

# Proiectare logică

## Curs 14

Instrumente software de implementare cu  
dispozitive logice programabile

Cristian Vancea

<https://users.utcluj.ro/~vcristian/PL.html>

# Cuprins

- Instrumente software de implementare cu dispozitive logice programabile
  - Translatare
  - Verificare
  - Simulare
  - Programare
  - Procesul de proiectare cu instrumente software
- Dispozitive logice programabile de tip FPGA

# Instrumente software de implementare cu dispozitive logice programabile

## Introducere

Dispozitive logice programabile = colecție de celule logice de bază plasate într-o rețea de interconectare => nevoie de traducare (compilare/conversie) de la schema proiectului la implementare cu celule.

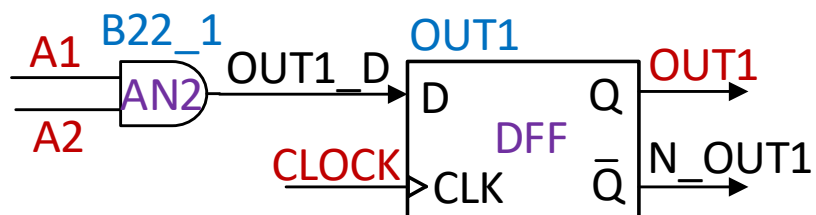
Instrumentele software realizează următoarele funcții:

- **traducare** – conversie de la schemă logică la funcțiile realizabile de celule;
- **verificare** – se verifică gradul de corectitudine obținut după traducare.

Suportul pentru aceste funcționalități are la bază reprezentarea proiectului sub formă de listă de conexiuni și unități utilizate – *netlist* (format text).

Ex:

Schema



Fișier *netlist*

```
NETSTART
B22_1 AN2 I(A1, A2) O(OUT1_D)
OUT1 DFF I(OUT1_D, CLOCK) O(OUT1, N_OUT1)
NETEND
NETIN A1, A2, CLOCK
NETOUT OUT1
```

# Instrumente software de implementare cu dispozitive logice programabile

## Traducere

Acțiuni întreprinse în cadrul procesului de traducere:

- Se verifică dacă numărul pinilor și al celulelor din dispozitiv este suficient pentru necesitățile proiectului.
- Se grupează componentele cu număr de intrări/ieșiri egal cu cel al celulelor de bază. Pentru celelalte componente cu număr mare de terminale se alocă mai multe celule/componentă.
- Se urmează un set de **reguli de conversie** și o strategie de **optimizare** a conversiei.

Etapele procesului de traducere:

1. maparea tehnologică;
2. plasarea;
3. rutarea.

# Instrumente software de implementare cu dispozitive logice programabile

## Traducere: Maparea tehnologică

- Reprezintă conversia componentelor logice ale proiectului în componente implementabile cu celulele dispozitivului logic programabil.
- Conversia se realizează concomitent cu **optimizarea *netlist*-ei** care constă în acțiuni pe mai multe direcții:
  - Se utilizează metode clasice de minimizare cu Diagrame Karnaugh, algoritmul Quinne McCluskey și teoremele algebrei booleene;
  - Se elimină componentele care nu sunt necesare (ex: un registru/numărător poate fi redus la numărul minim de biți utilizați/necesari în proiect);
  - Se elimină redundanța prin comasarea componentelor cu funcții identice.

# Instrumente software de implementare cu dispozitive logice programabile

## **Traducere: Plasarea**

- Reprezintă procesul găsire a unei soluții de amplasare fizică a componentelor logice rezultate din maparea tehnologică în celulele logice de bază.
  - Pentru îmbunătățirea timpilor de propagare a semnalelor în dispozitiv se urmărește amplasarea componentelor adiacente (ieșirile unei componente sunt intrările celeilalte) în celule vecine.
  - Se are în vedere găsirea unei soluții de amplasare care să permită conectarea tuturor celulelor implicate în proiect folosind rețeaua de interconectare disponibilă.

# Instrumente software de implementare cu dispozitive logice programabile

## Traducere: Rutarea

- Reprezintă procesul de interconectare a celulelor alocate proiectului după plasare.
  - Se analizează conexiunile ce apar în *netlist* și celulele alocate în pasul de plasare și se alocă treptat liniile de interconectare disponibile. Dacă o linie are la un capăt un semnal de ieșire și la celălalt capăt unul de intrare se marchează ca **linie ocupată**.
  - Dacă la un moment dat apare *congestia* (nu mai sunt linii disponibile) se reia atât amplasarea cât și rutarea -> procesul se numește **re-rutare**.
- **Probleme de temporizare:**
  - **Definiție:** **căile critice** sunt conexiuni ce trebuie favorizate în ceea ce privește timpul de propagare a semnalului prin ele (ex: semnalul de tact folosește căi critice ca să ajungă deodată la bistabilele registrelor și numărătoarelor).
  - În dispozitivele logice programabile există un număr limitat de celule dedicate cu acces la linii de interconectare de mare viteză.
  - Plasarea și rutarea încep întotdeauna cu celulele de pe căile critice.
  - Căile critice impun constrângeri de plasare și rutare și îngreunează găsirea unei soluții adecvate => se recomandă reducerea pe cât posibil a căilor critice.
- **Rezultatul:** un fișier care conține poziția celulelor alocate proiectului și interconexiunile între celule.

# Instrumente software de implementare cu dispozitive logice programabile

## Verificare

- Se realizează de către programe specializate în timpul translatării și/sau în procesele de simulare.
- Se analizează informația din *netlist* și proprietățile proiectului final rezultat.
- Exemple de verificări:
  - Se verifică numărul de intrări la care se conectează fiecare ieșire a unei celule; dacă numărul e prea mare dăunează d.p.d.v. electric. Soluție: se clonează celula cu mai multe celule identice, iar pe ieșirile lor se redistribuie în mod egal setul inițial de destinații.
  - Se elimină intrările lăsate neconectate pentru a evita zgomotele electrice.
  - Se identifică dacă sunt ieșiri conectate direct fără a utiliza buffer-e 3-state și se semnalează.



# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

- Se verifică funcționalitatea proiectului la nivel logic și performanțele temporale.
- Metodologia de lucru:
  - Se creează un model logic al circuitului, se aplică un set de valori pe intrări (stimuli) și se analizează corectitudinea valorilor obținute pe ieșiri.
  - Pe parcursul simulării se poate analiza propagarea semnalelor prin elementele interne ale circuitului.
- Tipuri de simulare:
  - funcțională;
  - temporală digitală;
  - a defectelor.

# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

### Simularea funcțională

- Se modelează funcționalitatea celulelor logice utilizate, se aplică stimuli și se generează setul de ieșiri rezultat.
- Modelul este simplu și generează rapid rezultatele – **avantaj**.
- Modelul nu oferă informații de temporizare – **dezavantaj**.

### Simularea temporală digitală

- La modelul celulelor utilizate se adaugă *blocuri de întârziere* pe toate ieșirile, care modelează întârzierea acumulată având în vedere 3 factori:
  - întârzierea introdusă de celulă la tranziția ieșirii din 1 în 0 sau din 0 în 1;
  - întârzierea introdusă de firele care conectează ieșirea cu alte intrări;
  - întârzierea datorată impedanței acumulate de la intrările comandate de ieșire.
- Detaliile despre întârzieri sunt cunoscute abia după plasare și rutare când se vor ști celulele și conexiunile utilizate. Întârzierile se adaugă la descrierea din fișierul *netlist*. Actualizarea *netlist* în acest sens se numește **back annotation** și are loc după fiecare iterație de plasare și rutare.
- Modelul este complex și oferă informații precise de temporizare – **avantaj**.

# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

### Simularea defectelor

- Sunt utilizate tehnici speciale prin care se verifică fiecare aspect al proiectului conform standardelor industriale.
- Se aplică stimuli și se simulează defecte artificiale în timp ce se urmărește comportamentul circuitului comparând valorile de pe ieșirile sale cu cele așteptate în lipsa defectelor.
- Ex: **simularea fundamentală** = proces prin care în timp ce se aplică stimuli pe intrări, ieșirile unor celule sunt menținute forțat pe 0 sau 1 și se analizează efectul asupra rezultatelor.
- Simularea defectelor necesită rulări multiple fiind o mare consumatoare de timp.
- La final se obține un *scor* numit **grad de defectare**.

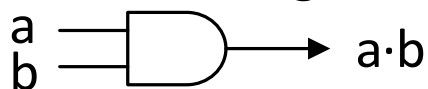
# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

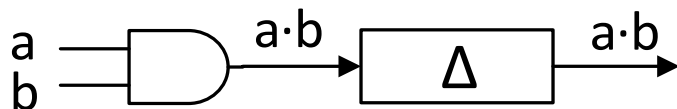
### Evaluarea semnalelor în timpul simulării

- Simulatorul are la bază o bibliotecă de funcții care evaluează intrările și ieșirile după un set extins de valori, ceea ce conferă o estimare mai precisă a comportamentului circuitului la nivel funcțional.
- În timpul simulării un semnal poate avea mai multe valori. Ex: 0 (logic), 1 (logic), X (nedeterminat), Z (întăită impedanță la 3-state), U (neinițializat).
- Evaluarea ieșirii din tabelul de adevăr al unei porți logice se realizează pentru toate combinațiile posibile pe intrări cu sau fără întârziere.
- Ex: Poarta ȘI are următorul tabel de adevăr cu 25 valori și 2 modele de simulare:

a) Modelul logic



b) Modelul cu întârziere adăugată



bloc de întârziere

ȘI	0	1	X	Z	U
0	0	0	0	0	0
1	0	1	X	X	U
X	0	X	X	X	U
Z	0	X	X	X	U
U	0	U	U	U	U

# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

### Modelarea

- a) Pentru o evaluare eficientă în simulare se utilizează **modele comportamentale** (*behavioural language models*) care facilitează descrierea comportamentală periferică (pe intrări/ieșiri) a circuitului testat.
- Se modelează ieșirile corecte pentru un set de stimuli de intrare.
  - Se utilizează o descriere generică, fără detalii de implementare => **portabilitate**.
  - Evaluarea este rapidă => se pretează la arhitecturi complexe.
  - Ex: Descrierea unui bistabil D flip-flop în limbajul VHDL

```
process(CLK)
begin
    if rising_edge(CLK) then
        Q <= D;
    end if;
end process;
```

Obs: Nu apare nicio referire la vreun tip de circuit sau poartă logică.

verificare front ascendent



# Instrumente software de implementare cu dispozitive logice programabile

## Simulare

### Modelarea

- b) Se poate modela comportamentul circuitului prin definirea elementelor sale constituate.
- Ex: Bistabilul D flip-flop se poate descrie cu porți ȘI-NU cu reacție inversă.
  - Oferă o vedere completă a tuturor operațiilor ce au loc în cadrul modelului.
  - Permite o evaluare completă a gradului de defecte.
  - Evaluarea este lentă => se pretează la arhitecturi simple.

Obs: Simulatorul vine cu **biblioteci** care încapsulează atât modele de simulare pentru un set larg de componente uzuale (porți logice, bistabile, seturi de intrări/ieșiri generice, etc.) cât și alte funcții adiționale utile (ex: funcții de aplicare a stimulilor, de evaluare a ieșirilor). Timpii de întârziere predefiniți se pot ajusta.

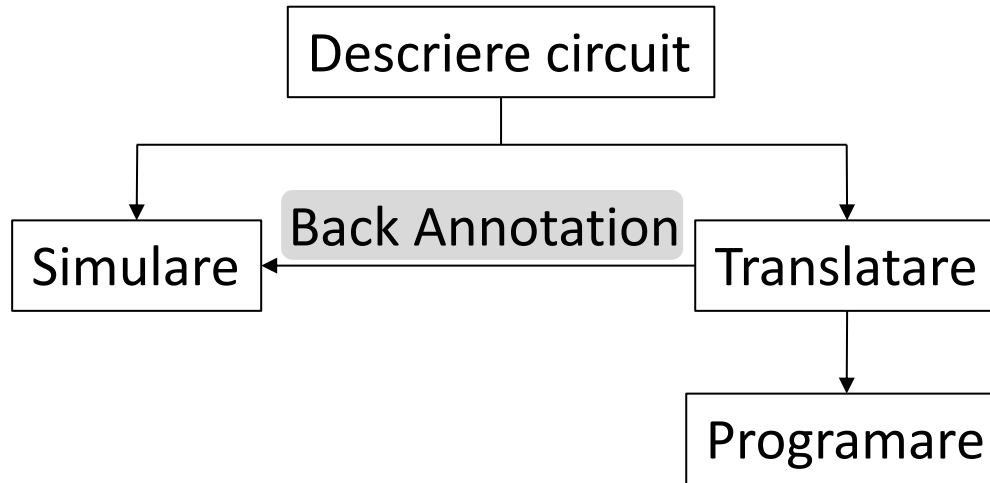
# Instrumente software de implementare cu dispozitive logice programabile

## Programare

- Fișierul obținut după rutare este convertit în format binar de configurare a celulelor și memoriilor alocate proiectului în interiorul dispozitivului logic programabil.
- Etape ale procesului de programare:
  1. Inițializare:
    - Se aplică alimentarea, se activează modul de programare a dispozitivului.
  2. Ștergere memorie de configurare.
  3. Încărcare date de configurare:
    - Datele de configurare specifică celulele interne utilizate, funcțiile pe care le implementează și legăturile dintre ele;
    - În funcție de tipul de dispozitiv utilizat fișierul de configurare poate avea dimensiuni diferite (50KB - 4GB);
    - Unele dispozitive au posibilitatea de decriptare și acceptă date de configurare criptate.
  4. Pornire (start-up)
    - Se verifică dacă datele de configurare sunt corecte.
    - Se dezactivează modul de programare.
    - Se activează intrările/ieșirile dispozitivului, bistabilele și memoriile.

# Instrumente software de implementare cu dispozitive logice programabile

## Procesul de proiectare cu instrumente software



1. Se descrie circuitul prin schemă sau limbaj de descriere hardware => fișierul *netlist*.
2. Se verifică funcționalitatea printr-o simulare funcțională pe baza fișierului *netlist*.
3. Se realizează traducerea. Se extrag întârzierile și se adaugă la *netlist* (back annotation).
4. Se realizează simularea temporală digitală. Dacă se satisfac toate condițiile de funcționare se poate programa circuitul în dispozitiv, altfel se fac ajustările necesare și se reia traducerea și simularea.

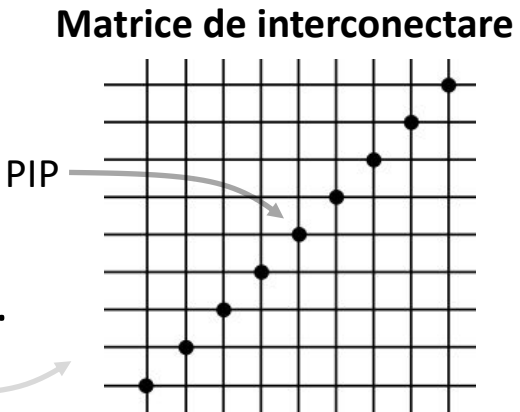
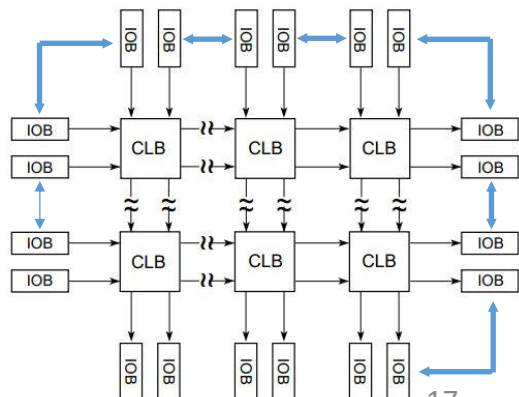


# Dispozitive logice programabile de tip FPGA

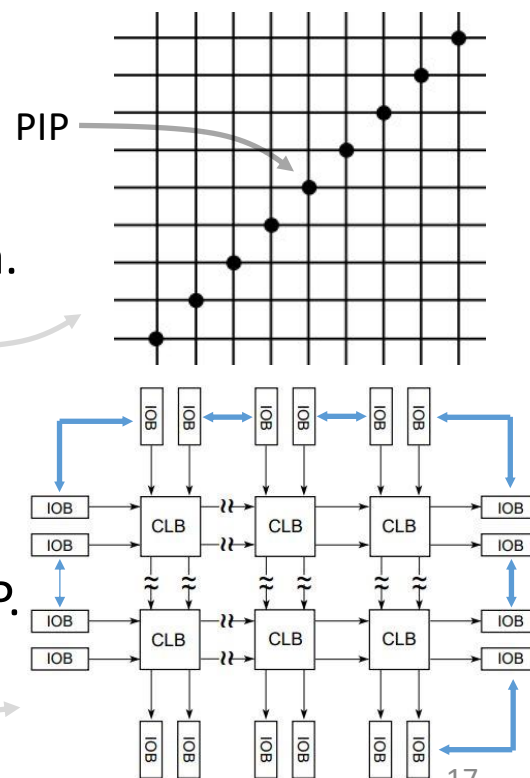
## Avantaje

- Dimensiuni reduse.
- Consum redus de putere.
- Creșterea vitezei, accelerare, scalabilitate, paralelizare.
- Timp redus de (re)configurare.
- Dezvoltarea de arhitecturi eterogene hardware/software.

## Familia Xilinx 4000

- Celula logică de bază - Configurable Logic Block (CLB): 3 generatoare de funcții logice combinaționale, 2 bistabile D flip-flop (pot fi latch), căi de multiplexare și conectare locală.
- Căile de interconectare sunt amplasate într-o matrice cu puncte de conexiuni programabile (Programmable Interconnection Point(s) - PIP). 
- Programarea conexiunilor se realizează prin memorii de interconectare SRAM în care fiecare bit este asociat unui PIP.
- Blocurile de intrare/ieșire (IOB) sunt interconectate într-un inel în jurul matricei de CLB. 

Matrice de interconectare



# Dispozitive logice programabile de tip FPGA

## **Familia Xilinx Spartan**

- Apar mai multe variante de FPGA-uri: Spartan-II, Spartan-IIE, Spartan-3, Spartan-3A, Spartan-3E, Spartan-6.
- Porți logice: 40 000 - 1 400 000.
- Viteză: 80 – 450 MHz.
- Conexiune: PCI.
- În CLB funcția logică se realizează cu tabelă de adevăr / Look-Up Table (LUT) implementată în RAM.
- Conține blocuri RAM dedicate pentru stocare date în memorie.
- Conține circuite de configurare a semnalelor de tact sub formă de unități DCM (Digital Clock Management).

# Dispozitive logice programabile de tip FPGA

## Familia Xilinx Virtex

- Mai multe variante: Virtex 2, Virtex E, Virtex II, Virtex II Pro, Virtex 4, 5, 6, 7.
- Conține module de transfer de date de mare viteză de tip Multi Gigabit Transceiver (MGT) cu viteze de 3 sau 6 Gb/s.
- Înglobează un procesor IBM Power PC pe lângă logica programabilă.

## Virtex 7

- Arhitectura programabilă este organizată pe unități de bază numite *slice*-uri care conțin:
  - 4 unități LUT cu 6 intrări care se pot folosi cu rol de RAM de 64 biți sau registre de deplasare pe 32 biți;
  - 8 bistabile;
  - Slice-urile speciale de tip DSP conțin o unitate de multiplicare pentru operanzi pe 25x18 biți, un sumator și un acumulator pe 48 biți.
- Sunt integrate blocuri dedicate RAM de dimensiune 36Kb, MGT cu viteză de transfer până la 28 Gb/s și 2 convertoare analog-digital (AD) pe 12 biți.
- Prezintă unități de monitorizare a curentului și temperaturii interne.



# Dispozitive logice programabile de tip FPGA

## Comparații între familii mai recente de dispozitive FPGA marca Xilinx

	Spartan-6	Artix-7	Kintex-7	Virtex-7	Virtex UltraScale
Celule logice	147 443	215 360	477 760	1 954 560	4 407 480
BlockRAM	4.8Mb	13Mb	34Mb	68Mb	115Mb
Slice-uri DSP	180	740	1 920	3 600	2 880
DSP performance	140GMACs	930GMACs	2 845GMACs	5 335GMACs	4 268GMACs
Număr MGT	8	16	32	96	104
Viteză MGT	3.2Gb/s	6.6Gb/s	12.5Gb/s	28.05Gb/s	32.75Gb/s
Capacitate transfer MGT (full duplex)	50Gb/s	211Gb/s	800Gb/s	2 784Gb/s	5 101Gb/s
Interfață memorie (DDR3)	800Mb/s	1 066Mb/s	1 866Mb/s	1 866Mb/s	2 400Mb/s
PCI Express	x1 Gen1	x4 Gen2	x8 Gen2	x8 Gen3	x8 Gen3
Pini I/O	576	500	500	1 200	1 456
Tensiune I/O	1.2-3.3V	1.2-3.3V	1.2-3.3V	1.2-3.3V	1.0-3.3V

GMACs – Giga Multiply-Accumulate Operations per second  $\Leftrightarrow 10^9$  sume de produse / s

# Dispozitive logice programabile de tip FPGA

## Sisteme de procesare eterogene

- Integrează programabilitatea unui procesor cu programabilitatea unui FPGA.
- Se poate folosi accelerarea hardware alături de funcționalități specifice unui procesor.
- Există 2 direcții de dezvoltare:
  - System on Chip (SoC)
    - Integrează programarea pe procesor cu logica programabilă (FPGA).
    - Familia Zynq de la Xilinx integrează procesoare model ARM în structuri de tip FPGA de diverse modele. Comunicarea se realizează prin canale directe folosind protocoale specifice.
  - Adaptive Compute Acceleration Platform (ACAP) / adaptive SoC
    - Combină programarea pe procesor cu elemente de calcul vectorial (specific DSP, GPU) și logică programabilă (FPGA) într-un singur chip care poate fi folosit pentru o gamă largă de aplicații industriale.
    - Legătura între elementele componente se realizează printr-un nod de comunicare de mare viteză integrat în chip.