# PROIECTAREA UNEI UNITATI ARITMETICE DE TIP MMX

Student: Petrenciuc Amelia-Andreea

Structura Sistemelor de Calcul

Universiatea Tehnica Cluj-Napoca

2025

1. **Introduction** 
   1. Context . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   2. Obiective . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
2. **Studiu Bibliografic**
   1. Ce inseamna MMX si la ce se foloseste? . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   2. Caracteristici principale . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   3. Saturare si intoarcere . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
3. **Analiza**
   1. Propunere proiect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   2. Analiza proiect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
4. **Design…………………………………………………………………………………… 6**
5. **Implementare………………………………………………………………………… 7**
6. **Testare si validare………………………………………………………………………. 9**
7. **Concluzii…………………………………………………………………………………….. 13**

**8 Bibliografie………………………………………………………………………………………………..13**

# Introducere

## 1.1 Context

## Proiectul propus are ca obiectiv proiectarea, implementarea și testarea a 6 operațiuni pe unitatea aritmetică, utilizând instrucțiuni MMX x86 pe 64 de biți. Instrucțiunile MMX sunt un set extins de instrucțiuni SIMD (Single Instruction, Multiple Data) care permit procesorului x86 să efectueze operații eficiente asupra datelor împachetate, cum ar fi byte-uri, word-uri, doubleword-uri și quadword-uri. Aceste instrucțiuni sunt esențiale pentru aplicațiile multimedia care necesită procesare rapidă și simultană a mai multor seturi de date.

## Proiectul va implica două componente principale:

## Design Hardware:

## Dezvoltarea unei unități aritmetice personalizate capabile să gestioneze instrucțiunile MMX x86.

## Integrarea unității cu o platformă FPGA pentru testarea funcționalității și a performanței.

## Dezvoltare Software:

## Crearea de testbenches pentru simulare și validare.

## Dezvoltarea de fragmente de cod software care să genereze fluxuri de instrucțiuni

## 1.2 Obiective

Proiectul are ca scop dezvoltarea și implementarea unei unități aritmetice MMX care să integreze instrucțiunile **paddd, psubd, psrld, pslld, pxor, pandn**. Această unitate va fi capabilă să efectueze operații aritmetice și logice pe date multimedia, conform standardului IEEE pe 64 de biți, oferind astfel performanță crescută în prelucrarea datelor și eficiență în aplicațiile de procesare paralelă.

Descriere instructiuni:

**L**

Aas m

1. PADDD: Adună un registru și un alt registru sau o locație de memorie ca întregi fără semn de 32 de biți (doublewords).
2. PSUBB: Scade un registru și un alt registru sau o locație de memorie ca întregi fără semn de 32 de biți (doublewords).
3. PANDN: Realizează o operație de negație logică urmată de un AND (NAND) între oricare două registre MMX, între un registru MMX și memoria principală sau între un registru MMX și o constantă.
4. PSRLD: Deplasează la dreapta un registru specificat cu un anumit număr de biți, operând asupra cuvintelor duble (32 de biți). Aceasta este o deplasare logică, nu aritmetică.
5. PSLLD: Deplasează la stanga un registru specificat cu un anumit număr de biți, operând asupra cuvintelor duble (32 de biți). Aceasta este o deplasare logică, nu aritmetică.
6. PXOR: Realizează o operație de exclusiv-sau (XOR) între oricare două registre MMX, între un registru MMX și memoria principală, sau între un registru MMX și o constantă.

# Studiu Bibliografic

**2.1 Ce inseamna MMX si la ce se foloseste?**

## Tehnologia MMX (Multi-Media eXtension) a fost dezvoltată pentru a îmbunătăți viteza de prelucrare a datelor multimedia, cum ar fi grafica, audio și video, printr-o abordare de procesare paralelă. Principalul obiectiv al acestei tehnologii este să permită aplicațiilor multimedia să funcționeze în timp real, acolo unde timpul de execuție este critic.

