ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

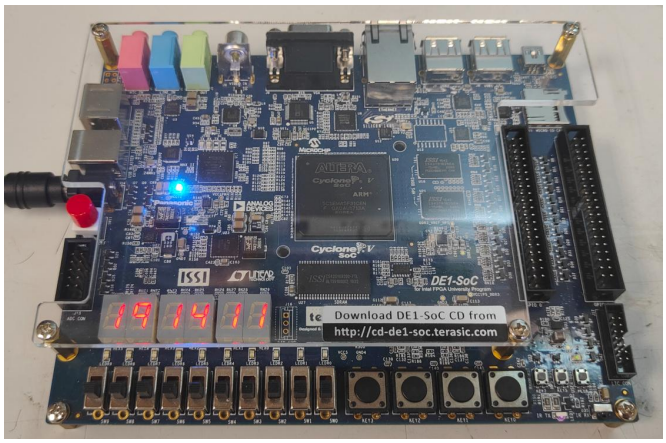
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση ψηφιακού ρολογιού σε FPGA



Του φοιτητή
Ανδρεάδη Ηλία
Αρ. Μητρώου: 52302Μ

Επιβλέπων
Παπακώστας Δημήτριος
Καθηγητής

Φεβρουάριος 2025

Τίτλος Δ.Ε. Υλοποίηση ψηφιακού ρολογιού σε FPGA
Κωδικός Δ.Ε. 24261
Ονοματεπώνυμο φοιτητή Ανδρεάδης Ηλίας
Ονοματεπώνυμο εισηγητή Παπακώστας Δημήτριος
Ημερομηνία ανάληψης Δ.Ε. 18/10/2024
Ημερομηνία περάτωσης Δ.Ε. 25/02/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Στους αγαπημένους μου.

Πρόλογος

Η επιλογή του θέματος της διπλωματικής εργασίας βασίστηκε στη σημασία των FPGA και της γλώσσας VHDL στη σύγχρονη σχεδίαση ηλεκτρονικών συστημάτων. Τα FPGA αποτελούν μια από τις πιο ευέλικτες και ισχυρές τεχνολογίες στον τομέα των ψηφιακών κυκλωμάτων, επιτρέποντας την ανάπτυξη πολύπλοκων εφαρμογών με υψηλή απόδοση και προσαρμοστικότητα. Η υλοποίηση ενός ψηφιακού ρολογιού σε FPGA συνδυάζει θεωρητικές γνώσεις και πρακτικές δεξιότητες, αποτελώντας μια ευκαιρία για εμβάθυνση στις γλώσσες περιγραφής υλικού και τις αρχές σχεδίασης ψηφιακών συστημάτων.

Η εργασία εστιάζει στην αρχιτεκτονική των FPGA, στη χρήση της VHDL και στη διαδικασία σχεδίασης, προσομοίωσης και υλοποίησης ενός πλήρους ψηφιακού συστήματος. Κατά την ανάπτυξη του έργου, εφαρμόστηκαν τεχνικές προγραμματισμού FPGA, διαχείρισης χρονισμού και προσομοίωσης ψηφιακών κυκλωμάτων. Η διπλωματική αυτή εργασία συμβάλλει στην κατανόηση των FPGA και της VHDL, παρέχοντας μια πρακτική εφαρμογή που μπορεί να αποτελέσει βάση για μελλοντικές αναπτύξεις.

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με τη σχεδίαση και υλοποίηση ενός ψηφιακού ρολογιού σε FPGA χρησιμοποιώντας τη γλώσσα περιγραφής υλικού VHDL. Το σύστημα περιλαμβάνει βασικές λειτουργίες όπως η εμφάνιση της ώρας και της ημερομηνίας, η ρύθμιση του χρόνου, η λειτουργία αφύπνισης, το χρονόμετρο και η αντίστροφη μέτρηση, καθώς και επιπλέον δυνατότητες όπως η επιλογή μεταξύ 12ωρης και 24ωρης μορφής ώρας, η νυχτερινή λειτουργία και η εξοικονόμηση ενέργειας.

Για την υλοποίηση του συστήματος, αναπτύχθηκαν και συνδυάστηκαν διάφορες λειτουργικές μονάδες, όπως κυκλώματα διαίρεσης συχνότητας, μετρητές BCD για τη διαχείριση του χρόνου και αποκωδικοποιητές ενδεικτών επτά τομέων για την οπτική αναπαράσταση των πληροφοριών.

Η υλοποίηση πραγματοποιήθηκε στο περιβάλλον Quartus II, όπου πραγματοποιήθηκαν προσομοιώσεις για την επαλήθευση της ορθής λειτουργίας όλων των λειτουργιών. Το σύστημα δοκιμάστηκε με επιτυχία, επιδεικνύοντας ακρίβεια στη μέτρηση του χρόνου και σταθερότητα στις μεταβάσεις μεταξύ των διαφόρων λειτουργιών. Η αντιστοίχιση των εισόδων και εξόδων με τα φυσικά pins του FPGA της αναπτυξιακής πλακέτας DE1-SoC εξασφάλισε τη σωστή λειτουργία του συστήματος σε πραγματικό χρόνο.

Τα αποτελέσματα της εργασίας επιβεβαίωσαν την ευελιξία και την αποδοτικότητα των FPGA στην υλοποίηση σύνθετων ψηφιακών συστημάτων, καθώς και τη σημασία της VHDL ως εργαλείου για την ακριβή περιγραφή και σχεδίαση της λογικής.

Implementation of a digital clock on FPGA

Ilias Andreadis

Abstract

This thesis focuses on the design and implementation of a digital clock on an FPGA using the VHDL hardware description language. The system includes essential functions such as displaying the time and date, time adjustment, alarm functionality, a stopwatch, and a countdown timer. Additionally, it offers extra features such as the selection between 12-hour and 24-hour time formats, night mode, and power-saving mode.

For the implementation of the system, various functional modules were developed and combined, including frequency division circuits, BCD counters for time management, seven-segment decoders for visual representation of information, and user control units.

The implementation was carried out in the Quartus II environment, where simulations were performed to verify the correct operation of all functions. The system was successfully tested, demonstrating accuracy in time measurement and stability in transitions between different functions. The mapping of inputs and outputs to the physical pins of the FPGA on the DE1-SoC development board ensured the proper operation of the system in real time.

The results of this work confirmed the flexibility and efficiency of FPGAs in implementing complex digital systems, as well as the significance of VHDL as a tool for precise logic description and design.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κύριο Δημήτρη Παπακώστα, για την εμπιστοσύνη, την ανάθεση της διπλωματικής και τη συνεργασία μας, καθώς και για την παραχώρηση της αναπτυξιακής πλακέτας στην οποία υλοποιήθηκε το πρακτικό κομμάτι της εργασίας. Επίσης, ευχαριστώ τον κύριο Γιάννη Ιντζέ για τη βοήθειά του και τις χρήσιμες συμβουλές που μου προσέφερε όποτε τις χρειάστηκα. Πάνω απ' όλα θέλω να ευχαριστήσω την οικογένειά μου και τους φίλους μου για τη συνεχή στήριξη, την κατανόηση και την ενθάρρυνση που μου προσέφεραν κατά τη διάρκεια αυτής της διαδικασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

BCD	Binary Coded Decimal
BRAM	Block Random-Access Memory
CAD	Computer-aided design
CB	Connection Boxes
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
DCM	Digital Clock Manager
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Arrays
FSM	Finite-State Machines
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
LSB	Least Significant Bit
LUT	Look-Up Table
MSB	Most Significant Bit
PAL	Programmable Array Logic
PLA	Programmable Logic Array
PLD	Programmable Logic Device
RAM	Random-Access Memory
ROM	Read-Only Memory
SB	Switch Box
RTL	Register Transfer Level Schematic
SRAM	Static Random-Access Memory
SPLD	Simple Programmable Logic Device
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VWF	Vector Waveform File

Εισαγωγή

Τα FPGA αποτελούν βασικό εργαλείο στη σύγχρονη σχεδίαση ηλεκτρονικών συστημάτων, καθώς προσφέρουν ευελιξία και δυνατότητα επαναπρογραμματισμού, καθιστώντας τα ιδανικά για την υλοποίηση πολύπλοκων λογικών συναρτήσεων και πρωτότυπων συστημάτων. Η ικανότητα τους να προσαρμόζονται στις ανάγκες της κάθε εφαρμογής τα καθιστά ιδανικά για ένα ευρύ φάσμα εφαρμογών. Στο πλαίσιο αυτό, η παρούσα Διπλωματική Εργασία επικεντρώνεται στη σχεδίαση και υλοποίηση ενός ψηφιακού ρολογιού χρησιμοποιώντας FPGA και τη γλώσσα περιγραφής υλικού VHDL, διερευνώντας παράλληλα τις σχεδιαστικές αρχές και τις τεχνικές προγραμματισμού που εμπλέκονται στη διαδικασία ανάπτυξης.

Ο βασικός σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός πλήρως λειτουργικού ψηφιακού ρολογιού σε FPGA, το οποίο να ενσωματώνει λειτουργίες όπως η εμφάνιση της ώρας και της ημερομηνίας, η δυνατότητα ρύθμισης του χρόνου, η λειτουργία αφύπνισης και το χρονόμετρο και η προσομοίωση και δοκιμή του συστήματος σε αναπτυξιακή πλακέτα. Επιπλέον, στόχοι είναι η μελέτη των FPGA, της αρχιτεκτονικής τους και των δυνατοτήτων που προσφέρουν, η διερεύνηση των δυνατοτήτων που προσφέρει η VHDL για τη σχεδίαση σύνθετων ψηφιακών συστημάτων.

Τα βασικά παραδοτέα της ΔΕ περιλαμβάνουν την ανάλυση και περιγραφή της αρχιτεκτονικής των FPGA και των τεχνολογιών διαμόρφωσης. Την υλοποίηση του ψηφιακού ρολογιού σε VHDL, με έμφαση στην ιεραρχική σχεδίαση και την επαναχρησιμοποίηση κώδικα. Την προσομοίωση και δοκιμή του συστήματος στο περιβάλλον Quartus II, για την επαλήθευση της ορθής λειτουργίας όλων των λειτουργιών. Την αντιστοίχιση των εισόδων και εξόδων του συστήματος με τα φυσικά pins του FPGA, για τη σωστή λειτουργία του συστήματος σε πραγματικό χρόνο.

Στο Κεφάλαιο 1ο γίνεται εισαγωγή στις Προγραμματιζόμενες Λογικές Διατάξεις, αναλύονται οι κύριες κατηγορίες τους καθώς και οι μέθοδοι προγραμματισμού τους.

Στο Κεφάλαιο 2ο παρουσιάζεται η αρχιτεκτονική των FPGA, συμπεριλαμβανομένων των λογικών μπλοκ, του δικτύου δρομολόγησης, των μπλοκ εισόδων/εξόδων, των μπλοκ ειδικού σκοπού, καθώς και οι τεχνολογίες διαμόρφωσης τους.

Στο Κεφάλαιο 3ο εξετάζονται οι βασικές έννοιες της VHDL, η δομή και η σύνταξή της, καθώς και η χρήση της για την ανάπτυξη ψηφιακών κυκλωμάτων.

Στο Κεφάλαιο 4ο παρουσιάζεται η σχεδίαση του ψηφιακού ρολογιού, τα επιμέρους υποσυστήματα, η υλοποίηση σε VHDL και η προσομοίωση του συστήματος.

Στο Κεφάλαιο 5ο παρουσιάζονται άλλες υλοποιήσεις ψηφιακών ρολογιών σε FPGA, αναδεικνύοντας τα πλεονεκτήματα και τις δυνατότητες της προτεινόμενης λύσης. Τέλος, παρουσιάζονται τα συμπεράσματα και οι προοπτικές για μελλοντικές βελτιώσεις.

Κεφάλαιο 1ο Προγραμματιζόμενες Λογικές Διατάξεις

Οι Προγραμματιζόμενες Λογικές Διατάξεις (Programmable Logic Device - PLD) είναι ολοκληρωμένα ψηφιακά κυκλώματα προγραμματιζόμενης λογικής που χρησιμοποιούνται στην σχεδίαση κυκλωμάτων για την υλοποίηση λογικών συναρτήσεων. Το πρώτο PLD κατασκευάστηκε στα μέσα της δεκαετίας του 1970 και αποτελούνταν από λίγες ασύνδετες μεταξύ τους λογικές πύλες και έναν πίνακα ασφαλειών [1]. Σε αντίθεση με τα τυπικά ολοκληρωμένα κυκλώματα, που έχουν σταθερή λειτουργία, τα PLD δεν έχουν προκαθορισμένη λειτουργία κατά την κατασκευή τους, και παρέχουν τη δυνατότητα προγραμματισμού από τον χρήστη ώστε να υλοποιήσουν ένα ευρύ φάσμα λειτουργιών ανάλογα με τις ανάγκες και τις ιδιαιτερότητες της κάθε εφαρμογής. Αυτή η ευελιξία καθιστά τις προγραμματιζόμενες λογικές διατάξεις ιδανικές για δοκιμές, πρωτότυπα, καθώς και για πρότζεκτ όπου η λογική λειτουργία μπορεί να αλλάξει ή να βελτιστοποιηθεί κατά τη διάρκεια της ανάπτυξης της εφαρμογής αλλά και του κύκλου ζωής του προϊόντος.

Τα PLD χωρίζονται σε τρείς κύριες κατηγορίες, ανάλογα με την αρχιτεκτονική, την πολυπλοκότητα και τον τρόπο προγραμματισμού τους.

- Απλές Προγραμματιζόμενες Λογικές Διατάξεις (Simple Programmable Logic Device - SPLD)
- Σύνθετες Προγραμματιζόμενες Λογικές Διατάξεις (Complex Programmable Logic Device - CPLD)
- Διατάξεις Πυλών Προγραμματιζόμενες στο Πεδίο (Field Programmable Gate Arrays - FPGA)

Παρακάτω θα αναφερθούμε στα χαρακτηριστικά των διατάξεων Simple PLD και Complex PLD, ενώ στα FPGA θα αναφερθούμε αναλυτικότερα στο Κεφάλαιο 2ο.

1.1 Απλές Προγραμματιζόμενες Λογικές Διατάξεις

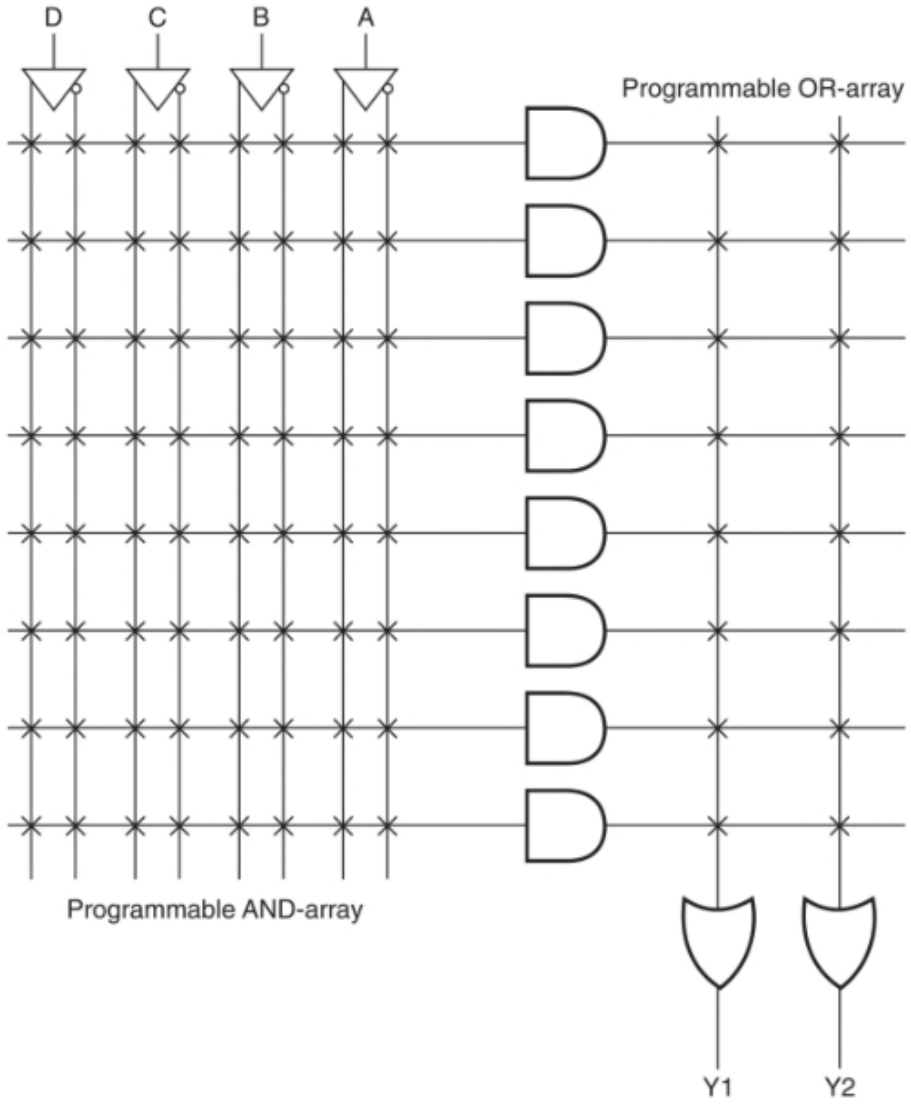
Οι Απλές Προγραμματιζόμενες Λογικές Διατάξεις (Simple Programmable Logic Device - SPLD) είναι, αρχιτεκτονικά, η βασική κατηγορία προγραμματιζόμενων λογικών διατάξεων, περιέχουν λογικές πύλες και προγραμματιζόμενους διακόπτες και χρησιμοποιούνται για την υλοποίηση απλών λογικών συναρτήσεων. Μέσω προγραμματισμού ρυθμίζονται οι διακόπτες ώστε να δημιουργηθούν οι κατάλληλες διασυνδέσεις μεταξύ των πυλών για την υλοποίηση του επιθυμητού, κάθε φορά, κυκλώματος. Οι δύο κύριες κατηγορίες SPLD είναι τα Programmable Logic Array (PLA) και τα Programmable Array Logic (PAL).

1.1.1 Προγραμματιζόμενοι Λογικοί Πίνακες

Η αρχιτεκτονική των Προγραμματιζόμενων Λογικών Πινακών (Programmable Logic Array - PLA) περιλαμβάνει ένα μπλοκ εισόδων, δύο πίνακες λογικών πυλών, έναν πίνακα πυλών AND, ο οποίος συνδέεται με έναν πίνακα πυλών OR και μπορούν να εκφράσουν κάθε λογική συνάρτηση ως άθροισμα γινομένων στο μπλοκ εξόδων της διάταξης [2].

Στο Σχήμα 1.1 απεικονίζεται η εσωτερική δομή μιας διάταξης PLA με τέσσερις εισόδους και τις ανάστροφες τιμές τους, οχτώ προγραμματιζόμενες πύλες AND και δύο προγραμματιζόμενες πύλες OR στην έξοδο της διάταξης, με (X) συμβολίζονται οι προγραμματιζόμενοι διακόπτες.

Η δυνατότητα προγραμματισμού τόσο των εισόδων των πυλών AND, όσο και των εισόδων των πυλών OR έχει σαν αποτέλεσμα τα PLA να είναι από τις πιο ευέλικτες προγραμματιζόμενες διατάξεις, ωστόσο καθιστά πιο δύσκολη την κατασκευή και τον προγραμματισμό τους, επίσης μειώνουν και την ταχύτητα απόκρισης του κυκλώματος [3].



Σχήμα 1.1: Εσωτερική δομή διάταξης PLA [3].

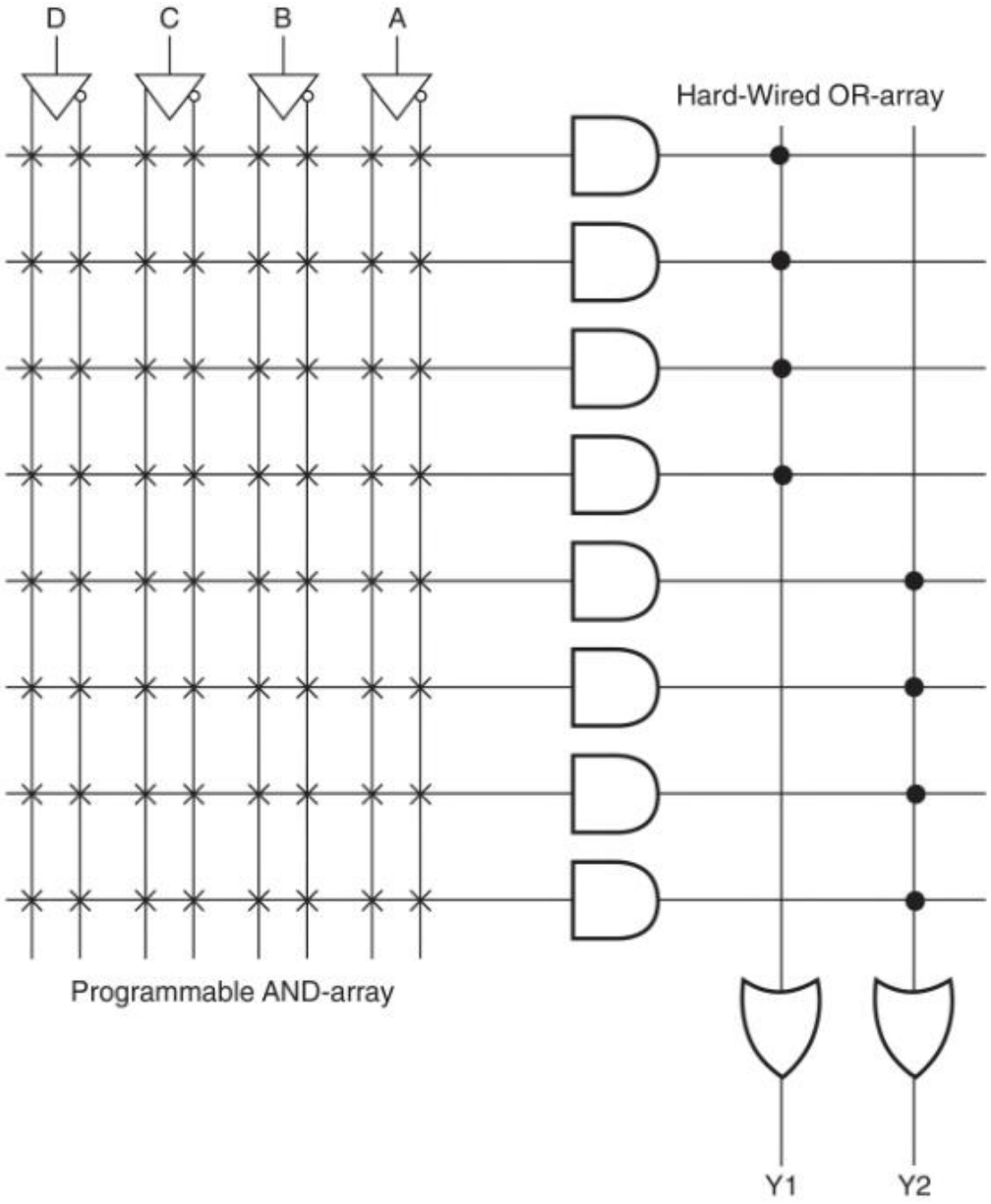
.

1.1.2 Προγραμματιζόμενοι Πίνακες Λογικής

Οι διατάξεις των Προγραμματιζόμενων Πινάκων Λογικής (Programmable Array Logic - PAL) διαθέτουν, όπως και οι διατάξεις PLA, δύο πίνακες λογικών πυλών, έναν πίνακα πυλών AND, μετά το μπλοκ εισόδων, και έναν πίνακα πυλών OR, οι έξοδοι των οποίων καταλήγουν στο μπλοκ εξόδων. Σε αντίθεση με τις διατάξεις PLA, στα PAL μόνο οι είσοδοι των πυλών AND είναι προγραμματιζόμενες, ενώ οι είσοδοι των πυλών OR είναι σταθερές. Οι διατάξεις PAL λόγω της απλούστερης κατασκευής τους, προσφέρουν καλύτερη απόδοση κυκλώματος και έχουν μικρότερο κόστος σε σχέση με τις διατάξεις PLA.

Στο Σχήμα 1.2 απεικονίζεται η εσωτερική δομή μιας διάταξης PAL με τέσσερις εισόδους και τις ανάστροφες τιμές τους, οχτώ προγραμματιζόμενες πύλες AND και δύο πύλες OR με σταθερές εισόδους, με (X) συμβολίζονται οι προγραμματιζόμενοι διακόπτες, ενώ με (•) συμβολίζονται οι σταθερές συνδέσεις των εξόδων των πυλών AND στις εισόδους των πυλών OR. Σε αντίθεση με τα

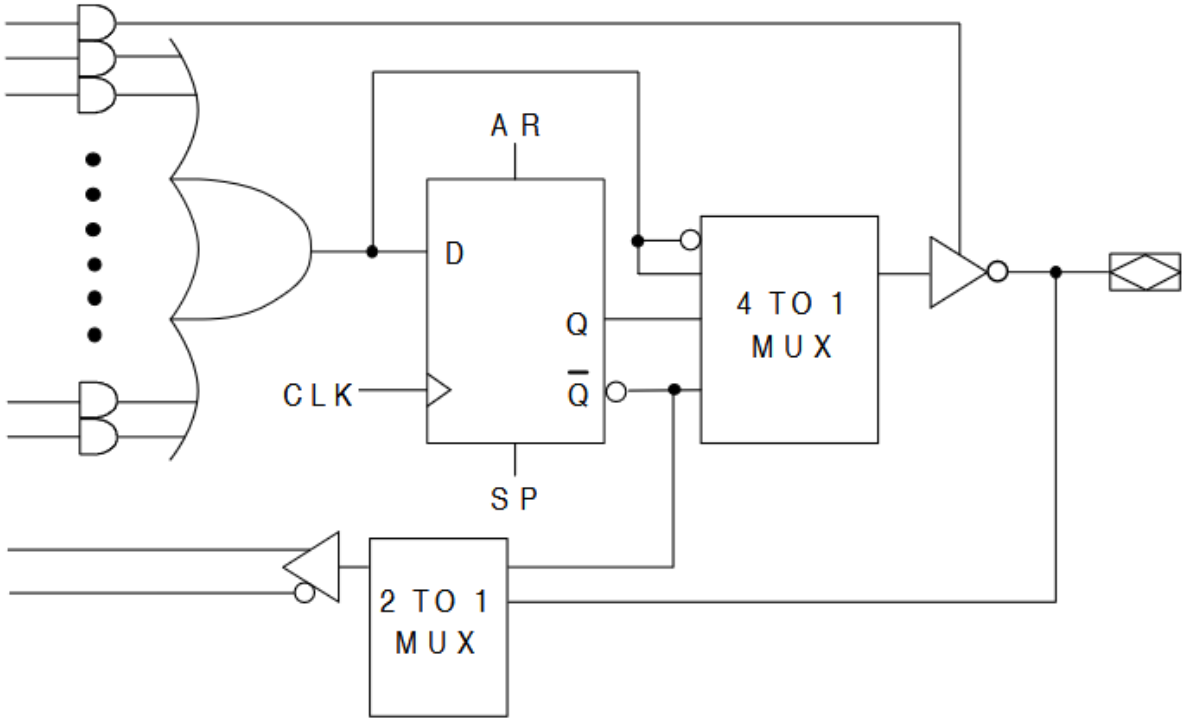
PLA όπου όλες οι έξοδοι των πυλών AND συνδέονται στις εισόδους των πυλών OR, στα PAL στις εισόδους κάθε πύλης OR συνδέονται μόνο τέσσερις από τις οχτώ έξοδοι των πυλών AND.



Σχήμα 1.2: Εσωτερική δομή διάταξης PAL [3].

Πολλές διατάξεις PAL διαθέτουν στην έξοδο, μετά τις πύλες OR, μια επιπλέον βαθμίδα, συνήθως αναφέρεται ως Output Logic Macrocell, η οποία προσθέτει επιπλέον ευελιξία στη διαμόρφωση του επιθυμητού κυκλώματος. Στο Σχήμα 1.3 απεικονίζεται διάγραμμα ενός τέτοιου κυκλώματος εξόδου της εταιρίας Lattice Semiconductor Corporation. Μπορεί να λειτουργήσει σε δύο καταστάσεις, registered mode και combinatorial I/O mode και η πολικότητα της εξόδου μπορεί να προγραμματιστεί ως ορθή ή ανάστροφη και στις δύο καταστάσεις. Σε registered mode η έξοδος του κυκλώματος οδηγείται από την έξοδο Q του flip-flop, ενώ η ανάστροφη έξοδος του flip-flop ανατροφοδοτείται, μαζί με τη συμπληρωματική τιμή της, στον πίνακα των πυλών AND. Σε combinatorial I/O mode η έξοδος του κυκλώματος οδηγείται απευθείας από την έξοδο της πύλης OR και η ανατροφοδότηση

προς τον πίνακα των πυλών AND γίνεται από την έξοδο του κυκλώματος [4]. Η ανατροφοδότηση προς τον πίνακα των πυλών AND επιτρέπει στον χρήστη την υλοποίηση πιο σύνθετων κυκλωμάτων με πολλαπλά επίπεδα.



Σχήμα 1.3: Διάγραμμα Output Logic Macrocell [4].

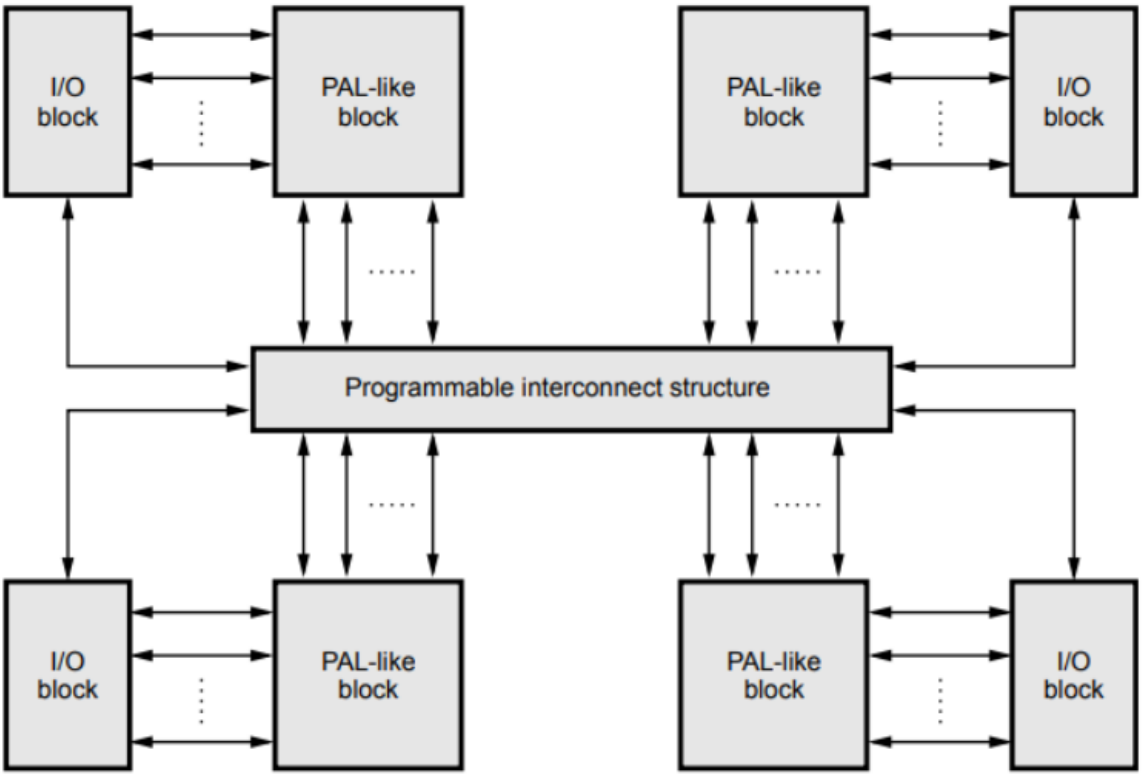
1.2 Σύνθετες Προγραμματιζόμενες Λογικές Διατάξεις

Τα CPLD είναι σύνθετες προγραμματιζόμενες λογικές διατάξεις οι οποίες δημιουργήθηκαν με σκοπό να προσφέρουν λύσεις στην υλοποίηση πιο σύνθετων και πολύπλοκων λογικών κυκλωμάτων, με μεγαλύτερες απαιτήσεις πόρων και εισόδων/εξόδων, συγκριτικά με τα PLA, PAL και άλλες ανάλογες διατάξεις,.

Τα CPLD αποτελούνται από πολλαπλά μπλοκ τύπου-PAL, τα οποία μπορούν να θεωρηθούν ως αυτόνομες μονάδες που υλοποιούν λογικές συναρτήσεις, και συνδέονται μεταξύ τους μέσω μιας προγραμματιζόμενης δομής καλωδίωσης σε ένα μοναδικό chip. Η εσωτερική προγραμματιζόμενη δομής καλωδίωσης επιτρέπει την ευέλικτη επικοινωνία των επιμέρους τμημάτων του CPLD. Στο Σχήμα 1.4 απεικονίζεται η εσωτερική δομή ενός CPLD, το οποίο περιλαμβάνει τέσσερα μπλοκ τύπου-PAL διασυνδεδεμένα μεταξύ τους μέσω της προγραμματιζόμενης καλωδίωσης, ενώ κάθε μπλοκ τύπου-PAL συνδέεται στο αντίστοιχο μπλοκ εισόδων εξόδων.

Ο αριθμός των μπλοκ τύπου-PAL μπορεί να ποικίλει από δύο μέχρι και πάνω από εκατό ανάλογα το chip και τον κατασκευαστή. Αυτό έχει σαν αποτέλεσμα, ενώ μια διάταξη PLD έχει πολυπλοκότητα μερικών εκατοντάδων λογικών πυλών, η αντίστοιχη πολυπλοκότητα των CPLD μπορεί να φτάσει κάποιες χιλιάδες λογικών πυλών.

Τα CPLD προσφέρουν προβλέψιμο χρόνο καθυστέρησης και υψηλή ταχύτητα απόκρισης, λόγω της προκαθορισμένης εσωτερικής δομής τους, και μικρή κατανάλωση ενέργειας, κάτι που τα κάνει ιδανικά για υλοποίηση εφαρμογών που απαιτείται υψηλό επίπεδο απόδοσης καθώς και για φορητές συσκευές που λειτουργούν με μπαταρία.



Σχήμα 1.4: Μπλοκ διάγραμμα CPLD [5].

1.3 Προγραμματισμός SPLD - CPLD

Ο προγραμματισμός των Simple PLD πραγματοποιείται μέσω εξειδικευμένων εργαλείων CAD τα οποία παρέχουν τη δυνατότητα σχεδίασης του λογικού κυκλώματος, προσομοίωσής του και δημιουργίας ενός αρχείου προγραμματισμού, το οποίο καθορίζει την κατάσταση κάθε προγραμματιζόμενου διακόπτη στο SPLD ώστε να υλοποιηθεί το σχεδιασμένο κύκλωμα.

Το SPLD τοποθετείται σε ειδική εξωτερική μονάδα προγραμματισμού, μέσω της οποίας το αρχείο προγραμματισμού μεταφέρεται από τον υπολογιστή στο chip και γίνεται η ρύθμιση των προγραμματιζόμενων διακοπτών. Μετά την ολοκλήρωση της διαδικασίας, η μονάδα διαβάζει τις ρυθμίσεις του SPLD για να επαληθεύσει ότι ο προγραμματισμός έχει πραγματοποιηθεί σωστά. Συνήθως τα SPLD τοποθετούνται στα τυπωμένα κυκλώματα σε βάσεις, ώστε να είναι δυνατή η εύκολη αφαίρεσή τους και η τοποθέτηση σε εξωτερική μονάδα για τη διαδικασία του προγραμματισμού.

Η μέθοδος προγραμματισμού σε εξωτερική μονάδα δεν είναι πρακτική για τα CPLD, τα οποία πολλές φορές διαθέτουν πάνω από διακόσιες επαφές κι έτσι αυξάνεται ο κίνδυνος να προκληθεί ζημιά σε κάποια από αυτές κατά τη διαδικασία της τοποθέτησης ή αφαίρεσης του chip. Επιπλέον, οι βάσεις για αφαίρεση και επανατοποθέτηση στην πλακέτα, πολλές φορές είναι πιο ακριβές για ολοκληρωμένα κυκλώματα σαν τα CPLD. Στις περισσότερες περιπτώσεις στα CPLD υποστηρίζεται η μέθοδος in-system programming όπου ο προγραμματισμός εκτελείται ενώ το ολοκληρωμένο κύκλωμα είναι συνδεδεμένο στο τυπωμένο κύκλωμα. Το αρχείο προγραμματισμού, που παράγεται από το CAD, μεταφέρεται απευθείας από τον υπολογιστή στην πλακέτα χρησιμοποιώντας το πρωτόκολλο JTAG.

1.4 Επίλογος

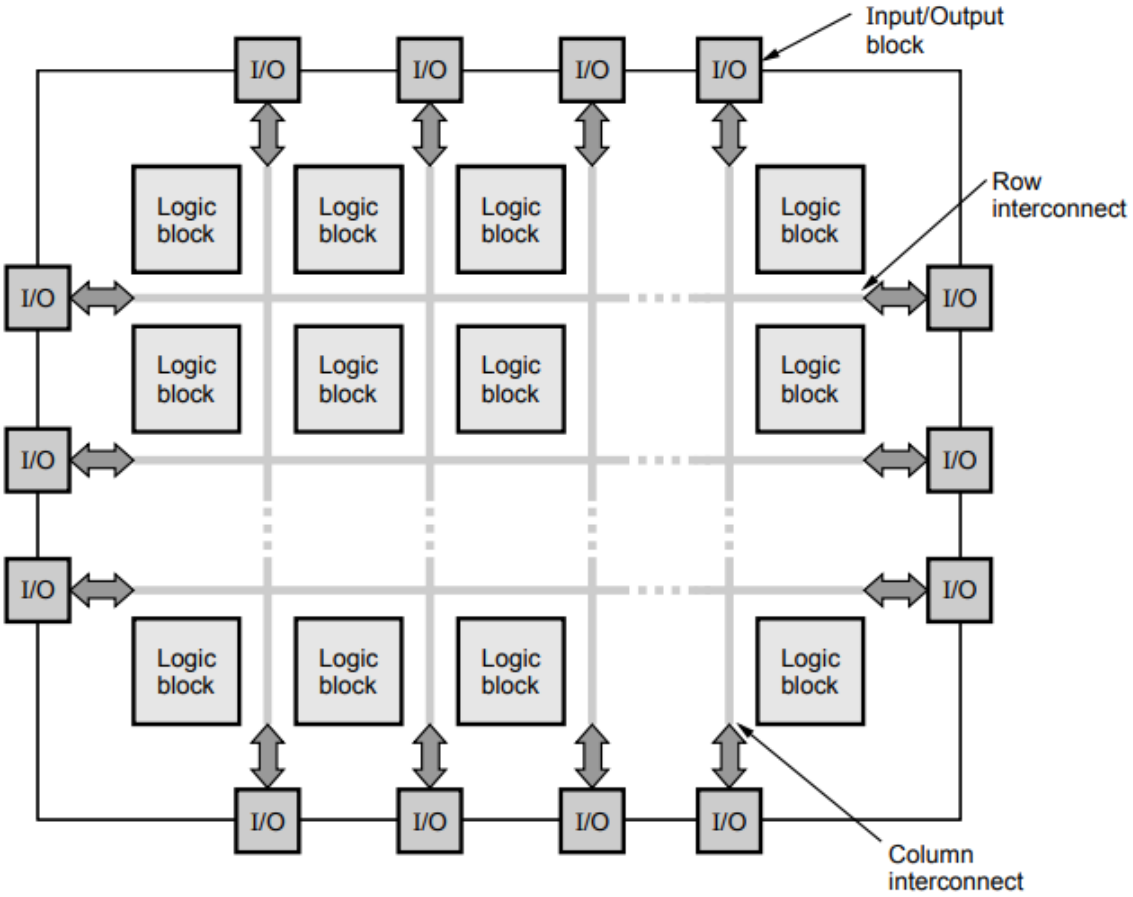
Στο παρόν κεφάλαιο αναλύθηκαν οι Προγραμματιζόμενες Λογικές Διατάξεις τα χαρακτηριστικά τους και οι βασικές κατηγορίες στις οποίες διακρίνονται. Αρχικά, παρουσιάστηκε η έννοια των PLD ως προγραμματιζόμενα ολοκληρωμένα κυκλώματα, τα οποία προσφέρουν ευελιξία στον σχεδιασμό ψηφιακών λογικών συστημάτων. Τα PLD επιτρέπουν τη διαμόρφωση της λειτουργίας τους μετά την κατασκευή τους, σε αντίθεση με τα ειδικά ολοκληρωμένα κυκλώματα, καθιστώντας τα κατάλληλα για πρωτότυπα και προσαρμοζόμενες εφαρμογές.

Εξετάστηκαν οι βασικές κατηγορίες των PLD. Τα Simple PLD, που περιλαμβάνουν τα Programmable Logic Arrays και τα Programmable Array Logic. Οι διατάξεις PLA είναι πιο ευέλικτες, αλλά πιο πολύπλοκες, ενώ οι διατάξεις PAL προσφέρουν καλύτερη απόδοση λόγω της απλούστερης αρχιτεκτονικής τους. Τα Complex PLD, τα οποία αποτελούνται από πολλαπλά μπλοκ τύπου-PAL, συνδεδεμένα μέσω μιας προγραμματιζόμενης δομής καλωδίωσης. Τα CPLD διαθέτουν μεγαλύτερη πολυπλοκότητα και είναι κατάλληλα για υλοποίηση μεγαλύτερων κυκλωμάτων με μεγαλύτερη ταχύτητα και μικρότερη κατανάλωση ενέργειας σε σχέση με τα SPLD.

Τέλος, έγινε αναφορά στη διαδικασία προγραμματισμού των SPLD και CPLD, η οποία επιτυγχάνεται μέσω εξειδικευμένων εργαλείων CAD. Ο προγραμματισμός των SPLD συνήθως πραγματοποιείται σε εξωτερική μονάδα προγραμματισμού, ενώ στα CPLD χρησιμοποιείται η μέθοδος In-System Programming μέσω του πρωτοκόλλου JTAG, προσφέροντας μεγαλύτερη ευκολία και αξιοπιστία στη διαδικασία προγραμματισμού.

