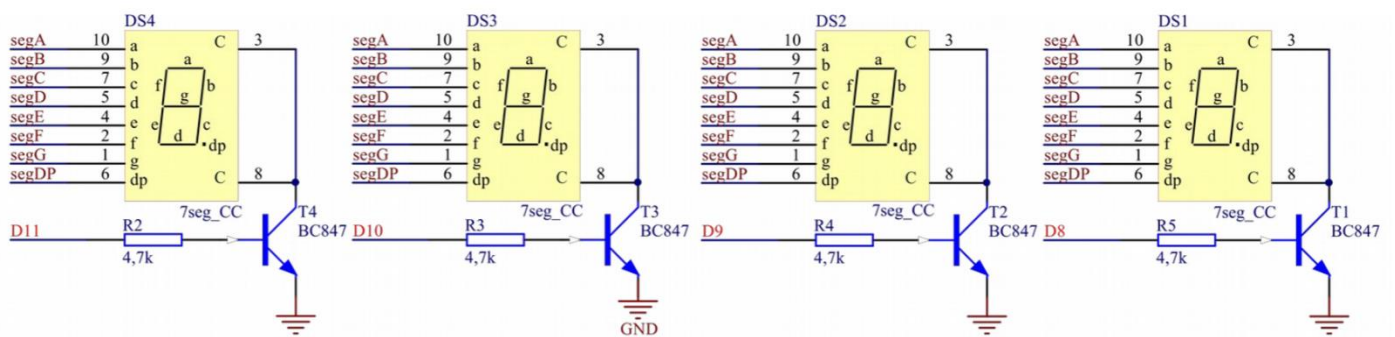
- Ο ταλαντωτής 10MHz (είσοδος)
- Η οθόνη με τα τέσσερα ψηφία δεκαδικής απεικόνισης LED (έξοδος).
- Το αναλογικό σήμα προς αναλογική/ψηφιακή μετατροπή (είσοδος)

Η είσοδος του αναλογικού σήματος **ANAIN1** που πρόκειται να μετατραπεί ψηφιακά, έχει ήδη επιλεγεί κατά την διαδικασία δημιουργίας και παραμετροποίησης της οντότητας MyADC. Για την είσοδο του ταλαντωτή των 10MHz, αν ανατρέξουμε στο σχηματικό διάγραμμα του τυπωμένου κυκλώματος (Σχήμα 33), θα διαπιστώσουμε ότι η έξοδος του ταλαντωτή, είναι συνδεδεμένη με την είσοδο χρονισμού **L3** του κυκλώματος FPGA.



Σχήμα 33. Η κυκλωματική διάταξη του ταλαντωτή 10MHz και η σύνδεσή του με το κύκλωμα FPGA (MAXimator, Schematics).

Για τις εξόδους που έχουν δημιουργηθεί για την οδήγηση της οθόνης των τεσσάρων ψηφίων δεκαδικής απεικόνισης LED, παρατηρούμε στο σχηματικό διάγραμμα του κυκλώματος (Σχήμα 34) ότι συνδέονται με τους ακροδέκτες του κυκλώματος FPGA ;οπως δείχνει ο Πίνακας 6.



Σχήμα 34. Σήματα οδήγησης των ψηφίων δεκαδικής απεικόνιση (MAXimator Schematics).

Πίνακας 6. Αντιστοιχία σημάτων των ψηφίων δεκαδικής απεικόνισης με τους ακροδέκτες του κυκλώματος FPGA (MAXimator, User Guide).

Έξοδος	Σήμα Τυπωμένου Κυκλώματος	Ακροδέκτης Κυκλώματος FPGA
Digit_Segments[6]	D6	G16
Digit_Segments [5]	D5	G15
Digit_Segments [4]	D4	H16
Digit_Segments [3]	D3	H15
Digit_Segments [2]	D2	J16
Digit_Segments [1]	D1	J15
Digit_Segments [0]	D0	L16
ConfigBit[4]	D7	F16
Digit[0]	D8	E15
Digit[1]	D9	E16
Digit[2]	D10	D15
Digit[3]	D11	D16

Στην τελευταία στήλη του παραπάνω πίνακα βρίσκονται τα ονόματα των ακροδεκτών (pins) του κυκλώματος FPGA στα οποία πρέπει να αντιστοιχιστούν οι εισοδοί/έξοδοι που δημιουργήθηκαν για την εφαρμογή.

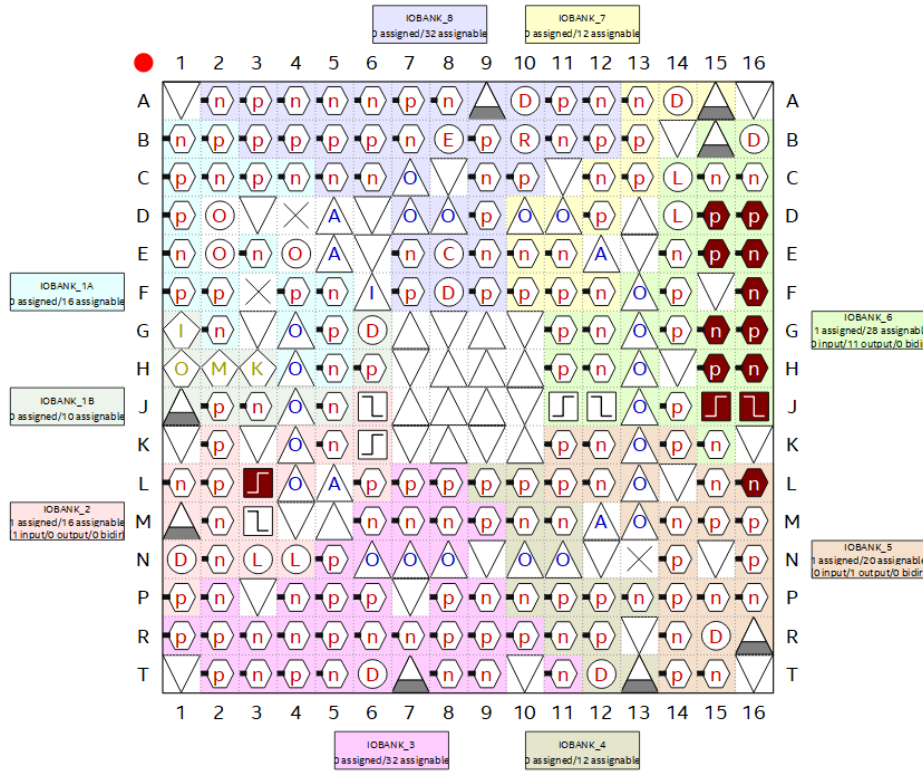
7.11 Ανάθεση των εισόδων-εξόδων της εφαρμογής στους ακροδέκτες του κυκλώματος FPGA.

Η ανάθεση των εξόδων της εφαρμογής σε συγκεκριμένους ακροδέκτες του κυκλώματος FPGA, πραγματοποιείται με την χρήση της διεπαφής **Pin Planner** που βρίσκεται στο μενού **Assignments**. Επιλέγουμε:

Assignments → Pin Planner

Άμεσα εκτελείται η διεπαφή Pin Planner όπου εμφανίζονται σχηματικά όλες οι ομάδες ακροδεκτών του κυκλώματος FPGA όπως απεικονίζεται στο Σχήμα 35.

Top View - Wire Bond MAX 10 - 10M08DAF256C8GES



Σχήμα 35. Διάγραμμα ακροδεκτών του κυκλώματος FPGA.

Επίσης στην εφαρμογή Pin Planner εμφανίζεται και ένας πίνακας που εμπεριέχει τις εισόδους/εξόδους της εφαρμογής στην στήλη **Node Name**. Στον πίνακα αυτό υπάρχει και η στήλη **Location** μέσω της οποίας πραγματοποιείται η ανάθεση των κατάλληλων ακροδεκτών του κυκλώματος FPGA, στις αντίστοιχες εισόδους/εξόδους που δημιουργήθηκαν για την εφαρμογή.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
clock_IN	Input	PIN_L3	2	B2_NO	PIN_L3	2.5V		12mA (default)			
ConfigBit[4]	Output	PIN_F16	6	B6_NO	PIN_F16	2.5V		12mA (default)	2 (default)		
Digit[3]	Output	PIN_D16	6	B6_NO	PIN_D16	2.5V		12mA (default)	2 (default)		
Digit[2]	Output	PIN_D15	6	B6_NO	PIN_D15	2.5V		12mA (default)	2 (default)		
Digit[1]	Output	PIN_E16	6	B6_NO	PIN_E16	2.5V		12mA (default)	2 (default)		
Digit[0]	Output	PIN_E15	6	B6_NO	PIN_E15	2.5V		12mA (default)	2 (default)		
Digit_Segments[6]	Output	PIN_L16	5	B5_NO	PIN_L16	2.5V		12mA (default)	2 (default)		
Digit_Segments[5]	Output	PIN_J15	6	B6_NO	PIN_J15	2.5V		12mA (default)	2 (default)		
Digit_Segments[4]	Output	PIN_J16	6	B6_NO	PIN_J16	2.5V		12mA (default)	2 (default)		
Digit_Segments[3]	Output	PIN_H15	6	B6_NO	PIN_H15	2.5V		12mA (default)	2 (default)		
Digit_Segments[2]	Output	PIN_H16	6	B6_NO	PIN_H16	2.5V		12mA (default)	2 (default)		
Digit_Segments[1]	Output	PIN_G15	6	B6_NO	PIN_G15	2.5V		12mA (default)	2 (default)		
Digit_Segments[0]	Output	PIN_G16	6	B6_NO	PIN_G16	2.5V		12mA (default)	2 (default)		

