Detalii privind circuitele FPGA (Field Programmable Gate Array).

1. Introducere.

Obiective urmarite:

- principii,
- implementare,
- programarea circuitelor logice configurabile din punctul de vedere al proiectarii celulelor, cat si al strategiei de interconectare

FPGA reprezinta circuite integrate care pot fi programate de catre utilizator.

FPGA contin:

- functii versatile,
- interconexiuni configurabile,
- interfete de I/E adaptabile conform specificatiilor utilizatorului.

Structura uzuala a unui circuit FPGA este data in fig.1.

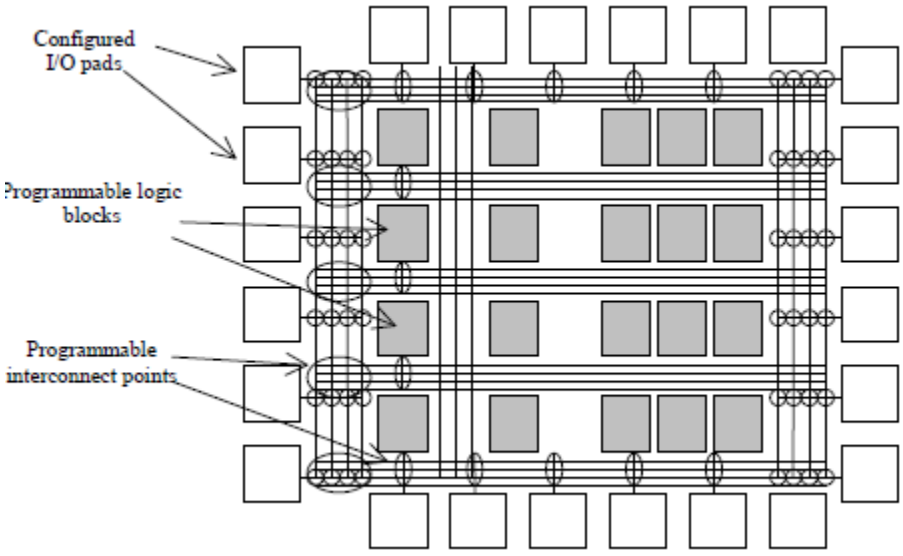


Fig.1. Structura uzuala a unui circuit FPGA este data in fig.1.

In figura 2 se prezinta implementarea unei functii simple (XOR cu 3 intrari):

- trei ploturi din stanga sunt configurate ca intrari,
- un bloc logic este utilizat pentru a crea un XOR cu 3 intrari,
- un plot din dreapta este configurat ca iesire,
- liniile de interconectare programabile asigura propagarea semnalelor in sistem.

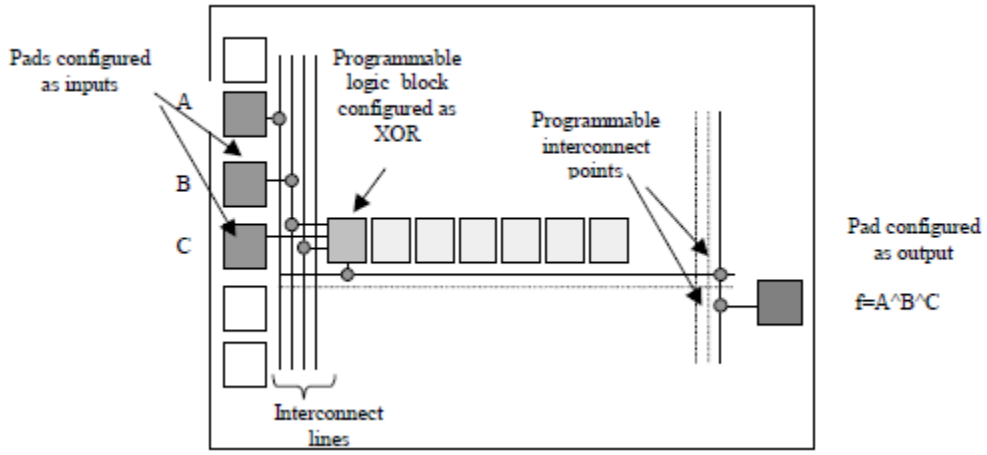


Fig.2 Implementarea unei functii XOR cu 3 intrari

Circuitul echivalent, corespunzator implementarii din figura 2, este prezentat in figura 3.

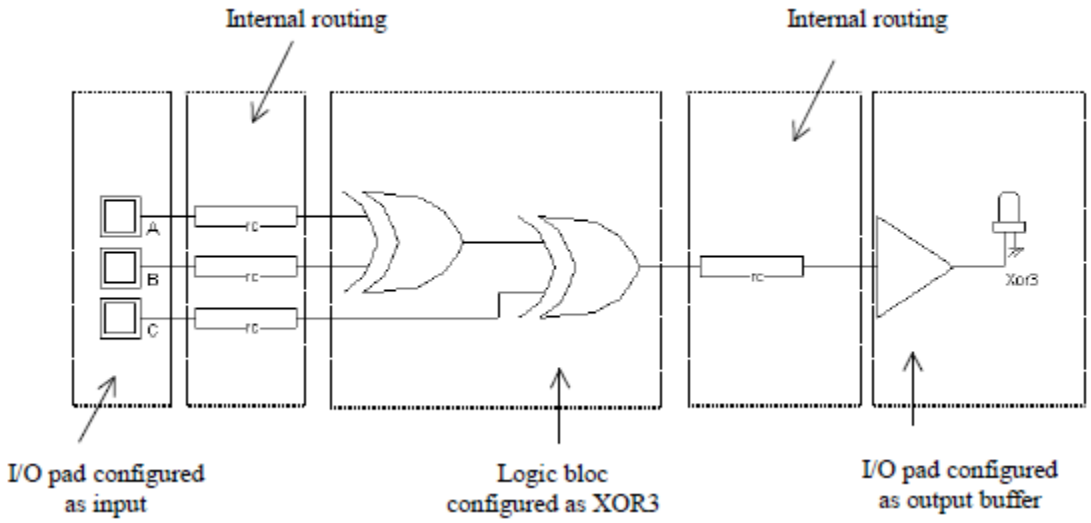


Fig.3. Circuitul echivalent, corespunzator implementarii din figura 2.

Circuitele FPGA se prezinta ca simple componente sau ca macroblocuri, in proiectele de sisteme pe o singura “pastila/chip” (Fig.4):

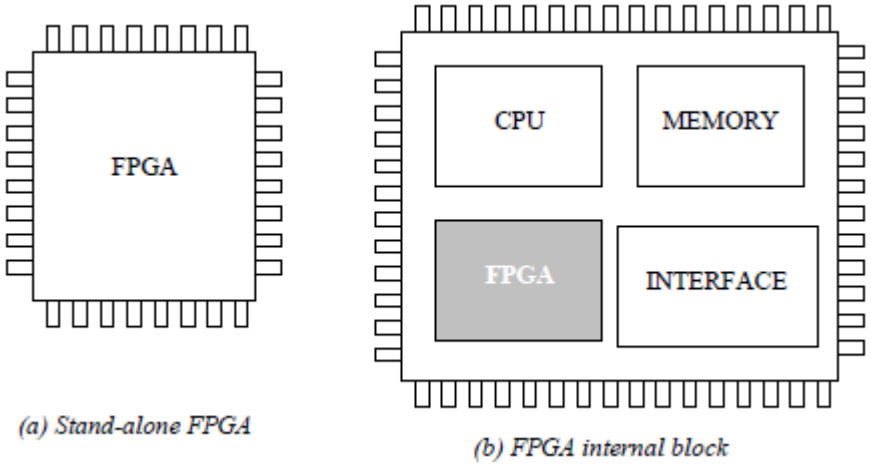


Fig.4. FPGA ca simple componente sau ca macroblocuri proiectele de sisteme pe un singur “chip”

- in cazul sistemelor de comunicatie, blocurile logice configurabile pot fi modificate dinamic in vederea adaptarii la protocoale de comunicatii mai performante;
- in situatia sistemelor de putere redusa, logica poate implementa mai multe task-uri in serie, in loc de a incorpora tot hardware-ul corespunzator, care nu va opera in paralel niciodata.

2. Circuite logice configurabile.

Blocurile logice programabile trebuie sa fie capabile sa implementeze toate functiile logice de baza: NOT, AND,…..,XOR, XNOR etc… Pentru implementarea acestora se pot folosi mai mule abordari, de exemplu:

- multiplexoare,
- tabele asociative (lookup tables).

2.1. Multiplexoare

Multiplexoarele cu doua intrari pot fi utilizate ca generatoare programabile de functii, dupa cum se poate vedea din tabela 1 si din figura 5.

Function	Boolean expression for output f	i0	i1	en
BUF(A)	f=A	0	1	A
NOT(A)	f=~(A)	1	0	A
AND(A,B)	f=A&B	0	B	A
OR(A,B)	f=A|B	B	1	A

Tab. 1. Programarea intrarilor unui multiplexor, in vederea implementarii a patru functii logice.

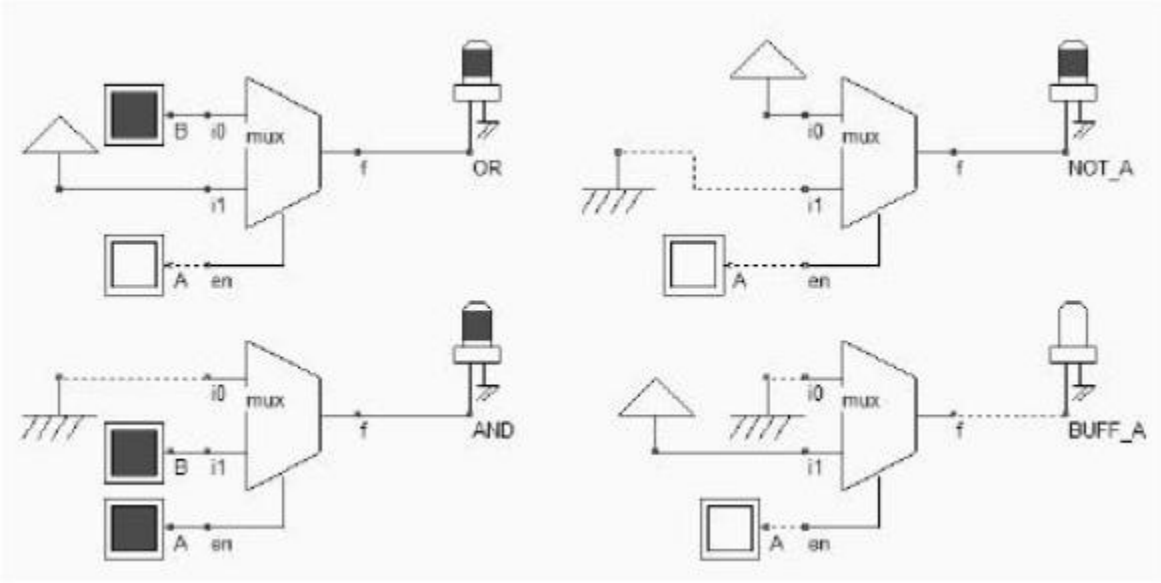


Fig.5. Implementarea unor functii logice cu ajutorul multiplexoarelor cu doua intrari de date si una de activare. (fpgaMux.SCH).