Κεφάλαιο 2ο Διατάξεις Πυλών Προγραμματιζόμενες στο Πεδίο

Τα FPGA (Field-Programmable Gate Arrays) είναι επαναπρογραμματιζόμενα ολοκληρωμένα κυκλώματα τα οποία επιτρέπουν την υλοποίηση σχετικά μεγάλων λογικών συναρτήσεων ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Τα FPGA προσφέρουν ευελιξία, καθώς ο σχεδιαστής μπορεί να ορίσει τη λειτουργία του κυκλώματος μέσω προγραμματισμού. Το πρώτο FPGA κατασκευάστηκε από την Xilinx το 1984 και τις επόμενες δεκαετίες έγιναν σημαντικές βελτιώσεις καθώς αυξήθηκε σημαντικά η χωρητικότητα και η ταχύτητα απόκρισης και μειώθηκε η κατανάλωση ενέργειας [6]. Σε αντίθεση με τα SPLD και CPLD, τα οποία χρησιμοποιούν κυρίως λογικές πύλες AND και OR για την υλοποίηση των συναρτήσεων, τα FPGA χρησιμοποιούν λογικά μπλοκ. Στο Σχήμα 2.1 απεικονίζεται η γενική αρχιτεκτονική δομή του FPGA, που περιλαμβάνει τα λογικά μπλοκ για την υλοποίηση των συναρτήσεων, τα μπλοκ εισόδων/εξόδων για την επικοινωνία με το περιβάλλον εκτός του ολοκληρωμένου κυκλώματος καθώς και το δίκτυο δρομολόγησης. Τα λογικά μπλοκ και τα μπλοκ εισόδων/εξόδων είναι τοποθετημένα σε δισδιάστατη διάταξη, ενώ τα δίκτυο δρομολόγησης, το οποίο περιλαμβάνει αγωγούς και προγραμματιζόμενους διακόπτες, διαμορφώνεται σε οριζόντια και κάθετα κανάλια μεταξύ των λογικών μπλοκ επιτρέποντας την εύκολη και ευέλικτη διασύνδεση τους. Οι προγραμματιζόμενοι διακόπτες έχουν τη δυνατότητα να διασυνδέσουν τα λογικά μπλοκ και τα μπλοκ εισόδων/εξόδων με το δίκτυο δρομολόγησης, αλλά και τα κάθετα με τα οριζόντια κανάλια του δικτύου μεταξύ τους [2].



Σχήμα 2.1: Αρχιτεκτονική δομή FPGA [5].

Στα FPGA όλοι οι πόροι, τα μπλοκ εισόδων/εξόδων, τα λογικά μπλοκ και το δίκτυο δρομολόγησης, είναι πλήρως προγραμματιζόμενοι. Αυτό παρέχει στον χρήστη απόλυτη ελευθερία και ευελιξία στην σύνθεση κώδικα, επιτρέποντάς του να σχεδιάσει και να υλοποιήσει οποιαδήποτε εφαρμογή επιθυμεί καθώς και να διαμορφώσει τους πόρους σύμφωνα με τις ανάγκες του.

Μετά τη σύνθεση κώδικα από τον χρήστη, που συνήθως σε μεγάλες εφαρμογές γράφεται σε γλώσσα περιγραφής υλικού, όπως η Verilog ή η VHDL, το λογισμικό ανάπτυξης του κάθε κατασκευαστή FPGA (όπως το Vivado της Xilinx ή το Quartus II της Intel) αναλαμβάνει τη μετατροπή της περιγραφής του κυκλώματος σε φυσική υλοποίηση. Η χαρτογράφηση των λογικών συναρτήσεων στα διαθέσιμα λογικά μπλοκ, η τοποθέτηση των μονάδων στο ολοκληρωμένο κύκλωμα και η δρομολόγηση των συνδέσεων αποτελούν τα βασικά στάδια αυτής της διαδικασίας. Σε αυτό το κεφάλαιο, θα εστιάσουμε στην αρχιτεκτονική των FPGA και στις βασικές αρχές λειτουργίας τους. Θα εξετάσουμε πώς τα λογικά μπλοκ, τα μπλοκ εισόδων/εξόδων και το δίκτυο δρομολόγησης συνεργάζονται για να υλοποιήσουν πολύπλοκες λογικές συναρτήσεις. Ενώ σε επόμενο κεφάλαιο θα αναφερθούμε στις γλώσσες περιγραφής υλικού και στις διαδικασίες σύνθεσης, προσομοίωσης και υλοποίησης που υποστηρίζονται από το λογισμικό των κατασκευαστών.

2.1 Τεχνολογίες Διαμόρφωσης

Τα FPGA διακρίνονται σε τρεις κύριες κατηγορίες ανάλογα με την τεχνολογία που χρησιμοποιούν για την αποθήκευση της πληροφορίας διαμόρφωσης SRAM-based, Antifuse-based και Flash-based. Κάθε κατηγορία έχει τα δικά της χαρακτηριστικά, πλεονεκτήματα και μειονεκτήματα, καθιστώντας την κατάλληλη για διαφορετικές εφαρμογές.

Τα SRAM-based FPGA αποτελούν την πιο διαδεδομένη κατηγορία FPGA λόγω της ευελιξίας και της δυνατότητας επαναπρογραμματισμού τους. Αυτή η τεχνολογία βασίζεται στη μνήμη SRAM (Static Random-Access Memory), η οποία επιτρέπει τον επαναπρογραμματισμό της συσκευής απεριόριστες φορές, καθιστώντας την ιδανική για εφαρμογές που απαιτούν συχνές αλλαγές ή αναβαθμίσεις. Ωστόσο, τα SRAM-based FPGA έχουν ορισμένα μειονεκτήματα, όπως η υψηλή κατανάλωση ενέργειας σε σύγκριση με άλλες τεχνολογίες, λόγω του μεγάλου αριθμού τρανζίστορ που απαιτούνται, καθώς και η πτητική (volatile) φύση της μνήμης, που συνεπάγεται την απώλεια της πληροφορίας διαμόρφωσης όταν η τροφοδοσία διακοπεί. Για τον λόγο αυτό, απαιτείται η χρήση εξωτερικής μη πτητικής μνήμης για την αποθήκευση της διαμόρφωσης, η οποία πρέπει να φορτώνεται εκ νέου στο FPGA κάθε φορά που η συσκευή ενεργοποιείται. Αυτή η ανάγκη κάνει πιο περίπλοκο τον σχεδιασμό του συστήματος, αλλά εξασφαλίζει την ευελιξία και την επαναχρησιμοποίηση της συσκευής.

Τα Flash-based FPGA διατηρούν την πληροφορία διαμόρφωσης (non-volatile) ακόμα και μετά την διακοπή της τροφοδοσίας. Αυτή η ιδιότητα εξαλείφει την ανάγκη για εξωτερική μνήμη, όπως συμβαίνει στα SRAM-based FPGA. Επιπλέον, τα Flash-based FPGA μπορούν να επαναπρογραμματιστούν, αν και όχι απεριόριστες φορές, καθώς η μνήμη Flash έχει περιορισμένο αριθμό κύκλων εγγραφής/διαγραφής. Ένα σημαντικό πλεονέκτημά τους είναι η μειωμένη κατανάλωση ενέργειας σε σύγκριση με τα SRAM-based FPGA. Η διαδικασία προγραμματισμού πραγματοποιείται εκτός κυκλώματος με χρήση εξωτερικής συσκευής προγραμματισμού, ενώ σε κάποιες συσκευές υπάρχει η δυνατότητα προγραμματισμού και εντός κυκλώματος (in-system programming), με την ταχύτητα εγγραφής, ωστόσο, της μνήμης Flash να είναι σχετικά αργή σε σύγκριση με την SRAM.

Τα Antifuse-based FPGA ανήκουν στην κατηγορία των μη πτητικών προγραμματιζόμενων συστοιχιών (non-volatile), καθώς διατηρούν μόνιμα τη διαμόρφωσή τους. Η τεχνολογία antifuse

επιτρέπει μία και μοναδική διαμόρφωση (one-time programmable), καθώς η διαδικασία διαμόρφωσης βασίζεται σε μια μη αναστρέψιμη φυσική αλλαγή στη δομή του υλικού. Αυτή η ιδιότητα τα καθιστά ιδανικά για εφαρμογές που απαιτούν υψηλή αξιοπιστία και σταθερότητα. Επιπλέον, τα Antifuse-based FPGA καταλαμβάνουν λιγότερο χώρο σε σύγκριση με τα FPGA που βασίζονται σε SRAM και flash, λόγω της απλότητας της δομής τους, και διακρίνονται για την ανθεκτικότητά τους στην ακτινοβολία, κάτι που τα κάνει κατάλληλα για κρίσιμες εφαρμογές σε τομείς όπως η αεροδιαστημική, η άμυνα και οι βιομηχανικές εφαρμογές υψηλής αξιοπιστίας [7].

2.2 Πίνακες Αναζήτησης Look-Up Table

Τα Look-Up Table (LUT), τα οποία βρίσκονται στα λογικά μπλοκ, είναι τα βασικά στοιχεία, συνδυαστικής λογικής, μέσω των οποίων υλοποιούνται οι λογικές συναρτήσεις στα FPGA. Είναι ο κύριος μηχανισμός μέσω του οποίου εκτελούνται λογικές πράξεις, όπως AND, OR, NOT κ.ά. Τα LUT είναι πίνακες αναζήτησης που αποθηκεύουν το αποτέλεσμα μιας λογικής συνάρτησης για κάθε πιθανό συνδυασμό εισόδων. Ουσιαστικά λειτουργεί ως πίνακας αληθείας ο οποίος προγραμματίζεται κατά τη διαδικασία σχεδίασης του FPGA.

Ένα LUT με n εισόδους μπορεί να αποθηκεύσει 2^n τιμές εξόδου και να υλοποιήσει οποιαδήποτε λογική συνάρτηση με n ή λιγότερες μεταβλητές [8]. Οι είσοδοι του LUT χρησιμοποιούνται ως διευθύνσεις για την επιλογή της αντίστοιχης τιμής εξόδου από τον πίνακα. Για παράδειγμα, θέλουμε να υλοποιήσουμε σε ένα LUT τριών εισόδων (LUT3) τη λογική συνάρτησης της σχέσης (3.1):

$$f = ab + c' \tag{3.1}$$

Ο πίνακας αληθείας που προκύπτει είναι ο Πίνακας 2.1.

Πίνακας 2.1: Πίνακας αληθείας σχέσης (3.1).

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Στη μνήμη του LUT θα αποθηκευτούν οι οκτώ δυνατές τιμές εξόδου και οι μεταβλητές των εισόδων *a, b, c* θα χρησιμοποιηθούν ως διευθύνσεις προσπέλασης της σωστής τιμής εξόδου. Κάθε δυνατός συνδυασμός των εισόδων αντιστοιχεί σε μια συγκεκριμένη διεύθυνση στη μνήμη του LUT, όπου είναι αποθηκευμένη η αντίστοιχη τιμή της λογικής συνάρτησης. Ουσιαστικά το LUT λειτουργεί ως αποθηκευτικός χώρος μνήμης μεγέθους 8x1 bits. Κάθε γραμμή του πίνακα αντιστοιχεί σε μια συγκεκριμένη τιμή εισόδου (*a, b, c*) και στη σχετική τιμή εξόδου f, όπως φαίνεται στον Πίνακας 2.1. Όταν το FPGA εκτελεί τη συνάρτηση, οι είσοδοι ακολουθούν τη διαδικασία μετατροπής τους σε διευθύνσεις μνήμης, όπου το LUT ανακαλεί άμεσα την αποθηκευμένη τιμή εξόδου. Στο συγκεκριμένο παράδειγμα, για τιμές εισόδου *a*=1, *b*=0, *c*=1, σχηματίζεται η διεύθυνση 101, η οποία, σύμφωνα με τον πίνακα αληθείας, αντιστοιχεί στην αποθηκευμένη τιμή εξόδου *f*=0.

Το LUT δεν περιορίζεται μόνο στην υλοποίηση απλών λογικών συναρτήσεων, αλλά μπορεί να χρησιμοποιηθεί και για πιο σύνθετες λογικές πράξεις και αριθμητικές λειτουργίες. Συνδυάζοντας πολλαπλά LUT μέσα στα λογικά μπλοκ ενός FPGA, είναι δυνατή η υλοποίηση πολυεπίπεδων λογικών συναρτήσεων και πολύπλοκων αλγορίθμων.

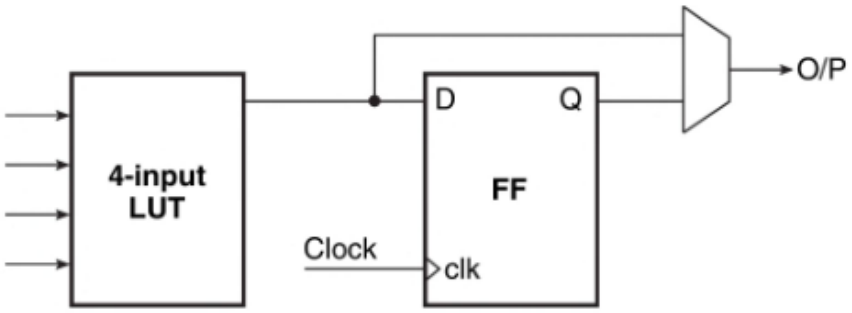
Ένα βασικό πλεονέκτημα των LUT είναι ότι ο χρόνος καθυστέρησης παραμένει σταθερός, ανεξάρτητα από την πολυπλοκότητα της λογικής συνάρτησης που υλοποιείται μέσα σε αυτά. Εφόσον μια λογική συνάρτηση χωράει σε ένα μόνο LUT, η καθυστέρηση διάδοσης δεν αλλάζει. Αυτή η ιδιότητα προσφέρει προβλεψιμότητα στην απόδοση και είναι ένα από τα βασικά πλεονεκτήματα της χρήσης LUT στα FPGA σε σύγκριση με άλλα ψηφιακά κυκλώματα όπου η καθυστέρηση αυξάνεται όσο μεγαλώνει η πολυπλοκότητα της συνάρτησης [9].

Παρόλο που το LUT αποτελεί το βασικό στοιχείο υλοποίησης λογικών συναρτήσεων στα εμπορικά FPGA, υπάρχει διαρκής έρευνα και διάλογος σχετικά με το ιδανικό του μέγεθος [7]. Οι μεγαλύτεροι πίνακες αναζήτησης επιτρέπουν την εκτέλεση πιο σύνθετων λογικών συναρτήσεων ανά λογικό μπλοκ, μειώνοντας έτσι τον χρόνο καθυστέρησης μεταξύ των μπλοκ. Ωστόσο, όσο αυξάνεται το μέγεθος των LUT, αυξάνεται και ο χρόνος καθυστέρησής τους, καθώς απαιτούν περισσότερο χρόνο για την ανάκληση της αποθηκευμένης τιμής, λόγω μεγαλύτερης εσωτερικής πολυπλοκότητας. Επιπλέον, αν μια λογική συνάρτηση δεν αξιοποιεί πλήρως όλες τις εισόδους και τη λειτουργικότητα ενός μεγάλου LUT, τότε σπαταλούνται πολύτιμοι πόροι του FPGA, οδηγώντας σε χαμηλότερη αποδοτικότητα και αυξημένη κατανάλωση ισχύος. Τα μικρότερα LUT παρουσιάζουν μικρότερο χρόνο καθυστέρησης, καθώς η διαδικασία ανάκλησης των τιμών από τη μνήμη είναι ταχύτερη. Επίσης, η χρήση μικρότερων LUT μπορεί να οδηγήσει σε καλύτερη αποδοτικότητα όσον αφορά την κατανάλωση ισχύος και τη χρήση πόρων. Ωστόσο, για να υλοποιηθούν πιο σύνθετες λογικές συναρτήσεις, απαιτείται η σύνθεση πολλών μικρότερων LUT, κάτι που ενδέχεται να οδηγήσει σε μεγαλύτερη καθυστέρηση λόγω αυξημένης πολυπλοκότητας των διασυνδέσεων και κατ’ επέκταση του κυκλώματος.

2.3 Βασικά στοιχεία αποθήκευσης Flip-Flop

Τα flip-flop είναι τα βασικά στοιχεία αποθήκευσης στα λογικά μπλοκ FPGA και χρησιμοποιούνται για την καταχώρηση και συγχρονισμό δεδομένων. Σε αντίθεση με τα LUT που λειτουργούν με συνδυαστική λογική, τα flip-flop προσφέρουν ακολουθιακή λογική στο κύκλωμα. Στα κυκλώματα συνδυαστικής λογικής η τιμή εξόδου εξαρτάται αποκλειστικά από τις τιμές των μεταβλητών εισόδου σε πραγματικό χρόνο, χωρίς να αποθηκεύονται προηγούμενες καταστάσεις. Αντίθετα, στα κυκλώματα ακολουθιακής λογικής η τιμή εξόδου εξαρτάται τόσο από τις τρέχουσες τιμές των εισόδων, όσο και από προηγούμενες καταστάσεις των τιμών εισόδου και εξόδου του κυκλώματος. Για τη λειτουργία των κυκλωμάτων ακολουθιακής λογικής είναι απαραίτητος ο συγχρονισμός τους με το σήμα ενός ρολογιού, καθώς η έξοδος του flip-flop αλλάζει κατάσταση μόνο όταν αλλάζει κατάσταση το ρολόι που το συγχρονίζει.

Στα λογικά μπλοκ των FPGA τα flip-flop χρησιμοποιούνται για την αποθήκευση των τιμών των εξόδων των LUT. Όπως φαίνεται στο Σχήμα 2.2 οι έξοδοι των flip-flop και των LUT καταλήγουν σε ένα πολυπλέκτη για την επιλογή της επιθυμητής, κάθε φορά, εξόδου. Τα flip-flop, τα look-up table και οι πολυπλέκτες είναι τα βασικά ηλεκτρονικά στοιχεία των λογικών μπλοκ.



Σχήμα 2.2:Λογικό μπλοκ ενός τυπικού FPGA [3].

2.4 Μπλοκ ειδικού σκοπού

Τα σύγχρονα FPGA, εκτός από τα λογικά μπλοκ στα οποία αναφερθήκαμε είδη, περιλαμβάνουν και μπλοκ ειδικού σκοπού που είναι ενσωματωμένα στη δομή τους. Αυτά τα μπλοκ έχουν σχεδιαστεί ώστε να εκτελούν εξειδικευμένες λειτουργίες με μεγαλύτερη αποδοτικότητα και χαμηλότερη κατανάλωση ενέργειας. Τα μπλοκ ειδικού σκοπού, όπως οι ενσωματωμένοι πολλαπλασιαστές, οι μνήμες και οι επεξεργαστές, προσφέρουν σημαντικά πλεονεκτήματα σε σχέση με τις παραδοσιακές λύσεις, καθώς παρέχουν έτοιμες λειτουργίες που δεν απαιτούν τη δημιουργία προσαρμοσμένων σχεδίων.

2.4.1 Μπλοκ RAM

Πολλές εφαρμογές απαιτούν τη χρήση ενσωματωμένης μνήμης στο FPGA. Παρόλο που μπορούν να χρησιμοποιηθούν τα λογικά μπλοκ για την υλοποίηση μνήμης RAM μεταβλητού μεγέθους, καθώς αυξάνεται η απαιτούμενη μνήμη καταναλώνονται και οι διαθέσιμοι πόροι. Για αυτόν τον λόγο, τα σύγχρονα FPGA περιλαμβάνουν ειδικά μπλοκ μνήμης, όπως τα μπλοκ RAM (BRAM), τα οποία προσφέρουν σταθερή και αποδοτική αποθήκευση δεδομένων. Αυτά τα μπλοκ μνήμης μπορούν να χρησιμοποιηθούν για την υλοποίηση RAM, ROM ή buffer, εξασφαλίζοντας βελτιωμένη απόδοση και εξοικονόμηση λογικών πόρων. Επιπλέον, τα BRAM μπορούν να συνδυαστούν για τη δημιουργία μεγαλύτερων μονάδων μνήμης, ενώ επιτρέπουν και ανεξάρτητες λειτουργίες ανάγνωσης και εγγραφής δεδομένων [9].

2.4.2 Μπλοκ Ψηφιακής Επεξεργασίας Σήματος DSP

Πολλές συσκευές FPGA ενσωματώνουν ειδικά μπλοκ ψηφιακής επεξεργασίας σήματος (Digital Signal Processing Blocks - DSP), τα οποία έχουν σχεδιαστεί για την αποδοτική υλοποίηση εφαρμογών που απαιτούν επεξεργασία σήματος ή βασικές αριθμητικές πράξεις, όπως πολλαπλασιασμός, πρόσθεση και αφαίρεση. Αν και αυτές οι λειτουργίες μπορούν να υλοποιηθούν χρησιμοποιώντας τα προγραμματιζόμενα λογικά μπλοκ του FPGA η ενσωμάτωση των DSP μπλοκ απευθείας στη δομή του FPGA προσφέρει σημαντικά πλεονεκτήματα τόσο σε απόδοση όσο και σε κατανάλωση ενέργειας. Όπως και τα μπλοκ RAM, τα DSP μπλοκ είναι σχεδιασμένα για βέλτιστη εκτέλεση των αντίστοιχων λειτουργιών, μειώνοντας τον φόρτο των γενικών λογικών πόρων και βελτιώνοντας τη συνολική αποδοτικότητα του συστήματος.

2.4.3 Επεξεργαστής

Μία από τις πιο σημαντικές προσθήκες στις σύγχρονες συσκευές FPGA είναι η ενσωμάτωση ενός επεξεργαστή απευθείας στη δομή τους. Αυτή η προσθήκη αποτελεί σημαντικό βήμα προς την απλοποίηση του σχεδιασμού συστημάτων, ιδιαίτερα για εφαρμογές που απαιτούν και επεξεργαστή και προγραμματιζόμενη λογική. Η χρήση ενός ολοκληρωμένου κυκλώματος με ενσωματωμένο επεξεργαστή μπορεί να αποτελέσει ιδανική λύση, καθώς μειώνει την πολυπλοκότητα του σχεδιασμού, εξοικονομεί πόρους και βελτιώνει την ενεργειακή απόδοση του συστήματος. Αντίθετα, σε FPGA που δεν διαθέτουν ενσωματωμένο επεξεργαστή, ο επεξεργαστής πρέπει να υλοποιηθεί μέσα στη δομή του FPGA ως πυρήνας επεξεργαστή λογισμικού (soft processor core). Αυτή η προσέγγιση απαιτεί επιπλέον πόρους για την υλοποίηση του επεξεργαστή και ενδέχεται να επηρεάσει την απόδοση σε σχέση με τη λύση των ενσωματωμένων επεξεργαστών, αλλά προσφέρει τη δυνατότητα πλήρους προσαρμογής της αρχιτεκτονικής στον εκάστοτε σχεδιασμό.

2.4.4 Διαχειριστής Ψηφιακού Ρολογιού DCM

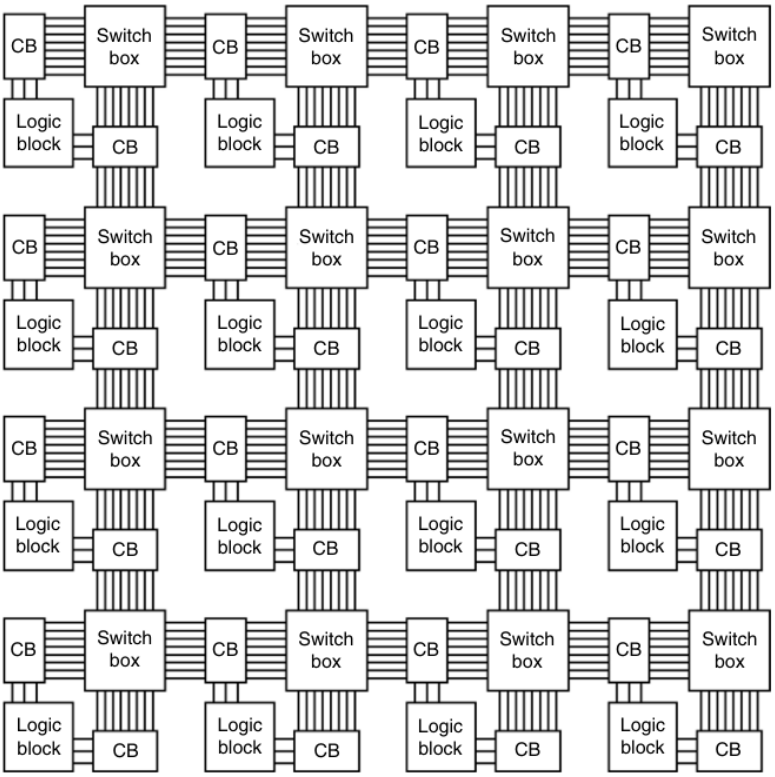
Με την ενσωμάτωση του Digital Clock Manager δίνεται η δυνατότητα δημιουργίας ρολογιών διαφορετικών συχνοτήτων από ένα μόνο ρολόι αναφοράς. Στα περισσότερα συστήματα, υπάρχει μόνο ένα εξωτερικό ρολόι με σταθερή συχνότητα, αλλά σε πολλές εφαρμογές απαιτείται διαφορετικά τμήματα του συστήματος να συγχρονίζονται σε διαφορετικές συχνότητες. Μέσω του DCM μπορούν να δημιουργηθούν νέες συχνότητας διαφορετικών φάσεων και με λιγότερη παραμόρφωση, σε σύγκριση με ένα ρολόι που έχει κατασκευαστεί μέσω των λογικών μπλοκ του FPGA.

2.5 Δίκτυο δρομολόγησης

Το δίκτυο δρομολόγησης παίζει πολύ σημαντικό ρόλο στη λειτουργία των FPGA, καθώς είναι υπεύθυνο για την επικοινωνία μεταξύ των λογικών μπλοκ, μπλοκ μνήμης και των άλλων πόρων του FPGA, επιτρέποντας τη ροή δεδομένων και σημάτων μέσα στο ολοκληρωμένο κύκλωμα. Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές διασύνδεσης, αλλά κυρίως χρησιμοποιείται η island style [7], σχηματική απεικόνιση της οποίας εμφανίζεται στο Σχήμα 2.3, η οποία επιτρέπει την ευέλικτη και αποδοτική σύνδεση των λογικών μπλοκ μέσω ενός δικτύου διασύνδεσης, διασφαλίζοντας τη σωστή επικοινωνία και τη χαμηλή καθυστέρηση μεταξύ των τμημάτων του FPGA.

Το δίκτυο δρομολόγησης αποτελείται από ένα πλέγμα κάθετων και οριζόντιων αγωγών που διασταυρώνονται, σχηματίζοντας ένα ευέλικτο σύστημα διασύνδεσης. Σε συγκεκριμένους κόμβους μέσα σε αυτό το πλέγμα, υπάρχουν προγραμματιζόμενοι διακόπτες (switch boxes – SB) που επιτρέπουν τη σύνδεση μεταξύ των αγωγών, διαμορφώνοντας έτσι τις διαδρομές επικοινωνίας. Τα λογικά μπλοκ και οι υπόλοιποι πόροι του συστήματος συνδέονται με τους αγωγούς του δικτύου δρομολόγησης μέσω των connection boxes (CB). Με αυτόν τον τρόπο, δημιουργείται πλήρης διασύνδεση μεταξύ των πόρων του FPGA, εξασφαλίζοντας την απαραίτητη ευελιξία για την υλοποίηση πολύπλοκων σχεδίων.

Ωστόσο, η πολυπλοκότητα του πλέγματος μπορεί να οδηγήσει σε μεγαλύτερες χρονικές καθυστερήσεις, ειδικά όταν τα σήματα πρέπει να διανύσουν μεγάλες αποστάσεις. Για αυτόν τον λόγο, στα σύγχρονα FPGA συχνά χρησιμοποιούνται αγωγοί μεγαλύτερου μήκους για τη διασύνδεση απομακρυσμένων μπλοκ αποφεύγοντας έτσι τα σήματα να περάσουν από πολλαπλούς προγραμματιζόμενους διακόπτες.



Σχήμα 2.3: Αρχιτεκτονική δομή island style [7].

Τέλος, η καθοριστική σημασία του δικτύου δρομολόγησης στα FPGA για την ευελιξία και την αποδοτικότητά τους αποτυπώνεται στο γεγονός ότι καταλαμβάνει το 80-90% του συνολικού χώρου του κυκλώματος. Αντίθετα, τα λογικά μπλοκ, τα οποία εκτελούν τις υπολογιστικές λειτουργίες, καταλαμβάνουν μόλις το 10-20% [10]. Αυτό υπογραμμίζει τον κρίσιμο ρόλο της δρομολόγησης στη συνολική απόδοση του FPGA, καθώς επηρεάζει τόσο την ταχύτητα όσο και την αποδοτική αξιοποίηση των διαθέσιμων πόρων.

2.6 Μπλοκ εισόδων/εξόδων

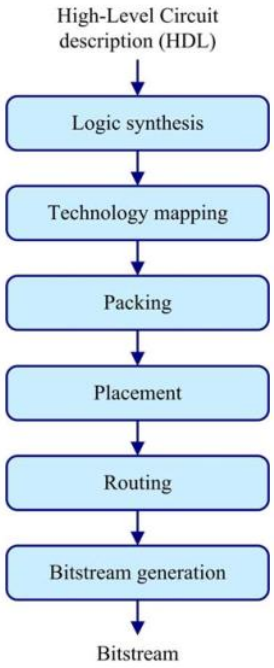
Τα μπλοκ εισόδων/εξόδων στα FPGA αποτελούν κρίσιμες μονάδες που επιτρέπουν τη διασύνδεση της συσκευής με εξωτερικά κυκλώματα και συστήματα. Ο βασικός τους ρόλος είναι η διαχείριση των σημάτων εισόδου και εξόδου, διασφαλίζοντας την αξιόπιστη επικοινωνία με περιφερειακές συσκευές, όπως αισθητήρες, ενεργοποιητές, μνήμες, οθόνες και επεξεργαστές. Αυτά τα μπλοκ μπορούν να λειτουργούν με διάφορους τρόπους, επιτρέποντας τη ρύθμιση των επαφών ως εισόδους, εξόδους ή και σε αμφίδρομη λειτουργία, προσφέροντας ευελιξία στη σχεδίαση του συστήματος. Επιπλέον, υποστηρίζουν πολλαπλά πρότυπα διασύνδεσης, τόσο μονοπολικά όσο και διαφορικά, επιτρέποντας στο FPGA να συνδέεται με διαφορετικούς τύπους συστημάτων χωρίς την ανάγκη πρόσθετου εξοπλισμού προσαρμογής.

Επιπλέον, τα μπλοκ I/O φροντίζουν για τη σωστή διαχείριση του σήματος, εξασφαλίζοντας ότι τα δεδομένα που μεταδίδονται ή λαμβάνονται είναι συγχρονισμένα και δεν προκαλούνται προβλήματα λόγω ανακολουθιών ή καθυστερήσεων. Τα μπλοκ εισόδων/εξόδων παρέχουν επίσης δυνατότητες διαχείρισης διαφορετικών επιπέδων τάσης και ρεύματος επιτρέποντας στο FPGA να συνεργάζεται με συσκευές που χρησιμοποιούν διαφορετικές τιμές τάσης, όπως 3.3V ή 1.8V. Συνολικά, τα μπλοκ

εισόδων/εξόδων είναι κρίσιμα για την αποτελεσματική λειτουργία του FPGA, καθώς εξασφαλίζουν την ομαλή επικοινωνία με το εξωτερικό περιβάλλον και προσφέρουν ευελιξία στη σχεδίαση και την υλοποίηση των εφαρμογών.

2.7 Σχεδιαστική ροή

Η απόδοση και η αποτελεσματικότητα ενός FPGA εξαρτώνται σε σημαντικό βαθμό από την ποιότητα του λογισμικού που το συνοδεύει, το οποίο κάθε εταιρεία αναπτύσσει και παρέχει εξειδικευμένο για τις δικές της πλατφόρμες. Το λογισμικό μετατρέπει τον κώδικα που έχει γραφτεί από τον χρήστη σε γλώσσα περιγραφής υλικού (HDL), συνήθως VHDL ή Verilog, σε ροή bit (bitstream), η οποία καθορίζει τις συνδέσεις και διαμορφώνει τους πόρους του FPGA.



Σχήμα 2.4: Διάγραμμα σχεδιαστικής ροής πληροφορίας [10].

Τα βήματα της σχεδιαστικής ροής, όπως φαίνεται και στο Σχήμα 2.4, είναι logic synthesis, technology mapping, Packing, Placement, Routing, Bitstream generation [10]. Αναλυτικότερα:

- Logic synthesis: Σε αυτό το στάδιο, ο κώδικας HDL μετατρέπεται σε ένα σύνολο από λογικές συναρτήσεις, οι οποίες εκφράζονται με πύλες και flip-flop και ορίζονται οι μεταξύ τους διασυνδέσεις. Το αποτέλεσμα είναι ένα ιεραρχικό δίκτυο που αναπαριστά τη λογική του σχεδιασμού ανεξάρτητα από την τεχνολογία του FPGA.

- Technology mapping: Η τεχνολογικά ανεξάρτητη λογική περιγραφή, του πρώτου σταδίου, μετασχηματίζεται έτσι ώστε να χρησιμοποιεί τους διαθέσιμους πόρους του FPGA, όπως LUT, flip-flop και άλλους διαθέσιμους λογικούς πόρους. Ο στόχος είναι να γίνει η αποδοτικότερη δυνατή αντιστοίχιση της λογικής στα συγκεκριμένα στοιχεία του FPGA.

- Packing: Σε αυτό το στάδιο, ομαδοποιούνται τα LUT και τα flip-flop σε συγκεκριμένες δομές του FPGA, όπως τα Configurable Logic Block (CLB) με στόχο τη μείωση της

πολυπλοκότητας των επόμενων σταδίων και την εξοικονόμηση πόρων.

- Placement: Τα λογικά μπλοκ που προέκυψαν από το Packing τοποθετούνται σε συγκεκριμένες φυσικές θέσεις στη διαθέσιμη αρχιτεκτονική του FPGA. Η διαδικασία της τοποθέτησης προσπαθεί να ελαχιστοποιήσει τις καθυστερήσεις και να βελτιστοποιήσει τη διασύνδεση μεταξύ των μπλοκ, λαμβάνοντας υπόψη τους περιορισμούς του FPGA.

- Routing: Στη φάση αυτή, οι φυσικά τοποθετημένες λογικές μονάδες συνδέονται μέσω του προγραμματιζόμενου δικτύου δρομολόγησης του FPGA. Ο στόχος είναι να διασφαλιστεί η σωστή επικοινωνία μεταξύ των λογικών μπλοκ με το ελάχιστο δυνατό κόστος σε καθυστέρηση και χρήση πόρων και η αποφυγή των αλληλοκαλύψεων των σημάτων.

- Bitstream generation: Το τελικό βήμα είναι η δημιουργία του αρχείου bitstream, το οποίο περιέχει όλες τις πληροφορίες που απαιτούνται για τον προγραμματισμό του FPGA. Το bitstream καθορίζει πώς θα ρυθμιστούν οι φυσικοί πόροι του FPGA για να υλοποιηθεί το επιθυμητό κύκλωμα.

Συνοψίζοντας, η σχεδιαστική ροή ενός FPGA περιλαμβάνει διαδοχικά στάδια που μετατρέπουν τον κώδικα HDL σε ένα πλήρως προγραμματισμένο κύκλωμα. Ο κώδικας περιγραφής υλικού μετατρέπεται σε λογικές συναρτήσεις, η λογική προσαρμόζεται στους διαθέσιμους πόρους, οι οποίοι ομαδοποιούνται σε λογικά μπλοκ και κατανέμονται στη φυσική αρχιτεκτονική του FPGA όπου μέσω της δρομολόγησης διασφαλίζει η σωστή σύνδεση των στοιχείων με βέλτιστη χρήση των πόρων. Τέλος, με τη δημιουργία του bitstram παράγεται το αρχείο το οποίο προγραμματίζει το FPGA.

2.8 Επίλογος

Σε αυτό το κεφάλαιο, παρουσιάστηκε μια ολοκληρωμένη ανάλυση των FPGA, εστιάζοντας στην αρχιτεκτονική, τις τεχνολογίες διαμόρφωσης, τη δομή των λογικών μπλοκ και τη διαδικασία σχεδιασμού τους.

Αρχικά, αναλύθηκε η βασική δομή των FPGA, η οποία περιλαμβάνει λογικά μπλοκ, μπλοκ εισόδων/εξόδων και το δίκτυο δρομολόγησης. Η ευελιξία αυτής της δομής καθιστά τα FPGA κατάλληλα για πληθώρα εφαρμογών, από ενσωματωμένα συστήματα μέχρι προηγμένες εφαρμογές τεχνητής νοημοσύνης.

Στη συνέχεια, εξετάστηκαν οι κύριες τεχνολογίες διαμόρφωσης FPGA, δηλαδή τα SRAM-based, Flash-based και Antifuse-based FPGA. Καθεμία από αυτές διαθέτει μοναδικά χαρακτηριστικά, με τα SRAM-based FPGA να προσφέρουν υψηλή επαναπρογραμματιζόμενη ευελιξία, τα Flash-based να εξασφαλίζουν μη πτητική αποθήκευση δεδομένων και τα Antifuse-based να προσφέρουν μόνιμη προγραμματισμένη λειτουργία για υψηλή αξιοπιστία.

Ιδιαίτερη έμφαση δόθηκε στη λειτουργία των Look-Up Table, τα οποία αποτελούν τον βασικό μηχανισμό υλοποίησης λογικών συναρτήσεων στα FPGA, καθώς και στη χρήση Flip-Flop, που επιτρέπουν την αποθήκευση και συγχρονισμό δεδομένων.

Συζητήθηκε, επίσης, η σημασία των μπλοκ ειδικού σκοπού, όπως οι ενσωματωμένες μνήμες, τα μπλοκ ψηφιακής επεξεργασίας σήματος, οι ενσωματωμένοι επεξεργαστές και τα συστήματα διαχείρισης ρολογιού. Αυτά τα στοιχεία συμβάλλουν στη βελτιστοποίηση της απόδοσης των FPGA, καθιστώντας τα ιδανικά για εφαρμογές υψηλών απαιτήσεων.

Επιπλέον, αναλύθηκε το δίκτυο δρομολόγησης, το οποίο παίζει καθοριστικό ρόλο στη διασύνδεση των λογικών μπλοκ και επηρεάζει σημαντικά την απόδοση του FPGA. Τα μπλοκ εισόδων/εξόδων επιτρέπουν την επικοινωνία με εξωτερικά συστήματα, καθιστώντας το FPGA ένα ευέλικτο εργαλείο

σχεδιασμού ηλεκτρονικών κυκλωμάτων. Τέλος, περιγράφηκε η διαδικασία σχεδιασμού και προγραμματισμού ενός FPGA. Η ποιότητα του λογισμικού ανάπτυξης FPGA επηρεάζει καθοριστικά την απόδοση του τελικού κυκλώματος.

Κεφάλαιο 3ο Γλώσσες Περιγραφής Υλικού

Στο προηγούμενο κεφάλαιο, αναλύσαμε την αρχιτεκτονική και τη λειτουργία των FPGA, τα οποία αποτελούν ένα από τα πιο ευέλικτα εργαλεία για την υλοποίηση ψηφιακών συστημάτων. Τα FPGA χαρακτηρίζονται από την ικανότητά τους να επαναπρογραμματίζονται, προσφέροντας τη δυνατότητα υλοποίησης πολύπλοκων κυκλωμάτων. Ωστόσο, για να αξιοποιηθεί πλήρως η δυναμική αυτών των συσκευών, απαιτείται η χρήση ειδικών γλωσσών που μπορούν να περιγράψουν τη λογική και τη συμπεριφορά του υλικού.

Οι γλώσσες περιγραφής υλικού (Hardware Description Languages - HDL), αναπτύχθηκαν ακριβώς για αυτόν τον σκοπό και αποτελούν ένα θεμελιώδες εργαλείο για το σχεδιασμό, τη μοντελοποίηση και την υλοποίηση ψηφιακών κυκλωμάτων και συστημάτων. Σε αντίθεση με τις κλασικές γλώσσες προγραμματισμού, οι οποίες χρησιμοποιούνται για να περιγράψουν τη ροή των δεδομένων και των εντολών ενός προγράμματος που εκτελείται σε έναν επεξεργαστή γενικού σκοπού, οι γλώσσες περιγραφής υλικού χρησιμοποιούνται για τον καθορισμό της δομής, της λειτουργικότητας και της χρονικής συμπεριφοράς ηλεκτρονικών κυκλωμάτων.

Οι γλώσσες HDL επιτρέπουν στους μηχανικούς να σχεδιάζουν και να προσομοιώνουν την απόδοση ενός κυκλώματος πριν από την κατασκευή του, μειώνοντας το κόστος και τον χρόνο ανάπτυξης. Οι περιγραφές HDL μπορούν να μετατραπούν σε φυσικά κυκλώματα μέσω εργαλείων σύνθεσης, δημιουργώντας έτοιμα σχέδια για τα FPGA. Επιπλέον, οι προσομοιώσεις επιτρέπουν τη δοκιμή της σωστής λειτουργίας των κυκλωμάτων πριν από την υλοποίησή τους σε υλικό. Η VHDL (VHSIC Hardware Description Language) και η Verilog είναι οι δύο κύριες γλώσσες περιγραφής υλικού που χρησιμοποιούνται ευρέως στη βιομηχανία και την ακαδημαϊκή κοινότητα.

Παραδοσιακά, τα ψηφιακά κυκλώματα σχεδιάζονταν με διαγράμματα λογικής και περιγραφές χαμηλού επιπέδου. Οι γλώσσες HDL εισήγαγαν έναν πιο αφηρημένο και δομημένο τρόπο σχεδίασης, επιτρέποντας την καλύτερη διαχείριση της πολυπλοκότητας και την αυτοματοποίηση της παραγωγής κυκλωμάτων.

3.1 Εισαγωγή στην VHDL

Η VHDL (VHSIC Hardware Description Language) είναι μια γλώσσα περιγραφής υλικού που χρησιμοποιείται για τη σχεδίαση, τη σύνθεση και την προσομοίωση ψηφιακών κυκλωμάτων. Χρησιμοποιείται κυρίως στην ανάπτυξη FPGA, CPLD και ASIC, επιτρέποντας την περιγραφή της λειτουργικότητας και της δομής των κυκλωμάτων, καθώς και την επαλήθευση της σωστής λειτουργίας τους μέσω προσομοίωσης.