Έπειτα από την ορθή συμπλήρωση του πίνακα ανάθεσης των ακροδεκτών επιλέγουμε στο μενού της διεπαφής Pin Planner:

Processing → Start I/O Assignment Analysis

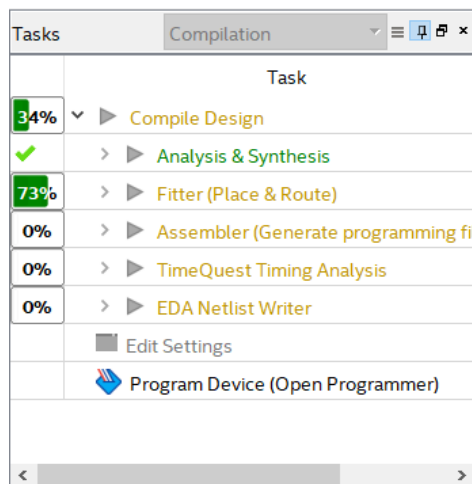
Με την παραπάνω επιλογή, έχει ολοκληρωθεί και η ανάθεση των εισόδων/εξόδων της εφαρμογής, στους κατάλληλους ακροδέκτες του κυκλώματος FPGA.

7.12 Compilation της εφαρμογής.


Η εφαρμογή που δημιουργήσαμε είναι έτοιμη για **compilation**. Κατά την διαδικασία αυτή μεταφράζεται ο κώδικας περιγραφής κυκλωμάτων VHDL, πραγματοποιείται η ανάλυση και η σύνθεση του κυκλώματος και δρομολογούνται οι συνδέσεις των κυκλωμάτων για την τελική διαμόρφωση του κυκλώματος FPGA. Για την έναρξη της διαδικασίας επιλέγουμε:

Processing → Start Compilation

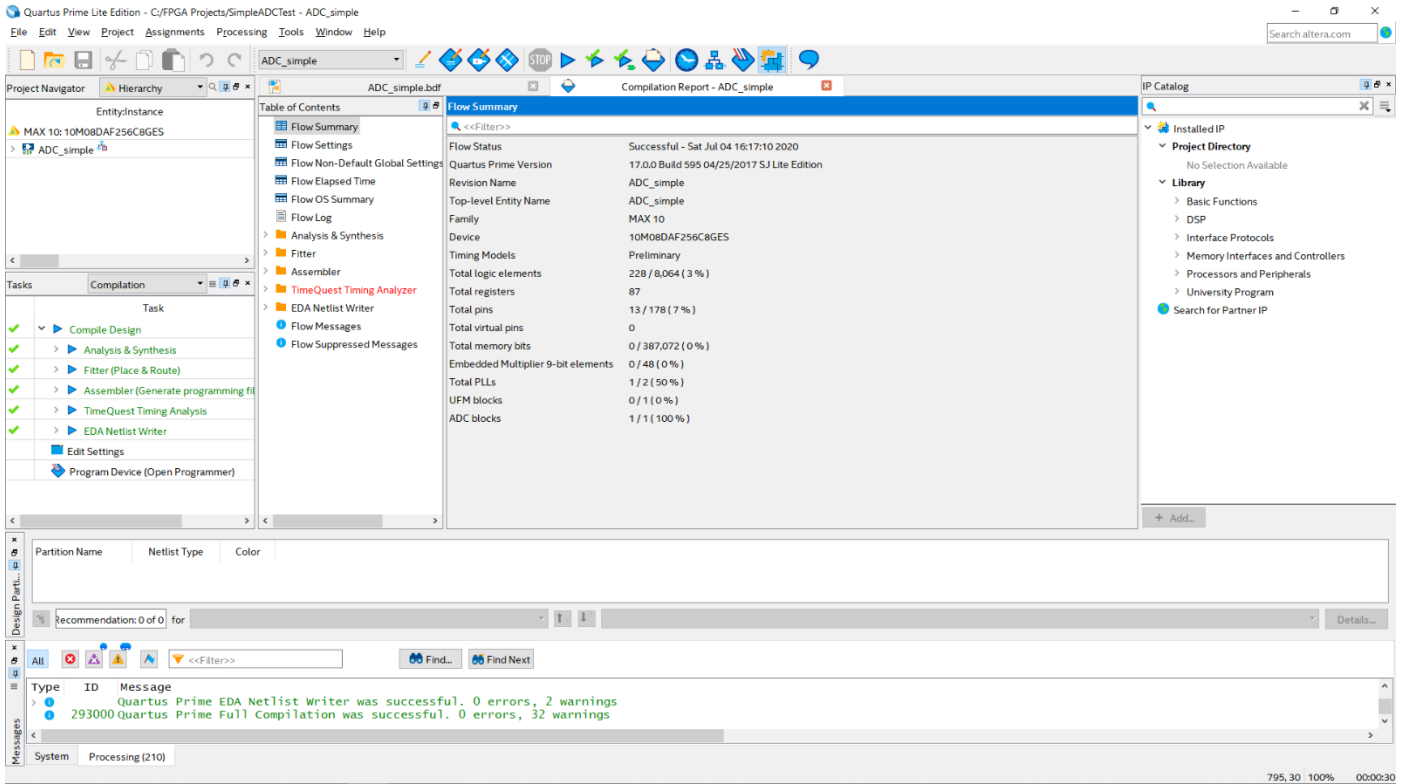
Κατά την διάρκεια εκτέλεσης της διαδικασίας, η πρόοδος εμφανίζεται στα δεξιά της οθόνης.



Με την ολοκλήρωση της διαδικασίας compilation, εμφανίζεται η αναφορά στην οποία παρουσιάζονται ορισμένα χρήσιμα στοιχεία σχετικά με την κατανάλωση των πόρων του υλικού για την υλοποίηση της εφαρμογής.

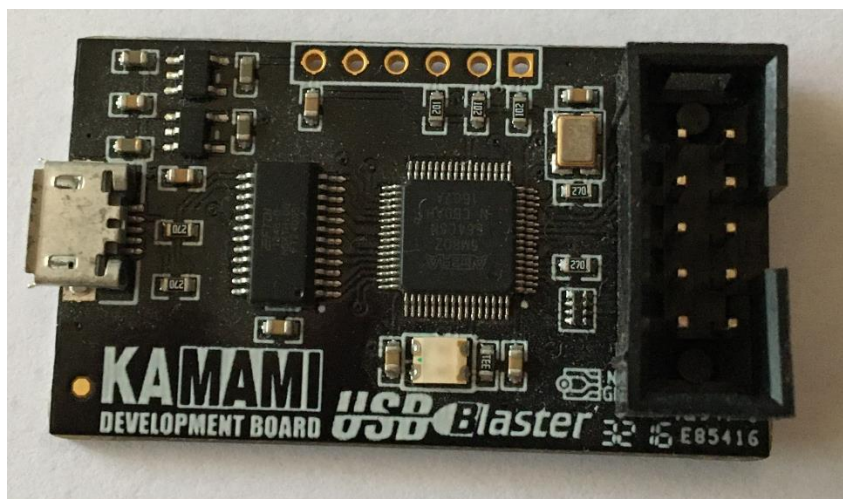
Flow Summary	
 <<Filter>>	
Flow Status	Successful - Sat Jul 04 16:41:34 2020
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	ADC_simple
Top-level Entity Name	ADC_simple
Family	MAX 10
Device	10M08DAF256C8GES
Timing Models	Preliminary
Total logic elements	228 / 8,064 (3 %)
Total registers	87
Total pins	13 / 178 (7 %)
Total virtual pins	0
Total memory bits	0 / 387,072 (0 %)
Embedded Multiplier 9-bit elements	0 / 48 (0 %)
Total PLLs	1 / 2 (50 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	1 / 1 (100 %)

Στην παραπάνω αναφορά της εφαρμογής που δημιουργήσαμε, εμφανίζεται η χρήση 228 λογικών στοιχείων από το σύνολο των 8064 που υπάρχουν στο συγκεκριμένο κύκλωμα FPGA και αντιστοιχούν σε ποσοστό 3%. Επίσης εμφανίζονται ο αριθμός των ακροδεκτών (13), καθώς και ο αριθμός των καταχωρητών (87), που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Στα τελευταία στοιχεία της αναφοράς συμπεριλαμβάνονται και η χρήση του ADC block καθώς και η χρήση του ενός εκ των δύο κυκλωμάτων χρονισμού PLLs που διαθέτει το συγκεκριμένο κύκλωμα FPGA. Στην περίπτωση που η διαδικασία δεν ολοκληρωθεί επιτυχώς, το περιβάλλον ανάπτυξης Quartus Prime ενημερώνει τον χρήστη για τα σφάλματα που εντοπίστηκαν.