Alte functii (XOR) necesita cel putin doua multiplexoare (Fig. 6):

- cu o iesire de tip buffer, un multiplexor cu doua intrari are minim 6 tranzistoare si 3 etaje care introduc intarzieri: doua inversoare si un tranzistor de trecere.
- un MUX cu 4 intrari are 18 etaje, care introduc intarzieri, astfel implementarile cu multiplexoare nu sunt eficiente.

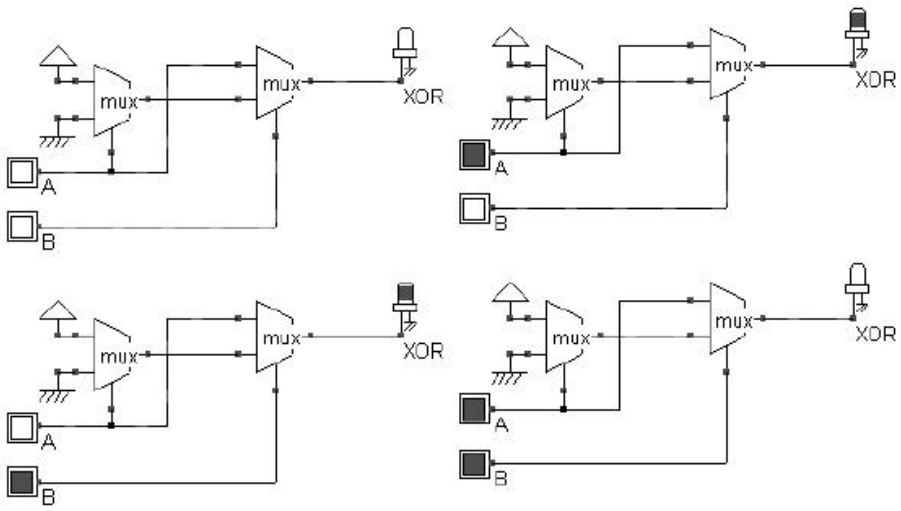


Fig.6. Functia XOR necesita cel putin doua multiplexoare, pentru implementare.

2.2. Tabele asociative (Look Up Tables).

Look Up Table (LUT) este cel mai versatil circuit pentru a crea o functie logica programabila/configurabila. LU.T descris in Tab.2. are ca intrari *F0,F1* si *F2*. Iesirea *Fout* genereaza o expresie logica bazata pe 8 informatii logice elementare *Value*[0], …, *Value*[7]. In cazul unui XOR, cu 3 intrari, setul de valori pentru iesirea Fout, dat in tabela de adevar, trebuie atribuit componentelor *Value*[0], …, *Value*[7]. In schema din figura 7 se atribuie manual Fout, din tabela de adevar, fiecaruia dintre cele 8 butoane. *Fout* genereaza functia XOR pentru intrarile *F0,F1* si *F2*.

Enable	F2	F1	F0	Fout= F0^F1^F2	Assigned to
0	0	0	0	0	Value[0]
1	0	0	1	1	Value[1]
2	0	1	0	1	Value[2]
3	0	1	1	0	Value[3]
4	1	0	0	1	Value[4]
5	1	0	1	0	Value[5]
6	1	1	0	0	Value[6]
7	1	1	1	1	Value[7]

Tab.2. Tabela de adevar a portii XOR, cu 3 intari, pentru implementarea in forma de tabela asociativa.

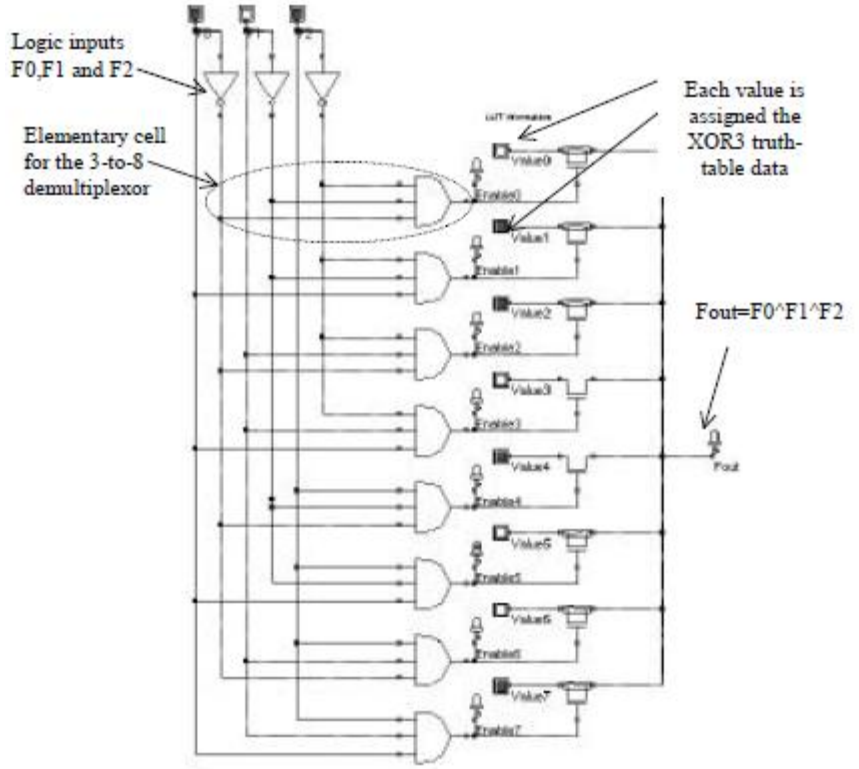


Fig. 7. Iesirea *f* genereaza functia logica *Fout*, in conformitate cu tabela asociativa stocata in punctele de memorare *Value*[i].

Punctele de memorare.

Punctele de memorare sunt utilizate pentru stocarea valorilor logice ale functiei date in tabela de adevar a acesteia. Punctele de memorare sunt componentele esentiale ale blocurilor logice configurabile. In ceea ce priveste posibilitatile de a stoca un singur bit de informatie, exista mai multe abordari. Astfel, in figura 8 se prezinta un registru de deplasare formatdin bistabile de tip D, in care bistabilui i memoreaza informatia logica *Value*[i]. Bistabilele D sunt inlantuite pentru a reduce numarul semnalelor de control la un semnal de ceas *ClockProg* si la un semnal de date *DataProg*.

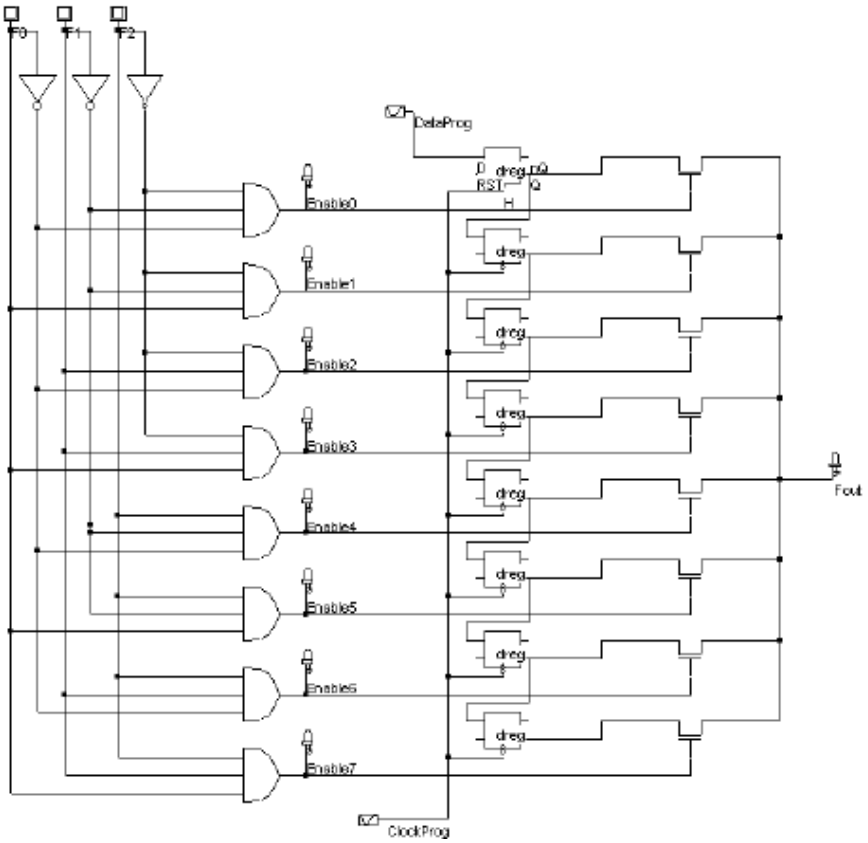


Fig.8. Informatia de tabela asociativa (look-up information – LUI) este data prin intermediul unui registru de deplasare cu bistabile de tip D (active pe fronturile negative- FpgaLutDreg.sch).