Στη σύνθεση, η VHDL περιγράφει τη λειτουργία ενός ψηφιακού κυκλώματος, επιτρέποντας στον μεταγλωττιστή να δημιουργήσει ένα φυσικό κύκλωμα που ανταποκρίνεται στις προδιαγραφές και υλοποιεί τη λειτουργία που έχει οριστεί. Στην προσομοίωση, η γλώσσα χρησιμοποιείται για να δοκιμάσει και να επαληθεύσει τη σωστή λειτουργία του σχεδιασμένου κυκλώματος. Αυτό γίνεται δημιουργώντας σήματα που εφαρμόζονται στο κύκλωμα, και στη συνέχεια συγκρίνοντας τις αποκρίσεις του σχεδιασμένου κυκλώματος με τα αναμενόμενα αποτελέσματα. Τα σήματα μπορούν να δημιουργηθούν είτε μέσω κώδικα, είτε χειροκίνητα σε γραφικό περιβάλλον. Αυτή η διαδικασία διευκολύνει σημαντικά την επαλήθευση της ορθότητας του σχεδιασμού, καθώς και τον έλεγχο της λειτουργίας του.

Η VHDL αναπτύχθηκε τη δεκαετία του 1980 ως μέρος του προγράμματος VHSIC (Very High Speed Integrated Circuits) του Υπουργείου Άμυνας των ΗΠΑ, με στόχο τη δημιουργία ενός τυποποιημένου τρόπου περιγραφής ψηφιακών κυκλωμάτων. Η πρώτη επίσημη έκδοσή της ήταν η VHDL-87, ενώ η έκδοση που χρησιμοποιείται σήμερα είναι η VHDL-2008 [11]. Παρόλο που η τελευταία έκδοση είναι αρκετών ετών, η VHDL συνεχίζει να εξελίσσεται και να χρησιμοποιείται ευρέως στη βιομηχανία και την ακαδημαϊκή κοινότητα.

Η γλώσσα VHDL είναι τυποποιημένη από το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE) με το πρότυπο IEEE 1076, εξασφαλίζοντας τη διαλειτουργικότητα και τη συμβατότητα μεταξύ διαφορετικών εργαλείων σχεδίασης και ανάπτυξης υλικού.

Στις επόμενες ενότητες, θα εξετάσουμε τη σύνταξη της VHDL, τις δομές δεδομένων που χρησιμοποιεί και τις βασικές αρχές προγραμματισμού της.

3.2 Δομή και βασική σύνταξη της VHDL

Στη γλώσσα προγραμματισμού VHDL, ένας πλήρης κώδικας αποτελείται από τρία βασικά μέρη: τις βιβλιοθήκες (libraries) και δηλώσεις χρήσης, την οντότητα (entity) και την αρχιτεκτονική (architecture). Οι βιβλιοθήκες περιλαμβάνουν τις απαραίτητες δηλώσεις με τους τύπους δεδομένων για τον κώδικα, η οντότητα καθορίζει τη διεπαφή του κυκλώματος με τις εισόδους και εξόδους του, ενώ η αρχιτεκτονική περιγράφει τη λειτουργικότητα του κυκλώματος. Στο Σχήμα 3.1 απεικονίζεται η βασική δομή ενός κώδικα σε VHDL.

Σχήμα 3.1:Βασική δομή κώδικα VHDL.

Το πρώτο μέρος ενός προγράμματος VHDL περιλαμβάνει τις δηλώσεις βιβλιοθηκών και τη χρήση των απαραίτητων πακέτων. Οι βιβλιοθήκες περιέχουν ορισμούς και συναρτήσεις που απαιτούνται για τη λειτουργία του κώδικα. Συνήθως χρησιμοποιείται η βιβλιοθήκη IEEE, η οποία περιλαμβάνει τύπους δεδομένων και συναρτήσεις που είναι απαραίτητες για την περιγραφή ψηφιακών κυκλωμάτων.

Η βιβλιοθήκη IEEE περιλαμβάνει σημαντικά πακέτα για την περιγραφή ψηφιακών κυκλωμάτων στην VHDL. Το πακέτο STD_LOGIC_1164 παρέχει τον τύπο std_logic για την αναπαράσταση ψηφιακών σημάτων με διαφορετικές καταστάσεις και τον τύπο std_logic_vector για διανύσματα αυτών των σημάτων, οι οποίοι, εκτός από τις τιμές ‘0’ και ‘1’ επιτρέπουν τη χρήση και άλλων τιμών, όπως ‘H’ για υψηλή ισχύ, ‘L’ για χαμηλή ισχύ, ‘X’ για άγνωστη τιμή ή ‘-’ για αδιάφορη τιμή. Το πακέτο NUMERIC_STD παρέχει υποστήριξη για αριθμητικές λειτουργίες μέσω των τύπων signed και

unsigned, που χρησιμοποιούνται για εκτέλεση πράξεων με προσημασμένους και μη προσημασμένους αριθμούς αντίστοιχα. Αυτά τα πακέτα είναι απαραίτητα για την περιγραφή, σύνθεση και προσομοίωση ψηφιακών συστημάτων στην VHDL.

Η δήλωση βιβλιοθηκών και πακέτων γίνεται με τις δεσμευμένες λέξεις library και use. Η library χρησιμοποιείται για να καθορίσει τη βιβλιοθήκη που περιέχει τα πακέτα, ενώ η use επιτρέπει την εισαγωγή συγκεκριμένων πακέτων από τη βιβλιοθήκη, ώστε να είναι διαθέσιμα στον κώδικα.

Η οντότητα καθορίζει τη διεπαφή της ψηφιακής μονάδας, ορίζοντας το όνομά της, τα σήματα εισόδων/εξόδων και τους τύπους δεδομένων που χρησιμοποιεί. Δεν περιγράφει τη λειτουργία του κυκλώματος, αλλά καθορίζει μόνο πώς αυτό επικοινωνεί με το εξωτερικό περιβάλλον. Τα σήματα εισόδων/εξόδων μπορεί να είναι είσοδοι που δηλώνονται με τη δεσμευμένη λέξη ‘in’, έξοδοι που δηλώνονται ως ‘out’, ως ‘inout’ δηλώνονται σήματα που μπορούν να λειτουργούν και ως είσοδοι και ως έξοδοι, ενώ σήματα εξόδου που διατηρούν την τιμή τους για χρήση στο εσωτερικό του κυκλώματος δηλώνονται με τη δεσμευμένη λέξη ‘buffer’. Για παράδειγμα, σε μια πύλη AND, η οντότητα μπορεί να περιλαμβάνει δύο εισόδους και μία έξοδο, όλες τύπου std_logic. Η λειτουργικότητα της μονάδας περιγράφεται στο architecture, το οποίο συνοδεύει το entity και καθορίζει τη συμπεριφορά του κυκλώματος.

Η αρχιτεκτονική καθορίζει τη λειτουργικότητα της ψηφιακής μονάδας, δηλαδή πώς οι έξοδοι παράγονται από τις εισόδους. Κάθε entity πρέπει να έχει τουλάχιστον μία architecture, η οποία περιγράφει είτε τη συμπεριφορά (behavioral modeling) είτε τη δομή (structural modeling) του κυκλώματος. Η σύνταξη της αρχιτεκτονικής περιλαμβάνει το όνομά της, το αντίστοιχο entity, και τη λογική που το υλοποιεί. Για παράδειγμα, σε μια πύλη AND, το architecture μπορεί να περιέχει μια απλή ανάθεση `Y <= A and B;`, που καθορίζει τη λειτουργία της.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
-- Ορισμός της οντότητας (entity)
Entity AND_GATE is
    Port (
        A, B : in  STD_LOGIC; -- Δύο είσοδοι τύπου std_logic
        Y    : out STD_LOGIC  -- Έξοδος τύπου std_logic
    );
End AND_GATE;

-- Ορισμός της αρχιτεκτονικής (architecture)
Architecture Behavioral of AND_GATE is
Begin
   Y <= A and B;--Η έξοδος Y είναι το λογικό AND των εισόδων A και B
End Behavioral;
```

Κώδικας 3.1: Κώδικας πύλης AND σε VHDL.

Ο Κώδικας 3.1 περιγράφει μια πύλη AND σε VHDL, χρησιμοποιώντας τη βιβλιοθήκη IEEE.STD_LOGIC_1164, η οποία παρέχει τον τύπο std_logic για την αναπαράσταση των ψηφιακών σημάτων. Η οντότητα AND_GATE καθορίζει τις εξωτερικές διεπαφές του κυκλώματος, με δύο εισόδους τύπου std_logic (A και B) και μία έξοδο τύπου std_logic (Y). Η αρχιτεκτονική Behavioral περιγράφει τη λειτουργία της πύλης, όπου η έξοδος Y υπολογίζεται ως το λογικό AND των εισόδων A και B. Ο κώδικας χρησιμοποιεί την έκφραση `Y <= A and B;` για να υλοποιήσει αυτή τη λογική πράξη, επιτυγχάνοντας έτσι τη λειτουργία της πύλης AND.

3.3 Δεσμευμένες λέξεις και ονόματα

Στη γλώσσα VHDL, όπως και σε πολλές άλλες γλώσσες προγραμματισμού, υπάρχουν δεσμευμένες λέξεις. Αυτές είναι λέξεις που έχουν προκαθορισμένο νόημα και ρόλο και χρησιμοποιούνται για τη δήλωση εντολών, τύπων, οντοτήτων κ.λπ. Δεν μπορούν να χρησιμοποιηθούν ως αναγνωριστικά (όπως ονόματα οντοτήτων, σημάτων, μεταβλητών κ.λπ.).

Στον Κώδικας 3.1 οι λέξεις Library, Use, Entity, is, Port, in, out, Architecture, End, AND είναι όλες δεσμευμένες και δεν μπορούν να χρησιμοποιηθούν για άλλα στοιχεία του προγράμματος, όπως τα ονόματα μεταβλητών ή σημάτων, διότι θα προκληθούν συντακτικά λάθη. Για παράδειγμα, το όνομα της οντότητας δεν μπορεί να είναι AND, καθώς η λέξη AND έχει ήδη προκαθορισμένο ρόλο, οπότε ονομάσαμε την οντότητα AND_GATE. Άλλες δεσμευμένες λέξεις είναι οι if, then, else, elsif, when, with, component κ.α.

Στη VHDL, τα ονόματα αναφέρονται σε αντικείμενα όπως μεταβλητές, σήματα, τύποι, διαδικασίες, οντότητες και αρχιτεκτονικές. Αυτά τα ονόματα πρέπει να ακολουθούν συγκεκριμένους κανόνες και περιορισμούς. Ένα όνομα μπορεί να περιλαμβάνει, αλφαριθμητικούς χαρακτήρες και το σύμβολο '_', αλλά πρέπει να ξεκινάει πάντα με γράμματα και να τελειώνει με γράμμα ή αριθμό. Η VHDL είναι case-insensitive, δηλαδή δεν κάνει διάκριση ανάμεσα σε κεφαλαία και πεζά γράμματα, για παράδειγμα τα ονόματα and_gate και AND_GATE θεωρούνται το ίδιο.

Το σύμβολο ';' χρησιμοποιείται ως τερματικό σύμβολο στο τέλος κάθε εντολής για να δηλώσει την ολοκλήρωσή της. Αυτό το σημείο είναι απαραίτητο για τη σωστή σύνταξη του κώδικα, καθώς χωρίς το ';' η εντολή θεωρείται ατελής και προκαλεί σφάλμα κατά την ανάλυση του κώδικα. Τέλος, το σύμβολο '--' χρησιμοποιείται για να εισάγει σχόλια στον κώδικα. Όλα τα κείμενα που ακολουθούν το '--' στην ίδια γραμμή αγνοούνται κατά την εκτέλεση του προγράμματος και χρησιμεύουν για να εξηγήσουν ή να περιγράψουν το σκοπό τμημάτων του κώδικα, βοηθώντας στην αναγνωσιμότητα και την κατανόηση του προγράμματος. Τα σχόλια είναι ιδιαίτερα χρήσιμα για τη τεκμηρίωση του κώδικα και για την επικοινωνία με άλλους προγραμματιστές ή για μελλοντική αναφορά.

3.4 Κυκλώματα συνδυαστικής λογικής

Τα κυκλώματα συνδυαστικής λογικής είναι τα βασικότερα ψηφιακά κυκλώματα των οποίων η έξοδος καθορίζεται αποκλειστικά από τις τρέχουσες τιμές των εισόδων τους, χωρίς να επηρεάζεται από προηγούμενες καταστάσεις ή χρονισμό. Ο Κώδικας 3.2 περιγράφει τη λειτουργία ενός πολυπλέκτη 4 σε 1 με την εντολή with-select και αποτελεί ένα παράδειγμα περιγραφής κυκλώματος συνδυαστικής λογικής. Σε αυτή την περίπτωση, το σήμα επιλογής (SEL), που αποτελείται από δύο bits, καθορίζει ποια από τις τέσσερις εισόδους (A, B, C, D) θα συνδεθεί στην έξοδο Y. Το κύκλωμα δεν αποθηκεύει καμία προηγούμενη κατάσταση, ούτε εξαρτάται από παλμό ρολογιού, αλλά επιλέγει και αποδίδει άμεσα την τιμή της αντίστοιχης εισόδου στην έξοδο, βασισμένο στις τρέχουσες τιμές του σήματος επιλογής.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;

Entity MUX4to1 is
    Port (
        A, B, C, D : in  STD_LOGIC;                     -- 4 είσοδοι
        SEL        : in  STD_LOGIC_VECTOR (1 downto 0); -- 2-bit επιλογή
        Y          : out STD_LOGIC                      -- Έξοδος
    );
end MUX4to1;
```

```
Architecture Behavioral of MUX4to1 is
Begin
    -- Επιλογή εξόδου με βάση την τιμή του SEL
    with SEL select
    Y <= A when "00",     -- Αν SEL = "00", Y = A
         B when "01",     -- Αν SEL = "01", Y = B
         C when "10",     -- Αν SEL = "10", Y = C
         D when "11",     -- Αν SEL = "11", Y = D
        '0' when others; -- Προεπιλεγμένη τιμή (για κάλυψη όλων των περιπτώσεων)
End Behavioral;
```

Κώδικας 3.2: Κώδικας συνδυαστικής λογικής.

Η VHDL λειτουργεί με παράλληλη εκτέλεση των εντολών, γεγονός που αντανακλά τη φύση των ψηφιακών κυκλωμάτων. Οι εντολές και οι διαδικασίες στη VHDL εκτελούνται ταυτόχρονα κατά τη διάρκεια της προσομοίωσης ή σύνθεσης, χωρίς να περιμένουν η μία την άλλη. Στην περίπτωση του Κώδικας 3.2, η εντολή with-select εκτελείται παράλληλα με οποιεσδήποτε άλλες εντολές βρίσκονται στο ίδιο επίπεδο της αρχιτεκτονικής, ανεξάρτητα από το σε ποια θέση του κώδικα βρίσκεται. Αυτό σημαίνει ότι το κύκλωμα εκτελεί όλους τους λογικούς υπολογισμούς για τις διάφορες καταστάσεις του σήματος επιλογής ταυτόχρονα, και η έξοδος Y ενημερώνεται αμέσως με την αντίστοιχη τιμή χωρίς καθυστερήσεις από εξωτερικούς χρονισμούς ή αποθηκευτικές καταστάσεις.

3.5 Κυκλώματα ακολουθιακής λογικής

Σε αντίθεση με τα κυκλώματα συνδιαστικής λογικής όπου η έξοδός τους εξαρτάται αποκλειστικά από τις τρέχουσες τιμές των εισόδων τους, η έξοδος των κυκλωμάτων ακολουθιακής λογικής εξαρτάται τόσο από τις τρέχουσες εισόδους όσο και από προηγούμενες καταστάσεις του συστήματος, διατηρώντας μνήμη της προηγούμενης λειτουργίας τους. Αυτό γίνεται δυνατό χάρη στη χρήση στοιχείων μνήμης, όπως flip-flop και register, τα οποία αποθηκεύουν πληροφορίες και επιτρέπουν στα κυκλώματα να "θυμούνται" προηγούμενες καταστάσεις. Συνήθως, τα κυκλώματα ακολουθιακής λογικής λειτουργούν με χρονισμό, δηλαδή η αλλαγή της κατάστασής τους συγχρονίζεται με ένα σήμα ρολογιού (clock). Το ρολόι καθορίζει τις χρονικές στιγμές στις οποίες μπορεί να αλλάξει η κατάσταση του κυκλώματος, εξασφαλίζοντας σταθερότητα και σωστή ακολουθία λειτουργίας.

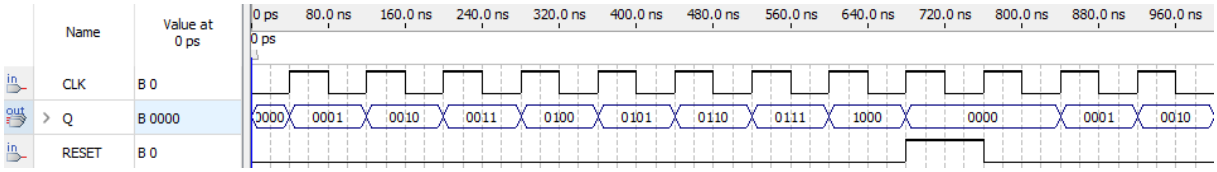
Η εντολή process παίζει καθοριστικό ρόλο στον σχεδιασμό κυκλωμάτων ακολουθιακής λογικής, καθώς επιτρέπει την περιγραφή της διαδοχής καταστάσεων και την ενσωμάτωση του χρονισμού. Μέσω του process, μπορούμε να διαχειριστούμε την αποθήκευση και την ενημέρωση της τρέχουσας κατάστασης του κυκλώματος, κάτι που είναι απαραίτητο για τη σωστή λειτουργία των ακολουθιακών συστημάτων. Επιπλέον, το process επιτρέπει τη χρήση χρονισμού μέσω ενός σήματος ρολογιού, συνήθως με την εντολή rising_edge(clk), ώστε οι αλλαγές κατάστασης να γίνονται σε συγκεκριμένες χρονικές στιγμές. Αυτό διασφαλίζει σταθερότητα και προβλεψιμότητα στη λειτουργία του κυκλώματος.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Counter_4bit is
    Port (
```

```
        CLK    : in  STD_LOGIC;                     -- Σήμα ρολογιού
        RESET  : in  STD_LOGIC;                     -- Σήμα reset (αρχικοποίηση)
        Q      : out STD_LOGIC_VECTOR (3 downto 0) -- Έξοδος 4-bit
    );
end Counter_4bit;

architecture Behavioral of Counter_4bit is
    signal count : STD_LOGIC_VECTOR (3 downto 0) := "0000"; -- Αρχική τιμή 0
begin
    process (CLK, RESET)
    begin
        if RESET = '1' then
            count <= "0000"; -- Όταν το RESET είναι ενεργό, ο μετρητής μηδενίζεται
        elsif rising_edge(CLK) then
            count <= count + 1; -- Αυξάνουμε την τιμή του μετρητή κατά 1 σε κάθε
παλμό ρολογιού
        end if;
    end process;

    Q <= count; -- Αντιστοίχιση της εξόδου
end Behavioral;
```

Κώδικας 3.3: Κώδικας κυκλώματος ακολουθιακής λογικής.

Ο Κώδικας 3.3 αποτελεί ένα παράδειγμα περιγραφής κυκλώματος ακολουθιακής λογικής. Η λειτουργία του κυκλώματος βασίζεται στη χρήση ενός process, το οποίο ενεργοποιείται κάθε φορά που ανιχνεύεται είτε αλλαγή στο CLK είτε στο RESET. Τα CLK και RESET βρίσκονται μέσα στην παρένθεση στη δήλωση του process, εντός της οποίας δηλώνεται η λίστα ευαισθησίας του. Όπως φαίνεται και στο Σχήμα 3.2, αν το σήμα RESET είναι ενεργό ‘1’, ο μετρητής τίθεται στην αρχική του κατάσταση, δηλαδή ‘0000’. Διαφορετικά, αν ανιχνευτεί ανερχόμενη ακμή του ρολογιού (rising_edge(CLK)), τότε η τιμή του μετρητή αυξάνεται κατά 1.



Σχήμα 3.2: Προσομοίωση κώδικα 4.3.

Αν και το process εκτελείται παράλληλα με τα υπόλοιπα τμήματα της αρχιτεκτονικής, οι τιμές της εξόδου Q ανανεώνονται μόνο όταν το σήμα RESET γίνει '1' ή όταν ανιχνευθεί ανερχόμενη ακμή του σήματος ρολογιού. Αυτή η διαδικασία διασφαλίζει ότι οι αλλαγές στην έξοδο συμβαίνουν με χρονικό συγχρονισμό και ελεγχόμενη επαναφορά στις αρχικές τιμές όταν απαιτείται.

3.6 Υποσυστήματα components

Ένα από τα πιο σημαντικά χαρακτηριστικά της VHDL είναι η δυνατότητα δημιουργίας και χρήσης υποσυστημάτων, τα οποία επιτρέπουν την οργάνωση και την επαναχρησιμοποίηση κώδικα.

Στην VHDL τα component χρησιμοποιούνται για τη δημιουργία ιεραρχικών σχεδιάσεων, επιτρέποντας την αναπαράσταση πολύπλοκων ψηφιακών κυκλωμάτων ως συνδυασμό μικρότερων, επαναχρησιμοποιήσιμων υποσυστημάτων και την καλύτερη οργάνωση του κώδικα.

Ένα component αντιστοιχεί σε ένα υποκύκλωμα, το οποίο μπορεί να δηλωθεί και να χρησιμοποιηθεί μέσα σε ένα μεγαλύτερο κύκλωμα. Για να γίνει αυτό, απαιτούνται δύο βασικά βήματα. Πρώτον, το component πρέπει να δηλωθεί μέσα στην αρχιτεκτονική του ανώτερου κυκλώματος, καθορίζοντας τις εισόδους και τις εξόδους του. Στη συνέχεια, πρέπει να γίνει η αντιστοίχισή του με την οντότητα (entity) του ανώτερου κυκλώματος, ώστε να συνδεθεί με την πραγματική λειτουργικότητά του και να αντιστοιχισθούν τα σήματα του κύριου κυκλώματος με τα σήματα του component.

Η χρήση των component προσφέρει σημαντικά πλεονεκτήματα στον σχεδιασμό ψηφιακών κυκλωμάτων. Επιτρέπει την αναλυτική περιγραφή μεγάλων σχεδιάσεων μέσω της σύνθεσης μικρότερων υπομονάδων, γεγονός που διευκολύνει τη συντήρηση, την κατανόηση και την επαναχρησιμοποίηση κώδικα. Επιπλέον, η ιεραρχική σχεδίαση που προκύπτει καθιστά τον σχεδιασμό πιο οργανωμένο και διαχειρίσιμο.

Ο Κώδικας 3.4 αποτελεί ένα παράδειγμα περιγραφής κυκλώματος με χρήση component, το οποίο επιτρέπει την αναφορά και επαναχρησιμοποίηση ενός υποκυκλώματος μέσα σε ένα ανώτερο κύκλωμα. Στην προκειμένη περίπτωση, το κύκλωμα περιλαμβάνει δύο πολυπλέκτες 4 σε 1, που έχουν την ίδια λειτουργία με τον πολυπλέκτη που περιγράφεται στον Κώδικας 3.2.

Αντί να γράψουμε δύο φορές τον ίδιο κώδικα για τον πολυπλέκτη, δηλώσαμε αρχικά το component, καθορίζοντας το όνομά του καθώς και τις εισόδους και εξόδους του. Με αυτόν τον τρόπο, το component λειτουργεί ως μια αφηρημένη περιγραφή του υποκυκλώματος, η οποία μπορεί να χρησιμοποιηθεί πολλαπλές φορές μέσα στο ανώτερο κύκλωμα χωρίς περιττή επανάληψη κώδικα.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;

-- Ανώτερη οντότητα που περιέχει δύο MUX4to1
Entity DualMUX is
    Port (
        A1, B1, C1, D1 : in  STD_LOGIC;  -- Είσοδοι για τον πρώτο MUX
        A2, B2, C2, D2 : in  STD_LOGIC;  -- Είσοδοι για τον δεύτερο MUX
        SEL1, SEL2     : in  STD_LOGIC_VECTOR (1 downto 0); -- Επιλογές για
τους δύο MUX
        Y1, Y2         : out STD_LOGIC  -- Έξοδοι των δύο MUX
    );
end DualMUX;

Architecture Structural of DualMUX is

    -- Δήλωση του component MUX4to1
    Component MUX4to1
        Port (
            A, B, C, D : in  STD_LOGIC;
            SEL        : in  STD_LOGIC_VECTOR (1 downto 0);
            Y          : out STD_LOGIC
        );
    end Component;

Begin

    -- Πρώτος πολυπλέκτης 4 προς 1
    MUX1: MUX4to1 port map (
        A   => A1,
        B   => B1,
        C   => C1,
        D   => D1,
        SEL => SEL1,
```

```
        Y   => Y1
    );


    -- Δεύτερος πολυπλέκτης 4 προς 1
    MUX2: MUX4to1 port map (
        A   => A2,
        B   => B2,
        C   => C2,
        D   => D2,
        SEL => SEL2,
        Y   => Y2
    );

End Structural;
```

Κώδικας 3.4: Κώδικας περιγραφής κυκλώματος με χρήση component.

Στη συνέχεια, στο κύριο σώμα της αρχιτεκτονικής, δημιουργήσαμε δύο στιγμιότυπα του component και πραγματοποιήσαμε την αντιστοίχιση των σημάτων του ανώτερου κυκλώματος με τις εισόδους και εξόδους κάθε πολυπλέκτη. Αυτή η διαδικασία διασφαλίζει ότι κάθε πολυπλέκτης λαμβάνει τα κατάλληλα δεδομένα εισόδου και επιλέγει τη σωστή έξοδο σύμφωνα με τα αντίστοιχα σήματα επιλογής.

Η αντιστοίχιση σημάτων στο port map μπορεί να γίνει είτε με ονομαστική αντιστοίχιση '=>', όπως στον Κώδικας 3.4, όπου κάθε σήμα αντιστοιχίζεται ρητά στο αντίστοιχο port, είτε με αντιστοίχιση θέσης, όπου τα σήματα δηλώνονται με τη σειρά που έχουν οριστεί στο entity. Η ονομαστική αντιστοίχιση είναι πιο ευανάγνωστη και λιγότερο επιρρεπής σε λάθη, ενώ η αντιστοίχιση θέσης απαιτεί αυστηρή τήρηση της σειράς των σημάτων. Για παράδειγμα, η δήλωση

```
MUX1: MUX4to1 port map (A1, B1, C1, D1, SEL1, Y1);
MUX2: MUX4to1 port map (A2, B2, C2, D2, SEL2, Y2);
```

χρησιμοποιεί αντιστοίχιση θέσης, καθώς τα σήματα περνιούνται στη σωστή σειρά χωρίς ρητή αναφορά στα ports.

Η χρήση του component συμβάλλει στην καλύτερη οργάνωση του κώδικα, διευκολύνει την επαναχρησιμοποίησή του και καθιστά το κύκλωμα πιο ευέλικτο και εύκολο στη συντήρηση. Επιπλέον, επιτρέπει την τροποποίηση ή αντικατάσταση του υποκυκλώματος χωρίς να επηρεάζονται τα υπόλοιπα μέρη του κυκλώματος, βελτιώνοντας έτσι τη δομή και την επεκτασιμότητα του σχεδιασμού.

3.7 Μηχανές Πεπερασμένων Καταστάσεων FSM

Οι Μηχανές Πεπερασμένων Καταστάσεων (Finite-State Machines - FSM) είναι ένα πολύ σημαντικό εργαλείο στο πεδίο του ψηφιακού σχεδιασμού και χρησιμοποιούνται ευρέως στην VHDL για την περιγραφή κυκλωμάτων και συστημάτων που περνούν από μια σειρά πεπερασμένων καταστάσεων, με κάθε κατάσταση να έχει καθορισμένη έξοδο και συγκεκριμένους κανόνες για την μετάβασή της σε άλλες καταστάσεις. Οι FSM χρησιμοποιούνται για την αναπαράσταση και την υλοποίηση ακολουθιακής λογικής, όπου η έξοδος εξαρτάται όχι μόνο από τις τρέχουσες εισόδους αλλά και από τις προηγούμενες καταστάσεις του συστήματος [11].

Οι καταστάσεις (states), οι μεταβάσεις (transitions) και οι έξοδοι (outputs) είναι τα βασικά στοιχεία τα οποία καθορίζουν μία FSM. Οι καταστάσεις είναι οι διαφορετικές λειτουργικές συνθήκες στις οποίες μπορεί να βρεθεί ένα σύστημα. Σε κάθε παλμό ρολογιού που χρονίζει την FSM το σύστημα μπορεί να βρίσκεται σε μία μόνο κατάσταση. Οι μεταβάσεις καθορίζουν πώς και πότε το σύστημα αλλάζει από τη μια κατάσταση στην άλλη. Η μετάβαση από μία κατάσταση σε μια άλλη γίνεται βάσει συγκεκριμένων συνθηκών, οι οποίες συνήθως περιλαμβάνουν τις εισόδους του συστήματος και την τρέχουσα κατάσταση. Οι έξοδοι μιας μηχανής πεπερασμένων καταστάσεων εξαρτώνται από την τρέχουσα κατάσταση και, σε ορισμένες περιπτώσεις, από τις εισόδους του συστήματος. Ανάλογα με τη δομή της FSM, οι έξοδοι μπορούν να παράγονται είτε μόνο από την κατάσταση, όπως στις μηχανές Moore, είτε να επηρεάζονται και από τις εισόδους, όπως στις μηχανές Mealy [11]. Οι έξοδοι αυτές μπορεί να καθορίζουν ενέργειες που πρέπει να εκτελεστούν ή σήματα που πρέπει να σταλούν σε άλλα τμήματα του συστήματος.

Συνολικά, η FSM λειτουργεί ως ένας μηχανισμός ελέγχου που μεταβαίνει από κατάσταση σε κατάσταση, με βάση τις εισόδους και τις καθορισμένες μεταβάσεις, καθορίζοντας έτσι την επιθυμητή συμπεριφορά του συστήματος.

Ο Κώδικας 3.5 είναι ένα παράδειγμα κώδικα VHDL υλοποίησης ενός ελεγκτή φαναριών κυκλοφορίας βασισμένο σε FSM. Στόχος είναι ο έλεγχος της λειτουργίας των φαναριών ενός δρόμου για αυτοκίνητα και πεζούς.

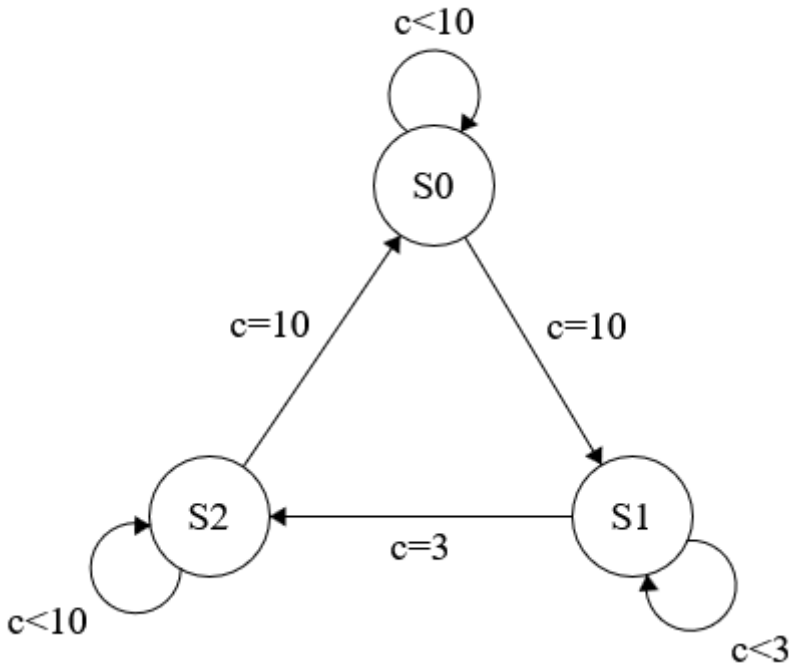
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
--use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Traffic_Light_FSM is
    Port (
        CLK         : in  STD_LOGIC; -- Ρολόι συστήματος
        RESET       : in  STD_LOGIC; -- Επαναφορά
        MAIN_RED    : out STD_LOGIC;
        MAIN_AMBER  : out STD_LOGIC;
        MAIN_GREEN  : out STD_LOGIC;
        PED_RED     : out STD_LOGIC;
        PED_GREEN   : out STD_LOGIC
    );
end Traffic_Light_FSM;

architecture Behavioral of Traffic_Light_FSM is
    type state_type is (S0, S1, S2); -- Καταστάσεις φαναριών
    signal current_state, next_state : state_type;
    signal counter : INTEGER := 0; -- Μετρητής καθυστέρησης

    constant GREEN_TIME  : INTEGER := 10; -- Διάρκεια πράσινου
    constant AMBER_TIME  : INTEGER := 3;  -- Διάρκεια πορτοκαλί
    constant RED_TIME    : INTEGER := 10; -- Διάρκεια κόκκινου στον κύριο
δρόμο (πράσινο στον παράδρομο)

begin

    -- Process για αλλαγή καταστάσεων σε κάθε παλμό του ρολογιού
    process (CLK, RESET)
    begin
        if RESET = '1' then
            current_state <= S0;
            counter <= 0;
        elsif rising_edge(CLK) then
```

```
            if counter = GREEN_TIME and current_state = S0 then
                current_state <= S1;
                counter <= 0;
            elsif counter = AMBER_TIME and current_state = S1 then
                current_state <= S2;
                counter <= 0;
            elsif counter = RED_TIME and current_state = S2 then
                current_state <= S0;
                counter <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    -- Process για ελέγχους εξόδου των φαναριών
    process (current_state)
    begin
        case current_state is
            when S0 =>  -- Πράσινο
                MAIN_GREEN <= '1';
                MAIN_AMBER <= '0';
                MAIN_RED <= '0';
                PED_GREEN <= '0';
                PED_RED <= '1';

            when S1 =>  -- Πορτοκαλί
                MAIN_GREEN <= '0';
                MAIN_AMBER <= '1';
                MAIN_RED <= '0';
                PED_GREEN <= '0';
                PED_RED <= '1';

            when S2 =>  -- Κόκκινο
                MAIN_GREEN <= '0';
                MAIN_AMBER <= '0';
                MAIN_RED <= '1';
                PED_GREEN <= '1';
                PED_RED <= '0';

            when others =>
                MAIN_GREEN <= '0';
                MAIN_AMBER <= '0';
                MAIN_RED <= '1';
                PED_GREEN <= '0';
                PED_RED <= '1';
        end case;
    end process;

