# ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

## ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

### ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ ΚΑΙ ΥΛΙΚΟΥ

**Διπλωματική Εργασία**

***Σχεδίαση ενός μοντέλου IP core του μικροελεγκτή AVR AT90S8515 σε VHDL***

**Θεοδωρόπουλος Δημήτρης**

Επιβλέπων καθηγητής: Καθηγητής Απόστολος Δόλλας

Εξεταστική Επιτροπή: Καθηγητής Α. Δόλλας
Αν. Καθηγητής Κ. Καλαϊτζάκης
Αν. Καθηγητής Δ. Πνευματικάτος

Ιούλιος 2003
Χανιά

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Απόστολο Δόλλα για τις πολύτιμες συμβουλές του που με βοήθησαν στην εκπόνηση αυτής της διπλωματικής εργασίας, τους αναπληρωτές καθηγητές κ. Διονύσιο Πνευματικάτο και κ. Κωνσταντίνο Καλαϊτζάκη που βρίσκονται στην εξεταστική επιτροπή και τον κ. Μάρκο Κυμιωνή για τη βοήθειά του.

Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδερφό μου που μου συμπαραστάθηκαν στην προσπάθεια αυτή, καθώς και όλους τους συμφοιτητές και φίλους μου που με βοήθησαν πολύ με τις συμβουλές τους και τις προτάσεις τους πάνω σε σημαντικές αποφάσεις.

Με εκτίμηση,

Θεοδωρόπουλος Δημήτρης

If at first you don’t succeed,
redefine success.

# Περιεχόμενα.

**1 Εισαγωγή. ..... 6**

**2 Σχετική έρευνα. ..... 9**

*2.1 Κατηγορίες IP cores. ..... 9*

*2.2 Μοντέλα IP cores μικροελεκτών και μικροεπεξεργαστών που υπάρχουν. ..... 10*

2.2.1 Μοντέλα μικροεπεξεργαστών. ..... 11

2.2.2 Μοντέλα IP cores μικροελεγκτών. ..... 12

*2.3 Μοντέλο IP core της οικογένειας μικροελεγκτών AVR. ..... 13*

*2.4 Εφαρμογές που χρησιμοποιούν μικροελεγκτές και αναδιατασσόμενη λογική. ..... 14*

**3 Ο μικροελεγκτής AT90S8515 της Atmel. ..... 24**

*3.1 Γενική περιγραφή του μικροελεγκτή. ..... 24*

*3.2 Αρχιτεκτονική του μικροελεγκτή. ..... 26*

3.2.1 Κατανομή της μνήμης μέσα στον μικροελεγκτή. ..... 27

3.2.2 Τρόποι διευθυνσιοδότησης. ..... 27

3.2.3 Τα interrupts του μικροελεγκτή. ..... 30

*3.3 Τα modules του μικροελεγκτή AT90S8515. ..... 32*

*3.4 Υποστηριζόμενες εντολές ..... 33*

**4 Το μοντέλο VHDL του AT90S8515 της Atmel. ..... 38**

*4.1 Γενική εικόνα του μοντέλου. ..... 39*

*4.2 Επιμέρους κομμάτια της σχεδίασης. ..... 41*

4.2.1 Μονάδα αριθμητικών και λογικών πράξεων (ALU). ..... 41

4.2.2 Μνήμες. ..... 43

4.2.3 IO pins. ..... 53

*4.3 Λειτουργία του UART. ..... 57*

4.3.1 Γεννήτρια κατάλληλου ρολογιού. ..... 57

4.3.2 UART Receiver. ..... 58

4.3.3 UART Transmitter. ..... 59

4.3.4 Αρχιτεκτονική του UART. ..... 59

*4.4 Κομμάτια του AT90S8515 που ΔΕΝ έχουν σχεδιαστεί. ..... 61*

*4.5 Ανάκληση, αποκωδικοποίηση και εκτέλεση των εντολών. ..... 63*

4.5.1 Ανάκληση εντολών. ..... 63

4.5.2 Αποκωδικοποίηση και εκτέλεση εντολών. ..... 66

*4.6 Λειτουργία των interrupts. ..... 71*

*4.7 Instruction Receiver. ... 74*

**5 Απεικόνιση του μοντέλου σε αναδιατασσόμενη λογική και βελτίωση 1ης έκδοσης. ... 78**

*5.1 Απεικόνιση της 1ης έκδοσης της σχεδίασης σε αναδιατασσόμενη λογική. ... 78*

*5.2 Βελτιώσεις και αλλαγές. ... 79*

5.2.1 Βελτιωμένη έκδοση της μνήμης εντολών. ... 80

5.2.2 Βελτιωμένη έκδοση της μνήμης δεδομένων. ... 81

5.2.3 Βελτιωμένη έκδοση της RF. ... 81

*5.3 Νέα απεικόνιση της σχεδίασης σε αναδιατασσόμενη λογική. ... 82*

*5.4 Απεικόνιση του μοντέλου IP core σε FPGAs της Altera. ... 83*

**6 Δοκιμή και πιστοποίηση του συστήματος. ... 86**

*6.1 Δοκιμές χωρίς τον instruction receiver. ... 87*

*6.2 Δοκιμές μαζί με τον instruction receiver. ... 99*

*6.3 Περιγραφή του κυκλώματος του συστήματος. ... 103*

**7 Συμπεράσματα και μελλοντικές επεκτάσεις. ... 107**

*7.1 Συμπεράσματα. ... 107*

*7.2 Μελλοντικές επεκτάσεις. ... 107*

*7.3 Επίλογος. ... 111*

**8 Παράρτημα Α – Ακρώνυμα. ... 113**

**9 Παράρτημα Β – Ιεραρχία του μοντέλου. ... 115**

**10 Παράρτημα Γ – Πίνακες σημάτων. ... 117**

**11 References. ... 131**

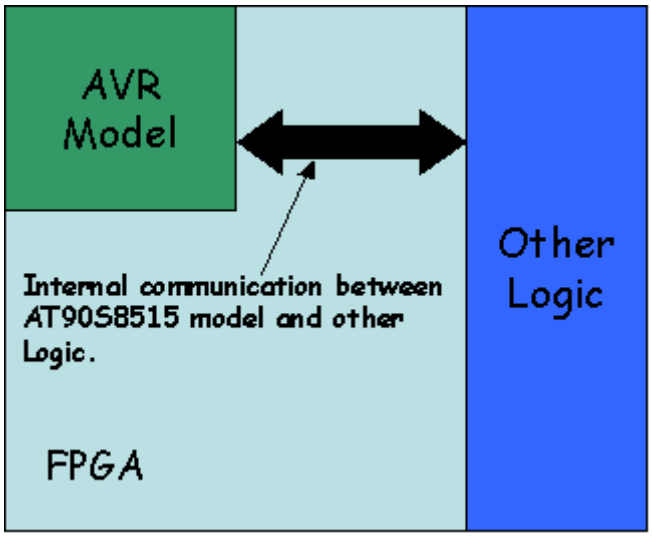
# 1 Εισαγωγή.

Είναι γεγονός ότι η σημερινή εποχή χαρακτηρίζεται από την τεχνολογική εξέλιξη και την πρόοδο της επιστήμης σε πολλούς τομείς. Σε καθημερινή βάση, χρησιμοποιούνται συσκευές που προσφέρονται απλόχερα από τη τεχνολογία, με σκοπό τη διευκόλυνση της ζωής και τη βελτίωση των συνθηκών διαβίωσης. Σημαντικός παράγοντας που συνέβαλλε στην πρόοδο αυτή, αποτελεί η εξέλιξη των τρανζίστορ, δηλαδή η μείωση του μεγέθους και της κατανάλωσής τους [7]. Το αποτέλεσμα είναι η σχεδίαση μικρότερων ολοκληρωμένων κυκλωμάτων, τα οποία συγχρόνως είναι πολύ πιο εξελιγμένα, έχοντας τη δυνατότητα να κάνουν πιο περίπλοκους υπολογισμούς, και παράλληλα χρειάζονται λιγότερη ενέργεια για τη λειτουργία τους.

Μία ομάδα ολοκληρωμένων κυκλωμάτων, η οποία γνώρισε ιδιαίτερη άνθιση τα τελευταία χρόνια, είναι αυτή των μικροελεγκτών. Πρόκειται για μία ειδική κατηγορία, όπου μέσα στο ίδιο το ολοκληρωμένο κύκλωμα περιλαμβάνονται πολλά διαφορετικά είδη πόρων (μετρητές, χρονιστές, δυνατότητα σύνδεσης με άλλον υπολογιστή, εσωτερικά και εξωτερικά interrupts), όπου ο χρήστης έχει τη δυνατότητα να τους προγραμματίσει να λειτουργούν σύμφωνα με τις απαιτήσεις της εκάστοτε εφαρμογής. Το γεγονός αυτό προσδίδει ιδιαίτερη ευελιξία στις αποφάσεις του μηχανικού, βοηθώντας τον να σχεδιάσει περίπλοκες εφαρμογές πολύ πιο εύκολα και γρήγορα. Δεν είναι τυχαίο ότι σήμερα οι μικροελεγκτές χρησιμοποιούνται στα περισσότερα τεχνολογικά προϊόντα, από τα φρένα των αυτοκινήτων (Anti Blocking System – ABS) μέχρι τις οικιακές συσκευές (κουζίνες, πλυντήρια, ψυγεία).

Σκοπός αυτής της διπλωματικής εργασίας είναι η σχεδίαση ενός επαναχρησιμοποιήσιμου μοντέλου (Intellectual Property – IP core) του μικροελεγκτή AT90S8515 της εταιρείας Atmel [29] με τη χρήση μιας γλώσσας περιγραφής υλικού και συγκεκριμένα της VHDL. Είναι αλήθεια ότι πολλές πρότυπες εφαρμογές (παράγραφος 2.3) χρησιμοποιούν κάποιον μικροελεγκτή μαζί με συσκευές αναδιατασσόμενης λογικής (Field Programmable Gate Arrays – FPGAs) [30, 31] και για να επικοινωνήσουν μεταξύ τους χρειάζεται είτε να κατασκευαστεί κάποια πλακέτα (Printed Circuit Board – PCB), είτε να χρησιμοποιηθούν καλώδια και breadboard. Και στις δύο περιπτώσεις ενδέχεται να υπάρξουν σφάλματα στις συνδέσεις, δυσκολεύοντας αρκετά την ανάπτυξη της σχεδίασης. Αντίθετα, αν χρησιμοποιηθεί μια σχετικά καινούρια FPGA, τότε

υπάρχει η δυνατότητα να ενσωματωθεί το μοντέλο του μικροελεγκτή στην FPGA, μαζί με την υπόλοιπη λογική που θα υπήρχε ούτως ή άλλως. Η σκέψη αυτή φαίνεται στο σχήμα 1.1.




Σχήμα 1.1 – Κατανομή του χώρου της FPGA.

Όπως φαίνεται και από το σχήμα, ένα σχετικά μικρό ποσοστό της FPGA καταλαμβάνεται από το μοντέλο IP core που έχει σχεδιαστεί και το υπόλοιπο που μένει, μπορεί να χρησιμοποιηθεί για την εξυπηρέτηση της εκάστοτε εφαρμογής. Η επικοινωνία μεταξύ του μοντέλου IP core και της λογικής γίνεται εσωτερικά της FPGA, γεγονός που απλουστεύει πολύ το σύνολο των κυκλωμάτων που πρόκειται να χρειαστούν, ενώ παράλληλα θα μπορεί να γίνει αύξηση της λειτουργικότητάς του επεκτείνοντας την αρχιτεκτονική του.

Η οργάνωση της διπλωματικής εργασίας είναι η παρακάτω. Στο 2$^{ο}$ κεφάλαιο γίνεται μια σχετική έρευνα σε ό,τι αφορά τη σχεδίαση IP cores γενικά, των ενσωματωμένων μικροελεγκτών για αναδιατασσόμενη λογική, καθώς επίσης και τις εφαρμογές που κάνουν τη χρήση μικροελεγκτών και αναδιατασσόμενης λογικής μαζί. Στο 3$^{ο}$ κεφάλαιο περιγράφεται ο μικροελεγκτής του οποίου σχεδιάστηκε το μοντέλο. Στο 4$^{ο}$ κεφάλαιο περιγράφεται αναλυτικά η πρώτη έκδοση του μοντέλου που σχεδιάστηκε σε VHDL. Στο 5$^{ο}$ κεφάλαιο αναφέρονται ορισμένες βελτιώσεις που έγιναν, ώστε να μειωθεί πολύ ο όγκος που καταλάμβανε η σχεδίαση στην FPGA και περιγράφεται η απεικόνιση του μοντέλου σε αναδιατασσόμενη λογική και σε συγκεκριμένες FPGAs. Στο 6$^{ο}$ κεφάλαιο αναφέρονται οι δοκιμές που έγιναν για τη πιστοποίηση

του συστήματος και τέλος στο 7$^{ο}$ κεφάλαιο περιγράφονται τα συμπεράσματα που βγήκαν, καθώς και οι μελλοντικές επεκτάσεις που μπορούν να εφαρμοστούν, ώστε να γίνει βελτίωση ή και αναβάθμιση του μοντέλου.

# 2 Σχετική έρευνα.

Στο κεφάλαιο αυτό παρουσιάζεται η σχετική έρευνα σε ό,τι αφορά τη σχεδίαση και χρησιμοποίηση των IP cores σε διάφορες εφαρμογές. Είναι αλήθεια ότι τα τελευταία χρόνια, η εκτεταμένη χρήση των γλωσσών περιγραφής υλικού όπως VHDL, Verilog και SystemVerilog έχει οδηγήσει στη σχεδίαση πολλών IP cores, οι οποίοι έχουν τη δυνατότητα να επαναχρησιμοποιηθούν σε τελείως διαφορετικές εφαρμογές [24, 25, 26 27, 28]. Κάθε φορά που κάποιος θέλει να ξεκινήσει μια σχεδίαση με μια γλώσσα περιγραφής υλικού, μπορεί να αναζητήσει από διάφορες πηγές (Internet, βιβλία) κάποια κομμάτια κώδικα που είναι έτοιμα και ελεγμένα ως προς τη σωστή λειτουργία τους, με αποτέλεσμα να μη σπαταλήσει το χρόνο για την εκ νέου σχεδίασή τους. Άλλωστε οι σημερινές σχεδιάσεις κυκλωμάτων έχουν φτάσει σε τέτοια πολυπλοκότητα που, αν κάποιο project ξεκινούσε από την αρχή χωρίς την επαναχρησιμοποίηση έστω και κάποιων μικρών κομματιών, σίγουρα θα καθυστερούσε υπερβολικά την ολοκλήρωσή του.

Στη συνέχεια γίνεται αναφορά στις κατηγορίες των IP cores που μπορεί να συναντήσει κάποιος σήμερα στο Internet και περιγράφονται ορισμένα χαρακτηριστικά παραδείγματα για κάθε μία από αυτές. Ειδική αναφορά γίνεται στη κατηγορία των μικροελεγκτών, μιας και είναι αυτή που συνδέεται άμεσα με το αντικείμενο της διπλωματικής, δίνοντας αρκετά παραδείγματα παρόμοιων σχεδιάσεων. Κατόπιν περιγράφονται και κάποια είδη εφαρμογών που χρησιμοποιούν μικροελεγκτές με αναδιατασσόμενη λογική και που θα μπορούσαν να επωφεληθούν πολύ από τη χρήση κάποιου μοντέλου μικροελεγκτή στο θέμα της πολυπλοκότητας και της ευκολίας.

## 2.1 Κατηγορίες IP cores.

Οι κατηγορίες των IP cores που μπορεί να συναντήσει κανείς στο Internet ή βιβλία είναι αρκετές και στις περισσότερες περιπτώσεις το πιθανότερο είναι να μπορεί να γίνει επαναχρησιμοποίηση κάποιων IP cores σε καινούρια projects. Για παράδειγμα, κάποιο έτοιμο IP core θα μπορούσε να χρησιμοποιηθεί ως «μαύρο κουτί» σε μια μεγαλύτερη σχεδίαση, αρκεί να είναι γνωστή η συμπεριφορά του. Ακόμα, θα μπορούσε να υλοποιηθεί σε μια FPGA έτσι ώστε η τελευταία να συμπεριφέρεται όπως ακριβώς περιγράφεται στο κώδικα του IP core και στη συνέχεια να γίνουν εξωτερικές συνδέσεις με την FPGA ώστε να υλοποιηθεί κάποιο μεγαλύτερο κύκλωμα.

Μια κατηγορία που υπάρχει είναι αυτή των IP cores που χρησιμοποιούνται για διάφορες υπολογιστικές πράξεις [26, 28]. Έτσι, μπορεί κανείς να βρει μοντέλα που υλοποιούν απλές ALU (Arithmetic Logic Units), κάνουν διαιρέσεις, κάνουν πράξεις με δεκαδικούς ή και ακόμα υπολογίζουν, με τη βοήθεια αλγορίθμων, το μετασχηματισμό Fourier. Σε αυτή τη κατηγορία θα μπορούσαν να ενταχθούν και οι λεγόμενοι συνεπεξεργαστές, οι οποίοι συνυπάρχουν με κάποιο επεξεργαστή και ως σκοπό τους έχουν να κάνουν δύσκολες μαθηματικές πράξεις για να αποφορτίσουν τον επεξεργαστή.

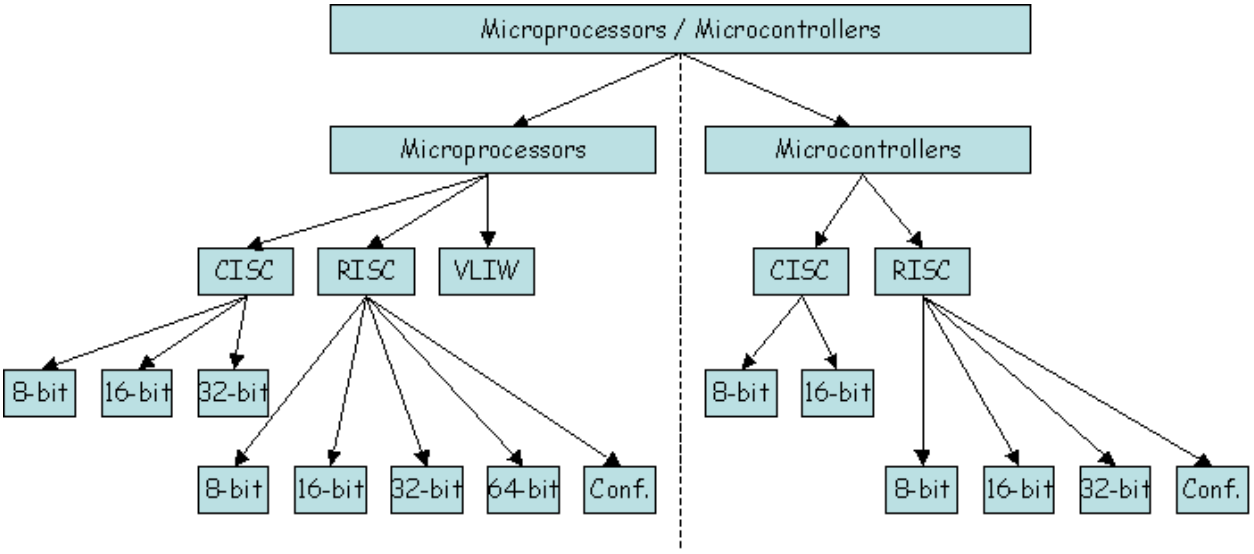
Αρκετά μεγαλύτερη είναι η κατηγορία των IP cores που χρησιμοποιούνται ως ελεγκτές (controllers) σε διάφορα είδη επικοινωνίας [26, 28]. Έτσι, μπορεί κανείς να βρει IP cores που υλοποιούν ελεγκτές για Ethernet 10/100 Mbps, επικοινωνία SPI (Serial Peripheral Interface), πρωτόκολλο CAN (Controller Area Network), UART, USB και USB 2.0 και επικοινωνία σειριακή ή με υπέρυθρες. Επίσης υπάρχουν και μοντέλα που υλοποιούν αλγόριθμους για κρυπτογράφηση και αποκρυπτογράφηση κατά τη διάρκεια της μετάδοσης πληροφοριών, όπως ο DES/Triple DES (Data Encryption Standard), AES (Advanced Encryption Standard) και ο SCAN [22],

Μια εξίσου μεγάλη κατηγορία IP cores είναι αυτή που υλοποιεί διάφορα είδη μνημών [11, 12, 26, 28]. Έτσι, μπορεί κανείς να βρει μνήμες σύγχρονες ή συνδυαστικές, μιας ή και περισσοτέρων θυρών ανάγνωσης, καθώς επίσης FIFOs (First In First Out). Ένα χαρακτηριστικό που κάνει ιδιαίτερα χρήσιμη τη κατηγορία αυτή είναι ότι οι περισσότερες σχεδιάσεις γίνονται πολύ εύκολα παραμετροποιήσιμες, γεγονός που σημαίνει ότι ο εκάστοτε σχεδιαστής μπορεί να αλλάξει το μέγεθος και το πλάτος της μνήμης ανάλογα με τις δικές του ανάγκες.

Διάφορες άλλες κατηγορίες περιλαμβάνουν τη σχεδίαση IP cores που υλοποιούν φίλτρα για ψηφιακή επεξεργασία σήματος, όπως τα FIR (Finite Impulse Response), ελεγκτές για συστήματα, όπως γέφυρα PCI (Peripheral Controller Interface), AC 97 και μνήμες [26]. Αξίζει να σημειωθεί επίσης η ύπαρξη ειδικής κατηγορίας IP cores που υλοποιούν ελεγκτές Video, οι οποίοι μπορεί να είναι για VGA (Video Graphic Adapter) ή να κάνουν κάποια συμπίεση. Σαν τελευταία κατηγορία αναφέρεται σκόπιμα αυτή των μικροεπεξεργαστών και μικροελεγκτών, η οποία περιγράφεται αναλυτικά στην επόμενη παράγραφο.

## 2.2 Μοντέλα IP cores μικροελεκτών και μικροεπεξεργαστών που υπάρχουν.

Η συγκεκριμένη κατηγορία των IP cores είναι ιδιαίτερα πλούσια σε επιλογές [26,28], καθώς έχουν γίνει υλοποιήσεις πολλών μικροελεγκτών και μικροεπεξεργαστών που ήδη κυκλοφορούν στο εμπόριο. Τα μοντέλα που μπορεί να βρει κανείς ποικίλουν ανάλογα με το εύρος των σημάτων που χρησιμοποιούν (4-bit, 8-bit, 12-bit, 16-bit, 32-bit, 64-bit), καθώς επίσης και από το είδος του σετ εντολών τους (RISC – Reduced Instruction Set Computer, CISC – Complex Instruction Set Computer, VLIW – Very Long Instruction Word) [36, 37]. Έτσι, στη συνέχεια γίνεται ξεχωριστή αναφορά στους μικροεπεξεργαστές και ξεχωριστή αναφορά στους μικροελεγκτές με βάση τους παραπάνω διαχωρισμούς, γεγονός που φαίνεται και στο σχήμα 2.1a.




Σχήμα 2.1a – Διαχωρισμός Μικροεπεξεργαστών και μικροελεγκτών.

### 2.2.1 Μοντέλα μικροεπεξεργαστών.

Τα μοντέλα των μικροεπεξεργαστών που έχουν υλοποιηθεί μέχρι σήμερα με τη βοήθεια κάποιας γλώσσας περιγραφής υλικού είναι αρκετά και καλύπτουν πολλές κατηγορίες [28]. Ύστερα από σχετική έρευνα, βρέθηκε ότι ιδιαίτερη βαρύτητα έχει δοθεί σε CISC μικροεπεξεργαστές και πιο συγκεκριμένα στα 16, 8 και 32 bit. Στα 16-bit μπορεί κανείς να βρει τα μοντέλα των 8086 και 80186 της Intel από διάφορες εταιρίες, όπως ARC International, CAST Inc. και Infinior Microsystems. Στα 8-bit υπάρχουν κυρίως τα μοντέλα των Zilog 80 (Z80) από

την Arc International και Z180 από την Systemyde International Corporation, 6809 από την Actel Corporation και 65c02 από την Sierra Circuit Design. Στην κατηγορία των 32-bit υπάρχει κυρίως το μοντέλο του 68000 της Motorola από τις Evatronix SA και VLSI Concepts.

Στην κατηγορία των RISC μικροεπεξεργαστών υπάρχει επίσης αρκετά μεγάλη ποικιλία υλοποιήσεων. Στα 32-bit βρίσκονται, μεταξύ άλλων, τα μοντέλα των ARCTangent A5/A4 από την ARC International, PowerPC 440/405 της IBM από την Synopsys Inc., ARM 922T από την Global Unichip Corp., MIPS32 4Kec από την LSI Logic Corporation, ένα μοντέλο μικροεπεξεργαστή χαμηλής κατανάλωσης για εφαρμογές με SmartCards από την ARM Ltd, τα μοντέλα ARM 966E-S/946E-S/926EJ-S, 7TDMI-S/9TDMI-S και ARM 7Thumb από την LSI Logic Corporation και τα μοντέλα των ARM 1136 JF-S/ J-S από την ARM Ltd. Στα 64-bit υπάρχει κυρίως το μοντέλο του MIPS64 5kf από την LSI Logic Corporation. Στα 8-bit υπάρχουν τα μοντέλα των ARClite της ARC International και των 80C51/80C52 της Samsung Electronics, ενώ στην κατηγορία των 16-bit μπορεί κανείς να βρει τα μοντέλα διαφόρων επεξεργαστών από εταιρίες όπως Eleven Engineering, Cambridge Consultants Ltd., Loarant Corporation και LSI Logic Corporation. Αξίζει να σημειωθεί ότι σε αυτή την κατηγορία υπάρχουν και επεξεργαστές που είναι παραμετροποιήσιμοι ως προς το εύρος και σαν παράδειγμα αναφέρονται τα μοντέλα των Nios SoftCore Embedded Processor από την Atmel, ARCTangent A5 από την ARC International και Proteus 3 Protocol Processor από την Silaria.

Στην κατηγορία των VLIW μικροεπεξεργαστών υπάρχει μικρότερη ποικιλία μοντέλων που έχουν υλοποιηθεί σε σχέση με τις παραπάνω κατηγορίες [28] και αναφέρεται ένα μοντέλο για ψηφιακή επεξεργασία σήματος από την Improv Systems Corp.

### 2.2.2 Μοντέλα IP cores μικροελεγκτών.

Όπως και στους μικροεπεξεργαστές, έτσι και οι μικροελεγκτές χωρίζονται σε κατηγορίες, ανάλογα με το εύρος τους (16-bit, 8-bit, 32-bit) και το είδος του σετ εντολών τους (RISC, CISC) [26, 28]. Στην κατηγορία των CISC μικροελεγκτών βρέθηκαν μοντέλα στα 8 και 16 bit. Στα 8-bit υπάρχει αυτό του 80530 της Dallas από την Evatronix SA. Στα 16-bit υπάρχουν κυρίως τα μοντέλα του C166S από τις Infineon Technologies A.G. και Synopsys Inc. και το μοντέλο του 65C265S από την Western Design Center Inc.

Στην κατηγορία των RISC μικροελεγκτών υπάρχουν μοντέλα στα 32, 16 και 8 bit, με την τελευταία να είναι ιδιαίτερα ενδιαφέρουσα μιας και σε αυτή τη κατηγορία ανήκει και το μοντέλο που σχεδιάστηκε στα πλαίσια της διπλωματικής. Στα 32-bit βρέθηκαν τα μοντέλα των ARM966E-S και ARM7TDMI από τις LSI Logic Corporation και R&D Technologies Inc. αντίστοιχα. Στα 16-bit υπάρχουν τα μοντέλα των ZSP 400/500/600 από την LSI Logic Corporation, καθώς και ένα μοντέλο μικροελεγκτή χαμηλής κατανάλωσης της e-vision Inc. Ειδικά για τον ZSP600, υπάρχει και ένα μοντέλο που είναι παραμετροποιήσιμο ως προς το εύρος και είναι πάλι από την LSI Logic Corporation. Στα 8-bit βρέθηκαν, μεταξύ άλλων, ο 8051 της Intel από τις Mentor Graphics (IP Division), Alatec Inc., Silicon Cores, VLSI Concepts, RIIC και Evatronix SA, ο 6811 από τις Digital Core Design και Dolphin Integration, τα μοντέλα των 8031/8032/8051 από την QualCore Logic Inc. και Faraday Technology Corporation, το μοντέλο του 80C51 από την Institute of Electron Technology, ο 1802 από την Sierra Circuit Design και οι Atmega103, AT90S1200 και AT90S2313 της Atmel.

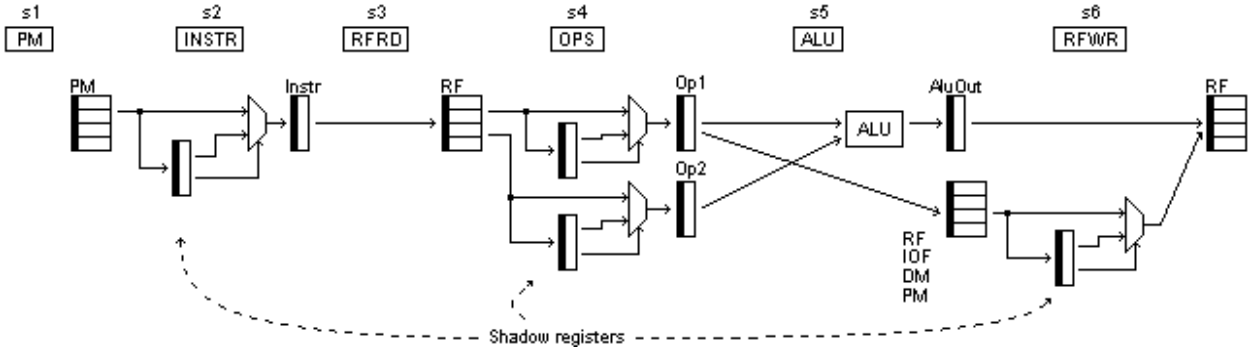
## 2.3 Μοντέλο IP core της οικογένειας μικροελεγκτών AVR.

Εκτός από τα παραπάνω IP core μοντέλα, ένα ακόμα που παρουσιάστηκε κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας και συγκλίνει πολύ στο στόχο της τελευταίας, ήταν αυτό ενός παραμετροποιήσιμου μικροελεγκτή που είναι συμβατό με την αρχιτεκτονική AVR της Atmel [32]. Όπως αναφέρεται από τους σχεδιαστές, σκοπός ήταν να υλοποιηθεί ένα γενικό μοντέλο IP core που να υποστηρίζει τις εντολές αρκετών μικροελεγκτών της Atmel, χωρίς να δοθεί ιδιαίτερο βάρος στα χαρακτηριστικά (αριθμός θυρών εισόδου/εξόδου, υποστηριζόμενα interrupts, Timer/Counters) κάποιου μεμονωμένου. Έτσι, το συγκεκριμένο μοντέλο IP Core αποτελείται από τα εξής χαρακτηριστικά:

- Έχει 6 επίπεδα pipeline (σχήμα 2.1b). Στο $1^o$ γίνεται ανάγνωση της μνήμης εντολών (PM – Program Memory), στο $2^o$ γίνεται μεταφορά της εξόδου της μνήμης εντολών στον καταχωρητή εντολής (INSTR – Instruction Register), στο $3^o$ γίνεται αποκωδικοποίηση της εντολής και διάβασμα της RF (RFRD – RF Read), στο $4^o$ γίνεται η εξαγωγή των περιεχομένων της RF, ενώ στο $5^o$ γίνεται η εκτέλεση κάποιας πράξης στην ALU

(Arithmetic Logic Unit) ή προσπέλαση στη μνήμη δεδομένων και τέλος στο 6$^{o}$ γίνεται εγγραφή των αποτελεσμάτων πίσω στην RF (RFWR – RF Write).

- Συνήθως χρειάζεται 1 κύκλος για την ολοκλήρωση κάθε εντολής. Όπως αναφέρεται, το μοντέλο είναι πιο αποδοτικό σε εντολές υπολογισμού, ενώ οι εντολές διακλάδωσης κοστίζουν περισσότερους κύκλους.
- Έχει υπολογιστεί ότι μπορεί να τρέξει περίπου στα 50 MHZ, αλλά δεν έχει δοκιμαστεί σε κάποια FPGA.
- Περιλαμβάνει μέχρι και 32 διαφορετικές πηγές για interrupts με ξεχωριστή διεύθυνση εξυπηρέτησης για το καθένα.
- Έχει 1 πόρτα για επικοινωνία με τον έξω κόσμο, 1 Timer/Counter 8-bit, 1 εξωτερικό interrupt.
- Δεν περιλαμβάνει UART και SPI.




Σχήμα 2.1b – Pipeline μοντέλου IP core.

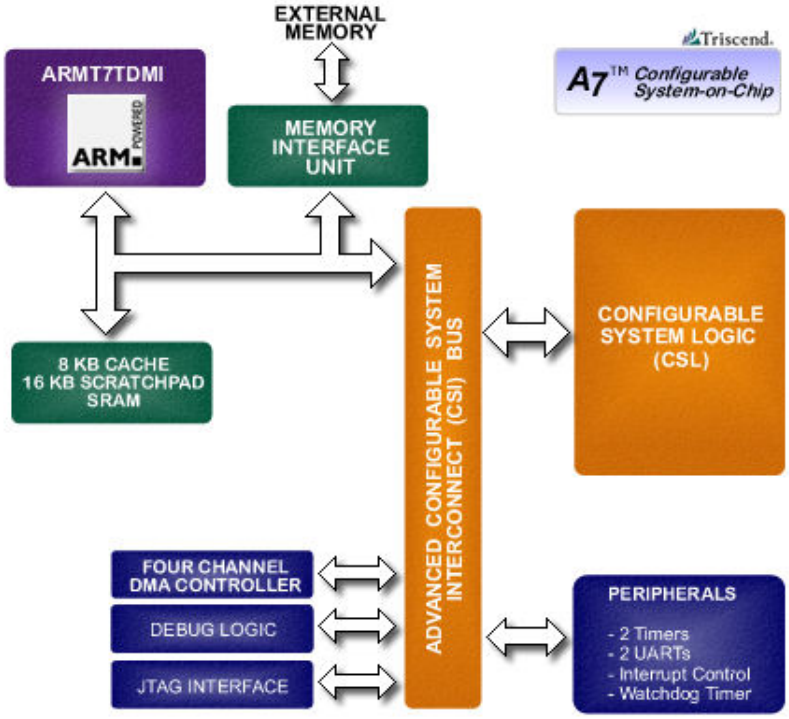
## 2.4 Εφαρμογές που χρησιμοποιούν μικροελεγκτές και αναδιατασσόμενη λογική.

Στην προηγούμενη παράγραφο αναφέρθηκαν, μεταξύ άλλων, πολλά μοντέλα IP cores μικροελεγκτών που υπάρχουν σήμερα. Στην παρούσα παράγραφο γίνεται αναφορά σε διάφορα προϊόντα και εφαρμογές που χρησιμοποιούν μικροελεγκτές με κάποια FPGA. Είναι αλήθεια ότι τα τελευταία χρόνια ο συνδυασμός των τελευταίων με αναδιατασσόμενη λογική χρησιμοποιείται αρκετά, μιας και προσφέρει αρκετές δυνατότητες και ευκολίες στην υλοποίηση κυκλωμάτων. Δεν είναι τυχαίο το γεγονός ότι όλο και περισσότερα development kits εμφανίζονται στο

εμπόριο από πολλές εταιρίες, τα οποία περιλαμβάνουν μια FPGA και έναν μικροελεγκτή / μικροεπεξεργαστή ή χρησιμοποιούν πολλά περιφερειακά κυκλώματα, όπως UART, seven segment displays, υπέρυθρες ακτίνες, Bluetooth και διάφορα άλλα Χαρακτηριστικό παράδειγμα αποτελούν τα αναπτυξιακά κυκλώματα που έχουν σχεδιαστεί από τις εταιρίες Xilinx, Altera και Triscend.

Από την εταιρεία Triscend [43] υπάρχουν development kits που βασίζονται σε ένα ολοκληρωμένο κύκλωμα το οποίο περιλαμβάνει αναδιατασσόμενη λογική και συγχρόνως κάποιο ARM μικροεπεξεργαστή. Πιο συγκεκριμένα, το πρώτο είναι το Triscend A7, το οποίο είναι ένα Configurable SoC (System On a Chip) και περιλαμβάνει χώρο για αναδιατασσόμενη λογική και τον 32-bit ARM7TDMI μικροεπεξεργαστή. Στο σχήμα 2.2a φαίνεται ένα block διάγραμμά του και στο 2.2b μια φωτογραφία του. Ακόμα περιλαμβάνει:

- Embedded SRAM-based Configurable System Logic (CSL) matrix.
- Flexible and seamless interface to external memories.
- Integrated Peripherals functions.
- Advanced system interconnect architecture.

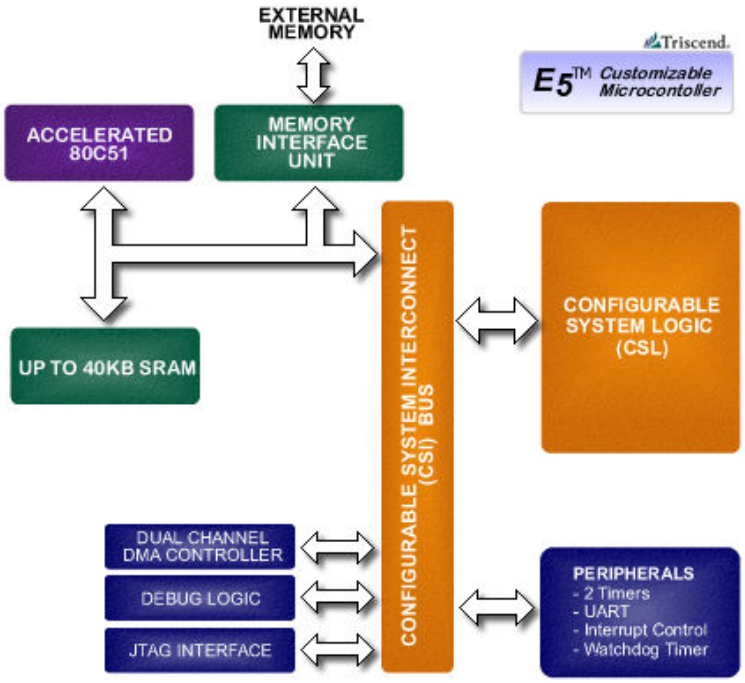



Σχήμα 2.2a – Block διάγραμμα του Triscend A7.

Σχήμα 2.2b – Triscend A7.

Επίσης υπάρχει και το Triscend E5, το οποίο είναι και αυτό ένα Configurable SoC και περιλαμβάνει χώρο για αναδιατασσόμενη λογική και τον μικροελεγκτή 8-bit 8051. Στο σχήμα 2.2c φαίνεται ένα block διάγραμμά του και στο 2.2d μια φωτογραφία του. Ακόμα περιλαμβάνει:

- Enhanced, high-performance, 8051-based microcontroller.
- Embedded Configurable System Logic (CSL).
- High-performance, dedicated Configurable System Interconnect (CSI) system bus.
- Enhanced programmable input/output (PIO) ports.
- Memory interface unit (MIU) for flexible, glue-less interface to external memory.




Σχήμα 2.2c – Block διάγραμμα του E5.

Σχήμα 2.2d – Triscend E5.

Με βάση τα δύο παραπάνω ολοκληρωμένα κυκλώματα η συγκεκριμένη εταιρεία έχει κατασκευάσει αρκετά development kits. Σαν παράδειγμα που χρησιμοποιεί το Triscend A7, αναφέρεται το Triscend A7 GNU Ethernet, το οποίο φαίνεται στο σχήμα 2.2e και περιλαμβάνει:

- Triscend A7 Configurable SoC.
- 8MB Flash, 32 MB SDRAM.
- RS232, Ethernet Interface.
- LED display, 2 push buttons, 4 DIP switches.

Σχήμα 2.2e – Triscend A7 GNU Ethernet Development Kit.

Επίσης αναφέρεται το Triscend A7 GNU Standard Development Kit, το οποίο φαίνεται στο σχήμα 2.2f και περιλαμβάνει:

- Triscend A7 Configurable SoC.
- JTAG, 2 RS232 Interfaces.
- 2 MB Flash, 32MB SDRAM.

Σχήμα 2.2f – Triscend A7 GNU Standard Development Kit.

Σαν παράδειγμα που χρησιμοποιεί το Triscend E5, αναφέρεται το Triscend Graphic LCD Development Kit, το οποίο φαίνεται στο σχήμα 2.2g και περιλαμβάνει:

- Triscend E5 Configurable SoC.
- Graphic LCD Interface Board.
- Supports any passive LCD panel up to QVGA.

Σχήμα 2.2g – Triscend Graphic LCD Development Kit.

Από την Xilinx [30], υπάρχουν πολλά development kits, τα οποία χρησιμοποιούν κάποια FPGA σε συνδυασμό με πολλά περιφερειακά κυκλώματα ή κάποιον μικροελεγκτή / μικροεπεξεργαστή. Σαν παράδειγμα, αναφέρεται από τη Xilinx το Virtex-II XC2V4000XP που περιλαμβάνει μια Virtex 2 Pro σε συνδυασμό με τον ενσωματωμένο μικροεπεξεργαστή

MicroBlaze [30] της ίδιας εταιρείας και φαίνεται στο σχήμα 2.3a. Πέρα από αυτά που ήδη αναφέρθηκαν, περιλαμβάνει:

- USB 2.0 Controller.
- PCI / PCI-X Connector.
- JTAG, RS232, 10 / 100 / 1000 Ethernet Interfaces.
- 128 MB DDR- SDRAM (Double Data Rate – Synchronous Dynamic Random Access. Memory), 24MB Flash, 64MB SDRAM, 1MB SRAM, 4KB EEPROM.

Σχήμα 2.3a – XC2V4000XP με MicroBlaze.

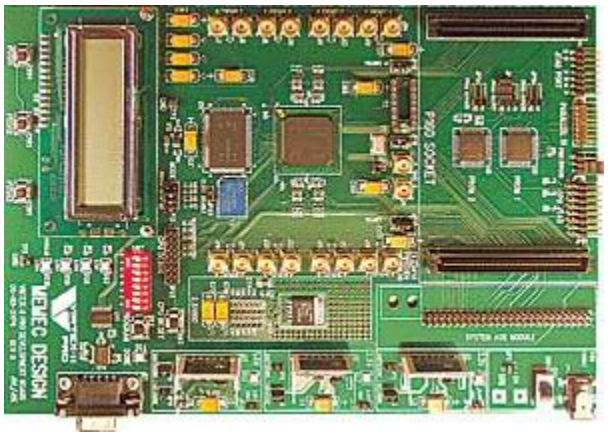
Επίσης αναφέρεται το Virtex-E Development Kit, το οποίο περιλαμβάνει μια VirtexE σε συνδυασμό πάλι με τον ενσωματωμένο μικροεπεξεργαστή Microblaze και φαίνεται στο σχήμα 2.3b. Πέρα από αυτά που ήδη αναφέρθηκαν, περιλαμβάνει:

- Video Decoder, Video και Stereo DAC (Digital to Analog Converter).
- VGA out, Video In.
- SDRAM, Flash.
- JTAG, RS232, PCI / PCI-X, 10 / 100 Ethernet, USB 2.0, CAN Bus.

Σχήμα 2.3b – Virtex-E Development Kit με MicroBlaze.

Ένα ακόμα development kit που αξίζει να αναφερθεί ακόμα ως αντιπροσωπευτικό παράδειγμα είναι το Virtex-II Pro P4/P7 FG456, το οποίο περιλαμβάνει PowerPC μικροεπεξεργαστή και φαίνεται στο σχήμα 2.3c. Πέρα από αυτά που αναφέρθηκαν, περιλαμβάνει:

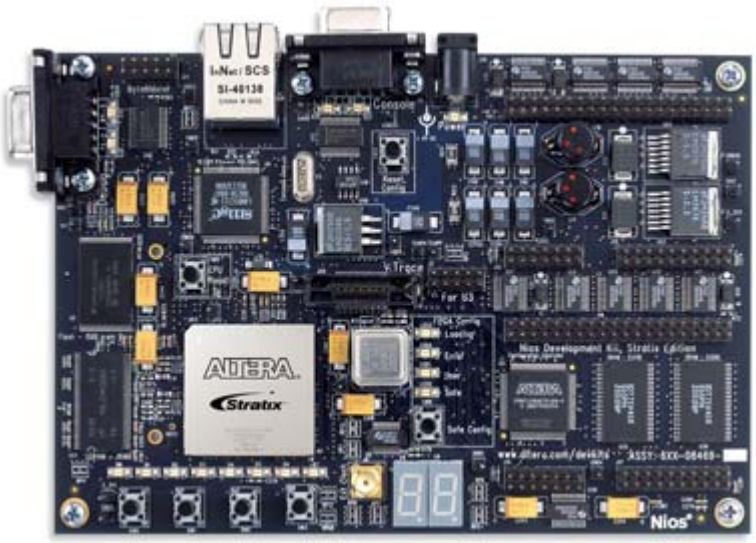
- RS232 and additional user support circuits.
- SDRAM.
- Four I/O Rocket Tranceivers.



Σχήμα 2.3c – Virtex-II Pro P4/P7 FG456 με PowerPC.

Από την Altera, υπάρχουν επίσης πολλά development kits που βασίζονται στο συνδυασμό της αναδιατασσόμενης λογικής και κάποιου μικροεπεξεργαστή. Σαν πρώτο παράδειγμα αναφέρεται το Nios Development Kit, Stratix Edition που περιλαμβάνει την Stratix EP1S10F780 FPGA και τον Nios ενσωματωμένο μικροεπεξεργαστή [31] και φαίνεται στο σχήμα 2.4a. Κάποια επιπλέον χαρακτηριστικά του είναι:

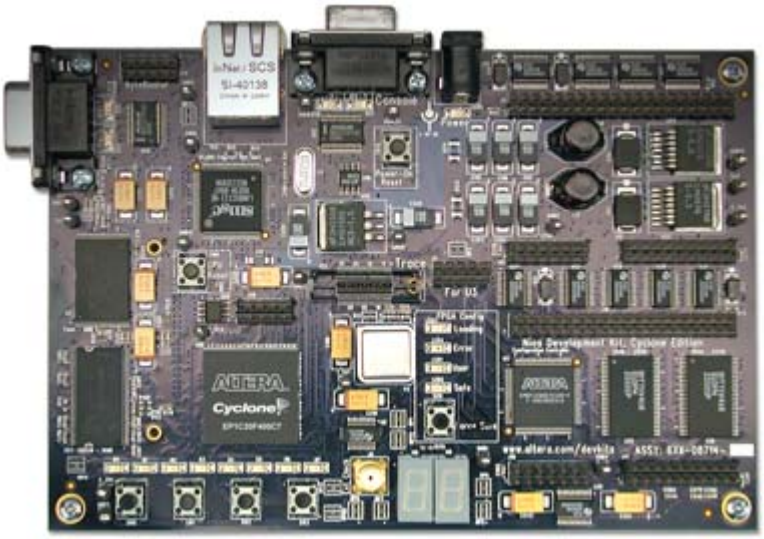
- 1 MB SRAM, 16 MB SDRAM, 8 MB Flash.
- 10/100 Ethernet physical layer/media access control (PHY/MAC).
- CompactFlash connector header για Type I CompactFlash cards.
- Ethernet connector (RJ-45).
- RS-232 DB9 port.
- 2 Joint Test Action Group (JTAG) connectors.




Σχήμα 2.4a – Nios Development Kit, Stratix Edition.

Σαν ένα δεύτερο παράδειγμα αναφέρεται το Nios Development Kit, Cyclone Edition που περιλαμβάνει την Cyclone EP1C20FC400 FPGA και τον ενσωματωμένο μικροεπεξεργαστή Nios και φαίνεται στο σχήμα 2.4b. Κάποια επιπλέον χαρακτηριστικά του είναι:

- 1 MB SRAM, 16 MB SDRAM, 8 MB Flash.
- 10/100 Ethernet physical layer/media access control (PHY/MAC).
- Ethernet connector (RJ-45).
- EPCS4 Serial Configuration Device (4 Mbits)
- RS-232 DB9 port.
- 2 Joint Test Action Group (JTAG) connectors.




Σχήμα 2.4b – Nios Development Kit, Cyclone Edition.