Configurarea unui LUT cu 3 intrari intr-o poarta XOR cu 3 intrari respecta strict protocolul descris in figura 9. Se genereaza o serie de 8 fronturi cazatoare de catre semnalul *ClockProg*. Aceasta se obtine prin configurarea unui generator de impulsuri cu o serie de 0-1, dupa cum se prezinta mai jos:

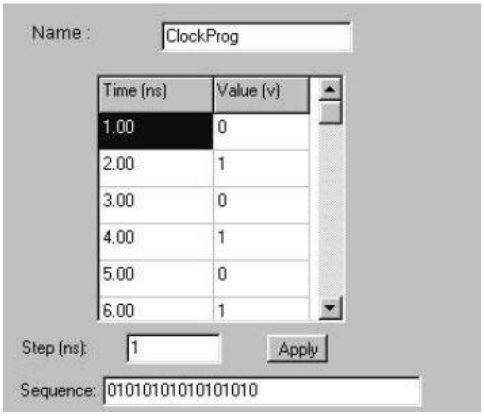


Fig.9 Programarea pentru ClockProg pentru a genera 8 fronturi active (FpgaLutDreg.sch)

La fiecare front activ , registrul de deplasare este alimentat cu o noua valoare prezentata secvential la intrarea *DataProg*. Mai intai trebuie sa fie intraodusa *Value*[7] si ultima *Value*[0]. Impulsul *DataProg* trebuie sa descrie tabela de adevar in ordine inversa, dupa cum se arata in figura 10.

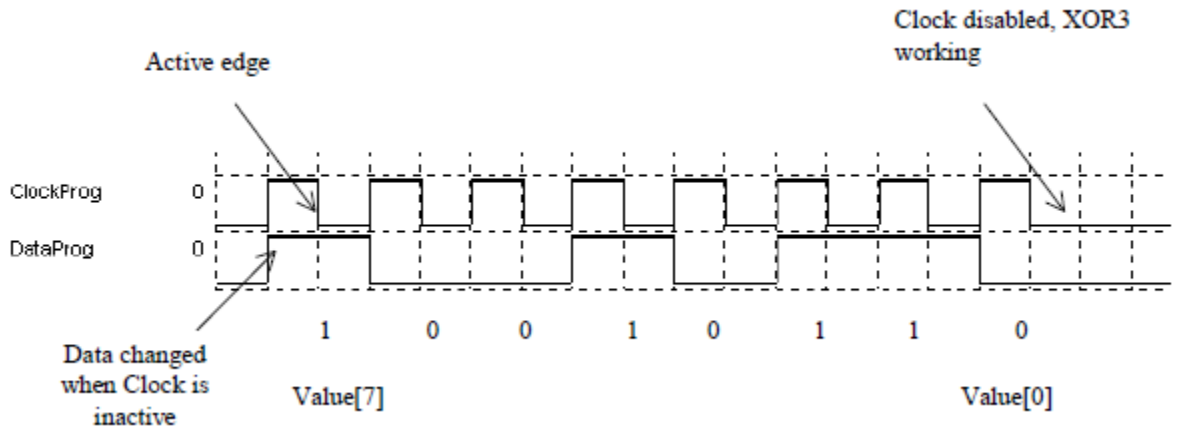


Fig.10. La sfarsitul celei de-a opta perioada de ceas, LUT este configurat ca un XOR, cu 3 intrari (FpgaLutDreg.sch).

Cele mai multe proiecte FPGA utilizeaza registre de tip D pentru a stoca configurarea LUT. Informatia de configurare se pierde la disparitia tensiunii de alimentare.

2.3. Fuzibile si Antifuzibile

Pentru a retine configurarea in absenta tensiunii de alimentare trebuie sa se utilizeze memorii nonvolatile. O astfel de memorie nonvolatila, programabila pentru o singura data, poarta denumirea de fuzibil. De regula, se utilizeaza ca fuzibil un contact intre doua straturi de metal, contact care va fi distrus de la aplicarea unui curent de intensitate mare (Fig11).Desi aceasta metoda provoaca distrugeri si in vecinatatea contactului, nu este nevoie de un strat specific, din punct de vedere tehnologic, ca in tehnologia CMOS λ

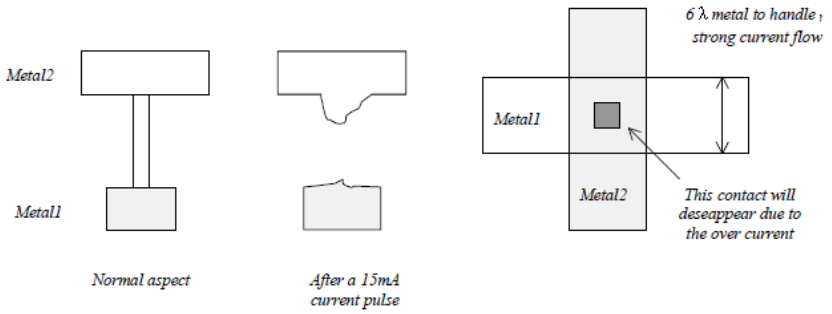


Fig.11. Contacte bazate pe fuzibile.

Un circuit de comanda cu o latime mare a canalului (cativa μm), alimentat de la o sursa de tensiune mare (V_{DDH}) va asigura un curent apreciabil prin driver. Schema circuitului fuzibil este prezentata in figura 12.

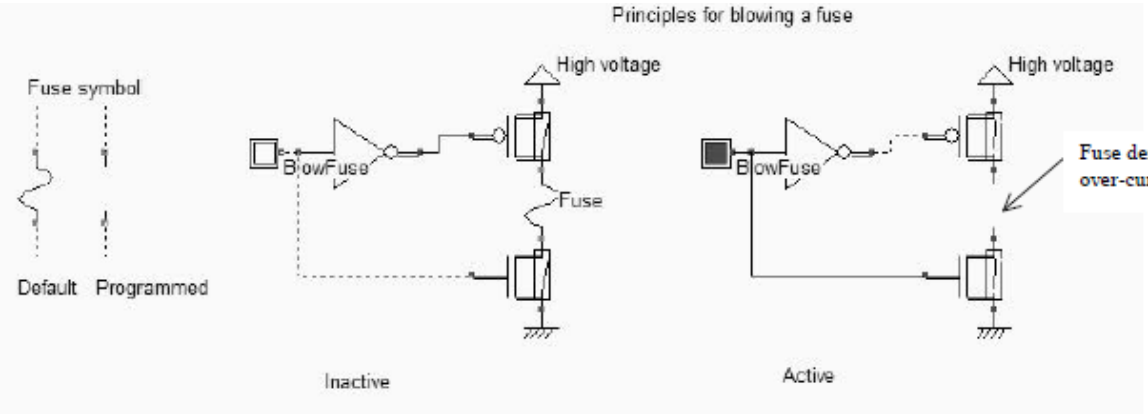


Fig.12. Programarea fuzibilelor.

Atunci cand se activeaza comanda BlowFuse sunt activate si ambele tranzistoare NMOS si PMOS, ceea ce conduce la trecerea unui curent apreciabil prin fuzibil, care topeste fuzibilul si intrerupe calea de curent. Curentul trebuie sa depaseasca valoarea de 15mA.

In contrast cu fuzibilul antifuzibilul are starea normala “deschis” (figura.13). O tensiune mare (cca 10V) aplicata intre straturile metal1 si metal2 distruge oxidul, asigurand un traseu conductiv intre straturile de metal. Utilizarea unei tensiuni ridicate in circuit implica o utilizare atenta a tehnologiei MOS si a diver-elor de I/E specifice pentru astfel de valori ale tensiunii manipulate.

O alta structura intalnita , care se chiama ONO (Oxid-Nitrura-Oxid), asigura o cale rezistiva atunci cand este programata. Valoarea tipica a rezistentei este de 500 Ohm. Statistic, valoarea

rezistentei este in limite mult mai mari pentru S_iO_2 decat pentru ONO, ceea ce face fuzibilul ONO mult mai atractiv, in ciuda unor etape suplimentare in procesul de fabricatie.

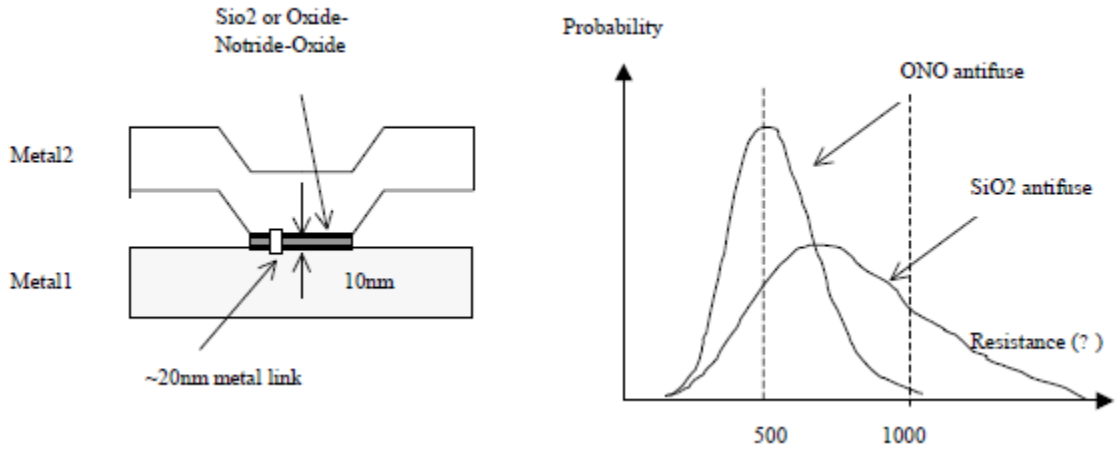


Fig.13. Principiul antifuzibilului si distributiile comparative ale rezistentelor pentru ONO si S_iO_2

Pentru programarea circuitelor FPGA, EEPROM si FRAM se folosesc alte tipuri de memorii nonvolatile. Aceste memorii nu sunt alterate atunci cand nu sunt alimentate si pot fi reprogramate de un numar mare de ori.

2.4. Implementare in DSCH.

In DSCH in meniul de simboluri (Fig.14) s-a introdus simbolul tebelei asociative LUT, care este echivalenta cu schema din figura 8. O proprietate importanta a simbolului LUT consta in abilitatea de a stoca programarea interna conform cu ceea ce realizeaza o memorie nonvolatila. Simbolul pentru pentru LUT, in interfata cu utilizatorul, este dat in figula 14.

