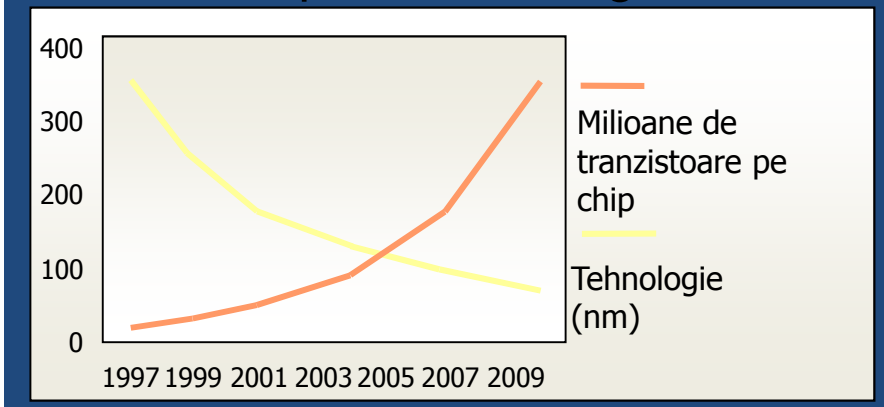
Sisteme Incorporate

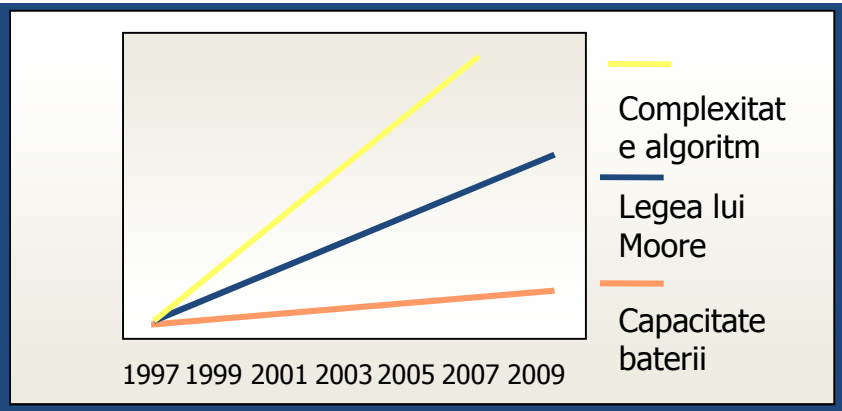
Cursul 11

Sisteme Reconfigurabile

De ce avem nevoie de reconfigurabilitate?

- Densitatea foarte mare a tranzistoarelor in circuitele moderne
- Costuri sporite de integrare





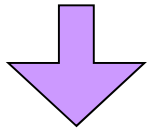
- Complexitate crescuta a algoritmilor
- Limitari puternice a capabilitatilor bateriilor

Se doreste
si performanta
si flexibilitate

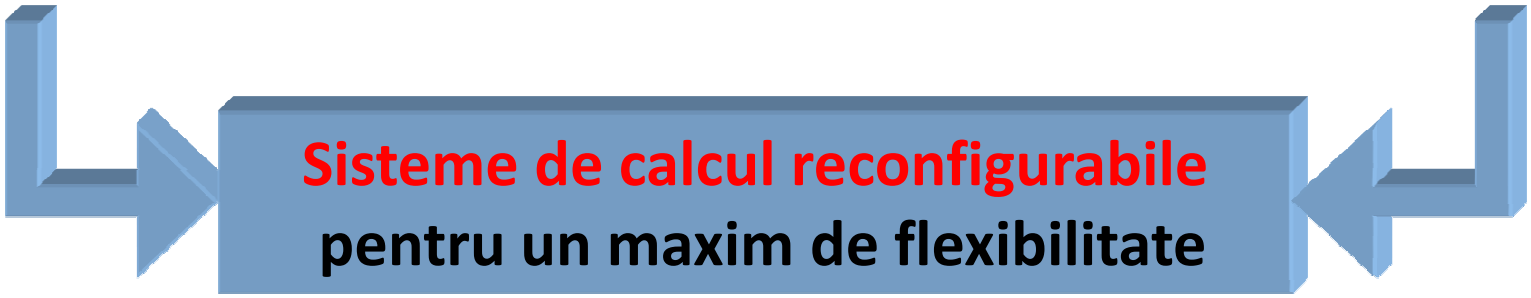
Constrangeri severe pentru consumul de energie

Analiza algoritmilor pentru sisteme embedded

- **90% din complexitatea unui algoritm este concentrata in portiuni bine definite ce constituie o parte foarte mica din codul total**
- **Multi algoritmi au portiuni de cod care pot fi paralelizate**
  - ✓ Performanta este imbunatatita prin folosirea cailor paralele de date
- **Granularitatea operanzilor este de obicei destul de diferita de 32 de biti**
  - ✓ Un UAL traditional este ineficient (consuma prea multa energie)

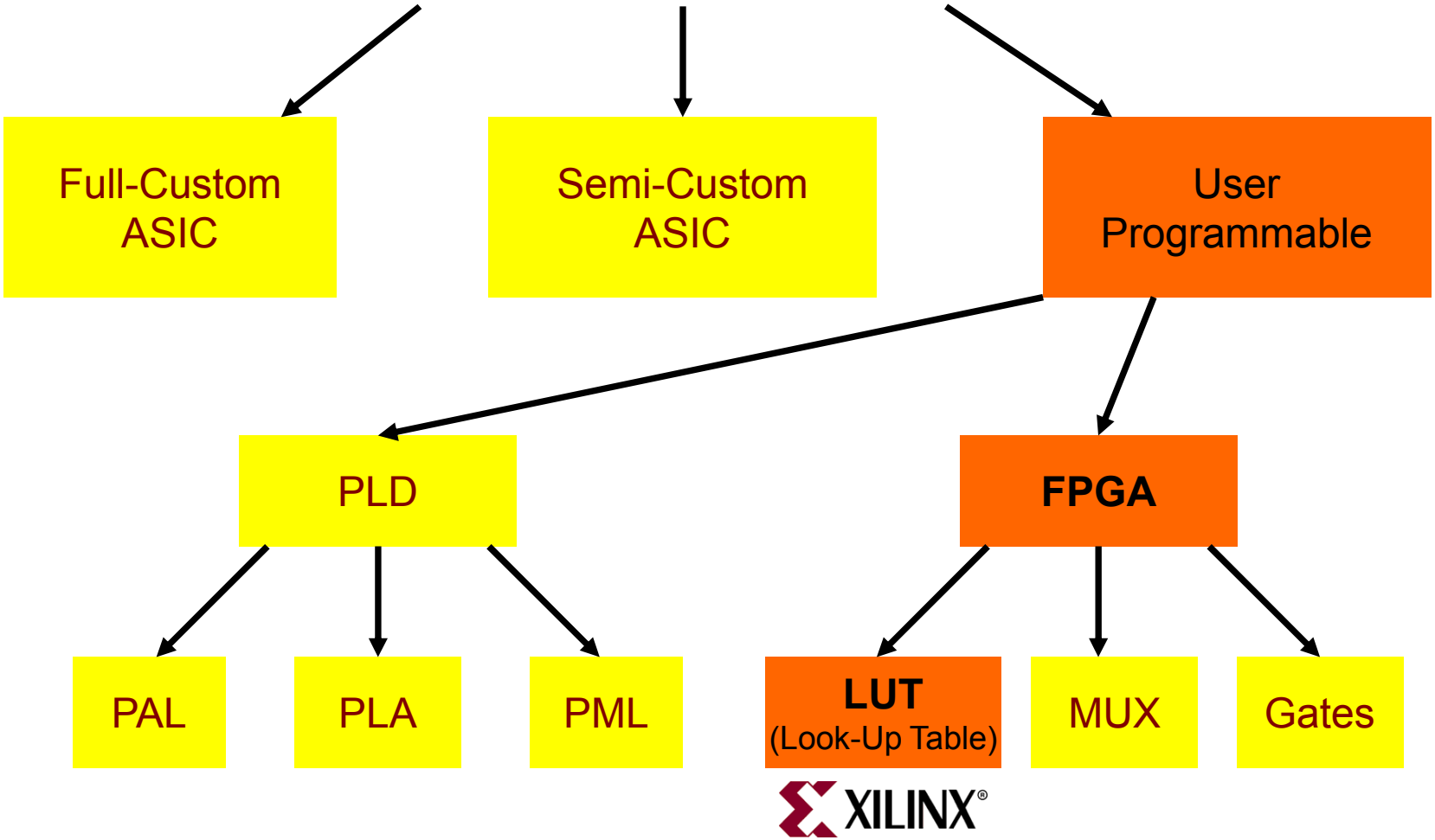


Se pot obtine imbunatatiri semnificative daca functionalitatea procesoarelor embedded este extinsa cu functii specifice aplicatiei



Tipuri de circuite integrate

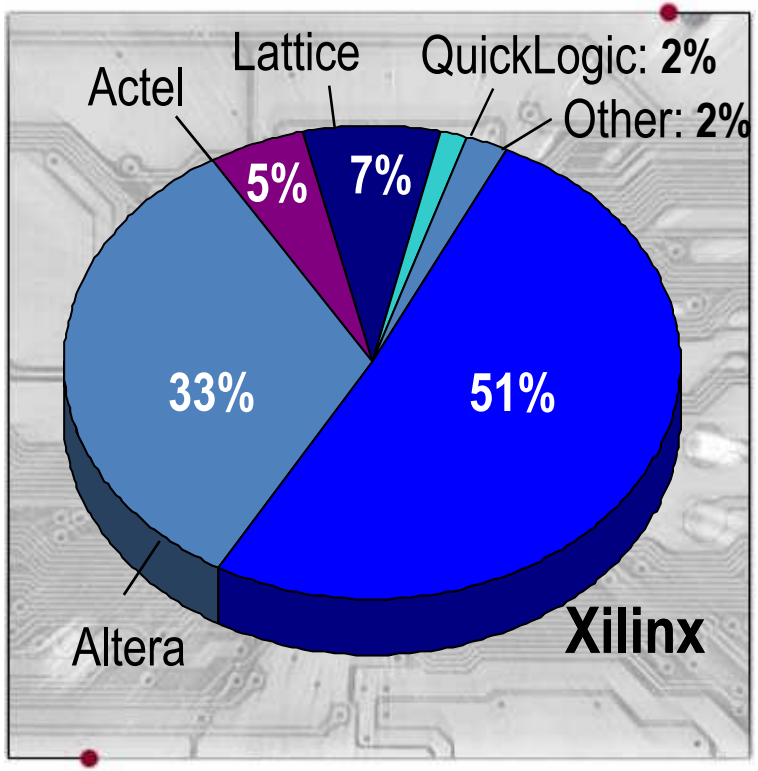
Integrated Circuits



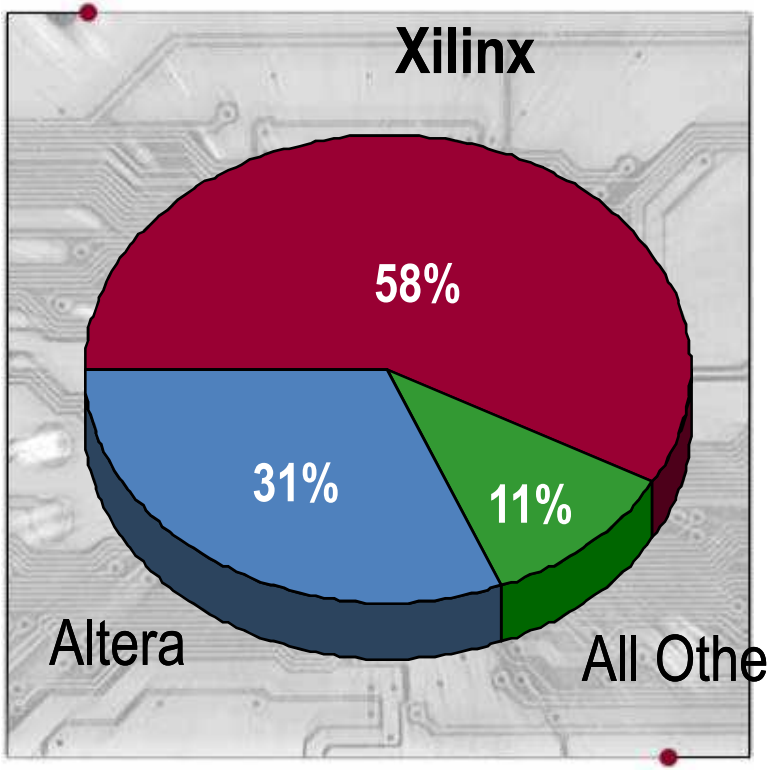
The Programmable Marketplace

Q1 Calendar Year 2005

PLD Segment



FPGA Sub-Segment



Two dominant suppliers, indicating a maturing market

Doua implementari concurente

ASIC
Application Specific Integrated Circuit

- un design trebuie trimis pentru fabricare intr-un fab de semiconductoare. Procesul dureaza timp si este foarte costisitor.
- sunt proiectate complet, de la descrierea comportamentala pana la layout-ul fizic

FPGA
Field Programmable Gate Array

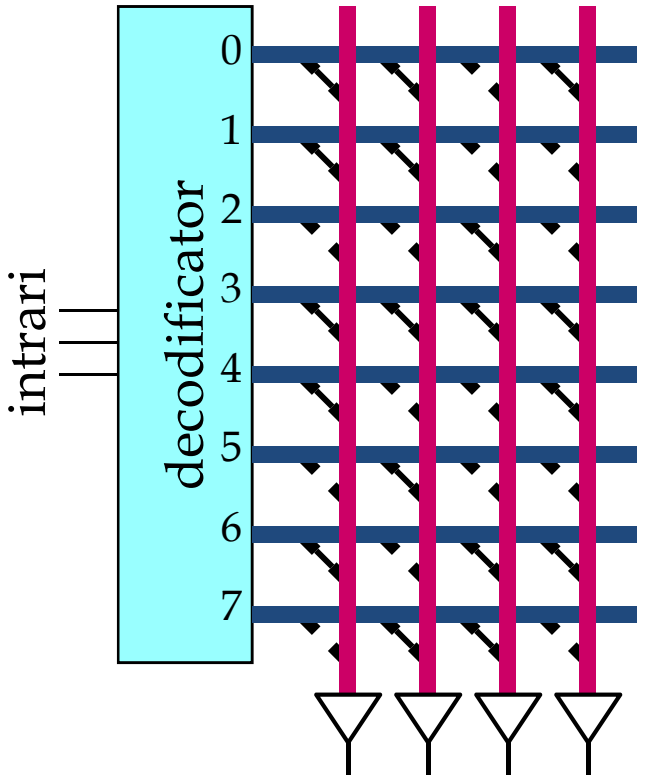
- reconfigurat de fiecare data de catre proiectanti
- nu se proiecteaza un layout fizic al componentelor. Proiectarea are ca rezultat un bitstream cu care se configureaza dispozitivul.

Programmable Logic Device (PLD)

- Sunt matrici simple de circuite logice
  - Implementeaza functii logice pe doua niveluri (AND/OR)
  - Au o structura simpla de interconectare programabila
  - Sunt de mia multe tipuri
    - Read Only Memory (ROM si PROM)
    - Programmable Logic Array (PLA)
    - Programmable Array Logic (PAL)
- Field Programmable Gate Arrays (FPGA)
  - Multe copii ale aceleiasi structuri configurabile de baza
  - Fiecare bloc poate fi configurat pentru a indeplini orice functie logica si include de obicei un flip-flop si un generator de functii cu 4 intrari
  - Interconectare programabila
  - Blocuri de memorie SRAM
  - Un FPGA mare are de obicei 100k+ circuite flip-flop, 100k generatoare de functii si 10Mb SRAM

Implementarea unei memorii ROM

- ROM poate fi implementata printr-un aranjament ortogonal de conexiuni
  - Conexiune la fiecare intersectie = 1 logic
  - Decodificatorul pune 1 logic pe linia selectata iar daca conexiunea este facuta, la iesire se poate citi un octet de date

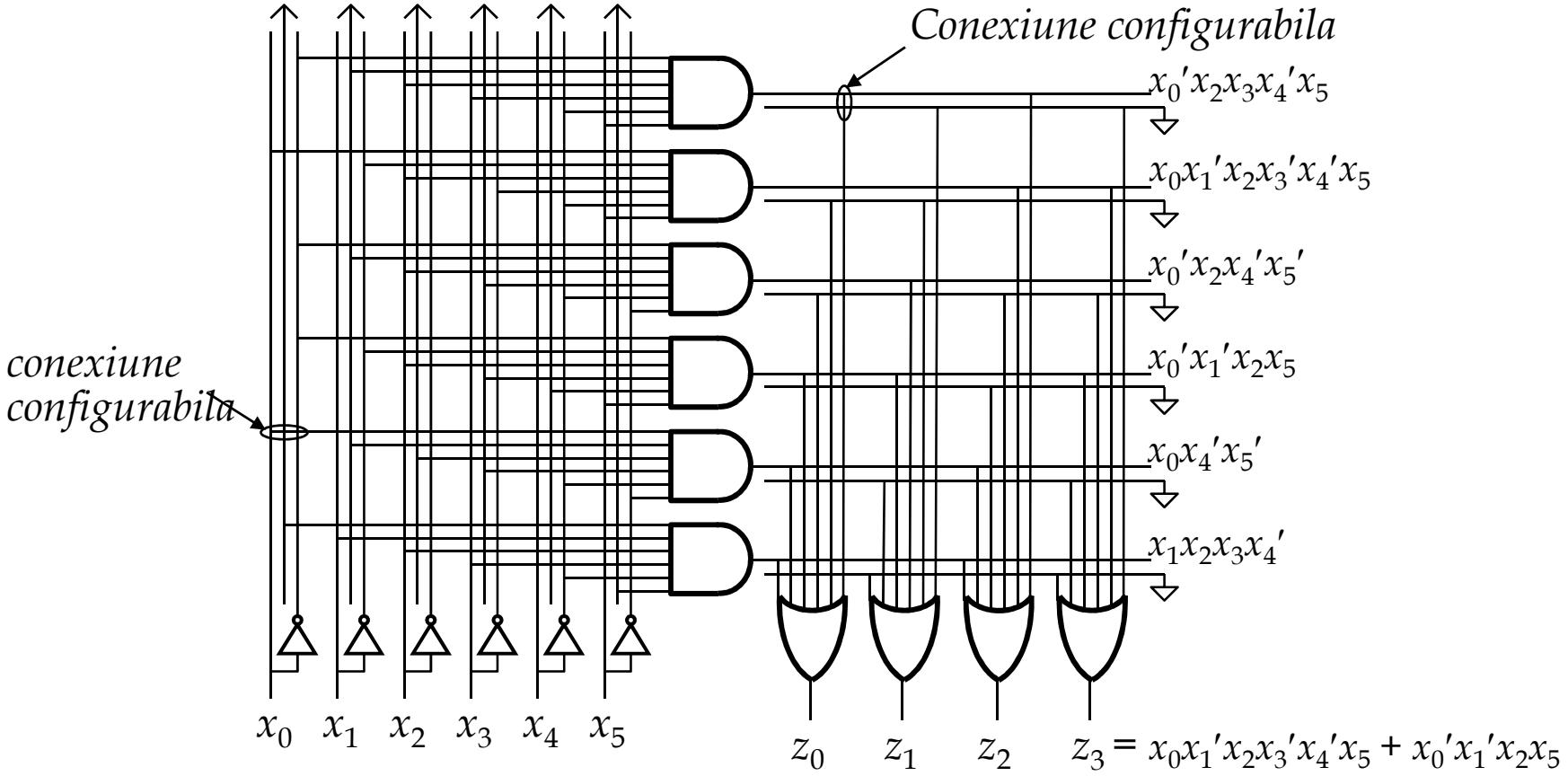


- Unele PROM-uri sunt scrise prin eliminarea conexiunilor
  - » Tensiune mare aplicata pe linia si coloana pe care se vrea marcarea unui 0 logic
  - » Curentul mare aplicat legaturii linie-coloana determina “arderea” legaturii
- Alte memorii pot fi arse si reprogramate (EPROM, EEPROM)