Αρκετές κατηγορίες εφαρμογών εκμεταλλεύονται τη χρήση των μικροελεγκτών σε συνδυασμό με την αναδιατασσόμενη λογική. Σαν παράδειγμα μπορεί να αναφερθεί η κατασκευή κυκλωμάτων ασφαλείας, όπου ανάλογα με τις ανάγκες της κάθε περίπτωσης, μπορεί να προσαρμοστεί και η σχεδίαση του αντίστοιχου συστήματος ασφαλείας. Αν σε άλλη περίπτωση, είναι επιθυμητή κάποια αλλαγή ή προσθήκη επιπλέον λειτουργιών, ο συνδυασμός της αναδιατασσόμενης λογικής και των μικροελεγκτών / μικροεπεξεργαστών, βοηθάει πολύ στη μείωση του χρόνου ανάπτυξης του νέου συστήματος. Μια ακόμα κατηγορία εφαρμογών είναι στη βιομηχανία και πιο συγκεκριμένα στον έλεγχο λειτουργίας αισθητήρων, μηχανισμούς κίνησης ρομποτικών βραχιόνων, καθώς και στον έλεγχο της κατάστασης διαφόρων βαλβίδων.

Ο συνδυασμός των μικροελεγκτών με τις FPGAs βρίσκει απήχηση και στην κατηγορία των εφαρμογών που με τους πρώτους ελέγχεται ο προγραμματισμός των δεύτερων [39]. Σαν παράδειγμα, αναφέρεται ένα application note της Atmel, όπου ο προγραμματισμός μιας AT6000 FPGA [38] ελέγχεται από έναν AT89C51 μικροελεγκτή.

Επίσης, σαν ένα ακόμα παράδειγμα εφαρμογής που χρησιμοποιεί το συνδυασμό του μικροελεγκτή με την FPGA, είναι η κατασκευή ειδικών ασανσέρ και πλατφόρμων για άτομα με ειδικές ανάγκες, προκειμένου να μπορούν να κινηθούν άνετα σε κάποιο ιδιωτικό χώρο με διαφορετικά επίπεδα [20]. Παρόλο που αντίστοιχα προϊόντα υπάρχουν ήδη στο εμπόριο, η απόφαση της χρήσης μιας FPGA με μικροελεγκτή οδήγησε σε σημαντική μείωση των εξόδων εγκατάστασης και πολύ πιο αξιόπιστη λειτουργία. Επιπροσθέτως, μια ακόμα εφαρμογή του

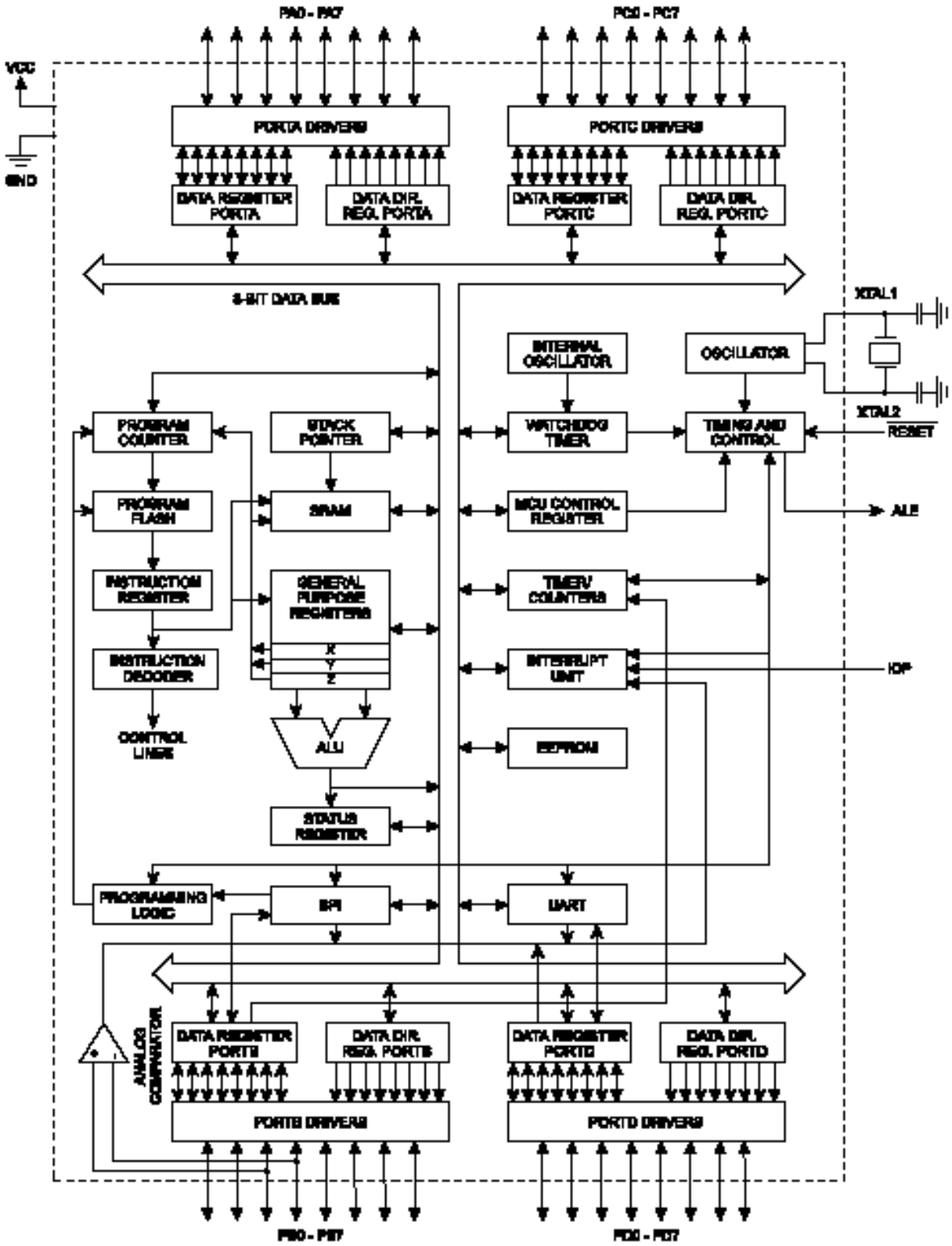
παραπάνω συνδυασμού που θα μπορούσε να αναφερθεί, είναι αυτή που εκπονήθηκε στα πλαίσια της μεταπτυχιακής εργασίας του κ. **Κυπριανού Παπαδημητρίου** [19] και στη συνέχεια βελτιώθηκε από τον κ. **Σταμάτη Σωτηρόπουλο** [18] στα πλαίσια της διπλωματικής του εργασίας. Πρόκειται για μια βιοϊατρική εφαρμογή για άτομα με ειδικές ανάγκες, η οποία έχει ως σκοπό να απλοποιήσει τη διαδικασία εκπόνησης ορισμένων καθημερινών συνηθειών, όπως το άναμμα των φώτων ή τη χρήση του κλιματιστικού. Για να γίνει κάτι τέτοιο, έχει χρησιμοποιηθεί, μεταξύ άλλων, μια FPGA, η οποία ενεργοποιεί την αντίστοιχη έξοδο ανάλογα με το είδος της κίνησης που αναγνωρίστηκε και ένας μικροελεγκτής, που επεξεργάζεται τις μετρήσεις κάποιων επιταχυνσιομέτρων που χρησιμοποιούνται για την ανίχνευση της κίνησης.

# 3 Ο μικροελεγκτής AT90S8515 της Atmel.

Στο κεφάλαιο αυτό γίνεται αναφορά στον AT90S8515 και πιο συγκεκριμένα στα επιμέρους χαρακτηριστικά του και στόχο έχει την παρουσίαση του μικροελεγκτή, πάνω στον οποίο βασίστηκε η υλοποίηση του μοντέλου IP core στα πλαίσια της διπλωματικής εργασίας. Μπορεί να πρόκειται για έναν 8-bit μικροελεγκτή, δεν παύει όμως να είναι από τους ισχυρότερους στη κατηγορία του σε ό,τι αφορά τη ταχύτητα, τη λειτουργικότητα και τη χαμηλή κατανάλωση ενέργειας.

## 3.1 Γενική περιγραφή του μικροελεγκτή.

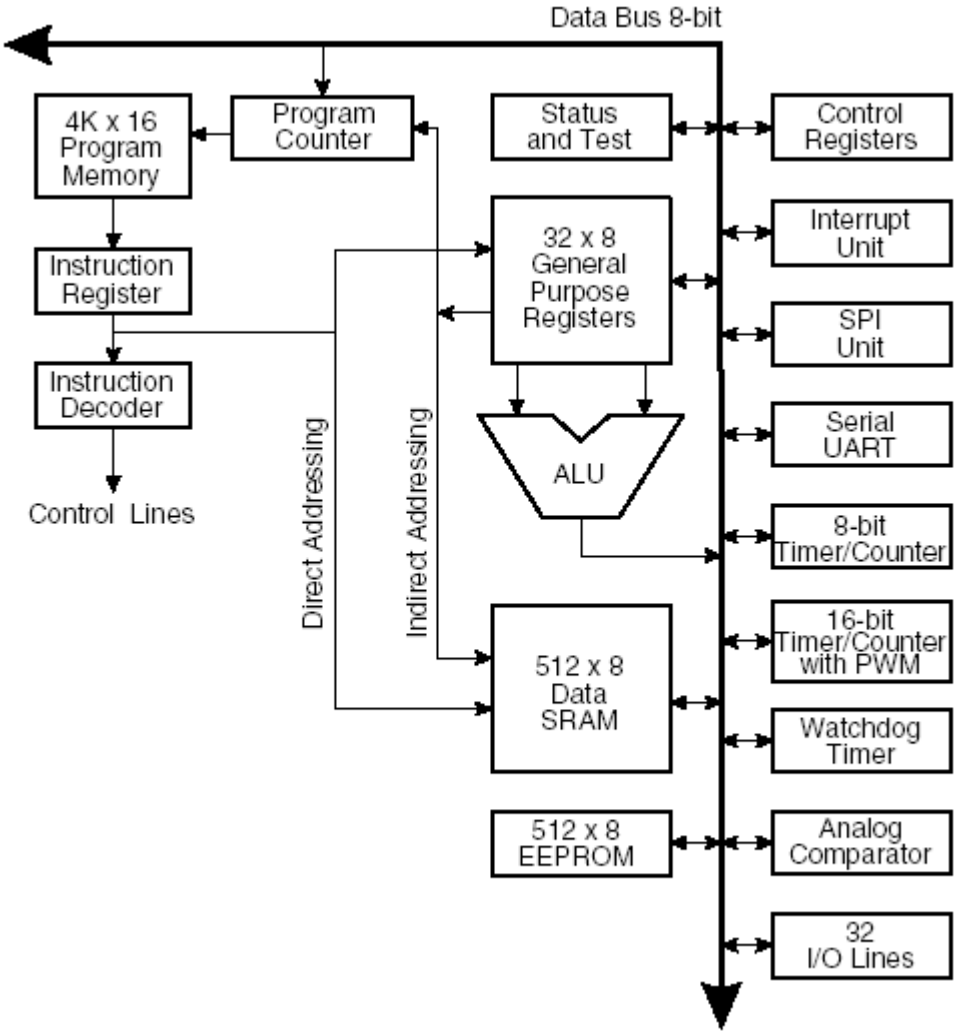
Ο AT90S8515 είναι ένας μικροελεγκτής με κύρια χαρακτηριστικά τη χαμηλή κατανάλωση ισχύος (Low Power) και την υποστήριξη πολλών εντολών, όπου οι περισσότερες χρειάζονται μόνο 1 κύκλο για την εκτέλεσή τους [9, 10]. Από πλευράς αρχιτεκτονικής, περιλαμβάνει 32 καταχωρητές γενικής χρήσεως (General Purpose Registers – GPRs), οι οποίοι είναι άμεσα συνδεδεμένοι με τη μονάδα λογικών πράξεων (Arithmetic Logic Unit – ALU), επιτρέποντας την άμεση πρόσβαση σε 2 ανεξάρτητους GPRs κατά τη διάρκεια ενός κύκλου. Ακόμα, υποστηρίζονται 2 εξωτερικά και 11 εσωτερικά interrupts. Παράλληλα, εντός του ολοκληρωμένου κυκλώματος, υπάρχουν 8Kbytes μνήμης εντολών (Instruction Memory), 512 bytes EEPROM (Electrically Erasable Programmable Read Only Memory), η οποία δε σβήνεται όταν σταματήσει η παροχή ενέργειας, και 512 bytes SRAM (Synchronous Random Access Memory) που χρησιμοποιείται ως μνήμη αποθήκευσης δεδομένων. Επιπροσθέτως, ο AT90S8515 περιλαμβάνει 4 8-bit θύρες (Ports) εισόδου/εξόδου, 1 UART (Universal Asynchronous Receiver Transmitter), 1 θύρα για SPI (Serial Peripheral Interface) επικοινωνία, 2 χρονιστές/μετρητές όπου ο ένας είναι 8-bit και ο άλλος 16-bit, 1 Watchdog timer και τέλος 1 αναλογικό συγκριτή. Αξίζει να σημειωθεί ότι, επειδή ο συγκεκριμένος μικροελεγκτής ανήκει στην κατηγορία χαμηλής κατανάλωσης, προσφέρει τη δυνατότητα επιλογής μεταξύ 2 καταστάσεων μείωσης της ενέργειας σε ό,τι αφορά τον τρόπο λειτουργίας. Στο σχήμα 3.1 φαίνεται ένα block διάγραμμα που δείχνει τη γενική δομή του AT90S8515. Στη συνέχεια γίνεται αναλυτική παρουσίαση όλων των παραπάνω χαρακτηριστικών του μικροελεγκτή, ξεκινώντας από την εσωτερική δομή της αρχιτεκτονική του και ύστερα παρουσιάζονται οι μονάδες εισόδου/εξόδου (UART, SPI, Ports, Analog Comparator).




Σχήμα 3.1. – Block διάγραμμα του AT90S8515.

## 3.2 Αρχιτεκτονική του μικροελεγκτή.

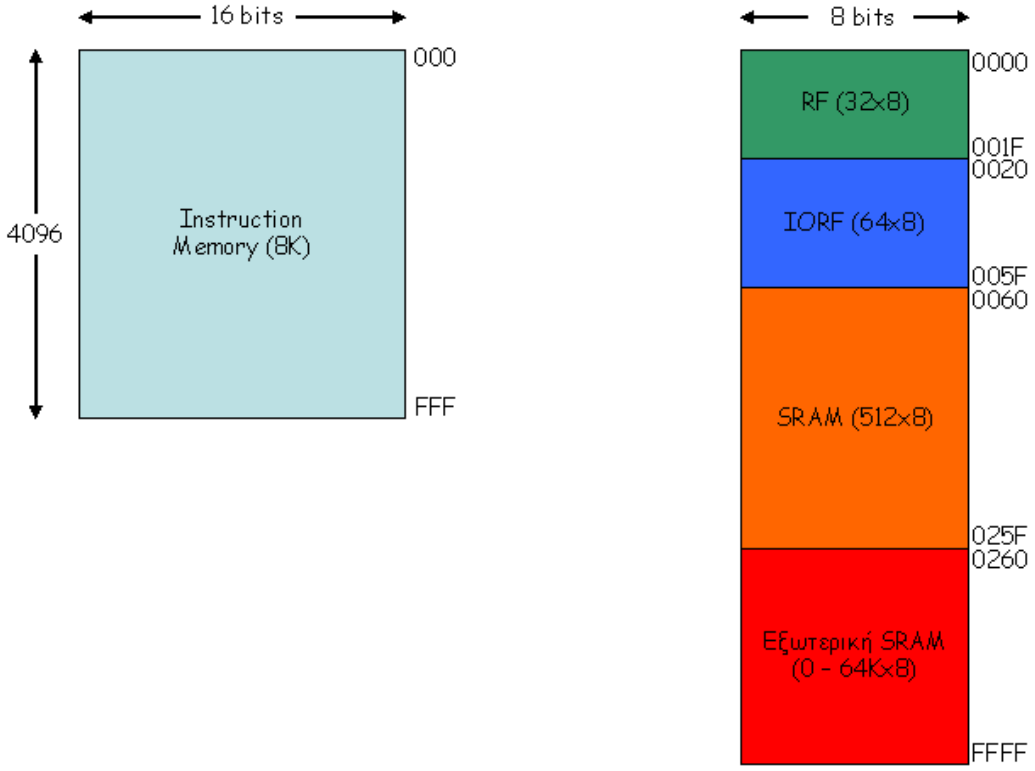
Στο σχήμα 3.2 φαίνεται ένα block διάγραμμα της αρχιτεκτονικής του AT90S8515, όπου φαίνεται η διαδρομή που ακολουθείται από την κάθε εντολή προκειμένου να ολοκληρωθεί η εκτέλεσή της. Πιο συγκεκριμένα, κατά τη διάρκεια 1 κύκλου, αρχικά διαβάζονται 2 τελεστέοι από την RF (Register File), στη συνέχεια εκτελείται η αντίστοιχη πράξη και κατόπιν το αποτέλεσμα αποθηκεύεται πάλι στην RF. Παρακάτω περιγράφονται αναλυτικά η κατανομή της μνήμης, ο τρόπος λειτουργίας της RF, η IORF (Input Output Register File), οι υποστηριζόμενοι τρόποι διευθυνσιοδότησης και τα interrupts που παρέχει ο μικροελεγκτής.




Σχήμα 3.2 – Block διάγραμμα της αρχιτεκτονικής του AT90S8515.

### 3.2.1 Κατανομή της μνήμης μέσα στον μικροελεγκτή.

Στο σχήμα 3.3 φαίνεται η κατανομή της μνήμης μέσα στο μικροελεγκτή [9]. Στο χρώμα του πάγου φαίνεται η μνήμη αποθήκευσης των εντολών που πρόκειται να εκτελεστούν. Η χωρητικότητά της είναι 8Kbytes, αλλά επειδή οι εντολές έχουν μήκος 16-bit (κάποιες έχουν 32-bit), είναι διαρρυθμισμένη έτσι ώστε να παρέχει 4096 θέσεις επί 16 bit στην κάθε θέση. Η RF καταλαμβάνει τις 32 πρώτες θέσεις διευθυνσιοδότησης (0000 ως 001F), η IORF καταλαμβάνει τις επόμενες 64 θέσεις διευθυνσιοδότησης (0020 ως 005F) η SRAM καταλαμβάνει τις επόμενες 512 θέσεις διευθυνσιοδότησης (0060 ως 025F). Αξίζει να σημειωθεί πως ο χρήστης θα πρέπει να χρησιμοποιεί το συγκεκριμένο εύρος για να προσπελάσει σωστά τη μνήμη. Τέλος, η πιθανή εξωτερική μνήμη μπορεί να έχει μέγεθος μέχρι 64K, οπότε το εύρος διευθυνσιοδότησής της είναι από 0260 ως FFFF.



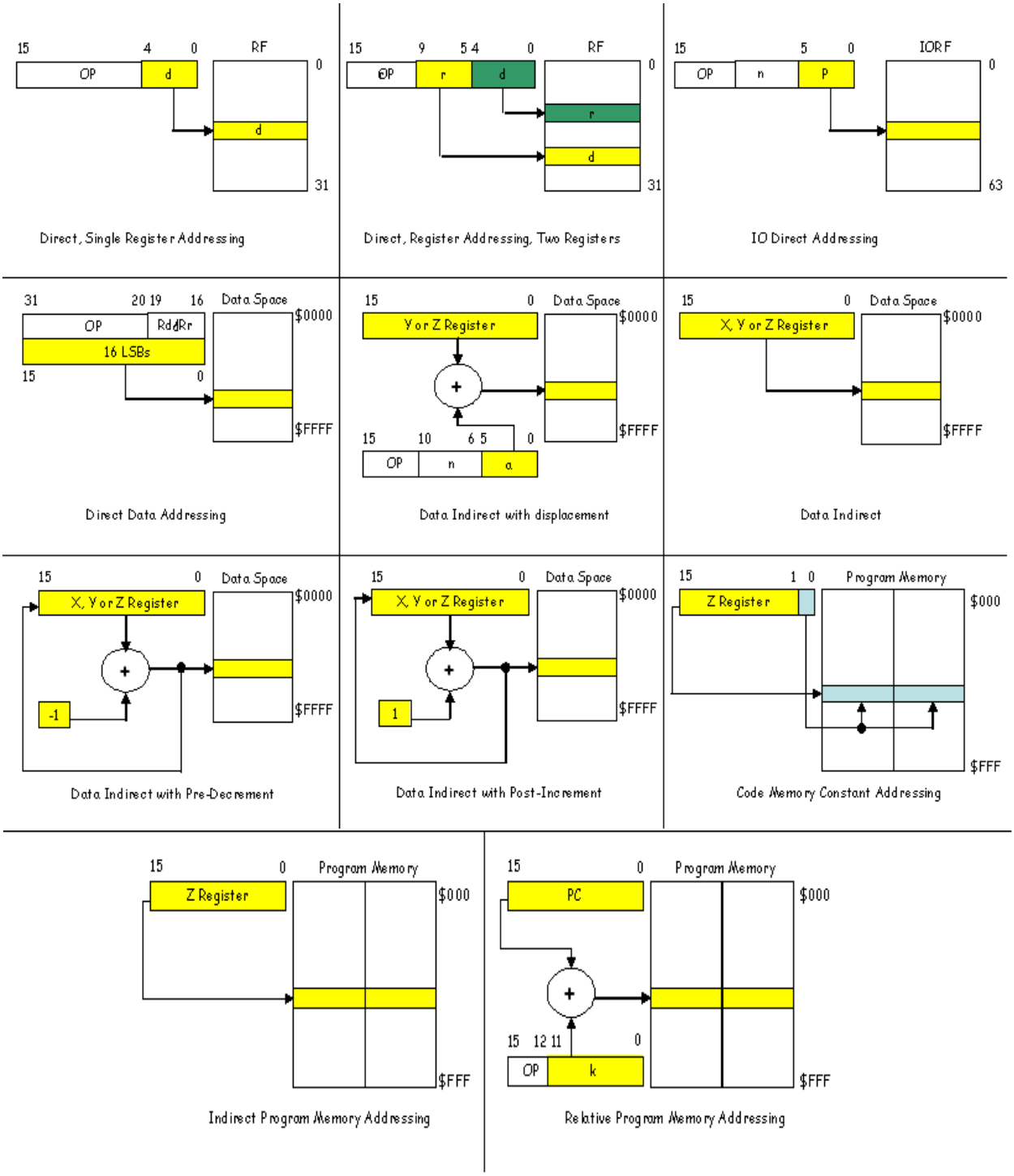

Σχήμα 3.3 – Κατανομή της μνήμης.

### 3.2.2 Τρόποι διευθυνσιοδότησης.

Ο AT90S8515 υποστηρίζει πολλούς διαφορετικούς τρόπους για την προσπέλαση των μνημών εντολών και δεδομένων [9]. Ανάλογα με το είδος της εκάστοτε εντολής που εκτελείται, χρησιμοποιείται και ο αντίστοιχος τρόπος προσπέλασης στις μνήμες. Παρακάτω, στο σχήμα 3.6, φαίνονται όλοι οι πιθανοί τρόποι που μπορούν να χρησιμοποιηθούν και στη συνέχεια γίνεται η επεξήγηση του καθενός.

- Direct Single Register Addressing: Η διεύθυνση του καταχωρητή στην RF που θέλουμε να προσπελάσουμε βρίσκεται στο πεδίο d της εντολής.
- Direct Register Addressing, Two Registers: Γίνεται ταυτόχρονη προσπέλαση σε 2 καταχωρητές της RF και οι διευθύνσεις τους βρίσκονται στα πεδία r και d της εντολής.
- IO Direct Addressing: Η διεύθυνση του IO καταχωρητή στην IORF που θέλουμε να προσπελάσουμε βρίσκεται στο πεδίο P και η διεύθυνση του καταχωρητή στην RF βρίσκεται στο n πεδίο.
- Direct Data Addressing: Στη συγκεκριμένη περίπτωση, η εντολή είναι 32-bit και τα 16 Least Significant Bits (LSBs) χρησιμοποιούνται για την προσπέλαση στη μνήμη δεδομένων. Στο πεδίο Rr/Rd βρίσκεται η διεύθυνση του καταχωρητή πηγής ή προορισμού της RF.
- Data Indirect with Displacement: Η διεύθυνση προσπέλασης στη μνήμη δεδομένων είναι το άθροισμα του Y ή Z καταχωρητή και του πεδίου a της εντολής. Το πεδίο n είναι η διεύθυνση του καταχωρητή στην RF.
- Data Indirect: Η διεύθυνση προσπέλασης στη μνήμη δεδομένων είναι ο X, Y ή Z καταχωρητής.
- Data Indirect with Pre-Decrement: Η διεύθυνση προσπέλασης στη μνήμη δεδομένων είναι ο X, Y ή Z καταχωρητής, αφού πρώτα μειωθεί κατά 1.
- Data Indirect with Post-Increment: Η διεύθυνση προσπέλασης στη μνήμη δεδομένων είναι ο X, Y ή Z καταχωρητής και μετά θα αυξηθεί κατά 1.
- Code Memory Constant Addressing: Ο συγκεκριμένος τρόπος διευθυνσιοδότησης χρησιμοποιείται από την εντολή LPM (Load Program Memory). Τα 15 Most Significant Bits (MSBs), χρησιμοποιούνται για την επιλογή της σωστής λέξης μέσα από τη μνήμη εντολών και το LSB για την επιλογή του κατάλληλου byte από τη συγκεκριμένη λέξη.
- Indirect Program Memory Addressing: Ο καταχωρητής Z χρησιμοποιείται ως δείκτης για την προσπέλαση της μνήμης εντολών.

- <u>Relative Program Memory Addressing:</u> Χρησιμοποιείται από τις εντολές RJMP (Relative Jump) και RCALL (Relative Call). Η διεύθυνση από όπου συνεχίζεται η εκτέλεση του κώδικα προέρχεται από το άθροισμα του πεδίου k και του Program Counter (PC).




Σχήμα 3.6 – Τρόποι διευθυνσιοδότησης.

### 3.2.3 Τα interrupts του μικροελεγκτή.

Όπως έχει αναφερθεί και πιο πριν, ο AT90S8515 υποστηρίζει 2 εξωτερικά και 11 εσωτερικά interrupts [9]. Όλα έχουν ξεχωριστή διεύθυνση που εξυπηρετούνται και το καθένα έχει δικά του bits που θα πρέπει να ενεργοποιηθούν προκειμένου να λειτουργήσουν. Στον παρακάτω πίνακα 3.1 φαίνονται όλα τα υποστηριζόμενα interrupts από το συγκεκριμένο μικροελεγκτή. Να σημειωθεί ότι όσο πιο μικρή αριθμητικά είναι η διεύθυνση εξυπηρέτησης, τόσο μεγαλύτερη είναι και η προτεραιότητα εξυπηρέτησης του αντίστοιχου interrupt. Για παράδειγμα, αν συμβούν ταυτόχρονα τα INT0 και TIMER1 OVF, πρώτα θα εξυπηρετηθεί το INT0, θα εκτελεστεί μια εντολή και κατόπιν θα εξυπηρετηθεί το TIMER1 OVF. Παρακάτω περιγράφονται συνοπτικά το καθένα από αυτά.

| Interrupt Vector # | Διεύθυνση Εξυπηρέτησης | Πηγή του interrupt |
|---|---|---|
| 1 | $000 | RESET |
| 2 | $001 | INT0 |
| 3 | $002 | INT1 |
| 4 | $003 | TIMER1 CAPT |
| 5 | $004 | TIMER1 COMPA |
| 6 | $005 | TIMER1 C0MPB |
| 7 | $006 | TIMER1 OVF |
| 8 | $007 | TIMER0 OVF |
| 9 | $008 | SPI, STC |
| 10 | $009 | UART, RX |
| 11 | $00A | UART, UDRE |
| 12 | $00B | UART, TX |
| 13 | $00C | ANA_COMP |

Πίνακας 3.1 – Υποστηριζόμενα interrupts.

- RESET: Ενεργοποιείται όταν κάνουμε reset ή γίνει reset από τον Watchdog timer.

- INT0: Ενεργοποιείται όταν στο PIND0 του μικροελεγκτή υπάρξει μετάβαση από θετικό σε αρνητικό παλμό ή μετάβαση από αρνητικό σε θετικό παλμό ή μόνιμο αρνητικό παλμό, ανάλογα με την επιλογή που έχει γίνει.
- INT1: Ενεργοποιείται όταν στο PIND1 του μικροελεγκτή υπάρξει μετάβαση από θετικό σε αρνητικό παλμό ή μετάβαση από αρνητικό σε θετικό παλμό ή μόνιμο αρνητικό παλμό, ανάλογα με την επιλογή που έχει γίνει.
- TIMER1 CAPT: Ενεργοποιείται όταν στο ICP του μικροελεγκτή υπάρξει μετάβαση από θετικό σε αρνητικό παλμό ή μετάβαση από αρνητικό σε θετικό παλμό, ανάλογα με την επιλογή που έχει γίνει και γίνεται μεταφορά της τιμής του Timer/Counter1 στον ICR1 IO καταχωρητή.
- TIMER1 COMPA: Ενεργοποιείται όταν η τιμή του Timer/Counter1 γίνει ίδια με αυτή που έχει επιλέξει ο χρήστης μέσα στον TCCR1A IO καταχωρητή.
- TIMER1 COMPB: Ενεργοποιείται όταν η τιμή του Timer/Counter1 γίνει ίδια με αυτή που έχει επιλέξει ο χρήστης μέσα στον TCCR1B IO καταχωρητή.
- TIMER1 OVF: Ενεργοποιείται όταν η τιμή του Timer/Counter1 πάρει τη μέγιστη που μπορεί και στη συνέχεια μηδενιστεί.
- TIMER0 OVF: Ενεργοποιείται όταν η τιμή του Timer/Counter0 πάρει τη μέγιστη που μπορεί και στη συνέχεια μηδενιστεί.
- SPI, STC: Ενεργοποιείται όταν χρησιμοποιείται το SPI πρωτόκολλο επικοινωνίας και τερματιστεί η μετάδοση.
- UART, RX: Ενεργοποιείται όταν χρησιμοποιείται το UART του μικροελεγκτή και ολοκληρωθεί η υποδοχή των bits που μεταδίδονται σειριακά.
- UART, UDRE: Ενεργοποιείται όταν χρησιμοποιείται το UART του μικροελεγκτή και ο UDR καταχωρητής είναι άδειος, δηλαδή μπορεί να δεχτεί νέο χαρακτήρα προς μετάδοση.
- UART, TXC: Ενεργοποιείται όταν χρησιμοποιείται το UART του μικροελεγκτή και ολοκληρωθεί η σειριακή μετάδοση των bits.
- ANA_COMP: Ενεργοποιείται όταν χρησιμοποιείται ο Analog Comparator και, ανάλογα με την επιλογή του χρήστη, υπάρξει διαφορά ή ισότητα στις δύο συγκρινόμενες τιμές.

## 3.3 Τα modules του μικροελεγκτή AT90S8515.

Όπως είχε αναφερθεί και πιο πριν, ο AT90S8515 περιλαμβάνει πολλά modules, τα οποία μπορούν να χρησιμοποιηθούν για την επικοινωνία του με τον εξωτερικό κόσμο. Αυτά είναι Timer/Counters, Watchdog timer, SPI, UART, αναλογικό συγκριτή και 32 θύρες εισόδου/εξόδου. Παρακάτω, αναφέρονται πολύ συνοπτικά τα χαρακτηριστικά του καθενός.

Οι Timer/Counters που έχει ο μικροελεγκτής AT90S8515 είναι 2, ένας 8-bit και ένας 16-bit. Μεταξύ τους έχουν πολλές κοινές λειτουργίες, αλλά ο 16-bit έχει και κάποιες επιπλέον. Ο Timer/Counter0 είναι ένα μετρητής, ο οποίος μπορεί να μετράει από το 0 ως το 255 με δυνατότητα επιλογής της συχνότητας που θα αυξάνεται η τιμή του [9]. Ο χρήστης μπορεί δηλαδή, να επιλέξει ο μετρητής να αυξάνεται σε κάθε θετικό παλμό του ρολογιού ή να υποδιαιρέσει τη συχνότητα αυτή κατά 16, 64, 256, 1024 ή κάποια εξωτερική πηγή. Υπάρχει, επίσης, η δυνατότητα (εφόσον έχει ενεργοποιηθεί) να εκτελεστεί και το interrupt TIMER0 OVF. Ο Timer/Counter1 είναι ένας μετρητής, που μπορεί να μετράει από 0 ως 65535 με δυνατότητα επιλογής της συχνότητας που θα αυξάνεται η τιμή του [9]. Ο χρήστης μπορεί δηλαδή, να επιλέξει ο μετρητής να αυξάνεται σε κάθε θετικό παλμό του ρολογιού ή να υποδιαιρέσει τη συχνότητα αυτή κατά 16, 64, 256, 1024 ή κάποια εξωτερική πηγή. Επίσης, υπάρχουν διαθέσιμα 4 διαφορετικά interrupts που μπορούν να εξυπηρετηθούν από το μικροελεγκτή, το TIMER1 OVF, το TIMER1 CAPT και τα TIMER1 COMPA και TIMER1 COMPB. Να σημειωθεί ότι ο συγκεκριμένος μετρητής μπορεί να χρησιμοποιηθεί και ως 8, 9 ή 10 bit Pulse Width Modulator (PWM).

Ο Watchdog timer τρέχει στο 1 MHZ από εσωτερικό ταλαντωτή για τάση 5V και χρησιμοποιείται στην περίπτωση όπου ο μικροελεγκτής για οποιοδήποτε λόγο δεν ανταποκρίνεται [9]. Εφόσον έχει ενεργοποιηθεί, αναλαμβάνει να τον επανεκκινήσει εκτελώντας τον κώδικα από την αρχή (RESET interrupt).

Ακόμα, όπως αναφέρθηκε, υποστηρίζεται και το SPI, επιτρέποντας έτσι τη σύγχρονη επικοινωνία μεταξύ του μικροελεγκτή και άλλων περιφερειακών συσκευών ή και μεταξύ μικροελεγκτών [9]. Το AT90S8515 SPI χρησιμοποιεί full duplex σύγχρονη μετάδοση δεδομένων. Επίσης μπορεί να λειτουργήσει σε τύπο λειτουργίας αφέντη ή σκλάβου (master/slave). Η μετάδοση των δεδομένων μπορεί να γίνει μέχρι και 5Mbit/sec και μπορεί να ξεκινήσει με το LSB ή με το MSB. Ακόμα, όπως έχει αναφερθεί και πιο πριν, υπάρχει ειδικό interrupt (SPI, STC) που μπορεί να εξυπηρετηθεί όταν τελειώσει κάποια μετάδοση, ενώ αξίζει

να σημειωθεί ότι περιλαμβάνεται και η δυνατότητα ο μικροελεγκτής να «ξυπνήσει» από κάποια κατάσταση λειτουργίας για εξοικονόμηση ενέργειας και να ξεκινήσει η μετάδοση δεδομένων, γεγονός που όμως ισχύει μόνο όταν λειτουργεί σε κατάσταση σκλάβου.

Το UART έχει χαρακτηριστικά που κάνουν αρκετά ευέλικτο τον AT90S8515 στη λειτουργία και τη χρήση [9].Πιο συγκεκριμένα, υπάρχει γεννήτρια baud rate, η οποία επιτρέπει την επικοινωνία σε όλες τις ταχύτητες (από 2400bps μέχρι 115200bps). Επίσης, ο χρήστης έχει τη δυνατότητα να στέλνει 8 ή 9 bits δεδομένα κάθε φορά και μπορεί να αναγνωριστεί η παρουσία θορύβου κατά τη μετάδοση (Framing Error Detection). Ακόμα, υπάρχει ξεχωριστό bit που δείχνει αν έγινε σφάλμα στο start bit (False Start Bit Detection), καθώς επίσης 3 ξεχωριστά interrupts (UART TCX, UART UDRE και UART RX) που μπορούν να εξυπηρετηθούν.

Ο αναλογικός συγκριτής βρίσκεται εντός του ολοκληρωμένου κυκλώματος και μπορεί να συγκρίνει τις εισόδους που εφαρμόζονται στα pins PB2 (AIN0) και PB3 (AIN1) [9]. Όταν η τάση στο AIN0 είναι μεγαλύτερη από αυτή στο AIN1, τότε η έξοδος του συγκριτή ACO (Analog Comparator Output) γίνεται 1, η οποία μπορεί να χρησιμοποιηθεί κατά τέτοιο τρόπο ώστε να ενεργοποιεί τη λειτουργία Input Capture Function του Timer/Counter1. Επίσης, υπάρχει η δυνατότητα της εξυπηρέτησης δικού του interrupt (ANALOG COMP), το οποίο μπορεί να ενεργοποιηθεί με 3 διαφορετικούς τρόπους (θετική έξοδος, αρνητική έξοδος, μετάβαση του σήματος στην ACO).

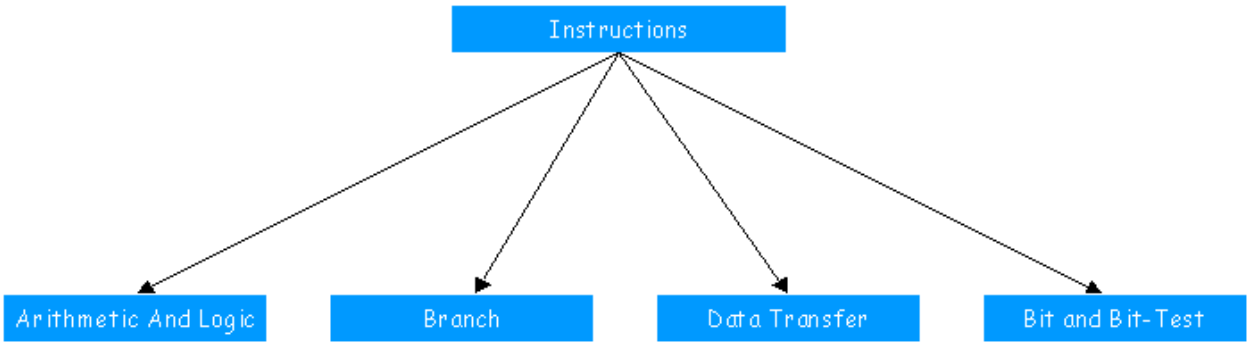
Τέλος, οι 32 θύρες εισόδου/εξόδου μπορούν να χρησιμοποιηθούν για την επικοινωνία του AT90S8515 με τον εξωτερικό κόσμο [9]. Τα ονόματά τους είναι PORTA, PORTB, PORTC και PORTD και σε αρκετές περιπτώσεις κάποια pins εξυπηρετούν παραπάνω από μία λειτουργίες. Το PORTB χρησιμοποιείται για τον αναλογικό συγκριτή και το SPI, τα PORTA και PORTC για την επικοινωνία του μικροελεγκτή με την εξωτερική μνήμη (στην περίπτωση που χρησιμοποιείται) και το PORTD για τα εξωτερικά interrupts INT0, INT1, καθώς και για το UART.

## 3.4 Υποστηριζόμενες εντολές

Ο AT90S8515 υποστηρίζει μια πληθώρα εντολών, οι οποίες μπορούν να χωριστούν σε 4 μεγάλες κατηγορίες [10]. Η $1^{η}$ είναι αυτή που περιλαμβάνει εντολές για αριθμητικές και λογικές πράξεις, η $2^{η}$ είναι αυτή που περιλαμβάνει εντολές διακλάδωσης, η $3^{η}$ είναι αυτή που

περιλαμβάνει εντολές μεταφοράς δεδομένων και η 4η είναι αυτή που περιλαμβάνει εντολές που εξετάζουν την τιμή ορισμένων bit σε κάποιους καταχωρητές. Η διάκριση αυτή φαίνεται στο σχήμα 3.7, ενώ στο 3.8 φαίνονται συνοπτικά όλες οι υποστηριζόμενες εντολές.

Στη στήλη **Mnemonic** φαίνεται η ονομασία της εντολής, στην **Operands** φαίνονται οι τελεστέοι που χρησιμοποιούνται, στην **Description** γίνεται μια σύντομη περιγραφή της λειτουργίας της, στην **Operation** φαίνεται το αποτέλεσμα της εντολής, στην **Flags** φαίνεται ποια bits του IO καταχωρητή SREG (Status REGister) επηρεάζονται και τέλος στην **Cycles** φαίνονται οι κύκλοι που χρειάζονται για την ολοκλήρωση της εντολής.




Σχήμα 3.7 – Διαχωρισμός σε ομάδες των υποστηριζόμενων εντολών.

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|---|---|---|---|---|---|
| ADD | Rd,Rr | Add without Carry | Rd = Rd + Rr | Z,C,N,V,H,S | 1 |
| ADC | Rd,Rr | Add with Carry | Rd = Rd + Rr + C | Z,C,N,V,H,S | 1 |
| ADIW | Rd, K | Add Immediate To Word | Rd+1 : Rd , K | Z,C,N,V,S | 2 |
| SUB | Rd,Rr | Subtract without Carry | Rd = Rd - Rr | Z,C,N,V,H,S | 1 |
| SUBI | Rd,K8 | Subtract Immediate | Rd = Rd - K8 | Z,C,N,V,H,S | 1 |
| SBC | Rd,Rr | Subtract with Carry | Rd = Rd - Rr - C | Z,C,N,V,H,S | 1 |
| SBCI | Rd,K8 | Subtract with Carry Immedtiate | Rd = Rd - K8 - C | Z,C,N,V,H,S | 1 |
| AND | Rd,Rr | Logical AND | Rd = Rd · Rr | Z,N,V,S | 1 |
| ANDI | Rd,K8 | Logical AND with Immediate | Rd = Rd · K8 | Z,N,V,S | 1 |
| OR | Rd,Rr | Logical OR | Rd = Rd V Rr | Z,N,V,S | 1 |
| ORI | Rd,K8 | Logical OR with Immediate | Rd = Rd V K8 | Z,N,V,S | 1 |
| EOR | Rd,Rr | Logical Exclusive OR | Rd = Rd EOR Rr | Z,N,V,S | 1 |
| COM | Rd | One's Complement | Rd = $FF - Rd | Z,C,N,V,S | 1 |
| NEG | Rd | Two's Complement | Rd = $00 - Rd | Z,C,N,V,H,S | 1 |
| SBR | Rd,K8 | Set Bit(s) in Register | Rd = Rd V K8 | Z,C,N,V,S | 1 |
| CBR | Rd,K8 | Clear Bit(s) in Register | Rd = Rd · ($FF - K8) | Z,C,N,V,S | 1 |
| INC | Rd | Increment Register | Rd = Rd + 1 | Z,N,V,S | 1 |
| DEC | Rd | Decrement Register | Rd = Rd -1 | Z,N,V,S | 1 |
| TST | Rd | Test for Zero or Negative | Rd = Rd · Rd | Z,C,N,V,S | 1 |
| CLR | Rd | Clear Register | Rd = 0 | Z,C,N,V,S | 1 |
| SER | Rd | Set Register | Rd = $FF | None | 1 |
| SBIW | Rdl,K6 | Subtract Immediate from Word | Rdh:Rdl = Rdh:Rdl - K 6 | Z,C,N,V,S | 2 |

Arithmetic and Logic Instructions.

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|---|---|---|---|---|---|
| RJMP | k | Relative Jump | PC = PC + k +1 | None | 2 |
| IJMP | None | Indirect Jump to (Z) | PC = Z | None | 2 |
| RCALL | k | Relative Call Subroutine | STACK = PC+1, PC = PC + k + 1 | None | 3/4* |
| ICALL | None | Indirect Call to (Z) | STACK = PC+1, PC = Z | None | 3/4* |
| RET | None | Subroutine Return | PC = STACK | None | 4/5* |
| RETI | None | Interrupt Return | PC = STACK | I | 4/5* |
| CPSE | Rd,Rr | Compare, Skip if equal | if (Rd ==Rr) PC = PC 2 or 3 | None | 1/2/3 |
| CP | Rd,Rr | Compare | Rd -Rr | Z,C,N,V,H,S | 1 |
| CPC | Rd,Rr | Compare with Carry | Rd - Rr - C | Z,C,N,V,H,S | 1 |
| CPI | Rd,K8 | Compare with Immediate | Rd - K | Z,C,N,V,H,S | 1 |
| SBRC | Rr,b | Skip if bit in register cleared | if(Rr(b)==0) PC = PC + 2 or 3 | None | 1/2/3 |
| SBRS | Rr,b | Skip if bit in register set | if(Rr(b)==1) PC = PC + 2 or 3 | None | 1/2/3 |
| SBIC | P,b | Skip if bit in I/O register cleared | if(I/O(P,b)==0) PC = PC + 2 or 3 | None | 1/2/3 |
| SBIS | P,b | Skip if bit in I/O register set | if(I/O(P,b)==1) PC = PC + 2 or 3 | None | 1/2/3 |
| BRBC | s,k | Branch if Status flag cleared | if(SREG(s)==0) PC = PC + k + 1 | None | 1/2 |
| BRBS | s,k | Branch if Status flag set | if(SREG(s)==1) PC = PC + k + 1 | None | 1/2 |
| BREQ | k | Branch if equal | if(Z==1) PC = PC + k + 1 | None | 1/2 |
| BRNE | k | Branch if not equal | if(Z==0) PC = PC + k + 1 | None | 1/2 |
| BRCS | k | Branch if carry set | if(C==1) PC = PC + k + 1 | None | 1/2 |
| BRCC | k | Branch if carry cleared | if(C==0) PC = PC + k + 1 | None | 1/2 |
| BRSH | k | Branch if same or higher | if(C==0) PC = PC + k + 1 | None | 1/2 |
| BRLO | k | Branch if lower | if(C==1) PC = PC + k + 1 | None | 1/2 |
| BRMI | k | Branch if minus | if(N==1) PC = PC + k + 1 | None | 1/2 |
| BRPL | k | Branch if plus | if(N==0) PC = PC + k + 1 | None | 1/2 |
| BRGE | k | Branch if greater than or equal (signed) | if(S==0) PC = PC + k + 1 | None | 1/2 |

| | | | | | |
|---|---|---|---|---|---|
| BRLT | k | Branch if less than (signed) | if(S==1) PC = PC + k + 1 | None | 1/2 |
| BRHS | k | Branch if half carry flag set | if(H==1) PC = PC + k + 1 | None | 1/2 |
| BRHC | k | Branch if half carry flag cleared | if(H==0) PC = PC + k + 1 | None | 1/2 |
| BRTS | k | Branch if T flag set | if(T==1) PC = PC + k + 1 | None | 1/2 |
| BRTC | k | Branch if T flag cleared | if(T==0) PC = PC + k + 1 | None | 1/2 |
| BRVS | k | Branch if overflow flag set | if(V==1) PC = PC + k + 1 | None | 1/2 |
| BRVC | k | Branch if overflow flag cleared | if(V==0) PC = PC + k + 1 | None | 1/2 |
| BRIE | k | Branch if interrupt enabled | if(I==1) PC = PC + k + 1 | None | 1/2 |
| BRID | k | Branch if interrupt disabled | if(I==0) PC = PC + k + 1 | None | 1/2 |

Branch Instructions.

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|---|---|---|---|---|---|
| MOV | Rd,Rr | Copy register | Rd = Rr | None | 1 |
| LDI | Rd,K8 | Load Immediate | Rd = K | None | 1 |
| LDS | Rd,k | Load Direct | Rd = (k) | None | 2* |
| LD | Rd,X | Load Indirect | Rd = (X) | None | 2* |
| LD | Rd,X+ | Load Indirect and Post-Increment | Rd = (X), X=X+1 | None | 2* |
| LD | Rd,-X | Load Indirect and Pre-Decrement | X=X-1, Rd = (X) | None | 2* |
| LD | Rd,Y | Load Indirect | Rd = (Y) | None | 2* |
| LD | Rd,Y+ | Load Indirect and Post-Increment | Rd = (Y), Y=Y+1 | None | 2* |
| LD | Rd,-Y | Load Indirect and Pre-Decrement | Y=Y-1, Rd = (Y) | None | 2* |
| LDD | Rd,Y+q | Load Indirect with displacement | Rd = (Y+q) | None | 2* |
| LD | Rd,Z | Load Indirect | Rd = (Z) | None | 2* |
| LD | Rd,Z+ | Load Indirect and Post-Increment | Rd = (Z), Z=Z+1 | None | 2* |
| LD | Rd,-Z | Load Indirect and Pre-Decrement | Z=Z-1, Rd = (Z) | None | 2* |
| LDD | Rd,Z+q | Load Indirect with displacement | Rd = (Z+q) | None | 2* |
| STS | k,Rr | Store Direct | (k) = Rr | None | 2* |
| ST | X,Rr | Store Indirect | (X) = Rr | None | 2* |
| ST | X+,Rr | Store Indirect and Post-Increment | (X) = Rr, X=X+1 | None | 2* |
| ST | -X,Rr | Store Indirect and Pre-Decrement | X=X-1, (X)=Rr | None | 2* |
| ST | Y,Rr | Store Indirect | (Y) = Rr | None | 2* |
| ST | Y+,Rr | Store Indirect and Post-Increment | (Y) = Rr, Y=Y+1 | None | 2 |
| ST | -Y,Rr | Store Indirect and Pre-Decrement | Y=Y-1, (Y) = Rr | None | 2 |
| ST | Y+q,Rr | Store Indirect with displacement | (Y+q) = Rr | None | 2 |
| ST | Z,Rr | Store Indirect | (Z) = Rr | None | 2 |
| ST | Z+,Rr | Store Indirect and Post-Increment | (Z) = Rr, Z=Z+1 | None | 2 |
| ST | -Z,Rr | Store Indirect and Pre-Decrement | Z=Z-1, (Z) = Rr | None | 2 |
| ST | Z+q,Rr | Store Indirect with displacement | (Z+q) = Rr | None | 2 |
| LPM | None | Load Program Memory | R0 = (Z) | None | 3 |
| IN | Rd,P | In Port | Rd = P | None | 1 |
| OUT | P,Rr | Out Port | P = Rr | None | 1 |
| PUSH | Rr | Push register on Stack | STACK = Rr | None | 2 |
| POP | Rd | Pop register from Stack | Rd = STACK | None | 2 |

Data Transfer Instructions.