Pentru completarea tabelei asociative se pot utiliza trei metode: una consta in definirea fiecarui element al tabloului cu cifrele 0 si 1. Numerele corespund combinatiilor logice ale intrarilor F2F1F0. De exemplu cifra 4 in zecimal va fi codificata 100 in binar, ceea ce corespunde la F2=1, F1=0 si F0=0. O alta solutie consta in selectarea descrierii functiei din lista. Informatia logica Fout este atribuita fiecarei combinatii de intrari ale LUT. O a treia solutie rezida in aceea ca utilizatorul introduca o descriere bazata pe intrarile F0,F1 si F2, cat si pe operatorii: "~" (NOT), "&' (And), "|" (Or) si "^" sa (XOR**),** urmand ca sa activeze butonul “Fill LUT”, pentru transferarea rezultatului expresiei catre tabela.

1. **Blocul Logic Programabil.**

Blocul Logic Programabil (BLP) consta in doua tabele asociative , un registru D si cateva multiplexoare. Pentru realizarea BLP exista numeroase solutii.

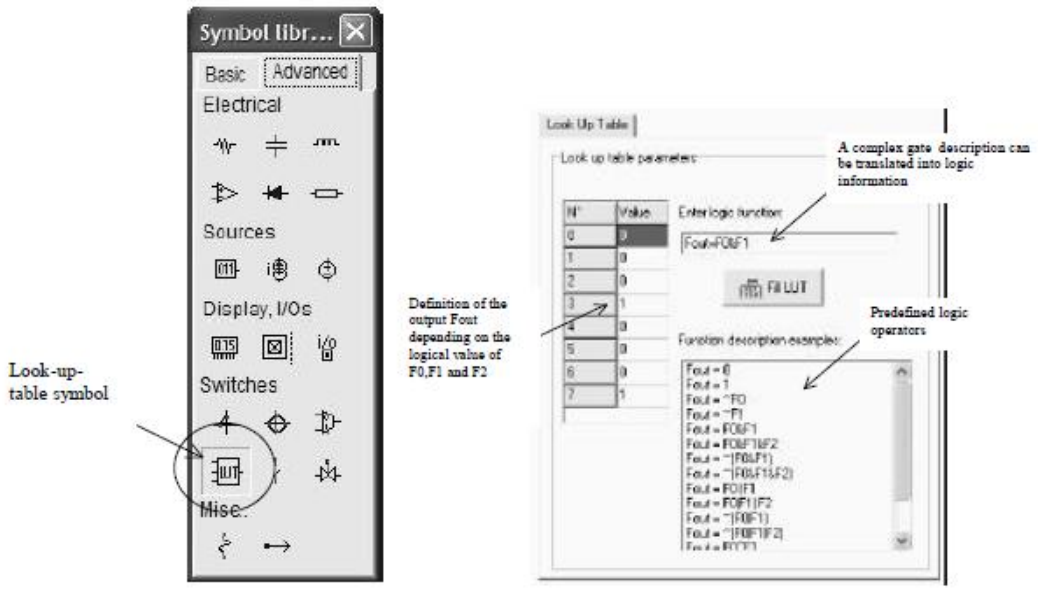


Fig.14. Simbolul LUT.

In figura 15 se prezinta o structura simpla care are sismilaritati cu cea utilizata in circuitele XC5200. BLP contine doua structuri active : LUT si registrul D, care pot functiona indepenedent sau impreuna.

Iesirea LUT este conectata direct la iesirea blocului Fout. Iesirea poate fi aplicata si la intrarea registrului D, datorita multiplexorului contraolat de DataIn_Fout. Reteaua DataOut poate transfera direct semnalul DataIn, ceea ce face ca celula sa fie transparenta. Semnalaul DataOut poate, de asemenea, furniza semnalul nQ, in functie de starea multiplexorului controlat de DataIn_nQ.

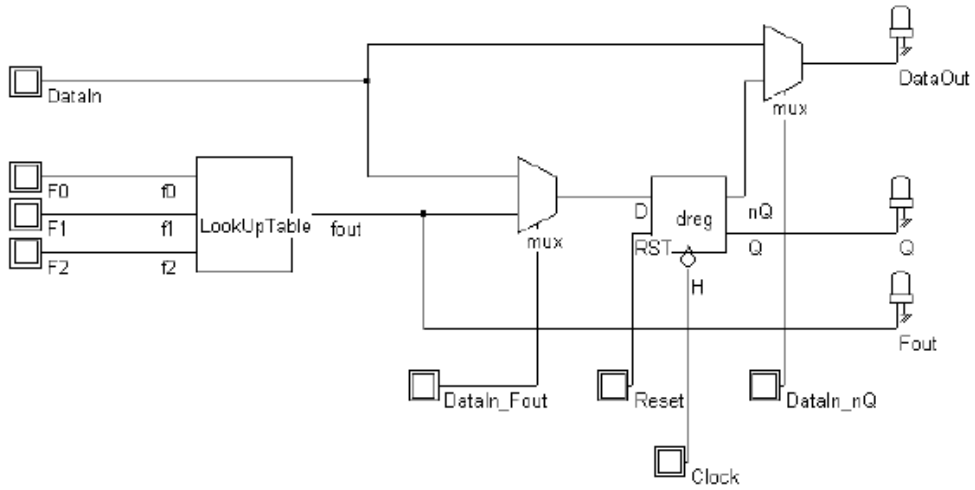


Fig. 15. BLP simplu, care include un LUT si un registru D. (FpgaCell.SCH).

Blocul consta acum intr-un LUT si un bistabil D. In continuare se vor inlantui semnalele DataIn_Fout si DataIn_nQ in structura unui registru de deplasare format din doua bistabile suplimentare. Fiecare bistabil utilizeaza acelasi semnal de ceas si aceeasi date de intrare inlantuite. Circuitul complet se prezinta in figura 16.

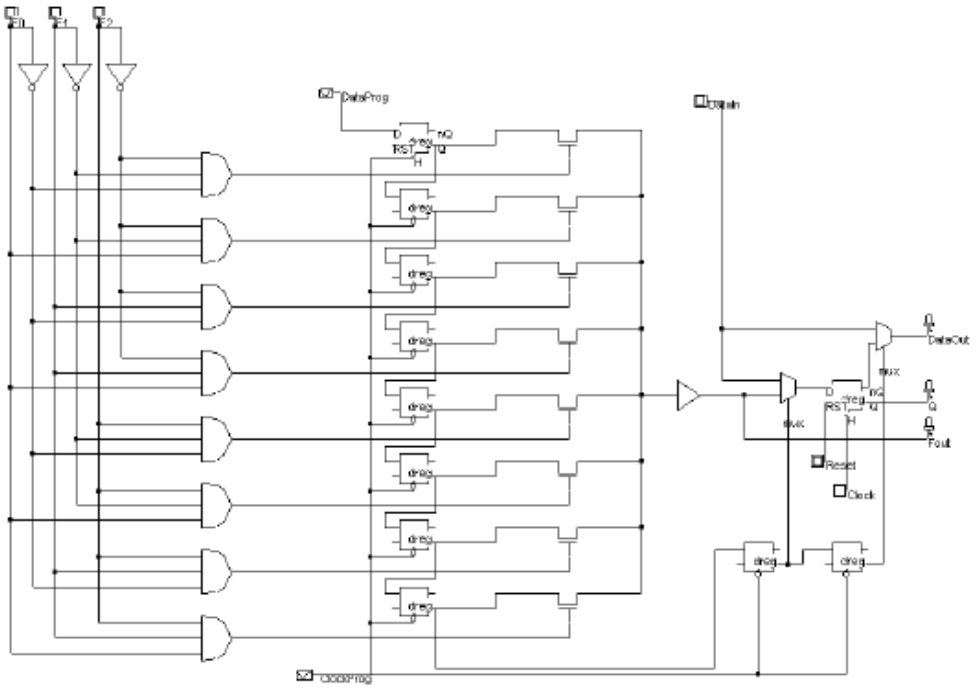


Fig.16. LUT, bistabilul D si registrul de deplasare, inclusiv doua circuite multiplexoare. (FpgaBlockStructure.SCH)

Configurarea blocului se efectueaza pe baza a 10 fronturi active, ale semnalului ColckProg, si a 10 semnale de date pe intrarea DataProg. Informatia care se transfera catre capatul cel mai indepartat al registrului este definita in primul ciclu, in timp ce bistabilul cel mai apropiat al registrului este configurat cu data transferata in ultimul ciclu (Tab.3)

Cycle 1	2	3	4	5	6	7	8	9	10
DataIn Nq	Datain Fout	Val[7]	Val[6]	Val[5]	Val[4]	Val[3]	Val[2]	Val[1]	Val[0]

Tab.3. Informatia seriala (data)utilizata pentru programarea punctelor de memorie ale LUT-ului.

2. Interconectarea blocurilor.

In acest paragraf este prezentata strategia de interconectare a blocurilor logice. Mai intai se vor examina punctele de interconectare programabile si matricile de conectare programabile. In continuare se va discuta implementarea globala a structurii.

2.1. Punctele programabile de interconectare (PPI).

PPI (PIP Programmable Interconnect Point) se regaseste in meniul "Advanced Switches Symbols), din figura 17. PPI au doua stari: "On" si "Off", care pot fi impuse prin dubla activare pe simbol si un click pe butonul "On/off".

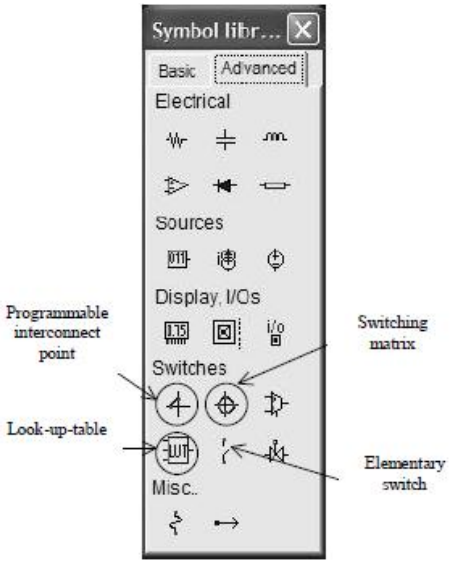


Fig.17. PPI (PIP) in paleta de simboluri.

