****

**Facultate: Automatică și Calculatoare**

**Specializare: Calculatoare și Tehnologia Informației**

**Disciplina: Proiectarea Sistemelor Numerice**

**Proiect: Calculator de buzunar**

**Idrumator laborator , Student,**

**Ing. Pop Diana Vîrnă Ioana, gr. 30212**

**Cuprins**

[1.Specificație](file:///E:\Calculator%20de%20buzunar\Calculator-de-buzunar.docx#Spec)

* Cerință
* Placa de dezvoltare Basys 3

[2.Lista de componente. Proiectarea şi implementarea](file:///E:\Calculator%20de%20buzunar\Calculator-de-buzunar.docx#ListaComponenteee)

* Schema de detaliu
* Proiectarea componentelor
* Semnificaţia notaţiilor

[3.Justificarea soluţiei alese](file:///E:\Calculator%20de%20buzunar\Calculator-de-buzunar.docx#JustificareaSolutiei)

[4.Utilizare şi rezultate](file:///E:\Calculator%20de%20buzunar\Calculator-de-buzunar.docx#UtilizareaAndRez)

* Resurse utilizate
* Paşi necesari pentru utilizare
* Rezultate

1. ***Specificația proiectului***

***Cerința:***

Să se proiecteze un calculator de buzunar cu operații aritmetice fundamentale (adunare, scădere, înmulțire, împărțire). Operațiile de înmulțire și împărțire se vor implementa folosind algoritmi specifici, nu operatorii limbajului. Operanzii sunt reprezentați pe 8 biți cu semn. Operanzii si operatorii vor fi introduși secvențial in formă zecimală. Se vor folosi afișajele cu 7 segmente de pe plăcuțele FPGA.

**Placa de dezvoltare Basys 3**

Placa de dezvoltare folosită pentru proiect este una dintre familiile FPGA, denumită BASYS 3, această placă fiind produsă de firma Digilent, ea oferind toate instrumentele necesare pentru dezvoltarea proiectului cerut.

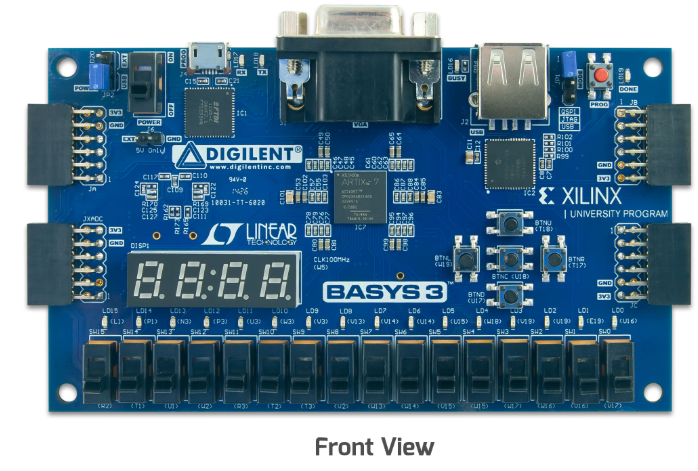
Printre componentele de care dispune putem enumera câteva care sunt de folos în realizarea acestui proiect:

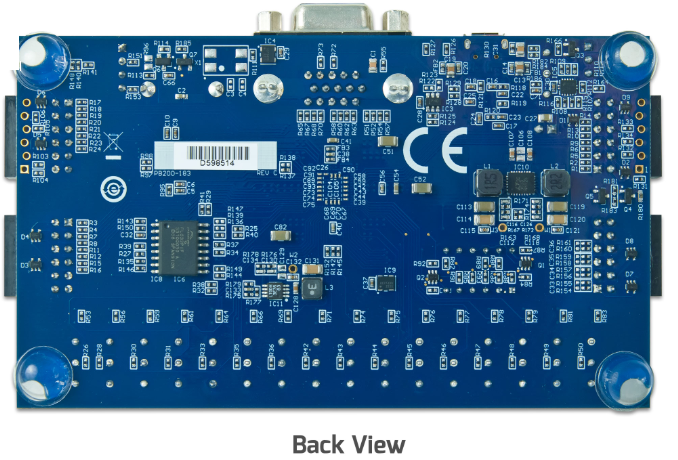
• patru afișoare de tip 7 segmente;

• 16 switch-uri

• 16 LED–uri;

• cinci butoane cu apǎsare temporarǎ;





Afisorul pe 7 segmente

In cele ce urmeaza voi prezenta pe scurt principalele caracteristici ale afisorului pe 7 segmente deoarece aceasta este una din cele mai importante componente ale placii BASYS 3 care ne permite sa vizualizam intr-un format usor de inteles datele de iesire.

Placa Basys 3 conține un anod comun din patru cifre cu afișaj LED format din șapte segmente. Fiecare dintre cele patru cifre este compusă din șapte segmente sub forma cifrei 8, cu câte un LED integrat în fiecare segment. LED-urile segmentelor pot fi iluminate individual, așa că oricare dintre cele 128 de modele poate fi afișat pe o cifră prin iluminarea anumitor segmente și neiluminarea celorlalte. Dintre cele 128 de modele, cele mai folosite sunt cele zece corespunzătoare cifrelor zecimale. Anodurile celor șapte LED-uri care formează fiecare cifră sunt legate într-un anod comun, dar catodurile LED-urilor rămân separate. Semnalele anodului comun sunt disponibile ca 4 semnale de activare cifră pentru afișajul de tip 4-cifre.

2***. Lista de componente. Proiectarea şi implementarea***

* Schema bloc

Calculatorul primeste ca intrari 2 numere ce sunt date in complement fata de 2 deoarece avem nevoie si de semnul numerelor.

Rezultatul este un numar pe 16 biti (maximul numarului de biti dupa cea mai mare operatie) si este dat in hexazecimal. Dupa conversie, rezultatul va fi in zecimal si primul afisor va contine semnul, iar celelalte 3 vor indica rezultatul final. De asemenea, exista o limita pentru numerele introduce astfel incat rezultatul sa nu se poata afisa.

NR1 8

NR2 8

Operatie 2

16 Rezultat

Calculator

Figure 1: Cutia neagra

* Componente

Calculatorul este format din 3 componente principale, fiecare avand functii diferite. Dupa realizarea operatiilor si a conversiei, rezultatul este trimis catre afisor(SSD).

ALU

Conversie din binar in zecimal

Afisor 7-segmente

16

Rezultatul pe afisor

16

1

16

2

8

8

Nr1 pe switch

Nr2 pe switch

Selectia operatiei pe butoane

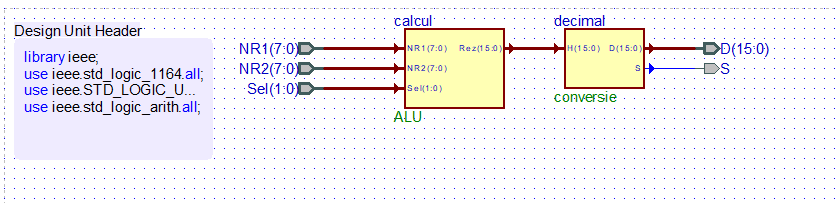


Figure 2 - Calculator logic

***A) Unitatea aritmetica-logica(ALU)***

Aceasta este componenta principala a calculatorului, care realizeaza operatiile aritmetice. Primeste ca intrari 2 numere pe 8 biti si selecteaza pe 2 biti ce operatie va fi realizata. Iesirea este rezultatul operatiei, reprezentat pe 16 biti.

Componenta ALU este formata din 4 componente: un sumator complet pe 8 biti, un inmlutitor, un impartitor si o componenta pentru conversia in complement fata de 2.

Toate operatiile sunt realizate in acelasi timp si depend de selectia cu ajutorul careia afisam rezultatul. Selectia se realizeaza cu un MUX 4:1 chiar daca nu e o componenta diferita.



Fig 3 - ALU

Adunarea si scaderea au la baza aceeasi componenta ce realizeaza adunarea pe 8 biti(este bazata pe sumator complet pe un bit). Diferenta este obtinuta adunand al doilea numar sub forma complement fata de 2 la primul. Chiar daca rezultatul are 8 biti, in final va fi pe 16 biti (extensie a semnului).