end Behavioral;
```

Κώδικας 3.5: Κώδικας περιγραφής ελεγκτή φαναριών.

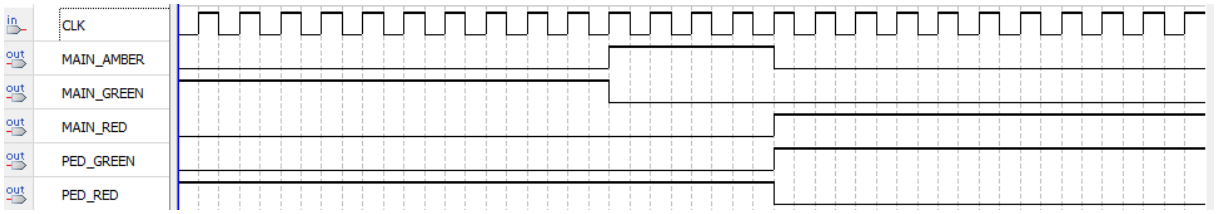
Ο κώδικας χρησιμοποιεί τα τρία χρώματα των φωτεινών σηματοδοτών, κόκκινο, πορτοκαλί και πράσινο, και αποτελείται από τρεις καταστάσεις, S0, S1 και S2, οι οποίες μεταβαίνουν κυκλικά με βάση έναν μετρητή χρονισμού. Στην βασική κατάσταση S0 το φανάρι του κύριου δρόμου είναι πράσινο, επιτρέποντας στα οχήματα να περάσουν, ενώ το φανάρι των πεζών είναι κόκκινο, εμποδίζοντας τη διέλευσή τους. Η κατάσταση διαρκεί 10 χρονικούς κύκλους, ορισμένους από τη σταθερά GREEN_TIME. Όταν περάσουν 10 κύκλοι, το κύκλωμα μεταβαίνει στην κατάσταση S1,

όπου το φανάρι του κύριου δρόμου γίνεται πορτοκαλί, προειδοποιώντας τα οχήματα να επιβραδύνουν, ενώ το φανάρι των πεζών παραμένει κόκκινο. Αυτή η κατάσταση διαρκεί 3 κύκλους (AMBER_TIME). Στη συνέχεια, η FSM μεταβαίνει στην S2, όπου το φανάρι του κύριου δρόμου γίνεται κόκκινο και το φανάρι των πεζών γίνεται πράσινο, επιτρέποντας στους πεζούς να διασχίσουν τον δρόμο με ασφάλεια. Αυτή η κατάσταση διαρκεί 10 κύκλους (RED_TIME). Μετά το πέρας των 10 κύκλων στην κατάσταση S2, το κύκλωμα επιστρέφει στην S0, επαναλαμβάνοντας την διαδικασία. Στο Σχήμα 3.3 απεικονίζεται το σχηματικό διάγραμμα του συστήματος, όπου c είναι ο μετρητής. Ενώ στο Σχήμα 3.4 απεικονίζονται οι αλλαγές των καταστάσεων των σημάτων εξόδου.



Σχήμα 3.3: Σχηματικό διάγραμμα FSM.

Το σύστημα χρησιμοποιεί τους παλμούς ενός ρολογιού (clk) για να συγχρονίζει τις μεταβάσεις των καταστάσεων και έναν μετρητή (counter), ο οποίος, αυξάνοντας σε κάθε θετική ακμή του ρολογιού, καταγράφει το χρονικό διάστημα που το σύστημα παραμένει σε κάθε κατάσταση. Όταν ο μετρητής φτάσει σε μια προκαθορισμένη τιμή (GREEN_TIME, AMBER_TIME, RED_TIME), η FSM μεταβαίνει στην επόμενη κατάσταση, στέλνει σήματα που να ανάψουν ή να σβήνουν τα χρώματα στα φανάρια και μηδενίζεται ο μετρητής. Τέλος με την εντολή when others καλύπτεται κάθε περίπτωση που δεν έχει καθοριστεί ρητά και το σύστημα προχωρά σε μια ασφαλή κατάσταση όπου ενεργοποιεί το κόκκινο για όλα τα φανάρια.



Σχήμα 3.4: Προσομοίωση κώδικα 3.5.

3.8 Επίλογος

Στο κεφάλαιο αυτό, παρουσιάσαμε τις γλώσσες περιγραφής υλικού (HDL), οι οποίες επιτρέπουν τον σχεδιασμό, τη μοντελοποίηση, την προσομοίωση και την υλοποίηση ψηφιακών κυκλωμάτων, με έμφαση στην VHDL.

Εξετάσαμε τη δομή ενός προγράμματος VHDL, η οποία περιλαμβάνει βιβλιοθήκες, οντότητα και αρχιτεκτονική. Οι βιβλιοθήκες περιέχουν τύπους δεδομένων και συναρτήσεις, η οντότητα καθορίζει τη διεπαφή εισόδων/εξόδων, ενώ η αρχιτεκτονική περιγράφει τη λειτουργικότητα του κυκλώματος. Τα κυκλώματα μπορεί να είναι συνδυαστικής ή ακολουθιακής λογικής. Τα συνδυαστικά κυκλώματα καθορίζονται αποκλειστικά από τις εισόδους τους, ενώ τα ακολουθιακά αποθηκεύουν προηγούμενες καταστάσεις και συγχρονίζονται μέσω ρολογιού.

Αναφερθήκαμε στη χρήση components, η οποία εξυπηρετει την ιεραρχική σχεδίαση κυκλωμάτων και την επαναχρησιμοποίηση κώδικα, διευκολύνοντας τη συντήρηση και την οργάνωση πολύπλοκων σχεδιάσεων. Επιπλέον, έγινε αναφορά στις μηχανές πεπερασμένων καταστάσεων που χρησιμοποιούνται για τη διαχείριση ακολουθιακών λειτουργιών, επιτρέποντας την αναπαράσταση σύνθετων συμπεριφορών κυκλωμάτων.

Συνολικά, η VHDL αποτελεί ένα ισχυρό εργαλείο για τον σχεδιασμό ψηφιακών συστημάτων, προσφέροντας ακριβή έλεγχο της συμπεριφοράς των κυκλωμάτων και τη βελτίωση της απόδοσης μέσω προσομοιώσεων και αυτόματης σύνθεσης.

Κεφάλαιο 4ο Σχεδίαση Ψηφιακού Ρολογιού

Στο παρόν κεφάλαιο, θα αναλυθεί η διαδικασία σχεδίασης και υλοποίησης ενός ψηφιακού ρολογιού χρησιμοποιώντας προγραμματιζόμενη λογική σε FPGA, με στόχο την κατανόηση των τεχνικών και των βημάτων που απαιτούνται για την ανάπτυξη ενός πλήρως λειτουργικού συστήματος. Η υλοποίηση του ψηφιακού ρολογιού πραγματοποιήθηκε στη γλώσσα περιγραφής υλικού VHDL, η οποία προσφέρει την απαραίτητη ευχέρεια και ακρίβεια για την περιγραφή των ψηφιακών κυκλωμάτων. Το σύστημα υλοποιήθηκε στην αναπτυξιακή πλακέτα DE1-SoC της εταιρίας Terasic, η οποία ενσωματώνει το FPGA 5CSEMA5F31C6 από τη σειρά Cyclone V της Intel.

Το κεφάλαιο αυτό επικεντρώνεται στην ανάλυση των απαιτήσεων και των βασικών λειτουργιών του ψηφιακού ρολογιού, καθώς και των τεχνικών που χρησιμοποιήθηκαν για να επιτευχθούν οι στόχοι της σχεδίασης. Ο σαφής ορισμός των απαιτήσεων είναι κρίσιμος για την κατανόηση των λειτουργιών που το σύστημα πρέπει να υποστηρίζει, εξασφαλίζοντας έτσι την επιτυχή υλοποίηση ενός αξιόπιστου και ακριβούς ψηφιακού ρολογιού. Συγκεκριμένα, θα γίνει ανάλυση των λειτουργικών απαιτήσεων, όπως η εμφάνιση του χρόνου, η δυνατότητα ρύθμισης της ώρας και η υποστήριξη επιπλέον λειτουργιών.

Επιπλέον, θα παρουσιαστούν τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του ψηφιακού ρολογιού. Αυτά περιλαμβάνουν τη χρήση του περιβάλλοντος σχεδίασης Quartus II της Intel για τη σύνθεση και την προσομοίωση του κώδικα, καθώς και την αναπτυξιακή πλακέτα DE1-SoC. Η διαδικασία ανάπτυξης περιλαμβάνει την περιγραφή των λειτουργικών μονάδων του συστήματος, την υλοποίηση τους σε VHDL και την ενσωμάτωσή τους σε ένα ενιαίο σύστημα.

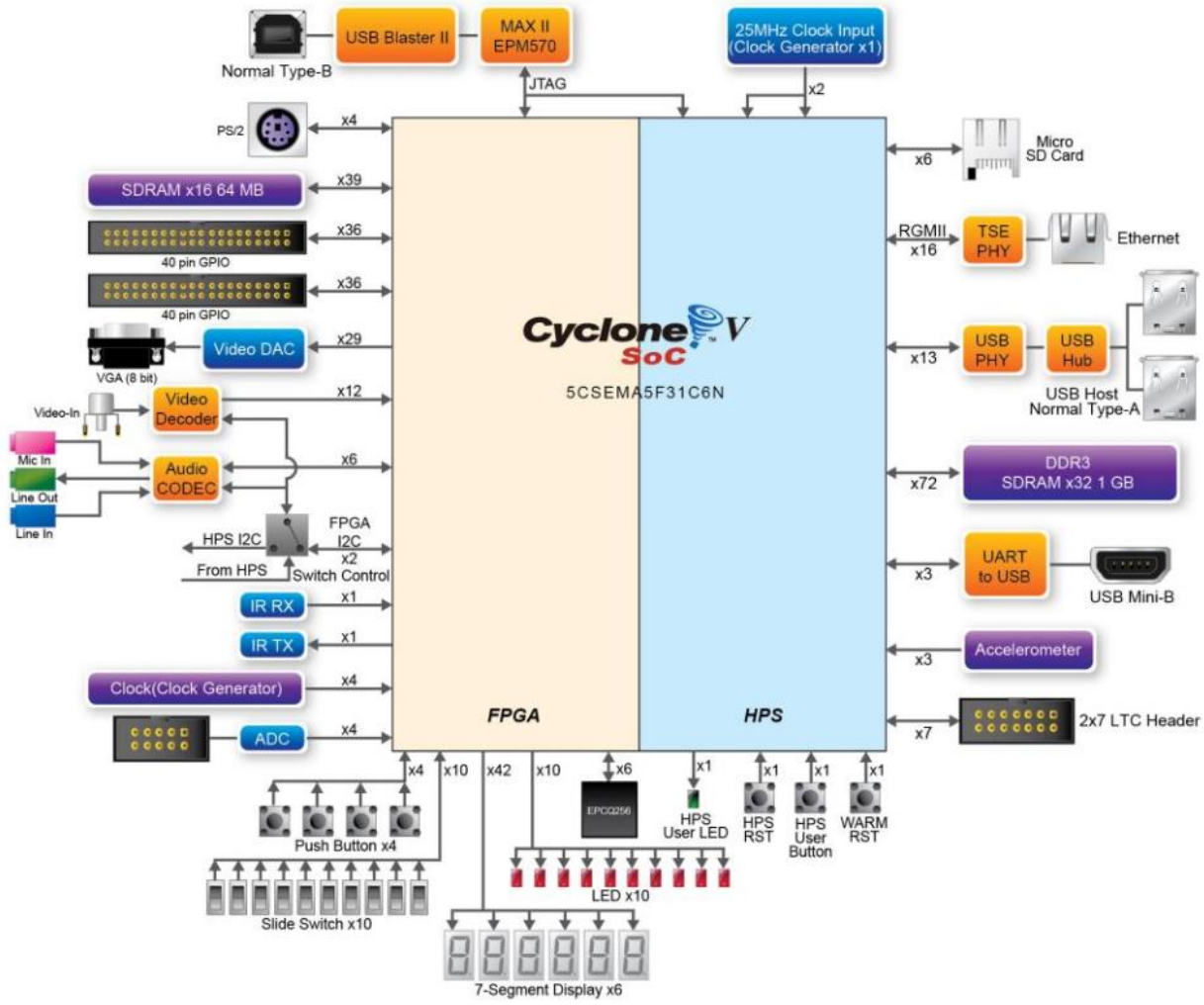
Στόχος του κεφαλαίου είναι να προσφέρει μια ολοκληρωμένη εικόνα της διαδικασίας ανάπτυξης του ψηφιακού ρολογιού, από τον ορισμό των απαιτήσεων έως την τελική υλοποίηση. Μέσα από αυτή την ανάλυση, θα γίνει σαφές το πλαίσιο λειτουργίας του συστήματος, οι προκλήσεις που αντιμετωπίστηκαν και οι λύσεις που υιοθετήθηκαν για την επίτευξη μιας λειτουργικής και αξιόπιστης υλοποίησης.

4.1 Αναπτυξιακή πλακέτα DE1-SoC

Η αναπτυξιακή πλακέτα DE1-SoC είναι μια ευέλικτη πλατφόρμα σχεδιασμού για την ανάπτυξη και τον πειραματισμό με συστήματα SoC (System on Chip), στο Σχήμα 4.1 απεικονίζεται το μπλοκ διάγραμμά της. Κεντρικό χαρακτηριστικό της DE1-SoC είναι το ολοκληρωμένο FPGA 5CSEMA5F31C6 από τη σειρά Cyclone V της Intel, το οποίο ενσωματώνει και έναν διπύρηνο επεξεργαστή ARM Cortex-A9. Αυτός ο συνδυασμός προσφέρει τη δυνατότητα ανάπτυξης συστημάτων που συνδυάζουν την προγραμματισμένη λογική του FPGA με τη δύναμη του επεξεργαστή, επιτρέποντας την υλοποίηση τόσο hardware όσο και software εφαρμογών. Το ολοκληρωμένο 5CSEMA5F31C6 διαθέτει 85.000 λογικά στοιχεία, 457 επαφές εισόδου/εξόδου γενικού σκοπού, 6 PLL, 4.450 kb ενσωματωμένη μνήμη και 87 μπλοκ DSP [12].

Ενδεικτικά η αναπτυξιακή πλακέτα περιλαμβάνει [13]:

- Altera Cyclone® V SE 5CSEMA5F31C6N
- Συσκευή αποθήκευσης μνήμης σειριακής διαμόρφωσης EPCQ256
- USB Blaster II για προγραμματισμό και έλεγχο από τον χρήστη
- 64MB μνήμη SDRAM
- 4 πλήκτρα push-buttons
- 10 συρόμενοι διακόπτες
- 10 κόκκινα LED
- 6 ενδείκτες 7 τομέων
- Γεννήτρια ρολογιού που παράγει 4 ρολόγια συχνότητας 50MHz
- Audio 24-bits CODEC με line-in, line-out και μικρόφωνο



Σχήμα 4.1: Μπλοκ διάγραμμα αναπτυξιακής πλακέτας DE1-SoC [13].

- Έξοδο VGA με μετατροπέα A/D
- TV Decoder (NTSC/PAL/SECAM) με είσοδο TV
- Θύρα διασύνδεσης ποντικιού/πληκτρολογίου τύπου PS/2
- Πομπό και δέκτη IR
- 2 θύρες επέκτασης 40 επαφών
- Μετατροπέα A/D

Επίσης περιλαμβάνει πληθώρα άλλων εισόδων/εξόδων για τον επεξεργαστή ARM Cortex-A9.

4.2 Λογισμικό Quartus II

Για την ανάπτυξη και υλοποίηση του ψηφιακού ρολογιού, χρησιμοποιήθηκε το περιβάλλον σχεδίασης Quartus II v.13.1.0 της Intel. Το Quartus II είναι ένα εργαλείο που επιτρέπει τη σύνθεση, προσομοίωση και υλοποίηση κώδικα VHDL σε συσκευές FPGA. Μέσω αυτού του περιβάλλοντος, πραγματοποιήθηκε η σύνθεση του κώδικα και η παραγωγή του αρχείου bitstream, το οποίο φορτώθηκε στο FPGA της πλακέτας DE1-SoC. Το Quartus II χρησιμοποιήθηκε επίσης για την προσομοίωση και τον έλεγχο της ορθότητας του σχεδιασμού, εξασφαλίζοντας τη σωστή λειτουργία του συστήματος σύμφωνα με τις προδιαγραφές.

Στο περιβάλλον του Quartus II προσφέρεται η δυνατότητα προσομοίωσης του κυκλώματος μέσω του University Program VWF (Vector Waveform File), ένα εργαλείο που επιτρέπει την ανάλυση της λειτουργίας του σχεδίου σε διάφορες συνθήκες. Το VWF επιτρέπει στον χρήστη να δημιουργήσει και

να τροφοδοτήσει σήματα στις εισόδους του συστήματος, προσομοιώνοντας έτσι την πραγματική λειτουργία του κυκλώματος. Μέσω της γραφικής αναπαράστασης των σημάτων σε κυματομορφές, ο χρήστης μπορεί να παρακολουθήσει την απόκριση του συστήματος, να διαπιστώσει τη σωστή χρονική συμπεριφορά και να εντοπίσει πιθανά σφάλματα ή αποκλίσεις.

Για την επιτυχή υλοποίηση ενός ψηφιακού κυκλώματος στο FPGA, είναι απαραίτητος ο σαφής ορισμός των εισόδων και εξόδων (I/O) του συστήματος, καθώς και η εφαρμογή κατάλληλων περιορισμών (constraints). Οι είσοδοι και οι έξοδοι του συστήματος ορίζονται μέσω του Pin Planer σε γραφικό περιβάλλον, όπου αντιστοιχίζονται οι φυσικές συνδέσεις του FPGA με τις εισόδους/εξόδους του σχεδίου. Εναλλακτικά, ο ορισμός των εισόδων/εξόδων μπορεί να γίνει μέσω του Assignment Editor. Επιπλέον, το εργαλείο TimeQuest Timing Analyzer του Quartus II παρέχει τη δυνατότητα ανάλυσης του χρονισμού του κυκλώματος καθώς και την εφαρμογή περιορισμών συγχρονισμού (timing constraints) μέσω ενός αρχείου .sdc, εξασφαλίζοντας τη σωστή λειτουργία του συστήματος σε πραγματικό χρόνο.

Το Quartus II περιλαμβάνει επίσης το εργαλείο MegaWizard Plug-In Manager, το οποίο επιτρέπει στους χρήστες να ενσωματώσουν εύκολα προ-σχεδιασμένα IP cores ή IP cores που έχουν δημιουργήσει οι ίδιοι. Το MegaWizard διευκολύνει τη διαδικασία δημιουργίας και ενσωμάτωσης υποσυστημάτων, όπως μονάδες επεξεργασίας, μνήμες, μονάδες επικοινωνίας και άλλες λογικές μονάδες, χωρίς να απαιτείται η κατασκευή του κώδικα από την αρχή. Ο MegaWizard δημιουργεί αυτόματα τον απαραίτητο κώδικα VHDL για την υλοποίηση αυτών των στοιχείων στο FPGA, εξοικονομώντας χρόνο και μειώνοντας την πολυπλοκότητα της ανάπτυξης.

Τέλος, το Quartus II παρέχει επίσης τη δυνατότητα προβολής του σχηματικού διαγράμματος RTL (Register Transfer Level Schematic), το οποίο αναπαριστά γραφικά την υλοποίηση του σχεδιασμού σε επίπεδο καταχωρητών και λογικών πυλών. επιτρέποντας την ανάλυση και την επαλήθευση της σύνδεσης και λειτουργίας των επιμέρους ψηφιακών κυκλωμάτων. Αυτή η δυνατότητα είναι χρήσιμη για τον εντοπισμό πιθανών σφαλμάτων ή αδυναμιών στη σχεδίαση, καθώς και για την καλύτερη κατανόηση της ροής των δεδομένων και των συνδέσεων μεταξύ των διάφορων μονάδων του συστήματος.

4.3 Ανάλυση των Απαιτήσεων και των Λειτουργιών του Ψηφιακού Ρολογιού

Το ψηφιακό ρολόι που σχεδιάστηκε και υλοποιήθηκε διαθέτει μια σειρά από βασικές και προηγμένες λειτουργίες, προκειμένου να προσφέρει ένα ευέλικτο και πλήρως λειτουργικό σύστημα. Η σχεδίαση του βασίζεται σε προγραμματιζόμενη λογική και αξιοποιεί τη γλώσσα VHDL για την περιγραφή και την υλοποίηση των επιμέρους λειτουργικών μονάδων. Παρακάτω περιγράφονται αναλυτικά οι βασικές λειτουργίες και οι απαιτήσεις που καθορίστηκαν για το σύστημα.

Ψηφιακό Ρολόι

Η κύρια λειτουργία του συστήματος είναι η εμφάνιση της τρέχουσας ώρας σε μορφή ώρα:λεπτά:δευτερόλεπτα, με δυνατότητα επιλογής μεταξύ 12ωρης και 24ωρης ένδειξης σε ψηφιακή μορφή σε ενδείκτες 7 τομέων. Παράλληλα, εμφανίζει την ημερομηνία σε μορφή ημέρα/μήνας/έτος.

Ρύθμιση Ώρας και Ημερομηνίας

Το σύστημα παρέχει τη δυνατότητα χειροκίνητης ρύθμισης της ώρας και της ημερομηνίας. Ο χρήστης μπορεί να ρυθμίσει τις τιμές της ώρας και των λεπτών καθώς και την ημερομηνία, προκειμένου να συγχρονίσει το ρολόι σε οποιαδήποτε χρονική στιγμή. Η διαδικασία ρύθμισης γίνεται μέσω πλήκτρων που επιτρέπουν την εύκολη πλοήγηση και την αποθήκευση των νέων τιμών.

Λειτουργίας αφύπνισης

Το σύστημα περιλαμβάνει λειτουργία αφύπνισης, η οποία επιτρέπει στον χρήστη να ορίσει μια συγκεκριμένη ώρα ειδοποίησης. Όταν η ώρα του ρολογιού ταυτιστεί με την ώρα αφύπνισης, το σύστημα ενεργοποιεί μια οπτική ειδοποίηση μέσω των LED. Η ενεργοποίηση και απενεργοποίηση της ειδοποίησης γίνεται, ανά πάσα στιγμή, μέσω συρόμενου διακόπτη, ενώ η ρύθμιση της ώρας γίνεται, όπως και για το ρολόι μέσω πλήκτρων.

Χρονόμετρο

Μια επιπλέον λειτουργία του ψηφιακού ρολογιού είναι η χρήση του ως χρονόμετρο. Ο χρήστης μπορεί να ξεκινήσει, να διακόψει ή να μηδενίσει τη μέτρηση του χρόνου, επιτρέποντας έτσι τη μέτρηση χρονικών διαστημάτων με ακρίβεια. Αρχικά στους ενδείκτες 7 τομέων εμφανίζονται εκατοστά του δευτερολέπτου, δευτερόλεπτα, λεπτά, ενώ όταν ο χρόνος υπερβεί τη μία ώρα εμφανίζονται δευτερόλεπτα, λεπτά και ώρες.

Αντίστροφη Μέτρηση

Η λειτουργία της αντίστροφης μέτρησης (countdown timer) επιτρέπει στον χρήστη να ορίσει, μέσω πλήκτρων, μια χρονική διάρκεια και να ξεκινήσει την αντίστροφη μέτρηση μέχρι το μηδέν. Επίσης υπάρχει η δυνατότητα παύσης και μηδενισμού του χρόνου μέτρησης.

Νυχτερινή Λειτουργία - Λειτουργία Εξοικονόμησης Ενέργειας

Το ψηφιακό ρολόι υποστηρίζει λειτουργία νυχτερινής προβολής, όπου η φωτεινότητα της οθόνης μειώνεται αυτόματα τις βραδινές ώρες. Για τη βελτίωση της απόδοσης και τη μείωση της κατανάλωσης ενέργειας, το ψηφιακό ρολόι διαθέτει λειτουργία εξοικονόμησης ενέργειας η οποία ενεργοποιείται από τον χρήστη. Σε αυτήν τη λειτουργία η οθόνη απενεργοποιείται και εμφανίζει την ώρα ανά ένα λεπτό για δέκα δευτερόλεπτα.

4.4 Δομή και Αρχιτεκτονική του Κώδικα

Η υλοποίηση του συστήματος βασίστηκε σε ιεραρχική δομή κώδικα. Κάθε λειτουργική μονάδα αναπτύχθηκε σε ξεχωριστό component, υλοποιώντας μια συγκεκριμένη λειτουργία, και στη συνέχεια ενώθηκε σε ένα top-level module, το οποίο συντονίζει τη λειτουργία του συστήματος. Αυτή η προσέγγιση επιτρέπει την ευκολότερη κατανόηση, ανάπτυξη και συντήρηση του κώδικα, καθώς κάθε μονάδα είναι αυτόνομη και εστιάζει σε μια συγκεκριμένη λειτουργία. Επιπλέον, η ιεραρχική δομή καθιστά το σύστημα πιο επεκτάσιμο, διευκολύνοντας την προσθήκη νέων λειτουργιών ή την τροποποίηση των υπαρχουσών, χωρίς να επηρεάζεται η συνολική αρχιτεκτονική.

Η ιεραρχική δομή του κώδικα περιλαμβάνει διάφορα components, τα οποία αντιστοιχούν στις βασικές λειτουργίες του ψηφιακού ρολογιού, όπως η εμφάνιση της ώρας και της ημερομηνίας, η ρύθμιση του χρόνου, τη λειτουργία αφύπνισης, το χρονόμετρο, η αντίστροφη μέτρηση ή η επιλογή μεταξύ 12ωρης ή 24ωρης μορφής της ώρας. Κάθε component σχεδιάστηκε με βάση τις απαιτήσεις του συστήματος και στη συνέχεια ενσωματώθηκε στο top-level module, όπου διαχειρίζεται η επικοινωνία και ο συντονισμός μεταξύ των μονάδων και των εισόδων και εξόδων του συστήματος.

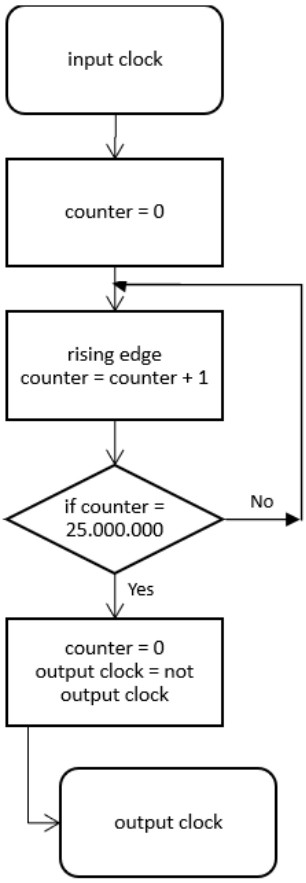
Σε αυτό το κεφάλαιο, θα περιγραφούν αναλυτικά οι βασικές λειτουργικές μονάδες του συστήματος, η αρχιτεκτονική του κώδικα και ο τρόπος με τον οποίο συνδέονται μεταξύ τους για να επιτευχθεί η ολοκληρωμένη λειτουργία του ψηφιακού ρολογιού.

4.4.1 Διαιρέτης συχνότητας

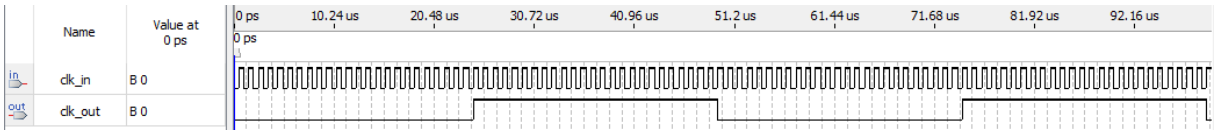
Η σωστή λειτουργία του ψηφιακού ρολογιού απαιτεί την παραγωγή ενός παλμού ακριβείας ανά δευτερόλεπτο (1Hz), ο οποίος θα χρησιμοποιείται για την προσαύξηση του μετρητή των δευτερολέπτων. Δεδομένου ότι η αναπτυξιακή πλακέτα DE1-SoC διαθέτει ενσωματωμένο ρολόι με συχνότητα 50MHz, η δημιουργία του απαιτούμενου 1Hz σήματος απαιτεί τη διαίρεση της συχνότητας του κύριου ρολογιού.

Για τον σκοπό αυτό, αναπτύχθηκε ένα component το οποίο δέχεται ως είσοδο το ρολόι των 50MHz, ένας μετρητής (counter) αυξάνει την τιμή του κατά ένα κάθε φορά που λαμβάνει έναν παλμό από το ρολόι. Όταν ο μετρητής μετρήσει 25.000.000 παλμούς, το component παράγει στην έξοδό του έναν παλμό. Όταν ο μετρητής φτάσει ξανά το ίδιο όριο των 25.000.000 παλμών, η έξοδος του component επιστρέφει στην τιμή ‘0’, και η διαδικασία επαναλαμβάνεται, δημιουργώντας με ακρίβεια τον απαιτούμενο παλμό ανά δευτερόλεπτο. Το διάγραμμα ροής του διαιρέτη συχνότητας απεικονίζεται στο Σχήμα 4.2.

Το component που δημιουργήσαμε για τη διαίρεση συχνότητας επιτρέπει την ακριβή μέτρηση του χρόνου, ώστε το ψηφιακό ρολόι να λειτουργεί με την απαιτούμενη ακρίβεια και να μπορεί να μετρά τα δευτερόλεπτα με συνέπεια. Χρησιμοποιώντας αυτόν τον παλμό 1Hz, το ψηφιακό ρολόι μπορεί να ανανεώνει την ένδειξή του κάθε δευτερόλεπτο, παρέχοντας μία αξιόπιστη και ακριβή μέτρηση του χρόνου, η οποία είναι απαραίτητη για την καλή λειτουργία του.



Σχήμα 4.2: Διάγραμμα ροής διαιρέτη συχνότητας 1Hz.



Σχήμα 4.3: Προσομοίωση διαιρέτη συχνότητας.

Στο Σχήμα 4.3 εμφανίζονται τα αποτελέσματα της προσομοίωσης που πραγματοποιήθηκε στο Quartus II. Αξίζει να σημειωθεί ότι ο μέγιστος χρόνος προσομοίωσης στο Quartus είναι 100 μsec. Δεδομένου ότι η είσοδος του component λαμβάνει σήμα 50MHz και η έξοδός του παράγει έναν παλμό ανά δευτερόλεπτο, η διαφορά στη συχνότητα είναι τόσο μεγάλη που δεν θα ήταν δυνατό να παρατηρήσουμε κάποια αλλαγή στην έξοδο κατά τη διάρκεια της προσομοίωσης. Για τον λόγο αυτό, προκειμένου να είναι εφικτή η οπτική επιβεβαίωση της λειτουργίας του κυκλώματος, προσαρμόσαμε τον μετρητή ώστε να μετρά μόνο μέχρι το 25 αντί για 25.000.000. Με αυτόν τον τρόπο, η έξοδος αλλάζει κατάσταση πολύ πιο συχνά, επιτρέποντας την επαλήθευση της σωστής λειτουργίας του διαιρέτη συχνότητας κατά την προσομοίωση στο Quartus II.

Εκτός από τον διαιρέτη συχνότητας που παράγει τον παλμό 1Hz και τον ονομάσαμε clk1hz, δημιουργήθηκαν τρεις ακόμα διαιρέτες συχνότητας, που χρησιμοποιούνται για άλλες λειτουργίες του συστήματος, όπως μέτρηση εκατοστών του δευτερολέπτου για την ακριβέστερη λειτουργία του χρονομέτρου ή μέτρηση της διάρκειας πατήματος των πλήκτρων, ώστε να είναι δυνατή η αναγνώριση σύντομων και παρατεταμένων πατημάτων Αυτοί οι διαιρέτες προσαρμόζουν τη συχνότητα του ρολογιού ανάλογα με τις απαιτήσεις κάθε λειτουργίας.

Όλοι οι διαιρέτες υλοποιήθηκαν με παρόμοια αρχιτεκτονική, χρησιμοποιώντας μετρητές που καταγράφουν έναν συγκεκριμένο αριθμό παλμών από το ρολόι των 50MHz πριν αλλάξουν την έξοδό τους. Ανάλογα με την επιθυμητή έξοδο, καθορίστηκε διαφορετικός διαιρέτης για κάθε περίπτωση, ώστε να επιτευχθεί η απαιτούμενη χρονική ακρίβεια σε κάθε επιμέρους λειτουργία του ψηφιακού ρολογιού.

4.4.2 Ενδείκτες 7 τομέων

Οι ενδείκτες 7 τομέων (7-segment displays) αποτελούν την κύρια έξοδο του συστήματος και χρησιμοποιούνται για την απεικόνιση της τρέχουσας ώρας και ημερομηνίας, καθώς και άλλων πληροφοριών, όπως η ρύθμιση της ώρας, η λειτουργία του χρονομέτρου και η ένδειξη ειδοποιήσεων. Κάθε ενδεικτής 7 τομέων αποτελείται από εφτά LED, τα οποία μπορούν να ενεργοποιούνται σε διαφορετικούς συνδυασμούς για να εμφανίσουν δεκαδικά ψηφία (0-9) και ορισμένους χαρακτήρες (π.χ. A, b, C, d, E, F). Η σωστή διαχείριση και έλεγχος αυτών των ενδεικτών είναι απαραίτητη για την ακριβή και σαφή εμφάνιση των πληροφοριών.

Στο πλαίσιο αυτής της εργασίας, χρησιμοποιήθηκαν έξι ενδείκτες 7 τομέων για την αλληλεπίδραση με τον χρήστη. Κάθε ενδεικτής ελέγχεται μέσω ενός αποκωδικοποιητή, ο οποίος μετατρέπει τις τετραψήφιες δυαδικές τιμές, που αντιστοιχούν στα ψηφία, σε σήματα που καθορίζουν ποια τμήματα του ενδεικτή θα ανάψουν και ποια όχι. Η σχεδίαση και η υλοποίηση του αποκωδικοποιητή για τους ενδείκτες 7 τομέων αποτέλεσε ένα βασικό βήμα για την ολοκλήρωση του συστήματος, καθώς εξασφάλισε την ορθή λειτουργία της εμφάνισης των πληροφοριών.

Για αρχή θα πρέπει να αναφέρουμε ότι για τη μέτρηση του χρόνου, όπως θα δούμε σε επόμενες ενότητες χρησιμοποιήσαμε μετρητές BCD (Binary Coded Decimal), οι οποίοι είναι ειδικοί ψηφιακοί μετρητές που μετρούν σε δεκαδική μορφή, αλλά χρησιμοποιώντας δυαδική αναπαράσταση.

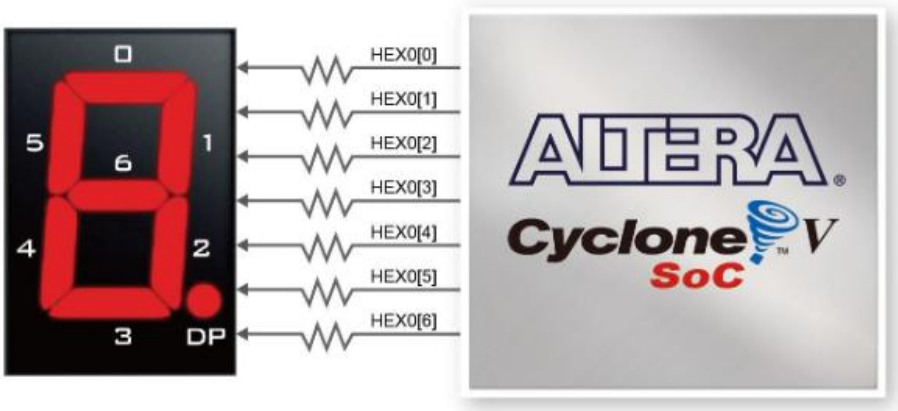
Πίνακας 4.1: Πίνακας μετατροπής από BCD σε 7-segment code.

Δεκαδικό ψηφίο	*BCD Code*	$7-segment\ code$
0	0000	1000000
1	0001	1111001
2	0010	0100100
3	0011	0110000
4	0100	0011001
5	0101	0010010
6	0110	0000010
7	0111	1111000
8	1000	0000000
9	1001	0011000

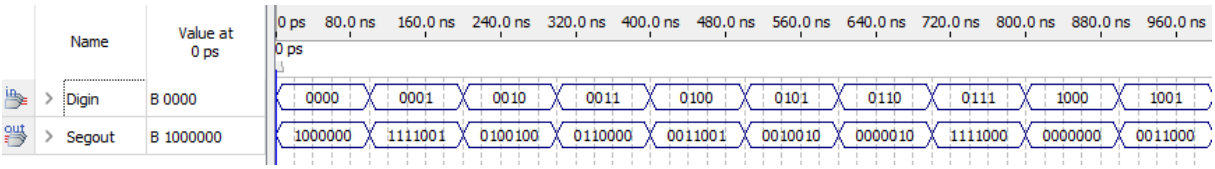
Όπως φαίνεται και στον Πίνακας 4.1 στους μετρητές BCD κάθε ψηφίο ενός δεκαδικού αριθμού (0-9) αναπαρίσταται με τέσσερα δυαδικά ψηφία (bits).

Η βασική λειτουργία ενός BCD μετρητή είναι να μετρά δεκαδικές τιμές σε μορφή που είναι εύκολα συμβατή με συσκευές απεικόνισης, όπως οι ενδείκτες εφτά τομέων. Για παράδειγμα, ένας μετρητής BCD μετρά από 0000 (0) έως 1001 (9) και έπειτα επιστρέφει στο 0000 αντί να συνεχίσει στο 1010 (10 σε δυαδικό). Αυτό επιτρέπει την απευθείας χρήση των εξόδων του για εμφάνιση σε ψηφιακές οθόνες καθώς κάθε δεκαδικό ψηφίο μπορεί να απεικονιστεί απευθείας χωρίς την ανάγκη πολύπλοκων μετατροπών.

Η είσοδος του αποκωδικοποιητή λαμβάνει τον τετραψήφιο κώδικα BCD και στην έξοδο παρέχει έναν επταψήφιο κωδικό, σύμφωνα με τον Πίνακας 4.1. Σε αυτόν τον κώδικο, το αριστερότερο bit αντιστοιχεί στο MSB (Most Significant Bit), ενώ το δεξιότερο στο LSB (Least Significant Bit). Κάθε bit του κωδικού αυτού καθορίζει την κατάσταση ενός από τα LED του ενδεικτή 7 τομέων, όπως φαίνεται στο Σχήμα 4.4. Αξίζει να σημειωθεί ότι η αναπτυξιακή πλακέτα DE1-SoC διαθέτει ενδείκτες κοινής ανόδου, κάτι που σημαίνει ότι για να ενεργοποιηθεί ένα LED, το αντίστοιχο bit του κώδικα 7-segment πρέπει να είναι 0. Για παράδειγμα, ο δεκαδικός αριθμός 8, που αντιστοιχεί στον BCD κωδικό 1000, έχει τον κωδικό 7-segment 0000000, όπου τα LED ενεργοποιούνται σύμφωνα με αυτή τη διάταξη.



Σχήμα 4.4: Ενδείκτης 7 τομέων της αναπτυξιακής πλακέτας DE1-Soc [13].



Σχήμα 4.5: Προσομοίωση αποκωδικοποιητή.

Στο Σχήμα 4.5 παρουσιάζονται τα αποτελέσματα της προσομοίωσης του αποκωδικοποιητή που πραγματοποιήθηκε στο Quartus II, όπως είναι εμφανές τα αποτελέσματα είναι αυτά που προβλέπονται σύμφωνα με τον Πίνακας 2.1.

4.4.3 Υλοποίηση Μετρητή Ψηφιακού Ρολογιού

Αφού υλοποιήθηκαν ο διαιρέτης συχνότητας clk1hz και ο αποκωδικοποιητής για την οδήγηση των ενδεικτών 7 τομέων, το επόμενο βήμα στη σχεδίαση του ψηφιακού ρολογιού είναι η ανάπτυξη του μετρητή. Ο διαιρέτης συχνότητας είναι απαραίτητος, καθώς παρέχει τον χρονισμό του 1Hz, ο οποίος καθορίζει τον ρυθμό αλλαγής των ενδείξεων του ρολογιού. Παράλληλα, ο αποκωδικοποιητής επιτρέπει την ορθή απεικόνιση των ψηφίων στην οθόνη.

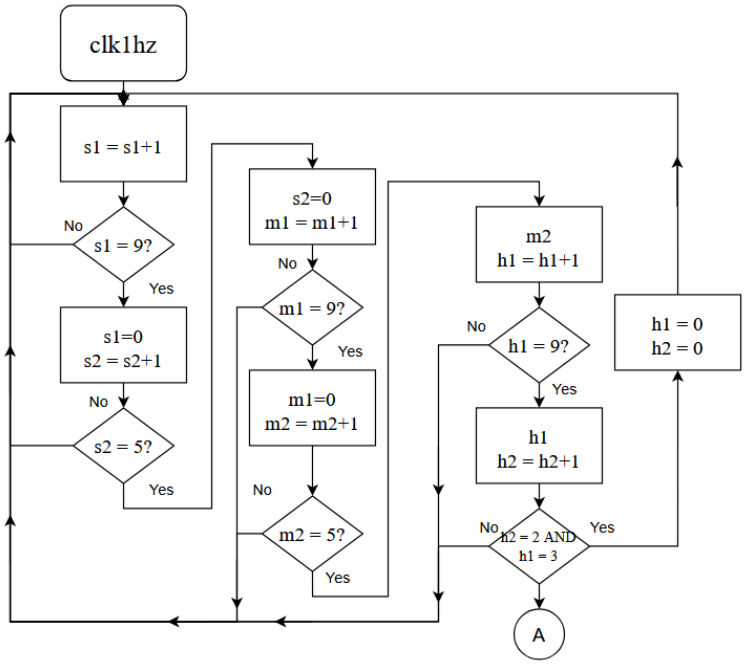
Με αυτά τα δύο βασικά υποσυστήματα σε λειτουργία, μπορούμε πλέον να προχωρήσουμε στην υλοποίηση του μετρητή, ο οποίος είναι υπεύθυνος για την ακριβή μέτρηση του χρόνου, συμπεριλαμβανομένων των δευτερολέπτων, των λεπτών και της ώρας, καθώς και της ημερομηνίας. Η σωστή λειτουργία του μετρητή είναι απαραίτητη για την ομαλή λειτουργία του ψηφιακού ρολογιού.

Για τον σκοπό αυτό, δημιουργήσαμε ένα ξεχωριστό component, το οποίο λαμβάνει στην είσοδό του παλμούς από το clk1hz. Όπως αναφέραμε και στην προηγούμενη ενότητα, η υλοποίηση του μετρητή του ψηφιακού ρολογιού έγινε με μετρητές BCD. Συγκεκριμένα, σχεδιάσαμε έξι μετρητές BCD, έναν για κάθε ψηφίο των μονάδων του χρόνου (δευτερόλεπτα, λεπτά, ώρες). Κάθε μετρητής είναι υπεύθυνος για την ακριβή μέτρηση και ενημέρωση της αντίστοιχης χρονικής μονάδας, εξασφαλίζοντας τη σωστή λειτουργία του συστήματος.

Η λειτουργία του μετρητή βασίζεται στη μετάβαση των μονάδων κάθε χρονικής μονάδας στις δεκάδες και στη συνέχεια στην επόμενη μονάδα χρόνου. Πιο συγκεκριμένα, ο πρώτος μετρητής, που μετράει τις μονάδες των δευτερολέπτων, αυξάνεται κατά ένα σε κάθε ανοδική ακμή των παλμών του clk1hz μέχρι να φτάσει στο "1001" (9 σε δεκαδική μορφή), στον επόμενο παλμό μηδενίζεται και αυξάνει κατά μία μονάδα ο δεύτερος μετρητής, των δεκάδων των δευτερολέπτων. Όταν ο μετρητής των δεκάδων των δευτερολέπτων φτάσει στο "0101" (5 σε δεκαδική μορφή), μηδενίζεται και αυξάνει ο μετρητής των μονάδων των λεπτών.

Η ίδια διαδικασία ακολουθείται για τα λεπτά και τις ώρες. Όταν ο μετρητής των μονάδων των λεπτών φτάσει στο "1001", μηδενίζεται και αυξάνει ο μετρητής των δεκάδων των λεπτών. Στη συνέχεια, όταν ο μετρητής των δεκάδων των λεπτών φτάσει στο "0101", μηδενίζεται και αυξάνει τον μετρητή των μονάδων των ωρών.

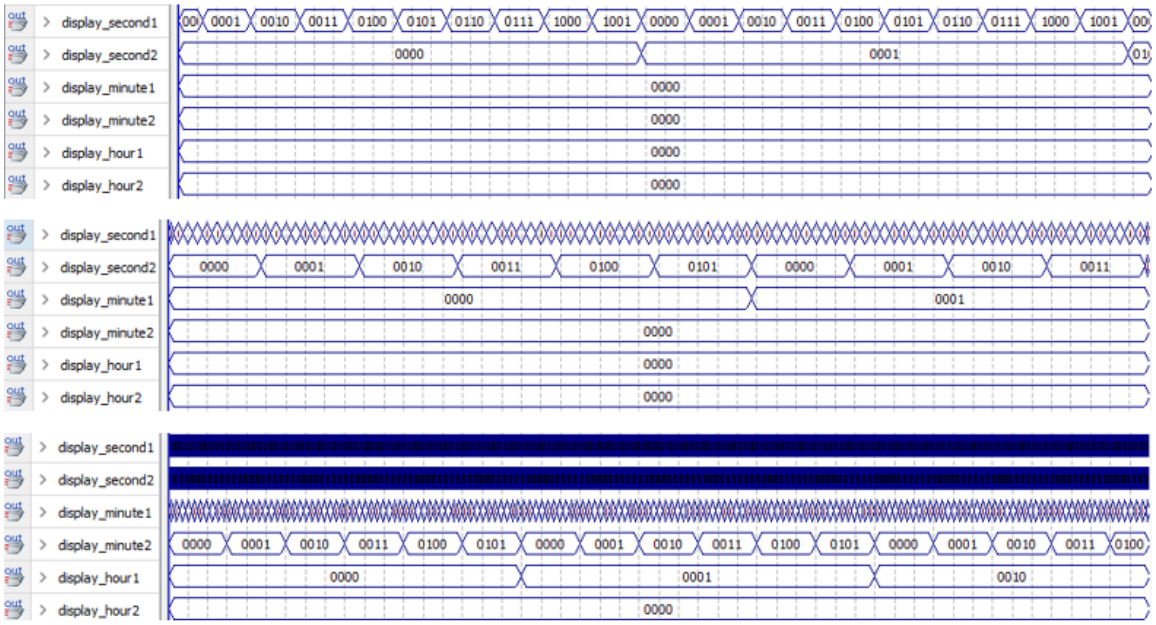
Ο μηχανισμός αλλαγής της ώρας ακολουθεί την ίδια λογική. Όταν ο μετρητής των μονάδων των ωρών φτάσει στο "1001", μηδενίζεται και αυξάνει τον μετρητή των δεκάδων των ωρών. Τέλος, όταν το ρολόι φτάσει στην ένδειξη "23:59:59", στον επόμενο παλμό όλοι οι μετρητές (δευτερολέπτων, λεπτών και ωρών) μηδενίζονται, επαναφέροντας την ώρα στο ‘00:00:00’ και ξεκινώντας έναν νέο κύκλο μέτρησης.



Σχήμα 4.6: Διάγραμμα ροής μετρητή ψηφιακού ρολογιού.

Στο Σχήμα 4.6 απεικονίζεται το διάγραμμα ροής του μετρητή του ψηφιακού ρολογιού. Όπου s1 και s2 είναι οι μετρητές των μονάδων και των δεκάδων των δευτερολέπτων, m1 και m2 είναι οι μετρητές των μονάδων και των δεκάδων των λεπτών και αντίστοιχα h1 και h2 είναι οι μετρητές των μονάδων και των δεκάδων των ωρών.

Θα πρέπει να αναφέρουμε ότι, στην VHDL, οι τιμές των σημάτων ανανεώνονται στο τέλος του κύκλου ρολογιού. Για παράδειγμα, στον έλεγχο για το αν το s1 είναι ίσο με το 9, η σύγκριση γίνεται με την τιμή που είχε το s1 στην αρχή της διαδικασίας κατά την ανίχνευση του παλμού, δηλαδή πριν την αύξηση s1=s1+1. Αυτό συμβαίνει γιατί οι αλλαγές των σημάτων προγραμματίζονται για την επόμενη χρονική στιγμή του ρολογιού.



Σχήμα 4.7: Προσομοίωση μετρητή ψηφιακού ρολογιού.

Στο Σχήμα 4.7 απεικονίζονται τα αποτελέσματα από τις διάφορες φάσεις της προσομοίωσης του μετρητή ψηφιακού ρολογιού που πραγματοποιήθηκε στο Quartus II και επιβεβαίωσε την ορθή λειτουργία του μετρητή, καθώς και τη σωστή διαχείριση των μεταβάσεων μεταξύ δευτερολέπτων, λεπτών και ωρών.

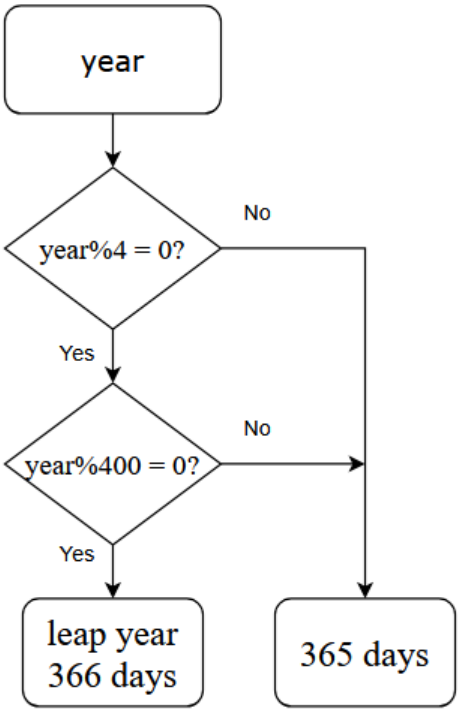
Στο πρώτο μέρος της προσομοίωσης, παρατηρείται η αύξηση της τιμής του μετρητή BCD που αντιστοιχεί στις μονάδες των δευτερολέπτων (second1). Όταν ο μετρητής φτάσει στην τιμή ‘1001’ (9 σε δεκαδική μορφή), στον επόμενο παλμό του ρολογιού μηδενίζεται και αυξάνεται η τιμή του μετρητή που αντιστοιχεί στις δεκάδες των δευτερολέπτων.

Στο δεύτερο μέρος της προσομοίωσης, παρατηρείται η μετάβαση σε νέο λεπτό. Όταν οι μετρητές δευτερολέπτων φτάσουν στο μέγιστο όριο (‘1001’ ο second1 και ‘0101’ ο second2), οι μετρητές δευτερολέπτων επιστρέφουν στις τιμές ‘0000’, ενώ ο μετρητής μονάδων των λεπτών (minute1) αυξάνεται κατά ένα.

