ΒΙΒΛΙΟΘΗΚΗ
ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ
026000265348

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Αρ. εισ.: 264 2005

Σχεδιασμός και ανάπτυξη διεπαφών ATM/PDH και CPU/αισθητήρα

183
ΜΠΛΕ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παγώνης Αθανάσιος
Φυσικός

Επίβλεψη : Επίκουρος Καθηγητής Ν. Μάνθος
Εργαστήριο Φυσικής Υψηλών Ενεργειών

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΙΣ ΣΥΓΧΡΟΝΕΣ ΗΛΕΚΤΡΟΝΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

Ιωάννινα, Οκτώβριος 1999

Π.Α. ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΕΝΤΡΙΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Η έγκριση της μεταπτυχιακής διπλωματικής εργασίας από το τμήμα Φυσικής του Πανεπιστημίου Ιωαννίνων δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα (Ν.5343/1932, αρθρ. 202)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Πίνακας Περιεχομένων

Περίληψη III
Abstract V

Εισαγωγή 1
1. Γενική περιγραφή του ATM 5
1.1 Εισαγωγή στο ATM 5
1.2 Λειτουργία του ATM 6
1.3 Το πρότυπο ATM 7
2. Στρώμα ATM 11
2.1 Δομή πακέτου ATM 11
2.2 Επικεφαλίδα του πακέτου 11
2.2.1 Γενικός έλεγχος ροής (Generic Flow Control, GFC) 12
2.2.2 Αναγνωριστής εικονικού μονοπατιού/καναλιού (Virtual Path/Channel Identifier, VPI/VCI) 12
2.2.3 Τύπος ωφέλιμου φορτίου (Palyolad Type, PT) 13
2.2.4 Πακέτο απώλειας προτεραιότητας (Cell Loss Priority, CLP) 13
2.2.5 Έλεγχος λάθους επικεφαλίδας (Header Error Control, HEC) 13
3. Σχετικά με τη VHDL 15
.4. Γενική περιγραφή του συστήματος PDH/ATM/SDH mapper 19
4.1 Ορισμοί 19
4.2 Γενική περιγραφή 21
5. Περιγραφή του δέκτη ATM/PDH 25
5.1 Στρώμα ATM του δέκτη (ATMRX) 25
5.1.1 Γενικά 25
5.1.2 TNETA (Texas Instruments) και η διεπαφή με την μονάδα ATMRX 26
5.1.3 Η μονάδα ATMRX 29
5.1.3.1 Υπομονάδα αυτόματης αναγνώρισης 30
5.1.3.2 Υπομονάδα αποθήκευσης και αναγώρισης των επικεφαλίδων 32
5.1.3.3 Υπομονάδα ακολουθίας καταστάσεων (state machine) 34
5.2 Στρώμα AAL του δέκτη (AALRX) 34
5.2.1 Γενικά 34
5.2.2 Η μονάδα AALRX 35
5.2.2.1 Αποσυναρμολογητής των SAR_Protocol Data Units (SAR_PDU DISSASEMBER) 36
5.2.2.2 Διορθωτής επικεφαλίδας τύπου AAL-1 (AAL-1 HEADER_COR) 37
5.2.2.3 Ελεγκτής ακολουθιακής αρίθμησης (SN_CHECK) 39
5.2.2.4 Μονάδα ανάκτησης ρολογιού PDHCLK (PDHCLK RECOVERY) 39
5.2.2.5 Μονάδα συγχρονισμού του πλαισίου PDH (FRAME_SYNC) 41
5.2.2.6 Δημιουργία πεδίου RTS (RTS CREATION) 43
5.2.2.7 Ταλαντωτής ελεγχόμενος από τάση (VOLTAGE CONTROLLED OSCILLATOR) 43

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.2.2.8 Εισαγωγή αδρανών bytes (IDLE BYTE INSERTION) 44
5.2.2.9 Πολυπλέκτης της FIFO (FIFO MULTIPLEXER) 44
5.2.2.10 Μονάδα εξόδου (OUTPUT UNIT) 45
5.2.2.11 Διεπαφές PDH (PDH INTERFACES) 45
5.2.2.12 Διεπαφή μικροεπεξεργαστή (MICROPROCESSOR INTERFACE) 46
6. Προσομοίωση λειτουργίας του δέκτη ATM/PDH 47
6.1 Βήματα κατά το σχεδιασμό με VHDL 47
6.2 Το πρόγραμμα υλοποίησης του δέκτη ATM/PDH σε VHDL 47
6.3 Αποτελέσματα του προγράμματος VHDL 51
7. Εισαγωγή στο σύστημα ΤΗΛ2 75
7.1 Γενική περιγραφή του συτήματος ΤΗΛ2 75
7.1.1 Πλαίσιο με το δίαυλο επικοινωνίας (back plane) 77
7.1.2 Τροφοδοτικό 77
7.1.3 Κάρτα κεντρικής μονάδας επεξεργασίας (CPU) 78
7.1.4 Κάρτα αναλογικών εισόδων (A/I) 78
7.1.5 Κάρτα ψηφιακών εξόδων (D/O) 78
7.1.6 Κάρτα interface RS-232 79
7.1.7 Κάρτα interface RS-485 79
8. Η κάρτα αναλογικών εισόδων 81
8.1 Γενική περιγραφή 81
8.2 Λογικός σχεδιασμός 81
8.2.1 Υπομονάδα προσαρμογής σήματος 4-20mA 83
8.2.2 Υπομονάδα απομόνωσης και μετατροπής 85
8.2.3 Υπομονάδα κωδικού κάρτας 86
8.3 Φυσικός σχεδιασμός (Τυπωμένο κύκλωμα) 87
8.4 Έλεγχος καλής λειτουργίας 89
9. Η κάρτα ψηφιακών εξόδων 91
9.1 Γενική περιγραφή 91
9.2 Λογικός σχεδιασμός 91
9.2.1 Υπομονάδα εξόδου 91
9.2.2 Υπομονάδα οδήγησης 93
9.2.3 Υπομονάδα κωδικού κάρτας 93
9.3 Φυσικός σχεδιασμός (Τυπωμένο κύκλωμα) 94
9.4 Έλεγχος καλής λειτουργίας 96
Αναφορές 97
Παράρτημα Α 99
Παράρτημα Β 161
Παράρτημα Γ 165

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Περίληψη

Παρουσιάζεται η ανάπτυξη διεπαφής από σύγχρονη ψηφιακή ιεραρχία (Synchronous Digital Hierarchy, SDH) σε πολυπλεγμένες γραμμές σχεδόν σύγχρονης ψηφιακής ιεραρχίας (Plesiochronous Digital Hierarchy, PDH), χρησιμοποιώντας τα στρώματα μεθόδου ασύγχρονης μεταφοράς (Asychronous Transfer Mode, ATM) και προσαρμογής ATM (ATM Adaptation Layer, ATM) του πρωτοκόλλου του ψηφιακού δικτύου ολοκληρωμένων υπηρεσιών ευρείας ζώνης (Broadband -Integrated Service Digital Network, B-ISDN). Παρουσιάζεται επίσης η ανάπτυξη διεπαφής CPU/αισθητήρα. Η αρχιτεκτονική που χρησιμοποιήθηκε στον σχεδιασμό της πρώτης διεπαφής εκμεταλεύεται πλήρως τους πόρους του πρωτοκόλλου του δικτύου B-ISDN, το οποίο χρησιμοποιείται για την διακίνηση πακέτων ATM. Ο αντικειμενικός σκοπός της υλοποίησης της διεπαφής είναι η αποπολυπλεξία ενός καναλιού SDH 155 Mb/s σε 64 κανάλια PDH 2 Mb/s. Ο σχεδιασμός της διεπαφής έγινε με γλώσσα περιγραφής υλικού VHDL (Very high speed Hardware Description Language) και η υλοποίησή της έγινε σε FPGA (Field Programmable Gate Array). Η διεπαφή CPU/αισθητήρα αποτελείται από μια κάρτα αναλογικών εισόδων για ψηφιοποίηση αναλογικών σημάτων αισθητήρα 4-20 mA και μια κάρτα ψηφιακών εξόδων 8 καναλιών που αναπτύχθηκαν για χρήση σε σύστημα τηλεμετρίας-τηλεδιαχείρησης.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Abstract

The development of a SDH (Synchronous Digital Hierarchy) to multiplexed PDH (Plesiochronous Digital Hierarchy) lines interface is presented. This interface makes use of the ATM (Asynchronous Transfer Mode) and the AAL (ATM Adaptation Layer) layers of the B-ISDN (Broandband Integrated Service Digital Network) protocol. A CPU/sensor interface is also presented. The architecture used in the first interface fully utilizes the B-ISDN protocol resources, used by the ATM cells. This interface demultiplexes a SDH 155 Mbps line to 64 PDH 2Mbps lines and it was designed using VHDL (Very high speed Hardware Description Language) and implemented in a FPGA (Field Programmable Gate Array). The CPU/sensor interface includes an analog input card for digitization of 4-20 mA analog sensor signals and an 8 channels digital output card. The latter interface was developed for use in remote control and measurement systems.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Εισαγωγή

Στον τηλεπικοινωνιακό κόσμο έχουν προστεθεί νέες υπηρεσίες στις ήδη υπάρχουσες όπως μεταφορά δεδομένων υψηλής ταχύτητας (Real-Time Control, MAN interconnection), κινούμενη εικόνα και ήχο (Tele-education, Medical images, Videoconference), τηλεδιαχείρηση κ.λ.π. Η έλευση των νέων αυτών τηλεπικοινωνιακών υπηρεσιών δημιουργεί νέες απαιτήσεις για το τηλεπικοινωνιακό δίκτυο. Επίσης, η σημερινή κατάσταση στο τομέα των συστημάτων τηλεμετρίας και τηλεδιαχείρησης μπορεί να χαρακτηριστεί από μια προσπάθεια ανάπτυξης νέων συστημάτων με τα οποία θα προσφέρονται πολλές υπηρεσίες και θα στηρίζονται στο μεγαλύτερο μέρος στην ψηφιακή τεχνολογία.

Οι παλιές υπηρεσίες, όπως για παράδειγμα, το απαρχαιωμένο πλέον αναλογικό τηλεφωνικό δίκτυο, τα απαρχαιωμένα αναλογικά συστήματα διαχείρησης είναι απαραίτητο να υποστηρίζονται από τα νέα συστήματα. Επομένως, είναι επιβεβλημένη η μετατροπή όλης αυτής της αναλογικής πληροφορίας σε ψηφιακή πληροφορία γιατί η δεύτερη μπορεί ναμεταδοθεί αξιόπιστα και με ασφάλεια. Η αντικατάσταση των υπαρχόντων αναλογικών συστημάτων με ψηφιακά είναι δαπανηρή και επομένως πρέπει να βρεθεί μια οικονομική λύση που θα λύνει το παραπάνω πρόβλημα. Η προσαρμογή των δύο παραπάνω τεχνολογιών γίνεται με την χρήση διεπαφών (interfaces), οι οποίες υποστηρίζουν τις παλιές υπηρεσίες και τις προσαρμόζουν στα νέα συστήματα.

Τα κεφάλαια 1 έως 6 αφορούν την ανάπτυξη διεπαφής ATM/PDH και συγκεκριμένα τον λογικό σχεδιασμό δέκτη ATM/PDH με την χρήση γλώσσας περιγραφής υλικού VHDL. Τα κεφάλαια 7 έως 9 αφορούν την υλοποίηση διεπαφής CPU/αισθητήρα και συγκεκριμένα την υλοποίηση δύο καρτών εισόδων/εξόδων που αποτελούν τμήμα μιας πρότυπης μονάδας τηλεμετρίας και τηλεχειρισμού.

Συγκεκριμένα:

Στο κεφάλαιο 1 δίνεται μια γενική περιγραφή του ATM (Ansychronous Transfer Mode) και τονίζονται οι επιπλέον δυνατότητες που προσφέρει σε σχέση με τις προϋπάρχουσες μεθόδους μεταφοράς δεδομένων.

Στο κεφάλαιο 2 δίνεται λεπτομερής περιγραφή του επιπέδου ATM.

Στο κεφάλαιο 3 δίνεται μια μικρή εισαγωγή στην γλώσσα VHDL, που είναι πλέον το καθιερωμένο πρότυπο περιγραφής υλικού. Δίνεται μια περιγραφή για τις δυνατότητες που προσφέρει αυτή σε αντιστοίχηση με με τον κλασσικό τρόπο σχεδίασης υλικού.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Στο κεφάλαιο 4 γίνεται μια γενική περιγραφή του συστήματος PDH/ATM/SDH mapper με μια μικρή εισαγωγή στους όρους PDH και SDH.

Στο κεφάλαιο 5 δίνεται λεπτομερής περιγραφή του δέκτη ATM/PDH, παρουσιάζονται τα επίπεδα από τα οποία αποτελείται και σχολιάζονται οι λειτουργίες των επιπέδων αυτών.

Στο κεφάλαιο 6 περιγράφεται η διαδικασία με την οποία έγινε η υλοποίηση του λογικού σχεδιασμού με χρήση της VHDL.

Στο κεφάλαιο 7 περιγράφεται το σύστημα ΤΗΛ2.

Στο κεφάλαιο 8 περιγράφεται λεπτομερώς η κάρτα αναλογικών εισόδων.

Στο κεφάλαιο 9 περιγράφεται λεπτομερώς η κάρτα ψηφιακών εξόδων.

Στα παραρτήματα παρατίθονται ο κώδικας VHDL για το δέκτη ATM/PDH, η εκχώρηση των ακίδων του κοινού διαύλου (back plane) μέσω του οποίου επικοινωνούν οι κάρτες εισόδων/εξόδων με την κεντρική μονάδα επεξεργασίας και το λογισμικό που χρησιμοποιήθηκε για την εξομοίωση του κοινού διαύλου. Τέλος, θέλω να αναφέρω ότι το κύριο μέρος της εργασίας αποτελείται από τα κεφάλαια 6, 8 και 9.

Η μεταπτυχιακή αυτή εργασία υλοποιήθηκε σε δύο εργαστήρια και αποτελείται από δύο τμήματα. Με κριτήριο την χρονική περίοδο, το πρώτο τμήμα της έγινε με την συνεργασία του Εργαστηρίου Τηλεπικοινωνιακών Συστημάτων του Εθνικού Μετσόβιου Πολυτεχνείου στα πλαίσια του ερευνητικού προγράμματος « Βιομηχανικά Προιόντα Μικροηλεκτρονικής » σε συνεργασία με τον Αντώνη Κατσούλη, ενώ το δεύτερό της έγινε στο Εργαστήριο Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων στα πλαίσια του ερευνητικού προγράμματος « ΕΚΒΑΝ-Π, Ανάπτυξη Συστήματος Τηλεδιαχείρησης Υδάτινων Πόρων» σε συνεργασία με τον Παναγιώτη Κολοβό. Στο σημείο αυτό θα ήθελα να ευχαριστήσω πολλούς ανθρώπους που με βοήθησαν κατά την διάρκεια της ερευνητικής μου εργασίας. Ιδιαίτερα :

Τον Επίκουρο Καθηγητή Ν. Μάνθο του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων και επιβλέποντα στην παρούσα μεταπτυχιακή διπλωματική εργασία, ευχαριστώ θερμά για την επιστημονική και ηθική συμπαράσταση στην διεκπεραίωση της εργασίας.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Τον Καθηγητή Φ. Τριάντη, διευθυντή του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων, ευχαριστώ θερμότατα για τον συντονισμό, την επιστημονική και ηθική συμπαράσταση στην διεκπεραίωση της εργασίας.

Τον Αναπληρωτή Καθηγητή Ν. Μήτρου του Εθνικού Μετσόβιου Πολυτεχνείου, ευχαριστώ θερμά για την επιστημονική και ηθική συμπάσταση, κατά την διάρκεια της συνεργασία μου με την ομάδα του Εθνικού Μετσόβιου Πολυτεχνείου στα πλαίσια του αντίστοιχου ερευνητικού προγράμματος.

Τον Επίκουρο Καθηγητή του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων, Ι. Ευαγγέλου ευχαριστώ για την μακρόχρονη επιστημονική συμπαράσταση καθ' όλη την διάρκεια της εκπόνησης της μεταπτυχιακής διπλωματικής εργασίας.

Ευχαριστώ θερμότατα τον Π. Τριανταφύλλου, ηλεκτρονικό του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων για την επιστημονική και τεχνική γνώση που αφειδώς μου διέθεσε, την ηθική συμπαράσταση κατά την διάρκεια της συνεργασίας μας.

Ευχαριστώ τους Α. Κατσούλη, Α. Ευαγγελάτο, Π. Κολοβό για το ευχάριστο κλίμα που δημιούργησαν κατά την διάρκεια της επιστημονικής συνεργασίας μας.

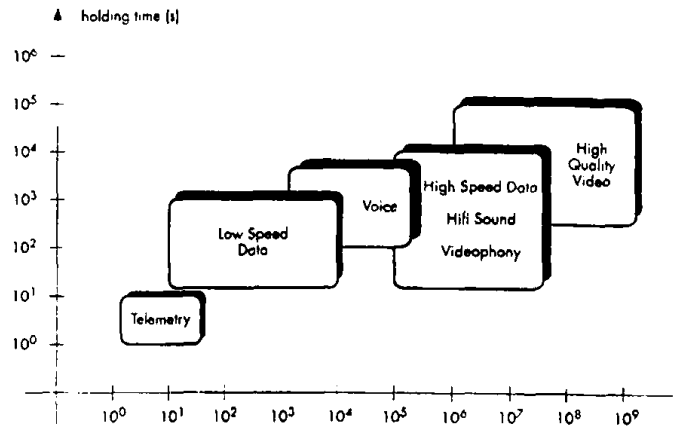
Τέλος, θέλω να ευχαριστήσω θερμά τους γονείς μου και τον αδερφό μου για την ηθική υποστήριξη καθ' όλη την διάρκεια της μεταπτυχιακής εργασίας μου, στους οποίους και την αφιερώνω.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

1. Γενική περιγραφή του ATM

1.1 *Εισαγωγή στο ATM*

Όπως αναφέρθηκε στον πρόλογο, η έλευση των νέων τηλεπικοινωνιακών υπηρεσιών δημιουργεί νέες απαιτήσεις για τα δίκτυα τηλεπικοινωνιών, σχήμα 1.1. Νέες τεχνικές τηλεπικοινωνιών, ή μέθοδοι μεταφοράς (transfer mode), απαιτούνται και θα προσφέρουν πλεονεκτήματα σε σχέση με τις υπάρχουσες τεχνικές. Η χρήση μιας συγκεκριμένης μεθόδου μεταφοράς καθορίζεται από την διαθέσιμη τεχνολογία της εκάστοτε εποχής.



Σχήμα 1.1 : Απαιτήσεις ταχύτητας

Εξαιτίας των απαιτήσεων νέων υπηρεσιών, όπως κινούμενη εικόνα και ήχος, και της υπάρχουσας τεχνολογίας, μία νέα μέθοδος μεταφοράς έχει οριστεί: το ψηφιακό δίκτυο ολοκληρωμένων υπηρεσιών (Integrated Service Digital Network, ISDN). Το ISDN είναι ένα τηλεφωνικό δίκτυο. Παρόλα αυτά, υπολογιστές μπορούν να το χρησιμοποιήσουν για να επικοινωνούν μεταξύ τους, να ανταλλάσουν δεδομένα, κλπ. Το ISDN σχεδιάστηκε για να εκμεταλεύεται τις υπάρχουσες χάλκινες καλωδιώσεις, οι οποίες έρχονται από το τηλεφωνικό κέντρο στα δικά μας αναλογικά τηλέφωνα με σκοπό να φέρει το ψηφιακό τηλεφωνικό σύστημα στα σπίτια μας.

Σύντομα όμως υπερκεράσθηκε από το ψηφιακό δίκτυο ολοκληρωμένων υπηρεσιών ευρείας ζώνης (Broadband Integrated Service Digital Network, B-ISDN) [2], το οποίο χρησιμοποιεί σαν μέσο διάδοσης την οπτική ίνα. Μια από τις κύριες διαφορές είναι η ταχύτητα. Το ISDN προσφέρει ρυθμό μεταφοράς δεδομένων της τάξης των 100Kbits/s, ενώ το B-ISDN προσφέρει από 100Mbits/s έως και 2Gbits/s.

Η ασύγχρονη μέθοδος μεταφοράς (Asynchronous Transfer Mode, ATM) [1] απλώς είναι ένα πρότυπο, το οποίο τρέχει πάνω στο B-ISDN. Κυριολεκτικά δεν είναι τίποτα παραπάνω από μια προδιαγραφή ενός πακέτου πληροφορίας 53-bytes εκ των οποίων τα 5-bytes είναι επικεφαλίδα. Η επικεφαλίδα περιέχει πληροφορίες δρομολόγησης. Το ATM μπορεί να τρέξει πάνω σ' ένα αριθμό από διαφορετικά φυσικά μέσα, όπως UTP (Unshielded Twisted Pair), οπτική ίνα (fiber optic). Το ATM προτάθηκε το 1987 από την CCITT (International Telegraph and Telephone Consultative Committee). Το 1990, η ITU (International Telecommunication Union, πρώην CCITT) εξέδοσε την πρώτη σειρά συστάσεων, προσδιορίζοντας τις λεπτομέρειες της ATM τεχνικής για χρήση στο B-ISDN.

Η ποικιλία B-ISDN υπηρεσιών και εφαρμογών, όπως φαίνονται στο πίνακα 1.1, απαιτούν ένα δίκτυο με δυνατότητες μεταφοράς οι οποίες:

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

- τροφοδοτούν υπηρεσίες που μπορεί να έχουν διαφορετικούς ρυθμούς μεταφοράς.
- υποστηρίζουν κυκλοφορίες με ρυθμό που μεταβάλλεται δραστικά (burst-type).
- λαμβάνουν υπ'όψιν καθυστερήσεις και εφαρμογές περιορισμένης εμβέλειας.

Service	Bit rate (Mbit/s)	Burstiness
Data transmission (connection-oriented)	1.5 to 130	1 - 50
Data transmission (connectionless)	1.5 to 130	1 - 50
Document transfer/retrieval	1.5 to 45	1 - 20
Videoconference/videotelephony	1.5 to 130	1 - 5
Broadband videotex/video retrieval	1.5 to 130	1 - 20
TV distribution	30 to 130	1
HDTV distribution	130	1

Burstiness = peak bit rate/average bit rate

Πίνακας 1.1 : Χαρακτηριστικά υπηρεσιών ευρείας ζώνης

1.2 *Λειτουργία του ATM*

Το πρότυπο ATM είναι μια τεχνολογία πακέτου, η οποία δρομολογεί την κίνηση μέσω μιας ετικέτας, η οποία περιέχεται στην επικεφαλίδα του πακέτου. Αντίθετα από άλλες τεχνολογίες πακέτου, όπως για παράδειγμα X.25 ή αναμετάδοση πλαισίου (frame relay), το ATM χρησιμοποιεί μικρά, σταθερού μήκους πακέτα (cells). Κάθε πακέτο αποτελείται από 53 bytes : 48 bytes για το πεδίο πληροφορίας και 5 bytes για επικεφαλίδα.

Η αρχή λειτουργίας του ATM βασίζεται στην αρχή της μεταγωγής πακέτου (cell switching) και στην αρχη εικονικής σύνδεσης (virtual connection). Η μεταγωγή πακέτου επιτρέπει στο σύστημα να χειρίζεται ποικίλους τύπους δεδομένων με ίση ευκολία. Η εικονική σύνδεση επιτρέπει να γίνεται χρήση του διαθέσιμου εύρους ζώνης αποδοτικά στην κίνηση δικτύου.

Η μεταγωγή πακέτου παρεκκλίνει από τις παραδοσιακές τηλεπικοινωνιακές μεθόδους. Παραδοσιακά δίκτυα χρησιμοποιούν πολύπλεξη διαίρεσης χρόνου (TDM) ορίζοντας ένα καθορισμένο εύρος χρονικού διαστήματος (time slot) μέσα σ' ένα επαναλαμβανόμενο πλαίσιο για κάθε κανάλι χρήστη. Τα δεδομένα περνάνε από κόμβους, οι οποίοι τα συγχρονίζουν μέσα στο πλαίσιο και τα εξάγουν στο επιθυμητό χρονικό διάστημα.

Κατά την μετάδοση, ενός πακέτου ATM για ένα δοσμένο χρήστη δεν υπάρχει καθορισμένο χρονικό διάστημα μέσα στο πλαίσιο (εξού και ο όρος ασύγχρονος). Το πακέτο μεταφέρει την επικεφαλίδα ταυτότητάς του σαν να ανήκει σ' ένα κανάλι για αυτόν τον συγκεκριμένο χρήστη. Τα δεδομένα του χρήστη μπορούν να καταλάβουν

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

από λίγα έως πολλά χρονικά διαστήματα δηλαδή όσα χρειάζονται, ενώ τα μη καταλαμβανόμενα χρονικά διαστήματα είναι διαθέσιμα για άλλους χρήστες.

Το ATM είναι σχεδιασμένο για να ταιριάζει καλύτερα με όλα τα σημερινά ψηφιακά δίκτυα, οπότε, σε χαμηλούς ρυθμούς μεταφοράς, εξαλείφεται η αναγκαιότητα χρήσης της ανίχνευσης/ανάκτησης λάθους των δεδομένων ανάμεσα σε κόμβους.

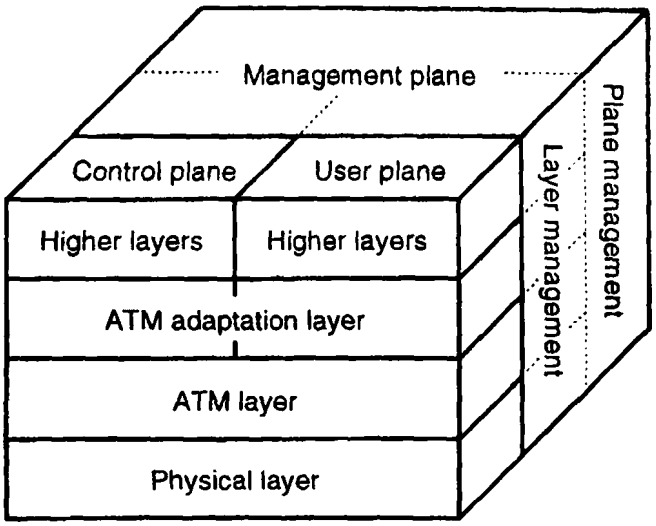
Η εισαγωγή δεδομένων μέσα στο πεδίο πληροφορίας των 48-bytes του πακέτου ATM πραγματοποιείται από το στρώμα προσαρμογής ATM (ATM adaptation Layer).

1.3 *Το πρότυπο ATM*

Το πρότυπο του ψηφιακού δικτύου ολοκληρωμένων υπηρεσιών ευρείας ζώνης (B-ISDN) όπως και το πρότυπο OSI, ορίζει μια ποικιλία από στρώματα, τα οποία περιέχουν ξεχωριστές λειτουργίες μοιράζοντας δεδομένα με καθορισμένη μορφή στα στρώματα πάνω και κάτω από αυτές, βλέπε σχήμα 1.2.

Το δίκτυο B-ISDN χρησιμοποιεί τρία στρώματα (σχήμα 1.2) για να περιγράψει εφαρμογές χρήστη υποστηριζόμενες από το δίκτυο: το φυσικό στρώμα (PHY), το στρώμα ATM και το στρώμα προσαρμογής ATM (AAL) [2]. Το PHY χειρίζεται λειτουργίες όπως: πακετάρισμα των δεδομένων για εκπομπή, μετακίνηση δεδομένων μέσω του ενδιάμεσου εκπομπής, ανάκτηση δεδομένων και προσδιορισμό των ορίων των πακέτων (πλαισίωση) μέσα σε μια ακολουθία από bits. Το στρώμα ATM κατευθύνει τα δεδομένα μέσα από κόμβους, χειρίζεται την μεταγωγή, και ορίζει τα περιεχόμενα της επικεφαλίδας. Το AAL προσαρμόζει δεδομένα χρήστη σε μορφή ATM και παρέχει ανίχνευση και ανάκτηση λάθους.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



ATM Asynchronous transfer mode

Σχήμα 1.2 : Πρότυπο αναφοράς B-ISDN

Η λειτουργία διαχείρησης εκτελείται διαμέσου όλων των στρωμάτων, όπως φαίνεται και στο σχήμα 1.2 σαν ένα επίπεδο πίσω από αυτά. Τα στρώματα περιέχουν τόσο λειτουργίες ελέγχου και σηματοδοσίας όσο και λειτουργίες μεταφοράς δεδομένων και ανταλλαγής με μια καθορισμένη μορφή.

Το AAL είναι υπεύθυνο για την χαρτογράφηση των δεδομένων του χρήστη μέσα στα πακέτα ATM. Αυτή η χαρτογράφηση ποικίλει με την ποιότητα και τον τύπο της υπηρεσίας μεταφοράς δεδομένων που χρειάζονται τα εκάστοτε δεδομένα.

Το AAL έχει δύο υποστρώματα: το υπόστρωμα σύγκλισης (Convergence Sublayer) και το υπόστρωμα κατάτμησης και επανασυναρμολόγησης (Segmentation And Reassembly) [10]. Η λειτουργία του AAL εξαρτάται από τη κατηγορία της υπηρεσίας. Σήμερα, οι προδιαγραφές ATM καθορίζουν 4 κατηγορίες υπηρεσιών:

- AAL1, υπηρεσίες σταθερού ρυθμού (CBR).

  Το πρότυπο AAL1 χειρίζεται την κίνηση στην οποία υπάρχει συσχέτιση χρονισμού ανάμεσα στην πηγή και στον προορισμό. Τέτοιες υπηρεσίες θυμίζουν την συμπεριφορά του υπάρχοντος τηλεφωνικού συστήματος και μεταφέρεται ένας καθορισμένος αριθμός από πακέτα ATM ανά μονάδα χρόνου. Τέτοιου είδους υπηρεσία είναι και η μεταφορά video σταθερού ρυθμού.

- AAL2, υπηρεσίες μεταβαλόμενου ρυθμού (VBR) ευαίσθητες στο χρονισμό.

  Το πρότυπο AAL2 χειρίζεται κίνηση στην οποία απαιτείται μια ισχυρή συσχέτιση χρονισμού ανάμεσα σε εκπομπό και δέκτη αλλά ο ρυθμός

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

μετάδοσης μπορεί να ποικίλει. Εφαρμογή αυτής της υπηρεσίας είναι η μεταφορά συμπιεσμένου video. Η πληροφορία video πρέπει να φτάνει στον προορισμό έγκαιρα αλλά δεν απαιτείται σταθερός ρυθμός μετάδοσης.

- AAL3/4, υπηρεσίες με ή χωρίς σύνδεση, μεταφορά δεδομένων VBR.

  Το πρότυπο AAL3/4 μπορεί να χειρίζεται δεδομένα VBR με ή χωρίς πρσεγκατάσταση μιας σύνδεσης ATM. Παράδειγμα μεταφοράς με σύνδεση ATM είναι η μεταφορά μεγάλων αρχείων, όπως αρχεία CAD ή δεδομένα backup. Η μεταφορά δεδομένων χωρίς σύνδεση χρησιμοποιείται για μικρού όγκου μεταφορά, με ρυθμό που μεταβάλλεται δραστικά.

- AAL5, απλό και αποδοτικό στρώμα προσαρμογής (SEAL).

  Το πρότυπο AAL5 είναι μια απλοποιημένη μορφή του AAL3/4 και είναι σχεδιασμένο να ικανοποιεί τις απαιτήσεις τοπικών δικτύων (LAN) υψηλών ταχυτήτων. Το AAL5 προορίζεται για υπηρεσίες VBR με ή χωρίς σύνδεση.

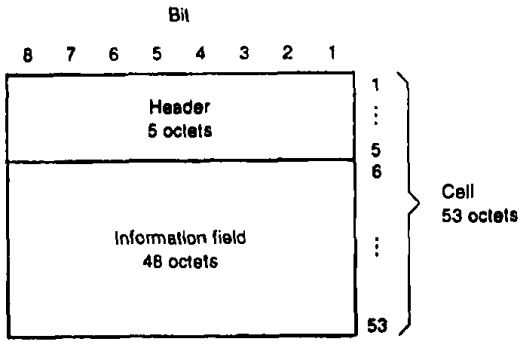
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

2. Στρώμα ATM

2.1 *Δομή πακέτου ATM*

Το πακέτο είναι το βασικό στοιχείο του στρώματος ATM. Ένα πακέτο αποτελείται από επικεφαλίδα των 5 bytes και χρήσιμη πληροφορία των 48 bytes (ATM Service Data Units, ATM-SDUs). Η δομή αυτή φαίνεται στο σχήμα 2.1. Οι ακόλουθες προδιαγραφές ορίζονται από την σύσταση I.361 της ITU-T [2]:

- Τα bytes αποστέλλονται με αύξουσα σειρά αρχίζοντας με το byte 1. Συνεπώς η επικεφαλίδα του πακέτου θα σταλεί πρώτα, ακολουθούμενη από το πεδίο πληροφορίας
- Τα bits εντός του byte στέλνονται με φθίνουσα σειρά αρχίζοντας από το 8° bit.

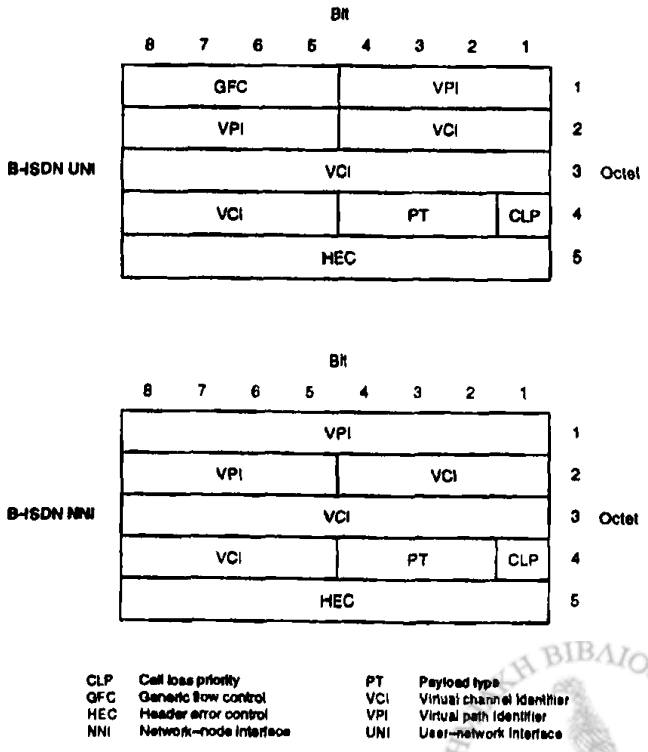


Σχήμα 2.1 : Δομή του cell

2.2 *Επικεφαλίδα του πακέτου*

Στο σχήμα 2.2 απεικονίζεται η επικεφαλίδα του πακέτου για το B-ISDN. Η επικεφαλίδα του πακέτου στη διεπαφή χρήστη-δικτύου B-ISDN (B-ISDN User-Network Interface, UNI) διαφέρει από την διεπαφή δικτύου-δικτύου ανάμεσα σε κόμβους του δικτύου (B-ISDN NNI) [3]. Η διαφορά έγκειται στην χρήση των bits 5-8 του πρώτου byte. Στο B-ISDN NNI, αυτά τα bits είναι μέρος του VPI.

Στις επόμενες παραγράφους περιγράφονται τα διάφορα πεδία της επικεφαλίδας.



Σχήμα 2.2 : Επικεφαλίδα του cell στο B-ISDN UNI και στο B-ISDN NNI

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

2.2.1 Το πεδίο *γενικού ελέγχου ροής (Generic Flow Control, GFC)*

Το πεδίο του γενικού ελέγχου ροής (GFC) αποτελείται από 4 bits και βοηθάει στον έλεγχο της ροής κυκλοφορίας από ATM συνδέσεις σε B-ISDN UNI. Χρησιμοποιείται για να ανακουφίζει συνθήκες υπερφόρτωσης.

2.2.2 Το πεδίο *αναγνώρισης εικονικού μονοπατιού/καναλιού (Virtual Path/Channel Identifier, VPI/VCI)*

Το πεδίο ανάγνωρισης του εικονικού μονοπατιού (VPI) στο B-ISDN UNI αποτελείται από 8 bits, ενώ το VPI στο NNI συντίθεται από τα πρώτα 12 bits της επικεφαλίδας του πακέτου. Το πεδίο αναγνώρισης του εικονικού καναλιού (VCI) αποτελείται από 16 bits. Το πεδίο VPI μαζί με το πεδίο VCI δίνουν την δυνατότητα δρομολόγησης του πακέτου. Στο πίνακα 3.1 φαίνονται οι προκαθορισμένες τιμές των VPI/VCI.

Προκαθορισμένες τιμές επικεφαλίδας του πακέτου χρησιμοποιούνται για την διαφοροποίηση των πακέτων για χρήση στο στρώμα ATM, από εκείνα τα πακέτα για χρήση στο φυσικό στρώμα. Πακέτα για χρήση στο φυσικό στρώμα και μη-προκαθορισμένα πακέτα χαρακτηρίζονται από μηδενικά στα bits 5-28 της επικεφαλίδας. Όλες οι άλλες δυνατές τιμές της επικεφαλίδας του πακέτου μπορούν να χρησιμοποιηθούν από προκαθορισμένα πακέτα. Η διαφορά μεταξύ πακέτων για χρήση στο φυσικό στρώμα και μη-προκαθορισμένων πακέτων εκφράζεται από το λιγότερο σημαντικό bit του τέρτατού byte της επικεφαλίδας του πακέτου.

Type	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5
Reserved for use of the physical layer	pppp0000	00000000	00000000	0000ppp1	HEC valid code
Unassigned cell	gggg0000	00000000	00000000	0000xxx0	HEC valid code
Meta-signalling	ggggyyyy	yyyy0000	00000000	00010a00	HEC valid code
General broadcast signalling	ggggyyyy	yyyy0000	00000000	00100aa0	HEC valid code
Point-to-point signalling	ggggyyyy	yyyy0000	00000000	01010aa0	HEC valid code
F4 segment OAM flow	ggggzzzz	zzzz0000	00000000	00110a0a	HEC valid code
F4 end-to-end OAM flow	ggggzzzz	zzzz0000	00000000	01000a0a	HEC valid code
Resource management	ggggzzzz	zzzzvvvv	vvvvvvvv	vvvv110a	HEC valid code
Reserved for future functions	ggggzzzz	zzzzvvvv	vvvvvvvv	vvvv111a	HEC valid code

VCI values from decimal 8 to 15 are reserved for future use
a indicates the bit is available for use by the appropriate ATM layer function
g indicates the bit is available for use by the GFC protocol
p indicates the bit is available for use by the physical layer
v indicates any VCI value other than 0
x indicates the bit is a 'don't care' bit
y any VPI value. For VPI = 0 the specified VCI value is reserved for user signalling with the local exchange
z any VPI value

Πίνακας 2.1 : Προκαθορισμένες τιμές επικεφαλίδας cell στο UNI

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

2.2.3 *Τύπος ωφέλιμου φορτίου (Payload Type, PT)*

Τρία bits επικεφαλίδας χρησιμοποιούνται για την αναγνώριση του τύπου του ωφέλιμου φορτίου. Στον πίνακα 2.2 παρατίθενται οι τύποι ωφέλιμου φορτίου (PTI). Τα πακέτα του ωφέλιμου φορτίου του χρήστη περιέχουν τόσο πληροφορίες για το χρήστη όσο και πληροφορίες για λειτουργίες της υπηρεσίας προσαρμογής.

PTI	Interpretation
000	User data cell, congestion not experienced, ATM-layer-user-to-ATM-layer-user indication = 0
001	User data cell, congestion not experienced, ATM-layer-user-to-ATM-layer-user indication = 1
010	User data cell, congestion experienced, ATM-layer-user-to-ATM-layer-user indication = 0
011	User data cell, congestion experienced, ATM-layer-user-to-ATM-layer-user indication = 1
100	OAM F5 segment associated cell
101	OAM F5 end-to-end associated cell
110	Resource management cell
111	Reserved for future functions

Πίνακας 2.2 : Τιμές PTI

2.2.4 *Απώλεια προτεραιότητας πακέτου (Cell Loss Priority, CLP)*

Το πεδίο της απώλειας προτεραιότητας πακέτου (CLP) αποτελείται από ένα bit, το οποίο χρησιμοποιείται για να δηλώσει την απώλεια προτεραιότητας του πακέτου. Εάν η τιμή του CLP bit είναι '1' το πακέτο ίσως απορρηφθεί. Αυτό εξαρτάται από τις συνθήκες του δικτύου. Στην περίπτωση CLP = '0', το πακέτο έχει υψηλή προτεραιότητα, και επομένως πρέπει να δοθούν για την μεταφορά του επαρκείς πόροι του δικτύου. Πακέτα που ανήκουν σε διασυνδέσεις σταθερού ρυθμού μεταφοράς (Constant Bit Rate) έχουν υψηλή προτεραιότητα.

2.2.5 Πεδίο *ελέγχου λάθους επικεφαλίδας (Header Error Control, HEC)*

Το πεδίο HEC είναι μέρος της επικεφαλίδας του πακέτου αλλά δεν χρησιμοποιείται στο στρώμα ATM. Περιέχει την ακολουθία ελέγχου λάθους επικεφαλίδας (HEC), η οποία δημιουργείται και ελέγχεται στο φυσικό στρώμα. Ο μηχανισμός HEC περιγράφεται στην σύσταση I.432 της ITU-T [3].

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

3. Σχετικά με τη VHDL

Η VHDL [4] ξεκίνησε στα πλαίσια του προγράμματος ανάπτυξης ολοκληρωμένων κυκλωμάτων πολύ υψηλής ταχύτητας (Very High Speed Integrated Circuits, VHSIC) που προώθησε η Αμερικανική Κυβέρνηση -και ειδικά το Υπουργείο Άμυνας- στις αρχές της δεκαετίας του 1980. Ο στόχος ήταν να αναπτυχθεί ένας τρόπος σχεδίασης που θα προσέφερε ευελιξία, ευκολία και ανεξαρτησία από το χρησιμοποιούμενο hardware, ενώ ταυτόχρονα θα βοηθούσε στην προσομοίωση μέσω ηλεκτρονικού υπολογιστή του σχεδιαζόμενου κυκλώματος καθώς και στην ανεύρεση πιθανών βλαβών πάνω σ' αυτό. Το πρόγραμμα VHSIC κατέληξε στη δημιουργία μιας ιδιαίτερα ισχυρής γλώσσας, η οποία εξ' αιτίας της ευελιξίας της προτυποποιήθηκε σχεδόν αμέσως από την IEEE με αποτέλεσμα σήμερα να αποτελεί ένα διεθνές πρότυπο. Βεβαίως, από τότε μέχρι σήμερα έχουν αναπτυχθεί αρκετές και ισχυρές γλώσσες περιγραφής υλικού, καμμιά όμως δεν φαίνεται προς το παρόν να πλησιάζει τα προσόντα και τις δυνατότητες της VHDL. Σήμερα, η απευθείας σχεδίαση κυκλωμάτων μέσω σχηματικών διαγραμμάτων γίνεται όλο και σπανιότερη και περιορίζεται σταδιακά σε μερικές μόνο ιδιόρυθμες περιπτώσεις που απαιτούν άμεσο έλεγχο πάνω στο hardware. Πολλοί χρήστες μεταξύ των οποίων και το Αμερικανικό Υπουργείο Άμυνας απαιτούν οι σχεδιάσεις συστημάτων ASIC που γίνονται για λογαριασμό του να περιγράφονται με κώδικα VHDL.

Πριν τη VHDL δεν υπήρχε στην αγορά κάποιο καθιερωμένο πρότυπο γλώσσας περιγραφής υλικού. Οι προϋπάρχουσες γλώσσες ήταν προσαρμοσμένες στα χαρακτηριστικά του πακέτου σχεδίασης και του αντίστοιχου προσομοιωτή και -το κυριότερο- ήταν κάθε φορά εξαρτώμενες από την τεχνολογία κατασκευής. Η VHDL κατάφερε να ξεπεράσει αυτό το εμπόδιο, κάτι που της δίνει τη δυνατότητα και την ευελιξία να ενσωματώσει οποιεσδήποτε μελλοντικές τεχνολογικές αλλαγές και προόδους. Επιπλέον, διαθέτει έναν σημαντικό αριθμό από πλεονεκτήματα σε σχέση με άλλες "συμβατικές" γλώσσες περιγραφής υλικού, όπως:

- **Αποτελεί ένα "ανοικτό για όλους" πρότυπο:** Ένα από τα βασικότερα χαρακτηριστικά της γλώσσας είναι ότι έχει καταβληθεί μεγάλη προσπάθεια για την ευρεία διάδοσή της σε κάθε ενδιαφερόμενο στην τεχνολογία σχεδίασης hardware, χωρίς να αποτελεί κρατικό "μυστικό" ή πνευματική ιδιοκτησία κάποιας εταιρίας που την έχει δημιουργήσει. Αυτός ήταν άλλωστε ο λόγος που η προτυποποίησή της από την IEEE έγινε σχεδόν αμέσως μετά τη λήξη του κυβερνητικού προγράμματος VHSIC χάρη στο οποίο αναπτύχθηκε.
- **Υποστηρίζει ένα μεγάλο αριθμό από μεθοδολογίες και τεχνικές σχεδίασης** (π.χ. σύγχρονη ή ασύγχρονη), κάτι που καθιστά συμβατούς μεταξύ τους χρήστες που χρησιμοποιούν διαφορετικές σχεδιαστικές προσεγγίσεις. Απ' την άλλη, οι βιομηχανίες κατασκευής ηλεκτρονικών παρέχουν βιβλιοθήκες

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

περιγραφής των εξαρτημάτων που παράγουν, καθώς η γλώσσα είναι τυποποιημένη και διαθέσιμη στον καθένα.

- **Είναι ανεξάρτητη από την τεχνολογία και τις διαδικασίες υλοποίησης και κατασκευής.** Η γλώσσα δεν εμπεριέχει κάποιο στοιχείο χαμηλού επιπέδου π.χ. αν πρόκειται για τεχνολογία CMOS, GaAs, Bipolar, HMOS κλπ στο επίπεδο πύλης, αν και τέτοιου είδους πληροφορίες μπορούν πολύ εύκολα να ενσωματωθούν και να γίνουν κατανοητές από τη VHDL.
- **Χρησιμοποιεί ένα ευρύ φάσμα δυνατοτήτων περιγραφής,** τόσο σε χαμηλό (με τη μορφή λογικών πυλών) όσο και σε υψηλό επίπεδο (περιγραφή της συμπεριφοράς ολόκληρου του συστήματος). Έτσι, είναι δυνατό η όλη σχεδίαση να διαχωριστεί σε υποσυστήματα για τα οποία υπάρχει η "υψηλού" επιπέδου περιγραφή τους χωρίς να απαιτούνται λεπτομέρειες της εσωτερικής τους δομής προκειμένου να πραγματοποιηθεί η προσομοίωση λειτουργίας τους. Οι περισσότερες από τις προϋπάρχουσες γλώσσες εστιάζονταν είτε μόνο στο χαμηλό είτε μόνο στο υψηλό επίπεδο περιγραφής, αλλά όχι και στα δυο. Η VHDL αντιθέτως, είναι ικανή να κάνει και τα δύο -και μάλιστα τότε αποδίδει τα μέγιστα- κάτι που δίνει την ευελιξία στο σχεδιαστή να συγχωνεύσει ταυτόχρονα στη σχεδίαση του μονάδες για τις οποίες υπάρχει τόσο λεπτομερής όσο και γενική περιγραφή της συμπεριφοράς τους.
- **Προσφέρει μεγάλες δυνατότητες ανταλλαγής πληροφοριών για κάθε σχεδίαση που αποτελείται από ένα ικανό πλήθος πολύπλοκων υποσυστημάτων.** Η VHDL είναι πλέον διεθνές πρότυπο, και επομένως είναι συμβατή με όλα τα σχεδιαστικά πακέτα και προγράμματα που ανταποκρίνονται στις προδιαγραφές της. Αυτό σημαίνει ότι ο κώδικας VHDL που έχει γραφτεί σε ένα μηχάνημα μπορεί να μεταφερθεί χωρίς τροποποιήσεις σε οποιοδήποτε άλλο (ακόμα κι αν αυτό χρησιμοποιεί διαφορετικό λειτουργικό σύστημα) και να παράγει ακριβώς τα ίδια αποτελέσματα κατά τη διαδικασία της προσομοίωσης. Επι πλέον, οι βιβλιοθήκες εξαρτημάτων που κατασκευάζονται από τις εταιρίες, είναι δυνατό να μεταφερθούν αυτούσιες σε οποιοδήποτε υπολογιστή και σε οποιοδήποτε σχεδιαστικό πρόγραμμα που ακολουθεί πιστά τις δομές και την αρχιτεκτονική της γλώσσας. Συνέπεια όλων αυτών είναι ότι το προς σχεδίαση σύστημα είναι δυνατόν να τεμαχιστεί σε έναν αριθμό υποσυστημάτων, καθένα εκ των οποίων μπορεί να υλοποιηθεί και από ένα διαφορετικό φορέα, εταιρία ή οργανισμό. Ο συντονιστής του προγράμματος έχει μια λειτουργική περιγραφή υψηλού επιπέδου του όλου συστήματος καθώς και των υπομονάδων του και μπορεί να κάνει την προσομοίωση χρησιμοποιώντας μόνο τις πληροφορίες αυτές, χωρίς να περιμένει το αποτέλεσμα της δουλειάς των επι μέρους εταίρων. Απ' την άλλη, οι τελευταίοι μπορούν να αναπτύξουν την κάθε μια μονάδα ξεχωριστά, διαθέτοντας μόνο την περιγραφή συμπεριφοράς της και χωρίς να ενδιαφέρονται καθόλου για το υπόλοιπο τμήμα του έργου. Με τον τρόπο αυτό η διαδικασία της σχεδίασης διευκολύνεται κατά πολύ, καθώς το debugging

γίνεται πλέον μια αρκετά απλούστερη υπόθεση, ενώ ο απαιτούμενος χρόνος υλοποίησης μειώνεται επίσης αισθητά.

- **Έχει δομή που ανταποκρίνεται στις σύγχρονες απαιτήσεις και προδιαγραφές,** κατά τον ίδιο τρόπο ακριβώς που ακολουθούν οι μοντέρνες γλώσσες προγραμματισμού. Τα υποπρογράμματα, οι διαδικασίες, τα αντικείμενα κλπ. είναι παρόντα και στη VHDL με τη μορφή διεργασιών (processes), αρχιτεκτονικών (architectures), οντοτήτων (entities) και συνιστωσών (components). Το γεγονός αυτό βοηθά ιδιαίτερα το σχεδιαστή κατά τη φάση της συγγραφής του προγράμματος, αφού του δίνει τη δυνατότητα να το τεμαχίσει σε επιμέρους ανεξάρτητα τμήματα τα οποία μπορούν στη συνέχεια να προσομοιωθούν ξεχωριστά το ένα απ' το άλλο.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

4. Γενική περιγραφή του συστήματος PDH/ATM/SDH mapper

4.1 *Ορισμοί*

Οι ψηφιακές υπηρεσίες που μπορούν να λειτουργούν με ένα απλό κάλεσμα ενός αναλογικού τηλεφώνου, γνωστές στην Ευρώπη σαν CEPT-0 και στην Αμερική σαν DS-0, έχουν ρυθμό μετάδοσης 64Kb/s και είναι σύγχρονες [5].

Η έννοια του συγχρόνου προέρχεται από τους τεχνικούς των τηλεπικοινωνιών και σημαίνει ότι τα bits για κάθε συγκεκριμένη τηλεφωνική επικοινωνία είναι τοποθετημένα πάντα στην ίδια συγκεκριμένη θέση μέσα στο ψηφιακό πλαίσιο μεταφοράς. Οι CEPT-1 και DS-1 αποτελούνται από 32 κανάλια της CEPT-0 και 24 κανάλια της DS-0, αντίστοιχα. Δηλαδή έχουν προδιαγραφεί ιεραρχικά ανάλογα με την ταχύτητα διάφορα πρότυπα CEPT-2, DS-2, CEPT-3 κλπ. Ανάλογα είναι και οι ανώτερες υπηρεσίες. Όμως όλα τα παραπάνω πρότυπα εκτός του CEPT-0 και DS-0 δεν είναι σύγχρονα αλλά σχεδόν σύγχρονα. Αυτή η ιεραρχία προτύπων είναι γνωστή ως σχεδόν σύγχρονη ψηφιακή ιεραρχία (Plesiochronous Digital Hierarchy). Οι απαιτήσεις για σύγχρονη επικοινωνία οδήγησαν στον καθορισμό της σύγχρονης ψηφιακής ιεραρχίας (Synchronous Digital Hierarchy).

Στο σημείο αυτό είναι αναγκαίο να δοθεί μια σύντομη περιγραφή των ιεραρχιών:

- Plesiochronous Digital Hierarchy – PDH : Η πληροφορία μεταδίδεται σε πλαίσια, το καθένα από τα οποία έχει χρόνο μετάδοσης 125 μs (ή το 1/8000 του δευτερολέπτου). Κάθε πλαίσιο αποτελείται από 32 κανάλια CEPT-0s, των 8 bytes. Έτσι το συνολικό μέγεθος του πλαισίου είναι 256 bytes και το εύρος ζώνης είναι 2,048 Mb/s. Ο παραπάνω τρόπος μεταφοράς είναι γνωστός σαν E-1 και αποτελεί Ευρωπαϊκό πρότυπο (η επίσημη ονομασία του είναι CEPT-1). Το αντίστοιχο Αμερικάνικο πρότυπο είναι το T-1, το οποίο διαφέρει στον αριθμό των καναλιών που μπορεί να μεταφέρει ένα πλαίσιο που σε αυτή τη περίπτωση είναι 24 DS-0s των 8 bits. Έτσι το πλαίσιο έχει 192 bits σύν 1 bit πλαισίου και έτσι το εύρος ζώνης είναι 1,544 Mb/s (DS-1). Το E-1 πολυπλέκεται σε μεγαλύτερης χωρητικότητας καλώδια, δημιουργώντας έτσι μια ιεραρχία από τρόπους μεταφοράς (καθένας από τους οποίους προκύπτει από πολύπλεξη των κατώτερων ιεραρχιών), η οποία είναι γνωστή ως σχεδόν σύγχρονη ψηφιακή ιεραρχία (PDH – Plesiochronous Digital Hierarchy). Όπως προαναφέρθηκε, ο τρόπος μετάδοσης δεν είναι σύγχρονος και η λύση που βρέθηκε ώστε να υπάρχει σύγχρονη μεταφορά, ήταν η εισαγωγή επιπλέον bits (stuffing bits) σε κάθε πλαίσιο για να απαληφθούν οι διαφορές χρονισμού των καναλιών (jitter). Στο παρακάτω πίνακα φαίνεται η σχεδόν σύγχρονη ψηφιακή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

ιεραρχία σε Αμερική, Ευρώπη και Ιαπωνία καθώς και οι διεθνείς ρυθμοί μετάδοσης.