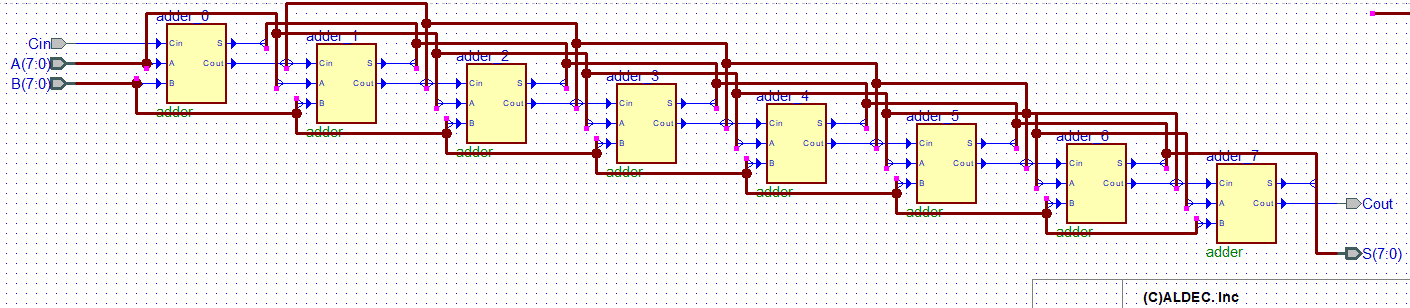


Fig 4 – Adunare si Scadere

Pentru inmultire si impartire vom folosi algoritmi, si nu operatori ( \* sau /).

Metoda aleasa pentru inmultire este inmultirea matricilor si poate fi aplicata doar numerelor pozitive. De aceea numerele negative vor fi transformate in complement fata de 2 inaite de inmultire. Rezultatul va fi pe 16 biti si este in complement fata de 2 daca este negativ ( sunt verificati primii biti ale numerelor).

Impartirea o vom realiza prin scaderi successive. Se va lua in considerare doar catul. Asemenea inmultitorului, impartirea se realizeaza doar pentru numere pozitive si inainte de operatie sunt convertite in complement fata de 2 daca sunt negative. Rezultatul va fi pe 8 biti, de aceea va fi concatenat cu semnul si cu zerouri.

Va fi convertit in complement fata de 2 daca este negative, utilizand componenta C2.

Conversia in complement fata de 2 va fi realizata de o componenta ce schimba numarul in C2 doar daca este negativ, verificand primul bit al numarului. Altfel, numarul nu se schimba. Componenta C2 este utilizata pentru rezultatul inmultirii si impartirii, deoarece aceste operatii sunt realizate doar pentru numere positive.

***B) Conversia***

Aceasta componenta converteste rezultatul preluat din ALU din binar (hexazecimal) in zecimal. Este utiliazata o alta componenta cu un algoritm simplu pentru transformarea din binary in BCD. Conversia se realizeaza doar pentru numere positive. Cele negative sunt convertite in C2. Aceasta componenta are ca intrare un numar pe 16 biti si ca iesire un numar pe 16 biti si un bit separate pentru semn, ce este transmis catre afisorul 7-segmente.

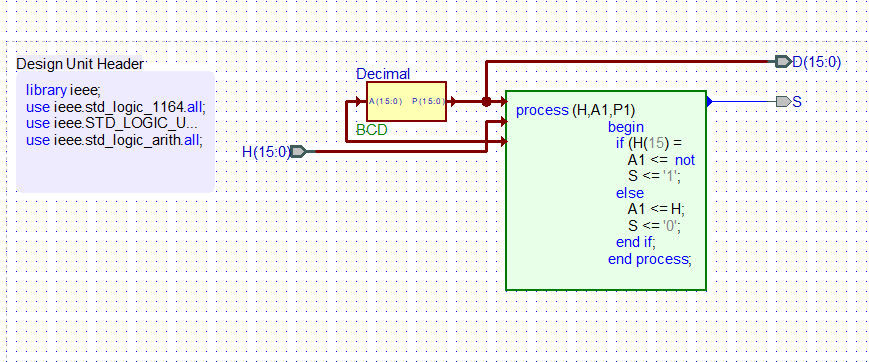


Fig 5 – Conversia componentelor

***C) Afisor 7-segmente***

Aceasta componenta primeste ca intrare rezultatul de la conversie si bitul de semn. Rezultatul va fi afisat pe 3 afisoare si unul este rezervat semnului (b daca este negativ).

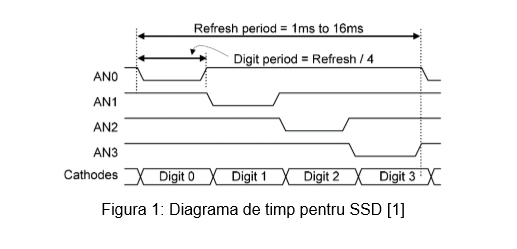


Fig 6 – Diagrama pentru afisorul 7-segmente

Este posibil sa afisam o singura cifra in acelasi timp. Pentru a afisa 4 numere, este necesar sa implementam un circuit care transmite semnalele pentru catozi. Maximul perioadei va fi calculate astfel incat vederea umana sa nu poata percepe aprinderea separate a fiecarui numar(16 ms = 60 MHz). Este realizata o afisare ciclica a numerelor(rand pe rand, insa noi nu observam asta).

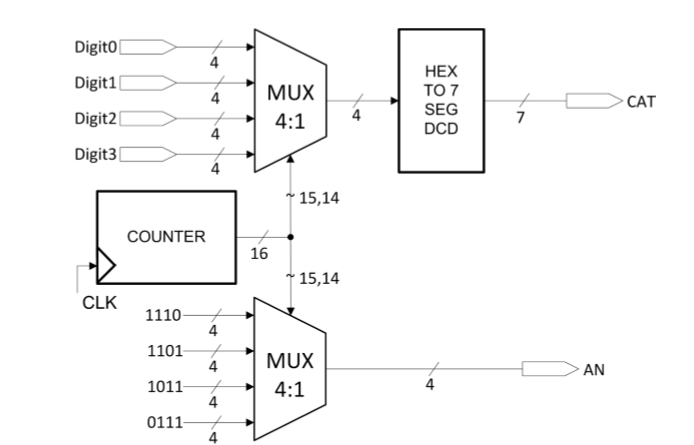


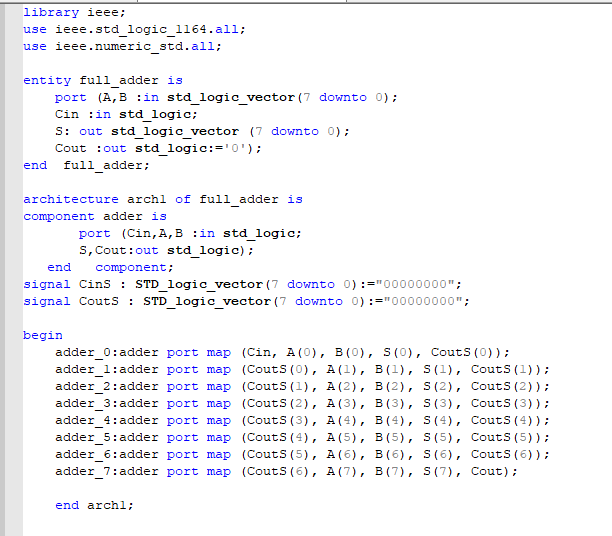
Fig 7 – SSD

Codul VHDL

VHDL este un limbaj hardware.

Componenta principala numita Calculator, este formata din ALU si display.

Adunarea si scaderea:



Sunt cascadate mai multe sumatoare complete pe 1 bit ca in diagrama prezentata anterior. Pentru diferenta, singura optiune e transformarea celui de al doilea numar in complement fata de 2 inaintea realizarii operatiilor.