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|---|---|---|---|---|---|
| LSL | Rd | Logical shift left | Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7) | Z,C,N,V,H,S | 1 |
| LSR | Rd | Logical shift right | Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0) | Z,C,N,V,S | 1 |
| ROL | Rd | Rotate left through carry | Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7) | Z,C,N,V,H,S | 1 |
| ROR | Rd | Rotate right through carry | Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0) | Z,C,N,V,S | 1 |
| ASR | Rd | Arithmetic shift right | Rd(n)=Rd(n+1), n=0,...,6 | Z,C,N,V,S | 1 |
| SWAP | Rd | Swap nibbles | Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0) | None | 1 |
| BSET | s | Set flag | SREG(s) = 1 | SREG(s) | 1 |
| BCLR | s | Clear flag | SREG(s) = 0 | SREG(s) | 1 |
| SBI | P,b | Set bit in I/O register | I/O(P,b) = 1 | None | 2 |
| CBI | P,b | Clear bit in I/O register | I/O(P,b) = 0 | None | 2 |
| BST | Rr,b | Bit store from register to T | T = Rr(b) | T | 1 |
| BLD | Rd,b | Bit load from register to T | Rd(b) = T | None | 1 |
| SEC | None | Set carry flag | C =1 | C | 1 |
| CLC | None | Clear carry flag | C = 0 | C | 1 |
| SEN | None | Set negative flag | N = 1 | N | 1 |
| CLN | None | Clear negative flag | N = 0 | N | 1 |
| SEZ | None | Set zero flag | Z = 1 | Z | 1 |
| CLZ | None | Clear zero flag | Z = 0 | Z | 1 |
| SEI | None | Set interrupt flag | I = 1 | I | 1 |
| CLI | None | Clear interrupt flag | I = 0 | I | 1 |
| SES | None | Set signed flag | S = 1 | S | 1 |
| CLN | None | Clear signed flag | S = 0 | S | 1 |
| SEV | None | Set overflow flag | V = 1 | V | 1 |
| CLV | None | Clear overflow flag | V = 0 | V | 1 |
| SET | None | Set T-flag | T = 1 | T | 1 |
| CLT | None | Clear T-flag | T = 0 | T | 1 |
| CLH | None | Clear half carry flag | H = 0 | H | 1 |
| NOP | None | No operation | None | None | 1 |
| SLEEP | None | Sleep | See instruction manual | None | 1 |
| WDR | None | Watchdog Reset | See instruction manual | None | 1 |

Bit and Bit-Test Instructions.

Σχήμα 3.8 – Υποστηριζόμενες εντολές από τον AT90S8515.

# 4 Το μοντέλο VHDL του AT90S8515 της Atmel.

Τα τελευταία χρόνια έχουν δημιουργηθεί αρκετές γλώσσες περιγραφής υλικού, με κάποιες να ξεχωρίζουν. Μία από αυτές είναι η VHDL, της οποίας το πρότυπο περιγράφτηκε αρχικά το 1987. Το 1993 προστέθηκαν και κάποια επιπλέον στοιχεία στη γλώσσα, κάνοντας τον τρόπο σύνταξης πιο «σταθερό» και ολοκληρωμένο [3, 4, 40]. Σήμερα, η VHDL είναι μία από τις ισχυρότερες γλώσσες περιγραφής υλικού και πολλές εταιρίες (για παράδειγμα Xilinx [30], Altera [31] και Mentor Graphics [34]) έχουν αναπτύξει σχεδιαστικά εργαλεία για αυτήν (CAD tools).

Η VHDL έχει αρκετές ιδιομορφίες σε σχέση με τις υπόλοιπες γλώσσες [3, 4]. Ένα από τα σημαντικότερα χαρακτηριστικά της είναι η παράλληλη εκτέλεση κομματιών κώδικα, όπως ακριβώς συμβαίνει και με το πραγματικό hardware. Ο χρήστης, δηλαδή, έχει τη δυνατότητα να περιγράψει κάποια σχεδίαση, της οποίας ορισμένα κομμάτια λειτουργούν ταυτόχρονα και μάλιστα να μπορούν να επικοινωνούν μεταξύ τους. Φυσικά υπάρχουν και τα κομμάτια κώδικα που εκτελούνται σειριακά, τα οποία βρίσκονται μέσα στις λεγόμενες διαδικασίες (processes).

Επίσης, η VHDL είναι μια γλώσσα στην οποία χρησιμοποιείται πολύ η ιεραρχία.. Πιο συγκεκριμένα, είναι δυνατόν να σχεδιαστεί κάποιο κύκλωμα με συγκεκριμένες εισόδους και εξόδους και κατόπιν αυτό να χρησιμοποιηθεί σαν «μαύρο κουτί» σε κάποια μεγαλύτερη σχεδίαση. Έτσι, ξεκινώντας από τα πολύ απλά κομμάτια ενός κυκλώματος, μπορεί κανείς να καταλήξει στην ολοκλήρωση πολύπλοκων σχεδιάσεων σχετικά γρήγορα.

Οι τρόποι που μπορεί να χρησιμοποιηθεί η VHDL κατά τη διάρκεια της σχεδίασης είναι ουσιαστικά 2. Ο πρώτος είναι ο behavioral όπου με σειριακό κώδικα (process) περιγράφεται ο τρόπος λειτουργίας ενός κυκλώματος. Μια process μπορεί να έχει τη λεγόμενη sensitivity list, μπορεί και όχι. Η τελευταία ουσιαστικά είναι το σύνολο των σημάτων, τα οποία θα την ενεργοποιήσουν. Δηλαδή, η αλλαγή της τιμής ενός και μόνο σήματος από τη sensitivity list είναι αρκετή για να ενεργοποιήσει τη process. Έτσι σχεδιάζονται διάφορα συνδυαστικά κυκλώματα. Αν μια process δεν έχει sensitivity list, τότε υπάρχει το σήμα του ρολογιού που την ενεργοποιεί και έτσι σχεδιάζονται σύγχρονα κυκλώματα. Συνήθως χρησιμοποιούνται εντολές και μεταβλητές παρόμοιες με αυτές στις γλώσσες λογισμικού (if-then-else, case, variables). Ο δεύτερος τρόπος είναι ο structural όπου περιγράφεται η ένωση μεταξύ μικρότερων κομματιών υλικού για τη δημιουργία ενός ενιαίου.

Σαν ένα τελευταίο από τα βασικά στοιχεία της VHDL αξίζει να αναφερθεί η χρήση των σημάτων (signals) και μεταβλητών (variables). Τα πρώτα συμπεριφέρονται όπως ακριβώς και οι πραγματικοί αγωγοί, δηλαδή θα παρουσιάσουν μια καθυστέρηση μέχρι να πάρουν την τιμή που τους ανατίθεται κάθε φορά και μπορούν να χρησιμοποιηθούν σε οποιοδήποτε μέρος του κώδικα. Οι μεταβλητές, αντίθετα, παίρνουν ακαριαία την τιμή που τους ανατίθεται και μπορούν να χρησιμοποιηθούν μόνο μέσα σε process. Δεν είναι δυνατόν, δηλαδή, να υπάρξει οποιαδήποτε επικοινωνία μεταξύ κάποιων κομματιών κώδικα που εκτελούνται παράλληλα με τη χρήση μεταβλητών, παρά μόνο με τη χρήση σημάτων.

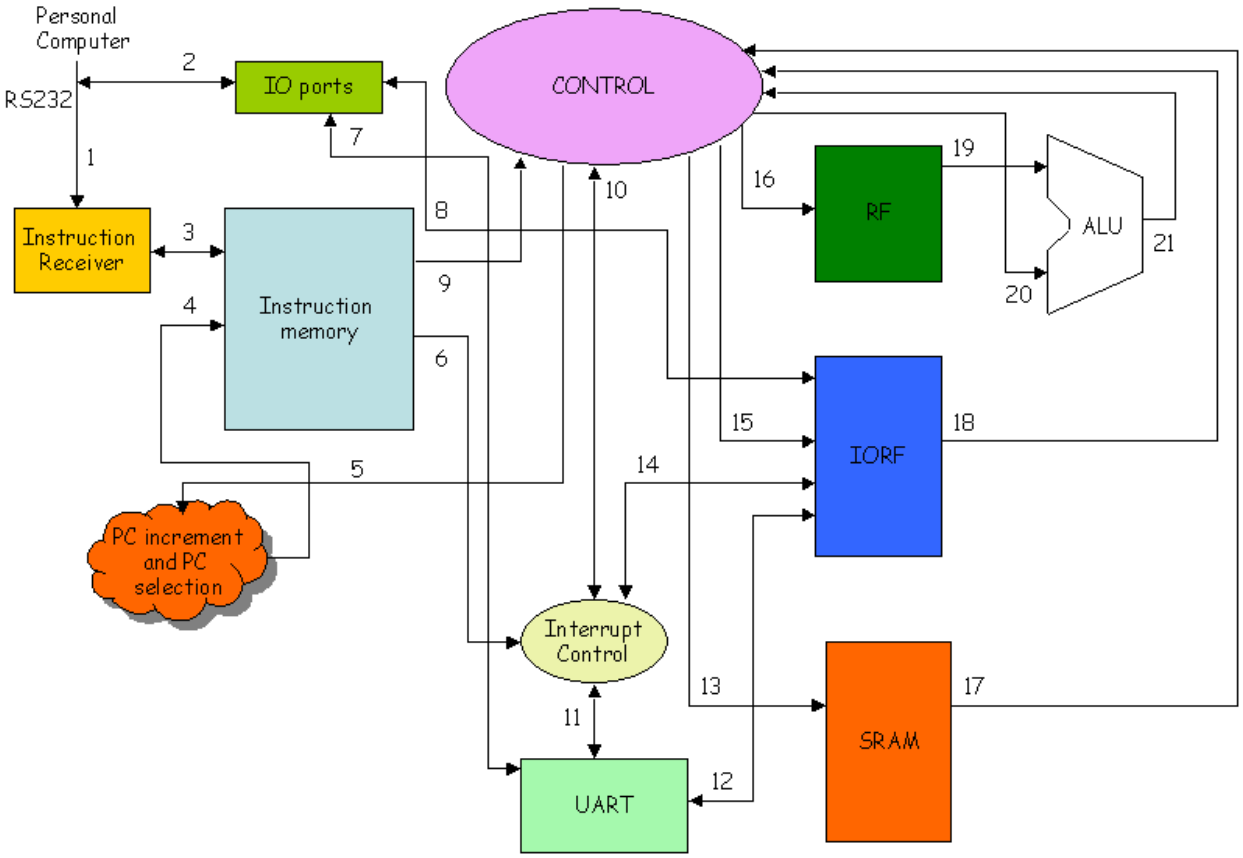
Χρησιμοποιώντας λοιπόν τη VHDL, αποφασίστηκε η σχεδίαση ενός μοντέλου του μικροελεγκτή AT90S8515 της Atmel, το οποίο περιλαμβάνει ορισμένα χαρακτηριστικά του τελευταίου. Πιο συγκεκριμένα, για τη σχεδίαση και την εξομοίωση λειτουργίας χρησιμοποιήθηκε το πρόγραμμα Modelsim 5.5d SE της Mentor Graphics [34], ενώ για την υλοποίηση το ISE 4.2i της Xilinx σε συνδυασμό με το FPGA Express της Synopsys [30]. Στη συνέχεια του κεφαλαίου αναλύεται ολόκληρη η σχεδίαση του μοντέλου.

## 4.1 Γενική εικόνα του μοντέλου.

Όπως αναφέρθηκε και πιο πριν, η ιεραρχία στη VHDL βοηθάει πολύ στη σχεδίαση πολύπλοκων κυκλωμάτων, ξεκινώντας από τα απλούστερα. Προκειμένου, λοιπόν, να γίνει εφικτή η σχεδίαση του μοντέλου του μικροελεγκτή, είναι απαραίτητος ο καθορισμός των επιμέρους κομματιών που θα χρειαστεί να σχεδιαστούν, καθώς και ο τρόπος που θα συνδεθούν και θα επικοινωνούν μεταξύ τους.

Σαν μία πρώτη έκδοση της σχεδίασης, πέρα από τις εντολές που υποστηρίζονται σχεδόν όλες, αποφασίστηκε να σχεδιαστεί ένα UART [15] και ένα κύκλωμα για τον προγραμματισμό της μνήμης εντολών μέσω σειριακής από το PC, ενώ να υποστηρίζονται 9 από τα 13 interrupts. Με βάση τα παραπάνω γενικά χαρακτηριστικά, είναι απαραίτητο να σχεδιαστεί η μνήμη εντολών (instruction memory), η μνήμη δεδομένων (data memory), η RF, η IORF, μια ALU που να εκτελεί τις επιθυμητές αριθμητικές και λογικές πράξεις, ένα UART, ένα κύκλωμα για την εγγραφή της μνήμης εντολών σειριακά από το PC, το γενικό κύκλωμα ελέγχου και ο έλεγχος των interrupts. Όλα τα παραπάνω βέβαια, δεν είναι τα μόνα που χρειάστηκαν για την

ολοκλήρωση της σχεδίασης, απλά στο σημείο αυτό αναφέρονται τα πολύ βασικά μέρη της. Στο σχήμα 4.1 φαίνεται ένα πολύ απλό block διάγραμμα της ιεραρχίας και του τρόπου σύνδεσης των επιμέρους κομματιών και ως σκοπό του έχει να δείξει τη γενική εικόνα της σχεδίασης του μοντέλου. Τα βελάκια αντιπροσωπεύουν ομάδες σημάτων που υπάρχουν για να επικοινωνούν τα επιμέρους κομμάτια μεταξύ τους. Στον πίνακα 4.1 περιγράφονται ποιες είναι οι ομάδες αυτές και ποια λειτουργία εξυπηρετούν. Να σημειωθεί ότι στο παράρτημα Β φαίνεται η ιεραρχία των συνδέσεων μεταξύ των κυκλωμάτων και στο παράρτημα Γ αναφέρονται τα ονόματα των σημάτων και πού συνδέεται το καθένα από αυτά.




Σχήμα 4.1 – Απλό block διάγραμμα του μοντέλου.

| Αριθμός βέλους | Περιγραφή της ομάδας |
|---|---|
| 1 | Σειριακή υποδοχή των δεδομένων από τον υπολογιστή στο κύκλωμα Instruction Receiver (παράγραφος 4.7). |
| 2 | Σειριακή αποστολή και υποδοχή των δεδομένων από τον υπολογιστή με το κύκλωμα UART (παράγραφος 4.3). |

| | |
|---|---|
| 3 | Εγγραφή των εντολών, που έρχονται σειριακά, στη μνήμη εντολών. |
| 4 | Έλεγχος του PC (Program Counter) για τη σωστή ανάγνωση των εντολών. |
| 5 | Επιλογή του καινούριου PC με βάση την εκάστοτε ροή του προγράμματος. |
| 6 | Έλεγχος της εντολής που πηγαίνει προς επεξεργασία, ανάλογα με το αν υπάρχει interrupt που περιμένει να εκτελεστεί ή όχι. |
| 7 | Μεταφορά των δεδομένων που ανταλλάσσονται σειριακά μεταξύ του UART και του υπολογιστή. |
| 8 | Έλεγχος της συμπεριφοράς των IO θυρών αν θα είναι είσοδοι ή έξοδοι. |
| 9 | Μεταφορά της εκάστοτε εντολής προς επεξεργασία. |
| 10 | Έλεγχος αν έχει προκύψει κάποιο Interrupt που περιμένει να εξυπηρετηθεί. |
| 11 | Έλεγχος της κατάστασης του UART, ώστε να είναι γνωστό αν και πότε θα υπάρξει κάποιο interrupt του πρώτου, που θα περιμένει να εξυπηρετηθεί. |
| 12 | Έλεγχος της κατάστασης του UART μέσω των IO καταχωρητών. |
| 13 | Ανάγνωση ή εγγραφή στη μνήμη δεδομένων. |
| 14 | Έλεγχος των Timer/Counters, ώστε να είναι γνωστό αν και πότε θα υπάρξει κάποιο interrupt δικό τους, που θα περιμένει να εξυπηρετηθεί. |
| 15 | Ανάγνωση ή εγγραφή στη μνήμη IO καταχωρητών. |
| 16 | Ανάγνωση ή εγγραφή στην RF. |
| 17 | Έξοδος της μνήμης δεδομένων προς τη μονάδα κεντρικού ελέγχου. |
| 18 | Έξοδος της μνήμης IO καταχωρητών προς τη μονάδα κεντρικού ελέγχου. |
| 19 | Έξοδος της RF προς τη μονάδα κεντρικού ελέγχου. |
| 20 | Έλεγχος της πράξης της ALU. |
| 21 | Έξοδος της ALU προς τη μονάδα κεντρικού ελέγχου. |

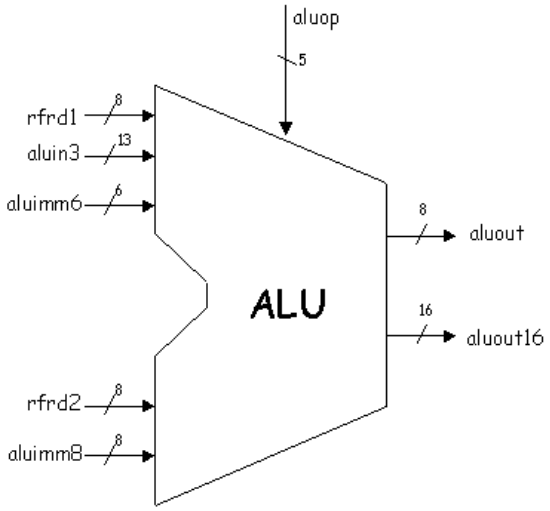
Πίνακας 4.1 – Περιγραφή των ομάδων των σημάτων.

## 4.2 Επιμέρους κομμάτια της σχεδίασης.

Στην παράγραφο αυτή παρουσιάζονται αναλυτικά όλα τα κομμάτια της σχεδίασης που χρησιμοποιήθηκαν για την ολοκλήρωσή της. Πιο συγκεκριμένα, αναφέρονται τα χαρακτηριστικά τους, ο τρόπος σκέψης που ακολουθήθηκε, καθώς και η λειτουργία τους.

### 4.2.1 Μονάδα αριθμητικών και λογικών πράξεων (ALU).

Προκειμένου να υποστηριχτούν όλες οι αριθμητικές και λογικές πράξεις που έχει ο AT90S8515, έπρεπε να σχεδιαστεί μια ALU με τις απαραίτητες εισόδους και εξόδους, οι οποίες φαίνονται στο σχήμα 4.2.




Σχήμα 4.2 – ALU.

Πρόκειται για ένα συνδυαστικό κύκλωμα [5] που σχεδιάστηκε με behavioral τρόπο στη VHDL, δηλαδή χρησιμοποιήθηκε μια process και μια sensitivity list για την περιγραφή του. Όταν αλλάξει η τιμή κάποιας εισόδου μετά από κάποια καθυστέρηση βγάζει και την αντίστοιχη έξοδο. Οι είσοδοι rfrd1 και rfrd2 έρχονται απευθείας από τις εξόδους της RF, ενώ οι υπόλοιπες έρχονται από η μονάδα γενικού ελέγχου του μοντέλου. Η aluin3 χρησιμοποιείται σε ορισμένες περιπτώσεις που χρειάζεται να γίνει κάποια πράξη με τελεστές που δεν προέρχονται από την RF, ενώ οι aluimm6 και aluimm8 χρησιμοποιούνται σε εντολές τύπου I-Format. Η είσοδος aluop είναι αυτή που περιγράφει ποια πράξη θα εκτελεστεί και στο πίνακα 4.2 φαίνονται η αντιστοιχία μεταξύ τιμών, πράξεων και ποια εντολή τη χρησιμοποιεί. Σε περίπτωση όπου θελήσει κανείς να υποστηρίζονται επιπλέον εντολές, μπορεί να χρησιμοποιήσει κάποια άδεια θέση ή να αυξήσει τον αριθμό των bits του aluop.

| Aluop (bits) | Περιγραφή πράξης | Εντολή που χρησιμοποιείται |
|---|---|---|
| 00000 | Οι έξοδοι γίνονται μηδέν | - |
| 00001 | Dout<=in1+in2 | Add |
| 00010 | dout<=in1+in2+”00000001” | Adc |
| 00011 | dout<=in1-in2 | Sub |
| 00100 | dout16<=(in1 & in2)+(“0000000000” & imm6) | Adiw |
| 00101 | dout<=in1 and in2 | And |
| 00110 | dout<=in1 and imm8 | Andi |
| 00111 | dout16<=”0000000” & in1(0) & in1(7) & in1(7 downto 1) | Asr |

| | | |
|---|---|---|
| 01000 | dout<=in1 + imm8 | Add with immediate |
| 01001 | Αν imm8(6)='1' τότε dout16<=("000" & in3) + ("111111111" & imm8(6 downto 0)), αλλιώς dout16<=("000" & in3) + ("000000000" & imm8(6 downto 0)) | PC+1+K |
| 01010 | dout<=imm8-in1 | 1's complement |
| 01011 | dout<=in1-in2-"00000001" | Sbc |
| 01100 | dout<=in1-imm8 | Subi |
| 01101 | dout<=in1 xor in2 | Eor |
| 01110 | dout16<=(in3(7 downto 0) & imm8) – "0000000000000010" | SPH:SPL-2 (icall) |
| 01111 | dout16<=(in2 & in1)+"0000000000000001" | ld Rd,X+, st X+,Rr |
| 10000 | dout16<=(in2 & in1)-"0000000000000001" | ld Rd,-X, st –X,Rr, ld Rd,-Y, st –Y,Rr, ld Rd,-Z, st –Z,Rr |
| 10001 | dout16<=(in2 & in1)+("00000000" & imm8) | ldd Rd,Y+q, std Y+q,Rr, ldd Rd,Z+q, std Z+q,Rr (0<=q<=63) |
| 10010 | dout<='0' & in1(7 downto 1) | Lsr |
| 10011 | dout<=in1 or in2 | Or |
| 10100 | dout<=in1 or imm8 | Ori |
| 10101 | dout16<=(in3(7 downto 0) & imm8) – "0000000000000001" | SPH:SPL-1 (push) |
| 10110 | Οι έξοδοι γίνονται μηδέν | - |
| 10111 | dout16<=(in3(7 downto 0) & imm8) + "0000000000000001" | SPH:SPL+1 (pop) |
| 11000 | dout16<="0000" & (in3(11 downto 0) + (imm8 & imm6(3 downto 0))) | rcall, rjmp |
| 11001 | dout16<=(in3(7 downto 0) & imm8) + "0000000000000010" | SPH:SPL+2 (ret) |
| 11010 | dout<='0' & in1(7 downto 1) | ror without carry |
| 11011 | dout<='1' & in1(7 downto 1) | ror with carry |
| 11100 | dout16<=(in1 & in2)-("0000000000" & imm6) | Sbiw |
| 11101 | dout<=in1 – imm8 – "00000001" | Sbci |
| 11110 | dout<=in1(3 downto 0) & in1(7 downto 4) | Swap |
| 11111 | dout16<="000" & (in3-"0000000000001") | PC+1-1 |

Πίνακας 4.2 – Πράξεις που υποστηρίζονται από την ALU.

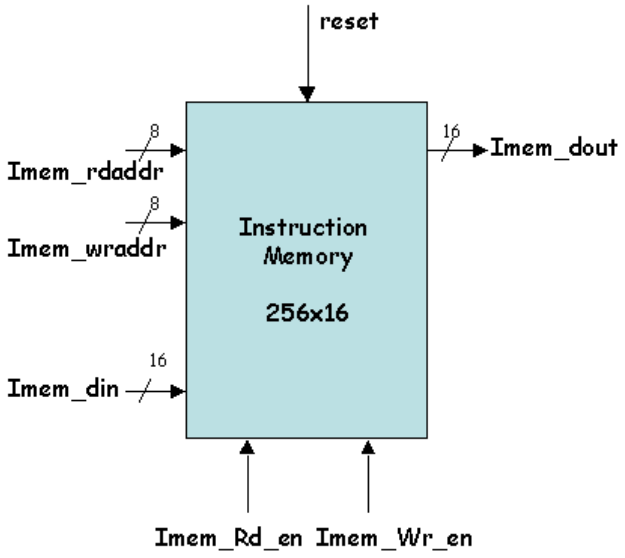
### 4.2.2 Μνήμες.

Σαν μνήμες αναφέρονται τα αποθηκευτικά στοιχεία που χρησιμοποιήθηκαν στη σχεδίαση. Πιο συγκεκριμένα, είναι η RF, η IORF, η μνήμη εντολών και η μνήμη δεδομένων. Σε όλες τις περιπτώσεις, εκτός από την IORF, χρησιμοποιήθηκε ένα συνδυαστικό κύκλωμα μνήμης, στο οποίο πρώτα θα πρέπει να εφαρμοστεί η διεύθυνση και τα δεδομένα εισόδου (αν πρόκειται για εγγραφή) και ύστερα να ενεργοποιηθεί το σήμα επίτρεψης, ενώ παράλληλα μπορούσε να

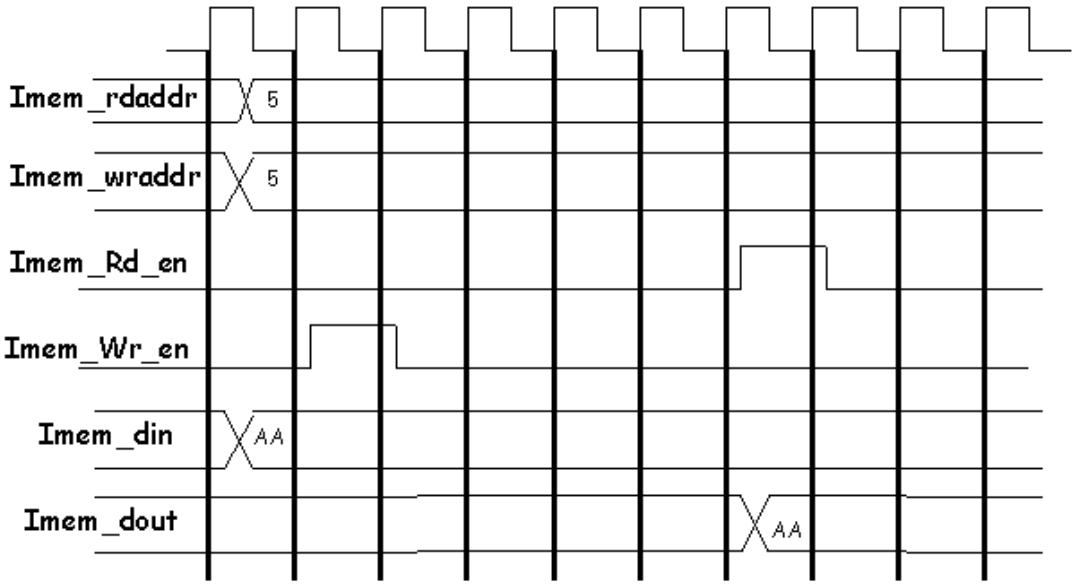
πάρει ως παραμέτρους το μέγεθος και το εύρος. Έτσι, ανάλογα με τις ανάγκες της κάθε μνήμης, το μόνο που χρειαζόταν ήταν να γίνουν οι αντίστοιχες αλλαγές στις παραμέτρους αυτές. Πιο συγκεκριμένα, είχε σχεδιαστεί μια παραμετροποιήσιμη μνήμη σε VHDL με behavioral τρόπο, χρησιμοποιώντας μια process με sensitivity list τις βασικές εισόδους της μνήμης, οπότε ανάλογα με τις ανάγκες της κάθε μνήμης, εισαγόντουσαν και τα αντίστοιχα μεγέθη (αρχεία **par_mem.vhd** και **par_mem_file.vhd**).

#### 4.2.2.1 Μνήμη εντολών.

Όπως αναφέρθηκε και πιο πριν, η μνήμη εντολών (στην 1η έκδοση) έχει 256 θέσεις επί 16 bit στην κάθε μία. Είναι ένα συνδυαστικό κύκλωμα, στο οποίο πρώτα θα πρέπει να εφαρμοστεί η διεύθυνση και τα δεδομένα εισόδου (αν πρόκειται για εγγραφή) και ύστερα να ενεργοποιηθεί το σήμα επίτρεψης και χρησιμοποιείται για την αποθήκευση των εντολών που πρόκειται να εκτελεστούν, καθώς και για σταθερές. Έχει ξεχωριστές εισόδους για τη διεύθυνση ανάγνωσης (imem_rdaddr) και εγγραφής (imem_wraddr), καθώς και αντίστοιχα σήματα που επιτρέπουν ανάγνωση (imem_rd_en) ή εγγραφή (imem_wr_en). Το reset αρχικοποιεί τα περιεχόμενα της μνήμης στο μηδέν, το imem_din είναι για την είσοδο των εντολών και το imem_dout είναι η έξοδός της. Στο σχήμα 4.3a φαίνεται η μνήμη εντολών με τις εισόδους και τις εξόδους της και στο σχήμα 4.3b φαίνεται μια κυματομορφή που δείχνει τον τρόπο λειτουργίας της.




Σχήμα 4.3a – Μνήμη εντολών (1η έκδοση).

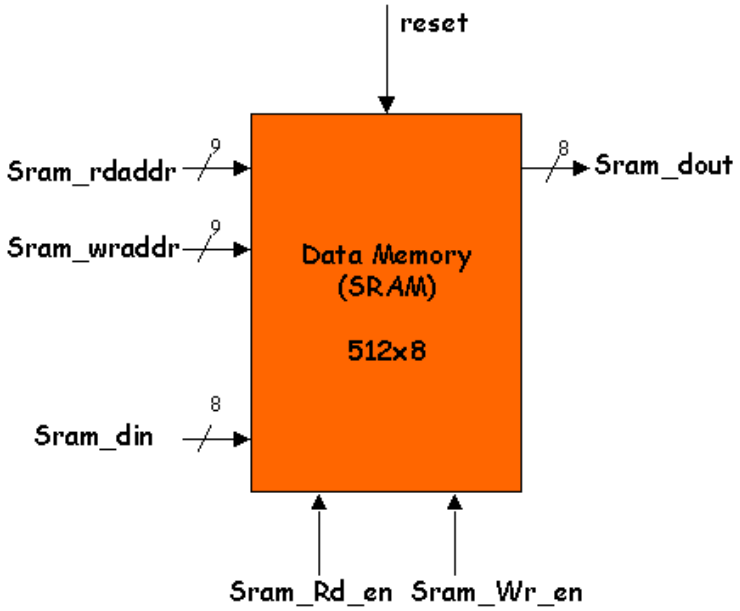



Σχήμα 4.3b – Λειτουργία της μνήμης εντολών.

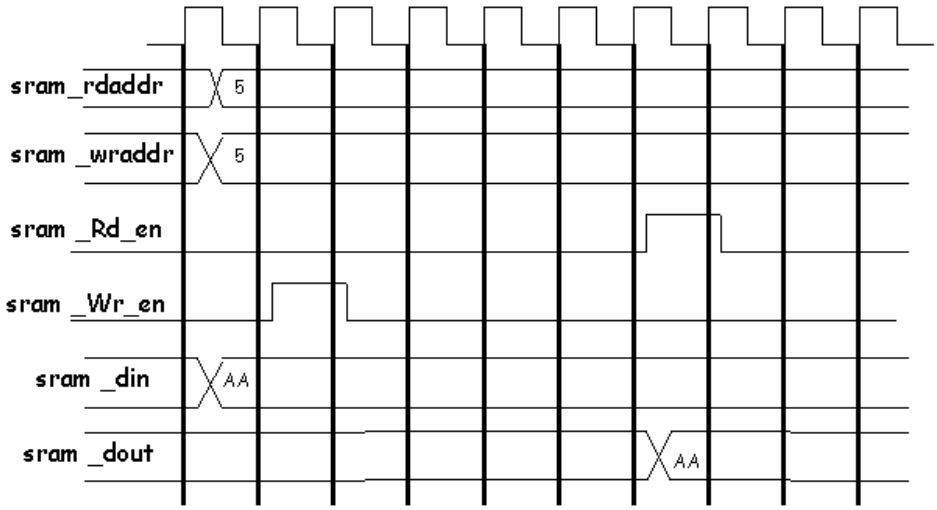
Για τη σχεδίαση της μνήμης εντολών σε VHDL, χρησιμοποιήθηκε η παραμετροποιήσιμη μνήμη που περιγράφτηκε στην εισαγωγή της παραγράφου 4.2.2 ως πυρήνας Αυτό που χρειάστηκε ήταν να σχεδιαστεί ένα top level αρχείο για να επιλεχτεί πλάτος 16-bit και μέγεθος 256 θέσεις, υλοποιώντας έτσι τη μνήμη εντολών (αρχεία **lpm_imem4096x16.vhd** – **par_mem_file.vhd** και **lpm256x16nofile.vhd** – **par_mem_file.vhd**).

### 4.2.2.2 Μνήμη δεδομένων.

Η μνήμη δεδομένων έχει 512 θέσεις επί 8 bit στην κάθε μία. Είναι ένα συνδυαστικό κύκλωμα, στο οποίο πρώτα θα πρέπει να εφαρμοστεί η διεύθυνση και τα δεδομένα εισόδου (αν πρόκειται για εγγραφή) και ύστερα να ενεργοποιηθεί το σήμα επίτρεψης και χρησιμοποιείται για δεδομένα. Έχει ξεχωριστές εισόδους για τη διεύθυνση ανάγνωσης (sram_rdaddr) και εγγραφής (sram_wraddr), καθώς και αντίστοιχα σήματα που επιτρέπουν ανάγνωση (sram_rd_en) ή εγγραφή (sram_wr_en). Το reset αρχικοποιεί τα περιεχόμενα της μνήμης στο μηδέν, το sram_din είναι για την είσοδο των δεδομένων και το sram_dout είναι η έξοδός της. Στο σχήμα 4.4a φαίνεται η μνήμη δεδομένων με τις εισόδους και τις εξόδους της και στο σχήμα 4.4b φαίνεται μια κυματομορφή που δείχνει τον τρόπο λειτουργίας της.




Σχήμα 4.4a – Μνήμη δεδομένων (SRAM).

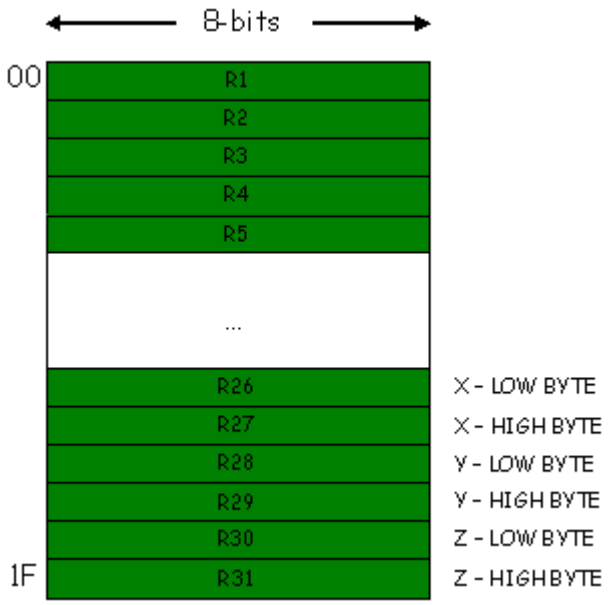



Σχήμα 4.4b – Λειτουργία της μνήμης δεδομένων.

Για τη σχεδίαση της μνήμης δεδομένων σε VHDL, χρησιμοποιήθηκε η παραμετροποιήσιμη μνήμη που περιγράφτηκε στην εισαγωγή της παραγράφου 4.2.2 ως πυρήνας. Αυτό που χρειάστηκε ήταν να σχεδιαστεί ένα top level αρχείο για να επιλεχτεί πλάτος 8-bit και μέγεθος 512 θέσεις, υλοποιώντας έτσι τη μνήμη δεδομένων (αρχεία **lpm_sram512x8.vhd – par_mem.vhd**).

### 4.2.2.3 Αρχείο καταχωρητών (RF).

Στο σχήμα 4.4a φαίνεται η δομή της RF [9] στον μικροελεγκτή AT90S8515. Συνολικά, αποτελείται από 32 θέσεις με 8 bit στην κάθε μία. Δύο βασικά χαρακτηριστικά της RF είναι ότι οι πρώτες 16 θέσεις δεν μπορούν να χρησιμοποιηθούν από εντολές τύπου I – Format (ldi, subi, sbci, cpi, andi, ori) και ότι οι τελευταίοι 6 8-bit καταχωρητές μπορούν να χρησιμοποιηθούν ως 3 16-bit καταχωρητές. Πιο συγκεκριμένα, υπάρχει ο X καταχωρητής που αποτελείται από τους R27:R26, ο Y καταχωρητής που αποτελείται από τους R29:R28 και ο Z καταχωρητής που αποτελείται από τους R31:R30. Η λειτουργία αυτή είναι ιδιαίτερα χρήσιμη στην περίπτωση όπου γίνεται προσπέλαση στην SRAM, μιας και οι X, Y, Z καταχωρητές μπορούν να χρησιμοποιηθούν ως 16-bit δείκτες ή όταν χρειάζεται να γίνει κάποιος υπολογισμός μεταξύ 16-bit αριθμών.



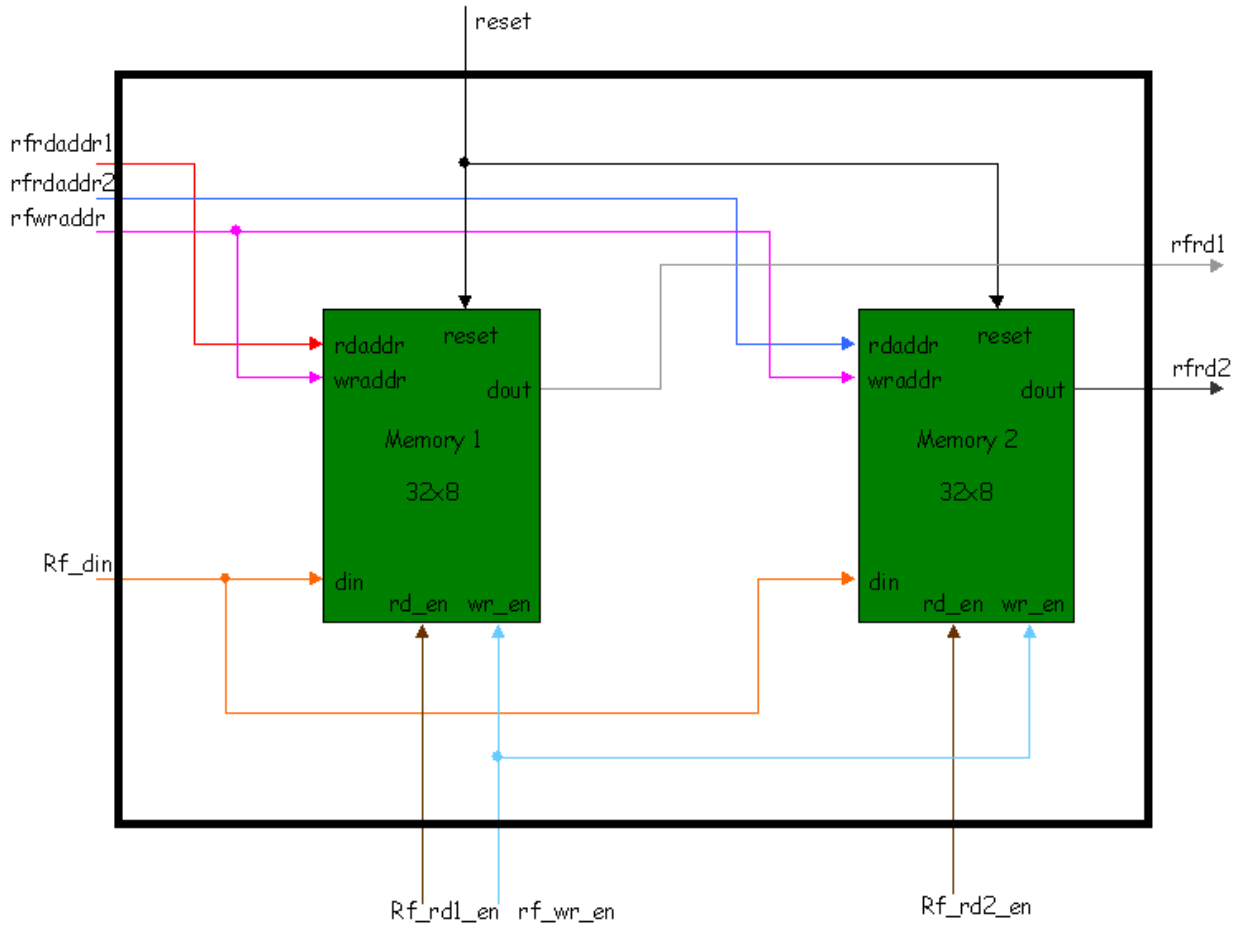

Σχήμα 4.4a – RF.

Σε ό,τι αφορά τη σχεδίαση της RF για το μοντέλο IP core, δε διαφέρει στον τρόπο λειτουργίας σε σχέση με τις υπόλοιπες μνήμες. Πρόκειται για ένα συνδυαστικό κύκλωμα, στο οποίο πρώτα θα πρέπει να εφαρμοστεί η διεύθυνση και τα δεδομένα εισόδου (αν πρόκειται για εγγραφή) και ύστερα να ενεργοποιηθεί το σήμα επίτρεψης και χρησιμοποιείται για τους

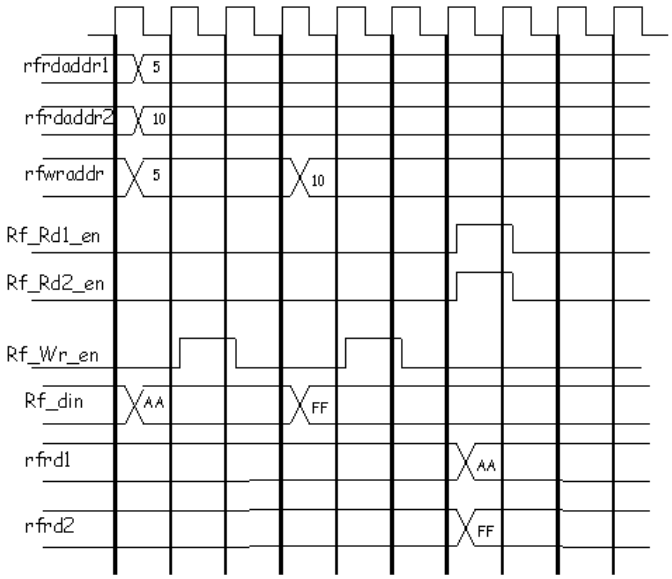
καταχωρητές. Η μόνη διαφορά είναι ότι την ίδια στιγμή είναι δυνατόν να διαβαστούν 2 θέσεις ταυτόχρονα, οπότε να εμφανιστούν στις εξόδους τα περιεχόμενα 2 καταχωρητών.

Για να γίνει αυτό εφικτό, εσωτερικά της RF υπάρχουν 2 μικρές μνήμες 32 θέσεων επί 8 bits στην κάθε θέση. Όταν γίνεται κάποια εγγραφή, τότε τα δεδομένα πάνε και στις 2 ταυτόχρονα, ώστε να έχουν πάντα ακριβώς τα ίδια περιεχόμενα. Όταν γίνεται κάποια ανάγνωση, τότε διαβάζονται ξεχωριστά τα περιεχόμενα της καθεμίας και εμφανίζονται στις αντίστοιχες εξόδους. Έτσι, υπάρχουν οι είσοδοι rfrdaddr1 και rfrdaddr2 για τους 2 καταχωρητές που πρέπει να διαβαστούν, η rfwraddr για να τον καταχωρητή που πρέπει να ενημερωθούν τα περιεχόμενά του, οι rf_rd1_en και rf_rd2_en που επιτρέπουν την ανάγνωση, η rf_wr_en που επιτρέπει την εγγραφή, η reset που αρχικοποιεί τα περιεχόμενα της RF στο μηδέν και η din για την είσοδο των νέων τιμών. Από εξόδους, υπάρχουν οι rfrd1 και rfrd2 για τους καταχωρητές που διαβάστηκαν. Όλα τα παραπάνω φαίνονται στο σχήμα 4.5b, ενώ στο 4.5c φαίνεται ο τρόπος λειτουργίας της RF.

Για τη σχεδίαση της μνήμης δεδομένων σε VHDL, χρησιμοποιήθηκε 2 φορές η παραμετροποιήσιμη μνήμη που περιγράφτηκε στην εισαγωγή της παραγράφου 4.2.2 ως πυρήνας. Αυτό που χρειάστηκε ήταν να σχεδιαστεί ένα top level αρχείο για να επιλεχτεί πλάτος 8-bit και μέγεθος 32 θέσεις και κατόπιν να ενωθούν όπως φαίνεται στο σχήμα 4.5b, υλοποιώντας έτσι την RF (αρχεία **lpm_rf32x8.vhd** – **par_mem.vhd**).



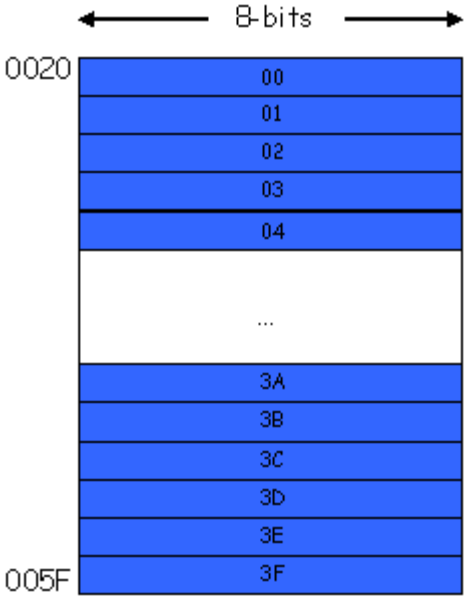

Σχήμα 4.5b – Δομή της RF.




Σχήμα 4.5c – Λειτουργία της RF.

### 4.2.2.4 Μνήμη IO καταχωρητών (IORF).

Στο σχήμα 4.6a φαίνεται η δομή της IORF [9] στον μικροελεγκτή AT90S8515. Συνολικά, αποτελείται από 64 θέσεις με 8-bit στην κάθε μία, όπου εκεί αποθηκεύονται οι τιμές των IO (Input/Output) καταχωρητών. Για να μπορέσει ο σχεδιαστής να χρησιμοποιήσει κάποιο κομμάτι του μικροελεγκτή (για παράδειγμα, UART, SPI, Ports, Timers), θα πρέπει να αποθηκεύσει και την κατάλληλη τιμή στον αντίστοιχο IO καταχωρητή. Όπως γίνεται εύκολα αντιληπτό, η IORF παίζει καθοριστικό ρόλο για τη σωστή λειτουργία του μικροελεγκτή, μιας και μέσα από αυτήν ο σχεδιαστής μπορεί να τον ελέγξει πλήρως.




Σχήμα 4.6a – IORF.

Σε ό,τι αφορά το μοντέλο IP core, το συγκεκριμένο κομμάτι της σχεδίασης είναι ένα από τα βασικότερα, καθώς περιλαμβάνει όλους τους IO καταχωρητές, με βάση τους οποίους ελέγχεται. Πρόκειται για ένα συνδυαστικό κύκλώμα, στο οποίο πρώτα θα πρέπει να εφαρμοστεί η διεύθυνση και τα δεδομένα εισόδου (αν πρόκειται για εγγραφή) και ύστερα να ενεργοποιηθεί το σήμα επίτρεψης και αποθηκεύονται οι τιμές των IO καταχωρητών για να μπορούν να διαβαστούν όταν χρειαστεί. Επίσης, περιλαμβάνει τον έλεγχο της εκκίνησης σειριακής μετάδοσης και την επιλογή του baud rate από το UART, τον έλεγχο των 2 Timer/Counter, καθώς και τον έλεγχο μερικών από τα υποστηριζόμενα interrupts. Όπως γίνεται αντιληπτό, η συγκεκριμένη μνήμη δεν έχει μόνο το σκοπό αποθήκευσης δεδομένων γενικά, αλλά εξυπηρετεί

και ένα μεγάλο κομμάτι του γενικού ελέγχου του μικροελεγκτή. Ο λόγος που συμβαίνει αυτό είναι ότι κάποιοι IO καταχωρητές δεν αλλάζουν μόνο όταν το επιθυμεί ο χρήστης, δηλαδή γράψει στην IO μνήμη, αλλά μπορεί να χρειαστεί να αλλάξουν επειδή συνέβη κάποιο γεγονός (για παράδειγμα, υπάρχει interrupt που περιμένει να εξυπηρετηθεί), οπότε θα πρέπει να γίνεται αυτόματα η ενημέρωση του αντίστοιχου IO καταχωρητή.

