**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

**Proiect**

la disciplina

**Structura sistemelor de calcul**

**Titlul**

***„Unitatea aritmetica si logica”***

Nume : Trif Gheorghe Andrei Indrumator : Cozma Andreea

Grupa: 30235 Data : 22.10.2019

Cuprins

[1 Rezumat 3](#_Toc27417792)

[2 Introducere 4](#_Toc27417793)

[3 Fundamentare teoretică 5](#_Toc27417794)

[3.1 Diagrama bloc: 5](#_Toc27417795)

[4 Proiectare şi implementare 6](#_Toc27417796)

[4.1 Metoda experimentala 7](#_Toc27417797)

[4.2 Operatii 7](#_Toc27417798)

[4.2.1 Adunarea si scaderea 7](#_Toc27417799)

[4.2.2 Operatiile logice: AND ,NAND,OR,NOR,XOR,XNOR 8](#_Toc27417800)

[4.2.3 Operatiile <,=,> 8](#_Toc27417801)

[4.2.4 Operatiile SLL(Shift logical left) si SLR (Shift logical right) 9](#_Toc27417802)

[4.2.5 Operatii complexe 9](#_Toc27417803)

[4.3 Module 10](#_Toc27417804)

[4.3.1 Afisorul led cu 7 segmente. 10](#_Toc27417805)

[4.3.2 Circuitul de debounce 10](#_Toc27417806)

[5 Rezultate experimentale 12](#_Toc27417807)

[6 Concluzii 13](#_Toc27417808)

# Rezumat

Unitatea aritmetico logica este aceea componenta a unui sistem de calcul care are ca rol efectuarea tuturor operatiilor aritmetice si logice de care acesta are nevoie.Astfel in acest proiect accentul cade pe implementarea unei unitati aritmetice si logice impreuna cu operatiile pe care aceasta le poate face intr-un limbaj de descriere hardware precum Vhdl ,dar si testarea acesteia pe o placa FPGA.Proiectarea acestei unitati sa facut pe ideea ca o instructiune se va executa intr-un singur ciclu de ceas .Astfel sa realizat proiectarea acestei unitati care executa majoritatea operatiilor logice , dar si a celor aritmetice.Dupa proiectarea acestei unitati aritmetico-logice am ajuns la concluzia ca aceasta este componenta fundamentala fara de care astazi un sistem de calcul nu ar putea exista.

# Introducere

In prim pas spre inceperea proiectari untati aritmetico logice este alcela de a stabili contextul temei si tendintele tehnologice legate de aceasta tema .Astfel se doreste implementarea unei unitati aritmetic-logice care sa se poata integra intr-un sistem mai complex .Se cere ca odata ce unitatea aritmetica este functionala aceasta se va testa pe o placa FPGA pentru a observa functionarea sa cat si pentru indeplinirea cerintei care sta la baza acestui proiect.

Acest proiect face parte din domeniul sistemelor de calcul.Un sistem de calcul este un ansamblu de componente hardware (dispozitive fizice) si software (programe) care permit solutionarea unor probleme a caror rezolvare se poate descrie sub forma unui algoritm.Intr-o abordare logica un sistem de calcul este un sistem stratificat pe mai multe nivele ierarhice: *Maşina fizică* alcătuită din componente electronice şi mecanice. La acest nivel se lucrează cu secvenţe de biţi care reprezintă coduri de instrucţiuni şi date. *Limbajul de asamblare*, permite programarea calculatorului prin instrucţiuni simple exprimate prin mnemonici. Fiecare mnemonică corespunde unui cod de instrucţiune. *Sistemul de operare* uşurează accesul la resursele calculatorului oferind un limbaj sau un mediu de operare şi un set de rutine predefinite (drivere) pentru lucrul cu aceste resurse. *Limbajele de nivel înalt şi mediile de programare.*Majoritatea limbajelor de programare dispun de o bibliotecă bogată de funcţii prin care se pot utiliza resursele calculatorului. *Aplicaţiile* sunt produse program care facilitează accesul la resursele unui calculator şi pentru personal nespecializat în domeniul calculatoarelor. Conform modelului clasic definit de J. Von Neumann, un calculator se compune din 5 tipuri de componente si anume: dispozitive de intrare,memorie interna si externa ,unitatea aritmetico logica,unitate de comanda,dispozitive de iesire.[2]