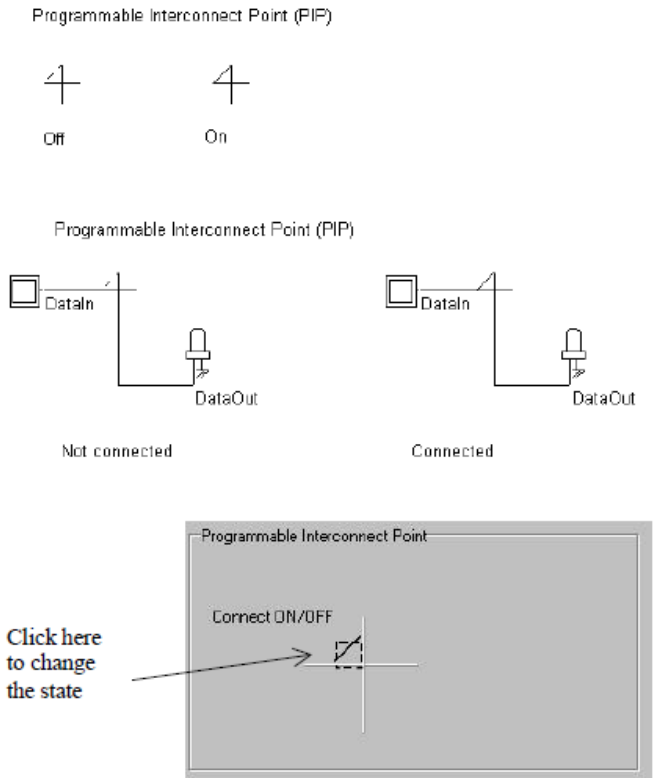
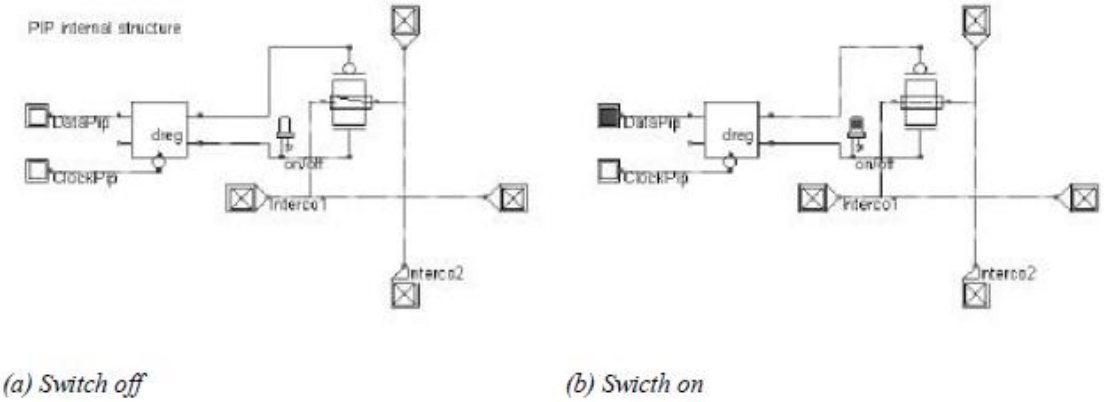


Fig.18. Modificarea starii PIP (FpgaPip.SCH).

Puntea poate fi realizata cu o poarta de transmisie (PT), controlata cu un bistabil D (Fig.19). Atunci cand informatia de programare este "0" PT este deschisa, legatura absenta intre Interco1 si Interco2. PT asigura o legatura rezistiva (≈ 100 Ω) intre Interco1 si Interco2.



(a) Switch off *(b) Swicth on*

Fig. 19. Structura interna a PIP si ilustrarea comportarii sale On/Off (FpgaPip.SCH).

Pentru a asigura cea mai mare flexibilitate de rutare PPI se vor grupa sub forama de matrice. In figura 20 se prezinta exemple de matrici 3 x 3 si 3 x 2. Legatura intre In1 si Out1, In2 si Out2, In3 si Out3 se realizeaza prin programarea unor PIP-puri. O unealta specifica de rutare asigura acest task, dar se poate face si manual. In DSCH pentru a schimba starea (0→1→0) unui PIP se tasteaza "O" pe simbolu grafic.

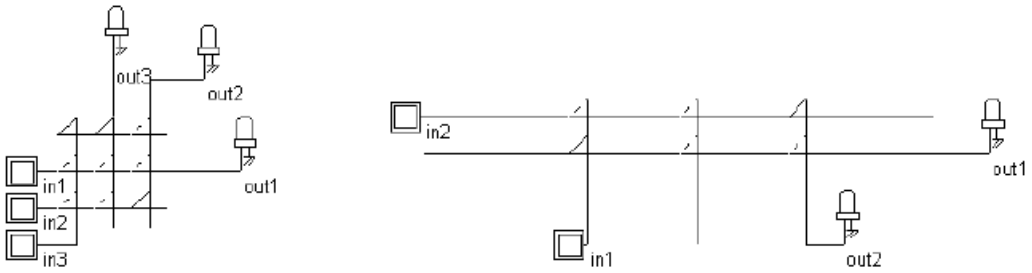


Fig.20. Matrice de PIP-uri (FpgaPip.SCH).

Matrice de comutare.

Matricea de comutarea reprezinta un PPI complex, care implica o gama larga de combinatii in cadrul unei intersectii de interconexiuni. Figura 21 reprezinta o matrice de comutare . Matricea contine 6 punti configurabile intre doua interconexiuni principale. Simbolul matricii de comutare se regaseste in meniul asigurat de aplicatia DSCH la sectiunea: "Advanced" set of "Switches" symbols. Printr-un click dublu pe simbolul matricii se pot accesa cele 6 comutatoare "On/Off"

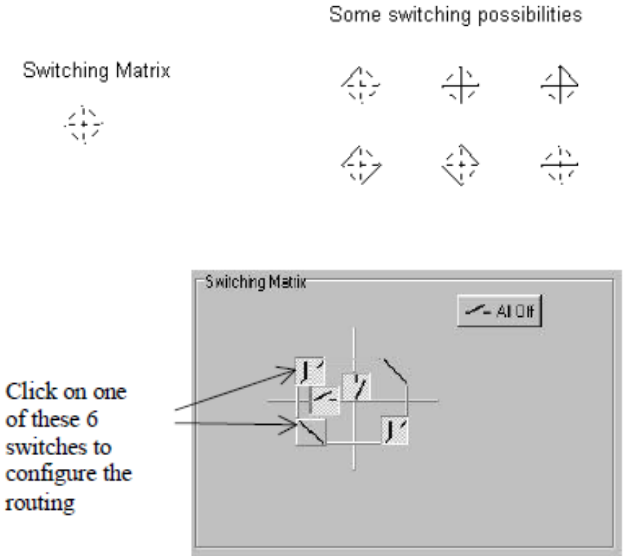


Fig.21. Modificarea starii matricii (FpgaMatrix.SCH).

Pentru a usura programarea matricei, in DSCH exista comenzi simple. Astfel starea matricii poate fi modificata plasand cursorul pe simbol si activand urmatoarele taste:

- Dezactivarea matricii:”o”;
- Activarea matricii: “O”;
- Activarea legaturii orizontale: “-“
- Activarea legaturii verticale:”|”

In figura 22 se dau exemple de matrici de comutare 3x2 si 3x3. Exemplele de rutare sunt numeroase. Ele imbunatatesc configurabilitatea blocurilor logice

.

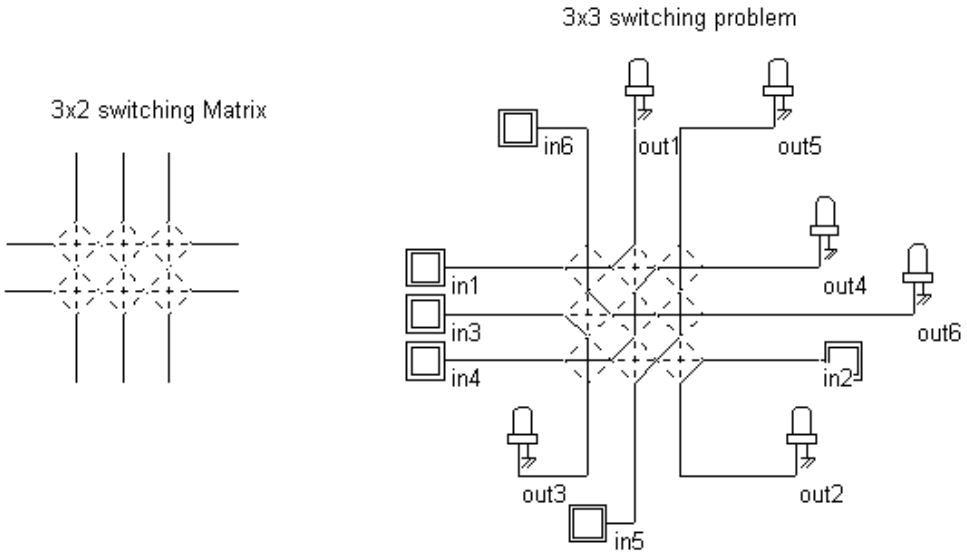


Fig.22 matrice de comutare 3x2 si un exemplu de strategie de rutare intre 6 intrari si 6 iesiri (fpgaMatrix.SCH).

Implementarea Matricilor de Comutare.

Din punct de vedere practic, matricea de comutare poate fi construita prin regruparea a 6 porti de transmisie (Fig.23). Fiecare poarta de transmisie este controlata de un bistabil D asociat care memoreaza configuratia dorita. Bistabilele D sunt inlantuite pentru utilizarea unei singure intrari DataIn si a unui semnal de ceas LoadClock, care sunt suficiente pentru configurarea matricii.

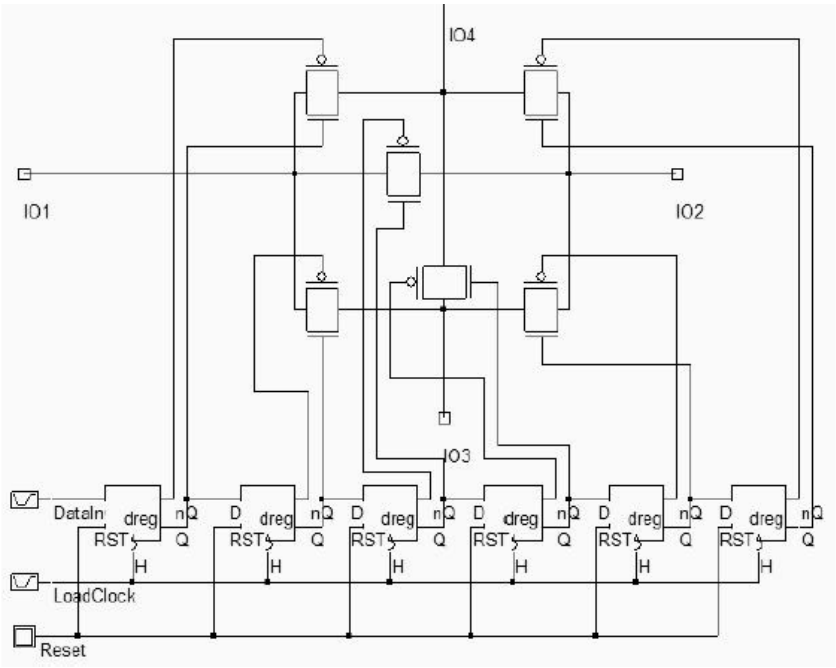


Fig. 23. Porti de transmisie amplasate pe liniile de rutare in vederea construirii matricii. (FpgaMatrix3.SCH).