## Caracteristici principale

## SIMD (Single Instruction Multiple Data): Permite procesarea mai multor elemente de date în paralel, folosind o singură instrucțiune.

## Registre MMX: Există 8 registre MMX, fiecare de 64 de biți, denumite MM0 până la MM7. Acestea pot gestiona date de tip octet, cuvânt, dublu-cuvânt și cuadruplu cuvânt, împachetate într-un format de 64 de biți.

A screenshot of a test

Description automatically generated

## Instructiuni MMX: Include un set extins de 57 de instrucțiuni pentru operații precum adunarea, scăderea, multiplicarea, compararea, deplasarea logică și aritmetică, operații logice și împachetare/despachetare.

## 2.3 Saturare si intoarcere (wraparound)

Instrucțiunile MMX permit procesarea simultană a mai multor întregi printr-un singur apel de instrucțiune. Întregii cu semn sunt reprezentați în format \*\*complement față de doi\*\*, unde primul bit indică semnul (0 pentru numere pozitive, 1 pentru numere negative). Complementul unui număr se obține prin inversarea tuturor biților și adăugarea valorii 1 la rezultatul inversării.

În timpul efectuării operațiilor, unele rezultate pot depăși domeniul de reprezentare al tipului de date utilizat. Tehnologia MMX oferă două modalități de gestionare a acestor situații:

1.**Întoarcerea (Wraparound):** rezultatul este trunchiat, păstrând doar cei mai puțin semnificativi biți.

2. **Saturarea:** rezultatul care depășește domeniul permis este înlocuit cu valoarea maximă sau minimă corespunzătoare acelui tip de date.

A table with numbers and symbols

Description automatically generated

# Analiza

* 1. **Propunerea proiectului**

.

.

Dupa finalizarea proiectului, acesta ar trebui sa execute toate operatiile mentionate anterior, sa treaca cu success toate testele de verifcare si sa functioneze corect pe placa Basys 3. Vom folosi switch-urile, butoanele si afisajul cu 7 segmente de pe placuta pentru introducerea datelor si afisarea rezultatelor.

Instrucțiunile selectate sunt de tip R, adică instrucțiuni care operează pe conținutul registrelor. Operanzii implicați sunt stocați în registre, iar rezultatul obținut este, de regulă, salvat tot într-un registru.

* 1. **Analiza proiect**

In cadrul acestui proiect, vom analiza fiecare instructiune si vom stabili opcode-ul acestora, dar stabilim si semnalele care fac posibila calcularea datelor.

Instructiunile alese sunt de tip R, avand urmatorul format:

A close-up of a number

Description automatically generated

Descrierea instructiunilor alese:

* + PADDD:
    - * RTL: $d <- $s + $t; PC <- PC + 4;
      * Sintaxa: paddd $d, $s, $t
      * Format: 000000 sssss ttttt ddddd 00000 000000
* PSUBD:
  + - * RTL: $d <- $s - $t; PC <- PC + 4;
      * Sintaxa: psubd $d, $s, $t
      * Format: 000000 sssss ttttt ddddd 00000 000001
* PSRLD:
  + - * RTL: $d <- $s >> $t; PC <- PC + 4;
      * Sintaxa: psrld $d, $s, $t
      * Format: 000000 00000 sssss ddddd 00001 000010
* PSLLD:
  + - * RTL: $d <- $s << $t; PC <- PC + 4;
      * Sintaxa: pslld $d, $s, $t
      * Format: 000000 00000 sssss ddddd 00001 000011
* PXOR:
  + - * RTL: $d <- $s ^ $t; PC <- PC + 4;
      * Sintaxa: pxor $d, $s, $t
      * Format: 000000 sssss ttttt ddddd 00000 000100
* PANDN:
  + RTL: $d <- ~$s & $t; PC <- PC + 4;
    - * Sintaxa: pandn $d, $s, $t
      * Format: 000000 sssss ttttt ddddd 00000 000101

# Design

A table with black text

Description automatically generatedIn etapa de design am stabilit semnalele de control pe care le vom folosi in proiect pentru a efectua instructiunile mentionate.