8

8

cin

B 7:0

A 7:0

S 7:0

cout

8

Semnificatia notatiilor:

* A[7:0] – primul numar pe 8 biti cu semn
* B[7:0] – al doilea numar pe 8 biti cu semn
* SUM[7:0] – suma celor 2 numere
* cout – in cazul unei depasiri se pune pe 1
* cin – intrare (cout de la adunarea precedenta), valoarea initiala este 0

Complement fata de 2:

Reprezentarea in complement fata de 2 este vitala pentru a efectua usor din punct de vedere logic scaderea. Complementul fata de 2 este realizat cu ajutorul operatiei XOR. Numarul este primit de catre componenta prin vectorul inside[15:0], complementul fata de 2 se va realiza prin repetare operatiei XOR intre inside(15) ,bitul de semn, si inside[14:0] dupa care se va aduna inside(15).

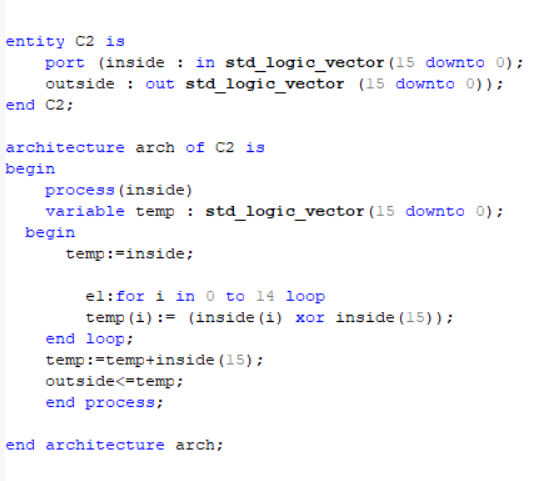
Asadar, daca bitul de semn este 1, se va forma complementul fata de 2 iar daca bitul de semn este 0 nu se va intampla nimic.

inside 15:0

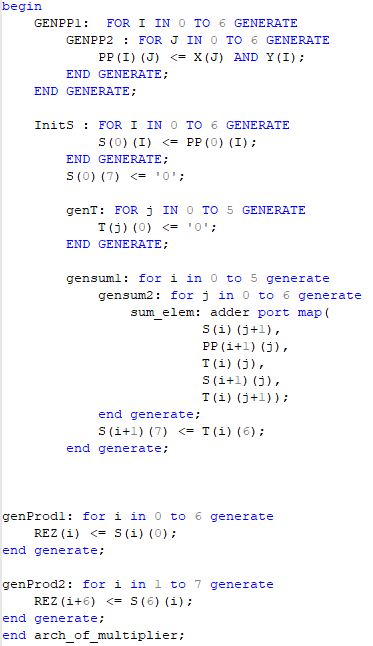
outside 15:0

Semnificatia notatiilor:

* inside[15:0] – numarul care va fi transformat in complement fata de 2
* outside[15:0] – numarul in complement fata de 2



Inmultire:



Pentru inmultire este utilizata inmultirea matricilor ca metoda, ceea ce este explicate in alegerea solutiei.

Y 7:0

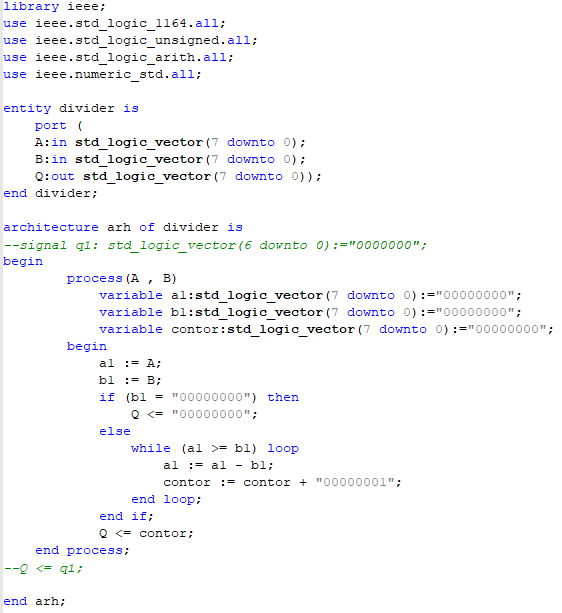
X 7:0

REZ 13:0

Semnificatia notatiilor:

* X[7:0] – primul numar pe 8 biti cu semn
* Y[7:0] – al doilea numar pe 8 biti cu semn
* REZ[13:0] – rezultatul inmultirii

Impartirea:



Impartirea se realizeaza utilizand scaderi repetate si rezultatul va memora doar catul.

B 7:0

A 7:0

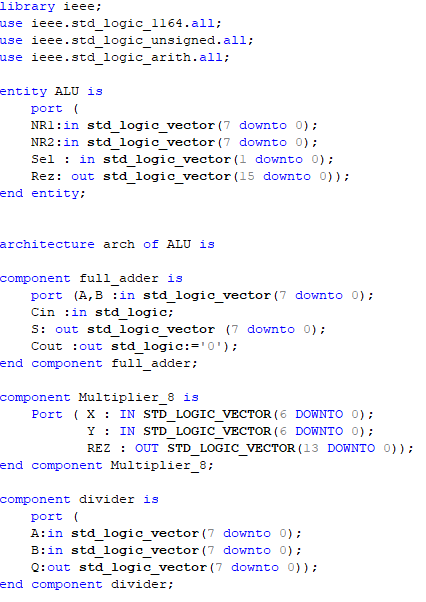
Q 7:0

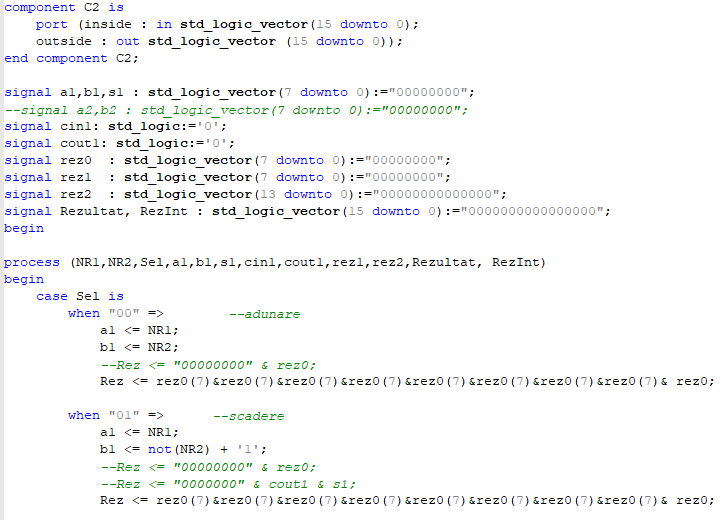
Semnificatia notatiilor:

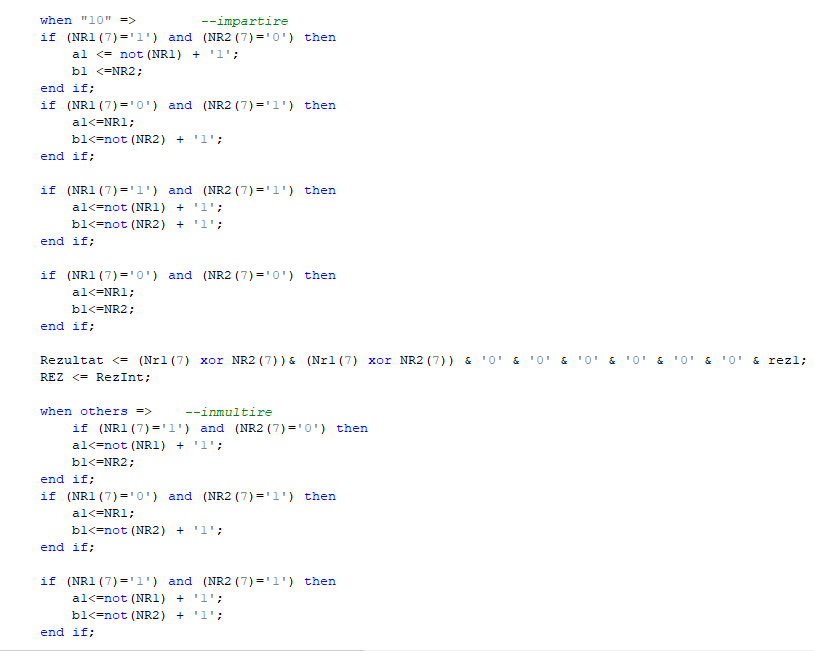
* A[7:0] – primul numar pe 8 biti cu semn
* B[7:0] – al doilea numar pe 8 biti cu semn
* Q[7:0] – rezultatul impartirii