Logica in RAM/ROM

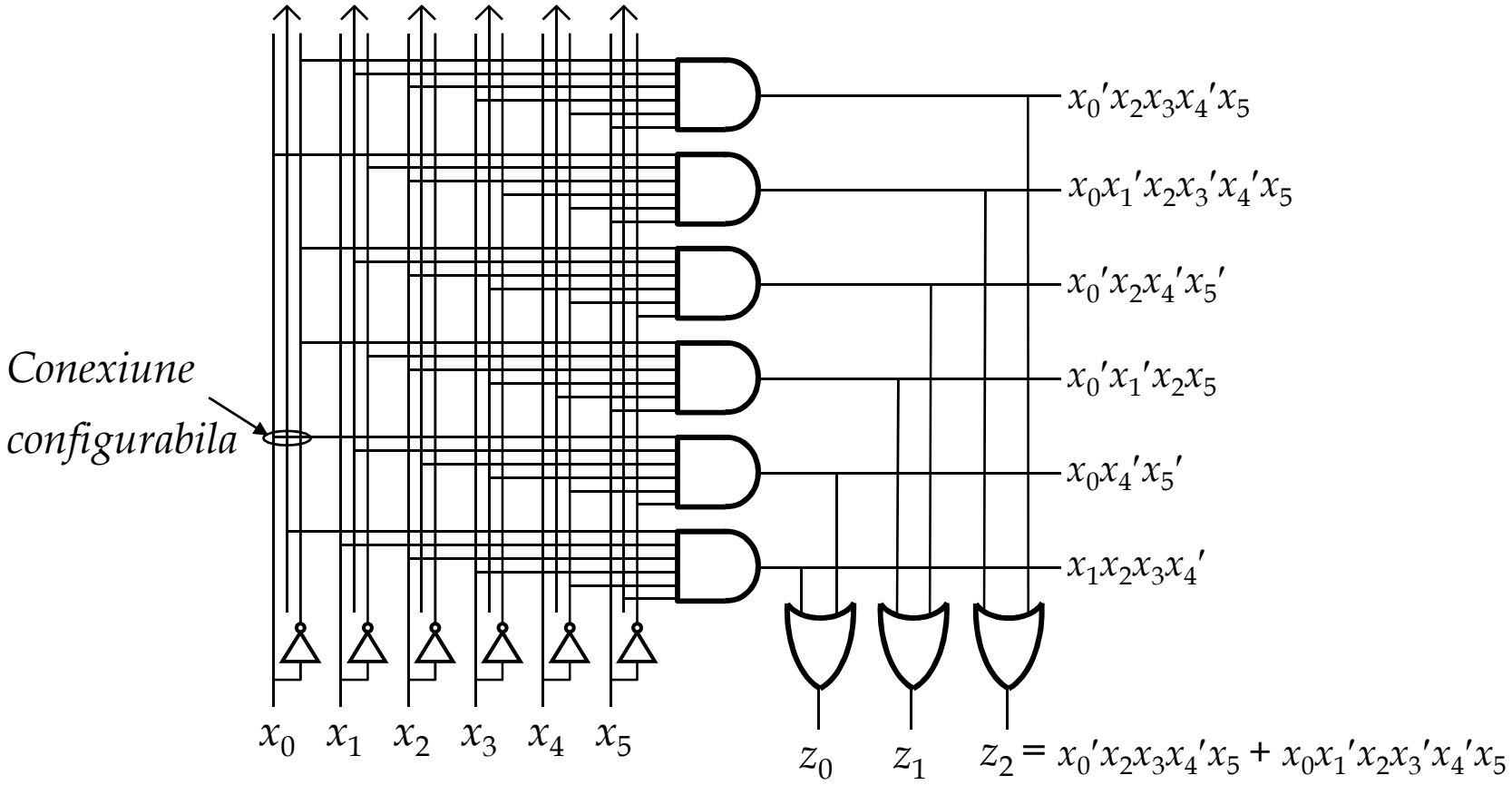
- Orice memorie RAM poate implementa functii logice reconfigurabile
  - Stocheaza tabela de adevar a functiei logice in memorie
  - Exemplu: folosesc RAM de 4 biti pentru a simula o poarta SI cu doua intrari stocand 0 la adresele 00, 01 si 10 si 1 la adresa 11
  - cu 2^m cuvinte de 1 bit se poate implementa orice functie de m intrari
  - O memorie cu 2^m cuvinte de w biti latime poate implementa w functii logice diferite cu m intrari
- Memoriile ROM sau EEPROM pot implementa aceleasi functii logice dar cu anumite avantaje:
  - Memoria este nevolatila
  - Datele sunt stocate la programare si pot fi reconfigurate printr-un update de firmware
  - Densitate mai mare decat memoria RAM

Programmable Logic Array (PLA)



- PLA-urile au un plan SI si un plan SAU
- Pot sa implementeze orice circuit de doua niveluri (SAU/SI)
- Implementate in CMOS.

Programmable Array Logic (PAL)

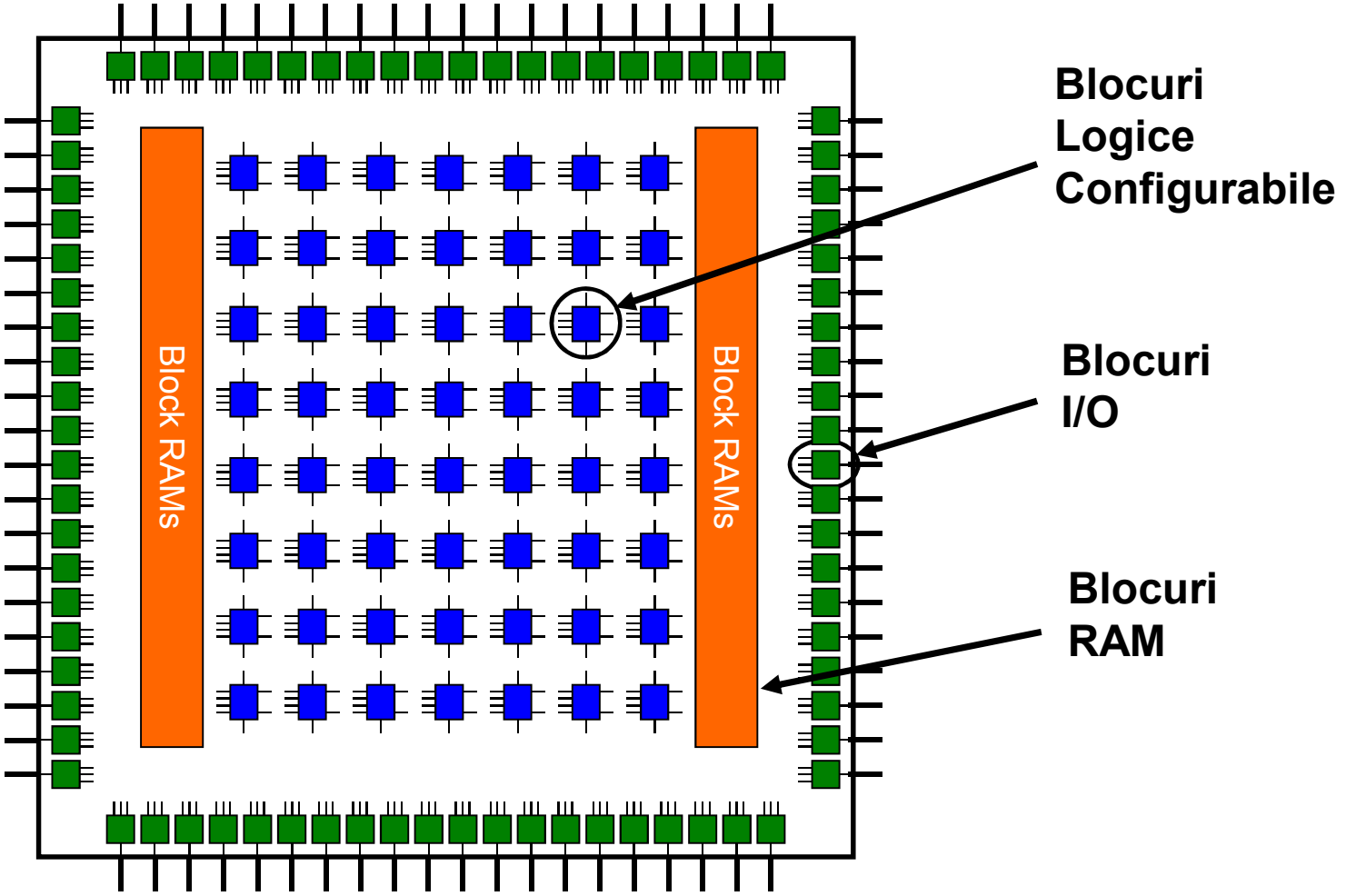


- PAL este similar cu PLA dar are un plan SAU fix.
- Mai usor de programat (si mai ieftin de produs)
- Dezavantaj: numar limitat de termeni generati la iesire

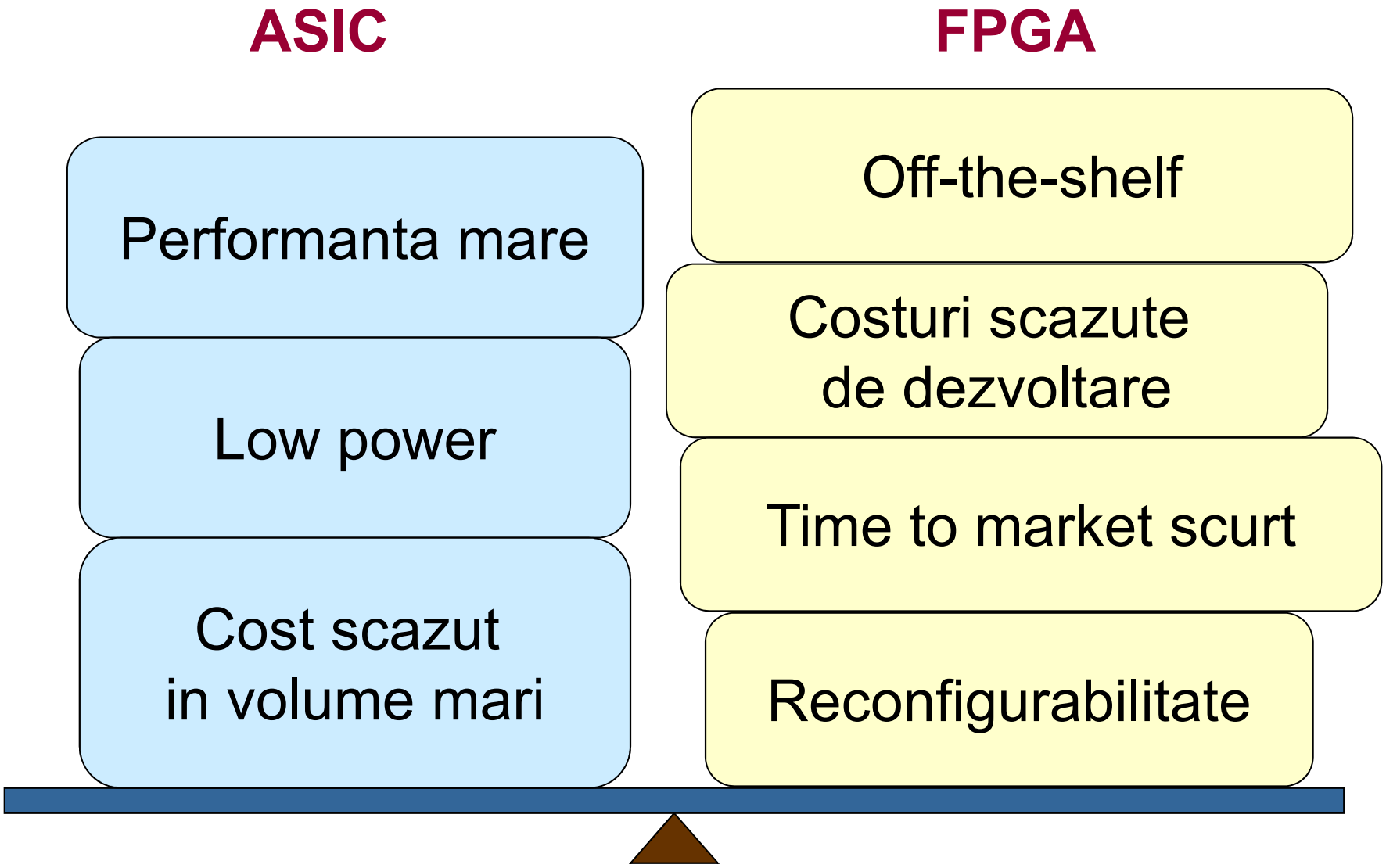
Field Programmable Gate Array (FPGA)

- Circuitele FPGA sunt folosite pentru generarea circuitelor logice complexe.
- Un chip contine un numar foarte mare (zeci de mii) de blocuri logice configurabile.
  - Programele de tip CAD mapeaza circuitele de nivel inalt peste matricea de blocuri de baza prin configurarea generatoarelor de functii, interconexiunilor si altor elemente configurabile
- Blocurile logice sunt rutate folosind interconexiuni programabile
  - Segmentele sunt conectate la blocurile logice si la alte segmente invecinate prin switch-uri configurabile
  - Programele CAD determina configuratia optima pentru toate switch-urile folosite.

Ce este un FPGA?



Care sunt optiunile?

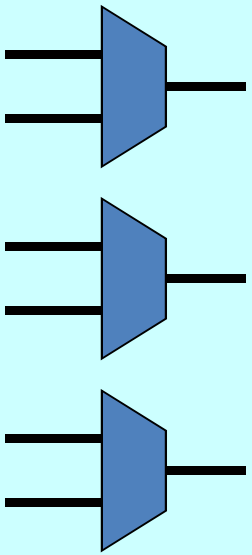


Alte avantaje FPGA

- Ciclul de fabricatie al unui ASIC este foarte costisitor, lung si necesita proiectanti din mai multe domenii
  - Greselile care nu sunt detectate la proiectare au un impact foarte mare asupra aspura costurilor si a timpului de dezvoltare
- FPGA=urile sunt perfecte pentru prototiparea rapida a unui circuit digital
- Se pot face upgrade-uri foarte usor, ca si cum ar fi in cazul unui software
- Domenii unice de aplicatii
  - Sisteme reconfigurabile

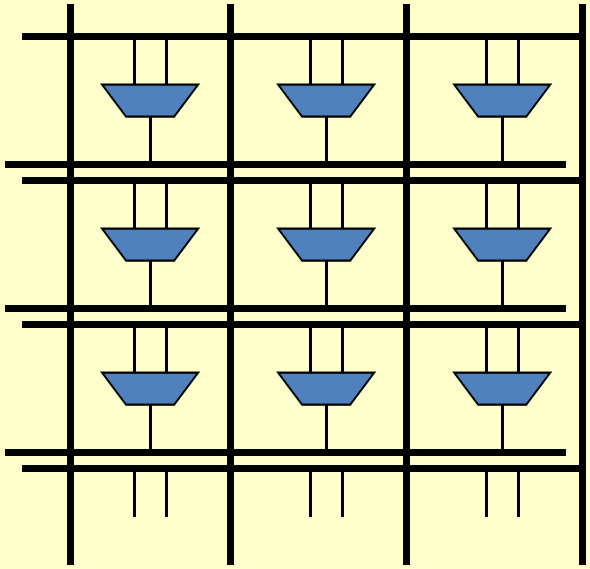
Latima de banda computationala

Fixa



CPU

“Fara limite”



RC

Producatorii majori de FPGA

FPGA SRAM

- **Xilinx, Inc.**
- **Altera Corp.**

} **Aproape 60% din piata**

- Atmel
- Lattice Semiconductor

FPGA Flash

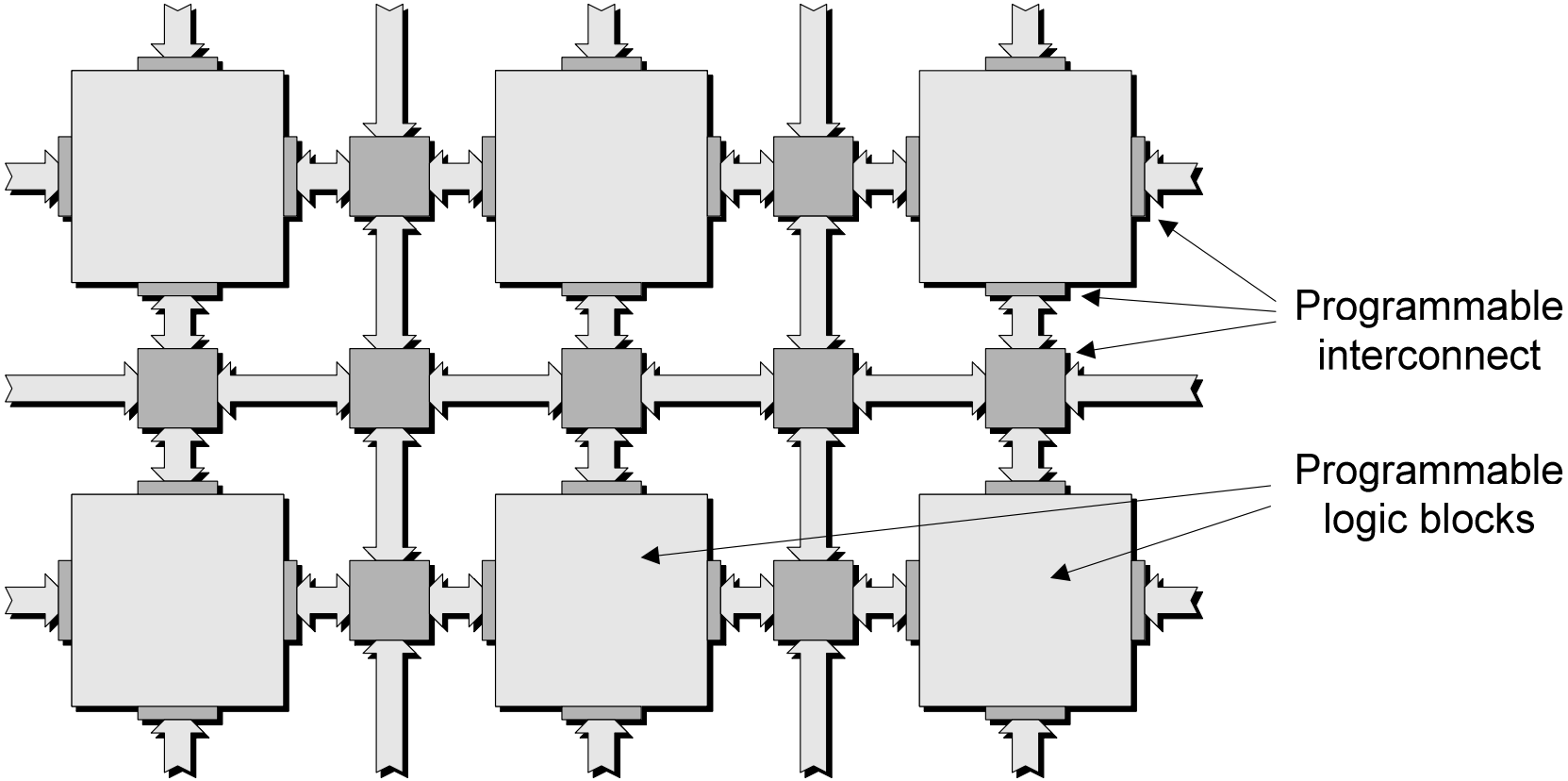
- Actel Corp.
- Quick Logic Corp.