Proiectul are un design asemănător arhitecturii MIPS32, incluzând aceleași componente, însă cu implementări mai simplificate datorită numărului redus de instrucțiuni. Principala modificare va fi la nivelul unității ALU, care va fi adaptată pentru a suporta instrucțiunile MMX ce operează pe seturi de biți.

Programul este compus din IF, aici se afla codul masina a fiecarei instructiuni si aicise va efectua trecerea la urmatoarea instructiune.

In ID, am stocat datele intr-un Register File. Aici se vor parsa bitii din instructiune pentru a stabili primul registru, respectiv al doilea registru. In UC se aleg semnalele pentru fiecare tip de instructiune.

In EX, se produc calculele efective al fiecarei instructiuni in parte.

A diagram of a computer

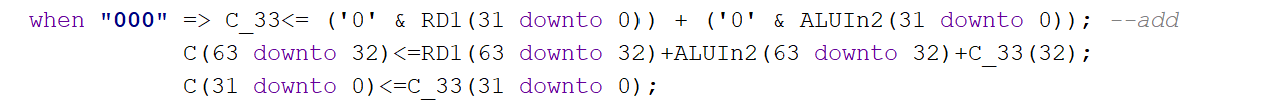
Description automatically generated

# Implementare

Deoarece am ales sa lucrez pe doublword, instructiunea de 64 de biti este impartial in doua parti.

**Instructiunea de adunare PADDD** s-a realizat astfel:

.



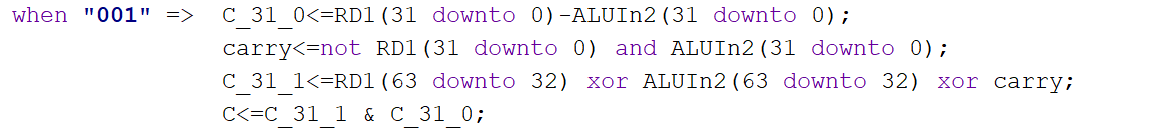
Se adună primii 32 de biți ai celor două numere (RD1 și ALUIn2). Pentru a permite gestionarea transportului, fiecare operand este extins de la 32 la 33 de biți prin adăugarea unui bit suplimentar 0 pe poziția cea mai semnificativă. Rezultatul acestei adunări este reținut în variabila C\_33, care include atât cei 32 de biți ai rezultatului, cât și bitul suplimentar de transport generat (al 33-lea bit).

Segmentul superior al celor două numere (ultimii 32 de biți) este adunat împreună cu bitul de transport generat de adunarea segmentului inferior (bitul 32 din C\_33). Aceasta asigură continuitatea adunării pe 64 de biți.

Segmentul inferior al rezultatului final (C(31 downto 0)) este luat direct din cei 32 de biți inferiori calculați în C\_33.

Segmentul superior (C(63 downto 32)) este calculat separat, incluzând transportul dintre segmente.

**Instructiunea PSUBD:**



Se realizează scăderea primilor 32 de biți ai celor două operanzi (RD1 și ALUIn2), iar rezultatul parțial este stocat în C\_31\_0. Se calculează împrumutul generat de scăderea segmentului inferior. Împrumutul apare atunci când o valoare din RD1 este mai mică decât valoarea corespunzătoare din ALUIn2. Segmentul superior al celor doi operanzi (RD1(63 downto 32) și ALUIn2(63 downto 32)) este scăzut, luând în considerare împrumutul calculat anterior (carry). Operația este realizată folosind operatorul xor, care implementează efectiv scăderea binară pe segmentul superior, cu ajustarea corespunzătoare pentru împrumut. ezultatul final pe 64 de biți este format prin concatenarea rezultatului segmentului superior (C\_31\_1) cu rezultatul segmentului inferior (C\_31\_0).