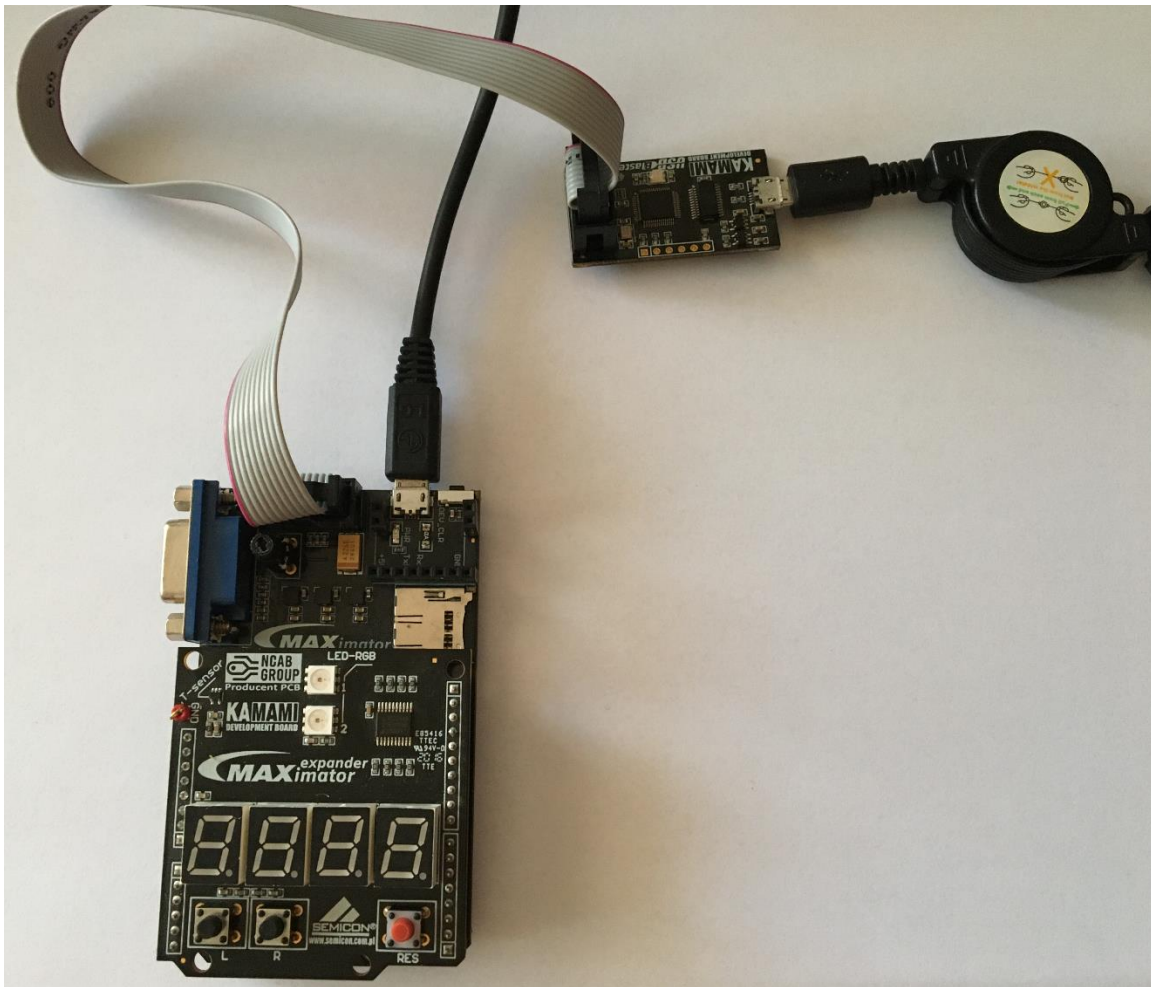
7.13 Προγραμματισμός και διαμόρφωση του κυκλώματος FPGA με τον κώδικα της εφαρμογής.

Κατά την ολοκλήρωση της διαδικασίας compilation, δημιουργείται το αρχείο διαμόρφωσης του κυκλώματος FPGA σύμφωνα με τον κώδικα περιγραφής κυκλωμάτων. Η εισαγωγή του αρχείου διαμόρφωσης στο κύκλωμα FPGA πραγματοποιείται με την βοήθεια ειδικού κυκλώματος προγραμματιστή **USB Blaster** Εικόνα 9.



Εικόνα 1. Τυπωμένο κύκλωμα του προγραμματιστή κυκλωμάτων FPGA, USB Blaster.

Ο προγραμματιστής USB Blaster μεταφέρει το αρχείο διαμόρφωσης που δημιουργείται από το περιβάλλον ανάπτυξης εφαρμογών Quartus Prime στο κύκλωμα FPGA. Η σύνδεση πραγματοποιείται σειριακά με σύνδεσμο USB ενώ η σύνδεση για τον προγραμματισμό του κυκλώματος FPGA πραγματοποιείται με σύνδεσμο JTAG (Εικόνα 10).

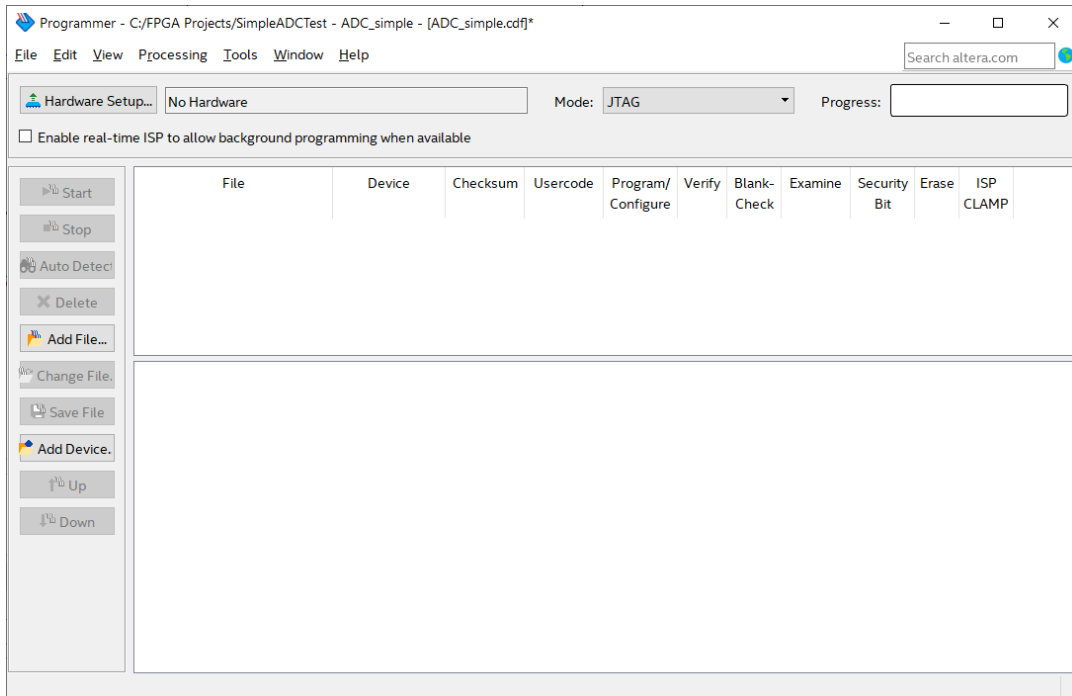


Εικόνα 2. Η σύνδεση του προγραμματιστή USB Blaster με το αναπτυξιακό κύκλωμα MAXimator.

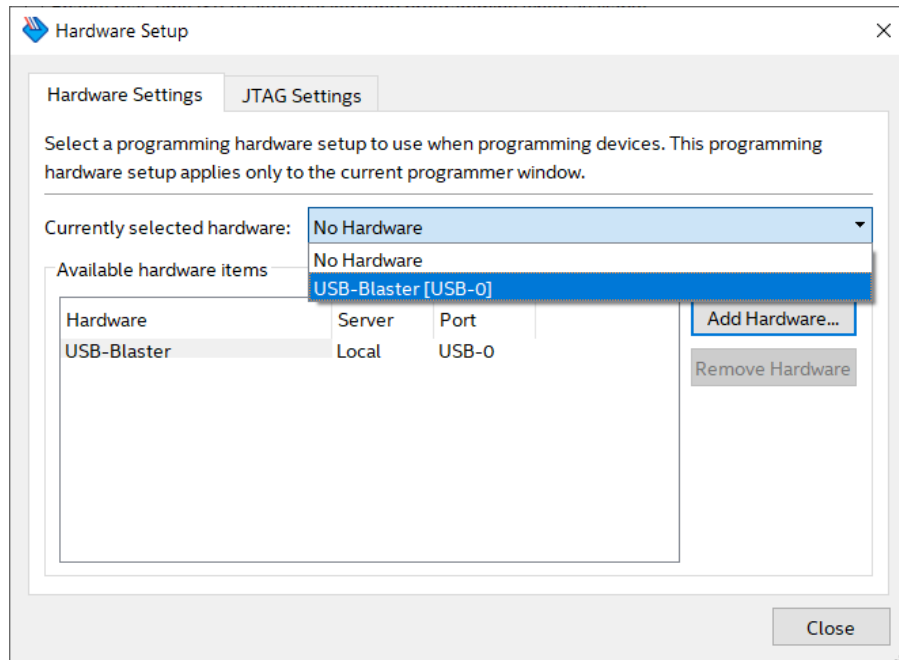
Για τον προγραμματισμό του κυκλώματος FPGA επιλέγουμε:

Tools → Programmer

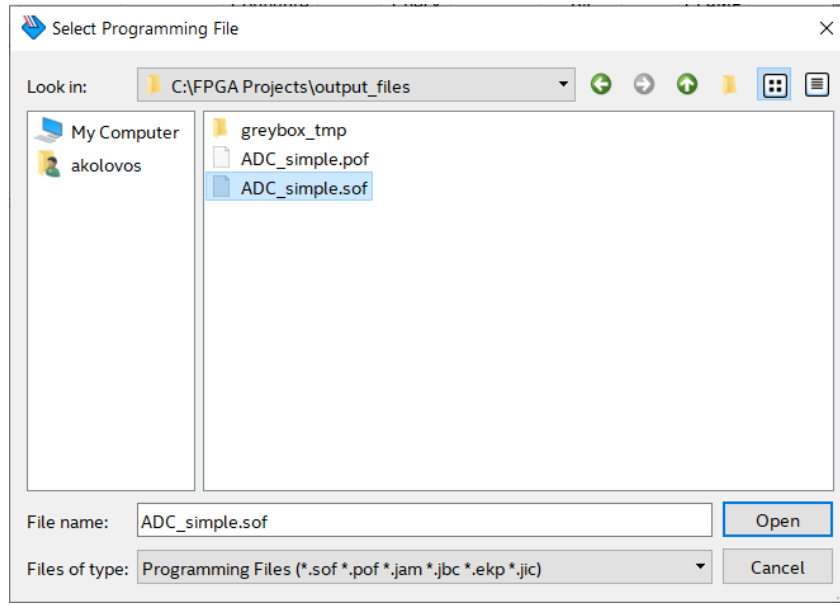
Στην διεπαφή που εμφανίζεται πρέπει πρώτα να δηλωθεί η συσκευή προγραμματισμού καθώς στην αρχική κατάσταση δεν υπάρχει δηλωμένο υλικό προγραμματισμού (No Hardware)..



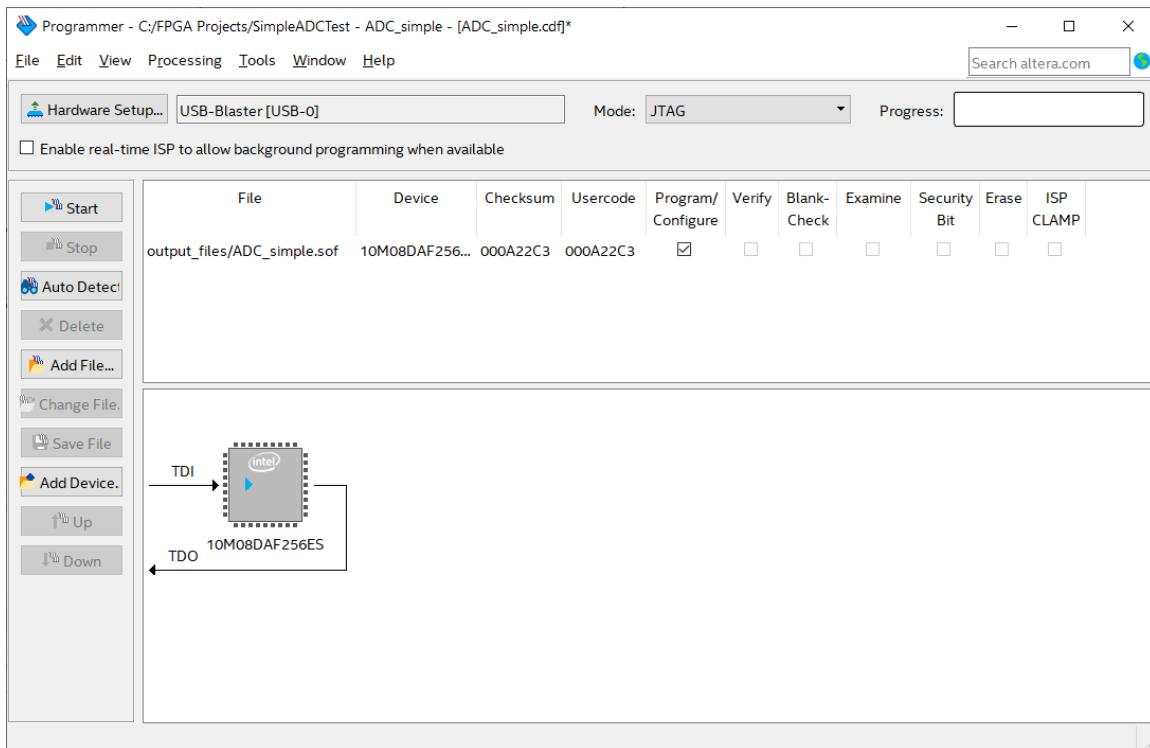
Επιλέγουμε Hardware Setup και ενεργοποιούμε τη συσκευή USB Blaster.



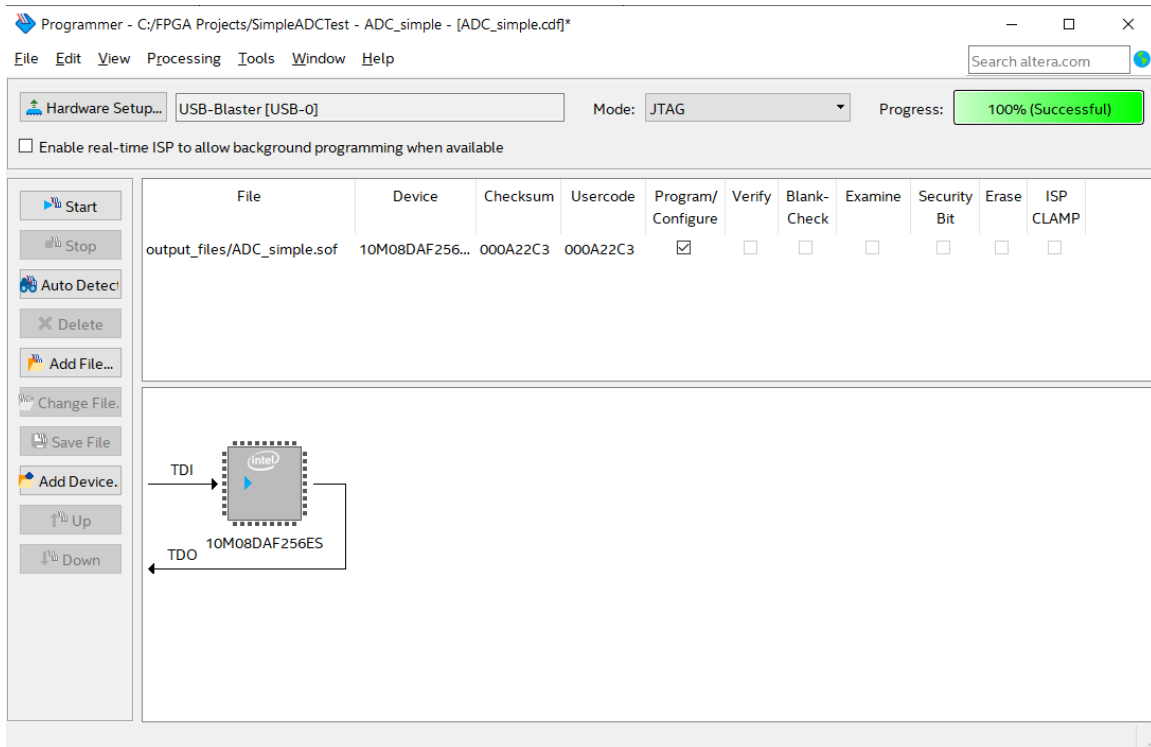
Στη συνέχεια ορίζουμε το αρχείο διαμόρφωσης για την εφαρμογή. Επιλέγουμε **Add File** και στην διεπαφή επιλογής αρχείου, επιλέγουμε το αρχείο ADC_simple.sof που βρίσκεται στον φάκελο C:\FPGA Projects\output_files.



Το αρχείο προστίθεται και εμφανίζεται και η σχηματική παράσταση του κυκλώματος FPGA που έχουμε επιλέξει αρχικά.