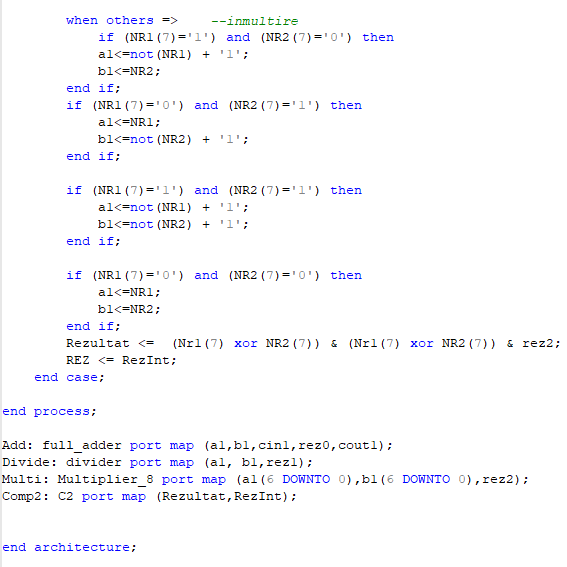
ALU

ALU este formata din cele 4 operatii realizate in acelasi timp si rezultatul afisat este selectat printr-o anumita optiune selectata:









NR2 7:0

SEL 1:0

NR1 7:0

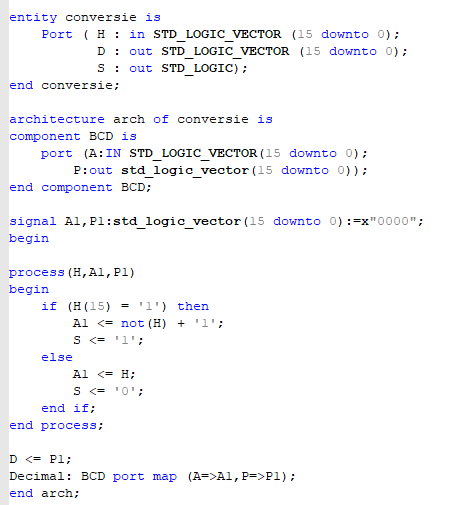
REZ 15:0

Semnificatia notatiilor:

* NR1[7:0] – primul numar pe 8 biti cu semn
* NR2[7:0] – al doilea numar pe 8 biti cu semn
* SEL[1:0] – selectarea operatiei
* REZ[7:0] – rezultatul impartirii

Conversia:

Conversia din binar in zecimal se bazeaza pe algoritmul care va fi explicat in Alegerea solutiei.



H 15:0

REZ 15:0

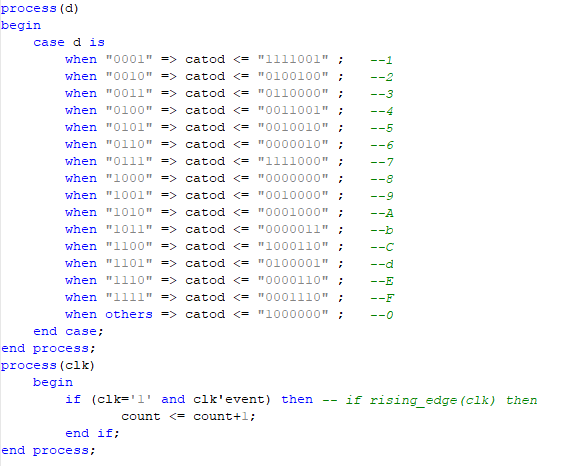
S

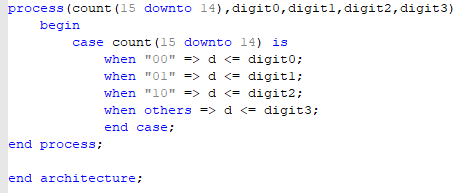
Semnificatia notatiilor:

* H[7:0] – rezulatatul in binar
* REZ[7:0] – rezultatul in zecimal
* S – semnul

***Afisorul:***

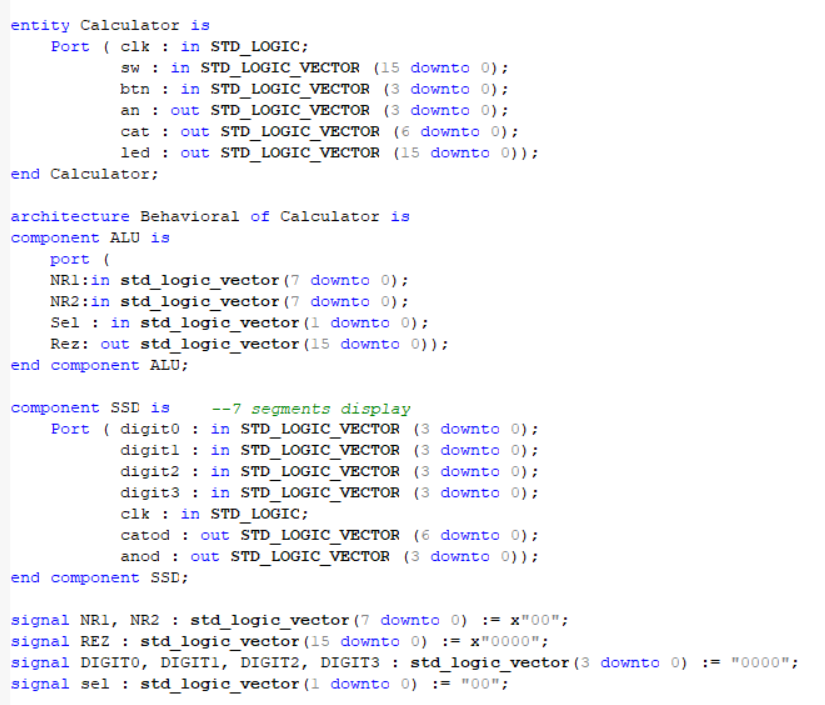
Primeste valorile pentru catozi si anozi astfel:

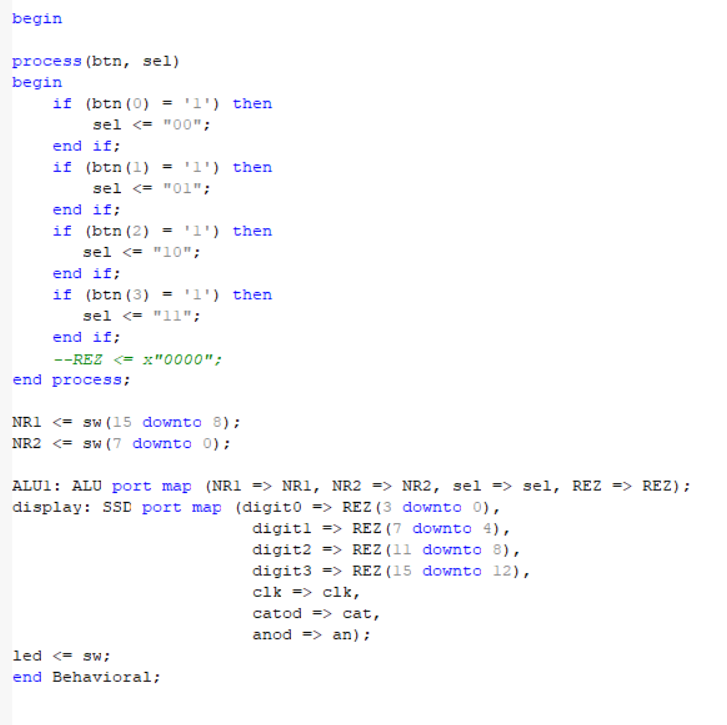




***Componenta finala:***

Componenta finala in VHDL este formata doar din ALU si afisorul 7-segmente.

Aceasta componenta, ia rezultatele din ALU in functie de ce selectie este introdusa, iar prin intermediul componentei SSD este afisat.



3. ***Justificarea solutiei alese***

Acest proiect ce efectueaza cele 4 operatii a fost unul interesant. Am cautat cei mai eficienti algoritmi pentru operatii, insa am decis sa folosesc unii care sunt mai putin eficienti, insa mai simplii de implementat in hardware.