**Instructiunea PANDN si PXOR**



A black and blue text

Description automatically generated

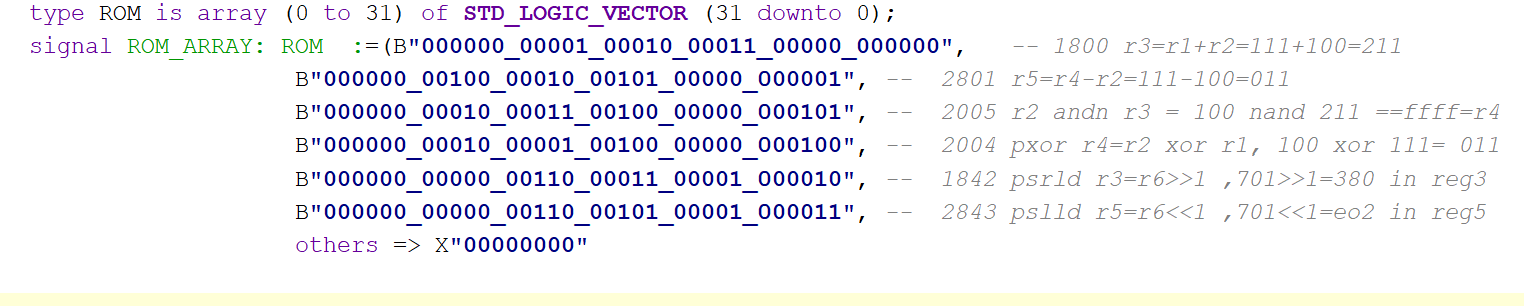
**Instructiunea PSRLD si PSLLD**

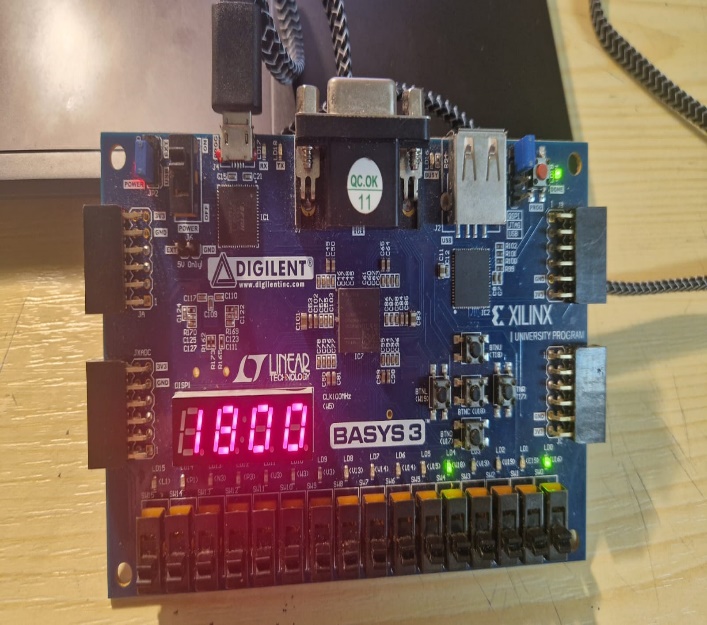
A screenshot of a computer program

Description automatically generated

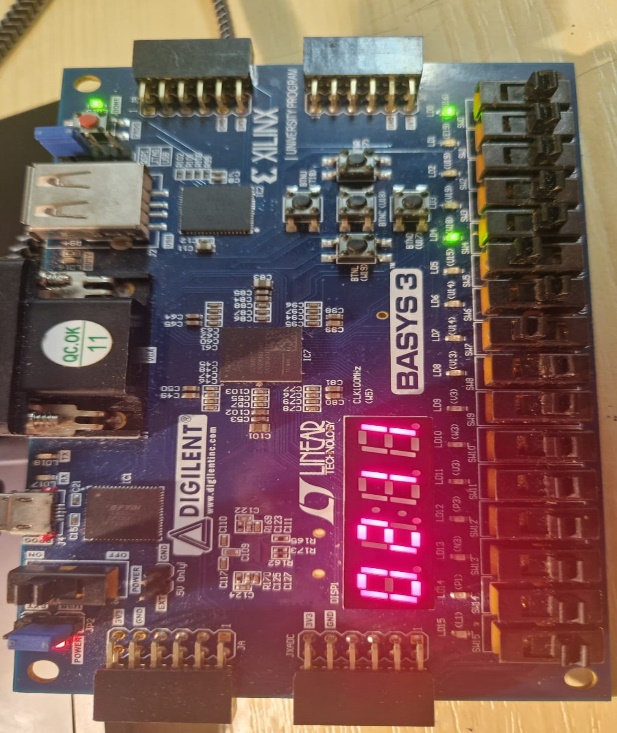
Se realizeaza shiftarea stanga/dreapta prin adaugarea de 0 in functie de valoarea din sa.