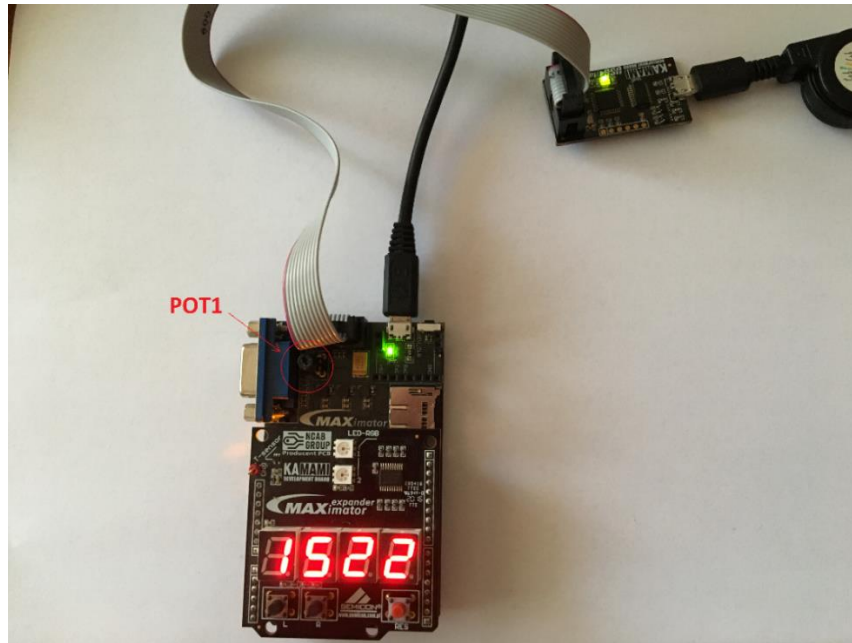
Επιλέγουμε **Start** και το αρχείο μεταφέρεται στο κύκλωμα FPGA.



Η διαμόρφωση που πραγματοποιείται με το αρχείο ADC_simple.sof, αποθηκεύεται σε μνήμη RAM και χάνεται με την απώλεια της ηλεκτρικής ενέργειας από το κύκλωμα FPGA. Για να παραμείνει η διαμόρφωση και κατά την απώλεια της ηλεκτρικής ενέργειας από το κύκλωμα FPGA, πρέπει να επιλεγεί το αρχείο ADC_simple.prof, το οποίο έχει δημιουργηθεί στον ίδιο φάκελο (C:\FPGA Projects\output_files) και έχει την δυνατότητα να αποθηκευτεί στην μνήμη Flash που παρέχει το κύκλωμα FPGA για τον χρήστη (UFM User Flash Memory). Στην περίπτωση αυτή το κύκλωμα FPGA συνεχίζει να διατηρεί την διαμόρφωση και κατά την απώλεια της ηλεκτρικής ενέργειας και εκτελεί την εφαρμογή άμεσα με την επαναφορά της.

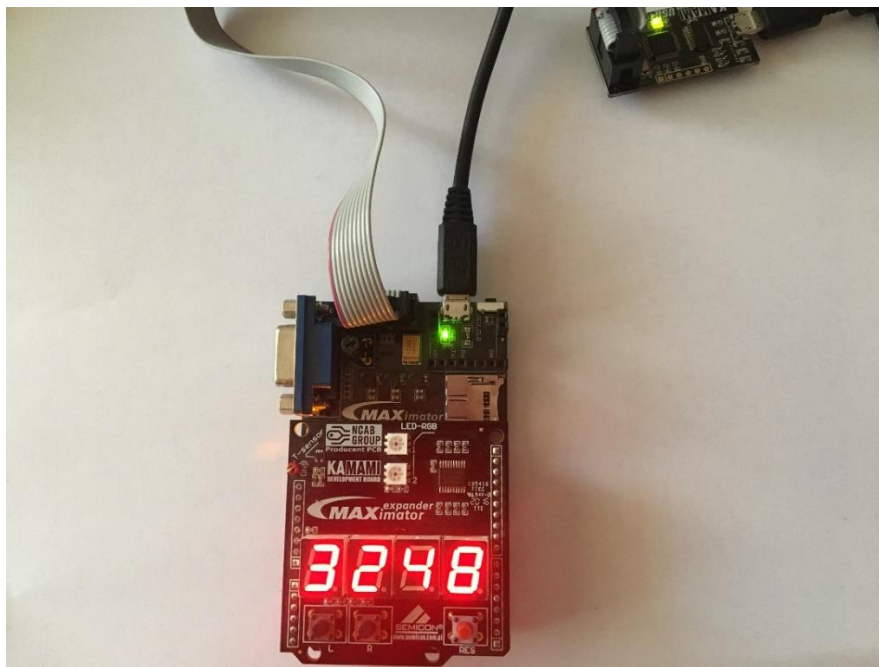
7.14 Εκτέλεση της εφαρμογής από το αναπτυξιακό τυπωμένο κύκλωμα MAXimator.

Με την ολοκλήρωση του προγραμματισμού του κυκλώματος FPGA με το αρχείο διαμόρφωσης, άμεσα η εφαρμογή που δημιουργήσαμε εκτελείται από το κύκλωμα FPGA. Στην παρακάτω εικόνα ο δεκαδικός αριθμός 1522, είναι η δεκαδική αναπαράσταση της τάσης που δέχεται ο ακροδέκτης αναλογικής εισόδου **ANAIN** του κυκλώματος FPGA και όπως προαναφέρθηκε η τάση αυτή καθορίζεται από την θέση του ποτενσιόμετρου POT1 που επισημαίνεται με κόκκινο κύκλο (Εικόνα 11).



Εικόνα 9: Το αναπτυξιακό κύκλωμα με την εφαρμογή σε λειτουργία.

Με την περιστροφή του ποτενσιόμετρου POT1, μπορούμε να αλλάξουμε τα επίπεδα τάσης του ακροδέκτη αναλογικής εισόδου ANAIN και κατά συνέπεια να αλλάξει και η ένδειξη της δεκαδικής απεικόνισης στην οθόνη LED του κυκλώματος (Εικόνα 12).



Εικόνα 10: Αλλαγή της δεκαδικής ένδειξης με περιστροφή του ποτενσιόμετρου POT1.

8 Συμπεράσματα

Η ρύθμιση και η χρήση του αναλογικού/ψηφιακού μετατροπέα που παρέχει η εταιρεία Intel στο κύκλωμα FPGA **10M08DAF256C8G** είναι πολύ απλή και οι τέσσερις διαφορετικές διαμορφώσεις του υλικού μπορούν να καλύψουν μεγάλο εύρος αναλογικών/ψηφιακών εφαρμογών.

Η διαμόρφωση της **διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM**, μπορεί να χρησιμοποιηθεί σε εφαρμογές που απαιτούν εγγραφή των αποτελεσμάτων σε μνήμη RAM με σκοπό την μετέπειτα ψηφιακή επεξεργασία τους.

Η διαμόρφωση της **διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM και ανίχνευση παραβίασης ορίων**, παρέχει την δυνατότητα στην εφαρμογή, την εγγραφή των αποτελεσμάτων σε προσωρινή μνήμη RAM αλλά και την δημιουργία σημάτων ανίχνευσης της υπέρβασης προκαθορισμένων ορίων των ψηφιακών αποτελεσμάτων.

Η διαμόρφωση της **διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε εξωτερική μνήμη RAM**, είναι χρήσιμη για εφαρμογές που απαιτείται εγγραφή μεγάλου όγκου αποτελεσμάτων αναλογικής/ψηφιακής μετατροπής. Ο όγκος των δεδομένων μπορεί να ξεπερνά τις δυνατότητες αποθήκευσης σε εσωτερική μνήμη του κυκλώματος FPGA. Με την διαμόρφωση αυτή παρέχεται η δυνατότητα ελέγχου της ροής των αποτελεσμάτων αναλογικής/ψηφιακής μετατροπής σε εξωτερική μνήμη RAM.

Τέλος, η διαμόρφωση της **χρήσης μόνο του πυρήνα ADC control**, αποτελεί την πιο ευέλικτη διαμόρφωση για την υλοποίηση εφαρμογών που απαιτούν αναλογική/ψηφιακή μετατροπή. Η συγκεκριμένη διαμόρφωση, χρησιμοποιήθηκε στην παρούσα εργασία για την υλοποίηση της εφαρμογής αναλογικών/ψηφιακών μετατροπών και την δεκαδική απεικόνιση του αποτελέσματος σε πραγματικό χρόνο.

Ο αναλογικός ψηφιακός μετατροπέας των κυκλωμάτων FPGA της οικογένειας MAX10 αποτελεί μια καλή λύση για την υλοποίηση εφαρμογών που απαιτούν δειγματοληψία μέχρι 1Msps. Αυτό είναι και το βασικό μειονέκτημα των αναλογικών/ψηφιακών μετατροπέων της οικογένειας MAX10, ο σχετικά μικρός ρυθμός

δειγματοληψίας τους, που αποτελεί περιοριστικό παράγοντα για την χρήση τους σε εφαρμογές που απαιτούν μεγάλους ρυθμούς δειγματοληψίας (π.χ. ψηφιακός παλμογράφος). Παρόλα αυτά μπορεί να χρησιμοποιηθεί σε πλήθος άλλων εφαρμογών, όπως έλεγχο περιβαλλοντικών συνθηκών, καταγραφές μετρήσεων φυσικών παραμέτρων, ψηφιακό φιλτράρισμα ήχου (φίλτρα DSP) κ.α.