Στο τρίτο μέρος της προσομοίωσης, παρατηρείται η μετάβαση σε νέα ώρα. Όταν οι μετρητές των λεπτών φτάσουν στο μέγιστο όριο, επιστρέφουν στις τιμές ‘0000’, ενώ ο μετρητής μονάδων των ωρών (hour1) αυξάνεται κατά ένα.

4.4.4 Υλοποίηση Μετρητή Ημερομηνίας

Αφού υλοποιήθηκε η μετρητής χρόνου σε επίπεδο ωρών, λεπτών και δευτερολέπτων, το επόμενο στάδιο στη σχεδίαση του συστήματος είναι η ανάπτυξη του μετρητή για τη διαχείριση της ημερομηνίας. Ο μετρητής αυτός είναι υπεύθυνος για την καταγραφή και την ορθή μετάβαση μεταξύ ημερών, μηνών και ετών, λαμβάνοντας υπόψη τη διαφοροποίηση των ημερών ανά μήνα καθώς και τον κανόνα των δίσεκτων ετών.

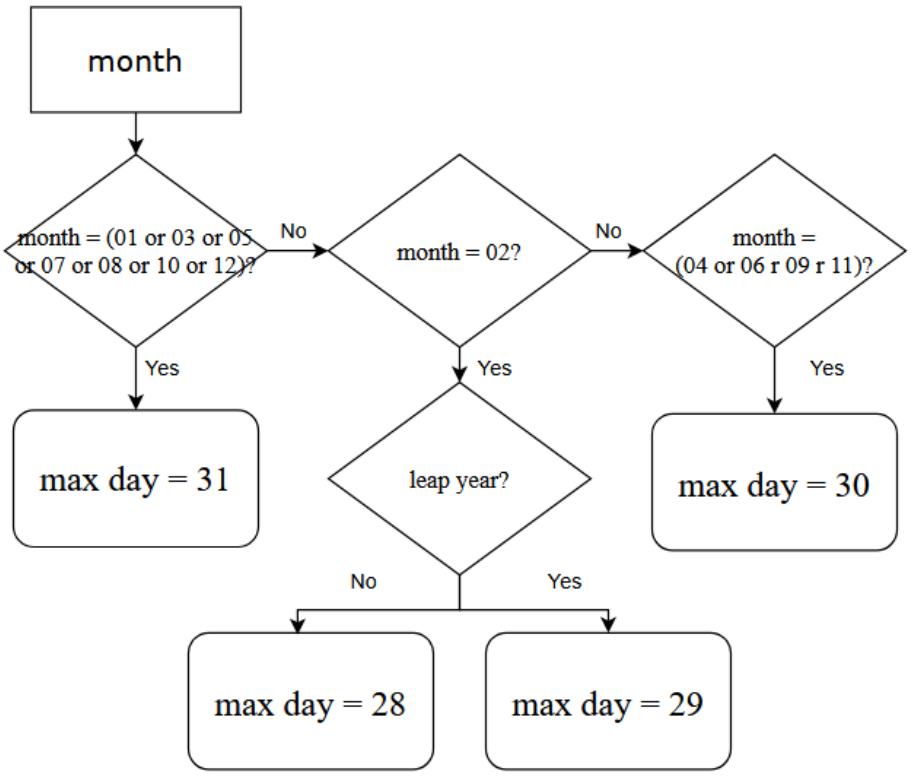


Σχήμα 4.8: Λογικό διάγραμμα υπολογισμού δίσεκτων ετών.

Σύμφωνα με τον κανόνα υπολογισμού των δίσεκτων ετών του Γρηγοριανού ημερολογίου, όπως απεικονίζεται στο λογικό διάγραμμα στο Σχήμα 4.8, ένα έτος είναι δίσεκτο αν διαιρείται με το 4, ενώ από τα επαιώνια έτη (1600, 1700, 1800, 1900, 2000, 2100 κ.ο.κ) δίσεκτα είναι μόνο τα έτη που διαιρούνται με το 400. Οπότε, σύμφωνα με τον κανόνα, το έτος 2000 είναι δίσεκτο, ενώ το έτος 2100 δεν είναι. Αυτός ο κανόνας βοηθά στη διατήρηση της ακρίβειας του ημερολογίου σε σχέση με την τροχιά της Γης γύρω από τον Ήλιο.

Η λειτουργία του μετρητή βασίζεται στην προσαρμογή της αρίθμησης των ημερών σύμφωνα με το μήκος του εκάστοτε μήνα (28, 29, 30 ή 31 ημέρες), ενώ παράλληλα διαχειρίζεται την εναλλαγή των μηνών και την προσαύξηση του έτους όταν ο μετρητής φτάσει την τιμή στις 31 Δεκεμβρίου. Για την αποδοτική διαχείριση αυτών των μεταβάσεων, απαιτείται ένας μηχανισμός που λαμβάνει υπόψη τόσο τις σταθερές διάρκειες των μηνών όσο και τον έλεγχο του δίσεκτου έτους για τον Φεβρουάριο.

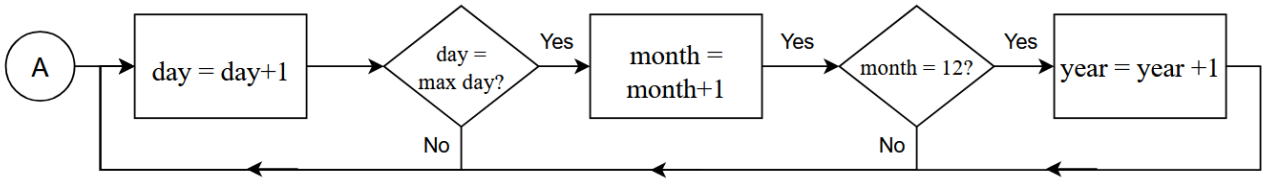
Η υλοποίηση του μετρητή ημερομηνίας πραγματοποιήθηκε με μια ιεραρχική δομή μετρητών BCD, αντίστοιχη με αυτήν που χρησιμοποιείται στη μέτρηση της ώρας. Κάθε μονάδα ημερολογιακής μέτρησης ακολουθεί τη λογική της αλληλουχίας των αριθμητικών μεταβάσεων, ώστε να διασφαλίζεται η ορθή λειτουργία του ψηφιακού ρολογιού σε πραγματικό χρόνο.



Σχήμα 4.9: Λογικό διάγραμμα υπολογισμού συνολικού αριθμού ημερών.

Για την υλοποίηση του μετρητή ημερολογίου, δημιουργήσαμε δύο μεταβλητές, maxday1 και maxday2, οι οποίες αντιστοιχούν στις μονάδες και τις δεκάδες του συνολικού αριθμού ημερών κάθε μήνα σε κώδικα BCD. Όπως φαίνεται και στο απλοποιημένο λογικό διάγραμμα του Σχήμα 4.9, οι μεταβλητές αυτές καθορίζουν το ανώτατο όριο της μέτρησης των ημερών, το οποίο εξαρτάται από τον εκάστοτε μήνα και το αν το έτος είναι δίσεκτο ή όχι. Μέσω αυτών, το σύστημα αναγνωρίζει πότε πρέπει να μηδενιστούν οι μετρητές των ημερών και να αυξηθεί αντίστοιχα ο μετρητής του μήνα, διασφαλίζοντας έτσι την ορθή λειτουργία του ημερολογίου.

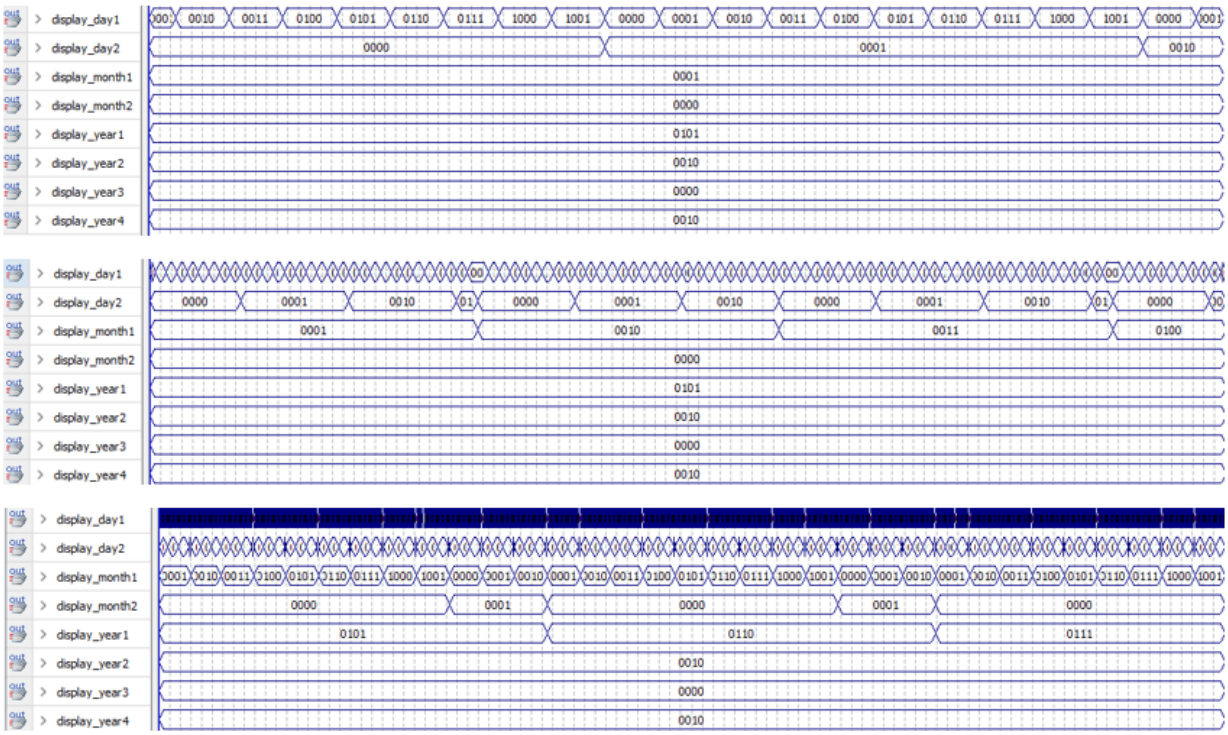
Το τελικό μπλοκ διάγραμμα του μετρητή ημερομηνίας απεικονίζεται στο Σχήμα 4.10. όπου φαίνεται η συνολική αρχιτεκτονική της διαχείρισης της ημερομηνίας στο ψηφιακό ρολόι. Ο μετρητής ημερολογίου συνεργάζεται άμεσα με τον μετρητή της ώρας. Συγκεκριμένα, κάθε φορά που ο μετρητής της ώρας ολοκληρώνει έναν πλήρη κύκλο (ώρα 00.00.00), στέλνει ένα σήμα (το σήμα 'A') το οποίο σηματοδοτεί την αλλαγή της ημέρας και ενεργοποιεί τον μετρητή ημερών.



Σχήμα 4.10: Μπλοκ διάγραμμα μετρητή ημερομηνίας.

Στο Σχήμα 4.11 απεικονίζονται τα αποτελέσματα από διάφορες φάσεις της προσομοίωσης του μετρητή ημερολογίου. Η προσομοίωση που πραγματοποιήθηκε στο περιβάλλον Quartus II επιβεβαίωσε την ορθή λειτουργία του μετρητή, καθώς και τη σωστή διαχείριση των μεταβάσεων μεταξύ ημερών, μηνών και ετών.

Στο πρώτο μέρος της προσομοίωσης, παρατηρείται η αύξηση της τιμής του μετρητή BCD που αντιστοιχεί στις μονάδες της ημέρας (day1). Όταν ο μετρητής φτάσει στην τιμή '1001' (9 σε δεκαδική μορφή), στο επόμενο σήμα που δέχεται από την έξοδο του μετρητή του ψηφιακού ρολογιού, μηδενίζεται και αυξάνεται η τιμή του μετρητή που αντιστοιχεί στις δεκάδες της ημέρας (day2).



Σχήμα 4.11: Προσομοίωση μετρητή ημερολογίου.

Στο δεύτερο μέρος της προσομοίωσης, παρατηρείται η μετάβαση σε νέο μήνα. Όταν οι μετρητές ημέρας φτάσουν στο μέγιστο όριο (στη συγκεκριμένη περίπτωση 31 μέρες για τον Ιανουάριο και τον

Μάρτιο και 28 μέρες για τον Φεβρουάριο), οι μετρητές ημέρας επιστρέφουν στις τιμές ‘0001’, ο μετρητής μονάδων (day1), και ‘0000’ ο μετρητής δεκάδων (day2). Ενώ ο μετρητής μονάδων του μήνα (month1) αυξάνεται κατά ένα. Παρομοίως, στο τρίτο μέρος της προσομοίωσης, παρατηρείται η μετάβαση σε νέο έτος.

Για τη διαχείριση των διαφορετικών ημερομηνιών και των αλλαγών των μηνών, έχουν υλοποιηθεί κατάλληλοι μηχανισμοί ελέγχου, ώστε να αποφεύγονται λανθασμένες μεταβάσεις. Για παράδειγμα, το σύστημα εξασφαλίζει ότι ο Φεβρουάριος έχει 28 ημέρες σε κανονικά έτη και 29 ημέρες σε δίσεκτα έτη, ενώ οι υπόλοιποι μήνες τηρούν τις καθορισμένες διάρκειες των 30 ή 31 ημερών.

Ο μετρητής ώρας, που αναλύσαμε στο Κεφάλαιο 4.4.3 και ο μετρητής ημερολογίου, που αναλύσαμε στο Κεφάλαιο 4.4.4, αποτελούν τις δύο βασικές λειτουργίες του component που ονομάσαμε ClockDigitCounter. Επιπλέον, όταν το σύστημα προβάλλει την ώρα στους ενδείκτες επτά τομέων, κατά τη διάρκεια της αλλαγής της ώρας, στα πρώτα πέντε δευτερόλεπτα εμφανίζεται η ημερομηνία και ο μήνας, ενώ κατά τα επόμενα πέντε δευτερόλεπτα εμφανίζεται το έτος, έπειτα εμφανίζεται και πάλι η ώρα. Αυτή η εναλλαγή εξασφαλίζει την ταυτόχρονη προβολή της τρέχουσας ώρας και της ημερομηνίας στους ενδεικτές, επιτρέποντας στον χρήστη να παρακολουθεί εύκολα και τις δύο παραμέτρους του χρόνου.

Ο κώδικας για τους μετρητές ώρας και ημερομηνίας, υπολογισμού συνολικών ημερών του μήνα και υπολογισμού δίσεκτων ετών παρατίθεται στο ΠΑΡΑΡΤΗΜΑ Ι.

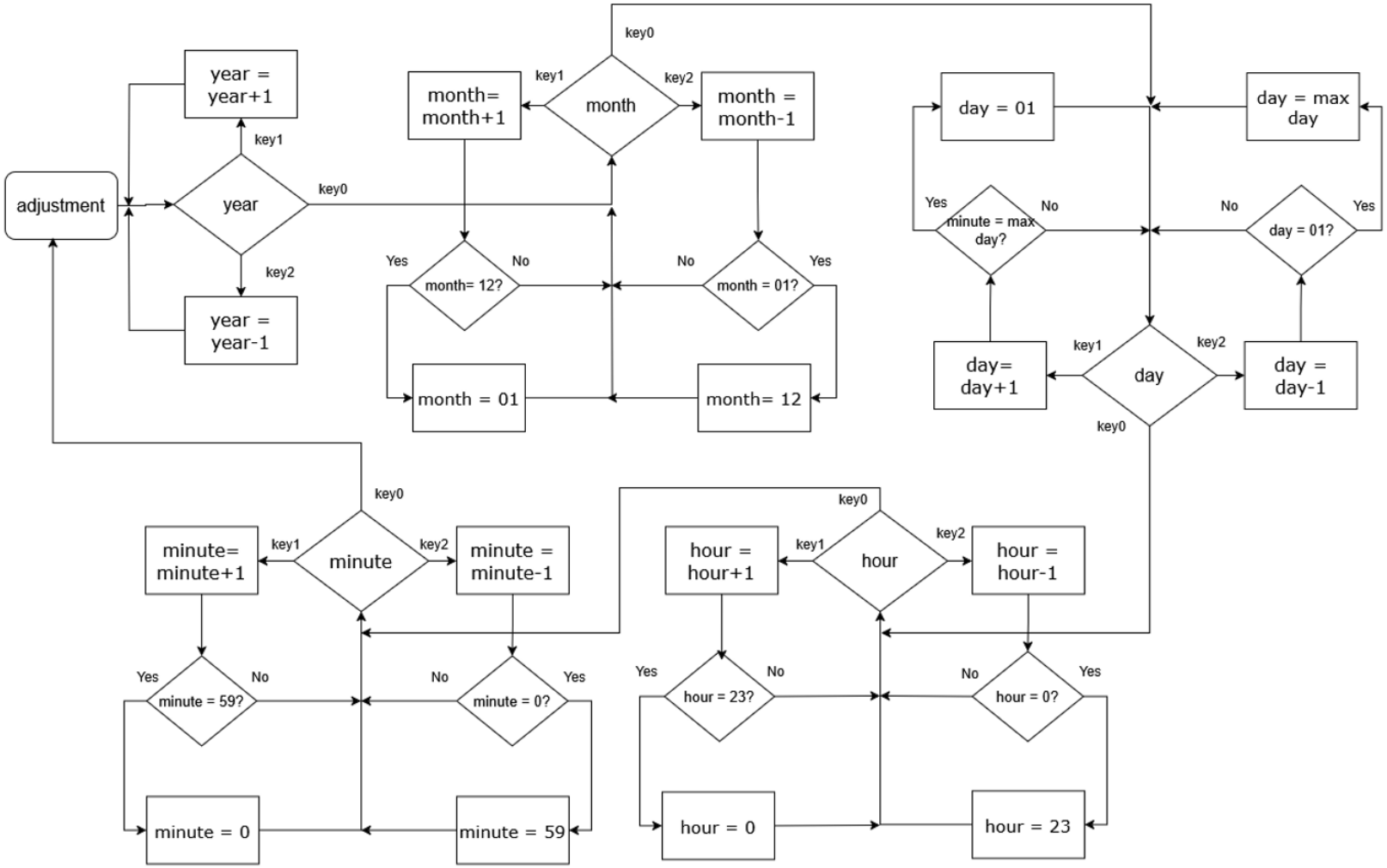
4.4.5 Ρύθμιση ημερομηνίας και ώρας

Η δυνατότητα ρύθμισης της ώρας και της ημερομηνίας αποτελεί βασικό χαρακτηριστικό για τη λειτουργία του ψηφιακού ρολογιού, καθώς εξασφαλίζει την ακρίβεια και την αξιοπιστία του συστήματος. Μετά την υλοποίηση των βασικών λειτουργιών του ρολογιού και του ημερολογίου, είναι απαραίτητη η ενσωμάτωση μηχανισμών που επιτρέπουν την εύκολη και ακριβή ρύθμιση των παραμέτρων χρόνου. Μέσω αυτού του μηχανισμού, το ρολόι και το ημερολόγιο μπορούν να λαμβάνουν τις επιθυμητές αρχικές τιμές ή να διορθώνονται σε περίπτωση αποκλίσεων, εξασφαλίζοντας έτσι την ακρίβεια και την αξιοπιστία του συστήματος.

Στην περίπτωση του συστήματος της παρούσας εργασίας, η διαδικασία αυτή περιλαμβάνει την προσαρμογή των ωρών, των λεπτών, καθώς και της ημερομηνίας, ενώ λαμβάνει υπόψη τη σωστή μετάβαση των ημερών, των μηνών και των ετών. Η διαδικασία ρύθμισης είναι σχεδιασμένη έτσι ώστε να είναι εύκολη και γρήγορη, επιτρέποντας στον χρήστη να εισάγει την ακριβή ώρα και ημερομηνία, είτε κατά την αρχική εκκίνηση του συστήματος, είτε όταν χρειάζεται να γίνει κάποια αλλαγή.

Η είσοδος στη διαδικασία ρύθμισης καθώς και η ίδια η διαδικασία ρύθμισης πραγματοποιούνται μέσω πλήκτρων. Κατά την είσοδο στη λειτουργία ρύθμισης, ο χρήστης ακολουθεί μια προκαθορισμένη και κυκλική ακολουθία βημάτων για την τροποποίηση της ώρας και της ημερομηνίας. Συγκεκριμένα, ο χρήστης προχωρά με τη σειρά, ρυθμίζοντας πρώτα το έτος, ακολουθούμενο από το μήνα, την ημέρα, την ώρα και τέλος τα λεπτά.

Κάθε βήμα παρέχει τη δυνατότητα τροποποίησης της αντίστοιχης μονάδας του χρόνου, είτε μέσω αύξησης είτε μέσω μείωσης της τιμής της. Η μετάβαση στο επόμενο βήμα, όπως και η καταχώρηση των νέων τιμών, πραγματοποιείται με το πάτημα ενός πλήκτρου, επιτρέποντας στον χρήστη να ολοκληρώσει την ρύθμιση του χρόνου και της ημερομηνίας με εύκολο και άμεσο τρόπο. Σημειώνεται ότι η ρύθμιση των δευτερολέπτων δεν κρίνεται απαραίτητη, και για τον λόγο αυτό δεν περιλαμβάνεται στη διαδικασία ρύθμισης.



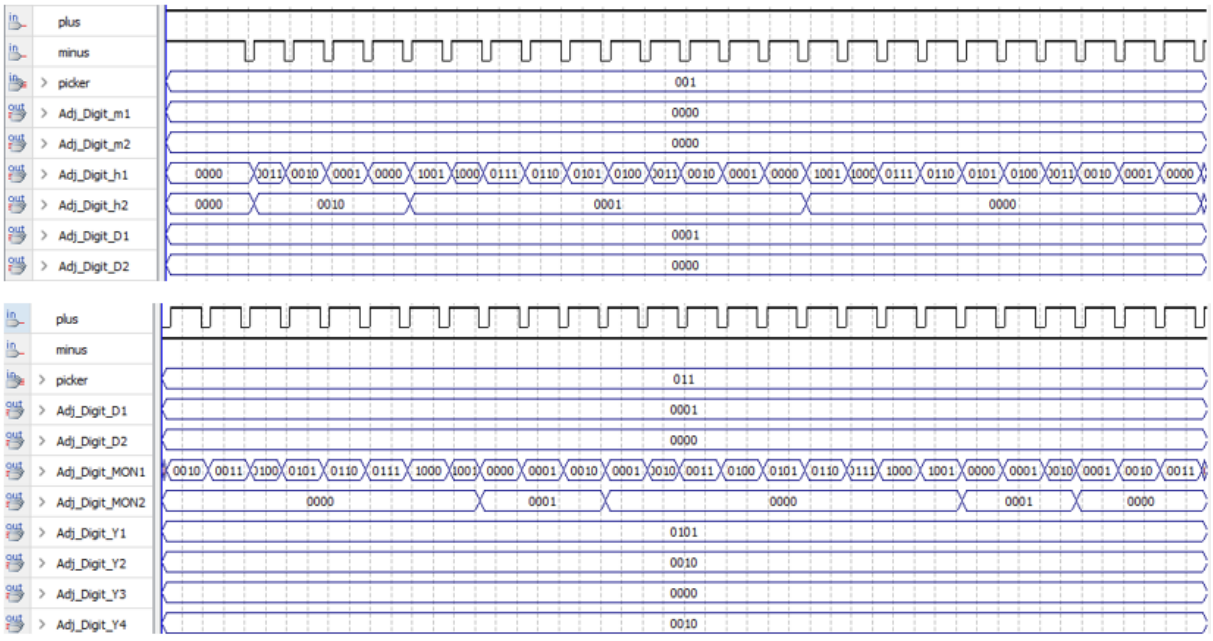
Σχήμα 4.12: Μπλοκ διάγραμμα διαδικασίας ρύθμισης ώρας και ημερομηνίας.

Η διαδικασία ρύθμισης της ώρας και της ημερομηνίας ακολουθεί την ίδια μεθοδολογία που εφαρμόζεται για τη λειτουργία του ρολογιού και του ημερολογίου, χρησιμοποιώντας μετρητές BCD για κάθε ψηφίο των αντίστοιχων μονάδων χρόνου. Η κύρια διαφορά, ωστόσο, έγκειται στο γεγονός ότι, κατά τη διάρκεια της ρύθμισης, οι μετρητές κάθε μονάδας είναι ανεξάρτητοι και δεν επηρεάζουν τις άλλες μονάδες.

Η παραπάνω διαδικασία ρύθμισης περιγράφεται αναλυτικά και απεικονίζεται στο μπλοκ διάγραμμα στο Σχήμα 4.12, το οποίο απεικονίζει τα βήματα που ακολουθεί ο χρήστης και την αλληλουχία των ενεργειών κατά τη διάρκεια της ρύθμισης της ώρας και της ημερομηνίας

Συγκεκριμένα, σε αντίθεση με τη λειτουργία του ρολογιού, όπου η ολοκλήρωση ενός κύκλου (π.χ., οι 60 λεπτά) οδηγεί στην αύξηση της επόμενης μονάδας, δηλαδή των ωρών, στη διαδικασία ρύθμισης οι μετρητές μηδενίζονται και ρυθμίζονται ανεξάρτητα, χωρίς να επηρεάζουν τις άλλες μονάδες. Έτσι, όταν ο χρήστης ρυθμίζει μία χρονική μονάδα, οι υπόλοιπες παραμένουν σταθερές και δεν επηρεάζονται.

Επιπλέον, η διαδικασία ρύθμισης κάθε χρονικής μονάδας μπορεί να πραγματοποιηθεί είτε μέσω αύξησης είτε μέσω μείωσης της τιμής της, ανάλογα με την εντολή του χρήστη, ο οποίος έχει τη δυνατότητα να τροποποιήσει κάθε μονάδα του χρόνου μέσω πλήκτρων. Ενώ η μετάβαση στην επόμενη μονάδα (π.χ., από το έτος στο μήνα ή από το μήνα στις ώρες) πραγματοποιείται μέσω άλλου πλήκτρου.



Σχήμα 4.13: Προσομοίωση διαδικασίας ρύθμισης ώρας και ημερομηνίας.

Στο Σχήμα 4.13 απεικονίζονται τα αποτελέσματα από τις διάφορες φάσεις της προσομοίωσης κατά τη διαδικασία ρύθμισης της ώρας και της ημερομηνίας. Η προσομοίωση πραγματοποιήθηκε στο περιβάλλον Quartus II και επιβεβαίωσε την ορθή λειτουργία του συστήματος.

Για αρχή να αναφέρουμε ότι μέσω του picker γίνεται η επιλογή μονάδας που θα ρυθμιστεί και ότι plus και minus είναι η ονομασία των πλήκτρων μέσω των οποίων ο χρήστης αυξάνει ή μειώνει τις τιμές.

Επίσης τα πλήκτρα της πλακέτας DE1-SoC κατά τη διάρκεια πίεσης, παράγουν λογικό 0, ενώ όταν δεν πιέζονται, το λογικό επίπεδο που παράγεται είναι 1.

Στο πρώτο μέρος της προσομοίωσης, ρυθμίζεται η ώρα με μείωση της τιμής, και παρατηρείται η συμπεριφορά των μετρητών Adj_Digit_h1 και Adj_Digit_h2, που αντιστοιχούν στις μονάδες και στις δεκάδες της ώρας, αντίστοιχα.

Συγκεκριμένα, όταν ο μετρητής Adj_Digit_h1 έχει την τιμή ‘0000’ και ταυτόχρονα ο Adj_Digit_h2 έχει επίσης την τιμή ‘0000’, τότε ο Adj_Digit_h1 μεταβαίνει στην τιμή ‘0011’, ενώ ο Adj_Digit_h2 λαμβάνει την τιμή ‘0010’, δηλαδή από ώρα 00:00 σε 23:00. Αντίστοιχα, όταν ο Adj_Digit_h1 βρίσκεται στην κατάσταση ‘0000’ και ο Adj_Digit_h2 έχει την τιμή ‘0010’, η επόμενη κατάσταση που λαμβάνουν είναι ‘1001’ για τον Adj_Digit_h1 και ‘0001’ για τον Adj_Digit_h2, δηλαδή από ώρα 20:00 σε 19:00. Επίσης παρατηρείται ότι δεν επηρεάζονται οι τιμές των μετρητών για τα λεπτά (Adj_Digit_m1 και Adj_Digit_m2) και για τις μέρες (Adj_Digit_D1 και Adj_Digit_D2).

Αυτή η συμπεριφορά επιβεβαιώνει την ορθή λειτουργία του συστήματος, καθώς οι μεταβάσεις ανταποκρίνονται στις αναμενόμενες τιμές και διασφαλίζουν τη σωστή αλλαγή των ψηφίων κατά τη διαδικασία ρύθμισης της ώρας.

Στο δεύτερο μέρος της προσομοίωσης, ρυθμίζεται ο μήνας με αύξηση της τιμής, και παρατηρείται η συμπεριφορά των μετρητών Adj_Digit_MON1 και Adj_Digit_MON2, που αντιστοιχούν στις μονάδες και στις δεκάδες του μήνα, αντίστοιχα.

Και σε αυτή την περίπτωση, παρατηρείται η ομαλή αύξηση των μετρητών, επιβεβαιώνοντας τη σωστή λειτουργία του συστήματος την ομαλή αύξηση των μετρητών. Επιπλέον, όταν ο Adj_Digit_MON1 φτάσει στην τιμή ‘0010’ (2 σε δεκαδική μορφή) και ο Adj_Digit_MON2 έχει την τιμή ‘0001’, τότε ο Adj_Digit_MON1 μεταβαίνει στην τιμή ‘0001’ και ο Adj_Digit_MON2 μεταβαίνει στην τιμή ‘0000’, δηλαδή από τον μήνα 12 σε 01.

Επίσης, παρατηρείται ότι οι τιμές των μετρητών για τα έτη (Adj_Digit_Y1, Adj_Digit_Y2, Adj_Digit_Y3 και Adj_Digit_Y4) παραμένουν ανεπηρέαστες, γεγονός που επιβεβαιώνει την απομόνωση και την ανεξαρτησία των διαφόρων μηχανισμών ρύθμισης.

Η συνολική συμπεριφορά της προσομοίωσης επιβεβαιώνει την ορθή λειτουργία του συστήματος, καθώς οι μεταβάσεις ακολουθούν τις αναμενόμενες τιμές, παραμένοντας εντός των προκαθορισμένων ορίων. Επιπλέον, διασφαλίζεται η ορθή αλλαγή των ψηφίων της εκάστοτε μονάδας χρόνου, χωρίς να επηρεάζονται οι υπόλοιπες.

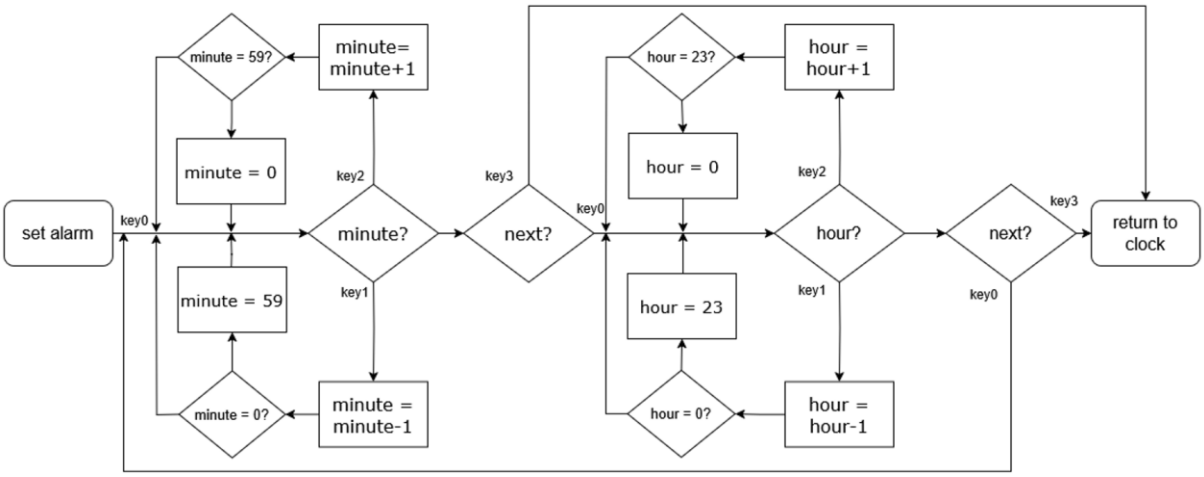
4.4.6 Λειτουργία αφύπνισης

Αφού ολοκληρώθηκε η υλοποίηση του ψηφιακού ρολογιού και του μηχανισμού ρύθμισης της ώρας, το επόμενο βήμα είναι η ανάπτυξη της λειτουργίας αφύπνισης. Η ενσωμάτωση μιας τέτοιας λειτουργίας αποτελεί ένα από τα βασικά χαρακτηριστικά ενός ψηφιακού ρολογιού, καθώς επιτρέπει την ειδοποίηση του χρήστη σε προκαθορισμένη ώρα. Ο μηχανισμός της αφύπνισης επιτρέπει στον χρήστη να ορίσει μια συγκεκριμένη ώρα ενεργοποίησης, κατά την οποία το σύστημα θα παράγει μια οπτική ειδοποίηση.

Για την υλοποίηση της λειτουργίας αφύπνισης, απαιτείται η δημιουργία μηχανισμών αποθήκευσης της ώρας ειδοποίησης, καθώς και η συνεχής σύγκρισή της με την τρέχουσα ώρα. Επιπλέον, ο μηχανισμός πρέπει να διαθέτει δυνατότητες ενεργοποίησης και απενεργοποίησης της αφύπνισης, ώστε ο χρήστης να μπορεί να ελέγχει τη λειτουργία του ανά πάσα στιγμή. Η βασική λειτουργία του συστήματος

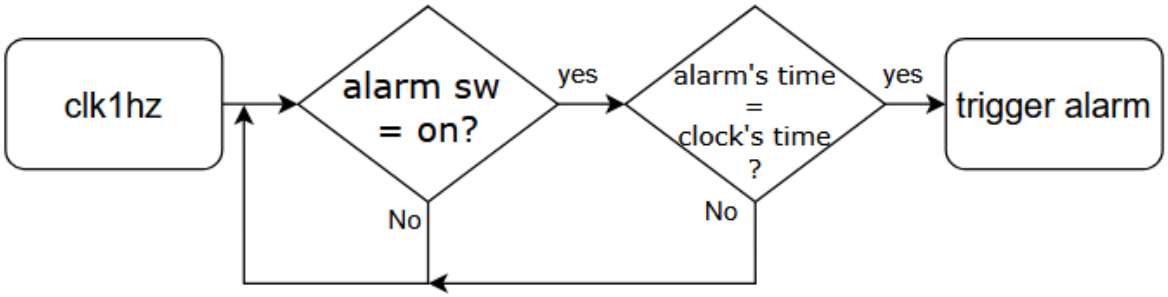
στηρίζεται στη συνεχή παρακολούθηση της τρέχουσας ώρας και στη σύγκρισή της με την ώρα που έχει προγραμματίσει ο χρήστης. Όταν οι δύο τιμές συμπίπτουν, ενεργοποιείται το σήμα ειδοποίησης.

Η διαδικασία ρύθμισης της αφύπνισης πραγματοποιείται μέσω των πλήκτρων του συστήματος, ακολουθώντας μια κυκλική ακολουθία επιλογών. Συγκεκριμένα, ο χρήστης έχει τη δυνατότητα να εισέλθει στη λειτουργία ρύθμισης μέσω παρατεταμένου πατήματος πλήκτρου και να ορίσει την επιθυμητή ώρα αφύπνισης, τροποποιώντας διαδοχικά τα λεπτά και τις ώρες. Η διαδικασία επιλογής των τιμών, η οποία παρουσιάζεται στο Σχήμα 4.14 είναι παρόμοια με τη διαδικασία ρύθμισης της ώρας και γίνεται είτε μέσω αύξησης είτε μέσω μείωσης της τιμής.



Σχήμα 4.14: Διάγραμμα ροής ρύθμισης αφύπνισης.

Η αποθήκευση της ώρας αφύπνισης πραγματοποιείται μέσω αντίστοιχων μετρητών BCD, όπως συμβαίνει και με το ψηφιακό ρολόι. Ωστόσο, σε αντίθεση με τη βασική λειτουργία του ρολογιού, η τιμή της αφύπνισης παραμένει σταθερή έως ότου μεταβληθεί από τον χρήστη, χωρίς να επηρεάζεται από τη φυσική ροή του χρόνου.



Σχήμα 4.15: Διάγραμμα ροής ενεργοποίησης λειτουργίας αφύπνισης.

Για την ενεργοποίηση της ειδοποίησης, το διάγραμμα ροής της οποίας παρουσιάζεται στο Σχήμα 4.15 το σύστημα συγκρίνει σε κάθε παλμό του clk1hz τις τιμές των μετρητών BCD της αφύπνισης με τις αντίστοιχες τιμές του ψηφιακού ρολογιού. Συγκεκριμένα, ελέγχεται αν οι τιμές των δεκάδων ωρών (Alarm_Digit_h2) και μονάδων ωρών (Alarm_Digit_h1) είναι ίδιες με τις αντίστοιχες τιμές του ρολογιού (Clock_Digit_h2 και Clock_Digit_h1), καθώς και αν οι τιμές των δεκάδων λεπτών

(Alarm_Digit_m2) και μονάδων λεπτών (Alarm_Digit_m1) ταυτίζονται με εκείνες του ρολογιού (Clock_Digit_m2 και Clock_Digit_m1). Όταν όλες οι παραπάνω συνθήκες ικανοποιούνται, η λειτουργία αφύπνισης ενεργοποιείται, παράγοντας την προβλεπόμενη ειδοποίηση.

Η ενεργοποίηση και απενεργοποίηση της αφύπνισης γίνεται χειροκίνητα από τον χρήστη μέσω συρόμενου διακόπτη. Η συνολική λειτουργία της λειτουργίας αφύπνισης είναι σχεδιασμένη ώστε να είναι απλή και αποτελεσματική, παρέχοντας στον χρήστη ευελιξία στη ρύθμιση και τη διαχείρισή του. Η αρχιτεκτονική του μηχανισμού διασφαλίζει ότι η ειδοποίηση θα ενεργοποιείται μόνο όταν είναι απαραίτητο, ενώ ταυτόχρονα επιτρέπει την προσαρμογή των ρυθμίσεων ανάλογα με τις ανάγκες του χρήστη.

Ο κώδικας για την ρύθμιση και ενεργοποίηση της λειτουργίας αφύπνισης παρατίθεται στο ΠΑΡΑΡΤΗΜΑ ΙΙ.

4.4.7 Επιλογή 12ωρης/24ωρης απεικόνισης

Η δυνατότητα ρύθμισης της απεικόνισης της ώρας μεταξύ 12ωρης και 24ωρης μορφής αποτελεί ένα σημαντικό χαρακτηριστικό στις σύγχρονες ψηφιακές εφαρμογές και συσκευές. Αυτή η λειτουργία επιτρέπει στους χρήστες να προσαρμόζουν την εμφάνιση της ώρας ανάλογα με τις προσωπικές τους προτιμήσεις ή τις τοπικές συμβάσεις.

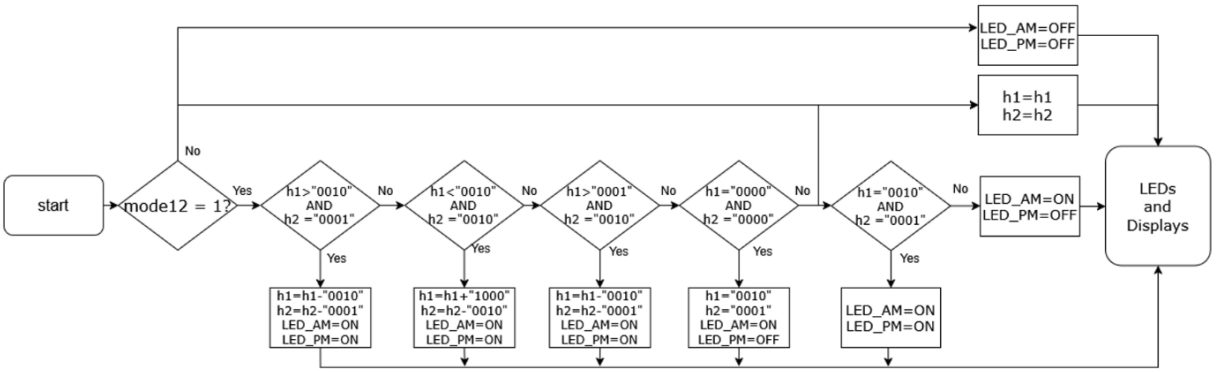
Η 12ωρη μορφή, η οποία χρησιμοποιεί τους δείκτες ‘π.μ.’ (προ μεσημβρίας) και ‘μ.μ.’ (μετά τη μεσημβρία), αποτελεί μέρος της καθημερινής ζωής για πολλούς ανθρώπους. Αντίθετα, η 24ωρη μορφή, που βασίζεται σε μια συνεχή αρίθμηση από το 00:00 έως το 23:59 προσφέρει μεγαλύτερη ακρίβεια και αποφεύγει τυχόν ασάφειες που μπορεί να προκύψουν από τη χρήση της 12ωρης μορφής, χρησιμοποιείται ευρέως σε επιστημονικές και τεχνικές εφαρμογές.

Στο ψηφιακό ρολόι που αναπτύχθηκε, ο χρήστης διαθέτει τη δυνατότητα επιλογής μεταξύ της 12ωρης και της 24ωρης μορφής εμφάνισης της ώρας, μέσω της χρήσης ενός πλήκτρου. Αυτή η λειτουργία είναι προσβάσιμη μόνο κατά τη διάρκεια της κανονικής λειτουργίας του ρολογιού, επιτρέποντας στον χρήστη να αλλάζει τη μορφή εμφάνισης της ώρας ανά πάσα στιγμή, ανάλογα με τις προτιμήσεις ή τις ανάγκες του. Η αλλαγή αυτή εφαρμόζεται επίσης στη ρύθμιση της ώρας και της λειτουργία αφύπνισης, διασφαλίζοντας ενιαία απεικόνιση σε όλες τις λειτουργίες της συσκευής.

Για την υλοποίηση της μετατροπής από 24ωρη σε 12ωρη μορφή, δημιουργήσαμε ένα component όπου στις εισόδους του δέχεται τις μονάδες και τις δεκάδες της ώρας του μετρητή του ψηφιακού ρολογιού, καθώς και την επιλογή του χρήστη για την επιθυμητή μορφή ώρας. Οι μονάδες και οι δεκάδες της ώρας είναι σε μορφή BCD. Οι δύο έξοδοι του component καταλήγουν στους αντίστοιχους ενδείκτες εφτά τομέων για την προβολή των δύο ψηφίων της ώρας. Επίσης, όταν χρειάζεται, ενεργοποιείται ένα LED ως ένδειξη ‘π.μ’ και δύο LED ως ένδειξη ‘μ.μ’.

Αξίζει να σημειωθεί ότι, εάν η ώρα ήταν αποθηκευμένη σε δεκαδική μορφή, η μετατροπή θα ήταν απλούστερη, καθώς για τις ώρες μεγαλύτερες του 12 η διαδικασία θα μπορούσε να πραγματοποιηθεί απλά με αφαίρεση του 12 (hour = hour - 12). Ωστόσο, δεδομένου ότι το σύστημα χρησιμοποιεί BCD ψηφία, η μετατροπή απαιτεί ειδικούς ελέγχους και προσαρμογές των δεκάδων και μονάδων της ώρας.

Στο Σχήμα 4.16 απεικονίζεται το διάγραμμα ροής της διαδικασίας μετατροπής της ώρας από 24ωρη σε 12ωρη μορφή. Αρχικά γίνεται έλεγχος της επιλογής του χρήστη. Εάν ο χρήστης έχει επιλέξει τη 24ωρη μορφή, δεν γίνεται καμία μετατροπή και η ώρα εμφανίζεται απευθείας όπως εισήχθη και τα δύο LED παραμένουν σβηστά. Εάν επιλεγεί η 12ωρη μορφή, η διαδικασία συνεχίζεται με τους παρακάτω ελέγχους και ενέργειες:



Σχήμα 4.16: Διάγραμμα ροής εμφάνισης 12ωρης ή 24ωρης μορφής.

Εάν η ώρα βρίσκεται μεταξύ 13:00 και 19:59 (δηλαδή, h1 > 0010 και h2 = 0001), τότε πραγματοποιείται η μετατροπή: h1 = h1 – 0010 και h2 = h2 – 0001. Ταυτόχρονα ανάβουν δύο LED για τη σήμανση ‘μ.μ.’.

Για τις ώρες μεταξύ 20:00 και 21:59 (δηλαδή, h1 < 0010 και h2 = 0010), η μετατροπή γίνεται ως εξής: h1 = h1 + 1000 και h2 = h2 – 0010. Όπως και πριν, ανάβουν δύο LED που υποδεικνύουν τη σήμανση ‘μ.μ.’.

Εάν η ώρα είναι μεταξύ 22:00 και 23:59 (δηλαδή, h1 > 0001 και h2 = 0010), η μετατροπή ακολουθεί τον ίδιο κανόνα με την πρώτη περίπτωση: h1 = h1 – 0010 και h2 = h2 – 0001. Επίσης ανάβουν δύο LED για τη σήμανση ‘μ.μ.’.

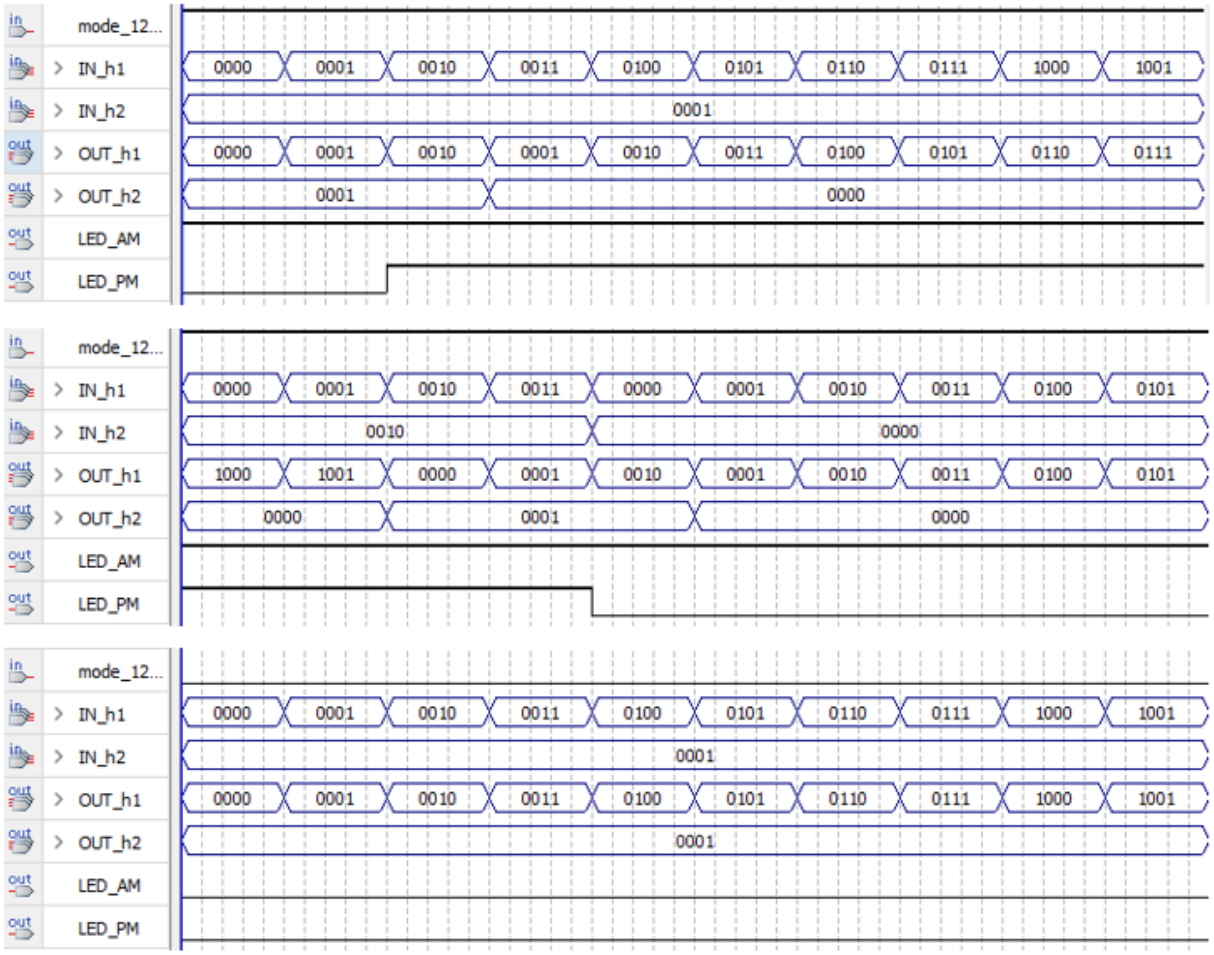
Στην περίπτωση που η ώρα είναι μεταξύ 00:00 και 00:59 (δηλαδή, h1 = 0000 και h2 = 0000), τότε η μετατροπή γίνεται ως εξής: h1 = 0010 και h2 = 0001. Σε αυτήν την περίπτωση, ανάβει μόνο ένα LED, υποδεικνύοντας ‘π.μ.’.

Τέλος, αν η ώρα είναι μεταξύ 01:00 και 12:59, δεν απαιτείται καμία αλλαγή στα BCD ψηφία. Για τις ώρες από 01:00 έως 11:59, παραμένει αναμμένο μόνο ένα LED για τη σήμανση ‘π.μ.’, ενώ για το χρονικό διάστημα 12:00 έως 12:59, ανάβουν δύο LED για την ένδειξη ‘μ.μ.’.

Στον Πίνακας 4.2 εμφανίζεται πιο συνοπτικά η λογική μετατροπής της ώρας από 24ωρη σε 12ωρη μορφή, καθώς και η αντίστοιχη ένδειξη LED.

Πίνακας 4.2: Μετατροπή από 24ωρη σε 12ωρη μορφή.

Ώρα (24ωρη)	Ώρα (12ωρη)	Μετατροπή BCD	Ένδειξη LED (π.μ/μ.μ)
00:00 – 00:59	12:00-12:59 π.μ.	h2=0001, h1=0010	1 LED (π.μ.)
01:00 – 11:59	01:00-11:59 π.μ.	Καμία αλλαγή	1 LED (π.μ.)
12:00 – 12:59	12:00-12:59 μ.μ.	Καμία αλλαγή	2 LED (μ.μ.)
13:00 – 19:59	01:00-07:59 μ.μ.	h2 = h2–0001, h1 = h1-0010	2 LED (μ.μ.)
20:00 – 21:59	08:00-09:59 μ.μ.	h2 = h2–0010, h1 = h1+1000	2 LED (μ.μ.)
22:00 – 23:59	10:00-11:59 μ.μ.	h2 = h2–0001, h1 = h1-0010	2 LED (μ.μ.)



Σχήμα 4.17:Προσομοίωση μετατροπής 24ωρης σε 12ωρης μορφή.

Στο Σχήμα 4.17 απεικονίζονται τρεις φάσεις από την προσομοίωση του συστήματος που υλοποιήθηκε στο VWF του Quartus II και περιλαμβάνει τη χρήση των εισόδων και εξόδων του component για την επεξεργασία της ώρας και της επιλογής του χρήστη για την 12ωρη ή 24ωρη μορφή. Συγκεκριμένα, οι είσοδοι και έξοδοι του component είναι οι εξής: οι είσοδοι IN_h1 και IN_h2 είναι οι μονάδες και δεκάδες της ώρας, η είσοδος mode_12 είναι η επιλογή του χρήστη για την μορφή ώρας (‘1’ για 12ωρη και ‘0’ για 24ωρη). Οι έξοδοι OUT_h1 και OUT_h2 είναι οι μονάδες και δεκάδες της ώρας μετά τη μετατροπή και οι έξοδοι LED_AM και LED_PM, τα LED για τις ενδείξεις ‘π.μ.’ (ανάβει το LED_AM) και ‘μ.μ’ (ανάβουν και τα δύο LED).

Στις πρώτες δύο φάσεις που η είσοδος mode_12 έχει τιμή ‘1’, οπότε ο χρήστης έχει επιλέξει 12ωρη μορφή, η μετατροπή γίνεται όπως παρουσιάστηκε πιο πάνω στο διάγραμμα ροής του Σχήμα 4.16 και στον Πίνακας 4.2, με τις αντίστοιχες ενέργειες και αλλαγές στις μονάδες και τις δεκάδες της ώρας, καθώς και την ενεργοποίηση των LED για την ένδειξη ‘π.μ.’ ή ‘μ.μ.’.

Στην τρίτη φάση, όταν το mode_12 είναι 0 (δηλαδή έχει επιλεγεί η 24ωρη μορφή), δεν πραγματοποιείται καμία μετατροπή στην ώρα. Η ώρα εμφανίζεται όπως εισάγεται στην 24ωρη μορφή χωρίς αλλαγές, και τα LED παραμένουν σβηστά.

Αυτή η διαδικασία εξασφαλίζει τη σωστή μετατροπή και εμφάνιση της ώρας σε 12ωρη μορφή, διατηρώντας παράλληλα την ακρίβεια και την αξιοπιστία του συστήματος. Η χρήση των LED για τη

σήμανση ‘π.μ.’ και ‘μ.μ.’ προσφέρει μια απλή και αποτελεσματική λύση για τη διαφοροποίηση των ωρών πριν και μετά το μεσημέρι.

Ο κώδικας επιλογής και μετατροπής της ώρας από 24ωρη σε 12ωρη μορφή παρατίθεται στο ΠΑΡΑΡΤΗΜΑ ΙΙΙ.

4.4.8 Χρονόμετρο

Η λειτουργία του χρονομέτρου αποτελεί ένα βασικό και απαραίτητο χαρακτηριστικό σε πολλές ψηφιακές συσκευές, προσφέροντας τη δυνατότητα μέτρησης του χρόνου με ακρίβεια και αξιοπιστία. Στο πλαίσιο της παρούσας εργασίας, η υλοποίηση του χρονομέτρου στο ψηφιακό ρολόι αποσκοπεί στην παροχή μιας πρόσθετης λειτουργικότητας που μπορεί να χρησιμοποιηθεί σε διάφορες εφαρμογές, όπως αθλητικές δραστηριότητες, βιομηχανικές εφαρμογές ή απλές καθημερινές ανάγκες.

Το χρονόμετρο επιτρέπει στον χρήστη να ξεκινά, να σταματά και να επαναφέρει τη μέτρηση του χρόνου, προσφέροντας ευελιξία και ευκολία στη διαχείριση του χρόνου. Σε αντίθεση με το ψηφιακό ρολόι, λειτουργεί ανεξάρτητα από την τρέχουσα ώρα και ελέγχεται πλήρως από τον χρήστη.

Η σχεδίασή του βασίζεται σε μετρητές BCD για την αναπαράσταση των μονάδων χρόνου, ενώ προσφέρει αυξημένη ακρίβεια καθώς μετρά όχι μόνο δευτερόλεπτα, λεπτά και ώρες, αλλά και εκατοστά του δευτερολέπτου. Για να επιτευχθεί αυτή η λειτουργία, δημιουργήθηκε ένα σήμα ρολογιού στα 100Hz, το οποίο χρησιμεύει ως η βάση χρόνου για την μέτρηση των εκατοστών του δευτερολέπτου. Οι μετρητές BCD λειτουργούν στο δεκαδικό σύστημα και αυξάνονται με βάση προκαθορισμένα όρια. Συγκεκριμένα, οι μετρητές των εκατοστών του δευτερολέπτου μετρούν από 00 έως 99, οι μετρητές των δευτερολέπτων και των λεπτών κυμαίνονται από 00 έως 59, ενώ οι μετρητές των ωρών από 00 έως 99.

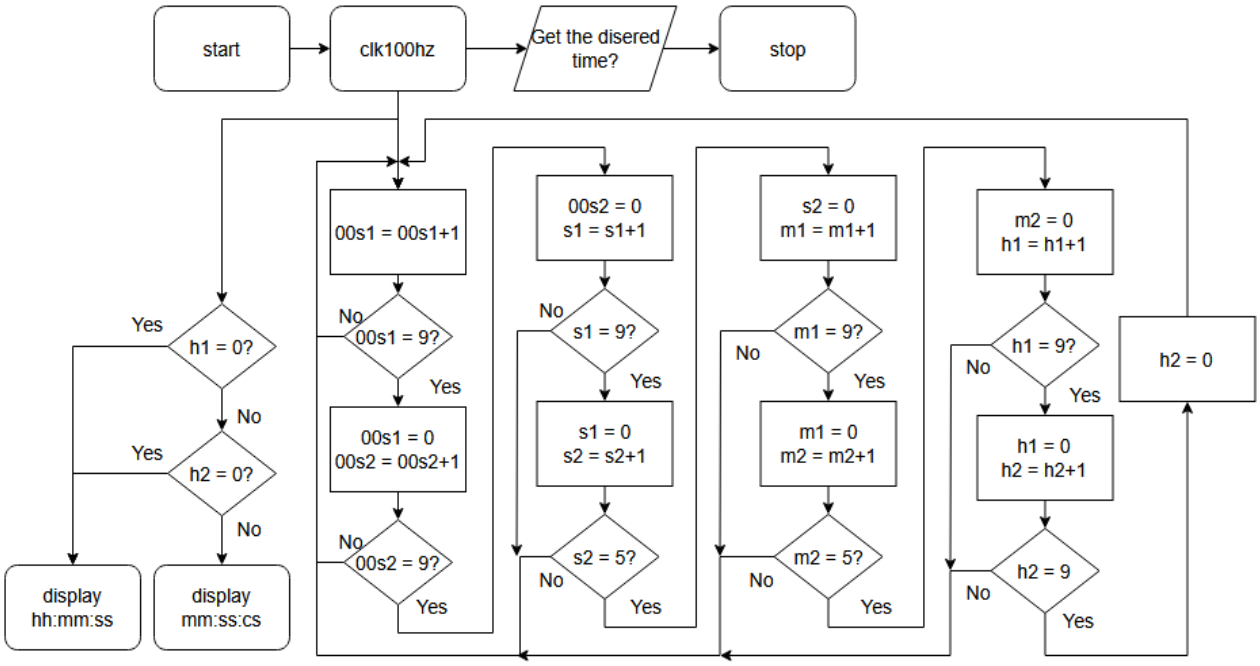
Το χρονόμετρο εμφανίζει τον χρόνο στους ενδείκτες με τη μορφή λλ:δδ:εε (λεπτά, δευτερόλεπτα, εκατοστά του δευτερολέπτου) για το πρώτο διάστημα της ώρας, ενώ μετά τη συμπλήρωση της μιας ώρας, η μορφή αλλάζει σε ωω:λλ:δδ (ώρες, λεπτά, δευτερόλεπτα). Αυτή η προσέγγιση εξασφαλίζει ότι ο χρήστης μπορεί να παρακολουθεί τον χρόνο με ακρίβεια, ανεξάρτητα από τη διάρκεια της μέτρησης.

Ο χρήστης μπορεί να ελέγχει το χρονόμετρο μέσω τριών βασικών λειτουργιών: start, stop και reset. Με το πάτημα του πλήκτρου start, το χρονόμετρο ξεκινά τη μέτρηση του χρόνου. Με το πάτημα του πλήκτρου stop, η μέτρηση σταματά, επιτρέποντας στον χρήστη να διαβάσει τον χρόνο που έχει μετρηθεί. Τέλος, με το πάτημα του πλήκτρου reset, το χρονόμετρο επαναφέρεται στην αρχική του κατάσταση, με όλους τους μετρητές να μηδενίζονται.

Η υλοποίηση του χρονομέτρου περιλαμβάνει τη δημιουργία ενός component που διαχειρίζεται τους μετρητές BCD για τα εκατοστά του δευτερολέπτου, τα δευτερόλεπτα, τα λεπτά και τις ώρες. Οι έξοδοι του component συνδέονται με τους ενδείκτες εφτά τομέων για την εμφάνιση του χρόνου. Ενώ η λειτουργία του ελέγχεται μέσω των start, stop και reset.

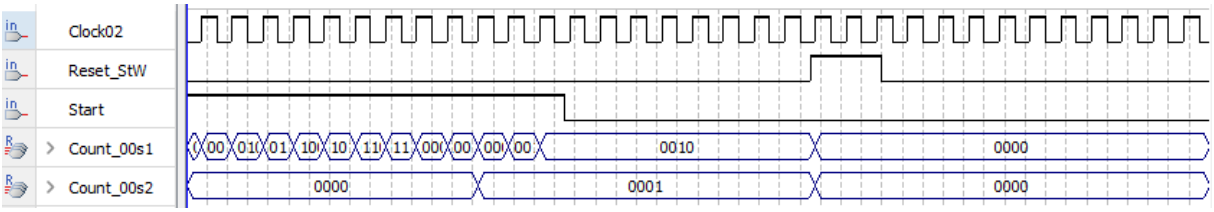
Στο Σχήμα 4.18 απεικονίζεται το διάγραμμα ροής του μετρητή του χρονομέτρου. Όπου ‘00s’ είναι οι μετρητές BCD των μονάδων και των δεκάδων των εκατοστών των δευτερολέπτων, ‘ss’ είναι οι μετρητές BCD των μονάδων και των δεκάδων των δευτερολέπτων, ‘mm’ είναι οι μετρητές των μονάδων και των δεκάδων των λεπτών και αντίστοιχα ‘hh’ είναι οι μετρητές των μονάδων και των δεκάδων των ωρών.

Το σύστημα ξεκινάει και σταματάει με τις αντίστοιχες ενέργειες του χρήστη και ελέγχει συνεχώς την τιμή των BCD μετρητών των ωρών. Αν και οι δύο είναι ίσοι με ‘0000’, η ένδειξη στην οθόνη εμφανίζεται στη μορφή λλ:δδ:εε. Αν τουλάχιστον ένας από τους μετρητές των ωρών είναι διάφορος του ‘0000’, τότε το χρονόμετρο μεταβαίνει στη μορφή ωω:λλ:δδ, ώστε να προσαρμόζεται σωστά η απεικόνιση του χρόνου.



Σχήμα 4.18: Διάγραμμα ροής μετρητή χρονομέτρου.

Στο Σχήμα 4.19 απεικονίζεται η προσομοίωση του χρονομέτρου που πραγματοποιήθηκε στο VWF του Quartus II. Παρατηρούμε ότι όσο το start είναι σε κατάσταση ‘1’, σε κάθε παλμό του ρολογιού αυξάνει ο μετρητής Count_00s1 και, στη συνέχεια, ο Count_00s2. Όταν το start γίνει ‘0’, δηλαδή ο χρήστης έχει πατήσει το stop, η μέτρηση σταματά χωρίς να μηδενιστούν οι μετρητές. Οι μετρητές μηδενίζονται μόνο όταν ο χρήστης πατήσει το reset.