**Testare si Validare**

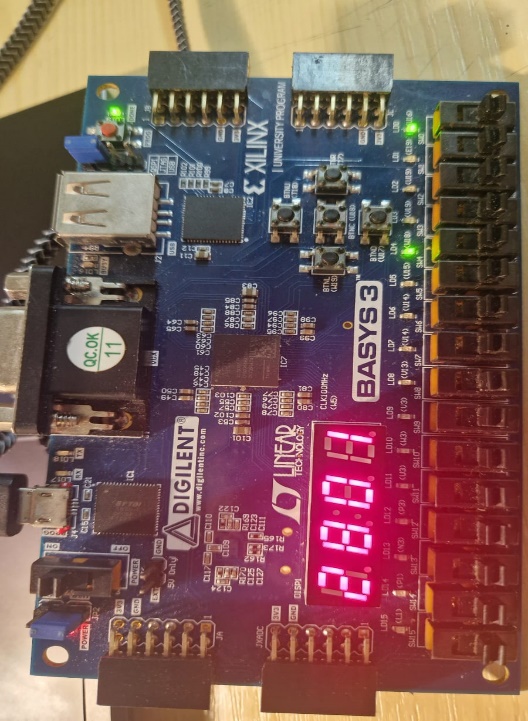


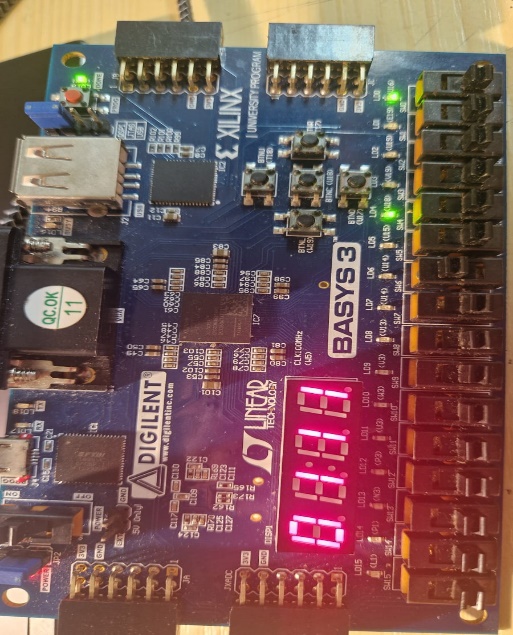
**A close up of a circuit board

Description automatically generatedPADDD**: R1

A close up of a circuit board