Το γεγονός ότι βρίσκεται ενσωματωμένος μέσα στο κύκλωμα FPGA, αποτελεί συγκριτικό πλεονέκτημα σε σχέση με τα διακεκριμένα ολοκληρωμένα κυκλώματα αναλογικής/ψηφιακής μετατροπής. Με τον τρόπο αυτό συνδυάζεται η δυνατότητα της αναλογικής/ψηφιακής μετατροπής, με την ταχύτατη επεξεργαστική ισχύ που παρέχουν τα κυκλώματα FPGA. Έτσι η ανάπτυξη εφαρμογών που απαιτούν αναλογική/ψηφιακή μετατροπή, μπορεί να πραγματοποιηθεί γρήγορα και οικονομικά σε συνδυασμό και με την ευελιξία της εύκολης επαναδιαμόρφωσης που διαθέτουν τα κυκλώματα FPGA.

9 ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ

9.1 ΣΧΗΜΑΤΑ

Σχήμα 1. Διαγραμματική απεικόνιση Λογικού Στοιχείου (Logic Element LE) κυκλώματος FPGA	8
Σχήμα 2. Σχηματική απεικόνιση ενός Logic Array Block (LAB).....	9
Σχήμα 3. Σχηματική απεικόνιση της δομής ενός κυκλώματος FPGA.....	10
Σχήμα 4. Κώδικας περιγραφής κυκλωμάτων VHDL	11
Σχήμα 5. Κωδικοποίηση ονοματολογίας των κυκλωμάτων της οικογένειας MAX 10....	13
Σχήμα 6. Το κύκλωμα FPGA 10M08DAF256C8G	14
Σχήμα 7: Βασικά στάδια σχεδίασης εφαρμογής για κύκλωμα FPGA.....	19
Σχήμα 8. Η θέση του αναλογικού/ψηφιακού μετατροπέα (ADC1) μέσα στο κύκλωμα FPGA	23
Σχήμα 9. Διάγραμμα του ADC Block.....	24
Σχήμα 10. Διάγραμμα του δεκαεξαδικού αποτελέσματος της αναλογικής/ψηφιακής μετατροπής σε συνάρτηση με την τάση εισόδου.....	25
Σχήμα 11. Ενσωματωμένος διαιρέτης τάσης εισόδου	26
Σχήμα 12. Η διασύνδεση του λογισμικού εφαρμογής με το ADC Block μέσω του Altera Modular ADC IP core.....	27
Σχήμα 13. Σχηματική απεικόνιση του μικροπυρήνα ADC control	28
Σχήμα 14. Διάγραμμα χρονισμού αναλογικών/ψηφιακών μετατροπών του ADC Block.....	29
Σχήμα 15. Σχηματική απεικόνιση του ADC sequencer microcore.....	31
Σχήμα 16. Σχηματική απεικόνιση του ADC sample storage core	32
Σχήμα 17. Σχηματική απεικόνιση του ADC threshold detection core	33
Σχήμα 18. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM.....	34
Σχήμα 19. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε ενσωματωμένη μνήμη RAM και ανίχνευση παραβίασης ορίων.....	35
Σχήμα 20. Σχηματική απεικόνιση της διαμόρφωσης διαδοχικής μετατροπής αναλογικών καναλιών με εγγραφή του αποτελέσματος σε εξωτερική μνήμη RAM.....	35
Σχήμα 21. Σχηματική απεικόνιση της διαμόρφωσης με χρήση μόνο το ADC control microcore.	36
Σχήμα 22. Διαγραμματική απεικόνιση της εφαρμογής αναλογικής/ψηφιακής μετατροπής.	46
Σχήμα 23. Η διασύνδεση του κυκλώματος χρονισμού (MyPLL) με τον αναλογικό/ψηφιακό μετατροπέα MyADC.....	61
Σχήμα 24. Σχηματικό διάγραμμα του κυκλώματος του ποτενσιομέτρου P1 (10kΩ) (MAXimator, Schematics).....	63
Σχήμα 25. Η σύνδεση της επιλογής αναλογικού καναλιού ChannelSelection με τον διάλογο command channel[4..0]	66
Σχήμα 26. Χρονικό διάγραμμα αναλογικής/ψηφιακής μετατροπής.....	66
Σχήμα 27. Η σύνδεση των ConfigurationBits στα αντίστοιχα σήματα του MyADC.....	67

Σχήμα 28. Η σύνδεση του απαριθμητή DelayCounter και του μετατροπέα BinarytoBCD στα κυκλώματα της εφαρμογής.	82
Σχήμα 29. Σχηματικό διάγραμμα του κυκλώματος οδήγησης των ψηφίων δεκαδικής απεικόνισης LED επτά τομέων (MAXimator, Schematics).	83
Σχήμα 30. Η σύνδεση των τεσσάρων συμβόλων BCD_segments με τον δίαυλο BCD_result[15..0]	87
Σχήμα 31. Η σύνδεση του αποκωδικοποιητή BCD_select με τους οδηγούς BCD_segments	92
Σχήμα 32. Οι έξοδοι Digit[3..0] και η σύνδεσή τους με τα κυκλώματα της εφαρμογής. .	94
Σχήμα 33. Η κυκλωματική διάταξη του ταλαντωτή 10MHz και η σύνδεσή του με το κύκλωμα FPGA (MAXimator, Schematics).	95
Σχήμα 34. Σήματα οδήγησης των ψηφίων δεκαδικής απεικόνιση (MAXimator Schematics).	95
Σχήμα 35. Διάγραμμα ακροδεκτών του κυκλώματος FPGA.	97

9.2 ΕΙΚΟΝΕΣ

Εικόνα 1: Δύο μέλη της οικογένειας κυκλωμάτων FPGA MAX 10 σε φυσικές διαστάσεις	15
Εικόνα 2: Τυπωμένο αναπτυξιακό κύκλωμα MAXimator	16
Εικόνα 3: Τυπωμένο κύκλωμα διεπαφών MAXimator expander	17
Εικόνα 4: Το συνολικό αναπτυξιακό τυπωμένο κύκλωμα.	18
Εικόνα 5: Λογότυπο έναρξης της εφαρμογής Intel Quartus Prime Development Suite 17.0.	37
Εικόνα 6: Η αρχική διεπαφή της εφαρμογής :Quartus Prime.	38
Εικόνα 7: Τυπωμένο κύκλωμα διεπαφών MAXimator expander με την οθόνη LED τεσσάρων δεκαδικών ψηφίων.	46
Εικόνα 8: Το λογότυπο έναρξης της εφαρμογής Qsys.	48
Εικόνα 9: Το αναπτυξιακό κύκλωμα με την εφαρμογή σε λειτουργία.	106
Εικόνα 10: Αλλαγή της δεκαδικής ένδειξης με περιστροφή του ποτενσιόμετρου POT1.	106

9.3 ΠΙΝΑΚΕΣ

Πίνακας 1: Υπολογισμός των χρονικών περιόδων t1, t2 και t3.	30
Πίνακας 2. Η δυαδική κωδικοποίηση των αναλογικών καναλιών.	62
Πίνακας 3. Κωδικοποίηση BCD.	68
Πίνακας 4. Παράδειγμα μετατροπής του δυαδικού αριθμού 12-bits 1101111000 σε δεκαδικό 3586.	69
Πίνακας 5. Αντιστοιχία των σημάτων διαύλου segment[6..0] με τους τομείς LED ενός ψηφίου δεκαδικής απεικόνισης.	85
Πίνακας 6. Αντιστοιχία σημάτων των ψηφίων δεκαδικής απεικόνισης με τους ακροδέκτες του κυκλώματος FPGA (MAXimator, User Guide).	96

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ιωάννης Α. Καλόμοιρος. *Εισαγωγή στη γλώσσα VHDL*. ΤΕΙ Σερρών Τμήμα Πληροφορικής και Επικοινωνιών, έκδοση 1.2, 2012.

Altera Corporation. *MAX 10 FPGA Device Architecture*. Version 2016.05.02, May 2016.

Intel. *MAX 10 FPGA Device Overview*. Version 2017.12.15, December 2017α.

Intel. *MAX 10 Analog to Digital Converter User Guide*. Version 2017.12.15, December 2017β.

MAXimator Schematics, Διαθέσιμο στη διεύθυνση: <https://maximator-fpga.org/wp-content/uploads/2018/06/MAXimator-board.pdf>. [πρόσβαση 3/09/2020].