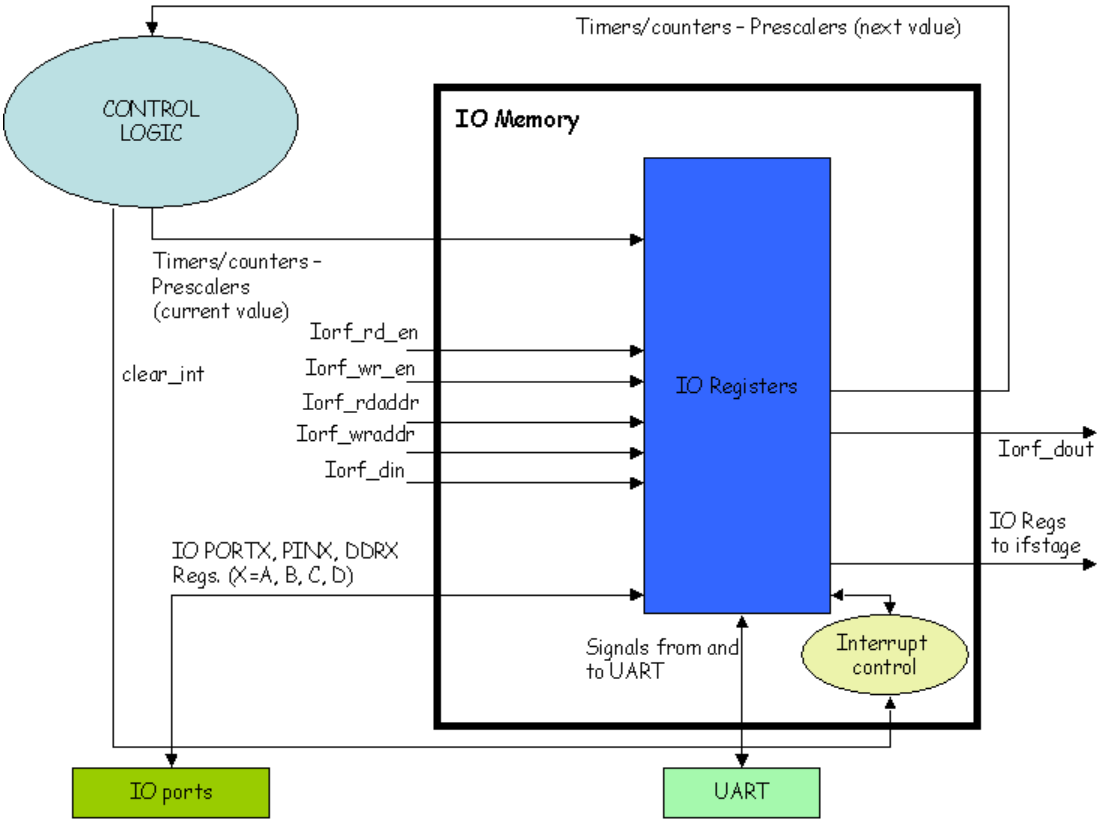
Κάποιοι από τους IO καταχωρητές του AT90S8515 δεν χρησιμοποιούνται, επειδή δεν έχουν συμπεριληφθεί ορισμένα κομμάτια του στη σχεδίαση (για παράδειγμα, IO καταχωρητές που έχουν σχέση με τον αναλογικό συγκριτή). Στον πίνακα 4.3 φαίνονται αυτοί που χρησιμοποιούνται από τη σχεδίαση και δίπλα η διεύθυνσή τους σε δεκαεξαδικό σύστημα.

| Όνομα IO καταχωρητή | Διεύθυνση (Hex) | Όνομα IO καταχωρητή | Διεύθυνση (Hex) |
|---|---|---|---|
| SREG | 3F | PORTA | 1B |
| SPH | 3E | DDRA | 1A |
| SPL | 3D | PINA | 19 |
| GIMSK | 3B | PORTB | 18 |
| GIFR | 3A | DDRB | 17 |
| TIMSK | 39 | PINB | 16 |
| TIFR | 38 | PORTC | 15 |
| MCUCR | 35 | DDRC | 14 |
| TCCR0 | 33 | PINC | 13 |
| TCNT0 | 32 | PORTD | 12 |
| TCCR1A | 2F | DDRD | 11 |
| TCCR1B | 2E | PIND | 10 |
| TCNT1H | 2D | UDR | 0C |
| TCNT1L | 2C | USR | 0B |
| OCR1AH | 2B | UCR | 0A |
| OCR1AL | 2A | UBRR | 09 |

Πίνακας 4.3 – IO καταχωρητές που χρησιμοποιούνται.

Το σχήμα 4.6b δείχνει ένα block διάγραμμα της IO μνήμης. Όπως φαίνεται, υπάρχουν αρκετές είσοδοι και έξοδοι (πέρα από τις βασικές για την εγγραφή και ανάγνωση στη μνήμη που είναι οι reset, iorf_rdaddr, iorf_wraddr, iorf_rd_en, iorf_wr_en, iorf_din, iorf_dout), γεγονός που οφείλεται στην ανάγκη για αυτόματη ενημέρωση των IO καταχωρητών. Πιο αναλυτικά, αρχικά φαίνονται τα σήματα των Timer/Counters και Prescalers που βγαίνουν από του IO καταχωρητές

και πάνε στο γενικό έλεγχο. Τα σήματα αυτά είναι η επόμενη τιμή τους (στην περίπτωση όπου χρησιμοποιούνται), η οποία ανατίθεται στον επόμενο κύκλο στους αντίστοιχους IO καταχωρητές. Ακόμα, οι τιμές των καταχωρητών SREG, GIMSK, GIFR, TIMSK και TIFR πάνε στη βαθμίδα ανάκλησης εντολών και χρησιμοποιούνται για την εξυπηρέτηση των interrupts. Κάποια σήματα χρειάζονται για την επικοινωνία του UART με τους IO καταχωρητές USR, UDR, UBRR και UCR ώστε οι τελευταίοι να ενημερώνονται όταν πρέπει, ενώ τα IO pins του μοντέλου έχουν άμεση σύνδεση με την RF. Θα πρέπει να διευκρινιστεί ότι οι IO καταχωρητές PINA, PINB, PINC και PIND δεν υπάρχουν ως κύκλωμα και όταν χρειαστεί να διαβαστεί η τιμή τους (δίνοντας την κατάλληλη IO διεύθυνση), γίνεται ανάγνωση των λογικών 1 ή 0 που υπάρχουν στα αντίστοιχα pins εκείνη τη στιγμή. Αξίζει επίσης να σημειωθεί ότι εσωτερικά της μνήμης IO καταχωρητών, υπάρχει ένα κομμάτι του ελέγχου εξυπηρέτησης όλων των εσωτερικών interrupts που υποστηρίζονται, το οποίο με τη βοήθεια του σήματος clear_int συνεργάζεται με τη μονάδα γενικού ελέγχου .




Σχήμα 4.6b – Block διάγραμμα IO μνήμης.

Για τη σχεδίαση της μνήμης IO καταχωρητών σε VHDL, χρησιμοποιήθηκε μια process με sensitivity list όλες τις εισόδους της μνήμης. Επίσης, χρησιμοποιήθηκε και μία σύγχρονη process (δηλαδή ενεργοποιείται σε κάθε θετική ακμή του ρολογιού) για τον έλεγχο του UART transmitter και πιο συγκεκριμένα για την επαλήθευση των συνθηκών που θα πρέπει να ισχύουν, αν είναι να ξεκινήσει να λειτουργεί (αρχείο **iorf_64x8_14_20.vhd**).

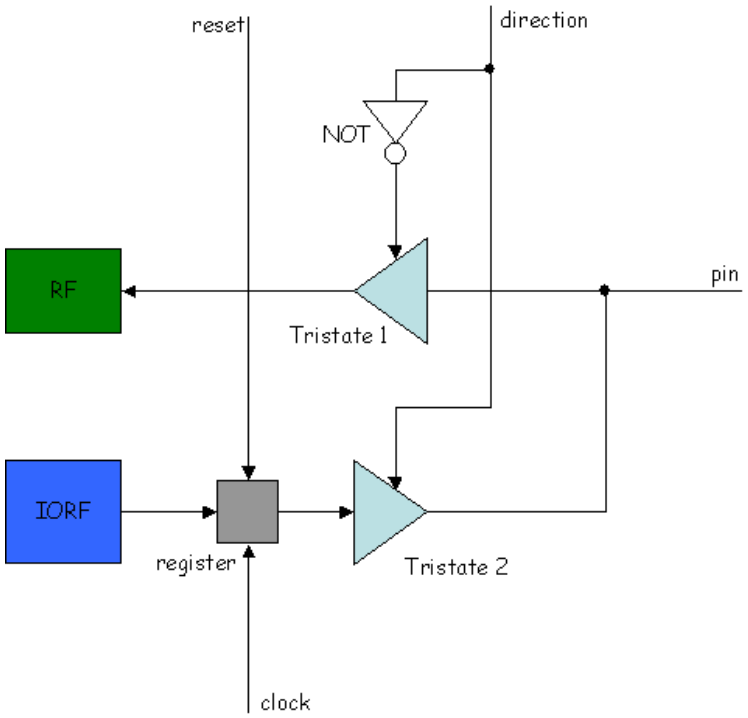
### 4.2.3 IO pins.

Ο AT90S8515 έχει 4 8-bit IO pins που χρησιμοποιούνται για την επικοινωνία του με τον εξωτερικό κόσμο. Προκειμένου λοιπόν να σχεδιαστούν σε VHDL, χρησιμοποιήθηκε η ειδική κατηγορία inout [3, 4, 8], η οποία επιτρέπει την αμφίδρομη λειτουργία των συγκεκριμένων pins, αλλά θα πρέπει να γίνεται σωστά η οδήγησή τους, γεγονός που επιβάλλει τη χρήση τρικατάστατων οδηγητών. Όταν το enable γίνει 1, τότε ο τρικατάστατος οδηγητής έχει το χαρακτηριστικό της διαφάνειας, δηλαδή ό,τι σήμα έρθει στην είσοδό του, το ίδιο θα περάσει στην έξοδο. Όταν όμως γίνει 0, τότε αποκόπτει τη σύνδεση και έτσι δημιουργείται η κατάσταση υψηλής εμπέδυσης.

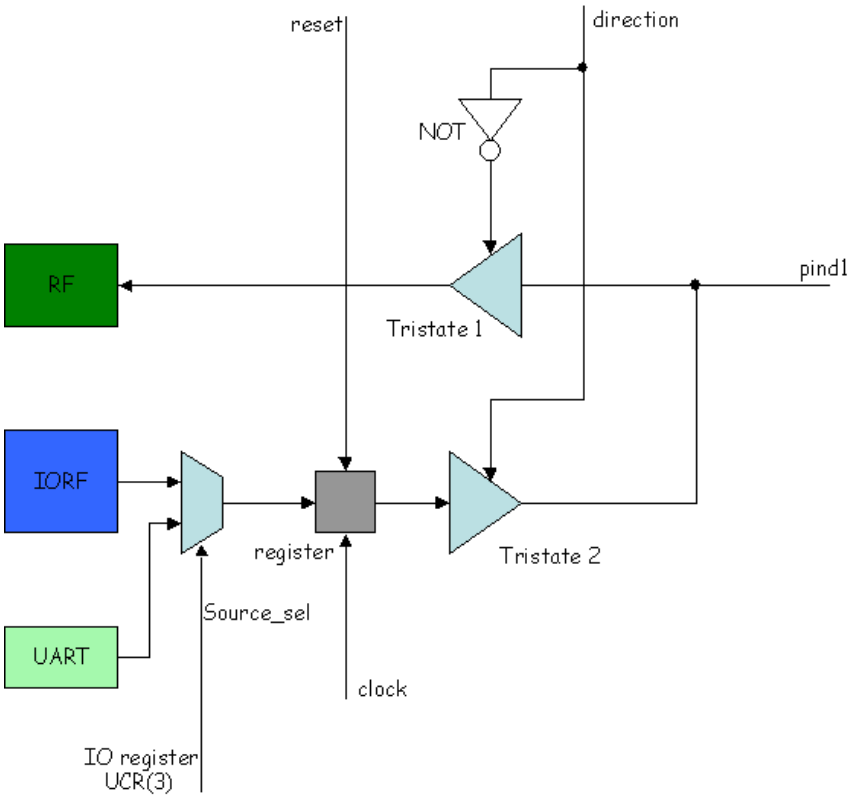
Στο συγκεκριμένο μικροελεγκτή, όταν τα IO pins του λειτουργούν ως είσοδοι, τότε πάνε απευθείας στη RF και όταν λειτουργούν ως έξοδοι παίρνουν τις τιμές τους από την IORF. Το κύκλωμα, επομένως, που χρειάζεται είναι τέτοιο που, με τη βοήθεια τρικατάστατων οδηγητών, τα pins να μπορούν να συνδέονται πότε με την RF και πότε με την IORF, με την προϋπόθεση όμως ότι είναι αμοιβαίως αποκλειόμενα. Στο σχήμα 4.7a φαίνεται το κύκλωμα αυτό για ένα pin. Όπως μπορεί να παρατηρηθεί, υπάρχει το σήμα direction που ενεργοποιεί ή όχι τους τρικατάστατους οδηγητές. Στον AT90S8515, όταν η τιμή του direction είναι 0, τότε τα pins λειτουργούν ως είσοδοι, ενώ όταν είναι 1 λειτουργούν ως έξοδοι. Έτσι, βάζοντας έναν αντιστροφέα πριν από το enable του τρικατάστατου οδηγητή 1, είναι σίγουρο πως όταν το σήμα direction πάρει την τιμή 0, το enable θα γίνει 1, άρα θα γίνει η σύνδεση μεταξύ των pins και της RF, ενώ συγχρόνως το enable του τρικατάστατου οδηγητή 2 θα γίνει 0, οπότε θα έχει αποκοπεί η σύνδεση μεταξύ των pins και της IORF. Να σημειωθεί ότι ο καταχωρητής παρεμβάλλεται για να περνούν τα δεδομένα στον επόμενο κύκλο στην έξοδο, όπως ακριβώς συμβαίνει και στον AT90S8515.

Στο σχήμα 4.7b φαίνεται το κύκλωμα που σχεδιάστηκε ειδικά για το pind1, το οποίο έχει και τη λειτουργία της αποστολής των χαρακτήρων σειριακά από το UART στον εξωτερικό κόσμο. Η διαφορά του σε σχέση με το προηγούμενο είναι ο επιπλέον πολυπλέκτης που υπάρχει μεταξύ του UART, IORF και του τρικατάστατου οδηγητή 2. Το σήμα με βάση που επιλέγει από πού θα έρχονται τα δεδομένα προς την έξοδο είναι το source_sel, το οποίο είναι 1 όταν είναι ενεργοποιημένο ο Transmitter του UART και 0 όταν δεν είναι. Έτσι, όταν χρησιμοποιείται ο Transmitter, τότε τα δεδομένα που στέλνει περνάνε στην έξοδο, αλλιώς η σύνδεση παραμένει μεταξύ IORF και pind1. Όπως και στον AT90S8515, έτσι και εδώ δεν είναι δυνατή η χρήση του pind1 από το UART και ταυτόχρονα ως IO pin.

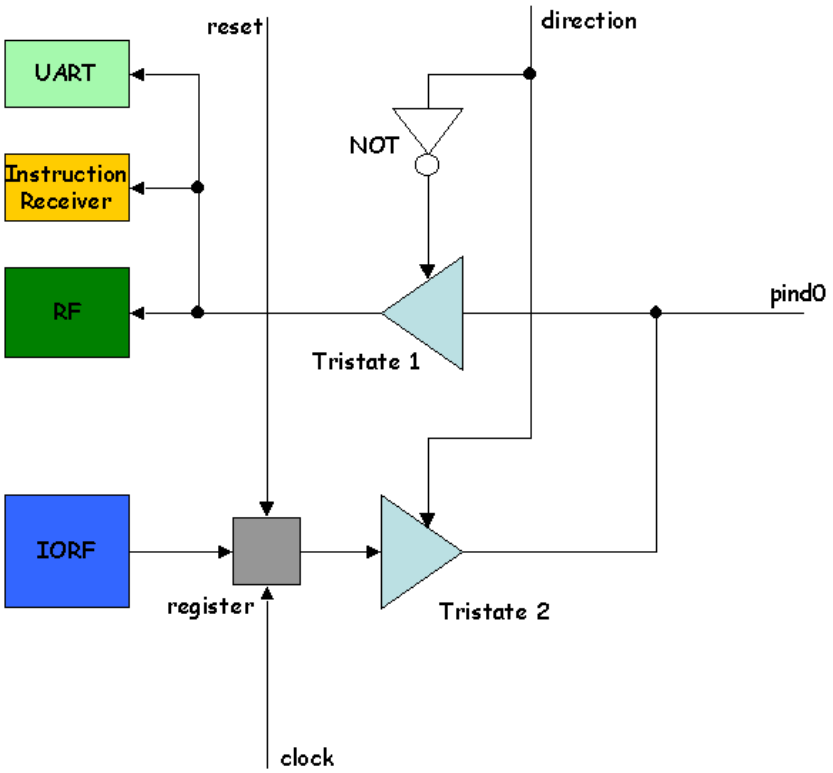
Στο σχήμα 4.7c φαίνεται το κύκλωμα που σχεδιάστηκε ειδικά για το pind0, το οποίο έχει τη λειτουργία της αποδοχής των χαρακτήρων που έρχονται σειριακά από υπολογιστή προς το μοντέλο IP core. Η διαφορά με αυτό στο 4.7b είναι ότι το συγκεκριμένο pin είναι άμεσα συνδεδεμένο και με την είσοδο του UART και του instruction receiver (παράγραφος 4.7), οι οποίες χρησιμοποιούνται για την αποδοχή των δεδομένων. Θα πρέπει να σημειωθεί ότι, επειδή οι IO καταχωρητές PINA, PINB, PINC, και PIND δεν υπάρχουν ως κύκλωμα, όταν διαβάζεται η τιμή κάποιου, εννοούνται οι τιμές των λογικών 1 ή 0 που υπάρχουν εκείνη τη στιγμή στα pins του. Για παράδειγμα, αν ο χρήστης χρησιμοποιήσει την εντολή **in r16, PINA** δεν πρόκειται να διαβαστεί κάποιος IO καταχωρητής PINA από την IORF, αλλά η τιμή της εισόδου pina εκείνη τη στιγμή. Για αυτό το λόγο, στα σχήματα 4.7a, 4.7b και 4.7c φαίνεται η σύνδεση των pins με την RF και όχι με την IORF (αρχεία **ristates.vhd – trastate_dff.vhd – PIND1tristate.vhd – tristates.vhd**).




Σχήμα 4.7a – Κύκλωμα για τα IO pins.

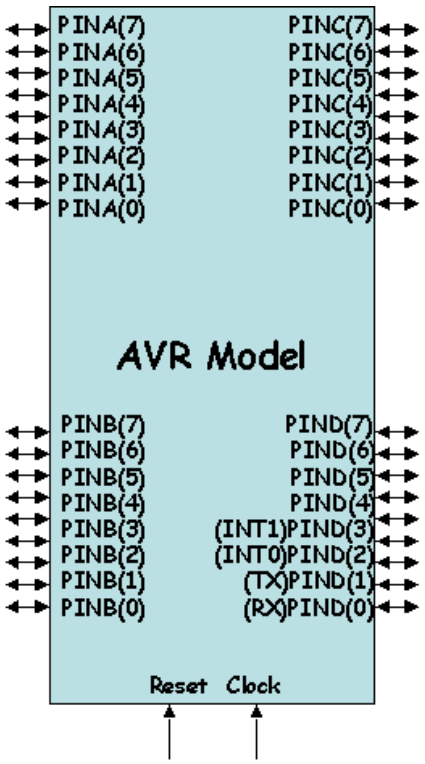



Σχήμα 4.7b – Κύκλωμα για το IO pind1.




Σχήμα 4.7c – Κύκλωμα για το IO pind0.

Επίσης, στο σχήμα 4.7d φαίνονται όλες οι είσοδοι και έξοδοι του μοντέλου IP core. Όπως μπορεί να παρατηρηθεί, τα PIND3 και PIND2 χρησιμοποιούνται και για τα 2 εξωτερικά interrupt INT0 και INT1, ενώ τα PIND1 και PIND0 χρησιμοποιούνται για τη σειριακή αποστολή και λήψη δεδομένων όπως ακριβώς συμβαίνει και στον μικροελεγκτή AT90S8515 (αρχεία **microAVR.vhd** και **microAVR2.vhd**).




Σχήμα 4.7d – IO θύρες του μοντέλου IP Core.

## 4.3 Λειτουργία του UART.

Όπως έχει αναφερθεί και παραπάνω, το μοντέλο του AT90S8515 που σχεδιάστηκε σε VHDL περιλαμβάνει ένα κύκλωμα UART [15] για τη σειριακή επικοινωνία με τον έξω κόσμο μέσω του πρωτοκόλλου RS232. Παρακάτω γίνεται παρουσίαση της γεννήτριας του ρολογιού με βάση το οποίο θα δουλεύει το UART, του Receiver, του Transmitter και ύστερα του UART συνολικά ως κύκλωμα και του τρόπου σύνδεσής του με την υπόλοιπη σχεδίαση.

### 4.3.1 Γεννήτρια κατάλληλου ρολογιού.

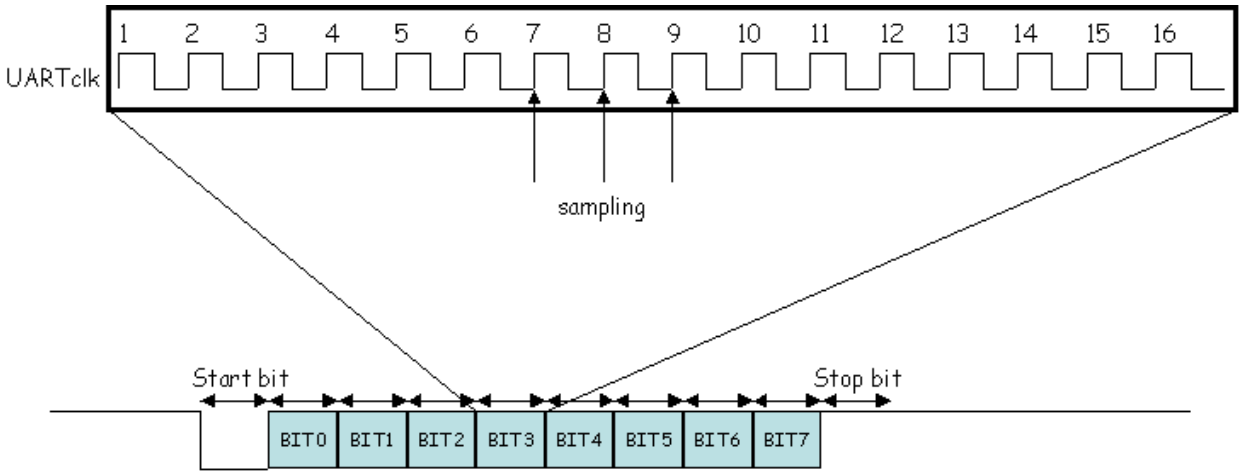
Για να λειτουργήσει σωστά ένα κύκλωμα UART, είναι απαραίτητη η προκαθορισμένη συχνότητα του ρολογιού του συστήματος, καθώς και ο ρυθμός μετάδοσης των bits από τη σειριακή. Έτσι σχεδιάστηκε μια γεννήτρια ρολογιού, η οποία δεν είναι τίποτε παραπάνω από έναν μετρητή που αυξάνεται σε κάθε θετικό παλμό του ρολογιού και όταν φτάσει σε ορισμένη τιμή, αντιστρέφει την έξοδο της γεννήτριας που είναι 1 bit. Έτσι, δημιουργείται ένα ρολόι με συχνότητα αρκετά μικρότερη από αυτή του συστήματος και μπορεί να συνεργαστεί με τη

σειριακή του υπολογιστή για τη σωστή ανταλλαγή δεδομένων. Τέλος, να σημειωθεί ότι χρησιμοποιείται και για τον έλεγχο των σημάτων που δείχνουν πότε τελείωσε ο Receiver ή ο Transmitter (αρχείο **ClkGenerator20.vhd**).

### 4.3.2 UART Receiver.

Ο UART Receiver είναι ένα κύκλωμα που δουλεύει με βάση την έξοδο της γεννήτριας ρολογιού. Προκειμένου να δεχτεί σωστά τα δεδομένα που έρχονται σειριακά από τον υπολογιστή, κάνει δειγματοληψία στην τιμή του καλωδίου σε συγκεκριμένες χρονικές στιγμές και ύστερα αποθηκεύει το bit που μεταδόθηκε.

Αρχικά, η τιμή στο καλώδιο είναι στο λογικό 1, που σημαίνει ότι δεν υπάρχει μετάδοση. Για να αρχίσει η λήψη των δεδομένων, θα πρέπει να εμφανιστεί το start bit που είναι ένας παλμός λογικού 0 για συγκεκριμένο χρονικό διάστημα. Κατά τη διάρκεια του χρονικού αυτού διαστήματος, στο ρολόι του UART σημειώνονται 16 θετικοί παλμοί. Στον $7^o$, $8^o$ και $9^o$ θετικό παλμό γίνεται δειγματοληψία της τιμής του καλωδίου και η πλειονότητα των τιμών που διαβάστηκαν είναι και αυτή που καθορίζει την τιμή του bit που λαμβάνεται. Αν στο start bit η πλειονότητα είναι λογικό 1, τότε υπάρχει σφάλμα στη μετάδοσή του και θα πρέπει να επαναληφθεί. Αν είναι λογικό 0, τότε με τον ίδιο ακριβώς τρόπο συνεχίζεται η δειγματοληψία και των υπολοίπων bit που μεταδίδονται μέχρι να έρθει η στιγμή του stop bit, που είναι ένας παλμός λογικού 1 συγκεκριμένης χρονικής διάρκειας ίσης με τη διάρκεια μετάδοσης όλων των άλλων bits. Να σημειωθεί ότι η συγκεκριμένη σχεδίαση δεν υποστηρίζει την παρουσία parity bit, ενώ γίνεται μετάδοση δεδομένων μόνο 8-bit. Στο σχήμα 4.8 φαίνεται ο τρόπος που λειτουργίας του Receiver, καθώς και πότε γίνεται η δειγματοληψία των δεδομένων (αρχείο **myRec3.vhd**).




Σχήμα 4.8 – Λειτουργία του UART Receiver.

### 4.3.3 UART Transmitter.

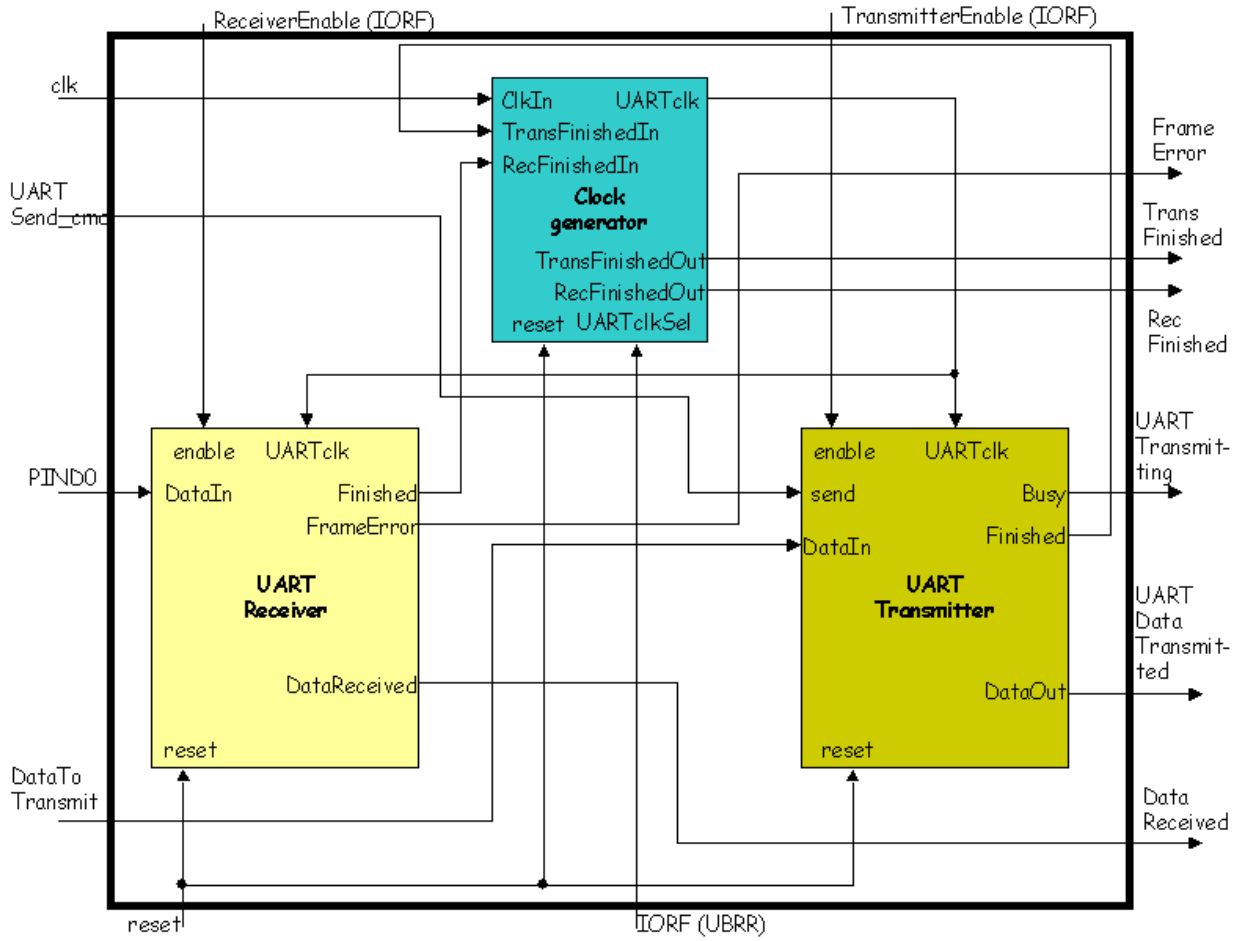
Ο UART Transmitter είναι ένα πιο απλό κύκλωμα από τον Receiver, μιας και το μόνο που χρειάζεται να κάνει είναι να κρατάει την έξοδό του σε λογικό 1 ή 0 για συγκεκριμένο χρονικό διάστημα. Και αυτό το κύκλωμα χρειάζεται απαραιτήτως τη γεννήτρια ρολογιού, μιας και με βάση αυτή γίνεται η σειριακή αποστολή των bits. Έτσι, εφόσον είναι προκαθορισμένο το ρολόι λειτουργίας του συστήματος, καθώς και ο ρυθμός μετάδοσης, μπορεί να γίνει η σωστή αποστολή των δεδομένων στον υπολογιστή (αρχείο **myTrans2.vhd**).

### 4.3.4 Αρχιτεκτονική του UART.

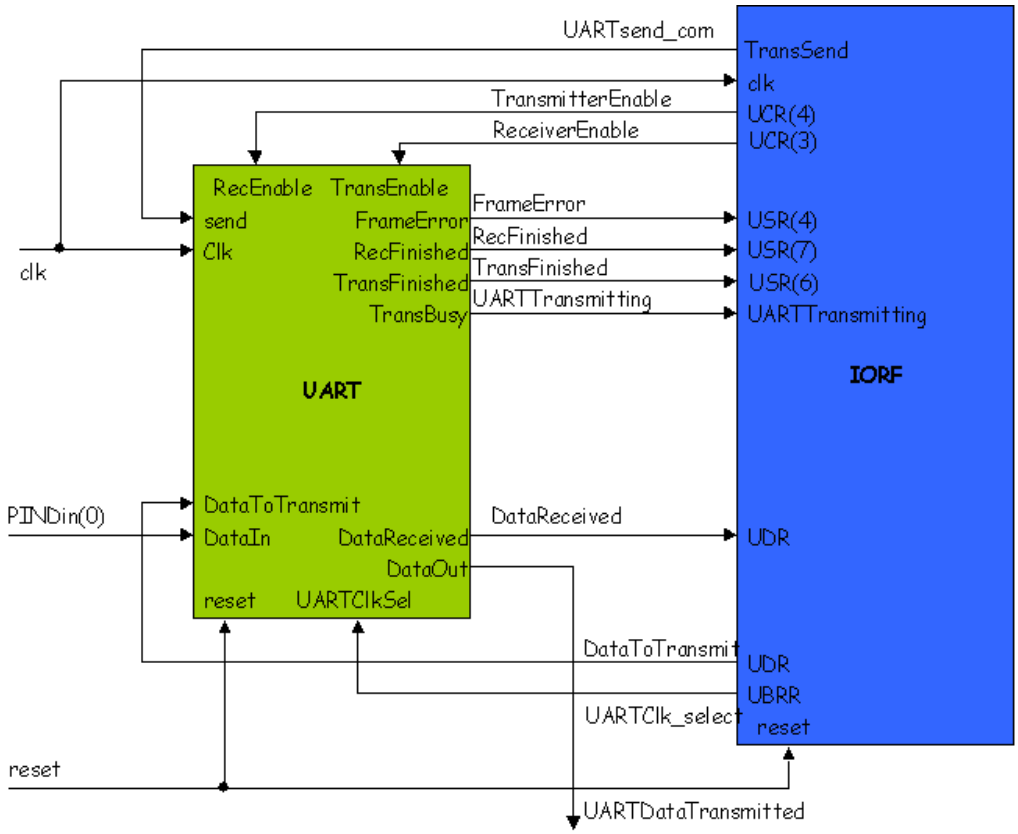
Αφού παρουσιάστηκαν τα υποκυκλώματα της γεννήτριας του ρολογιού, του Receiver και του Transmitter, στην παράγραφο αυτή θα γίνει αναφορά στο πλήρες κύκλωμα του UART, καθώς και στον τρόπο σύνδεσής του με την IORF.

Στο σχήμα 4.9 φαίνεται η εσωτερική σύνδεση των κυκλωμάτων στο UART και στο σχήμα 4.10 φαίνεται η σύνδεσή του με την IORF. Όπως μπορεί να παρατηρηθεί, το ρολόι στον Receiver και στον Transmitter προέρχεται από τη γεννήτρια ρολογιού. Όταν τελειώσει κάποιος από τους δύο, ενημερώνεται η γεννήτρια, η οποία με τη σειρά της ενημερώνει τα αντίστοιχα bits στον USR IO καταχωρητή στην IORF. Τα δεδομένα που στέλνονται από τον υπολογιστή, έρχονται από το PIND0, πάνε στον Receiver και από εκεί στον UDR, ενώ αυτά που στέλνονται, βγαίνουν από τον UDR, πάνε στον Transmitter και τελικά περνάνε από το PIND1. Τέλος, ο IO καταχωρητής UBRR είναι αυτός που αποτελεί την πηγή επιλογής του ρυθμού μετάδοσης των

δεδομένων σειριακά. Γενικά, το πλεονέκτημα της σύνδεσης του UART με την IORF είναι η άμεση ενημέρωση των απαραίτητων πεδίων σε συγκεκριμένους IO καταχωρητές (USR, UCR) που ελέγχουν την κατάσταση διαφόρων κυκλωμάτων, γεγονός που επιβάλλεται για τη σωστή λειτουργία του μοντέλου (αρχείο **myUART.vhd**).



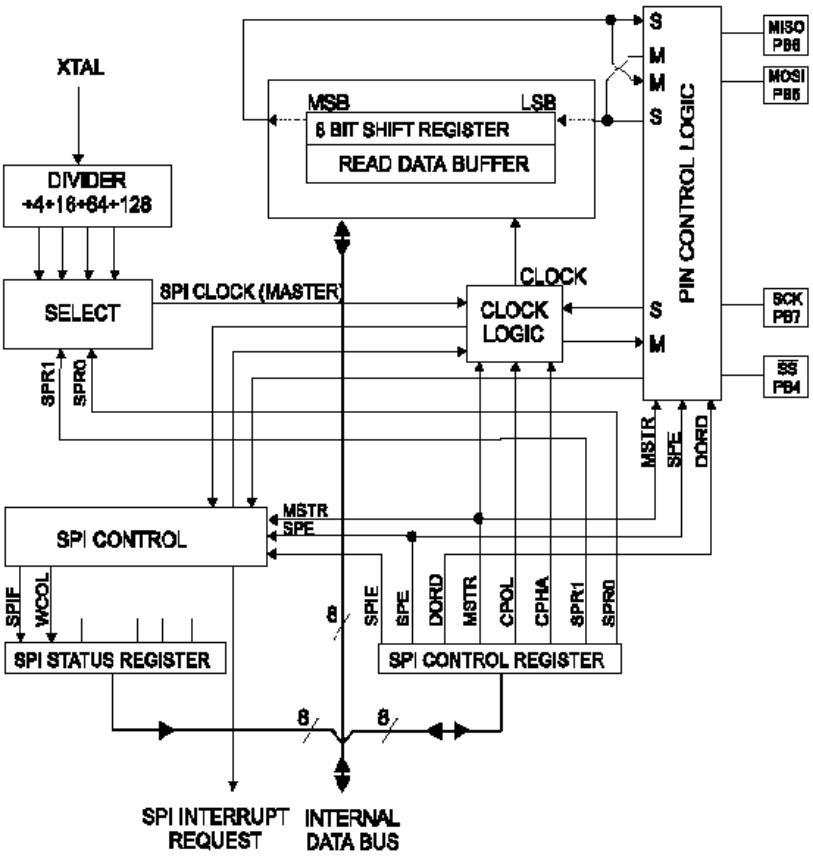

Σχήμα 4.9 – Αρχιτεκτονική του UART.



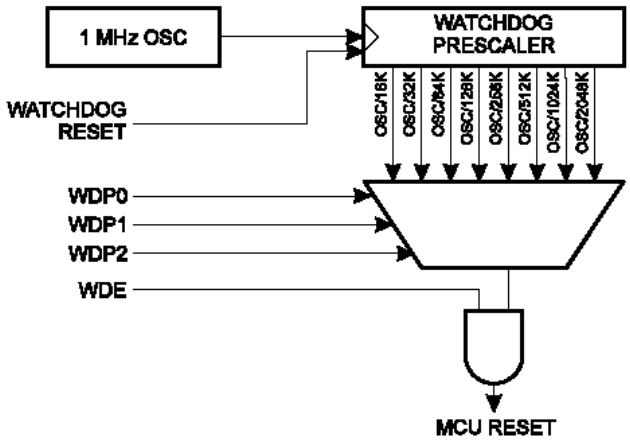

Σχήμα 4.10 – Σύνδεση του UART με την IORF.

## 4.4 Κομμάτια του AT90S8515 που ΔΕΝ έχουν σχεδιαστεί.

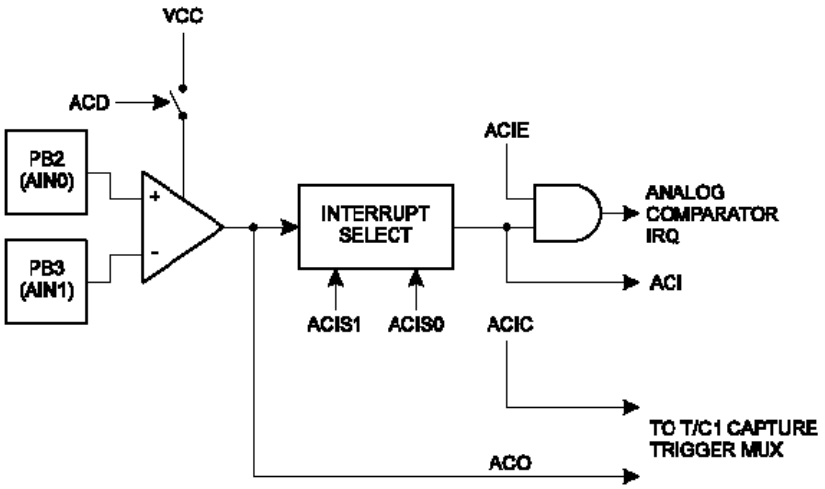
Μέχρι και την προηγούμενη παράγραφο περιγράφτηκαν όλα τα κομμάτια του μικροελεγκτή AT90S8515, τα οποία έχουν συμπεριληφθεί στο μοντέλο IP core. Πριν γίνει η περιγραφή του τρόπου που επιδρούν όλα μαζί, προκειμένου να γίνει η ανάκληση, αποκωδικοποίηση και εκτέλεση των εντολών, αξίζει να σημειωθεί ποια κομμάτια του δεν έχουν συμπεριληφθεί σε αυτή την πρώτη έκδοση. Έτσι, δεν έχει γίνει η σχεδίαση των κυκλωμάτων που υλοποιούν SPI, Watchdog timer, αναλογικό συγκριτή, τα interrupts SPI STC, UART UDRE, TIMER1_CAPT, ANA_COMP και κάποιο interface με EEPROM ή αναλογικό συγκριτή. Στα σχήματα 4.11, 4.12 και 4.13 φαίνονται τα block διαγράμματα στον AT90S8515 των SPI, Watchdog timer και αναλογικού συγκριτή αντίστοιχα.




Σχήμα 4.11 – SPI block διάγραμμα.




Σχήμα 4.12 – Watchdog timer.




Σχήμα 4.13 – Αναλογικός συγκριτής.

## 4.5 Ανάκληση, αποκωδικοποίηση και εκτέλεση των εντολών.

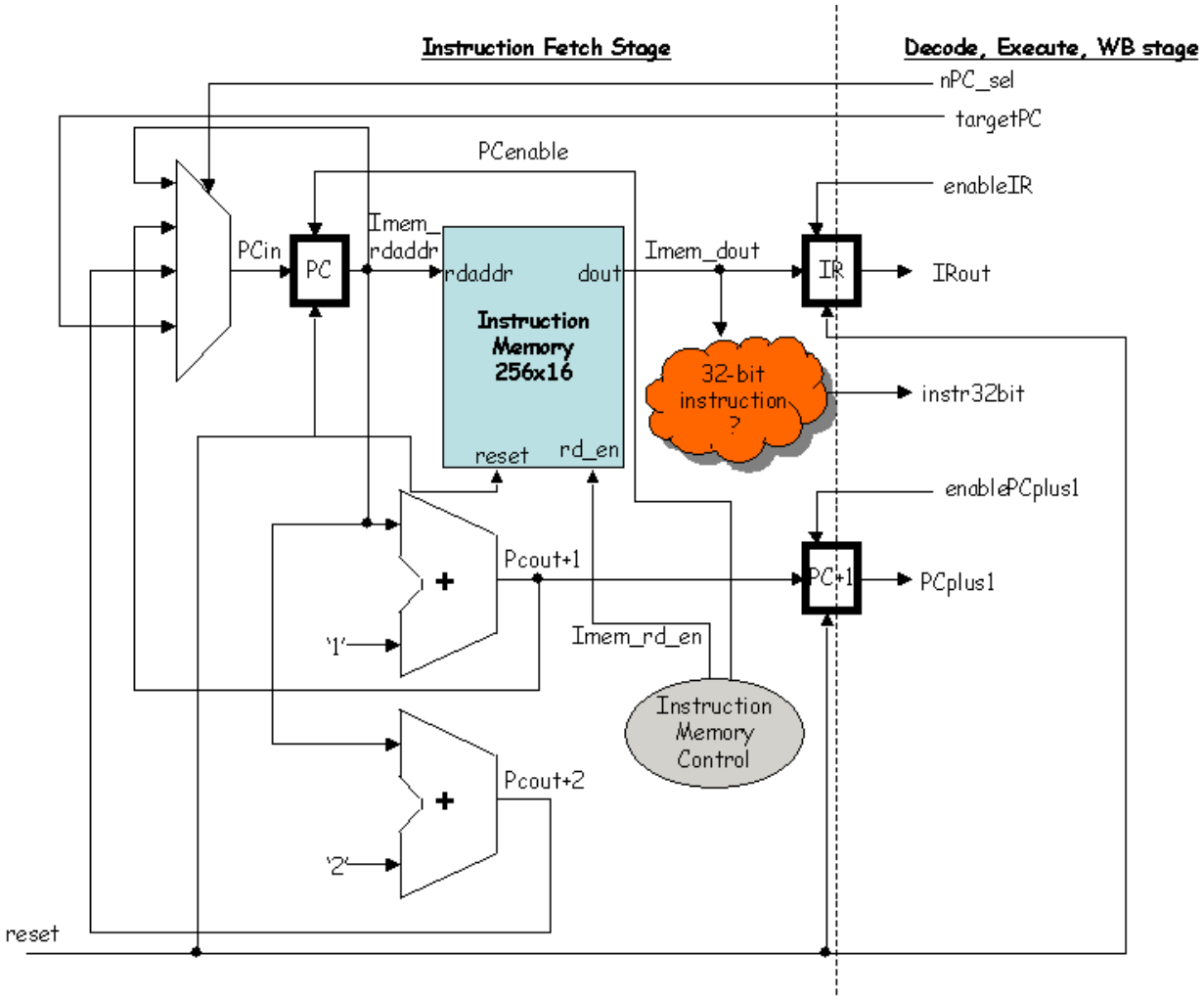
Αφού περιγράφτηκαν τα κομμάτια του AT90S8515 που έχουν σχεδιαστεί, στην παράγραφο αυτή φαίνεται ο τρόπος που έχει γίνει η σύνδεση μεταξύ τους, ώστε να μπορεί να γίνει επιτυχώς η ανάκληση, αποκωδικοποίηση και εκτέλεση των εντολών.

### 4.5.1 Ανάκληση εντολών.

Η σχεδίαση χωρίζεται ουσιαστικά σε 2 μέρη και ανάμεσά τους υπάρχουν pipeline καταχωρητές [1, 2]. Το πρώτο μέρος (Instruction fetch stage) είναι αυτό που αναλαμβάνει να διαβάσει την εντολή από τη μνήμη εντολών και να την πάει στον καταχωρητή IR (Instruction Register), από όπου και θα διαβαστεί πιο μετά για να αρχίσει η εκτέλεσή της και το δεύτερο μέρος είναι η αποκωδικοποίηση και εκτέλεση της εντολής.

Ο λόγος που αποφασίστηκε μια αρχιτεκτονική με 2 pipeline βαθμίδες, είναι για να αποφευχθεί η πολυπλοκότητα στη σχεδίαση της μονάδας γενικού ελέγχου που θα χρειαζόταν αν υπήρχε βαθύτερο pipeline [1, 2]. Είναι αλήθεια ότι η παρουσία πολλών pipeline επιπέδων αυξάνουν σημαντικά την πολυπλοκότητα της σχεδίασης, εισάγοντας παράλληλα control και data hazards. Έτσι, χρησιμοποιώντας μόνο 2 pipeline επίπεδα, αποφεύγονται τα data hazards, μιας και δεν υπάρχει περίπτωση να ξεκινήσει κάποια εντολή χωρίς να έχουν γραφτεί τα αποτελέσματα της προηγούμενης, οπότε αποφεύγονται και τα stalls, ενώ τα control hazards

μπορούν να ελεγχθούν αρκετά εύκολα. Αξίζει ακόμα να σημειωθεί ότι ο συγκεκριμένος τρόπος σχεδίασης βοηθάει πολύ στην επεκτασιμότητα του μοντέλου, καθώς διευκολύνει την προσθήκη νέων εντολών, μιας και δεν υπάρχουν πολλά pipeline επίπεδα.