Revenind la tema propusa trebuie sa proiectam o unitate aritmetica logica care poate executa operatii de adunare scadere inmultire si operatii logice precum siftare stanga drepta operatiile logice elementare precum: AND,NOT-AND,OR,XOR etc.Voi incerca sa implementez si anumite operatii auxiliare precum o comparatie care va avea ca efect setarea anmitor flaguri despre care se va discuta in capitolele urmatoare.Unitatea aritmetica logica are doi operanzi de intrare pe n biti si produce un rezultat de iesire, aceasta mai are si anumite flaguri de un bit care vor semnala anumite rezultate precum :>=,<=,= ,zero etc.Unitate aritmetica logica va mai avea o intrare prin cate se va selecta operatia pe care o va executa pe o anumita perioada de ceas.O data ce unitatea eritmetica logica va fi functionala pentru a reusi sa afisez un rezultat si pentru a putea testa corectitudinea operatiilor peo placa FPGA voi folosi un sistem de afisare bazat pe un afisor cu 7 segmente [1].Astfel se va putea vizualiza rezultatul operatiilor si se va usura testarea.

In cele ce urmeaza voi discuta despre solutia pe care o propun .In primul rand voi incerca sa proiectez o unitate aritmetica logica pe un bit care va face operatiile de adunare ,SI, SAU.Dupa aceasta se va incerca implemantarea operatiilor mai complexe percum scaderea,siftare extindere cu 0 sau 1, detector de pritate. Astfel dupa construirea acestei unitati pe 1 bit pentru a putea construi un ALU pe cel putin 8 biti vom putea cascada 8 ALU pe 1 bit astfel avand o metoda de compunere bottom-up.Aceasta metoda presupune plecarea de la blocuri si unitati simple, iar in final ajungandu-se la ansambluri mai complexe.[1],[3]