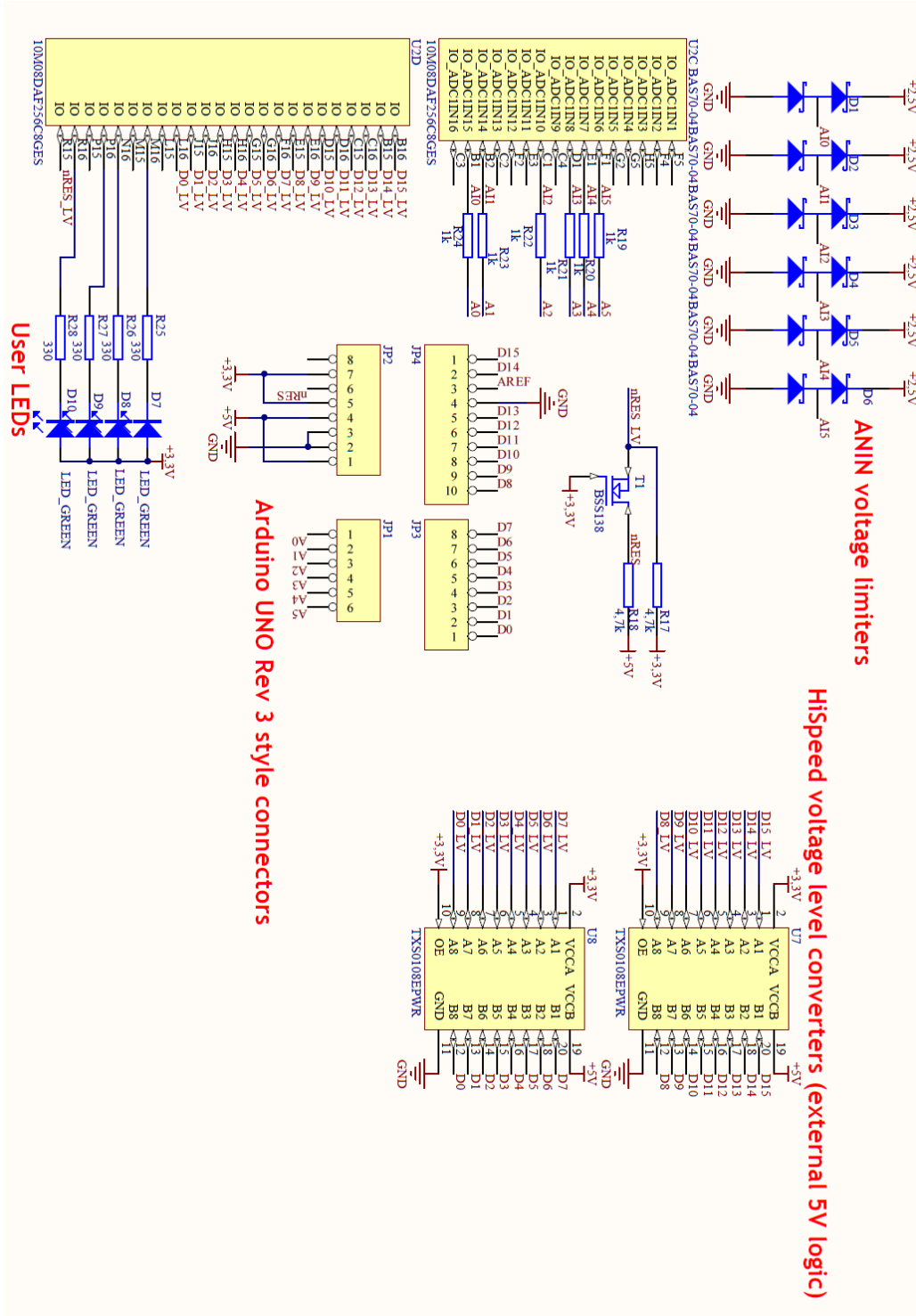
MAXimator User Guide, Διαθέσιμο στη διεύθυνση: https://maximator-fpga.org/wp-content/uploads/2016/03/MAXimator_user_manual.pdf. [πρόσβαση 03/09/2020].

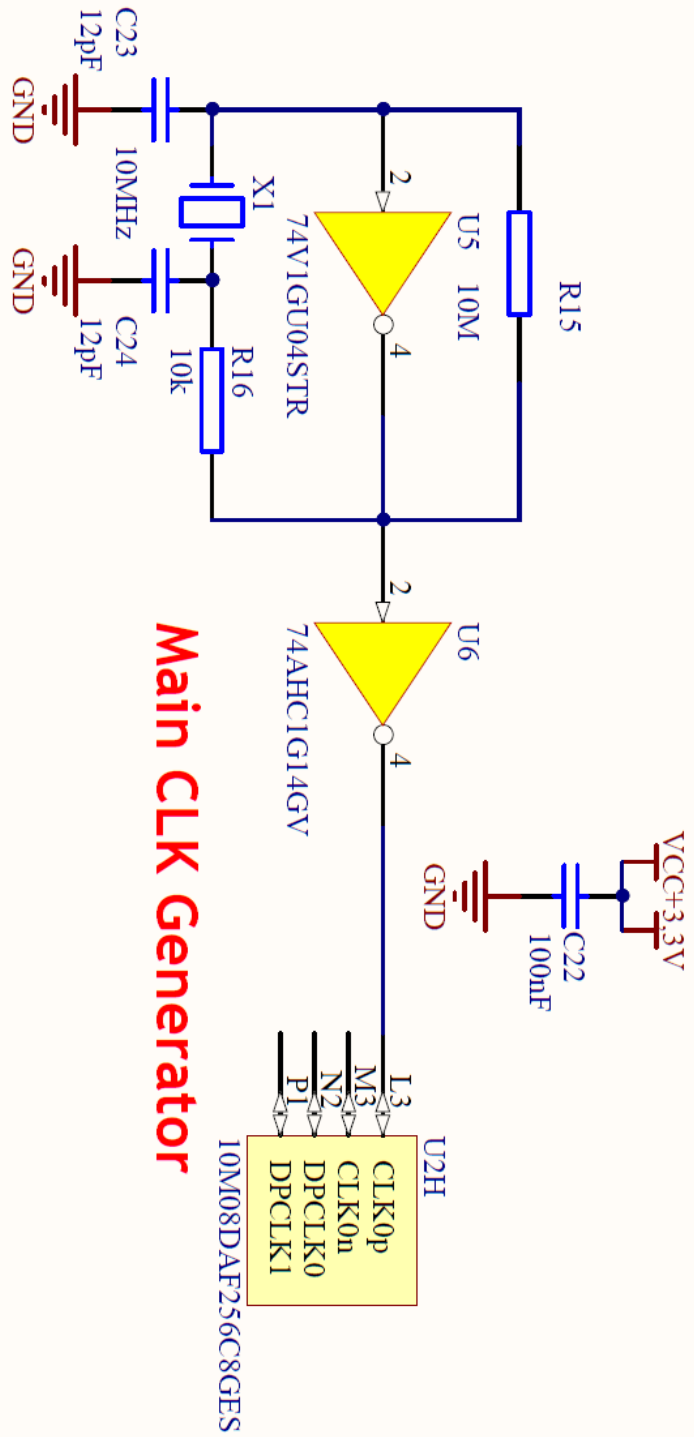
Wikipedia. Field-programmable gate array, Διαθέσιμο στην διεύθυνση: https://en.wikipedia.org/wiki/Field-programmable_gate_array. [πρόσβαση 03/09/2020].

Wikipedia. Double dabble algorithm Διαθέσιμο στην διεύθυνση: https://en.wikipedia.org/wiki/Double_dabble [πρόσβαση 03/09/2020].

ΠΑΡΑΡΤΗΜΑ Β

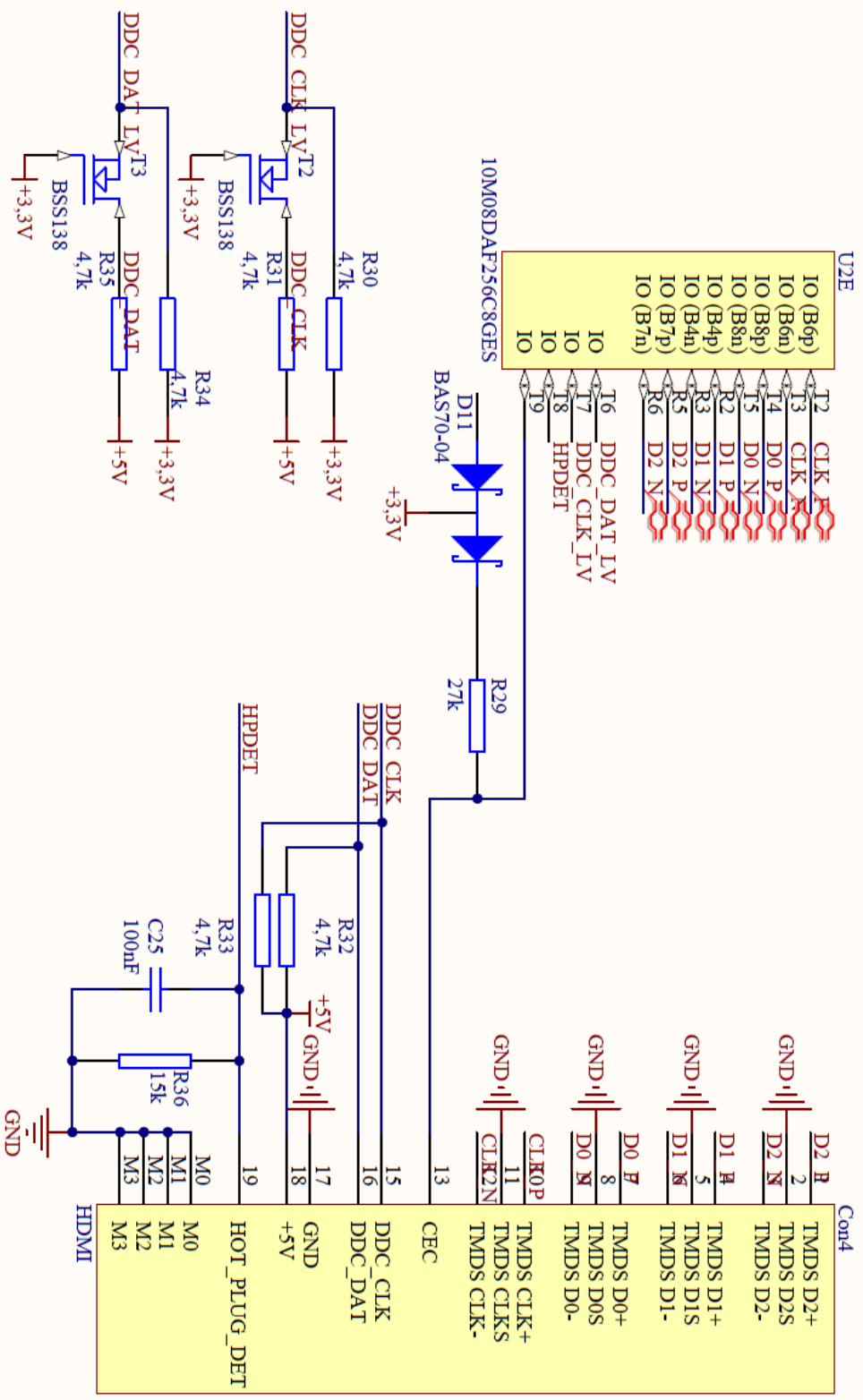
Σχηματικά διαγράμματα του τυπωμένου αναπτυξιακού κυκλώματος
MAXimator.