Tablouri de Blocuri.

Blocurile configurabile sunt asociate cu PIP-uri si cu matrici de comutare pentru a crea un nucleu complet configurabil. In figura 24 se propune un exemplu de bloc dublu configurabil si rutarea lui configurabila asociata.

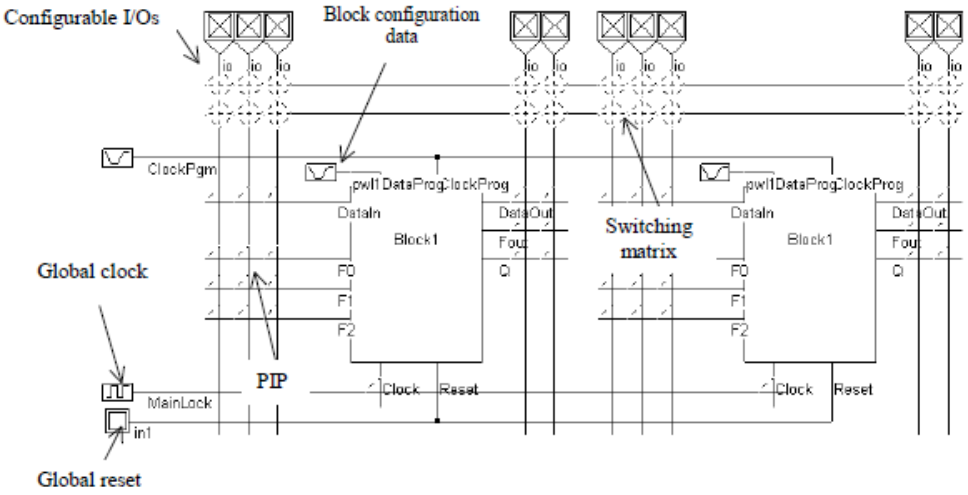


Fig. 24. Blocuri configurabile, matrice de comutare, I/E configurabile si tablouri de PIP-uri. (fpga2blocks.SCH).

Exemplu de Sumator Complet.

Tabela de adevar si expresia logica pentru Sumatorul Complet sunt reamintite in Tab.3.

Implementarea sumei (Sum) si a transportului (Carry) se realizeaza prin programarea a doua tabele asociative corespunzator tabelelor de adevar.

Sumator complet

A	B	C	SUM	CARRY	RESULT
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	2
1	0	0	1	0	1
1	0	1	0	1	2
1	1	0	0	1	2
1	1	1	1	1	3

Tab.3. Tabela de adevar a sumatorului complet.

Diagrama generala de implementare a unui sumator complet este data in figura 25. Unul din blocurile logice programabile genereaza functia Sum, pentru valorile date ale intrarilor A,B si C. Informatia necesara configurarii Blocului 1, ca functie suma (XOR cu 3 intrari) este data in Tab.4. Semnalul Sum se propaga in afara blocului, in regiunea interfetei de iesire, prin exploatarea resurselor de interconectare si a matricii de comutare. Celalalt bloc logic programabil, Blocul 2, asigura generarea semnalului Carry, ca functie de aceleasi intrari A,B si C. Programarea Blocului 2 rezulta din Tab.4. Rezultatul carry este exportat in zona de iesire, in modul in care s-a procedat si pentru semnalul Sum.

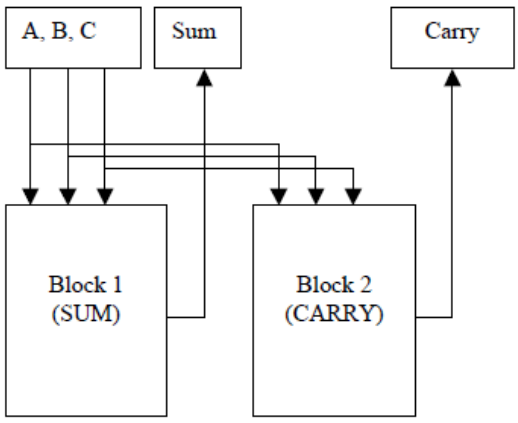


Fig. 25. Functiile SUM si CARRY care realizeaza un sumator complet in FPGA. (fpgaFullAdder.SCH).

Block 1 (Sum of F0,F1 and F2)									
Cycle 1	2	3	4	5	6	7	8	9	10
DataIn Ncy	Datain Fout	Val[7]	Val[6]	Val[5]	Val[4]	Val[3]	Val[2]	Val[1]	Val[0]
0	0	1	0	0	1	0	1	1	0

Block 2 (Carry of F0,F1 and F2)									
Cycle 1	2	3	4	5	6	7	8	9	10
DataIn Ncy	Datain Fout	Val[7]	Val[6]	Val[5]	Val[4]	Val[3]	Val[2]	Val[1]	Val[0]
0	0	1	1	1	0	1	1	0	0

Tab. 9. Datele sub forma seriala utilizate pentru a configura blocurile logice 1 si 2, ca SUM si CARRY.

Simularea Sumatorului Complet implementat in doua blocuri configurabile(fpgaFullAdder.SCH) este prezentata in figura 26.

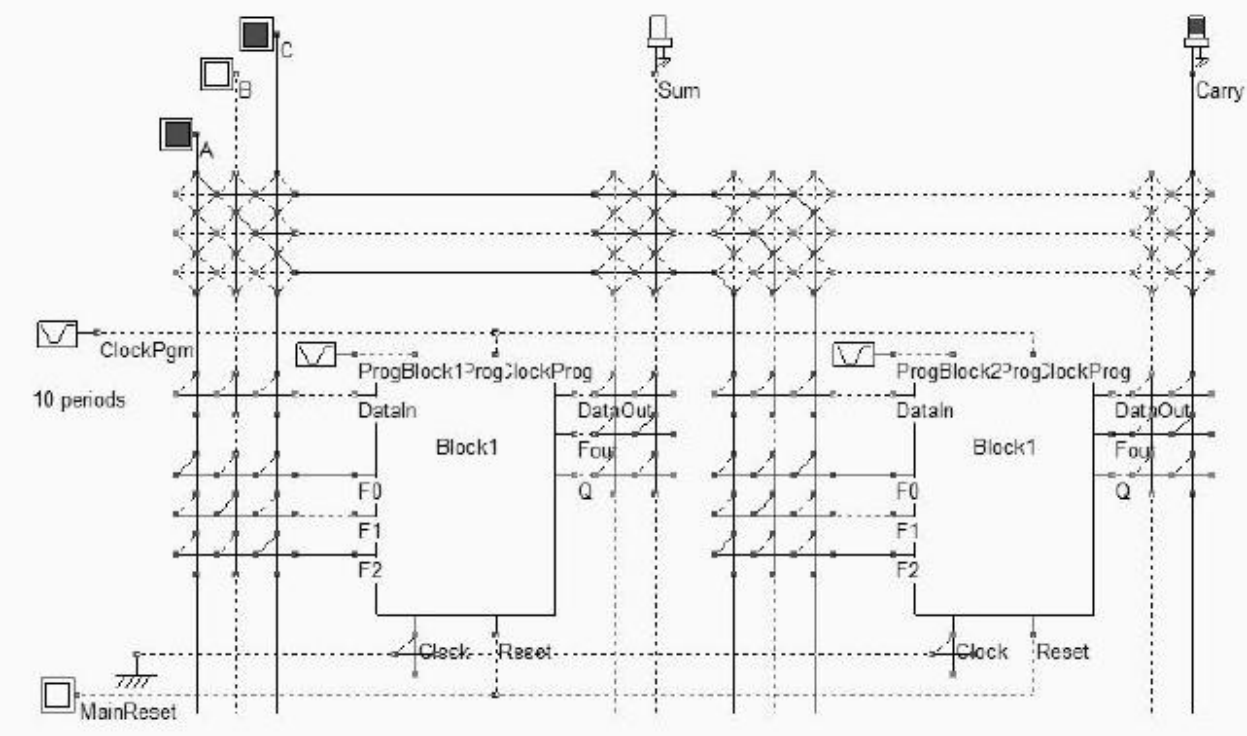


Fig. 26. Simularea Sumatorului Complet implementat in doua blocuri configurabile (fpgaFullAdder.SCH).

Secventa de programare este continuta sub forma unor transe liniare ProgBlock1 si ProgBlock2. Dupa cum se vede din diagrama temporala prezentata in figura 27, ceasul de programare ClockPgm este activ numai in faza de initializare, pentru a deplasa informatia de programare in punctele de memorare din blocuri, pentru configurarea fiecarui multiplexor. Rutarea semnalelor A,B si C ca si a semnalelor Sum si Carry a fost efectuata manual, in circuitul din figura 26. In realitate, exista unelte pentru plasare/rutare, care genereaza automat structura electrica pornind de la schema initiala. Astfel, se evita erorile manuale, limitandu-se conflictele si omisiunile.