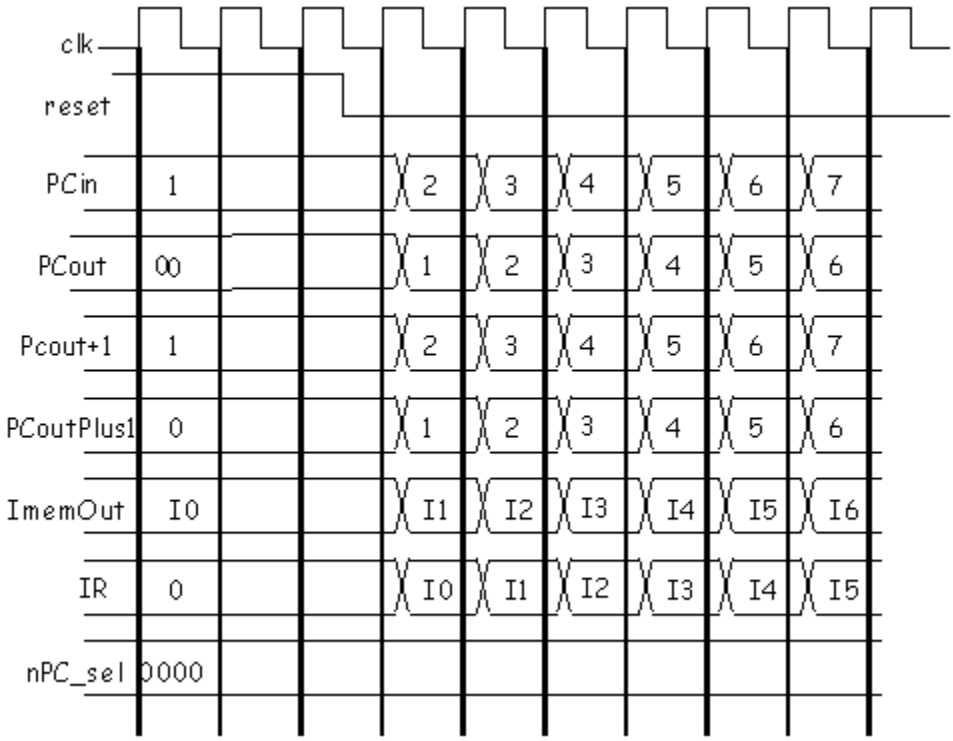

Σχήμα 4.8 – Βαθμίδα ανάκλησης εντολών.

Στο σχήμα 4.8 φαίνεται η βαθμίδα ανάκλησης εντολών. Για λόγους απλότητας έχουν σχεδιαστεί μόνο τα σήματα που χρειάζονται για την ανάκληση των εντολών από τη μνήμη. Όπως μπορεί να παρατηρήσει κανείς, πέρα από τη μνήμη εντολών, υπάρχουν οι καταχωρητές PC+1, PC και IR. Ο $1^{ος}$ κρατάει την τιμή του PC (Program Counter) συν 1, δηλαδή τη διεύθυνση της επόμενης εντολής από αυτήν που θα εκτελεστεί, ο $2^{ος}$ κρατάει την τιμή του PC και ο $3^{ος}$ κρατάει την εντολή που πρόκειται να εκτελεστεί. Αν η εντολή αυτή είναι 32-bit, τότε υπάρχει ειδικό σήμα (instr32bit) που ενημερώνει τη μονάδα γενικού ελέγχου και τότε δεν

χρησιμοποιείται η διεύθυνση PC+1, αλλά η διεύθυνση PC+2. Ο λόγος που συμβαίνει αυτό είναι επειδή οι 32-bit εντολές καταλαμβάνουν 2 συνεχόμενες θέσεις στη μνήμη εντολών, οπότε όταν θα πρέπει να ανακληθεί η επόμενη εντολή, ο PC θα πρέπει να «δείξει» 2 θέσεις παρακάτω και όχι 1. Ακόμα, κάποια σήματα έρχονται από τη $2^η$ βαθμίδα (nPCsel, enableIR, enablePCplus1, targetPC) και πιο συγκεκριμένα από τη μονάδα γενικού ελέγχου. Στο συγκεκριμένο σχήμα δεν φαίνεται η λογική που χρησιμοποιείται για την εξυπηρέτηση των interrupts, καθώς περιγράφεται αναλυτικά σε παρακάτω παράγραφο. Να σημειωθεί ότι υπάρχει και η μονάδα ελέγχου της μνήμης εντολών, η οποία αναλαμβάνει την σειριακή εγγραφή των εντολών από τον υπολογιστή και, αφού τελειώσει η εγγραφή, ενεργοποιεί το σήμα ανάγνωσης.

Στο σχήμα 4.9 φαίνεται ο τρόπος λειτουργίας της βαθμίδας ανάκλησης εντολών στην περίπτωση όπου σε κάθε κύκλο γινόταν ανάκληση της επόμενης εντολής. Ο αριθμός των κύκλων που περνάνε για να ανακληθεί η επόμενη εντολή στην τελική σχεδίαση, δεν είναι 1 αλλά παραπάνω και εξαρτάται από το είδος της εντολής. Ο έλεγχος της χρονικής στιγμής ανάκλησης της επόμενης εντολής γίνεται από το σήμα nPC_sel που προέρχεται από τη μονάδα γενικού ελέγχου από τη $2^η$ βαθμίδα. Όταν χρειαστεί κάποια εντολή να παραμείνει στον IR για περισσότερους από 1 κύκλους, τότε το nPC_sel αναλαμβάνει να στέλνει την ίδια εντολή και όχι την επόμενη, μέχρι να αλλαχτεί η τιμή του από τη μονάδα γενικού ελέγχου.

Όπως φαίνεται λοιπόν στο σχήμα 4.9, αρχικά το reset είναι 1 για να αρχικοποιηθούν οι καταχωρητές σωστά. Έτσι, το PCout έχει αρχική τιμή 0 και επειδή ο αθροιστής είναι συνδυαστικό κύκλωμα, μετά από λίγο, το PCout+1 γίνεται 1. Επειδή το reset είναι ακόμα 1, η έξοδος του PC+1 είναι 0, ενώ το ImemOut έχει την $1^η$ (I1) εντολή για εκτέλεση. Η έξοδος του IR είναι και αυτή 0 και το nPC_sel έχει σταθερή τιμή 0000, που σημαίνει ότι σε κάθε κύκλο έρχεται η επόμενη εντολή για εκτέλεση. Όταν το reset γίνει 0, το PCout παίρνει την τιμή του PCin που είναι το 1. Μετά από λίγο, λόγω του συνδυαστικού κυκλώματος του αθροιστή, το PCout+1 γίνεται 2, οπότε μετά από λίγο, το PCin γίνεται 2. Το Pcplus1 παίρνει την τιμή 1, ενώ το IR παίρνει την $1^η$ εντολή (I0) και το ImemOut τη $2^η$ (I1). Με τον ίδιο ακριβώς τρόπο συνεχίζεται η διαδικασία ανάκλησης εντολών στην περίπτωση όπου δε συμβεί κάτι που θα τη διακόψει (κάποιο interrupt). Έτσι, με το κύκλωμα αυτό υπάρχει η δυνατότητα της ανάκλησης των εντολών μέσα από τη μνήμη μετά από όσους κύκλους είναι αναγκαίο, γεγονός που ελέγχεται από το σήμα nCP_sel. Στον πίνακα 4.4 φαίνονται οι τιμές του nPC_sel, με τις οποίες επιλέγεται κάθε φορά η είσοδος προς τον καταχωρητή PC (αρχεία **ifstage3.vhd** και **ifstage5.vhd**).




Σχήμα 4.9 – Λειτουργία της βαθμίδας ανάκλησης εντολών.

| Τιμή του nPC_sel | Είσοδος προς PC | Τιμή του nPC_sel | Είσοδος προς PC |
|---|---|---|---|
| 0000 | Pcout+1 | 0110 | 4 (TIMER1 COMPA) |
| 0001 | Pcout | 0111 | 5 (TIMER1 COMPB) |
| 0010 | TargetPC | 1000 | 6 (TIMER1 OVF) |
| 0011 | PCout+2 | 1001 | 7 (TIMER0 OVF) |
| 0100 | 1 (INT0) | 1010 | 9 (UART RX) |
| 0101 | 2 (INT1) | 1100 | 11 (UART TXC) |

Πίνακας 4.4 – Τιμές του nPC_sel.

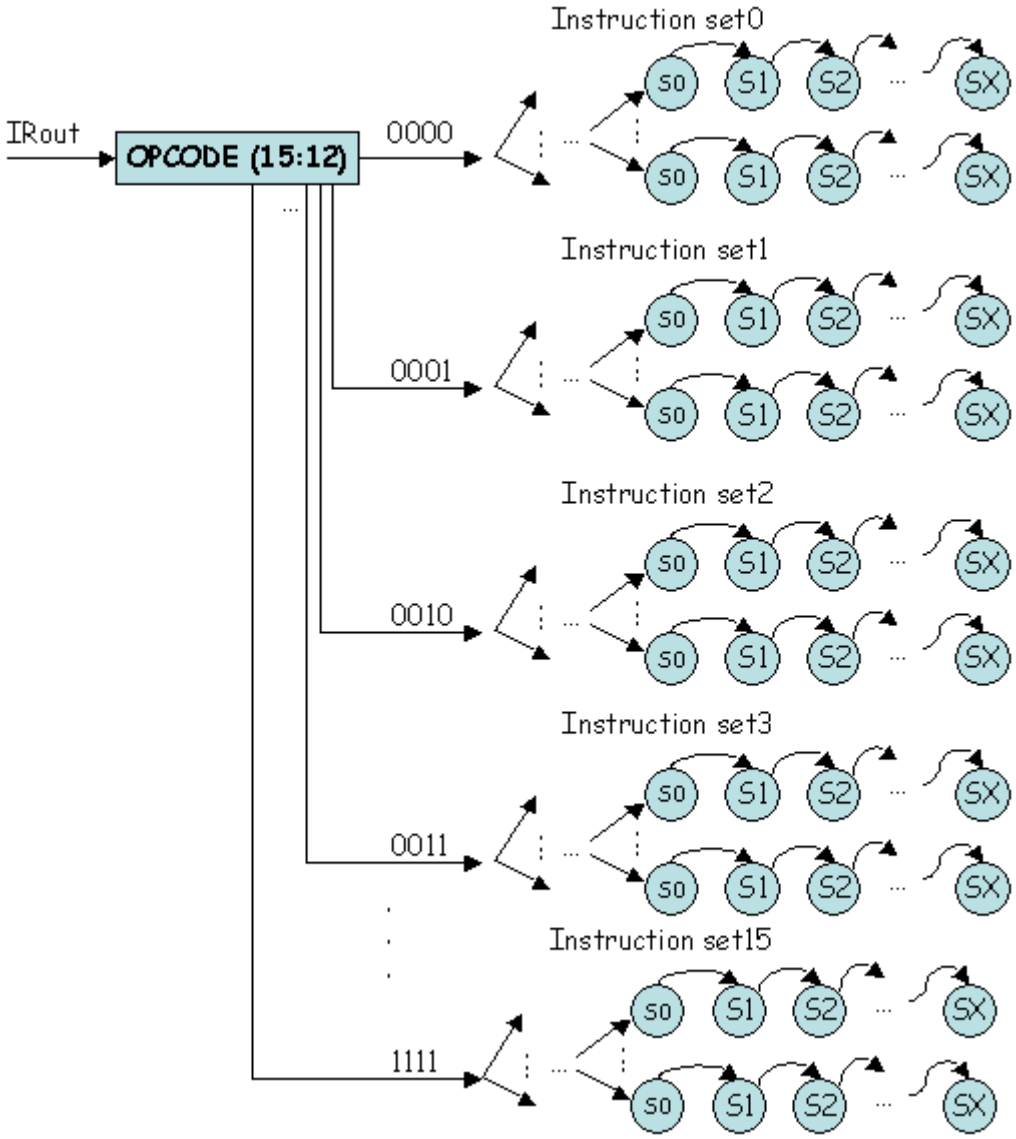
### 4.5.2 Αποκωδικοποίηση και εκτέλεση εντολών.

Η αποκωδικοποίηση των εντολών γίνεται στη μονάδα γενικού ελέγχου. Πιο συγκεκριμένα, έχει γίνει ομαδοποίηση των εντολών ανάλογα με το operation code τους (κάποια

bits από τα 16 συνολικά). Το αποτέλεσμα είναι μια δενδροειδής αποκωδικοποίηση της κάθε εντολής, μιας και υπάρχουν αρκετές υποκατηγορίες εντολών. Για να γίνει πιο εύκολα κατανοητός ο τρόπος ομαδοποίησης, θα γίνει αναφορά της 1ης κατηγορίας εντολών, η οποία είναι με βάση τα 4 MSBs. Στον πίνακα 4.5 φαίνονται όλες οι υποστηριζόμενες εντολές από το συγκεκριμένο μοντέλο, καθώς επίσης και το 1ο επίπεδο του διαχωρισμού τους σε υποομάδες, ενώ στο σχήμα 4.10 φαίνεται ο τρόπος που γίνεται η ομαδοποίηση των εντολών.

| 4 MSBs (15 ως 12) | Ομάδα εντολών |
|---|---|
| 0000 | add, lsl, cpc, nop, sbc |
| 0001 | adc, rol, cp, cpse, sub |
| 0010 | and, eor, clr, mov, or |
| 0011 | cpi |
| 0100 | sbci |
| 0101 | subi |
| 0110 | ori, sbr |
| 0111 | andi, cbr |
| 1000 | std(Y+q), std(Z+q), ldd(Y+q), ldd(Z+q) (για 0<=q<=31) |
| 1001 | ldd(-Y), ld(X,Rr), ld(X+,Rr), ld(-X,Rr), ldd(-Z), lds, pop, push, std(-Y), st(X,Rr), st(X+,Rr), st(-X,Rr), std(-Z), sts, asr, bclr, clc, clh, cli, cln, cls, clt, clz, bset, sec, seh, sei, sen, ses, set, sev, sez, pushPC, com, dec, icall, ijmp, inc, neg, ret, reti, ror, lsr, adiw, sbiw, cbi, sbic, sbi, sbis, swap |
| 1010 | std(Y+q), std(Z+q), ldd(Y+q), ldd(Z+q) (για 32<=q<=63) |
| 1011 | in, out |
| 1100 | rjmp |
| 1101 | rcall |
| 1110 | ldi, ser |
| 1111 | brbs, brcs, breq, brhs, brie, brlo, brlt, brmi, brts, brvs, brbc, brcc, brge, brhc, brid, brne, brpl, brsh, brtc, brvc, bld, bst, sbrs, sbrc |

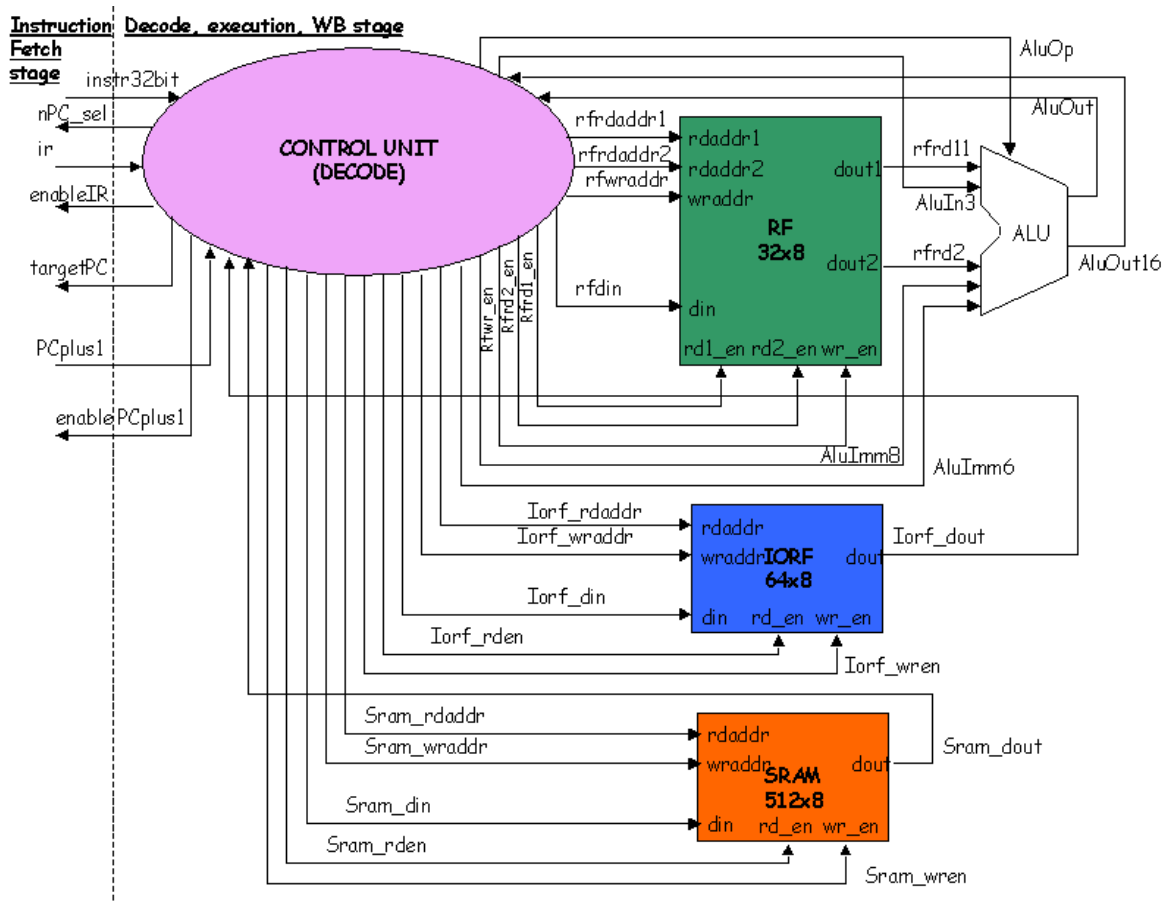
Πίνακας 4.5 – Υποστηριζόμενες εντολές και διαχωρισμός τους σε υποομάδες.




Σχήμα 4.10 – Ομαδοποίηση εντολών.

Για την εκτέλεση της κάθε εντολής, υπάρχει και η αντίστοιχη μηχανή πεπερασμένων καταστάσεων (FSM – Finite State Machine). Δηλαδή, κάθε εντολή, για να ολοκληρωθεί η εκτέλεσή της, χρειάζεται συγκεκριμένους κύκλους ρολογιού και μέχρι να τελειώσει η εκτέλεσή της, οι υπόλοιπες πρέπει να περιμένουν. Για την ακρίβεια, μέχρι 2 κύκλους πριν τελειώσει η υπό εκτέλεση εντολή, το σήμα nPC_sel έχει την τιμή 0001, που σημαίνει ότι δεν φέρνει κάποια καινούρια εντολή. Δύο κύκλους όμως πριν το τέλος, το nPC_sel παίρνει την τιμή που πρέπει, για να ξεκινήσει να έρχεται η επόμενη εντολή, ανάλογα με τα αποτελέσματα της παρούσας, μιας και θα μπορούσε να ήταν κάποια εντολή διακλάδωσης, οπότε το nPC_sel θα έπαιρνε την τιμή 0001

αν δεν ίσχυε η συνθήκη διακλάδωσης ή την τιμή 0010 αν ίσχυε. Ο λόγος που η επόμενη εντολή ξεκινάει 2 κύκλους πιο πριν, είναι επειδή χρειάζεται 1 κύκλος για να ανακληθεί από τη μνήμη εντολών και 1 κύκλος για να περάσει τον IR καταχωρητή. Η χρήση των FSMs προτιμήθηκε, ώστε να είναι αρκετά εύκολη η προσθήκη νέων εντολών στο μοντέλο IP core. Πράγματι, σε περίπτωση όπου είναι επιθυμητό κάτι τέτοιο, το μόνο που χρειάζεται να γίνει, είναι η δημιουργία της καινούριας FSM για την αντίστοιχη εντολή και η κατάλληλη ένταξή της στο δέντρο αποκωδικοποίησης.

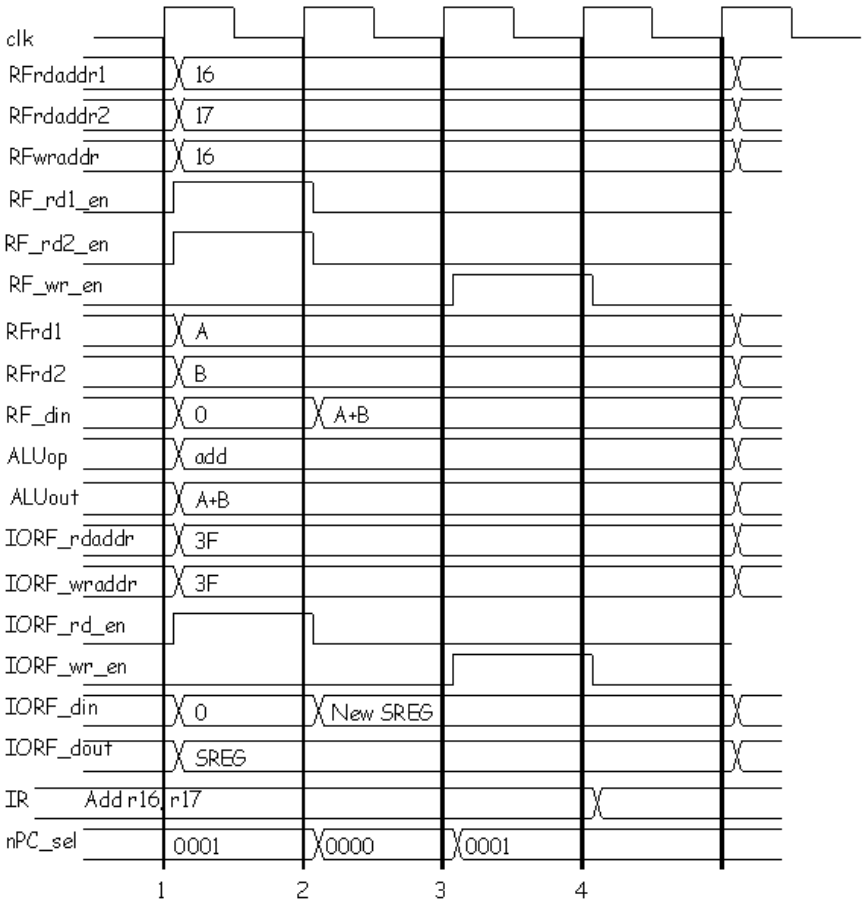



Σχήμα 4.11 – Βαθμίδα αποκωδικοποίησης και εκτέλεσης εντολών.

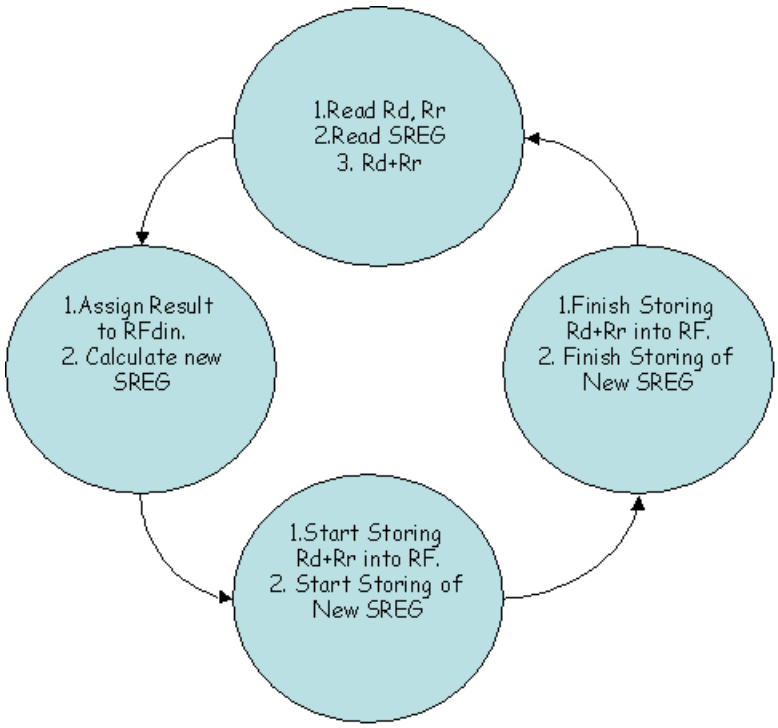
Στο σχήμα 4.11 φαίνεται η βαθμίδα αποκωδικοποίησης και εκτέλεσης εντολών και τα σήματα με τα οποία επικοινωνεί με τη βαθμίδα ανάκλησης των εντολών. Για λόγους απλότητας, έχουν σχεδιαστεί μόνο τα σήματα που χρειάζονται για την αποκωδικοποίηση των εντολών. Όπως μπορεί να παρατηρηθεί, στη βαθμίδα αυτή υπάρχει η RF, η IORF και η SRAM, οι οποίες επικοινωνούν άμεσα με τη μονάδα κεντρικού ελέγχου. Σε αυτήν πάει αρχικά και όποια εντολή

είναι υπό εκτέλεση, οπότε εκεί γίνεται η αποκωδικοποίηση των πεδίων που πρέπει να προσπελαστούν σε οποιοδήποτε από τα 3 παραπάνω αποθηκευτικά μέσα. Αφού ξεκινήσει η εκτέλεση της εκάστοτε εντολής, 2 κύκλου πριν το τέλος, ξεκινάει να έρχεται η επόμενη που θα πρέπει να εκτελεστεί.

Για να γίνει πιο εύκολα κατανοητός ο τρόπος εκτέλεσης των εντολών, στο σχήμα 4.12 φαίνονται οι κυματομορφές των σημάτων στην περίπτωση όπου εκτελείται μια πρόσθεση μεταξύ των περιεχομένων 2 καταχωρητών της RF, ενώ στο 4.13 φαίνεται η αντίστοιχη FSM.




Σχήμα 4.12 – Εκτέλεση πρόσθεσης.




Σχήμα 4.13 – FSM για την εκτέλεση της πρόσθεσης.

Με παρόμοιο τρόπο, εκτελούνται και οι υπόλοιπες εντολές από τις οποίες κάποιες είναι αρκετά πιο πολύπλοκες, ενώ άλλες πιο απλές. Για παράδειγμα, η εντολή reti χρειάζεται 15 κύκλους για να εκτελεστεί αφού περιλαμβάνει μεταξύ άλλων την ανάγνωση του PC+1 από τη στοίβα. Από την άλλη πλευρά, η εντολή ldi χρειάζεται μόνο 3 κύκλους μιας και το μόνο που γίνεται είναι η απευθείας εγγραφή ενός αριθμού στην RF (αρχείο **datapath70.vhd**).

## 4.6 Λειτουργία των interrupts.

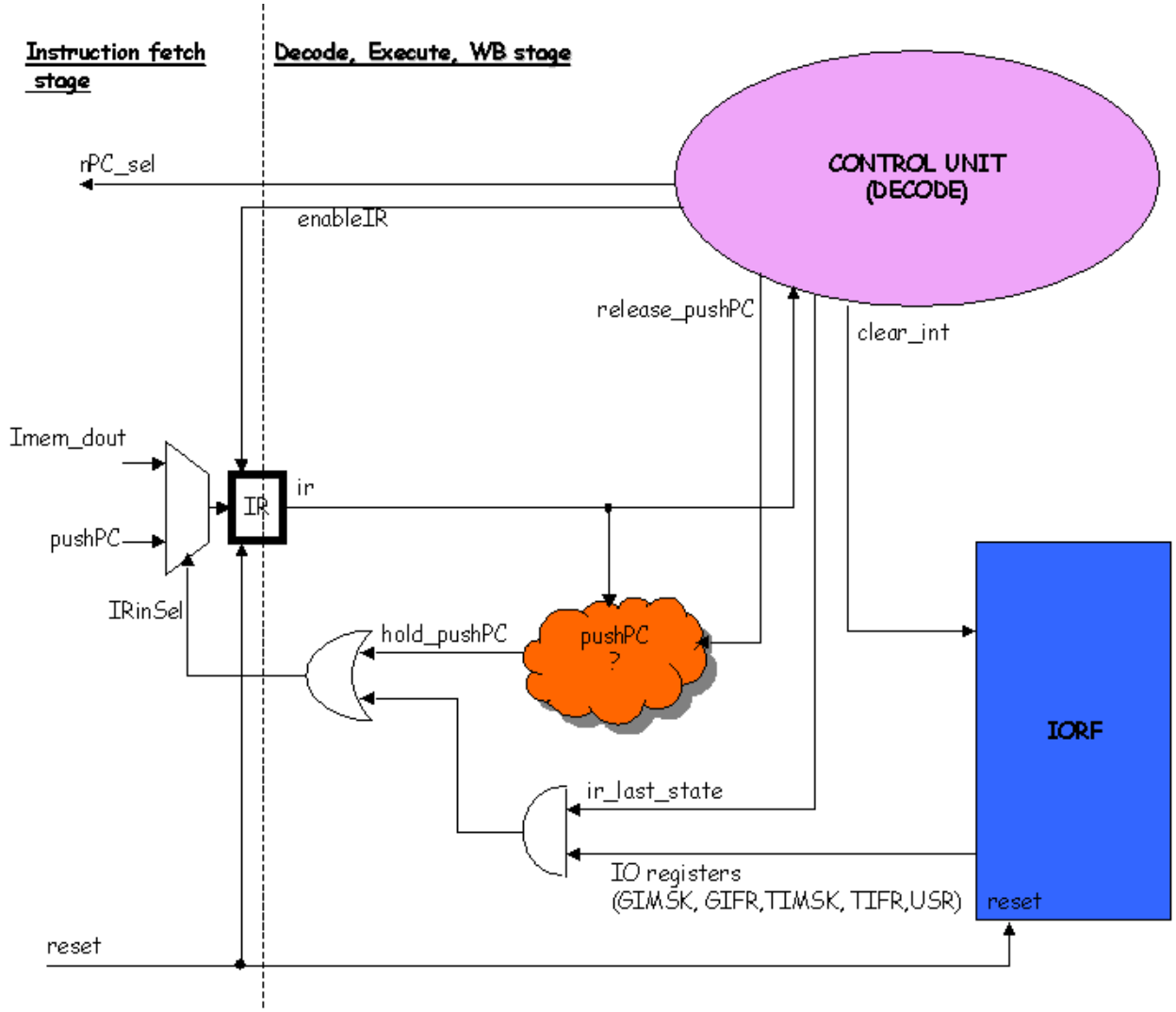
Η εξυπηρέτηση των interrupts ήταν ένα από τα δυσκολότερα και σημαντικότερα σημεία της σχεδίασης του μοντέλου. Ο λόγος που συμβαίνει αυτό είναι επειδή τα interrupts μπορούν να συμβούν σε οποιαδήποτε χρονική στιγμή, οπότε η σχεδίαση θα πρέπει να είναι τέτοια ώστε να μπορεί να αντεπεξέλθει σε όλες τις δυνατές περιπτώσεις εξυπηρέτησής τους. Κάτι τέτοιο χρειάζεται αρκετή προσοχή, καθώς υπάρχει πιθανότητα την ώρα που ξεκινάει η διαδικασία εξυπηρέτησης ενός interrupt, να υπάρξει και $2^o$ interrupt με μεγαλύτερη προτεραιότητα, οπότε αυτομάτως το τελευταίο θα πρέπει να εξυπηρετηθεί πρώτο και ύστερα να εξυπηρετηθεί αυτό που συνέβη αρχικά.

Για να γίνει η εξυπηρέτηση ενός interrupt που συνέβη, υπάρχει μια συγκεκριμένη διαδικασία που πρέπει να γίνει, η οποία αρχικά περιλαμβάνει την αποθήκευση του PC στη στοίβα. Κατόπιν, απενεργοποιείται το I bit του IO καταχωρητή SREG ώστε να μη ξεκινήσει η εξυπηρέτηση ενός interrupt την ώρα που είναι σε εκτέλεση κάποιο άλλο (προφανώς, μπορούν να συμβούν πολλά διαφορετικά interrupts την ώρα που εκτελείται κάποιο συγκεκριμένο, αλλά θα αρχίσουν να εκτελούνται ένα-ένα κάθε φορά ανάλογα με την προτεραιότητά τους, αφού τελειώσει η εξυπηρέτηση του προηγούμενου) και στο τέλος γίνεται η απενεργοποίηση του αντίστοιχου bit που εκφράζει το συγκεκριμένο interrupt, μιας και ξεκίνησε η εκτέλεσή του.

Η παραπάνω διαδικασία χρειάζεται 15 κύκλους για να εκτελεστεί και την αναλαμβάνει μία συγκεκριμένη εντολή, η pushPC. Η τελευταία εκτελείται κάθε φορά πριν ξεκινήσει η εκτέλεση της ρουτίνας που εξυπηρετεί κάποιο interrupt και αναλαμβάνει να κάνει όλα όσα αναφέρθηκαν στην προηγούμενη παράγραφο. Για να τελειώσει η εκτέλεση ενός interrupt, ο χρήστης θα πρέπει να χρησιμοποιήσει την εντολή reti ως την τελευταία στη ρουτίνα, μιας και αυτή είναι που αναλαμβάνει να διαβάσει τον PC από τη στοίβα για να συνεχίσει η κανονική ροή του προγράμματος. Επίσης, να σημειωθεί ότι, αν στο τέλος της εκτέλεσης ενός interrupt υπάρχει κάποιο άλλο που περιμένει, τότε θα εκτελεστεί η επόμενη εντολή του κανονικού προγράμματος και κατόπιν θα ξεκινήσει η διαδικασία εξυπηρέτησής του.

Στο σχήμα 4.14 φαίνεται το κύκλωμα που έχει σχεδιαστεί για την εξυπηρέτηση των interrupts. Όπως μπορεί να παρατηρηθεί, υπάρχει ένας πολυπλέκτης 2 σε 1 πριν από τον IR καταχωρητή, ο οποίος έχει σαν εισόδους την έξοδο της μνήμης εντολών και την εντολή pushPC, ενώ το σήμα IRinSel χρησιμοποιείται για το ποια είσοδος θα περνάει κάθε φορά. Όταν είναι 0, περνάει η έξοδος της μνήμης εντολών, δηλαδή συνεχίζεται κανονικά η εκτέλεση του προγράμματος. Για να γίνει 1, θα πρέπει να έχει συμβεί κάποιο interrupt που περιμένει να εξυπηρετηθεί και συγχρόνως να είναι ο τελευταίος κύκλος εκτέλεσης της παρούσας εντολής. Έτσι, δεν περνάει η επόμενη εντολή, αλλά η pushPC, η οποία αναλαμβάνει να εκκινήσει τη διαδικασία που περιγράφτηκε πιο πριν για να εκτελεστεί το interrupt. Επειδή όπως αναφέρθηκε παραπάνω η τελευταία χρειάζεται 15 κύκλους για να ολοκληρωθεί, υπάρχει το σήμα hold_pushPC, το οποίο αναλαμβάνει να τροφοδοτεί τη μονάδα γενικού ελέγχου με την ίδια την εντολή (ανάδραση κατά κάποιο τρόπο), μέχρι να φτάσει στον $14^{ο}$ κύκλο. Τότε, ενεργοποιείται το σήμα release_pushPC, το οποίο αναλαμβάνει να σταματήσει την ανάδραση που έχει δημιουργηθεί και αφήνει να περάσει η $1^{η}$ εντολή που είναι στη ρουτίνα του interrupt. Να

σημειωθεί ότι μέχρι και τον 13$^{ο}$ κύκλο εκτέλεσης της pushPC, μπορεί να ληφθεί υπόψη η προτεραιότητα των interrupts, καθώς σε αυτόν ακριβώς τον κύκλο γίνεται η απενεργοποίηση του bit που αντιπροσωπεύει το interrupt. Υπάρχει, δηλαδή, η δυνατότητα ακόμα και να έχει ξεκινήσει η διαδικασία για την εξυπηρέτηση ενός interrupt, να συμβεί κάποιο άλλο με μεγαλύτερη προτεραιότητα και τελικά να εξυπηρετηθεί (όπως θα έπρεπε) πρώτα το τελευταίο και ύστερα το προηγούμενο. Τέλος, στον πίνακα 4.6 φαίνονται τα interrupts που υποστηρίζονται από το μοντέλο σε αυτή την 1$^{η}$ έκδοση. Οι αριθμοί interrupt είναι με βάση το datasheet της Atmel και όσο πιο μικρός είναι κάποιος, τόσο μεγαλύτερη είναι και η προτεραιότητα του interrupt (αρχεία **ifstage3.vhd / ifstage5.vhd – datapath70.vhd – iorf64x8_14_20.vhd**).




Σχήμα 4.14 – Κύκλωμα εξυπηρέτησης των interrupts.

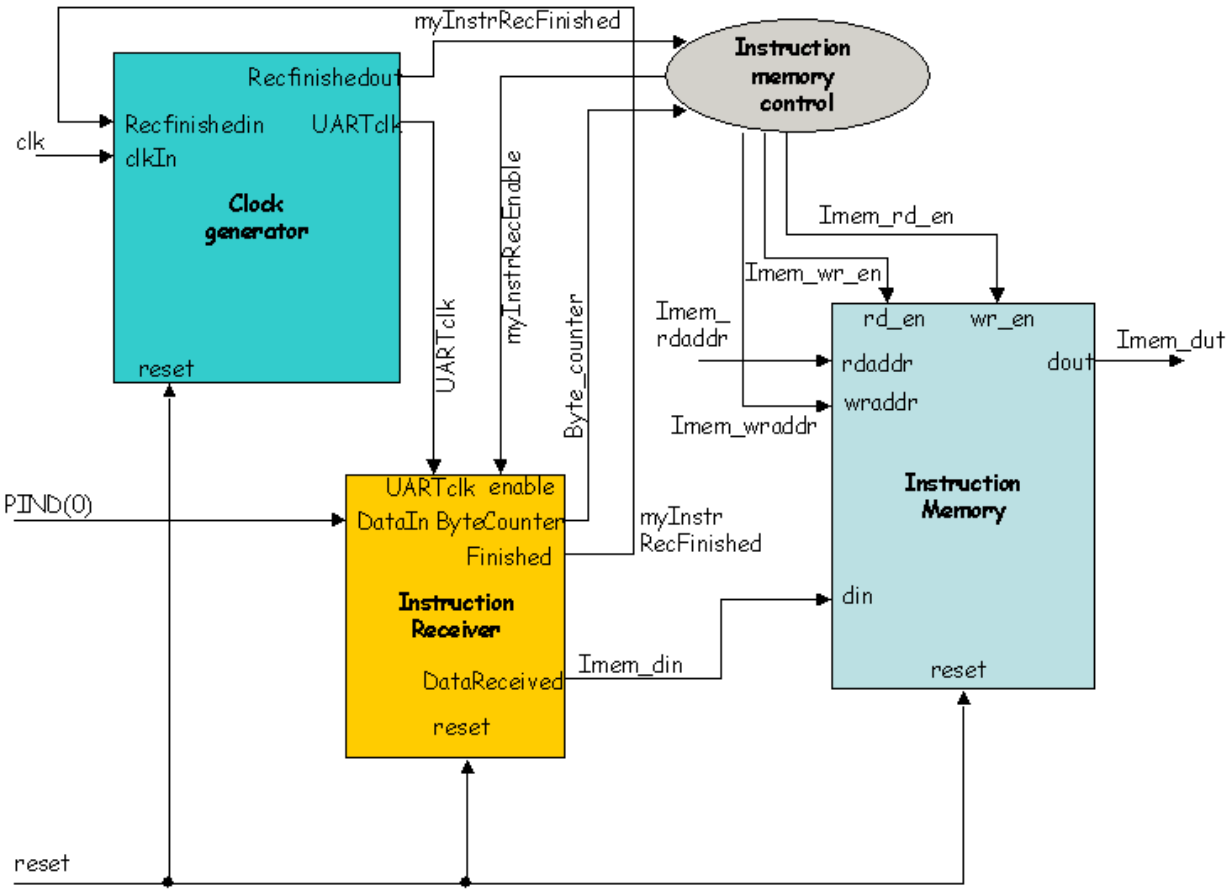
| Αριθμός interrupt | Διεύθυνση εκκίνησης | Πηγή | Περιγραφή |
|---|---|---|---|
| 1 | $000 | RESET | Γενικό reset |
| 2 | $001 | INT0 | Εξωτερικό interrupt 0 |
| 3 | $002 | INT1 | Εξωτερικό interrupt 1 |
| 5 | $004 | TIMER1 COMPA | $1^{η}$ ισότητα τιμών στον Timer1 |
| 6 | $005 | TIMER1 COMPB | $2^{η}$ ισότητα τιμών στον Timer1 |
| 7 | $006 | TIMER1 OVF | Υπερχείλιση του Timer1 |
| 8 | $007 | TIMER0 OVF | Υπερχείλιση του Timer0 |
| 10 | $009 | UART RX | Ο UART Receiver τελείωσε |
| 12 | $00B | UART TX | Ο UART Transmitter τελείωσε |

Πίνακας 4.6 – Υποστηριζόμενα interrupts.

## 4.7 Instruction Receiver.

Μέχρι τώρα έχει γίνει μια συνολική περιγραφή του μοντέλου IP core που σχεδιάστηκε βασισμένο στον AT90S8515. Προκειμένου όμως να μπορεί να προγραμματιστεί με κώδικα που έχει γραφτεί σε assembly στο AVR Studio, σχεδιάστηκε ένα ξεχωριστό κύκλωμα, το οποίο δεν υπάρχει στον μικροελεγκτή και αναλαμβάνει να προγραμματίσει τη μνήμη εντολών μέσω της σειριακής ενός υπολογιστή. Για λόγους απλότητας ονομάζεται instruction receiver.

Στο σχήμα 4.18 φαίνεται ένα block διάγραμμα του instruction receiver. Πιο αναλυτικά, μπορεί να παρατηρήσει κανείς την παρουσία ενός κυκλώματος που δέχεται σαν είσοδο το ρολόι του συστήματος και εξάγει ένα ρολόι με συχνότητα τέτοια, ώστε να μπορεί να γίνει η αποδοχή των δεδομένων σειριακά. Τα δεδομένα που έρχονται από τον υπολογιστή (οι εντολές δηλαδή που θα αποθηκευτούν στη μνήμη εντολών), μπαίνουν από το PIND0 και πάνε απευθείας στον instruction receiver. Η μονάδα ελέγχου Instruction Memory Control, είναι αυτή που φροντίζει για τη σωστή σειριακή υποδοχή των δεδομένων μέσα στη μνήμη εντολών. Όταν συγκεντρωθούν όλοι οι ASCII χαρακτήρες που σχηματίζουν μια εντολή (γεγονός που ελέγχεται από το σήμα Byte_Counter), τότε αναλαμβάνει, σε συνεργασία με τη γεννήτρια του ρολογιού, να χειριστεί τα σήματα imem_wr_en, imem_wraddr και imem_din με τέτοιο τρόπο, ώστε να γίνει η εγγραφή της εντολής στη παρούσα θέση μνήμης και συγχρόνως να ετοιμάσει την εγγραφή της επόμενης εντολής στην παρακάτω θέση.




Σχήμα 4.18 – Instruction Receiver.

Ο instruction receiver ουσιαστικά είναι ένα κύκλωμα, το οποίο γενικά λειτουργεί όπως και ο receiver του UART, αλλά για λόγους σαφήνειας, να σημειωθεί ότι τα 2 κυκλώματα είναι τελείως ξεχωριστά. Πιο αναλυτικά, αρχικά στην είσοδό του υπάρχει λογικό 1, όπως συμβαίνει στη σειριακή μετάδοση. Μόλις ξεκινήσει η μετάδοση κάποιου byte, θα πρέπει πρώτα να αναγνωριστεί το start bit και κατόπιν να αρχίσει η αποθήκευση των εντολών στη μνήμη εντολών. Στον πίνακα 4.7 φαίνονται συνολικά όλοι οι ASCII χαρακτήρες που αναμένονται να σταλούν από τον υπολογιστή και δίπλα η αντίστοιχη ακολουθία από bits που στέλνονται σειριακά.

| ASCII Χαρακτήρας | Ακολουθία από bits | ASCII Χαρακτήρας | Ακολουθία από bits |
|---|---|---|---|
| 0 | 00110000 | A | 01000001 |

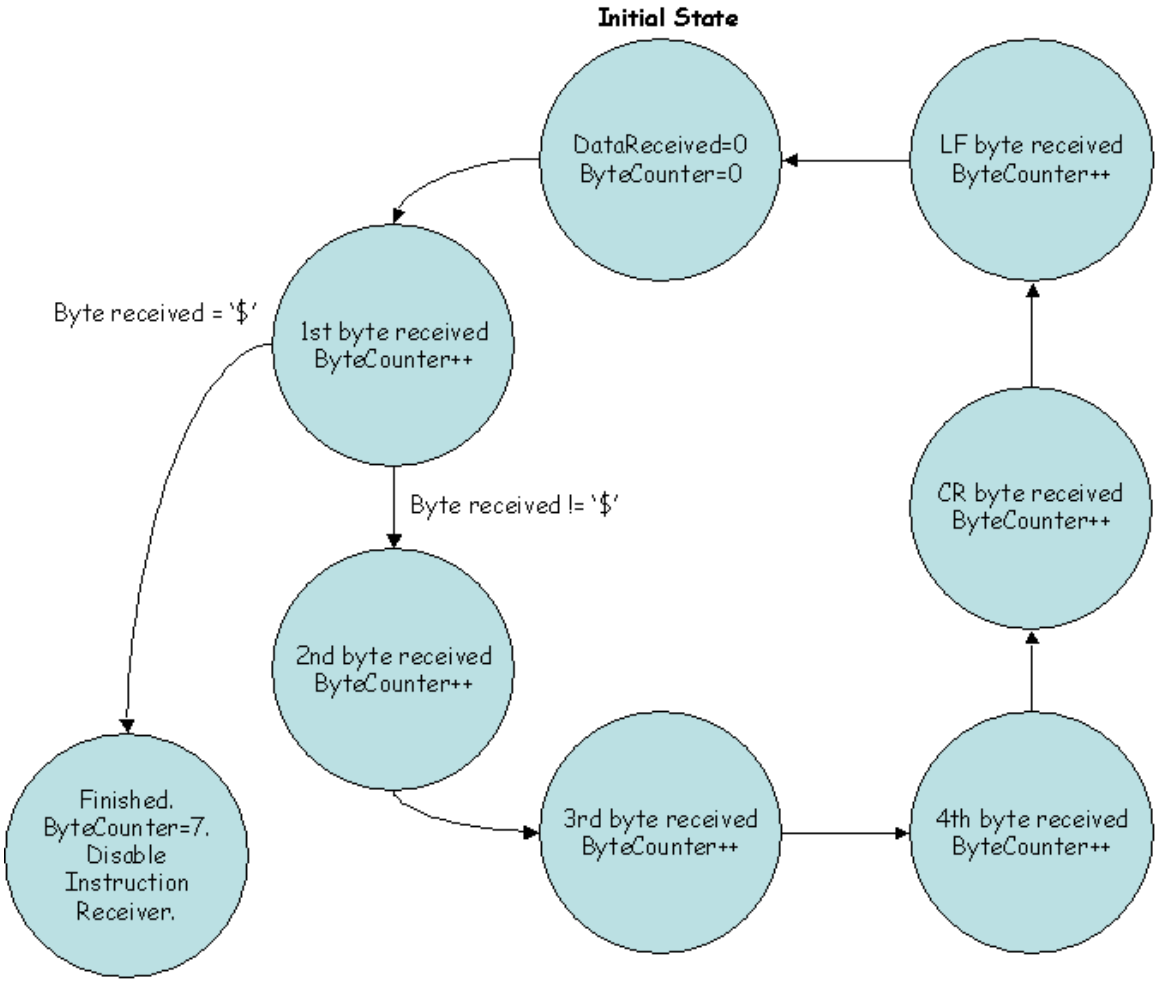
| | | | |
|---|---|---|---|
| 1 | 00110001 | B | 01000010 |
| 2 | 00110010 | C | 01000011 |
| 3 | 00110011 | D | 01000100 |
| 4 | 00110100 | E | 01000101 |
| 5 | 00110101 | F | 01000110 |
| 6 | 00110110 | LF | 00001010 |
| 7 | 00110111 | CR | 00001101 |
| 8 | 00111000 | $ | 00100100 |
| 9 | 00111001 | | |

Πίνακας 4.7 – Αναμενόμενοι χαρακτήρες από τον υπολογιστή.

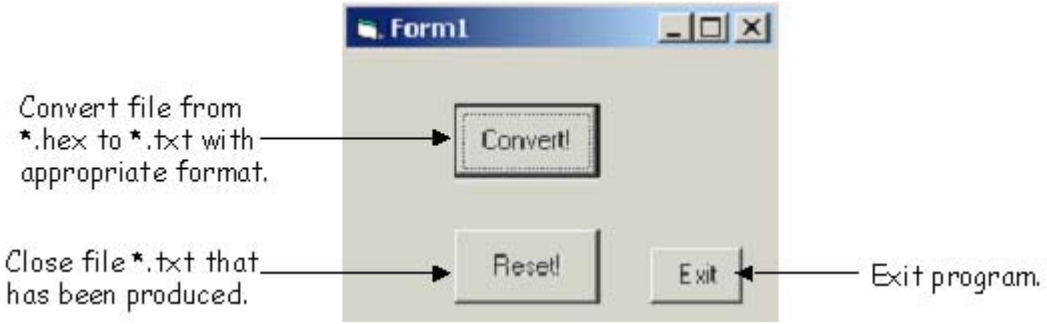
Για να σταλεί σωστά το πρόγραμμα που θέλει ο χρήστης σειριακά, θα πρέπει να έχει ένα συγκεκριμένο format το αρχείο που το περιέχει. Πιο συγκεκριμένα, σε κάθε γραμμή θα πρέπει να υπάρχουν μόνο οι 4 ASCII χαρακτήρες που αντιπροσωπεύουν μια εντολή και μετά την τελευταία εντολή, θα πρέπει να υπάρχει ο ASCII χαρακτήρας $, που σημαίνει ότι τελείωσε το πρόγραμμα. Προκειμένου να παραχθεί το συγκεκριμένου είδους .txt αρχείο, έχει σχεδιαστεί ένα μικρό πρόγραμμα σε Visual Basic που φαίνεται στο σχήμα 4.20 και αναλαμβάνει να διαβάσει το .hex αρχείο που εξάγει το AVR Studio και να το μετατρέψει όπως ακριβώς πρέπει. Ο χρήστης απλά πρέπει να υποδείξει στο πρόγραμμα ποιο είναι το αρχείο .hex που έχει εξάγει το AVR Studio και ύστερα αυτό αναλαμβάνει να το μετατρέψει στο πρέπων .txt αρχείο.