Adunarea si scaderea folosesc o cascadare simpla a sumatorului complet pe 1 bit, diferenta este aceea de introducere a primului numar.

Inmultirea se bazeaza pe metoda inmultirii matricilor si se aplica doar pentru numere pozitive.



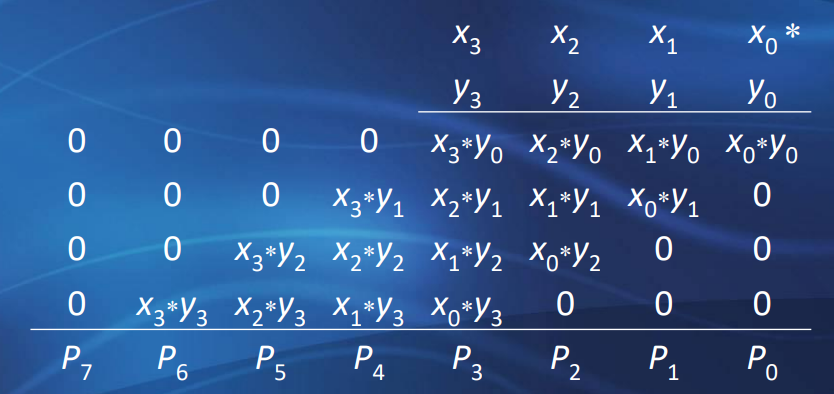
Metoda: Consideram 2 numere intregi fara semn:

X=x(n-1)\*…\*x(1)\*x(0)

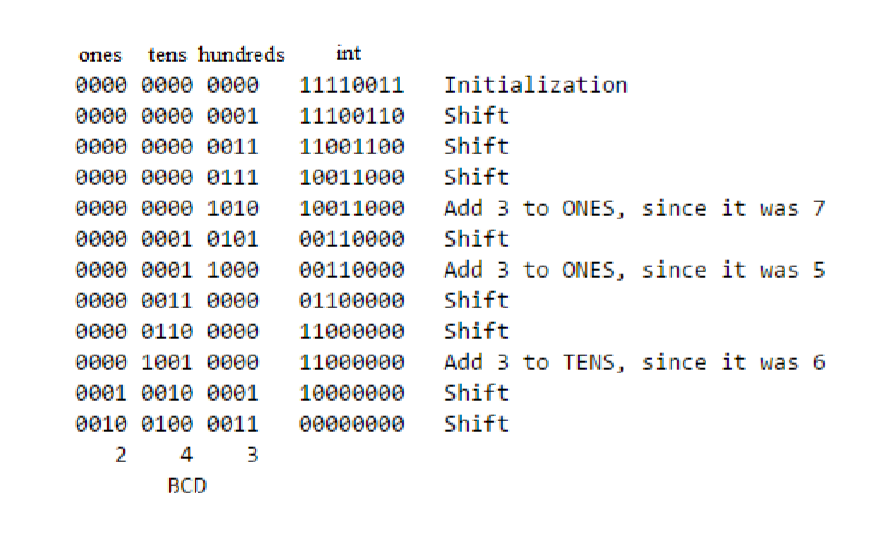
Y=y(n-1)\*…\*y(1)\*y(0).

P= X\*Y = \* .

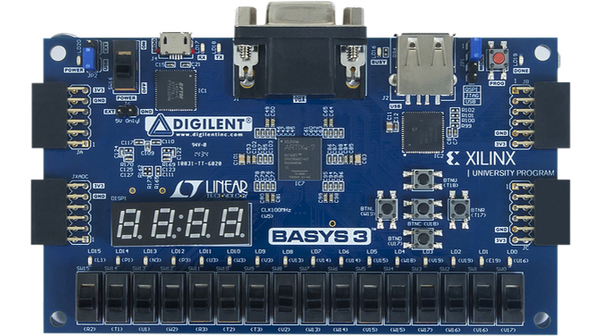
Fiecare produs x(i)\*y(j) poate fi realizat utilizand AND. O matrice n\*n de porti SI poate obtine toti termenii x(i)\*y(j) in acelasi timp. Toti termenii sunt adunati utilizand o matrice n\*(n-1) de sumator complet pe 1 bit. Shiftarea care rezulta din 2^I si 2^j este implementata are implemented printr-o mutare de spatiu la adunarea pe directiile x si y.



Pentru conversia in BCD din binary, este folosit urmatorul algoritm: Numarul are 3 coloane, pentru sute, zeci si unitati. Daca una este mai mare sau egala cu 5, se aduna 3 la acea coloanal. Apoi toate coloanele sunt shiftate la stanga cu cate o pozitie. Algortimul trebuie sa fie efectuat de cati biti are numarul initial in binar.



4. ***Utilizare si resurse***



Numerele sunt introduse utilizand switch-urile si de asemenea becurile sunt aprinse daca bitul are valoarea ‘1’. Numerele sunt date in complement fata de 2, ceea ce face ca operatiile sa fie posibile atat pe numere positive, cat si pe cele negative.

Selectia operatiilor se realizeaza utilizand butoane: cel din mijloc pentru adunare, cel de sus pentru scadere, cel din dreapta pentru inmultire sic el din stanga pentru impartire.

Rezultatul este afisat pe afisorul 7-segmente, primul reprezentand semnul(b-negativ si celelalte reprezinta numarul in zecimal. Pentru afisorul hexazecimal, toate cifrele reprezinta numere.

Fisierul de constrangeri este preluat de pe pagina oficiala a placii si este unul standard.

De asemenea, toate operatiile inclusive impartirea sunt perfect realizate in VHDL. Presupun ca problema este la bucle (for loop) deoarece este afectata frecventa afisorului.

Posibilitati de dezvoltare ulterioara

Cea mai necesara dezvoltare ulterioara este efectuarea operatiilor pe numere reale. Se vor putea folosi algoritmi mult mai eficienti pentru realizarea operatiilor, de exemplu algoritmul Booth pentru inmultire, care functioneaza atat pe numere positive, cat si pe numere negative.

De asemenea, calculatorul poate fi dezvoltat pentru a realiza operatii mult mai complicate.

Pentru acest proiect, cea mai buna dezvoltare ar fi sa functioneze correct pe placa, afisand rezultatele in forma zecimala, chiar daca algoritmul pentru conversie implica o bucla.

Resurse

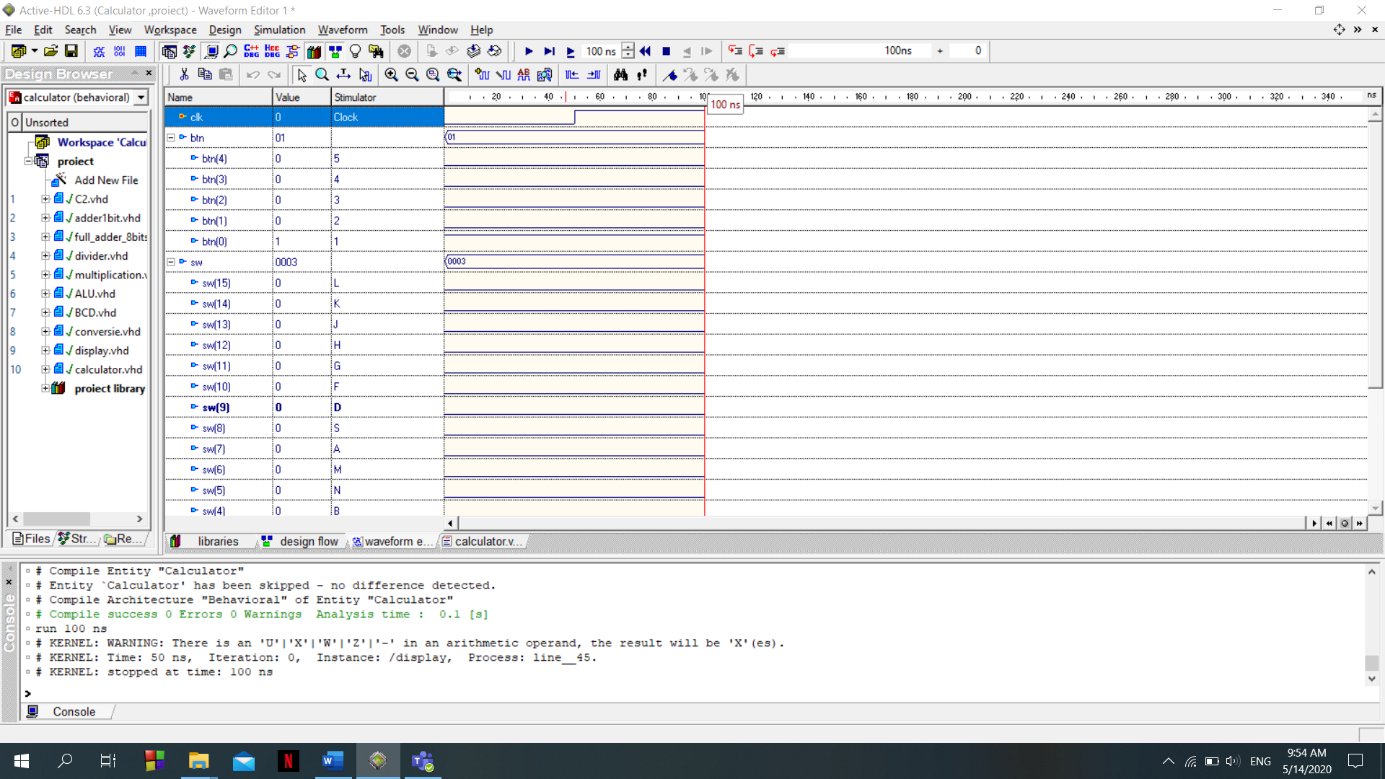
<https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/Basys3/Supporting%20Material/Basys3_Master.xdc>