# Fundamentare teoretică

## Diagrama bloc:

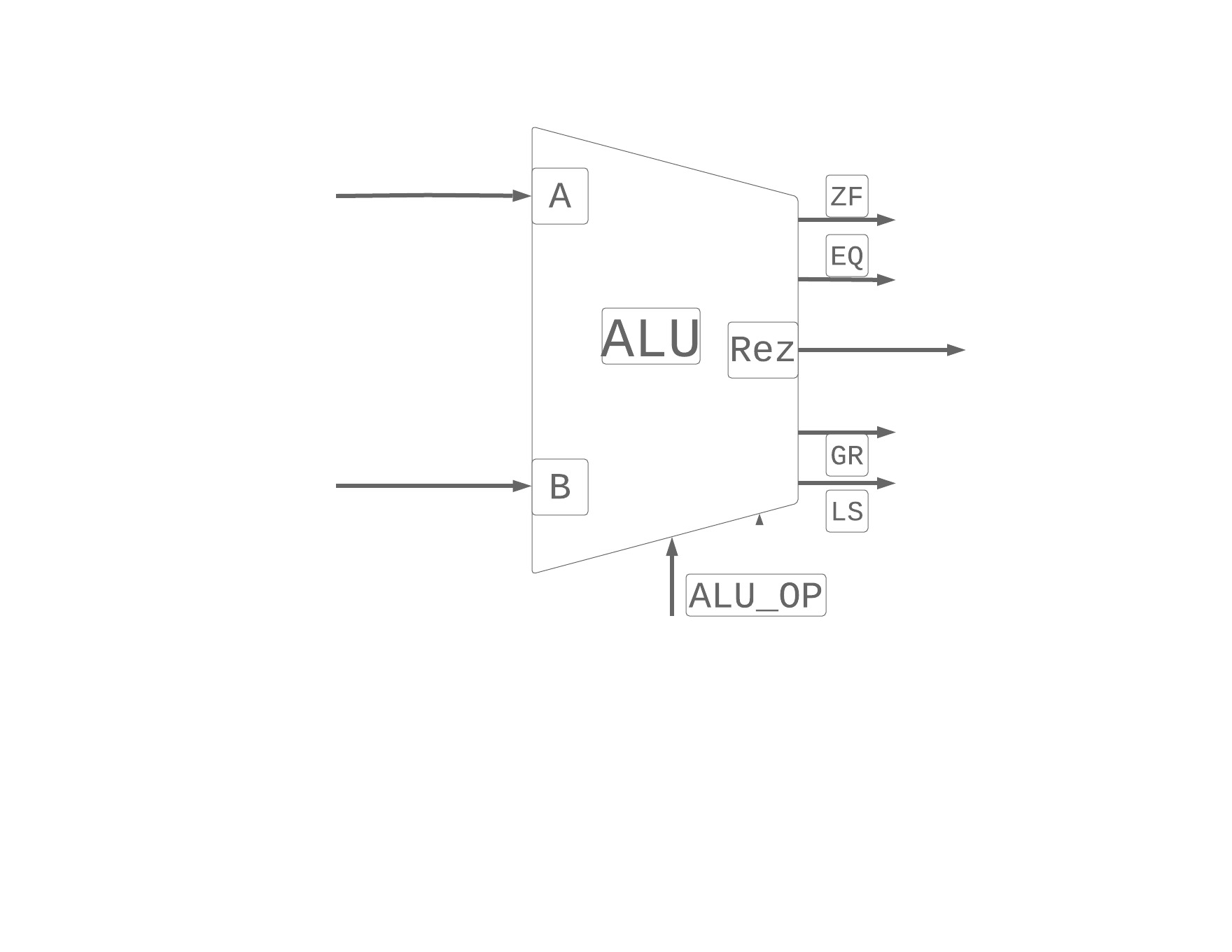


Figura 3.1

Schema bloc unitatea aritmetico-logica

* A si B - sunt cei doi operanzi pe care ii va primi ALU;
* ALU\_ OP - este intrea care are ca scop selectarea operatiiei aritmetice sau logice pe care utilizatorul o doreste.
* Rez - este iesirea pe care se va transmite rezlutatul operatiilor aritmetice si logice.
* ZF,GR,LS,EQ -sunt iesiri de un bit cu rol de flag-uri pentru a determinarea anumitor caracteristici despre operanzi.

# Proiectare şi implementare

In aceasta etapa se va folosi limbajul Vhdl si programul Vivado pentru implementarea unitati aritmetice si logice .Pentru a usura descrierea solutiei nu vom incepe prin a scrie codul la nivelul de porti logice .Astfel voi incerca sa implementez solutia utilizand circuite care au fost deja construite .

Pentru a putea incepe sa rezolvam problema mai intai trebuie sa stabilim ce operatii va executa unitatea aritmetica si va mai trebui sa codificam aceste operatii astfel incat sa le putem accesa in viitor.

|  |  |
| --- | --- |
| Operatie | Codificare |
| Adunare | 00000 |
| Scadere | 00001 |
| SI-Logic | 00010 |
| Sau-Logic | 00011 |
| SI-NU | 00100 |
| SAU-NU | 00101 |
| SAU -EXCLUSIV | 00110 |
| SAU-NU EXCUSIV  >  <  =  Media aritmetica  SLR  SLL  Radical(Sqrt)  Impartire  Inmultire | 00111  01000  01001  01010  01011  01100  01101  01111  10000  10001 |

Astfel aceste combinatii vor fii introduse pe intrarea ALU\_OP care reprezinta selectia unui multiplexor si astfel pe iesirea Rez ne va fi returnata valoarea operatiei efectuate.

Am inceput prin definirea operatiilor de baza pe care orice ALU le are in interior urmand ulterior construirea unor operatii mai complicate precum radical ,inmultire ,impartire.Pentru astfel de instructiuni intrarea ALU\_OP va fii extinsa de la 4 biti la 5 biti deoarece pe 4 biti nu vom avea destule combinatii.

## Metoda experimentala

Pentru implementarea proiectului se va folosi limbajul de descriere hardware in mediul Vivado Xilinx.In cea ce priveste ALU structura sa este aceiasi .In principiu este o cutie in care poate face diferite operatii, nu putem incerca acest o altfel de implementare , dar putem implementa diferit operatii pe care aceasta le contine .De exemplu pentru a face operatia de radical ar fifost mai simplu sa folosim o functie care face acest lucru pentru noi si care este implementata deja intr-o biblioteca.