Στο σχήμα 4.19 φαίνεται η FSM λειτουργίας του instruction receiver. Αρχικά ξεκινάει από την κατάσταση όπου τα σήματα DataReceived (κρατάει τα δεδομένα που έρχονται) και ByteCounter (μετράει τους ASCII χαρακτήρες που έρχονται) είναι 0. Η είσοδος στο PIND0 αρχικά είναι σε λογικό 1 και για να ξεκινήσει η μετάδοση, θα πρέπει να εμφανιστεί το start bit που είναι το λογικό 0 για κάποιο συγκεκριμένο χρονικό διάστημα. Μόλις ληφθεί το $1^{o}$ byte, εξετάζεται αν ήταν το $ ή όχι. Αν ήταν, τότε σημαίνει ότι τελείωσε το πρόγραμμα που θέλει να μετάδοση σειριακά ο χρήστης, οπότε γίνεται απενεργοποίηση του instruction receiver, ο ByteCounter παίρνει την τιμή 7, ξεκινάει η εκτέλεση του προγράμματος και από εκεί και ύστερα ο instruction receiver είναι απενεργοποιημένος. Για να ενεργοποιηθεί ξανά και να σταλεί καινούριο πρόγραμμα σειριακά, θα πρέπει να γίνει reset στο σύστημα. Αν δεν ήταν $, τότε λαμβάνονται με τη σειρά και τα υπόλοιπα bytes, φροντίζοντας πάντα να αυξάνεται η τιμή του ByteCounter, η οποία παραμένει συνεχώς μικρότερη από 7. Αφού ληφθούν τα CR και LF bytes, η FSM επανέρχεται στην αρχική κατάσταση και περιμένει τη σειριακή μετάδοση είτε του

χαρακτήρα $, είτε της επόμενης εξάδας από bytes (4 bytes για την εντολή και 2 bytes οι CR και LF) (αρχεία **myInstrRec2.vhd** και **myInstrRecClkGen.vhd**).




Σχήμα 4.19 – FSM του instruction receiver.




Σχήμα 4.20 – Interface του προγράμματος σε VB τη μετατροπή του *.hex αρχείου σε *.txt.

# 5 Απεικόνιση του μοντέλου σε αναδιατασσόμενη λογική και βελτίωση 1ης έκδοσης.

Στο προηγούμενο κεφάλαιο, αναλύθηκε ο τρόπος σχεδίασης του μοντέλου του AT90S8515 και παρουσιάστηκαν λεπτομερώς όλα τα επιμέρους κυκλώματά του. Στο παρόν κεφάλαιο, γίνεται αναφορά στο ποσοστό που καταλαμβάνει η σχεδίαση σε διάφορες FPGA, καθώς και ορισμένες βελτιώσεις που έγιναν, προκειμένου να υπάρξουν καλύτερα αποτελέσματα. Αρχικά γίνεται μια απεικόνιση της 1ης έκδοσης της σχεδίασης σε κάποιες FPGA και γίνεται σχολιασμός των αποτελεσμάτων. Κατόπιν, αναφέρονται ορισμένες αλλαγές που έγιναν, οι οποίες σε επόμενη έκδοση μπορούν να οδηγήσουν σε μια σαφώς βελτιωμένη σχεδίαση. Ύστερα, παρουσιάζεται η νέα απεικόνιση της (βελτιωμένης) σχεδίασης και γίνεται σύγκριση με αυτήν που υπήρχε πριν γίνουν οι αλλαγές και τέλος γίνεται αναφορά στη διαδικασία που ακολουθήθηκε για την απεικόνιση του μοντέλου IP core σε FPGAs από την εταιρεία Altera.

Στην 1η έκδοση της σχεδίασης έχει γίνει προσπάθεια να μη χρησιμοποιηθεί καμία βιβλιοθήκη συγκεκριμένης εταιρίας στη σχεδίαση, με σκοπό το μοντέλο να μπορεί να χρησιμοποιηθεί σε FPGAs διαφορετικών εταιριών. Είναι αλήθεια πως, αν γινόταν χρήση κάποιων κομματιών κώδικα συγκεκριμένης εταιρίας, η σχεδίαση δε θα μπορούσε να χρησιμοποιηθεί, όχι μόνο από άλλη εταιρία, αλλά πιθανώς ούτε και από ορισμένες οικογένειες FPGA της ίδιας εταιρίας. Για παράδειγμα, αν γινόταν χρήση κάποιων IP cores για μνήμες από τον Core Generator της Xilinx με σκοπό να χρησιμοποιηθεί η Virtex 1000 BG560 [13], τότε η σχεδίαση δε θα λειτουργούσε σε προγενέστερες οικογένειες αφού, σύμφωνα με τη Xilinx, οι μνήμες αυτές μπορούν να χρησιμοποιηθούν μόνο από τις Spartan 2, Virtex, VirtexE και Virtex2.

## 5.1 Απεικόνιση της 1ης έκδοσης της σχεδίασης σε αναδιατασσόμενη λογική.

Με στόχο τη δημιουργία ενός μοντέλου που να είναι ανεξάρτητο από οποιαδήποτε εταιρία, στη σχεδίαση χρησιμοποιήθηκε VHDL που να μπορεί να «τρέξει» σε οποιοδήποτε CAD εργαλείο. Όπως αναφέρθηκε και πιο πριν άλλωστε, στο μόνο σημείο που θα μπορούσε να χρησιμοποιηθούν IP cores από συγκεκριμένη εταιρία είναι αυτό της σχεδίασης των μνημών. Ως αρχικός στόχος ήταν η σχεδίαση να χρησιμοποιηθεί σε μια Virtex 1000 BG560, η οποία έχει

ξεχωριστό κομμάτι που μπορεί να λειτουργήσει αποκλειστικά ως σύγχρονη μνήμη, οπότε η υλοποίηση γινόταν με βάση τη χωρητικότητα και τα χαρακτηριστικά αυτής της FPGA.

Παρόλα αυτά όμως, στην 1η έκδοση της σχεδίασης δεν χρησιμοποιήθηκαν IP cores της Xilinx, αφήνοντας ανεκμετάλλευτο τον ειδικό χώρο της Virtex για μνήμη. Αντίθετα, οι μνήμες εντολών (instruction memory – IMEM), δεδομένων (SRAM) και καταχωρητών (RF), υλοποιήθηκαν ως συνδυαστικά κυκλώματα και πιο συγκεκριμένα, ως πίνακες από std_logic_vectors (κατηγορία σημάτων στη VHDL) [3, 4, 8]. Το γεγονός αυτό έχει από τη μία πλευρά το πλεονέκτημα της ανεξαρτησίας από το χώρο των εταιριών που αναφέρθηκε πιο πριν, από την άλλη όμως δημιουργεί κυκλώματα μνήμης που είναι πολύ πιο αργά στην ανταπόκριση και συγχρόνως καταλαμβάνουν αρκετό χώρο στην FPGA.

Στο πίνακα 5.1 φαίνεται το % ποσοστό σε διάφορες FPGAs που καταλαμβάνει η σχεδίαση του μοντέλου. Ειδικά για τη Virtex 1000 BG560, η οποία ήταν και ο αρχικός στόχος, έγινε μια πιο αναλυτική μελέτη σχετικά με τη κατάληψη του χώρου της και βρέθηκε ότι το 37% χρησιμοποιείτο από την SRAM, 43% από την μνήμη εντολών με μόνο 256 θέσεις, 6% από την RF, ενώ 10% ήταν όλη η υπόλοιπη λογική. Το γεγονός αυτό οδηγεί στην κατάληψη του 96% της συγκεκριμένης FPGA, ποσοστό απαγορευτικό, αφού με δυσκολία θα χωρούσε οποιαδήποτε άλλη σχεδίαση μέσα σε αυτήν. Τελικά, τα παραπάνω αποτελέσματα οδήγησαν στη προσπάθεια μείωσης του ποσοστού κατάληψης χώρου στην FPGA, δοκιμάζοντας και μια σχεδίαση που να περιλαμβάνει IP cores της Xilinx σε ό,τι αφορά τις μνήμες.

| FPGAs | % ποσοστό κατάληψης | Slices | Ταχύτητα (MHz) |
|---|---|---|---|
| Virtex 1000 BG560 (-5) | 96 | 12288 | 15 |
| VirtexE 1000 BG560 (-6) | 96 | 12288 | 21 |
| Virtex2 1000BG 575 (-5) | >100 | 5120 | - |

Πίνακας 5.1 – Κατάληψη της σχεδίασης σε διάφορες FPGAs.

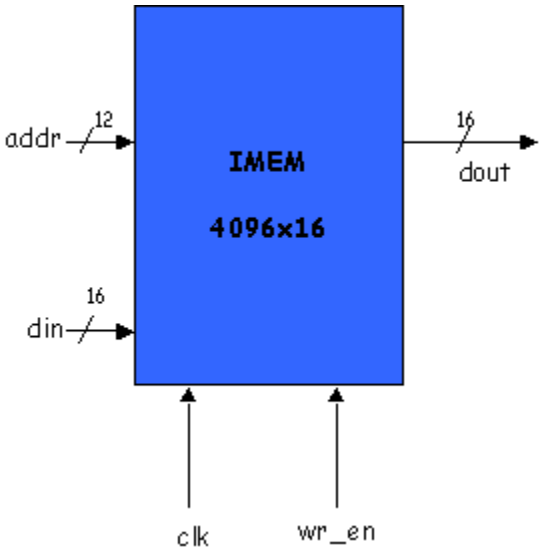
## 5.2 Βελτιώσεις και αλλαγές.

Έχοντας τώρα ως σκοπό τη μείωση του ποσοστού κατάληψης χώρου της FPGA, έγιναν ορισμένες αλλαγές στη σχεδίαση, οι οποίες θα μπορούσαν να οδηγήσουν σε βελτιωμένα αποτελέσματα όχι μόνο σε θέματα χώρου, αλλά και ταχύτητας. Έτσι έγιναν οι απαραίτητες

αλλαγές στη σχεδίαση, προκειμένου να προσαρμοστούν οι μνήμες που παράγει ο Core Generator της Xilinx [11, 12]. Οι τελευταίες, αυτή τη φορά, υλοποιούνται ως σύγχρονα κυκλώματα, τα οποία δεν καταλαμβάνουν χώρο στο κομμάτι της Virtex που είναι αφιερωμένο για την υλοποίηση της λογικής, αλλά σε αυτό που είναι ειδικά σχεδιασμένο για την υλοποίηση σύγχρονης μνήμης.

### 5.2.1 Βελτιωμένη έκδοση της μνήμης εντολών.

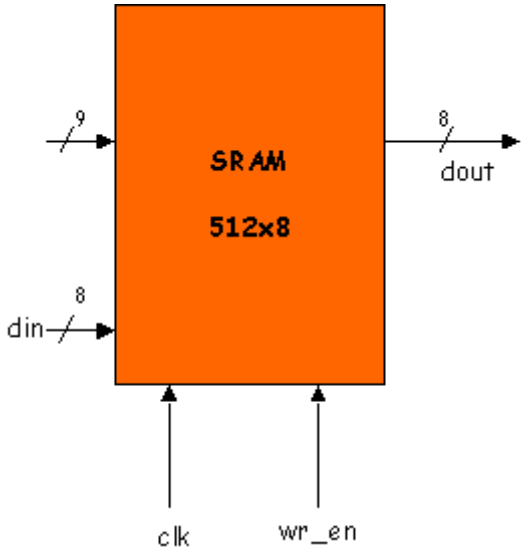
Για τη βελτίωση των χαρακτηριστικών της μνήμης εντολών, χρησιμοποιήθηκε ένα IP core από τον Core Generator της Xilinx και συγκεκριμένα η Single Port Block Memory [11], η οποία φαίνεται στο σχήμα 5.1. Όπως μπορεί να παρατηρηθεί, δεν υπάρχει ξεχωριστό σήμα για τη διεύθυνση ανάγνωσης και εγγραφής σε αντίθεση με την προηγούμενη σχεδίαση, απλά υπάρχει ένα σήμα addr το οποίο δηλώνει τη διεύθυνση που θα προσπελαστεί. Επίσης, δεν υπάρχει ξεχωριστό σήμα που να επιτρέπει ανάγνωση, παρά μόνο το wr_en για την εγγραφή. Έτσι, αν το τελευταίο είναι 1, τότε γίνεται εγγραφή στη διεύθυνση που δείχνει το σήμα addr, διαφορετικά γίνεται συνεχώς ανάγνωση από τη θέση που δείχνει το σήμα addr. Αξίζει να σημειωθεί ότι η $1^{η}$ έκδοση η ταχύτητα του κυκλώματος, για 256 θέσεις επί 16 bit στην κάθε μία, δεν ξεπερνούσε τα 25 MHz, ενώ στη $2^{η}$ έκδοση, με 4096 θέσεις και 16 bit στην κάθε μία, φτάνει περίπου τα 90MHz.




Σχήμα 5.1 – Βελτιωμένη μνήμη εντολών.

### 5.2.2 Βελτιωμένη έκδοση της μνήμης δεδομένων.

Για τη βελτίωση των χαρακτηριστικών της μνήμης δεδομένων, χρησιμοποιήθηκε επίσης το IP core Single Port Block Memory από τον Core Generator της Xilinx, η οποία φαίνεται στο σχήμα 5.2. Όπως μπορεί να παρατηρηθεί, και εδώ δεν υπάρχει ξεχωριστό σήμα για τη διεύθυνση ανάγνωσης και εγγραφής σε αντίθεση με την προηγούμενη σχεδίαση, απλά υπάρχει ένα σήμα addr το οποίο δηλώνει τη διεύθυνση που θα προσπελαστεί. Επίσης, δεν υπάρχει ξεχωριστό σήμα που να επιτρέπει ανάγνωση, παρά μόνο το wr_en για την εγγραφή. Έτσι, αν το τελευταίο είναι 1, τότε γίνεται εγγραφή στη διεύθυνση που δείχνει το σήμα addr, διαφορετικά γίνεται συνεχώς ανάγνωση από τη θέση που δείχνει το σήμα addr. Αξίζει να σημειωθεί ότι η $1^{η}$ έκδοση η ταχύτητα του κυκλώματος, δεν ξεπερνούσε τα 25 MHz, ενώ στη $2^{η}$ έκδοση, φτάνει περίπου τα 110 MHz.

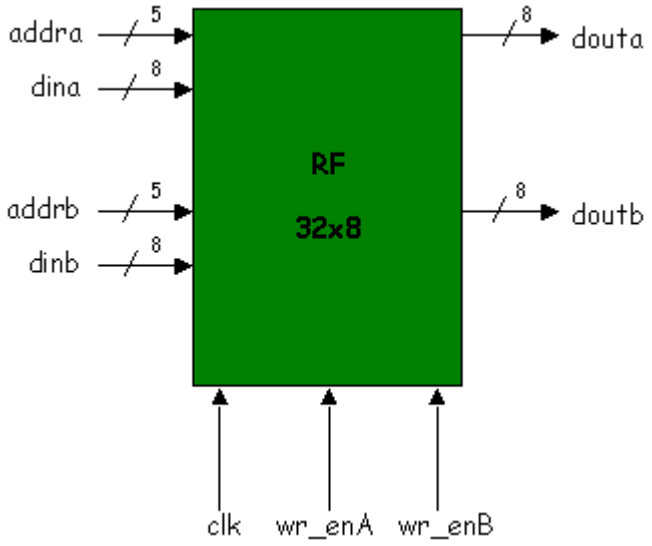



Σχήμα 5.2 – Βελτιωμένη μνήμη δεδομένων.

### 5.2.3 Βελτιωμένη έκδοση της RF.

Για τη βελτίωση των χαρακτηριστικών της RF, χρησιμοποιήθηκε ένα IP core από τον Core Generator της Xilinx και συγκεκριμένα η Dual Port Block Memory [12], η οποία φαίνεται στο σχήμα 5.3. Όπως μπορεί να παρατηρηθεί, ουσιαστικά πρόκειται για 2 Single Port Block

Memories ενωμένες, όπου στην κάθε μία δεν υπάρχει ξεχωριστό σήμα για τη διεύθυνση ανάγνωσης και εγγραφής σε αντίθεση με την προηγούμενη σχεδίαση, απλά υπάρχει ένα σήμα addr το οποίο δηλώνει τη διεύθυνση που θα προσπελαστεί. Επίσης, δεν υπάρχει ξεχωριστό σήμα που να επιτρέπει ανάγνωση, παρά μόνο το wr_en για την εγγραφή. Έτσι, αν το τελευταίο είναι 1, τότε γίνεται εγγραφή στη διεύθυνση που δείχνει το σήμα addr, διαφορετικά γίνεται συνεχώς ανάγνωση από τη θέση που δείχνει το σήμα addr. Αξίζει να σημειωθεί ότι η $1^{η}$ έκδοση η ταχύτητα του κυκλώματος δεν ξεπερνούσε τα 30 MHz, ενώ στη $2^{η}$ έκδοση, φτάνει περίπου τα 120MHz.




Σχήμα 5.3 – Βελτιωμένη RF.

## 5.3 Νέα απεικόνιση της σχεδίασης σε αναδιατασσόμενη λογική.

Έχοντας κάνει τις παραπάνω αλλαγές στη σχεδίαση, έγινε νέα υλοποίηση προκειμένου να συγκριθούν οι διαφορές στα αποτελέσματα. Πράγματι, με τη χρήση των παραπάνω IP cores του Core Generator της Xilinx, στη Virtex 1000 BG560, η κατάληψη χώρου έπεσε από το 96% στο 23%! Ο λόγος που συνέβη αυτό είναι επειδή η μνήμη εντολών, δεδομένων και καταχωρητών υλοποιήθηκαν στο ξεχωριστό κομμάτι της Virtex που είναι ειδικά για σύγχρονη μνήμη, καταλαμβάνοντας το 50% αυτού. Το αποτέλεσμα είναι να μειωθεί πολύ ο χώρος που θα χρειαζόταν στην FGPA, μιας και όλες οι παραπάνω μνήμες είχαν υλοποιηθεί σε ξεχωριστό κομμάτι. Έτσι, «σπαταλώντας» μόνο το ¼ περίπου της συγκεκριμένης FPGA, είναι πολύ πιο

εύκολο να χωρέσει και κάποια άλλη σχεδίαση. Αξίζει να σημειωθεί ακόμα, ότι ο χρόνος που χρειαζόταν για την υλοποίηση της 1ης έκδοσης ήταν περίπου 6 ώρες, ενώ της 2ης περίπου 50 λεπτά. Στον πίνακα 5.2 φαίνονται τα ποσοστά κατάληψης της νέας σχεδίασης και για άλλες FPGAs, γεγονός που μαρτυρά το μέγεθος της βελτίωσης.

| FPGAs | % ποσοστό κατάληψης | Slices | Ταχύτητα (MHz) |
|---|---|---|---|
| Virtex 1000 BG560 (-5) | 23 | 12288 | 15 |
| VirtexE 1000 BG560 (-6) | 25 | 12288 | 16 |
| Virtex2 1000BG 575 (-5) | 57 | 5120 | 62 |

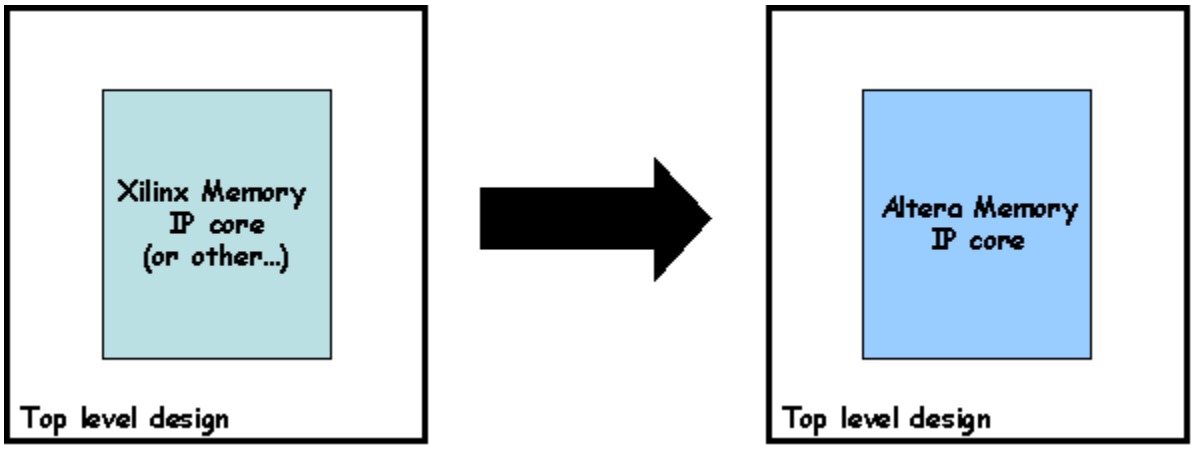
Πίνακας 5.2 – Κατάληψη της νέας σχεδίασης σε διάφορες FPGAs.

Τέλος, θα πρέπει να τονιστεί ότι, για να ολοκληρωθεί η 2η έκδοση του μοντέλου του AT90S8515, είναι απαραίτητο να γίνουν και κάποιες αλλαγές στον τρόπο αποκωδικοποίησης των εντολών. Πιο συγκεκριμένα, επειδή η «συμπεριφορά» των νέων μνημών που έχουν εξαχθεί από τον Core Generator είναι διαφορετική από αυτή των μνημών που χρησιμοποιούνταν στην 1η έκδοση, θα πρέπει να γίνουν ορισμένες αλλαγές στις FSMs αποκωδικοποίησης των εντολών. Ο λόγος που συμβαίνει αυτό, είναι επειδή οι καινούριες μνήμες είναι σύγχρονες και δεν έχουν ξεχωριστά σήματα για τη διεύθυνση και επίτρεψη ανάγνωσης και εγγραφής, σε αντίθεση με τις παλιές που είχαν και ήταν συνδυαστικά κυκλώματα.

## 5.4 Απεικόνιση του μοντέλου IP core σε FPGAs της Altera.

Όπως έχει αναφερθεί και σε προηγούμενες παραγράφους, η 1η έκδοση του μοντέλου IP core που έχει σχεδιαστεί, μπορεί να υλοποιηθεί σε FPGAs ανεξαρτήτως εταιρείας κατασκευής, αρκεί οι τελευταίες να έχουν αρκετό διαθέσιμο χώρο. Προκειμένου να μειωθεί το ποσοστό της κατάληψης του χώρου, στην προηγούμενη παράγραφο περιγράφτηκαν οι αλλαγές που έγιναν στις μνήμες και πιο συγκεκριμένα, η αντικατάστασή τους από IP cores μνήμες που είχαν παραχθεί από τον Core Generator της Xilinx. Στην παράγραφο αυτή γίνεται αναφορά στις αλλαγές που χρειάστηκαν να γίνουν, ώστε να αντικατασταθούν οι αρχικές μνήμες από IP cores μνήμες που παρήχθησαν από τον MegaFunction Wizard που βρίσκεται στο εργαλείο CAD MaxPlusII 9.02 της Altera και τελικά να απεικονιστεί το μοντέλο IP core του μικροελεγκτή σε FPGAs της συγκεκριμένης εταιρείας.

Αρχικά έγιναν ορισμένες αλλαγές σε ελάχιστα σημεία του κώδικα που έπρεπε, καθώς και στην ονομασία των αρχείων *.vhd μιας και το MaxPlusII επιβάλει οι ονομασίες των entities και των αρχείων, στο οποίο βρίσκεται το καθένα, να είναι ακριβώς ίδιες. Κατόπιν παρήχθησαν οι μνήμες που χρειάζονται από τον MegaFunction Wizard. Πιο συγκεκριμένα, σχεδιάστηκαν μια μνήμη εντολών 4096 θέσεων με 16 bit στην καθεμία και μια μνήμη δεδομένων 512 θέσεων με 8 bit στην καθεμία, ενώ για τη σχεδίασή τους χρησιμοποιήθηκε το παραμετροποιήσιμο IP core του MegaFunction Wizard "LPM_RAM_DQ". Για τη σχεδίαση της Register File, χρησιμοποιήθηκε το παραμετροποιήσιμο μοντέλο "DUAL_RAM" και παράχθηκε μια μνήμη 32 θέσεων με 8 bit στην καθεμία. Έτσι, ενώ οι μνήμες αλλάχτηκαν, τα top level αρχεία που χρησιμοποιούνται για την καθεμία από αυτές παραμένουν ακριβώς τα ίδια και με τις ίδιες εισόδους και εξόδους γεγονός που φαίνεται στο σχήμα 5.4.




Σχήμα 5.4 – Αντικατάσταση μνήμης.

Αφού έγιναν οι αλλαγές στις μνήμες, έγινε compile, όλων των αρχείων *.vhd που χρησιμοποιούνται, με επιτυχία. Το σημείο που συναντήθηκε πρόβλημα είναι όταν γίνεται πλήρης υλοποίηση του μοντέλου σε κάποια FPGA της Altera (στην προκειμένη περίπτωση είχε δοκιμαστεί η Flex 10KE). Πιο συγκεκριμένα, ενώ το compile και η υλοποίηση των αρχείων tristate.vhd και tristates.vhd γινόταν με επιτυχία, όταν ξεκινούσε η υλοποίηση του πλήρες μοντέλου IP core, προέκυπτε σφάλμα στο 1$^{ο}$ αρχείο και ειδικά στη χρήση των inout pins. Για την αποφυγή του προβλήματος αυτού, έγιναν οι απαραίτητες αλλαγές ώστε να μη χρειάζεται το tristate.vhd που υλοποιεί έναν τρικατάστατο οδηγητή και χρησιμοποιήθηκε το έτοιμο κύκλωμα "TRI" για τον τελευταίο, το οποίο υπάρχει στις βιβλιοθήκες της VHDL. Παρόλα αυτά όμως, το

MaxPlusII συνέχισε να παρουσιάζει σφάλμα στην υλοποίηση του μοντέλου IP core, γεγονός που ίσως οφείλεται σε κάποια παράμετρο του πρώτου που δεν έχει ληφθεί υπόψη. Έτσι, μόλις ξεπεραστεί το μικρό αυτό πρόβλημα, τότε θα μπορεί να χρησιμοποιηθεί το μοντέλο IP core και σε FPGAs της εταιρείας Altera.

# 6 Δοκιμή και πιστοποίηση του συστήματος.

Η δοκιμή και πιστοποίηση του συστήματος είναι το σημείο όπου γίνεται προσπάθεια να ανακαλυφθούν διάφορα λάθη που έγιναν κατά τη διάρκεια της σχεδίασης. Η διαδικασία που γίνεται για να ελεγχθεί η ορθή λειτουργία του μοντέλου είναι, αρχικά με τη βοήθεια των προγραμμάτων εξομοίωσης και κατόπιν η κατασκευή ενός κυκλώματος, με το οποίο παρατηρείται η συμπεριφορά της σχεδίασης σε πραγματικές συνθήκες.

Τα προγράμματα εξομοίωσης παρέχουν τη δυνατότητα στο σχεδιαστή να ελέγξει πότε και με ποιο τρόπο αλλάζει η τιμή των διαφόρων σημάτων. Είναι αλήθεια ότι τυχών σχεδιαστικά λάθη, μπορούν να δώσουν σε κάποιο σήμα τιμή μη καθορισμένη ή να το έχουν αφήσει ασύνδετο. Επίσης, μπορεί να συμβεί η τιμή κάποιου σήματος να οδηγείται από δύο διαφορετικά σήματα την ίδια στιγμή, γεγονός που οδηγεί σε σφάλμα κατά τη διάρκεια της υλοποίησης της σχεδίασης. Τέτοιου είδους λάθη είναι δυνατόν να ανακαλυφθούν με τα προγράμματα εξομοίωσης και μάλιστα υπό 2 βασικά είδη συνθηκών, ενώ σε αρκετά προγράμματα υπάρχουν ακόμα περισσότερα ενδιάμεσα στάδια συνθηκών λειτουργίας. Το $1^{ο}$ είδος είναι όταν το σύστημα λειτουργεί υπό ιδανικές συνθήκες, όπου τα σήματα δε θα παρουσίαζαν καθυστέρηση στην αλλαγή των τιμών τους και λέγεται functional simulation. Το $2^{ο}$ είδος είναι όταν πλέον λαμβάνονται υπόψη και οι καθυστερήσεις των σημάτων, γεγονός που συγκλίνει πολύ περισσότερο στις πραγματικές συνθήκες λειτουργίας και λέγεται timing simulation. Στο είδος αυτό μάλιστα, είναι δυνατόν να χρησιμοποιηθούν ειδικές βιβλιοθήκες που έχουν τις καθυστερήσεις των σημάτων για συγκεκριμένες FPGAs, με αποτέλεσμα ο σχεδιαστής να γνωρίζει ακριβώς τη συμπεριφορά του συστήματος όταν πρόκειται να υλοποιηθεί στη δική του FPGA.

Στο συγκεκριμένο κεφάλαιο, αρχικά παρουσιάζονται κάποιοι κώδικες που γράφτηκαν σε γλώσσα assembly για τον AT90S8515 μέσα από το AVR Studio και κατόπιν περιγράφεται το κύκλωμα που χρειάζεται να κατασκευαστεί, ώστε το σύστημα να μπορεί να λειτουργήσει σωστά.

## 6.1 Δοκιμές χωρίς τον instruction receiver.

Η αρχικοποίηση της μνήμης εντολών κατά τη διάρκεια των εξομοιώσεων είναι ένα πολύ βασικό στοιχείο, καθώς όσο πιο εύκολα γίνεται, τόσες περισσότερες δοκιμές μπορούν να πραγματοποιηθούν στο σύστημα. Αν εξαρχής όλες οι δοκιμές γινόντουσαν υπό την παρουσία του instruction receiver, θα έπρεπε κάθε φορά στην είσοδο του μοντέλου pind0, μέσω της οποίας προγραμματίζεται η μνήμη εντολών, να εισάγονται σε συγκεκριμένες χρονικές στιγμές οι τιμές των bits που θα στέλνονταν σειριακά από τον υπολογιστή. Κάτι τέτοιο είναι αρκετά χρονοβόρο και δυσλειτουργικό, ενώ παράλληλα είναι γνωστό ότι, κατά τη διάρκεια της πιστοποίησης της λειτουργίας ενός συστήματος, θα πρέπει να δοκιμαστούν πολλές περιπτώσεις σε σύντομο χρονικό διάστημα. Έτσι, αυτό που έγινε ήταν η αρχικοποίηση της μνήμης εντολών από ένα αρχείο .txt που εξαγόταν από το πρόγραμμα σε Visual Basic που περιγράφτηκε στην παράγραφο 4.7. Η διαδικασία αρχικοποίησης με αυτό τον τρόπο είναι ιδιαίτερα εύκολη και πολύ γρήγορη, καθώς το παραγόμενο .txt αρχείο, χάρη στη χρήση των βιβλιοθηκών ieee.std_logic_textio και std.text.io της VHDL [3, 4, 8], μπορεί αυτόματα να αρχικοποιήσει τη μνήμη εντολών. Να σημειωθεί ότι η τελευταία έχει μέγεθος 256 θέσεις επί 16 bits στην κάθε μία.

Προκειμένου, λοιπόν, να ελεγχθεί η συμπεριφορά του μοντέλου, κατά τη διάρκεια της σχεδίασης γινόντουσαν συνεχείς δοκιμές με πολλά προγράμματα που γραφόντουσαν στο AVR Studio. Παρακάτω φαίνεται ένα από αυτά, στο οποίο γίνεται χρήση των περισσοτέρων interrupts που υποστηρίζονται, ενός port, της στοίβας, ενός μετρητή με υποδιαίρεση συχνότητας, ρουτινών, του UART, ενώ προσπελαύνεται και η μνήμη δεδομένων:

```
.include "8515def.inc"

.org $000 ;hardware reset
rjmp RESET

.org $001 ;external interrupt 0
rjmp INT_0

.org $002 ;external interrupt 1
rjmp INT_1
```

```
.org $007 ;timer/counter0 overflow interrupt
rjmp OVF0

.org $009 ;UART receiver finished interrupt
rjmp RC

.org $00B ;UART transmitter finished interrupt
rjmp TC

RESET:
      ldi r16,low(RAMEND) ;initialize stack
      out SPL,r16
      ldi r16,high(RAMEND)
      out SPH,r16
      ldi r16,$d8
      out UCR,r16 ;enable UART RX, TX, RXI, TXI
      ldi r16,129
      out UBRR,r16 ;20MHz baud rate 9600

      ;enable external interrupts
      ldi r16,$c0
      out GIMSK,r16
      ldi r16,$0a
      out MCUCR,r16

      ;enable Timer/Counters interrupts
      ldi r16,$82
      out TIMSK,r16

      ;stop Timer/Counter0
      ldi r16,$00
      out TCCR0,r16
      clr r20
      clr r17
      clr r23
```

```
        ;initialize Z pointer
        clr r31
        ldi r30,$60

        ; initialize r21,r22, IO DDRD, DDRB
        ldi r21,10
        ldi r22,20
        ldi r16,2
        out DDRD,r16
        ldi r16,$ff
        out DDRB,r16

        sei

FOREVER:
        inc r20
        out PORTB,r20
        cpi r17,1 ;if r17=1, send what has been received
        breq SEND_UDR
        cpi r23,1 ;if r23=1 then call subroutine
        breq CALL_ROUTINE
        rjmp FOREVER

RC: ;receive and store to r24
        in r24,UDR
        add r21,r22
        inc r17
        reti

TC: ;subtract r21-r22 when transmitting is finished
        sub r21,r22
        reti

SEND_UDR: ;send contents of r24
        out UDR,r24
        clr r17
        rjmp FOREVER
```

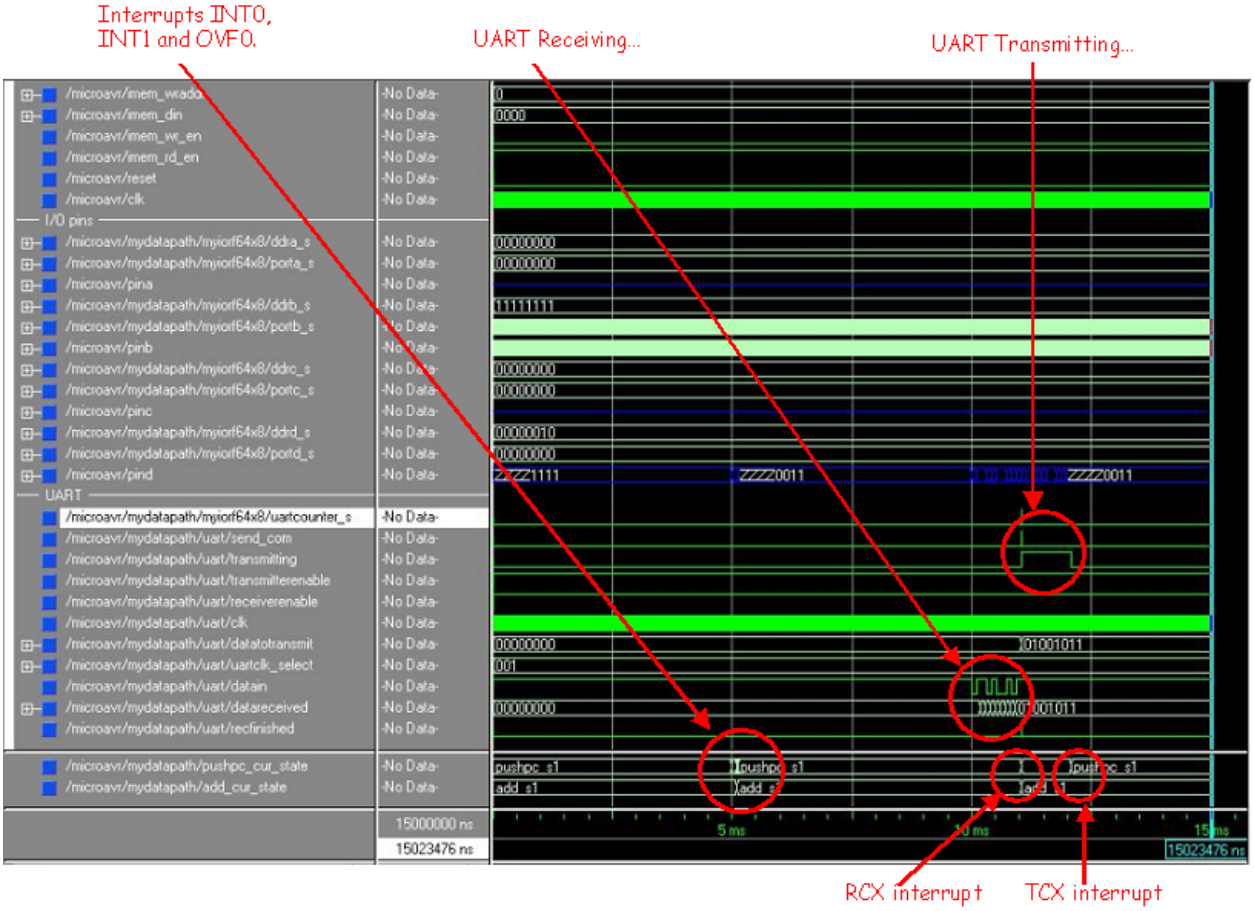
```
INT_0: ;external interrupt 0
      ;start Timer/Counter0 clk/8
      ldi r16,$02
      out TCCR0,r16
      reti

INT_1: ;external interrupt 1
      ;stop Timer/Counter0
      ldi r16,$00
      out TCCR0,r16
      reti

OVF0: ;timer/counter0 interrupt overflow
      inc r23
      reti

CALL_ROUTINE: ;when r23=1 MEMORY subroutine is called
      rcall MEMORY
      rjmp FOREVER

MEMORY: ;access data memory
      add r21,r22
      st Z+,r21
      st Z,r22
      ld r21,Z
      ld r22,-Z
      clr r23
      ret
```

Η λειτουργία του συγκεκριμένου προγράμματος είναι η εξής. Τα 2 εξωτερικά interrupts χρησιμοποιούνται για να ξεκινούν και να σταματούν τον Timer/Counter0 με υποδιαίρεση CLK/8 (κάθε 8 κύκλους, ο μετρητής αυξάνεται κατά 1). Επίσης, όταν μεταδοθεί σειριακά κάτι στο

μοντέλο, τότε αυτό το λαμβάνει και στη συνέχεια το μεταδίδει σειριακά πίσω και έτσι μπορεί να παρατηρηθεί η λειτουργία του UART. Έτσι, με αυτό το σχετικά απλό πρόγραμμα, γίνεται έλεγχος πολλών εντολών και λειτουργιών του μοντέλου σε ό,τι αφορά τη σωστή λειτουργία τους. Να σημειωθεί ότι για την πιστοποίηση της λειτουργίας του μοντέλου στις εξομοιώσεις στο Modelsim, δοκιμάστηκαν ακόμα και τα παρακάτω προγράμματα, τα οποία υπάρχουν στο CD της διπλωματικής μέσα στο φάκελο "Test Programs", μαζί με οδηγίες χρήσης για να δοκιμαστούν με το Modelsim:

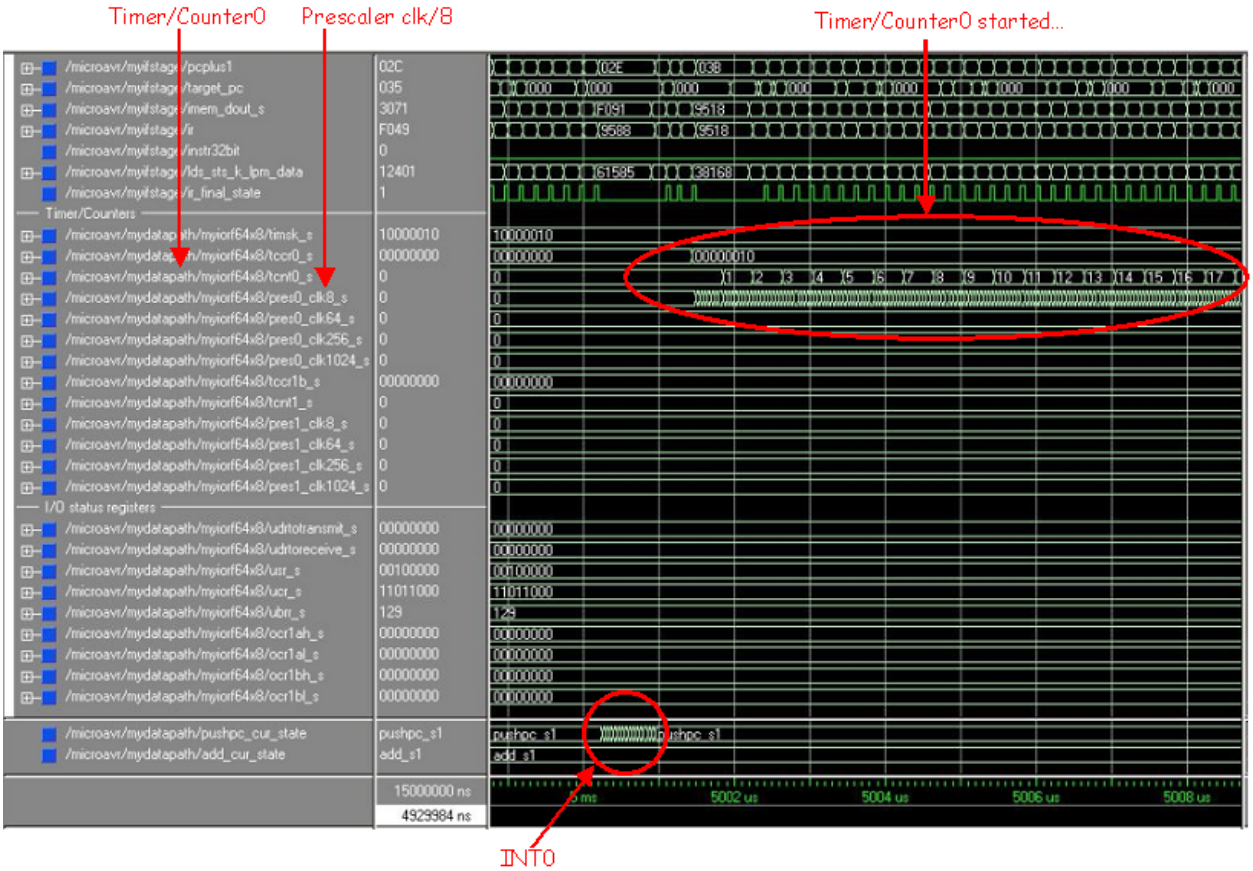
- "**Timer 1 Test**": Χρησιμοποιεί τον Timer/Counter1 και τα interrupts OVF1, OCR1A και OCR1B.
- "**LPM Test**": Χρησιμοποιεί την εντολή lpm και κάνει προσπέλαση στη μνήμη δεδομένων.
- "**Bubble Sort**": Application note της Atmel για την υλοποίηση του αλγόριθμου Bubble Sort [41].
- "**Multiply Divide**": Application note της Atmel για την υλοποίηση των πράξεων του πολλαπλασιασμού και της διαίρεσης μεταξύ 2 8-bit αριθμών και μεταξύ 2 16-bit αριθμών [42].

Στη συνέχεια φαίνονται κάποιες κυματομορφές που έχουν εξαχθεί από το Modelsim της Mentor Graphics, οι οποίες δείχνουν ορισμένα σημαντικά σημεία της εξομοίωσης του μοντέλου για το παραπάνω πρόγραμμα. Πιο συγκεκριμένα, στο σχήμα 6.1 φαίνεται ολόκληρη η εξομοίωση του προγράμματος. Στο 6.2 είναι η στιγμή που ενεργοποιείται το εξωτερικό interrupt 0 και ξεκινάει τον Timer/Counter0. Στο 6.3 είναι η στιγμή που ενεργοποιείται το εξωτερικό interrupt 1 και σταματάει τον Timer/Counter0. Στο 6.4 είναι η στιγμή που γίνεται υπερχείλιση του Timer/Counter0, οπότε ενεργοποιείται το αντίστοιχο interrupt. Στο 6.5 φαίνεται οι τιμές του PORTB και PINB, στα οποία καταλήγει η τιμή του καταχωρητή r20. Στο 6.6 είναι η στιγμή που τελειώνει η υποδοχή των δεδομένων σειριακά από το UART και ξεκινάει η εκτέλεση του αντίστοιχου interrupt, ενώ στο 6.7 είναι η στιγμή που τελειώνει η σειριακή αποστολή δεδομένων από το UART και ενεργοποιείται το αντίστοιχο interrupt. Τέλος, στο 6.8 φαίνεται ένα σημείο που εκτελούνται διάφορες εντολές και μπορούν να παρατηρηθούν οι τιμές που παίρνουν τα σήματα της RF, της IORF και της μνήμης δεδομένων.



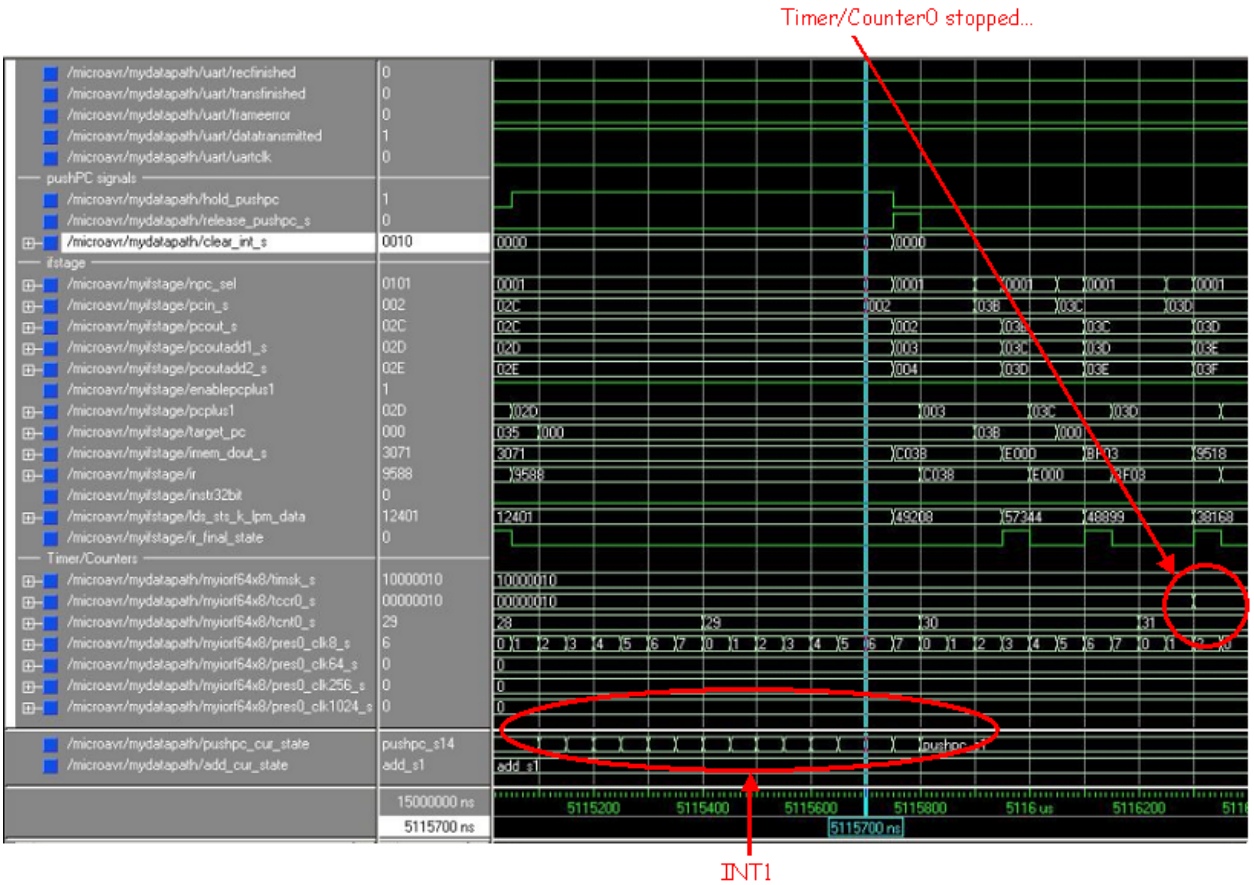

Σχήμα 6.1 – Πλήρης εξομοίωση το προγράμματος:

- Interrupts INT0, INT1 and OVF0: Σημείο της προσομοίωσης όπου συμβαίνουν τα εξωτερικά interrupts 0 και 1 και μετά γίνεται υπερχείλιση του Timer/Counter0 και συμβαίνει το OVF0 interrupt.
- UART Receiving: Σημείο όπου φαίνεται ο UART Receiver να δέχεται δεδομένα σειριακά.
- UART Transmitting: Σημείο όπου μεταδίδονται δεδομένα σειριακά.
- RCX interrupt: Σημείο όπου τελειώνει η σειριακή υποδοχή των δεδομένων και ενεργοποιείται το RCX interrupt.
- TCX interrupt: Σημείο όπου τελειώνει η σειριακή αποστολή των δεδομένων και ενεργοποιείται το TCX interrupt.