Σχήμα 4.19: Προσομοίωση του χρονομέτρου.

Η προσομοίωση επιβεβαίωσε την ορθή λειτουργία του συστήματος, καθώς οι μετρητές και οι ενδείξεις ανταποκρίνονται στις αναμενόμενες τιμές. Το χρονόμετρο διασφαλίζει ακριβή μέτρηση του χρόνου και ευέλικτο έλεγχο από τον χρήστη, προσφέροντας μια ολοκληρωμένη λύση για τις ανάγκες μέτρησης χρόνου.

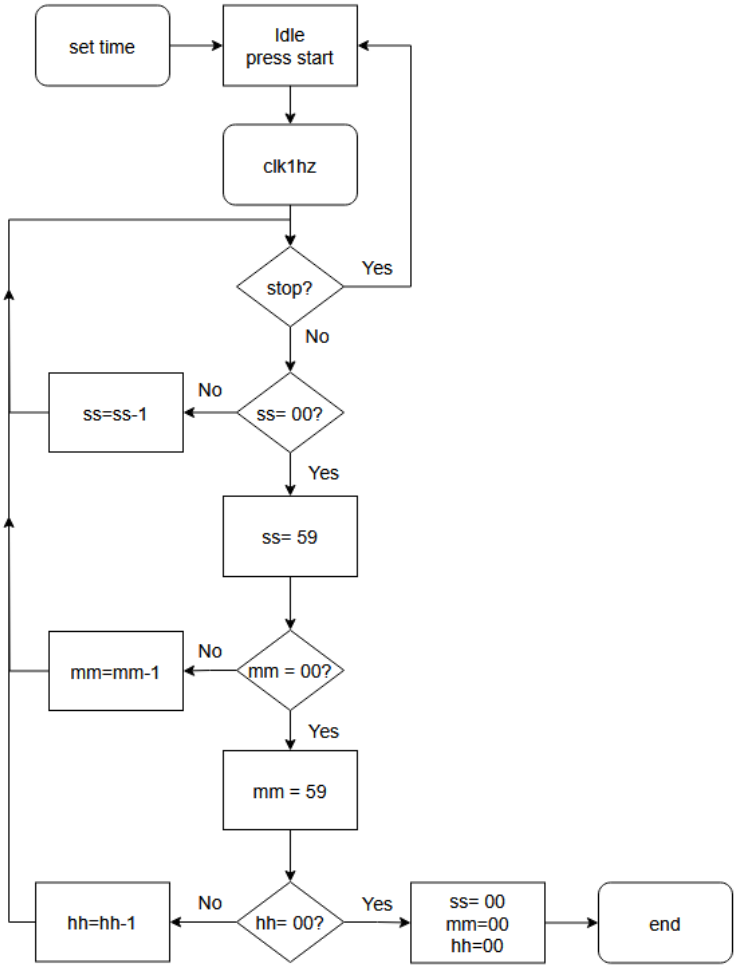
Ο κώδικας του χρονομέτρου παρατίθεται στο ΠΑΡΑΡΤΗΜΑ IV.

4.4.9 Αντίστροφη μέτρηση

Η λειτουργία της αντιστροφής μέτρησης αποτελεί ένα απαραίτητο και χρήσιμο χαρακτηριστικό σε πολλές ψηφιακές συσκευές, προσφέροντας τη δυνατότητα μέτρησης του υπολειπόμενου χρόνου από μια προκαθορισμένη τιμή. Αυτή η λειτουργία βρίσκει εφαρμογές σε διάφορες πρακτικές καταστάσεις, όπως χρονόμετρα για αθλητικές δραστηριότητες, συστήματα ειδοποίησης, βιομηχανικές διεργασίες ή ακόμα και σε καθημερινές ανάγκες, όπως η χρονική διαχείριση μαγειρικών συνταγών ή άλλων εργασιών.

Στο πλαίσιο της παρούσας εργασίας, η υλοποίηση της αντιστροφής μέτρησης στο ψηφιακό ρολόι αποσκοπεί στην παροχή μιας πρόσθετης λειτουργικότητας που επιτρέπει στον χρήστη να ορίσει ένα χρονικό διάστημα και να παρακολουθεί την αντίστροφη μέτρηση μέχρι να φτάσει στο μηδέν. Όταν η μέτρηση φτάσει στο μηδέν, το σύστημα παράγει μια οπτική ειδοποίηση, ενημερώνοντας τον χρήστη ότι ο χρόνος έχει εξαντληθεί.

Η σχεδίαση της αντιστροφής μέτρησης βασίζεται στις ίδιες αρχές με το χρονόμετρο, χρησιμοποιώντας μετρητές BCD για την αναπαράσταση των μονάδων χρόνου (ώρες, λεπτά, δευτερόλεπτα). Ωστόσο, σε αντίθεση με το χρονόμετρο, η αντιστροφή μέτρησης ξεκινά από μια προκαθορισμένη τιμή και μειώνεται προς το μηδέν. Ο χρήστης μπορεί να ορίσει την αρχική τιμή και να ελέγχει τη λειτουργία μέσω πλήκτρων, παρέχοντας ευελιξία και ευκολία στη διαχείριση του χρόνου.



Σχήμα 4.20: Διάγραμμα ροής αντίστροφης μέτρησης.

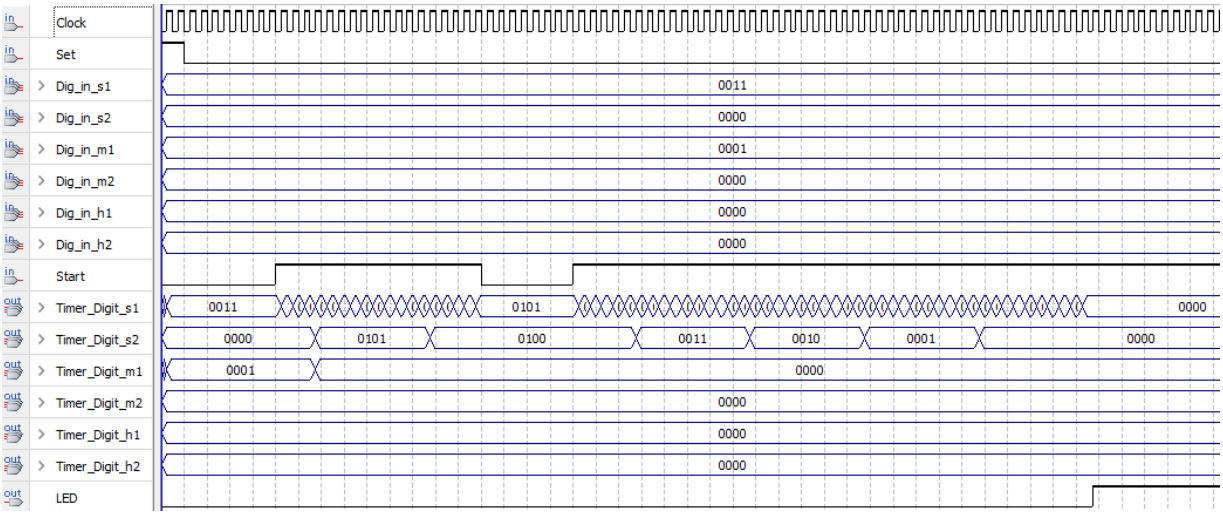
Η υλοποίηση αυτής της λειτουργίας περιλαμβάνει τη δημιουργία ενός component που διαχειρίζεται τους μετρητές BCD για τις ώρες, τα λεπτά και τα δευτερόλεπτα, καθώς και την εμφάνιση του χρόνου στους ενδείκτες εφτά τομέων. Ενώ η ρύθμιση του χρόνου γίνεται σε ξεχωριστό component.

Στο Σχήμα 4.20 απεικονίζεται το διάγραμμα ροής της αντίστροφης μέτρησης, το οποίο περιγράφει τη λειτουργία και τη διαδοχή των καταστάσεων του συστήματος. Η διαδικασία ξεκινά με την αρχικοποίηση του συστήματος, όπου ο χρήστης έχει τη δυνατότητα να ορίσει την αρχική τιμή της αντίστροφης μέτρησης. Η τιμή αυτή αποθηκεύεται στους αντίστοιχους BCD μετρητές για τις ώρες, τα λεπτά και τα δευτερόλεπτα.

Όταν ο χρήστης πατήσει το start, ξεκινά η αντίστροφη μέτρηση. Το σύστημα ελέγχει αν η τιμή των δευτερολέπτων είναι ίση με μηδέν. Εάν δεν είναι, μειώνει τα δευτερόλεπτα κατά ένα και επιστρέφει στην αρχή της διαδικασίας. Εάν τα δευτερόλεπτα φτάσουν στο μηδέν, επαναφέρονται στην τιμή 59 και το σύστημα ελέγχει την τιμή των λεπτών. Εάν τα λεπτά δεν είναι μηδέν, μειώνονται κατά ένα και η διαδικασία επιστρέφει στην αρχή. Εάν τα λεπτά φτάσουν στο μηδέν, επαναφέρονται στην τιμή 59 και το σύστημα ελέγχει την τιμή των ωρών. Εάν οι ώρες είναι μηδέν, όλοι οι μετρητές μηδενίζονται και ο χρήστης ειδοποιείται για την ολοκλήρωση της αντίστροφης μέτρησης. Εάν οι ώρες δεν είναι μηδέν, μειώνονται κατά μία και η διαδικασία επιστρέφει στην αρχή.

Κατά τη διάρκεια της αντίστροφης μέτρησης, ο χρήστης μπορεί να πατήσει το πλήκτρο stop για να διακόψει προσωρινά τη λειτουργία. Σε αυτή την περίπτωση, η τρέχουσα τιμή του χρόνου διατηρείται και η μέτρηση συνεχίζεται από το ίδιο σημείο όταν ο χρήστης πατήσει ξανά το start.

Η διαδικασία τερματίζεται όταν σε έναν παλμό ρολογιού όλοι οι μετρητές BCD είναι ‘0000’. Σε αυτό το σημείο, ενεργοποιείται ένα LED, για να ενημερώσει τον χρήστη ότι ο χρόνος έχει εξαντληθεί.



Σχήμα 4.21: Προσομοίωση αντίστροφης μέτρησης.

Στο Σχήμα 4.21 απεικονίζεται η προσομοίωση της αντίστροφης μέτρησης που πραγματοποιήθηκε στο VWF του Quartus II. Αρχικά, όταν το σήμα set τίθεται σε λογικό ‘1’, το σύστημα αποθηκεύει τις αρχικές τιμές που έχει εισαγάγει ο χρήστης για τις ώρες, τα λεπτά και τα δευτερόλεπτα. Οι τιμές αυτές φορτώνονται στους αντίστοιχους BCD μετρητές και παραμένουν σταθερές έως ότου ξεκινήσει η αντίστροφη μέτρηση.

Η αντίστροφη μέτρηση ξεκινά μόνο όταν το σήμα start γίνει ‘1’. Από εκείνο το σημείο και μετά, το σύστημα μειώνει σταδιακά την τιμή του χρόνου, ξεκινώντας από τα δευτερόλεπτα. Όταν οι μετρητές των δευτερολέπτων φτάσουν στο μηδέν, ο μετρητής των μονάδων των λεπτών μειώνεται κατά μία

μονάδα. Κατά τη διάρκεια της αντίστροφης μέτρησης, αν το σήμα start γίνει ‘0’, η μέτρηση σταματά προσωρινά και διατηρεί την τρέχουσα τιμή της. Όταν το start επιστρέψει σε ‘1’, η μέτρηση συνεχίζεται από το σημείο που είχε διακοπεί.

Η προσομοίωση επιβεβαιώνει ότι το σύστημα ανταποκρίνεται σωστά στις εντολές του χρήστη, διαχειρίζεται ορθά τη μείωση του χρόνου και ενεργοποιεί την ειδοποίηση όταν η μέτρηση φτάσει στο τέλος της.

Ο κώδικας της αντίστροφης μέτρησης παρατίθεται ΠΑΡΑΡΤΗΜΑ V.

4.4.10 Νυχτερινή λειτουργία – εξοικονόμηση ενέργειας

Η νυχτερινή λειτουργία και η εξοικονόμηση ενέργειας αποτελούν βασικά χαρακτηριστικά των σύγχρονων ψηφιακών συσκευών, ιδιαίτερα σε συστήματα που λειτουργούν συνεχώς, όπως τα ψηφιακά ρολόγια. Σε πολλές περιπτώσεις, οι χρήστες δεν χρειάζονται την πλήρη λειτουργία της συσκευής κατά τη διάρκεια της νύχτας, όταν η φωτεινότητα του περιβάλλοντος είναι χαμηλή ή ανύπαρκτη. Για αυτόν τον λόγο, η υλοποίηση μιας νυχτερινής λειτουργίας που μειώνει την κατανάλωση ενέργειας και προσαρμόζει τη συμπεριφορά της συσκευής στις συνθήκες φωτισμού είναι απαραίτητη.

Νυχτερινή λειτουργία

Η νυχτερινή λειτουργία στο ψηφιακό ρολόι που αναπτύχθηκε για την παρούσα εργασία έχει ως στόχο να εξοικονομήσει ενέργεια και να βελτιώσει την εμπειρία του χρήστη. Για την υλοποίησή της, χρησιμοποιείται διαμόρφωση εύρους παλμού (PWM – Pulse Width Modulation) ώστε να μειωθεί η φωτεινότητα των ενδείξεων του ρολογιού κατά τις βραδινές ώρες. Η τεχνική αυτή βασίζεται στην παροχή περιοδικών παλμών στους ενδείκτες επτά τομέων και στα LED, όπου η διάρκεια των παλμών καθορίζει τη φωτεινότητα. Όσο μεγαλύτερη είναι η διάρκεια των παλμών, τόσο μεγαλύτερη είναι η φωτεινότητα, και αντίστροφα. Συγκεκριμένα, η νυχτερινή λειτουργία ενεργοποιείται αυτόματα μεταξύ 19:00 και 07:00, μειώνοντας τη φωτεινότητα στο 10% της κανονικής έντασης.

Η μείωση της φωτεινότητας επιτυγχάνεται μέσω ενός μετρητή, ο οποίος αυξάνεται κυκλικά από το 0 έως το 99. Όταν η νυχτερινή λειτουργία είναι ενεργή, το σήμα PWM ενεργοποιείται μόνο για το 10% του κύκλου λειτουργίας, περιορίζοντας έτσι την κατανάλωση ενέργειας και μειώνοντας την εκπομπή φωτός από τις ενδείξεις. Αντιθέτως, όταν η νυχτερινή λειτουργία είναι απενεργοποιημένη η φωτεινότητα παραμένει στο 100%.

Η νυχτερινή λειτουργία ενεργοποιείται βάσει της τρέχουσας ώρας του ρολογιού και μειώνει τη φωτεινότητα ανεξάρτητα από την επιλεγμένη λειτουργία. Είτε το σύστημα βρίσκεται σε κανονική λειτουργία ρολογιού, είτε σε χρονομέτρηση, είτε σε αντίστροφη μέτρηση, η φωτεινότητα προσαρμόζεται αυτόματα σύμφωνα με τις ρυθμίσεις της νυχτερινής λειτουργίας. Ωστόσο, κατά τη ρύθμιση της ώρας από τον χρήστη, η ακρίβεια της αποθηκευμένης ώρας δεν μπορεί να θεωρηθεί δεδομένη. Για αυτόν τον λόγο, η νυχτερινή λειτουργία απενεργοποιείται προσωρινά όταν το ρολόι βρίσκεται σε κατάσταση ρύθμισης, αποτρέποντας έτσι την πιθανή εσφαλμένη ενεργοποίησή της.

Με αυτόν τον τρόπο, αποτρέπεται η ενεργοποίηση της χαμηλής φωτεινότητας σε λάθος χρονικές περιόδους, εξασφαλίζοντας ότι η νυχτερινή λειτουργία θα ενεργοποιηθεί μόνο όταν η ώρα έχει ρυθμιστεί σωστά και το σύστημα επανέλθει στη φυσιολογική του λειτουργία. Μόλις ολοκληρωθεί η ρύθμιση της ώρας, η νυχτερινή λειτουργία μπορεί να επανενεργοποιηθεί αυτόματα, βασισμένη στη νέα, σωστά καταχωρημένη ώρα, εξασφαλίζοντας την αποτελεσματική διαχείριση της φωτεινότητας.

Εξοικονόμηση ενέργειας

Η λειτουργία εξοικονόμησης ενέργειας ενεργοποιείται χειροκίνητα από τον χρήστη όταν το σύστημα βρίσκεται σε λειτουργία ρολογιού. Όταν αυτή η λειτουργία είναι ενεργή, η οθόνη του ρολογιού απενεργοποιείται για να μειωθεί η κατανάλωση ενέργειας, ενώ το ρολόι συνεχίζει να λειτουργεί κανονικά στο παρασκήνιο. Η ώρα εμφανίζεται για δέκα δευτερόλεπτα κάθε λεπτό, εξασφαλίζοντας ότι ο χρήστης μπορεί να δει την τρέχουσα ώρα όταν το χρειάζεται.

Η λειτουργία εξοικονόμησης ενέργειας απενεργοποιείται είτε με τον ίδιο χειροκίνητο τρόπο από τον χρήστη, είτε αυτόματα, όταν ο χρήστης αλλάξει λειτουργία (π.χ., μεταβεί σε χρονόμετρο). Με αυτόν τον τρόπο, η λειτουργία εξοικονόμησης ενέργειας προσαρμόζεται στις ανάγκες του χρήστη και τις απαιτήσεις της εκάστοτε λειτουργίας του συστήματος.

Οι κώδικες της νυχτερινής λειτουργίας και της εξοικονόμησης ενέργειας παρατίθενται στο ΠΑΡΑΡΤΗΜΑ VI.

4.4.11 Υλοποίηση Μηχανής Πεπερασμένων Καταστάσεων

Η Μηχανή Πεπερασμένων Καταστάσεων (Finite State Machine - FSM) αποτελεί τον βασικό μηχανισμό ελέγχου στο ψηφιακό ρολόι, καθώς διαχειρίζεται τις μεταβάσεις μεταξύ των διαφόρων λειτουργιών του συστήματος. Η FSM ορίζει τις καταστάσεις του ρολογιού και τις συνθήκες με τις οποίες το σύστημα μεταβαίνει από τη μια κατάσταση στην άλλη, διασφαλίζοντας ότι οι ενέργειες του συστήματος είναι σωστά συγχρονισμένες και ελεγχόμενες.

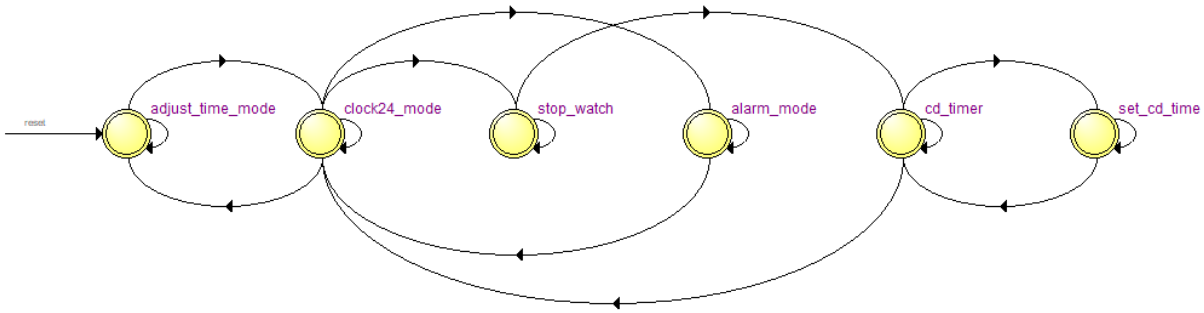
Η FSM που έχει σχεδιαστεί για το παρόν σύστημα αποτελείται από διακριτές καταστάσεις, καθεμία από τις οποίες αντιστοιχεί σε μία συγκεκριμένη λειτουργία. Οι μεταβάσεις μεταξύ αυτών των καταστάσεων καθορίζονται από σήματα εισόδου και εσωτερικούς μετρητές. Οι κύριες καταστάσεις του συστήματος περιλαμβάνουν:

- Ρολόι: Η βασική κατάσταση του συστήματος, όπου εμφανίζεται η τρέχουσα ώρα και ημερομηνία.
- Χρονόμετρο: Κατάσταση μέτρησης χρόνου με δυνατότητες έναρξης, παύσης και επαναφοράς.
- Αντίστροφη Μέτρηση: Κατάσταση αντίστροφης μέτρησης από μια προκαθορισμένη τιμή.
- Λειτουργία αφύπνισης: Κατάσταση ρύθμισης της ώρας ειδοποίησης.
- Ρύθμιση Ώρας/Ημερομηνίας: Κατάσταση ρύθμισης της ώρας και της ημερομηνίας του συστήματος.
- Ρύθμιση Αντίστροφης Μέτρησης: Κατάσταση ρύθμισης του χρόνου για την αντίστροφη μέτρηση.

Οι μεταβάσεις μεταξύ των καταστάσεων εξαρτώνται από συγκεκριμένες ενέργειες του χρήστη και συνθήκες που καθορίζουν την επόμενη κατάσταση. Παρακάτω παρουσιάζονται οι βασικές μεταβάσεις:

- Η μετάβαση από Ρύθμιση Ώρας/Ημερομηνίας σε Ρολόι πραγματοποιείται όταν ολοκληρωθεί η ρύθμιση και ο χρήστης πατήσει στιγμιαία το πλήκτρο KEY3.
- Η μετάβαση από Ρολόι σε Ρύθμιση Ώρας/Ημερομηνίας πραγματοποιείται όταν ο χρήστης πατήσει παρατεταμένα το πλήκτρο KEY0.
- Η μετάβαση από Ρολόι σε Χρονόμετρο πραγματοποιείται όταν ο χρήστης πατήσει στιγμιαία το πλήκτρο KEY3.
- Η μετάβαση από Χρονόμετρο σε Αντίστροφη Μέτρηση πραγματοποιείται όταν ο χρήστης πατήσει στιγμιαία το πλήκτρο KEY3.
- Η μετάβαση από Αντίστροφη Μέτρηση σε Ρύθμιση Αντίστροφης Μέτρησης πραγματοποιείται όταν ο χρήστης πατήσει παρατεταμένα το πλήκτρο KEY0.

- Η μετάβαση από Ρύθμιση Αντίστροφης Μέτρησης σε Αντίστροφη Μέτρηση πραγματοποιείται όταν ολοκληρωθεί η ρύθμιση του χρόνου πατήσει στιγμιαία το πλήκτρο KEY3.
- Η μετάβαση από Αντίστροφη Μέτρηση σε Ρολόι πραγματοποιείται όταν ο χρήστης πατήσει στιγμιαία το πλήκτρο KEY3.
- Η μετάβαση από Ρολόι σε Λειτουργία Αφύπνισης πραγματοποιείται όταν ο χρήστης πατήσει παρατεταμένα το πλήκτρο KEY3.
- Η μετάβαση από Λειτουργία Αφύπνισης σε Ρολόι πραγματοποιείται όταν ολοκληρωθεί η ρύθμιση και ο χρήστης πατήσει στιγμιαία το πλήκτρο KEY3.



Σχήμα 4.22: Διάγραμμα FSM του Ψηφιακού Ρολογιού.

Στο Σχήμα 4.22 παρουσιάζεται το διάγραμμα ροής της FSM, όπως προκύπτει από το εργαλείο State Machine Viewer του Quartus II. Απεικονίζονται οι καταστάσεις του συστήματος καθώς και οι μεταβάσεις μεταξύ τους, προσδιορίζοντας τη λογική λειτουργίας του ψηφιακού ρολογιού. Συγκεκριμένα, οι καταστάσεις περιλαμβάνουν: adjust_time_mode, που αφορά τη ρύθμιση της ώρας, clock24_mode, που είναι η βασική λειτουργία εμφάνισης της ώρας, stop_watch, το χρονόμετρο, alarm_mode, που αντιστοιχεί στο λειτουργία αφύπνισης, cd_timer, που αφορά την αντίστροφη μέτρηση, και set_cd_time, την κατάσταση ρύθμισης του χρόνου της αντίστροφης μέτρησης.

4.4.12 Κύρια μονάδα Top-level

Το Top-Level module αποτελεί τον κεντρικό πυρήνα του ψηφιακού ρολογιού, όπου όλες οι λειτουργικές μονάδες συνδέονται και συντονίζονται ώστε να εξασφαλιστεί η ομαλή λειτουργία του συστήματος. Αυτό το module είναι υπεύθυνο για τη διαχείριση των εισόδων, την επεξεργασία των δεδομένων και την εμφάνιση των πληροφοριών στους ενδείκτες 7 τομέων, καθώς και για τον έλεγχο των διαφόρων λειτουργιών, όπως το ρολόι, τη λειτουργία αφύπνισης, το χρονόμετρο και η αντίστροφη μέτρηση. Επιπλέον, αναλαμβάνει τη διασύνδεση και τον συγχρονισμό μεταξύ όλων των υπομονάδων του συστήματος, διασφαλίζοντας τη σωστή επικοινωνία και συνεργασία τους.

Το Top-Level Module οργανώνεται γύρω από μια ιεραρχική δομή, όπου κάθε υποσύστημα εκτελεί μια συγκεκριμένη λειτουργία. Οι διαιρέτες συχνότητας παράγουν τα απαραίτητα σήματα ρολογιού για το σύστημα, εξασφαλίζοντας τη σωστή λειτουργία του χρονισμού. Ο μετρητής χρόνου και ημερομηνίας είναι υπεύθυνος για τον υπολογισμό και την αποθήκευση της τρέχουσας ώρας, περιλαμβάνοντας ώρες, λεπτά, δευτερόλεπτα, καθώς και ημερομηνία με ημέρες, μήνες και έτη. Παράλληλα, ο αποκωδικοποιητής επτά τομέων μετατρέπει τις τιμές BCD σε σήματα για την οπτική αναπαράσταση της πληροφορίας στους ενδείκτες. Το υποσύστημα της λειτουργίας αφύπνισης διαχειρίζεται τις ειδοποιήσεις, ενεργοποιώντας τις αντίστοιχες εξόδους όταν πληρούνται οι προκαθορισμένες συνθήκες. Αντίστοιχα, το χρονόμετρο και η αντίστροφη μέτρηση προσφέρουν

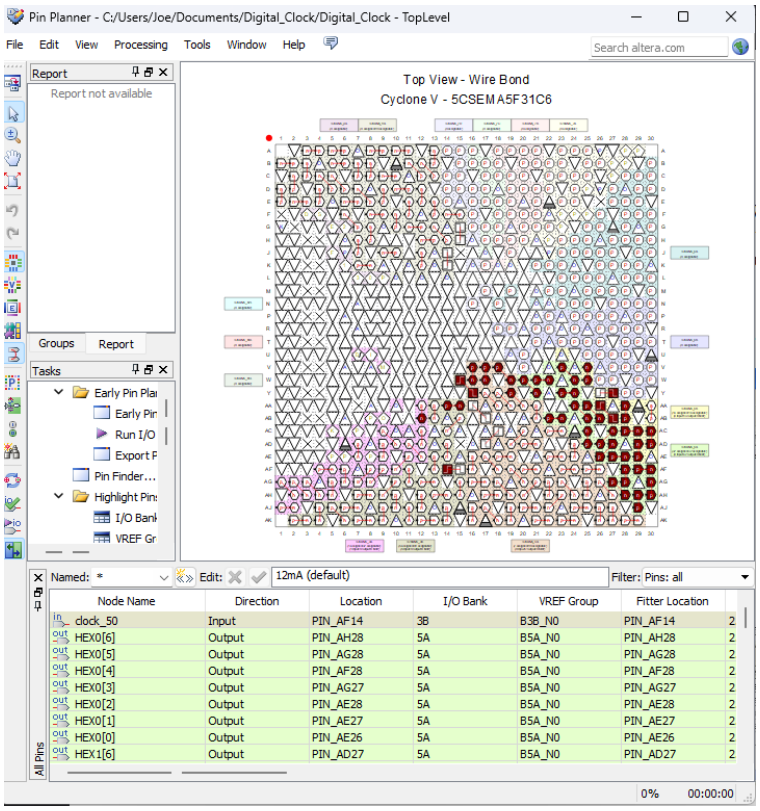
επιπλέον λειτουργικότητα, ενώ η Μηχανή Πεπερασμένων Καταστάσεων ελέγχει τις μεταβάσεις μεταξύ των διαφορετικών λειτουργιών του συστήματος.

Ένα σημαντικό πλεονέκτημα της ιεραρχικής σχεδίασης στο VHDL είναι η δυνατότητα επαναχρησιμοποίησης των components μέσα στο Top-Level. Για παράδειγμα, τον αποκωδικοποιητή επτά τομέων τον δηλώσαμε μία φορά ως component στη σχεδίαση, και τον χρησιμοποιήσαμε έξι φορές, μία για κάθε ενδείκτη επτά τομέων, μέσω της δημιουργίας στιγμιοτύπων (instances). Αυτό σημαίνει ότι αντί να επαναγράψουμε τον ίδιο κώδικα για κάθε ένδειξη, απλά δημιουργήσαμε ξεχωριστά στιγμιότυπα του component, καθένα από τα οποία αναλαμβάνει να αποκωδικοποιήσει και να προβάλλει μια συγκεκριμένη ψηφιακή τιμή στους αντίστοιχους ενδείκτες. Αυτή η προσέγγιση επιτρέπει τη βελτιωμένη διαχείριση του κώδικα, την αποφυγή επαναλήψεων και την ευκολότερη συντήρηση του συστήματος.

Το Top-Level module λαμβάνει ως εισόδους τα σήματα ρολογιού, που συγχρονίζουν το σύστημα, και τις εντολές του χρήστη μέσω των πλήκτρων που επιτρέπουν την αλληλεπίδραση. Οι έξοδοι του συστήματος περιλαμβάνουν τα σήματα προς τις οθόνες επτά τομέων, όπου εμφανίζονται η ώρα, η ημερομηνία και οι υπόλοιπες λειτουργίες, καθώς και ενδείξεις LED για ειδοποιήσεις, όπως η ενεργοποίηση της αφύπνισης ή όταν πληρούνται οι αντίστοιχες συνθήκες.

4.5 Ορισμός εισόδων/εξόδων

Αφού ολοκληρώσαμε τη σχεδίαση όλων των component και την υλοποίηση του Top-Level, το επόμενο βήμα είναι η αντιστοίχιση των εισόδων και εξόδων του συστήματος στα φυσικά pins του FPGA. Για να μπορέσει το ψηφιακό ρολόι να αλληλεπιδράσει με την αναπτυξιακή πλακέτα και τα εξωτερικά περιφερειακά, πρέπει να καθορίσουμε πώς συνδέονται τα σήματα του σχεδίου μας με τις φυσικές επαφές της συσκευής.

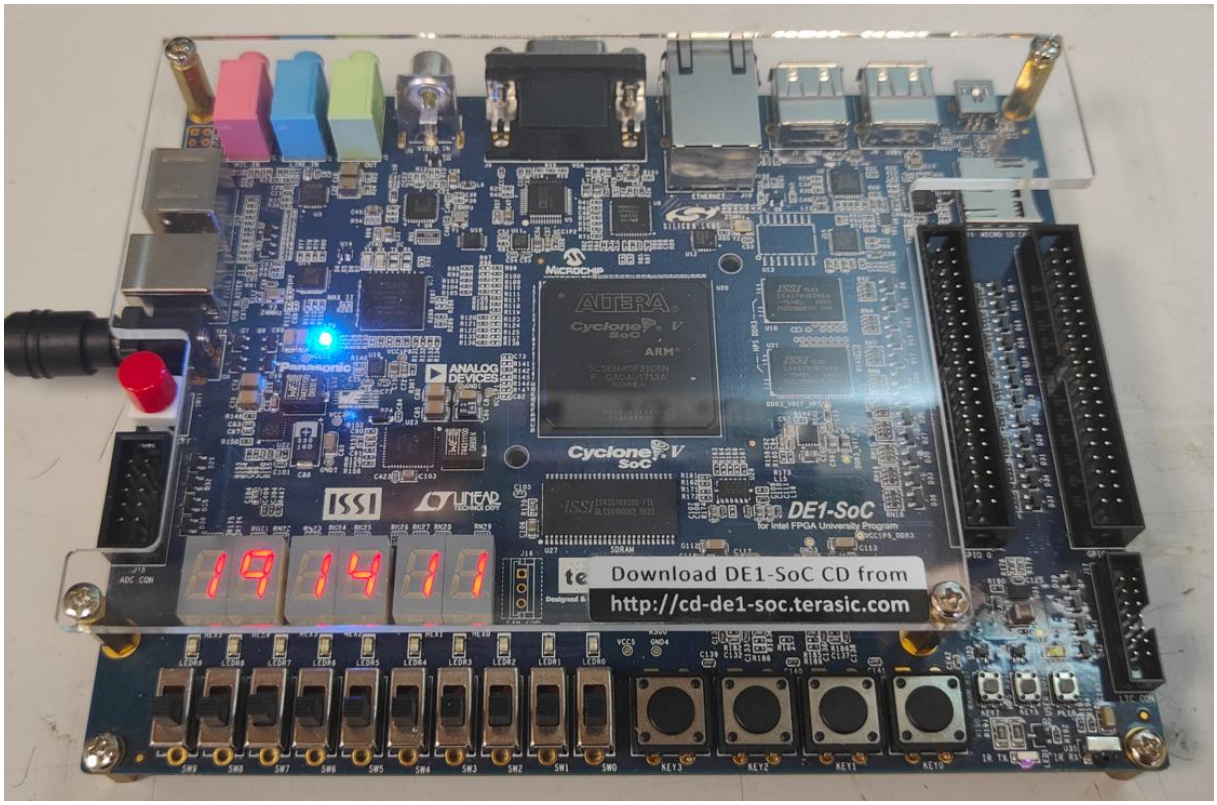


Σχήμα 4.23: Παράθυρο pin planner του Quartus II.

Αρχικά δηλώθηκαν οι είσοδοι και οι έξοδοι στο entity του Top-Level αρχείου, καθορίζοντας τη λειτουργία τους. Στη συνέχεια, μέσω του Pin Planner του Quartus II, πραγματοποιήθηκε η αντιστοίχιση των σημάτων του σχεδίου στις κατάλληλες επαφές του FPGA, ώστε να διασφαλιστεί η σωστή σύνδεση του συστήματος με το υλικό της πλακέτας.

Από το μενού Assignments του Quartus II επιλέγουμε Pin Planner, όπου ανοίγει ένα νέο παράθυρο στο οποίο μεταξύ άλλων εμφανίζονται το top view του FPGA, στη συγκεκριμένη περίπτωση το 5CSEMA5F31C6. Στο κάτω μέρος του παραθύρου εμφανίζονται οι είσοδοι και οι έξοδοι, ρολόι, LED, ενδείκτες επτά τομέων, πλήκτρα, διακόπτες, που έχουμε στο Top-Level αρχείο. Ακολουθώντας το εγχειρίδιο χρήσης της πλακέτας DE1-Soc αντιστοιχίζουμε τις εισόδους και εξόδους του κώδικά μας με τα φυσικά pins του FPGA. Αυτή η διαδικασία εξασφαλίζει ότι κάθε σήμα του σχεδίου μας θα συνδεθεί σωστά με το αντίστοιχο υλικό, επιτρέποντας τη σωστή λειτουργία του ψηφιακού ρολογιού.

Στο Σχήμα 4.24 απεικονίζεται η αναπτυξιακή πλακέτα DE1-SoC σε λειτουργία εμφάνισης ώρας. Περισσότερες λεπτομέρειες και φωτογραφίες σχετικά με την υλοποίηση και τα αποτελέσματα της δοκιμής θα βρεθούν στο ΠΑΡΑΡΤΗΜΑ VII, όπου θα παρουσιαστούν αναλυτικά φωτογραφίες της πλακέτας κατά τη λειτουργία του συστήματος.



Σχήμα 4.24: Η αναπτυξιακή πλακέτα DE1-SoC σε λειτουργία εμφάνισης ώρας.

4.6 Επίλογος

Στο παρόν κεφάλαιο αναλύθηκε η διαδικασία σχεδίασης και υλοποίησης ενός ψηφιακού ρολογιού με χρήση FPGA, αξιοποιώντας τη γλώσσα περιγραφής υλικού VHDL. Αναλύθηκαν οι βασικές απαιτήσεις του συστήματος, η αρχιτεκτονική του κώδικα, καθώς και οι επιμέρους λειτουργίες, όπως η ρύθμιση ώρας και ημερομηνίας, τη λειτουργία αφύπνισης, το χρονόμετρο και η επιλογή μεταξύ 12ωρης και 24ωρης μορφής.

Η ανάπτυξη πραγματοποιήθηκε στην αναπτυξιακή πλακέτα DE1-SoC, αξιοποιώντας το περιβάλλον Quartus II για τη σύνθεση, προσομοίωση και υλοποίηση του σχεδιασμού. Ιδιαίτερη έμφαση δόθηκε στη χρήση μετρητών BCD για την ακριβή διαχείριση του χρόνου και της ημερομηνίας, καθώς και στη δημιουργία κατάλληλων αποκωδικοποιητών για την απεικόνιση των τιμών στις ενδείξεις 7 τομέων.

Η υλοποίηση δοκιμάστηκε μέσω προσομοιώσεων στο περιβάλλον Quartus II, όπου επαληθεύτηκε η ορθή λειτουργία όλων των υπομονάδων του συστήματος. Οι δοκιμές επιβεβαίωσαν την ακρίβεια της μέτρησης του χρόνου, τη σωστή μετάβαση μεταξύ χρονικών μονάδων και τη λειτουργικότητα των πρόσθετων δυνατοτήτων. Τέλος, έγινε η αντιστοίχιση των εισόδων και εξόδων του κώδικα με τα φυσικά pin της του FPGA.

Κεφάλαιο 5ο Συμπεράσματα

Υλοποιήσεις ψηφιακών ρολογιών σε FPGA έχουν αναπτυχθεί σε πολλές διαφορετικές μορφές, τόσο σε ακαδημαϊκό όσο και σε ερασιτεχνικό επίπεδο, και αποτελούν ένα ενδιαφέρον πεδίο μελέτης λόγω της ποικιλίας των προσεγγίσεων και των τεχνολογιών που χρησιμοποιούνται. Σε αυτή την ανάλυση, επιλέγουμε και παρουσιάζουμε ορισμένες από τις πιο χαρακτηριστικές, προκειμένου να συγκρίνουμε τα βασικά τους χαρακτηριστικά με την προτεινόμενη υλοποίησή μας.

Το 2023 ο Hanchen Kang στο άρθρο «Design of electronic clock and stopwatch based on FPGA and VHDL language» [14] παρουσίασε την ανάπτυξη ενός ηλεκτρονικού ρολογιού και χρονομέτρου με χρήση FPGA και VHDL. Το σύστημα που σχεδιάστηκε διαθέτει λειτουργία ρολογιού, όπου προβάλλεται η ώρα σε 24ωρη μορφή με δυνατότητα ρύθμισης, λειτουργία αφύπνισης με δυνατότητα αναβολής για δέκα λεπτά και λειτουργία χρονομέτρου, που μετρά λεπτά, δευτερόλεπτα και εκατοστά του δευτερολέπτου, επιτρέποντας την προσωρινή αποθήκευση έως τριών προσωρινών μετρήσεων. Για την υλοποίηση σε FPGA, χρησιμοποιείται κύκλωμα διαιρέτη συχνότητας που παράγει σήματα 1Hz και 100 Hz, ενώ η μέτρηση του χρόνου βασίζεται σε μετρητές modulo-60 και modulo-24 για το ρολόι, και modulo-100 και modulo-60 για το χρονόμετρο. Επιπλέον χρησιμοποιείται ένας βομβητής για την αναπαραγωγή μουσικής κατά την ενεργοποίηση της λειτουργίας αφύπνησης.

Το 2013 οι Vedat Kiray και Meirambek Zhaparov στο άρθρο «FPGA based digital electronic education, clock calendar design» [15] παρουσίασαν την υλοποίηση ενός ρολογιού-ημερολογίου σε πλατφόρμα FPGA ως μέρος ενός εκπαιδευτικού προγράμματος για διδασκαλία ψηφιακής σχεδίασης. Ο σχεδιασμός περιλαμβάνει τρεις κύριες μονάδες: τη μονάδα καταμέτρησης για δευτερόλεπτα, λεπτά, ώρες, ημέρες, μήνες και έτη. Τη μονάδα ρύθμισης για την επιλογή και αλλαγή των τιμών και τη μονάδα εμφάνισης σε ενδείκτες επτά τομέων. Για την υλοποίηση δεν χρησιμοποιείται κάποια γλώσσα περιγραφής υλικού, αλλά γίνεται μέσω σχεδίασης σε γραφικό περιβάλλον για καλύτερη κατανόηση από τους φοιτητές και ενίσχυση των δεξιοτήτων τους στην ψηφιακή σχεδίαση. Ενώ χρησιμοποιούνται βασικά ψηφιακά κυκλώματα όπως μετρητές, πολυπλέκτες, συγκριτές και αποκωδικοποιητές.

Το 2015 οι Jun Yang, Hongye Li και Long Liu στο άρθρο «Design and implementation of the infrared remote-controlled digital clock based on FPGA» [16] παρουσίασαν τον σχεδιασμό και την υλοποίηση ενός ψηφιακού ρολογιού με υπέρυθρο τηλεχειριστήριο βασισμένο σε FPGA, χρησιμοποιώντας τη γλώσσα VHDL. Το σύστημα αποτελείται από τρία κύρια μέρη: τη μονάδα του υπέρυθρου τηλεχειριστηρίου, τη μονάδα του ψηφιακού ρολογιού και τις επιμέρους λειτουργικές μονάδες (ένδειξη ώρας, ξυπνητήρι, ηχητική ειδοποίηση και μέτρηση θερμοκρασίας). Η επικοινωνία μέσω υπέρυθρων επιτυγχάνεται με τη διαμόρφωση και αποδιαμόρφωση παλμών, ενώ η κύρια μονάδα του ρολογιού χρησιμοποιεί κύκλωμα διαίρεσης συχνότητας και μετρητές για τη ρύθμιση και μέτρηση της ώρας. Για τη μέτρηση της θερμοκρασίας χρησιμοποιείται ένας αισθητήρας θερμοκρασίας DS18B20.

Η σχεδίαση και η υλοποίηση του συστήματος της παρούσας εργασίας με χρήση FPGA και VHDL αποτέλεσε μια ολοκληρωμένη εμπειρία που ενσωμάτωσε πολλές βασικές έννοιες του ψηφιακού σχεδιασμού. Μέσα από αυτή την εργασία, επιτεύχθηκε η δημιουργία ενός πλήρως λειτουργικού συστήματος που περιλαμβάνει λειτουργίες όπως η εμφάνιση της ώρας και της ημερομηνίας, η ρύθμιση του χρόνου, η λειτουργία αφύπνισης, το χρονόμετρο και η αντίστροφη μέτρηση. Επιπλέον, υλοποιήθηκε η δυνατότητα επιλογής μεταξύ 12ωρης και 24ωρης μορφής ώρας και λειτουργίες εξοικονόμησης ενέργειας και νυχτερινής λειτουργίας. Μελλοντικά, το σύστημα μπορεί να επεκταθεί με την αυτοματοποίηση της αλλαγής θερινής/χειμερινής ώρας, με δυνατότητα προσωρινής

αποθήκευσης μετρήσεων του χρονομέτρου, δυνατότητα αναβολής της λειτουργίας αφύπνισης, καθώς και προσθήκη λειτουργίας μέτρησης υγρασίας και θερμοκρασίας.

Ο Πίνακας 5.1 παρουσιάζει συνοπτικά τις λειτουργίες που υποστηρίζει κάθε υλοποίηση, επιτρέποντας μια άμεση σύγκριση των χαρακτηριστικών τους.

Πίνακας 5.1: Σύγκριση Λειτουργιών.

Λειτουργία	H. Kang	V. Kiray & M. Zhaparov	J.Yang et al.	Δική μας Υλοποίηση
Εμφάνιση ώρας	24ωρη μορφή	24ωρη μορφή	24ωρη μορφή	12/24ωρη μορφή
Ημερομηνία	Όχι	Ναι	Όχι	Ναι
Αφύπνιση	Ναι (με αναβολή και ηχητική ειδοποίηση)	Όχι	Ναι (ηχητική ειδοποίηση)	Ναι
Χρονόμετρο	Ναι (με έως 3 προσωρινές αποθηκεύσεις)	Όχι	Όχι	Ναι
Αντίστροφη μέτρηση	Όχι	Όχι	Όχι	Ναι
Μέτρηση θερμοκρασίας	Όχι	Όχι	Ναι	Όχι
Εξοικονόμηση ενέργειας	Όχι	Όχι	Όχι	Ναι
Νυχτερινή λειτουργία	Όχι	Όχι	Όχι	Ναι
Επικοινωνία	Όχι	Όχι	Υπέρυθρο τηλεχειριστήριο	Όχι

Η χρήση της γλώσσας VHDL επέτρεψε την ακριβή περιγραφή της λογικής και της συμπεριφοράς του συστήματος, ενώ η ιεραρχική δομή του κώδικα διευκόλυνε την οργάνωση και την επαναχρησιμοποίηση των υποσυστημάτων. Η εφαρμογή των μετρητών BCD για τη διαχείριση του χρόνου και της ημερομηνίας εξασφάλισε ακρίβεια και αξιοπιστία, ενώ η χρήση αποκωδικοποιητών επτά τομέων επέτρεψε την οπτική αναπαράσταση των πληροφοριών στους ενδείκτες.

Η προσομοίωση του συστήματος στο περιβάλλον Quartus II επιβεβαίωσε την ορθή λειτουργία όλων των λειτουργιών, ενώ η αντιστοίχιση των εισόδων και εξόδων με τα φυσικά pins του FPGA εξασφάλισε τη σωστή λειτουργία του συστήματος σε πραγματικό χρόνο. Η εργασία αυτή αποτέλεσε μια σημαντική εμπειρία στην κατανόηση των FPGA και των δυνατοτήτων τους, καθώς και στην εφαρμογή των γλωσσών περιγραφής υλικού για την υλοποίηση πολύπλοκων ψηφιακών συστημάτων.

Συνολικά, η εργασία αυτή υπογραμμίζει τη σημασία των FPGA και της VHDL στη σύγχρονη σχεδίαση ηλεκτρονικών συστημάτων, προσφέροντας μια πρακτική εφαρμογή που μπορεί να αποτελέσει βάση για μελλοντικές αναπτύξεις και βελτιώσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] P. Horowitz και W. Hill, The Art of Electronics - Third Edition, New York, NY: Cambridge University Press, 2015.

[2] S. Brown και Z. Vranesic, Fundamentals of Digital Logic with VHDL Design – 3rd ed., New York: McGraw-Hill, 2009.

[3] A. K. Maini, Digital Electronics: Principles, Devices and Applications, Daryaganj, New Delhi: John Wiley & Sons, 2008.

[4] Lattice Semiconductor Corporation, *"High Performance E2 CMOS PLD", GAL22V10 datasheet,* 2006.

[5] «https://eee.poriyaan.in,» EEE Dept Anna University, [Ηλεκτρονικό]. Available: https://eee.poriyaan.in/topic/cpld--complex-programmable-logic-devices--11688/. [Πρόσβαση 02 2025].

[6] S. M. Trimberger, «Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology,» *Proceedings of the IEEE,* p. 318–331, March 2015.

[7] S. Hauck και A. DeHon, Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation, Morgan Kaufmann, 2008.

[8] R. C. Cofer και B. F. Harding, Rapid System Prototyping with FPGAs, Newnes, 2006.

[9] R. Sass και A. G. Schmidt, Embedded Systems Design with Platform FPGAs, Morgan Kaufmann, 2010.

[10] H. Parvez και H. Mehrez, Application-Specific Mesh-based Heterogeneous FPGA Architectures, Springer, 2011.

[11] V. A. Pedroni, Circuit Design with VHDL, Cambridge, Massachusetts: The M IT Press, 2020.

[12] Intel, *Cyclone V Device Overview,* 2018.

[13] Terasic, *DE1 - SoC User Manual.*

[14] H. Kang, «Design of electronic clock and stopwatch based on FPGA and VHDL language,» *Journal of Physics: Conference Series,* November 2023.

[15] V. Kiray και M. Zhaparov, «FPGA Based Digital Electronic Education, Clock-Calendar Design,» *PRZEGLĄD ELEKTROTECHNICZNY,* January 2013.

[16] J. Yang, H. Li και L. Liu, «Design and implementation of the infrared remote-controlled digital clock based on FPGA,» *Applied Mechanics and Materials,* March 2015.

ΠΑΡΑΡΤΗΜΑ Ι Κώδικας μετρητή ώρας/ημερομηνίας