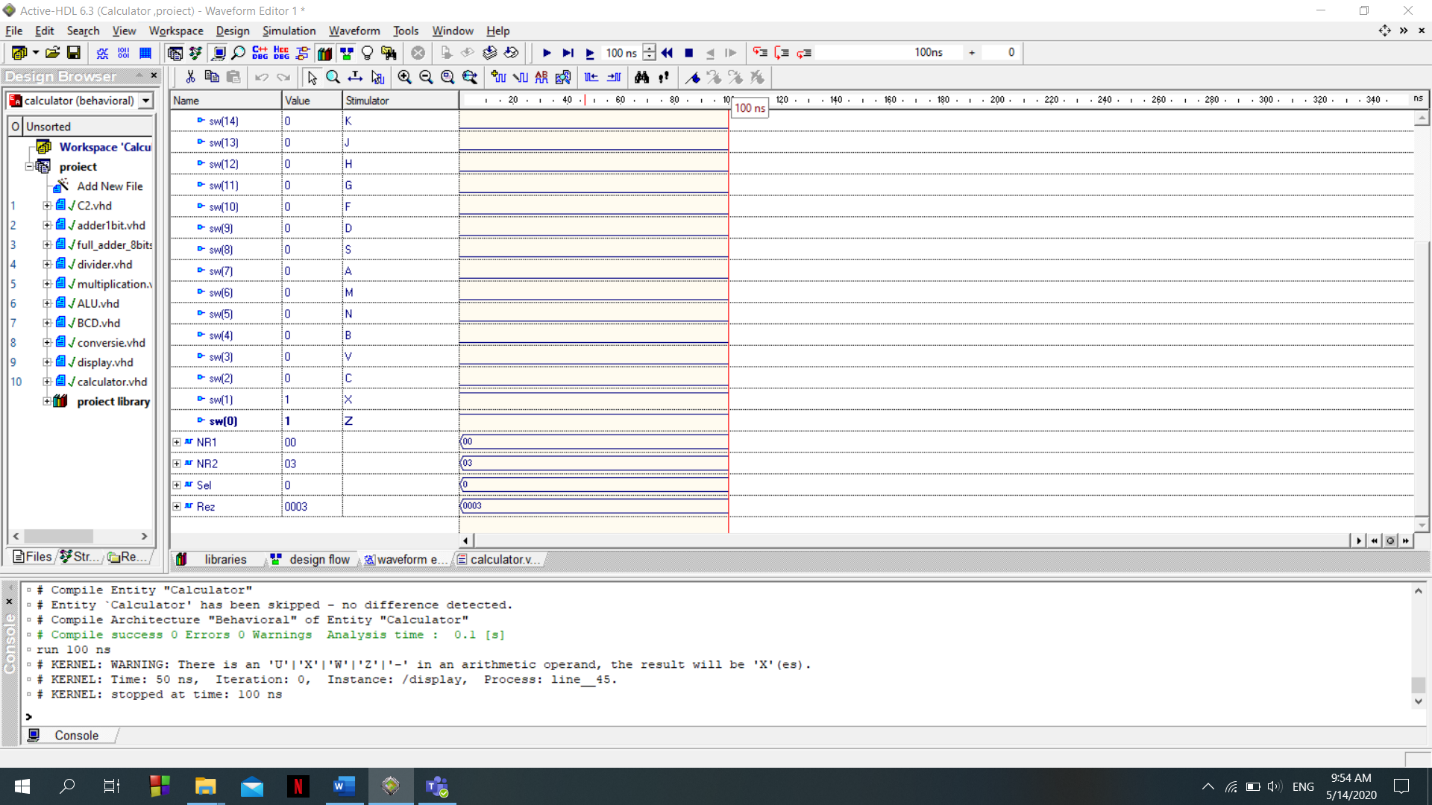
<https://www.google.com/search?q=basys+3&rlz=1C1CHBF_enRO847RO847&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiTy-yTw6riAhWNl4sKHb6pBjgQ_AUIDigB&biw=808&bih=647>

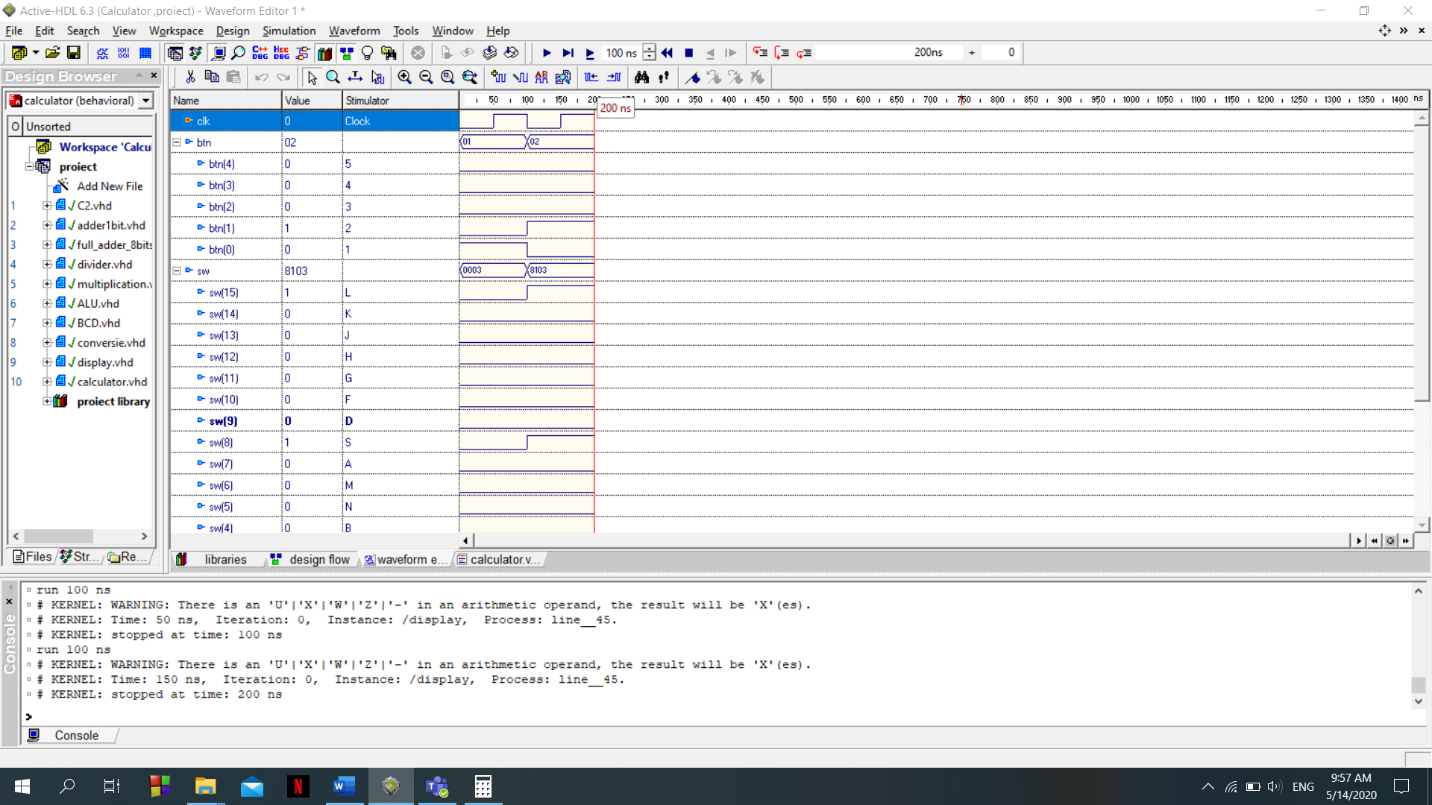
<http://users.utcluj.ro/~lucia/activdid.html>