## Operatii

In aceasta sectiune se vor explica operatiile pe care unitatea aritmetica logica le contine si cum pot fi acesta implementate.

### Adunarea si scaderea

Prima operatie pe care am implementat-o este operatia de adunare .Aceasta are codificare "00000" si are ca operanzi cele doua intrari de 8 biti ale lui ALU.Aceasta operatie este banala se putea face fie implementand un sumator pe 8 biti fie cascadand 8 sumatoare pe 1 bit sau folosind o functie deja implementata din biblioteca . Astfel intreaga operatie se reduce la o singura linie de cod.

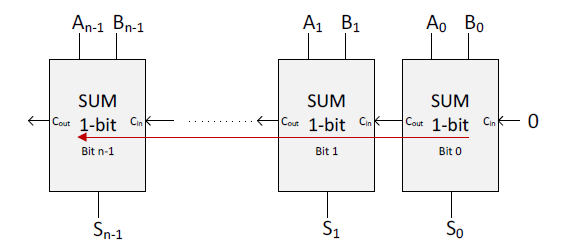


Fig.4.1Sumator pe n biti construit cu sumatoare pe 1 bit cascadate

Operatia de scadere este foarte asemanatoare cu adunarea singura diferenta fiind faptul ca in loc sa adunam la primul operand al doilea operand ,la primul operand va trebui sa adunam complementul fata de 2 al celui de-al doilea operand .

### Operatiile logice: AND ,NAND,OR,NOR,XOR,XNOR

Acestea sunt operatiile logice foarte comune intr-un ALU .Acestea se executa bit cu bit.Spre exemplu penrtru a face un "si" intre cei doi operanzi ai unitatii aritmetice logice ,va trebui sa folosim 8 porti "si" cu doua intrari , fiecare poarta primeste pe o intrare un bit de la cei doi operanzi si va returna pe iesire rezultatul.

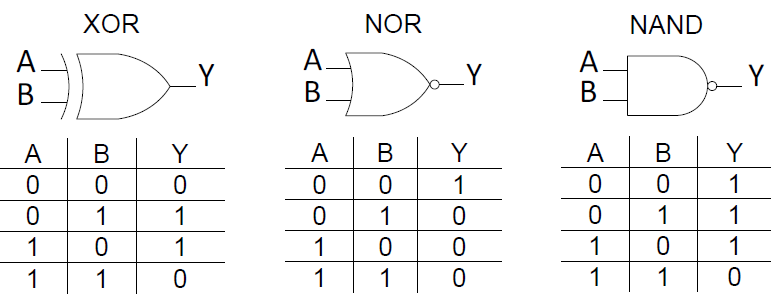
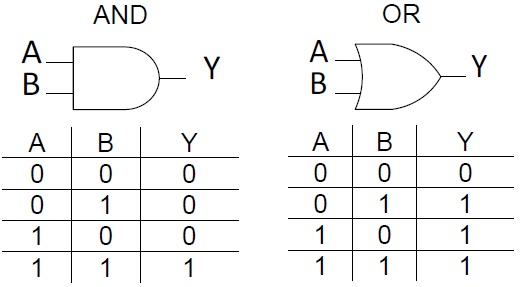


Fig.4.2.Tabele de adevar ale portilor elementare

### Operatiile <,=,>

Acestea sunt operatii de comparare .Acestea au ca scop compararea operanzilor si setarea anumitor flaguri care sa indice faptul ca compararea este adevarata.

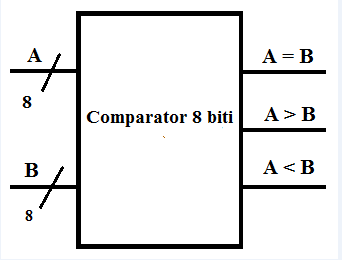


Fig.4.3.Comparator pe 8 biti

Dupa cum se poate observa in figura 4.3 comparatorul are iesirile pe 1 bit.Acest lucru insemnand ca acestea au rolul doar de a confirma daca una dintre operatii este adevarata sau nu.In ideea ca cineva va dori sa foloseasca operandul care este mai mic sau mai mare am implementat ALU in asa fel incat operandul sa fie transmis pe iesirea ALU .