Επίπεδο Ιεραρχίας n	*Ευρώπη CEPT – n (Kb/s)*	*Αμερική DS – n (Kb/s)*	*Ιαπωνία (Kb/s)*	*Διεθνείς Ρυθμοί (Kb/s)*
0	64	64	64	64
1	2048	1544	1544	2048
2	8448	6312	6312	6312
3	34368	44736	32064	44736
4	139264	139264	97728	139264

- Synchronous Digital Hierarchy – SDH : Αν και το PDH είναι αρκετά ικανοποιητικό και οπωσδήποτε μια τεράστια βελτίωση του προηγούμενου αναλογικού τηλεφωνικού συστήματος, εν τούτοις έχει αρκετά προβλήματα και ιδιαιτερότητες, μερικές από τις οποίες συνοψίζονται παρακάτω :

  1. Το Αμερικάνικο και το Ευρωπαϊκό τηλεφωνικό σύστημα έχουν λίγα κοινά στοιχεία. Για υπεραντλαντικές συνδέσεις χρειάζονται ακριβά συστήματα μετατροπής .
  2. Πολλές φορές συστήματα τα οποία έχουν κατασκευστεί για τι ίδιο πρότυπο από διαφορετικούς κατασκευαστές είναι λειτουργικά ασύμβατα μεταξύ τους
  3. Δεν υπάρχουν πρότυπα για διασυνδέσεις πολύ υψηλού εύρους ζώνης
  4. Το δίκτυο δεν είναι σύγχρονο για εύρος ζώνης μεγαλύτερο του T-1. Η μετατροπή του σε σύγχρονο γίνενται με την εισαγωγή επιπλέον ψηφίων (stuffing bits)

  Για να λύσουν το πρόβλημα του συγχρονισμού, οι διάφορες εταιρείες τηλεπικοινωνιών κατέληξαν στην ιδέα του δείκτη (pointer), σύμφωνα με την οποία η αρχή των περιεχομένων ενός πλαισίου γίνενται γνωστή από τη τιμή ενός δείκτη στην επικεφαλίδα του. Το αποτελέσμα είναι η αντικατάσταση του PDH με ένα καλύτερο σύστημα, όπως η Σύγχρονη Ψηφιακή Ιεραρχία (SDH – Sychronous Digital Hierarchy). Οι ρυθμοί μετάδοσης (σε Mb/s) για το διεθνές πρότυπο Synchronous Optical NETwork/SDH, μαζί με το αντίστοιχο επίπεδο ιεραρχίας του SONET OC-n, τον αριθμό των συνδέσεων DS-3s που υποστηρίζονται από τον οπτικό μεταφορέα (OC), το αντίστοιχο πρότυπο SDH STM-n "Synchronous Transmition Module" και τον αριθμό των CEPT-4s διασυνδέσεων που υποστηρίζονται από το SDH φαίνονται στο παρακάτω πίνακα :

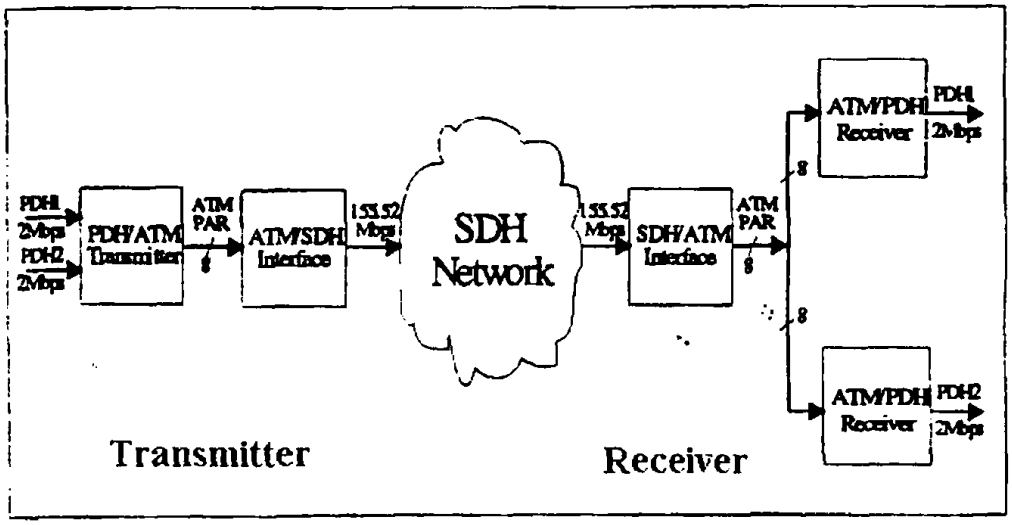
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Επίπεδο Ιεράρχίας	*Ρυθμός Μετάδοσης (Mb/s)*	*SONET*	*DS-3s*	*SDH*	*CEPT-4*
0	155,52	OC-3	3	STM-1	1
1	622,08	OC-12	12	STM-4	4
2	2488,22	OC-48	48	STM-16	16

Το SDH μπορεί να μεταφέρει πολλά πλαίσια μέσω οπτικής ίνας που το κάθε πλαίσιο μπορεί να είναι DS-3, FDDI, CEPT-4, ATM κ.λ.π. Τα επίσημα πρότυπα της CCITT γι' αυτό το υποσύνολο του SONET και του SDH είναι το G.707 [6], το G.708 [7] και το G.709 [8]. Αυτά προδιαγράφουν τους παραπάνω ρυθμούς βασιζόμενοι στην πολυπλεξία των DS-3s και CEPT-4s καθώς και αρκετών επιπλέον ψηφίων για λόγους διαχείρησης και συντήρησης.

4.2 *Γενική περιγραφή*

Το σύστημα απεικόνισης και πολυπλεξίας σχεδόν σύγχρονης ψηφιακής ιεραρχίας (PDH) δύο γραμμών 2 Mbit/s σε σύγχρονη ψηφιακή ιεραρχία (SDH) 155 Mbit/s με ATM [9], οι γενικές αρχές του οποίου παρουσιάζονται στο σχήμα 4.1, εκτελεί τις ακόλουθες βασικές εργασίες :



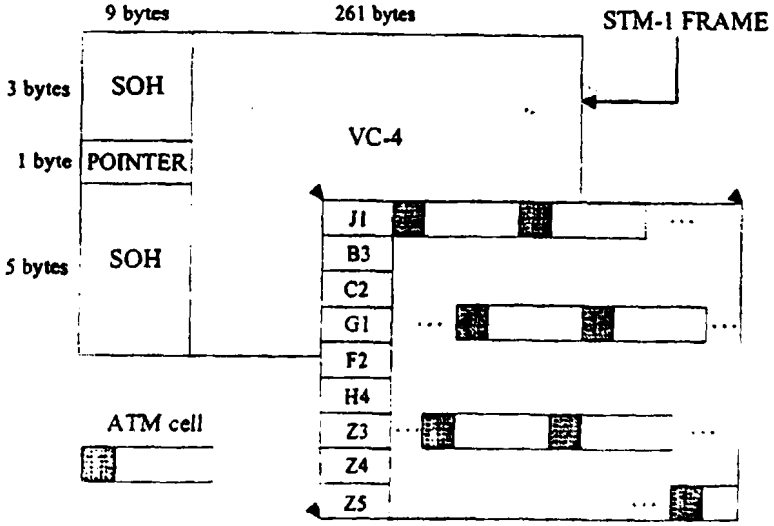
Σχήμα 4.1 : Γενική αρχιτεκτονική του συστήματος PDH/ATM/SDH mapper

1. Στην κατεύθυνση του εκπομπού (εκπομπός ονομάζεται η πλευρά που εκπέμπει πακέτα ATM προς το SDH), εισάγονται τα δεδομένα προς μετάδοση, τα οποία προέρχονται από δύο γραμμές PDH 2 Mbit/s. Τα εισερχόμενα δεδομένα, διασπώνται σε ομάδες των 47 bytes και αφού δημιουργηθούν κατάλληλες επικεφαλίδες του ενός byte, διαμορφώνονται πλαίσια που ονομάζονται AAL-SDUs. Οι 48άδες αυτές στη συνέχεια πλαισιώνονται με άλλα 5 bytes και έτσι

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

παράγονται τα πακέτα μετάδοσης των ATM συστημάτων. Η διεπαφή SDH που ακολουθεί στη συνέχεια, διαβάζει τα πακέτα, τα μετατρέπει σε σειριακά και τα μεταδίδει με τη κατάλληλη πλαισίωση προς το SDH, μαζί με όλα τα απαραίτητα σήματα χρονισμού και ελέγχου που απαιτούνται από τις προδιαγραφές του SDH. Ουσιαστικά δηλαδή, τα δεδομένα εισόδου επεξεργάζονται διαδοχικά από το στρώμα AAL και το στρώμα ATM του μοντέλου αναφοράς του B-ISDN δικτύου, σύμφωνα με τη σύσταση I.321 της ITU-T/CCITT [10]. Στα δύο εισερχόμενα κανάλια δίνονται στο στρώμα ATM διαφορετικές τιμές διευθύνσεων στις επικεφαλίδες και είναι δυνατόν να σταλούν με πολύπλεξη μέσω του εκπομπού ATM.

2.Στον δέκτη - δέκτης ονομάζεται η πλευρά που λαμβάνει πακέτα ATM από το SDH - αρχικά τα δεδομένα SDH που έρχονται σειριακά, περνούν μέσα από τις διεπαφές SDH, οι οποίες εκτελούν τη μετατροπή των σειριακών δεδομένων σε παράλληλα, πραγματοποιούν την ανάκτηση ρολογιού SDH, διαμορφώνοντας έτσι τα πακέτα ATM που αποστέλλονται στη συνέχεια στον δέκτη ATM. Ο τελευταίος, αφού αναγνώσει τις ανάλογες επικεφαλίδες, οι οποίες περιέχουν την διεύθυνση που δείχνει σε ποια από τις δύο συνδέσεις PDH ανήκει το πακέτο, μεταβιβάζει τα χρήσιμα δεδομένα στο στρώμα AAL, με σκοπό να γίνει λεπτομερέστερος έλεγχος λαθών (απώλειας πακέτων) και επί πλέον ανάκτηση του ρολογιού υπηρεσίας (ρολόϊ PDH) του πομπού. Στο στρώμα AAL τα δεδομένα αφού εισαχθούν πρώτα σε καταχωρητές FIFO οι οποίες βοηθούν στην



Σχήμα 4.2 : Mapping των ATM cells μέσα στα STM-1 frames

προσαρμογή των ρυθμών μετάδοσης, εκπέμπονται ξανά (μέσω των διεπαφών PDH), σε αντίστοιχες γραμμές PDH με αυτές που είχαν εισαχθεί στον εκπομπό. Η χαρτογράφηση (mapping) των πακέτων ATM μέσα στο πλαίσιο SDH γίνεται σύμφωνα με τη σύσταση G.709 της ITU-T [8]. Τα ολοκληρωμένα πακέτα των 53 bytes μεταφέρονται μέσα στο πλαίσιο VC-4 του STM-1 (155.52 Mbit/s), όπως φαίνεται στο σχήμα 4.2. Επειδή το μέγεθος του VC-4 (260 x 9 bytes) δεν είναι

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

ακέραιο πολλαπλάσιο του αντίστοιχου μεγέθους των πακέτων ATM (53 bytes), κάποια από αυτά μεταδίδονται ολόκληρα, ενώ άλλα διαχωρίζονται σε δύο μέρη και η εκπομπή του δεύτερου μέρους συνεχίζεται με το επόμενο πλαίσιο. Στη συνέχεια προστίθενται οι απαραίτητες πληροφορίες ελέγχου POH, Path Overhead και μαζί με το SOH, Section Overhead και τον ανάλογο δείκτη συναποτελούν το πλαίσιο του συστήματος STM-1 της σύγχρονης ψηφιακής ιεραρχίας [5].

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΠΑΤΡΩΝ

5. Περιγραφή του δέκτη ATM/PDH

Στο κεφάλαιο αυτό παρατίθονται πληροφορίες για το δέκτη ATM/PDH σε γλώσσα VHDL και προσομοιώθηκε. Ο δέκτης αποτελείται από την μονάδα ATMRX, η οποία εκτελεί όλες τις λειτουργίες του στρώματος ATM και από την μονάδα AALRX, η οποία εκτελεί τις λειτουργίες του στρώματος AAL [10]. Είναι απαραίτητο να σημειώσουμε ότι η μονάδα AALRX είναι ίδια και για τις δύο ροές χρήστη (user stream).

5.1 *Στρώμα ATM του δέκτη (ATMRX)*

5.1.1 *Γενικά*

Στο τμήμα αυτό του δέκτη, εισέρχονται τα πακέτα ATM (53 bytes) μέσω της διεπαφής SDH, η οποία υλοποιείται με το ολοκληρωμένο κύκλωμα TNETA [11], όπου αναγνωρίζονται (on fly) οι υποστηριζόμενες από τον δέκτη επικεφαλίδες πακέτων. Στην συνέχεια μεταβιβάζονται μόνο τα δεδομένα του χρήστη (payload) στο στρώμα AAL, με σκοπό να γίνει λεπτομερέστερος έλεγχος λαθών (απώλειας πακέτων) και επιπλέον ανάκτηση του ρολογιού (PDH clock) του πομπού. Το συγκεκριμένο στρώμα ATM υποστηρίζει δεκατέσσερα είδη επικεφαλίδων (άρα και δεκατέσσερα αντίστοιχα είδη πακέτων ATM), πιό αναλυτικά, δύο είδη για τις δύο ροές χρήστη, έξι είδη διαχείρησης και έξι είδη σηματοδοσίας [9].

Ένα ιδιαίτερο χαρακτηριστικό της μονάδας αυτής είναι ότι οι επικεφαλίδες που ελέγχει το σύστημα δεν είναι στατικές αλλά δυναμικές, δηλαδή μέσω της διεπαφής του υπαρχοντος στο PCB μικροελεγκτή, φορτώνονται οι εκάστοτε τιμές τους στη RAM της μονάδας ATMRX. Το παραπάνω κάνει την μονάδα ATMRX ευέλικτη ως προς την υποστήριξη διαφορετικών ειδών επικεφαλίδων. Ένα άλλο χαρακτηριστικό της μονάδας αυτής είναι η αυτόματη αναγνώριση των επικεφαλίδων (on fly operation). Δηλαδή, καθώς εισέρχεται στην μονάδα το πακέτο ATM γίνεται ταυτόχρονα και η αναγνώρισή του μέσω της παράλληλης σύγκρισής του με τις τιμές των επικεφαλίδων που είναι αποθηκευμένες στη RAM. Στην περίπτωση που η επικεφαλίδα βρεθεί λανθασμένη – δηλ. η τιμή της δεν ισούται με καμμία από τις αποθηκευμένες τιμές των επικεφαλίδων στη RAM - υπάρχει η δυνατότητα να αγνοηθεί η τιμή του CLP (Cell Loss Priority) διότι ίσως έχει αλλάξει λόγω συμφόρησης του δικτύου.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.1.2 *TNETA (Texas Instruments) και η διεπαφή με την μονάδα ATMRX*

Το ολοκληρωμένο TNETA (TDC 1500A) [6] είναι ένας πομπός/δέκτης που χρησιμοποιείται για την μεταφορά πακέτων ATM των 53-bytes μέσω του STM-1 (155,52 Mbit/s). Μία σύνοψη των χαρακτηριστικών του TNETA είναι :

- Υπάρχει αναλογικός βρόγχος κλειδώματος φάσης PLL, το οποίο παρέχει την ανάκτηση του εισερχόμενου ρολογιού από την εισερχόμενη ροή σειριακών δεδομένων και μεταδίδει ρολόϊ από εξωτερική πηγή ρολογιού των 19,44 MHz
- Δέχεται και εξάγει πακέτα ATM σε/από ένα σύγχρονο φάκελο ωφέλιμου φορτίου (Synchronous Payload Envelope, SPE) SONET/SDH STM-1
- Ανιχνεύει λάθη πολλαπλών-bits και διορθώνει λάθη ενός-bit στην 5-bit επικεφαλίδα του εισερχόμενου πακέτου ATM
- Προειδοποιεί στη περίπτωση : που το εισερχόμενο σήμα είναι εκτός πλαισίου (Out Of Frame), απώλειας του εισερχόμενου σειριακού σήματος (Loss of Incoming Serial Signal,LOS), απώλειας πλαισίου (Loss Of Frame), λάθους ομοτιμίας B1-Byte (B1ERR), απώλειας της σειράς του πακέτου ATM (Loss Of ATM Cell Allgnment), αποτυχία λήψης line far-end (Line Far-End Receive Failure), απώλειας του δείκτη (Loss Of Pointer)
- Είναι τεχνολογίας BiCMOS και έχει συσκευασία 144-Pin PQFP

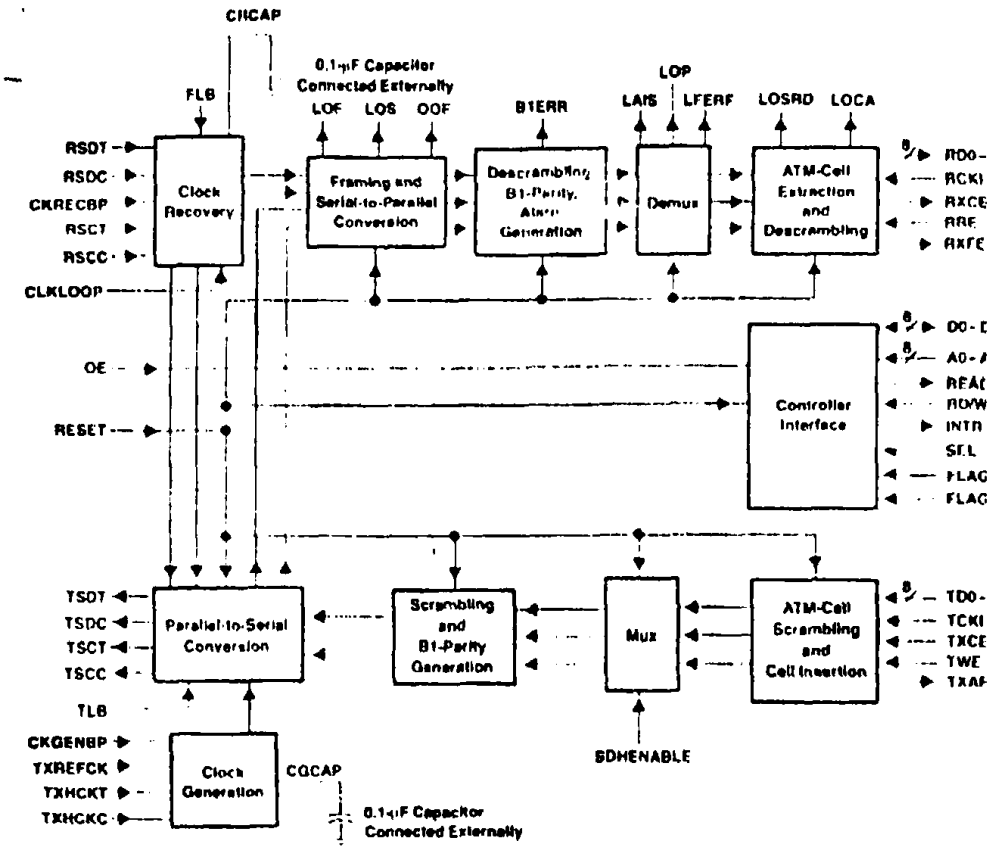
Ως δέκτης, το TNETA δέχεται: τα σειριακά δεδομένα των 155,52 Mbit/s, ανακτά το σήμα ρολογιού, εκτελεί συμμόρφωση πλαισίου SONET/SDH και μετατροπή σειριακής μετάδοσης σε παράλληλη, αναγνωρίζει το ωφέλιμο φορτίο SONET/SDH και οριοθετεί το πακέτο ATM. Τα εξαγόμενα πακέτα ATM από το ωφέλιμο φορτίο, διαχωρίζονται και περνάνε στην FIFO όπου είναι έτοιμα για έξοδο στην επόμενη μονάδα (π.χ. μονάδα ATMRX).

Ως εκπομπός : ολόκληρα τα πακέτα ATM των 53-bytes που είναι τοποθετημένα στη FIFO, εμπλέκονται και εισέρχονται σ' ένα STM-1 Synchronous Payload Envelope. Ένα αναλογικό PLL χρησιμοποιείται για να παράγει το ρολόϊ εξόδου των 155,52 Mbit/s από ένα ταλαντωτή των 19,44 MHz αποφεύγοντας την χρήση ταλαντωτή υψηλής ταχύτητας.

Το λειτουργικό διάγραμμα του ολοκληρωμένου φαίνεται στο σχήμα 5.1. Κατά την διαδικασία της λήψης, το TNETA δέχεται σειριακά δεδομένα, τα οποία αποτελούνται από δεδομένα αληθή και συμπληρωματικά ψευδό-ECL των 155,52 Mbit/s και ένα προαιρετικό ρολόϊ αληθές και συμπληρωματικό ψευδό-ECL των 155,52 Mbit/s. Το εισερχόμενο ρολόϊ των 155,52 Mbit/s χρησιμοποιείται μόνο εάν το εισερχόμενο ανακτώμενο ρολόϊ παράκαμψης (CKRECBP) είναι σε λογική στάθμη 1 και απενεργοποιεί το κύκλωμα ανάκτησης ρολογιού.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Το κύκλωμα ανάκτησης ρολογιού χρησιμοποιείται για να ανακτηθεί το εμπλεκόμενο ρολόϊ από τα εισερχόμενα σειριακά δεδομένα RSDT και RSDC. Το κύκλωμα ανάκτησης ρολογιού αποτελείται από ένα ανιχνευτή μετάβασης, ένα αναλογικό PLL και ένα κύκλωμα επαναχρονισμού:



Σχήμα 5.1 : Λειτουργικό διάγραμμα του TNETA

Ο ανιχνευτής μετάβασης χρησιμοποιείται για να διπλασιάζει την συχνότητα των εισερχόμενων σειριακών δεδομένων. Αυτό είναι απαραίτητο γιατί τα δεδομένα δεν περιέχουν μία δεύτερη αρμονική, η οποία είναι απαραίτητη για να ανακτηθεί το μεταδιδόμενο ρολόϊ.

Το αναλογικό PLL αποτελείται από ένα ανιχνευτή φάσης-συχνότητας, ένα φίλτρο και ένα εσωτερικό ταλαντωτή ελεγχόμενο από τάση (VCO). Ο ανιχνευτής φάσης-συχνότητας συγκρίνει την έξοδο του ανιχνευτή μετάβασης με την έξοδο του VCO και παράγει ένα σήμα στο φίλτρο, το οποίο χρησιμοποιείται για να αλλάξει την συχνότητα του VCO. Η συχνότητα του VCO ρυθμίζεται μέχρι αυτή να ταιριάξει με την συχνότητα του ανιχνευτή μετάβασης. Όταν αυτό συμβεί, το αναλογικό PLL κλειδώνει σ' αυτή τη συχνότητα.

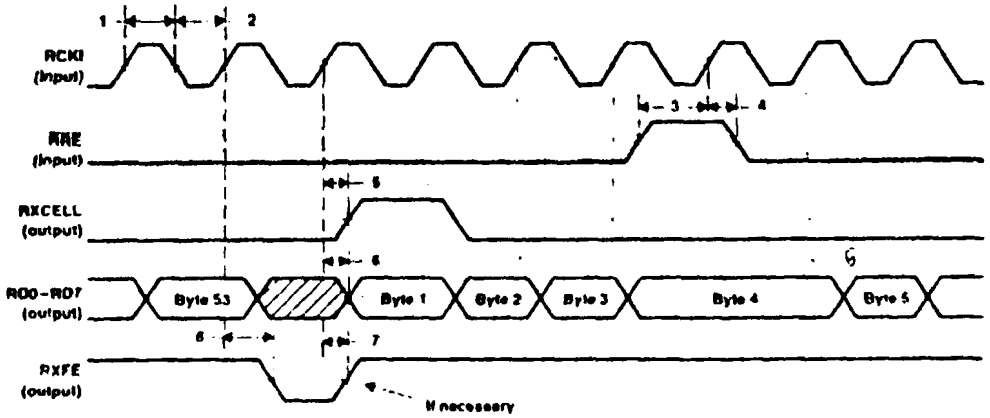
Το κύκλωμα του επαναχρονισμού επαναχρονίζει τα σειριακά εισερχόμενα δεδομένα με το ανακτημένο ρολόϊ εξόδου.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Το ανακτημένο σήμα ρολογιού από τα εισερχόμενα σειριακά δεδομένα μπορεί επίσης να χρησιμοποιηθεί σαν μεταδιδόμενο ρολόϊ στην πλευρά εκπομπής. Αυτό είναι γνωστό σαν ανακύκλωση ρολογιού (clock looping). Η χρήση ανακτημένου ρολογιού σαν μεταδιδόμενο ρολόϊ έχει ως αποτέλεσμα το μεταδιδόμενο ρολόϊ να έχει ακριβώς την συχνότητα του πηγαίου ρολογιού, που χρησιμοποιείται για να αναπαράγει τα εισερχόμενα δεδομένα. Εάν το πηγαίο ρολόϊ είναι πολύ ακριβές τότε το μεταδιδόμενο ρολόϊ είναι επίσης πολύ ακριβές. Το μειονέκτημα της χρήσης του clock looping βρίσκεται στη περίπτωση πουείναι το λαμβανόμενο σήμα για κάποιο λόγο χάνεται. Τότε το μεταδιδόμενο ρολόϊ επίσης χάνεται.

Μόλις το STM-1 SPE εξακριβώνεται, τα πακέτα ATM αναγνωρίζονται και εξάγονται. Η σκιαγράφηση του πακέτου ολοκληρώνεται με τον υπολογισμό του HEC για τα πρώτα τέσσερα bytes μετά το byte J1και συγκρίνοντας την υπολογιζόμενη τιμή με το πέμπτο byte. Εάν η τιμή δεν ταιριάζει, η διαδικασία επαναλαμβάνεται για τα 4 bytes μετά ένα byte από το byte J1 και ούτω καθ' εξής. Θεωρείται ότι υπάρχει συμμόρφωση του πακέτου όταν συμβούν έξι συνεχής επιτυχείς συγκρίσεις. Μέχρι να υπάρξει συμμόρφωση του πακέτου είναι ενεργοποιημένο το σήμα LOCA (lost-of-cell-alignment).

Η πλευρά του δέκτη ανιχνεύει λάθη πολλαπλών-bits και διορθώνει λάθη ενός-bit που λαμβάνουν χώρα στα 5 bytes της επικεφαλίδας των εισερχόμενων πακέτων ATM χρησιμοποιώντας το HEC byte. Η παραπάνω διαδικασία ανίχνευσης απενεργοποιείται θέτοντας ένα bit στο καταχωρητή ελέγχου (Control Register) του οκληρωμένου. Παρέχεται επίσης ένας μετρητής 8-bits, οποίος



Σχήμα 5.2 : Διάγραμμα χρονισμού του Receive-Cell-Interface

μετράει τον αριθμό των πακέτων ATM με λάθη επικεφαλίδας πολλαπλών-bits.

Τα πακέτα ATM αφού εξαχθούν, διαχωρίζονται. Το ωφέλιμο φορτίο των 48-bytes (payload) του πακέτου ATM συντίθεται στον πομπό χρησιμοποιώντας ένα πολυώνυμο $x^{43}+1$ ώστε να ξεχωρίζει το payload από τα bytes της επικεφαλίδας

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

και να βελτειώνει την αποδοτικότητα του αλγορίθμου σκιαγράφησης του πακέτου. Το TNETA παρέχει την δυνατότητα της απόρριψης αδρανών και απροσδιόριστων πακέτων που λαμβάνονται από τα εισερχόμενα δεδομένα. Ένας αδρανές πακέτο ορίζεται σαν ένα πακέτο με επικεφαλίδα ATM των 5-bytes και τιμή 00 00 00 01 52 (hex) και ένα απροσδιόριστο πακέτο ορίζεται σαν ένα πακέτο με επικεφαλίδα 00 00 00-00 55 (hex). Και στις δύο περιπτώσεις, το payload αγνοείται.

Μετά τον διαχωρισμό, το πακέτο ATM περνάει στον απομονωτή εξόδου, ο οποίος λειτουργεί σαν μία FIFO. Η διεπαφή του TNETA με την μονάδα ATMRX αποτελείται από τους ακροδέκτες των δεδομένων εξόδου (RD0 - RD7), του ρολογιού (RCKI), του σήματος ενεργοποίησης ανάγνωσης (/RRE), του σήματος (/RXFE), του δείκτη αρχής του πακέτου ATM (RXCELL) και της προειδοποίησης απώλειας δεδομένων (LOSRD). Τα δεδομένα στέλνονται στην έξοδο κατά την άνοδο της ακμής του RCKI εφόσον το /RRE είναι σε λογικό χαμηλό. Το LORSD ενεργοποιείται όταν υπερχειλίσει η FIFO. Σ' αυτή τη περίπτωση, το τελευταία τοποθετημένο πακέτο μέσα στη FIFO γράφεται στη θέση του προηγούμενου πακέτου που εισήχθηκε. Η FIFO κρατά τρία ολόκληρα πακέτα ATM. Στο σχήμα 5.3 παρουσιάζεται μια συνοπτική περιγραφή της διεπαφής, ενώ στο σχήμα 5.2 φαίνεται το διάγραμμα χρονισμού.

receive-cell interface

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
RCKI	120	I (TTL)	Receive clock input. Output signals are clocked out of the receive-cell interface on positive transitions of RCKI when RRE is low
RD0–RD07	92 - 95 98–101	O	Receive byte data. The ATM cells are clocked out of the TDC1500A through RD0 - RD7 one byte at a time on positive transitions of RCKI, which begins with the first byte of the ATM cell header
RRE	119	I (TTL)	Receive read enable. A low level on RRE enables the reading of data from the receive-cell interface
RXCELL	117	O	Receive start of ATM-cell indicator. RXCELL goes high to identify the first byte of an ATM cell. RXCELL is low during the remainder of the output
RXFE	118	O	Receive FIFO empty. RXFE goes low to denote that the receive FIFO is empty and that the current output byte is not a valid byte. RXFE goes high when a complete ATM cell is available for o...

Σχήμα 5.3: Συνοπτική περιγραφή του Receive-Cell-Interface

5.1.3 *Η μονάδα ATMRX*

Η μονάδα ATMRX αποτελείται από τρεις υπομονάδες: την υπομονάδα αυτόματης αναγνώρισης, την υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων και την υπομονάδα ροής καταστάσεων. Υποστηρίζει δύο ροές χρήστη, έξι ροές σηματοδοσίας και έξι ροές διαχείρησης [9]. Στο σχήμα 5.4 παρουσιάζεται το διάγραμμα της μονάδας αυτής.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Η υπομονάδα αυτόματης αναγνώρισης αποτελείται από οκτώ καταχωρητές (registers) και από μία μονάδα αποπολυπλεξίας και πολυπλεξίας (demultiplexer και multiplexer). Η υπομονάδα της αποθήκευσης και αναγνώρισης των επικεφαλίδων αποτελείται από μία RAM 4 διευθύνσεων 8 bits, ένα μετρητή τεσσάρων καταστάσεων και ένα συγκριτή. Τέλος η υπομονάδα ακολουθίας καταστάσεων (state machine) ελέγχει τη λειτουργία των προηγούμενων υπομονάδων.

Στις επόμενες παραγράφους περιγράφονται λεπτομερέστερα οι λειτουργίες των παραπάνω υπομονάδων.

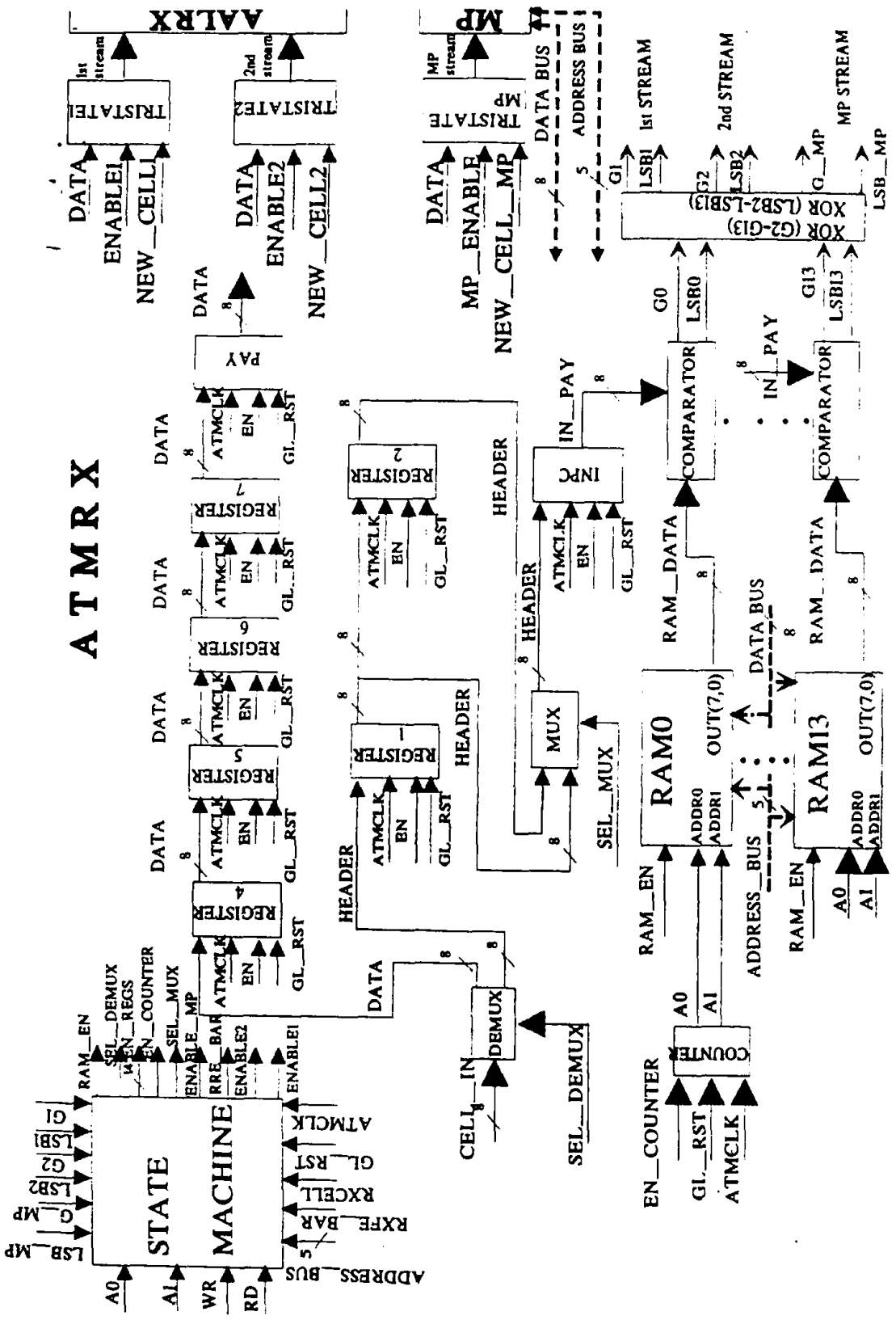
5.1.3.1 Υπομονάδα αυτόματης αναγνώρισης

Στην υπομονάδα αυτή εισέρχεται το πακέτο ATM, αναγνωρίζεται και διαχωρίζεται η επικεφαλίδα από τα δεδομένα και στη συνέχεια ερευνάται ο προορισμός των δεδομένων. Η λειτουργία της υπομονάδας βασίζεται στο γεγονός ότι η επικεφαλίδα του πακέτου ATM συγκρίνεται με τις επικεφαλίδες που υποστηρίζει ο δέκτης. Δηλαδή η επικεφαλίδα οδηγείται σε ένα μονοπάτι που αποτελείται από τρεις καταχωρητές σε σειρά και η έξοδος του τελευταίου καταχωρητή οδηγείται στο συγκριτή της υπομονάδας αποθήκευσης και αναγνώρισης των επικεφαλίδων. Οι τρείς καταχωρητές αποθηκεύουν την υπόλοιπη επικεφαλίδα όταν συγκρίνεται το πρώτο byte. Υπενθυμίζεται ότι για τη σύγκριση της επικεφαλίδας χρησιμοποιούνται τα τέσσερα πρώτα bytes, ενώ το πέμπτο που είναι το HEC και αφορά την ορθότητα του πακέτου, ελέγχεται από το TNETA (HEC verification).

Κατά την διάρκεια της διεύλεσης της επικεφαλίδας από το πρώτο μονοπάτι, τα δεδομένα του πακέτου ATM κατευθύνονται και σ' ένα δεύτερο μονοπάτι που αποτελείται από πέντε καταχωρητές. Οι πέντε αυτοί καταχωρητές χρησιμοποιούνται ώστε να δημιουργηθεί η αναγκαία καθυστέρηση του ωφέλιμου φορτίου που χρειάζεται για την αναγνώριση της επικεφαλίδας. Ανάλογα με το είδος της επικεφαλίδας οδηγούνται τα δεδομένα του χρήστη είτε προς το στρώμα AAL, αν πρόκειται για ένα από τις δύο ροές του χρήστη, είτε προς την διεπαφή του μικροελεγκτή, αν πρόκειται για τις ροές σηματοδοσίας και διαχείρησης, για λεπτομερέστερη επεξεργασία. Αλλοιώς απορρίπτονται.

Ο αποπολυπλέκτης χρησιμοποιείται για να διαχωρίζει την επικεφαλίδα από τα δεδομένα του χρήστη στο πακέτο ATM, ενώ ο πολυπλέκτης μαζί με τους καταχωρητές χρησιμοποιούνται για να διαβάζεται το πακέτο ATM ανά byte σε κάθε παλμό του ρολογιού ATM. Στο σχήμα 5.5 απεικονίζεται το διάγραμμα χρονισμού της υπομονάδας. Παρατηρώντας προσεκτικά το χρονικό διάγραμμα της υπομονάδας φαίνεται ότι δεν διαβάζεται το πέμπτο byte του πακέτου ATM (HEC, header error control) όπως προαναφέρθηκε.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ



Σχήμα 5.4 : Διάγραμμα της μονάδας ATMRX

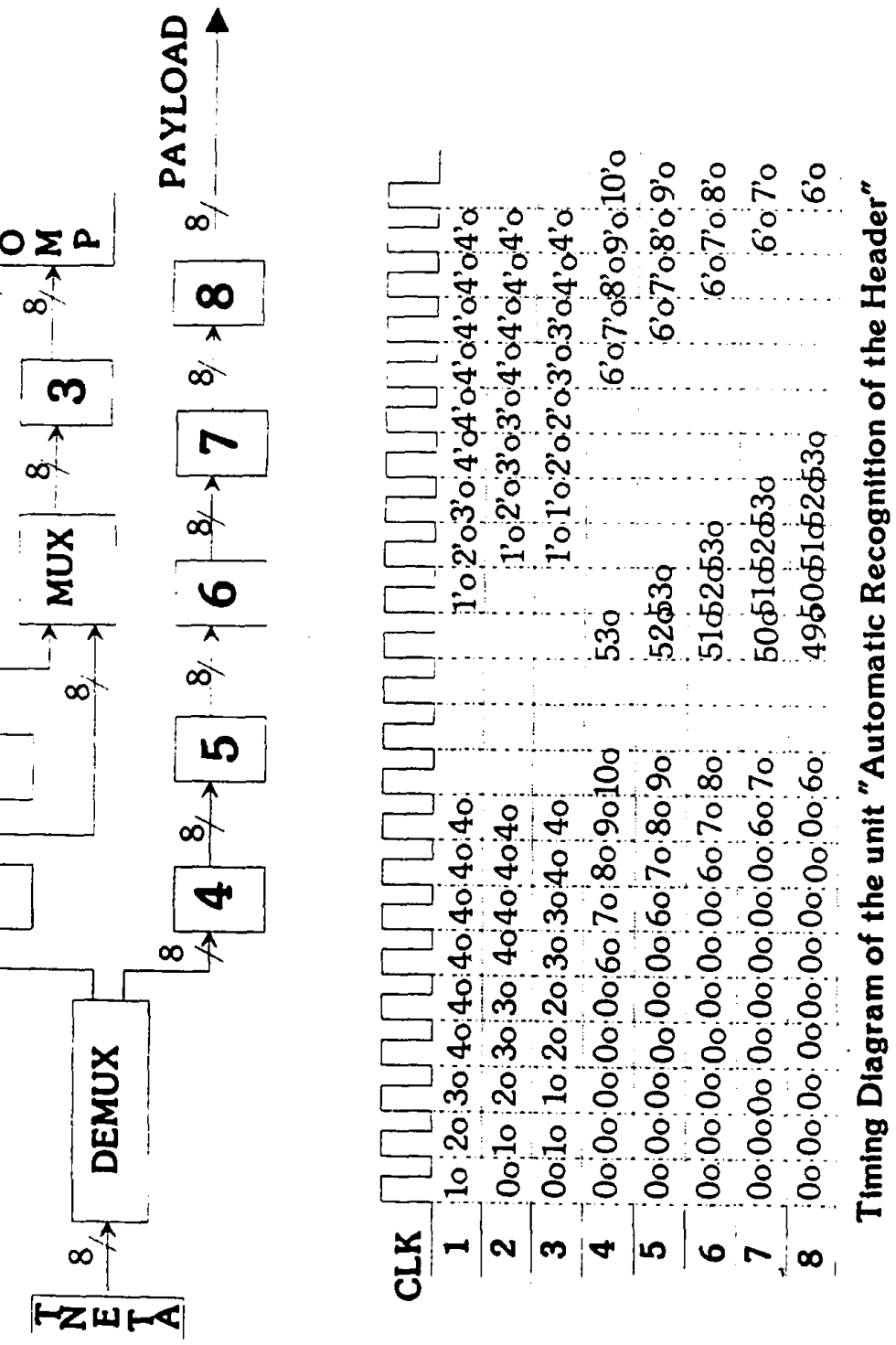
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.1.3.2 *Υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων*

Η εισερχόμενη επικεφαλίδα στην υπομονάδα της αυτόματης αναγνώρισης συγκρίνεται με τις υποστηριζόμενες επικεφαλίδες και εξάγονται δύο σήματα ελέγχου για κάθε ροή του χρήστη και άλλα δύο σήματα για τα έξι είδη πακέτων σηματοδοσίας και τα έξι είδη πακέτων διαχείρησης αντίστοιχα. Οι καταστάσεις των δύο σημάτων ελέγχου δηλώνουν την ορθότητα ή όχι των επικεφαλίδων και επομένως οδηγούν στην έγκριση ή απόριψη του εισερχόμενου πακέτου ATM. Τα δύο σήματα ελέγχου είναι το G (active high) και το LSB. Να σημειωθεί ότι το G, το ένα από τα δύο σήματα ελέγχου, προέρχεται από την σύγκριση και των τεσσάρων bytes της επικεφαλίδας, ενώ το LSB ενεργοποιείται όταν το G πάρει τη τιμή μηδέν (LSB active high) και προέρχεται από την σύγκριση των τριών πρώτων bytes της επικεφαλίδας (δεν λαμβάνεται υπόψη το CLP).

Η υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων αποτελείται κυρίως από RAMs 4x8 (4 διευθύνσεις των 8 bits) και συγκριτές (comparators). Κατά την αρχικοποίηση της μονάδας στην RAM αποθηκεύονται οι επικεφαλίδες μέσω της διεπαφής του μικροελεγκτή. Στην κατάσταση λειτουργίας της υπομονάδας κάθε ένα από τα τέσσερα πρώτα εισερχόμενα bytes της επικεφαλίδας του πακέτου ATM, συγκρίνεται με το πρώτο αποθηκευμένο byte όλων των RAMs και παράγονται τα σήματα ελέγχου. Οι καταστάσεις των σημάτων ελέγχου εισέρχονται στην υπομονάδα ροής καταστάσεων (state machine), η οποία παράγει τα κατάλληλα σήματα. Στην περίπτωση της μη απόρριψης, το ωφέλιμο φορτίο οδηγείται σ' ένα από τα δύο στρώματα AAL (οι δύο ροές χρήστη) ή στην διεπαφή του μικροελεγκτή (ροές σηματοδοσίας και διαχείρησης) ανάλογα με τις λογικές καταστάσεις που βρίσκονται τα έξι σήματα ελέγχου.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 5.5 : Διάγραμμα χρονισμού της διαδικασίας on fly της υπομονάδας αυτόματης αναγνώρισης της επικεφαλίδας του πακέτου ATM

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.1.3.3 *Υπομονάδα ακολουθίας καταστάσεων (state machine)*

Η υπομονάδα ακολουθίας καταστάσεων επικοινωνεί με το TNETA μέσω της ολοκληρωμένης διεπαφής από και προς το SDH. Τα σήματα της διεπαφής με το TNETA και κάποια σήματα ελέγχου από τις άλλες υπομονάδες παράγει, οι οποίες είναι είσοδοι των άλλων μονάδων, για την ομαλή και σωστή λειτουργία του στρώματος ATM (ATMRX). Συγκεκριμένα, κατά την αρχικοποίηση του συστήματος ενεγοποιεί το μικροελεγκτή να φορτώσει στη RAM τις επικεφαλίδες που θα υποστιρίζει ο δέκτης. Ελέγχει τη σωστή λειτουργία του TNETA μέσω των ακροδεκτών των δεδομένων. Στη περίπτωση που το TNETA δεν λειτουργεί σωστά διακόπτει τη λειτουργία της μονάδας ATMRX και ειδοποιεί με σήματα ελέγχου τις άλλες μονάδες (AALRX, μικροελεγκτή). Τέλος στη περίπτωση ομαλής λειτουργίας οδηγεί τη χρήσιμη πληροφορία ATM_SDUs προς τις αντίστοιχες μονάδες AALRX αν πρόκειται για ροή χρήστη, αλλιώς προς το μικροελεγκτή αν πρόκειται για ροή σηματοδοσίας ή διαχείρησης.

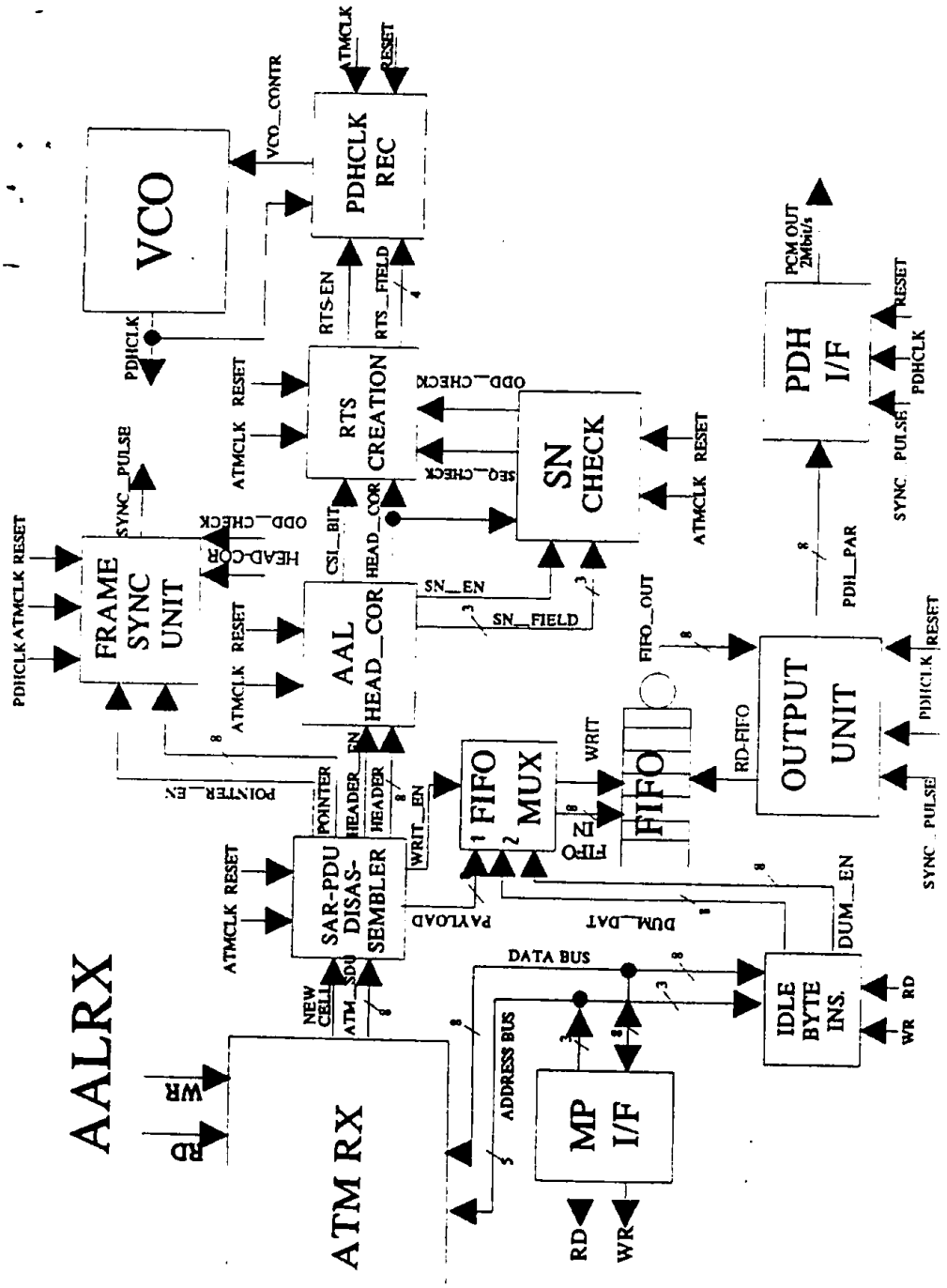
5.2 *Στρώμα AAL του δέκτη (AALRX)*

5.2.1 *Γενικά*

Στο στρώμα AALRX του δέκτη, ίδιο για κάθε ροή χρήστη, εισέρχονται τα ATM SDUs από το στρώμα ATM του δέκτη (ATMRX), με σκοπό να γίνει λεπτομερέστερος έλεγχος λαθών (απώλεια πακέτων) και επιπλέον ανάκτηση του ρολογιού υπηρεσίας (PDH) του εκπομπού. Στο σχήμα 5.6 παρουσιάζεται το διάγραμμα της μονάδας AALRX. Τα δεδομένα αφού πρώτα εισαχθούν σε μία FIFO, η οποία βοηθάει στην προσαρμογή της μετάδοσης, εκπέμπονται, μέσω των διεπαφών PDH στις αντίστοιχες γραμμές PDH.

Οι υποστηριζόμενες από το σύστημα υπηρεσίες, ανήκουν στην κατηγορία AAL-1, ή αλλοιώς πρόκειται για υπηρεσίες πραγματικού χρόνου και μάλιστα σταθερού ρυθμού (CBR, Constant Bit Rate services). Αυτό σημαίνει ότι στο δέκτη είναι επιβεβλημένη η ανάκτηση του ρολογιού υπηρεσίας του εκπομπού, το οποίο στο συγκεκριμένο επικοινωνιακό σύστημα δεν είναι άλλο από το ρολόϊ των διεπαφών PDH.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 5.6 : Διάγραμμα της μονάδας AALRX

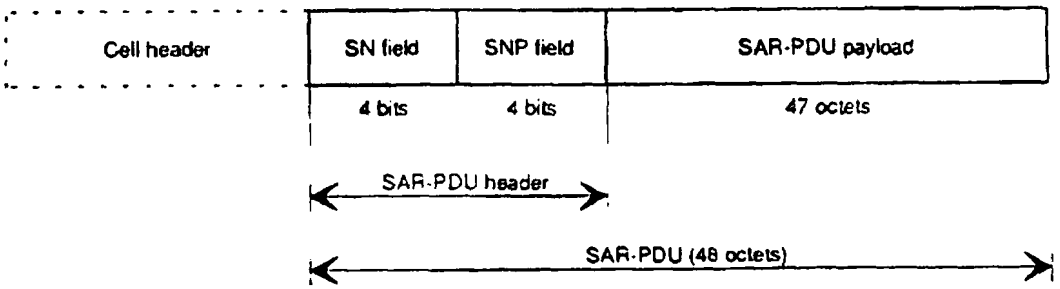
5.2.2 *Η μονάδα AALRX*

Η μονάδα AALRX αποτελείται από τα ακόλουθα τμήματα :

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.2.2.1 Αποσυναρμολογητής των SAR_Protocol Data Units (SAR_PDU DISASSEMBLER)

Ο SAR_PDU DISASSEMBLER διαχωρίζει το ATM_SDU, σε επικεφαλίδα, δείκτη και ωφέλιμο φορτίο. Όλες αυτές οι λειτουργίες υλοποιούνται στο υπόστρωμα κατάτμησης και επανασυναρμολόγησης (SAR) του στρώματος AAL-1 και γι'αυτό και το όνομα της μονάδας είναι SAR_PDU DISASSEMBLER. Το μέγεθος και η θέση των πεδίων του SAR_PDU φαίνονται στο σχήμα 5.7. Η επικεφαλίδα του SAR_PDU περιέχει την πληροφορία αρίθμησης των πακέτων και τα απαραίτητα bits για να γίνει η ανάκτηση του ρολογιού υπηρεσίας (PDHCLK) του πομπού [12]. Συγκεκριμένα τα τέσσερα σημαντικότερα bits αποτελούν το πεδίο ακολουθίας αριθμών (SN, Sequence Number Field) εκ των οποίων 1 bit



Σχήμα 5.7 : Μορφή του SAR-PDU του AAL-1

είναι ο δείκτης υποστρώματος σύγκλισης (CSI, Convergence Sublayer Identification), 1bit το MSB της επικεφαλίδας, και 3-bits το πεδίο ακολουθίας μέτρησης. Τα τέσσερα λιγότερα σημαντικά προστατεύουν το παραπάνω πεδίο (Sequence Number Protection Field). Εξ' αιτίας της μεγάλης σημασίας των παραπάνω πεδίων, υποστηρίζεται μια μέθοδος ανίχνευσης πολλαπλών σφαλμάτων και διόρθωσης μονών (του ενός bit) με έλεγχο κυκλικού πλεονασμού (Cyclic Redundancy Check) και πεδία ομοτιμίας. Το σχετικό αλγόριθμο εκτελεί η μονάδα AAL HEADER_COR όταν ειδοποιηθεί με το σήμα HEADER_EN (Active High).

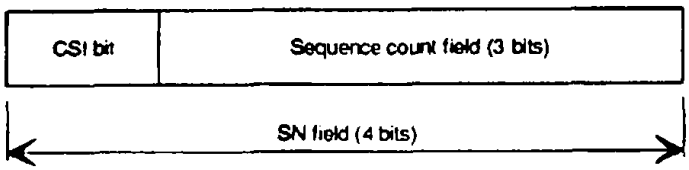
Ο δείκτης (βλέπε σχήμα 5.14), εξ' ίσου σημαντικό πεδίο, χρησιμοποιείται για να αναγνωριστεί η αρχή του πλαισίου PDH μέσα στα δεδομένα ATM. Τα 47 bytes του ωφέλιμου φορτίου SAR-PDU που χρησιμοποιούνται από το υπόστρωμα σύγκλισης (convergence sublayer, CS) έχουν δύο μορφές, non-P και P [12]. Στην μορφή P, η οποία χρησιμοποιείται στο συγκεκριμένο σύστημα, το πρώτο byte του ωφέλιμου φορτίου SAR-PDU είναι το πεδίο του δείκτη.

Περισσότερες λεπτομέρειες για την διαδικασία αυτή περιγράφονται στην μονάδα FRAME-SYNC. Τέλος, το ωφέλιμο φορτίο, που είναι οι προς μετάδοση πληροφορίες του χρήστη, αποθηκεύεται σε μια αρκετά μεγάλου μεγέθους μνήμη FIFO, της οποίας σκοπός είναι η προσαρμογή των ρυθμών μετάδοσης: "γρήγορο" ATM σε "αργό" PDH.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

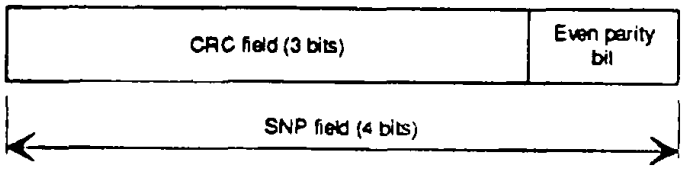
5.2.2.2 Διορθωτής επικεφαλίδας τύπου AAL-1 (AAL-1 HEADER_COR)

Η μονάδα αυτή υλοποιεί τον αλγόριθμο ανεύρεσης και διόρθωσης σφαλμάτων που περιγράφεται στη σύσταση I.363 της ITU-T [12]. Η χρήσιμη πληροφορία της επικεφαλίδας (δηλ. το πεδίο ακολουθίας αριθμού και το CSI bit), ελέγχονται με



Σχήμα 5.8 : Μορφή του πεδίου sequence number (SN)

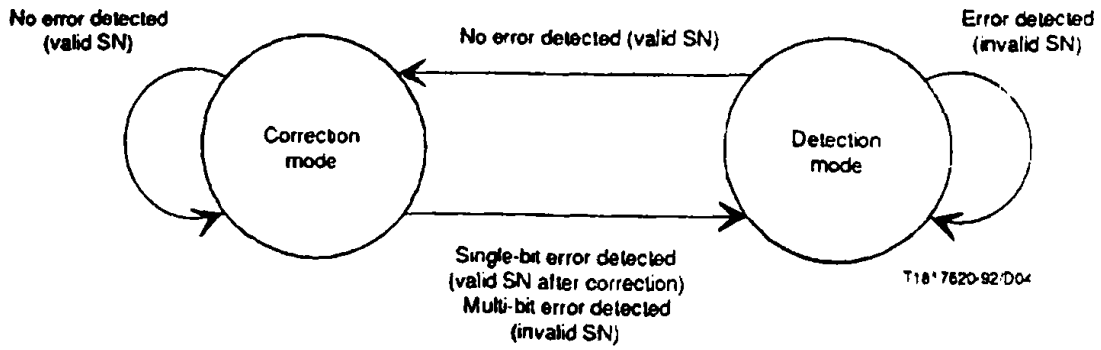
τη χρήση ενός πεδίου CRC 3 bits και ένα bit άρτιας ισοτιμίας. Το πεδίο SN χωρίζεται σε δύο επιπλέον πεδία όπως φαίνεται και στο σχήμα 5.8. Το πεδίο ακολουθιακής αρίθμησης μεταφέρει την τιμή που προβλέπεται από το υπόστρωμα σύγκλισης. Το CSI bit μεταφέρει την ένδειξη υποστρώματος σύγκλισης που προβλέπεται από το CS. Η προκαθορισμένη τιμή του CSI bit είναι '0'. Το πεδίο



Σχήμα 5.9 : Μορφή του πεδίου SNP

SNP παρέχει δυνατότητες ανίχνευσης και διόρθωσης λάθους στην επικεφαλίδα SAR-PDU [12]. Η μορφή του πεδίου παρουσιάζεται στο σχήμα 5.9. Δύο βήματα χρησιμοποιούνται για την προστασία, το πεδίο ακολουθιακής αρίθμησης ελέγχεται από 3 bits CRC, και το 7-bit αποτέλεσμα ελέγχεται από ένα bit άρτιας ομοτιμίας.

Λειτουργίες στην πλευρά του δέκτη : Ο δέκτης αναφορικά με την διόρθωση των λαθών έχει δύο διαφορετικές μεθόδους λειτουργίας, μέθοδο διόρθωσης και



Σχήμα 5.10 : SNP : receiver modes of operation

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

μέθοδο ανίχνευσης [12]. Στο σχήμα 5.10 παρουσιάζονται οι δύο μέθοδοι. Η προκαθορισμένη μέθοδος είναι η μέθοδος διόρθωσης, η οποία παρέχει διόρθωση λαθών ενός-bit. Κατά την εκκίνηση, ο δέκτης είναι τοποθετημένος στην προκαθορισμένη μέθοδο. Ο δέκτης εξετάζει καθεμιά επικεφαλίδα SAR-PDU ελέγχοντας τα CRC bits και το bit της άρτιας ομοτιμίας. Εάν στην επικεφαλίδα ανιχνευθεί λάθος, η διαδικασία που ακολουθείται εξαρτάται από την κατάσταση του δέκτη. Συγκεκριμένα, στην μέθοδο διόρθωσης μόνο λάθη ενός-bit μπορούν να διορθωθούν και ο δέκτης μετάγεται στην μέθοδο ανίχνευσης. Στην μέθοδο ανίχνευσης, όλες οι λανθασμένες επικεφαλίδες SAR-PDU δηλώνονται ότι έχουν άκυρο SN, εάν όμως μία επικεφαλίδα SAR-PDU εξετασθεί και βρεθεί να μην έχει λάθος τότε ο δέκτης μετάγεται στην μέθοδο διόρθωσης.

CRC Syndrome	*Parity*	*Action on current SN + SNP*	*Reaction for next SN + SNP*
Zero	No violation	No corrective action. Declare SN valid	Continue in correction mode
Non-zero	Violation	Single bit correction based on syndrome. Declare SN valid.	Switch to detection mode
Zero	Violation	Correct parity bit. Declare SN valid.	Switch to detection mode
Non-zero	No violation	No corrective action: multi-bit errors are uncorrectable. Declare SN invalid.	Switch to detection mode
SN SNP	Sequence Sequence	Number number protection	

Πίνακας 5.1 : Λειτουργίες κατά την μέθοδο διόρθωσης

CRC Syndrome	*Parity*	*Action on current SN + SNP*	*Reaction for next SN + SNP*
Zero	No violation	No corrective action. Declare SN valid	Swicth to correction mode
Non-zero	Violation	No corrective action. Declare SN invalid.	Continue in detection mode
Zero	Violation	No corrective action. Declare SN invalid.	Continue in detection mode
Non-zero	No violation	No corrective action. Declare SN invalid.	Continue in detection mode
SN SNP	Sequence Sequence	Number number protection	

Πίνακας 5.2 : Λειτουργίες κατά την μέθοδο ανίχνευσης

Οι πίνακες 5.1 και 5.2 δείχνουν τις λεπτομερείς λειτουργίες για την μέθοδο διόρθωσης και για την μέθοδο ανίχνευσης, αντίστοιχα. Η λειτουργία βασίζεται πάνω στην συνδυασμένη εγκυρότητα του CRC και της ομοτιμίας. Από τα παραπάνω γίνεται φανερό ότι το σήμα HEADER_COR - Active High σημαίνει ότι

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

η επικεφαλίδα έχει ληφθεί σωστά - ενεργοποιείται τόσο σε περίπτωση ορθής μετάδοσης όσο και σε περίπτωση λάθους του ενός bit. Η έξοδος της μονάδας αυτής μας δίνει το SN_FIELD (3 bits), το CSI_BIT, μαζί με το απαραίτητο σήμα (HEADER_COR).

5.2.2.3 Ελεγκτής ακολουθιακής αρίθμησης (SN_CHECK)

Η μονάδα αυτή εκτελεί τον επιβεβλημένο έλεγχο συνέχειας της ακολουθίας αριθμών που βρίσκονται στις επικεφαλίδες AAL [12]. Είναι προφανές, ότι οι αντίστοιχοι αριθμοί θα πρέπει να είναι διαδοχικοί, διαφορετικά συμπεραίνεται ότι κατά την διάρκεια της μετάδοσης είτε έχουν χαθεί κάποια πακέτα είτε ότι κάποια άλλα έχουν παρεμβληθεί.

Στην πρώτη περίπτωση, είναι απαραίτητο να συμπληρθούν κατά κάποιο τρόπο τα δεδομένα που λείπουν, στέλνοντας αδρανή bytes προς τις διεπαφές PDH. Βέβαια, εξαιτίας του γεγονότος αυτού είναι πιθανό να χαθεί κάποιο CSI bit , που αποτελεί πληροφορία σημαντική για την ανάκτηση του ρολογιού PDHCLK. Για το σκοπό αυτό, παράγεται το σήμα SEQ_CHECK λογικής κατάστασης μηδέν που μας ειδοποιεί ότι η ακολουθία των πακέτων είναι λανθασμένη. Στην δεύτερη περίπτωση (παρεμβολή επιπλέον SDU's), τα πράγματα είναι πιο απλά, καθώς η μόνη ενέργεια που απαιτείται είναι η απόρριψη των πακέτων που έχουν ληφθεί περισσότερες από μία φορές. Στη περίπτωση αυτή, παράγεται το σήμα REJ_CELL λογικής κατάστασης ένα για την απόρριψη των πακέτων. Τέλος, μια επιπλέον δυνατότητα της μονάδας αυτής , είναι η παραγωγή του σήματος ODD_CHECK λογικής κατάστασης ένα, το οποίο δηλώνει ότι έχει ληφθεί πακέτο με περιττό ακολουθιακό αριθμό, γεγονός που διευκολύνει την εξαγωγή του αποτυπώματος υπολοίπου χρόνου RTS από το CSI bit.

5.2.2.4 Μονάδα ανάκτησης ρολογιού PDHCLK (PDHCLK RECOVERY)

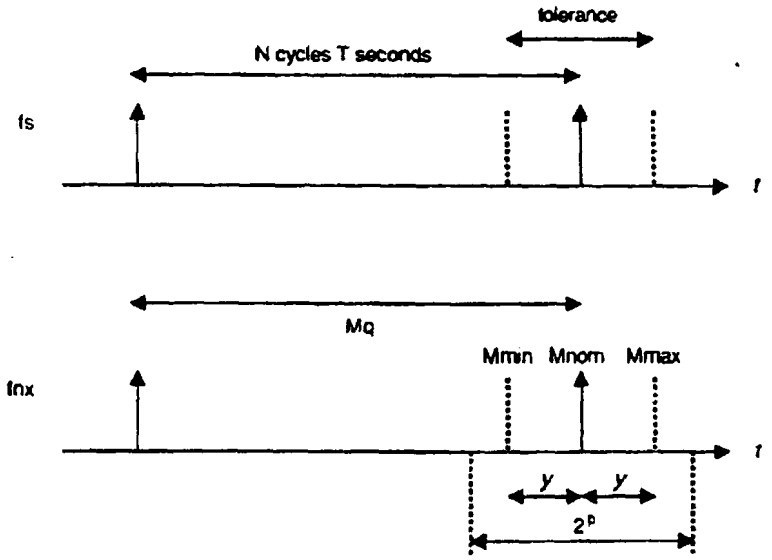
Η μονάδα αυτή δημιουργεί ξανά το ρολόϊ υπηρεσίας PDHCLK με βάση τα πεδία RTS που έρχονται στις επικεφαλίδες AAL και το ρολόι του δικτύου SDH ATMCLK σαν σήμα αναφοράς. Η μέθοδος του σύγχρονου αποτυπώματος υπολοίπου χρόνου SRTS χρησιμοποιεί το αποτύπωμα υπολοίπου χρόνου RTS για να μετρήσει και να μεταφέρει πληροφορία για την διαφορά συχνότητας ανάμεσα στο αντλούμενο από το ρολόϊ δικτύου και στο ρολόϊ υπηρεσίας [12]. Το αντλούμενο ρολόϊ δικτύου θεωρείται ότι είναι διαθέσιμο και στον εκπομπό και στον δέκτη. Η αρχή λειτουργίας του SRTS παρουσιάζεται στο σχήμα 5.11. Η ακόλουθη περιγραφή της μεθόδου SRTS ακολουθεί την πρακάτω σημειογραφία :

- fs: συχνότητα ρολογιού υπηρεσίας
- fn: συχνότητα ρολογιού δικτύου, π.χ. 155,52 MHz
- fnx: αντλούμενη συχνότητα ρολογιού δικτύου, fnx=fn/x, όπου x είναι ένας ακέραιος αριθμός
- N: περίοδος του RTS σε κύκλους της συχνότητας fs του ρολογιού υπηρεσίας

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

- T: περίοδος του RTS σε δευτερόλεπτα
- M Mnom/Mmax/Mmin: αριθμός των fnx κύκλων μέσα σε μια κανονική/ μέγιστη/ελάχιστη περίοδο RTS
- Mq: μεγάλος ακέραιος μικρότερος ή ίσος με το M

Μια σταθερή χρονική διάρκεια ίση με μια περίοδο T μετρούμενη από N



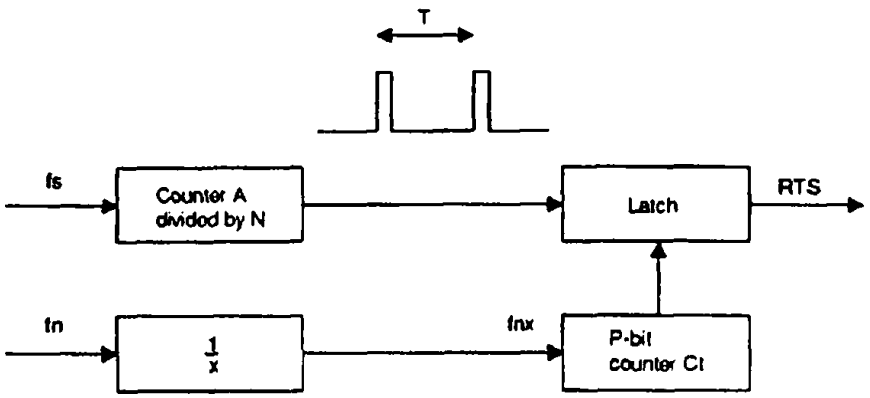
Σχήμα 5.11 : Η αρχή του σύγχρονου αποτυπώματος υπολοίπου χρόνου (SRTS)

κύκλους του ρολογιού υπηρεσίας, αντιστοιχεί σε κάποιο αριθμό αντλούμενων από το ρολόϊ δικτύου κύκλων Mq. Εάν το Mq μεταδοθεί στον δέκτη, το πηγαίο ρολόϊ υπηρεσίας μπορεί να ανακατασκευαστεί από τον δέκτη, δεδομένου ότι έχει την απαραίτητη πληροφορία : fnx, Mq και N. Όμως, το Mq καθορίζεται από ένα κανονικό μέρος και από ένα υπόλοιπο μέρος. Το κανονικό μέρος Mnom αντιστοιχεί στον ονομαστικό αριθμό των fnx κύκλων σε T δευτερόλεπτα και είναι σταθερό για την υπηρεσία. Το υπόλοιπο μέρος μεταβιβάζει την πληροφορία της διαφοράς συχνότητας. Μιας και το κανονικό μέρος είναι σταθερό, μπορεί να θεωρηθεί ότι είναι διαθέσιμο στον δέκτη. Μόνο το υπόλοιπο μέρος του Mq μεταδίδεται στον δέκτη.

Ένας απλός τρόπος αναπαράστασης του υπολοίπου μέρους του Mq είναι μέσω της έννοιας του RTS, του οποίου η παραγωγή φαίνεται στο σχήμα 5.12. Ο μετρητής Ct έιναι ένας μετρητής P-bit, ο οποίος είναι συνεχώς ενεργοποιημένος με το αντλούμενο ρολόϊ του δικτύου. Η έξοδος του μετρητή Ct δειγματοληπτείται κάθε N κύκλους του ρολογιού υπηρεσίας. Το δείγμα P-bit είναι το αποτύπωμα υπολοίπου χρόνου, RTS. Γνωρίζοντας το RTS και το κανονικό μέρος του Mq, το Mq είναι προσδιορισμένο πλήρως. Το Mq χρησιμοποιείται για να παράγει το σήμα χρονισμού αναφοράς ώστε να δημιουργήσει το ρολόϊ υπηρεσίας. Η όλη αρχιτεκτονική της μονάδας αυτής είναι αρκετά περίπλοκη, αλλά μπορεί να υλοποιηθεί σ' ένα ψηφιακό PLL, με τη διαφορά ότι εδώ δεν υπάρχει συγκριτής

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

φάσης, αλλά συγκριτής RTS. Συγκεκριμένα, το πεδίο RTS υπολογίζεται στο δέκτη όπως και στον εκπομπό (μέσω των ATMCLK και PDHCLK), συγκρίνεται με αυτό που αποστέλλεται από τον εκπομπό μέσω του CSI bit και στη συνέχεια τα δύο αυτά πεδία αφαιρούνται μεταξύ τους. Η διαφορά αυτή, που ονομάζεται και



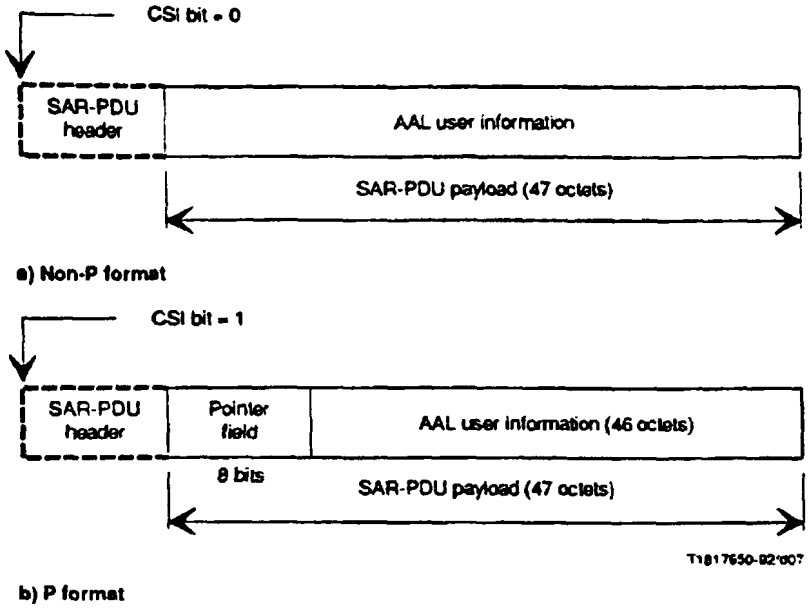
Σχήμα 5.12 : Παραγωγή του αποτυπώματος υπολοίπου χρόνου (RTS)

jitter, είναι η έξοδος (VCO_CONTR) της μονάδας PDHCLK RECOVERY. Στη συνέχεια, η διαφορά οδηγείται σ'έναν D/A converter και ύστερα σε ένα βαθυπερατό φίλτρο, έτσι ώστε να παραχθεί μια αργά μεταβαλλόμενη τάση ελέγχου που θα ρυθμίζει την ολίσθηση συχνότητας του ταλαντωτή ελεγχόμενο από τάση (Voltage Controlled Oscillator,VCO). Είναι προφανές ότι πρόκειται για ένα βρόχο κλειδώματος φάσης (PLL) ο οποίος θα χρειαστεί κάποιο χρόνο για να σταθεροποιηθεί και κατά συνέπεια, μέχρι εκείνη τη στιγμή οτιδήποτε δεδομένα PDH μεταδίδονται από τον πομπό θα χαθούν.

5.2.2.5 Μονάδα συγχρονισμού του πλαισίου PDH (FRAME_SYNC)

Εδώ παράγεται ο παλμός συγχρονισμού πλαισίου SYNC_PULSE που απαιτεί η διεπαφή PDH για να λειτουργήσει σωστά με βάση την πληροφορία του δείκτη που λαμβάνεται από τη βαθμίδα SAR_PDU DISASSEMBLER. Το μέγεθος του δείκτη είναι 8 bits και το περιεχόμενό του αντιπροσωπεύει τον αριθμό των bytes που απομένουν μέχρι να ξεκινήσει ένα καινούριο πλαίσιο PDH. Όπως αναφέρθηκε παραπάνω, οι 47 οκτάδες του ωφέλιμου φορτίου SAR-PDU που χρησιμοποιούνται από το υπόστρωμα σύγκλισης CS έχουν δύο μορφές, non-P και P, όπως φαίνεται στο σχήμα 5.13. Στην μορφή P, το πρώτο byte του ωφέλιμου φορτίου SAR-PDU είναι το πεδίο του δείκτη. Το υπόλοιπο πεδίο περιέχει πληροφορία του χρήστη. Η μορφή P χρησιμοποιείται, μόνο εάν η τιμή του ακολουθιακού αριθμού στην επικεφαλίδα SAR-PDU είναι 0,2,4, ή 6. Επομένως χρησιμοποιείται ανά δύο πακέτα. Στο δεύτερο πακέτο το συγκεκριμένο πεδίο χρησιμοποιείται για δεδομένα του χρήστη. Η μορφή του πεδίου του δείκτη φαίνεται στο σχήμα 5.14. Το πεδίο του δείκτη περιέχει τον αριθμό των υπολοίπων bytes ανάμεσα από το τέλος του πεδίου του δείκτη και του τελευταίου byte δεδομένων του επόμενου πακέτου. Η τιμή του πεδίου του δείκτη εκτείνεται από 0

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

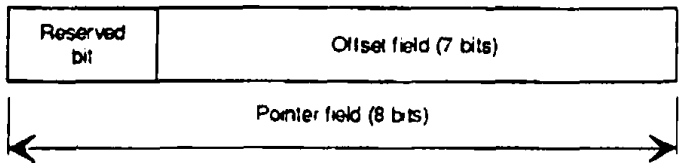


Σχήμα 5.13: Η μορφή του SAR-PDU στη μέθοδο μεταφοράς δομημένων δεδομένων

μέχρι και 92. Στην περίπτωση του PDH των 2 Mb/s (σύστημα PCM 1ης τάξης), το μέγεθος του πλαισίου είναι 32 bytes, άρα από τα 8 bits του πεδίου αυτού στη συγκεκριμένη εφαρμογή χρησιμοποιούνται μόνο τα 5. Στη γενική περίπτωση, ο αριθμός που περιέχει όπως αναφέρθηκε, κυμαίνεται από 0 μέχρι 92, ενώ το 93 συναντάται μόνο σε συστήματα PCM ανώτερης τάξης (π.χ. 34 Mbps), στα οποία το πλαίσιο έχει περισσότερα bytes από όσα θα μπορούσε να απεικονίσει ο οκτάμπιτος δείκτης.

Η βαθμίδα FRAME _SYNC διαβάζει δεδομένα με το ATMCLK και παράγει έξοδο SYNC_PULSE που είναι συγχρονισμένη με το PDHCLK. Επειδή το μέγεθος του ATM_SDU είναι 48 bytes, είναι πιθανό δύο σήματα SYNC_PULSE να πρέπει να παραχθούν στη διάρκεια μετάδοσης ενός πακέτου, γι' αυτό και γίνεται χρήση απαριθμητή. Ο απαριθμητής μετράει συνεχώς από 1 μέχρι 32, όσα είναι δηλαδή και τα bytes που αποτελούν το πλαίσιο PDH, καθορίζοντας έτσι τα όρια του πλαισίου στο ωφέλιμο φορτίο του κάθε πακέτου. Συγχρονίζεται δε σε κάθε άρτιο πακέτο, με την τιμή του δείκτη που βρίσκεται σε αυτό. Αν το σήμα HEAD_COR είναι '0' (δηλαδή η επικεφαλίδα δεν έχει ληφθεί σωστά) για το πακέτο στο οποίο αντιστοιχεί ο δείκτης εισόδου τότε η τιμή του δείκτη δεν λαμβάνεται υπόψη στο συγχρονισμό του SYNC_PULSE. Αν τα σήματα SEQ_CHECK και ODD_CHECK είναι συγχρόνως μηδέν, δηλαδή το πακέτο στο οποίο αντιστοιχεί ο δείκτης δεν έχει ληφθεί σωστά, τότε η τιμή του δείκτη επίσης δεν λαμβάνεται υπόψη στο συγχρονισμό του SYNC_PULSE.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 5.14 : Μορφή του πεδίου του δείκτη

5.2.2.6 Δημιουργία πεδίου RTS (RTS CREATION)

Η μονάδα αυτή έχει σαν αποστολή την επαναδημιουργία του πεδίου RTS (Residual Time Stamp), το οποίο είχε κατασκευαστεί στον πομπό και έχει διασπαρεί στα AAL SDUs με περιττό ακολουθιακό αριθμό υπό την μορφή των CSI bits [11]. Διαβάζοντας λοιπόν την τιμή του έβδομου bit των επικεφαλίδων με SN = 1, 3, 5 ή 7 επαναδημιουργείται το RTS - το bit που αντιστοιχεί σε SN =1 είναι το περισσότερο σημαντικό - ο ρόλος του οποίου είναι να συμβάλλει στην ανάκτηση του ρολογιού υπηρεσίας του πομπού. Στην παρούσα περίπτωση το ρολόι υπηρεσίας είναι το PDHCLK. Για να βεβαιωθούμε ότι τα CSI bits που λαμβάνονται ανήκουν στο ίδιο RTS, εξετάζεται το σήμα SEQ_CHECK, το οποίο ελέγχει αν οι ακολουθιακοί αριθμοί των SDUs που έρχονται είναι όντως διαδοχικά. Σε περίπτωση που δεν συμβαίνει αυτό, το RTS δεν υπολογίζεται, εκτός αν το σφάλμα προκύπτει από επανάληψη αποστολής πακέτου που έχει ήδη σταλεί.

Αν το σήμα HEAD_COR έχει την τιμή μηδέν, δηλαδή η επικεφαλίδα δεν έχει ληφθεί σωστά, τότε πάλι δεν υπολογίζεται το RTS.

5.2.2.7 Ταλαντωντής ελεγχόμενος από τάση (VOLTAGE CONTROLLED OSCILLATOR)

Πρόκειται για ένα κλασσικό ταλαντωτή ελεγχόμενο από τάση ο οποίος, όταν λειτουργεί ελεύθερα παράγει ένα σήμα τετραγωνικού παλμού συχνότητας 2 MHz περίπου. Η έξοδος αυτή θα πρέπει να συγχρονιστεί με το αντίστοιχο PDHCLK του εκπομπού. Για το σκοπό αυτό χρησιμοποιείται το αναλογικό σήμα VCO_CONTR, το οποίο ρυθμίζει τη συχνότητα ταλάντωσης του ταλαντωτή, ώστε η διαφορά φάσης με την αντίστοιχη του πομπού να είναι σταθερή. Όταν συμβεί αυτό δεν θα έχουμε πλέον ολίσθηση συχνότητας - η διάταξη θα έχει κλειδώσει - και η έξοδος του VCO θα μπορεί να χρησιμοποιηθεί σαν τοπικό ρολόι PDHCLK για τις υπόλοιπες μονάδες του συστήματος.

Η βαθμίδα VCO_CONTR δεν είναι ενσωματωμένη στα FPGAs, αλλά έχει τοποθετηθεί στην πλακέτα εξωτερικά από αυτά, καθώς, όπως αναφέρθηκε και προηγουμένως, πρόκειται για αναλογική μονάδα.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.2.2.8 Εισαγωγή αδρανών bytes (IDLE BYTE INSERTION)

Υπάρχουν ορισμένες περιπτώσεις που στις γραμμές PDH δεν θα πρέπει να στέλνονται τα δεδομένα που ελήφθησαν από το SDH δίκτυο. Για παράδειγμα, αν συμβεί απώλεια πακέτων ATM κατά τη μετάδοση ή σε ορισμένα από τα πακέτα που έφτασαν στον δέκτη η επικεφαλίδα τους δεν έχει ληφθεί σωστά, είναι σημαντικό να αποστέλλεται κάτι προς τα κανάλια PDH έτσι ώστε να κρατηθεί σταθερός ο ρυθμός μετάδοσης. Το ίδιο θα πρέπει να συμβαίνει και στην περίπτωση που είτε για κάποιο λόγο έχει διακοπεί η λειτουργία του πομπού, είτε τα σήματα εισόδου PDH είναι ανενεργά. Για όλους τους παραπάνω λόγους είναι αναγκαία η λειτουργία της μονάδας εισαγωγής αδρανών δεδομένων. Τα δεδομένα αυτά τοποθετούνται από τον μικροελεγκτή μέσω των διαύλων διευθύνσεων και δεδομένων ADDR_BUS και DATA_BUS αντίστοιχα και διαβάζονται από τον πολυπλέκτη της FIFO. Είναι σημαντικό να τονισθεί ότι ο χρήστης του υπολογιστικού συστήματος θα μπορεί μέσω του μικροελεγκτή να διαβάζει αλλά και να τροποποιεί τα περιεχόμενα της βαθμίδας IDLE BYTE INSERTION μέσω των επιλογών RD και WR αντίστοιχα. Επιπλέον, και στις διεπαφές PDH έχει προβλεφθεί η δυνατότητα εισαγωγής αδρανών δεδομένων FF. Η μονάδα IDLE BYTE INSERTION είναι ιδιαίτερα χρήσιμη καθώς οι πληροφορίες που μπορεί να αποθηκεύσει δεν είναι μόνο απλά FF αλλά είναι δυνατόν να βελτιστοποιηθεί το σύστημα αξιοποιώντας παλλαπλώς τα αδρανή bytes, π.χ. για να ειδοποιηθεί ένα απομακρυσμένο σύστημα στην περίπτωση βλάβης του δέκτη.

5.2.2.9 Πολυπλέκτης της FIFO (FIFO MULTIPLEXER)

Η βαθμίδα αυτή παρέχει τα απαιτούμενα σήματα για εγγραφή στη FIFO. Ανάλογα με την τιμή του σήματος ενεργοποίησης της FIFO τα δεδομένα που γράφονται στη FIFO (byte προς byte) σε κάθε παλμό του παράλληλου ρολογιού ATMCLK προέρχονται από την είσοδο PAYLOAD. Η είσοδος PAYLOAD παίρνει τιμές είτε από την μονάδα SAR_PDU DISASSEMBLER είτε από την είσοδο DUM_DATA. Το σήμα ενεργοποίησης της FIFO εξαρτάται από τα σήματα HEAD_COR και SEQ_CHECK.

Όταν το σήμα HEAD_COR έχει τη τιμή 1, δηλαδή η επικεφαλίδα του πακέτου ATM έχει βρεθεί να είναι σωστή, γράφονται στη FIFO τα δεδομένα της εισόδου PAYLOAD. Όταν το σήμα HEAD_COR έχει τη τιμή 0, δηλαδή η επικεφαλίδα του πακέτου ATM δεν έχει βρεθεί να είναι σωστή, γράφονται στη FIFO τα δεδομένα της εισόδου DUM_DATA. Όταν το σήμα SEQ_CHECK έχει τη τιμή 1, δηλαδή ο ακολουθιακός αριθμός του πακέτου ATM έχει βρεθεί να είναι σωστός, γράφονται στη FIFO τα δεδομένα της εισόδου PAYLOAD. Όταν το σήμα SEQ_CHECK έχει τη τιμή 0, δηλαδή ο ακολουθιακός αριθμός του πακέτου ATM δεν έχει βρεθεί να είναι σωστός, εν γράφεται στη FIFO, γράφονται στη FIFO τα δεδομένα της εισόδου DUM_DATA.

Επειδή η FIFO έχει εκλεγεί να είναι αρκετά μεγάλη, δεν λαμβάνεται φροντίδα για περιπτώσεις υπερχείλισης.

5.2.2.10 Μονάδα εξόδου (OUTPUT UNIT)

Η μονάδα αυτή παίρνει δεδομένα από τη FIFO και τα οδηγεί την διεπαφή PDH. Η ανάγνωση και η εκπομπή γίνεται με βάση το ρολόι PDHCLK που έχει παραχθεί ήδη από τη βαθμίδα VCO και τον παλμό συγχρονισμού SYNC_PULSE. Κάθε φορά που στέλνεται ένας σήμα SYNC_PULSE από τη βαθμίδα FRAME_SYNC ενεργοποιείται το σήμα RD για διάβασμα από τη FIFO. Η ενεργοποίηση παραμένει όσο το FRAME_SYNC είναι ενεργοποιημένο. Με τον τρόπο αυτό συγχρονίζονται τα πακέτα του καναλιού PDH πριν εισαχθούν στην διεπαφή PDH.

5.2.2.11 Διεπαφές PDH (PDH INTERFACES)

Οι διεπαφές εκπομπής των γραμμών PDH είναι ακριβώς οι ίδιες με τις αντίστοιχες του εκπομπού. Πρόκειται για τα δύο ολοκληρωμένα PEB2235 (ISDN Primary Access Tranceiver) [13] και PEB2035 (Advanced CMOS Frame Aligner, ACFA) [14], τα οποία αναλαμβάνουν να εκτελέσουν όλες τις λειτουργίες της διεπαφής : πλαισίωση των δεδομένων, εισαγωγή πληροφοριών σηματοδοσίας, αποσύζευξη ρυθμού μεταφοράς, προστασία έναντι λαθών με μια μέθοδο CRC 4 bits, δημιουργία συναγερμών σε περίπτωση απώλειας συγχρονισμού, προσαρμογή στη γραμμή μεταφοράς κλπ. Το πρώτο από αυτά πραγματοποιεί τις κατωτέρου επιπέδου (clock synthesis/recovery, line termination) λειτουργίες ενώ το δεύτερο λειτουργεί σε ανώτερο επίπεδο (Frame Alignment/Synthesis, CRC 4 bits generation/checking, signalling, alarm indications, etc.). Η σύνδεση των δύο ICs γίνεται μέσω ενός διαύλου έξι γραμμών, όπως μπορεί να φανεί και στα σχετικά κυκλώματα των τεχνικών χαρακτηριστικών [13], [14].

Λόγω της εσωτερικής FIFO (64 words x 9) του PEB2035, μπορούμε να αντιμετωπίσουμε κατά κάποιο τρόπο τυχόν μικροδιαφορές στην ταχύτητα των γραμμών PDH. Το σήμα SYNC_PULSE αποτελεί είσοδο για τις διεπαφές αυτές και δίνεται ταυτόχρονα και στις δύο. Έτσι αποφεύγονται πιθανά προβλήματα συγχρονισμού των ροών PDH. Βεβαίως δεν μπορεί να αποκλειστεί το ενδεχόμενο κάποια από τις εσωτερικές FIFO να ξεχειλίσει από δεδομένα, οπότε ένα σήμα προειδοποίησης (AINT) δημιουργείται από το ACFA προκαλώντας κάποιο σήμα διακοπής στον μικροελεγκτή του συστήματος. Τόσο το PEB2035 όσο και το PEB2235 μπορούν να δράσουν ταυτόχρονα σαν εκπομπός και σαν δέκτης για τις γραμμές PDH. Στη συγκεκριμένη περίπτωση χρησιμοποιείται μόνο η μία κατεύθυνση. Τα δεδομένα εκπέμπονται στο δίκτυο PCM ακριβώς με την ίδια μορφή που είχαν ληφθεί στο αντίστοιχο κανάλι του πομπού, απαλλαγμένα δηλαδή από επικεφαλίδες, πληροφορίες αρίθμησης κλπ. και γενικά οτιδήποτε επιπλέον είχε εισαχθεί εκεί προκειμένου να καταστήσει δυνατή την μετάδοση ATM/SDH.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

5.2.2.12 Διεπαφή μικροεπεξεργαστή (MICROPROCESSOR INTERFACE)

Η διεπαφή αυτή επιτρέπει σε ένα μικροελεγκτή να ελέγχει το όλο σύστημα. Το απαιτεί η λειτουργία τόσο του δέκτη ATM όσο και των διεπαφών PDH (ιδίως του PEB2035).

Στην περίπτωση, ο μικροελεγκτής αποθηκεύει στις μνήμες RAM του δέκτη κάποιες τιμές οι οποίες κάνουν την λογική αντιστοίχιση ανάμεσα στα πεδία VCI/VPI και σε κάθε μία από τις τελικές κατευθύνσεις που υποστηρίζονται από τον δέκτη.

Στη περίπτωση των διεπαφών PDH, γίνεται η αρχικοποίηση των εσωτερικών καταχωρητών ελέγχου των διεπαφών PDH που ρυθμίζουν διάφορες παραμέτρους συμπεριφοράς τνη, όπως για παράδειγμα την ταχύτητα μετάδοσης, τη σηματοδοσία, την παραγωγή συναγερμών σε περίπτωση σφάλματος και τη μέθοδο κωδικοποίησης. Ο μικροελεγκτής δεν βρίσκεται κατ' ανάγκη πάνω στη πλακέτα του δέκτη. Προτείνεται το interface να κατασκευαστεί σε μορφή κάρτας επέκτασης για PC, οπότε ο μικροελεγκτής του τελευταίου θα ελέγχει και τις βασικές λειτουργίες του συστήματος.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

6. Προσομοίωση λειτουργίας του δέκτη ATM/PDH

Ο λογικός σχεδιασμός (functional simulation) του ψηφιακού τμήματος του δέκτη έγινε με τη βοήθεια του λογισμικού ViewSynthesis της Viewlogic. Το σχεδιαστικό πακέτο χρησιμοποιεί ένα πλήρως γραφικό περιβάλλον μέσα από Windows 3.1/95, ενώ μπορεί να δεχθεί και εντολές απ' ευθείας από το DOS. Αρχικά, το πακέτο λογισμικού παίρνει σαν είσοδο κώδικα VHDL, ο οποίος μπορεί να είναι είτε τελείως αυτόνομος, είτε να καλεί άλλα υποπρογράμματα VHDL που είναι συνδεδεμένα με αυτόν. Στη συνέχεια, επιλέγεται η οικογένεια FPGAs μέσα στην οποία θα ενσωματωθεί η σχεδίαση. Στην περίπτωση της παρούσας εργασίας έχει επιλεχθεί η οικογένεια XC4000, εξ' αιτίας της μνήμης RAM που περιλαμβάνει και η οποία είναι απαραίτητη για την υλοποίηση του συστήματος.

6.1 *Βήματα κατά το σχεδιασμό με VHDL*

Το σχεδιαστικό πακέτο κάνει τη μεταγλώτιση και παράγει αρχεία (λειτουργικής) προσομοίωσης. Συνεχίζοντας, γίνεται η σύνθεση της σχεδίασης. Το αποτέλεσμα της ενέργειας αυτής είναι η δημιουργία ενός σχηματικού διαγράμματος που αντιστοιχεί στον κώδικα VHDL που αρχικά εισήχθη στο σύστημα. Επιπλέον, παράγονται καινούργια αρχεία προσομοίωσης (timing simulation), τα οποία στηρίζονται στο προκύπτον κύκλωμα. Προφανώς, τα αποτελέσματα από τις δυο προσομοιώσεις θα πρέπει να συμπίπτουν, αλλοιώς κάποιο λάθος έχει συμβεί κατά τη διαδικασία της σύνθεσης. Εάν τα αποτελέσματα από τις δύο προσομοιώσεις είναι ίδια, ακολουθεί η υλοποίηση της VHDL στο FPGA. Ο χρήστης μπορεί αν επιθυμεί να επέμβει απ' ευθείας στις εσωτερικές δομές του FPGA (με χρήση του editor XACT) και να διορθώσει "με το χέρι" κρίσιμα τμήματα, για τα οποία επιβάλλεται να υπάρχει η μικρότερη δυνατή καθυστέρηση. Η απαραίτητη προσομοίωση χρονισμού που γίνεται στο σημείο αυτό, συμπληρώνει τη φάση της σχεδίασης και υλοποίησης. Το μόνο που μένει πλέον είναι να μεταφερθούν οι πληροφορίες αρχιτεκτονικής μέσα στο FPGA. Τα FPGA είναι προγραμματίσιμα παραπάνω από μία φορές. Αυτό μάς δίνει τη δυνατότητα κατά τη φάση του ελέγχου του σχεδίου να δημιουργήσουμε πολλές εκδοχές της αρχικής σχεδίασης και να χρησιμοποιήσουμε τελικά εκείνη που παράγει τα καλύτερα και πιο αξιόπιστα αποτελέσματα.

6.2 *Το πρόγραμμα υλοποίησης του δέκτη ATM/PDH σε VHDL*

Ο κλασσικός τρόπος σχεδίασης γίνεται με βάση μια ιεραρχία. Η υλοποίηση ενός συστήματος αρχίζει από τα επιμέρους τμήματα, τα οποία συνήθως σχεδιάζονται σε επίπεδο πύλης. Στη συνέχεια, μερικά από αυτά αποτελούν ένα τμήμα ανωτέρου επιπέδου και ούτω καθ' εξής. Η υλοποίηση του σχεδίου φτάνει σ' ένα σημείο, όπου το περιγραφόμενο σύστημα αποτελείται από δύο ή τρία τμήματα ανωτάτου επιπέδου. Το πλεονέκτημα αυτού του τρόπου σχεδίασης είναι ότι πιθανόν εσωτερικές αλλαγές στα κατώτερα επίπεδα - τα επιμέρους τμήματα - δεν απαιτούν και αλλαγές στα ανώτερα επίπεδα.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

Εν τούτοις, η γλώσσα περιγραφής υλικού VHDL παρέχει την δυνατότητα στο χρήστη να υλοποιήσει το σύστημά του ιεραρχικά ή σαν ένα ενιαίο σύστημα. Επομένως, η VHDL έχει το πλεονέκτημα του κλασσικού τρόπου σχεδίασης αλλά επιπλέον δίνει την δυνατότητα το σύστημα να αντιμετωπιστεί σαν ένα μαύρο κουτί που εκτελεί μια διαδικασία (procedure) με εισόδους και εξόδους. Διαθέτει δηλαδή τις τεχνικές εκείνες μιας γλώσσας προγραμματισμού (functions, procedures, variables). Χαρακτηριστικό παράδειγμα υλοποίησης της δεύτερης μεθόδου είναι η περιγραφή ενός κυκλώματος ακολουθίας καταστάσεων. Στη περίπτωση του κλασσικού τρόπου σχεδίασης, η περιγραφή του κυκλώματος είναι χρονοβόρα και επίπονη.

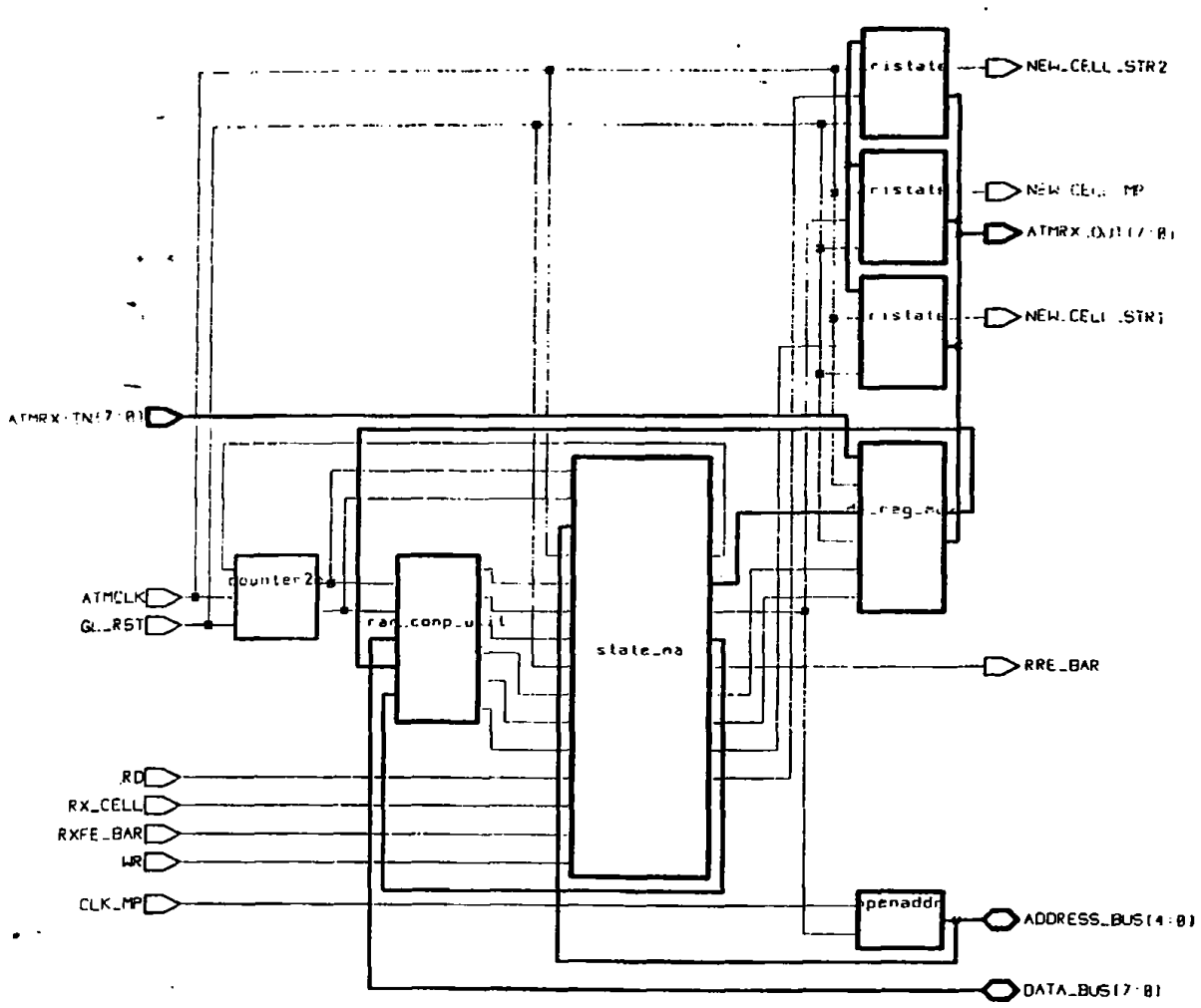
Το πρόγραμμα υλοποίησης του δέκτη ATM/PDH σε VHDL παρατίθεται στο παράρτημα Α. Αποτελείται από το πρόγραμμα υλοποίησης της μονάδας ATMRX και από δύο ίδια προγράμματα υλοποίησης της μονάδας AALRX, που αντιστοιχούν στις δύο ροές χρήστη. Στα σχήματα 6.1 και 6.2 παρατίθενται τα υλοποιημένα διαγράμματα του ATMRX και AALRX, αντίστοιχα. Όπως φαίνεται στα σχήματα, η κάθε μονάδα αποτελείται από επιμέρους υπομονάδες.

Η μονάδα ATMRX, όπως αναλύθηκε στο προηγούμενο κεφάλαιο αποτελείται από την υπομονάδα αυτόματης αναγνώρισης της επικεφαλίδας, την υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων και την υπομονάδα ακολουθίας καταστάσεων. Η υπομονάδα αναγνώρισης της επικεφαλίδας αποτελείται από το τμήμα de_reg_mux, όπως φαίνεται στο σχήμα 6.1. Το τμήμα αυτό χωρίζεται σε επιμέρους τμήματα: ένα demux1_2, οκτώ reg8bit και ένα mux2_1 (παράρτημα Α). Ο demux1-2 είναι ένας αποπολυπλέκτης ενός byte σε δύο και διαχωρίζει την επικεφαλίδα του πακέτου ATM (5 bytes) από τη χρήσιμη πληροφορία. Ο reg8bit είναι ένας καταχωρητής 8bits και ο mux2_1 είναι ένας πολυπλέκτης δύο byte σε ένα. Οι τρεις καταχωρητές μαζί με τον πολυπλέκτη χρησιμοποιούνται για να οδηγούν την επικεφαλίδα προς την υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων, ενώ οι υπόλοιποι καταχωρητές διατηρούν τη χρήσιμη πληροφορία όσο διαρκέσει η αναγνώριση του εισερχόμενου πακέτου.

Η υπομονάδα αποθήκευσης και αναγνώρισης των επικεφαλίδων αποτελείται από τα τμήματα ram_comp_unit και το counter2b, σχήμα 6.1. Το τμήμα ram_comp_unit αποτελείται από 14 μνήμες 4 διεθύνσεων των 8bits, ram4x8 και ένα συγκριτή, comp8bit. Το τμήμα couter2b είναι ένας μετρητής 2bits. Με κάθε παλμό ρολογιού ATMCLK μέσω του μετρητή επιλέγεται η εκάστοτε διεύθυνση των 14 μνημών, ώστε να συγκριθεί με το αντίστοιχο byte της επικεφαλίδας του εισερχόμενου πακέτου ATM.

Η υπομονάδα ακολουθίας καταστάσεων αποτελείται από τα τμήματα state_ma, mpenaddr, tristate. Εκτελεί τις λειτουργίες του στρώματος ATM, φορτώνει τις επικεφαλίδες που θα υποστηρίζει στις μνήμες μέσω του τμήματος mpenaddr και οδηγεί τη χρήσιμη πληροφορία προς την αντίστοιχη μονάδα AALRX για κάθε ροή χρήστη μέσω των τμημάτων tristate.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 6.1: Το υλοποιημένο διάγραμμα της μονάδας ATMRX με VHDL

Η μονάδα AALRX είναι ίδια για τις δύο ροές χρήστη. Αποτελείται από το τμήματα sar_pdu, aal1_head_correct, sn_check, rts_crte, idle_byte, fifo_mux, pdhclk_rec, fram_entity, out_unit, όπως φαίνονται στο σχήμα 6.2.

- Το sar_pdu διαχωρίζει το ATM_SDU σε επικεφαλίδα, δείκτη και δεδομένα.

- Το aal1_head_correct εκτελεί τον αλγόριθμο ανίχνευσης και διόρθωσης σφαλμάτων στην επικεφαλίδα του ATM_SDU και αποτελείται από τα επιμέρους τμήματα:
  - read_header, το οποίο αποθηκεύει προσωρινά την επικεφαλίδα
  - crc_gen που επαναδημιουργεί το πεδίο CRC
  - crc_compar, το οποίο συγκρίνει το εισερχόμενο πεδίο CRC με το παραγόμενο
  - corct_detect_mode που εκτελεί τη μέθοδο ανίχνευσης και διόρθωσης λάθους

- Το sn_check ανιχνεύει την ακολουθία αριθμών της επικεφαλίδας του ATM_SDU.

- Το rts_crte επαναδημιουργεί το πεδίο RTS.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

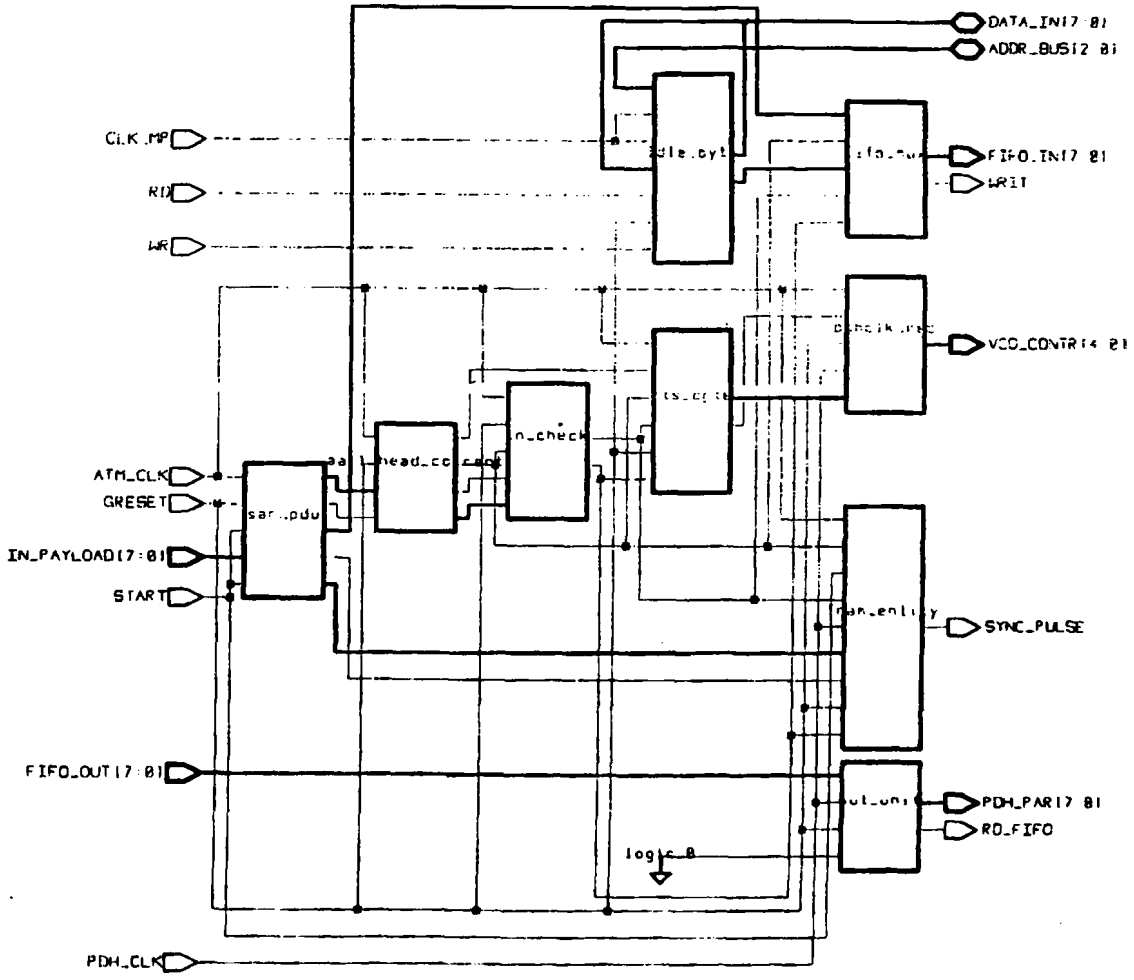
- Το pdhclk_rec ανακτά το ρολόϊ υπηρεσίας (PDHCLK) μέσω της διαφοράς από το αποτέλεσμα της σύγκρισης του εισερχόμενου πεδίου RTS με το πεδίο RTS που δημιουργήθηκε από το rts_crte.

- Το idle_byte εισάγει αδρανή δεδομένα στην περίπτωση που τα εισερχόμενα είναι λανθασμένα ώστε να μη χαθεί ο ρυθμός μεταδοσης.

- Το fifo_mux παρέχει τα απαιτούμενα σήματα εγγραφής της FIFO.

- Το fram_entity εξάγει το παλμό συχρονισμού του πλασίου PDH.

- Το out_unit οδηγεί τα προερχόμενα από τη FIFO δεδομένα στη διεπαφή PDH.



Σχήμα 6.2: Το υλοποιημένο διάγραμμα της μονάδας AALRX με VHDL

Αρχικά, οι επιμέρους υπομονάδες από τις οποίες αποτελούνται οι μονάδες ATMRX και AALRX, περιγράφηκαν σε VHDL. Σε κάθε υπομονάδα ξεχωριστά γίνεται λειτουργική προσομοίωση. Ο σωστός λειτουργικά πλέον κώδικας της VHDL της κάθε υπομονάδας συντίθεται με τους κώδικες των άλλων υπομονάδων που συνθέτουν την μονάδα ATMRX ώστε να δημιουργηθεί ένας ολοκληρωμένος κώδικας. Στη συνέχεια ακολουθείται η παραπάνω διαδικασία για την λειτουργική προσομοίωση. Η παραπάνω

διαδικασία γίνεται και για την μονάδα AALRX. Τέλος, δημιουργείται ένας ενοποιημένος κώδικας VHDL για όλο τον δέκτη.

Τις περισσότερες φορές ένας τόσο μεγάλος κώδικας VHDL δεν χωρά σε εμπορικά διαθέσιμα FPGA, οπότε υποχρεωτικά γίνεται διαχωρισμός του κώδικα σε μικρότερα ανεξάρτητα κομάτια ώστε το κάθε κομμάτι να χωρά σε ένα FPGA.

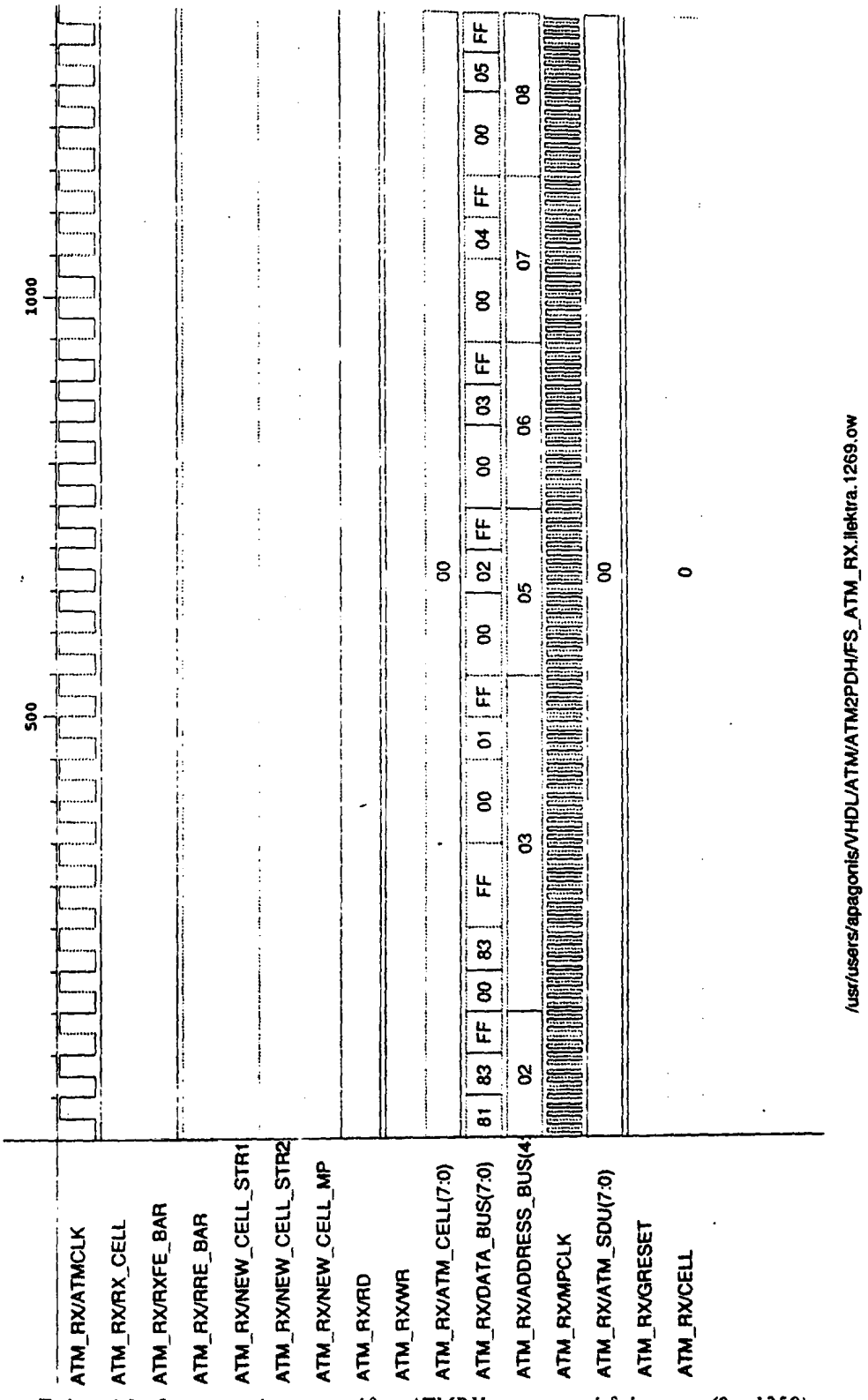
Τέλος, πραγματοποιείται η υλοποίηση του συστήματος σε ένα ή περισσότερα FPGA.

6.3 *Αποτελέσματα του προγράμματος VHDL*

Ο χρονισμός των σημάτων εισόδου, εξόδου της μονάδας ATMRX, της μονάδας AALRX καθώς επίσης και του δέκτη ATM/PDH που είναι το αποτέλεσμα της προσομοίωσης του προγράμματος του παραρτήματος Α φαίνεται στα επόμενα σχήματα. Συγκεκριμένα στα σχήματα 6.3, 6.4, 6.5, 6.6 παρατίθεται ο χρονισμός της μονάδας ATMRX, για τα διαστήματα χρόνου από 0-1350ns, 1350-2800ns, 2800-4300ns, 4300-5750ns, αντίστοιχα.

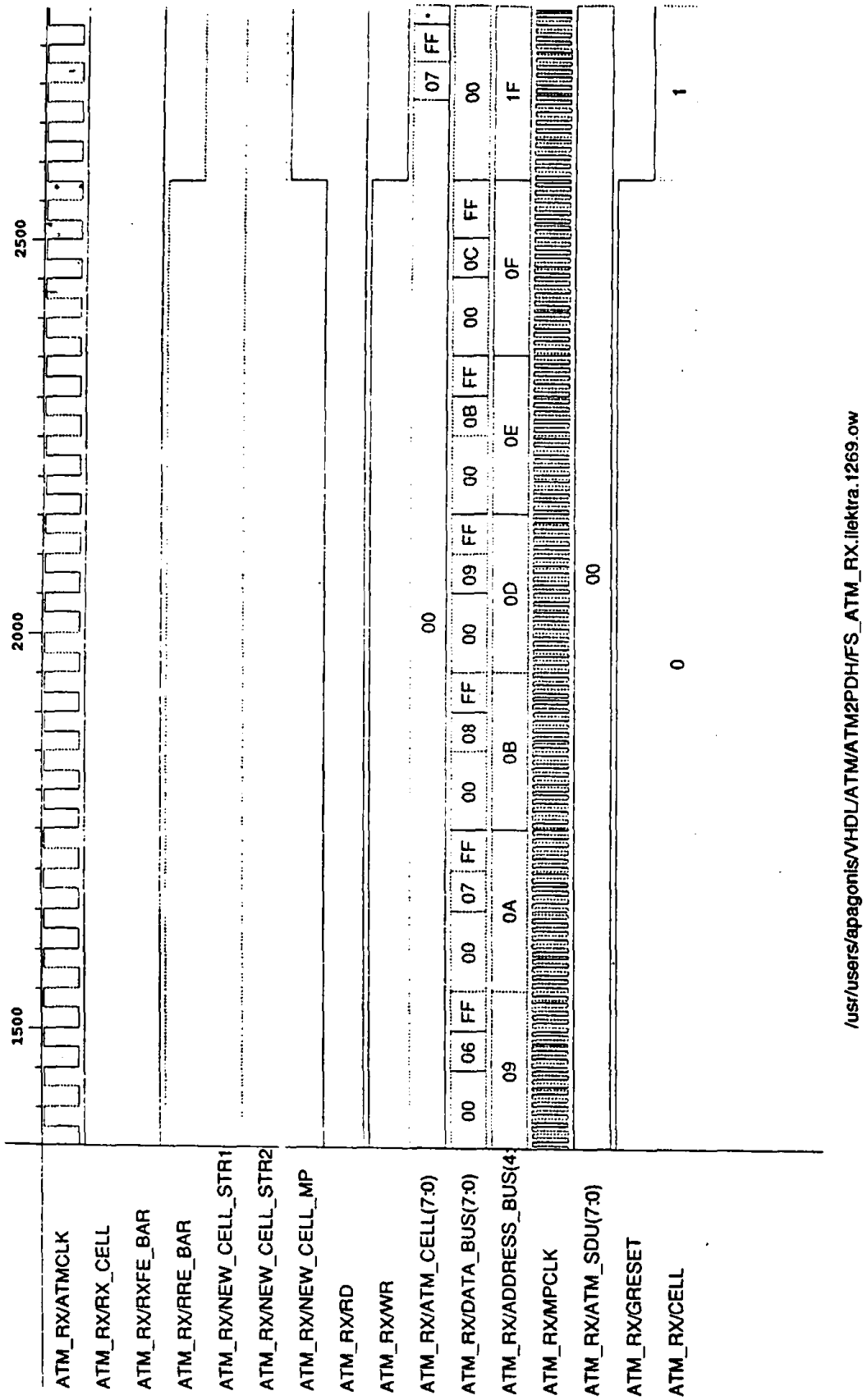
Τα σχήματα αυτά προέκυψαν από την λειτοπυργική προσομοίωση, του ATMRX με δεδομένα εισόδου σωστά πακέτα ATM και ενδιάμεσα λανθασμένα και αδρανή πακέτα ATM.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



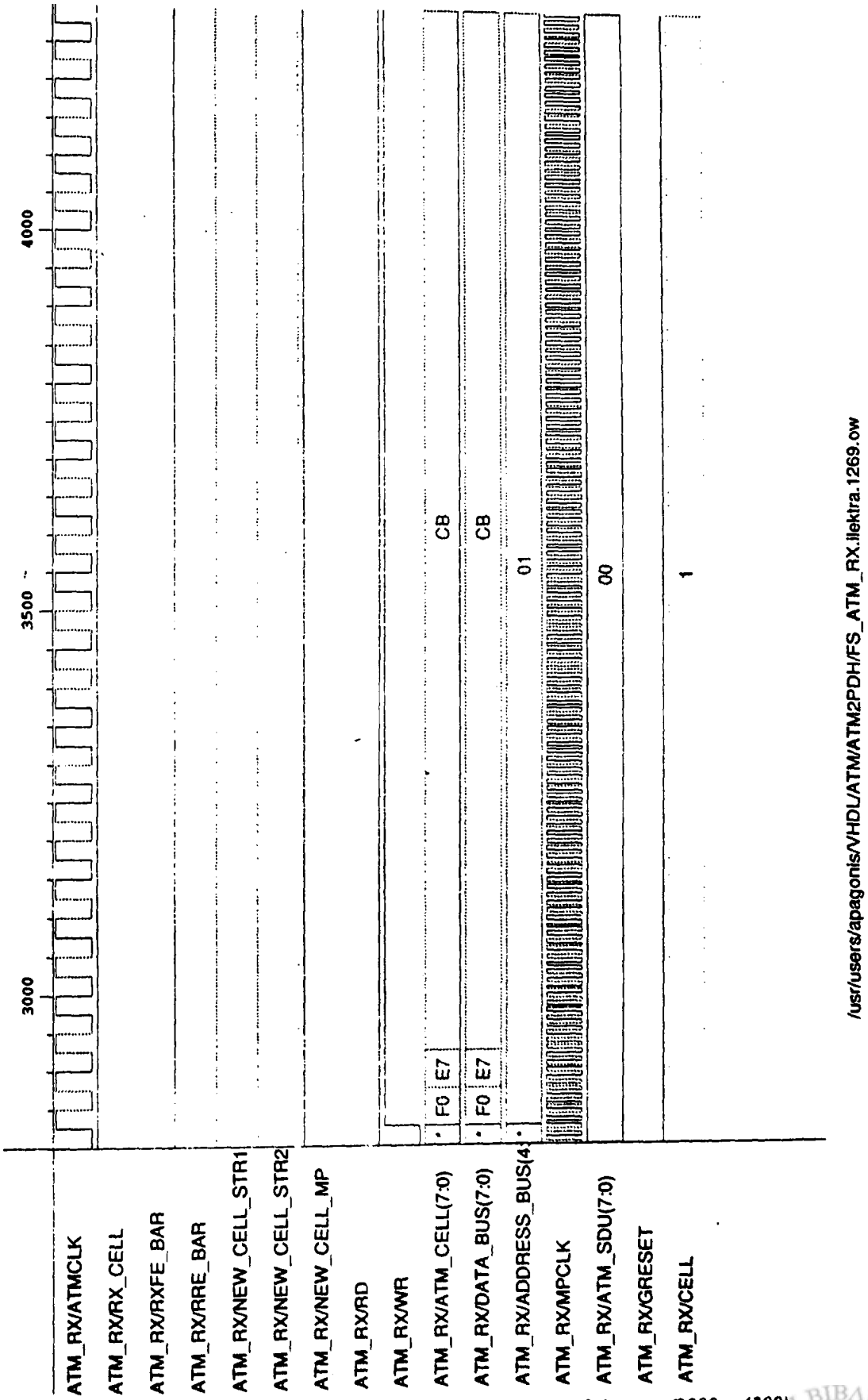
Σχήμα 6.3: Ο χρονισμός της μονάδας ATMRX για χρονικό διάστημα (0 – 1350)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



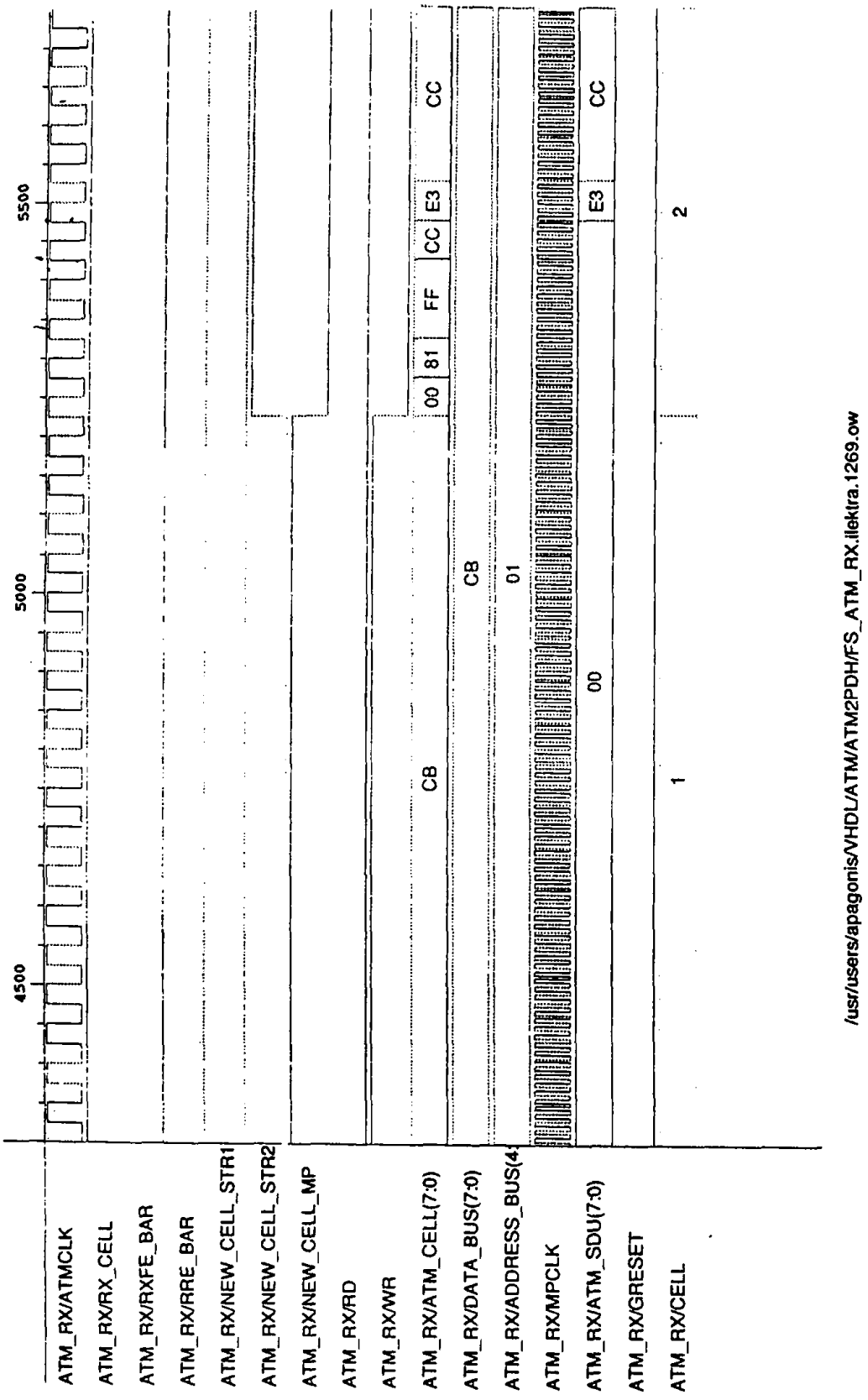
Σχήμα 6.4: Ο χρονισμός της μονάδας ATMRX για χρονικό διάστημα (1350 – 2800)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 6.5: Ο χρονισμός της μονάδας ATMRX για χρονικό διάστημα (2800 – 4300)ns

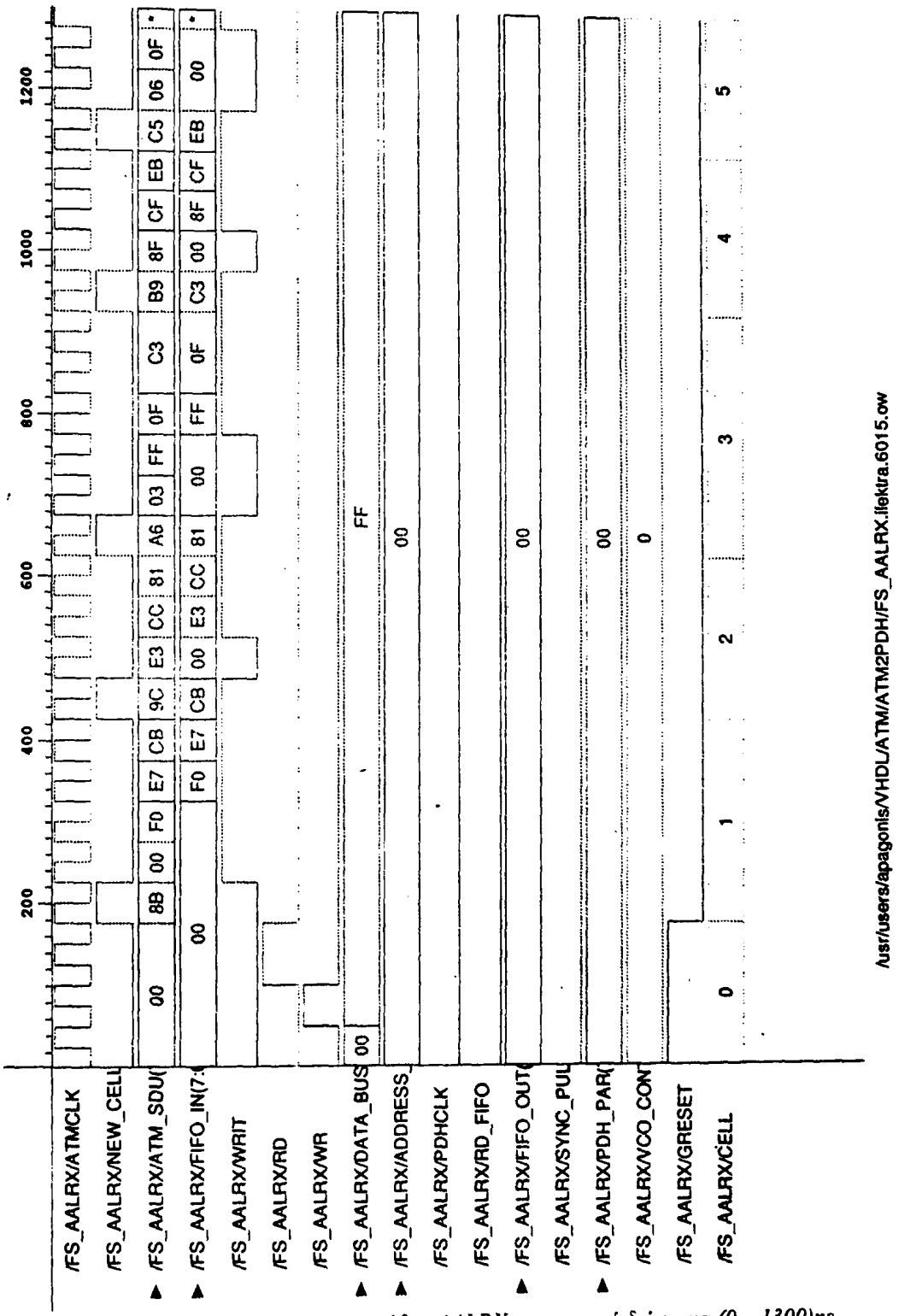
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 6.6: Ο χρονισμός της μονάδας ATMRX για χρονικό διάστημα (4300 – 5750)ns

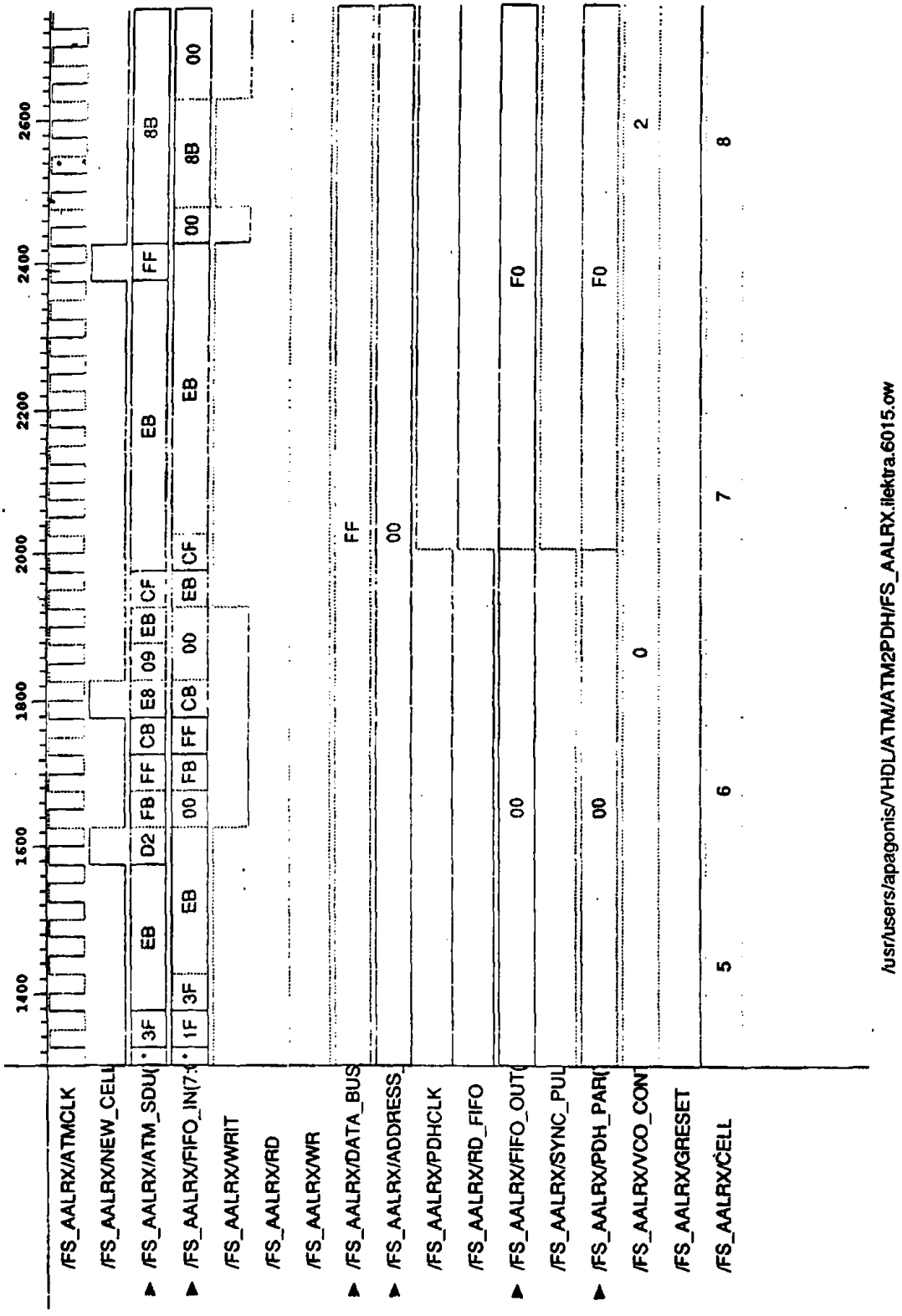
Στα σχήματα 6.7 και 6.8 παρατίθεται ο χρονισμός της μονάδας AALRX, για τα διαστήματα χρόνου 0-1300ns, 1300-2750ns, αντίστοιχα.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



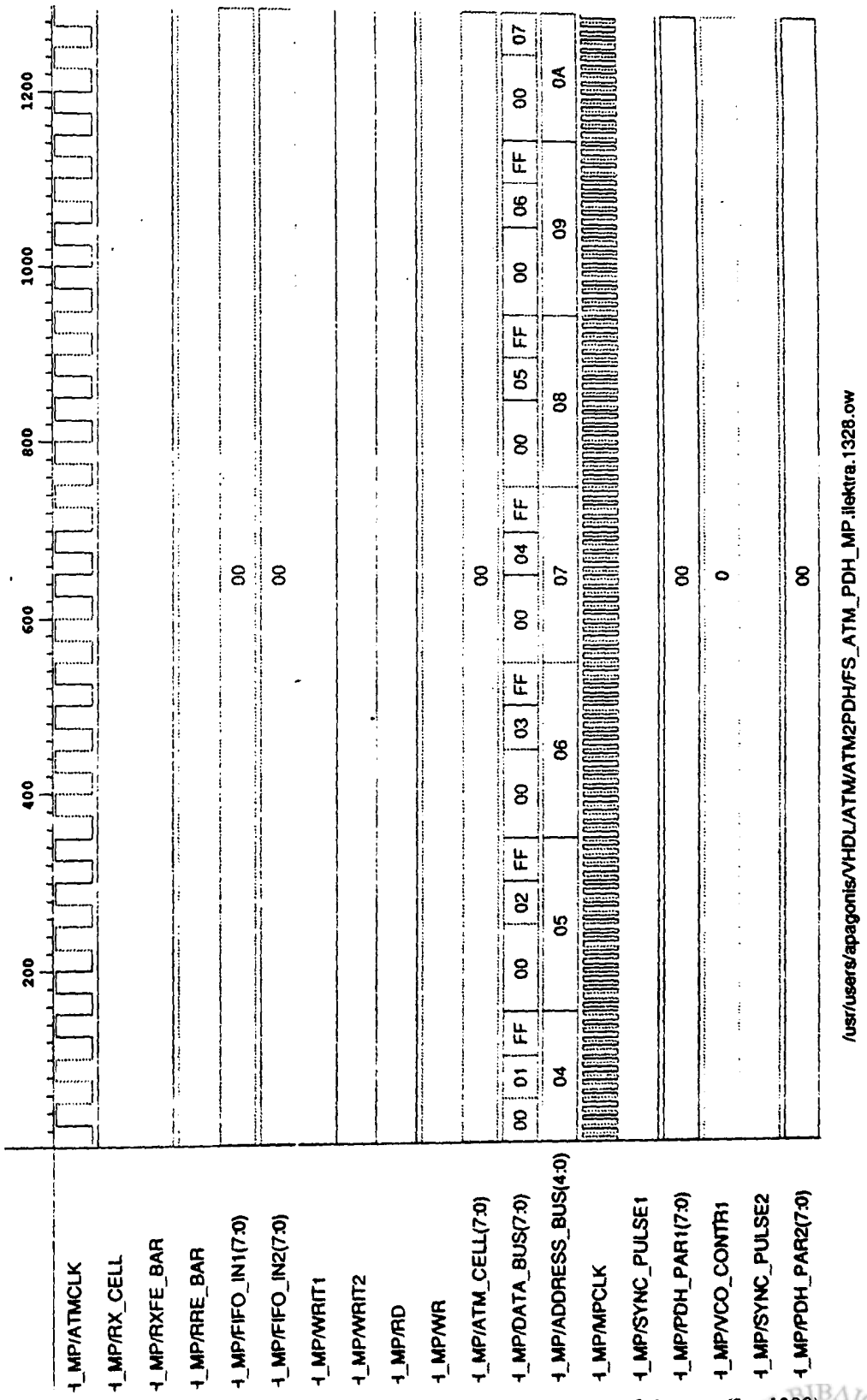
Σχήμα 6.7: Ο χρονισμός της μονάδας AALRX για χρονικό διάστημα (0 – 1300)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



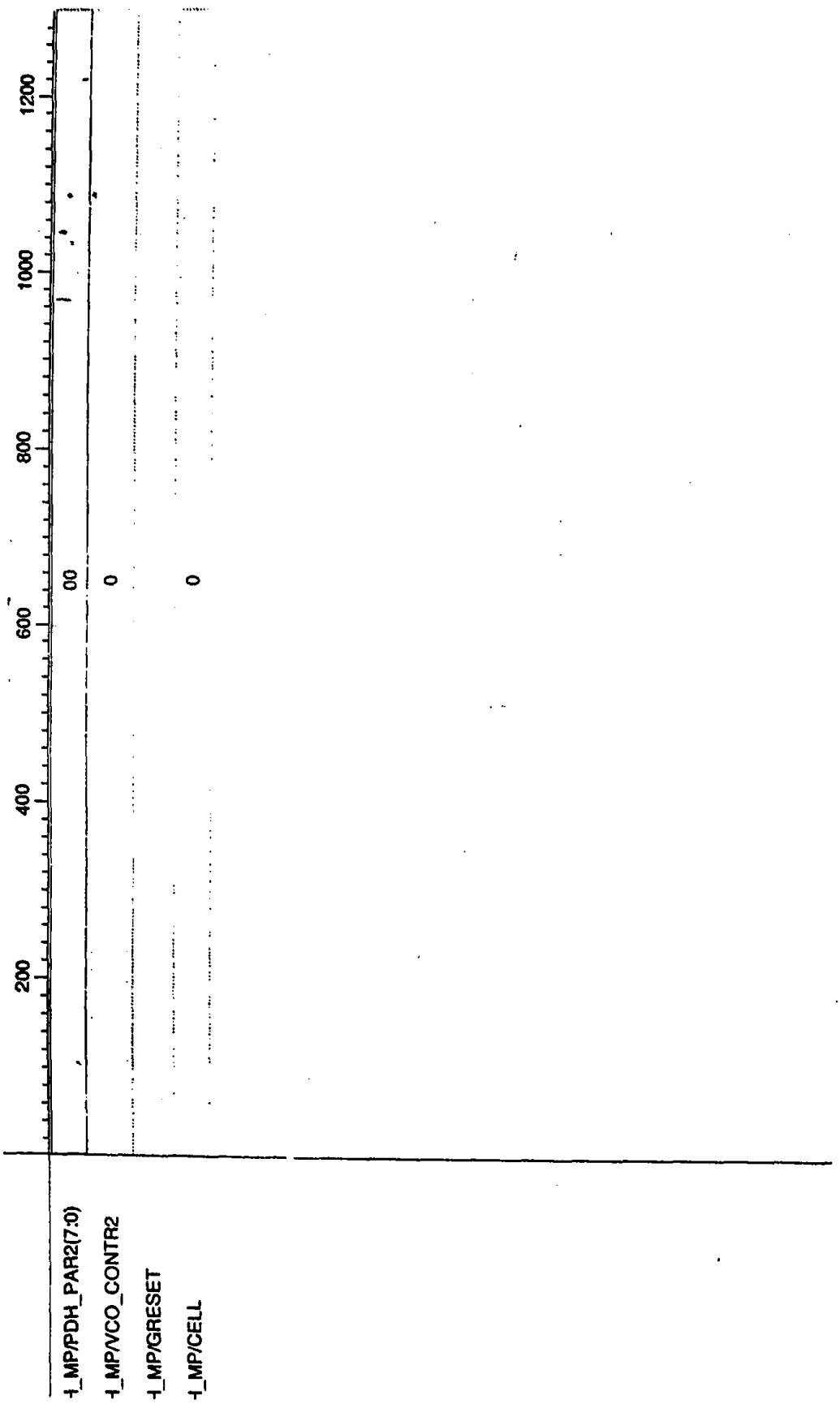
Σχήμα 6.8: Ο χρονισμός της μονάδας AALRX για χρονικό διάστημα (1300 – 2750)ns

Στα σχήματα 6.9, 6.10, 6.11, 6.12 παρατίθεται ο χρονισμός του δέκτη κατά την φόρτωση των υποστηριζώμενων επικεφαλίδων μέσω της διεπαφής μικροελεγκτή, για τα διαστήματα χρόνου 0-1300ns, 1300-2800ns, 2800-4300ns, 4300-5400ns, αντίστοιχα.



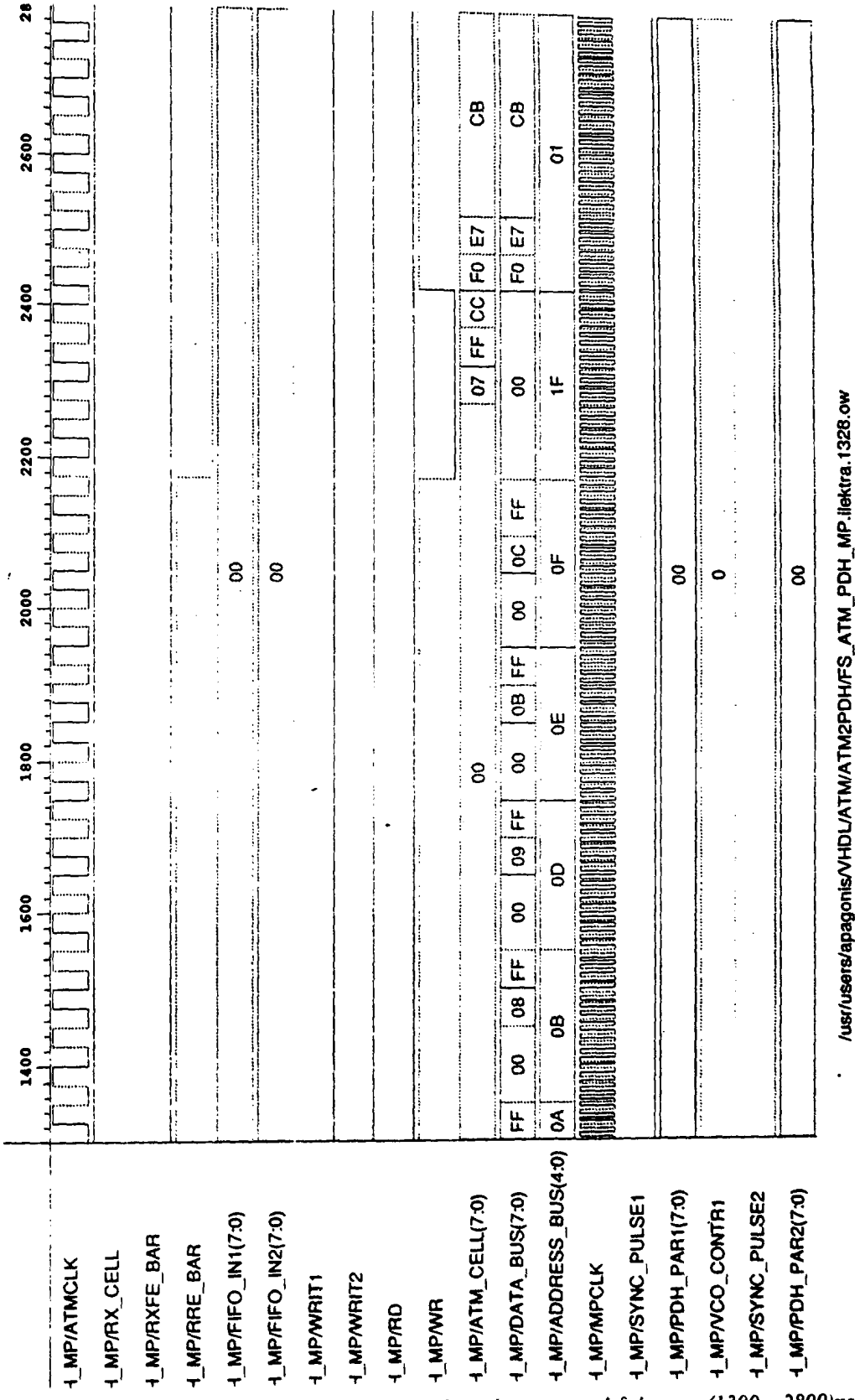
Σχήμα 6.9: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (0 – 1300)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



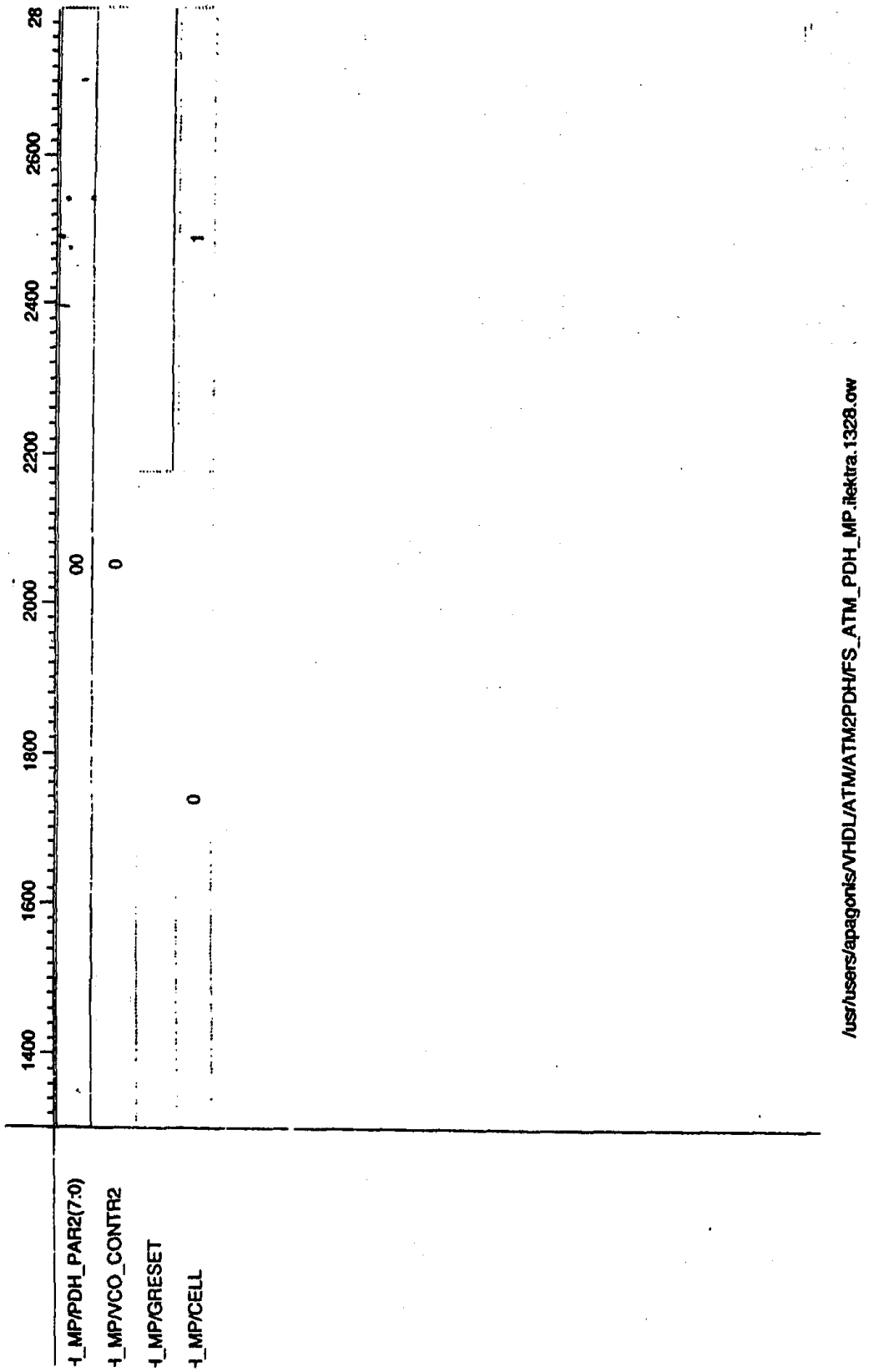
Σχήμα 6.9: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (0 – 1300)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΠΑΤΡΩΝ



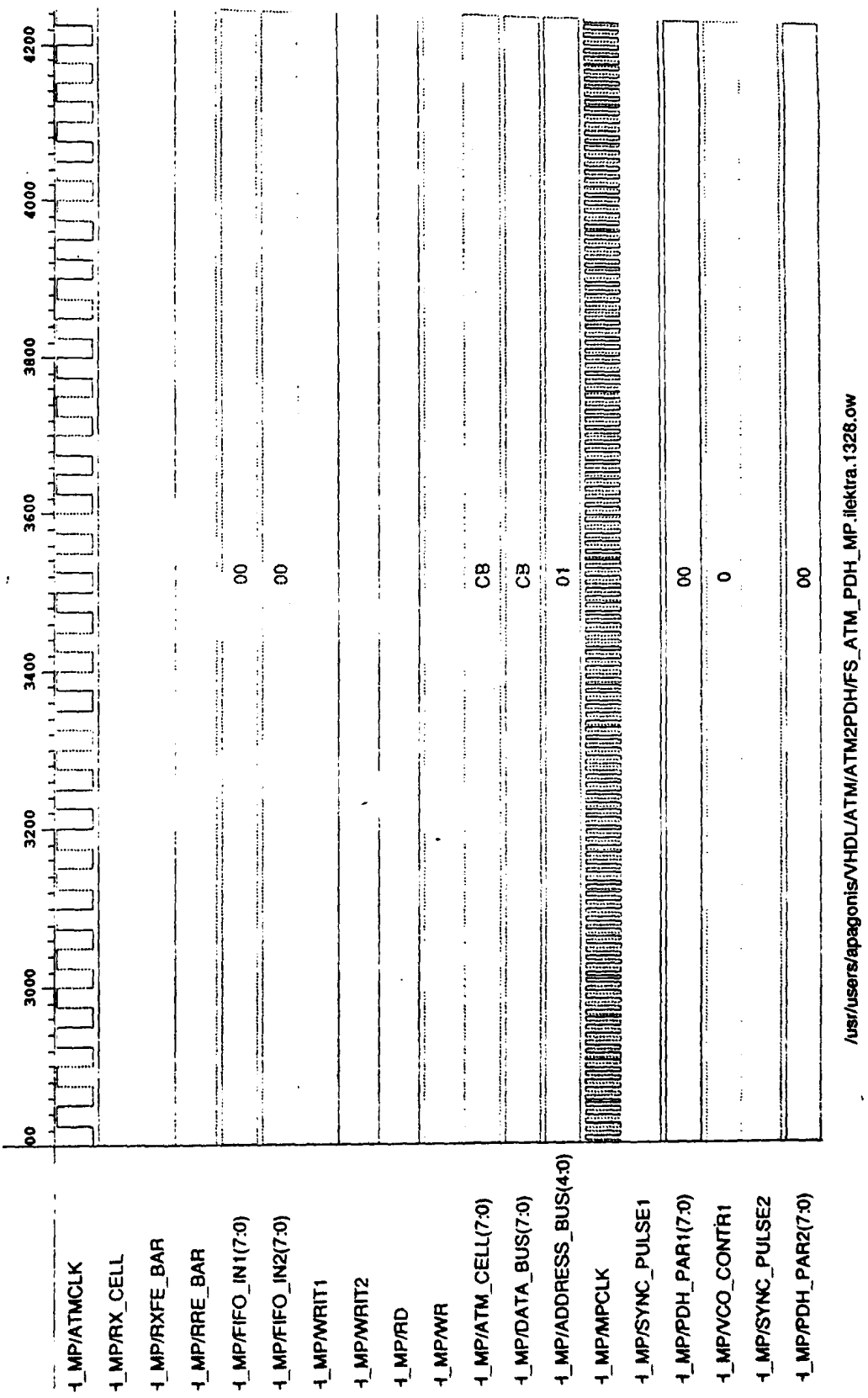
Σχήμα 6.10: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (1300 – 2800)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



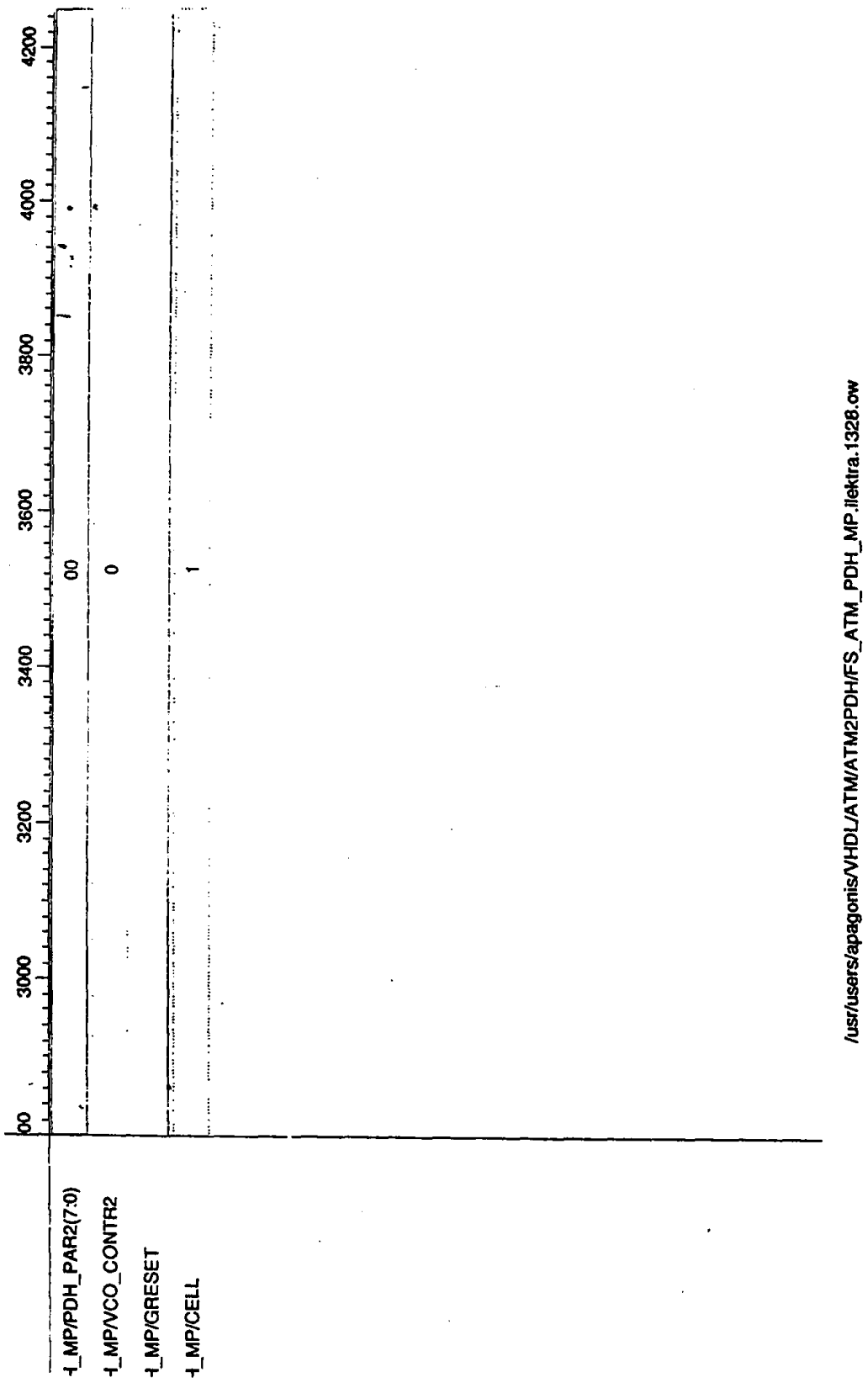
Σχήμα 6.10: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (1300 – 2800)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ



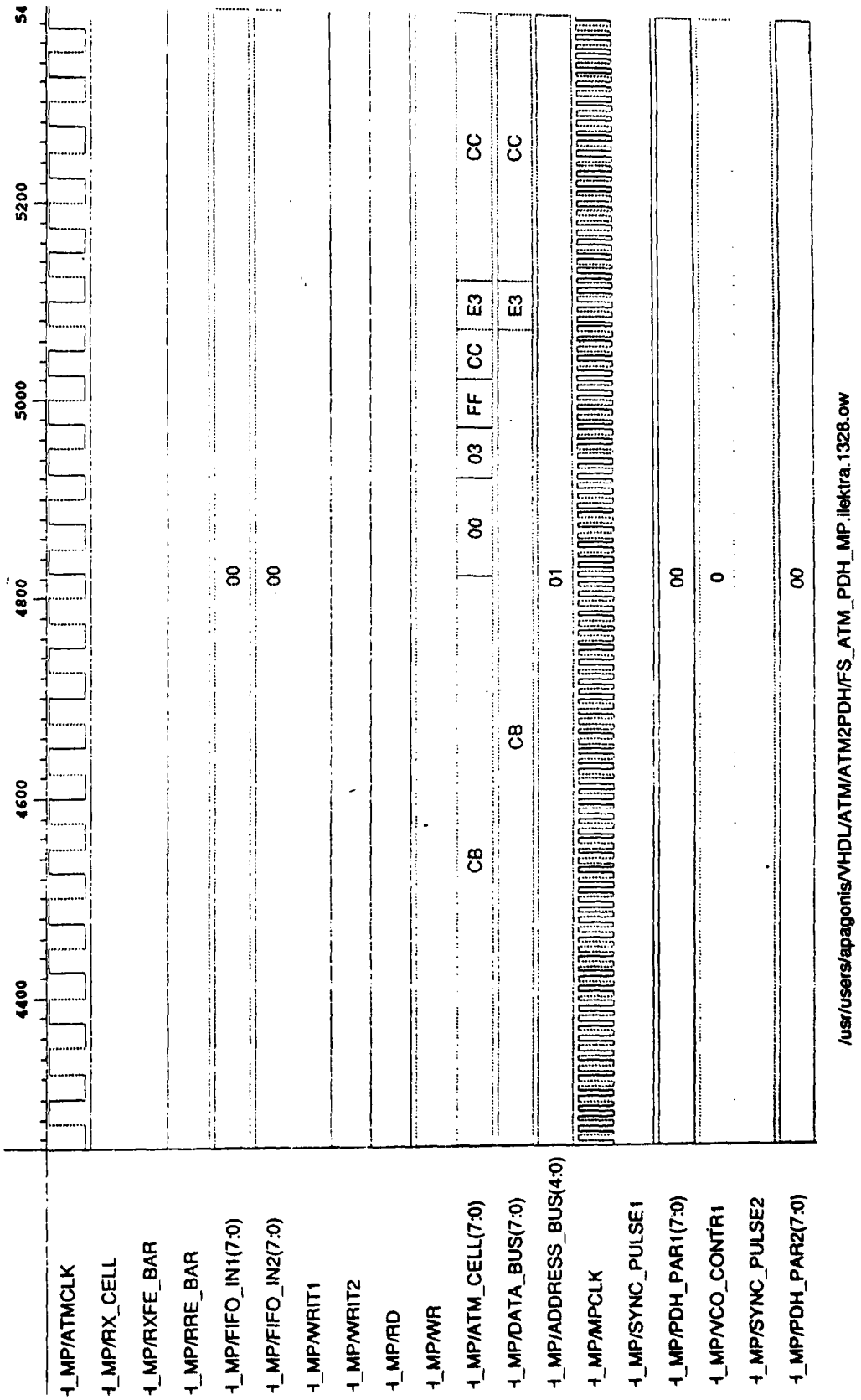
Σχήμα 6.11: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (2800 – 4300)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

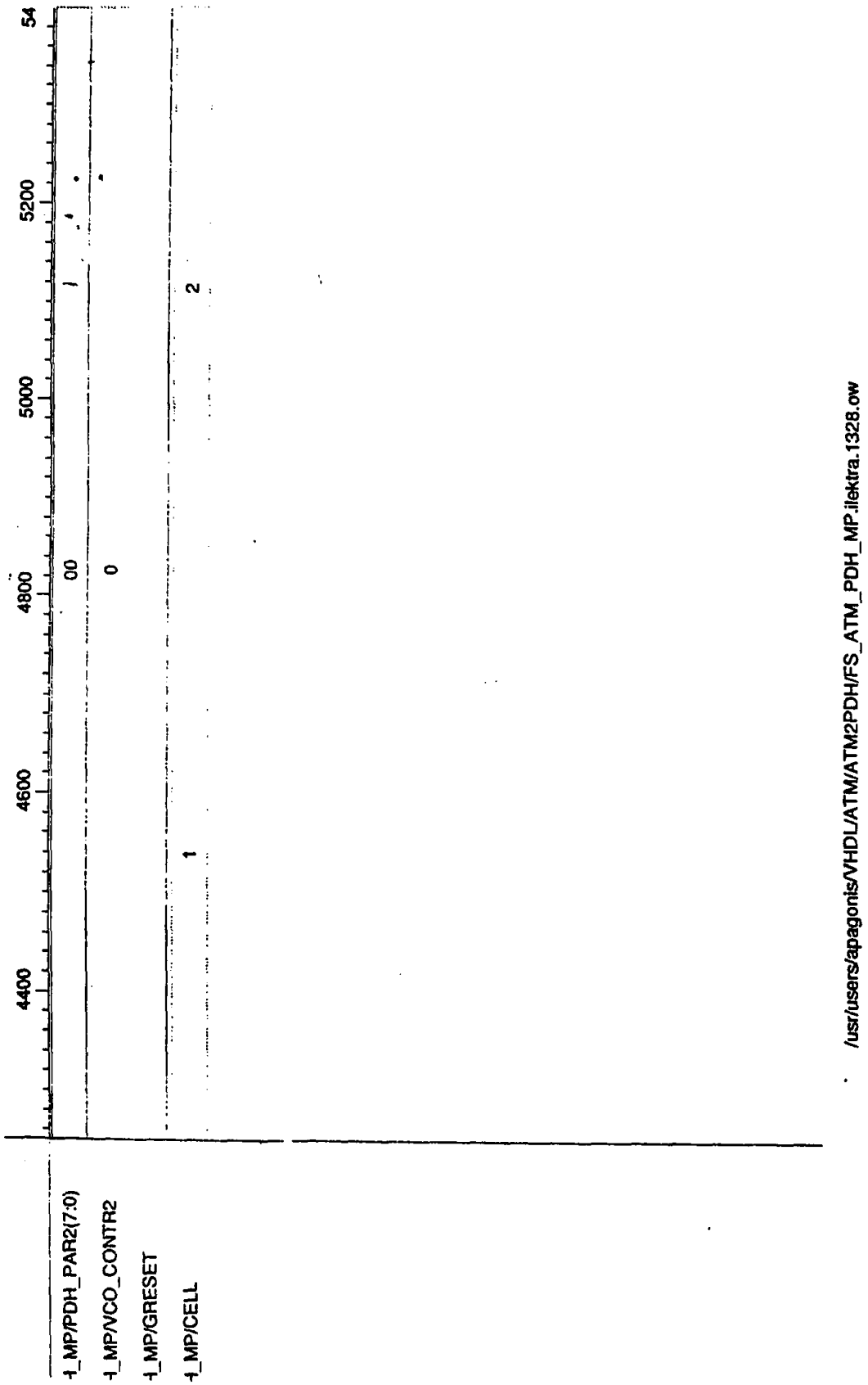


Σχήμα 6.11: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (2800 – 4300)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



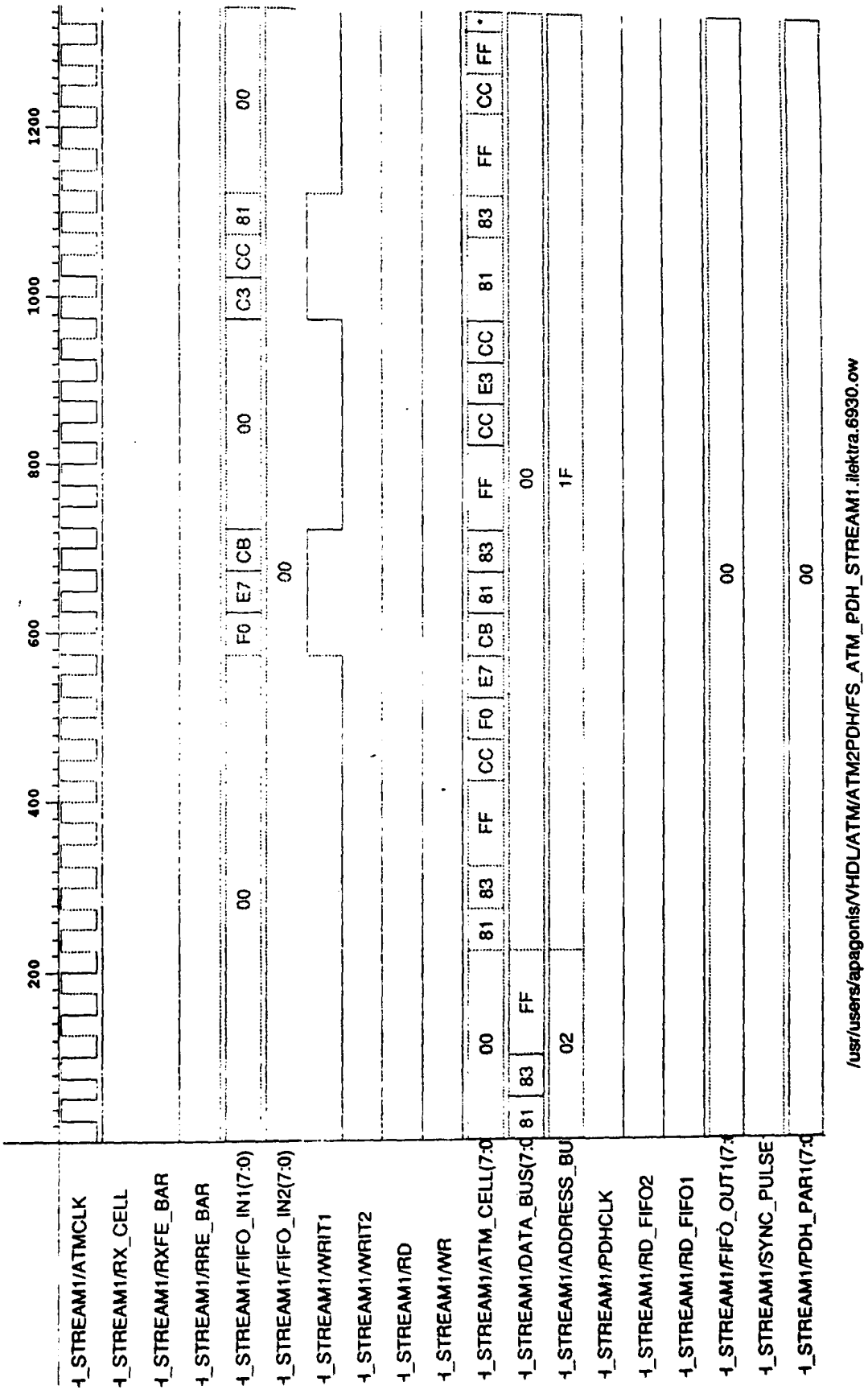
Σχήμα 6.12: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (4300 – 5400)ns



Σχήμα 6.12: Ο χρονισμός της διεπαφής μικροελεγκτή για χρονικό διάστημα (4300 – 5400)ns (συνέχεια)

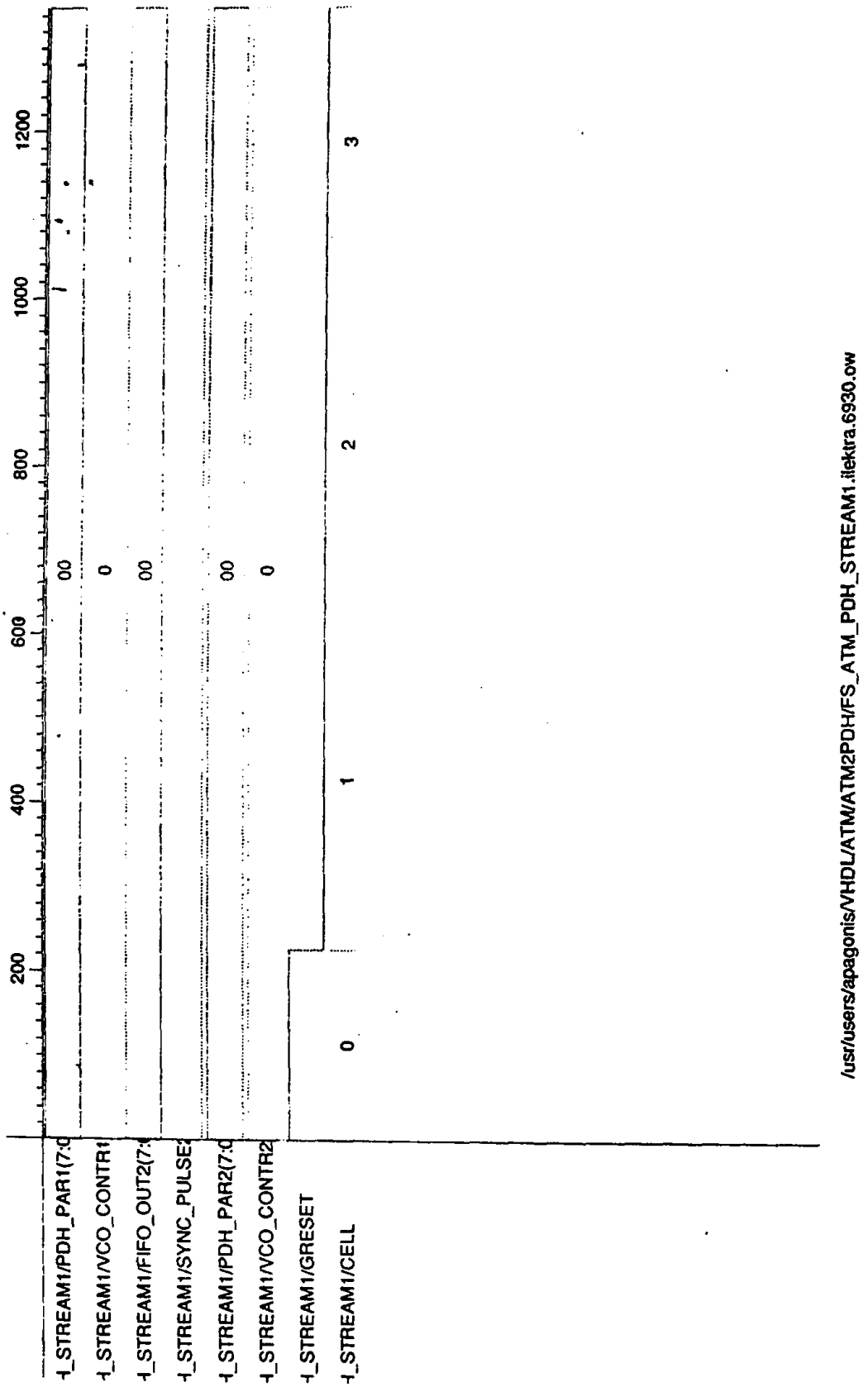
Στα σχήματα 6.13, 6.14 παρατίθεται ο χρονισμός του δέκτη ATM/PDH για τη μία ροή χρήστη για τα διαστήματα χρόνου 0-1350ns, 1350-2800ns, αντίστοιχα.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



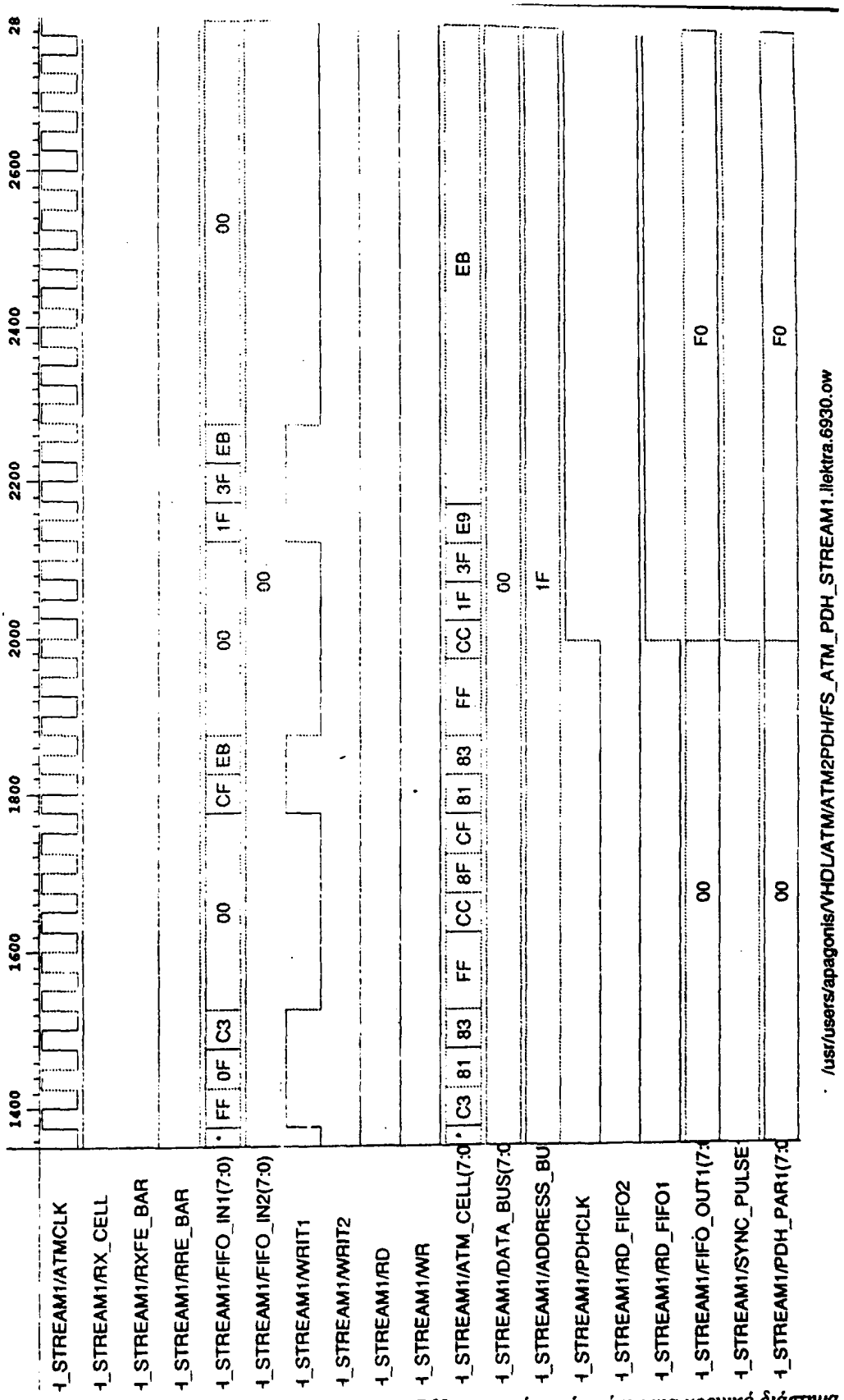
Σχήμα 6.13: Ο χρονισμός του δέκτη ATM/PDH για τη μία ροή χρήστη για χρονικό διάστημα (0 – 1350)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



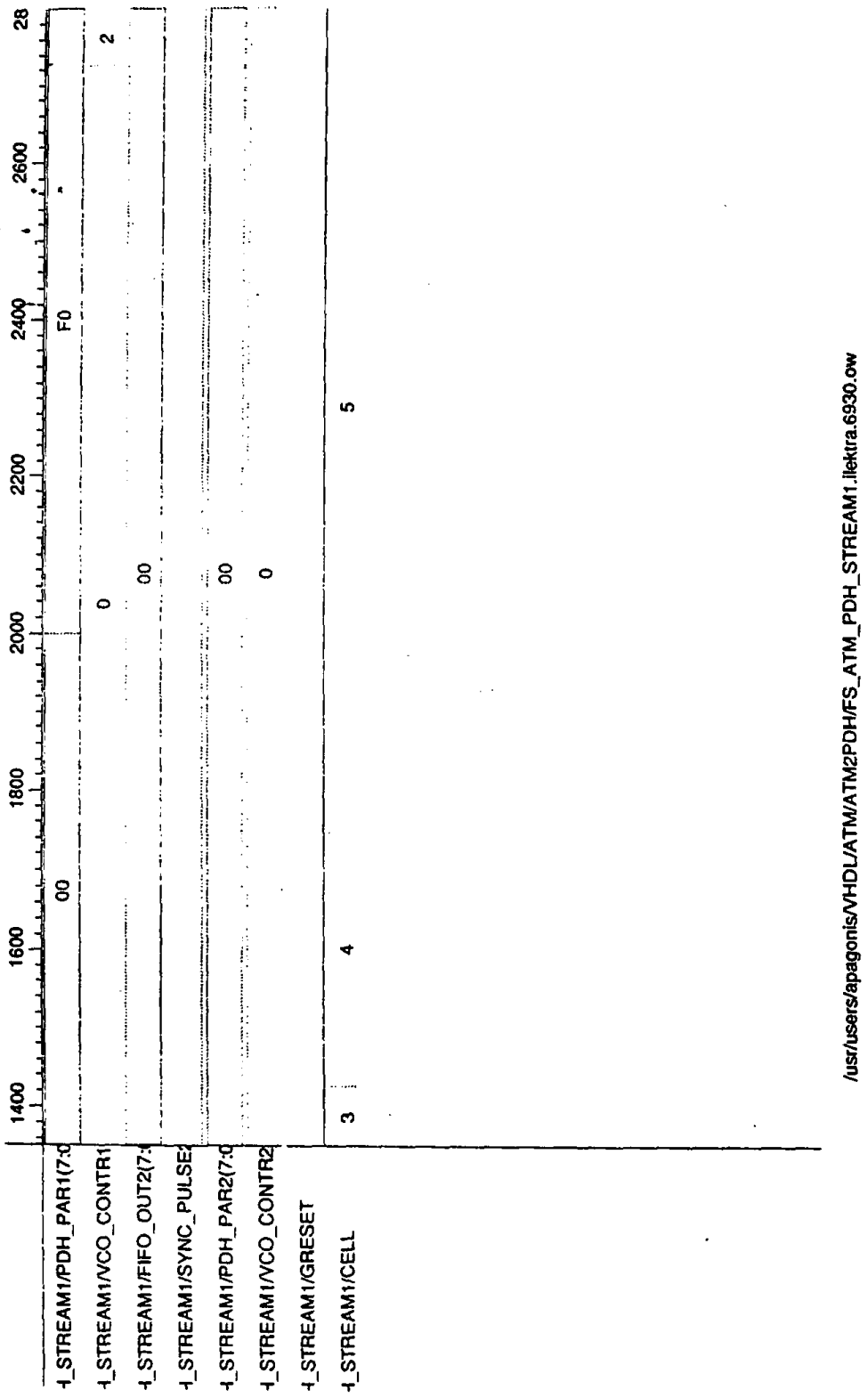
Σχήμα 6.13: Ο χρονισμός του δέκτη ATM/PDH για τη μία ροή χρήστη για χρονικό διάστημα (0 – 1350)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



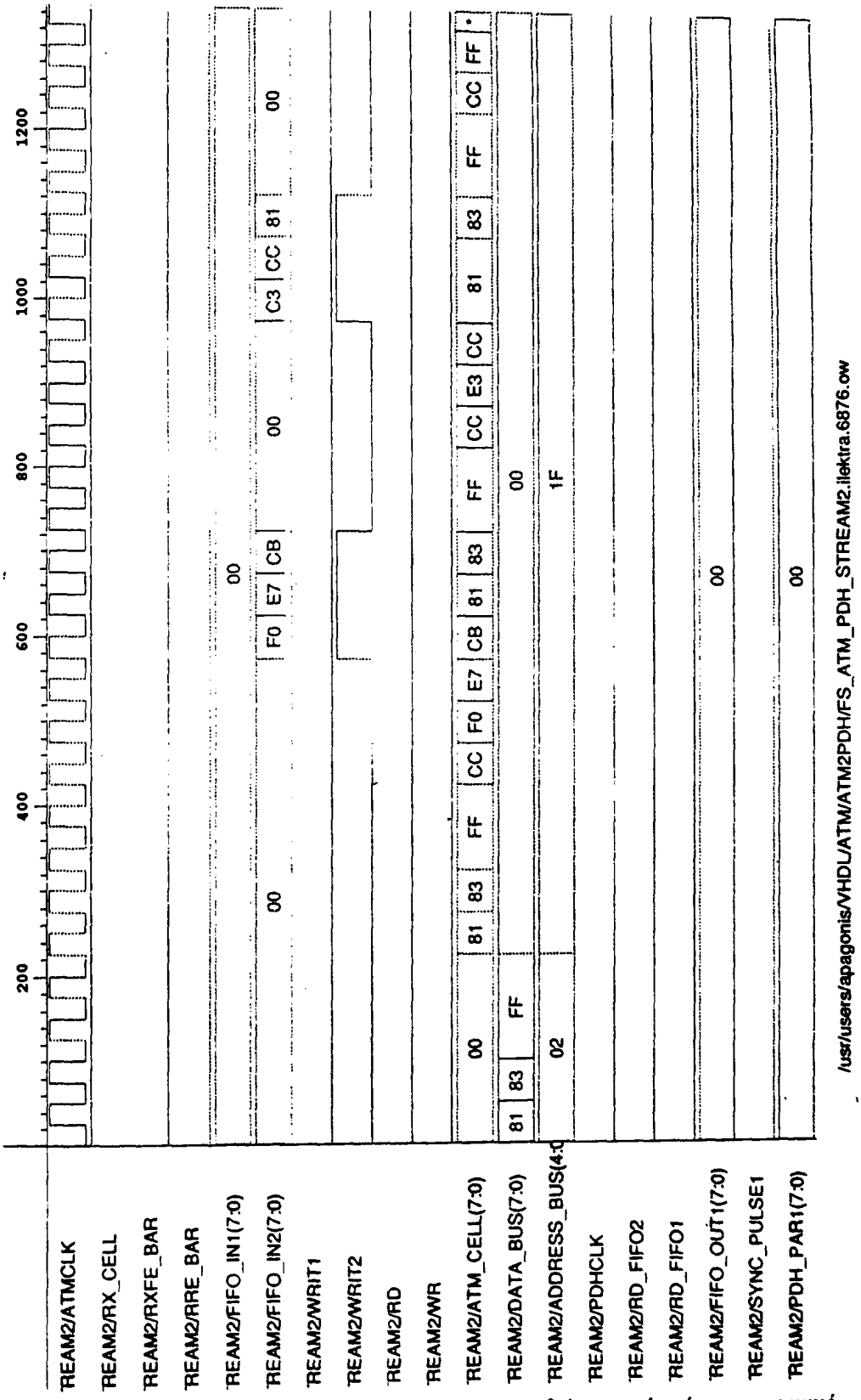
Σχήμα 6.14: Ο χρονισμός του δέκτη ATM/PDH για τη μία ροή χρήστη για χρονικό διάστημα (1350 – 2800)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



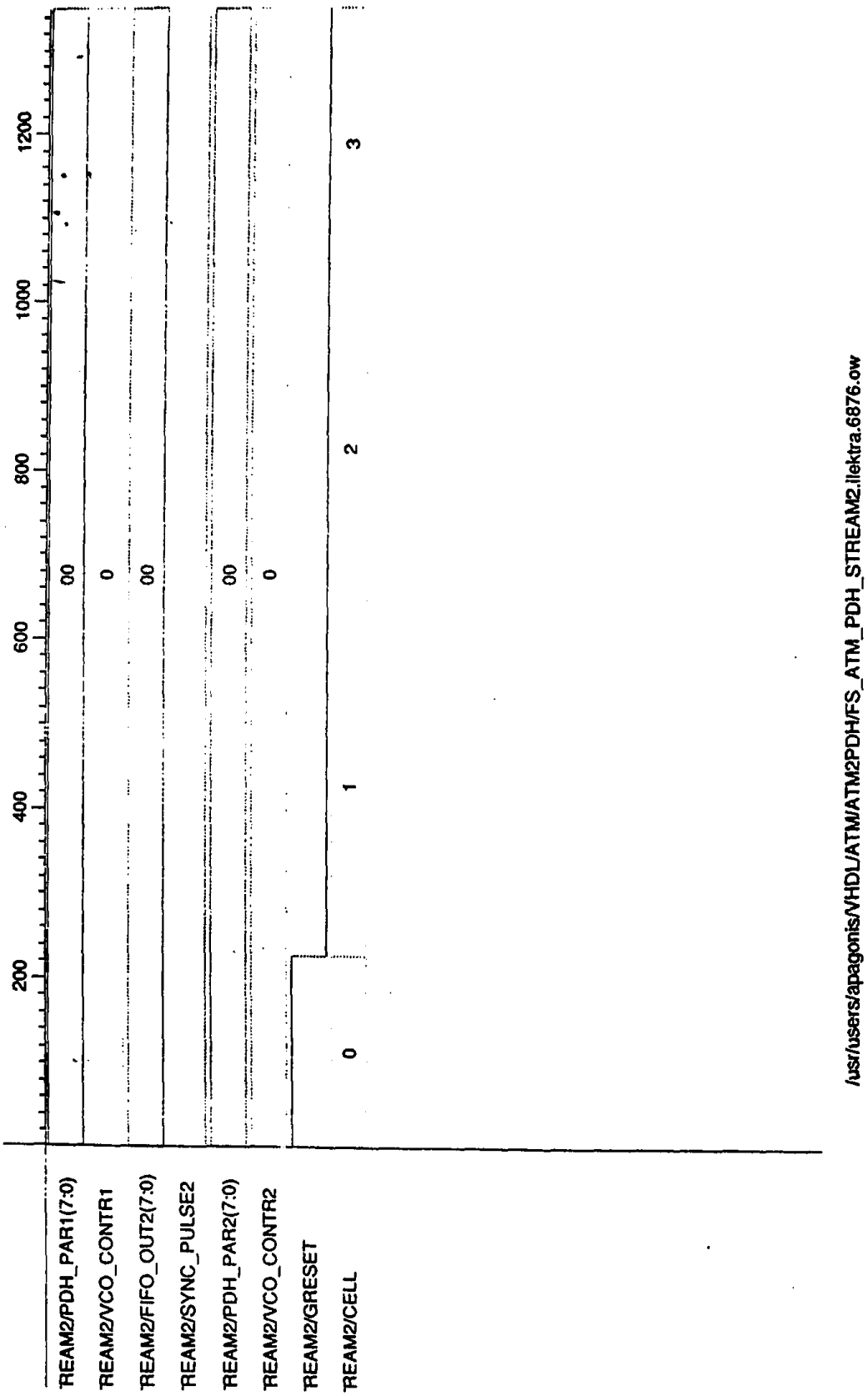
Σχήμα 6.14: Ο χρονισμός του δέκτη ATM/PDH για τη μία ροή χρήστη για χρονικό διάστημα (1350 – 2800)ns (συνέχεια)

Στα σχήματα 6.15, 6.16 παρατίθεται ο χρονισμός του δέκτη ATM/PDH για τη δεύτερη ροή χρήστη για τα διαστήματα χρόνου 0-1350ns, 1350-2800ns, αντίσοιχα.



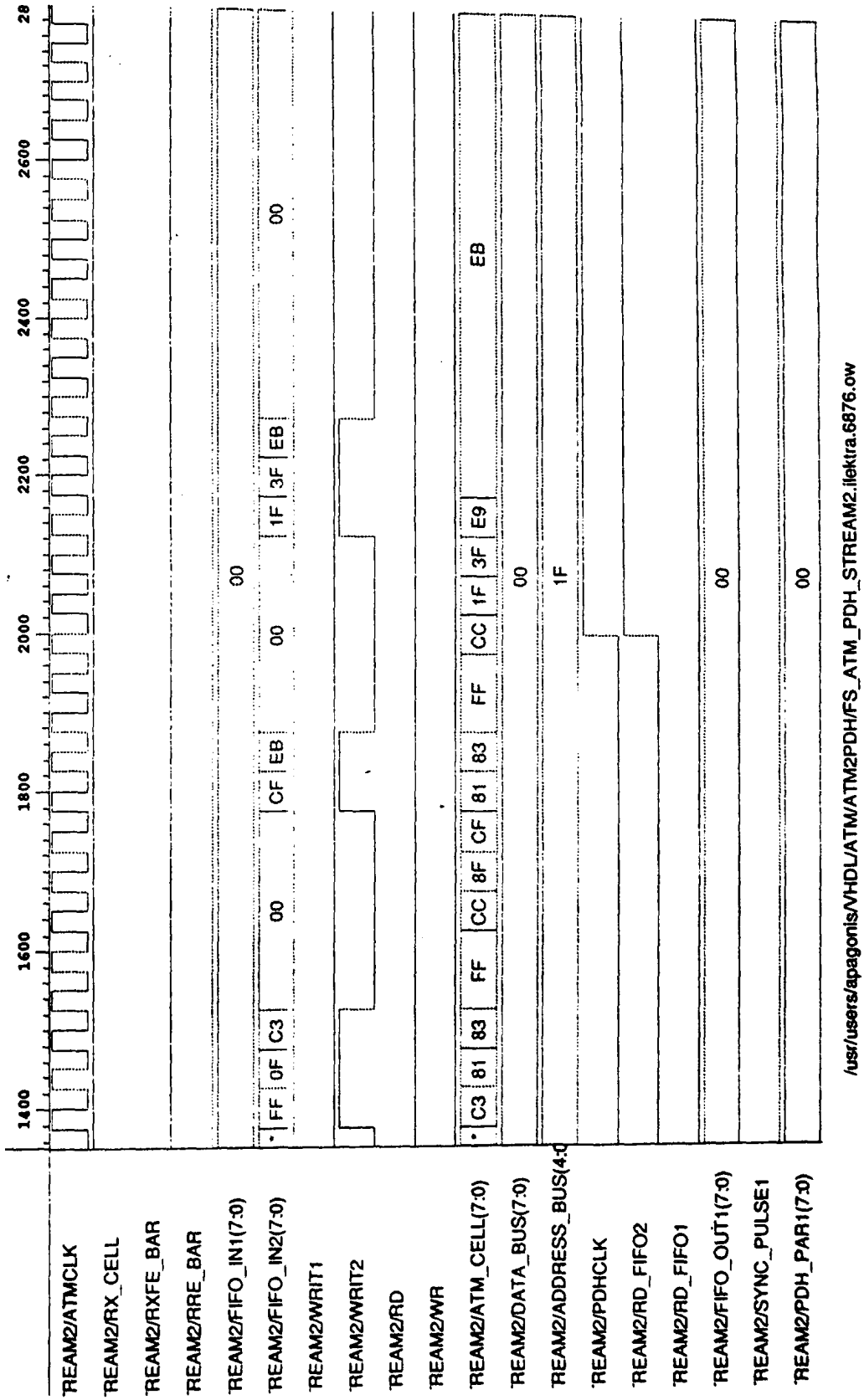
Σχήμα 6.15: Ο χρονισμός του δέκτη ATM/PDH για τη δεύτερη ροή χρήστη για χρονικό διάστημα (0 – 1350)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



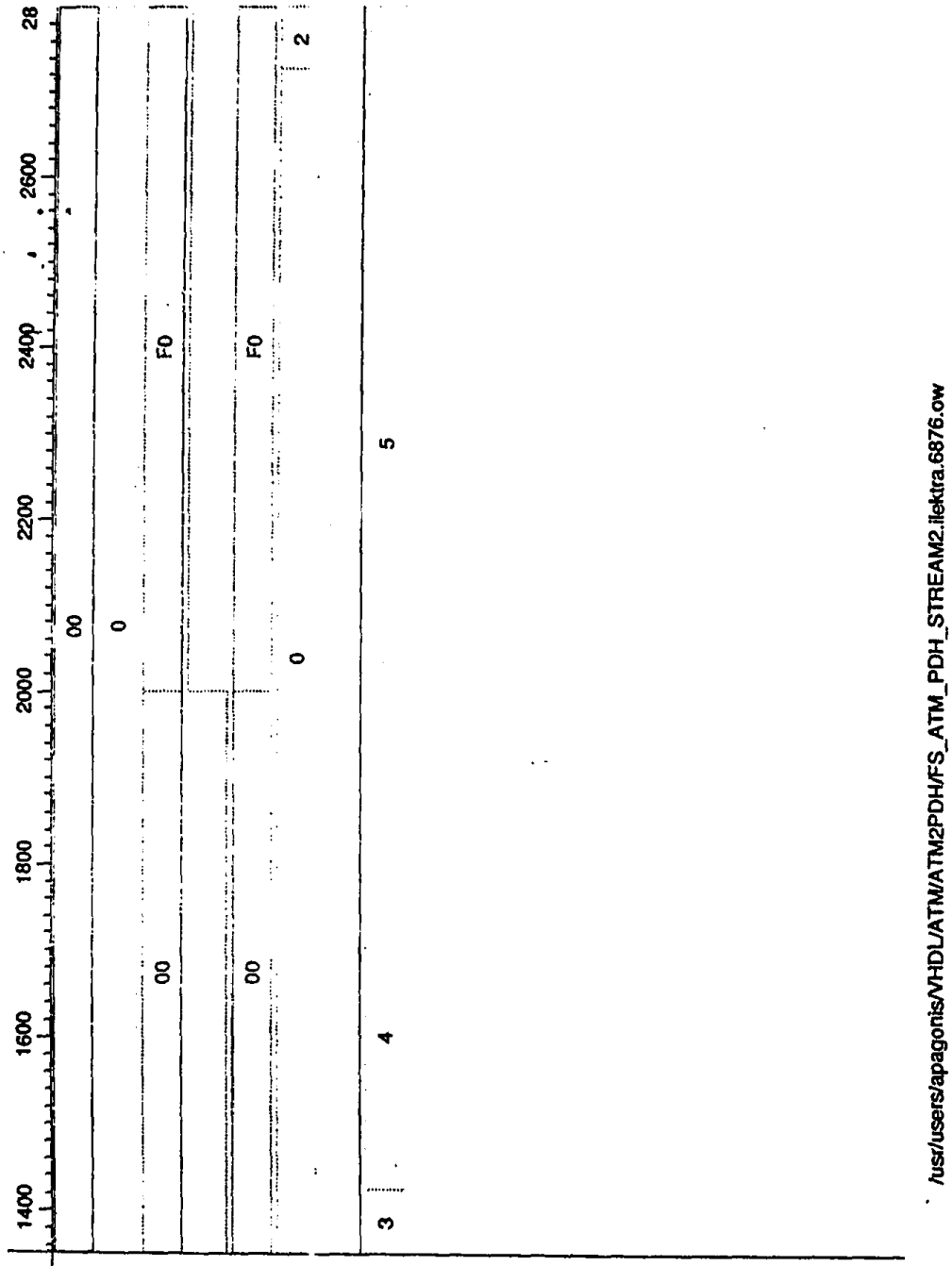
Σχήμα 6.15: Ο χρονισμός του δέκτη ATM/PDH για τη δεύτερη ροή χρήστη για χρονικό διάστημα (0 – 1350)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



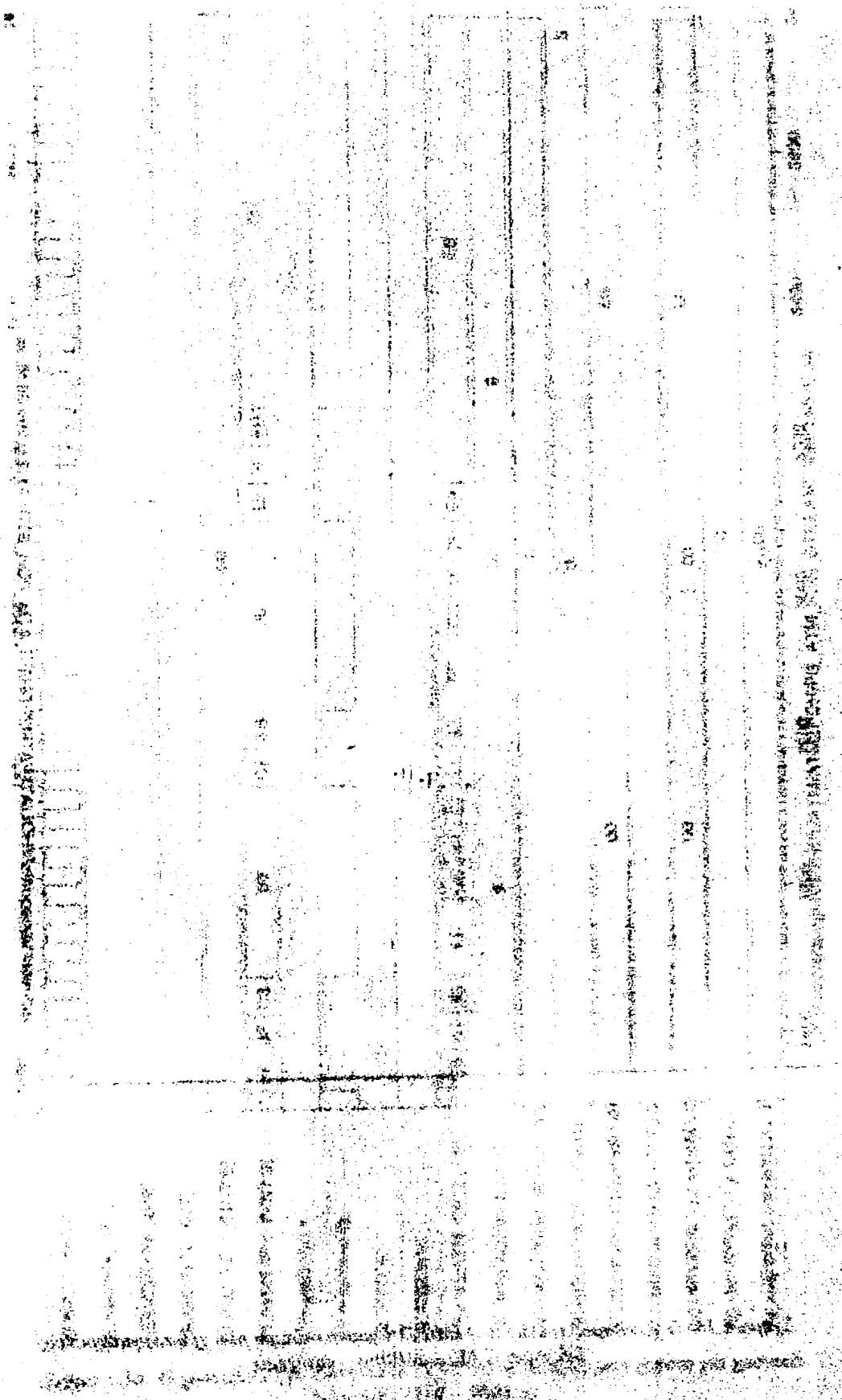
Σχήμα 6.16: Ο χρονισμός του δέκτη ATM/PDH για τη δεύτερη ροή χρήστη για χρονικό διάστημα (1350 – 2800)ns

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 6.16: Ο χρονισμός του δέκτη ATM/PDH για τη δεύτερη ροή χρήστη για χρονικό διάστημα (1350 – 2800)ns (συνέχεια)

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

7. Εισαγωγή στο σύστημα ΤΗΛ2

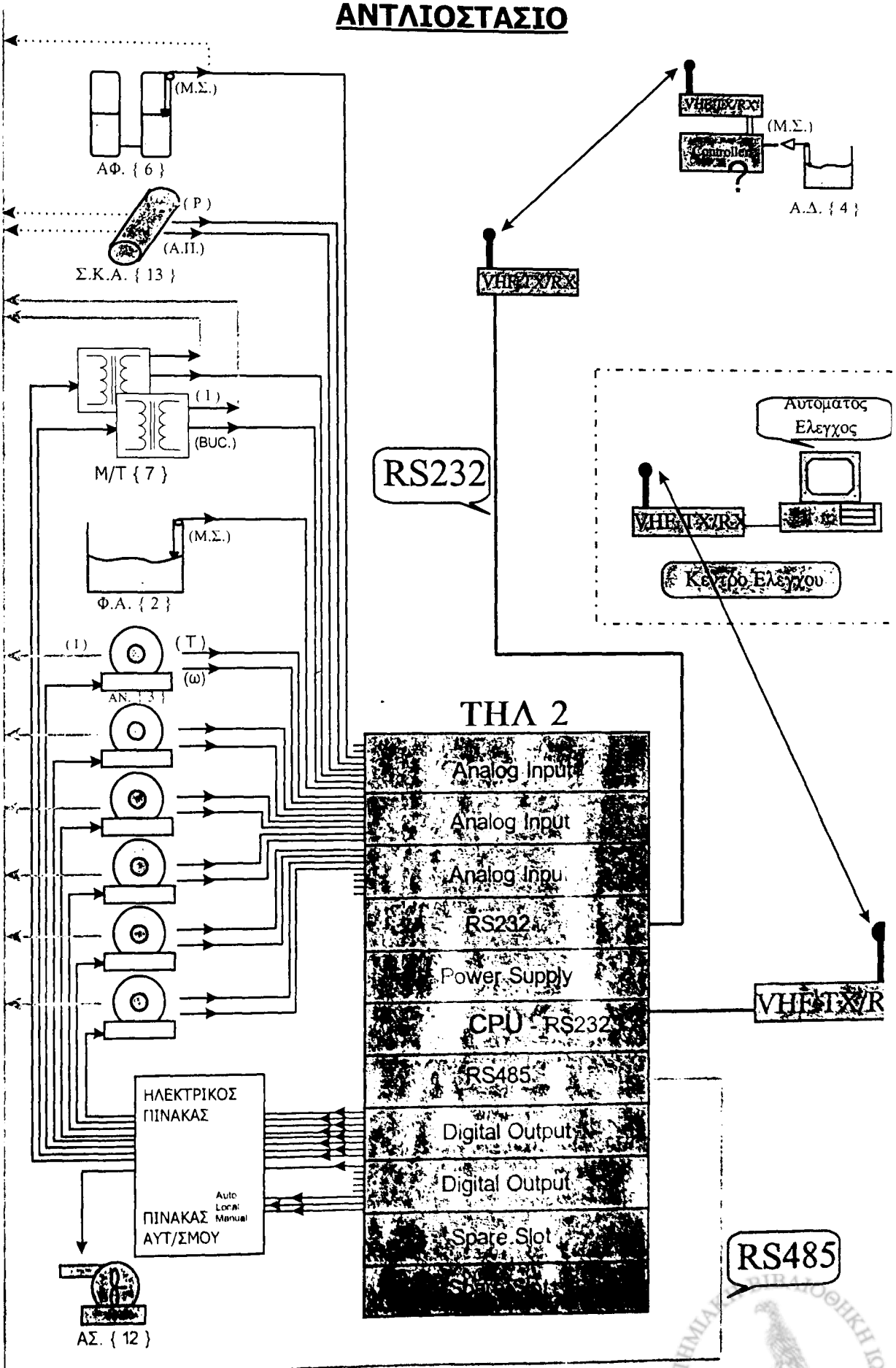
Ο σχεδιασμός και η υλοποίηση των τυπωμένων κυκλωμάτων των καρτών I/O έγινε στα πλαίσια του έργου με τίτλο "Ανάπτυξη συστήματος τηλεδιαχείρησης υδάτινων πόρων" (ΑΣΤΗΛΔΥΠ) του προγράμματος ΕΚΒΑΝ Π, της Γενικής Γραμματείας Έρευνας και Τεχνολογίας. Το πρόγραμμα αυτό έχει ως σκοπό την δημιουργία ενός πιλοτικού συστήματος ΤΗΛ2 τηλεμετρίας και τηλεχειρισμού, το οποίο να καλύπτει ανάγκες τηλεπίβλεψης, τηλεμετρίας, τηλεχειρισμού και αυτοματισμού εγκαταστάσεων άρδευσης, οι οποίες βρίσκονται σε μεγάλη απόσταση τόσο μεταξύ τους (έως 2Km) όσο και από το κέντρο ελέγχου (ασύρματα έως 10Km). Οι επιμέρους μονάδες του συστήματος ελέγχονται από ένα κεντρικό Η/Υ μέσω ενός ψηφιακού δικτύου συλλογής δεδομένων και εκτέλεσης χειρισμών.

Οι μονάδες τηλεμετρίας για τη συλλογή μετρήσεων συγκροτούν ένα σύστημα από τερματικές μονάδες (modules) οι οποίες συνδέονται απ΄ ευθείας με τα διάφορα αισθητήρια. Στη παρούσα εργασία περιγράφονται δύο τύποι μονάδων: ψηφιακές μονάδες με οκτώ ψηφιακές εξόδους και αναλογικές μονάδες με οκτώ αναλογικές εισόδους, οι οποίες σχεδιάστηκαν και υλοποιήθηκαν στο Εργαστήριο ΦΥΕ του τμήματος Φυσικής του Πανεπ. Ιωαννίνων. Ο σχεδιασμός έγινε με τη βοήθεια του σχεδιαστικού πακέτου CAD OrCAD και η υλοποίηση των τυπωμένων κυκλωμάτων στο εμφανιστήριο τυπωμένων κυκλωμάτων, του Εργαστηρίου Φυσικής Υψηλών Ενέργειών.

7.1 *Γενική περιγραφή του συστήματος ΤΗΛ2*

Σύμφωνα με το πρόγραμμα ΑΣΤΗΛΔΥΠ, η πρότυπη μονάδα ΤΗΛ2 θα υλοποιηθεί σε ένα

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 7.1 . Σχεδιάγραμμα του Αντλιοστασίου Α1

αντλιοστάσιο και θα επικοινωνεί τόσο με τους sensors-actuators, κ.λ.π. του ίδιου του αντλιοστασίου όσο και με την κεντρική μονάδα, όπως φαίνεται και στο διάγραμμα της προηγούμενης σελίδας. Το σύστημα ΤΗΛ2 σχεδιάστηκε με βάση τον ισχυρό μικροελεγκτή 8051 της INTEL. Στην προηγούμενη σελίδα φαίνεται το σχηματικό διάγραμμα του ΤΗΛ2 σ' ένα από τα αντλιοστάσια (αντλιοστάσιο Α1). Η αρχιτεκτονική του συστήματος είναι modular, ώστε να είναι γρήγορος και εύκολος ο εντοπισμός προβλημάτων του συστήματος και η συντήρησή του. Αποτελείται από ένα κουτί με το δίαυλο (back-plane) στο οποίο περιέχονται οι κάρτες : τροφοδοτικού, κεντρικής μονάδας επεξεργασίας (CPU), αναλογικών εισόδων, ψηφιακών εξόδων, διεπαφής RS-485 και διεπαφής RS-232 [15]. Οι κάρτες επικοινωνούν μεταξύ τους μέσω της κοινής διεπαφής στο δίαυλο (back-plane). Το πλήθος και το είδος των καρτών αυτών καθορίζεται από την φύση των σημάτων εξόδου των διαφόρων αισθητήρων (sensors) που θα χρησιμοποιηθούν στα αντλιοστάσια (αναλογικά, ψηφιακά, 4-20mA, RS-485) καθώς επίσης και από τα σήματα εισόδου ή τα πρωτόκολλα των συστημάτων επικοινωνίας με τον κεντρικό σταθμό. Ο δίαυλος δεν περιέχει κανένα κύκλωμα παρά μόνο τη κοινή διεπαφή. Για την πρότυπη μονάδα απαιτείται η σχεδίαση και ανάπτυξη των εξής ηλεκτρονικών καρτών:

7.1.1 *Πλαίσιο με το δίαυλο επικοινωνίας (back-plane)*

Τα χαρακτηριστικά του διάυλου (back-plane) είναι :

- Κάρτα σε δύο εκδόσεις με 6 θέσεις σε αποστάσεις 3cm μεταξύ τους.
- Δυνατότητα τροφοδοσίας με DC τάσεις από εξωτερικό τροφοδοτικό μέσω συνδετήρα (connector) ή από εσωτερικό τροφοδοτικό σε οποιαδήποτε θέση (slot) του πλαισίου.
- Τερματικές αντιστάσεις σε όλα τα ψηφιακά σήματα με δυνατότητα κοπής σε θέσεις πολλαπλάσιες του 2.
- Τοπικοί πυκνωτές σε κάθε θέση και για όλες τις DC τάσεις , για να μπορεί να γίνει αντικατάσταση μιας κάρτας κατά την λειτουργία του βιομηχανικού ελεγκτή χωρίς προβλήματα.
- Γαλβανισμένη η μία πλευρά της κάρτας για αυξημένη προστασία έναντι του θορύβου.

Το pin assignment του back plane και το διάγραμμα χρονισμού (timing diagram) παρατίθονται στο παράρτημα Β.

7.1.2 *Τροφοδοτικό*

Τα χαρακτηριστικά του τροφοδοτικού είναι :

Είσοδοι:

- 220V AC, 50Hz μέσω τριπολικού Αμερικάνικου τύπου ρευματολήπτη για σασί.
- Διακόπτης για την παροχή ισχύος, διπλός ON-OFF με ενσωματωμένο LED.
- Ασφάλεια 300mA.
- Δακτυλοειδής μετασχηματιστής 2x15V/3A.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Έξοδοι:
- +5V/1.5A
- 12V/1A
- 8V/1A (προαιρετικά)
- Μη ρυθμιζόμενη τάση για την ανίχνευση της πτώσης τάσης.

7.1.3 *Κάρτα κεντρικής μονάδας επεξεργασίας (CPU)*

Το σπουδαιότερο τμήμα της κάρτας είναι ο μικροελεγκτής 8051 του οποίου τα χαρακτηριστικά είναι :
- 8-bit CPU, σχεδιασμένη για εφαρμογές ελέγχου.
- Εκτεταμένες δυνατότητες δυαδικής επεξεργασίας (single-bit logic).
- Διευθυνσιοδότηση μνήμης προγράμματος μέχρι 64K.
- Διευθυνσιοδότηση μνήμης δεδομένων μέχρι 64K.
- Μνήμη προγράμματος 4K bytes πάνω στο chip.
- I/O γραμμές δικατευθυντήριες και ξεχωριστά διευθυνσιοδοτούμενες.
- Δύο απαριθμητές/μετρητές των 16-bit.
- Full duplex UART.
- Δομή διακοπών 5-πηγών με επίπεδα προτεραιότητας.
- Ταλαντωτής πάνω στο chip.

7.1.4 *Κάρτα αναλογικών εισόδων (A/I)*

Τα κυριότερα χαρακτηριστικά της κάρτας είναι :
- Επιλογή κωδικού κάρτας με dip-switches.
- 8 αναλογικές είσοδοι των 4-20mA.
- Μετατροπέα αναλογικού σήματος σε ψηφιακό ακριβείας 8 bits (ADC 0809).
- Χρόνος μετατροπής 70 μs.
- Σφάλμα στην γραμμικότητα ±1/2 LSB και ±1 LSB.
- Ξεχωριστή απομόνωση (buffer) κάθε εισόδου.
- Ψηφιακά επιλεγόμενη είσοδος.

7.1.5 *Κάρτα ψηφιακών εξόδων (D/O)*

Τα κυριότερα χαρακτηριστικά της κάρτας είναι :
- Επιλογή κωδικού κάρτας με dip-switches.
- Ικανό ρεύμα εξόδου για την κάλυψη των απαιτήσεων βιομηχανικών εφαρμογών.
- Οπτική ένδειξη ενεργοποίησης της εξόδου.
- Απενεργοποίηση των εξόδων της κάρτας σε περίπτωση εσφαλμένης λειτουργίας της κεντρικής μονάδας επεξεργασίας.
- Οπτική απομόνωση μεταξύ της υπομονάδας οδήγησης και της υπομονάδας εξόδου.
- 8 ψηφιακές έξοδοι των 12 V DC.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

7.1.6 *Κάρτα inerface RS-232*

Η τυποποίηση αυτή προσδιορίζει τη διασύνδεση μεταξύ μιας τερματικής συσκευής (Data Terminal Equipment DTE) και μιας συσκευής επικοινωνιών (Data Circuit - Terminating Equipment DCE). Οι συσκευές DTE μπορεί να είναι υπολογιστές, τερματικά, εκτυπωτές, ενώ DCE είναι τα modem, multiplexer και άλλες συσκευές επικοινωνίας.

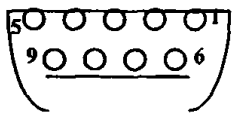
Το μέσο διασύνδεσης στο οποίο λειτουργεί το RS-232 είναι καλώδιο 25 ακροδεκτών.

Σύμφωνα με το πρότυπο RS-232 μια τάση περισσότερο αρνητική από –3 Volt είναι ένα δυαδικό 1 και μία τάση περισσότερο θετική από +4 Volt είναι ένα δυαδικό 0. Επιτρέπονται ρυθμοί μετάδοσης δεδομένων έως 20kb/s, όπως και καλώδια έως 15 μέτρα.

Όταν ο υπολογιστής τεθεί σε λειτουργία θέτει σε λογικό 1 το DTR (Data Terminal Ready). Οταν το modem τεθεί σε λειτουργία θέτει σε λογικό 1 το DSR (Data Set Ready). Οταν το modem ανιχνεύσει μια φέρουσα συχνότητα στην γραμμή, θέτει σε λογικό 1 το CTS (Clear To Send). Το RTS (Request To Send) δηλώνει ότι το τερματικό θέλει να στείλει δεδομένα. Τα δεδομένα μεταδίδονται στο κύκλωμα αποστολής (Transmit) και λαμβάνονται στο κύκλωμα υποδοχής (Receive). Η μέγιστη ταχύτητα είναι 19200 bps. Ο τρόπος μετάδοσης είναι σημείο - με - σημείο (point-to-point) με κοινή επιστροφή. Στο σχήμα 7.2 παραθέτονται οι ακίδες του προτύπου RS-232.

Pin assignmet προτύπου RS232

Αριθμός Pin	Ονομα Σήματος
1	N.N.
2	RxD
3	TxD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	N.C.



Σχήμα 7.2 : Παράθεση ακίδων προτύπου RS-232

7.1.7 *Κάρτα interface RS-485*

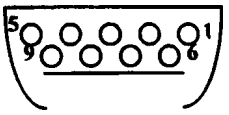
Έχει δυνατότητα πραγματικής σύνδεσης πολλαπλών χρηστών (multidrop) πάνω στην ίδια γραμμή. Αυτό επιτυγχάνεται επειδή το κύκλωμα του εκπομπού είναι tri-state, δηλαδή εκτός από τις δυο λογικές καταστάσεις μπορεί να τεθεί και σε κατάσταση υψηλής αντίστασης. Οι τάσεις της τυποποίησης αυτής είναι +1.5 V για το λογικό 0 και –1.5 V για το λογικό 1. Στο δέκτη αρκούν +200 mV και -200 mV για να είναι

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

αναγνωρίσιμες οι λογικές καταστάσεις. Η μέγιστη ταχύτητα είναι 10 Mbps. Ο τρόπος μετάδοσης είναι διαφορικός, με δυνατότητα διασύνδεσης 32 χρηστών πολαπλών σημείων. Στο σχήμα 7.3 παρατίθονται οι ακίδες του προτύπου RS-485.

Pin Assignment προτύπου RS485

Αριθμός Pin	Ονομα Σήματος
1	+12V DC
2	GND
3	-DATA
4	N.C.
5	RGN
6	+12V DC
7	GND
8	+DATA
9	N.C.



Σχήμα 7.3 : Παράθεση ακίδων προτύπου RS-485

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

8. Η κάρτα αναλογικών εισόδων

8.1 *Γενική περιγραφή*

Μια κάρτα αναλογικών εισόδων χρησιμοποιείται για την ανάγνωση ηλεκτρικών σημάτων από ηλεκτρονικές μονάδες. Η κάρτα αυτή μπορεί να είναι παθητική ή ενεργητική. Με τον όρο παθητική εννοούμε ότι διαβάζει τις εισόδους και τις μεταφέρει στην κεντρική μονάδα επεξεργασίας που βρίσκεται σε άλλη κάρτα, η οποία επεξεργάζεται τα σήματα των εισόδων και ανάλογα αποφασίζει. Η μεταφορά των δεδομένων μπορεί να γίνει παράλληλα, οπότε έχουμε υψηλό ρυθμό μετάδοσης αλλά μόνο για αποστάσεις μικρότερες του μισού μέτρου. Άλλος τρόπος μεταφοράς είναι ο σειριακός, όπου έχουμε χαμηλό ρυθμό μετάδοσης (περίπου 8 φορές μικρότερος από το παράλληλο) αλλά μπορεί να επεκταθεί σε μεγαλύτερες αποστάσεις. Για παράδειγμα με το πρωτόκολλο RS232 (σειριακό) πετυχαίνουμε ασφαλή μεταφορά δεδομένων σε απόσταση μέχρι 15 μέτρων περίπου, ενώ με το πρωτόκολλο RS485 (συμμετρικό σειριακό) πετυχαίνουμε απόσταση μέχρι 3 χιλιομέτρων περίπου. Με το όρο ενεργητική εννοούμε ότι η κάρτα αυτή διαβάζει τις εισόδους, τις επεξεργάζεται η ίδια και ανάλογα αποφασίζει.

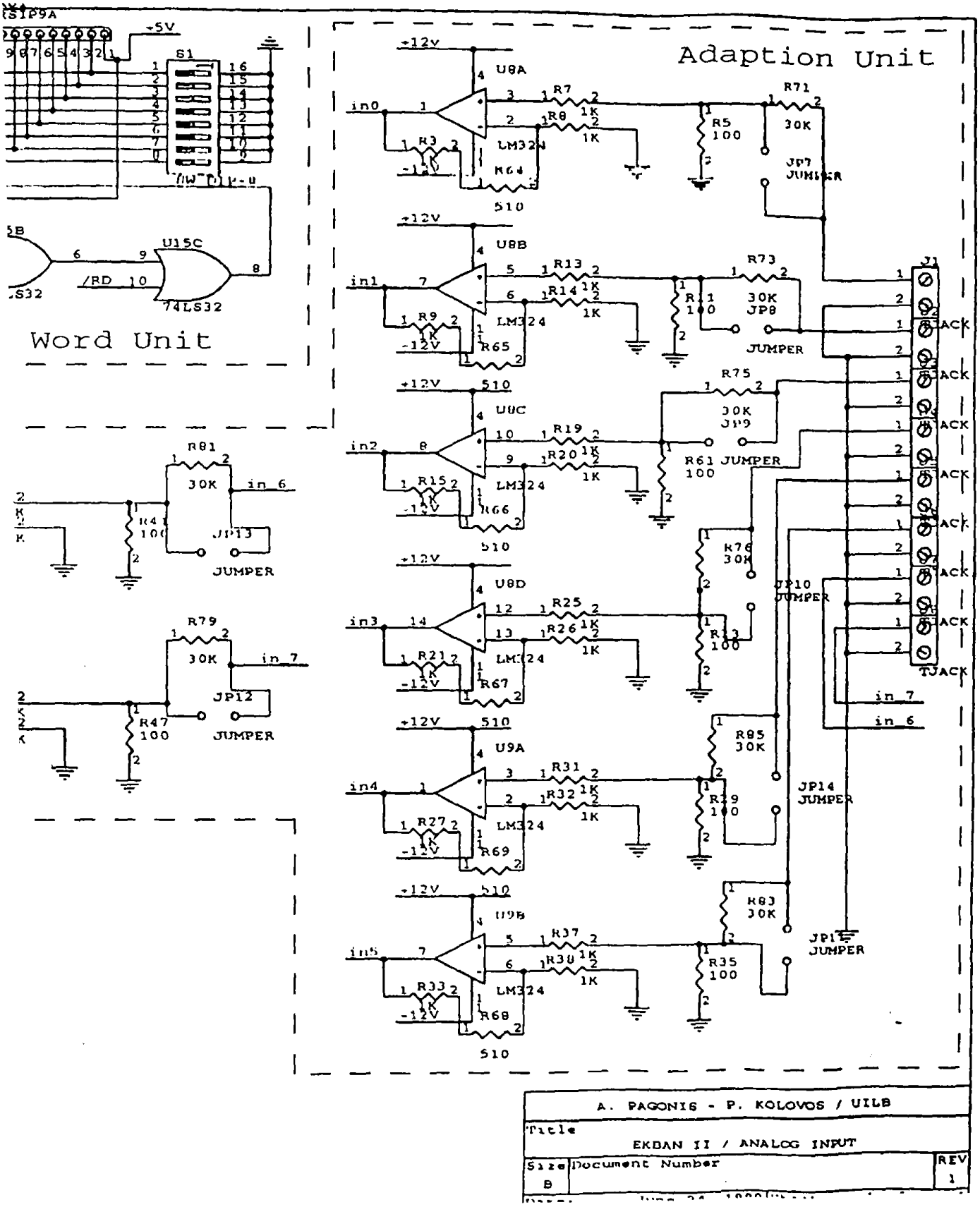
Η κάρτα αναλογικών εισόδων που σχεδιάστηκε στα πλαίσια αυτής της εργασίας είναι παθητική. Χρησιμοποιείται για την ψηφιοποίηση και την μεταφορά των ψηφιοποιημένων σημάτων εισόδου παράλληλα προς την κεντρική μονάδα επεξεργασίας (CPU) [16]. Η ψηφιοποίηση των σημάτων έχει εκτός των άλλων ένα πρόσθετο πλεονέκτημα την απουσία θορύβου κατά την μεταφορά. Η κάρτα έχει 8 αναλογικές εισόδους του βιομηχανικού προτύπου 4-20mA και έξοδο την δειγματοληψία 8-bit της επιλεγόμενης εισόδου.

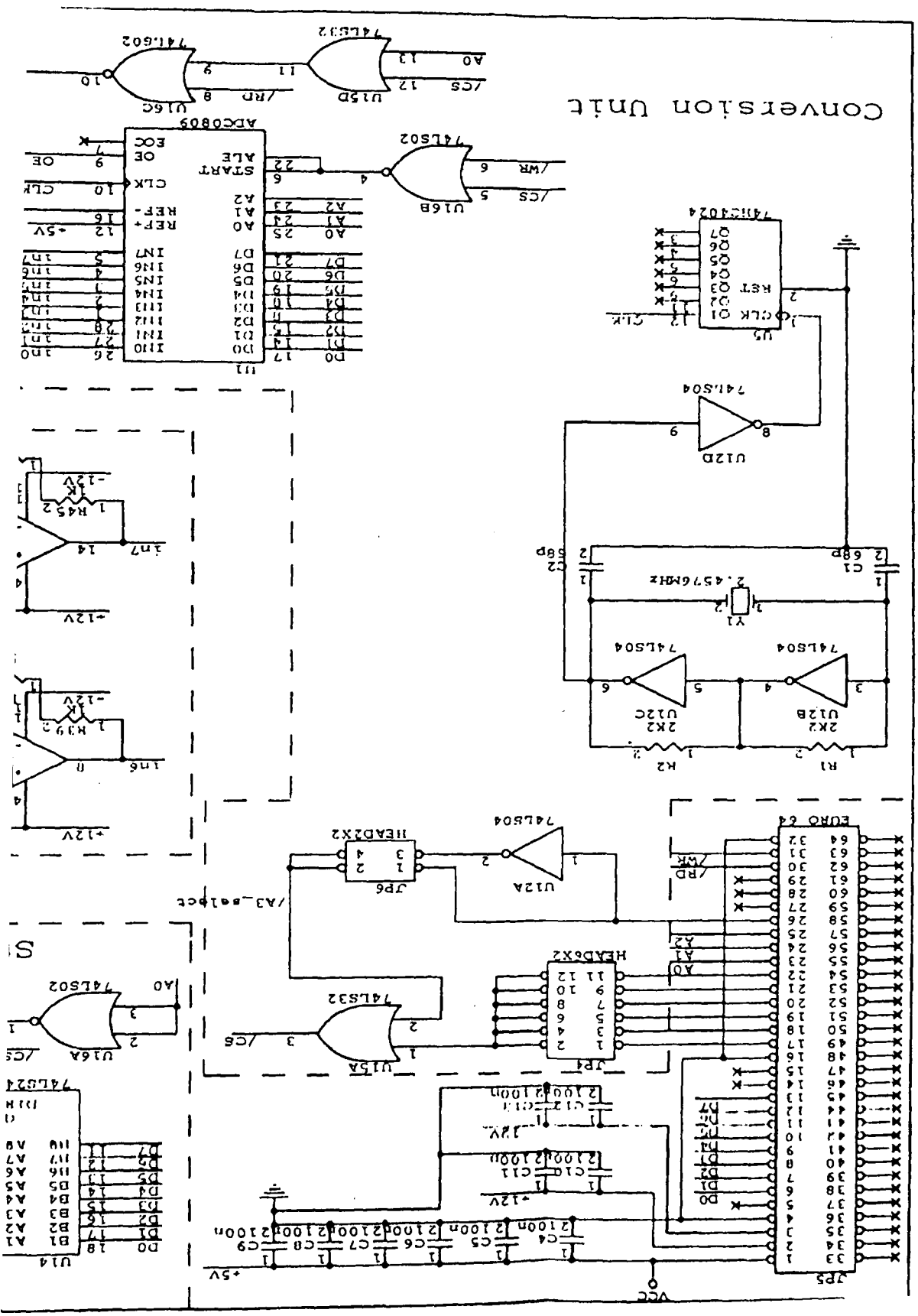
8.2 *Λογικός σχεδιασμός*

Η κάρτα αναλογικών εισόδων βασίζεται σε τρείς επιμέρους υπομονάδες : την υπομονάδα προσαρμογής του σήματος εισόδου 4-20mA, την υπομονάδα απομόνωσης και μετατροπής και την υπομονάδα κωδικού κάρτας. Το λογικό διάγραμμα της κάρτας παρατίθενται στο σχήμα 8.1.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ





Σχήμα 8.1 : Λογικό διάγραμμα της κάρτας αναλογικών εισόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

8.2.1 *Υπομονάδα προσαρμογής σήματος 4-20 mA*

Η υπομονάδα προσαρμογής είναι ουσιαστικά ένας μετατροπέας ρεύματος σε τάση. Συγκεκριμένα έχει σαν σκοπό να προσαρμόσει το εισερχόμενο σήμα το οποίο είναι ρεύμα στην περιοχή 4-20mA σε τάση στην περιοχή 1-5Volts. Όπως παρατηρούμε το κύριο στοιχείο της υπομονάδας είναι ένας τελεστικός ενισχυτής.

Ο τελεστικός ενισχυτής (operational amplifier) οφείλει την ονομασία του στο πλήθος των λειτουργιών που μπορεί να εκτελέσει. Μερικές από τις λειτουργίες αυτές είναι η ενίσχυση, η διαφόριση, η ολοκλήρωση, η πρόσθεση, η αφαίρεση και πολλές άλλες. Γενικά είναι ένας ενισχυτής με άμεση σύζευξη και υψηλή απολαβή. Έχει μεγάλη αντίσταση εισόδου και μικρή αντίσταση εξόδου, ενώ από άποψη συχνοτήτων μπορεί να ενισχύει ικανοποιητικά από συνεχή σήματα έως πολύ υψηλές συχνότητες. Οι σημαντικότερες από τις εφαρμογές του τελεστικού ανισχυτή είναι:

- ενισχυτής και αναστροφέας σήματος (inverting amplifier)
- ενισχυτής χωρίς αναστροφή σήματος (non-inverting amplifier)
- διαφορικός ενισχυτής
- ενισχυτής-ή υποβιβαστής- και αθροιστής (summing amplifier)
- συγκριτής τάσης (Voltage comparator)
- ενεργό φίλτρο (active filter)

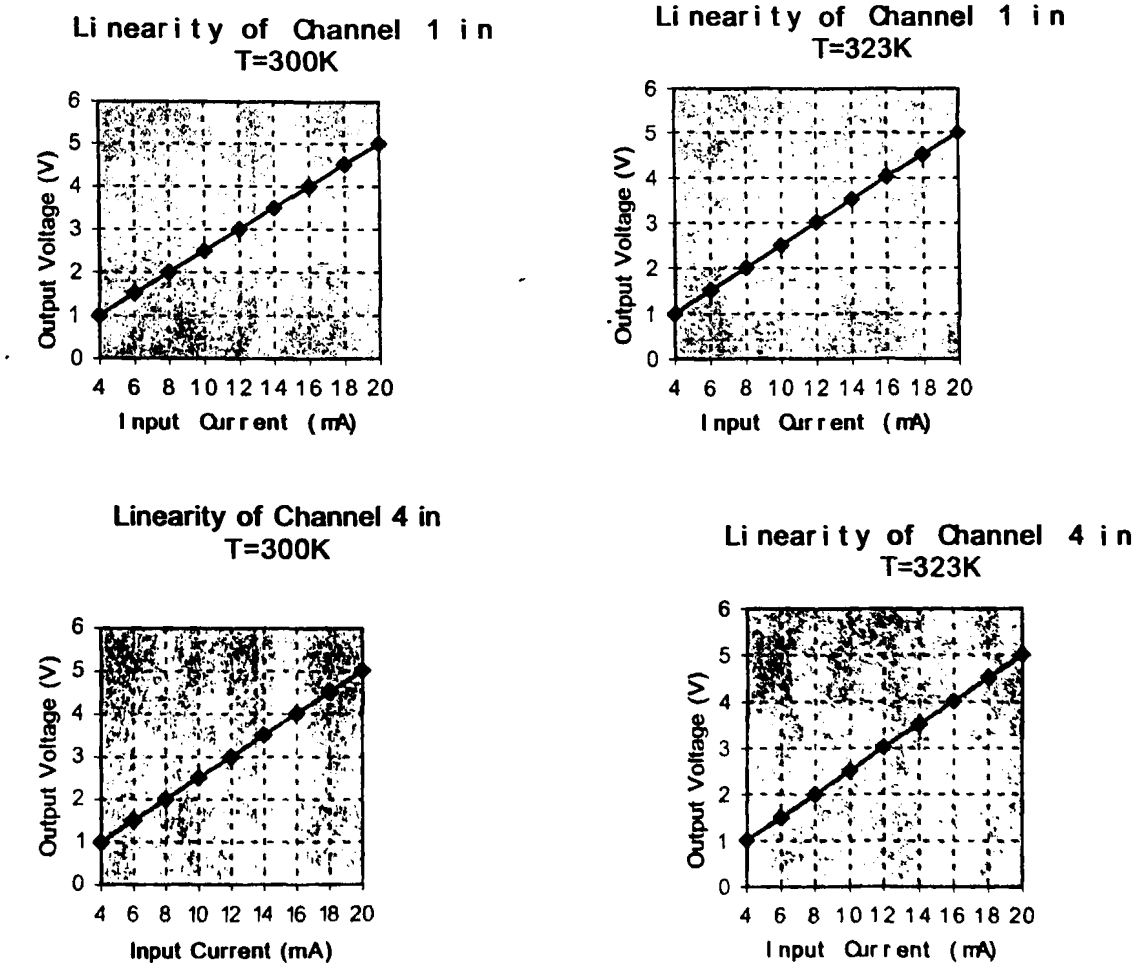
Στην περίπτωσή μας χρησιμοποιούμε τον τελεστικό ενισχυτή σαν διαφορικό, ο οποίος ενισχύει την διαφορά μεταξύ δύο σημάτων. Το κύριο χαρακτηστικό του διαφορικού ενισχυτή είναι ο λόγος απόρριψης κοινού-σήματος (CMRR), που ορίζεται σαν το λόγο του κέρδους τάσης εφαρμόζοντας διαφορικό σήμα προς το αντίστοιχο κέρδος εφαρμόζοντας κοινό σήμα, δηλ. το ίδιο σήμα και στις δύο εισόδους του. Προφανώς είναι επιθυμητό να έχουμε υψηλή τάση στην έξοδο εφαρμόζοντας διαφορικό σήμα ενώ στην περίπτωση που εφαρμόζεται κοινό-σήμα η τάση εξόδου να ισούται με το μηδέν.

Ο τελεστικός ενισχυτής που χρησιμοποιήθηκε είναι το ολοκληρωμένο LM324 [17] (στο σχήμα 8.1, τα ολοκληρωμένα U8 και U9. Το σήμα εισόδου της κάρτας, ουσιαστικά είναι μια πηγή ρεύματος 4-20mA, η οποία ενώνεται με την μη-ανάστροφη είσοδο του τελεστικού ενισχυτή μέσω μιας αντίστασης 1K, μιας αντίστασης 100Ω ως προς την γη και μιας επιλογής που μπορεί μέσω ενός διακόπτη να εισάγει μια αντίσταση 30K σε σειρά με το σήμα εισόδου. Αυτή η επιλογή υπάρχει για προστατευτικούς λόγους διότι μερικοί αισθητήρες 4-20mA λειτουργούν με τάση +24V ενώ άλλοι με μικρότερη. Η αντίσταση των 100Ω παίζει τον ρόλο του φόρτου για το σήμα εισόδου, συνεπώς εμφανίζεται μια τάση 0.4V για ρεύμα εισόδου 4mA πάνω στην αντίσταση ως προς την γη. Η αντίσταση ανάδρασης 1.5K (1K και 510 σε σειρά) μαζί με την αντίσταση 1K που ενώνεται με την αναστρέφουσα είσοδο του τελεστικού ρυθμίζει την απολαβή του τελεστικού ώστε για ρεύμα εισόδου 20mA να έχουμε τάση εξόδου 5V. Οι εργαστηριακές τιμές τάσης που λάβαμε για 4mA και για 20mA ήταν 0.99V και 4.98V με ακρίβεια 0.02V αντίστοιχα. Η ακρίβεια προέρχεται από την κβάντιση του μετατροπέα των 8-bits. Επίσης έγιναν μετρήσεις γραμμικότητας σε δύο

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

I_{in} (mA)	V_{ch1} (Volts)		V_{ch4} (Volts)	
	T=300°K	T=323°K	T=300°K	T=323°K
4.0	0.991	0.994	0.997	1.003
6.0	1.513	1.508	1.511	1.496
8.0	1.998	2.007	2.004	1.999
10.0	2.499	2.499	2.508	2.512
12.0	2.999	3.018	2.999	3.011
14.0	3.504	3.517	3.505	3.518
16.0	4.002	4.023	4.009	4.015
18.0	4.520	4.515	4.512	4.518
20.0	5.000	5.001	5.004	5.009

Πίνακας 8.1: Εργαστηριακές μετρήσεις της τάσης εξόδου για τα κανάλια 1 και 4 για ρεύμα εισόδου από 4 – 20 mA, σε θερμοκρασίες 300°K και 323°K.



Σχήμα 8.2: Τάση εξόδου των καναλιών 1 και 4 συναρτήσει του ρεύματος εισόδου για θερμοκρασίες 300°K και 323°K.

ξεχωριστά κανάλια τόσο σε θερμοκρασία δωματίου (T=300°K), όσο και σε θερμοκρασία T=323°K. Οι μετρήσεις φαίνονται στον πίνακα 8.1. Στο σχήμα 8.2 παρατίθενται οι γραφικές παραστάσεις των μετρήσεων, υποδεικνύοντας την

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ ΒΙΒΛΙΟΘΗΚΗ

γραμμικότητα της υπομονάδας προσαρμογής. Εδώ πρέπει να σημειωθεί ότι οι μετρήσεις πάρθηκαν μετά από μισή ώρα λειτουργία.
Με βάση τις τιμές του πίνακα 8.1 και με την θεωρία ελαχίστων τετραγώνων μπορεί να υπολογιστεί η κλήση των ευθειών και το σφάλμα της. Οι θεωρητικοί τύποι είναι οι εξής:
Για ευθεία $f = ax + b$ ισχύει : $a=(n\Sigma(x_i f_i)-\Sigma x_i \Sigma f_i)/(n\Sigma(x_i)^2-(\Sigma x_i)^2)$, $b=(\Sigma(x_i)^2\Sigma x_i\Sigma(x_i f_i))/(n\Sigma(x_i)^2-(\Sigma x_i)^2)$, $\sigma(a)=((\Sigma(\delta f_i)^2)/(n-2))^{0.5}(n/(n\Sigma(x_i)^2-(\Sigma x_i)^2))^{0.5}$,$\delta f_i=f_i-(ax_i+b)$, όπου f_i, x_i είναι η τάση εξόδου, το ρεύμα εισόδου αντίστοιχα.
Τα συμπεράσματα λοιπόν είναι:
Για το κανάλι 1 σε θερμ. T=300°K η κλήση είναι a=0.2505 με σ(a)=±0.0023.
Για το κανάλι 1 σε θερμ. T=323°K η κλήση είναι a=0.2508 με σ(a)=±0.0006.
Για το κανάλι 4 σε θερμ. T=300°K η κλήση είναι a=0.2503 με σ(a)=±0.0003.
Για το κανάλι 4 σε θερμ. T=323°K η κλήση είναι a=0.2511 με σ(a)=±0.0003.

Επομένως σε ένα βιομηχανικό περιβάλλον που συνήθως οι συνθήκες είναι ιδαίτερα αντίξοες, η θερμοκρασία παίζει ένα σημαντικό ρόλο στην συμπεριφορά ηλεκτρονικών καρτών. Συνεπώς, κάθε ηλεκτρονική κάρτα, η οποία προορίζεται για λειτουργία σε βιομηχανικό χώρο πρέπει να ελέγχεται πριν παραθοδεί. Από τις παραπάνω γραφικές παραστάσεις η διατήρηση της γραμμικότητας των τελεστικών ενισχυτών σε υψηλές θερμοκρασίες.

8.2.2 *Υπομονάδα απομόνωσης και μετατροπής*

Η υπομονάδα απομόνωσης και μετατροπής χρησιμοποιείται για την μετατροπή του αναλογικού σήματος (αναλογική έξοδος της υπομονάδας προσαρμογής) σε ψηφιακό και επιπλέον να απομονώνει ξεχωριστά κάθε αναλογική είσοδο. Όπως παρατηρούμε, το κύριο στοιχείο της υπομονάδας είναι το ολοκληρωμένο ADC0809 [18]. Το ολοκληρωμένο αυτό είναι μια μονολιθική CMOS διάταξη με ένα 8-bit μετατροπέα αναλογικού σήματος σε ψηφιακό, ένα πολυπλέκτη 8 καναλιών και μια λογική ελέγχου συμβατή με μικροεπεξεργαστή. Ο μετατροπέας χαρακτηρίζεται από ένα 256R διαιρέτη τάσης με αναλογικό "δέντρο" διακοπτών. Η προσπέλαση σε κάθε 8-single-ended αναλογικό σήμα γίνεται άμεσα με τον πολυπλέκτη 8-καναλιών. Εσωτερικά το ολοκληρωμένο διαθέτει μανδαλωτή και αποκωδικοποιητή της πολυπλεγμένης διεύθυνσης της αναλογικής εισόδου (ξεχωριστή απομόνωση για κάθε αναλογική είσοδο) και μανδαλωτή TTL τριών καταστάσεων για την ψηφιοποιημένη έξοδο. Τέλος, η περιοχή αναλογικής εισόδου είναι από 0V έως 5V. Ακολούθως παρατίθεται η λειτουργική περιγραφή του ADC0809: Κατ'αρχήν στέλνεται η 3-bit διεύθυνση για την επιλογή της αναλογικής εισόδου και ύστερα με ένα θετικό στενό παλμό (start) μανδαλώνεται η 3-bit διεύθυνση που σημαίνει ότι ειδοποιείται ο μετατροπέας να αρχίσει την μετατροπή. Ο στενός αυτός παλμός start δημιουργείται από την λογική πράξη NOR των σημάτων $\overline{CS}$ και $\overline{WR}$. Το σήμα $\overline{CS}$ προέρχεται από την λογική πράξη OR των σημάτων $\overline{CS0}$ - $\overline{CS5}$ μέσω του επιλογέα JP4 και της διεύθυνσης A3 μέσω του επιλογέα JP6. Είναι φυσικό ότι ο χρόνος μετατροπής εξαρτάται από την συχνότητα του ρολογιού. Η συχνότητα ρολογιού παράγεται από εξωτερικό κύκλωμα και είναι 1.2 MHz, όπως φαίνεται και στο λογικό διάγραμμα. Ο

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

χρόνος μετατροπής γι' αυτή τη συχνότητα έχει μετρηθεί εργαστηριακά και είναι 53μs. Μετά από αυτό το χρονικό διάστημα στέλνεται ένας θετικός στενός παλμός (output enable), ο οποίος ενεργοποιεί τον μανδαλωτή για να εξάγει την ψηφιοποιημένη μορφή του αναλογικού σήματος. Ο παραπάνω παλμός δημιουργείται από την λογική πράξη NOR των σημάτων $\overline{RD}$ και το αποτέλεσμα της λογικής πράξης OR των σημάτων $\overline{CS}$ και της διευθυνσης A0. Η ακρίβεια στη γραμμικότητα του μετατροπέα είναι ±3/2 LSB. Η υπομονάδα αυτή έχει σαν σήμα εισόδου τις εξόδους της υπομονάδας προσαρμογής, τις διευθύνσεις A0-A3 από τη κοινή διεπαφή (interface) των καρτών (back plane), καθώς επίσης και τα σήματα ελέγχου start, output enable. Έξοδός της είναι η ψηφιακή μορφή της επιλεγόμενης αναλογικής εισόδου 4-20mA, την οποία θα διαβάσει η κεντρική μονάδα επεξεργασίας.

8.2.3 *Υπομονάδα κωδικού κάρτας*

Η υπομονάδα κωδικού κάρτας διαθέτει ένα dip-switch μέσω του οποίου επιλέγεται μηχανικά ο 8-bit ψηφιακός αριθμός. Ο ρόλος αυτής της υπομονάδας είναι η αναγνώριση της κάρτας από την κεντρική μονάδα επεξεργασίας. Προφανώς κάθε κάρτα έχει τον δικό της κωδικό αριθμό, ταυτότητα της κάρτας.

Το πρωτόκολλο επικοινωνίας μεταξύ της κάρτας αναλογικών εισόδων και της κεντρικής μονάδας επεξεργασίας μέσω του back plane παρατίθενται στο πίνακα 8.2.

MODE	ADDRESS					SELECTION
	A0	A1	A2	A3	$\overline{CS}$	
$\overline{WR}$	0	0	0	*	**	Channel 1
$\overline{WR}$	1	0	0	*	**	Channel 2
$\overline{WR}$	0	1	0	*	**	Channel 3
$\overline{WR}$	1	1	0	*	**	Channel 4
$\overline{WR}$	0	0	1	*	**	Channel 5
$\overline{WR}$	1	0	1	*	**	Channel 6
$\overline{WR}$	0	1	1	*	**	Channel 7
$\overline{WR}$	1	1	1	*	**	Channel 8
$\overline{RD}$	1	X	X	*	**	Status Word
$\overline{RD}$	0	X	X	*	**	Read Selected Channel

Πίνακας 8.2: Διευκρίνηση του πρωτοκόλλου επικοινωνίας μεταξύ της κάρτας αναλογικών εισόδων με την κεντρική μονάδα επεξεργασίας

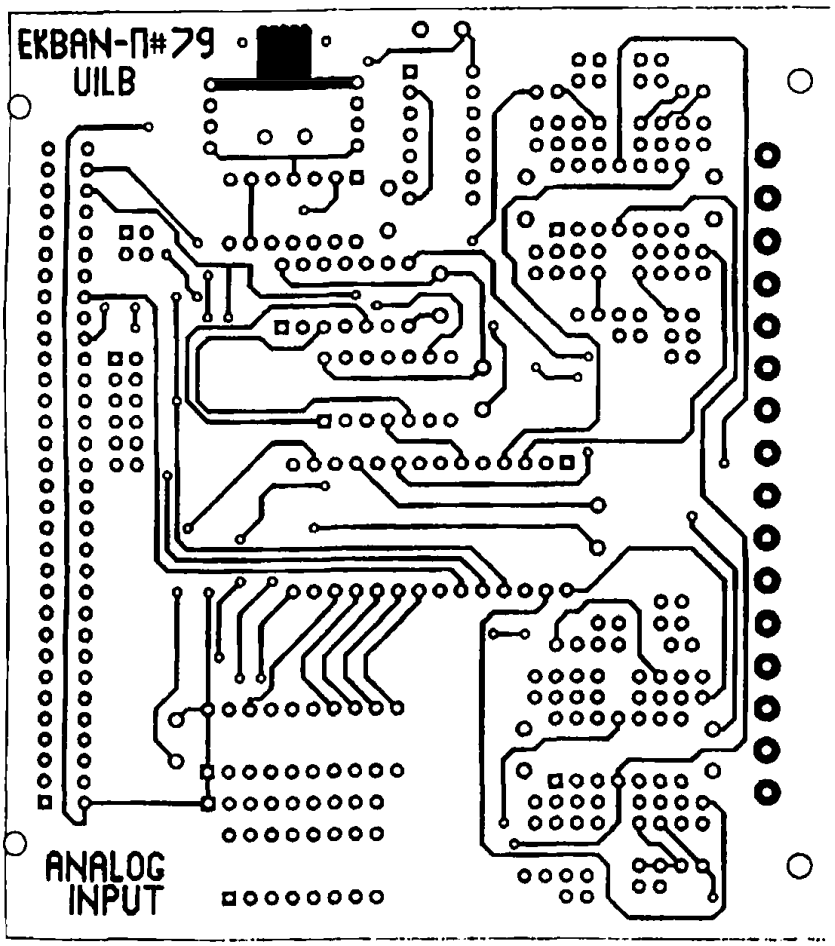
X : Don't care

* : Η κατάσταση της διεύθυνσης A3 εξαρτάται από την θέση του βραχυκυκλωτήρα στον jumper JP6, σχήμα 2. Συγκεκριμένα ενώνοντας τους ακροδέκτες 3 και 4 του JP6 επιλέγουμε A3='1', ενώ με τους ακροδέκτες 1 και 2 επιλέγουμε A3='0'.

** : Η επιλογή του card select εξαρτάται από τη θέση του βραχυκυκλωτήρα του jumper JP4. Συγκεκριμένα ενώνοντας τους ακροδέκτες 11 και 12 επιλέγουμε το $\overline{CS5}$, τα 9 και 10 το $\overline{CS4}$, και ούτω καθεξής μέχρι τα 1 και 2 για το $\overline{CS0}$.

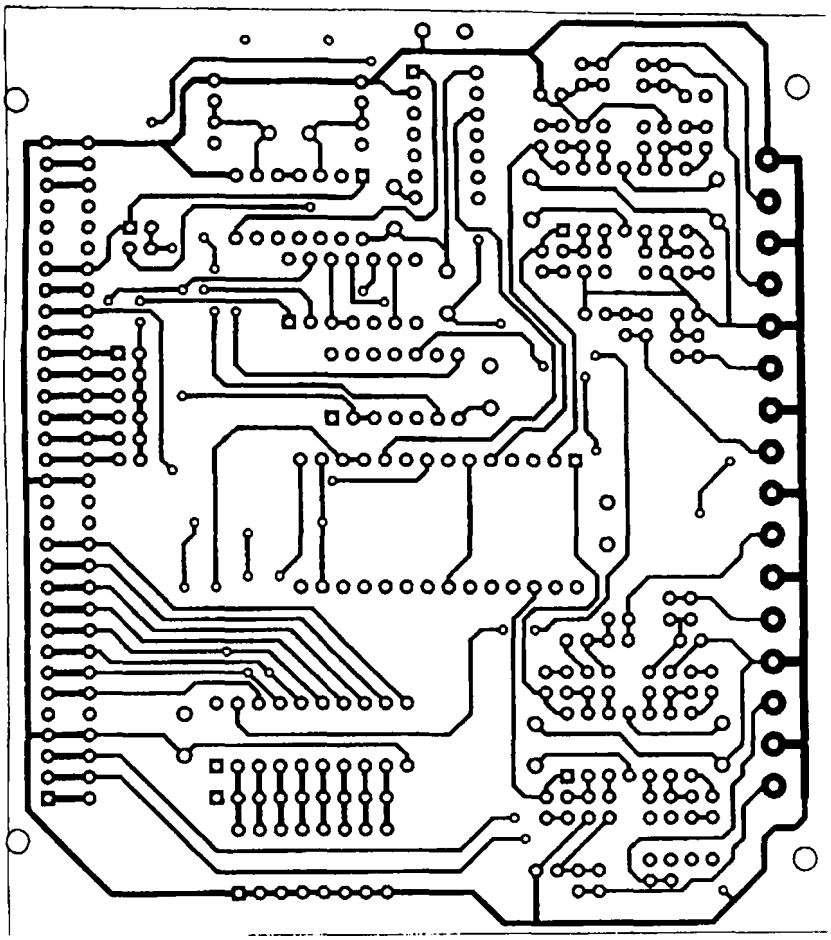
8.3 *Φυσικός σχεδιασμός (Τυπωμένο κύκλωμα)*

Ο φυσικός σχεδιασμός της κάρτας υλοποιήθηκε σε τυπωμένο κύκλωμα δύο όψεων, οι οποίες δίδονται στα σχήματα 8.3 και 8.4, όψη των εξαρτημάτων και όψη κολλήσεων αντίστοιχα. Στο σχήμα 8.5 φαίνεται το τοπογραφικό των εξαρτημάτων στο φυσικό επίπεδο. Οι διαστάσεις της κάρτας είναι (10.8 × 10.1) cm^2.

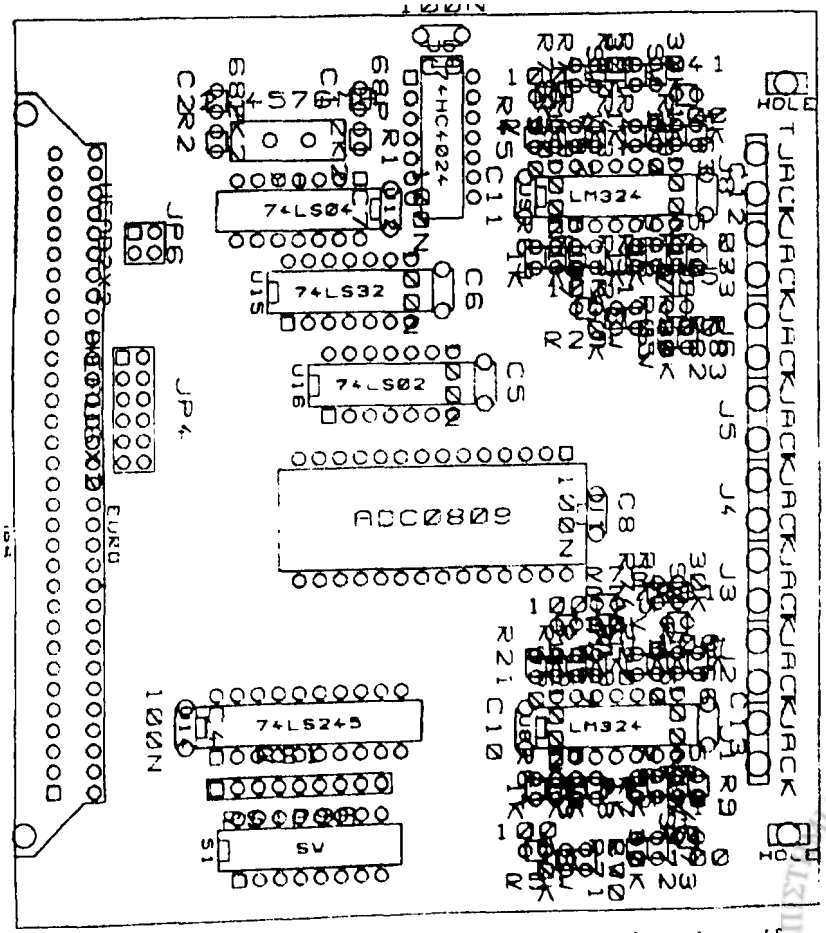


Σχήμα 8.3: Όψη των εξαρτημάτων (Component Side) της κάρτας αναλογικών εισόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 8.4: Όψη των κολλήσεων (Solder Side) της κάρτας αναλογικών εισόδων

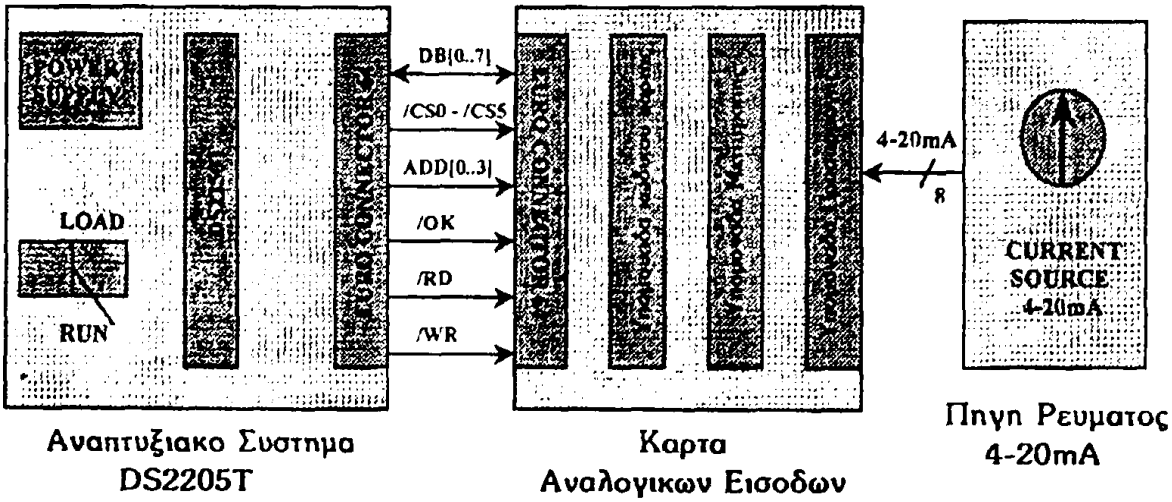


Σχήμα 8.5: Τοπολογία των εξαρτημάτων της κάρτας αναλογικών εισόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

8.4 *Έλεγχος καλής λειτουργίας*

Μετά την υλοποίηση του τυπωμένου κυκλώματος έπρεπε να ελεγχθεί η κάρτα, για πιθανότητα βραχυκυκλώματος ή ψυχρής κόλλησης και να εξομοιωθεί η παραγωγή σημάτων με τον χρονισμό τους του back plane (μέσω euro connector 64). Ύστερα από τον επιτυχή έλεγχο της κάρτας για βραχυκυκλώματα και ψυχρές κολλήσεις, η εξομοίωση του χρονισμού των σημάτων έγινε με την βοήθεια ενός αναπτυξιακού συστήματος με βάση τον μικροελεγκτή DS2250T της DALLAS SEMICONDUCTOR, ο οποίος είναι απολύτως συμβατός με τον 8051 της INTEL [19]. Στο σχήμα 8.6 φαίνεται το σχηματικό διάγραμμα διασύνδεσης του αναπτυξιακού με την κάρτα αναλογικών εισόδων.



Σχήμα 8.6 : Διάγραμμα διασύνδεσης αναπτυξιακού συστήματος DS2250T και κάρτας αναλογικών εισόδων

Η γλώσσα assembly που χρησιμοποιεί ο μικροελεγκτής DS2250T είναι πλήρως συμβατή με αυτή του μικροελεγκτή 8051. Το λογισμικό που χρησιμοποιήθηκε παρατίθεται στο παράρτημα Γ.

Στο πρώτο τμήμα (τμήμα A) του προγράμματος στέλνονται πρώτα τα σήματα $\overline{CS0}$ - $\overline{CS5}$, $\overline{OK}$, A0=1, A1 = A2 = X (Do not care) για να επιλέξουμε την διεύθυνση που αντιστοιχεί στο Status Word και μετά θέτοντας το σήμα $\overline{RD}$ σε λογικό μηδέν, διαβάζουμε την ψηφιακή λέξη η οποία πλέον βρίσκεται στο καταχωρητή ACC του μικροελεγκτή. Στην συνέχεια μεταφέρουμε τα περιεχόμενα του ACC στην πόρτα P0 όπου εμφανίζεται η ψηφιακή λέξη σε δυαδική μορφή μέσω μιας οθόνης από leds.

Στο δεύτερο τμήμα (τμήμα B) του προγράμματος στέλνονται πάλι τα σήματα $\overline{CS0}$ - $\overline{CS5}$, $\overline{OK}$ και A[0..2] = 00 για να επιλέξουμε το πρώτο αναλογικό κανάλι, μετά θέτοντας το σήμα $\overline{WR}$ σε λογικό μηδέν για την ενεργοποίηση της διαδικασίας μετατροπής. Στην συνέχεια περιμένουμε για χρονικό διάστημα 53μs, που είναι το χρονικό διάστημα που απαιτείται για την διαδικασία της μετατροπής. Εδώ πρέπει να επισημανθεί, ότι δεν είναι απαραίτητο να χρησιμοποιηθεί κύκλωμα για την ανίχνευση ολοκλήρωσης της μετατροπής (βλέπε παράρτημα, σήμα EOC), το οποίο χρησιμοποιείται συνήθως στην περίπτωση που η διάταξη μετατροπής δεν ελέγχεται

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

από προγραματιζόμενη μονάδα και επιπλέον καταλαμβάνει χώρο "άσκοπα". Μετά την χρονική αυτή καθυστέρηση στέλνεται η διεύθυνση A0 = 0 με την βοήθεια του σήματος $\overline{RD}$ αρνητικής ενεργοποίησης για την εισαγωγή της ψηφιοποιημένης τιμής του επιλεγμένου αναλογικού καναλιού στον καταχωρητή ACC. Και τέλος, το περιεχόμενο του ACC μεταφέρεται στην οθόνη από leds για την εμφάνισή του μέσω της πόρτας P0.

9. Η κάρτα ψηφιακών εξόδων

9.1 *Γενική περιγραφή*

Η κάρτα ψηφιακών εξόδων είναι τελείως παθητική και χρησιμοποιείται για την παροχή ικανού ρεύματος εξόδου που απαιτείται σε ένα βιομηχανικό χώρο [20]. Αυτή διαθέτει 8 ψηφιακές εξόδους των 12 V DC που παρέχεται από το τροφοδοτικό του ΤΗΛ2 ή 24V DC χρησιμοποιώντας εξωτερική τροφοδοσία για την οδήγηση των ρελέ [16]. Η μέγιστη δυνατότητα παροχής ισχύος ανά κανάλι της κάρτας είναι 18W, σύμφωνα με τα χαρακτηριστικά των τρανζίστορ εξόδου.

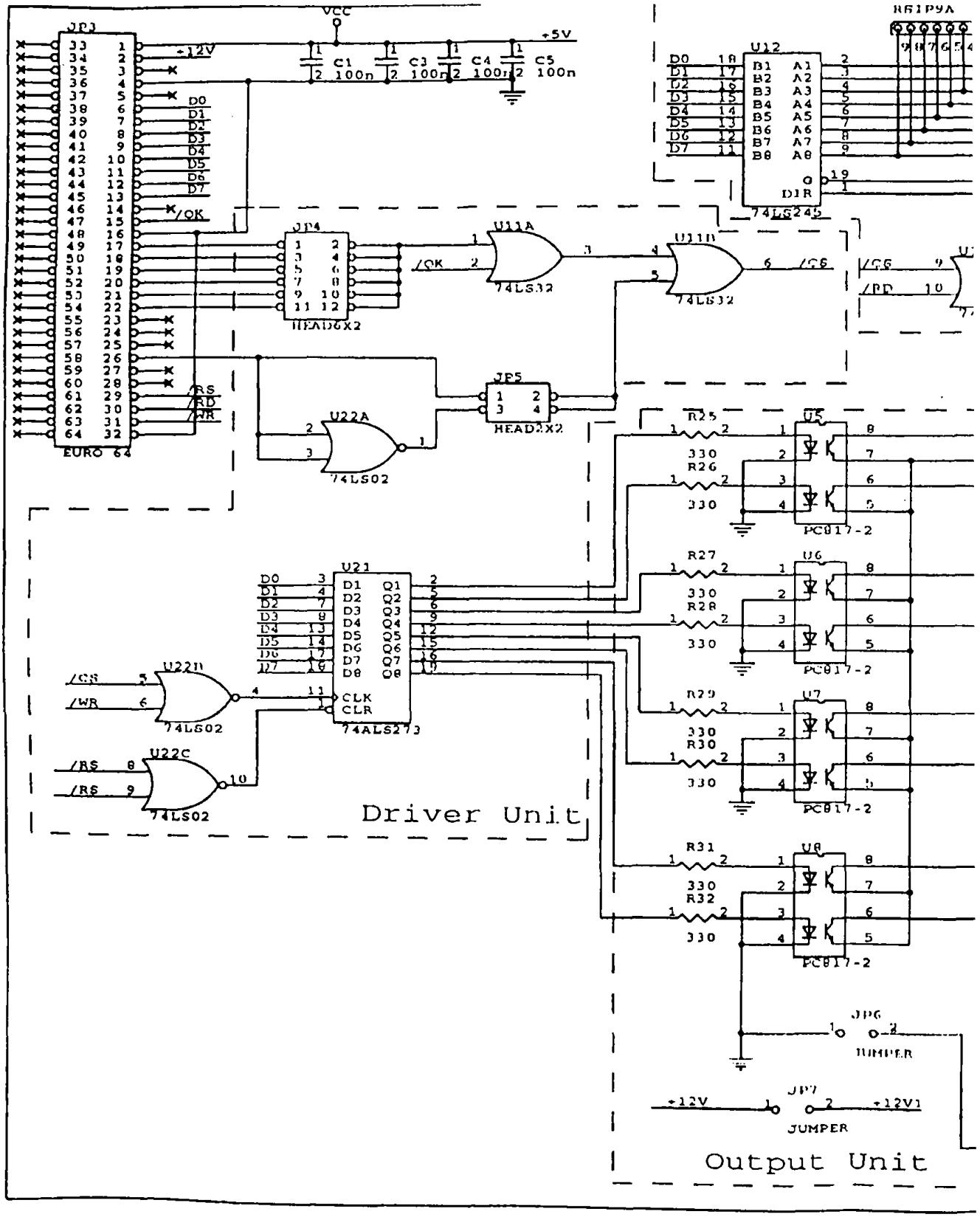
9.2 *Λογικός σχεδιασμός*

Η κάρτα ψηφιακών εξόδων βασίζεται σε τρεις επιμέρους υπομονάδες : την υπομονάδα εξόδου, την υπομονάδα οδήγησης και την υπομονάδα κωδικού κάρτας. Το λογικό διάγραμμα της κάρτας φαίνεται στο σχήμα 9.1.

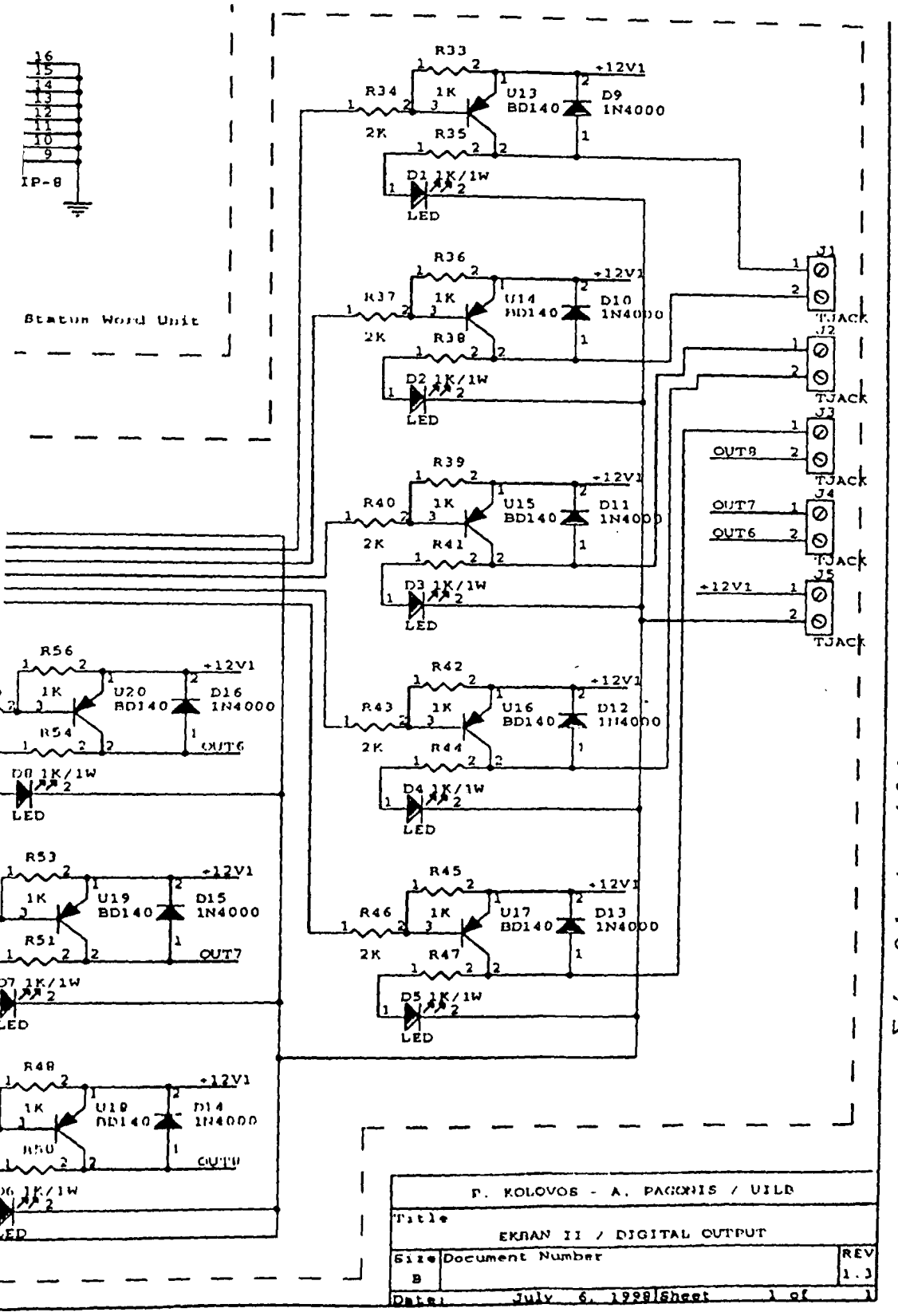
9.2.1 *Υπομονάδα εξόδου*

Η υπομονάδα εξόδου χρησιμοποιείται για την παροχή ικανού ρεύματος εξόδου που απαιτείται σε ένα βιομηχανικό χώρο. Όπως παρατηρούμε στο σχήμα 9.1, βασίζεται ουσιαστικά σε ένα τρανζίστορ ισχύος. Επομένως ελέγχοντας την τάση της βάσης του τρανζίστορ παρέχουμε στο συλλέκτη του τρανζίστορ την τάση των 12 V ή 24V με το απαιτούμενο ρεύμα. Πιο συγκεκριμένα θέτοντας τάση 0 V στην βάση του τρανζίστορ τότε στην έξοδο παίρνουμε 0 V ενώ θέτοντας τάση λίγο μεγαλύτερη από την τάση αγωγής της επαφής p-n

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 9.1 : Λογικό διάγραμμα της κάρτας ψηφιακών εξόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

(περίπου 0.7V) παίρνουμε στην έξοδο 12 V ή 24V. Ένα άλλο σημείο που πρέπει να παρατηρήσουμε είναι ότι το τρανζίστορ οδηγείται από ένα οπτοζεύκτη χρησιμοποιώντας έτσι την τεχνολογία "isolated ground" σύμφωνα με τα βιομηχανικά πρότυπα, επιτυγχάνοντας με οπτικό τρόπο, απομόνωση από την υπόλοιπη κάρτα. Τέλος, σημειώνουμε ότι υπάρχει οπτική ένδειξη της ενεργοποιούμενης εξόδου με αντίστοιχο LED.

9.2.2 *Υπομονάδα οδήγησης*

Η υπομονάδα οδήγησης χρησιμοιείται για την σωστή μεταφορά των δεδομένων εισόδου της κάρτας προς την υπομονάδα εξόδου. Όπως φαίνεται και στο σχήμα 9.1, η υπομονάδα αυτή αποτελείται ουσιαστικά από ένα μανδαλωτή με βάση το οκτώ με σκανδαλισμό ακμής. Τα δεδομένα εισόδου έρχονται από το κοινό δίαυλο των καρτών (euro connector 64) και οδηγούνται προς την υπομονάδα εξόδου. Το σήμα ενεργοποίησης εξασφαλίζει την ορθή μεταφορά των δεδομένων, το οποίο προέρχεται από την λογική πράξη NOR των σημάτων $\overline{WR}$ και $\overline{CS}$. Το σήμα $\overline{CS}$ προέρχεται από την λογική πράξη OR των σημάτων της διεύθυνσης A3 μέσω του επιλογέα JP5 και το αποτέλεσμα της λογικής πράξης OR των σημάτων $\overline{OK}$ και $\overline{CS0}$ - $\overline{CS5}$ μέσω του επιλογέα JP4. Με τον έλεγχο της κατάστασης του σήματος $\overline{OK}$ εξασφαλίζουμε ένα επιπλέον χαρακτηριστικό της υπομονάδας την απενεργοποίηση των εξόδων σε περίπτωση εσφαλμένης λειτουργίας της κεντρικής μονάδας επεξεργασίας. Τέλος, ελέγχοντας την κατάσταση του σήματος $\overline{RS}$ μηδενίζονται οι έξοδοι κατά την εκκίνηση του συστήματος.

9.2.3 *Υπομονάδα κωδικού κάρτας*

Η υπομονάδα κωδικού κάρτας διαθέτει ένα dip-switch μέσω του οποίου επιλέγεται μηχανικά ο 8-bit ψηφιακός αριθμός. Ο ρόλος αυτής της υπομονάδας είναι η αναγνώριση της κάρτας από την κεντρική μονάδα επεξεργασίας (προφανώς κάθε κάρτα θα έχει τον δικό της κωδικό αριθμό).

Το πρωτόκολλο επικοινωνίας μεταξύ της κάρτας αναλογικών εισόδων και της κεντρικής μονάδας επεξεργασίας μέσω του κοινού διάυλου παρατίθενται στο πίνακα 9.1.

MODE	ADDRESS					SELECTION
	A0	A1	A2	A3	$\overline{CS}$	
$\overline{WR}$	X	X	X	*	**	Digital Data Output
$\overline{RD}$	X	X	X	*	**	Status Word

Πίνακας 9.1: Διευκρίνηση του πρωτοκόλλου επικοινωνίας μεταξύ της κάρτας ψηφιακών εξόδων με την κεντρική μονάδα επεξεργασίας

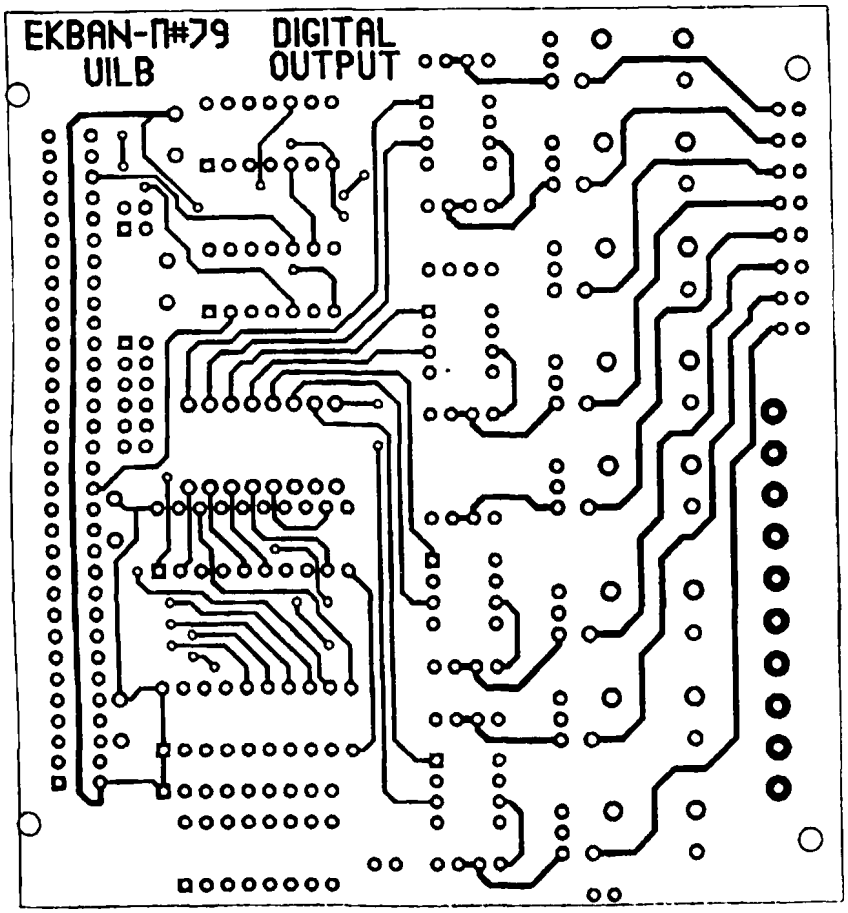
X : Don't care

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

* : Η κατάσταση της διεύθυνσης A3 εξαρτάται από την θέση του βραχυκυκλωτήρα στον jumper JP6, σχήμα 6. Συγκεκριμένα ενώνοντας τους ακροδέκτες 3 και 4 του JP6 επιλέγουμε A3='1', ενώ με τους ακροδέκτες 1 και 2 επιλέγουμε A3='0'.
** : Η επιλογή του card select εξαρτάται από τη θέση του βραχυκυκλωτήρα του jumper JP4. Συγκεκριμένα ενώνοντας τους ακροδέκτες 11 και 12 επιλέγουμε το $\overline{CS5}$, τα 9 και 10 το $\overline{CS4}$ και ούτω καθεξής μέχρι τα 1 και 2 το $\overline{CS0}$.

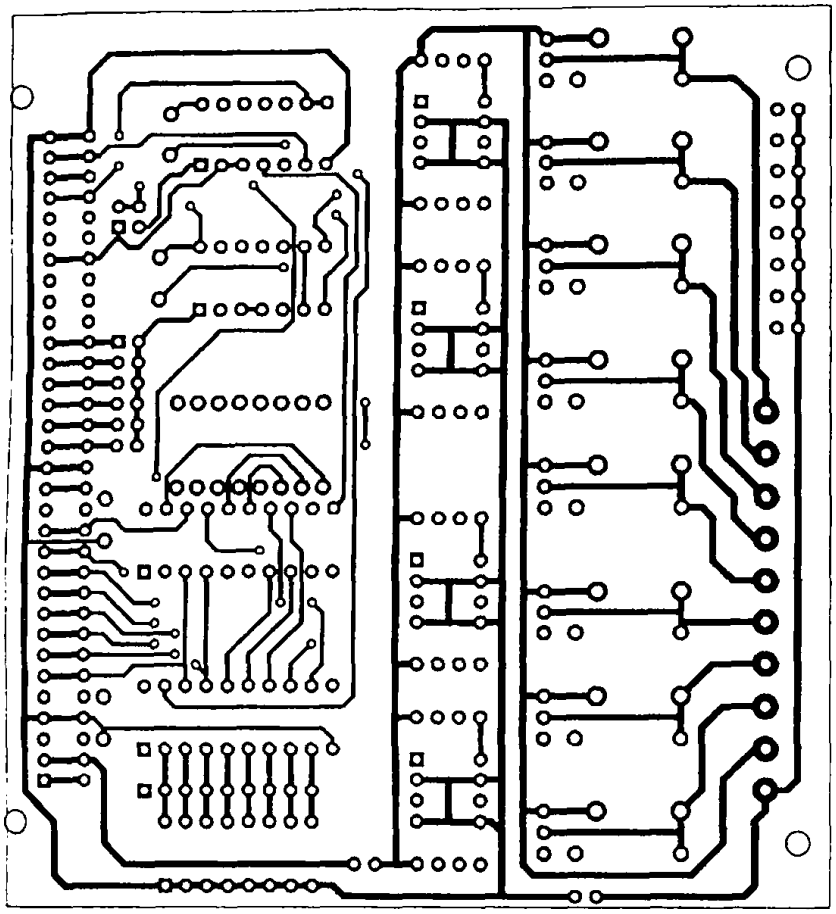
9.3 *Φυσικός σχεδιασμός (Τυπωμένο κύκλωμα)*

Ο φυσικός σχεδιασμός της κάρτας υλοποιήθηκε σε τυπωμένο κύκλωμα δύο όψεων, οι οποίες δίδονται στα σχήματα 9.2 και 9.3, με την όψη των εξαρτημάτων και την όψη των κολλήσεων αντίστοιχα. Στο σχήμα 9.4 φαίνεται το τοπογραφικό των εξαρτημάτων στο φυσικό επίπεδο. Οι διαστάσεις της κάρτας είναι (10.8 × 10.1) cm^2.

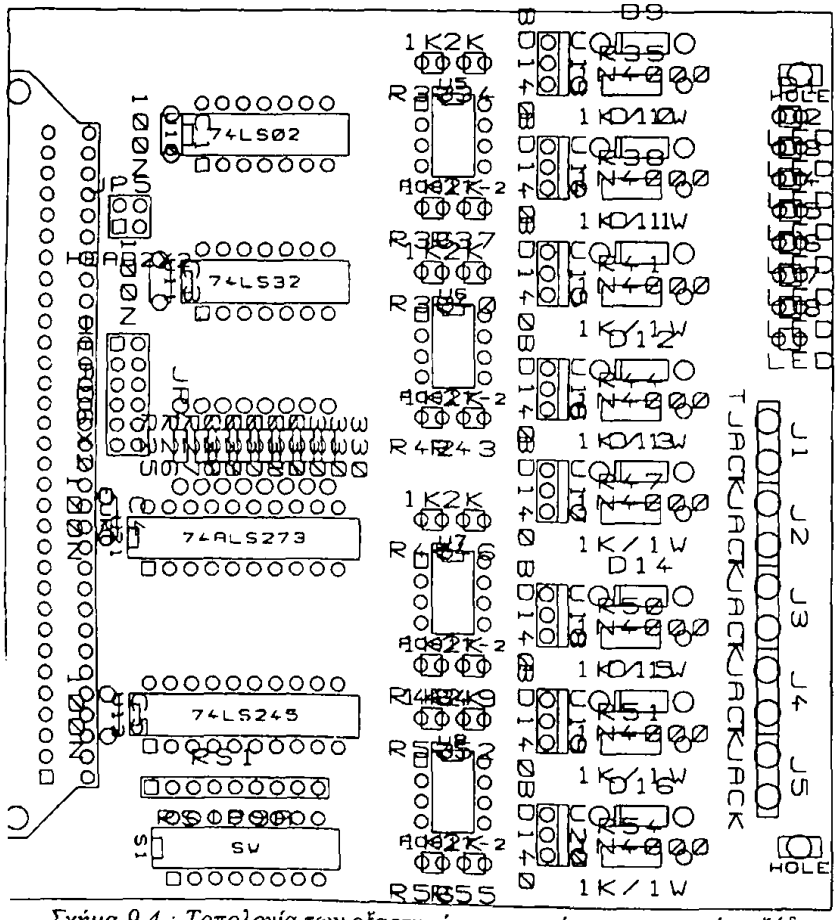


Σχήμα 9.2 : Όψη των εξαρτημάτων (Component Side) της κάρτας ψηφιακών εξόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ



Σχήμα 9.3 : Όψη των κολλήσεων (Solder Side) της κάρτας ψηφιακών εξόδων

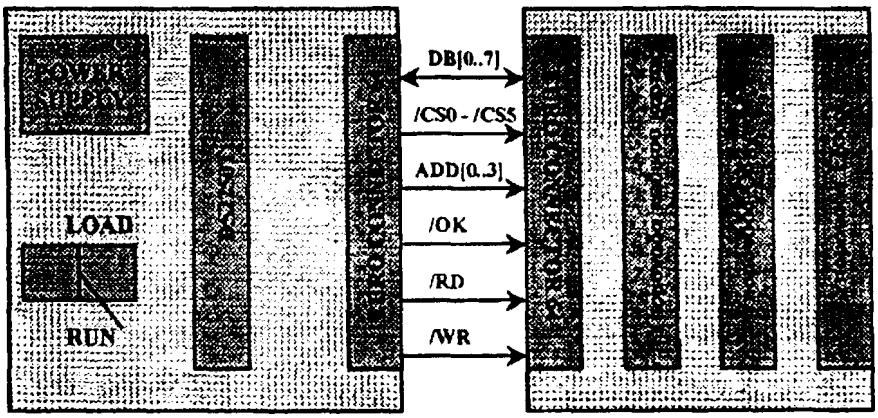


Σχήμα 9.4 : Τοπολογία των εξαρτημάτων της κάρτας ψηφιακών εξόδων

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

9.4 *Έλεγχος καλής λειτουργίας*

Μετά την υλοποίηση του τυπωμένου κυκλώματος ελέγχθηκε τόσο ηλεκτρικά η κάρτα, πιθανότητα βραχυκυκλώματος ή ψυχρής κόλλησης, όσο και έγινε εξομοίωση του χρονισμού και η παραγωγή των σημάτων (back-plane) για την ορθή λειτουργία της κάρτας χησιμοποιώντας το αναπτυξιακό σύστημα με βάση τον DS2250T, του οποίου το διάγραμμα διασύνδεσης φαίνεται στο σχήμα 9.5.



Σχήμα 9.5 : Διάγραμμα διασύνδεσης του αναπτυξιακού συστήματος DS2250T με την κάρτα ψηφιακών εξόδων

Το λογισμικό που χρησιμοποιήθηκε παρατίθεται στο παράρτημα Γ.

Στο πρώτο τμήμα (τμήμα A) του προγράμματος πρώτα στέλνονται τα σήματα $\overline{CS0}$ - $\overline{CS5}$, $\overline{OK}$, A0= A1 = A2 = X (Do not care) για να επιλέξουμε την διεύθυνση που αντιστοιχεί στο Status Word και μετά θέτοντας το σήμα $\overline{RD}$ σε λογικό μηδέν, διαβάζουμε την ψηφιακή λέξη η οποία πλέον βρίσκεται στο καταχωρητή ACC του μικροελεγκτή. Στην συνέχεια μεταφέρουμε τα περιεχόμενα του ACC στην πόρτα P0 όπου εμφανίζεται η ψηφιακή λέξη σε δυαδική μορφή μέσω μιας οθόνης από leds.

Στο δεύτερο τμήμα (τμήμα B) του προγράμματος στέλνονται πάλι τα σήματα $\overline{CS0}$ - $\overline{CS5}$, $\overline{OK}$ και A[0..2] = X (Do not care) για να οδηγήσουμε τα δεδομένα εισόδου της κάρτας, που στην προκειμένη περίπτωση είναι ο αριθμός 255, προς την έξοδο με την βοήθεια του σήματος $\overline{WR}$ σε λογικό μηδέν. Έτσι στην έξοδο της κάρτας θα πάρουμε τάση 12V ή 24V και 0V για λογικό 1 και 0 αντίστοιχα.

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΠΑΤΡΩΝ

Αναφορές

[1] Rainer Handel, Manfred N. Huber, Stefan Schroder, ATM Networks, Concepts, Protocols, Applications, Second Edition, 1995, Addison-Wesley

[2] William Stallings, ISDN and Broadband ISDN with frame relay and ATM, Third Edition, 1995, Prentice-Hall

[3] ITU-T: Recommendation I.432, 'B-ISDN User-Network Interface – Physical Layer Specification', Rev. 1, Geneva, 1993

[4] Roger Lipsett, Carl F. Schaefer, Cary Ussery, VHDL: Hardware Description and Design, 1993, Kluwer Academic Publishers

[5] Paul Reilly, PDH, Broadband ISDN, ATM, and All That: A guide to modern WAN networking, and how it evolved, April 10, 1994, Silicon Graphics Computer Systems

[6] Α. Κατσούλης, Α. Ευαγγελάτος, Κ. Καβιδόπουλος, Ν. Μήτρου (ICCS-LC), Α. Παγώνης (UILB), Λογικός σχεδιασμός ATM/PDH δέκτη, παραδοτέο έργου ΕΠΕΤ ΙΙ476: Βιομηχανικά Προϊόντα Μικροηλεκτρονικής, Αθήνα, 30.07.96

[7] ITU-T: Recommendation G.707, 'Synchronous Digital Hierarchy Bit Rates', Rev. 2, Geneva, 1993

[8] ITU-T: Recommendation G.708, 'Network Node Interface for the Synchronous Digital Hierarchy', Rev. 2, Geneva, 1993

[9] ITU-T: Recommendation G.709, 'Synchronous Multiplexing Structure', Rev. 2, Geneva, 1993

[10] Α. Ευαγγελάτος, Κ. Καβιδόπουλος, Ν. Μήτρου, Θ. Ραψομανίκης, Σχεδιασμός συστήματος (ATM Receiver), παραδοτέο ΕΠΕΤ ΙΙ476/Φ3/ICCS-LC/D3.21, Αθήνα, 30-11-95

[11] ITU-T: Recommendation I.321, 'B-ISDN protocol reference model and its application', Geneva, 1990

[12] "TDC1500A", 155.52-MBITS/S SONET/SDH ATM RECEIVER/TRANSMITTER, Product Information, ATM Interface Products, Texas Instruments, 1995

[13] ITU-T: Recommendation I.363, 'B-ISDN ATM Adaptation Layer Specification', Rev. 1, Geneva, 1993

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

[14] "PEB 2235", ISDN Primary Access Transceiver (IPAT), Product Information, Siemens, 1995

[15] "PEB 2035", Advanced CMOS Frame Aligner (ACFA), Product Information, Siemens, 1995

[16] Α. Παγώνης, Π.Κολοβός, Ν. Μάνθος, Ι. Ευαγγέλου, Φ. Α. Τριάντης, Ηλεκτρονικά σχέδια του συστήματος ΤΗΛ2, τμήμα του παραδοτέου του έργου ΕΚΒΑΝ-Π-79 : "Ανάπτυξη συστήματος τηλεδιαχείρησης υδάτινων πόρων", Εργαστήριο Φυσικής Υψηλών Ενεργειών, Τμήμα Φυσικής, Παν. Ιωαννίνων, Δεκέμβριος 31, 1996

[17] Α. Παγώνης, Ν. Μάνθος, Ι. Ευαγγέλου, Φ.Α. Τριάντης, Κάρτες I/O του Συστήματος ΤΗΛ2, Εσωτερικό Παραδοτέο του έργου ΕΚΒΑΝ-Π-79 : "Ανάπτυξη συστήματος τηλεδιαχείρησης υδάτινων πόρων", Εργαστήριο Φυσικής Υψηλών Ενεργειών, Τμήμα Φυσικής, Παν. Ιωαννίνων, Ιούλιος 19,1997

[18] "LM324", Low Power Quad Operational Amplifier, National Operational Amplifiers Databook, National Semiconductor, 1995

[19] "ADC0809", 8-bit μP Compatible A/D Converter with 8-Channel Multiplexer, Product Information, National Data Acquisition Databook, National Semiconductor, 1997

[20] Α. Παγώνης, Β. Θεοδώρου, Έλεγχος μέσω Η/Υ μιας τράπεζας Pan & Tilt υψηλής ακρίβειας, Διπλωματική εργασία, Εργαστήριο Φυσικής Υψηλών Ενεργειών, Τμήμα Φυσικής, Παν. Ιωαννίνων, Ιούνιος 1994

[21] Digital Design Seminar, Referrence Manual, Texas Instruments, 1997

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Παράρτημα Α

Παρατίθεται ο κώδικας VHDL του δέκτη ATM/PDH. Αποτελείται από τρία επιμέρους κομμάτια, τα δύο πρώτα είναι ανεξάρτητα και αφορούν το μεν ένα το κώδικα VHDL για την υπομονάδα ATMRX (τμήμα Α) το δε άλλο το κώδικα VHDL για την υπομονάδα AALRX (τμήμα Β). Το τρίτο κομμάτι εξαρτάται από τα δύο προηγούμενα και είναι ο ενοποιημένος κώδικας VHDL για τον δέκτη ATM/PDH (τμήμα Γ).

ΤΜΗΜΑ Α

```
-- ***************** RAM4X8 ******************************************
--     This unit is a RAM  with 4 addresses 8 bits  each. The header values
--      are stored here by the microprocessor at the initialization of the system.
--
*************************************************************************
*******
library IEEE;
use IEEE.std_logic_1164.all;
-- ===============================================================
--      RAM4X8 entity declaration
-- ===============================================================
entity ram4x8 is
   port(DATA_IN       : in  STD_LOGIC_VECTOR (7 downto 0);
        DATA_OUT      : out STD_LOGIC_VECTOR (7 downto 0);
        ADDR          : in  STD_LOGIC_VECTOR (1 downto 0);
        WE            : in  STD_LOGIC);
end;
-- ===============================================================
--      RAM4X8 architecture
-- ===============================================================
architecture  arch_ram4x8 of ram4x8  is
begin
 process(WE,ADDR)

  type CORE4X8 is array (0 to 3) of STD_LOGIC_VECTOR (7 downto 0);
  variable ram: CORE4X8;
  variable wrdeikths: integer range 0 to 3 :=0;
  variable rddeikths: integer range 0 to 3 :=0;

  begin
       if WE='1' then
         case ADDR is
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
            when "00" =>
               wrdeikths := 0;
               ram(wrdeikths) := DATA_IN;
               DATA_OUT <= ram(wrdeikths);
            when "01" =>
               wrdeikths := (wrdeikths + 1) mod 4;
               ram(wrdeikths) := DATA_IN;
               DATA_OUT <= ram(wrdeikths);
            when "10" =>
               wrdeikths := (wrdeikths + 2) mod 4;
               ram(wrdeikths) := DATA_IN;
               DATA_OUT <= ram(wrdeikths);
            when "11" =>
               wrdeikths := (wrdeikths + 3) mod 4;
               ram(wrdeikths) := DATA_IN;
               DATA_OUT <= ram(wrdeikths);
          end case;
       elsif WE='0' then
           case ADDR is
            when "00" =>
               rddeikths := 0;
               DATA_OUT <= ram(rddeikths);
            when "01" =>
               rddeikths := (rddeikths + 1) mod 4;
               DATA_OUT <= ram(rddeikths);
            when "10" =>
               rddeikths := (rddeikths + 2) mod 4;
               DATA_OUT <= ram(rddeikths);
            when "11" =>
               rddeikths := (rddeikths + 3) mod 4;
               DATA_OUT <= ram(rddeikths);
            end case;
       end if
  end process;
end arch_ram4x8;
-- ============================================================
-- RAM4X8 configuration
-- ============================================================
configuration ram4x8_cfg of ram4x8 is
  for arch_ram4x8
  end for;
end ram4x8_cfg;


-- ************************** COMPARATOR
**********************************
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--      This unit compares 2 8-bit bus signals and produces 2 control signals, EQ and
LSB.
-- EQ compares all bits, while LSB compares the 7 most significant bits.
--
**********************************************************************
***
library IEEE;
use IEEE.std_logic_1164.all;
-- ================================================
-- COMP8BIT entity declaration
-- ================================================
entity comp8bit is
  port (A       : in  STD_LOGIC_VECTOR (7 downto 0);
        B       : in  STD_LOGIC_VECTOR (7 downto 0);
        EQ      : out STD_LOGIC;
        LSB     : out STD_LOGIC); -- compares 7 downto 1
end;
-- ================================================
--    COMP8BIT architecture
-- ================================================
architecture arch_comp8bit of comp8bit is
begin
 update: process(A,B)
  begin
    LSB <= '1';
    if A(0) = B(0) then
      EQ <='1';
    elsif A(0) /= B(0) then
      EQ <='0';
    end if;
    for i in 1 to 7 loop
     if A(i) /= B(i)  then
      EQ <='0';
      LSB <='0';
     end if;
    end loop;
 end process;
end arch_comp8bit;
-- ================================================
--    COMP8BIT configuration
-- ================================================
configuration comp8bit_cfg of comp8bit is
  for arch_comp8bit
  end for;
end comp8bit_cfg;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ************ COUNTER2BIT ********************************
--     This unit is a 2-bit counter. It's output is connected to the address
-- input of the RAM4x8.
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================================
--       counter2b entity declaration
-- ============================================================
entity counter2b is
 port(CLK            : in  STD_LOGIC;
      CE             : in  STD_LOGIC;
      A0_COUNT       : out STD_LOGIC;
      A1_COUNT       : out STD_LOGIC;
      RESETAsync     : in  STD_LOGIC);
end;
-- ============================================================
--       counter2b architecture
-- ============================================================
architecture arch_counter2b of counter2b is

 signal a0 : STD_LOGIC;
 signal a1 : STD_LOGIC;
begin

 process(CLK,RESETAsync)
  begin
 if RESETAsync = '1' then
   a0 <= '0';
   a1 <= '0';
 elsif RESETAsync = '0' then
  if CLK'event and CLK ='1' then
   if CE ='1' and a0 ='1' and a1 ='1' then
    a0 <= '0';
    a1 <= '0';
   elsif CE ='1' and a0 ='0' and a1 ='0' then
    a0 <= '1';
    a1 <= '0';
   elsif CE ='1' and a0 ='1' and a1 ='0' then
    a0 <= '0';
    a1 <= '1';
   elsif CE ='1' and a0 ='0' and a1 ='1' then
    a0 <= '1';
    a1 <= '1';
   end if;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
   end if;
  end if;
 end process;
 A0_COUNT <= a0;
 A1_COUNT <= a1;
end arch_counter2b;
-- ============================================================
--        counter2b configuration
-- ============================================================
configuration counter2b_cfg of counter2b is
  for arch_counter2b
  end for;
end  counter2b_cfg;




-- ******************** DEMULTIPLEXER  1-2 ************************
--       This unit demultiplexes 1 bus of 8 bits into 2 8-bit buses.
-- ****************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ================================
--    demux1_2 entity declaration
-- ================================
entity demux1_2 is
 port(Q_DEMUX0       : out  STD_LOGIC_VECTOR (7 downto 0);
      Q_DEMUX1       : out  STD_LOGIC_VECTOR (7 downto 0);
      D_DEMUX        : in  STD_LOGIC_VECTOR (7 downto 0);
      S_DEMUX        : in  STD_LOGIC);
end;
-- ================================
--    demux1_2 architecture
-- ================================
architecture arch_demux1_2 of demux1_2 is
begin

 process(S_DEMUX,D_DEMUX)
  begin
    if S_DEMUX ='0' then
     Q_DEMUX0 <= D_DEMUX;
    elsif S_DEMUX ='1' then
     Q_DEMUX1 <= D_DEMUX;
    end if;
 end process;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
end arch_demux1_2;
-- ================================================
--   demux1_2 configuration
-- ================================================
configuration demux1_2_cfg of demux1_2 is
 for arch_demux1_2
 end for;
end demux1_2_cfg;



-- ******************** MULTIPLEXER 2-1 ****************
--  This unit multiplexes 2 8-bits buses into 1 8-bits bus.
-- ******************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ================================================
-- mux2_1 entity declaration
-- ================================================
entity mux2_1 is
 port(D_MUX0       : in  STD_LOGIC_VECTOR (7 downto 0);
      D_MUX1       : in  STD_LOGIC_VECTOR (7 downto 0);
      Q_MUX        : out STD_LOGIC_VECTOR (7 downto 0);
      S_MUX        : in  STD_LOGIC);
end;
-- ================================================
--    mux2_1 architecture
-- ================================================
architecture arch_mux2_1 of mux2_1 is
begin

 process(S_MUX,D_MUX0,D_MUX1)
  begin
    if S_MUX ='0' then
     Q_MUX <= D_MUX0;
    elsif S_MUX ='1' then
      Q_MUX <= D_MUX1;
    end if;
 end process;
end arch_mux2_1;
-- ================================================
--    mux2_1 configuration
-- ================================================
configuration mux2_1_cfg of mux2_1 is
 for arch_mux2_1
 end for;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
end mux2_1_cfg;


-- ************** 8 bit REGISTER ********************************
--      This unit is a 8-bit register.
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================
--      reg8bit entity declaration
-- ============================================
entity reg8bit is
  port(CLK_REG            : in  STD_LOGIC;
       EN_REG             : in  STD_LOGIC;
       DIN_REG            : in  STD_LOGIC_VECTOR (7 downto 0);
       DOUT_REG           : out STD_LOGIC_VECTOR (7 downto 0);
       REG_RESETAsync     : in  STD_LOGIC);
end;
-- ============================================
--   reg8bit architecture
-- ============================================
architecture arch_reg8bit of reg8bit is
begin

 process(CLK_REG,REG_RESETAsync)
 begin
  if REG_RESETAsync ='1' then
   DOUT_REG <= (others => '0');
  elsif (CLK_REG'event and CLK_REG ='1') then
   if EN_REG ='1' then
    DOUT_REG <= DIN_REG;
   end if;
  end if;
 end process;
end arch_reg8bit;
-- ============================================================
--    reg8bit configuration
-- ============================================================
configuration reg8bit_cfg of reg8bit is
  for arch_reg8bit
  end for;
end reg8bit_cfg;


-- ********************* TRISTATE ********************************
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- This unit is a tristate with clock
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================================
--        tristate entity declaration
-- ============================================================
entity tristate is
   port (A_D     : in  STD_LOGIC_VECTOR (7 downto 0);
         CLK_D         : in  STD_LOGIC;
         D_EN   : in  STD_LOGIC;
         RST_DFF : in  STD_LOGIC;
         NEW_CL  : out STD_LOGIC;
         O_D     : out STD_LOGIC_VECTOR (7 downto 0));
end;
-- ============================================================
--        tristate architecture
-- ============================================================
architecture arch_tristate of tristate is
begin
 process(A_D,CLK_D,D_EN,RST_DFF)
 begin
   if RST_DFF = '1' then
     O_D <= (others =>'0');
     NEW_CL <= '0';
   elsif RST_DFF = '0' then
    if CLK_D'event and CLK_D ='1' then
     if D_EN ='1' then
      NEW_CL <= '1';
        O_D <= A_D;
     else
      NEW_CL <= '0';
      O_D <= (others =>'X');
     end if;
    end if;
   end if;
 end process;
end arch_tristate;
-- ============================================================
--        tristate configuration
-- ============================================================
configuration tristate_cfg of tristate is
  for arch_tristate
  end for;
end tristate_cfg;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ********** MPENADDR **********
--
--      This unit sends the address of microproccessor in the address -bus
--  when  MP_EN gets high.
-- ******************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ================================================
--      mpenaddr entity declaration
-- ================================================
entity mpenaddr is
  port (MP_EN              :in  STD_LOGIC;
        MP_ADDRESS         :out STD_LOGIC_VECTOR(4 downto 0));
end;
-- ================================================
--      mpenaddr architecture
-- ================================================
architecture arch_mpenaddr of mpenaddr is
begin
   process(MP_EN)
   begin
        if MP_EN ='1' then
          MP_ADDRESS <="00001";
        end if;
   end process;
end arch_mpenaddr;
-- ================================================
--      mpenaddr configuration
-- ================================================
configuration mpenaddr_cfg of mpenaddr is
      for arch_mpenaddr
      end for;
end mpenaddr_cfg;


-- ************************************************************
--
--    DEMUX1_2 --> REG8BIT --> MUX2_1
--
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ================================================
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--    de_reg_mux entity declaration
-- ================================================
entity de_reg_mux is
  port(C_L_K              : in  STD_LOGIC;
       SEL_DEMUX          : in  STD_LOGIC;
       SEL_MUX            : in  STD_LOGIC;
       EN_REGISTERS       : in  STD_LOGIC_VECTOR (7 downto 0);
       CE_LL              : in  STD_LOGIC_VECTOR (7 downto 0);
       PAY_LOAD           : out STD_LOGIC_VECTOR (7 downto 0);
       INP_COMPAR         : out STD_LOGIC_VECTOR (7 downto 0);
       RST_REGISTERS      : in  STD_LOGIC_VECTOR (7 downto 0));
end;
-- ================================================
-- de_reg_mux architecture
-- ================================================
architecture arch_de_reg_mux of de_reg_mux is

 signal de1_mux: STD_LOGIC_VECTOR (7 downto 0);
 signal de0_mux: STD_LOGIC_VECTOR (7 downto 0);
 signal reg4th : STD_LOGIC_VECTOR (7 downto 0);
 signal reg5th : STD_LOGIC_VECTOR (7 downto 0);
 signal reg6th : STD_LOGIC_VECTOR (7 downto 0);
 signal reg7th : STD_LOGIC_VECTOR (7 downto 0);
 signal reg1st : STD_LOGIC_VECTOR (7 downto 0);
 signal reg2nd : STD_LOGIC_VECTOR (7 downto 0);
 signal inp3reg: STD_LOGIC_VECTOR (7 downto 0);
 signal en_reg : STD_LOGIC_VECTOR (7 downto 0);
 signal rst_reg: STD_LOGIC_VECTOR (7 downto 0);

 component demux1_2
   port(Q_DEMUX0       : out STD_LOGIC_VECTOR (7 downto 0);
        Q_DEMUX1       : out STD_LOGIC_VECTOR (7 downto 0);
        D_DEMUX        : in  STD_LOGIC_VECTOR (7 downto 0);
        S_DEMUX        : in  STD_LOGIC);
 end component;

 component reg8bit
   port(CLK_REG             : in  STD_LOGIC;
        EN_REG              : in  STD_LOGIC;
        DIN_REG             : in  STD_LOGIC_VECTOR (7 downto 0);
        DOUT_REG            : out STD_LOGIC_VECTOR (7 downto 0);
        REG_RESETAsync      : in  STD_LOGIC);
 end component;

 component mux2_1
   port(D_MUX0      : in  STD_LOGIC_VECTOR (7 downto 0);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
        D_MUX1      : in  STD_LOGIC_VECTOR (7 downto 0);
        Q_MUX       : out STD_LOGIC_VECTOR (7 downto 0);
        S_MUX       : in  STD_LOGIC);
 end component;

begin
 en_reg <= EN_REGISTERS;
 rst_reg <= RST_REGISTERS;
 U0: demux1_2
    port map(Q_DEMUX0 => de0_mux,Q_DEMUX1 => de1_mux,D_DEMUX =>
CE_LL,S_DEMUX => SEL_DEMUX);
 U1: reg8bit -- REGISTER 4
    port map(CLK_REG => C_L_K,EN_REG => en_reg(3),DIN_REG =>
de0_mux,DOUT_REG => reg4th,REG_RESETAsync => rst_reg(3));
 U2: reg8bit -- REGISTER 5
    port map(CLK_REG => C_L_K,EN_REG => en_reg(4),DIN_REG =>
reg4th,DOUT_REG => reg5th,REG_RESETAsync => rst_reg(4));
 U3: reg8bit -- REGISTER 6
    port map(CLK_REG => C_L_K,EN_REG => en_reg(5),DIN_REG =>
reg5th,DOUT_REG => reg6th,REG_RESETAsync => rst_reg(5));
 U4: reg8bit -- REGISTER 7
    port map(CLK_REG => C_L_K,EN_REG => en_reg(6),DIN_REG =>
reg6th,DOUT_REG => reg7th,REG_RESETAsync => rst_reg(6));
 U5: reg8bit -- REGISTER 8
    port map(CLK_REG => C_L_K,EN_REG => en_reg(7),DIN_REG =>
reg7th,DOUT_REG => PAY_LOAD,REG_RESETAsync => rst_reg(7));
 U6: reg8bit -- REGISTER 1
    port map(CLK_REG => C_L_K,EN_REG => en_reg(2),DIN_REG =>
de1_mux,DOUT_REG => reg1st,REG_RESETAsync => rst_reg(2));
 U7: reg8bit -- REGISTER 2
    port map(CLK_REG => C_L_K,EN_REG => en_reg(1),DIN_REG =>
reg1st,DOUT_REG => reg2nd,REG_RESETAsync => rst_reg(1));
 U8: mux2_1
    port map(D_MUX0 => reg2nd,D_MUX1 => reg1st,Q_MUX => inp3reg,S_MUX =>
SEL_MUX);
 U9: reg8bit -- REGISTER 3
    port map(CLK_REG => C_L_K,EN_REG => en_reg(0),DIN_REG =>
inp3reg,DOUT_REG => INP_COMPAR,REG_RESETAsync => rst_reg(0));

end arch_de_reg_mux;
-- ============================================================
--   de_reg_mux configuration
-- ============================================================
configuration de_reg_mux_cfg of de_reg_mux is
  for arch_de_reg_mux
  end for;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
end de_reg_mux_cfg;


-- ***************** RAM_COMP ***********************************
--       RAM + COMPARATOR
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================================
--    ram_comp entity declaration
-- ============================================================
entity ram_comp is
  port(D_RAM_COMP     : in  STD_LOGIC_VECTOR (7 downto 0);
       IN_PAYLOAD     : in  STD_LOGIC_VECTOR (7 downto 0);
       WRIT_EN        : in  STD_LOGIC;
       ADDR0          : in  STD_LOGIC;
       ADDR1          : in  STD_LOGIC;
       EQUAL_COMP     : out STD_LOGIC;
       LSB_COMP       : out STD_LOGIC);
end ram_comp;
-- ============================================================
--  ram_comp  architecture
-- ============================================================
architecture arch_ram_comp of ram_comp is

 signal stream  : std_logic_vector (7 downto 0);

 component ram4x8
   port(DATA_IN       : in  STD_LOGIC_VECTOR (7 downto 0);
        DATA_OUT      : out STD_LOGIC_VECTOR (7 downto 0);
        ADDRES0       : in  STD_LOGIC;
        ADDRES1       : in  STD_LOGIC;
        WE            : in  STD_LOGIC);
 end component;

 component comp8bit
    port(A      : in  STD_LOGIC_VECTOR (7 downto 0);
         B      : in  STD_LOGIC_VECTOR (7 downto 0);
         EQ     : out STD_LOGIC;
         LSB    : out STD_LOGIC);
 end component;

begin
 U0: ram4x8
 port map(D_RAM_COMP,stream,ADDR0,ADDR1,WRIT_EN);
 U1: comp8bit
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
 port map(IN_PAYLOAD,stream,EQUAL_COMP,LSB_COMP);

end arch_ram_comp;
-- ============================================================
--   ram_comp configuration
-- ============================================================
configuration ram_comp_cfg of ram_comp is
  for arch_ram_comp
  end for;
end ram_comp_cfg;



-- ***************** XOR_MP ****************************
--       This unit is a xor gate with 12-bits inputs
-- ****************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================================
--       xor_mp entity declaration
-- ============================================================
entity xor_mp is
  port(IN_XOR_MP   : in  STD_LOGIC_VECTOR (11 downto 0);
       OUT_MP      : out STD_LOGIC);
end xor_mp;
-- ============================================================
--       xor_mp architecture
-- ============================================================
architecture arch_xor_mp of xor_mp is
begin
 process(IN_XOR_MP)
 begin
   OUT_MP <= IN_XOR_MP(11) xor IN_XOR_MP(10) xor IN_XOR_MP(9) xor
IN_XOR_MP(8) xor IN_XOR_MP(7) xor IN_XOR_MP(6) xor IN_XOR_MP(5) xor
IN_XOR_MP(4) xor IN_XOR_MP(3) xor IN_XOR_MP(2) xor IN_XOR_MP(1) xor
IN_XOR_MP(0);
 end process;
end arch_xor_mp;
-- ============================================================
--       xor_mp configuration
-- ============================================================
configuration xor_mp_cfg of xor_mp is
  for arch_xor_mp
  end for;
end xor_mp_cfg;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ************* RAM_COMP_UNIT ****************************
--        This unit is the unit which stores and checks the supported
--        headers of the ATM cell.
-- ******************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ==================================================================
--        ram_comp_unit entity declaration
-- ==================================================================
entity ram_comp_unit is
  port(IN_PAY          : in  STD_LOGIC_VECTOR (7 downto 0);
       DATA_IN_UNIT    : in  STD_LOGIC_VECTOR (7 downto 0);
       WR_EN           : in  STD_LOGIC_VECTOR (13 downto 0);
       ADD0            : in  STD_LOGIC;
       ADD1            : in  STD_LOGIC;
       G_1             : out STD_LOGIC;
       G_2             : out STD_LOGIC;
       LSB_1           : out STD_LOGIC;
       LSB_2           : out STD_LOGIC;
       G_MP            : out STD_LOGIC;
       LSB_MP          : out STD_LOGIC);
end ram_comp_unit;
-- ==================================================================
--        ram_comp_unit architecture
-- ==================================================================
architecture arch_ram_comp_unit of ram_comp_unit is

 signal equal_unit : STD_LOGIC_VECTOR (11 downto 0);
 signal lsb_unit   : STD_LOGIC_VECTOR (11 downto 0);
 component ram_comp
  port(D_RAM_COMP    : in  STD_LOGIC_VECTOR (7 downto 0);
       IN_PAYLOAD    : in  STD_LOGIC_VECTOR (7 downto 0);
       WRIT_EN       : in  STD_LOGIC;
       ADDR0         : in  STD_LOGIC;
       ADDR1         : in  STD_LOGIC;
       EQUAL_COMP    : out STD_LOGIC;
       LSB_COMP      : out STD_LOGIC);
 end component;

 component xor_mp
  port(IN_XOR_MP  : in  STD_LOGIC_VECTOR (11 downto 0);
       OUT_MP     : out STD_LOGIC);
 end component;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
begin
 U0: ram_comp
 port map(DATA_IN_UNIT,IN_PAY,WR_EN(13),ADD0,ADD1,G_1,LSB_1);
 U1: ram_comp
 port map(DATA_IN_UNIT,IN_PAY,WR_EN(12),ADD0,ADD1,G_2,LSB_2);
 U2: ram_comp
port
map(DATA_IN_UNIT,IN_PAY,WR_EN(11),ADD0,ADD1,equal_unit(11),lsb_unit(11))
;
 U3: ram_comp
port
map(DATA_IN_UNIT,IN_PAY,WR_EN(10),ADD0,ADD1,equal_unit(10),lsb_unit(10))
;
 U4: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(9),ADD0,ADD1,equal_unit(9),lsb_unit(9));
 U5: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(8),ADD0,ADD1,equal_unit(8),lsb_unit(8));
 U6: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(7),ADD0,ADD1,equal_unit(7),lsb_unit(7));
 U7: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(6),ADD0,ADD1,equal_unit(6),lsb_unit(6));
 U8: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(5),ADD0,ADD1,equal_unit(5),lsb_unit(5));
 U9: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(4),ADD0,ADD1,equal_unit(4),lsb_unit(4));
 U10: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(3),ADD0,ADD1,equal_unit(3),lsb_unit(3));
 U11: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(2),ADD0,ADD1,equal_unit(2),lsb_unit(2));
 U12: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(1),ADD0,ADD1,equal_unit(1),lsb_unit(1));
 U13: ram_comp
 port
map(DATA_IN_UNIT,IN_PAY,WR_EN(0),ADD0,ADD1,equal_unit(0),lsb_unit(0));
 U14: xor_mp
 port map(equal_unit(11 downto 0),G_MP);
 U15: xor_mp
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
 port map(lsb_unit(11 downto 0),LSB_MP);
end arch_ram_comp_unit;
-- ============================================================
--        ram_comp_unit configuration
-- ============================================================
configuration ram_comp_unit_cfg of ram_comp_unit is
  for arch_ram_comp_unit
  end for;
end ram_comp_unit_cfg;


-- ******************** STATE_MACHINE ************************
--        This unit is  state machine of the unit ATMRX
-- ************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
-- ============================================================
--        state_ma entity declaration
-- ============================================================
entity state_ma is
  port(ATM_CLK        : in  STD_LOGIC;
       RXCELL         : in  STD_LOGIC;
       RXFE_BAR       : in  STD_LOGIC;
       A0             : in  STD_LOGIC;
       A1             : in  STD_LOGIC;
       G1             : in  STD_LOGIC;
       LSB1           : in  STD_LOGIC;
       G2             : in  STD_LOGIC;
       LSB2           : in  STD_LOGIC;
       G_MP           : in  STD_LOGIC;
       LSB_MP         : in  STD_LOGIC;
       ADDRES_BUS     : in  STD_LOGIC_VECTOR (4 downto 0);
       WR             : in  STD_LOGIC;
       RD             : in  STD_LOGIC;
       RRE_BAR        : out STD_LOGIC;
       TRIST1_EN      : out STD_LOGIC;    -- Active high
       TRIST2_EN      : out STD_LOGIC;    -- Active high
       MP_ENABLE      : out STD_LOGIC;    -- Active high
       RAM_ENABLE     : out STD_LOGIC_VECTOR (13 downto 0);
       SEL_MUX        : out STD_LOGIC;
       SEL_DEMUX      : out STD_LOGIC;
       EN_COUNTER     : out STD_LOGIC;
       EN_REGS        : out STD_LOGIC_VECTOR (7 downto 0);
       G_RST          : in  STD_LOGIC);
end;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ============================================================
--        state_ma architecturte
-- ============================================================
architecture arch_state_ma of state_ma is
signal G          : STD_LOGIC;
signal LSB        : STD_LOGIC;
signal ENABL      : STD_LOGIC;      -- Active high
signal state      : STD_LOGIC_VECTOR (3 downto 0);
signal en_ram     : STD_LOGIC;

begin

process(ATM_CLK,RXCELL,RXFE_BAR,A0,A1,G1,G2,G_MP,LSB1,LSB2,LSB_MP,
ADDRES_BUS,WR,RD,G_RST)
 begin
   state <= "0000";
  if G_RST = '1' then
   SEL_MUX <= '0';
   SEL_DEMUX <= '0';
   EN_COUNTER <= '0';
   MP_ENABLE <= '0';
   TRIST1_EN <= '0';
   TRIST2_EN <= '0';
   EN_REGS <= "00000000";  -- 1st-2nd-INPC-4th-5th-6th-7th-PAY
  elsif G_RST = '0' then
   if (G1 ='1') then
        G <= G1;
   elsif (G2 ='1') then
        G <= G2;
   elsif (G_MP ='1') then
        G <= G_MP;
   else
     G <= '0';
   end if;

   if (LSB1 ='1') then
     LSB <= LSB1;
   elsif (LSB2 ='1') then
     LSB <= LSB2;
   elsif (LSB_MP ='1') then
     LSB <= LSB_MP;
   else
     LSB <= '0';
   end if;

   if ATM_CLK'event and ATM_CLK = '1' then
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
case state is
 when "0000" =>
      RRE_BAR <='1';
      A0 <='0';
      A1 <='0';

  case ADDRES_BUS is
   when "00010" =>
    en_ram <= RAM_ENABLE(13);
   when "00011" =>
    en_ram <= RAM_ENABLE(12);
   when "00100" =>
    en_ram <= RAM_ENABLE(11);
   when "00101" =>
    en_ram <= RAM_ENABLE(10);
   when "00110" =>
    en_ram <= RAM_ENABLE(9);
   when "00111" =>
    en_ram <= RAM_ENABLE(8);
   when "01000" =>
    en_ram <= RAM_ENABLE(7);
   when "01001" =>
    en_ram <= RAM_ENABLE(6);
   when "01010" =>
    en_ram <= RAM_ENABLE(5);
   when "01011" =>
    en_ram <= RAM_ENABLE(4);
   when "01010" =>
    en_ram <= RAM_ENABLE(3);
   when "01011" =>
    en_ram <= RAM_ENABLE(2);
   when "01100" =>
    en_ram <= RAM_ENABLE(1);
   when "01101" =>
    en_ram <= RAM_ENABLE(0);
   when others =>
    en_ram <= (others =>'X');
   end case;

   if WR ='1' then
     en_ram ='1';
     if A0 ='0' and A1='0' then
        A0='1';
        A1='0';
     elsif A0='1' and A1='0' then
        A0='0';
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
            A1='1';
        elsif A0='0' and A1='1' then
            A0='1';
            A1='1';
        else
            A0='0';
            A1='0';
            en_ram='0';
        end if;
        else
          en_ram <='0';
    end if;

    if (RXFE_BAR = '0' and WR ='1') then
     state <= "0000";
    elsif (RXFE_BAR = '1' and WR='0') then
     RRE_BAR <= '0';
     EN_COUNTER <= '0';
     ENABLE <= '1';
     SEL_DEMUX <= '0'; -- Epilegw 1st reg.
     SEL_MUX <= '0';   -- Diavaze apo ton 1st reg.
     EN_REGS <= "00000000";
    end if;
    state <= "0001";
   when "0001" =>
    RRE_BAR <= '0';
    EN_COUNTER <= '0';
    ENABLE <= '1';
    SEL_DEMUX <= '0';
    SEL_MUX  <= '0';
    EN_REGS <= "00000000";
    state <= "0010";
   when "0010" =>
    RRE_BAR <= '0';
    SEL_DEMUX <= '0'; -- EPILEGW 1st REG.
    SEL_MUX <= '0';  -- EPILEGW 1st REG.
    ENABLE <= '1';
    EN_REGS <= "10001111";
    state <= "0011";
   when "0011" =>
    RRE_BAR <= '0';
    ENABLE <= '1';
    EN_REGS <= "11100111";
    state <= "0100";
   when "0100" =>
    SEL_MUX <= '1';  -- EPILEGW 2nd REG.
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
ENABLE <= '1';
EN_REGS <= "11000011";
state <= "0101";
when "0101" =>
RRE_BAR <= '0';
if G = '1' then
 if A0 = '1' and A1 = '0' then
  SEL_MUX <= '1';  -- EPILEGW  2nd REG.
     SEL_DEMUX <= '1';  -- EPILEGW 4th REG.
  EN_COUNTER <= '1';
  ENABLE <= '1';
     EN_REGS <= "01110000";
  state <= "1000";
 elsif A0 = '0' and A1 = '0' then
  SEL_MUX <= '1';
  EN_COUNTER <= '1';
  ENABLE <= '1';
  EN_REGS <= "11100001";
  state <= "0110";
 elsif A0 = '1' and A1 = '1' then
   SEL_DEMUX <= '1';
   EN_COUNTER <= '1';
   ENABLE <= '1';
   EN_REGS <= "00011111";
   state <= "1001";
  else
   SEL_MUX <= '1';
   SEL_DEMUX <= '1';
   EN_COUNTER <= '1';
   ENABLE <= '1';
   EN_REGS <= "00111100";
   state <= "0111";
 end if;
 elsif G = '0' then
if A0 = '1' and  A1 = '1' then
 if LSB = '1' then
  RRE_BAR <= '0';
  ENABLE <= '1';
  SEL_DEMUX <= '1';
  EN_COUNTER <= '1';
  EN_REGS <= "00011111";
  state <= "1001";
 elsif LSB = '0' then
  EN_COUNTER <= '1';
  ENABLE <= '1';
  EN_REGS <= "00000000";
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      state <= "1111";
      end if;
     elsif A0 = '0' and A1 = '0' then
      EN_COUNTER <= '0';
      ENABLE <= '1';
      EN_REGS <= "00000001";
      state <= "1110";
     else
      RRE_BAR <= '1';
      EN_COUNTER <= '1';   -- a=1 h' a=2
      ENABLE <= '1';
      EN_REGS <= "00000000";
      state <= "1111";
     end if;
     end if;
    ENABLE <= '1';
   when "0110" =>
    EN_COUNTER <= '0';
    ENABLE <= '0';
    EN_REGS <= "00000000";
    state <= "0101";
   when "0111" =>
    EN_COUNTER <= '0';
    SEL_DEMUX <= '1';
    ENABLE <= '0';
    EN_REGS <= "00011110";
    state <= "0101";
   when "1000" =>
    ENABLE <= '0';
    EN_COUNTER <= '0';
    SEL_DEMUX <= '1';
    EN_REGS <= "00011000";
    state <= "0101";
   when "1001" =>
    EN_COUNTER <= '0';
    if RXFE_BAR = '0' then
     RRE_BAR <= '0';
     ENABLE <= '1';
     EN_REGS <= "00001111";
     state <= "1010";
    elsif RXCELL = '1' then
    if RXFE_BAR = '1' then
     if RXCELL = '1' then
      SEL_DEMUX <= '0'; -- EPILEGW 1st REG.
      SEL_MUX <= '0';  -- EPILEGW 1st REG.
      ENABLE <= '1';
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
 EN_REGS <= "10001111";
 state <= "0011";
elsif RXCELL = '0' then
 SEL_DEMUX <= '1';
 EN_COUNTER <= '0';
 ENABLE <= '1';
 EN_REGS <= "00011111";
 state <= "1001";
end if;
end if;
end if;
when "1010" =>
RRE_BAR <= '0';
EN_COUNTER <= '0';
if RXFE_BAR = '0' then
 ENABLE <= '1';
 EN_REGS <= "00000111";
 state <= "1011";
elsif RXFE_BAR = '1' then
 SEL_DEMUX <= '0';
 ENABLE <= '1';
 EN_REGS <= "10000111";
 state <= "0011";
end if;
when "1011" =>
RRE_BAR <= '0';
ENABLE <= '1';
EN_COUNTER <= '0';
if RXFE_BAR = '0' then
 EN_REGS <= "00000011";
 state <= "1100";
elsif RXFE_BAR = '1' then
 SEL_DEMUX <= '0';
 ENABLE <= '0';
 EN_REGS <= "10000011";
 state <= "0011";
end if;
when "1100" =>
RRE_BAR <= '0';
ENABLE <= '0';
EN_COUNTER <= '0';
if RXFE_BAR = '0' then
 EN_REGS <= "00000001";
 state <= "1101";
elsif RXFE_BAR = '1' then
 SEL_DEMUX <= '0';
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
 ENABLE <= '0';
 EN_REGS <= "10000001";
 state <= "0011";
 end if;
when "1101" =>
 RRE_BAR <= '0';
 EN_COUNTER <= '0';
 if RXFE_BAR = '0' then
 ENABLE <= '1';
 EN_REGS <= "00000000";
 state <= "1101";
 elsif RXFE_BAR = '1' then
 ENABLE <= '1';
 SEL_DEMUX <= '0';
 EN_REGS <= "10000000";
 state <= "0011";
 end if;
when "1110" =>
 EN_COUNTER <= '0';
 ENABLE <= '0';
 RRE_BAR <= '0';
 EN_REGS <= "00000000";
 if RXCELL = '1' then
 state <= "0010";
 else if RXCELL = '0' then
 state <= "1110";
 end if;
 end if;
when "1111" =>
 if A0 = '1' and A1 = '0' then
 EN_COUNTER <= '1';
 state <= "1111";
 else if A0 = '0' and A1 = '1' then
 EN_COUNTER <= '1';
 state <= "1111";
 else if A0 = '1' and A1 = '1' then
 EN_COUNTER <= '1';
 state <= "1111";
 else if A0 = '0' and A1 = '0' then
 RRE_BAR <= '0';
 EN_COUNTER <= '0';
 ENABLE <= '1';
 EN_REGS <= "00000000";
 if RXCELL = '1' then
  state <= "0010";
 else if RXCELL = '0' then
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
            state <= "1111";
          end if;
        end if;
          end if;
          end if;
          end if;
          end if;
      when others =>
          state <="0000";
    end case;
   end if;
   if (G1 ='1'or LSB1 ='1') then
      TRIST1_EN <= ENABLE;
      TRIST2_EN <= '0';
      MP_ENABLE <= '0';
    elsif (G2 ='1' or LSB2 ='1') then
      TRIST2_EN <= ENABLE;
      TRIST1_EN <= '0';
      MP_ENABLE <= '0';
    elsif (G_MP ='1' or LSB_MP ='1') then
      MP_ENABLE <= ENABLE;
      TRIST1_EN <= '0';
      TRIST2_EN <= '0';
    else
      TRIST1_EN <= '0';
      TRIST2_EN <= '0';
      MP_ENABLE <= '0';
    end if;
  end if;
 end process;
end arch_state_ma;

-- ============================================
--        state_ma configuration
-- ============================================
configuration state_ma_cfg of state_ma is
      for arch_state_ma
      end for;
end state_ma_cfg;


-- ********** ATM_RECEIVER ********************************
--        The unit ATMRX
-- ********************************************************************
library IEEE;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
use IEEE.std_logic_1164.all;
-- ============================================
--       atm_receiver entity declaration
-- ============================================
entity atm_receiver is
  port (ATMCLK          :in  STD_LOGIC;
        GL_RST          :in  STD_LOGIC;
        ADDRESS_BUS     :inout  STD_LOGIC_VECTOR (4 downto 0);
        DATA_BUS        :inout  STD_LOGIC_VECTOR (7 downto 0);
        WR              :in  STD_LOGIC;
        RD              :in  STD_LOGIC;
        ATMRX_IN        :in  STD_LOGIC_VECTOR (7 downto 0);
        RX_CELL         :in  STD_LOGIC;     -- high active
        RXFE_BAR        :in  STD_LOGIC;          -- low active
        RRE_BAR         :out STD_LOGIC;          -- low active
      NEW_CELL_STR1     :out STD_LOGIC;
      NEW_CELL_STR2     :out STD_LOGIC;
      NEW_CELL_MP       :out STD_LOGIC;
     ATMRX_OUT          :out STD_LOGIC_VECTOR (7 downto 0));
end;
-- ============================================
--       atm_receiver architecture
-- ============================================
architecture arch_atm_receiver of atm_receiver is

signal count0       : STD_LOGIC;
signal count1       : STD_LOGIC;
signal g1_1         : STD_LOGIC;
signal lsb1_1       : STD_LOGIC;
signal g1_2         : STD_LOGIC;
signal lsb1_2       : STD_LOGIC;
signal g1_mp        : STD_LOGIC;
signal lsb1_mp      : STD_LOGIC;
signal enable1      : STD_LOGIC;
signal enable2      : STD_LOGIC;
signal mpenable     : STD_LOGIC;
signal muxen        : STD_LOGIC;
signal demuxen      : STD_LOGIC;
signal ramen        : STD_LOGIC_VECTOR (13 downto 0);
signal counten      : STD_LOGIC;
signal regenable    : STD_LOGIC_VECTOR (7 downto 0);
signal compinput    : STD_LOGIC_VECTOR (7 downto 0);
signal pay          : STD_LOGIC_VECTOR (7 downto 0);

  component state_ma
  port(ATM_CLK        : in  STD_LOGIC;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      RXCELL          : in STD_LOGIC;
      RXFE_BAR        : in STD_LOGIC;
      A0              : in STD_LOGIC;
      A1              : in STD_LOGIC;
      G1              : in STD_LOGIC;
      LSB1            : in STD_LOGIC;
      G2              : in STD_LOGIC;
      LSB2            : in STD_LOGIC;
      G_MP            : in STD_LOGIC;
      LSB_MP          : in STD_LOGIC;
      ADDRES_BUS      : in STD_LOGIC_VECTOR (4 downto 0);
      WR              : in STD_LOGIC;
      RD              : in STD_LOGIC;
      RRE_BAR         : out STD_LOGIC;
      TRIST1_EN       : out STD_LOGIC;   -- Active high
      TRIST2_EN       : out STD_LOGIC;   -- Active high
      MP_ENABLE       : out STD_LOGIC;    -- Active high
      RAM_ENABLE      : out STD_LOGIC_VECTOR (13 downto 0);
      SEL_MUX         : out STD_LOGIC;
      SEL_DEMUX       : out STD_LOGIC;
      EN_COUNTER      : out STD_LOGIC;
      EN_REGS         : out STD_LOGIC_VECTOR (7 downto 0);
      G_RST           : in STD_LOGIC);
end component;

component counter2b
port(CLK          : in  STD_LOGIC;
     CE           : in  STD_LOGIC;
     A0_COUNT     : out STD_LOGIC;
     A1_COUNT     : out STD_LOGIC;
     RESETAsync : in  STD_LOGIC);
end component;

component de_reg_mux
 port(C_L_K            : in STD_LOGIC;
      SEL_DEMUX        : in STD_LOGIC;
      SEL_MUX          : in STD_LOGIC;
      EN_REGISTERS     : in STD_LOGIC_VECTOR (7 downto 0);
      CE_LL            : in STD_LOGIC_VECTOR (7 downto 0);
      PAY_LOAD         : out STD_LOGIC_VECTOR (7 downto 0);
      INP_COMPAR       : out STD_LOGIC_VECTOR (7 downto 0);
      RST_REGISTERS    : in STD_LOGIC);
end component;

component ram_comp_unit
 port(IN_PAY          : in STD_LOGIC_VECTOR (7 downto 0);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      DATA_IN_UNIT : in STD_LOGIC_VECTOR (7 downto 0);
      WR_EN        : in STD_LOGIC_VECTOR (13 downto 0);
      ADD0         : in STD_LOGIC;
      ADD1         : in STD_LOGIC;
      G_1          : out STD_LOGIC;
      G_2          : out STD_LOGIC;
      LSB_1        : out STD_LOGIC;
      LSB_2        : out STD_LOGIC;
      G_MP         : out STD_LOGIC;
      LSB_MP       : out STD_LOGIC);
 end component;

 component tristate
  port (A_D       : in STD_LOGIC_VECTOR (7 downto 0);
        CLK_D     : in STD_LOGIC;
        D_EN      : in STD_LOGIC;
        RST_DFF   : in STD_LOGIC;
        NEW_CL    : out STD_LOGIC;
        O_D       : out STD_LOGIC_VECTOR (7 downto 0));
 end component;

 component mpenaddr
  port (MP_EN          :in STD_LOGIC;
        MP_ADDRESS     :out STD_LOGIC_VECTOR(4 downto 0));
 end component;

begin

 U0: state_ma
port map ( ATMCLK, RX_CELL, RXFE_BAR, count0, count1, g1_1, lsb1_1, g1_2,
lsb1_2, g1_mp,
lsb1_mp,ADDRESS_BUS,RRE_BAR,enable1,enable2,mpenable,ramen,muxen,demuxen
,counten, regenable,WR,RD,GL_RST);
 U1: counter2b
port map(ATMCLK,counten,count0,count1,GL_RST);
 U2: de_reg_mux
port
map(ATMCLK,demuxen,muxen,regenable,ATMRX_IN,ATMRX_OUT,compinput,GL_
RST);
 U3: ram_comp_unit
port
map(compinput,DATA_BUS,ramen,count0,count1,g1_1,g1_2,lsb1_1,lsb1_2,g1_mp,lsb1
_mp);
 U4: tristate
port map (pay,ATMCLK,enable1,GL_RST,NEW_CELL_STR1,ATMRX_OUT);
 U5: tristate
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
port map (pay,ATMCLK,enable2,GL_RST,NEW_CELL_STR2,ATMRX_OUT);
 U6: tristate
port map (pay,ATMCLK,mpenable,GL_RST,NEW_CELL_MP,DATA_BUS);
 U7: mpenaddr
 port map (mpenable,ADDRESS_OUT);

end arch_atm_receiver;
-- ========================================
--        atm_receiver configuration
-- ========================================
configuration atm_receiver_cfg of atm_receiver is
  for arch_atm_receiver
  end for;
end atm_receiver_cfg;
```

Τμήμα Β

```
-- ***************** SAR -PDU DISASSEMBLER
*******************************
--
--        This unit takes as input the ATM-SDU, separates the pointer, the AAL
--    header and the AAL PDU and takes them at the output.
--
--
*****************************************************************************
*******
library IEEE;
use IEEE.std_logic_1164.all;

-- ========================================
--        sar_pdu entity declaration
-- ========================================
entity sar_pdu is
  port(ATM_CLK      : in  STD_LOGIC;
       START        : in  STD_LOGIC;
       IN_G1        : in  STD_LOGIC;
       IN_PAYLOAD   : in  STD_LOGIC_VECTOR (7 downto 0);
       PTR_EN       : out STD_LOGIC;
       PTR_FLD      : out STD_LOGIC_VECTOR (7 downto 0);
       PDH_DATA     : out STD_LOGIC_VECTOR (7 downto 0);
       HEADER       : out STD_LOGIC_VECTOR (7 downto 0);
       HEADER_EN    : out STD_LOGIC;
       G_RST        : in  STD_LOGIC);
end;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ============================================
--        sar_pdu architecture
-- ============================================
architecture arch_sar_pdu of sar_pdu is
  signal even_odd   : STD_LOGIC;  -- '1' : even cell, '0' : odd cell
  signal count_byte  : integer range 0 to 47;

begin

 process(ATM_CLK,IN_G1,START,IN_PAYLOAD,G_RST)
 begin
  if G_RST ='1' then
    PTR_EN <= '0';
    PTR_FLD  <= "00000000";
    PDH_DATA <= "00000000";
    HEADER_EN <= '0';
    HEADER   <= "00000000";
    count_byte <= 0;
  elsif G_RST ='0' then
      if ATM_CLK'event and ATM_CLK ='1' then
       if START = '1' then
         if IN_G1 ='1' then
          case count_byte is
           when 0 =>
              even_odd <= IN_PAYLOAD(4);
              HEADER_EN <= '1';
              HEADER <= IN_PAYLOAD;
           when 1 =>
              if even_odd = '1' then
                PDH_DATA <= IN_PAYLOAD;
              elsif even_odd = '0' then
                PTR_EN <= '1';
                PTR_FLD <= IN_PAYLOAD;
              end if;
           when others =>
              PDH_DATA <= IN_PAYLOAD;
          end case;
          if count_byte = 47 then
            count_byte <= 0;
          else
            count_byte <= count_byte + 1;
          end if;
         end if;
       end if;
    end if;      -- end if clock
    else
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      PTR_EN <= '0';
      PTR_FLD  <= "00000000";
      PDH_DATA <= "00000000";
      HEADER_EN <= '0';
      HEADER   <= "00000000";
    end if;
 end process;

end arch_sar_pdu;

-- ============================================
--       sar_pdu configuration
-- ============================================
configuration sar_pdu_cfg of sar_pdu is
  for arch_sar_pdu
  end for;
end sar_pdu_cfg;



-- ********************** READ_HEADER
************************************
--       This unit reads and buffers thé AAL header.
--
**********************************************************************
******
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--       read_header entity declaration
-- ============================================
entity read_header is
  port(ATM_CLK          : in  STD_LOGIC;
       IN_HEADER_EN     : in  STD_LOGIC;
       IN_HEADER        : in  STD_LOGIC_VECTOR (7 downto 0);
       OUT_HEADER       : out STD_LOGIC_VECTOR (7 downto 0);
       G_RST            : in  STD_LOGIC);
end;

-- ============================================
--       read_header architecture
-- ============================================
architecture arch_read_header of read_header is

begin
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ

```
  process(ATM_CLK,IN_HEADER_EN,IN_HEADER,G_RST)
  begin
   if G_RST = '0' then
    if ATM_CLK'event and ATM_CLK = '1' then
     if IN_HEADER_EN = '1' then
      OUT_HEADER <= IN_HEADER;
     end if;
    end if;
   end if;
  end process;
end arch_read_header;

-- ============================================
--       read_header configuration
-- ============================================
configuration read_header_cfg of read_header is
  for arch_read_header
  end for;
end read_header_cfg;


-- ********************* CRC_GEN ********************************
--       This unit generates the CRC from the SN field of the
-- AAL header.
-- ***************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ================================
--       crc_gen entity declaration
-- ================================
entity crc_gen is
  port(IN_SN          : in  STD_LOGIC_VECTOR (3 downto 0);
       OUT_CRC_PAR    : out STD_LOGIC_VECTOR (3 downto 0));
end;

-- ================================================
--       crc_gen architecture
-- ================================================
architecture arch_crc_gen of crc_gen is

begin
 process(IN_SN)
 begin
  OUT_CRC_PAR(3) <= IN_SN(3) xor IN_SN(2) xor IN_SN(1);
  OUT_CRC_PAR(2) <= IN_SN(2) xor IN_SN(1) xor IN_SN(0);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
   OUT_CRC_PAR(1) <= IN_SN(3) xor IN_SN(2) xor IN_SN(0);
   OUT_CRC_PAR(0) <= IN_SN(3) xor IN_SN(1) xor IN_SN(0);
 end process;
end arch_crc_gen;

-- ============================================
--        crc_gen configuration
-- ============================================
configuration crc_gen_cfg of crc_gen is
  for arch_crc_gen
  end for;
end crc_gen_cfg;


-- ********************** CRC_COMPAR ***************************
--        This unit compares the CRC generated with the CRC field
--  of the AAL header.
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--        crc_compar entity declaration
-- ============================================
entity crc_compar is
  port(CRC_GENE     : in  STD_LOGIC_VECTOR (3 downto 0);
       CRC_HEAD     : in  STD_LOGIC_VECTOR (3 downto 0);
       CRC_STATE    : out STD_LOGIC;
       PAR_STATE    : out STD_LOGIC);
end;
-- ============================================
--        crc_compar architecture
-- ============================================
architecture arch_crc_compar of crc_compar is

 signal crc_bar2  : STD_LOGIC;
 signal crc_bar1  : STD_LOGIC;
 signal crc_bar0  : STD_LOGIC;
 signal crc_bar   : STD_LOGIC;
 signal par_bar   : STD_LOGIC;

begin
 process(CRC_GENE,CRC_HEAD)
 begin
   CRC_STATE <=  (not(CRC_GENE(3) xor CRC_HEAD(3))) and (not(CRC_GENE(2)
xor CRC_HEAD(2))) and (not(CRC_GENE(1) xor CRC_HEAD(1)));
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
   PAR_STATE <= not ( CRC_GENE(0) xor CRC_HEAD(0));  -- Active high
  end process;
end arch_crc_compar;

-- ============================================
--       crc_compar configuration
-- ============================================
configuration crc_compar_cfg of crc_compar is
  for arch_crc_compar
  end for;
end crc_compar_cfg;


-- ************************ CORCT_DETECT_MODE
**********************
--       This unit implements the algorithm which finds and corrects errors
-- of the AAL header according I.363 of ITU -T recommendation.
--
**************************************************************************
******
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--       corct_detect_mode entity declaration
-- ============================================
entity corct_detect_mode is
  port(ATM_CLK       : in  STD_LOGIC;
       HEADER_EN     : in  STD_LOGIC;
       HEADER_IN     : in  STD_LOGIC_VECTOR (7 downto 0);
       CRC_STATE_IN  : in  STD_LOGIC;
       PAR_STATE_IN  : in  STD_LOGIC;
       CSI_BIT       : out STD_LOGIC;
       HEAD_COR      : out STD_LOGIC;
       SN_EN         : out STD_LOGIC;
       SN_FIELD      : out STD_LOGIC_VECTOR (2 downto 0);
       G_RST         : in  STD_LOGIC);
end;

-- ============================================
--       corct_detect_mode architecture
-- ============================================
architecture arch_corct_detect_mode of corct_detect_mode is

 signal mode_state   : STD_LOGIC;
 signal sn_valid     : STD_LOGIC;  -- Active high
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
signal bit_correct : STD_LOGIC; -- Active high
signal head_stable : STD_LOGIC_VECTOR (7 downto 0);

begin

process(ATM_CLK,HEADER_EN,HEADER_IN,CRC_STATE_IN,PAR_STATE_IN,G
_RST)
 begin
  if G_RST = '1' then
    mode_state <= '1'; -- Correction Mode
    sn_valid  <= '0';  -- Invalid SN
    bit_correct <= '0';
    HEAD_COR <= '0';
    SN_EN   <= '0';
  elsif G_RST = '0' then

    if HEADER_EN = '1' then

     if mode_state = '1' then
      if CRC_STATE_IN = '1' and PAR_STATE_IN = '1' then
       sn_valid <= '1'; -- Valid SN
      elsif CRC_STATE_IN = '0' and PAR_STATE_IN = '0' then
       bit_correct <= '1'; --Single bit correction
       sn_valid <= '1';
       mode_state <= '0'; -- Detection Mode
      elsif CRC_STATE_IN = '1' and PAR_STATE_IN = '0' then
       sn_valid <= '1';
       mode_state <= '0';
      elsif CRC_STATE_IN = '0' and PAR_STATE_IN = '1' then
       -- No corrective action
       sn_valid <= '0';   -- Invalid SN
       mode_state <= '0';
      end if;
     elsif mode_state = '0' then
      if CRC_STATE_IN = '1' and PAR_STATE_IN = '1' then
       -- No corrective action
       sn_valid <= '1'; -- Valid SN
       mode_state <= '1'; -- Correction mode
      elsif CRC_STATE_IN = '0' and PAR_STATE_IN = '0' then
       -- No corrective action
       sn_valid <= '0';
      elsif CRC_STATE_IN = '1' and PAR_STATE_IN = '0' then
       -- No corrective action
       sn_valid <= '0';
      elsif CRC_STATE_IN = '0' and PAR_STATE_IN = '1' then
       -- No corrective action
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
        sn_valid <= '0';
       end if;
      end if;
     end if;
    if ATM_CLK'event and ATM_CLK = '1' then
     if sn_valid = '1' then
      if bit_correct = '1' then
       bit_correct <= '0';
       HEAD_COR <= '1';
       SN_EN <= '1';
       CSI_BIT <= HEADER_IN(3) xor HEADER_IN(1) xor HEADER_IN(0);
       SN_FIELD(2) <= HEADER_IN(3) xor HEADER_IN(2) xor HEADER_IN(1);
       SN_FIELD(1) <= HEADER_IN(3) xor HEADER_IN(2) xor HEADER_IN(0);
       SN_FIELD(0) <= HEADER_IN(2) xor HEADER_IN(1) xor HEADER_IN(0);
      elsif bit_correct = '0' then
       HEAD_COR <= '1';
       SN_EN <= '1';
       CSI_BIT <= HEADER_IN(7);
       SN_FIELD <= HEADER_IN(6 downto 4);
      end if;
     elsif sn_valid = '0' then
      HEAD_COR <= '0';
      SN_EN <= '0';
     end if;
    end if;      -- end if clock
   end if;  -- end if reset
  end process;

end arch_corct_detect_mode;

-- ============================================
--       corct_detect_mode configuration
-- ============================================
configuration corct_detect_mode_cfg of corct_detect_mode is
  for arch_corct_detect_mode
  end for;
end corct_detect_mode_cfg;


-- ***************** AAL- HEAD -COR *******************************
--
--       This unit checks the correction the header of the AAL layer
--
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ============================================
--       aal1_head_correct entity declaration
-- ============================================
entity aal1_head_correct is
  port(ATM_CLK     : in STD_LOGIC;
       IN_HEAD_EN  : in STD_LOGIC;
       IN_HEAD     : in STD_LOGIC_VECTOR (7 downto 0);
       CSI_BIT     : out STD_LOGIC;
       HEAD_COR    : out STD_LOGIC;
       SN_EN       : out STD_LOGIC;
       SN_FLD      : out STD_LOGIC_VECTOR (2 downto 0);
       G_RST       : in STD_LOGIC);
end;

-- ============================================
--       aal1_head_correct architecture
-- ============================================
architecture arch_aal1_head_correct of aal1_head_correct is

 signal crc_state_out   : STD_LOGIC;
 signal par_state_out   : STD_LOGIC;
 signal out_head        : STD_LOGIC_VECTOR (7 downto 0);
 signal crc_gen_par     : STD_LOGIC_VECTOR (3 downto 0);

 component read_header
  port(ATM_CLK       : in STD_LOGIC;
       IN_HEADER_EN  : in STD_LOGIC;
       IN_HEADER     : in STD_LOGIC_VECTOR (7 downto 0);
       OUT_HEADER    : out STD_LOGIC_VECTOR (7 downto 0);
       G_RST         : in STD_LOGIC);
 end component;

 component crc_gen
  port(IN_SN         : in STD_LOGIC_VECTOR (3 downto 0);
       OUT_CRC_PAR   : out STD_LOGIC_VECTOR (3 downto 0));
 end component;

 component crc_compar
   port(CRC_GENE    : in STD_LOGIC_VECTOR (3 downto 0);
        CRC_HEAD    : in STD_LOGIC_VECTOR (3 downto 0);
        CRC_STATE   : out STD_LOGIC;
        PAR_STATE   : out STD_LOGIC);
 end component;

 component corct_detect_mode
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
  port(ATM_CLK      : in  STD_LOGIC;
       HEADER_EN    : in  STD_LOGIC;
       HEADER_IN    : in  STD_LOGIC_VECTOR (7 downto 0);
       CRC_STATE_IN : in  STD_LOGIC;
       PAR_STATE_IN : in  STD_LOGIC;
       CSI_BIT      : out STD_LOGIC;
       HEAD_COR     : out STD_LOGIC;
       SN_EN        : out STD_LOGIC;
       SN_FIELD     : out STD_LOGIC_VECTOR (2 downto 0);
       G_RST        : in  STD_LOGIC);
 end component;

begin

 U0: read_header
  port map(ATM_CLK,IN_HEAD_EN,IN_HEAD,out_head,G_RST);
 U1: crc_gen
  port map(out_head(7 downto 4),crc_gen_par);
 U2: crc_compar
  port map(crc_gen_par,out_head(3 downto 0),crc_state_out,par_state_out);
 U3: corct_detect_mode
  port
map(ATM_CLK,IN_HEAD_EN,out_head,crc_state_out,par_state_out,CSI_BIT,HEAD_
COR,SN_EN,SN_FLD,G_RST);

end arch_aal1_head_correct;

-- ============================================
--       aal1_head_correct configuration
-- ============================================
configuration aal1_head_correct_cfg of aal1_head_correct is
  for arch_aal1_head_correct
  end for;
end aal1_head_correct_cfg;


-- ************************ FRAME -SYNC
**********************************
--       This unit produces the pulse which synchronizes the whole AALRX.
--
**************************************************************************
****
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--        fra_sync entity declaration
-- ================================================
entity fra_sync is
  port(FRAM_CLK        :in  STD_LOGIC;
       RESET           :in  STD_LOGIC;
       NEW_CELL        :in  STD_LOGIC;
       POINTER_EN      :in  STD_LOGIC;
       POINTER         :in  STD_LOGIC_VECTOR (7 downto 0);
       HEAD_COR        :in  STD_LOGIC;
       SEQ_CHECK       :in  STD_LOGIC;
       ODD_CHECK       :in  STD_LOGIC;
       SYNC_PULSE      :out STD_LOGIC);
end;

-- ================================================
--        fra_sync architecture
-- ================================================
architecture arch_fra_sync of fra_sync is
   signal ptr     : integer range 0 to 31;
   signal COUNTER : integer range 0 to 31 :=0;
   signal sync_ind: STD_LOGIC;

begin
  process(FRAM_CLK,RESET,NEW_CELL,POINTER_EN,POINTER)
        begin
          if (RESET ='1') then
               COUNTER <= 0;
           ptr <= 0;
               SYNC_PULSE <='0';
          elsif (RESET ='0') then
             if (sync_ind ='1') then
                  sync_ind <='0';
                  SYNC_PULSE <='0';
                   COUNTER <=0;
                end if;
              if (FRAM_CLK'event and FRAM_CLK ='1') then
                if (NEW_CELL ='1') then
               for i in 0 to 4  loop
                  if POINTER(i) = '1' then
                    ptr <= ptr + 2**i;
                  end if;
               end loop;

                if (POINTER_EN ='1') then
                  if ((HEAD_COR ='1' or SEQ_CHECK ='1') and ODD_CHECK ='0')
then
```

```
                    if (COUNTER = ptr) then
                   SYNC_PULSE <='1';
                     sync_ind <='1';
                  end if;
                end if;
              end if;
                if (COUNTER = 31) then
                 SYNC_PULSE <= '1';
                 sync_ind <='1';
              else
                COUNTER <= COUNTER + 1;
              end if;
               end if;   -- end if new cell
             end if;      -- end if ATM_CLK
          end if;        --- end if reset
      end process;
end arch_fra_sync;

-- ============================================
--      fra_sync configuration
-- ============================================
configuration fra_sync_cfg of fra_sync is
      for arch_fra_sync
      end for;
end fra_sync_cfg;


-- ********************* D Flip-Flop ********************************
--      This unit is a d  flip -flop.
-- ******************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ===================================
--      dflflop entity declaration
-- ===================================
entity dflflop is
  port (A_D        : in  STD_LOGIC;
        CLK_D      : in  STD_LOGIC;
        RST_DFF    : in  STD_LOGIC;
        O_D        : out STD_LOGIC);
end;

-- ===================================
--      dflflop architecture
-- ===================================
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
architecture arch_dflflop of dflflop is

begin
 process(A_D,CLK_D,RST_DFF)
 begin
   if RST_DFF = '1' then
     O_D <= '0';
   elsif RST_DFF = '0' then

    if CLK_D'event and CLK_D ='1' then
          O_D <= A_D;
    end if;
   end if;
 end process;
end arch_dflflop;

-- ============================================
--       dflflop configuration
-- ============================================
configuration dflflop_cfg of dflflop is
  for arch_dflflop
  end for;
end dflflop_cfg;


--****************** FRAME SYNC UNIT
**********************************
--
--   FRAME SYNCHRONIZE UNIT
--
--
********************************************************************************
***
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--       fram_entity entity declaration
-- ============================================
entity fram_entity is
port (ATM_CLK              :in STD_LOGIC;
     PDH_CLK               :in STD_LOGIC;
     RESET                 :in STD_LOGIC;
     NEW_CELL              :in STD_LOGIC;
     POINTER_EN            :in STD_LOGIC;
     POINTER               :in STD_LOGIC_VECTOR (7 downto 0);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
    HEAD_COR            :in STD_LOGIC;
    SEQ_CHECK           :in STD_LOGIC;
    ODD_CHECK           :in STD_LOGIC;
    SYNC_OUT            :out STD_LOGIC);
end;

-- ============================================
--      fram_entity architecture
-- ============================================
architecture arch_fram_entity of fram_entity is

signal fram_dfl      :STD_LOGIC;
component fra_sync
  port(FRAM_CLK       :in STD_LOGIC;
       RESET          :in STD_LOGIC;
       NEW_CELL       :in STD_LOGIC;
       POINTER_EN     :in STD_LOGIC;
       POINTER        :in STD_LOGIC_VECTOR (7 downto 0);
       HEAD_COR       :in STD_LOGIC;
       SEQ_CHECK      :in STD_LOGIC;
       ODD_CHECK      :in STD_LOGIC;
       SYNC_PULSE     :out STD_LOGIC);
end component;

component dflflop
  port(A_D      : in STD_LOGIC;
       CLK_D    : in STD_LOGIC;
       RST_DFF  : in STD_LOGIC;
       O_D      : out STD_LOGIC);
end component;

begin

 U0: fra_sync
 port map(FRAM_CLK => ATM_CLK,RESET => RESET,NEW_CELL =>
NEW_CELL,POINTER_EN => POINTER_EN,POINTER =>
POINTER,HEAD_COR => HEAD_COR,SEQ_CHECK =>
SEQ_CHECK,ODD_CHECK
=> ODD_CHECK,SYNC_PULSE => fram_dfl);
 U1: dflflop
 port map(A_D => fram_dfl,CLK_D => PDH_CLK,RST_DFF => RESET, O_D =>
SYNC_OUT);

end arch_fram_entity;

-- ============================================
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--        fram_entity configuration
-- ============================================
configuration fram_entity_cfg of fram_entity is
        for arch_fram_entity
        end for;
end fram_entity_cfg;


-- ********************* IDLE BYTE INS. ******************************
--        This unit is a buffer that inserts the dummy bytes that
-- reads from the microprocessor into FIFO -MUX unit.
-- ***********************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--        idle_byte entity declaration
-- ============================================
entity idle_byte is
  port(ADDR_BUS          :in  STD_LOGIC_VECTOR (2 downto 0);
       DATA_IN           :inout  STD_LOGIC_VECTOR (7 downto 0);
        CLK_MP           :in  STD_LOGIC;
        CLK_DUM          :in  STD_LOGIC;
        RESET            :in  STD_LOGIC;
        WR               :in  STD_LOGIC;
        RD               :in  STD_LOGIC;
       DUM_EN            :out STD_LOGIC;
        DUM_DATA         :out STD_LOGIC_VECTOR (7 downto 0));
end;

-- ============================================
--        idle_byte architecture
-- ============================================
architecture arch_idle_byte of idle_byte is

   signal DATA : STD_LOGIC_VECTOR (7 downto 0);
begin

  process(ADDR_BUS,DATA_IN,CLK_MP,CLK_DUM,RESET,WR,RD)
     begin
        if (RESET ='1') then
         DUM_DATA <=(others=>'0');
        end if;
        if (ADDR_BUS ="000") then
          if (CLK_MP'event and CLK_MP='1') then
           if (WR ='1') then
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
            DATA <= DATA_IN;
          elsif (WR ='0') then
              if (RD ='1') then
              DATA_IN <= DATA;
              end if;
          end if;
        end if;
        if (CLK_DUM'event and CLK_DUM='1') then
          DUM_EN <= '1';
          DUM_DATA <= DATA;
        end if;
      end process;
end arch_idle_byte;

-- ============================================
--      idle_byte configuration
-- ============================================
configuration idle_byte_cfg of idle_byte is
      for arch_idle_byte
      end for;
end idle_byte_cfg;


-- ********************** FIFO_MUX
**************************************
--      This unit arranges the input interface with the FIFO. It multiplexes
-- the AAL PDU from SAR_PDU DISASSEMBLER and the dummy bytes
-- from IDLE BYTE INS.
--
**********************************************************************
******
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--      fifo_mux entity declaration
-- ============================================
entity fifo_mux is
      port(WRIT_EN          :in STD_LOGIC;
          DIASS_DATA        :in STD_LOGIC_VECTOR (7 downto 0);
          DUM_EN            :in STD_LOGIC;
          IDLE_DATA         :in STD_LOGIC_VECTOR (7 downto 0);
          HEAD_COR          :in STD_LOGIC;
          ODD_CHECK         :in STD_LOGIC;
          SEQ_CHECK         :in STD_LOGIC;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
        IN_FIFO                 :out STD_LOGIC_VECTOR (7 downto 0);
        WRIT                    :out STD_LOGIC);
end;

-- ============================================
--      fifo_mux architecture
-- ============================================
architecture arch_fifo_mux of fifo_mux is
      signal COUNTER      : INTEGER range 1 to 47;
      signal COUNT_END : INTEGER range 46 to 47;
begin

process(WRIT_EN,DIASS_DATA,DUM_EN,IDLE_DATA,HEAD_COR,ODD_CHEC
K,SEQ_CHECK)
      begin
        if (ODD_CHECK ='1') then
            COUNT_END <= 47;
        else
            COUNT_END <= 46;
        end if;
        if (HEAD_COR ='1') then
            if (SEQ_CHECK ='1') then
              if WRIT_EN ='1' then
                WRIT <='1';
              IN_FIFO <= DIASS_DATA;
              end if;
            else
             if (COUNTER <= COUNT_END) then
                if DUM_EN ='1' then
                WRIT <='1';
                 IN_FIFO <= IDLE_DATA;
                end if;
             end if;
         end if;
        else
          if (COUNTER <= COUNT_END) then
            if DUM_EN ='1' then
            WRIT <='1';
             IN_FIFO <= IDLE_DATA;
            end if;
        end if;
        end if;
        COUNTER <= COUNTER + 1;
        if (COUNTER = COUNT_END) then
            COUNTER <= 1;
        end if;
```

```
    end process;
end arch_fifo_mux;

-- ============================================
--        fifo_mux configuration
-- ============================================
configuration fifo_mux_cfg of fifo_mux is
      for arch_fifo_mux
      end for;
end fifo_mux_cfg;


-- ************************ SN_CHECK ****************************
--        This unit checks the SN of the AAL header.
-- ***************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
-- ============================================
--        sn_check entity declaration
-- ============================================
entity sn_check is
  port(ATM_CLK      : in  STD_LOGIC;
       HEAD_COR     : in  STD_LOGIC;
       SN_EN        : in  STD_LOGIC;
       SN_FLD       : in  STD_LOGIC_VECTOR (2 downto 0);
       SEQ_CHECK    : out STD_LOGIC;
       ODD_CHECK    : out STD_LOGIC;
       G_RST        : in  STD_LOGIC);
end;

-- ============================================
--        sn_check architecture
-- ============================================
architecture arch_sn_check of sn_check is
signal sn_state : integer range 0 to 7;
signal sn_count : integer range 0 to 7;
signal st_check : STD_LOGIC;
signal odd_st   : STD_LOGIC;
signal i        : integer range 0 to 2 := 0;
begin
 process(ATM_CLK,HEAD_COR,SN_EN,SN_FLD,G_RST)
 begin
  if G_RST = '1' then
    SEQ_CHECK <= '0';
    ODD_CHECK <= '0';
    sn_state <= 0;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      sn_count <= 0;
      st_check <= '0';
      odd_st <= '0';
    elsif G_RST = '0' then
     if ATM_CLK'event and ATM_CLK = '1' then
      if HEAD_COR = '1' then
       if SN_EN = '1' then
          for i in 0 to 2 loop
           if SN_FLD(i) = '1' then
           sn_count <= sn_count + 2**i;
           end if;
         end loop;
        if SN_FLD(0) = '1' then
            ODD_CHECK <= '1';
            elsif SN_FLD(0) = '0' then
            ODD_CHECK <= '0';
             end if;
             if (sn_count <= sn_state) then
                    SEQ_CHECK <= '0';
             end if;
           if (sn_state = (sn_count - 1) and (sn_count /= 0)) then
                  SEQ_CHECK <= '1';
           elsif (sn_count /= 0) then
                SEQ_CHECK <= '0';
        else
              SEQ_CHECK <= '1';
          end if;
              sn_state <= sn_count;
      end if;
      end if;
     end if;
    end if;
end process;
end arch_sn_check;

-- ============================================
--        sn_check configuration
-- ============================================
configuration sn_check_cfg of sn_check is
  for arch_sn_check
  end for;
end sn_check_cfg;


--****************************** RTS CREATION
**********************************
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--
--      Takes as inputs the header of the AAL_PDH,the HEAD_COR,the
-- ATMCLK and the RESET and generates the RTS_FIELD and the signal rts_we. The
SN
-- check is calculated when the register was reading.
--
**********************************************************************
*****
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;

-- ================================
--      rts_crte entity declaration
-- ================================
entity rts_crte is
   port(ATM_CLK          :in  STD_LOGIC;
        CSI_BIT          :in  STD_LOGIC;
        HEAD_COR         :in  STD_LOGIC;
        ODD_CHECK        :in  STD_LOGIC;
        SEQ_CHECK        :in  STD_LOGIC;
        RESET            :in  STD_LOGIC;
        RTS              :out STD_LOGIC_VECTOR (3 downto 0);
        RTS_WE           :out STD_LOGIC);
end;

-- ================================
--      rts_crte architecture
-- ================================
architecture arch_rts_crte of rts_crte is
 signal I   :INTEGER range 0 to 3 := 3;
begin

 process (ATM_CLK,CSI_BIT,HEAD_COR,SEQ_CHECK,RESET)
 begin
       if (RESET = '1') then
         RTS <= (others => '0');
         RTS_WE <= '0';
       elsif (ATM_CLK'event and ATM_CLK = '1') then
         if (I = 3) then
           RTS_WE <='0';
         end if;
              if (HEAD_COR ='1') then
                if (SEQ_CHECK = '1') then
                  if (ODD_CHECK ='1') then
                      RTS(I) <= CSI_BIT;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
                    if (I = 0) then
                      RTS_WE <='1';
                      I <= 3;
                    else
                      I <= I - 1;
                    end if;
                  end if;
                else
                  RTS_WE <='0';
                end if;
              else
                RTS_WE <= '0';
            end if;
        end if;
        end process;
end arch_rts_crte;

-- ============================================
--       rts_crte configuration
-- ============================================
configuration rts_crte_cfg of rts_crte is
        for arch_rts_crte
        end for;
end rts_crte_cfg;


--  ************ Divide by 372 the parallel PDHCLK ***************
--
--       Divides the input clock by 372 providing an one period
--  low pulse on the output when the count equals 371
--      There are synchronous and asynchronous reset signals
--  and there is an enable signal.
--
--  ************************************************************************

library IEEE;
use IEEE.STD_LOGIC_1164.all;


-- ============================
--       div372 entity declaration
-- ============================

entity div372 is
 port (div        : out STD_LOGIC;
     clk          : in  STD_LOGIC;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      enable      : in  STD_LOGIC;
      resetAsync : in  STD_LOGIC);
end;

-- ============================================
--       div372 architecture
-- ============================================

architecture arch_div372 of div372 is

 constant tpd_reset_to_count : time := 10 ns;
 constant tpd_clk_to_count   : time :=  5 ns;

 signal count: STD_LOGIC_VECTOR(8 downto 0);

 -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 function increment(val : STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
  -- normalize the indexing
  alias input    : STD_LOGIC_VECTOR(val'length downto 1) is val;
  variable result : STD_LOGIC_VECTOR(input'range) := input;
  variable carry  : STD_LOGIC := '1';

 begin
  for i in input'low to input'high loop
   result(i) := input(i) xor carry;
   carry := input(i) and carry;
   exit when carry = '0';
  end loop;

  return result;
 end increment;
 -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

begin

 ctr: process(clk, resetAsync)
 begin
  if (resetAsync = '1') then
   if resetAsync'event then
    count <= (others => '0') after tpd_reset_to_count;
    div <= '1';
   end if;
  elsif (resetAsync = '0') then
   if clk'event and (clk = '1') then
    if (enable = '1') then
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      if count = "101110011" then -- count = 371
        count <= (others => '0') after tpd_clk_to_count;
        div <= '1' after tpd_clk_to_count;
      else
        count <= increment(count) after tpd_clk_to_count;
        if count = "101110010" then
          div <= '0' after tpd_clk_to_count;
        else
          div <= '1' after tpd_clk_to_count;
        end if;
      end if;
    end if;
   end if;
  end if;
 end process;

end arch_div372;


-- ============================================
--       div372 configuration
-- ============================================
configuration div372_cfg of div372 is
 for arch_div372
 end for;
end div372_cfg;


-- ************************** Divide by 64 the ATMCLK *****************
--
--       Divides the input clock by 372 providing an one period
--  low pulse on the output when the count equals 371
--      There are synchronous and asynchronous reset signals
--  and there is an enable signal.

--
*********************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--       div64 entity declaration
-- ============================================
entity div64 is
  port(ATM_CLK    : in  STD_LOGIC;
       ENABLE     : in  STD_LOGIC;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      OUT_CLK     : out STD_LOGIC;
      G_RST       : in  STD_LOGIC);
end;

-- ============================================
--       div64 architecture
-- This unit generates the PDHCLK from RTS field of the AAL header.
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

-- ============================================
--       pdhclk_rec entity declaration
-- ============================================
entity pdhclk_rec is

  port(ATM_CLK     : in  STD_LOGIC;
       PDH_CLK     : in  STD_LOGIC;
       ENABLE      : in  STD_LOGIC;
       RTS_FIELD   : in  STD_LOGIC_VECTOR (3 downto 0);
       VCO_CONTR   : out integer range -15 to 15;
       G_RST       : in  STD_LOGIC);
end;

-- ============================================
--       pdhclk_rec architecture
-- ============================================
architecture arch_pdhclk_rec of pdhclk_rec is

 signal ena_ble  : STD_LOGIC;
 signal clk_64   : STD_LOGIC;
 signal c_out    : integer range 0 to 15;
 signal rts_g    : integer range 0 to 15;

component div372

 port (div        : out STD_LOGIC;
       clk        : in  STD_LOGIC;
       enable     : in  STD_LOGIC;
       resetAsync : in  STD_LOGIC);

end component;

component div64

  port(ATM_CLK    : in  STD_LOGIC;
       ENABLE     : in  STD_LOGIC;
       OUT_CLK    : out STD_LOGIC;
       G_RST      : in  STD_LOGIC);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
end component;

component counter4bit
  port(CLK      : in  STD_LOGIC;
       COUNT    : out integer range 0 to 15;
       G_RST    : in STD_LOGIC);
end component;

component latch4bit
  port(LATCH_EN    : in  STD_LOGIC;
       IN_COUNT    : in  integer range 0 to 15;
       RTS_OUT     : out integer range 0 to 15);
end component;

component rts_compar
  port(RTS_FLD     : in  STD_LOGIC_VECTOR (3 downto 0);
       RTS_GEN     : in  integer range 0 to 15;
       ENABLE      : in  STD_LOGIC;
       VCO_CONTR   : out integer range -15 to 15);
end component;

begin
    U0: div372
    port map(ena_ble,PDH_CLK,ENABLE,G_RST);
    U1: div64
    port map(ATM_CLK,ENABLE,clk_64,G_RST);
    U2: counter4bit
    port map(clk_64,c_out,G_RST);
    U3: latch4bit
    port map(ena_ble,c_out,rts_g);
    U4: rts_compar
    port map(RTS_FIELD,rts_g,ena_ble,VCO_CONTR);

end arch_pdhclk_rec;
-- ============================================
--       pdhclk_rec configuration
-- ============================================
configuration pdhclk_rec_cfg of pdhclk_rec is
    for arch_pdhclk_rec
    end for;
end pdhclk_rec_cfg;

-- ********************* OUTPUT UNIT *********************************
--
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--      This unit adjusts the data from the FIFO to the PDH format.
--
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ========================================
--      out_unit entity declaration
-- ========================================
entity out_unit is
        port(OUT_CLK        :in  STD_LOGIC;
             RESET          :in  STD_LOGIC;
             SYNC_PULSE     :in  STD_LOGIC;
             FIFO_OUT       :in  STD_LOGIC_VECTOR (7 downto 0);
             RD             :out  STD_LOGIC;
             PDH_PAR        :out  STD_LOGIC_VECTOR (7 downto 0));
end;
-- ========================================
--      out_unit architecture
-- ========================================
architecture arch_out_unit of out_unit is
signal counter :INTEGER range 0 to 31 := 0;
begin
    process(OUT_CLK,RESET,SYNC_PULSE,FIFO_OUT)
    begin
    if (RESET ='1') then
    PDH_PAR <= (others =>'0');
    elsif (RESET ='0') then
    if (OUT_CLK'event and OUT_CLK ='1') then
    if (SYNC_PULSE ='1' and counter <=31) then
    if (counter = 31) then
    counter <=0;
    else
    counter <= counter + 1;
    end if;
        RD <= '1';
        PDH_PAR <= FIFO_OUT;
    end if;
    end if;
    end if;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
    end process;
end arch_out_unit;
-- ============================================
--        out_unit configuration
-- ============================================
configuration out_unit_cfg of out_unit is
    for arch_out_unit
    end for;
end out_unit_cfg;

--*****************************************************************************
--
--        A A L -1  layer of the Receiver for one stream (AALRX)
--
--
--*****************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--        aal1 entity declaration
-- ============================================
entity aal1 is
  port(ATM_CLK          :in  STD_LOGIC;
       PDH_CLK          :in  STD_LOGIC;
       CLK_MP           :in  STD_LOGIC;
       GRESET           :in  STD_LOGIC;
       START            :in  STD_LOGIC;
       IN_G1            :in  STD_LOGIC;
       IN_PAYLOAD       :in  STD_LOGIC_VECTOR (7 downto 0);
       DATA_IN          :inout  STD_LOGIC_VECTOR (7 downto 0);
       ADDR_BUS         :in  STD_LOGIC_VECTOR (2 downto 0);
       RD               :in  STD_LOGIC;
       WR               :in  STD_LOGIC;
      ENR               :in  STD_LOGIC;
      FIFO_OUT          :in STD_LOGIC_VECTOR (7 downto 0);
       SYNC_PULSE       :out STD_LOGIC;
      WRIT              :out  STD_LOGIC;
      FIFO_IN           :out STD_LOGIC_VECTOR (7 downto 0);
       VCO_CONTR        :out integer range -15 to 15;
      PDH_PAR           :out STD_LOGIC_VECTOR (7 downto 0));

end;
-- ============================================
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
--      aal1 architecture
-- ================================================
architecture arch_aal1 of aal1 is

signal pointer_ena   : STD_LOGIC;
signal pointer_flda  : STD_LOGIC_VECTOR (7 downto 0);
signal pdh_ddata     : STD_LOGIC_VECTOR (7 downto 0);
signal heeader       : STD_LOGIC_VECTOR (7 downto 0);
signal heeader_ena   : STD_LOGIC;
signal csi_byte      : STD_LOGIC;
signal heead_corct   : STD_LOGIC;
signal sn_ena        : STD_LOGIC;
signal sn_flda       : STD_LOGIC_VECTOR (2 downto 0);
signal odda_chk      : STD_LOGIC;
signal seqq_chk      : STD_LOGIC;
signal rts_field     : STD_LOGIC_VECTOR (3 downto 0);
signal rts_ena       : STD_LOGIC;
signal pulse_ena     : STD_LOGIC;
signal wr_en         : STD_LOGIC;
signal idle_en       : STD_LOGIC;
signal idle_ddatta   : STD_LOGIC_VECTOR (7 downto 0);

component sar_pdu

  port(ATM_CLK      : in STD_LOGIC;
       START        : in STD_LOGIC;
       IN_G1        : in STD_LOGIC;
       IN_PAYLOAD   : in STD_LOGIC_VECTOR (7 downto 0);
       PTR_EN       : out STD_LOGIC;
       PTR_FLD      : out STD_LOGIC_VECTOR (7 downto 0);
       WE_FIFO      : out STD_LOGIC;
       PDH_DATA     : out STD_LOGIC_VECTOR (7 downto 0);
       HEADER       : out STD_LOGIC_VECTOR (7 downto 0);
       HEADER_EN    : out STD_LOGIC;
       G_RST        : in STD_LOGIC);
end component;

component aal1_head_correct

  port(ATM_CLK     : in STD_LOGIC;
       IN_HEAD_EN  : in STD_LOGIC;
       IN_HEAD     : in STD_LOGIC_VECTOR (7 downto 0);
       CSI_BIT     : out STD_LOGIC;
       HEAD_COR    : out STD_LOGIC;
       SN_EN       : out STD_LOGIC;
       SN_FLD      : out STD_LOGIC_VECTOR (2 downto 0);
       G_RST       : in STD_LOGIC);
end component;
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
component rts_crte
   port(ATM_CLK      :in STD_LOGIC;
      CSI_BIT        :in STD_LOGIC;
        HEAD_COR     :in STD_LOGIC;
      ODD_CHECK      :in STD_LOGIC;
        SEQ_CHECK    :in STD_LOGIC;
        RESET        :in STD_LOGIC;
        RTS          :out STD_LOGIC_VECTOR (3 downto 0);
        RTS_WE       :out STD_LOGIC);
end component;

component fram_entity
 port(ATM_CLK      :in STD_LOGIC;
    PDH_CLK        :in STD_LOGIC;
    RESET          :in STD_LOGIC;
    NEW_CELL       :in STD_LOGIC;
    POINTER_EN     :in STD_LOGIC;
    POINTER        :in STD_LOGIC_VECTOR (7 downto 0);
    HEAD_COR       :in STD_LOGIC;
    SEQ_CHECK      :in STD_LOGIC;
    ODD_CHECK      :in STD_LOGIC;
    SYNC_OUT       :out STD_LOGIC);
end component;

component pdhclk_rec
 port(ATM_CLK     : in STD_LOGIC;
    PDH_CLK       : in STD_LOGIC;
    ENABLE        : in STD_LOGIC;
    RTS_FIELD     : in STD_LOGIC_VECTOR (3 downto 0);
    VCO_CONTR     : out integer range -15 to 15;
    G_RST         : in STD_LOGIC);
end component;

component fifo_mux
 port(WRIT_EN       :in STD_LOGIC;
      DIASS         :in STD_LOGIC_VECTOR (7 downto 0);
      DUM_EN        :in STD_LOGIC;
    IDLE_DATA       :in STD_LOGIC_VECTOR (7 downto 0);
    HEAD_COR        :in STD_LOGIC;
    ODD_CHECK       :in STD_LOGIC;
    SEQ_CHECK       :in STD_LOGIC;
    IN_FIFO         :out STD_LOGIC_VECTOR(7 downto 0);
    WRIT            :out STD_LOGIC);
end component;

component idle_byte
 port(ADDR_BUS    :in STD_LOGIC_VECTOR (2 downto 0);
```

```
    DATA_IN      :inout STD_LOGIC_VECTOR (7 downto 0);
    CLK_MP       :in STD_LOGIC;
    CLK_DUM      :in STD_LOGIC;
    RESET        :in STD_LOGIC;
    WR           :in STD_LOGIC;
    RD           :in STD_LOGIC;
    DUM_EN       :out STD_LOGIC;
    DUM_DATA     :out STD_LOGIC_VECTOR (7 downto 0));
end component;

component out_mux

    port(OUT_CLK      :in STD_LOGIC;
        RESET         :in STD_LOGIC;
        SYNC_PULSE :in STD_LOGIC;

        FIFO_OUT     :in STD_LOGIC_VECTOR (7 downto 0);
        RD           :out STD_LOGIC;
        PDH_PAR      :out STD_LOGIC_VECTOR (7 downto 0));
end component;

component sn_check

 port(ATM_CLK    : in STD_LOGIC;
    HEAD_COR   : in STD_LOGIC;
    SN_EN      : in STD_LOGIC;
    SN_FLD     : in STD_LOGIC_VECTOR (2 downto 0);
    SEQ_CHECK  : out STD_LOGIC;
   ODD_CHECK   : out STD_LOGIC;
   G_RST       : in STD_LOGIC);
end component;

begin

U0 :sar_pdu
port
map(ATM_CLK,START,IN_G1,IN_PAYLOAD,PTR_EN,PTR_FIELD,wr_en,pdh_ddat
a,heeader,heeader_ena,GRESET);

U1 :aal1_head_correct

port
map(ATM_CLK,heeader_ena,heeader,csi_byte,heead_corct,sn_ena,sn_flda,GRESET);

U2: rts_crte

port
map(ATM_CLK,csi_byte,heead_corct,odda_chk,seqq_chk,GRESET,rts_field,rts_ena);


U4 :pdhclk_rec

port map(ATM_CLK,PDH_CLK,rts_ena,rts_field,VCO_CONTR,GRESET);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
U5 :fifo_mux
port map(wr_en, pdh_ddata, idle_en, idle_ddatta, heead_corct, odda_chk,
seqq_chk,FIFO_IN,WRIT);
U6 :idle_byte
port
map(ADDR_BUS,DATA_IN,CLK_MP,CLK_MP,GRESET,WR,RD,idle_en,idle_ddatta
);
U7 :out_mux
port map(PDH_CLK,GRESET,pulse_ena,FIFO_OUT,RD_FIFO,PDH_PAR);
U8 : sn_check
port map(ATM_CLK,heead_corct,sn_ena,sn_flda,seqq_chk,odda_chk,GRESET);

U9 : fram_entity
port map
(ATM_CLK,PDH_CLK,GRESET,START,pointer_ena,pointer_flda,heead_corct,odda_c
hk,seqq_chke,pulse_ena);
SYNC_PULSE <= pulse_ena;
end arch_aal1;
-- ============================================
--       aal1 configuration
-- ============================================
configuration aal1_cfg of aal1 is
    for arch_aal1
    end for;
end aal1_cfg;
```

==Τμήμα Γ==

```
-- ************************** ATM_PDH
************************************
--       This unit is an ATM/PDH receiver
--
**************************************************************************
**

library IEEE;
use IEEE.std_logic_1164.all;

-- ============================================
--       atm_pdh entity declaration
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
-- ============================================================
entity atm_pdh is
  port(ATM_CLK        : in  STD_LOGIC;
       PDH_CLK        :in STD_LOGIC;
       CLK_MP         :in STD_LOGIC;
       GL_RST         : in  STD_LOGIC;
       ADDRESS_BUS    :inout  STD_LOGIC_VECTOR (4 downto 0);
       DATA_BUS       :inout  STD_LOGIC_VECTOR (7 downto 0);
       WR             :in  STD_LOGIC;
       RD             :in  STD_LOGIC;
       ATMRX_IN       :in  STD_LOGIC_VECTOR (7 downto 0);
       RX_CELL        :in  STD_LOGIC; -- high active
       RXFE_BAR       :in  STD_LOGIC; -- low active
       RRE_BAR        :out STD_LOGIC; -- low active
       FIFO_IN1       :out STD_LOGIC;
       RD_FIFO1       :out STD_LOGIC;
       SYNC_PULSE1    :out STD_LOGIC;
       VCO_CONTR1     :out integer range -15 to 15;
       PDH_PAR1       :out STD_LOGIC_VECTOR (7 downto 0);
       FIFO_IN2       :out STD_LOGIC;
       RD_FIFO2       :out STD_LOGIC;
       SYNC_PULSE2    :out STD_LOGIC;
       VCO_CONTR2     :out integer range -15 to 15;
       PDH_PAR2       :out STD_LOGIC_VECTOR (7 downto 0));
end;

-- ============================================================
--       atm_pdh architecture
-- ============================================================
architecture arch_atm_pdh of atm_pdh is

 signal stream1    : STD_LOGIC;
 signal stream2    : STD_LOGIC;
 signal atm_sdu    : STD_LOGIC_VECTOR (7 downto 0);
component atm_receiver
 port (ATMCLK            :in  STD_LOGIC;
       GL_RST            :in  STD_LOGIC;
       ADDRESS_BUS       :in  STD_LOGIC_VECTOR (4 downto 0);
       DATA_BUS          :in  STD_LOGIC_VECTOR (7 downto 0);
       WR                :in  STD_LOGIC;
       RD                :in  STD_LOGIC;
       ATMRX_IN          :in  STD_LOGIC_VECTOR (7 downto 0);
       RX_CELL           :in  STD_LOGIC; -- high active
       RXFE_BAR          :in  STD_LOGIC; -- low active
       RRE_BAR           :out STD_LOGIC; -- low active
       ADDRESS_OUT       :out STD_LOGIC_VECTOR (4 downto 0);
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
      NEW_CELL_STR1  :out STD_LOGIC;
      NEW_CELL_STR2  :out STD_LOGIC;
      NEW_CELL_MP    :out STD_LOGIC;
      ATMRX_OUT      :out STD_LOGIC_VECTOR (7 downto 0));
end component;

component aal1
 port(ATM_CLK        :in  STD_LOGIC;
      PDH_CLK        :in  STD_LOGIC;
      CLK_MP         :in  STD_LOGIC;
      GRESET         :in  STD_LOGIC;
      START          :in  STD_LOGIC;
      IN_G1          :in  STD_LOGIC;
      IN_PAYLOAD     :in  STD_LOGIC_VECTOR (7 downto 0);
      DATA_IN        :inout  STD_LOGIC_VECTOR (7 downto 0);
      ADDR_BUS       :in  STD_LOGIC_VECTOR (2 downto 0);
      RD             :in  STD_LOGIC;
      WR             :in  STD_LOGIC;
      ENR            :in  STD_LOGIC;
      FIFO_OUT       :in  STD_LOGIC_VECTOR (7 downto 0);
      SYNC_PULSE     :out STD_LOGIC;
      WRIT           :out STD_LOGIC;
      FIFO_IN        :out STD_LOGIC_VECTOR (7 downto 0);
      VCO_CONTR      :out integer range -15 to 15;
      PDH_PAR        :out STD_LOGIC_VECTOR (7 downto 0));
end component;


begin
  U0: atm_receiver
port
map(ATM_CLK,GL_RST,ADDRESS_BUS,DATA_BUS,WR,RD,ATMRX_IN,RX_CE
LL,RXFE_BAR,RRE_BAR,stream1,stream2,stream_mp,atm_sdu);
  U1: aal1
port
map(ATM_CLK,PDH_CLK,CLK,_MP,GRESET,stream1,atm_sdu,DATA_BUS,ADDR
ESS_BUS,RD,WR,ENR,FIFO_OUT1,SYNC_PULSE1,WRIT1,FIFO_IN1,VCO_CONT
R1,PDH_PAR1);
  U2: aal1
port
map(ATM_CLK,PDH_CLK,CLK_MP,GRESET,stream2,atm_sdu,DATA_BUS,ADDRE
SS,RD,WR,ENR,FIFO_OUT2,SYNC_PULSE2,WRIT2,FIFO_IN2,VCO_CONTR2,PD
H_PAR2);
end arch_atm_pdh;

-- ==========
```

```
--        atm_pdh configuration
-- ============================================================
configuration atm_pdh_cfg of atm_pdh is
  for arch_atm_pdh
  end for;
end atm_pdh_cfg;
```

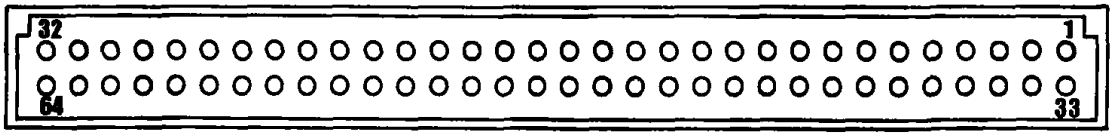
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
        NEW_CELL_STR1   :out STD_LOGIC;
        NEW_CELL_STR2   :out STD_LOGIC;
        NEW_CELL_MP     :out STD_LOGIC;
        ATMRX_OUT       :out STD_LOGIC_VECTOR (7 downto 0);
end component;

component sub1
 port(ATM_CLK           :in STD_LOGIC;
      PDH_CLK           :in STD_LOGIC;
      CLK_MP            :in STD_LOGIC;
      GRESET            :in STD_LOGIC;
      START             :in STD_LOGIC;
      IN_GE             :in STD_LOGIC;
      IN_PAYLOAD        :in STD_LOGIC_VECTOR (7 downto 0);
      DATA_IN           :inout STD_LOGIC_VECTOR (7 downto 0);
      ADDR_BUS          :in STD_LOGIC_VECTOR (2 downto 0);
      RD                :in STD_LOGIC;
      WR                :in STD_LOGIC;
      ENR               :in STD_LOGIC;
      FIFO_OUT          :in STD_LOGIC_VECTOR (7 downto 0);
      SYNC_PULSE        :out STD_LOGIC;
      WRIT              :out STD_LOGIC;
      FIFO_IN           :out STD_LOGIC_VECTOR (7 downto 0);
      VCO_CONTR         :out integer range -1 to 15;
      PDH_PAR           :out STD_LOGIC_VECTOR (7 downto 0));
end component;


begin
 [illegible]
port
 [illegible]ATM_CLK,CL_[illegible],ADDR_BUS,DATA_BUS,WR,RD,ATMRX_IN,RX_CE
LL,[illegible],PAR,[illegible],DAR,[illegible],[illegible],[illegible],[illegible],
 [illegible]
port
 [illegible]ATM_CLK,PDH_CLK,CLK_MP,GRESET,[illegible],DATA_BUS,ADDR
ESS_BUS,RD,WR,ENR,FIFO_OUT,[illegible],PULSE1,WRIT1,FIFO_IN1,VCO_CONT
R1,PDH_PAR1);
 [illegible]
port
 [illegible]ATM_CLK,PDH_CLK,CLK_MP,GRESET,[illegible],DATA_BUS,ADDRE
SS,RD,WR,ENR,FIFO_OUT2,[illegible],PULSE1,WRIT2,FIFO_IN2,VCO_CONTR2,PD
H_PAR2);
end arch_atm_pdh
```

Παράρτημα Β

Στο παράρτημα παρατίθονται ο προσδιορισμός των ακροδεκτών και το διάγραμμα χρονισμού (timining diagram) του κοινού διαύλου (back plane).

Για την επικοινωνία με τις μονάδες I/O επιλέχθηκε το βιομηχανικό standard euro connector 64 επαφών, όπως απεικονίζεται στο επόμενο σχήμα και στο οποίο χρησιμοποιούνται 2 επαφές ανά γραμμή για μεγαλύτερη αξιοπιστία.



Στην επόμενη σελίδα παρατίθενται πίνακας με την αντιστοιχία των επαφών (Pins).

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Αριθμός Pin	Όνομα σήματος
1 – 33	+5V DC
2 – 34	+12V DC
3 – 35	-12V DC
4 – 36	GND
5 – 37	PSC
6 – 38	D0
7 – 39	D1
8 – 40	D2
9 – 41	D3
10 - 42	D4
11 - 43	D5
12 - 44	D6
13 - 45	D7
14 - 46	$\overline{INT1}$
15 - 47	$\overline{OK}$
16 - 48	GND
17 - 49	$\overline{CS0}$
18 - 50	$\overline{CS1}$
19 - 51	$\overline{CS2}$
20 - 52	$\overline{CS3}$
21 - 53	$\overline{CS4}$
22 - 54	$\overline{CS5}$
23 - 55	A0
24 - 56	A1
25 - 57	A2
26 - 58	A3
27 - 59	$\overline{INT0}$
28 - 60	P1.2
29 - 61	RS
30 - 62	$\overline{RD}$
31 - 63	$\overline{WR}$
32 - 64	GND

Παρακάτω γίνεται η περιγραφή των σημάτων.

Τάσεις τροφοδοσίας

Χρησιμοποιούνται τρία επίπεδα τάσης: +5V DC, +12V DC, -12V DC.

PSC

Η γραμμή αυτή είναι η ανορθωμένη τάση του τροφοδοτικού πριν περάσει από τους σταθεροποιητές τάσης. Χρησιμοποιείται για την έγκαιρη ανίχνευση πτώσης τάσης.

D0 - D7, Data bits 0 έως 7

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Οι γραμμές αυτές σχηματίζουν τον δίαυλο δεδομένων (data bus), bit 0 έως 7, για τον ελεγκτή, τις μνήμες και τις μονάδες I/O. Το D0 είναι το λιγότερο σημαντικό bit (LSB) και το D7 ειναι το πλέον σημαντικό bit (MSB).

$\overline{INT1}$

Η γραμμή αυτή χρησιμοποιείται για να ειδοποιηθεί ασύγχρονα ο ελεγκτής οτι μια μονάδα I/O χρειάζεται εξυπηρέτηση. Συνδέεται με το αντίστοιχο INT1 του 8051.

$\overline{OK}$

Ενεργοποιείται στο λογικό 0. Η στάθμη του λογικού 1 στο σήμα αυτό δηλώνει την λανθασμένη λειτουργία της CPU. Χρησιμοποιείται από τις μονάδες ψηφιακών εξόδων, για την απενεργοποίηση τους σε περίπτωση μη σωστής λειτουργίας της CPU.

$\overline{CS0}$ - $\overline{CS5}$, Card Select από 0 έως 5

Χρησιμοποιούνται για την επιλογή 6 μονάδων I/O. Σε καθένα απο αυτά αντιστοιχούν 16 διευθύνσεις (*A0-A3*). Ενεργοποιούνται σε λογικό 0.

A0 -A3

Είναι γραμμές διεύθυνσης οι οποίες χρησιμοποιούνται σε συνδυασμο με τα card-select σηματα για την επιλογή 16 τοπικών διευθύνσεων σε μια μονάδα I/O. A0 είναι το λιγότερο σημαντικό bit.

$\overline{INT0}$

Η γραμμή αυτή χρησιμοποιείται για να ειδοποιηθεί ασύγχρονα ο ελεγκτής οτι μια μονάδα I/O χρειάζεται εξυπηρέτηση. Συνδέεται με το αντίστοιχο INT0 του 8051.

P1.2

Συνδέεται με το αντίστοιχο P1.2 του 8051

RS, Reset Out

Κατά την εκκίνηση του βιομηχανικού ελεγκτή δημιουργείται ένα reset. Επίσης σε περίπτωση λανθασμένης λειτουργίας της CPU, το κύκλωμα watch-dog δημιουργεί παλμούς reset. Η γραμμή reset out χρησιμοποιείται για να μεταδοθεί το reset στις κάρτες που χρησιμοποιούν προγραμματιζόμενα ολοκληρωμένα κυκλώματα.

$\overline{RD}$, σήμα ανάγνωσης

Είναι σήμα ελέγχου και χρησιμοποιείται για να μεταφέρει τα δεδομένα από μία διεύθυνση στο data bus. Συνδεεται με το σημα RD του 8051 (buffered).

$\overline{WR}$, σήμα εγγραφής

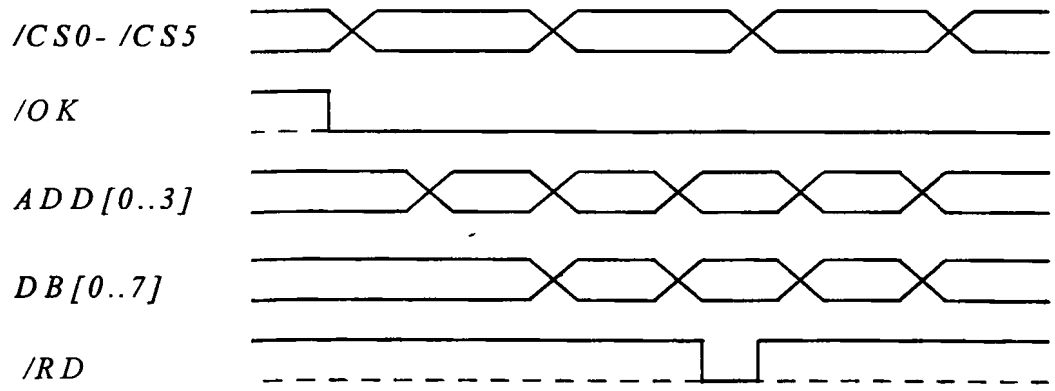
Είναι σήμα ελέγχου και χρησιμοποιείται για να μεταφέρει σε μια διεύθυνση δεδομένα από το κοινό δίαυλο. Συνδέεται με το σημα WR του 8051 (buffered).

Στο επόμενο διάγραμμα παρατίθενται ο χρονισμός των σημάτων του κοινού διάυλου (back-plane) τόσο σε κατάσταση ανάγνωσης (read mode) όσο και σε

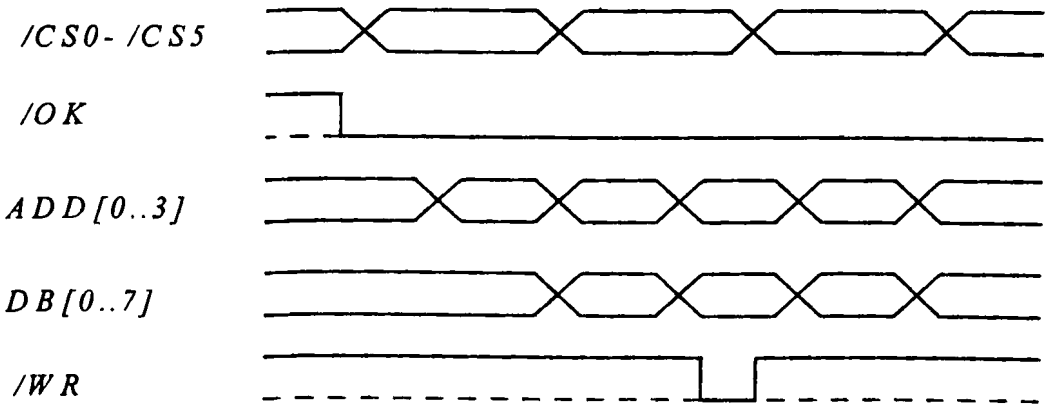
ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

κατάσταση εγγραφής (write mode). Στο χρονικό αυτό διάγραμμα βλέπουμε ότι πρώτα στέλνονται τα σήματα $\overline{CS0}$ - $\overline{CS5}$ για την επιλογή της κάρτας και το σήμα $\overline{OK}$ που δηλώνει την σωστή κατάσταση λειτουργίας ή όχι της κεντρικής μονάδας επεξεργασίας. Μετά την επαλήθευση των σημάτων αυτών από την κάρτα στέλνονται οι διευθύνσεις ADD[0..3] για την επιλογή διαδικασίας που θα υλοποιηθεί σύμφωνα με τους πίνακες 8.1 και 9.1 στην κάρτα αναλογικών εισόδων και κάρτα ψηφιακών εξόδων αντίστοιχα. Τέλος, τα δεδομένα στέλνονται στην περίπτωση εγγραφής ή διαβάζονται στην περίπτωση ανάγνωσης από την κεντρική μονάδα προς την επιλεγμένη κάρτα και αντίστροφα. Επιπρόσθετα, τα δεδομένα είναι ενεργά μόνο στο χρονικό διάστημα κατά το οποίο ή το σήμα $\overline{WR}$ ή το σήμα $\overline{RD}$ είναι σε λογικό μηδέν.

Timing Diagram of the Back Plane - Read Mode



Timing Diagram of the Back Plane - Write Mode



Η κάθετος (/) δηλώνει ενεργοποίηση σήματος στο λογικό μηδέν (active low).

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Παράρτημα Γ

Στο παράρτημα αυτό παρατίθεται το λογισμικό σε γλώσσα ASSEMBLY που χρησιμοποιήθηκε για την εξομοίωση και την παραγωγή σημάτων του κοινού διαύλου (back plane) για τις κάρτες αναλογικών εισόδων και ψηφιακών εξόδων.

➲ Παρακάτω παρατίθεται το κομμάτι του λογισμικού για την κάρτα αναλογικών εισόδων.

```
            ORG         0000H
*************************************************************************
AGAIN           CLR         ACC
            MOV         P2,#11000001B  ; SEND ADDRESS FOR STATUS WORD
            MOV         P0,#00         ; CLEAR PORT 0
            SETB        P2.3           ; A0=1 ==> READ STATUS WORD
            CLR         P2.7           ; CLEAR RD
            MOV         P1,#0255
            MOV         A,P1           ;STORE STATUS WORD DATA BUS TO A
            SETB        P2.7           ; SET RD
            MOV         P0,A           ; DISPLAY REG. A ON PORT 0
Τμήμα Α
*************************************************************************
            ACALL           DELAY              ; NOT REQUIRED
*************************************************************************
            MOV         P2,#11000001B  ; SEND ADDRESS FOR 1st CHAN. OF ADC
            CLR         P2.6           ; CLEAR WR
            NOP                        ; WAIT (NOT REQUIRED)
            SETB P2.6                  ; SET WR ==> START CONVERSION
            ACALL       DEL53          ; 53 usec DELAY FOR CONVERSION
            CLR         P2.3           ; A0=0 ==> READ ADC
            CLR         P2.7           ; CLEAR RD
            MOV         P1,#0255
            MOV         A,P1           ; STORE ADC DATA BUS TO A
            SETB        P2.7           ; SET RD
            MOV         P0,A           ; DISPLAY REG. A ON PORT 0
            ACALL       DELAY
            SJMP        AGAIN
            MOV         PCON,#01       ; MC IDLE
Τμήμα Β
*************************************************************************
```

```
;-----------------------------------------------------------------
;General purpose delay(the big one)
DELAY   PUSH    B
        PUSH    DPH
        PUSH    DPL
        MOV     B,#010
D3      MOV     DPH,#0255
D2      MOV     DPL,#0255
D1      DJNZ    DPL,D1
        DJNZ    DPH,D2
        DJNZ    B,D3
        POP     DPL
        POP     DPH
        POP     B
        RET
;-----------------------------------------------------------------
;delay aprox. 53 usec with instruction ACALL.
DEL53   PUSH    DPL
        MOV     DPL,#020
REP     DJNZ    DPL,REP
        POP     DPL
        RET
        END
```

➲ Το υπόλοιπο κομμάτι του λογισμικού αφορά την κάρτα ψηφιακών εξόδων.

```
        ORG     0000H
*************************************************************************
AGAIN   CLR     ACC
        MOV     P2,#11000001B     ; SEND ADDRESS FOR STATUS WORD
        MOV     P0,#00            ; CLEAR PORT 0
        CLR     P2.7              ; CLEAR RD
        MOV     P1,#0255
        MOV     A,P1              ; STORE STATUS DATA BUS TO A
        SETB    P2.7              ; SET RD
        MOV     P0,A              ; DISPLAY REG. A ON PORT 0
Τμήμα Α
*************************************************************************
        ACALL   DELAY             ; NOT REQUIRED
*************************************************************************
        MOV     P2,#11000001B     ; ADDRESS FOR DIGIT OUT
        MOV     P1,#0255      ; OUT DATA = 255: ALL OUTPUTS ENABLED
        CLR     P2.6              ; CLEAR WR
        NOP                       ; WAIT (NOT REQUIRED)
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
            SETB  P2.6                  ; SET WR
Τμήμα Β
*************************************************************************
            ACALL       DELAY           ; NOT REQUIRED

;-------------------------------------------------------------------
;General purpose delay(the big one)
DELAY       PUSH        B
            PUSH        DPH
            PUSH        DPL
            MOV         B,#010
D3          MOV         DPH,#0255
D2          MOV         DPL,#0255
D1          DJNZ        DPL,D1
            DJNZ        DPH,D2
            DJNZ        B,D3
            POP         DPL
            POP         DPH
            POP         B
            RET
            END
```

ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

```
;General purpose delay(the big one)
DELAY   PUSH    B
        PUSH    DPH
        PUSH    DPL
        MOV     B,#010
D3      MOV     DPH,#0255
D2      MOV     DPL,#0255
D1      DINZ    DPL,D1
        DINZ    DPH,D2
        DINZ    B,D3
        POP     DPL
        POP     DPH
        POP     B
        RET

;Delay [illegible] with instruction ACALL
DELAY   PUSH    DPL
        MOV     DPL,#0255
REP     DINZ    DPL,REP
        POP     DPL
        RET
        END
```

⇒ Το υπόλοιπο κομμάτι του λογισμικού ασορά της κάρτα ψηφιακών εξόδων.

```
        ORG     0000H
AGAIN   CLR     ACC
        MOV     P2,#11000000B   ; BIND ADDRESS FOR STATUS WORD
        MOV     P0,#00          ; CLEAR PORT 2
        CLR     P1.7            ; CLEAR [illegible]
        MOV     P1,#0255
        MOV     [illegible]     ; [illegible] DATA [illegible] TO A
        SETB    [illegible]     ; [illegible]
        MOV     [illegible]     ; [illegible] PORT 0
Τμήμα Α
        ACALL   DELAY           ; NOT [illegible]
        MOV     P2,#11000000B   ; ADDRESS FOR [illegible]
        MOV     P1,#0255        ; [illegible] DATA [illegible] ALL [illegible] ENABLED
        CLR     P1.7            ; CLEAR [illegible]
        NOP                     ; WAIT (NOT [illegible])
```