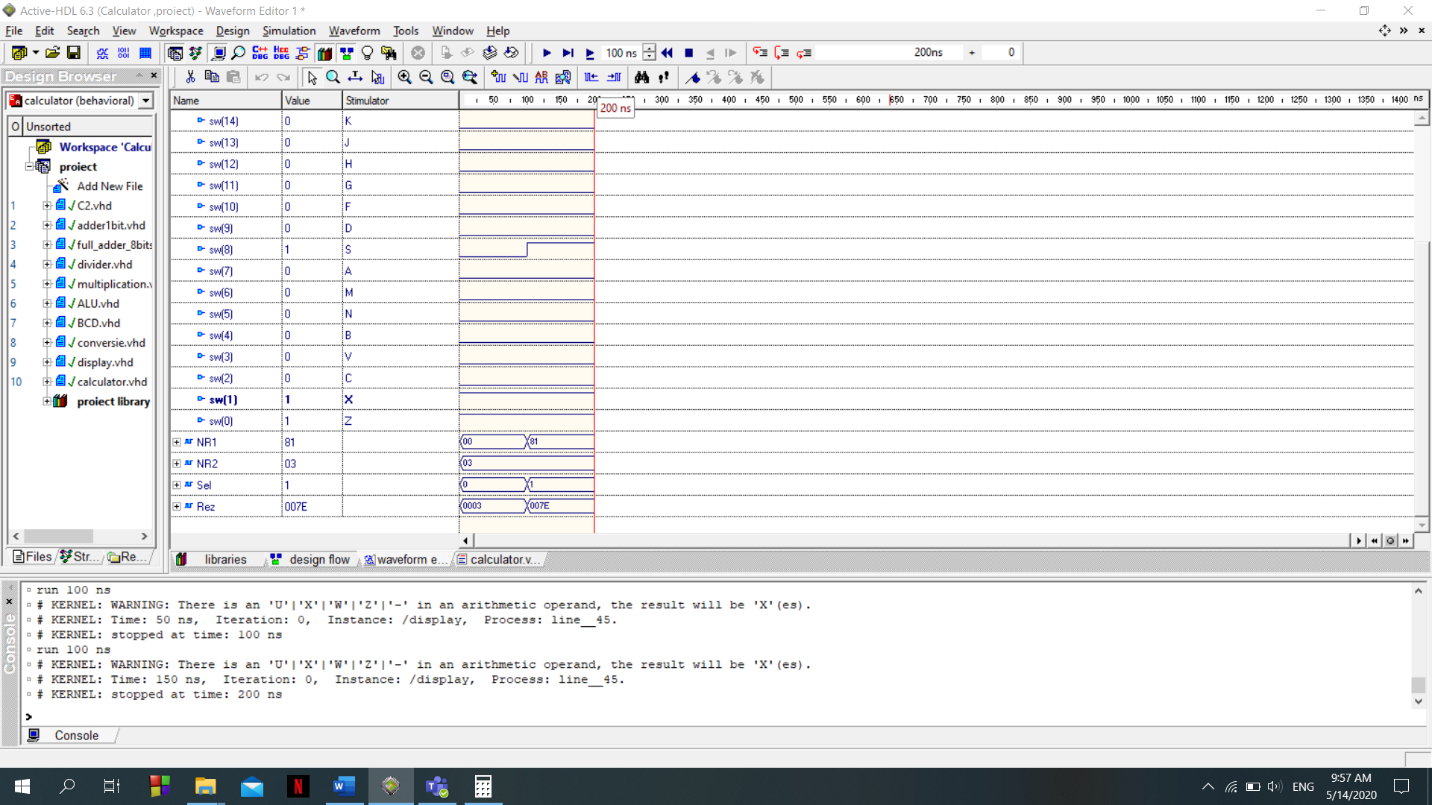
Simulari:

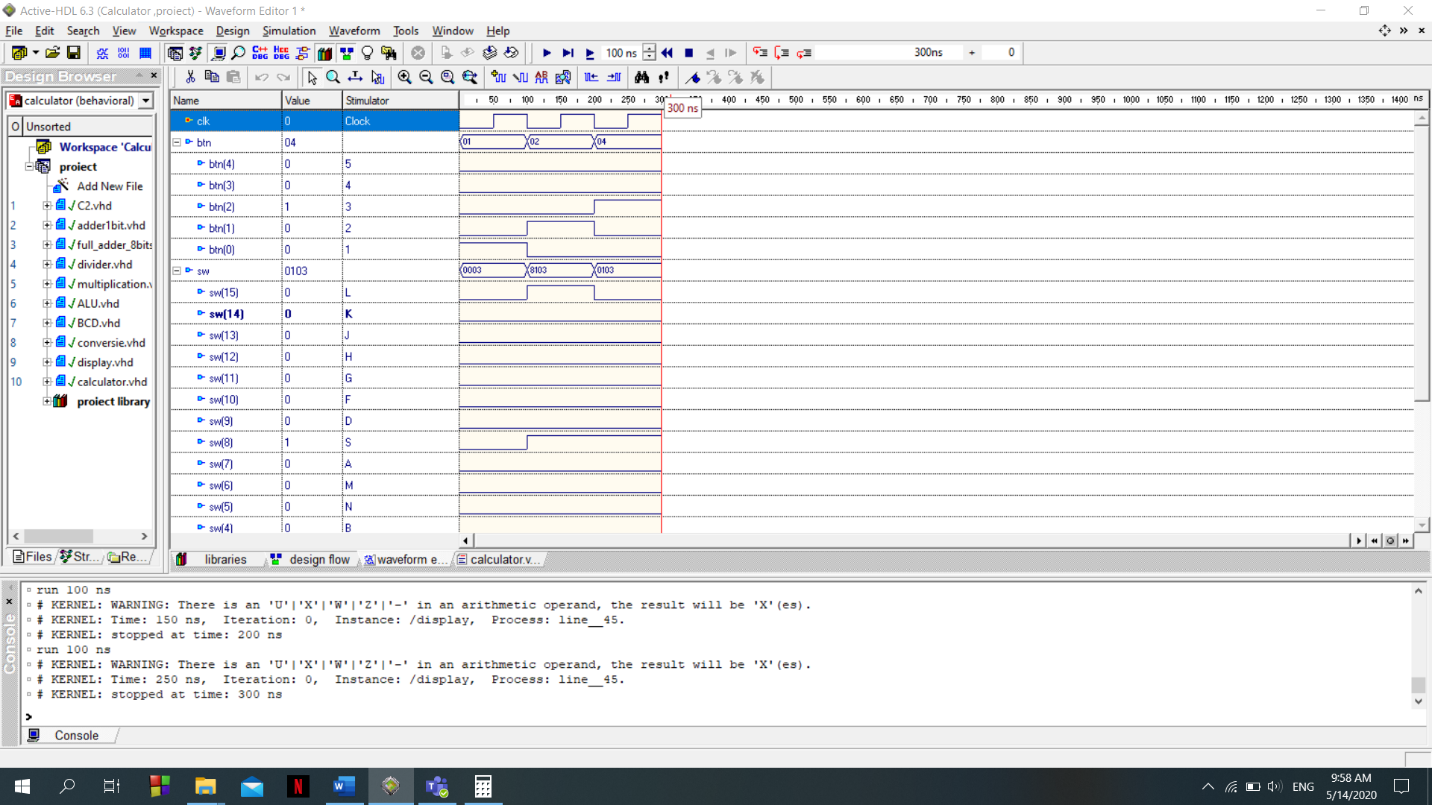
1)