```
Process (clock)
    Variable year_int : Integer;
Begin

    If Rising_Edge(clock) Then

        Count_s1  <= Count_s1 + 1;
        year1_int <= to_integer(unsigned(year1));
        year2_int <= to_integer(unsigned(year2));
        year3_int <= to_integer(unsigned(year3));
        year4_int <= to_integer(unsigned(year4));

        If (Count_s1 = "1001") Then
            Count_s1 <= "0000";
            Count_s2 <= Count_s2 + 1;
            If (Count_s2 = "0101") Then
                Count_s2 <= "0000";
                Count_m1 <= Count_m1 + 1;
                If (Count_m1 = "1001") Then
                    Count_m1 <= "0000";
                    Count_m2 <= Count_m2 + 1;
                    If (Count_m2 = "0101") Then
                        Count_m2 <= "0000";
                        Count_h1 <= Count_h1 + 1;
                        If (Count_h1 = "1001") Then
                            Count_h1 <= "0000";
                            Count_h2 <= Count_h2 + 1;
                        End If;
                        If (Count_h2 = "0010" AND Count_h1 = "0011") Then
                            Count_s1 <= "0000";
                            Count_s2 <= "0000";
                            Count_m1 <= "0000";
                            Count_m2 <= "0000";
                            Count_h1 <= "0000";
                            Count_h2 <= "0000";
                            day1     <= day1 + 1;

                            If (day1 = "1001") Then
                                day1 <= "0000";
                                day2 <= day2 + 1;
                            End If;

                            If day2 >= max_day2 AND day1 >= max_day1 Then

                                day1   <= "0001";
                                day2   <= "0000";
                                month1 <= month1 + 1;
                                If (month1 = "1001") Then
                                    month1 <= "0000";
                                    month2 <= month2 + 1;
                                End If;
                                If (month2 = "0001" AND month1 = "0010")
Then

                                    month1 <= "0001";
                                    month2 <= "0000";
                                    year1  <= year1 + 1;
                                End If;
                                If (year1 = "1001") Then
```

```
                                    year1 <= "0000";
                                    year2 <= year2 + 1;
                                End If;
                                If (year2 = "1001" AND year1 = "1001") Then
                                    year1 <= "0000";
                                    year2 <= "0000";
                                    year3 <= year3 + 1;
                                End If;
                                If (year3 = "1001" AND year2 = "1001" AND
year1 = "1001") Then
                                    year1 <= "0000";
                                    year2 <= "0000";
                                    year3 <= "0000";
                                    year4 <= year4 + 1;
                                End If;
                                If (year4 = "1001" AND year3 = "1001" AND
year2 = "1001" AND year1 = "1001") Then
                                    year1 <= "0000";
                                    year2 <= "0000";
                                    year3 <= "0000";
                                    year4 <= "0000";
                                End If;
                            End If;

                        End If;
                    End If;
                End If;
            End If;
        End If;

        if Adjust = '1' Then

            Count_s1 <= "0000";
            Count_s2 <= "0000";

            Count_m1 <= Adj_Digit_m1;
            Count_m2 <= Adj_Digit_m2;
            Count_h1 <= Adj_Digit_h1;
            Count_h2 <= Adj_Digit_h2;

            day1   <= Adj_Digit_D1;
            day2   <= Adj_Digit_D2;
            month1 <= Adj_Digit_MON1;
            month2 <= Adj_Digit_MON2;
            year1  <= Adj_Digit_Y1;
            year2  <= Adj_Digit_Y2;
            year3  <= Adj_Digit_Y3;
            year4  <= Adj_Digit_Y4;
        End If;

        If (month2 = "0000" AND month1 = "0001") OR  -- Ιανουάριος
            (month2 = "0000" AND month1 = "0011") OR -- Μάρτιος
            (month2 = "0000" AND month1 = "0101") OR -- Μάιος
            (month2 = "0000" AND month1 = "0111") OR -- Ιούλιος
            (month2 = "0000" AND month1 = "1000") OR -- Αύγουστος
            (month2 = "0001" AND month1 = "0000") OR -- Οκτώβριος
            (month2 = "0001" AND month1 = "0010")    -- Δεκέμβριος
        Then
            max_day1 <= "0001";
            max_day2 <= "0011";
        Elsif (month2 = "0000" AND month1 = "0100") OR -- Απρίλιος
```

```
            (month2 = "0000" AND month1 = "0110") OR    -- Ιούνιος
            (month2 = "0000" AND month1 = "1001") OR    -- Σεπτέμβριος
            (month2 = "0001" AND month1 = "0001")       -- Νοέβριος
        Then
            max_day1 <= "0000";
            max_day2 <= "0011";
        Elsif (month2 = "0000" AND month1 = "0010") -- Φεβρουάριος
        Then
            max_day2 <= "0010";
            If (full_year_int MOD 4 = 0 AND
               (full_year_int MOD 100 /= 0 OR full_year_int MOD 400 = 0))
            Then
                max_day1 <= "1001";
            Else
                max_day1 <= "1000";
            End If;
        End If;

    End If;
End Process;
```

ΠΑΡΑΡΤΗΜΑ ΙΙ Κώδικας ρύθμισης και ενεργοποίησης λειτουργίας αφύπνισης

```
Process(clock)
Begin
    If rising_edge(clock) Then
        If mode = "010" Then
            If picker = "00" Then --Ρύθμιση λεπτών
                If plus = '0' Then
                    Count_m1 <= Count_m1 + 1;
                    If (Count_m1 = "1001") Then
                        Count_m1 <= "0000";
                        Count_m2 <= Count_m2 + 1;
                        If (Count_m2 = "0101") Then
                            Count_m2 <= "0000";
                        End If;
                    End If;
                Elsif minus = '0' Then
                    Count_m1 <= Count_m1 - 1;
                    If (Count_m1 = "0000") Then
                        Count_m1 <= "1001";
                        Count_m2 <= Count_m2 - 1;
                        If (Count_m2 = "0000") Then
                            Count_m2 <= "0101";
                        End If;
                    End If;
                End If;

            Elsif picker = "01" Then --Ρύθμιση ώρας
                If plus = '0' Then
                    Count_h1 <= Count_h1 + 1;
                    If (Count_h1 = "1001") Then
                        Count_h1 <= "0000";
                        Count_h2 <= Count_h2 + 1;
                    End If;
                    If (Count_h2 = "0010" AND Count_h1 = "0011") Then
                        Count_h1 <= "0000";
                        Count_h2 <= "0000";
                    End If;

                Elsif minus = '0' Then
                    Count_h1 <= Count_h1 - 1;
                    If (Count_h1 = "0000") Then
                        Count_h1 <= "1001";
                        Count_h2 <= Count_h2 - 1;
                    End If;
                    If (Count_h2 = "0000" AND Count_h1 = "0000") Then
                        Count_h1 <= "0011";
                        Count_h2 <= "0010";
                    End If;
                End If;
            End If;
        End If;

        -- Έλεγχος και ενεργοποίηση συναγερμού όταν η ώρα του ρολογιού
ταιριάζει με την προγραμματισμένη ώρα
        If (Count_m1 = Digit_m1 AND Count_m2 = Digit_m2 AND Count_h1 =
Digit_h1 AND Count_h2 = Digit_h2 AND SW_Alarm = '1') Then
            blink_counter <= blink_counter + 1;
```

```
            alarm_active  <= '1';
        End If;

        If alarm_active = '1' Then
            blink_counter <= blink_counter + 1;
            If blink_counter = 2 Then
                leds_int      <= not leds_int;
                blink_counter <= 0;
            End If;
        End If;

        --Απενεργοποίηση από τον χρήστη
        If SW_Alarm = '0' Then
            alarm_active  <= '0';
            leds_int      <= "0000";
            blink_counter <= 0;
        End If;
    End If;
End Process;
```

ΠΑΡΑΡΤΗΜΑ ΙΙΙ Κώδικας μετατροπής 24ωρης σε 12ωρη μορφή

```
Process (Digin_h1, Digin_h2, mode_12_24, Date)
Begin

    If mode_12_24 = '1' AND Date = '0' Then

        -- Ελέγχουμε αν η ώρα είναι από 13:00 μέχρι 19:59
        If Digin_h2 = "0001" AND Digin_h1 > "0010" Then
            Digout_h1 <= Digin_h1 - "0010";
            Digout_h2 <= Digin_h2 - "0001";
            LED_AM    <= '1';
            LED_PM    <= '1';
        -- Ελέγχουμε αν η ώρα είναι από 20:00 μέχρι 21:59
        Elsif Digin_h2 = "0010" AND Digin_h1 < "0010" Then
            Digout_h1 <= Digin_h1 + "1000";
            Digout_h2 <= Digin_h2 - "0010";
            LED_AM    <= '1';
            LED_PM    <= '1';
        -- Ελέγχουμε αν η ώρα είναι από 22:00 μέχρι 23:59
        Elsif Digin_h2 = "0010" AND Digin_h1 > "0001" Then
            Digout_h1 <= Digin_h1 - "0010";
            Digout_h2 <= Digin_h2 - "0001";
            LED_AM    <= '1';
            LED_PM    <= '1';
        -- Ελέγχουμε αν η ώρα είναι από 00:00 μέχρι 00:59
        Elsif (Digin_h2 = "0000" And Digin_h1 = "0000") Then
            Digout_h1 <= "0010";
            Digout_h2 <= "0001";
            LED_AM    <= '1';
            LED_PM    <= '0';

        Else
            -- Καμία ενέργεια
            Digout_h1 <= Digin_h1;
            Digout_h2 <= Digin_h2;
            -- Έλεγχος για πμ/μμ και ενεργοποίηση των led
            If (Digin_h2 = "0001" And Digin_h1 = "0010") Then
                LED_AM <= '1';
                LED_PM <= '1';
            Else
                LED_AM <= '1';
                LED_PM <= '0';

            End if;
        End If;

    Else
        -- Επιλογή 24 καμία ενέργεια
        Digout_h1 <= Digin_h1;
        Digout_h2 <= Digin_h2;
        LED_AM    <= '0';
        LED_PM    <= '0';
    End If;
    -- Έλεγχος νυχτερινής λειτουργίας
    If Digin_h2 = "0000" AND Digin_h1 > "0110" Then
```

```
        Night_Mode <= '0';
    Elsif Digin_h2 = "0001" AND Digin_h1 < "1001" Then
        Night_Mode <= '0';
    Else
        Night_Mode <= '1';
    End If;
End Process;
```

ΠΑΡΑΡΤΗΜΑ IV Κώδικας χρονομέτρου

```
Architecture Behavior Of StopWatch Is

    Signal Count_00s1 : STD_LOGIC_vector(3 Downto 0);
    Signal Count_00s2 : STD_LOGIC_vector(3 Downto 0);
    Signal Count_s1   : STD_LOGIC_vector(3 Downto 0);
    Signal Count_s2   : STD_LOGIC_vector(3 Downto 0);
    Signal Count_m1   : STD_LOGIC_vector(3 Downto 0);
    Signal Count_m2   : STD_LOGIC_vector(3 Downto 0);
    Signal Count_h1   : STD_LOGIC_vector(3 Downto 0);
    Signal Count_h2   : STD_LOGIC_vector(3 Downto 0);

Begin


Process (Clock02, mode, Start, Reset_StW)
Begin

If Rising_Edge(Clock02) Then

    If Reset_StW = '1' Then
        Count_00s1 <= "0000";
        Count_00s2 <= "0000";
        Count_s1   <= "0000";
        Count_s2   <= "0000";
        Count_m1   <= "0000";
        Count_m2   <= "0000";
        Count_h1   <= "0000";
        Count_h2   <= "0000";

    ElsIf Start = '1' Then
        Count_00s1 <= Count_00s1 + 1;
        If (Count_00s1 = "1001") Then
            Count_00s1 <= "0000";
            Count_00s2 <= Count_00s2 + 1;
            If (Count_00s2 = "1001") Then
                Count_00s2 <= "0000";
                Count_s1   <= Count_s1 + 1;
                If (Count_s1 = "1001") Then
                    Count_s1 <= "0000";
                    Count_s2 <= Count_s2 + 1;
                    If (Count_s2 = "0101") Then
                        Count_s2 <= "0000";
                        Count_m1 <= Count_m1 + 1;
                        If (Count_m1 = "1001") Then
                            Count_m1 <= "0000";
                            Count_m2 <= Count_m2 + 1;
                            If (Count_m2 = "0101") Then
                                Count_m2 <= "0000";
                                Count_h1 <= Count_h1 + 1;
                                If Count_h1 = "1001" Then
                                    Count_h1 <= "0000";
                                    Count_h2 <= Count_h2 + 1;
                                End If;
                    If (Count_h2 = "1001" AND Count_h1 =    "1001") Then
                                    Count_00s1 <= "0000";
                                    Count_00s2 <= "0000";
                                    Count_s1   <= "0000";
                                    Count_s2   <= "0000";
```

```
                                        Count_m1    <= "0000";
                                        Count_m2    <= "0000";
                                        Count_h1    <= "0000";
                                        Count_h2    <= "0000";
                                    End If;
                                End If;
                            End If;
                        End If;
                    End If;
                End If;
            End If;
        End If;
    End If;
End If;
End Process;
-- Κάτω από μία ώρα εμφανίζει λλ:δδ:εκ, πάνω από μία ώρα εμφανίζει ωω:λλ:δδ
StW_Digit_1 <= Count_00s1 when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_s1;
StW_Digit_2 <= Count_00s2 when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_s2;
StW_Digit_3 <= Count_s1   when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_m1;
StW_Digit_4 <= Count_s2   when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_m2;
StW_Digit_5 <= Count_m1   when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_h1;
StW_Digit_6 <= Count_m2   when (Count_h2 = "0000" AND Count_h1 = "0000")
else Count_h2;
End Behavior;
```

ΠΑΡΑΡΤΗΜΑ V Κώδικας αντίστροφης μέτρησης

```
Process (Clock, mode)
Begin

if Rising_Edge(Clock) Then

    if mode = "101" then
        led <= '0';
    end if;

    If Set = '1' Then
        -- Εισαγωγή αρχικής τιμής για αντίστροφη μέτρηση
        Count_s1 <= Dig_in_s1;
        Count_s2 <= Dig_in_s2;
        Count_m1 <= Dig_in_m1;
        Count_m2 <= Dig_in_m2;
        Count_h1 <= Dig_in_h1;
        Count_h2 <= Dig_in_h2;

    ElsIf Set = '0' Then
        If Start = '1' Then
            -- Αντίστροφη μέτρηση
            If Count_s1 = "0000" Then
                Count_s1 <= "1001"; -- Επαναφορά σε 9
                If Count_s2 = "0000" Then
                    Count_s2 <= "0101"; -- Επαναφορά σε 5
                    If Count_m1 = "0000" Then
                        Count_m1 <= "1001"; -- Επαναφορά σε 9
                        If Count_m2 = "0000" Then
                            Count_m2 <= "0101"; -- Επαναφορά σε 5
                            If Count_h1 = "0000" Then
                                Count_h1 <= "1001"; -- Επαναφορά σε 9
                                If Count_h2 = "0000" Then
                                   -- Τερματισμός όταν όλα τα ψηφία είναι 0
                                    Count_s1 <= "0000";
                                    Count_s2 <= "0000";
                                    Count_m1 <= "0000";
                                    Count_m2 <= "0000";
                                    Count_h1 <= "0000";
                                    Count_h2 <= "0000";
-- Ενεργοποίηση ειδοποίησης, απενεργοποίηση όταν μπει σε λειτουργία
ρύθμισης
                                    if mode = "101" then
                                        led      <= '0';
                                    else led <= '1';
                                    end if;

                                Else
                                    Count_h2 <= Count_h2 - 1;
                                End If;
                            Else
                                Count_h1 <= Count_h1 - 1;
                            End If;
                        Else
                            Count_m2 <= Count_m2 - 1;
                        End If;
                    Else
                        Count_m1 <= Count_m1 - 1;
                    End If;
```

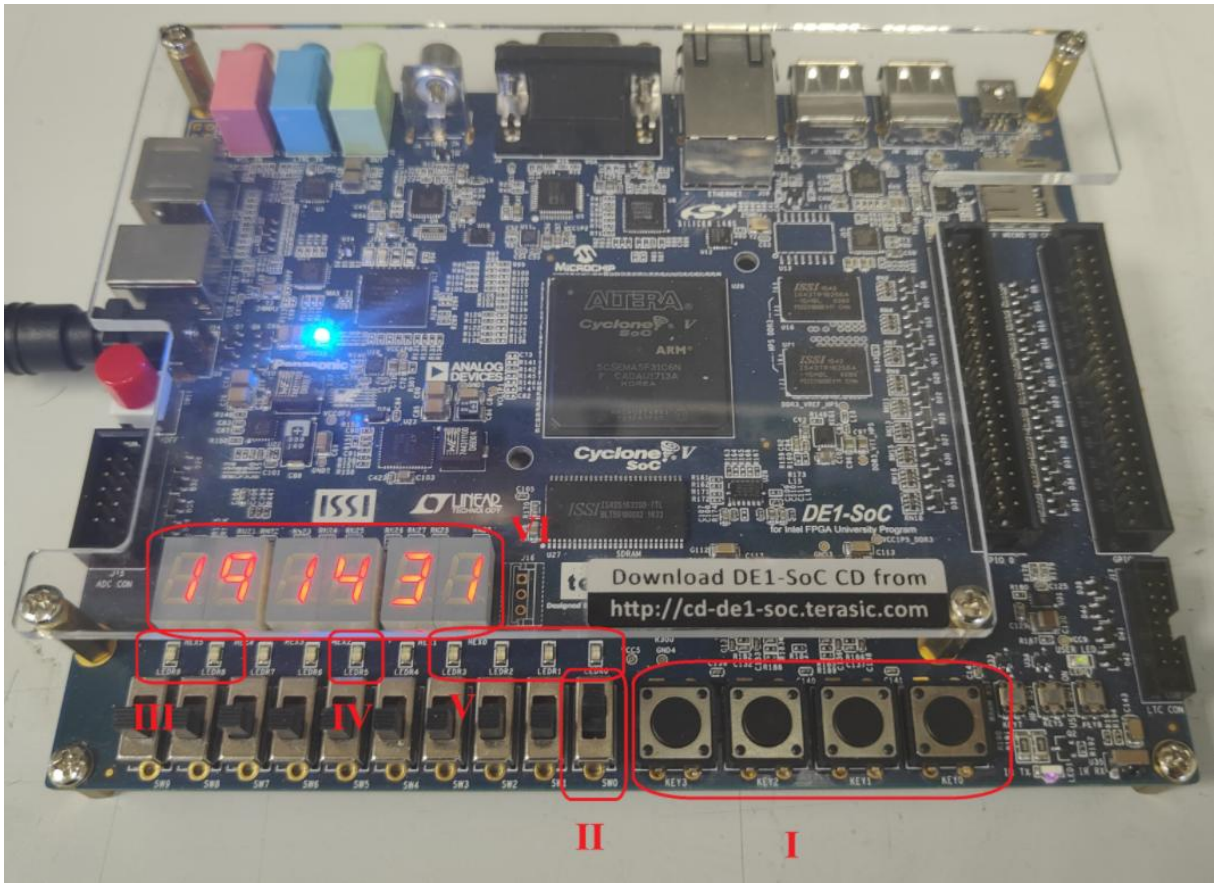
```
                Else
                    Count_s2 <= Count_s2 - 1;
                End If;
            Else
                Count_s1 <= Count_s1 - 1;
            End If;
        End If;
    End If;
End If;
End Process;
```

ΠΑΡΑΡΤΗΜΑ VI Κώδικας εξοικονόμησης ενέργειας – νυχτερινής λειτουργίας

```
Process(clk50)
Begin
    If rising_edge(clk50) Then
        -- Αύξηση του μετρητή PWM
        If pwm_counter = 99 Then
            pwm_counter <= 0;
        Else
            pwm_counter <= pwm_counter + 1;
        End If;

        -- Ενεργοποίηση PWM όταν το Night_Mode είναι ενεργό
        If pwm_counter < 10 And Night_Mode = '1' Then -- 10% φωτεινότητα
            pwm_enable <= '1';
        Elsif Night_Mode = '0' OR mode = "011" Then
            pwm_enable <= '1';
        Else
            pwm_enable <= '0';
        End If;
    End If;
End Process;


Process(Clock)
Begin
    If rising_edge(clock) Then
        -- Έλεγχος αν είναι σε λειτουργία ρολογιού και ο χρήστης έχει
ενεργοποιήσει την εξοικονόμηση ενέργειας
        If mode = "001" And Low_Power_Mode = '1' Then
            If LPM_counter = 99 Then
                LPM_counter <= 0;
            Else
                LPM_counter <= LPM_counter + 1;
            End If;

            If LPM_counter < 50 Then
                LPM_enable <= '1';
            Else
                LPM_enable <= '0';
            End If;
        Else
            LPM_enable <= '0';
        End If;
    End if;
End Process;
```

ΠΑΡΑΡΤΗΜΑ VII Υλοποίηση συστήματος

Στο παρόν παράρτημα παρουσιάζεται μέσω μιας σειράς εικόνων η λειτουργία του ψηφιακού ρολογιού που υλοποιήθηκε. Ειδικότερα, περιλαμβάνονται εικόνες που αποτυπώνουν την ένδειξη της ώρας, τη ρύθμιση των παραμέτρων του συστήματος, καθώς και τη λειτουργία πρόσθετων χαρακτηριστικών, όπως η αφύπνιση και η εναλλαγή μεταξύ 12ωρης και 24ωρης μορφής, η αντίστροφη μέτρηση και η νυχτερινή λειτουργία.



Εικόνα 1: Είσοδοι/έξοδοι του συστήματος.

Στην Εικόνα 1 απεικονίζεται η αναπτυξιακή πλακέτα DE1-SoC, όπου βλεπουμε:

I. Τα πλήκτρα KEY3, KEY2, KEY1, KEY0, από αριστερά προς τα δεξιά, μέσω των οποίων γίνεται η εναλλαγή μεταξύ των καταστάσεων του συστήματος, καθώς και οι ανάλογες ρυθμίσεις.

II. Τον διακόπτη της λειτουργίας αφύπνισης.

III. Τα δύο LED ένδειξης π.μ./μ.μ όταν ο χρήστης επιλέξει 12ωρης μορφή ώρας.

IV. Το LED ειδοποίησης όταν ολοκληρωθεί ο χρόνος της αντίστροφης μέτρησης.

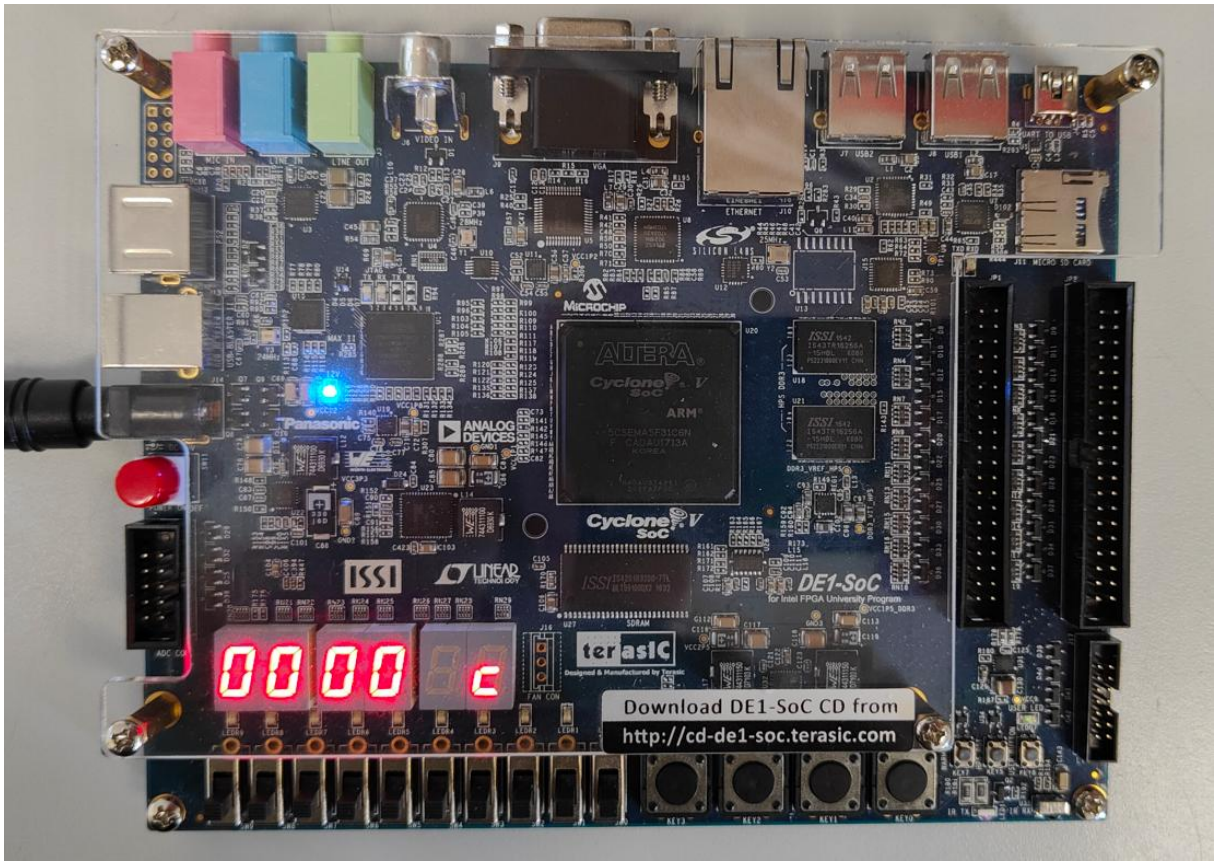
V. Τα τέσσερα LED ειδοποίησης της λειτουργίας αφύπνισης.

VI. Τους έξι ενδείκτες επτά τομέων.

Στην Εικόνα 2 απεικονίζεται το σύστημα σε κατάσταση ρύθμισης ώρας και ημερομηνίας. Η κατάσταση ρύθμισης ενεργοποιείται αυτόματα κατά την εκκίνηση του συστήματος, ενώ στη συνέχεια μπορεί να επανεκκινηθεί μέσω παρατεταμένου πατήματος του KEY0 από τον χρήστη όταν το σύστημα βρίσκεται σε κατάσταση εμφάνισης ρολογιού.

Όταν το σύστημα βρίσκεται σε κατάσταση ρύθμισης, στον δεξιά ενδείκτη δεξιά εμφανίζεται το γράμμα “c” προκειμένου ο χρήστης να αντιλαμβάνεται ότι βρίσκεται στη διαδικασία διαμόρφωσης

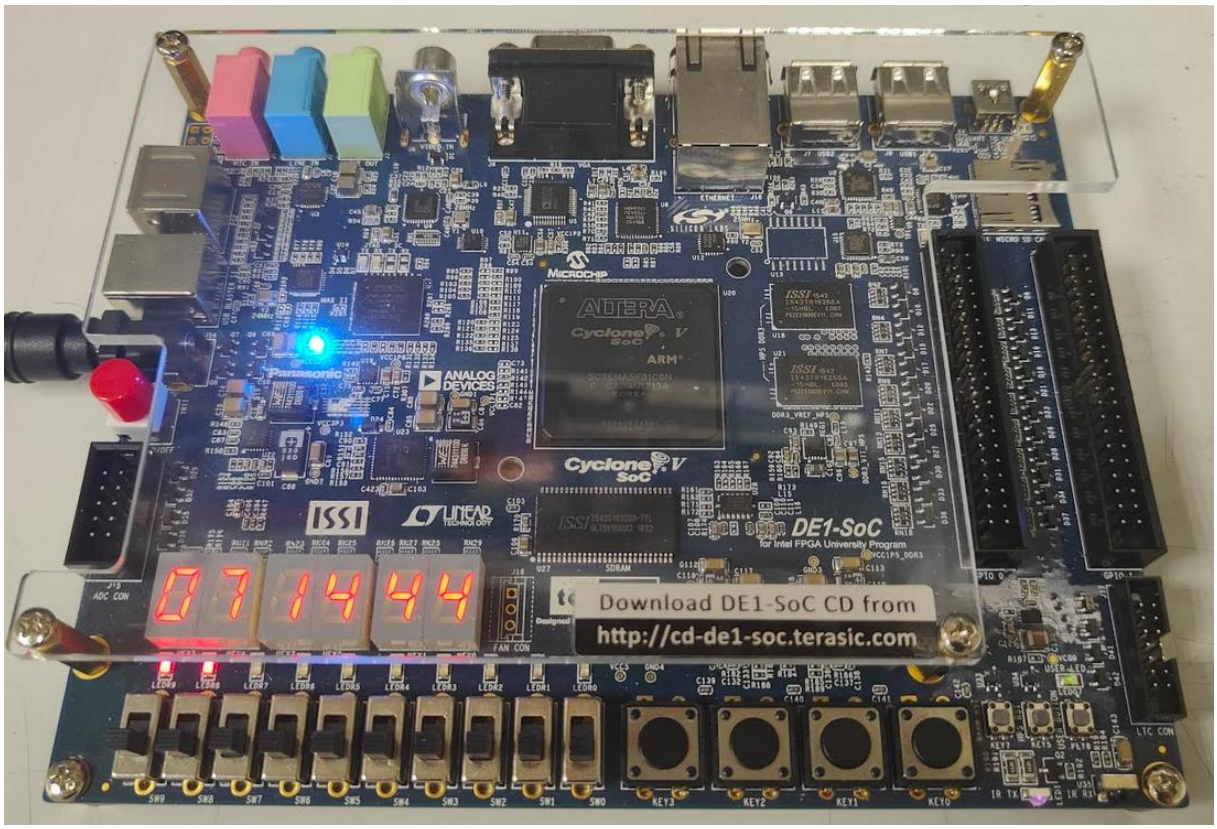
ώρας. Η μετάβαση μεταξύ των μονάδων του χρόνου (κατά σειρά: έτος, μήνας, ημέρα, ώρα, λεπτά) πραγματοποιείται με στιγμιαίο πάτημα του πλήκτρου KEY0. Οι αντίστοιχες μονάδες που επιλέγονται αναβοσβήνουν, υποδεικνύοντας ότι βρίσκονται σε λειτουργία επεξεργασίας. Η αύξηση ή μείωση των τιμών των επιλεγμένων μονάδων γίνεται μέσω των πλήκτρων KEY2 για αύξηση και KEY1 για μείωση. Με το πάτημα του KEY3, οι νέες τιμές αποθηκεύονται και το σύστημα εξέρχεται από τη λειτουργία ρύθμισης, εισερχόμενο στην κανονική κατάσταση εμφάνισης της ώρας.



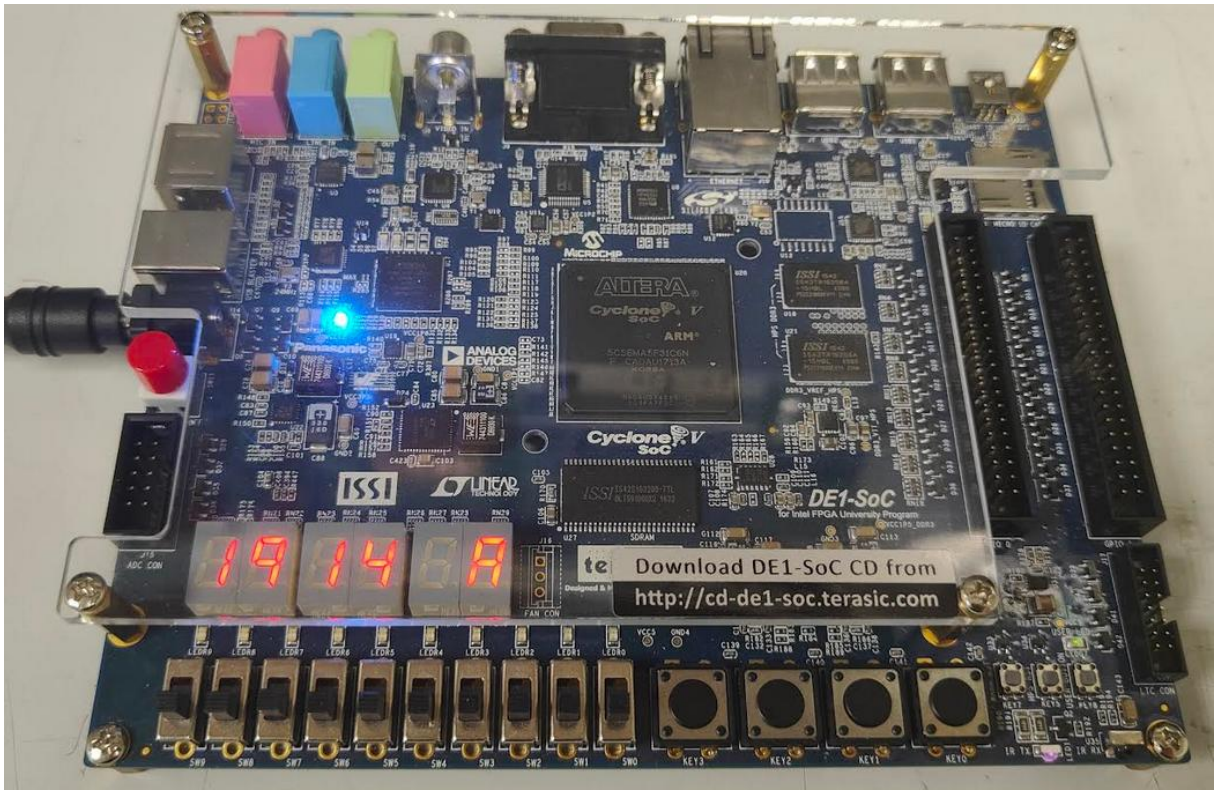
Εικόνα 2: Ρύθμιση ώρας.

Όταν το σύστημα βρίσκεται σε κατάσταση εμφάνισης ώρας, ο χρήστης μπορεί να επιλέξει ανάμεσα σε 12ωρη ή 24ωρη μορφή μέσω του πλήκτρου KEY2. Όπως φαίνεται στην Εικόνα 3, στην περίπτωση της 12ωρης μορφής, κατά τις ώρες 12.00 με 23.59 ανάβουν δύο LED για να δηλώσουν ότι είναι μ.μ., ενώ κατά τις ώρες 00.00 με 11.59 ανάβει μόνο ένα LED για να δηλώσει ότι είναι π.μ. Στην περίπτωση που επιλεγεί η 24ωρη μορφή, όπως στην Εικόνα 1, δεν ανάβει κανένα LED, καθώς η ώρα εμφανίζεται άμεσα σε 24ωρη αναπαράσταση χωρίς την ανάγκη διαχωρισμού μεταξύ π.μ. και μ.μ. Επίσης υπάρχει η δυνατότητα λειτουργίας εξοικονόμησης ενέργειας μέσω του πλήκτρου KEY1, όπου εμφανίζεται η ώρα κάθε λεπτό για δέκα μόνο δευτερόλεπτα.

Η ρύθμιση της ώρας αφύπνισης πραγματοποιείται από την κατάσταση εμφάνισης ώρας μέσω παρατεταμένου πατήματος του πλήκτρου KEY3. Η διαδικασία ρύθμισης ακολουθεί την ίδια λογική με αυτήν της ρύθμισης της ώρας, με τη μετάβαση μεταξύ των μονάδων του χρόνου να γίνεται μέσω του KEY0 και η τροποποίηση των τιμών μέσω των KEY2 και KEY1. Ωστόσο, κατά τη ρύθμιση της αφύπνισης (Εικόνα 4), στον τέρμα δεξιά ενδείκτη εμφανίζεται το γράμμα “A”, υποδεικνύοντας ότι ο χρήστης βρίσκεται στη συγκεκριμένη λειτουργία. Με το πάτημα του KEY3, οι νέες τιμές αποθηκεύονται και το σύστημα εξέρχεται από τη λειτουργία ρύθμισης της αφύπνισης και επιστρέφει στην κανονική κατάσταση εμφάνισης της ώρας.

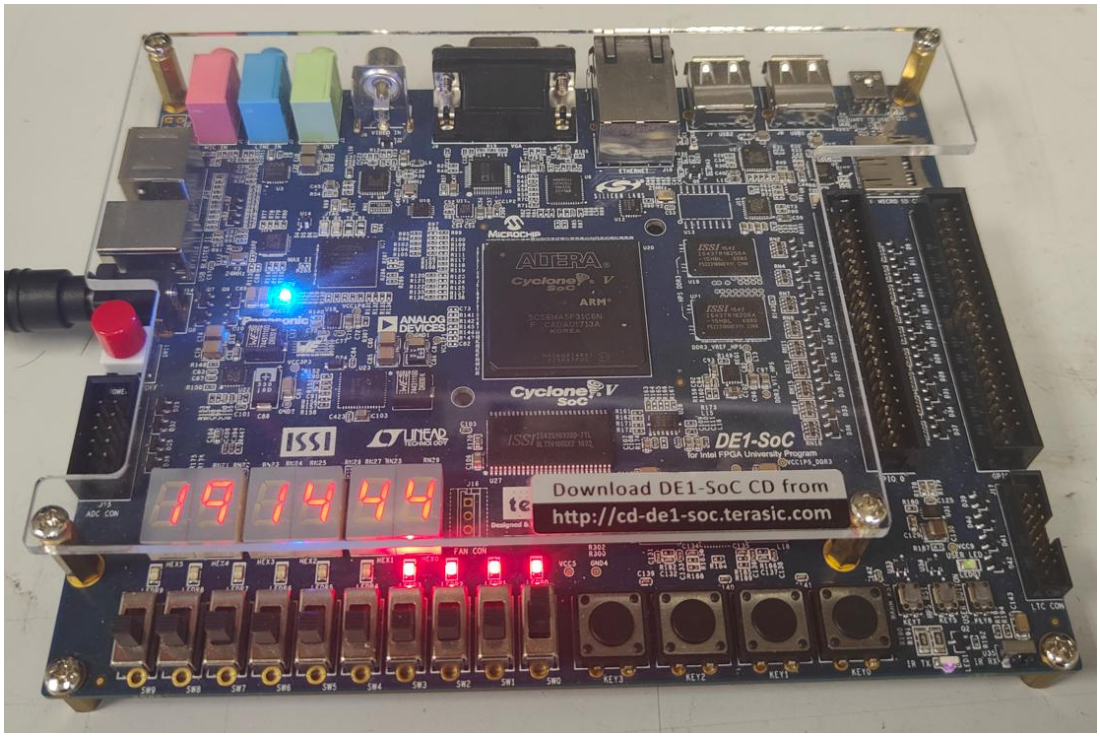


Εικόνα 3: 12ωρη απεικόνιση ώρας.



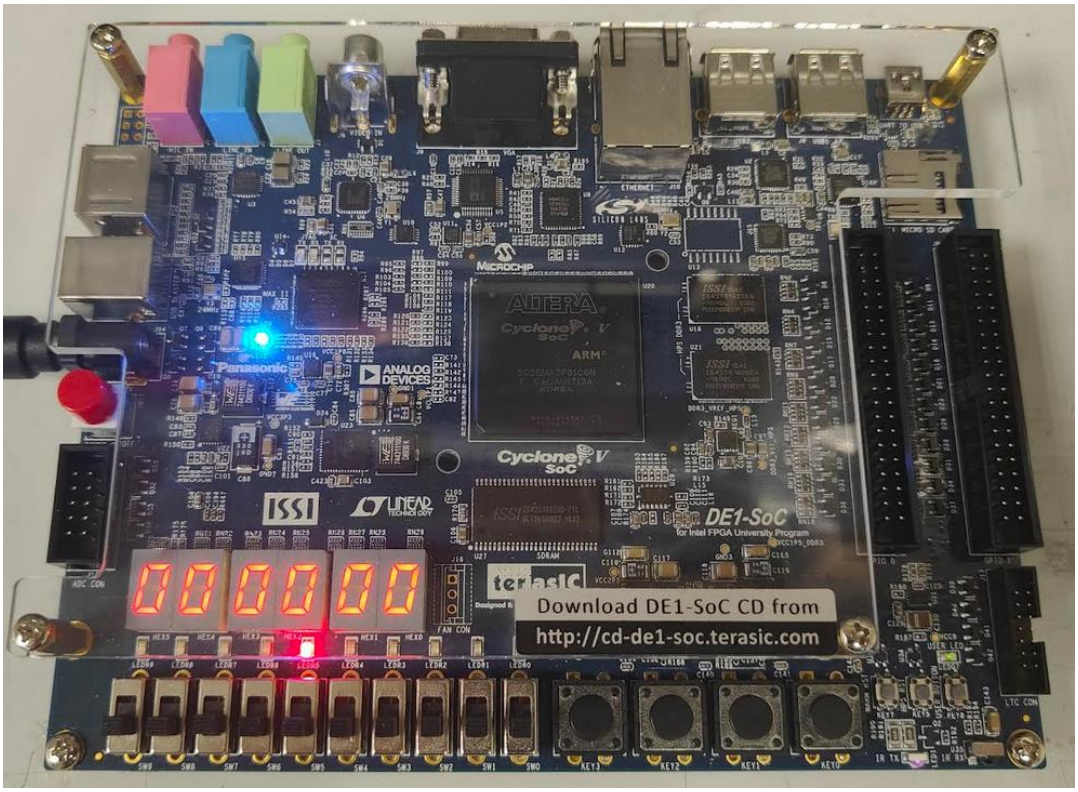
Εικόνα 4: Ρύθμιση λειτουργίας αφύπνισης.

Η ενεργοποίηση και απενεργοποίηση της λειτουργίας αφύπνισης πραγματοποιείται μέσω του διακόπτη αφύπνισης, επιτρέποντας στον χρήστη να ελέγχει αν η αφύπνιση είναι ενεργή ή ανενεργή. Όταν η προκαθορισμένη ώρα αφύπνισης επιτευχθεί, το σύστημα ενεργοποιεί μια οπτική ένδειξη, αναβοσβήνοντας ρυθμικά τέσσερα LED, όπως φαίνεται στην Εικόνα 4.



Εικόνα 4: Οπτική ένδειξη αφύπνισης.

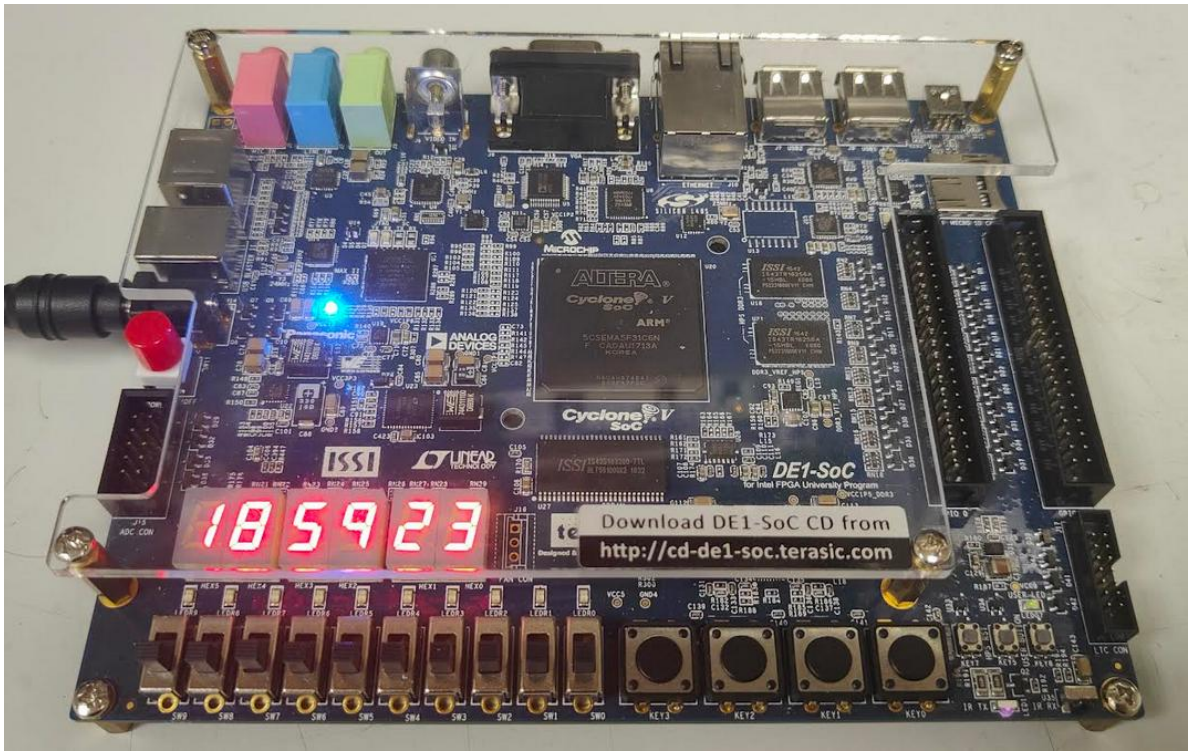
Η μετάβαση από την κατάσταση εμφάνισης ώρας σε χρονόμετρο γίνεται μέσω του KEY3, η έναρξη και παύση του χρονομέτρου γίνεται με το πλήκτρο KEY2 και ο μηδενισμός με το KEY0.



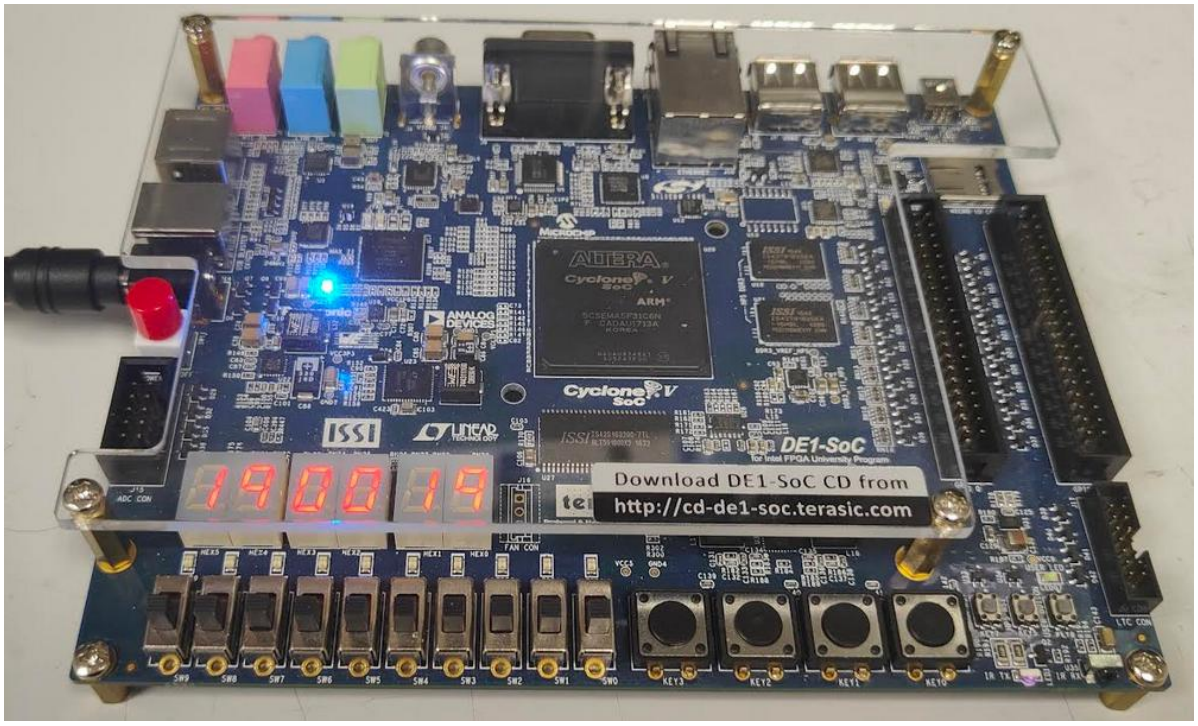
Εικόνα 5: Οπτική ένδειξη τέλους αντίστροφης μέτρησης.

Η μετάβαση από το χρονόμετρο στην αντίστροφη μέτρηση γίνεται επίσης μέσω του KEY3. Η ρύθμιση της επιθυμητής διάρκειας αντίστροφης μέτρησης ακολουθεί την ίδια διαδικασία με αυτήν της ρύθμισης ώρας, με τη διαφορά ότι στην αντίστροφη μέτρηση δίνεται η δυνατότητα ρύθμισης της ώρας, των λεπτών και των δευτερολέπτων.

Αφού ολοκληρωθεί η ρύθμιση, οι τιμές αποθηκεύονται πατώντας το KEY3. Η εκκίνηση και παύση της αντίστροφης μέτρησης πραγματοποιείται με το KEY2, ενώ όταν ο χρόνος τελειώσει, το σύστημα ειδοποιεί τον χρήστη με μέσω LED όπως φαίνεται στην Εικόνα 5.

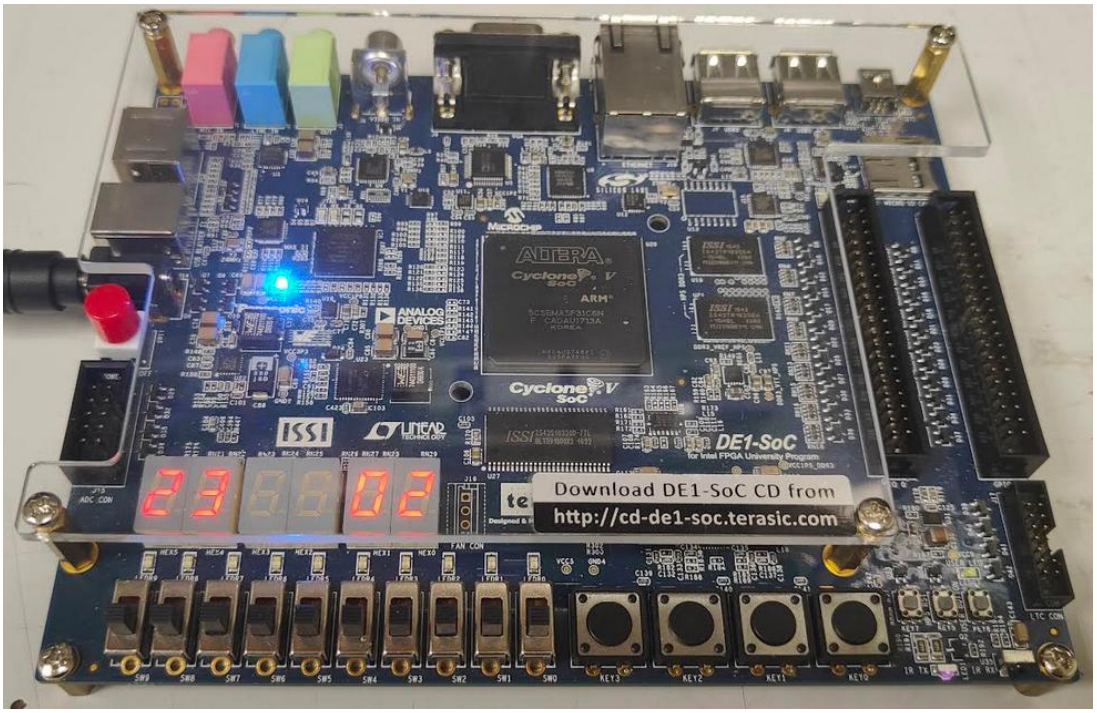


Εικόνα 6: Κανονική λειτουργία.

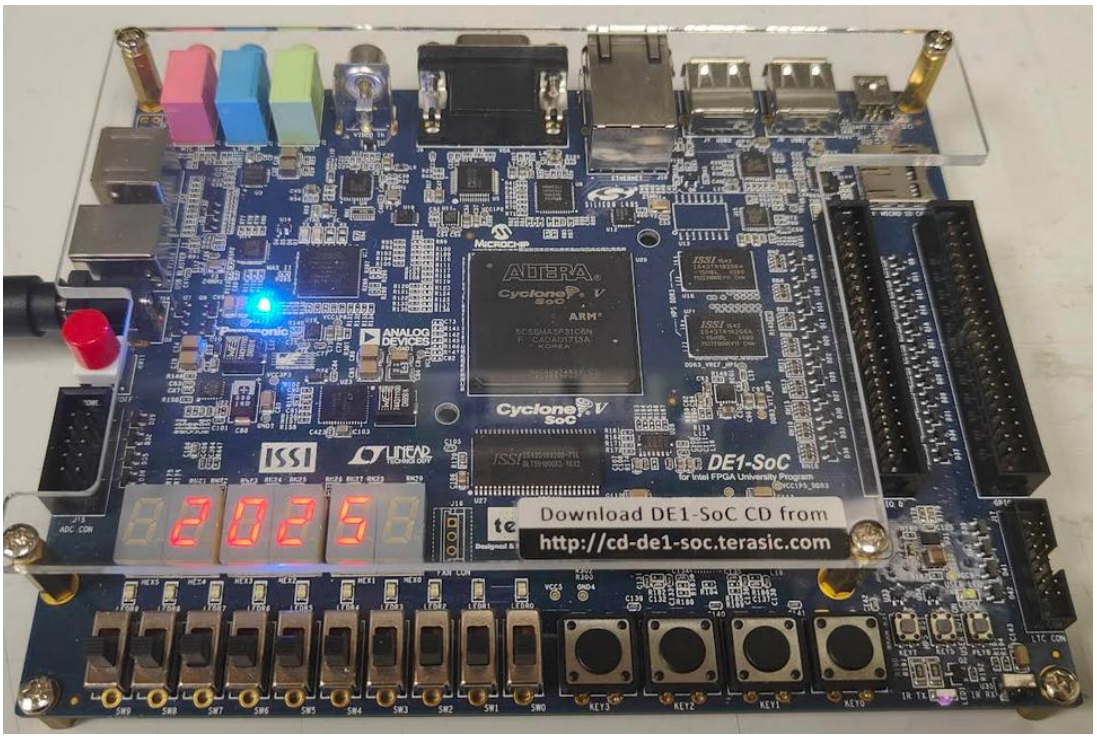


Εικόνα 7: Νυχτερινή λειτουργία.

Στην Εικόνα 6 και στην Εικόνα 7 απεικονίζεται η διαφορά στη φωτεινότητα των LED, λόγω της νυχτερινής λειτουργίας του συστήματος. Κατά τις ώρες 19.00 με 06.59, που ενεργοποιείται η νυχτερινή λειτουργία, η φωτεινότητα των LED μειώνεται αυτόματα για να προσαρμοστεί στις συνθήκες χαμηλού φωτισμού.

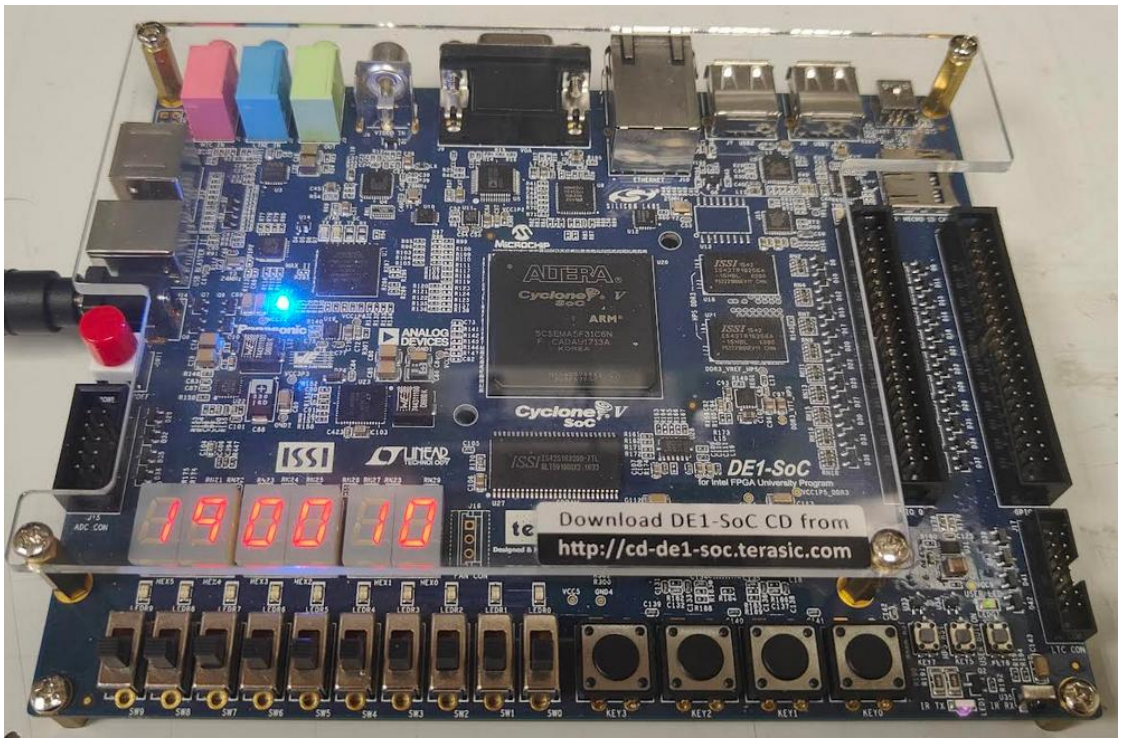


Εικόνα 8: Εμφάνιση ημέρας και μήνα.



Εικόνα 9: Εμφάνιση έτους.

Όταν αλλάζει η ώρα στα πρώτα πέντε δευτερόλεπτα εμφανίζεται η ημέρα και ο μήνας, όπως φαίνεται στην Εικόνα 8, στα επόμενα πέντε δευτερόλεπτα εμφανίζεται το έτος, όπως απεικονίζεται στην Εικόνα 9. Τέλος, στην Εικόνα 10 φαίνεται η ώρα, η οποία επανέρχεται μετά από συνολικά δέκα δευτερόλεπτα, επιστρέφοντας το σύστημα στην κανονική του λειτουργία εμφάνισης της ώρας.



Εικόνα 10: Εμφάνιση ώρας μετά τα δέκα πρώτα δευτερόλεπτα της ώρας.