Familii de FPGA Xilinx

- Familii vechi
  - XC3000, XC4000, XC5200
  - Tehnologie 0.5µm, 0.35µm si 0.25µm . Nerecomandate pentru produse noi.
- **Familii High-Performance**
  - Virtex (0.22µm)
  - Virtex-E, Virtex-EM (0.18µm)
  - Virtex-II, Virtex-II PRO (0.13µm)
  - Virtex-4 (0.09µm)
  - Virtex-5
- **Familii Low Cost**
  - Spartan/XL – derivat din XC4000
  - Spartan-II – derivat din Virtex
  - Spartan-IIE – derivat din Virtex-E
  - **Spartan-3**
  - **Spartan-3E**

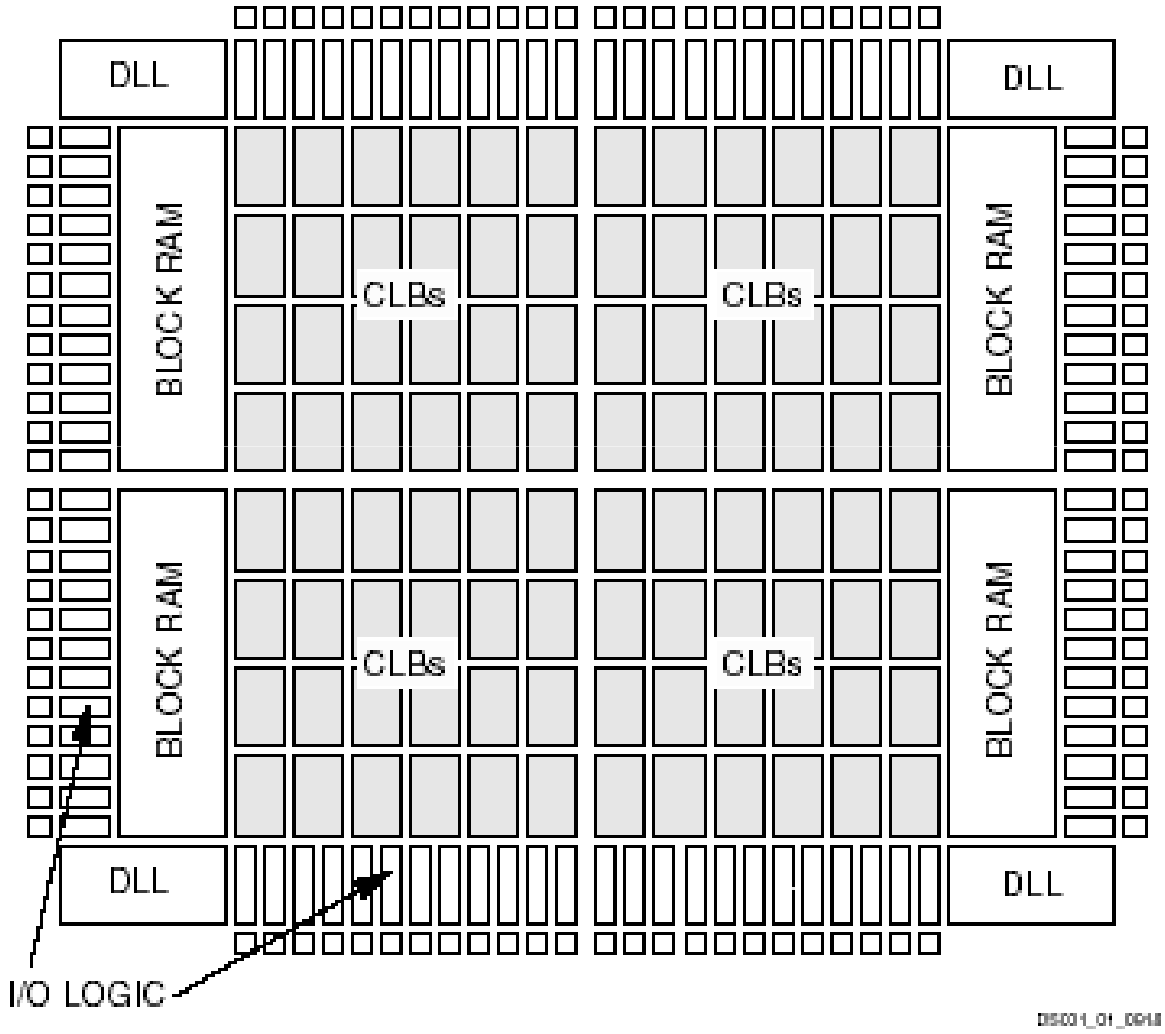
Prezentare

Structura generala a unui FPGA

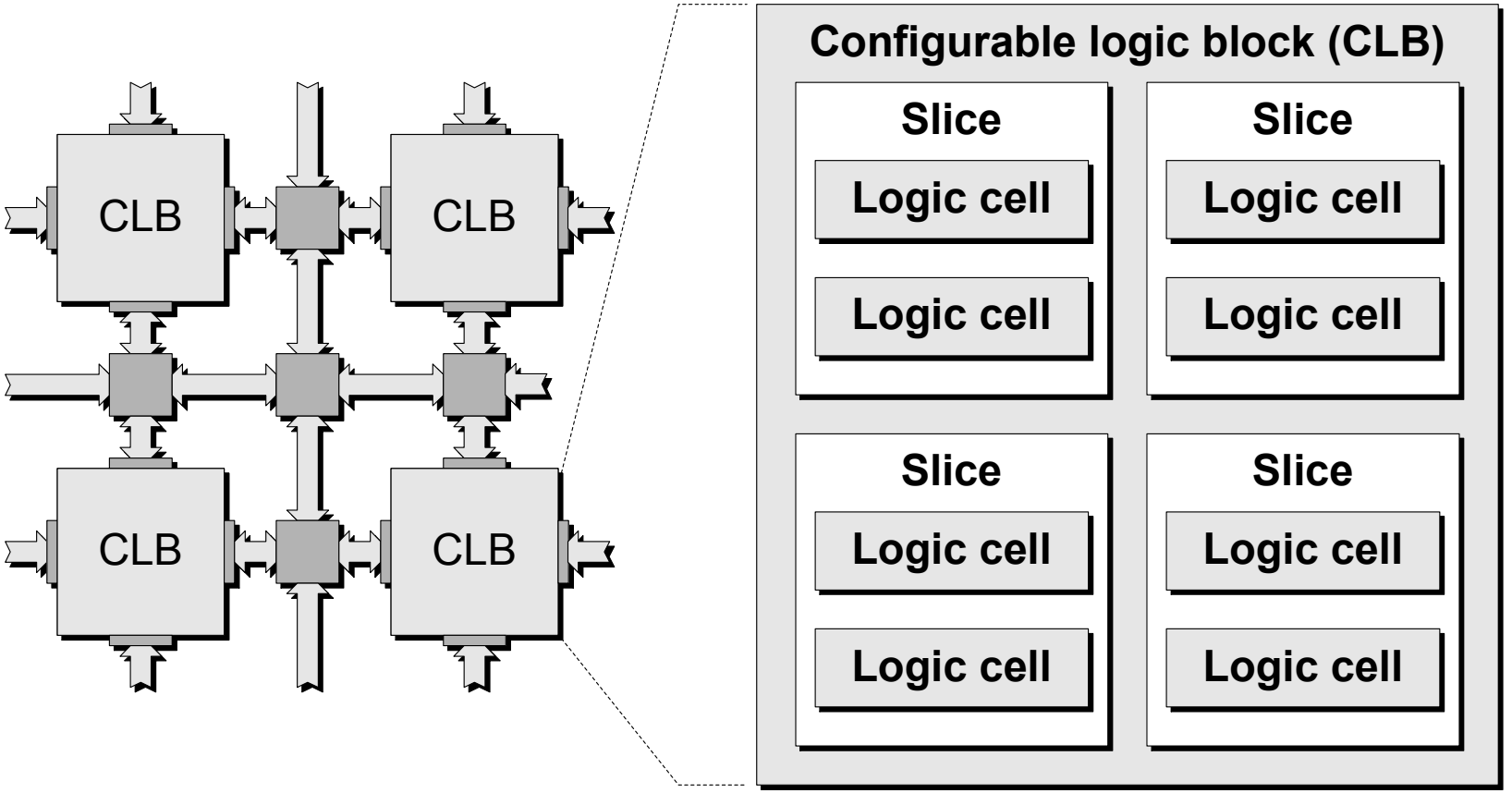


The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)

Xilinx FPGA Block Diagram

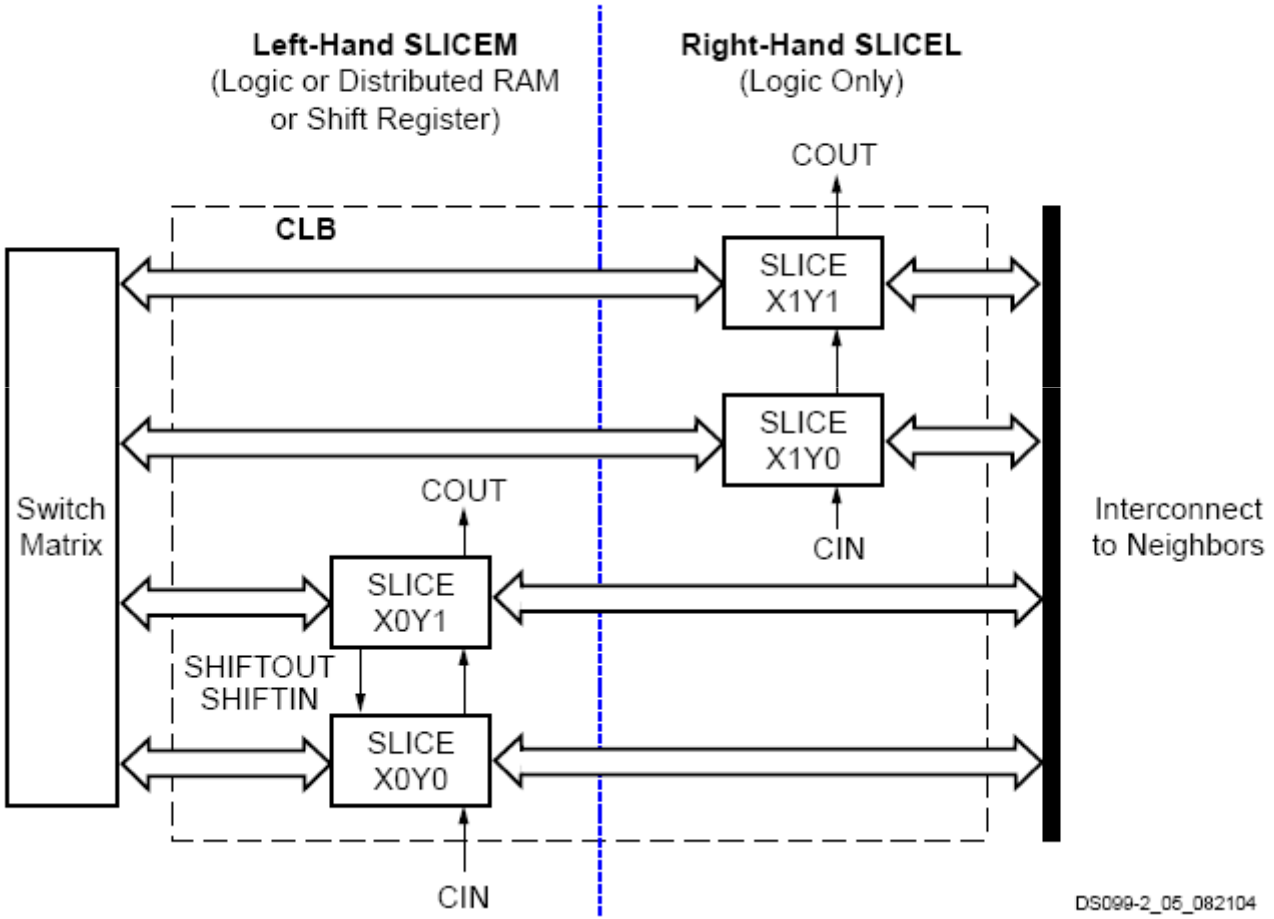


Xilinx CLB

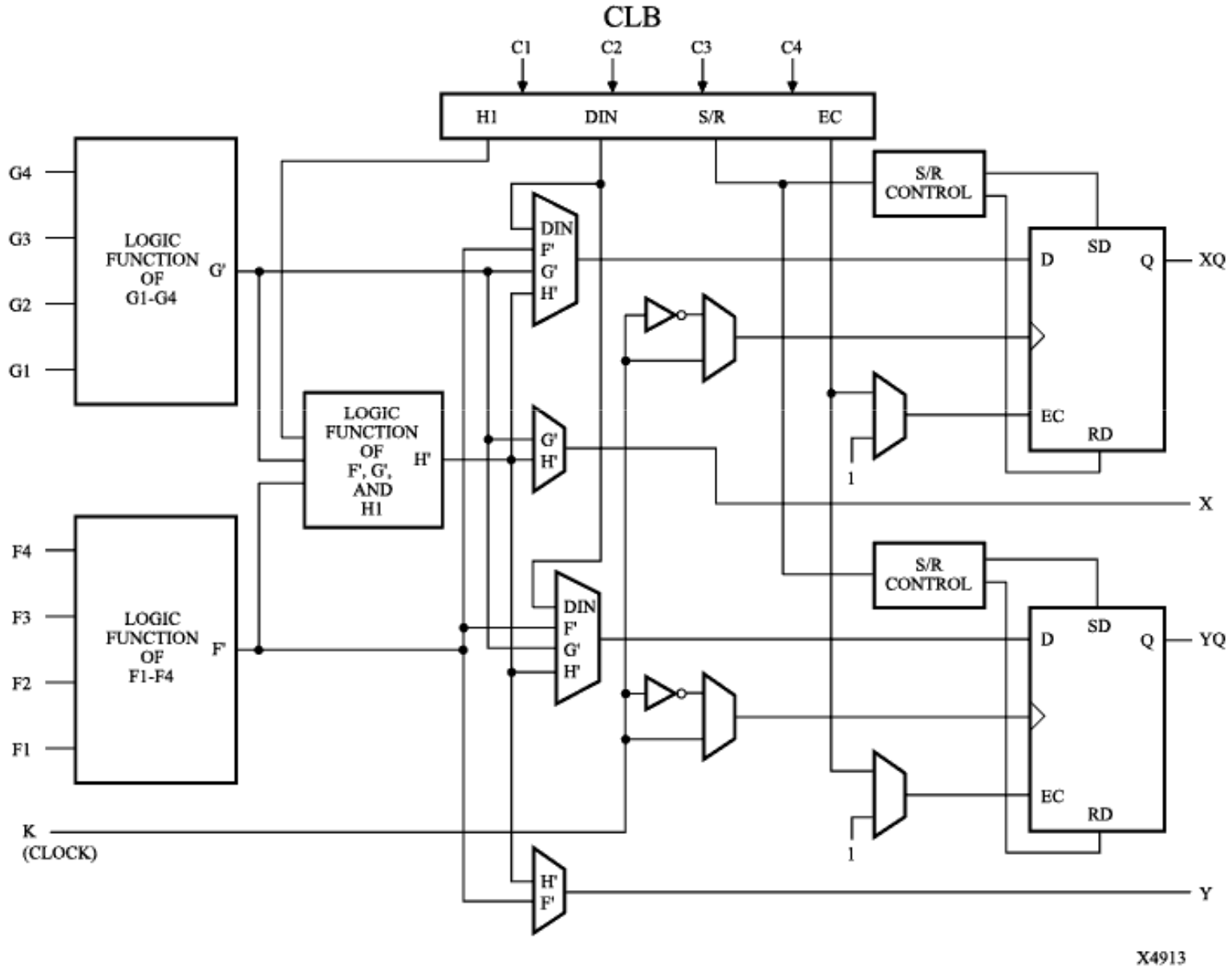


The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)

Configurable Logic Block CLB

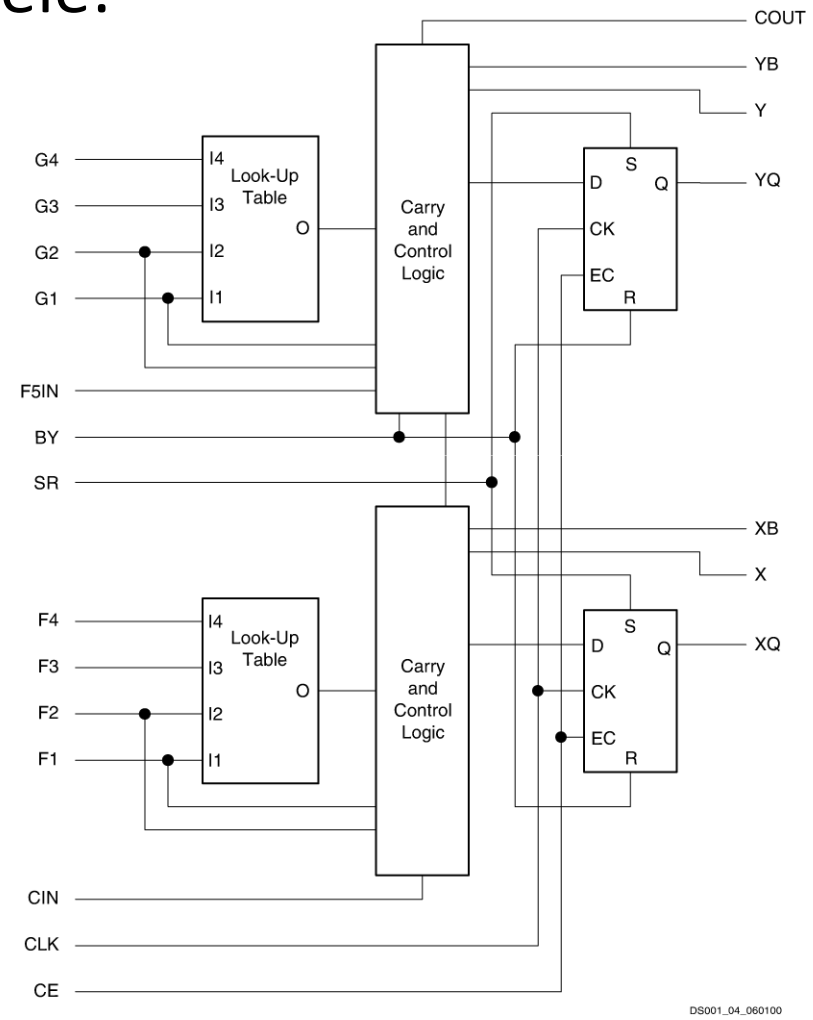


CLB Slice


CLB
C1
C2
C3
C4
H1
DIN
S/R
EC
G4
G3
G2
G1
LOGIC FUNCTION OF G1-G4
G'
LOGIC FUNCTION OF F', G', AND H1
H'
F4
F3
F2
F1
LOGIC FUNCTION OF F1-F4
F'
K (CLOCK)
DIN
F'
G'
H'
G'
H'
H'
F'
S/R CONTROL
S/R CONTROL
D
SD
Q
EC
RD
XQ
YQ
1
1
X
Y
X4913

Structura unui slice CLB

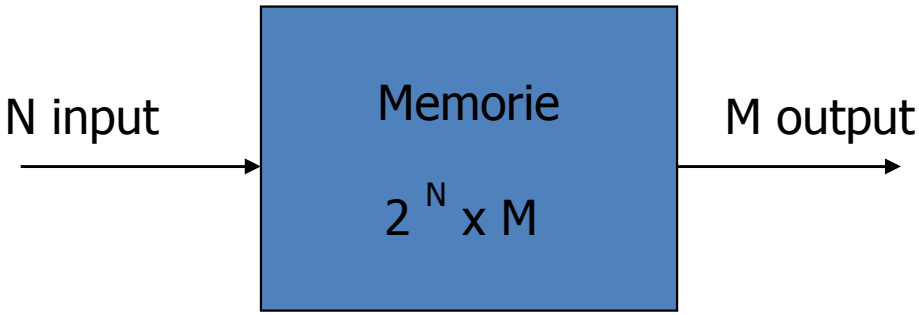
- Fiecare slice contine urmatoarele:
  - LUT cu patru intrari
    - Orice functie logica cu 4 intrari,
    - sau RAM 16bit x 1
    - sau Registru Shift pe 16 biti
  - Carry & Control
    - Logica aritmetica rapida
    - Logica pentru inmultire
    - Logica de multiplexare
  - Elemente de stocare
    - Latch sau flip-flop
    - Set / reset
    - Iesiri normale sau inversate
    - Control pentru functionare sincron/asincron



Functiile Logice

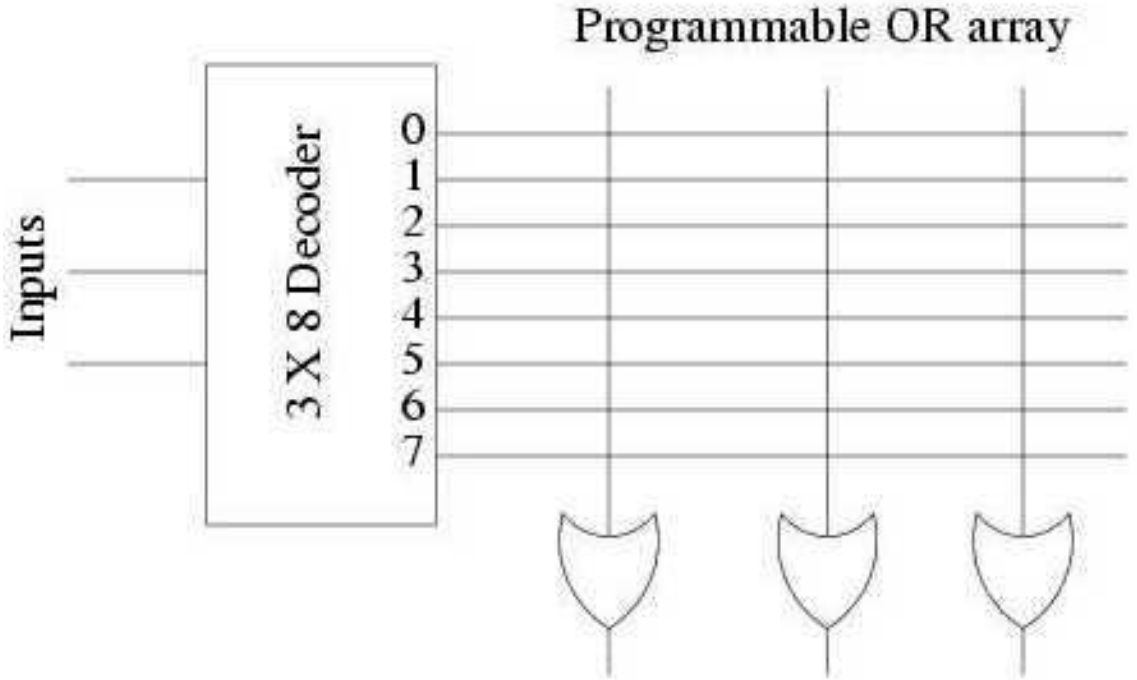
- Implementate folosind Look-up Table (LUT)
- Un LUT cu k intrari corespunde unei memorii de 2^K x 1 bit
- Un LUT cu k intrari poate implenta orice functie logica cu k intrari si o singura iesire function

Lookup Table (LUT)



- Adrese: N biti; Iesire: M biti
- Memoria contine 2^N cuvinte a cate M biti
- Valorile intrarilor decid cuvantul care va fi disponibil la iesire la un moment dat

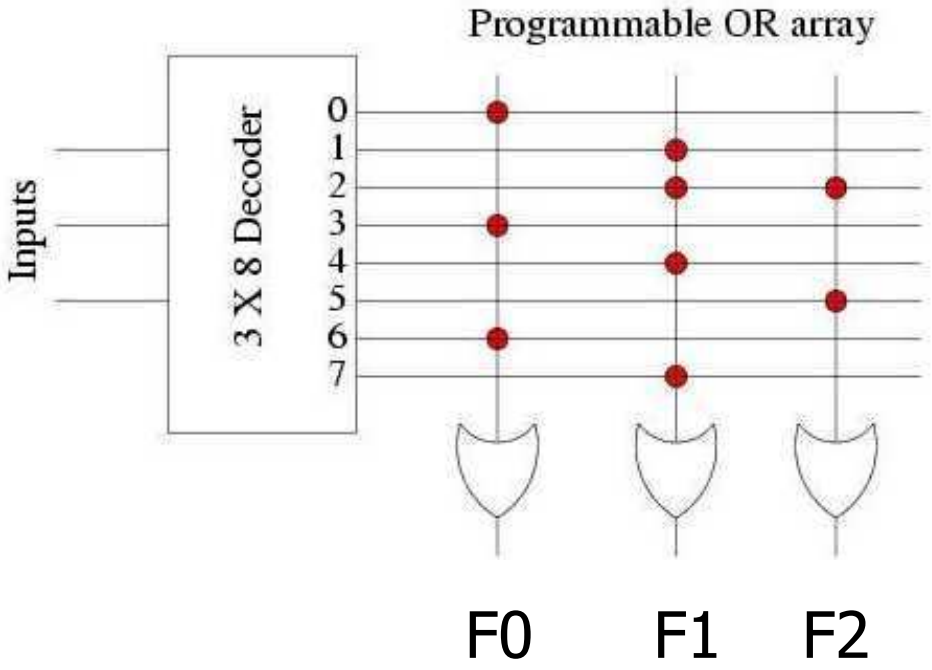
Diagrama logica a unui LUT 8x3



Suma de mintermeni

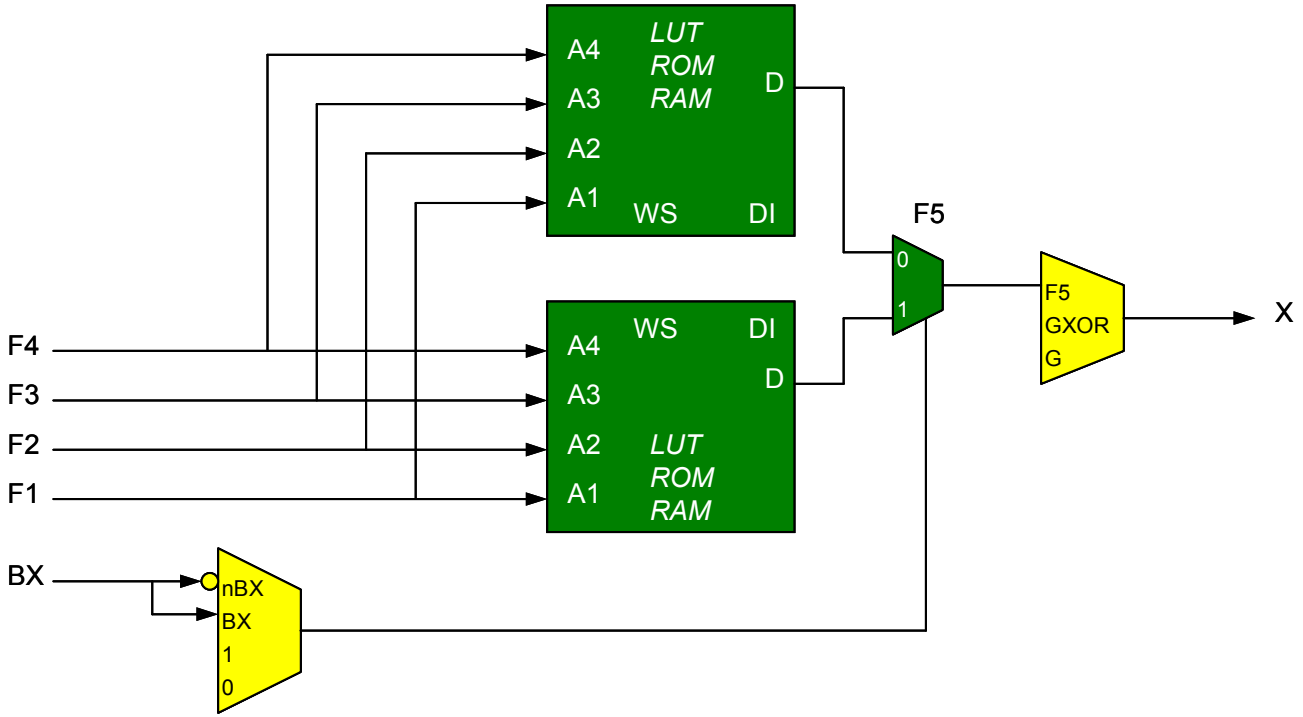
Implementarea unei functii logice folosind LUT

I0	I1	I2	F0	F1	F2
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	1	0

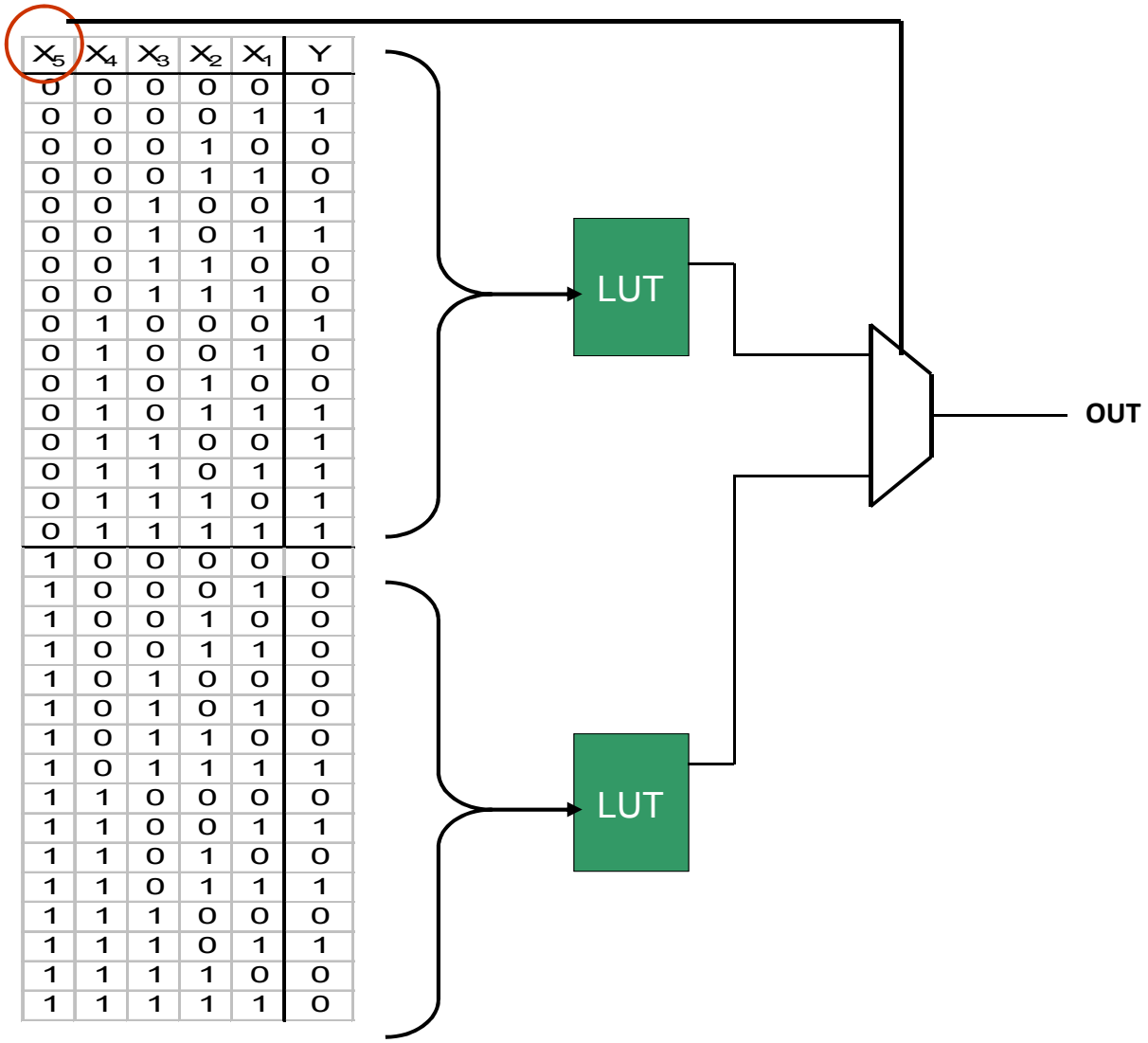


Implementarea unei functii logice cu 5 intrari folosind 2 LUT

- Un slice CLB poate sa implementeze si fucntii cu mai mult de 4 intrari logice, folosind doua LUT
- Functia este partitionata intra cele doua LUT
- Multiplexorul final selecteaza iesirea corecta

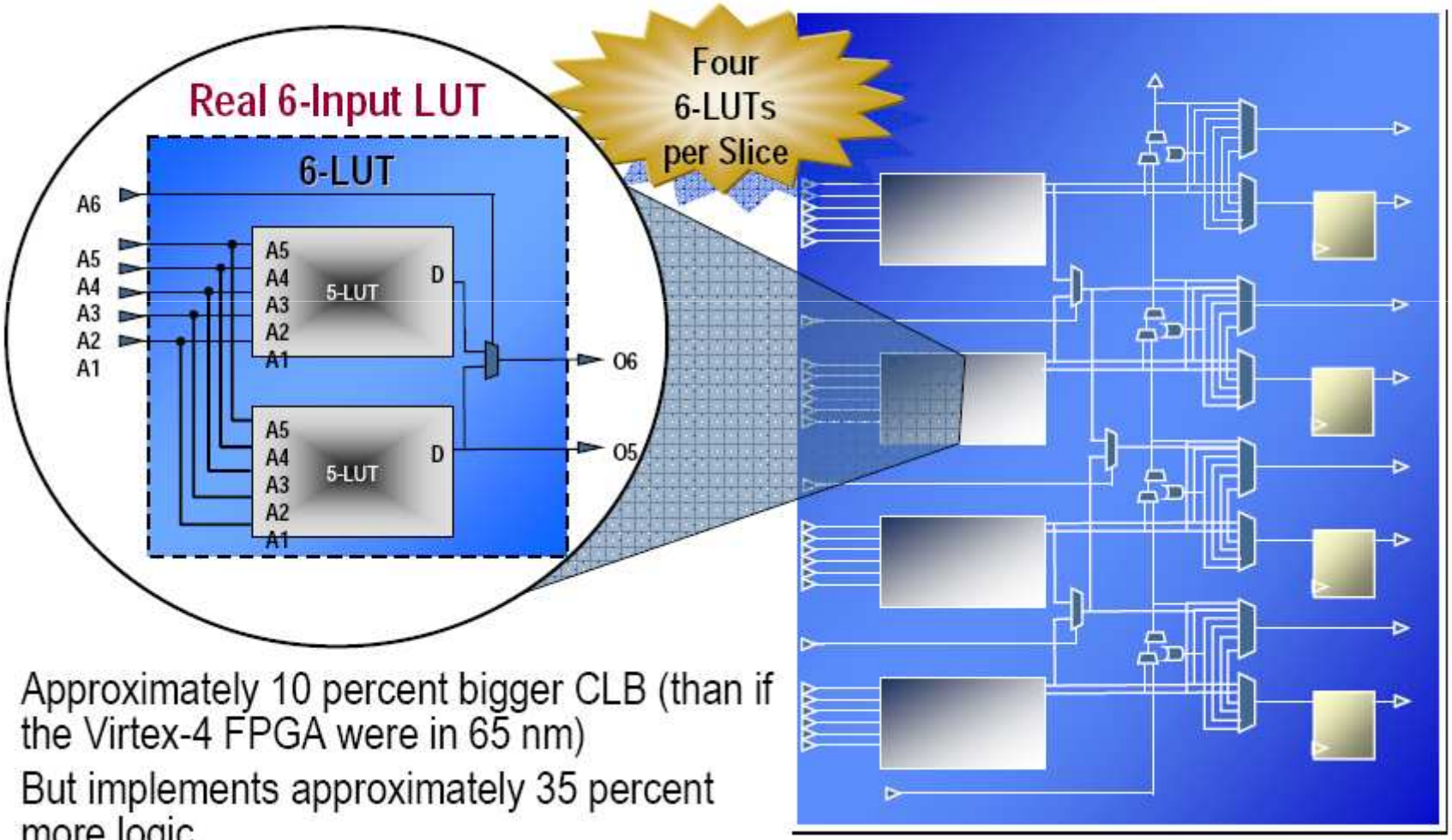


Implementarea unei functii logice cu 5 intrari folosind 2 LUT



X_5	X_4	X_3	X_2	X_1	Y
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	0

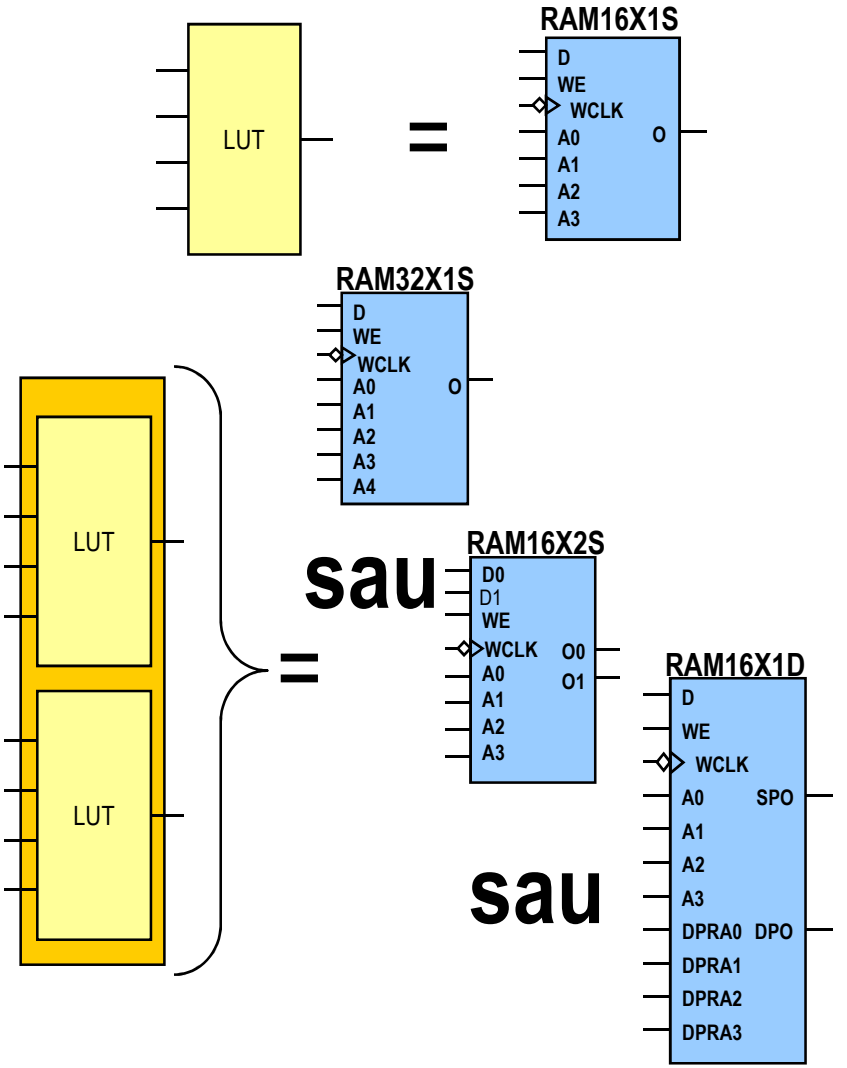
Exemplu Virtex



- Approximately 10 percent bigger CLB (than if the Virtex-4 FPGA were in 65 nm)
- But implements approximately 35 percent more logic

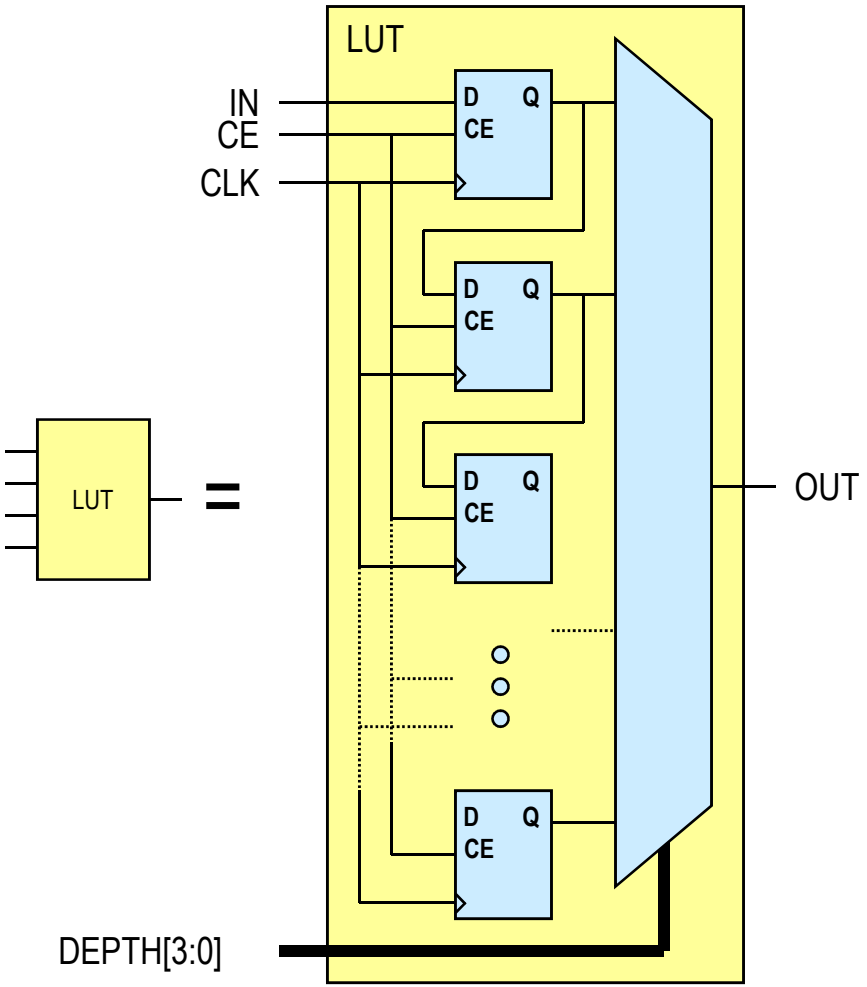
RAM Distribuit

- CLB LUT configurabil ca RAM
  - Un LUT egal 16x1 RAM
  - Memoria poate fi Single sau Dual-Port
  - Cascadarea LUT-urilor mareste latimea memoriei
- Scriere Sincrona
- Citire Sincrona/Asincrona
  - Circuitele flip-flop pot fi folosite pentru implmentarea citirii sincrone

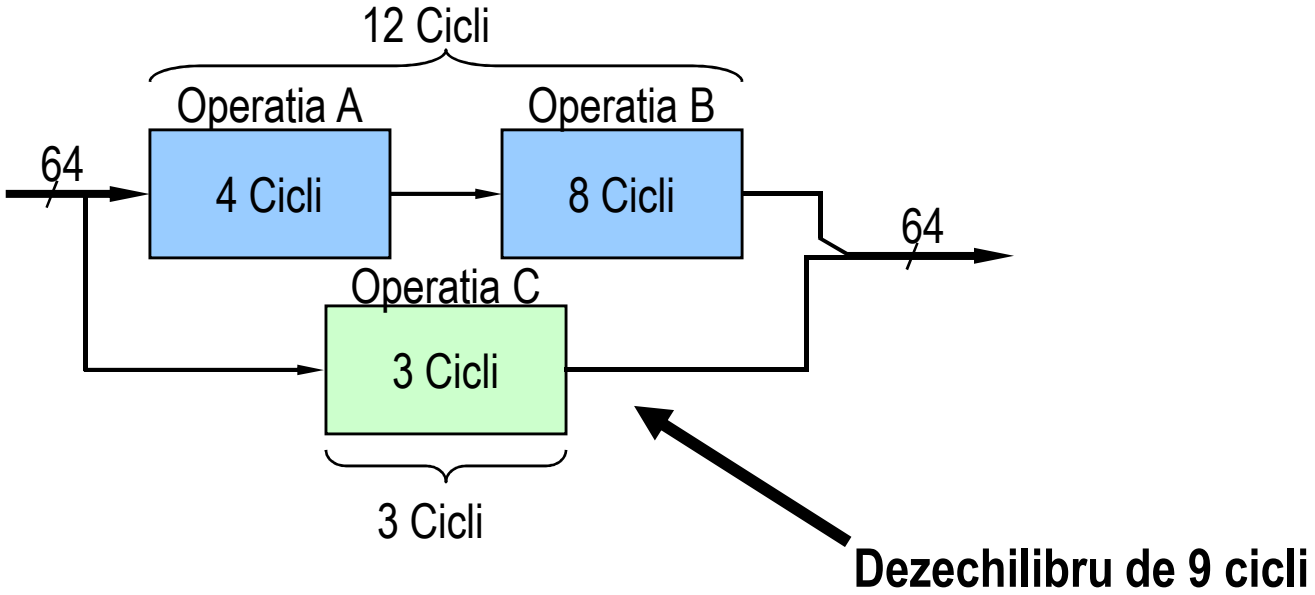


Registru Shift

- Fiecare LUT poate fi configurat ca un registru shift
  - Serial in, serial out
- Intarziere dinamica de pana la 16 cicli
- Cascadarea mareste numarul ciclilor de intarziere



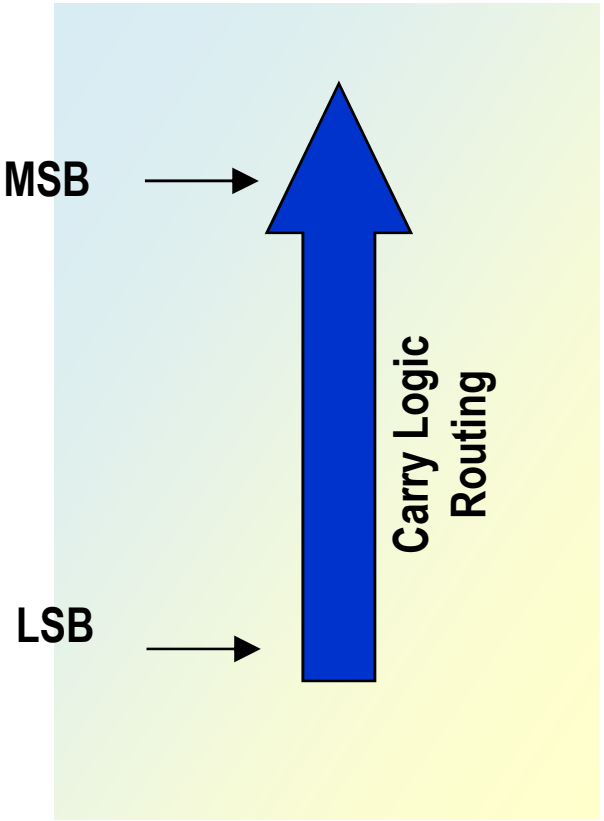
Registru Shift



- FPGA bogat in registre
  - Permite adaugarea de etaje pipeline pentru a mari productivitatea
- Caile de date trebuie sa fie echilibrate pentru a pastra funcionalitatea sistemului

Fast Carry Logic

- Fiecare CLB contine logica separata pentru rutare si generarea rapida a semnalelor de suma si carry
  - Mareste eficienta si performantele sumatoarelor , multiplicatoarelor, acumulatoarelor, comaparatoarelor si numaratoarelor
- Logica de carry este independenta de logica normala si poate lucra in conjunctie cu LUT



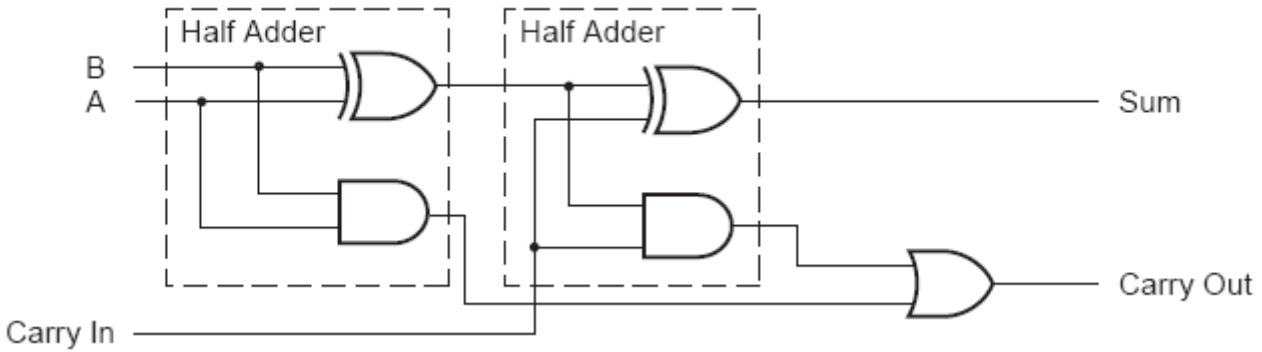
Accesarea Carry Logic

- Toate tool-urile majore de siteza pot invoca carry logic pentru functiile aritmetice
  - Adunare (SUM <= A + B)
  - Scadere (DIFF <= A - B)
  - Comparatii (if A < B then…)
  - Numaratoare(count <= count +1)

Implementarea Carry Logic

Abordarea clasica: Half Adder

A	B	Sum (A XOR B)	Carry Out (A AND B)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Dezavantaje:

1. Propagare lenta a carry
2. Utilizeaza 2 LUT-uri

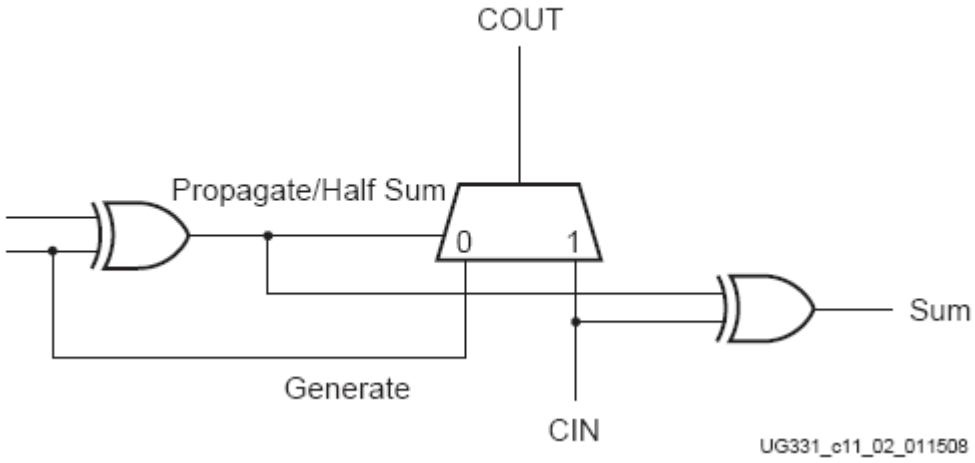
Implementarea Carry Logic

- Carry Look-Ahead

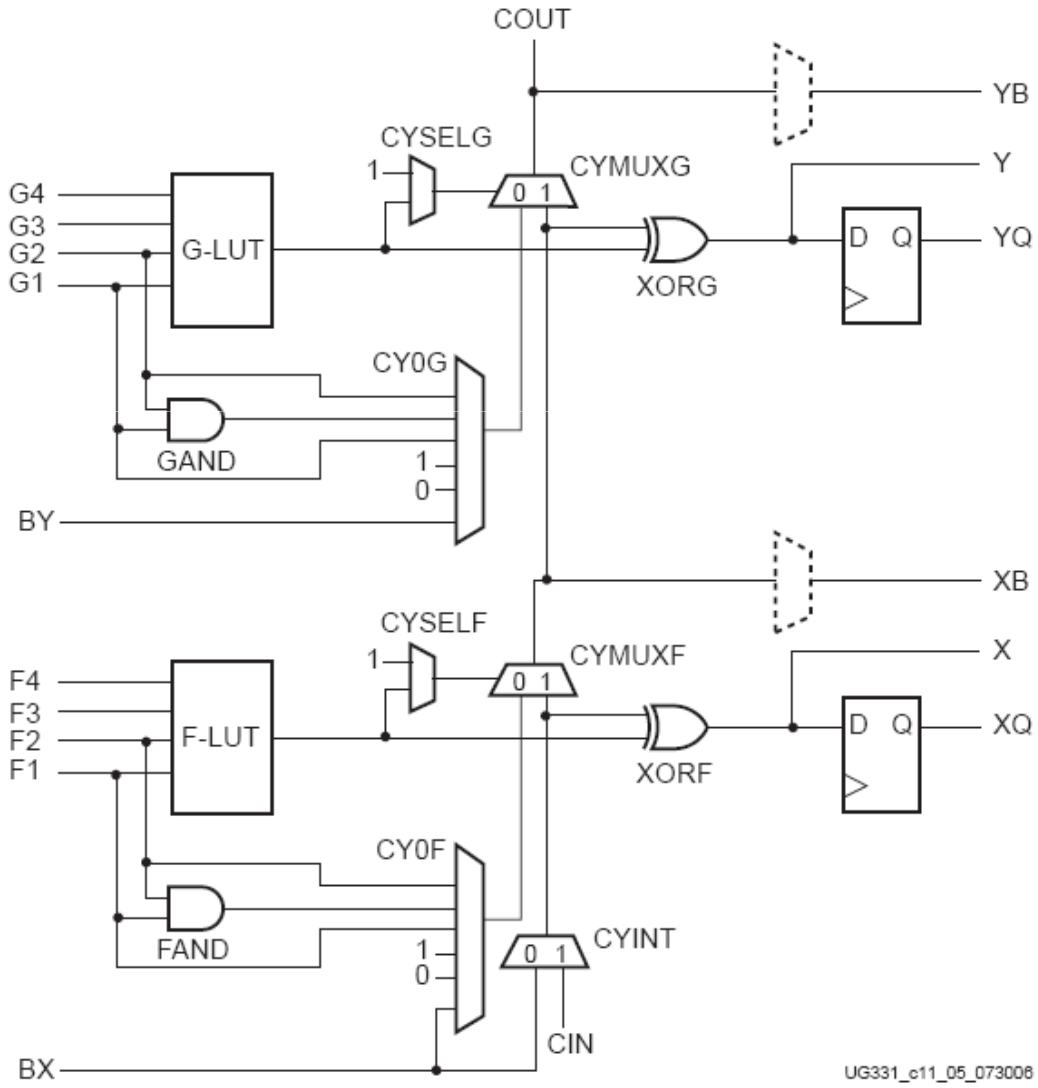
A	B	Propagate	Generate
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Avantaje:

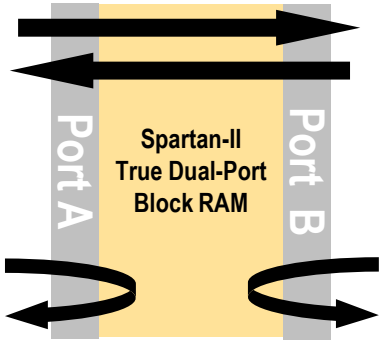
1. Foloseste un singur LUT (trei intrari)
2. Carry se propaga rapid – un singur MUX



Structura unui slice cu logica adaugata



Block RAM



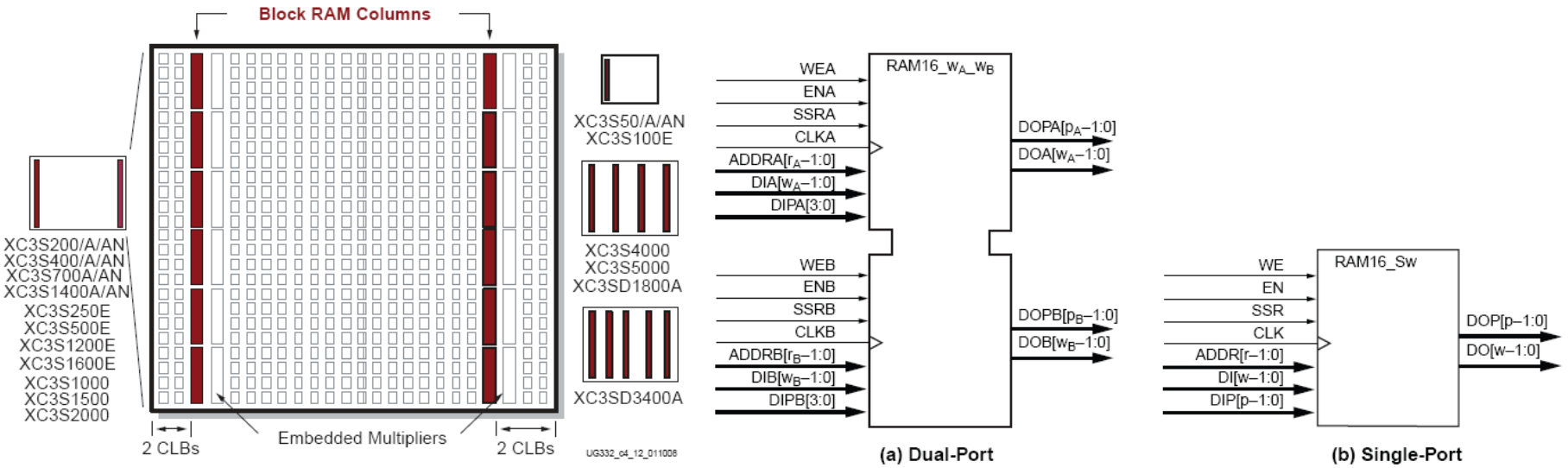
- Implementare eficienta a memoriei intre CLB si porturile IO
  - Blocuri dedicate
- Toata logica de control este implementata in celula de RAM
  - De la 4 la 104 blocuri de memorie
    - 18 kbiti pe bloc
  - Blocurile se pot cascada pentru implementarea unei memorii mai mari
- Poate functiona ca single sau dual-port RAM

Cantitatea de RAM familia Spartan 3

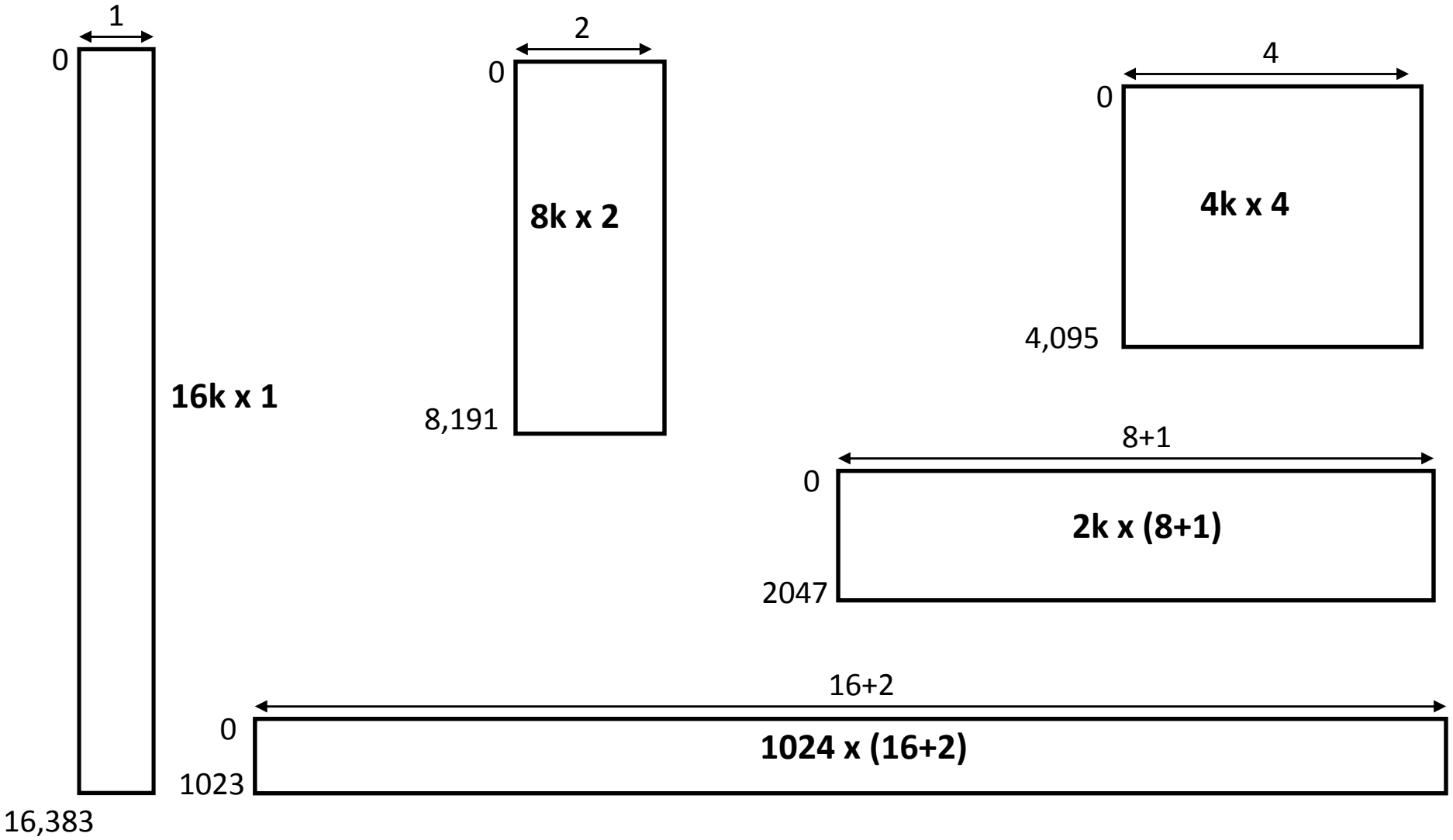
Device	Total Number of RAM Blocks	Total Addressable Locations (bits)	Number of Columns
XC3S50	4	73,728	1
XC3S200	12	221,184	2
XC3S400	16	294,912	2
XC3S1000	24	442,368	2
XC3S1500	32	589,824	2
XC3S2000	40	737,280	2
XC3S4000	96	1,769,472	4
XC3S5000	104	1,916,928	4

Block RAM Port Aspect Ratios

Total RAM bits, including parity	18,432 (16K data + 2K parity)
Memory Organizations	16Kx1 8Kx2 4Kx4 2Kx8 (no parity) 2Kx9 (x8 + parity) 1Kx16 (no parity) 1Kx18 (x16 + 2 parity) 512x32 (no parity) 512x36 (x32 + 4 parity) 256x72 (single-port only)

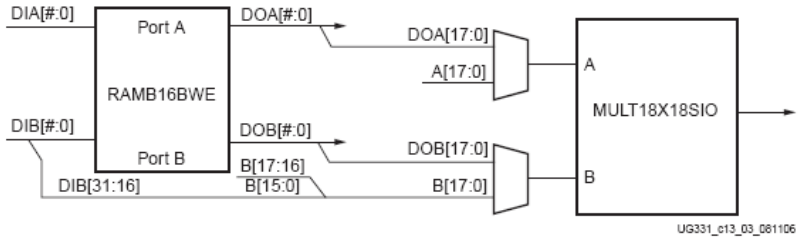


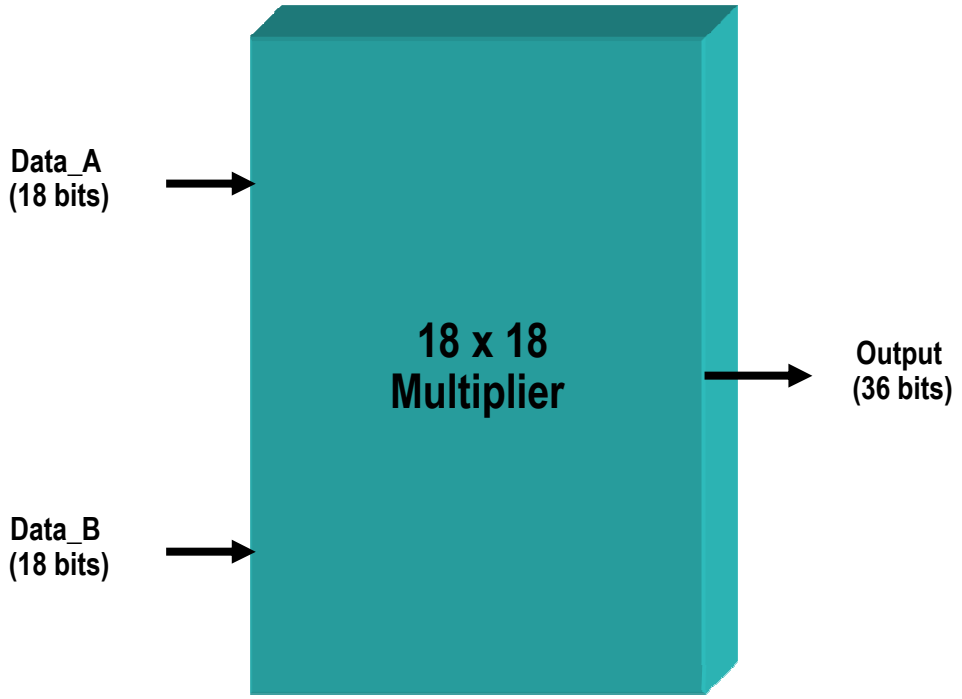
Block RAM Port Aspect Ratios



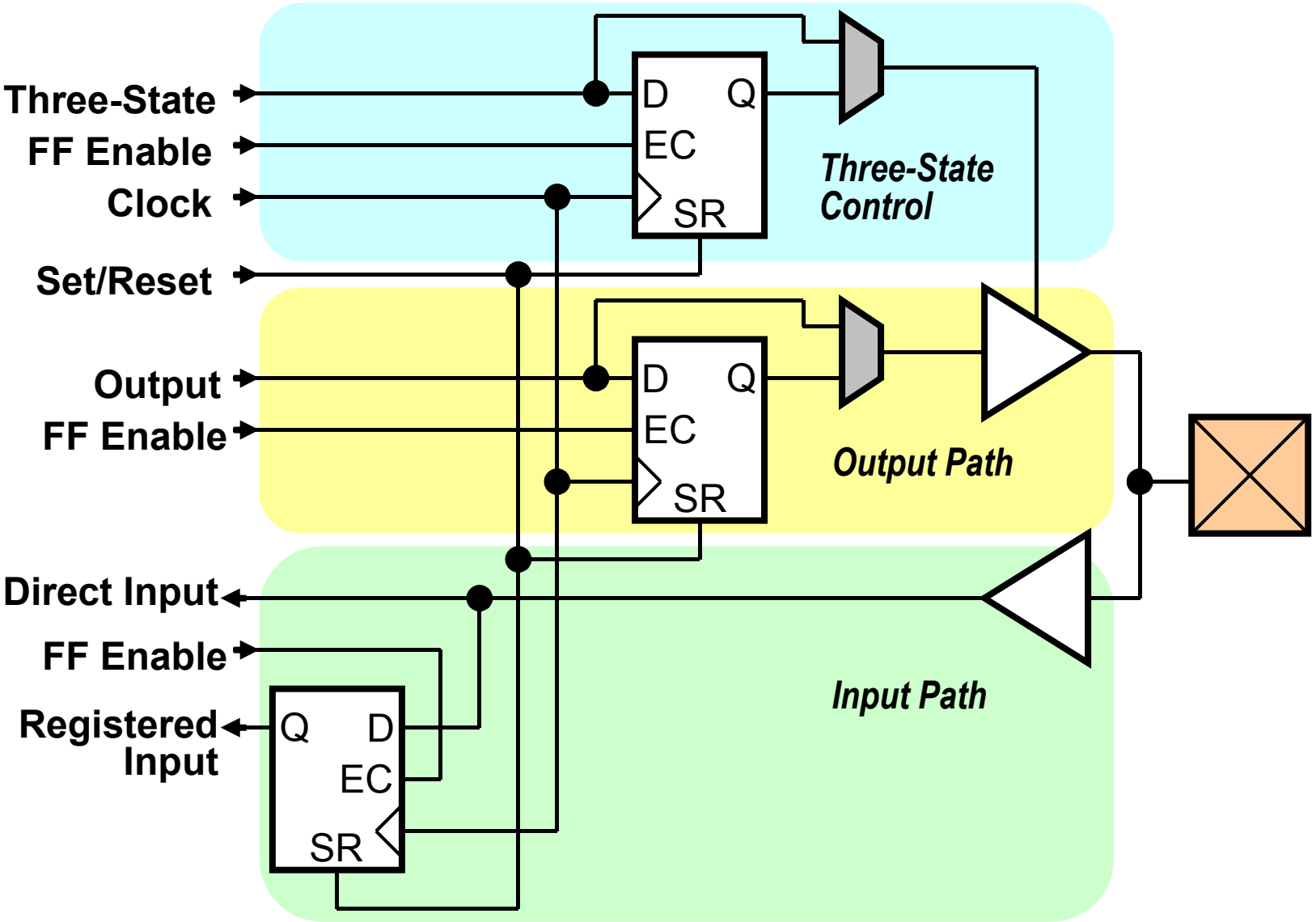
18 x 18 Embedded Multiplier

- Intareste functionalitatea de DSP a circuitului
  - Optimizat pentru viteza si performanta maxima. Implementeaza module tip multiply/accumulate
  - Sunt organizate in coloane adiacente coloanelor de blocuri RAM
  - Fiecare multiplicator are doi operanzi de 18 biti latime si este implementat in logica pur combinationala. Se conecteaza prin magistrala la blocul RAM adiacent





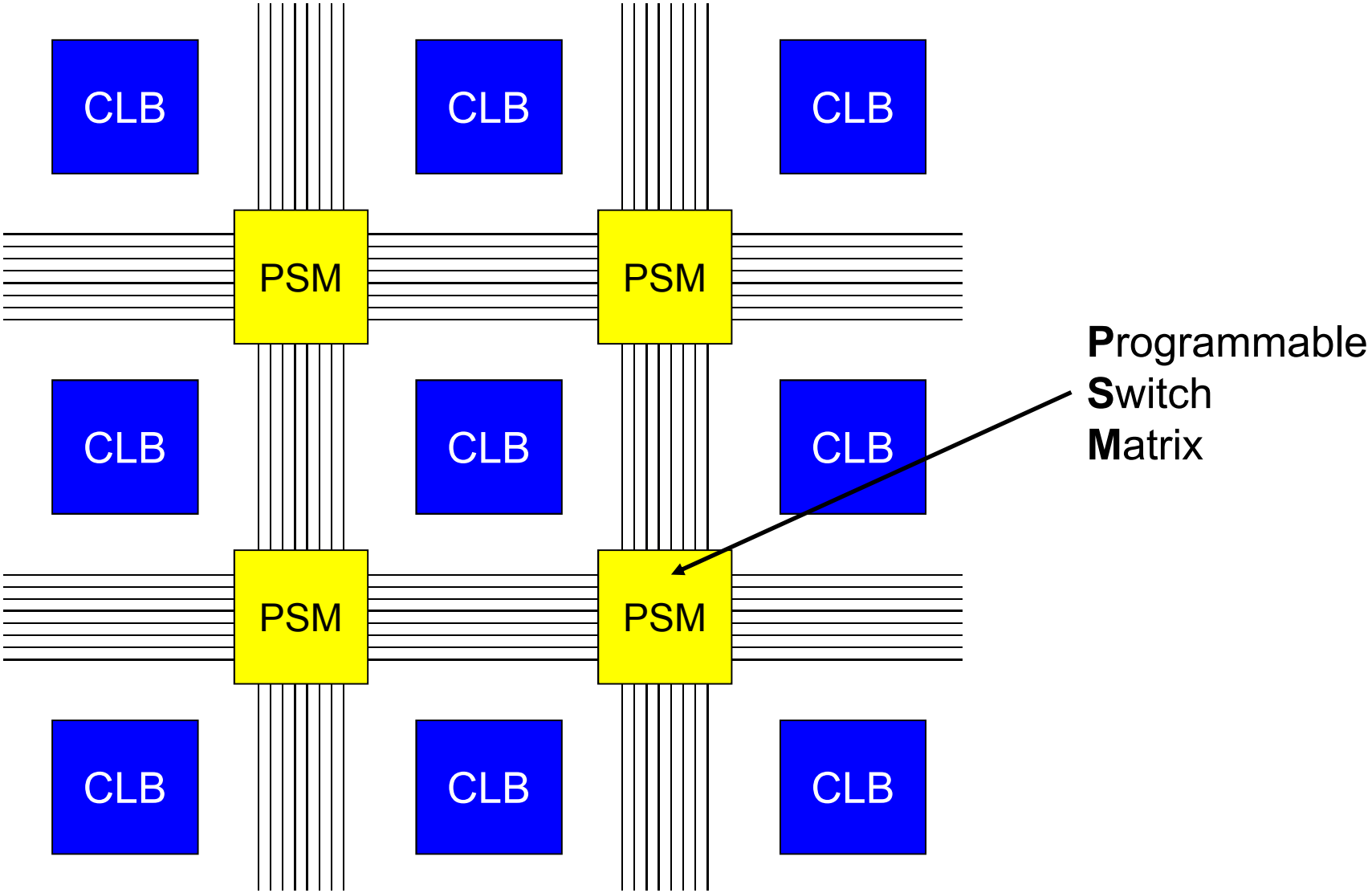
Structura de baza a unui bloc I/O



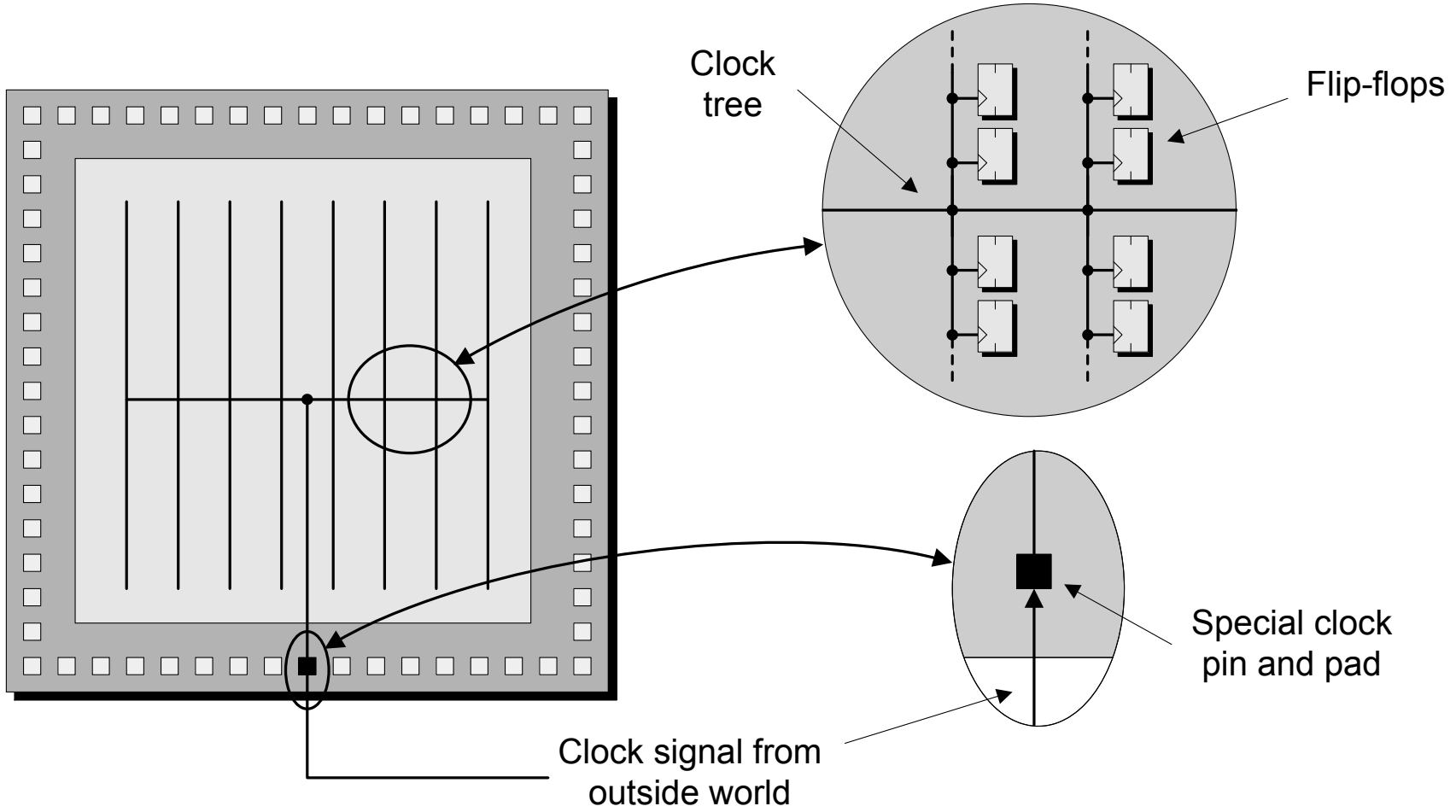
Funtionalitatea unui bloc I/O

- Blocurile I/O (IOB) permit interconectarea pinilor circuitului la structura interna de blocuri CLB
- Fiecare IOB poate sa functioneze ca un port uni sau bidirectional
- Iesirile pot fi fortate in starea de impedanta marita
- Intrarile si iesirile pot fi trecute printr-un buffer de tip registru
- Intrarile pot fi amanate

Resurse de rutare

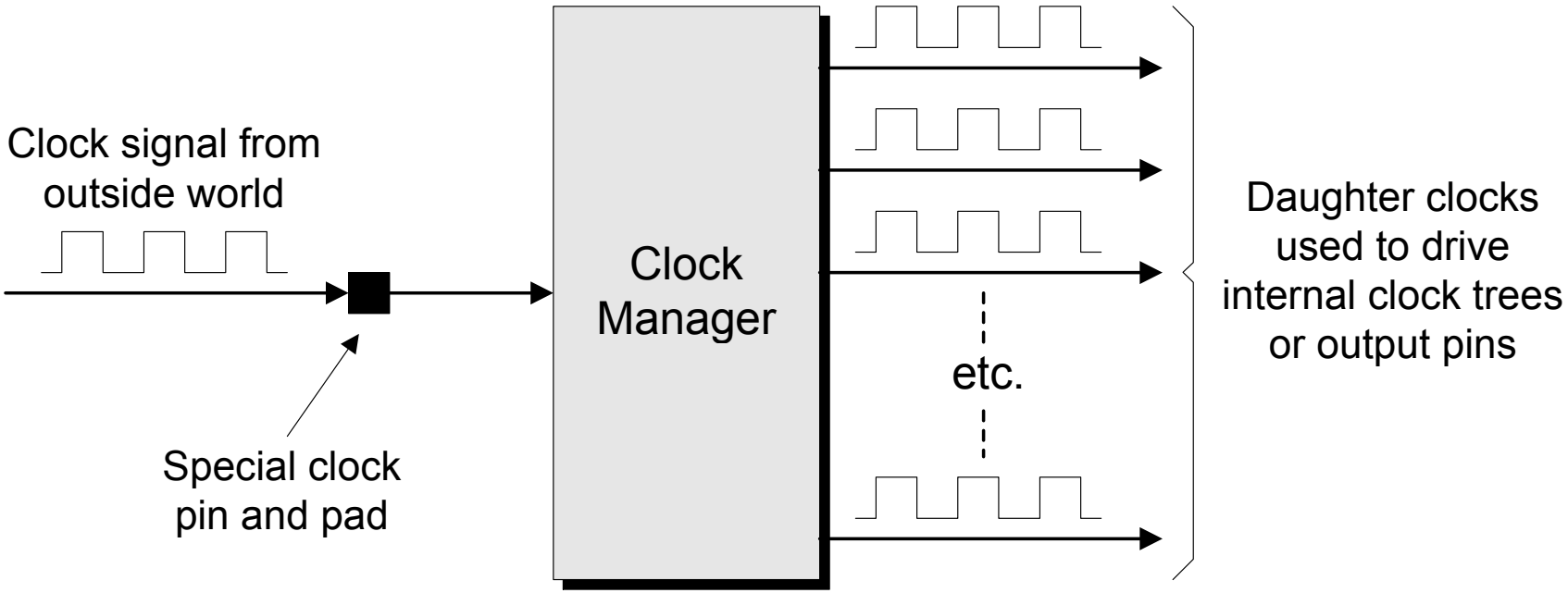


Clock Tree



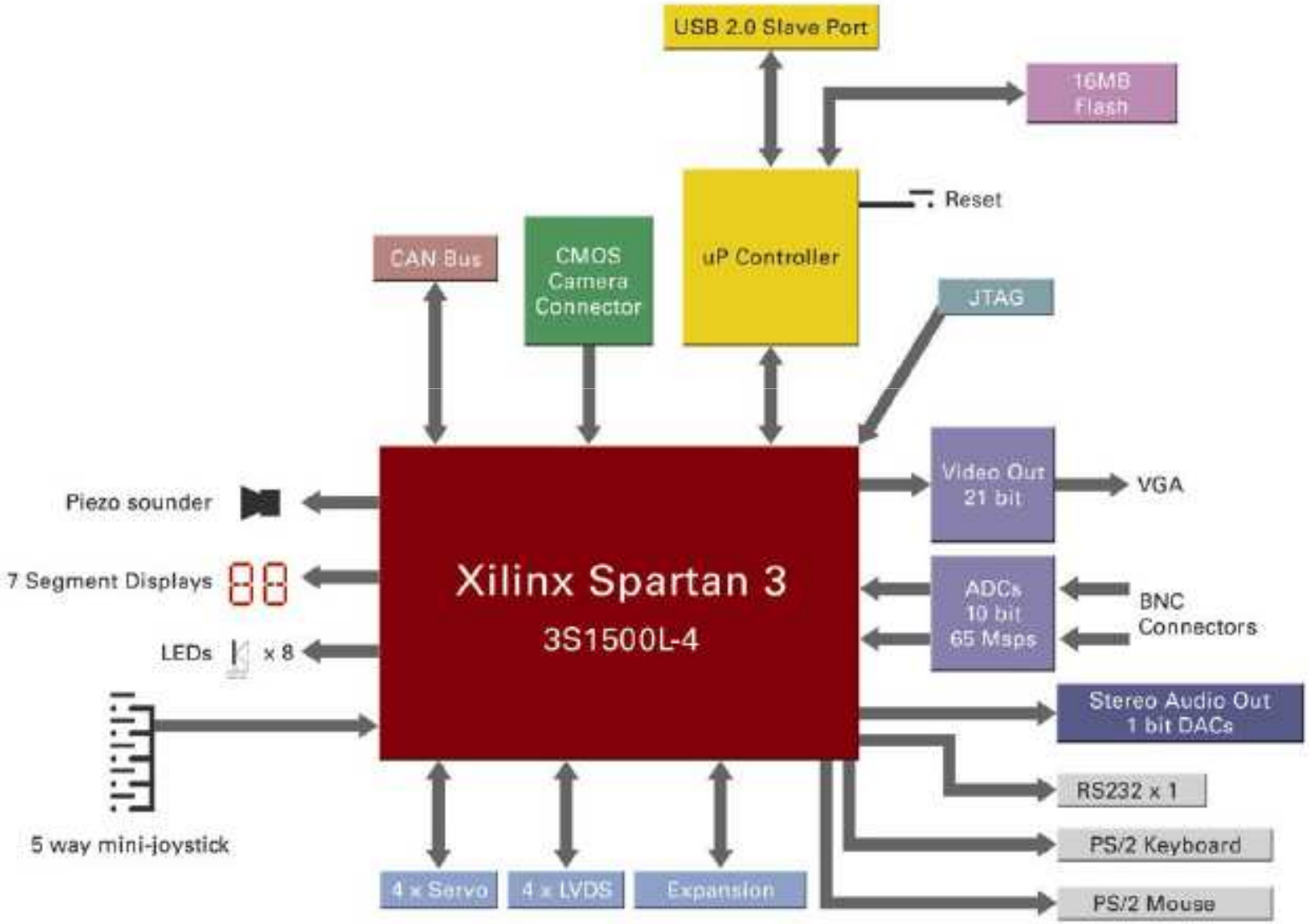
The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)

Digital Clock Manager (DCM)



The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)

Exemplu de sistem cu FPGA



FPGA Design Flow

Design flow (1)

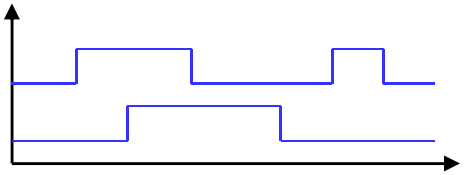
Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

Specificatii (descrierea functionalitatii)

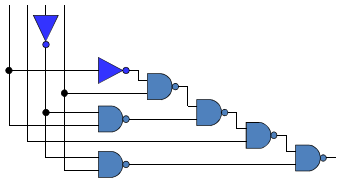
```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
                clock, reset, encr_decr: in std_logic;
                data_input: in std_logic_vector(31 downto 0);
                data_output: out std_logic_vector(31 downto 0);
                out_full: in std_logic;
                key_input: in std_logic_vector(31 downto 0);
                key_read: out std_logic;
        );
end AES_core;
```

Descriere VHDL (Fisiere sursa)

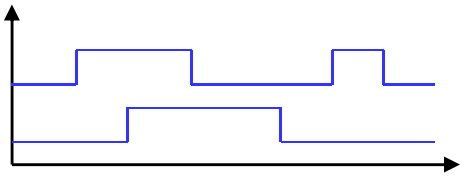
Simulare Functionala



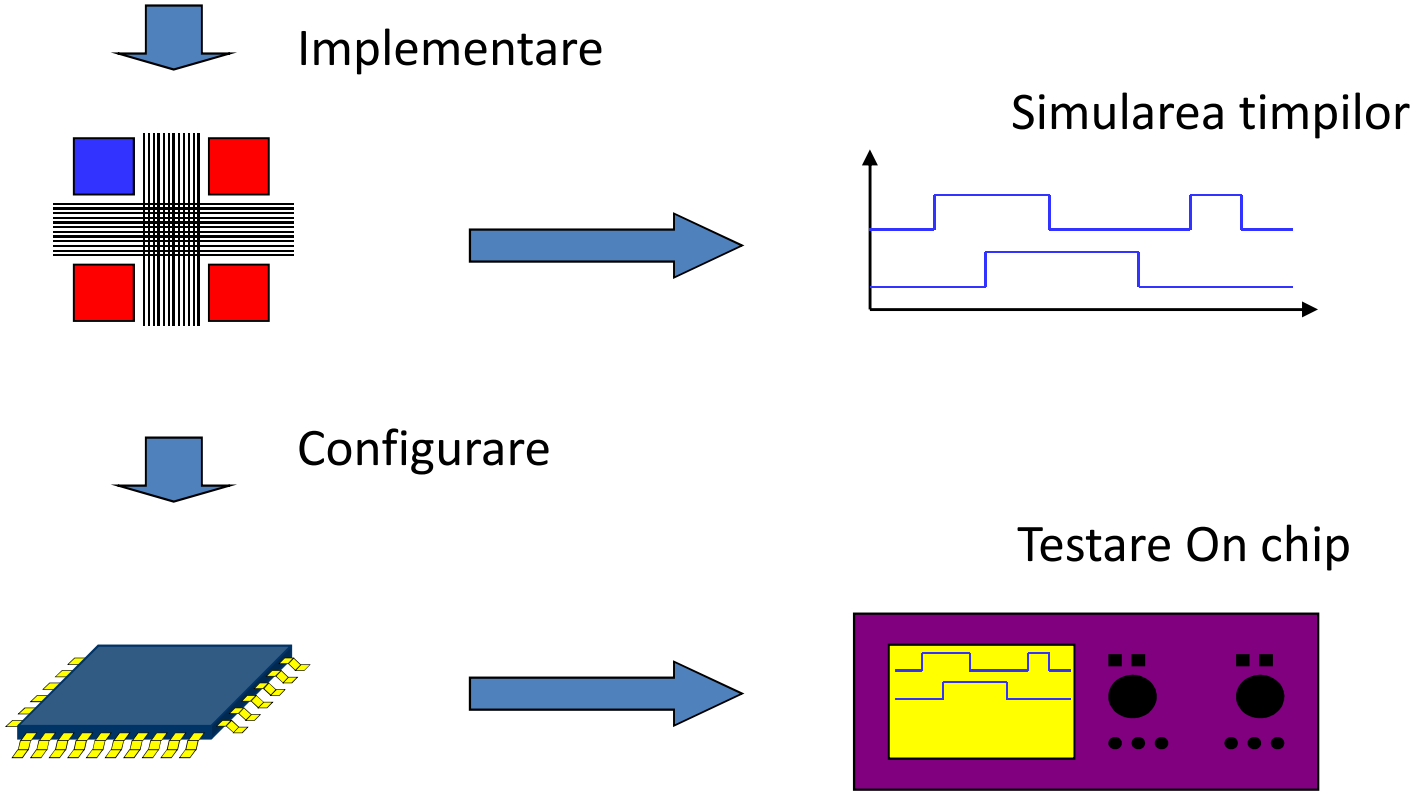
Sinteza



Simulare Post-synthesis



Design flow (2)



Sinteza

Tool-uri de Sinteza

Synplify Pro

Xilinx XST

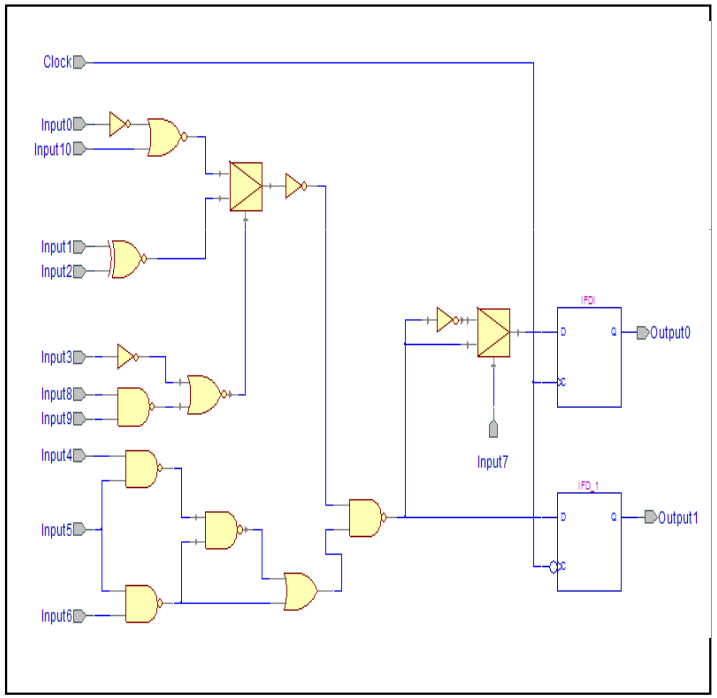
… si altele

Sinteza Logica

Descriere VHDL

```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
                A1<=A when (NEG_A='0') else
                                not A;
                B1<=B when (NEG_B='0') else
                                not B;
                Y<=Y1 when (NEG_Y='0') else
                                not Y1;

                MUX_0<=A1 and B1;
                MUX_1<=A1 or B1;
                MUX_2<=A1 xor B1;
                MUX_3<=A1 xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0 when "00",
                                                MUX_1 when "01",
                                                MUX_2 when "10",
                                                MUX_3 when others;

end MLU_DATAFLOW;
```

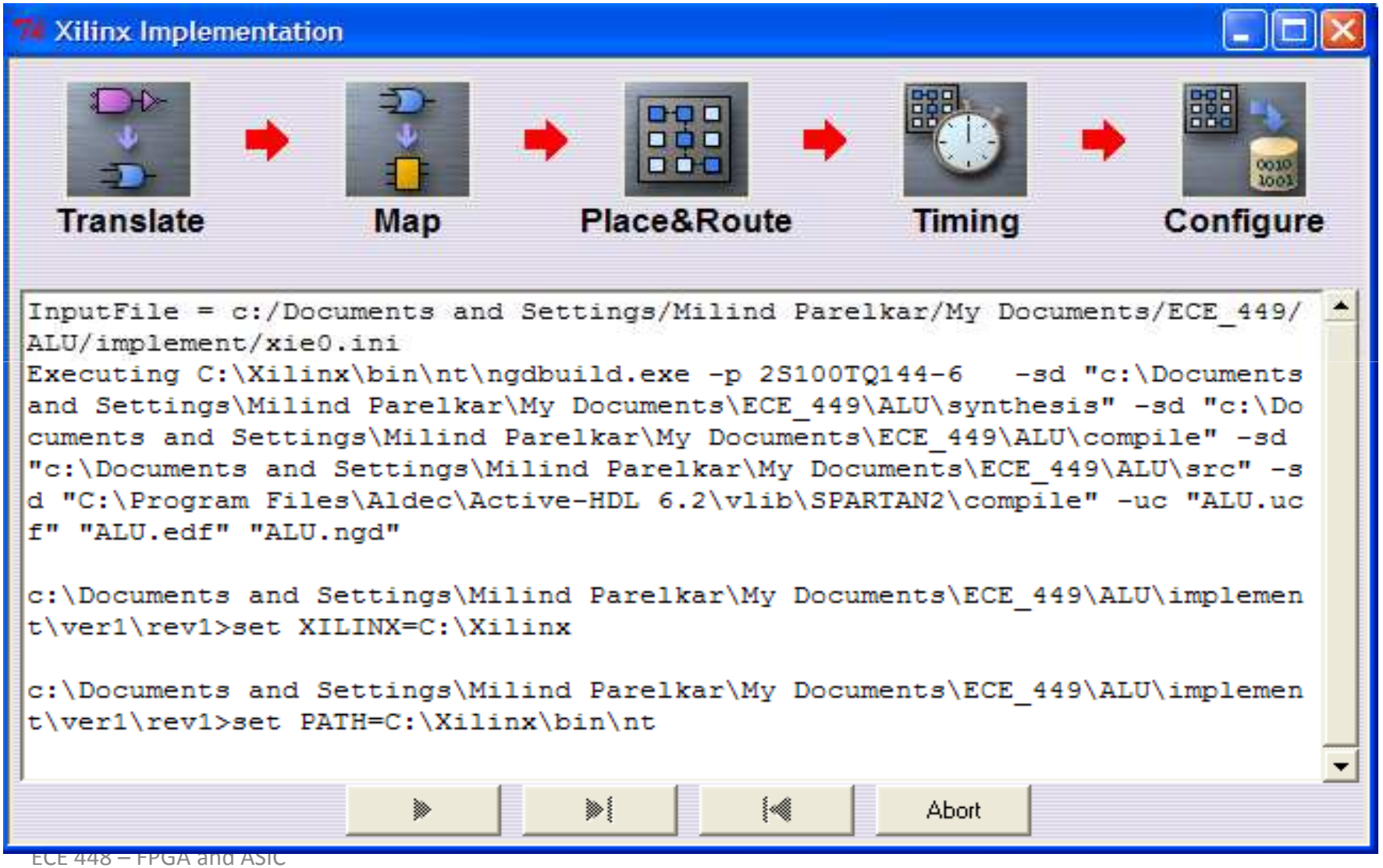
Netlist Circuit



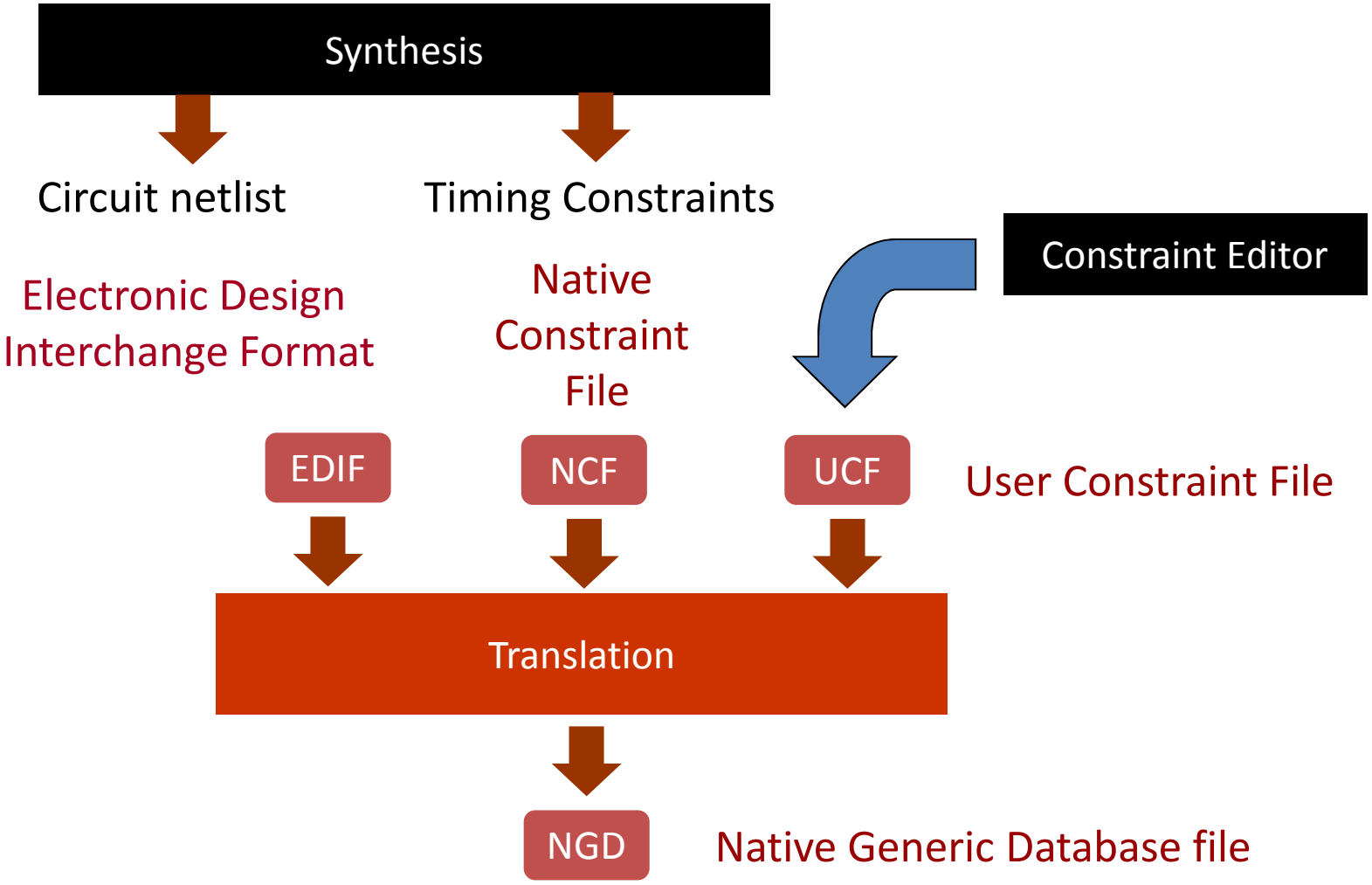
Implementare

Implementare

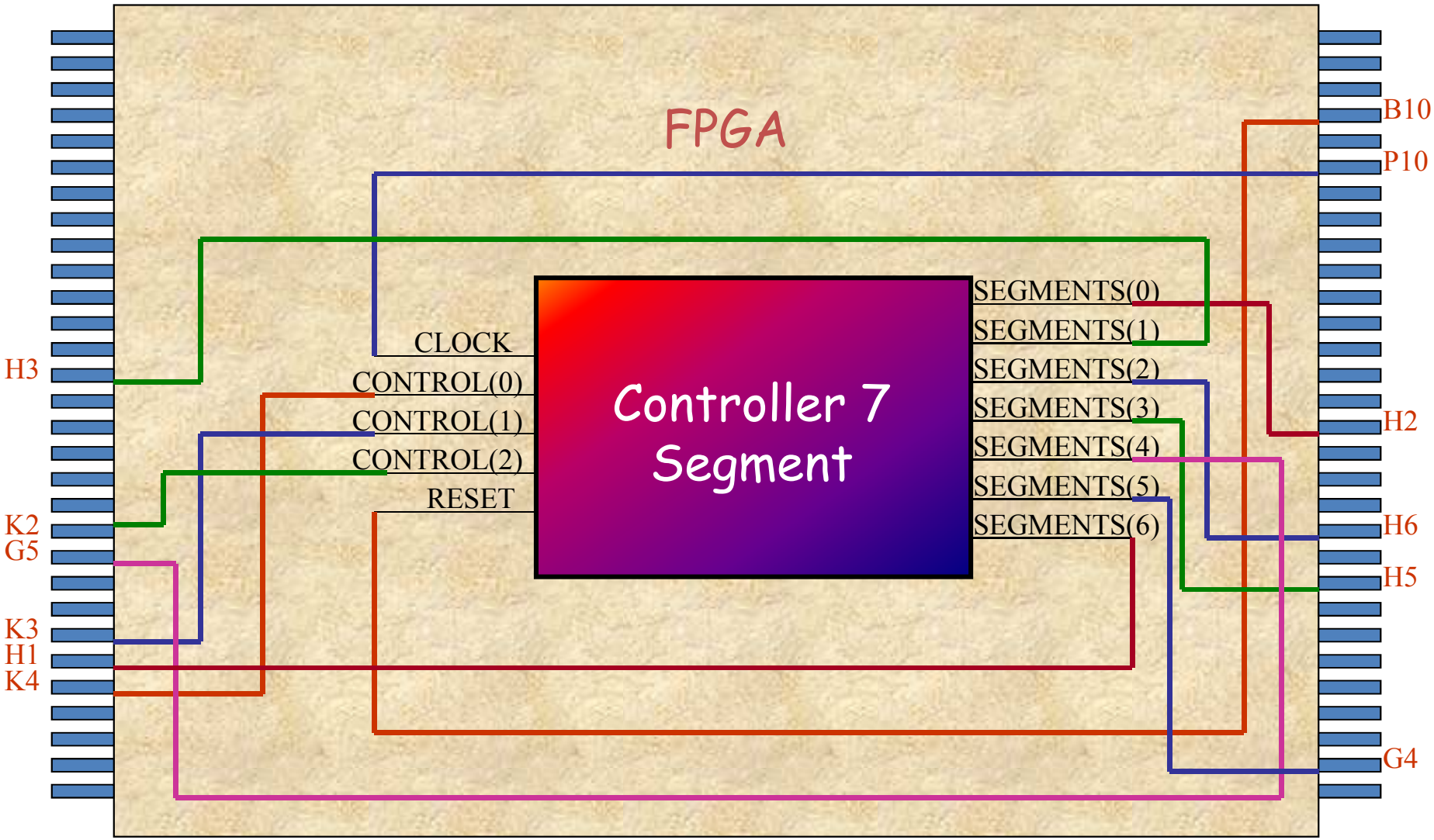
- Dupa siteza intregul proces de implementare este realizat de tool-uri FPGA proprietare


Xilinx Implementation
Translate
Map
Place&Route
Timing
Configure
InputFile = c:/Documents and Settings/Milind Parelkar/My Documents/ECE_449/ALU/implement/xie0.ini
Executing C:\Xilinx\bin\nt\ngdbuild.exe -p 2S100TQ144-6 -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\synthesis" -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\compile" -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\src" -sd "C:\Program Files\Aldec\Active-HDL 6.2\vlib\SPARTAN2\compile" -uc "ALU.ucf" "ALU.edf" "ALU.ngd"
c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\implement\ver1\rev1>set XILINX=C:\Xilinx
c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\implement\ver1\rev1>set PATH=C:\Xilinx\bin\nt
Abort

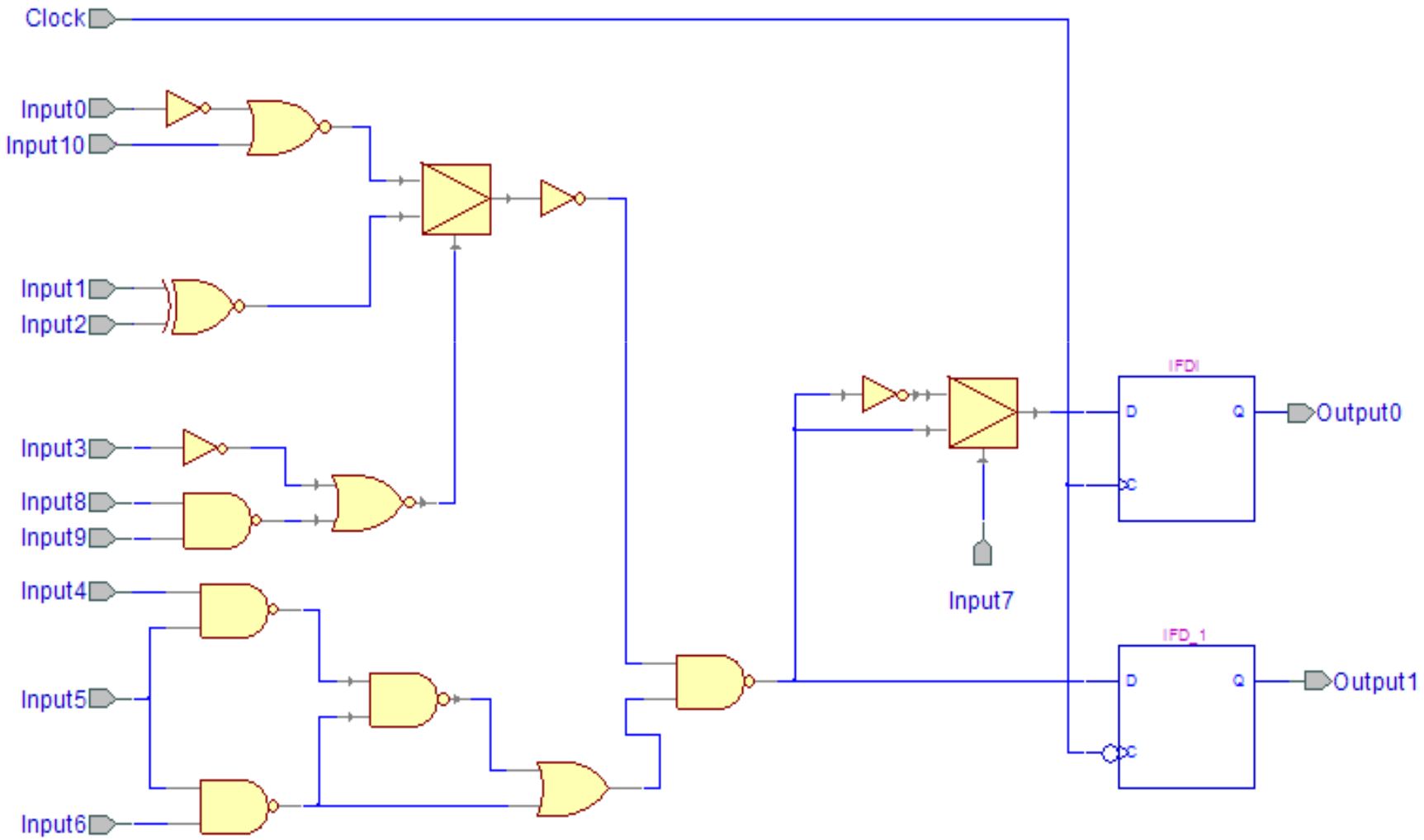
Translatie


Synthesis
Circuit netlist
Timing Constraints
Constraint Editor
Electronic Design Interchange Format
Native Constraint File
EDIF
NCF
UCF
User Constraint File
Translation
NGD
Native Generic Database file

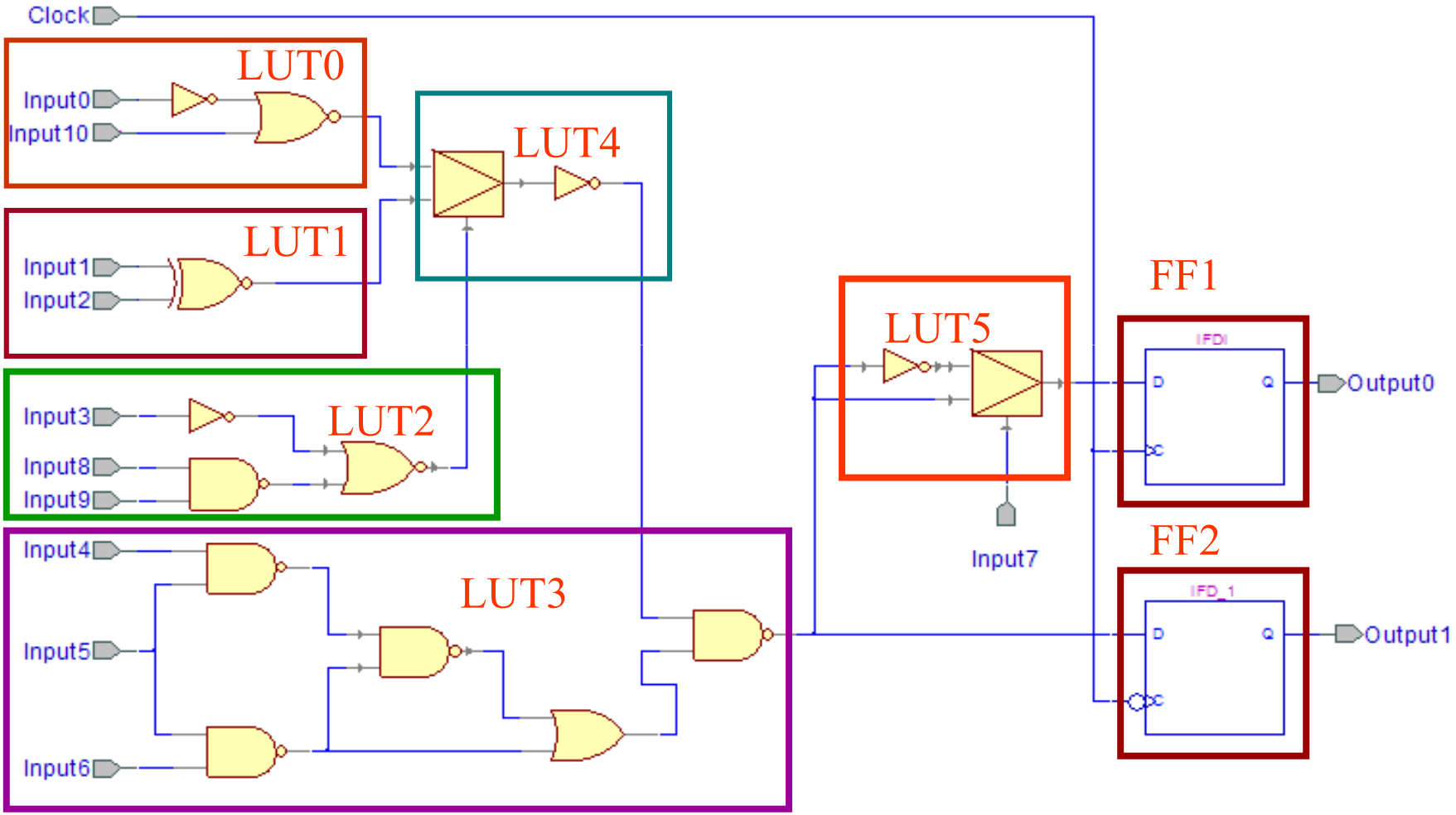
Asocierea Pinilor



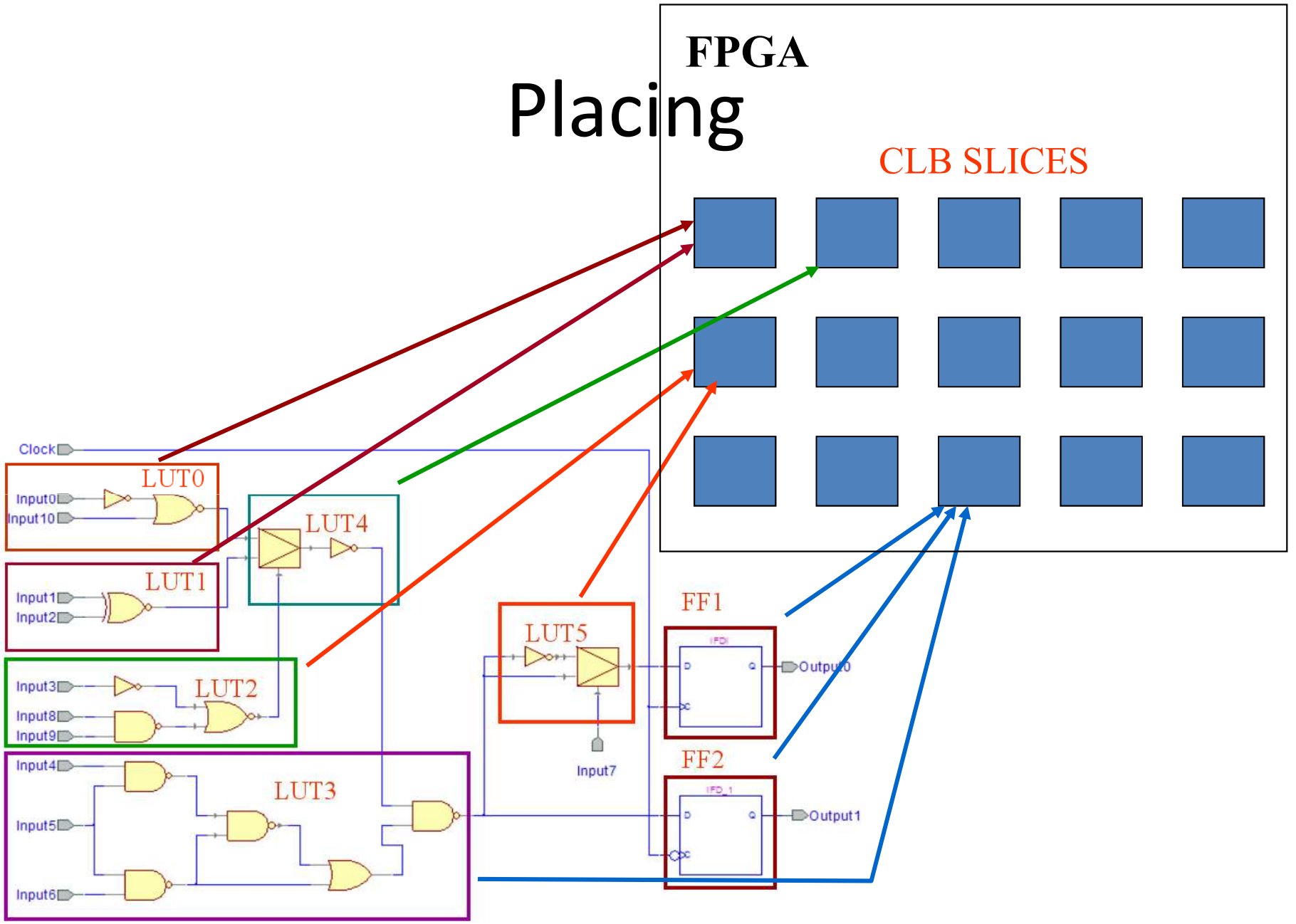
Netlist Circuit



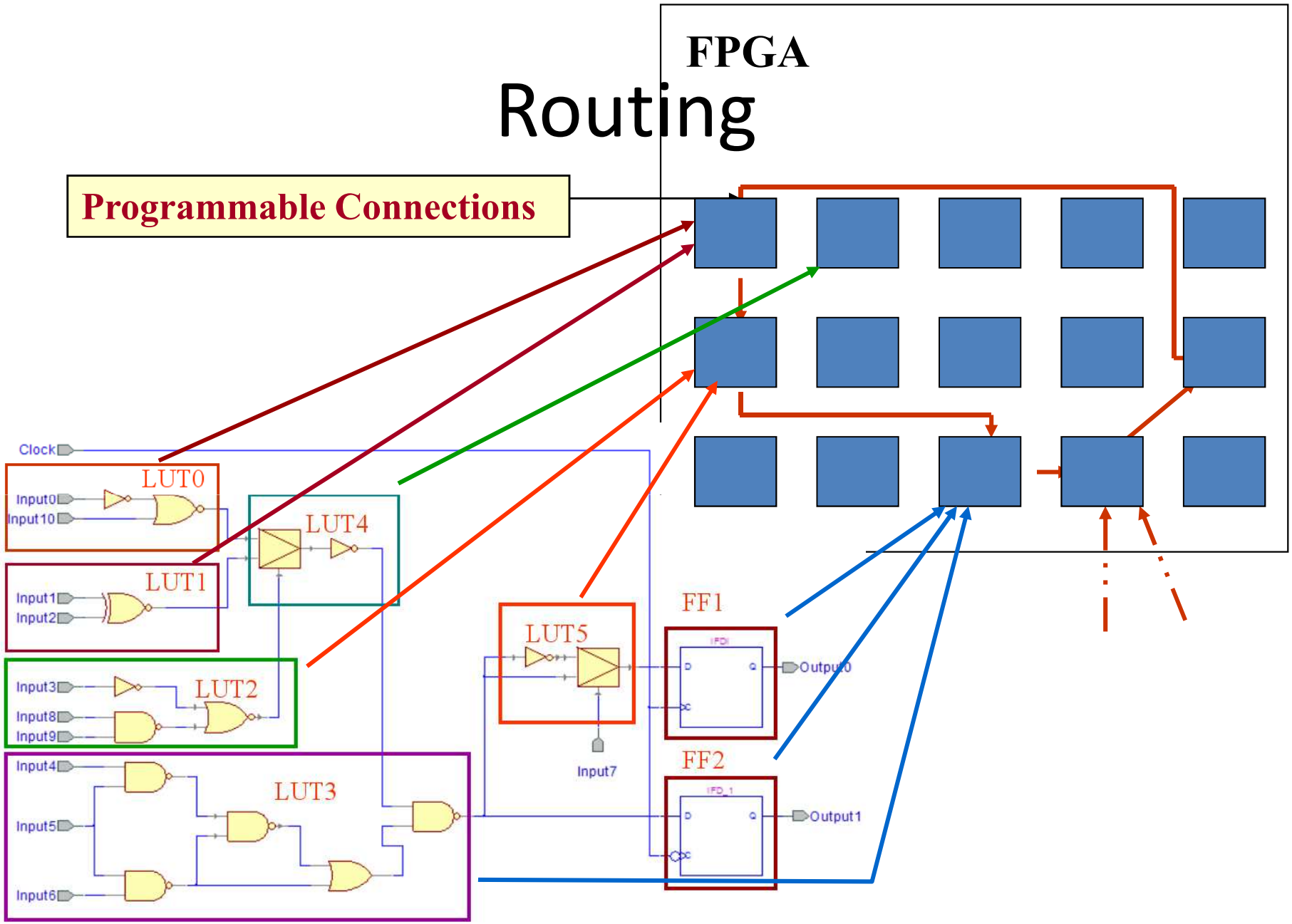
Mapare


Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
IFDI
D
Q
C
Output0
FF2
IFD_1
D
Q
C
Output1

Placing


FPGA
CLB SLICES
Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT4
Input3
Input8
Input9
LUT2
Input4
Input5
Input6
LUT3
LUT5
Input7
FF1
FF2
Output0
Output1

Routing


FPGA
Programmable Connections
Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT4
LUT5
Input7
FF1
FF2
Output0
Output1

Configurare

- Odata ce design-ul este implementat, trebuie creat un fisier pe care FPGA-ul poate sa-l inteleaga.
  - Acest fisier este numit bit stream: BIT file (extensia .bit)

- Fisierul BIT poate fi descarcat direct in FPGA sau poate fi transformat intr-un fisier PROM care stocheaza informatiile despre programare.