### Operatiile SLL(Shift logical left) si SLR (Shift logical right)

Aceste doua operatii sunt operatii de shiftare care au ca scop deplasarea unui nr de biti la stanga sau dreapta si umplerea pozitiilor siftate cu 0.



Fig 4.4 Siftarea stanga sau dreapta

### Operatii complexe

Acestea sunt operatii putin mai complexe cum ar fii media aritmetica ,radical ,impartire.

Prima dintre aceste operatii este media aritmetica care va face suma celor doi operanzi si ii va imparti la numarul lor ,in cazul nostru 2.Pentru aceasta operatie vom avea nevoie de anumite functii de conversie deoarece in acest moment nu avem operatia de impartire implementata .Astfel secventa de

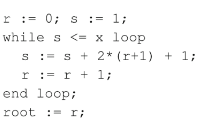
cod care face media aritmetica este urmatoarea:



Dupa cum se poate observa am folosit functia to\_integer() care va transforma cei 8 biti ai primului operand si neva returna un intreg .Dupa aceasta operatie vom folosi to\_unsigned() pentru a face intregul fara semn .Am facut acelasi lucru si cu cel de-al doilea operand .Dupa aceasta urmeaza sa ii adunam si sa ii imapartim la 2.Odata ce rezultatul este gata acesta va trebui convertit inapoi in binar folosind functia std\_logic\_vector() deoarece acest rezultat va trebui transmis pe iesirea ALU.

A doua operatie este operatia de radical.Pentru aceasta operatie se va folosi un FSM cu 3 stari.

Entitatea sistemului este formata din patru semnale de intrare: x-operandul ,clk-semnalul de ceas ,un semnal de start -start si un semnal de reset-reset si doua semnale de iesire : r-rezultatul si un semnal done care indica ca operatia sa terminat.Aceasta operatie a fost implementata dupa urmatorul algoritm:



Schema acestui circuit ese prezentata in figura urmatoare:

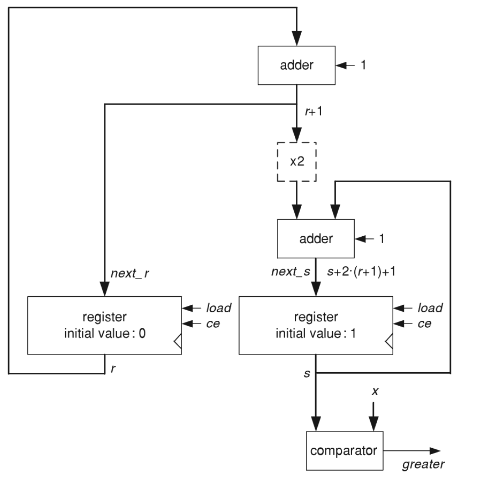


Fig.4.5 Unitatea de calcul a radicalului

Pentu a completa circuitul este nevoie de adaugarea unei unitati de control care se va ocupa de generarea semnalelor de load si ce.Aceasta este o masina cu stari finite care are ca intrare semnalul grater care simbolizeaza finalul executiei buclei algoritmului si doua iesiri load and ce. Diagrama de stare a acestei unitati de control este urmatoarea[4]:

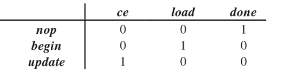
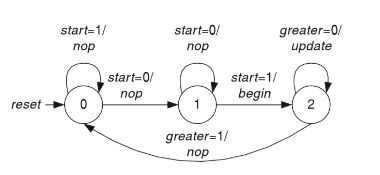


Fig.4.6.Diagrama de stare a unuitati de control

A treia operatie este impartirea aceasta operatie implica doi operanzi : deimpartitul care este o valoare pe 8 biti si impartitorul care este o valoare pe 4 biti.Aceasta opertatie va genera un cat pe 8 biti si un rest pe 4 biti .Atat catul cat si restul vor fi afisate pe placa .Catul se va afisa pe afisorul cu led 7 segmente iar restul se va afisa pe ledurile de sub afisor.Am implementat operatia de impartire conform urmatorului algoritm de imaprtie:

1.Stocam deimpartitul in concatenare a doua variabile n1 si n2 de 5 respectiv 4 biti;

2.Stocam impartitorul in variabila d pe 4 biti;

3.Repetam urmatorul set de operatii de 4 ori:

siftam concatenarea lui n1 cu n2 la stanga cu 1 bit;

daca n1 > d atunci

n1=n1-d;

n2(0)=1;

4.Dupa ce se iese din bucla catul va fi in n2 si restul se va afla in cei mai putini 4 biti ai lui n1 adica n1(3 downto 0);

Acest algoritm a fost implementat sub forma unei proceduri in vhdl deoarece pentru a reusi implementarea operatiei de impartire va trebui sa respectam diagrama care dupa cum se poate vedea necesita apelarea de doua ori a proceduri div4 .

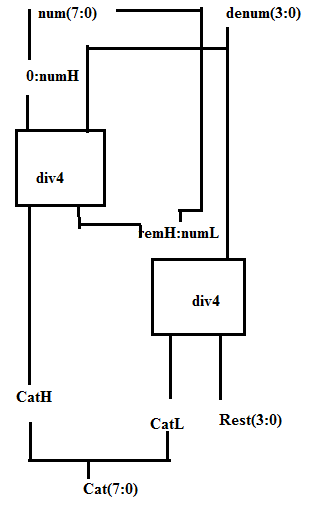


Fig.4.5 Schema unitati de imaprtire 8:4

A patra operatie este inmultirea matriciala a doua numere pe 4 biti.Pentru a realiza acrasta operatie am ales folosirea unui sumator elementar de 1 bit.Astfel prin legarea mai multor sumatoare elementare se va putea obine rezultatul inmultiri care este pe 8 biti.Acest rezultat va fi trimis pe iesirea ALU care este conecata la afisor.Legarea sumatoarelor sa facut conform diagramei [5]:

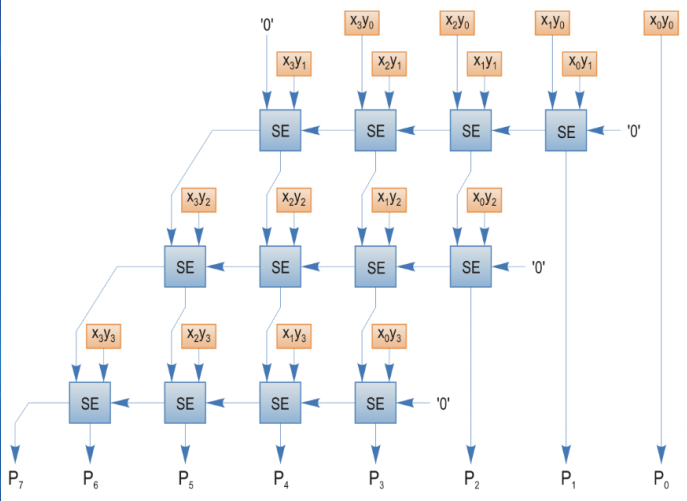


Fig.4.6 Shema inmultiri matriciale

Schema bloc al unui astfel de sumator se poate observa in figura urmatoare impreuna cu tabelul de adevar al acestuia.Pe langa aceste sumatoare sau mai folosit si cateva porti si cu doua intrari deoarece pe aprosimativ toate intrarile acestor sumatoarele este nevoie de aceste porti [6].

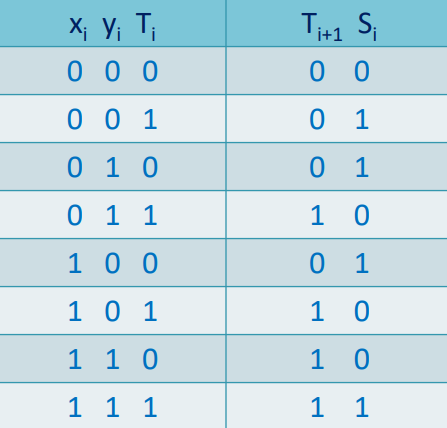
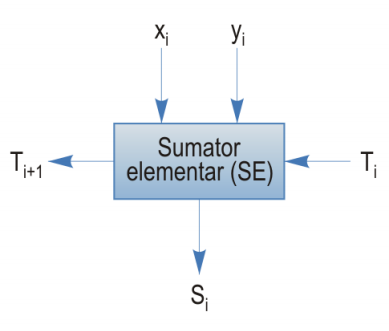


Fig.4.7 Sumatorul elementar si tabelul de adevar al acestuia

## Module

Pentru a putea rula proiectul vom avea nevoie un afisor led cu 7 segmente ,si de un circuit debunce pentru a putea utiliza butoanle placi.