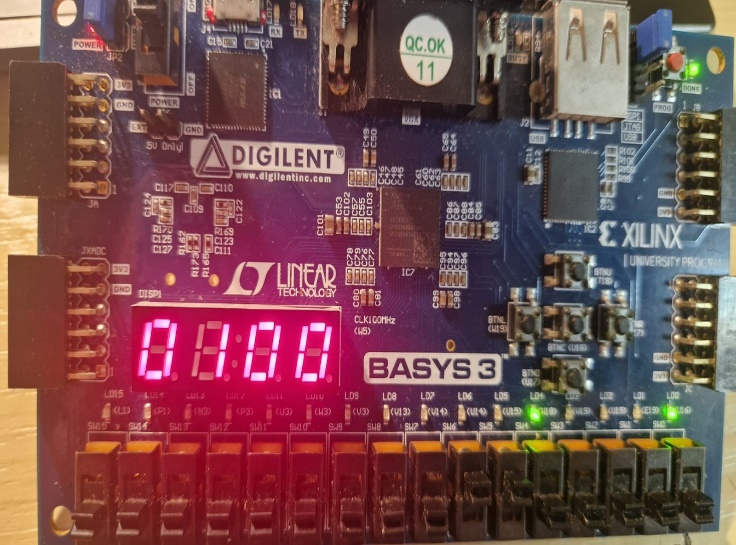
Description automatically generatedR2 REZ



**PSUBD**

R1

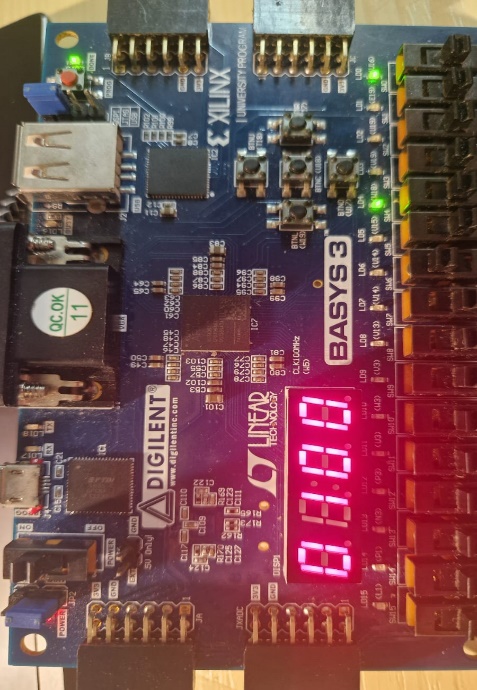
A close up of a circuit board

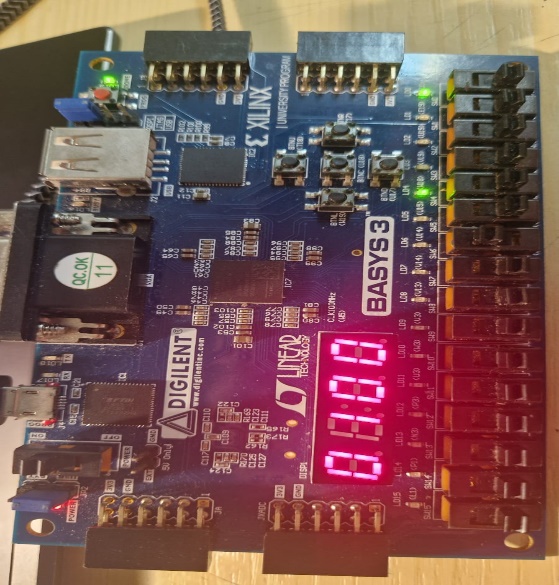
Description automatically generatedR2 REZ

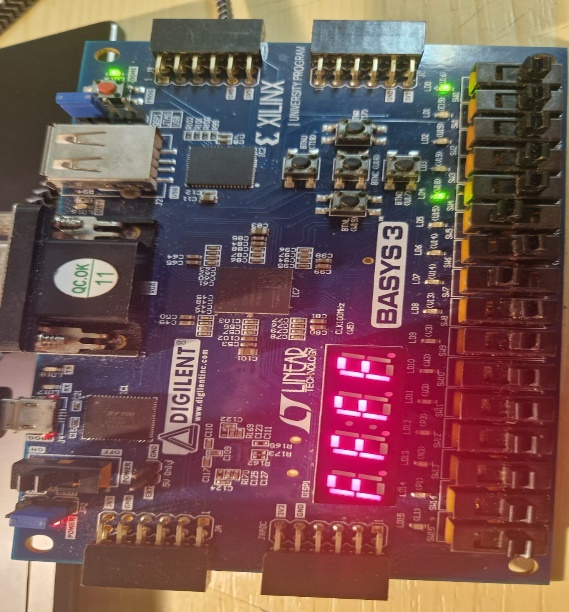
A close up of a circuit board

Description automatically generated

**PANDN**: R1 R2



REZ **PXOR**: R1



A close up of a circuit board

Description automatically generatedA close up of a circuit board

Description automatically generatedR2 REZ

**A close up of a circuit board

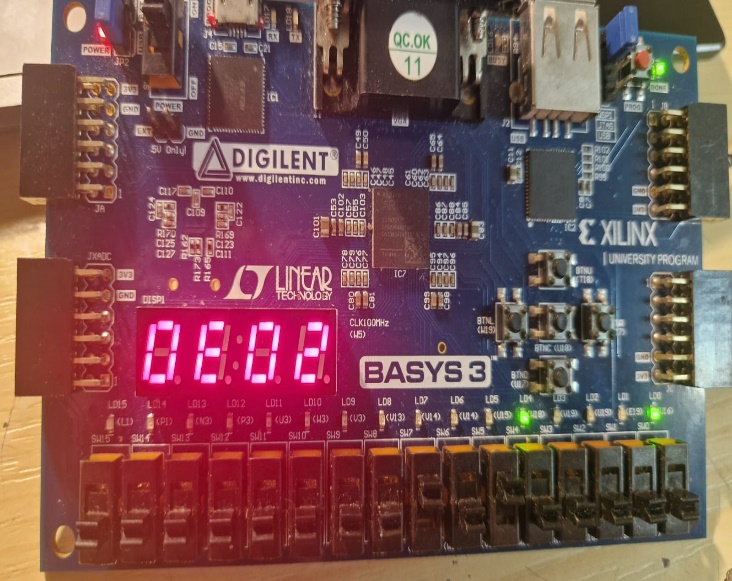
Description automatically generatedPSRLD**: R1 REZ

A close up of a circuit board

Description automatically generated

**A close up of a circuit board

Description automatically generated**

**PSLLD**: R1 REZ

**Concluzii**

Proiectul „Instrucțiuni MMX” a avut ca scop dezvoltarea și testarea unei Unități Aritmetice care să poată executa instrucțiuni MMX x86 pe o arhitectură de 64 de biți. Acesta a implicat integrarea pe o platformă FPGA, placuta Basys3 și a inclus atât partea de dezvoltare hardware, cât și cea software.

Etapele proiectului au fost planificate riguros, pornind de la definirea arhitecturii, continuând cu implementarea în VHDL și testarea funcțională. Printre componentele dezvoltate se numără unitatea de procesare a datelor MMX, o memorie cu date, un generator de pulsuri și un cod pentru afișajul cu șapte segmente. Aceste elemente au permis utilizarea eficientă a setului de instrucțiuni MMX, care includea operații precum adunare, scadere, shiftari logice si operații logice (ANDN, XOR), toate aceste fiind impachetate pe double word.

Prin realizarea acestei Unități Aritmetice, proiectul a demonstrat capacitatea de a gestiona în paralel date împachetate de diferite dimensiuni (byte, word, double-word), punând în evidență beneficiile procesării SIMD (Single Instruction, Multiple Data). Rezultatele obținute confirmă aplicabilitatea acestei unități în domenii precum multimedia și alte procese de calcul intensiv.

**Bibliografie**

1. <https://docs.oracle.com/cd/E18752_01/html/817-5477/eojdc.html>
2. <https://www.ee.torontomu.ca/~courses/ele818/mmx.pdf>
3. Curs 10 - PLA