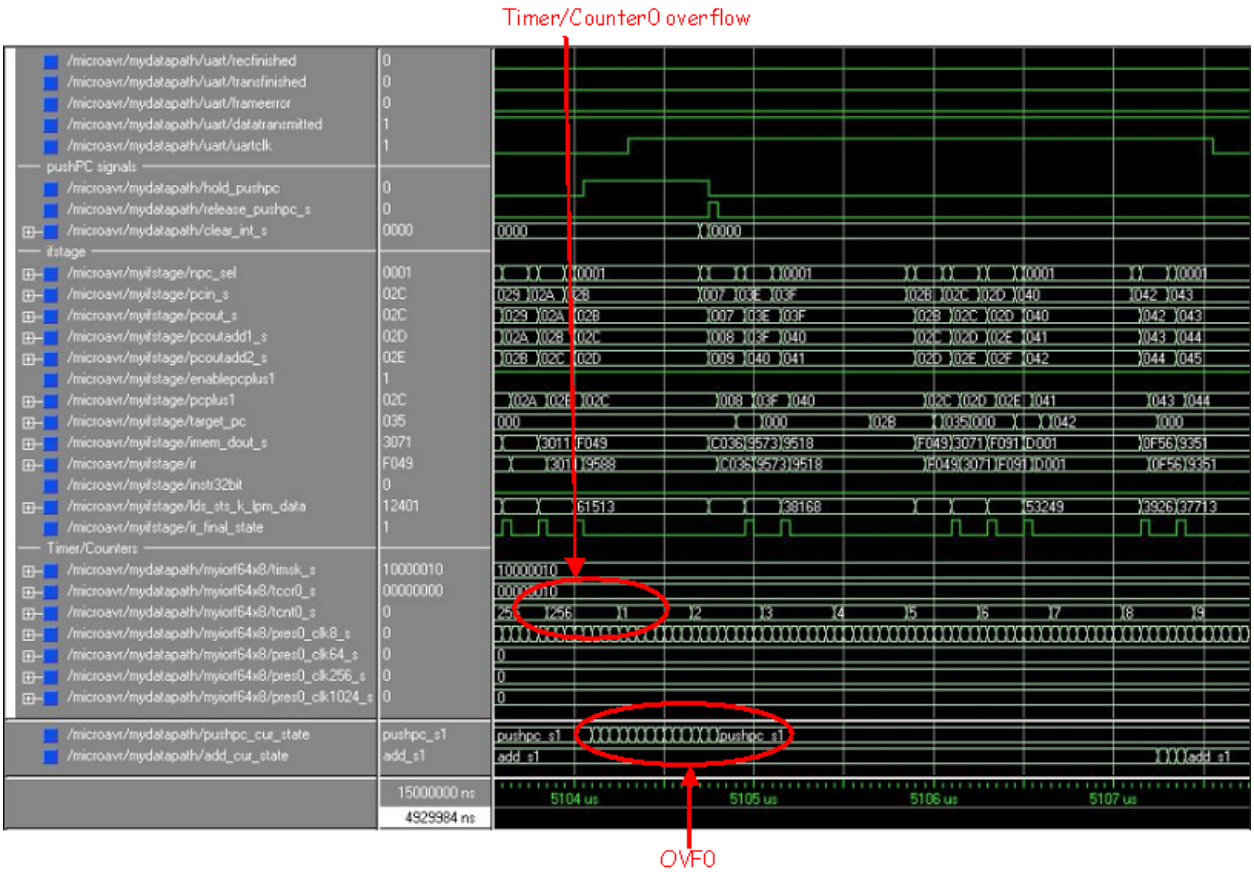

Σχήμα 6.2 – Εξωτερικό interrupt 0:

- Timer/Counter0: Φαίνεται η τιμή που έχει.
- Prescaler/8: Φαίνεται η τιμή του prescaler του Timer/Counter0 που έχει.
- Timer/Counter started: Έχει ξεκινήσει να μετράει ο prescaler και ο Timer/Counter0.
- INT0: Σημείο που γίνεται το εξωτερικό interrupt 0.



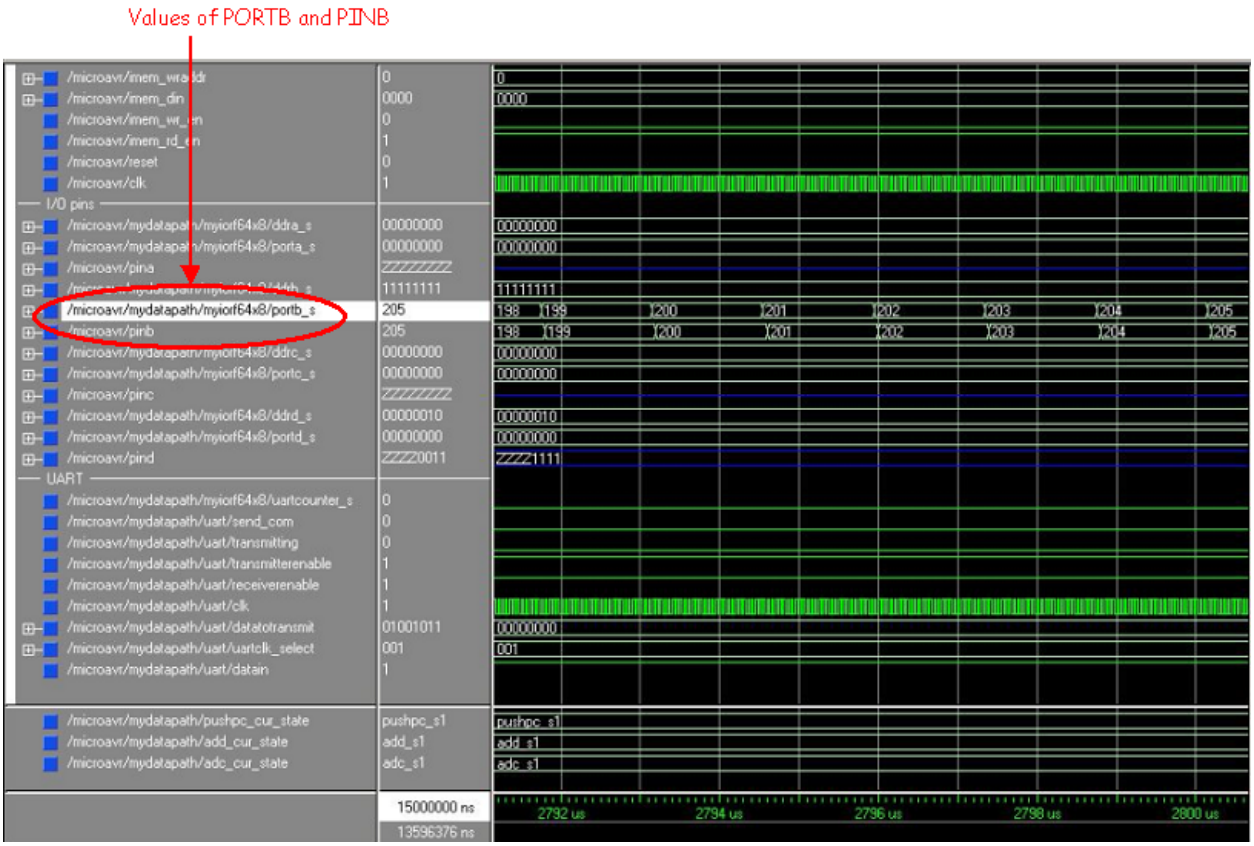

Σχήμα 6.3 – Εξωτερικό interrupt 1:

- INT1: Σημείο που γίνεται το εξωτερικό interrupt 1.
- Timer/Counter0 stopped: Έχει σταματήσει ο Timer/Counter0 και φαίνεται από την τιμή του IO καταχωρητή TCCR0.



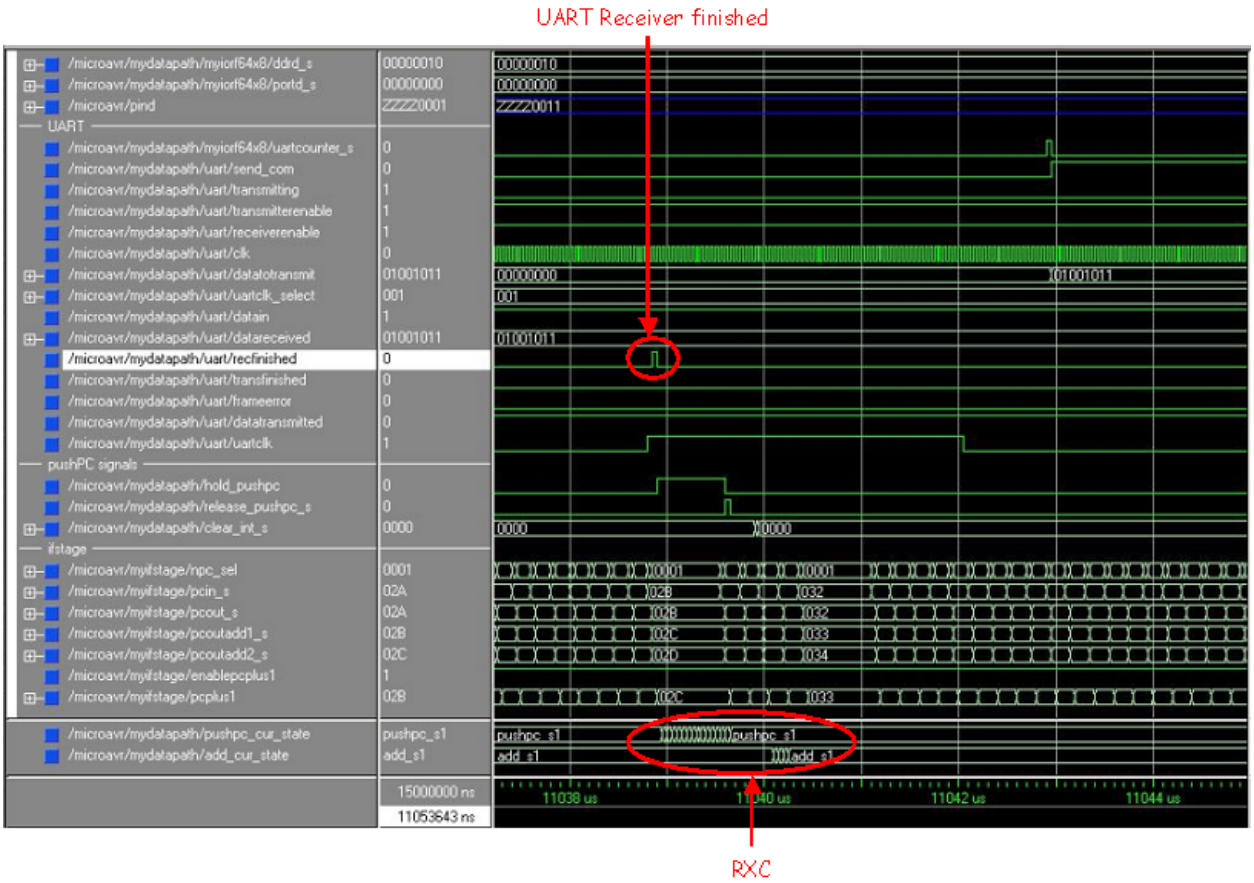

Σχήμα 6.4 – Υπερχείλιση του Timer/Counter0:

- Timer/Counter0 overflow: Σημείο που γίνεται υπερχείλιση του Timer/Counter0.
- OVF0: Μόλις γίνεται η υπερχείλιση, ξεκινάει το OVF0 interrupt.



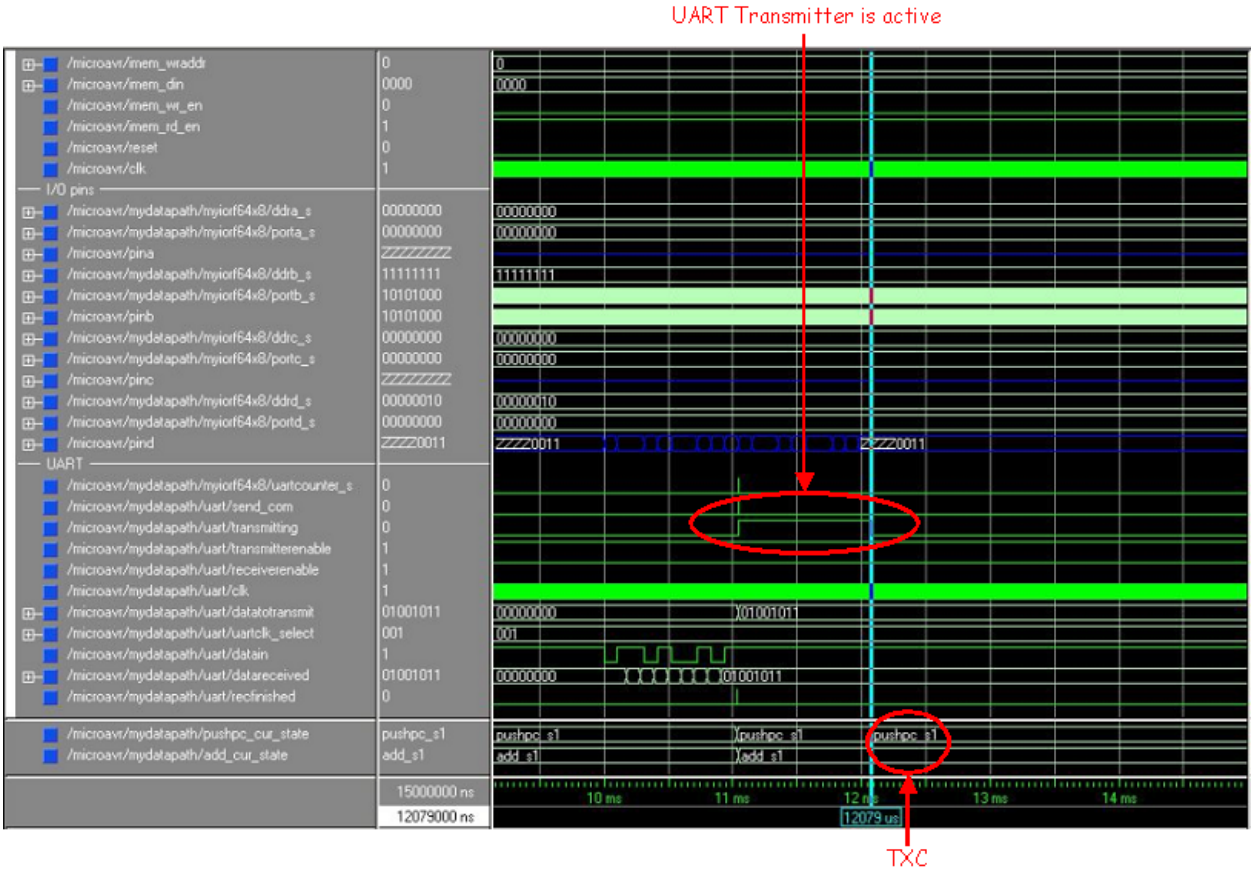

Σχήμα 6.5 – PORTB και PINB:

- Values of PORTB and PINB: Φαίνονται οι τιμές που παίρνουν οι PORTB και PINB αντίστοιχα.



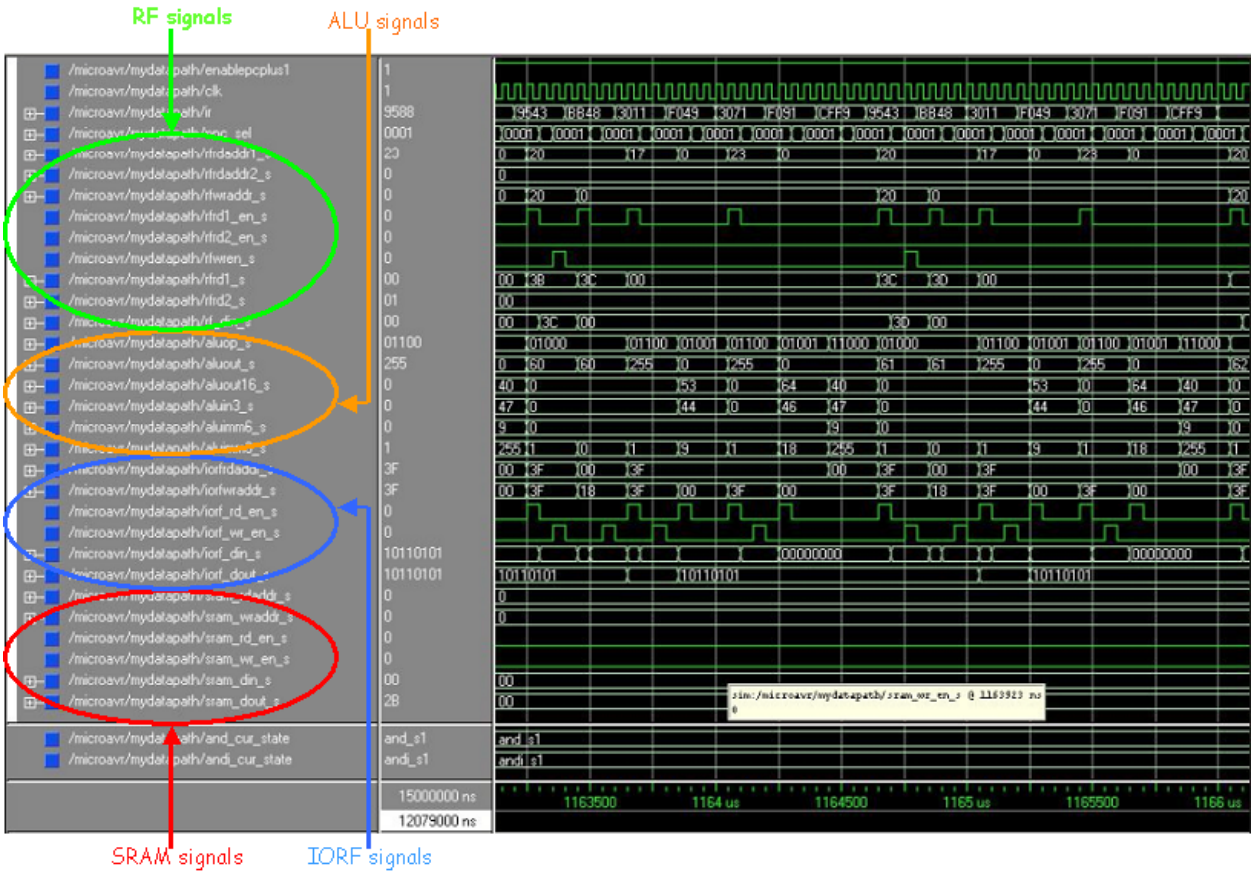

Σχήμα 6.6 – RXC interrupt:

- UART Receiver finished: Στιγμή που τελειώνει η σειριακή υποδοχή των δεδομένων.
- RCX: Μόλις τελειώνει η σειριακή υποδοχή, ενεργοποιείται το RCX interrupt.




Σχήμα 6.7 – TXC interrupt :

- UART Transmitter is active: Φαίνεται η χρονική περίοδος όπου ο UART Transmitter είναι ενεργοποιημένος.
- TXC: Μόλις ολοκληρωθεί η σειριακή αποστολή δεδομένων, ενεργοποιείται το TXC interrupt.




Σχήμα 6.8 – Εκτέλεση εντολών:

- RF signals: Σήματα που δείχνουν τη λειτουργία της RF.
- ALU signals: Σήματα που δείχνουν τη λειτουργία της ALU.
- SRAM signals: Σήματα που δείχνουν τη λειτουργία της SRAM.
- IORF signals: Σήματα που δείχνουν τη λειτουργία της IORF.

## 6.2 Δοκιμές μαζί με τον instruction receiver.

Στην προηγούμενη παράγραφο περιγράφτηκε ο τρόπος δοκιμής του συστήματος χωρίς την παρουσία του instruction receiver. Μετά τις παραπάνω δοκιμές, θεωρήθηκε απαραίτητο να γίνει μια εξομοίωση του μοντέλου στην περίπτωση που ο κώδικας μεταφερόταν σειριακά από τον υπολογιστή. Έτσι στη σχεδίαση προστέθηκε ο instruction receiver και συνδέθηκε με τον τρόπο που περιγράφτηκε στην παράγραφο 4.7 και στη συνέχεια δόθηκαν οι απαραίτητες τιμές στις κατάλληλες χρονικές στιγμές στην είσοδο pind0 του μοντέλου, προκειμένου η μνήμη εντολών να αρχικοποιηθεί με το παρακάτω πρόγραμμα:

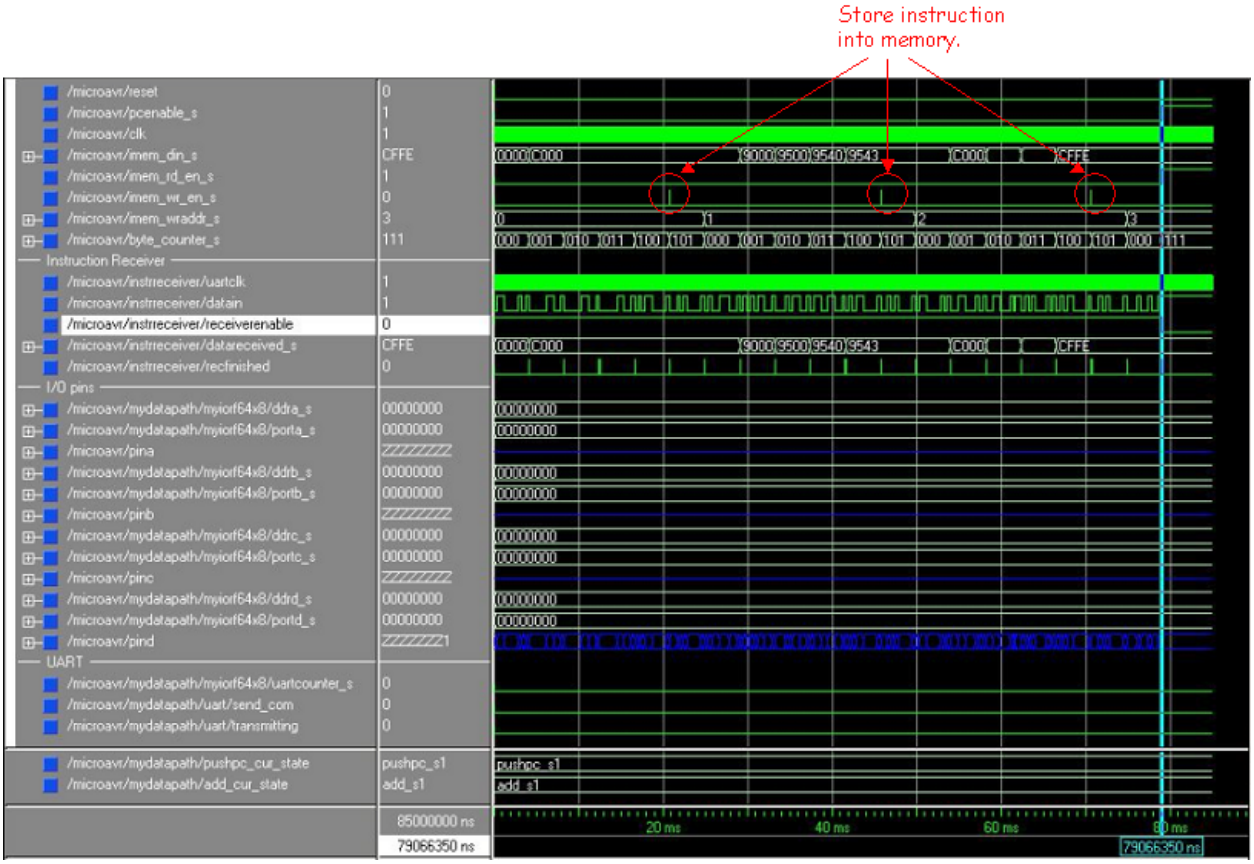
```
.org $000
rjmp RESET

RESET:
        inc r20
        rjmp RESET
```

Πρόκειται για ένα πολύ απλό πρόγραμμα που ως σκοπό έχει να ελέγξει τη σωστή λειτουργία του instruction receiver και αυτό που κάνει είναι να αυξάνει συνεχώς τον καταχωρητή r20. Στο σχήμα 6.9 φαίνεται η κυματομορφή που δείχνει τη λειτουργία του Instruction receiver. Συγκεκριμένα, μπορεί να παρατηρηθεί στην αρχή η διαδικασία που εκτελείται για τον προγραμματισμό της μνήμης εντολών και στη συνέχεια η εκκίνηση εκτέλεσης του προγράμματος. Στους κύκλους φαίνεται η στιγμή που κάποια εντολή αποθηκεύεται στη μνήμη εντολών ενεργοποιώντας και ύστερα απενεργοποιώντας το write enable και, ακριβώς από κάτω, η διεύθυνση στην οποία συμβαίνει κάθε φορά.

Επειδή, μετά την τελευταία εντολή του προγράμματος, στέλνεται ο ASCII χαρακτήρας $ (όπως έχει περιγραφτεί στην παράγραφο 4.7), ο instruction receiver απενεργοποιείται και αμέσως μετά, αρχίζουν να διαβάζονται οι εντολές που βρίσκονται μέσα στη μνήμη εντολών. Αυτό αρχίζει να συμβαίνει μετά τα 79066350 ns, όπου ο byte counter του instruction receiver παίρνει την τιμή 111 και στη συνέχεια γίνεται η εκκίνηση της εκτέλεσης του προγράμματος, γεγονός που φαίνεται στο σχήμα 6.10.

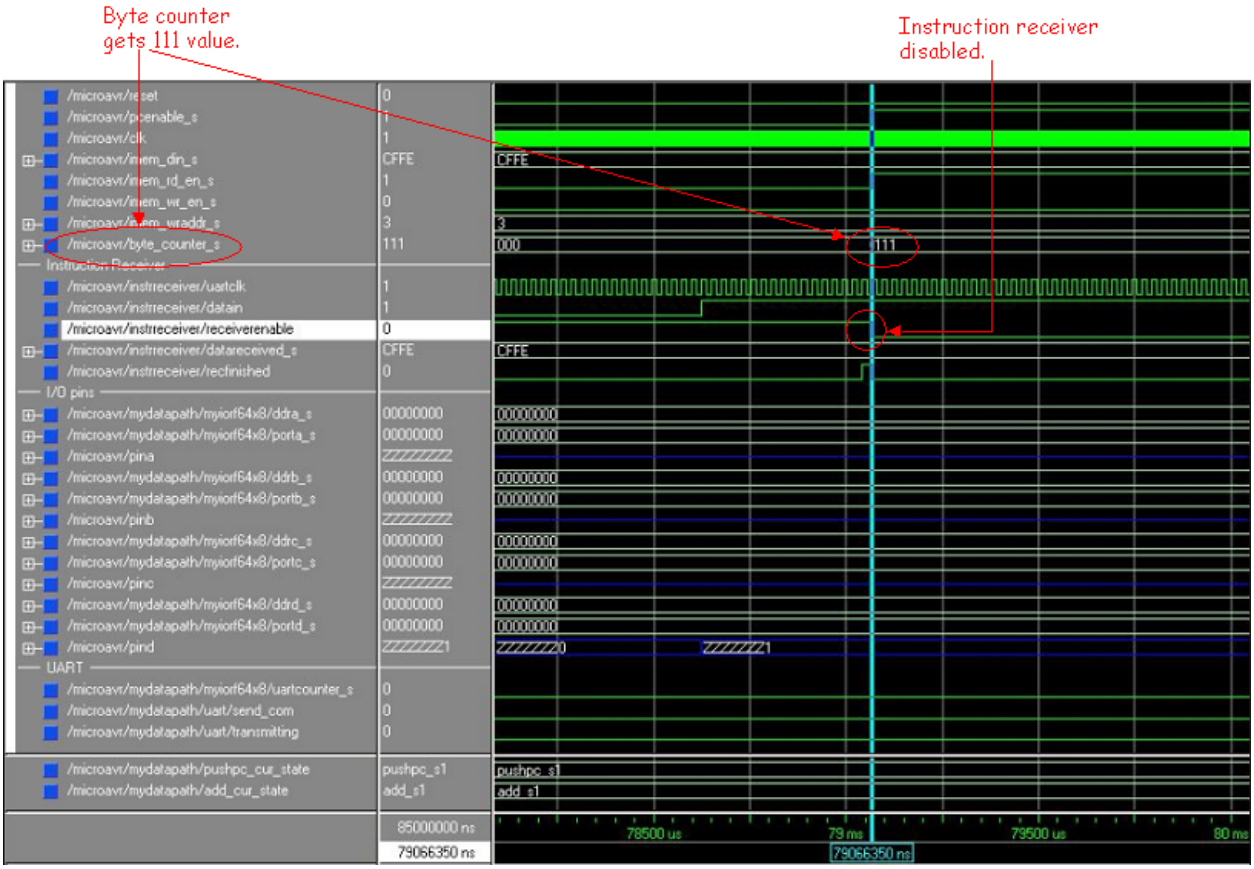
Τέλος, στο σχήμα 6.11 φαίνεται το σημείο ακριβώς μετά την απενεργοποίηση του instruction receiver, όπου έχει αρχίσει η εκτέλεση του προγράμματος. Το σήμα ir είναι η εντολή που εκτελείται, στα σήματα rfrdaddr1 και rfwraddr φαίνεται ο καταχωρητής που προσπελαύνεται και στο rfdin φαίνεται η καινούρια τιμή του καταχωρητή που αποθηκεύεται στην RF. Όπως μπορεί να παρατηρηθεί, αρχικά εκτελείται η εντολή C000 που είναι η **rjmp RESET** και σημαίνει τη μετάβαση στο σημείο όπου ξεκινάει ο κώδικας μετά από reset. Ύστερα εκτελείται η εντολή 9543, η οποία είναι η **inc r20**, οπότε διαβάζεται ο καταχωρητής r20, περνάει η τιμή του μέσα από την ALU όπου αυξάνεται κατά 1 και στη συνέχεια αποθηκεύεται η νέα τιμή

πάλι στον r20. Κατόπιν εκτελείται πάλι η **rjmp RESET** μόνο που αυτή τη φορά είναι η CFFE και γυρίζει τον PC μια θέση πίσω για να ξαναεκτελεστεί η **inc r20**, οπότε επαναλαμβάνεται συνεχώς ο κύκλος.



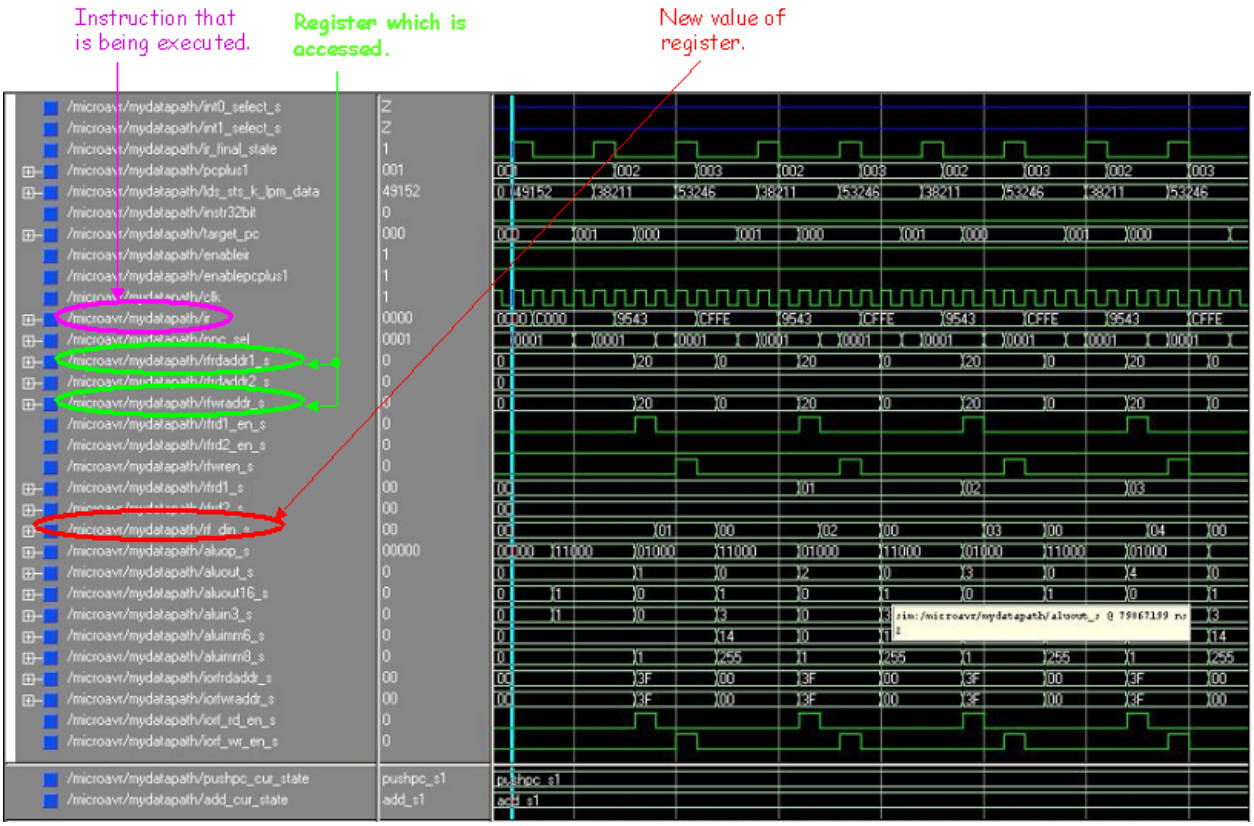

Σχήμα 6.9 – Κυματομορφή που δείχνει τη λειτουργία του instruction receiver:

- Store instruction into memory: Φαίνονται οι στιγμές που ενεργοποιείται το σήμα επίτρεψης εγγραφής στη μνήμη εντολών, προκειμένου να αποθηκευτούν οι εντολές που στέλνονται σειριακά.




Σχήμα 6.10 – Στιγμή που ο instruction receiver σταματάει να λειτουργεί:

- Byte Counter gets 111 value: Το σήμα byte counter που μετράει τον αριθμό των bytes που έρχονται, παίρνει τη δυαδική τιμή 111.
- Instruction Receiver disabled: Μόλις ο byte counter πάρει τη δυαδική τιμή 111, ο instruction receiver απενεργοποιείται.




Σχήμα 6.11 – Εκτέλεση του προγράμματος:

- Instruction that is being executed: Φαίνεται η εντολή που εκτελείται κάθε φορά.
- Register, which is accessed: Φαίνεται ο καταχωρητής που προσπελαύνεται κάθε φορά.
- New value of register: Φαίνεται η καινούρια τιμή του καταχωρητή.

## 6.3 Περιγραφή του κυκλώματος του συστήματος.

Στις δύο προηγούμενες παραγράφους, περιγράφτηκαν οι εξομοιώσεις που έγιναν προκειμένου να πιστοποιηθεί η λειτουργία του μοντέλου σε functional εξομοιώσεις. Στην παράγραφο αυτή παρουσιάζεται το κύκλωμα που θα πρέπει να κατασκευαστεί για να λειτουργήσει το μοντέλο στα 10 MHz και να μπορεί να προγραμματιστεί σειριακά από τον υπολογιστή. Να σημειωθεί ότι στην προκειμένη περίπτωση, χρησιμοποιείται η 1η έκδοση του μοντέλου με 256 θέσεις στη μνήμη εντολών.

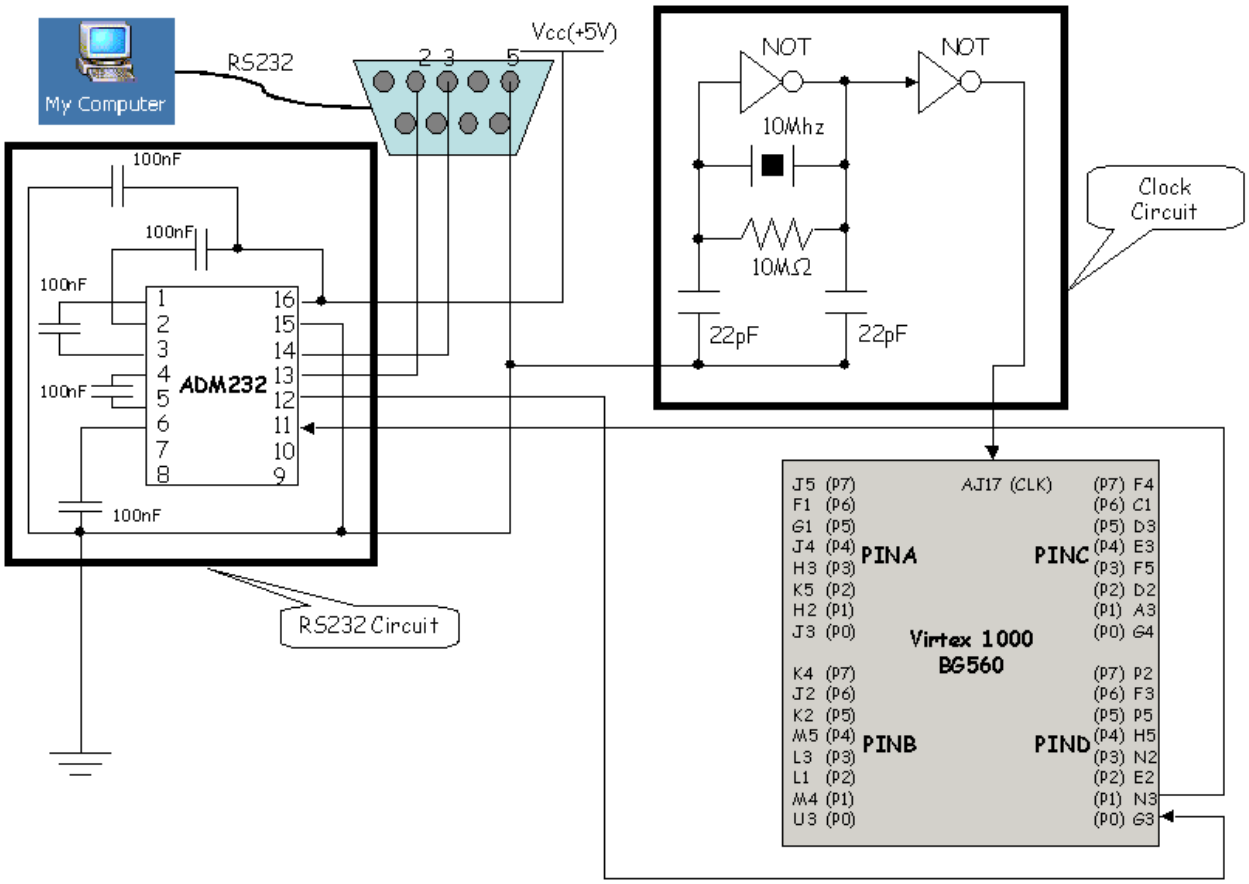
Στο σχήμα 6.12 φαίνεται το κύκλωμα που θα πρέπει να κατασκευαστεί για να λειτουργήσει το μοντέλο του AT90S8515 μέσα στη Virtex, ενώ στο 6.13 φαίνεται μια φωτογραφία του. Το αριστερό κύκλωμα (RS232 circuit) είναι αυτό που χρειάζεται για τη

σειριακή επικοινωνία μεταξύ του υπολογιστή και της FPGA. Πρόκειται για ένα κύκλωμα που χρησιμοποιεί το ολοκληρωμένο ADM232 [35]. Το τελευταίο ανήκει στην οικογένεια υψηλής ταχύτητας RS232 line drivers και μπορεί να προσφέρει ταχύτητα μετάδοσης μέχρι και 200KB/sec. Για να λειτουργήσει χρειάζεται τάση τροφοδοσίας +5V και με τη βοήθεια εξωτερικών πυκνωτών μπορεί να επιτύχει τάσεις +/-12V, με τις οποίες λειτουργεί η σειριακή του υπολογιστή. Τα δεδομένα που έρχονται από τον υπολογιστή, βγαίνουν από το pin 12 του ADM232 και πάνε στο pind0 του μοντέλου, ενώ αυτά που εξέρχονται από το τελευταίο πάνε στο pin 11 του ADM232 και ύστερα πάνε στο υπολογιστή.

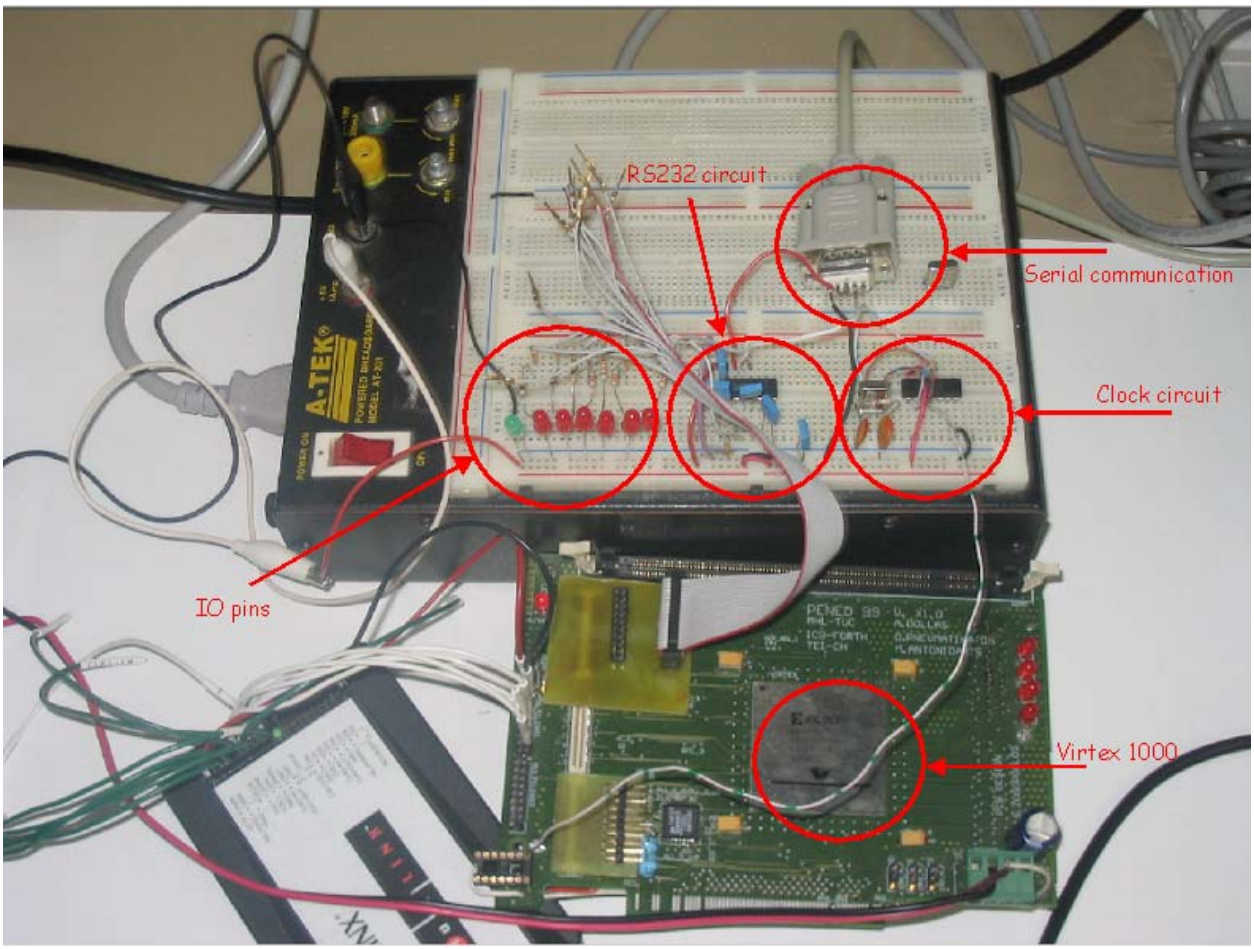
Το δεξί κύκλωμα (Clock circuit) είναι αυτό που αναλαμβάνει την παραγωγή «καλής ποιότητας» παλμού ρολογιού [6]. Έτσι, χρησιμοποιήθηκε ένας κρύσταλλος που λειτουργούσε σε αυτή τη συχνότητα και με τη βοήθεια της αντίστασης 10MΩ, των 2 πυκνωτών 22pF και των 2 αντιστροφέων μπορεί να παραχθεί παλμός ίδιας ποιότητας με αυτόν που θα προερχόταν από έναν ταλαντωτή. Πιο αναλυτικά, το κύκλωμα αυτό χωρίζεται σε 3 μικρότερα μέρη, όπου το $1^{ο}$ είναι αυτό με τον κρύσταλλο και τα παθητικά στοιχεία (πυκνωτές, αντίσταση) και αναλαμβάνει την παραγωγή του ημίτονου με μια συγκεκριμένη χρονική καθυστέρηση RC που προκύπτει από τα τελευταία, ενώ το $2^{ο}$ είναι ο $1^{ος}$ αντιστροφέας [17] και αναλαμβάνει την ενίσχυση και ανάδραση του σήματος που παράγεται, ώστε να υπάρχει καλής ποιότητας τετραγωνικός παλμός. Το $3^{ο}$ μέρος είναι ο $2^{ος}$ αντιστροφέας και χρησιμοποιείται για να μην συνδέσουμε φορτίο στο $2^{ο}$ μέρος μιας και κάτι τέτοιο θα μπορούσε να επηρεάσει αρνητικά τη συμπεριφορά του, οπότε στην έξοδό του έχουμε ένα τετραγωνικό παλμό που να μπορεί να χρησιμοποιηθεί σε ψηφιακά κυκλώματα.

Η έξοδος του κυκλώματος αυτού συνδέεται με το pin AJ17, που είναι μία από τις εισόδους για ρολόι στη Virtex. Επίσης, η επιλογή της σύνδεσης του κάθε pin του μοντέλου με συγκεκριμένο IO pin της Virtex [16, 21], έγινε μέσω του .ucf αρχείου που πρέπει να γραφτεί πριν από την υλοποίηση του κυκλώματος και φαίνεται στο σχήμα αυτό. Προφανώς, αν για κάποιο λόγο χρειαστεί να αλλάξουν οι αντιστοιχίες μεταξύ των pins του μοντέλου και αυτών της Virtex, θα πρέπει να γίνει εκ νέου υλοποίηση της σχεδίασης. Προκειμένου να επαλειφθεί η ορθότητα των συνδέσεων του κυκλώματος, σχεδιάστηκε ένα κύκλωμα UART σε VHDL, υλοποιήθηκε στη Virtex και λειτούργησε επιτυχώς, γεγονός που πιστοποιεί τη σωστή λειτουργία του.

Αφού δημιουργήθηκε το κατάλληλο *.ucf αρχείο, στο οποίο αναφέρεται ποια είσοδος ή έξοδος της σχεδίασης θα συνδεθεί σε ποιο pin της Virtex, κατασκευάστηκε το κύκλωμα του σχήματος 6.12, το οποίο είχε επαληθευτεί ότι λειτουργεί σωστά. Κατόπιν έγινε το implementation της σχεδίασης χρησιμοποιώντας το ISE 4.2i της Xilinx και, αφού τελείωσε, προγραμματίστηκε επιτυχώς η FPGA. Παρόλα αυτά όμως, δυστυχώς η τελική σχεδίαση δε δούλεψε σε ορισμένες δοκιμές που έγιναν, γεγονός που πιθανώς να οφείλεται σε κάποια κακή λειτουργία του κώδικα, η οποία δεν φαίνεται από τις εξομοιώσεις στο Modelsim 5.5d SE.




Σχήμα 6.12 – Κύκλωμα που θα πρέπει να κατασκευαστεί.




Σχήμα 6.13 – Φωτογραφία του κυκλώματος.

# 7 Συμπεράσματα και μελλοντικές επεκτάσεις.

Στο κεφάλαιο αυτό γίνεται αναφορά πάνω σε κάποια συμπεράσματα που εξήχθησαν μετά την ολοκλήρωση της σχεδίασης, καθώς και σε διάφορες επεκτάσεις που μπορούν να γίνουν στο μέλλον, ώστε να υπάρξουν καλύτερα αποτελέσματα σχετικά με την ταχύτητα και τη λειτουργικότητα του μοντέλου.

## 7.1 Συμπεράσματα.

Τα συμπεράσματα που εξήχθησαν μετά την ολοκλήρωση της σχεδίασης της 1$^{ης}$ έκδοσης του μοντέλου, προκύπτουν κυρίως από τις διαφορές του σε σχέση με τη βελτιωμένη έκδοση. Πιο συγκεκριμένα, έγινε σαφώς κατανοητό ότι η χρήση κάποιων έτοιμων IP cores από τον Core Generator της Xilinx βοήθησαν καταλυτικά στη δημιουργία μίας πολύ μικρότερης, από πλευράς κατάληψης όγκου στην FPGA, σχεδίασης, γεγονός που επιτρέπει την ενσωμάτωση και άλλων σχεδιάσεων μέσα στην τελευταία. Αυτό συμβαίνει, επειδή οι συγκεκριμένες μνήμες έχουν σχεδιαστεί με απώτερο σκοπό την ενσωμάτωσή τους σε ορισμένες FPGAs, οπότε χρησιμοποιούν ειδικές βιβλιοθήκες της Xilinx και καταλαμβάνουν τον χώρο που είναι ειδικά διαμορφωμένος για αυτές. Είναι αλήθεια ότι, το 96% που καταλάμβανε η 1$^{η}$ έκδοση, εξαιτίας των μνημών που είχαν χρησιμοποιηθεί, είναι τελείως απαγορευτικό για περαιτέρω χρήση της FPGA και δε θα ήταν λογικό να χρησιμοποιηθεί μια Virtex ως ένας μικροελεγκτής και μόνο.

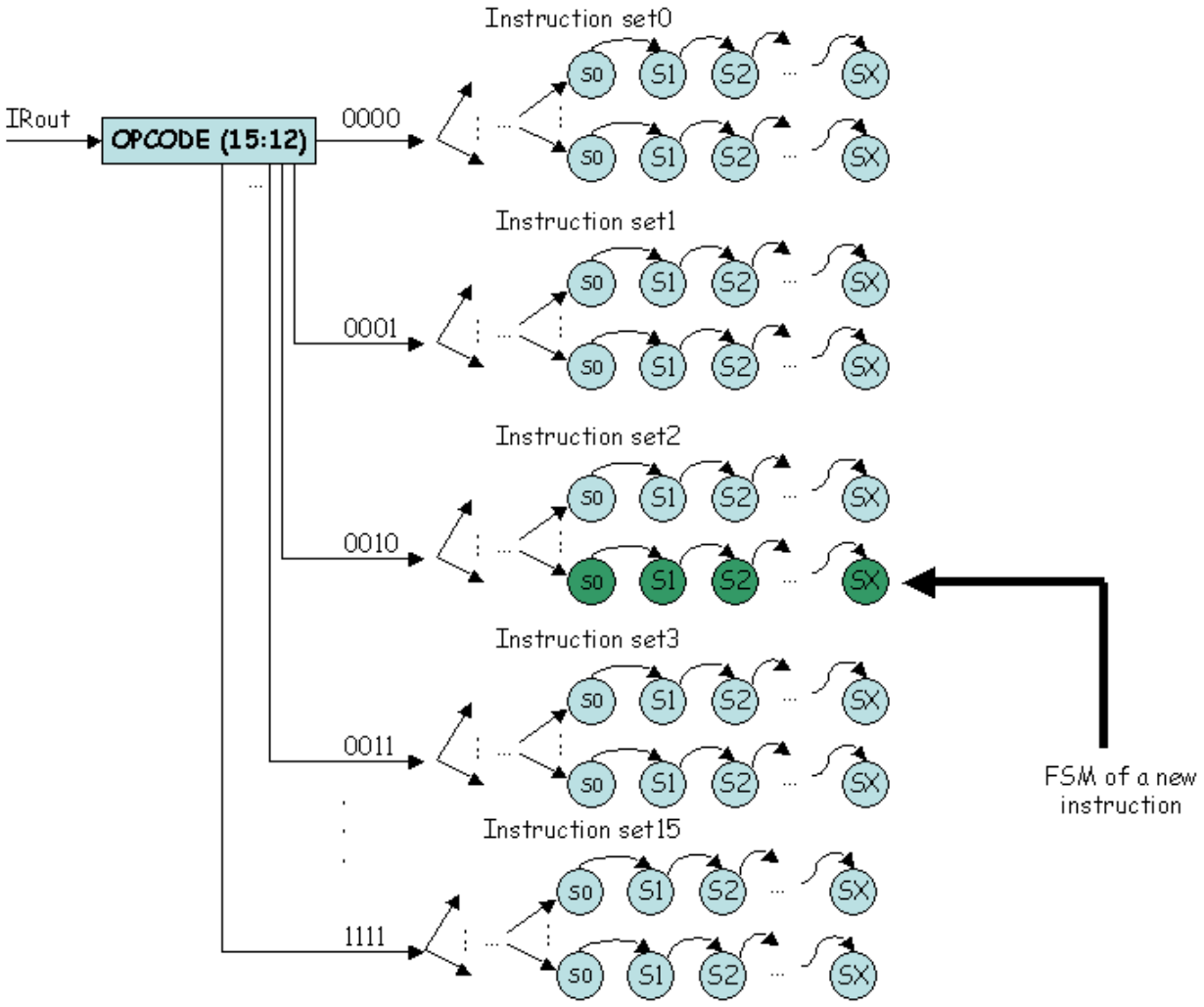
Από την άλλη πλευρά όμως, χρησιμοποιώντας τις έτοιμες μνήμες από τον Core Generator, χάνεται το πλεονέκτημα της σχεδίασης ενός μοντέλου που μπορεί να χρησιμοποιηθεί από οποιοδήποτε CAD εργαλείο και FPGAs από διαφορετικές εταιρίες. Έτσι, το βασικό συμπέρασμα που προκύπτει είναι ότι κάθε σχεδίαση έχει ορισμένα πλεονεκτήματα και μειονεκτήματα, αν και η βελτιωμένη έκδοση, όταν ολοκληρωθεί, ίσως να είναι προτιμότερη για μελλοντική χρήση, μιας και είναι πολύ πιο μικρή και συμφέρουσα από πλευράς κατανάλωσης πόρων.

## 7.2 Μελλοντικές επεκτάσεις.

Οι επεκτάσεις που μπορούν να γίνουν και στις δύο εκδόσεις είναι αρκετές και μπορούν να προσφέρουν ακόμα μεγαλύτερη λειτουργικότητα στο μοντέλο. Άλλωστε, η σχεδίαση της

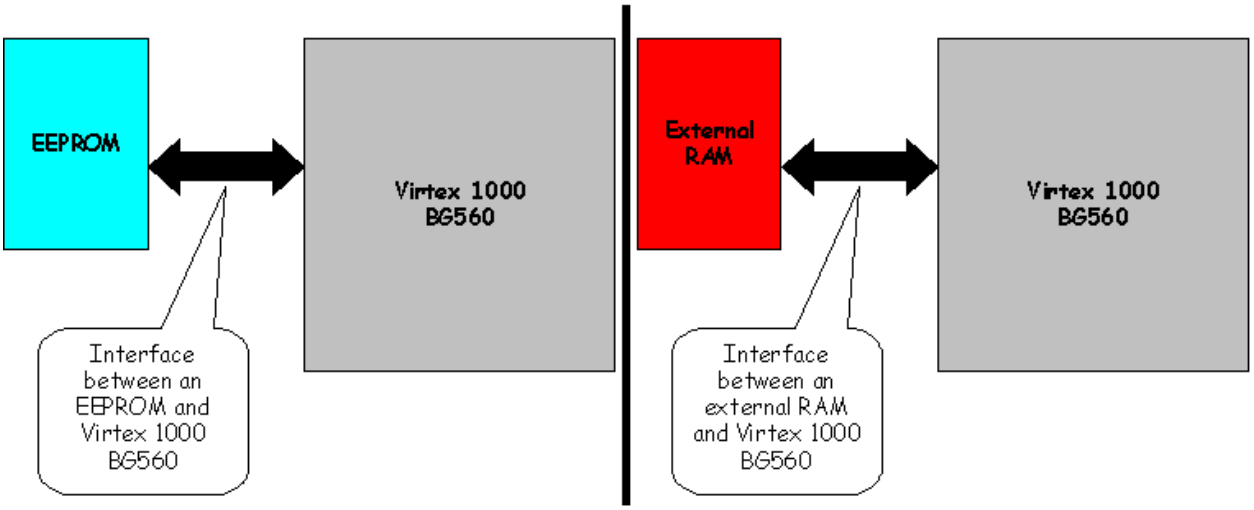
αρχιτεκτονικής του έγινε με βασικό γνώμονα την εύκολη επεκτασιμότητα, προκειμένου να είναι δυνατή η υποστήριξη περισσότερων λειτουργιών.

Μία βασική επέκταση που μπορεί να γίνει, είναι η αύξηση των εντολών που θα υποστηρίζονται. Ως τώρα, μπορούν να χρησιμοποιηθούν οι περισσότερες εντολές όλων των οικογενειών AVR που είναι μικρότερες ή ίσες του AT90S8515. Για να γίνει η προσθήκη κάποιας επιπλέον εντολής, το μόνο που χρειάζεται είναι να συμπεριληφθεί στο δέντρο αποκωδικοποίησης των εντολών και στη συνέχεια να σχεδιαστεί η FSM της που θα επικοινωνεί με τα υπόλοιπα κυκλώματα, όπως μνήμη δεδομένων, RF και IORF. Λόγω της σχετικά απλής pipeline αρχιτεκτονικής, δε χρειάζεται να σχεδιαστούν πολύπλοκοι μηχανισμοί για κάποια καινούρια εντολή, όπως θα συνέβαινε αν ο αριθμός των pipeline επιπέδων ήταν μεγαλύτερος. Έτσι, υπάρχει η δυνατότητα να προστεθούν αρκετά πολύπλοκες εντολές που υποστηρίζονται από μεγαλύτερες (και ακριβότερες) AVR οικογένειες, όπως για παράδειγμα η εντολή του πολλαπλασιασμού, παρέχοντας μεγάλες ευκολίες στο χρήστη κατά τη διάρκεια της σχεδίασης του κώδικα. Στο σχήμα 7.1 φαίνεται ο τρόπος που μπορεί να μπει μια καινούρια εντολή στο ήδη υπάρχον σύνολο. Το σχήμα είναι παρόμοιο με το 4.10, μόνο που στην προκειμένη περίπτωση, το παχύ βέλος δηλώνει την εισαγωγή της FSM κάποιας νέας εντολής.



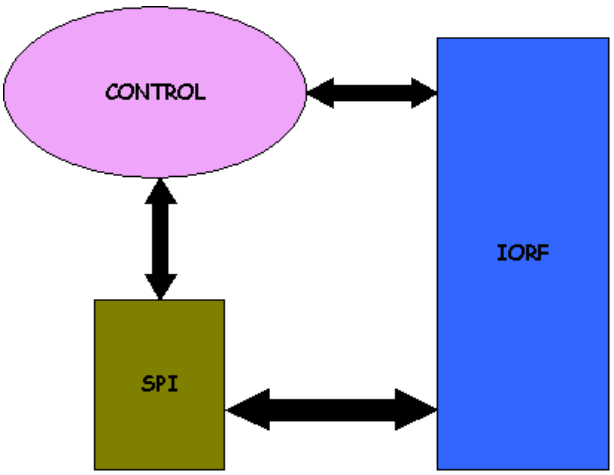

Σχήμα 7.1 – Προσθήκη νέας εντολής στο δέντρο αποκωδικοποίησης.

Στην IORF, υπάρχει η δυνατότητα να προστεθούν επιπλέον IO καταχωρητές, προκειμένου να υποστηρίζονται περισσότερες λειτουργίες. Μία από αυτές για παράδειγμα, μπορεί να είναι η σύνδεση εξωτερικής μνήμης με το μοντέλο. Όλα τα απαραίτητα πεδία για τη λειτουργία αυτή, υπάρχουν ήδη στον IO καταχωρητή SREG και το μόνο που μένει, είναι η προσθήκη της κατάλληλης λογικής στη μονάδα γενικού ελέγχου. Επίσης, πάλι στον τομέα των μνημών, μπορεί να σχεδιαστεί το κατάλληλο interface για την επικοινωνία του μοντέλου με κάποια EEPROM και να χρησιμοποιούνται οι αντίστοιχες εντολές όπως και στον AT90S8515. Στο σχήμα 7.2 φαίνεται ένα απλό διάγραμμα που δείχνει πώς θα μπορούσε να είναι η επικοινωνία μεταξύ της EEPROM και της εξωτερικής μνήμης με την FPGA.

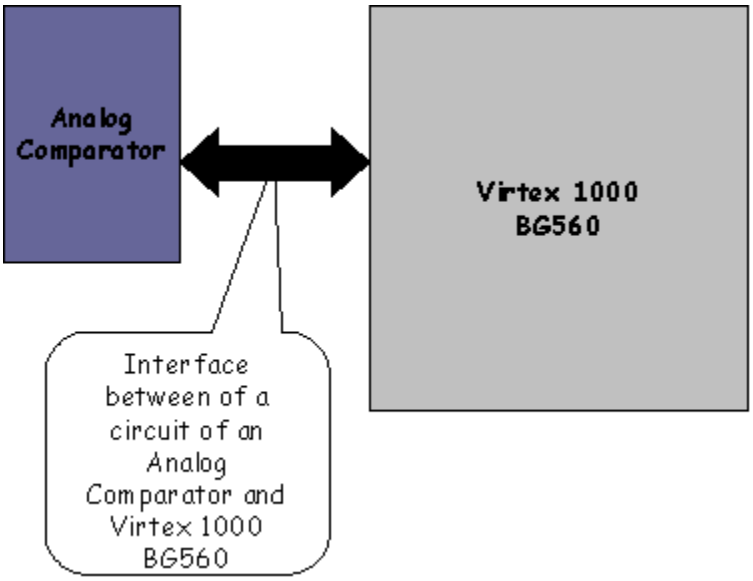



Σχήμα 7.2 – Απλό διάγραμμα επικοινωνίας EEPROM ή εξωτερικής μνήμης με FPGA.

Τέλος, μια ακόμα βασική επέκταση που θα μπορούσε να γίνει, είναι η προσθήκη στη σχεδίαση ορισμένων κυκλωμάτων όπως το SPI ή η επικοινωνία του μοντέλου με κάποιον εξωτερικό αναλογικό συγκριτή. Για το SPI, χρειάζεται η προσθήκη των αντίστοιχων καταχωρητών στην IORF, η οποία είναι έτσι σχεδιασμένη ώστε να είναι σχετικά εύκολο, ενώ θα μπορούσε να προστεθεί και το interrupt SPI Complete, που ενεργοποιείται όταν τελειώσει η μετάδοση δεδομένων. Ο αναλογικός συγκριτής, λόγω της φύσης του, δεν είναι δυνατόν να σχεδιαστεί με ψηφιακά κυκλώματα (άρα προφανώς ούτε σε VHDL), αλλά αυτό που μπορεί να γίνει είναι η σχεδίαση ενός κατάλληλου interface, ώστε να μπορεί να υπάρχει εξωτερική επικοινωνία του μοντέλου με αυτόν. Στο σχήμα 7.3a φαίνεται η σύνδεση του SPI με το υπάρχον κύκλωμα, ενώ στο 7.3b, φαίνεται ένα απλό διάγραμμα επικοινωνίας μεταξύ του αναλογικού συγκριτή και της FPGA.




Σχήμα 7.3a – Σύνδεση του SPI με το υπόλοιπο κύκλωμα.




Σχήμα 7.3b – Απλό διάγραμμα επικοινωνίας αναλογικού συγκριτή και FPGA.

## 7.3 Επίλογος.

Σταχυολογώντας, στα πλαίσια της διπλωματικής αυτής σχεδιάστηκε ένα μοντέλο IP core του μικροελεγκτή AT90S8515 της Atmel. Στόχος της σχεδίασης αυτής ήταν το τελευταίο να είναι σχετικά μικρό σε κατάληψη χώρου στις FPGAs, ενώ παράλληλα να μπορεί να επεκταθεί σε αρκετούς τομείς. Έτσι, δόθηκε ιδιαίτερο βάρος στον τρόπο σχεδίασης, ώστε αναβαθμίσεις όπως προσθήκη εντολών ή κάποιου interface ώστε να συμπεριληφθούν νέα κυκλώματα (SPI, Analog

Comparator), να μπορεί να γίνει αρκετά εύκολα, γεγονός που θα αυξήσει ακόμα περισσότερο τη λειτουργικότητα του μοντέλου.

,

# 8 Παράρτημα Α – Ακρώνυμα.

## A

**AES** – **A**dvanced **E**ncryption **S**tandard

**ALU** – **A**rithmetic **L**ogic **U**nit

## C

**CAN** – **C**ontroller **A**rea **N**etwork

## D

**DAC** – **D**igital to **A**nalog **C**onverter

**DDR** – **D**ouble **D**ata **R**ate

**DES** – **D**ata **E**ncryption **S**tandard

## E

**EEPROM** – **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory

## F

**FIFO** – **F**irst **I**n **F**irst **O**ut

**FIR** – **F**inite **I**mpulse **R**esponse

**FPGA** – **F**ield **P**rogrammable **G**ate **A**rray

**FSM** – **F**inite **S**tate **M**achine

## G

**GPR** – **G**eneral **P**urpose **R**egister

## I

**IMEM** – **I**nstruction **M**emory

**IORF** – **I**nput **O**utput **R**egister **F**ile

## J

**JTAG** - **J**oint **T**est **A**ction **G**roup

## M

**MAC** - **M**edia **A**ccess **C**ontrol

## P

**PCB** – **P**rinted **C**ircuit **B**oard

**PCI** – **P**eripheral **C**omponent **I**nterface

## R

**RF** – **R**egister **F**ile

## S

**SDRAM** – **S**ynchronous **D**ynamic **R**andom **A**ccess **M**emory

**SoC** – **S**ystem on a **C**hip

**SPI** – **S**erial **P**eripheral **I**nterface

**SRAM** – **S**tatic **R**andom **A**ccess **M**emory

## U

**UART** – **U**niversal **A**synchronous **R**eceiver **T**ransmitter

**USB** – **U**niversal **S**erial **B**us

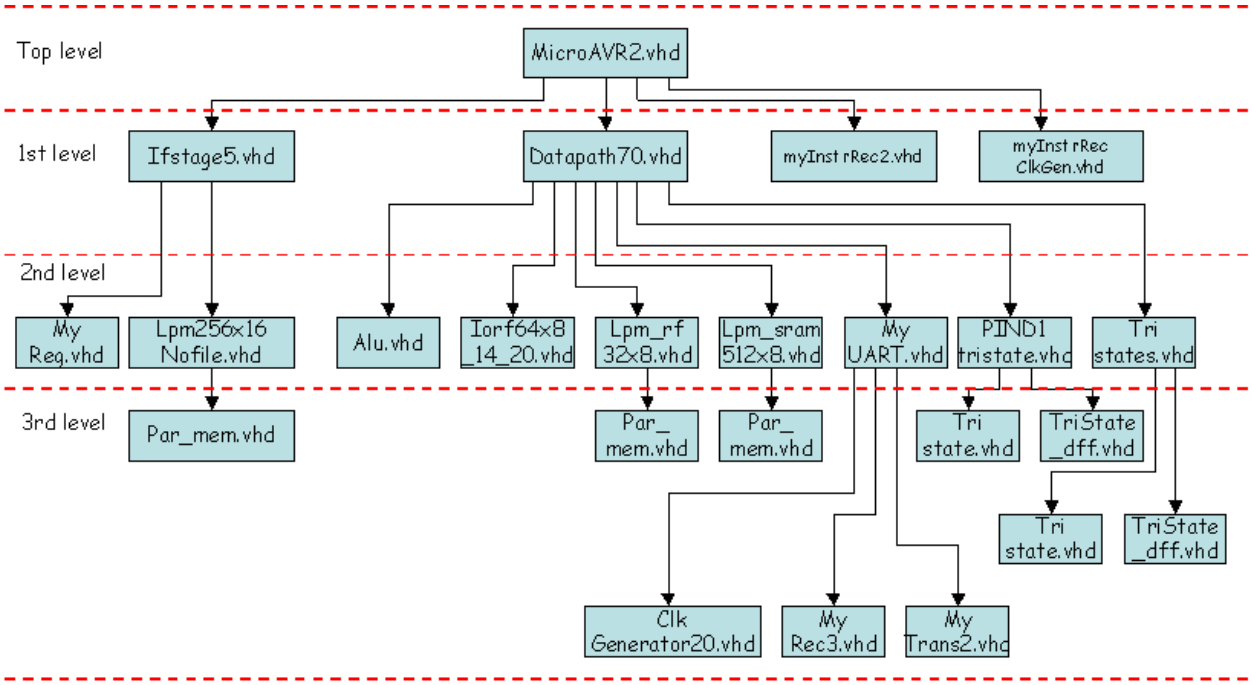
## V

**VHDL** – **V**ery high speed **H**ardware **D**escription **L**anguage

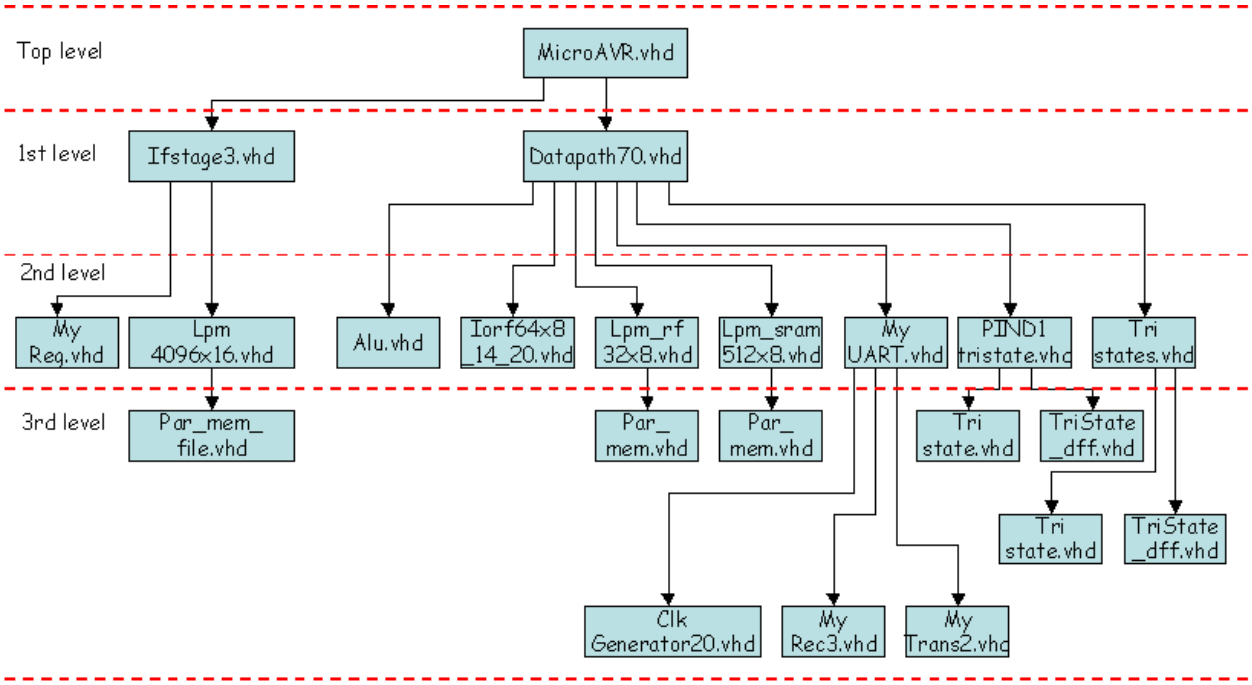
**VGA** – **V**ideo **G**raphic **A**dapter

# 9 Παράρτημα Β – Ιεραρχία του μοντέλου.

Στο συγκεκριμένο παράρτημα φαίνεται η ιεραρχία που υπάρχει στη σχεδίαση μεταξύ των κυκλωμάτων που υλοποιήθηκαν σε VHDL. Επειδή έγιναν δοκιμές με και χωρίς τον instruction receiver (παράγραφοι 6.1 και 6.2), θεωρήθηκε σκόπιμο να παρουσιαστούν 2 διαφορετικά σχήματα, τα οποία να δείχνουν τις μικρές διαφορές που υπάρχουν. Στα πλαίσια φαίνεται κάθε φορά το όνομα του συγκεκριμένου αρχείου VHDL που αντιπροσωπεύει το εκάστοτε κύκλωμα. Στο σχήμα Β1 φαίνεται η ιεραρχία παρουσία του instruction receiver και στο Β2 χωρίς αυτόν.




Σχήμα Β1 - Ιεραρχία του μοντέλου IP core με τον instruction receiver.




Σχήμα B2 - Ιεραρχία του μοντέλου IP Core χωρίς τον instruction receiver.

# 10 Παράρτημα Γ – Πίνακες σημάτων.

Στο παράρτημα Γ αναφέρονται τα σήματα καθενός από τα αρχεία *.vhd που σχεδιάστηκαν, καθώς και πού συνδέονται. Για ευκολία στην ανάγνωση, έχουν χωριστεί σε πίνακες, όπου καθένας από τους τελευταίους περιγράφει τα σήματα ενός *.vhd αρχείου. Για λόγους συντομίας ισχύουν οι παρακάτω αντιστοιχίες:

- IMEM ~ lpm_imem4096x16.vhd / lpm_imem256x16file.
- RF ~ rf32x8.vhd.
- Alu ~ alu.vhd.
- Clkgenerator ~ clkgenerator20.vhd
- UART ~ myUART.vhd.
- UARTReceiver ~ myRec3.vhd.
- UARTTransmitter ~ myTrans2.vhd.
- Datapath ~ datapath70.vhd.
- Ifstage ~ ifstage3.vhd / ifstage5.vhd.
- MicroAVR ~ microAVR.vhd / microAVR2.vhd.
- IORF ~ iorf64x8_14_20.vhd.
- SRAM ~ lpm_sram512x8.vhd.
- PIND1Tristate ~ PIND1Tristate.vhd.
- Tristates ~ tristates.vhd
- Tristate ~ tristates.vhd.
- Register ~ myReg.vhd.
- Memory ~ par_mem.vhd / par_mem_file.vhd.
- Tristates_dff ~ tristates_dff.vhd.
- InstrRec ~ myInstrRec2.vhd.
- InstrRecClkGen ~ myInstrRecClkGen.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Rfrd1_s | Από RF προς alu |
| Rfrd2_s | Από RF προς alu |
| Aluin3_s | Από control προς alu |

| | |
|---|---|
| Aluimm6_s | Από control προς alu |
| Aluimm8_s | Από control προς alu |
| Aluop_s | Από control προς alu |
| Aluout_s | Από alu προς control |
| Aluout16_s | Από alu προς control |

Πίνακας Γ1 – alu vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς clkgenerator |
| Clk | Από εξωτερική πηγή προς clkgenerator |
| Recfinished_s | Από UART προς clkgenerator |
| TransFinished_s | Από UART προς clkgenerator |
| UARTclk_select_s | Από UART προς clkgenerator |
| TransFinished | Από clkgenerator προς UART |
| RecFinished | Από clkgenerator προς UART |
| UARTclk | Από clkgenerator προς UARTReceiver και UARTTransmitter |

Πίνακας Γ2 – clkgenerator20.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς datapath |
| Clk | Από εξωτερική πηγή προς dtatpath |
| Ir_s | Από ifstage προς datapath |
| Pcplus1_s | Από ifstage προς datapath |
| Lds_sts_k_lpm_data_s | Από ifstage προς datapath |
| Instr32bit_s | Από ifstage προς datapath |
| TargetPC_s | Από datapath προς ifstage |
| EnableIR_s | Από datapath προς ifstage |
| EnablePCplus1_s | Από datapath προς ifstage |
| Pina | Από microAVR προς datapath |
| Pinb | Από microAVR προς datapath |
| Pinc | Από microAVR προς datapath |
| Pind | Από microAVR προς datapath |
| GIFR_s | Από datapath προς ifstage |
| GIMSK_s | Από datapath προς ifstage |
| TIMSK_s | Από datapath προς ifstage |
| SREG_s | Από datapath προς ifstage |
| USR_s | Από datapath προς ifstage |

| | |
|---|---|
| Ir_final_state_s | Από datapath προς ifstage |
| HoldpushPC | Από datapath προς ifstage |
| NPC_sel | Από datapath προς ifstage |
| TIFR_s | Από datapath προς ifstage |

Πίνακας Γ3 – datapath70.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς ifstage |
| Clk | Από εξωτερική πηγή προς ifstage |
| Ir_final_state_s | Από datapath προς ifstage |
| Hold_pushPC_s | Από datapath προς ifstage |
| GIFR_s | Από datapath προς ifstage |
| GIMKS_s | Από datapath προς ifstage |
| TIFR_s | Από datapath προς ifstage |
| TIMSK_s | Από datapath προς ifstage |
| SREG_s | Από datapath προς ifstage |
| USR_s | Από datapath προς ifstage |
| Imem_wraddr | Από microAVR προς ifstage |
| Imem_din | Από microAVR προς ifstage |
| Imem_wr_en | Από microAVR προς ifstage |
| Imem_rd_en | Από microAVR προς ifstage |
| EnablePCplus1_s | Από datapath προς ifstage |
| TargetPC_s | Από datapath προς ifstage |
| EnableIR_s | Από datapath προς ifstage |
| NPC_sel_s | Από datapath προς ifstage |
| Ir_s | Από ifstage προς datapath |
| Pcplus1_s | Από ifstage προς datapath |
| Lds_sts_k_lpm_data_s | Από ifstage προς datapath |
| Instr32bit_s | Από ifstage προς datapath |

Πίνακας Γ4 – ifstage3.vhd / ifstage5.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς IORF |
| Clk | Από εξωτερική πηγή προς IORF |
| Clear_int_s | Από control προς IORF |

| | |
|---|---|
| Pres0_clk8in_s | Από control προς IORF |
| Pres0_clk64in_s | Από control προς IORF |
| Pres0_clk256in_s | Από control προς IORF |
| Pres0_clk1024in_s | Από control προς IORF |
| Pres1_clk8in_s | Από control προς IORF |
| Pres1_clk64in_s | Από control προς IORF |
| Pres1_clk256in_s | Από control προς IORF |
| Pres1_clk1024in_s | Από control προς IORF |
| Iorfrdaddr_s | Από control προς IORF |
| Iorfwraddr_s | Από control προς IORF |
| Iorf_rd_en_s | Από control προς IORF |
| Iorf_wr_en_s | Από control προς IORF |
| Iorf_din_s | Από control προς IORF |
| GIRFin_s | Από control of GIFR προς IORF (εσωτερικά του datapath) |
| TCNT0in_s | Από control προς IORF |
| TCNT1in_s | Από control προς IORF |
| SREGout_s | Από IORF προς ifstage |
| GIFRout_s | Από IORF προς ifstage |
| TIFRout_s | Από IORF προς ifstage |
| TIMSKout_s | Από IORF προς ifstage |
| MCUCRout_s | Από IORF προς control of GIFR (εσωτερικά του datapath) |
| TCNT0out_s | Από IORF προς control |
| USRout_s | Από IORF προς ifstage |
| PORTAout_s | Form IORF προς tristates |
| DDRAout_s | Form IORF προς tristates |
| PORTBout_s | Form IORF προς tristates |
| DDRBout_s | Form IORF προς tristates |
| PORTCout_s | Form IORF προς tristates |
| DDRCout_s | Form IORF προς tristates |
| PORTDout_s | Form IORF προς tristates |
| DDRDout_s | Form IORF προς tristates |
| UARTsend_com_s | Από IORF προς UART |
| DataReceived_s | Από UART προς IORF |
| RecFinished_s | Από UART προς IORF |
| TransFinished_s | Από UART προς IORF |
| FrameError_s | Από UART προς IORF |
| UARTTransmitting_s | Από UART προς IORF |
| UARTClk_select_s | Από IORF προς UART |
| TransmitterEnable_s | Από IORF προς UART και PIND1Tristate |
| ReceiverEnable_s | Από IORF προς UART |
| DataTo Transmit_s | Από IORF προς UART |
| Pres0_clk8out_s | Από IORF προς control |
| Pres0_clk64out_s | Από IORF προς control |
| Pres0_clk256out_s | Από IORF προς control |
| Pres0_clk1024out_s | Από IORF προς control |

| | |
|---|---|
| Pres1_clk8out_s | Από IORF προς control |
| Pres1_clk64out_s | Από IORF προς control |
| Pres1_clk256out_s | Από IORF προς control |
| Pres1_clk1024out_s | Από IORF προς control |

Πίνακας Γ5 – iorf64x8_14_20.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς IMEM |
| PCout_s | Από PC προς IMEM (εσωτερικά του ifstage) |
| Imem_wraddr | Από microAVR προς IMEM |
| Imem_rdaddr | Από microAVR προς IMEM |
| Imem_rd_en | Από microAVR προς IMEM |
| Imem_wr_en | Από microAVR προς IMEM |
| Imem_din_s | Από microAVR προς IMEM |
| Imem_dout_s | Από IMEM προς multiplexor Imemout (εσωτερικά του ifstage) |

Πίνακας Γ6 – lpm_imem4096x16.vhd / lpm_imem256x16nofile.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Form εξωτερική πηγή προς SRAM |
| Sram_rdaddr_s | Από control προς SRAM |
| Sram_wraddr_s | Από control προς RF |
| Sram_rd_en_s | Από control προς RF |
| Sram_wr_en_s | Από control προς RF |
| Sram_din_s | Από control προς RF |
| Sram_dout_s | Από RF προς control |

Πίνακας Γ7 – lpm_sram512x8.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς microAVR |
| Clk | Από εξωτερική πηγή προς microAVR |
| Pina | Από εξωτερική πηγή προς microAVR |
| Pinb | Από εξωτερική πηγή προς microAVR |
| Pinc | Από εξωτερική πηγή προς microAVR |
| Pind | Από εξωτερική πηγή προς microAVR |

Πίνακας Γ8 – microAVR.vhd / microAVR2.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς UARTreceiver |
| ReceiverEnable_s | Από UART προς UARTReceiver |
| UARTclk | Από clkgenerator προς UARTreceiver |
| DataIn | Από UART προς UARTReceiver |
| RecFinished_s | Από UARTReceiver προς UART |
| FrameError | Από UARTReceiver προς UART |
| DataReceived | Από UARTReceiver προς UART |

Πίνακας Γ9 – myRec3.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς register |
| Din | Είσοδος του register |
| Clk | Από εξωτερική πηγή προς register |
| Enable | Ενεργοποίηση / απενεργοποίηση του register |
| Dout | Έξοδος του register |
| Width | Παράμετρος για το εύρος του καταχωρητή |

Πίνακας Γ10 – myReg.vhd.(*)

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς UARTTransmitter |
| Send_com | Από UART προς UARTTransmitter |
| TransmitterEnable | Από UART προς UARTTransmitter |
| UARTclk | Από clkgenerator προς UARTTransmitter |
| DataToTransmit | Από UART προς UARTTransmitter |
| TransFinished_s | Από UARTTransmitter προς UART |
| Transmitting | Από UARTTransmitter προς UART |
| DataTransmitted | Από UARTTransmitter προς UART |

Πίνακας Γ11 – myTrans2.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς UART |
| UARTsend_com_s | Από IORF προς UART |
| TransmitterEnable_s | Από IORF προς UART |
| ReceiverEnable_s | Από IORF προς UART |
| Clk | Από εξωτερική πηγή προς UART |
| DataToTransmit_s | Από IORF προς UART |
| UARTclk_select_s | Από IORF προς UART |
| PINDin_s(0) | Από PIND0 προς UART |
| DataReceived_s | Από UART προς IORF |
| RecFinished | Από UART προς IORF |
| TransFinished_s | Από UART προς IORF |
| UARTTransmitting_s | Από UART προς IORF |
| FrameError_s | Από UART προς IORF |
| UARTDataTransmitted_s | Από UART προς PIND1Tristate |

Πίνακας Γ12 – myUART.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς memory |
| Rdaddress | Address προς read Από memory |
| Rden | Διάβασμα από τη memory |
| Data | Data προς αποθήκευση στη memory |
| Wraddress | διεύθυνση εγγραφής στη memory |
| Wren | Γράψιμο στη memory |
| Q | Έξοδος της memory |
| Lpm_width | Παράμετρος για το εύρος της memory |
| Lpm_widthad | Παράμετρος για το μέγεθος της memory |

Πίνακας Γ13 – par_mem.vhd / par_mem_file.vhd. (*)

| Όνομα σήματος | Περιγραφή |
|---|---|
| Clk | Από εξωτερική πηγή προς PIND1tristate |
| DDRDout_s(1) | Από IORF προς PIND1tristate |
| TransmitterEnable_s | Από UART προς PIND1tristate |
| Pind(1) | Από PIND1 of microAVR προς PIND1tristate |
| Pind1in_s(1) | Από PIND1tristate προς control και RF |
| PORTDout_s(1) | Από IORF προς PIND1 of microAVR |

| | |
|---|---|
| UARTDataTransmitted_s | Από UART προς PIND1 of microAVR |

Πίνακας Γ14 – PIND1tristate.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Clk | Από εξωτερική πηγή προς tristates_dff |
| Direction_s | Από tristates προς tristates_dff |
| N_dir_s | Από tristates προς trastate |
| B1 | Από tristates προς tristates |
| A | Από tristates προς tristate |

Πίνακας Γ15 – tristate.vhd / tristate_dff.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Clk | Από εξωτερική πηγή προς tristates |
| DDRAout_s | Από IORF προς tristates |
| DDRBout_s | Από IORF προς tristates |
| DDRCout_s | Από IORF προς tristates |
| DDRDout_s | Από IORF προς tristates |
| PINA | Από PINA of microAVR προς tristates |
| PINB | Από PINA of microAVR προς tristates |
| PINC | Από PINA of microAVR προς tristates |
| PIND | Από PINA of microAVR προς tristates |
| PINAin_s | Από tristates προς control και RF |
| PINBin_s | Από tristates προς control και RF |
| PINCin_s | Από tristates προς control και RF |
| PINDin_s | Από tristates προς control και RF |
| PORTAout_s | Από IORF προς PINA of microAVR |
| PORTBout_s | Από IORF προς PINB of microAVR |
| PORTCout_s | Από IORF προς PINC of microAVR |
| PORTDout_s | Από IORF προς PIND of microAVR |

Πίνακας Γ16 – tristates.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς RF |
| Rfrd1_en_s | Από control προς RF |

| | |
|---|---|
| Rfrd2_en_s | Από control προς RF |
| Rfwr_en_s | Από control προς RF |
| Rfrdaddr1_s | Από control προς RF |
| Rfrdaddr2_s | Από control προς RF |
| Rfwraddr_s | Από control προς RF |
| Rf_din_s | Από control προς RF |
| Rfrd1_s | Από RF προς control |
| Rfrd2_s | Από FR προς control |

Πίνακας Γ17 – lpm_rf32x8.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς InstrRec |
| UARTclk_s | Από InstrRecClkGen προς InstrRec |
| Pind(0) | Από microAVR προς InstrRec |
| MyInstrRecEnable_s | Από microAVR προς InstrRec |
| Byte_counter_s | Από InstrRec προς microAVR |
| MyInstrRecFinished_s | Από InstrRec προς InstrRecClkGen |
| Imem_din_s | Από InstrRec προς ifstage |

Πίνακας Γ18 – myInstrRec2.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|
| Reset | Από εξωτερική πηγή προς InstrRecClkGen |
| MyInstrRecFinished_s | Από InstrRec προς InstrRecClkGen |
| Clk | Από εξωτερική πηγή προς InstrRecClkGen |
| InstrRecFinished_s | Από InstrRecClkGen προς microAVR |
| UARTclk_s | Από InstrRecClkGen προς InstrRec |

Πίνακας Γ19 – myInstrRecClkGen.vhd.

Ο τελευταίος πίνακας δεν αντιπροσωπεύει κάποιο *.vhd αρχείο, αλλά συγκεντρώνει όλα τα σήματα ελέγχου που υπάρχουν εσωτερικά του αρχείου datapath70.vhd.

| Όνομα σήματος | Περιγραφή |
|---|---|

| | |
|---|---|
| Target_PC_s | Από control προς ifstage |
| Enable_IR_s | Από control προς ifstage |
| NPC_sel | Από control προς ifstage |
| EnablePCplus1_s | Από control προς ifstage |
| Release_PushPC_s | Από control προς hold_pushPC signal control (εσωτερικά του datapath) |
| Clear_int_s | Από control προς IORF and external interrupt control (εσωτερικά του datapath) |
| Rfrd1_en_s | Από control προς RF |
| Rfrd2_en_s | Από control προς RF |
| Rfrdwr_en_s | Από control προς RF |
| Rfrdaddr1_s | Από control προς RF |
| Rfrdaddr2_s | Από control προς RF |
| Rfwraddr_s | Από control προς RF |
| Aluop_s | Από control προς alu |
| Aluin3_s | Από control προς alu |
| Aluimm6 | Από control προς alu |
| Aluimm8 | Από control προς alu |
| Rf_din_s | Από control προς RF |
| Iorfrdaddr_s | Από control προς IORF |
| Iorfwraddr_s | Από control προς IORF |
| Iorf_rd_en_s | Από control προς IORF |
| Iorf_wr_en_s | Από control προς IORF |
| Iorf_din_s | Από control προς IORF |
| Sram_rdaddr_s | Από control προς SRAM |
| Sram_wraddr_s | Από control προς SRAM |
| Sram_rd_en_s | Από control προς SRAM |
| Sram_wr_en_s | Από control προς SRAM |
| Sram_din_s | Από control προς SRAM |
| Pindin_s(2) | Από microAVR προς control |
| Pindin_s(3) | Από microAVR προς control |
| Add_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Ldi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Adc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Adiw_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| And_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Andi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Asr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Bclr_cur_state | Από control προς instruction final state control (εσωτερικά του |

| | |
|---|---|
| | datapath) |
| Bld_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Brbc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Brbs_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Bset_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Bst_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Out_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| In_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Cbi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Com_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Cp_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Cpc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Cpi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Subi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Cpse_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sub_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Eor_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Mov_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbic_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Ori_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddZ0q31_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddY0q31_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StdZ0q31_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StdY0q31_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Lds_cur_state | Από control προς instruction final state control (εσωτερικά του |

| | |
|---|---|
| | datapath) |
| LddZplus_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddminusZ_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddYplus_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddminusY_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LdXplusRd_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LdminusXRd_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LdXRd_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Pop_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sts_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| stdZplus_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| stdminusZ_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| stdYplus_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| stdminusY_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StXplusRr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StminusXRr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| stXRr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Push_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Neg_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Inc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Ror_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Ret_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| PushPC_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Icall_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Ijmp_cur_state | Από control προς instruction final state control (εσωτερικά του |

| | |
|---|---|
| | datapath) |
| Dec_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbiw_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbi_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddZ32q63_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| LddY32q63_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StdZ32q63_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| StdY32q63_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Rjmp_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Rcall_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbrs_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbci_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbis_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Sbrc_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Lsr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Nop_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Or_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Swap_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Lpm_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| wdr_cur_state | Από control προς instruction final state control (εσωτερικά του datapath) |
| Aluout_s | Από alu προς control |
| Aluout16_s | Από alu προς control |
| Iorf_dout_s | Από IORF προς control |
| Sram_dout_s | Από SRAM προς control |

Πίνακας Γ20 – control unit.

(*) Το αρχείο αυτό έχει γενική χρήση και γι' αυτό περιγράφεται η λειτουργία των εισόδων και εξόδων τους.

# 11 References.

[1] David A. Patterson & John L. Hennessy, "*Computer Organization & Design: The Hardware/Software Interface"*, Morgan Kaufmann, 2nd edition (August 1997).

[2] David A. Patterson & John L. Hennessy, "*Computer Architecture: A Quantitative Approach"*, Morgan Kaufmann, 1996.

[3] Mark Zwolinski, "*Digital System Design With VHDL"*, Southampton, Prentice Hall, August 1999.

[4] Stefan Sjoholm & Lennart Lindh, "*VHDL For Designers"*, Prentice Hall, 1997.

[5] M. Morris Mano, "*Ψηφιακή Σχεδίαση 2η Έκδοση"*, Αθήνα, Prentice Hall, 1992.

[6] Adel S. Sedra & Kenneth C. Smith, "*Μικροηλεκτρονικά Κυκλώματα Τόμος Β'"*, Αθήνα, Saunders College Publishing, 1994.

[7] N.H. Weste & K.Eshraghian, *"Σχεδίαση Ολοκληρωμένων Κυκλωμάτων CMOS VLSI"*, Παπασωτηρίου, Αθήνα 1996.

[8] *"FPGA Express VHDL Reference Manual"*, Synopsys, December 1997.

[9] "*8-bit Microcontroller with 8K In-System Programmable Flash: AT90S8515"*, Atmel Corporation.

[10] "*8-bit Microcontroller with 8K In-System Programmable Flash: AT90S8515 Instruction Set"*, Atmel Corporation.

[11] "*Single-Port Block Memory for Virtex-II, Virtex, and Spartan-II V3.1"*, Xilinx, March 2, 2001.

[12] "*Dual-Port Block Memory for Virtex-II, Virtex, and Spartan-II V3.1"*, Xilinx, March 2, 2001.

[13] "*Virtex™ 2.5V Field Programmable Gate Arrays"*, Xilinx, July 19, 2001.

[14] "*Modelsim 5.7 SE Performance Guidelines"*, Model technology, November 2002.

[15] Jeung Joon Lee, *"Synthesizable Universal Asynchronous Receiver Transmitter"*, 4/27/2001.

[16] "*Virtex™ 2.5V Field Programmable Gate Arrays: Pinout Περιγραφή"*, Xilinx, July 19, 2001.

[17] "*SN54HC04, SN74HC04 Hex Inverters"*, Texas Instruments, November 2002, http://www.ti.com .

[18] K. Papademetriou, A. Dollas, S. Sotiropoulos, *"A Second Generation Embedded Reconfigurable Input Device for Kinetically Challenged Persons",* 2003 IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM) Posted Paper, IEEE Computer Society Press, 2003.

[19] A. Dollas, K. Papademetriou, N. Aslanides, T. Kean, *"A Reconfigurable Embedded Input Device for Kinetically Challenged Persons"*, Presented at the European Conference on Field Programmable Logic (FPL), Proceedings, pp.326-335, Belfast, N.Ireland, August 27-29, 2001, Springer-Verlag.

[20] Thyssen de Reus BV, *"Wheelchair Lift for the Disabled"*, http://www.fuse-7network.com/fuse/demonstration/29/29493.

[21] Ζωγόπουλος Σωτήρης, "*Σχεδίαση και Υλοποίηση Πειραματικής Πλατφόρμας Για Ενεργά Δίκτυα, Βασισμένη Σε Αναδιατασσόμενη Λογική"*, Εργαστήριο Μικροεπεξεργαστών και Υλικού, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Η/Υ, Πολυτεχνείο Κρήτης, Χανιά 2001.

[22] Κάχρης Χριστόφορος, *"Design and FPGA Implementation of the SCAN Encryption Algorithm*, Εργαστήριο Μικροεπεξεργαστών και Υλικού, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Η/Υ, Πολυτεχνείο Κρήτης, Χανιά, Οκτώβριος 2002.

[24] http://www.cast-inc.com

[25] http://www.freeip.com

[26] http://www.opencores.org

[27] http://www.atmel.com/products/ipcores/

[28] http://www.us.design-reuse.com/sip

[29] http://www.atmel.com

[30] http://www.xilinx.com

[31] http://www.altera.com

[32] http://www.opencores.org/projects/pavr/

[33] http://www.xilinx.com/xlnx/xebiz/onlinestore.jsp?sGlobalNavPick=PURCHASE

[34] http://www.model.com

[35] Analog Devices, *"High Speed, +5V, 0.1μF CMOS Rs-232 Drivers/Receivers: ADM222 / ADM232A / ADM 242"*.

[36] http://www.whatis.com

[37] http://www.interfacebus.com/Controllers.html

[38] *"Coprocessor Filed Programmable Gate Arrays: AT6000 (SV) Series"*, Atmel Corporation.

[39] "*Controlling FPGA Configuration With A Flash Base Microcontroller"*, Atmel Corporation.

[40] Steven Bailey, *"Comparison of VHDL, Verilog and SystemVerilog"*, Model Technology

[41] *"AVR220 : Bubble Sort, Application Note"*, Atmel Corporation.

[42] *"AVR220 : Multiply and Divide Routines, Application Note"*, Atmel Corporation.

[43] http://www.triscend.com