### Afisorul led cu 7 segmente.

Display-ul de șapte segmente (de asemenea denumit, afisaj LED cu 7 segmente) este o modalitate de a reprezenta numere în sisteme electronice. Acesta este compus din șapte segmente care pot fi activate sau dezactivate individual. Fiecare segment are forma unei linii scurte.Utilizand astfel afisorul placi si setand in fisierul de constrangeri anozi si catozi se va putea afisa pe placuta iesirea ALU.

### Circuitul de debounce

În circuitele secvențiale, orice variabilă de intrare care nu este dependentă de ceasul sistemului, este o intrare asincronă. Majoritatea acestor intrări asincrone provin de la dispozitive mecanice, cum ar fi cele patru butoane de pe placa de dezvoltare din laborator. Pe lângă faptul ca sunt asincrone, aceste semnale nu sunt perfect dreptunghiulare, elementele mecanice din interiorul butoanelor avand nevoie de un mic timp pentru a se stabiliza. In acest timp semnalul generat este format din multe spike-uri, care conduc la funcționarea gresită a circuitului nostru .Ne propunem să rezolvăm această problemă prin ignorarea acelei perioade din semnal (cateva milisecunde). Acest lucru îl face circuitul de Debounce, care primește la intrare un semnal, asemanator lui **in** din figura de mai jos și generează la ieșire un semnal perfect dreptunghiular, de durata unei singure perioade de ceas [7].

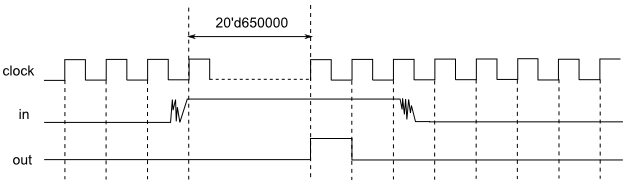


Fig.4.5 Generare semnal perfect dreptungiular cu ajutorul unui circuit debounce

Schema bloc a unui astfel de circuit este urmatoarea:

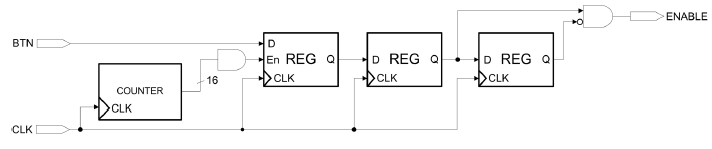


Fig 4.6 Circuit debounce

Am folosit acest circuit de pentru trei butoane doua dintre ele sunt pentu unitatea de calcul a radicalului pentru semnalele de start si reset ,iar al treilea debounce este folosit la un buton care imi incrementeaza selectia operatiei pe ALU .A fost nevoie de acest buton deoarece pe placa de dezvoltare exista doar 16 swich-uri pe care le-am folosit la pentru operanzi .Astfel de fiecare data cand acest buton este apasat valoarea semnalului ALU\_OP se va incrementa cu o unitate.

# Rezultate experimentale

# Concluzii

Bibliografie

[1]. Carte BazeleArhitecturiCalculatoarelor - de Florin Oniga;

[2]. http://users.utcluj.ro/~baruch/ac/curs/UAL.pdf;

[3].Curs 05 Arhitectura Calculatoarelor pag 1-13 ,Florin Oniga;

[4]. Carte Guide to fpga implementation of arithmetic functions ,pag 23-27 -

de Jean Pierre Deschamps,Gustavo D. Sutter,Enrique Cantó;

[5].Curs 4 Structura Sistemelor de calcul , Dr. Baruch Zoltan Francisc;

[6].Curs 4 Structura Sistemelor de calcul , Dr. Baruch Zoltan Francisc;

[7].Curs 01 Arhitectura Calculatoarelor ,Dl. Florin Oniga;