UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

PROIECT: **APLICAȚIE COD CIFRU PENTRU SECURIZAREA DULAPURILOR**

PENTRU: **PROIECTAREA SISTEMELOR NUMERICE**

STUDENȚI: **MORAR TIMOTEI CRISTIAN**

TIUCA MARIA DANIELA

PROFESOR: **DIANA IRENA POP**

CUPRINS

1. SPECIFICAȚIE

Descriere: Să se implementeze o aplicație care permite utilizatorului adăugarea unui cifru din 3 caractere distincte pentru securizarea unui dulap (asemănător dulapurilor folosite la vestiarele de sport, mall, etc)

Cerințe funcționale:

1. Un led LIBER_OCUPAT va avea funcția de a semnala faptul că dulapul este liber (led stins) sau ocupat (led aprins)
2. Utilizatorul va apăsa un buton ADAUGA_CIFRU pentru a semnala începerea introducerii codului. Un led INTRODU_CARACTERE se va aprinde pentru a marca starea
3. Utilizatorul va adaugă pe rând 3 caractere cu ajutorul butoanelor UP și DOWN
4. Caracterele sunt cuprinse în intervalul 0-1-..-8-9-A-B-..-F
5. Caracterul curent este afișat pe SSD
6. Pentru trecerea la următorul caracter utilizatorul va apăsa butonul ADAUGA_CIFRU
7. Caracterul anterior introdus rămâne afișat
8. Următorul caracter este vizibil pe afișaj pe poziția următoare
9. După introducerea celui de al treilea caracter, la apăsarea butonului ADAUGA_CIFRU, afișajul SSD se va stinge, iar cifrul va fi în starea blocat prin aprinderea ledului LIBER_OCUPAT.
10. Ledul INTRODU_CARACTERE se va stinge
11. Existența unui buton/switch RESET în timpul introducerii cifrului pentru revenire în starea inițială (ledul LIBER_OCUPAT se va stinge, afișajul SSD este gol, ledul INTRODUC_CARACTERE se va stinge)
12. Utilizatorul va apăsa butonul/switch ADAUGA_CIFRU pentru a începe introducerea codului pentru deblocarea cifrului
13. Se vor relua pașii 2-8
14. La introducerea ultimului caracter, la apăsarea butonului ADAUGA_CIFRU se va face verificarea, daca codul introdus corespunde cu codul anterior.
15. În cazul de egalitate, ledul LIBER_OCUPAT se va stinge, ledul INTRODU_CARACTERE se va stinge, afisajul SSD se golește
16. În cazul de inegalitate, ledul LIBER_OCUPAT va rămâne aprins, ledul INTRODU_CARACTERE se va stinge, afișajul SSD se golește

Cerințe non-funcționale:

- Implementare pe plăcuță
- Utilizare SSD
- Utilizare switch, led, butoane cu frecvența de 1 Hz.

2. SCHEMA BLOC

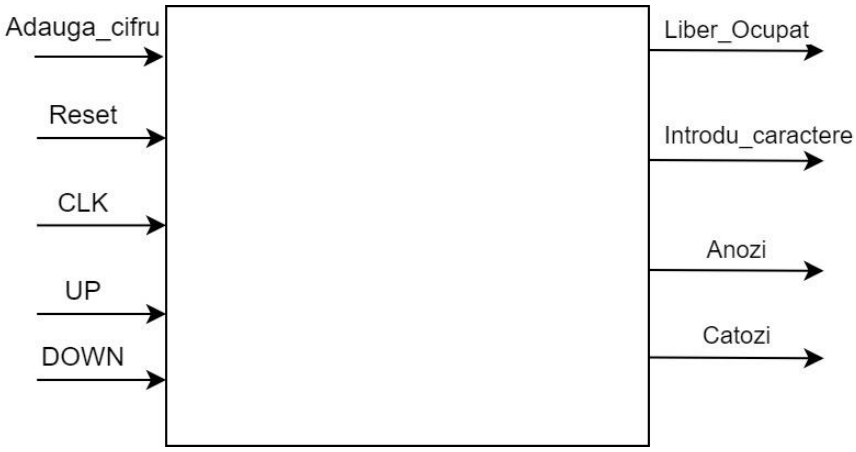
2.1 BLACK BOX – CUTIA NEAGRĂ

Intrări:

- 3 butoane: Adaugă_cifru, UP și DOWN (1 bit)
- 1 switch: Reset (1 bit)
- semnal de tact clk 100MHz

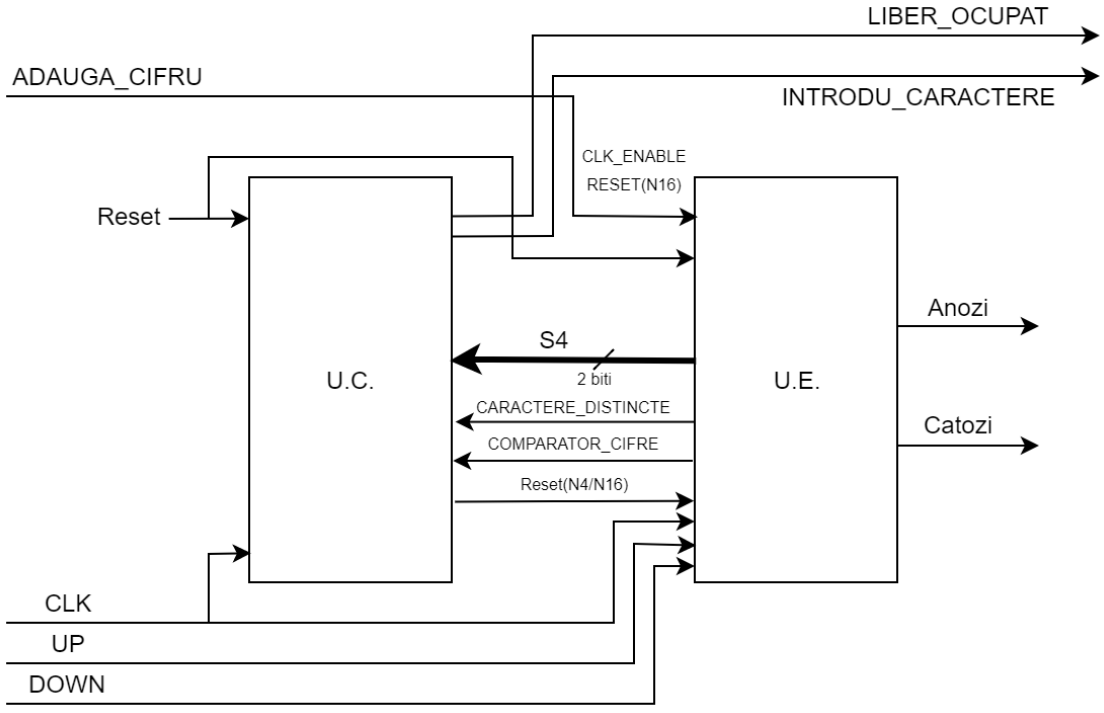
Ieșiri:

- 3 afișoare 7 segmente
- 2 leduri: Liber_ocupat si Introdu_caractere



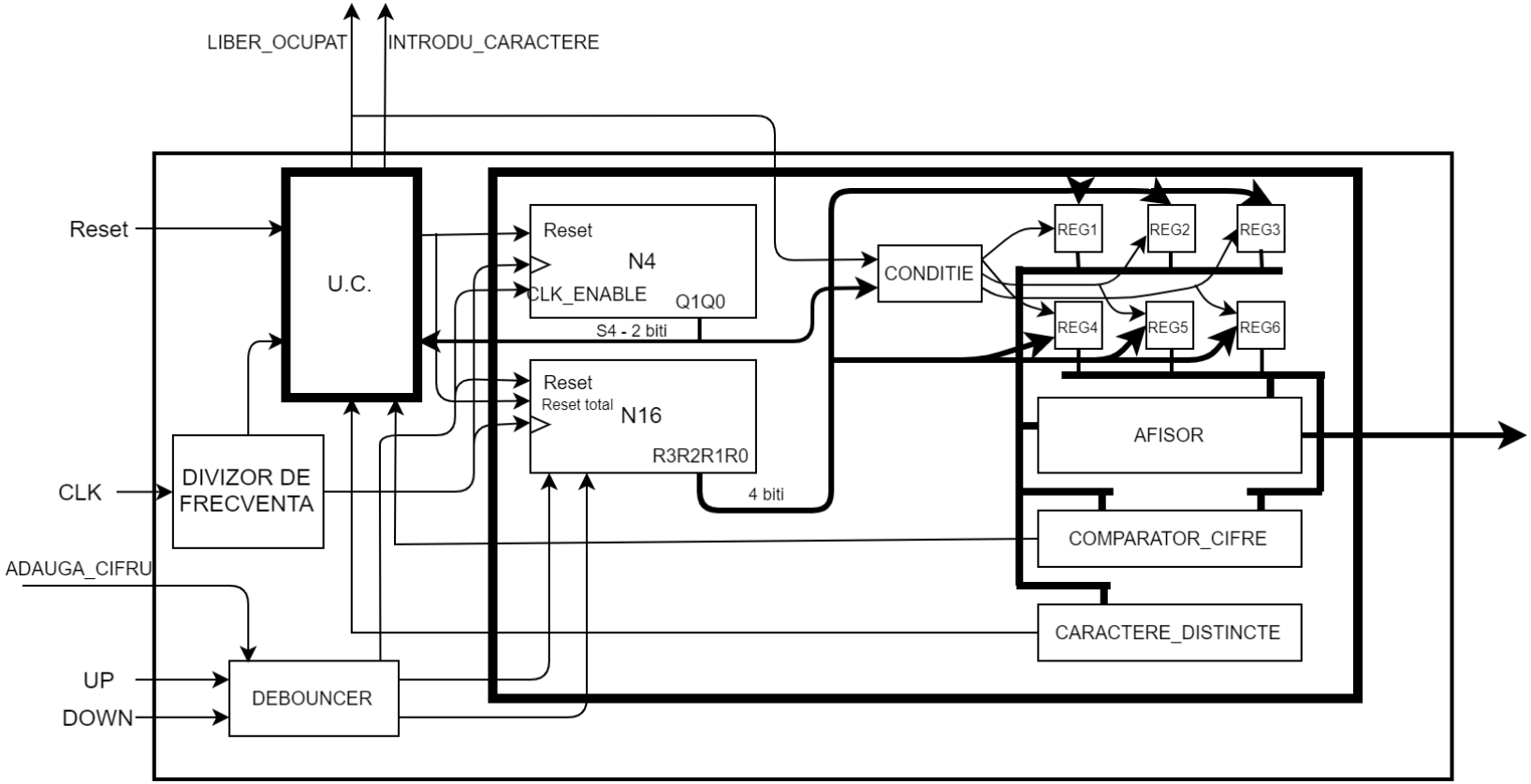
Ca și intrări de control avem: **adaugă_cifru** și **reset**, iar ca și intrări de date avem: **up** și **down**.

Ca și ieșiri de control avem: **liber_ocupat** și **introdu_caractere**, iar ca și ieșire de date avem **cifrul** care se afișează pe SSD.



2.2 DESCOMPUNEREA UC ȘI UE

Pentru a simplifica schema precizăm că butoanele ADAUGA_CIFRU, UP și DOWN au fost introduse într-un singur debouncer, iar resursele: debouncer și divizor de frecvență au fost plasate în afara unității de execuție, deși acestea aparțin în mod evident unității.



Unitatea de comandă (UC) este cea care controlează în mod individual fiecare resursă din unitatea de execuție UE, furnizându-i acesteia din urmă semnalele de comandă pentru toate resursele sale. Pentru a lua decizii, UC are nevoie să citească atât variabilele de intrare ale sistemului, cât și starea resurselor din UE.

Atât intrările cât și ieșirile sistemului pot fi citite/generate atât în/din UC cât și în/din UE.

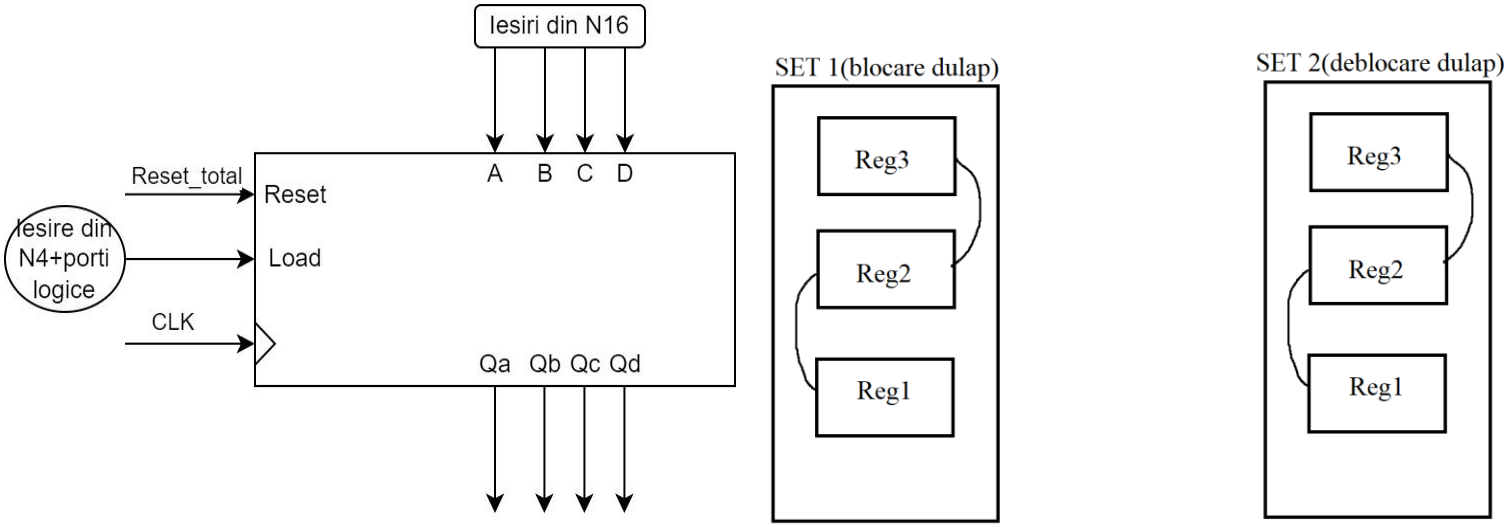
Pentru realizarea proiectului am folosit:

- Regiștri
- Numărător modulo 16
- Numărător modulo 4
- Logica combinațională pentru unirea componentelor
- Debouncer
- Divizor de frecvență 100MHz $\longrightarrow$ 1 Hz
- SSD

2.3 LISTA RESURSELOR

1. **Registre**(Morar)

Pentru memorarea cifrului vom utiliza 2 seturi a câte 3 registre, unul dintre seturi va conține primul cifru pentru **blocarea** dulapului, iar al doilea set va conține cel de-al doilea cifru folosit pentru **deblocarea** dulapului.



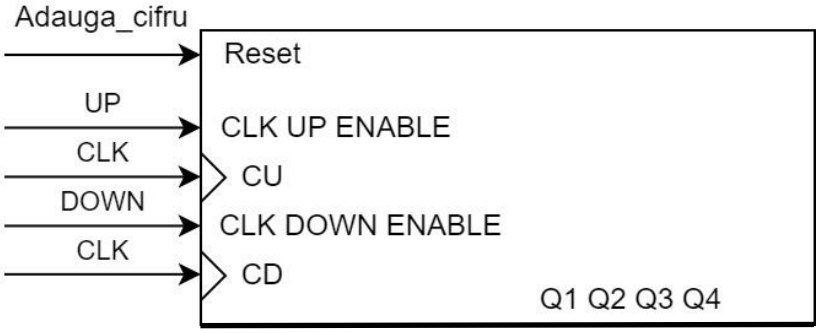
Fiecare din regiștrii utilizați în unul dintre seturile precedente (Set1 sau Set2) va avea aceeași structură și anume cea de mai sus. *Resetăm* conținutul regiștrilor atunci când dorim să resetăm tot cifrul. *Pentru a încărca* într-unul dintre regiștrii ne vom folosii de numărătorul modulo 4 (N4) și cu ajutorul unei logice combinaționale('CONDITIE') vom activa LOAD-ul corespunzător fiecărui registru în funcție de numărul de apăsării a butonului Adaugă_cifru.

Operație	Descriere	Reset_total	LOAD
CLEAR	(Qa, Qb, Qc, Qd) <= (0,0,0,0)	1	X
HOLD	(Qa, Qb, Qc, Qd) <= (Qa, Qb, Qc ,Qd)	0	0
LOAD	(Qa, Qb, Qc, Qd) <= (A, B, C, D)	0	1

De asemenea, pentru a nu reseta și primul set de regiștri atunci când suntem la cel de-al doilea cifru, utilizăm 2 reseturi diferite(reset_registri1 & reset_registri2) pe care le activăm în funcție de starea în care ne aflăm în UC.

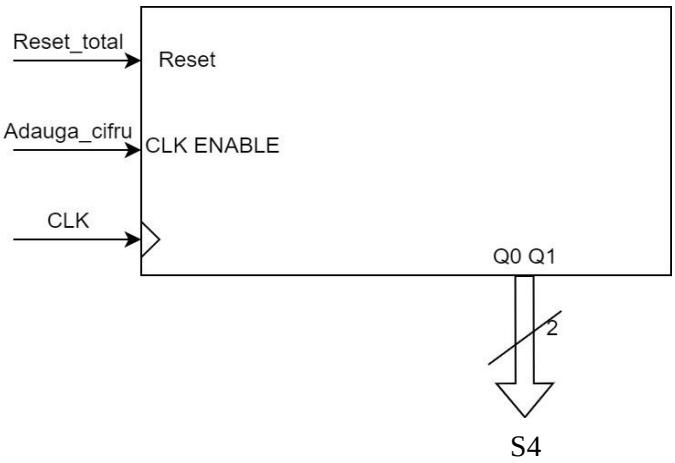
2. **Numărător reversibil modulo 16**(Tiuca)

Avem nevoie de un numărător modulo 16 pentru afișarea succesivă a simbolurilor din intervalul 0-...-9-A-...-F. Astfel acesta va număra 16 stări pe care le vom decodifica în hexazecimal. De asemenea acesta trebuie să fie reversibil datorită butoanelor de UP și DOWN care stabilesc sensul numărătorul (crescător/descrescător). Pentru fiecare iterație ieșirile vor fi salvate în registrele discutate anterior.



3. **Numărător modulo 4**(Morar)

De asemenea vom avea nevoie de un numărător modulo 4 care să contorizeze numărul de apăsări a butonului 'adauga_cifru'. Așadar 'adauga_cifru' va fi clock enable pentru acest numărător, iar resetul_total din Black box mare va fi conectat la resetul din numărător.

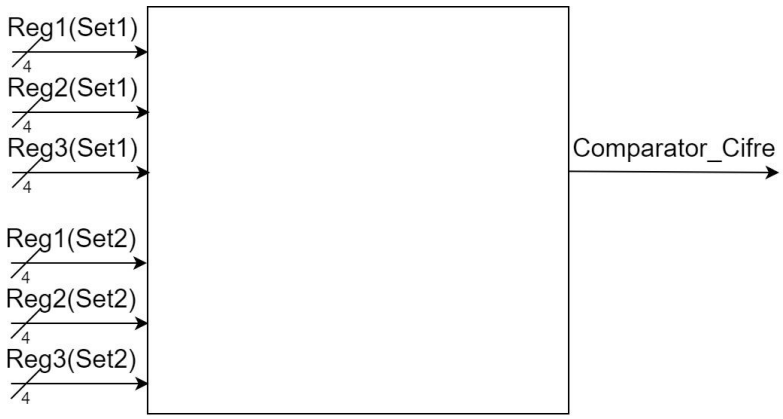


4. **Verificare caractere distincte**(Morar)

Vom realiza un circuit pentru verificarea condiției ca toate caracterele introduse care formează un cifru să fie distincte. Pentru descompunerea acestui circuit vom utiliza un algoritm simplu de comparare a conținutului din registre folosind un if.

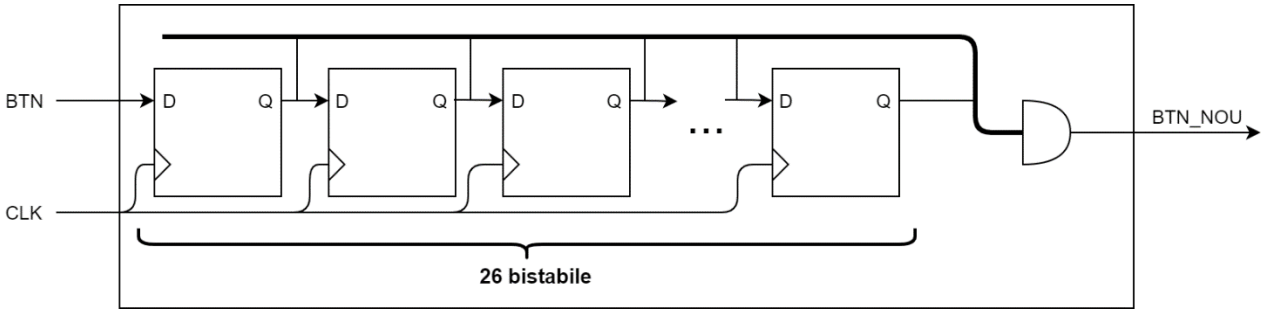
5. **Comparator cifre**(Tiuca)

Scopul acestui circuit este de a debloca dulapul. Dacă cifrul introdus la blocare coincide cu cel introdus pentru deblocare atunci dulapul se va deschide. Pentru descompunerea acestui circuit vom utiliza un algoritm simplu de comparare a conținutului din registre folosind un if.



6. **Debouncer** (Tiuca)

Vom utiliza un debouncer pentru stabilizarea semnalelor primite de la butoanele plăcuței. Acesta constă în cascadarea a 12 bistabile D, ieșirea bistabilului devenim noua intrare pentru următorul. La final, pe toate ieșirile se aplică "ȘI"(&) logic. Acest semnal va fi "noul buton".



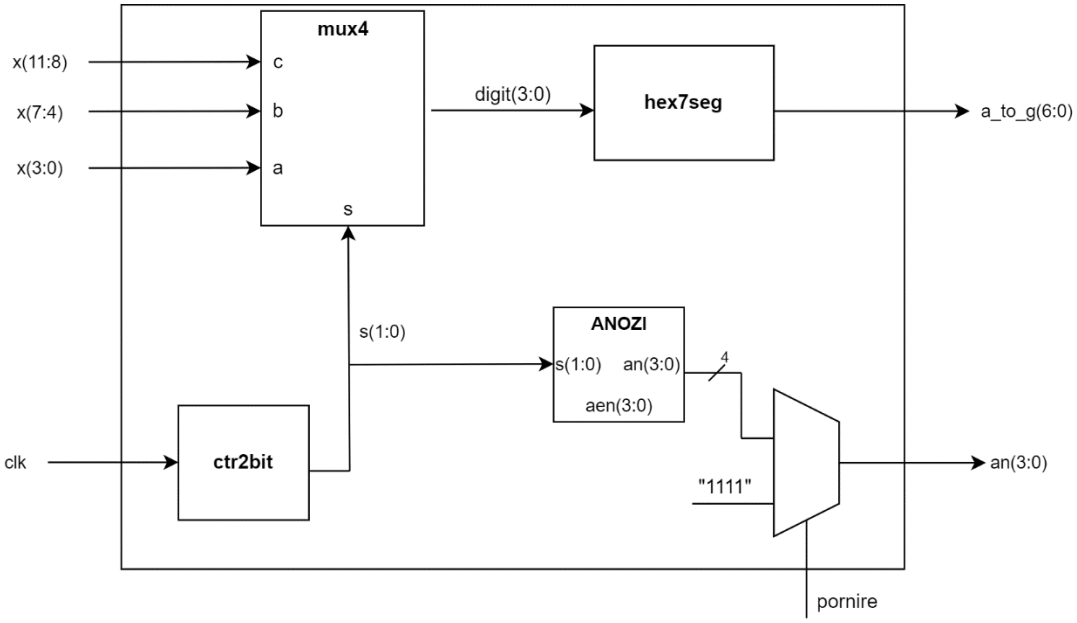
7. **Divizor de frecventa**(Tiuca)

Divizorul de frecvență este obținut prin realizarea unui numărător pe 26 de biți cu scopul de a se crea o întârziere de 1 secundă. Acest nou clock întârziat va rezulta din bitul cel mai semnificativ al numărătorului, adică bitul 26.

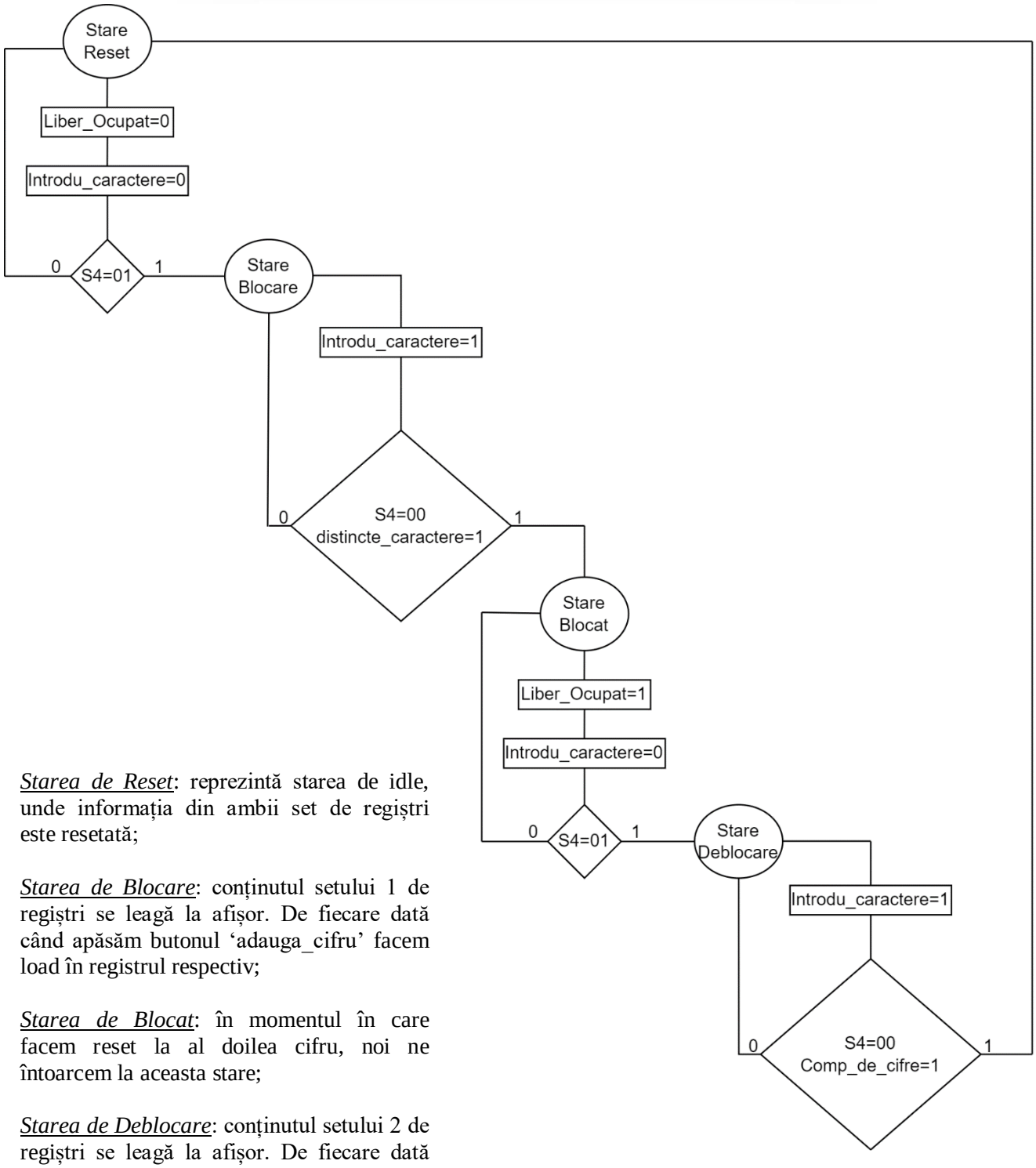
8. **7 Segment Display**(Morar)

În componența afișorului avem un multiplexor 3:1 cu lățimea căii de date de 4 biți pe care îl utilizăm pentru a accesa conținutul regiștrilor în care se află cifrul. Din acesta avem o ieșire care intră în decodificatorul hexazecimal și convertește numărul pentru a putea fi afișat pe SSD.

De asemenea, pentru ca toate caracterele să se afișeze concomitent folosim clock-ul de pe placă pe care îl divizăm pentru ca pe leduri să se poată percepe informația. Din acesta ieșirea s(1:0) intră in decodificatorul "anozi" care efectuează schimbarea rapidă a anozilor. Există o intrare aen(enable) prin care decidem câte din cele 4 leduri dorim să fie pornite. La final mai utilizăm un multiplexor 2:1 cu lățimea căii de date de 4 biți unde intrările sunt fie ieșirea din decodificator pentru activarea anozilor, fie "1111" care ne stinge complet afișorul. Mai departe din acest multiplexor iese valoarea anozilor.



3. ORGANIGRAMA



Starea de Reset: reprezintă starea de idle, unde informația din ambii set de regiștri este resetată;

Starea de Blocare: conținutul setului 1 de regiștri se leagă la afișor. De fiecare dată când apăsăm butonul ‘adauga_cifru’ facem load în registrul respectiv;

Starea de Blocat: în momentul în care facem reset la al doilea cifru, noi ne întoarcem la aceasta stare;

Starea de Deblocare: conținutul setului 2 de regiștri se leagă la afișor. De fiecare dată când apăsăm butonul ‘adauga_cifru’ facem load în registrul respectiv;

4. JUSTIFICAREA SOLUȚIEI ALESE

Considerăm că soluția aleasă este cea care se aproprie cel mai mult de realitate. Utilizarea Cifrului de dulap este foarte lizibilă, iar procesul de construcție și logica din spatele lui sunt ușor de înțeles, utilizând cunoștințe elementare de Proiectare logică și Proiectarea Sistemelor Numerice.

Pentru memorarea cifrului utilizăm regiștri ca soluție optimă, deoarece putem scrie și citi într-un mod mai simplu, spre exemplu dacă utilizăm memorii RAM aveam nevoie de un semnal auxiliar prin care să știm dacă citim sau scriem în memorie. Utilizarea a 2 numărătoare(numărător modulo 16 și numărător modulo 4) simplifică modul prin care putem să selectăm simbolul dorit în cifru, precum și trecerea la caracterul următor. Pentru a ajuta la sincronizarea acestora, precum și pentru a vedea în timp real modul cum se schimbă simbolurile, utilizăm clock-ul plăcii divizat pentru a obține aproximativ o secundă. De asemenea am optat pentru utilizarea butoanelor ca metodă de selecție deoarece este mai natural utilizatorului să folosească butoane în loc de switch-uri. Pentru a putea realiza aceasta folosim un debouncer care să ne stabilizeze butoanele. Pentru legarea tuturor componentelor se folosește o logică combinațională.

5. MANUAL DE UTILIZARE ȘI ÎNTREȚINERE

Basys 3

Placa de dezvoltare Basys3 face parte din familia Artix-7 FPGA(XC7A35T-1CPG236C) a celor de la Xilinx și este cea mai nouă din gama Basys.

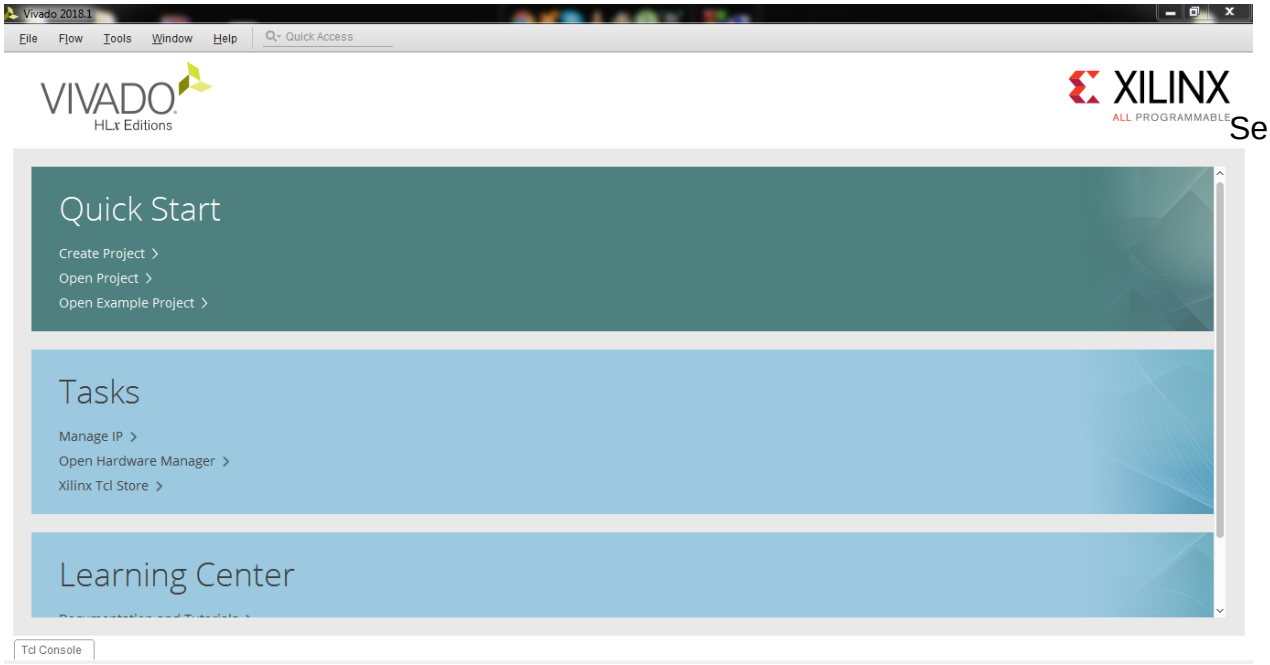
Placa dispune de 33.280 celule organizate în 5200 de slice-uri, fiecare slice conținând 4 LUT-uri cu câte 6 intrări fiecare și 8 bistabile. De asemenea, placa conține cristale quartz care generează un semnal de clock(tact) de 100 MHz. Din punct de vedere al interferenței cu clientul, placa dispune de 16 LED-uri, 16 switch-uri, 5 butoane, 4 afișoare BCD-7 segmente, un port VGA și un port USB HID. Mai mult decât atât, placa poate fi programată fie prin portul Micro-USB (Modul JTAG), fie prin portul USB-UART cu ajutorul unei memorii flash. Programarea acestei plăci se face exclusiv cu programul Vivado Design Suite în limbajul VHDL.

➢ *Pașii necesari pentru utilizare*

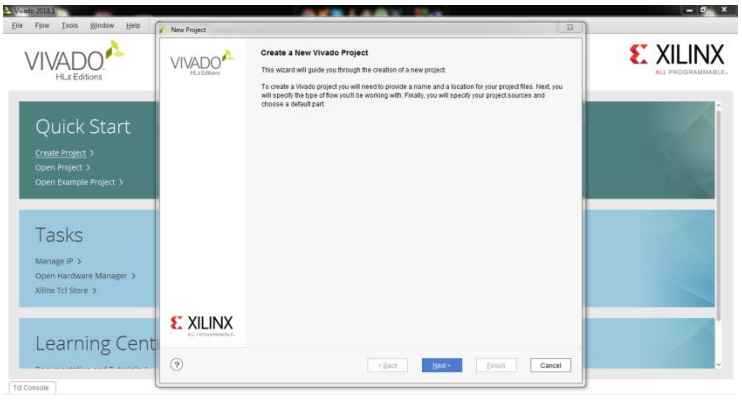
Pentru a putea utiliza acest proiect este nevoie de programul Vivado Design Suite, dar și de o plăcută Basys 3.

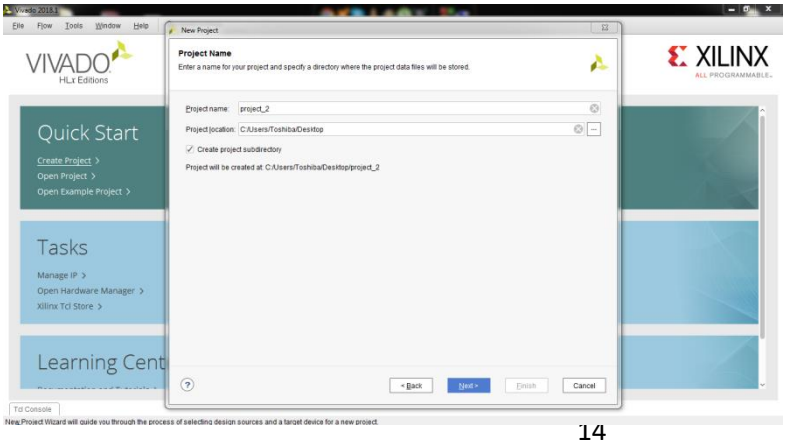
În cazul în care utilizatorul nu are codul sursă într-un program cu terminația .vhd, va trebui să îl creeze cu ajutorul programului Active-HDL și să îl compileze .

În următoarele imagini se va prezenta pas cu pas crearea unui nou proiect în VIVADO și progamarea plăcuței FPGA. În primul rând se creează un proiect nou.

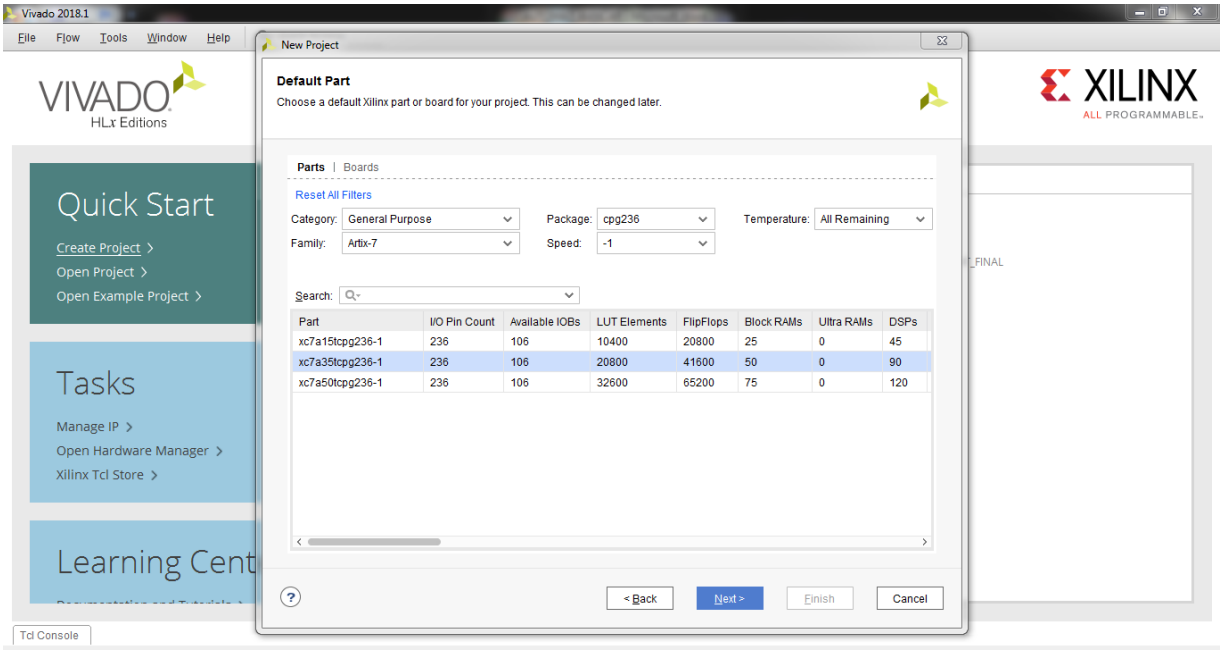


Se deschide o fereastră unde apăsăm next, iar apoi se alege un nume și selectăm next.

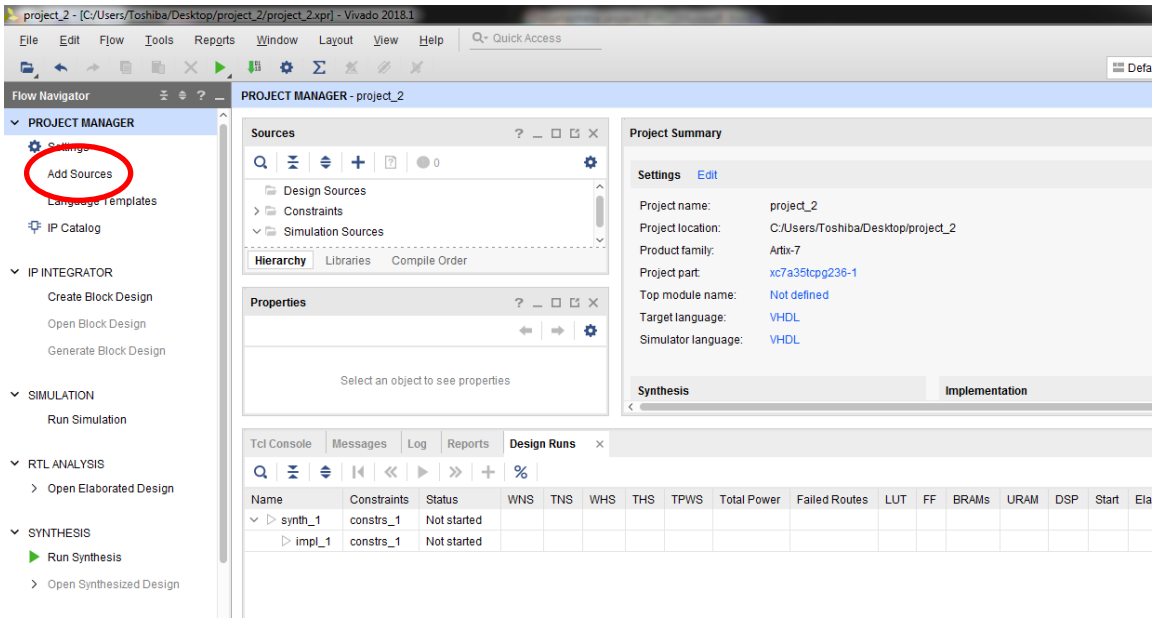




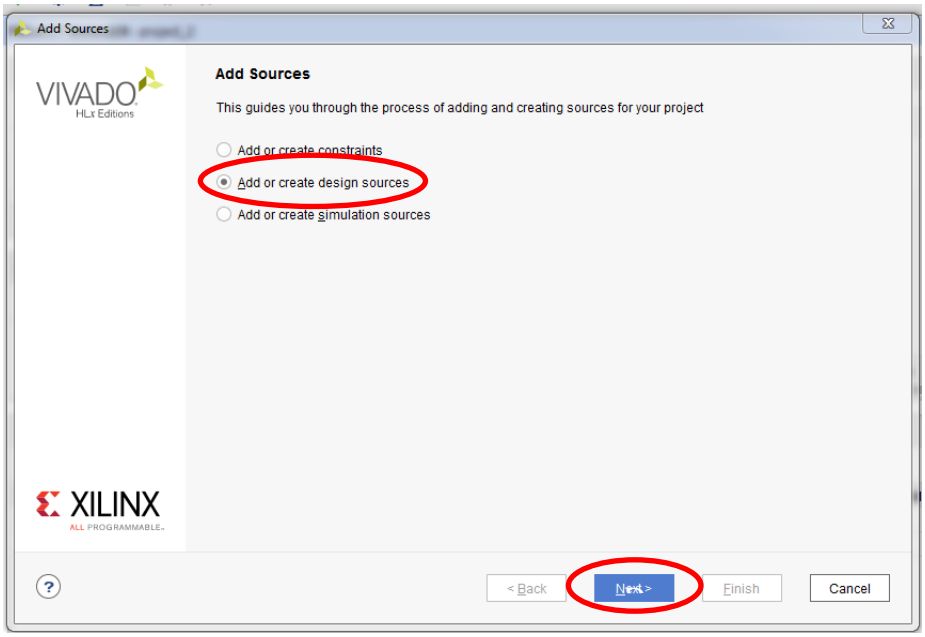
Next, iar în continuare se completează după cum urmează, iar apoi se apasă next și finish.

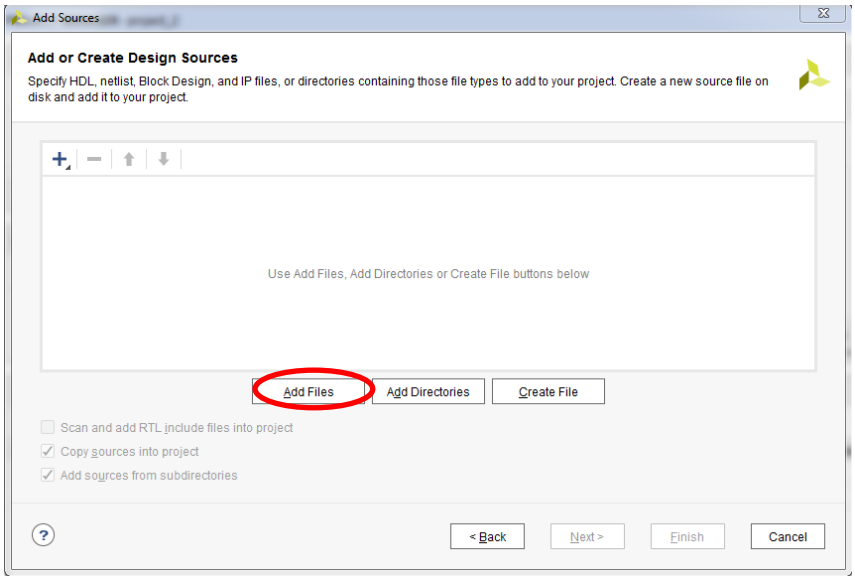


Se introduce codul VHDL scris în ACTIVE-HDL cu ajutorul butonului ADD SOURCES:



Apoi cu Add Files:



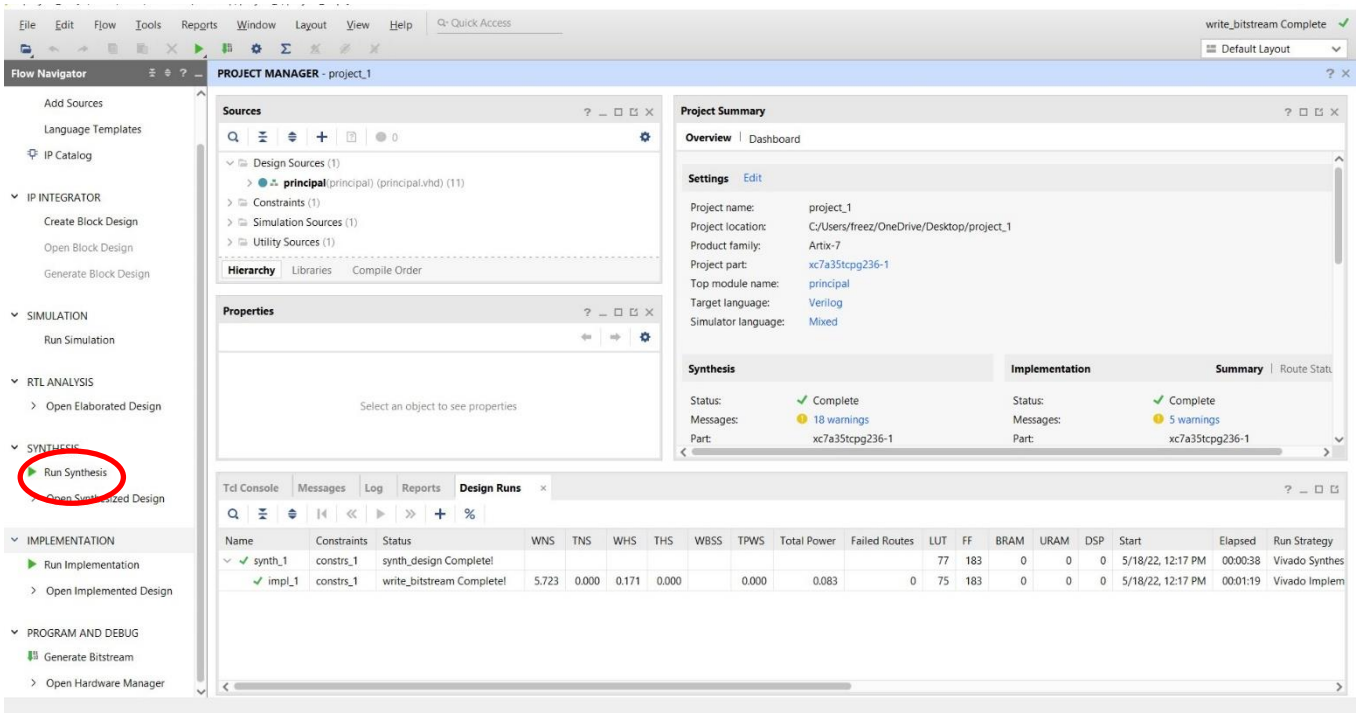


Se selectează fișierele și se apasă ulterior pe finish:

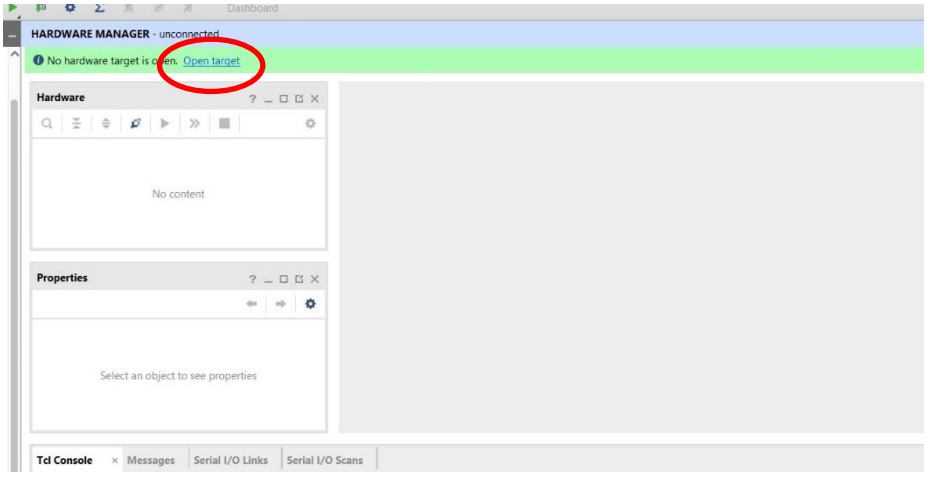
În continuare, se revine la opțiunea Add Sources =>Add or create constraints și se realizează fișierul de constrângeri.

Pentru placuța Basys 3 am descărcat fișierul const.xdc și am făcut modificările necesare.

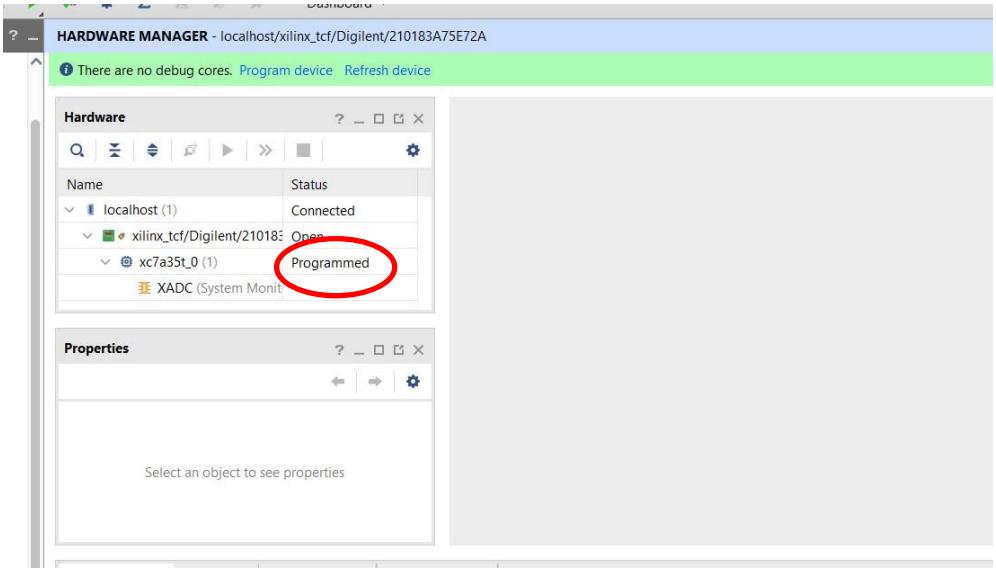
După apăsăm dublu click pe „Run Synthesis” și așteptăm. După terminarea procesului de sinteză, va apărea o fereastră, selectăm „Run Implementation” și din nou așteptăm terminarea rulării. După terminarea procesului de implementare selectăm opțiunea 2 din noua fereastră apărută „Generate Bitstream”, iar după terminarea procesului închidem fereastra nou apărută.

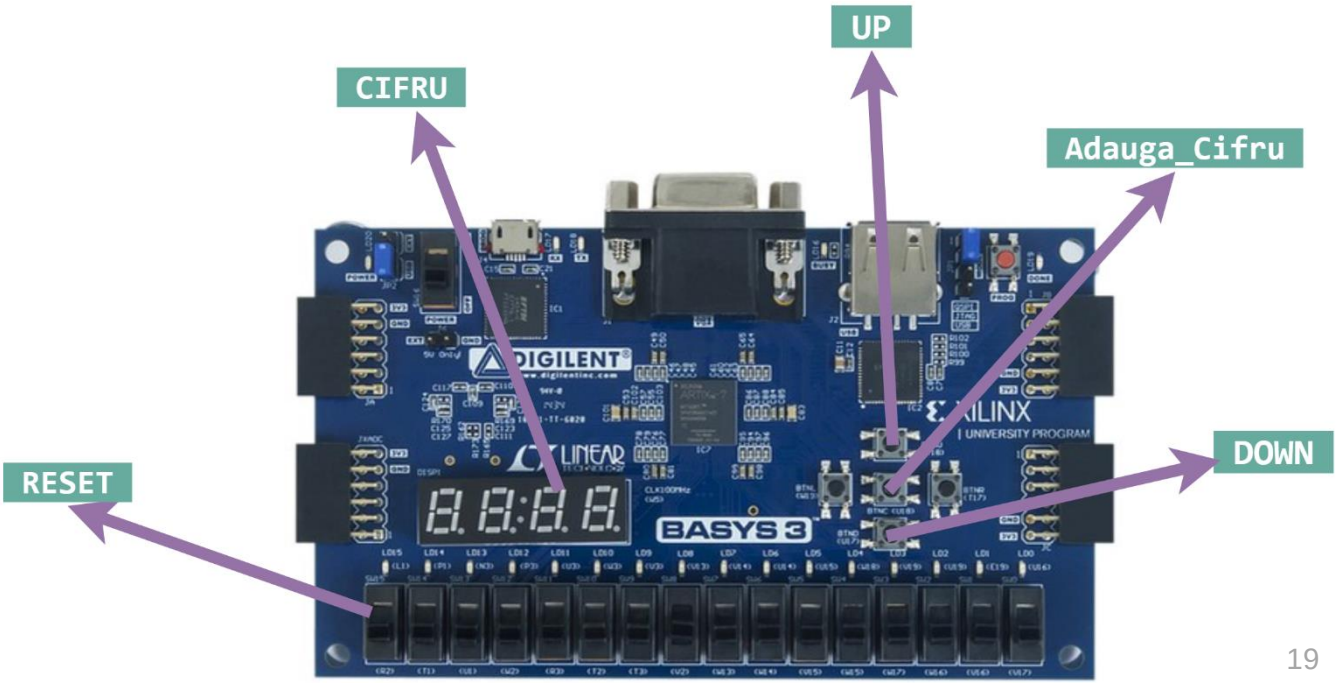


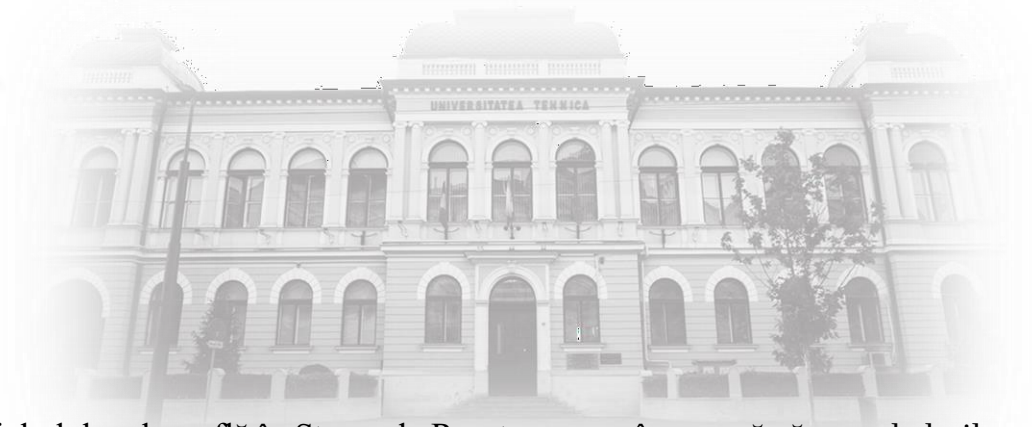
După aceea în fereastra din stânga se selectează “Open Hardware Manager”, și la Open Target alegem “Auto-Connect”. Din momentul acesta, dispozitivul dumneavoastră realizează operațiile necesare pentru a realiza conectarea la placa Basys3.



După terminarea acestei acțiuni, ar trebui să vă apară anumite denumiri, care reprezintă specificațiile plăcii(ca mai jos). Pe linia unde scrie Programmed/Not Programmed apăsăm click-dreapta, “Program Device” și Program. În momentul acesta codul dumneavoastră va fi încărcat pe placă, după finalizarea procesului.







Inițial, dulapul se află în Starea de Reset, ceea ce înseamnă că toate ledurile sunt stinse, de asemenea afișorul este stins, iar conținutul atât setului 1 cât și setului 2 de regiștrii este gol, acesta urmează să fie setat cu cifrul creat de către utilizator în pașii următori.

Pași:

1. Pentru a se începe setarea cifrului dorit utilizatorul trebuie să apese mai întâi pe butonul Adaugă_Cifru. Această acțiune va induce Starea de Blocare a dulapului. Afișorul se va aprinde arătând "000" împreună cu ledul Introdu_Caractere ce indică utilizatorului începerea introducerii codului dorit.

2. Utilizatorul se va folosi de butoanele UP și DOWN pentru a introduce caracterul dorit. Apăsarea continuă acestora va fi vizibilă pe afișor, butonul UP va crea apariția caracterelor în ordine crescătoare de la 0-9-A-F, iar butonul DOWN va crea apariția caracterelor în ordine descrescătoare. Când utilizatorul va ajunge la caracterul dorit pentru salvare este nevoit să mai apese încă o dată butonul Adaugă_Cifru. Această operație va face direct trimitere către următoarea poziție , iar caracterul salvat anterior va rămâne afișat.

3. Pasul 2. se va repeta pentru toate caracterele rămase. Singura diferență va fi la ultimul caracterul, deoarece la apăsarea butonului Adaugă_cifru pentru salvarea caracterului, acesta se va salva, dar afișorul și ledul Introdu_Caractere se vor stinge, iar ledul Liber_Ocupat se va aprinde.

Observații:

a) Dacă utilizatorul a greșit sau se răzgândește asupra cifrului introdus până când acesta este salvat, atunci poate reseta cifrul utilizând switch-ul destinat resetării. Atenție: această operație va șterge conținutul setului 1 de regiștrii, afișorul va afișa din nou "000", iar pentru reluarea pasului 2 și 3, utilizatorul este nevoit să mai apese o dată Adaugă_Cifru, în ciuda faptului că ledul Introdu_Caractere este aprins.

b) Întoarcerea la caracterul introdus anterior este imposibilă. Pentru a realiza această operație utilizatorul este nevoit să reseteze tot cifrul.

c) Dacă cifrul este complet și salvat nu se poate reseta.

d) La sfârșitul introducerii cifrului și apăsarea butonului Adaugă_Cifru, caracterele introduse sunt comparate pentru a se stabili distincția lor, o măsură ce sporește securitatea. Dacă acestea sunt distincte, cifrul se va salva, iar dulapul va fi blocat ceea ce înseamnă că ajunge în Starea de Blocat. Dacă această condiție nu este îndeplinită, cifrul se va reseta singur, conținutul afișorului va fi setat pe "000", iar utilizatorul este nevoit să introducă alt cifru, se reia pasul 2 si 3. Din nou, în acest caz începerea introducerii cifrului se va face prin apăsarea butonului Adaugă_Cifru, deși ledul Introdu_Caractere este aprins.

4. Ajuns în Starea de blocat ledul Introdu_Caractere și afișorul sunt stinse, iar ledul Liber_Ocupat este aprins, fapt ce marchează ocuparea dulapului. Dulapul va rămâne în această stare până la următoarea apăsare a butonului Adaugă_Cifru ceea ce va face trimitere la Starea de Deblocare.
5. În Starea de Deblocare ledul Liber_Ocupat este aprins și ledul Introdu_Caractere împreună cu afișorul se vor aprinde. Utilizatorul este nevoit să introducă cifrul deja salvat. Această operație se va realiza reluând pasul 2 pentru toate caractere.

   Observații:

   a. Dacă utilizatorul a greșit sau se răzgândește asupra cifrului introdus până la momentul apăsării butonului Adaugă_Cifru după ce ultimul caracter este introdus, atunci poate reseta cifrul utilizând switch-ul destinat resetării. Atenție: această operație va șterge conținutul celui de-al doilea cifru, afișorul va afișa din nou “000”, iar pentru reluarea pasului 2, utilizatorul este nevoit să mai apese o dată Adaugă_Cifru, în ciuda faptului că ledul Introdu_Caractere este aprins.
   b. Întoarcerea la caracterul introdus anterior este imposibilă. Pentru a realiza această operație utilizatorul este nevoit să reseteze tot cifrul introdus. Această operație nu va reseta primul cifru, doar pe al 2-lea.
   c. La sfârșitul introducerii cifrului și apăsarea butonului Adaugă_Cifru, cifrul introdus este comparat cu cel salvat în setul 1 de regiștrii pentru a se stabili egalitatea lor. Dacă acestea nu sunt egale, ultimul cifrul introdus se va reseta automat, afișorul va arăta “000”, ledurile vor rămâne aprinse. Dulapul se va afla în așteptare pentru introducerea cifrului pentru deblocare. Pentru a reîncepe introducerea caracterelor, utilizatorul este nevoit să apese încă o dată butonul Adaugă_Cifru și apoi poate relua pasul 2 pentru toate caracterele. Dacă cifrul introdus este egal cu cel inițial, atunci dulapul este deblocat, se deschide și ajunge înapoi în Starea inițiala de Reset.

Întreținere:

Se recomandă un număr minim de resetări asupra codurilor introduse.

De asemenea, pe parte materială, este recomandat verificarea periodică a componentelor sistemului, în deosebit funcționalitatea butoanelor: UP, DOWN, Adaugă_Cifru, și a elementelor de ieșire: ledurile Introdu_Caractere și Liber_Ocupat, afișorul.

6. POSIBILITĂȚI DE DEZVOLTARE ULTERIOARĂ

Acest cifru de dulap a fost creat la o scară mică, însă poate fi dezvoltat prin mărirea capacității regiștrilor(se utilizează mai mulți regiștrii pentru memorarea unui cod mai lung). De asemenea se pot adăuga mai multe caractere în librărie pentru blocarea cifrului, sporind astfel securitatea obiectelor.

Se poate adăuga o comandă în plus care să fie disponibilă doar personalului autorizat întreținerii dulapurilor securizate. Dacă se întâmplă ca cineva să își fi uitat codul, un reset care sa nu fie disponibil oricărui utilizator, ar fi soluția optimă care să poată reseta întru totul dulapul. Aceasta ar putea fi extinsă prin crearea unei interfețe specializată, care să restricționeze accesul utilizatorilor la toate instrucțiunile posibile.

7. BIBLIOGRAFIE

- Notițe curs Proiectare Logică An 1 Sem. 1
- Îndrumător laborator Proiectare Sistemelor Numerice An 1 Sem. 2
- Neso Academy
- LBEbooks