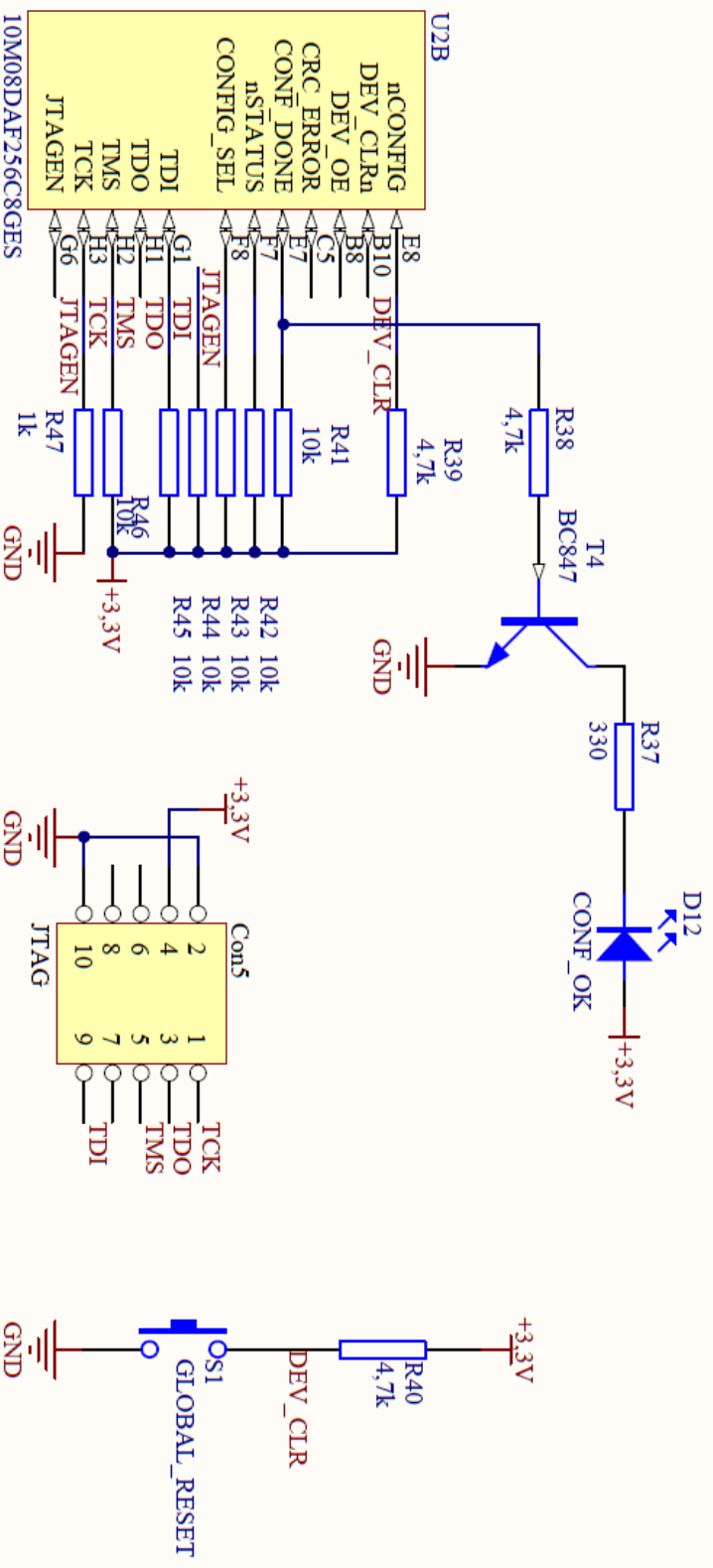


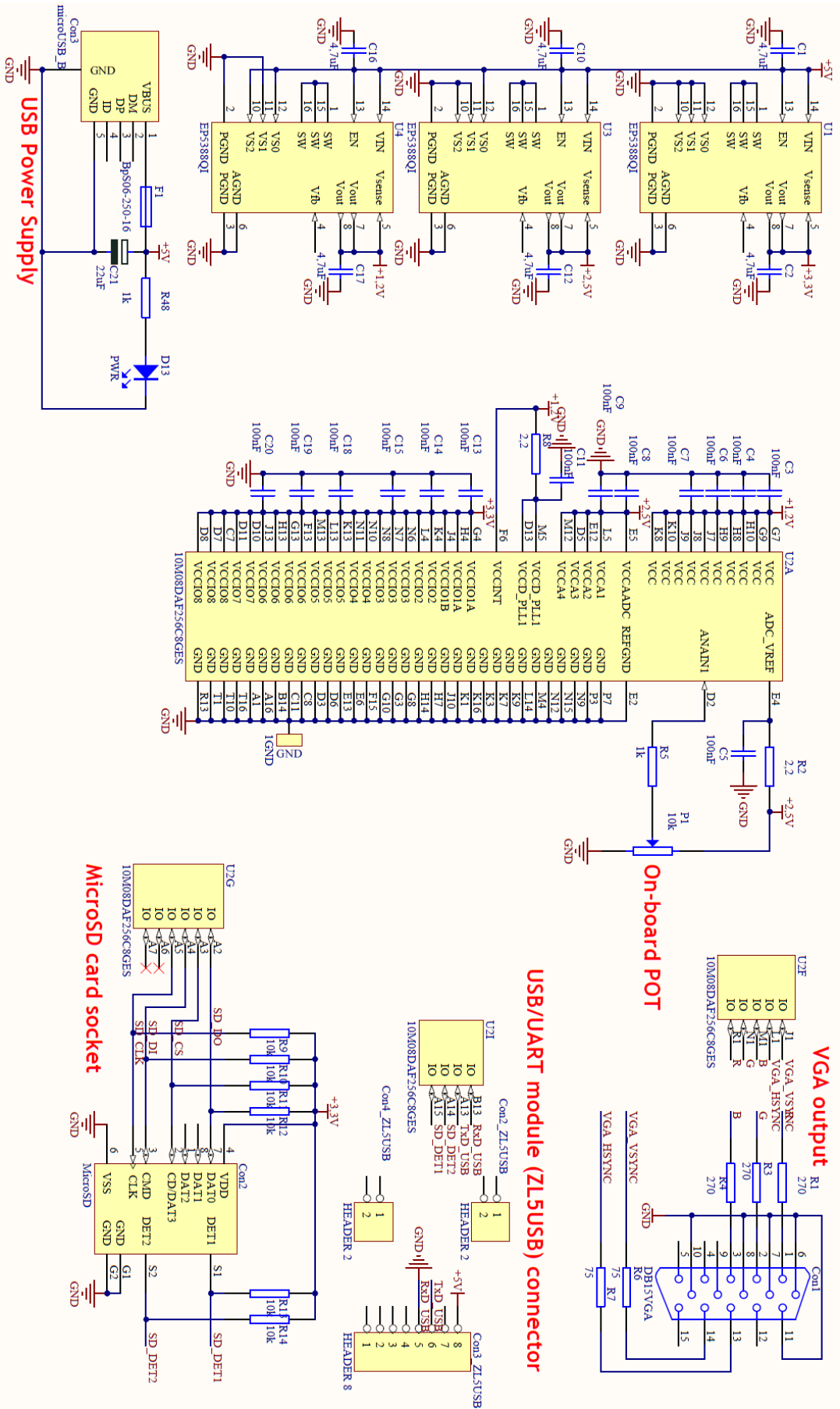
Main CLK Generator

HDMI/CEC/DDC connector



JTAG connector + configuration signalling + nGRES





USB Power Supply

On-board POT

VGA output

USB/UART module (ZL5USB) connector

MicroSD card socket

ΠΑΡΑΡΤΗΜΑ Γ

Σχηματικό διάγραμμα του κυκλώματος διεπαφών MAXimator expander.