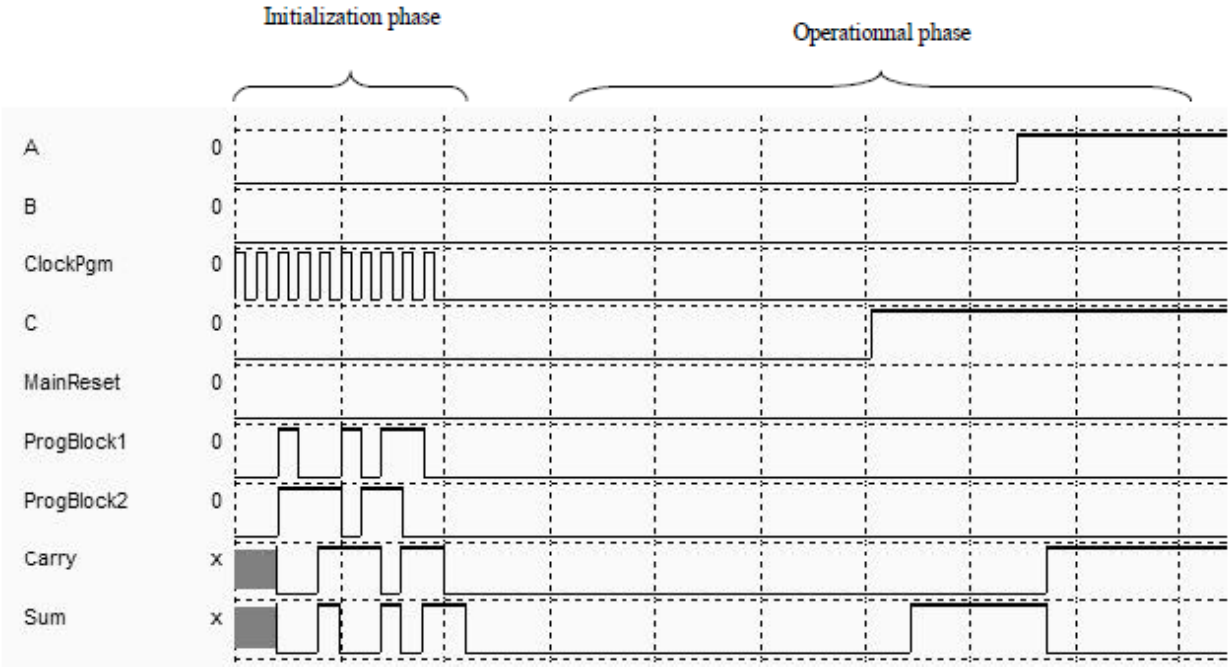


Fig.27. Diagrama temporala a Sumatorului Complet.(fpgaFullAdder.SCH)

Exemplu Divizor de ceas.

Cel de-al doilea exemplu propus pentru implementare in FPGA este cel al unui divizor de ceas. In figura 28 sunt date structura generala si diagrama temporala ale divizorului de ceas cu 4. Acesta necesita doua registre D, cu reactie de la nD la intrarea D.

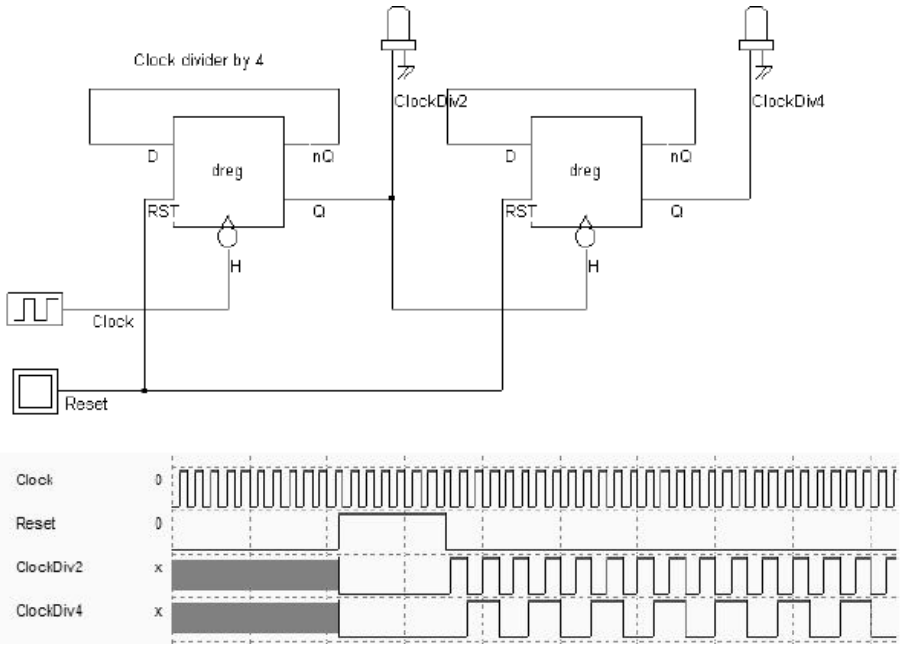


Fig.28. Diagrama si simularea tipica pentru un divizor de ceas cu4. (ClockDiv4.SCH)

Schema bloc generala a implementarii unui divizor de ces este data in figura 29.

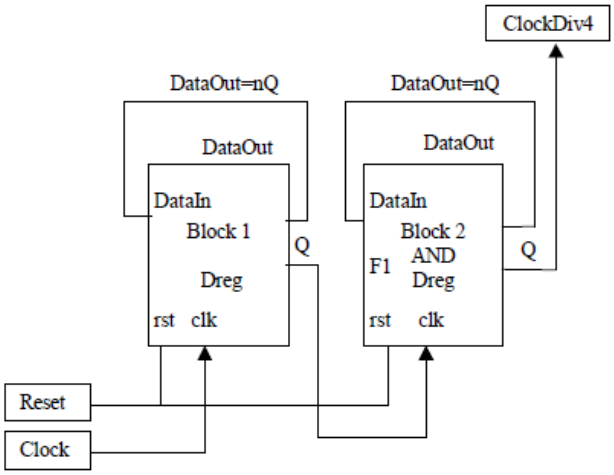


Fig.29: Implementarea unui divizor de ceas cu 2 in 2 blocuri configurabile (FpgaDiv4.SCH).

Fiecare bloc logic programabil este configurat ca un etaj de divizare a ceasului. Informatia necesara pentru configurarea Blocului1, ca un simplu bistabil D, este prezentata in Tab.9. Aceasta informatie/data sub forma seriala creaza o cale directa de la DataIn catre intrarea bistabilului D, al celulei Dreg, iar nD se propaga conform figurii 30. In afara blocului programabil, semnalul nQ se propaga catre intrarea DataIn. De observat ca tabela asociativa este

inactiva in aceasta configuratie. Celalalt bloc logic Block2 este, de asemenea, programat ca un bistabil Dreg cu reactie de la nQ la DataIn.

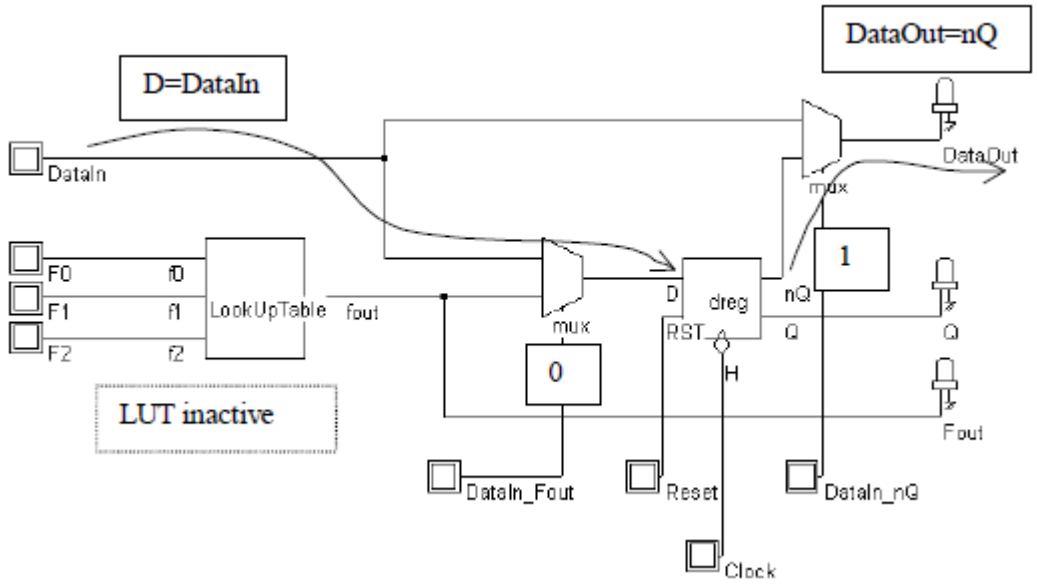


Fig.30. Utilizarea blocului configutabil ca bistabil Dreg.SCH)

Block 1 (DataOut=nQ, D=DataIn)									
Cycle 1	2	3	4	5	6	7	8	9	10

DataIn Nq	Datain Fout	Val[7]	Val[6]	Val[5]	Val[4]	Val[3]	Val[2]	Val[1]	Val[0]
1	0	0	0	0	0	0	0	0	0

Block 2 (DataOut=nQ, D=DataIn)									
Cycle 1	2	3	4	5	6	7	8	9	10
DataIn Nq	Datain Fout	Val[7]	Val[6]	Val[5]	Val[4]	Val[3]	Val[2]	Val[1]	Val[0]
1	0	0	0	0	0	0	0	0	0

Tab. 9 Data seriala utilizata pentru a configura blocurile logice 1 si 2 ca divizoare de ceas. (FpgaDiv4.SCH)

In figura 32 se prezinta rezultatele simularii contorului. Primele nanosecunde sunt dedicate programarii blocului. Dupa programare contorul opereaza in conformitate cu specificatiile din figura 28. De remarcat intarzierea importanta a raspunsului la fronturile active, datorate complexitatii intrinseci a blocului configurat si a intarzierii in calea de interconectare prin punctele de conectare si matricea de comutare.

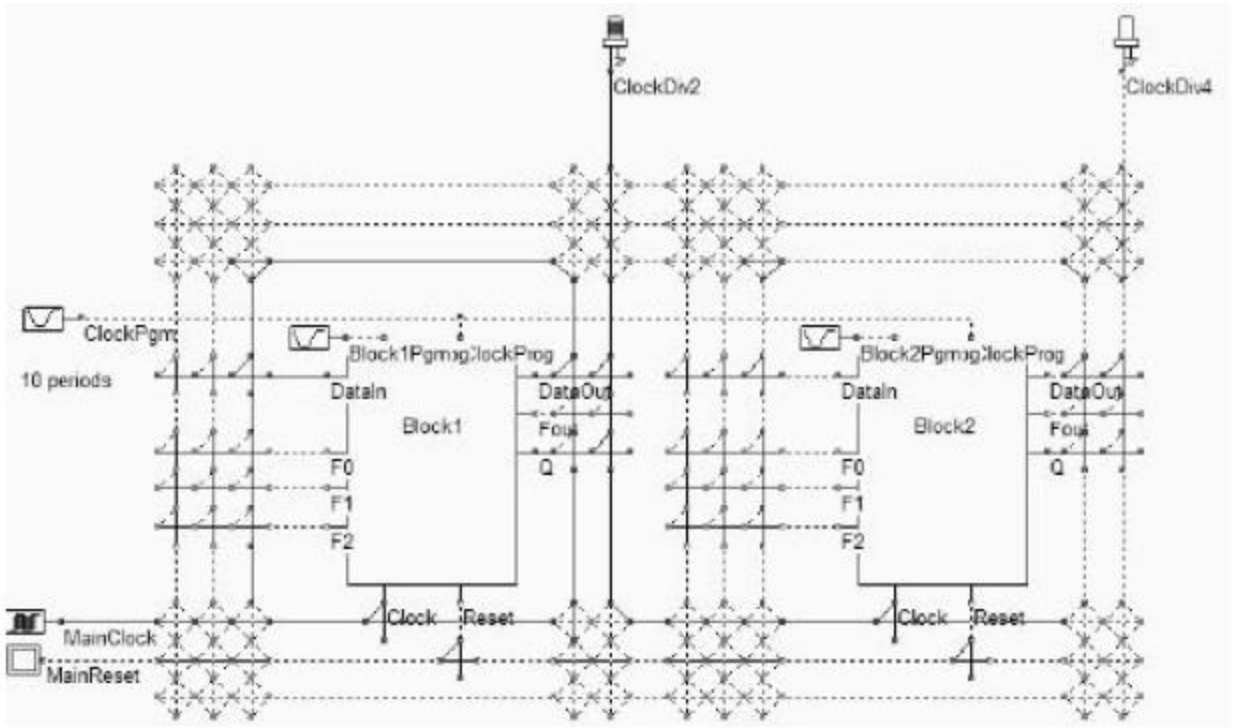


Fig.31. Rutarea divizorului de ceas cu 2 in doua blocuri configurabile. (FpgaDiv4.SCH)

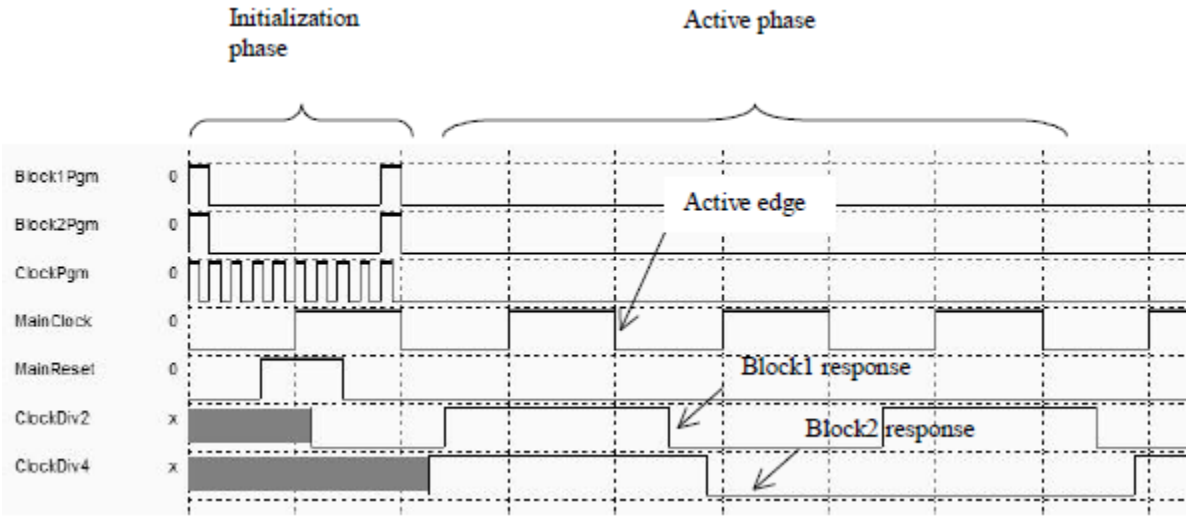


Fig.32. Diagramele temporale ale circuitului divizor de ceas. (ClockDiv4.SCH).

5 Concluzii.

Acest curs constituie o introducere in domeniul FPGA, din punctul de vedere al proiectarii celulei de baza:

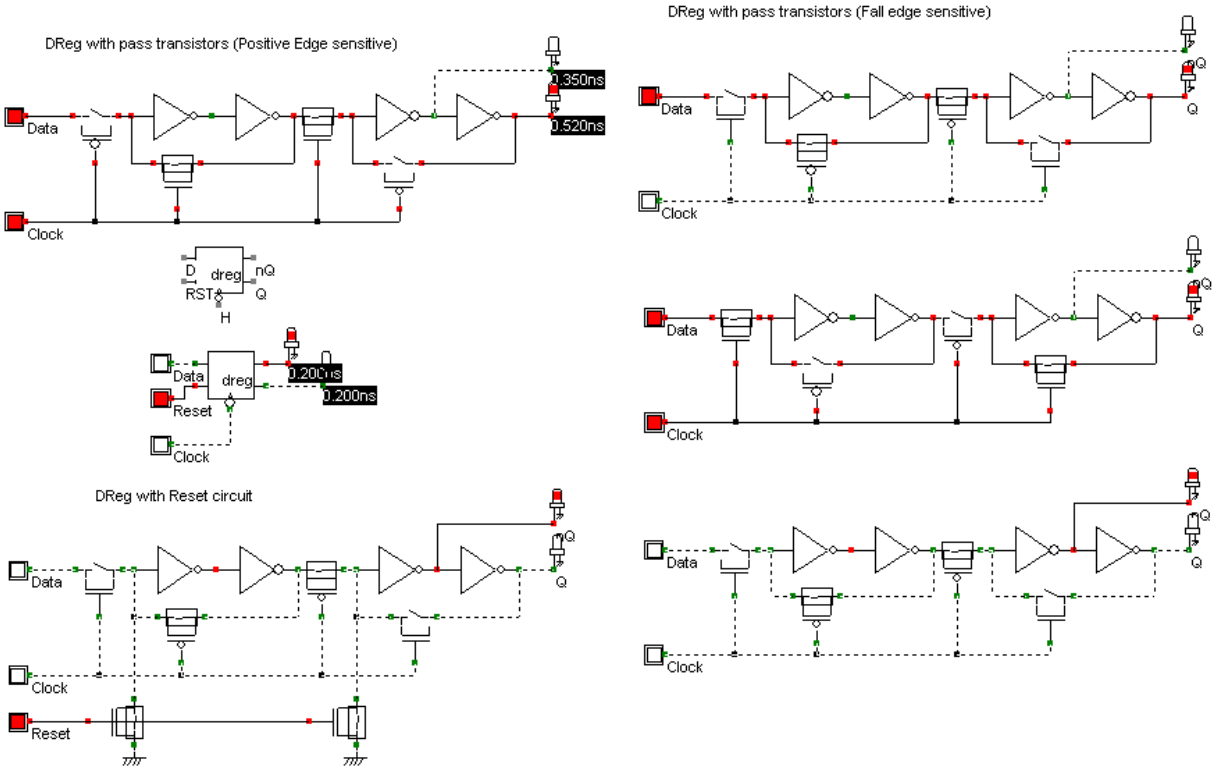
- s-au prezentat multiplexorul si tabela asociativa pentru constructia circuitelor logice configurabile;
- au fost examinate programarea punctelor de memorie folosind bistabile Dreg inlantuite, cat si fuzibilele;
- s-au descris punctele de interconectare programabile, matricile de comutare, cat si implementarile lor in DSCH;
- s-au realizat: implementarea unui sumator complet si a unui divizor de ceas utilizand doua blocuri logice configurabile, puncte de interconectare programabile si o matrice de comutare.

.Bibliografie.

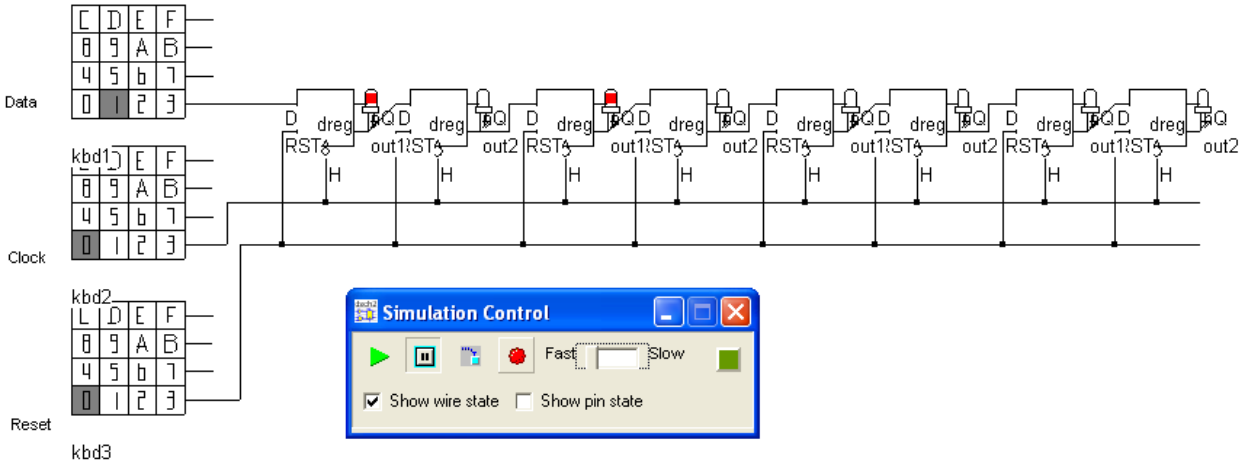
[1]. E.Sicard, S. Delmas-Bendhia. DEEP SUBMICRON CMOS DESIGN.
Cap 9. Field Programmable Gate Array.

ANEXE.

A1. Diverse implementari pentru Dreg (dreg.sch).



A2. Registru de deplasare pe 8 biti realizat cu bistabile Dreg, active pe frontul negativ al ceasului (Dreg_de_deplasare_8_1.sch):



A3. LUT cu 3 intrari (LutAND.sch)**:**