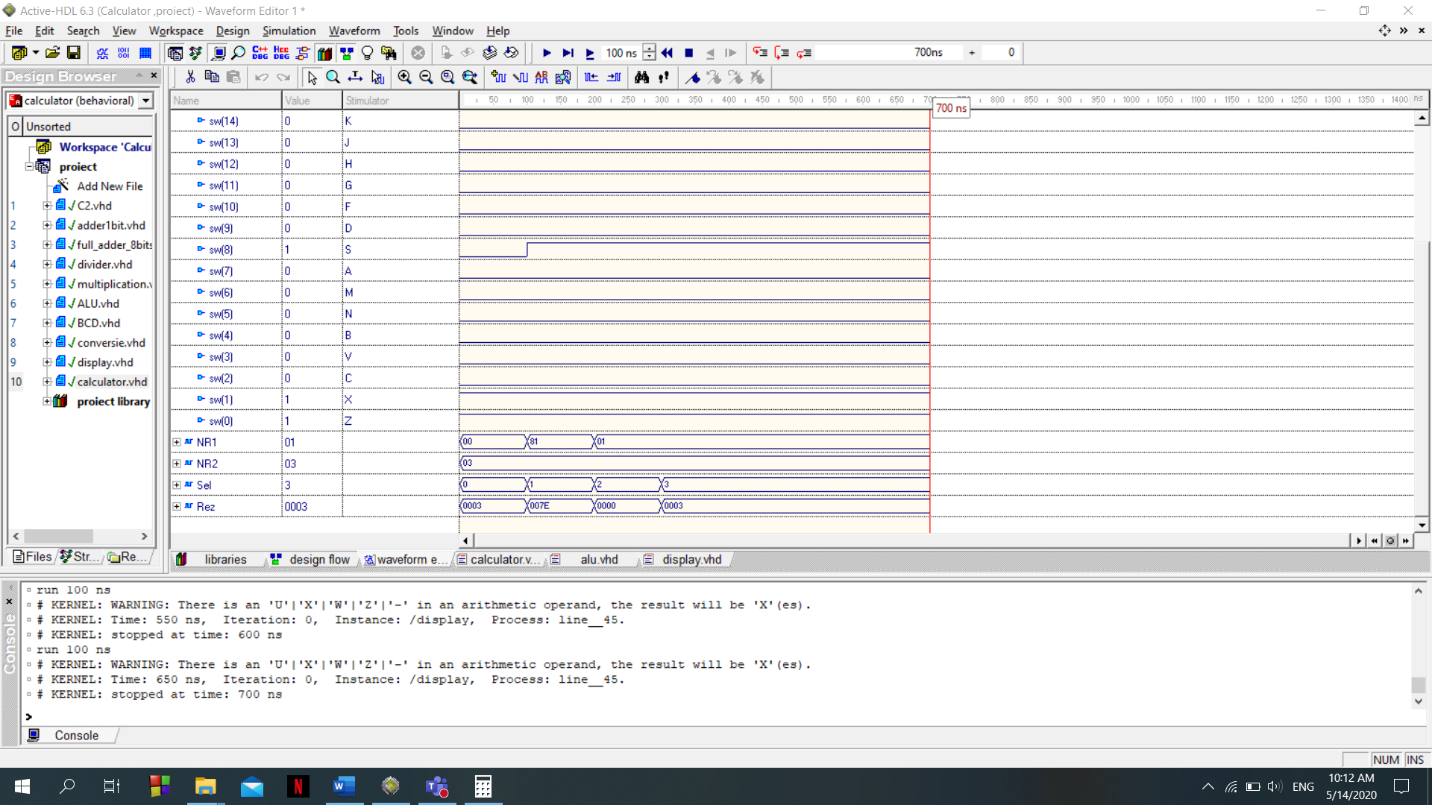


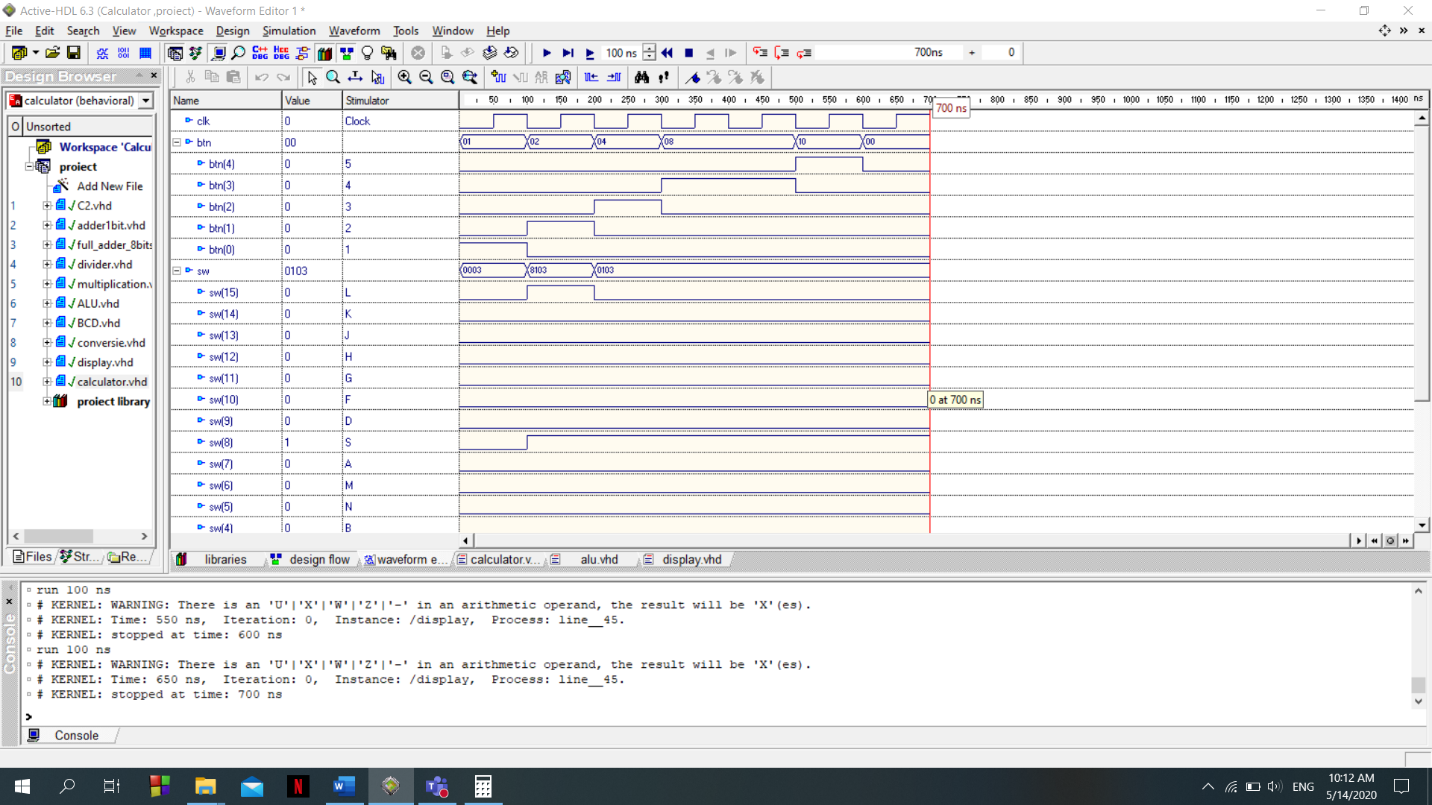
2) 



3) 



4) 



Sintetizare si implementare in vivado:

