Sisteme cu dispozitive reconfigurabile

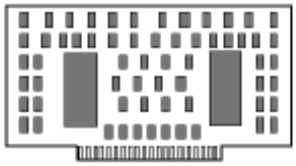
Arhitecturi de calcul

Dezvoltarea tehnologiei mijloacelor automate de calcul oferă mai multe soluții de implementare a sistemelor digitale:

1. Procesoare de uz general (GP), programate software;
2. Structuri hardware fixe (ASIC Aplication Specific Integrated Circuits) - circuite integrate specifice aplicației;
3. Circuite reconfigurabile (cele mai reprezentative fiind circuitele FPGA).

La începutul deceniului 9, al secolului trecut, cea mai mare parte a circuitelor logice, din sistemele numerice tipice, era realizată cu ajutorul unui număr relativ mic de circuite standard integrate pe scară largă (LSI): microprocesoare, controloare de magistrale, controloare de I/E, circuite de sincronizare etc.

Toate sistemele aveau încă nevoie de o logică "aleatoare", atât pentru interconectarea circuitelor integrate pe scară largă, cât şi pentru: generarea semnalelor de comandă globală (reset s.a.), formatarea datelor (serial/ paralel, paralel/serial, multiplexare) etc. Sistemele erau alcătuite dintr-un număr mic de componente LSI şi din numeroase componente integrate pe scară mică (SSI) şi medie (MSI).



Plachetă prevăzută cu componente LSI, MSI şi SSI.

Pentru realizarea structurii de interconectare adesea se proiectau circuite la cerere (custom, ASIC), care conduceau la:

- reducerea complexității sistemului şi a costurilor de fabricație, cât şi la mărirea performanței;
- costuri ridicate de dezvoltare a circuitelor la cerere, creşterea timpului de proiectare şi a timpului în care produsul ajungea pe piață.

În acest context apar două componente ale costurilor:

- costul de dezvoltare, denumit uneori: *non-recurring engineering* (NRE) şi
- costurile de fabricație.

În realizarea sistemelor numerice, abordarea cu circuite la cerere este viabilă pentru produsele realizate în număr foarte mare, la care costurile de dezvoltare se pot amortiza şi care nu sunt critice în raport cu timpul de lansare pe piață (TTM -Time To Market). FPGA-urile au fost introduse ca o alternativă la circuitele la cerere, pentru implementarea logicii de interconectare, ceea ce a permis mărirea densității de circuite de circa 10 ori în raport cu soluția SSI/MSI, reducerea costurile de dezvoltare şi scurtarea TTM. Cu ajutorul mijloacelor automate de proiectare (CAD), circuitele au putut fi implementate într-un timp foarte scurt, lipsind etapele de proiectare a măştilor şi fabricare a circuitelor.

În conformitate cu legea lui Moore, densitatea (porți/suprafață) FPGA-urilor a crescut între anii 80-90, ai secolului 20, până la punctul în care funcțiile importante de prelucrare a datelor au fost implementate direct într-un singur circuit FPGA. FPGA este în continuare în competiție cu circuitele la cerere pentru funcții speciale de prelucrare şi pentru logica de interconectare, dar este în competiție şi cu microprocesoarele în aplicații dedicate şi încorporate. FPGA-urile au avantajul performanței, în raport cu microprocesoarele, deoarece circuitele pot fi

adaptate uşor la aplicație. Microprocesoarele realizează funcțiile speciale în software, în condițiile operării în mai multe cicluri.

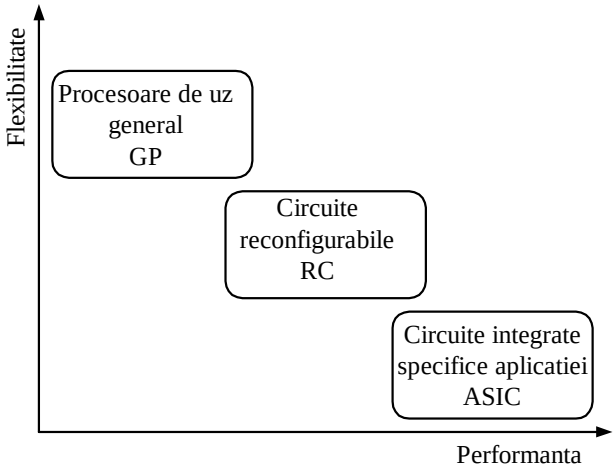
Comparație între soluțiile bazate pe ASIC, FPGA şi procesoare pentru implementarea unui sistem numeric, referitor la: performanță, NRE, cost/unitate şi TTM.

Performanță	NRE	Cost/unitate	TTM
ASIC	ASIC	FPGA	ASIC
FPGA	FPGA	Procesoare	FPGA
Procesoare	Procesoare	ASIC	Procesoare

Vom compara aceste arhitecturi şi din punctul de vedere al criteriilor de flexibilitate şi performanță.

Performanța determină capacitatea unui produs de a îndeplini o anumită sarcină. Există mai multe moduri de a măsura performanța unui produs, în general aceasta este exprimată în: MIPS (milioane de instrucțiuni per secundă), MMACS (milioane de multiplicări şi acumulări per secundă) sau mai simplu în MHz (milioane de perioade de tact per secundă).

Flexibilitatea se referă la posibilitatea de a schimba caracteristicile unui produs pentru a îndeplini cerințe noi. Implementările software au cel mai scurt timp de lansare pe piață, datorită marii flexibilități şi interfeței prietenoase a limbajelor de programare şi a compilatoarelor. Acesta este motivul principal pentru care ele sunt mai prezente pe piață decât implementările hardware.



Procesoarele programate prin software

- Procesoarele (microprocesoare, procesoare digitale de semnal DSPs) furnizează resursele logice necesare implementării unor platforme de calcul **variate**.
- Permit implementarea unor platforme de calcul foarte **flexibile**, capabile să execute aplicații din domenii diferite: comunicații, sisteme de comandă şi control, prelucrarea semnalelor, etc.
- Aplicațiile sunt executate prin decodarea unui flux de instrucțiuni preluate dintr-un cod software, pe baza informațiilor stocate în blocuri de memorie.
- Caracterul secvențial de executare a aplicațiilor, limitele impuse de viteza de acces la blocurile de memorie şi arhitectura fixă a procesoarelor, limitează performanța acestor sisteme.

Structurile hardware fixe (ASIC)

- ASIC-urile furnizează o soluție alternativă procesoarelor convenționale pentru implementarea unor aplicații performante.
- ASIC-urile sunt dispozitive hardware dedicate, proiectate pentru a implementa un număr foarte mic de aplicații sau chiar numai una singură.
- Pentru o aplicație dată, ASIC-urile obțin performanțe foarte bune, ocupă un spațiu mult mai mic şi consumă mai puțină energie decât procesoarele.
- În cazul unor producții de serie mari, prețul de cost unitar devine foarte bun.
- Durata mare a ciclului de dezvoltare a circuitului.
- Prețul ridicat al proiectării circuitului integrat (50 000$ - 1 000 000$)
- Sunt excluse optimizări post-design.
- Sunt puțin flexibile.

Circuitele reconfigurabile (FPGA)

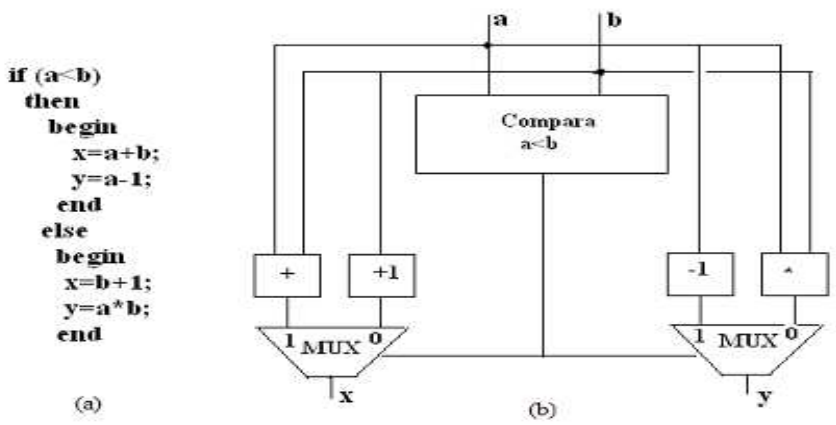
- Asigură un compromis între cele 2 tendințe: flexibilitate vs. performanță.
- Permit implementarea circuitelor specializate direct în hardware.
- Permit reconfigurarea, deoarece conțin resurse funcționale ce pot fi uşor modificate după implementarea în cadrul aplicației, ca urmare a schimbării parametrilor şi datelor de lucru.
- Risc scăzut în faza de proiectare.
- Cost inițial redus.
- Timp de proiectare redus.
- Permit optimizări post-design.

Modalități de efectuare a calculelor

Pornind de la posibilitățile de implementare a algoritmilor: prin software sau prin hardware (fix/reconfigurabil), se conturează două modalități de efectuare a calculelor:

1. **Temporală -** executarea secvențială a algoritmului în baza instrucțiunilor din program, care necesită etapele de extragere, decodare şi executare;
2. **Spațială** - executarea paralelă a algoritmului care se efectuează în baza unor blocuri funcționale care formează o structură hardware. Instrucțiunile, în acest caz, lipsesc. Distribuirea algoritmului nu doar în timp dar şi în spațiu permite atingerea unui grad înalt de paralelism.

Pentru exemplificare se presupune execuția următorului fragment de program pe un calculator convențioanal şi într-o structură ASIC/FPGA (fig.1.1.).



(a) soluția temporală/software. (b) soluția spațială.

Fragment de program pe un calculator vN (a) şi într-o structură ASIC (b).

Timpul total de execuție pe un calculator vN este: $3*t_{instrucțiune}$, care poate cuprinde mai multe cicluri de ceas. Structura ASIC/FPGA efectuează calculele într-un interval de timp egal cu întârzierea cea mai mare în propagarea semnalului de la intrare la ieşire.

Circuite PLD

Circuitele logice programabile, cunoscute şi sub forma acronimului PLD (Programmable Logic Device), sunt circuite integrate care conțin un număr mare de porți sau celule a căror interconexiune poate fi configurată sau "programată" pentru a implementa orice funcție combinațională sau secvențială dorită.

Pentru programarea circuitelor PLD se utilizează două tehnici:

- programarea prin măşti, care se efectuează în timpul procesului de fabricație;
- programarea de către utilizator, pentru care se utilizează echipamente de programare cu costuri reduse.

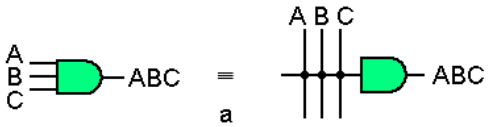
Multe circuite PLD pot fi reprogramate de utilizator de multe ori, motiv pentru care ele sunt avantajoase pentru realizarea prototipurilor unui nou produs. Conexiunile programabile între elementele logice ale unui circuit PLD conțin comutatoare realizate de obicei cu tranzistoare sau antifuzibile (uneori fuzibile).

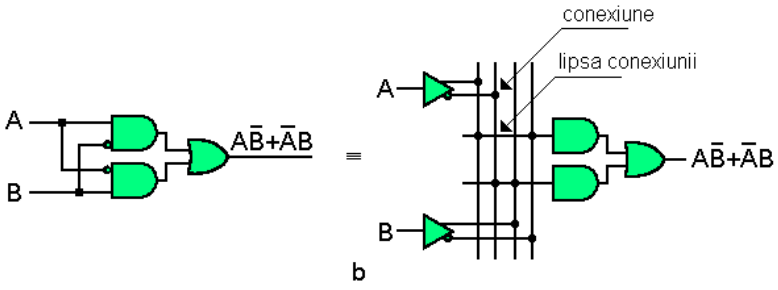
Firme producătoare de PLD: Xilinx, Altera, Lattice, Actel, Cypress, Atmel, QuickLogic.

Există mai multe tipuri de circuite care sunt denumite în mod generic circuite logice programabile (PLD).

SPLD (Simple Programmable Logic Device)
CPLD (Complex Programmable Logic Device)
FPGA (Field Programmable Gate Array)

Porțile logice programabile ale unui circuit PLD pot fi reprezentate în mod simplificat: în locul unor linii de intrare multiple la fiecare din aceste porți s-a figurat o singură linie. Semnul × indică o conexiune programabilă a unei linii de intrare la o poartă logică. Absența semnului × indică faptul că respectiva conexiune a fost programată în starea deconectată.



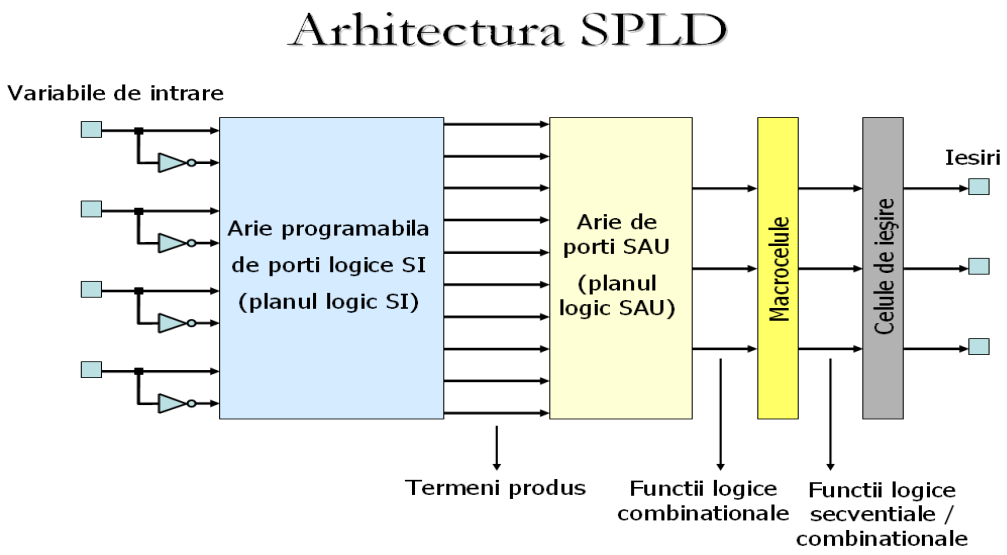


Circuite SPLD

Din punct de vedere arhitectural se clasifică în :

1. **PLA** – Programmable Logic Array.
2. **PAL/GAL** – Programmable Array Logic/Generic Array Logic

Ambele conțin arii de porți logice ŞI şi SAU, conectate consecutiv, cu ajutorul cărora pot fi realizate formele disjunctive ale funcțiilor logice.



Parametrii de bază:
n – numărul variabilelor logice
p – numărul termenilor produs
m – numărul funcțiilor logice realizate (numărul ieşirilor)

În planul logic ŞI variabilele intră atât în formă directă cât şi în formă inversă. La ieşirile acestui plan se formează temenii produs. Aceşti termeni servesc drept intrări pentru planul logic SAU. În circuitele PLA acest plan este programabil, în circuitele PAL – este fix.

Complexitatea circuitului se determină din formula:

$$C=(2n+m)p$$

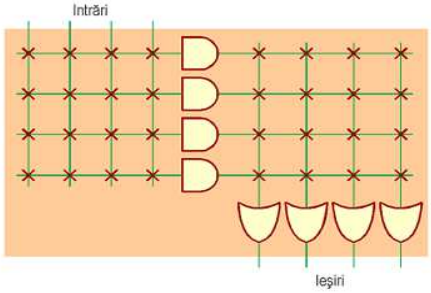
La intrările porților SAU din PLA poate fi aplicată orice combinație a termenilor produs, iar termenii pot fi utilizați de mai multe ori.

În cazul circuitelor PAL, la fiecare poartă SAU pot fi aplicați doar anumiți termeni produs. În cazul când este necesar de a utiliza acelaşi termen produs la altă poartă SAU, el este generat de mai multe ori. Dar programarea circuitelor PAL este mai simplă.

O rețea logică programabilă PLA (Programmable Logic Array) este similară ca şi concept cu o memorie ROM, cu excepția faptului că nu realizează decodificarea completă a variabilelor şi nu generează toți mintermii. Decodificatorul este înlocuit cu o rețea de porți ŞI care poate fi programată pentru a genera termenii produs ai variabilelor de intrare. Termenii produs sunt apoi conectați în mod selectiv cu porți SAU pentru a genera suma termenilor produs pentru funcțiile booleene necesare.

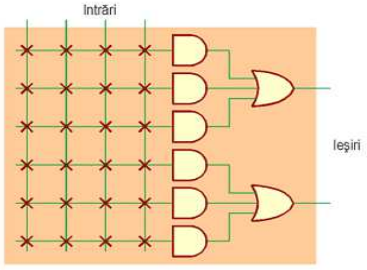
Un circuit PLA poate implementa în mod direct un set de funcții logice exprimate printr-un tabel de adevăr. Fiecare intrare pentru care valoarea funcției este adevărată necesită un termen produs, şi acestuia îi corespunde o linie de porți ŞI din primul etaj al circuitului PLA. Fiecare ieşire corespunde la o linie de porți SAU din al doilea etaj al circuitului. Numărul de porți SAU corespunde cu numărul de intrări din tabelul de adevăr pentru care ieşirea este adevărată.
Pentru proiectarea unui sistem digital cu un circuit PLA, nu este necesar să se indice conexiunile interne ale circuitului, ci trebuie să se specifice doar tabelul de programare.

Structura de bază a unui circuit PLA:



Avantaj: flexibilitate
Dezavantaje: cost ridicat de fabricare, viteză de lucru redusă

Circuitele PAL (Programmable Array Logic) conțin o rețea de porți ŞI programabilă, şi o rețea de porți SAU cu conexiuni fixe. Fiecare linie de ieşire este conectată la un set fix de linii ale rețelei de porți ŞI. O asemenea ieşire a circuitului PAL poate implementa o expresie pe două nivele conținând cel mult opt termeni. Avantajele circuitelor PAL sunt simplitatea utilizării în anumite aplicații şi viteza mai ridicată. Aceste circuite sunt însă mai puțin flexibile decât circuitele PLA.



Avantaje: cost de fabricare mai redus decât PLA, viteza de lucru mai ridicată decât PLA
Dezavantaj: flexibilitate redusă

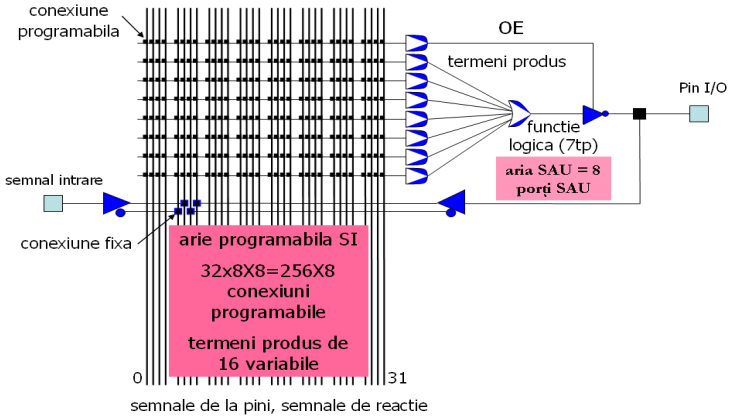
Există şi circuite PLA sau PAL care conțin bistabile ataşate prin conexiuni programabile la ieşirile rețelei de porți SAU, ceea ce permite implementarea unor circuite secvențiale de dimensiuni medii.

Unele circuite PAL conțin macrocelule, formate din bistabile, multiplexoare şi porți logice.

Dispozitivul PAL16L8 (combinațional).

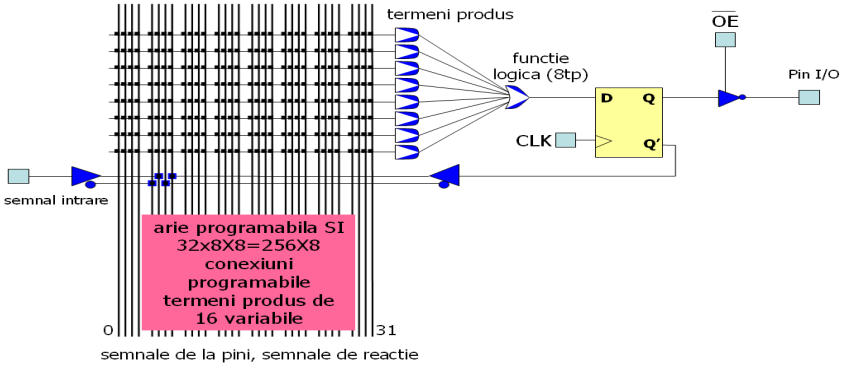
Rețeaua programabilă de porți ŞI conține 64 rânduri şi 32 coloane (64 x 32 = 2048 fuzibile). Fiecare din cele 64 porți ŞI are 32 intrări – 16 variabile în formă directă şi inversă. =pt elemente ŞI sunt asociate cu fiecare pin de ieşire. Şapte dintre aceste porți sunt conectate la poarta fixă SAU. O poartă ŞI serveşte pentru validarea ieşirii (în cazul când primeşte valoarea „1” logic) şi este conectată la un buffer cu trei stări. Circuitul conține 6 pini bidirecționali (13-18)

Structura dispozitivului PAL16L8

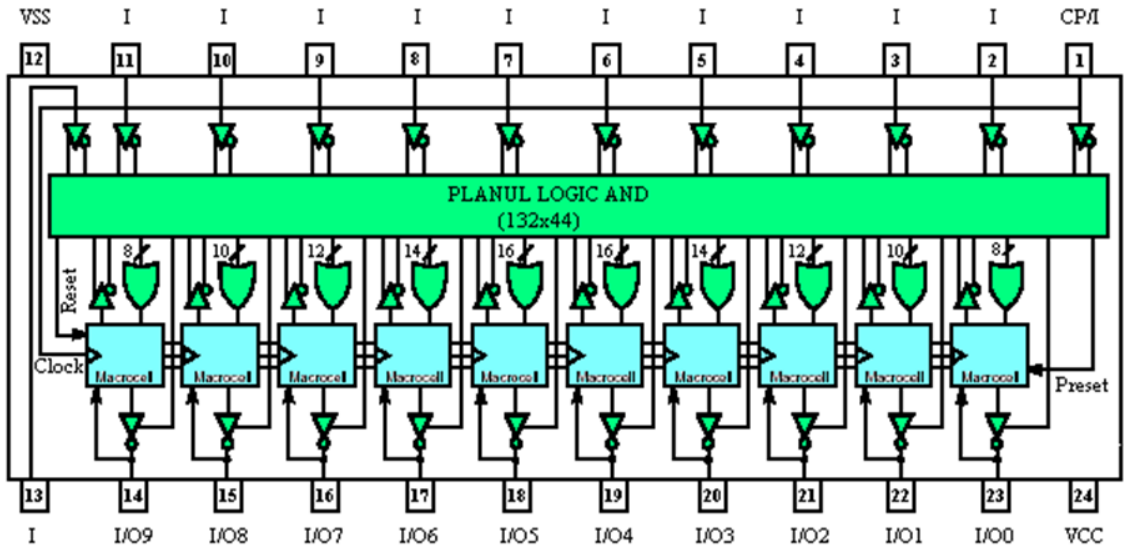


Dispozitivul PAL16R8 (secvențial)

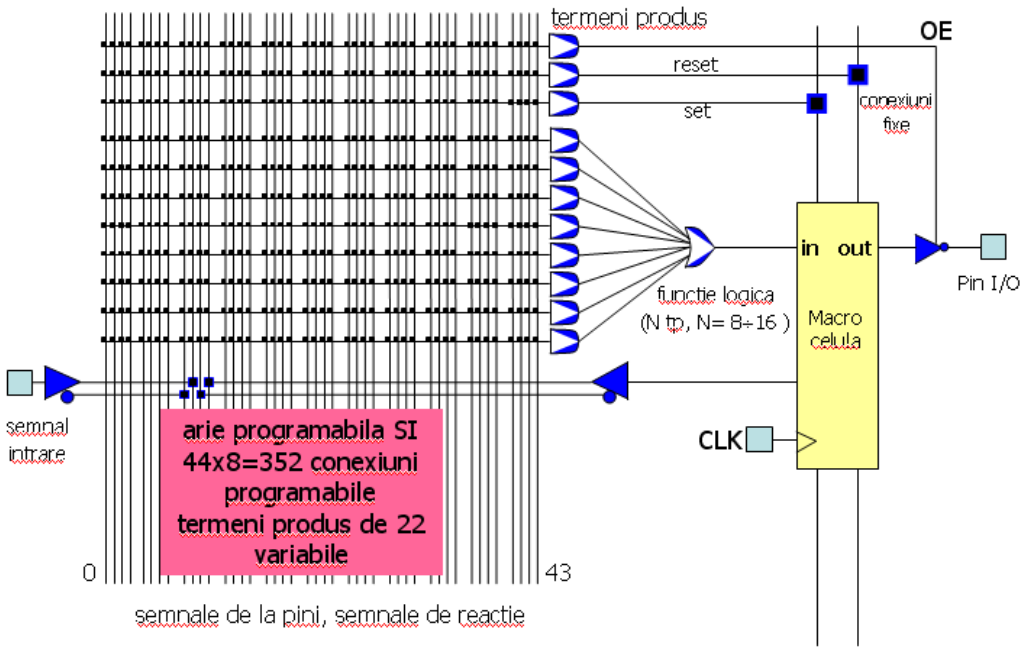
Structura dispozitivului PAL16R8

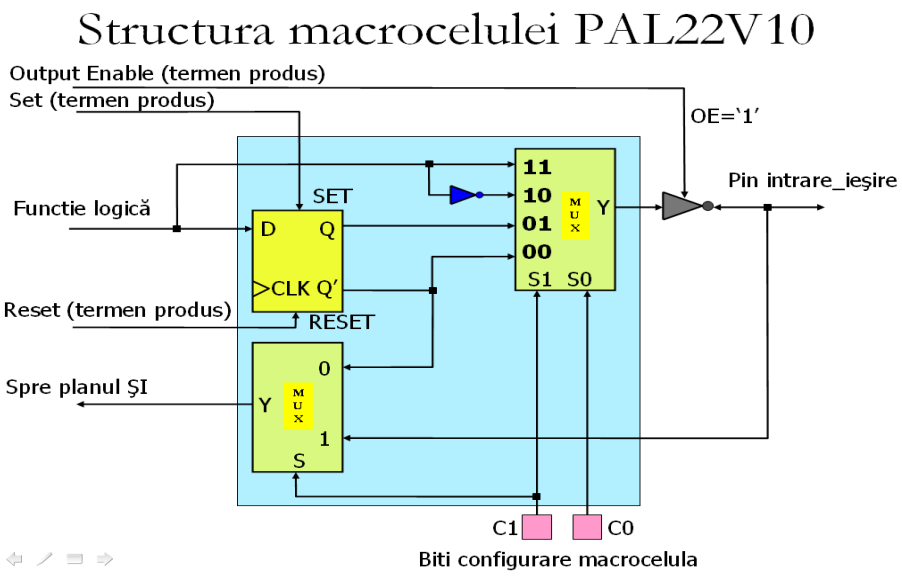


Dispozitivul PAL22V10



Structura dispozitivului PAL22V10



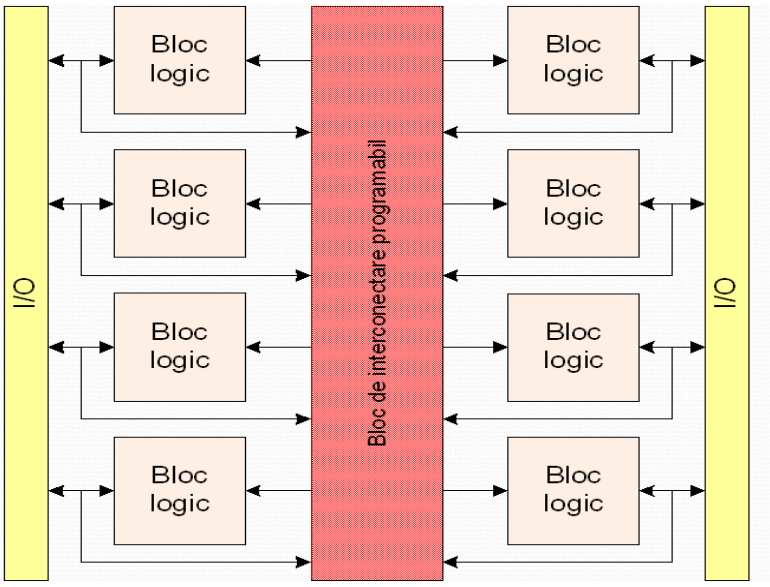


Circuite CPLD

Circuitele CPLD sunt circuite VLSI ale căror părți componente sunt:

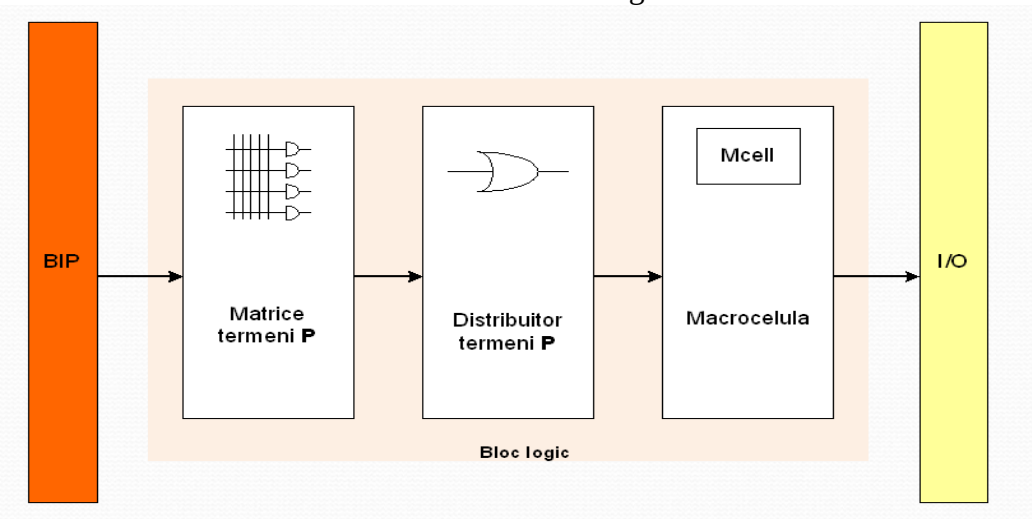
- PAL (GAL), care formează blocurile logice (funcționale), fiecare bloc fiind compus din mai multe macrocelule;
- Bloc de interconectare programabil;
- Blocuri de intrare/ieșire

Funcțiile logice simple pot fi implementate în cadrul uni singur bloc. Funcțiile mai complexe pot necesita mai multe blocuri, care vor fi interconectate prin matricea de rutare.



În circuitele CPLD, spre deosebire de circuitele PAL, matricea SAU nu este complet fixă şi, prin intermediul distribuitorului termenilor produs permite de a forma funcții logice disjunctive, variind numărul termenilor, utilizând aceiași termeni pentru diferite funcții. Macrocelula reprezintă un bistabil care poate fi programat sa lucreze în regim D sau T, multiplexoare pentru alegerea modului de lucru (combinațional sau secvențial), elemente XOR pentru a obține funcția în forma directă sau inversă.

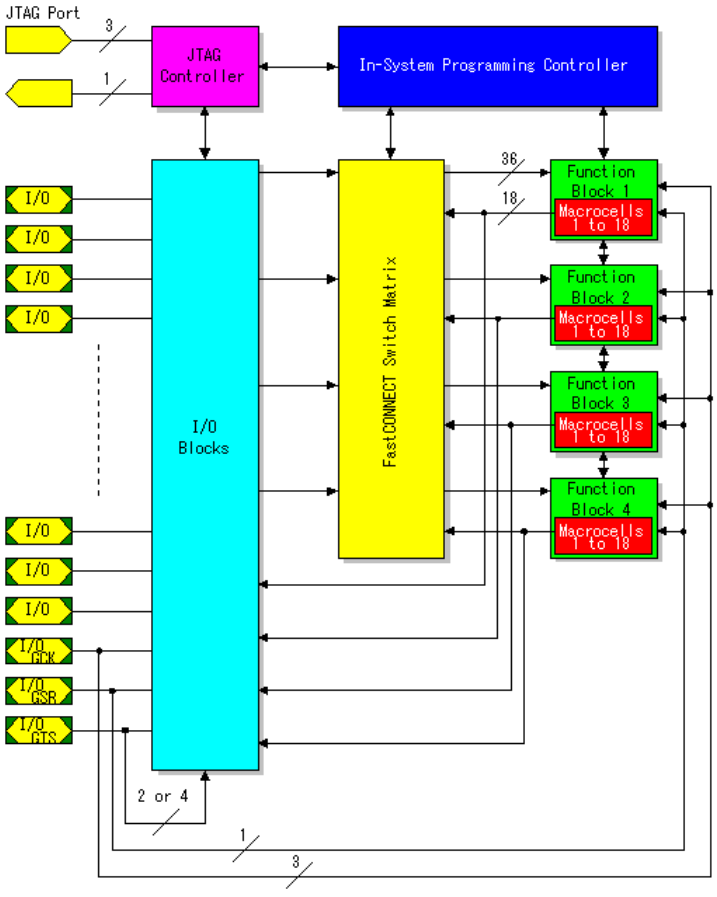
Structura blocului logic



Seria XC9500

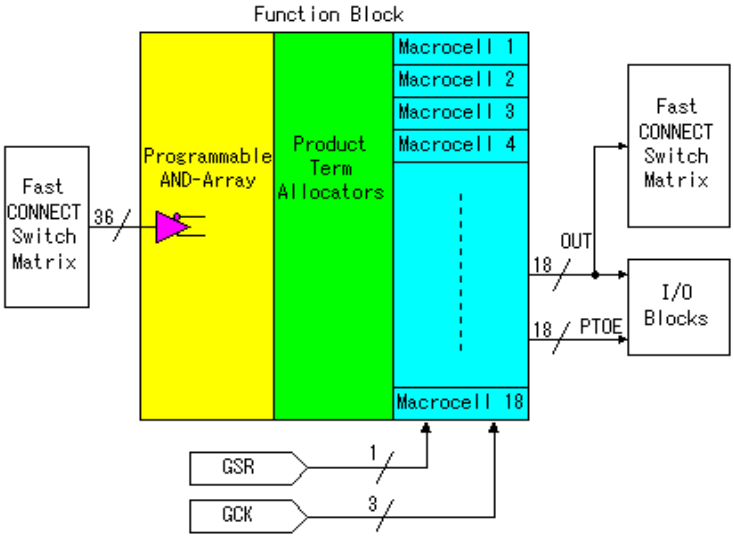
Circuitele înglobează un anumit număr de Blocuri Funcționale (Functional Block) şi de Blocuri de Intrare/Ieşire (I/O Block) interconectate prin intermediul unei matrici de interconectare numită FastConnect. Fiecare Bloc Funcțional are maxim 36 de intrări şi maxim 18 ieşiri. Fiecare Bloc Funcțional este compus din 18 macro celule (macrocell). La nivelul Blocului Funcțional implementarea este tip sumă de produse, fiind disponibili până la 90 de termeni produs (Product Terms). Fiecare macro celulă conține un bistabil - register (numărul de bistabile disponibile este egal cu numărul de macro celule al circuitului) care poate fi configurat ca fiind de tip D sau T sau absent când este neutilizat, în cazul în care funcția implementată este pur combinațională.

Structura funcțională de bază a circuitului CPLD XC9572XL:



Fiecare bloc funcțional constă din aria programabilă de porți ŞI, distribuitorul de termeni şi macrocelule. Macrocelula constă dintr-un bistabil de tip D/T. Semnalele de setare/resetare şi ceas pentru bistabile sunt alocate de către distribuitorul de termeni. PTOE (product term output enable) este direcționat spre Blocul de intrae/ieşire, la fel, de către distribuitorul de termeni.

Structura unui bloc funcțional



Familiile CPLD sunt bazate pe o tehnologie combinată FLASH-EPROM şi RAM. Configurația este memorată permanent într-o memorie de tip FLASH, fiind transferată la punerea sub tensiune (power-up) într-o memorie de configurare de tip RAM (oarecum similară cu cea de la FPGA). Structura de configurare astfel rezultată este nevolatilă.

Familiile reprezentative de circuite CPLD ale firmei Xilinx sunt: XC9500, XC9500XL, XC9500XV, Cool Runner II, Cool Runner XPLA3; ale firmei ALTERA: MAX7000, MAX3000A.

Circuite FPGA

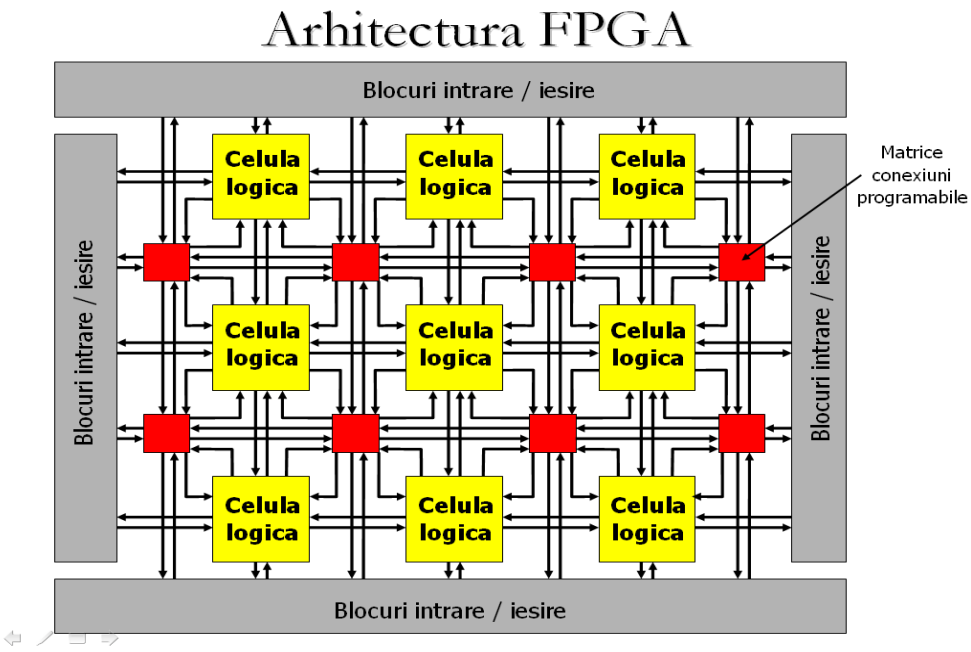
Primul circuit *FPGA* a fost propus de Ross Freeman, unul din co-fondatorii firmei Xilinx Inc., în 1985. De atunci au fost elaborate diferite tipuri de circuite *FPGA* de un număr de alte companii ca Actel, Altera, Atmel, Texas Instruments etc. În comparație cu primele dispozitive programabile (*PLD*), care se deosebesc prin conexiuni rigide, *FPGA*-urile se bazează pe legături flexibile prin intermediul interconexiunilor programabile.

FPGA sunt circuite VLSI cu următoarele componente principale:

- celule logice (blocuri funcționale), amplasate in formă de matrice bidimensională;
- matrice de conexiuni programabile, amplasate în jurul fiecărui bloc funcțional;
- blocuri de intrare/ieşire.

Toate componentele FPGA sunt programabile (reconfigurabile) de către utilizator.

Fiecare celulă logică poate fi programată pentru a implementa orice funcție logică a intrărilor sale. De aceea, aceste blocuri sunt numite blocuri logice configurabile (*CLB – Configurable Logic Block*, Xilinx) Cele mai multe celule logice conțin de asemenea unul sau două bistabile. Fiecare celulă este înconjurată de interconexiuni programabile. Ansamblul acestor interconexiuni poartă numele de matrice de conexiuni programabile. Întreg ansamblul de celule şi interconexiuni se află într-un inel format de blocurile de intrare / ieşire.



Tehnologii de programare a circuitelor FPGA

În prezent sunt utilizate trei tehnologii de programare a circuitelor FPGA:
- Antifuzibile
- SRAM
- EPROM, EEPROM/FLASH

Circuite cu antifuzibile.

Un antifuzibil este un dispozitiv cu două terminale care în mod normal se află în starea de înaltă impedanță, iar atunci când este expus la o tensiune ridicată, trece în starea cu rezistență redusă (300-500 Ω). Antifuzibilele au dimensiuni reduse, astfel încât o arhitectură bazată pe antifuzibile poate conține sute de mii sau milioane de antifuzibile. Pentru simplificarea arhitecturii şi a programării, circuitele FPGA bazate pe antifuzibile constau de obicei din rânduri de elemente logice configurabile cu canale de interconectare între ele, ca şi rețelele de porți tradiționale.

Un bloc logic poate fi programat prin conectarea pinilor săi de intrare la valori fixe sau la rețele de interconectare. Există antifuzibile la fiecare punct de intersecție între interconexiuni şi pini din canal şi la toate punctele de intersecție între interconexiuni în locurile în care canalele se intersectează.

Din categoria circuitelor FPGA cu antifuzibile fac parte circuitele firmelor Actel, Quicklogic, Cypress.

Avantaje.

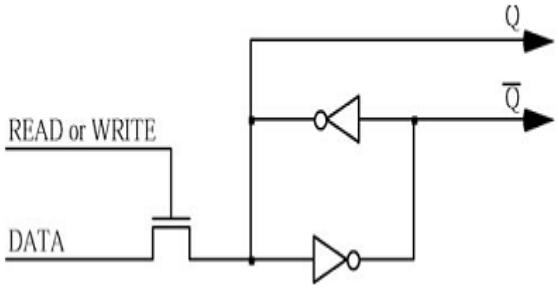
1. Sunt nevolatile
2. Timpul de rutare este mic şi deci sunt mai rapide
3. Consum mai mic de putere
4. Securitate înaltă a datelor de configurare, deoarece circuitele sunt configurate o singură dată şi informaşia de configurare nu este accesibilă concurenților.

Dezavantaje

1. Necesită un proces de fabricație nestandard complex.
2. Necesită un programator extern şi, o dată programate, nu mai pot fi reprogramate.
   Aceste dezavantaje au contribuit la dezvoltarea relativ lentă a acestor circuite.

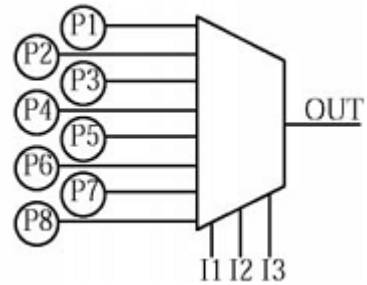
Circuite cu memorii SRAM

Programarea acestor circuite se realizează prin celule de memorie statică.
O celulă SRAM dintr-un FPGA are structura:



După cum se poate observa există perechea de inversoare tipică celulei SRAM care va memora valoarea programată. O singură poartă de tranzistor de tip MOSFET n (MOS Field-Effect-Transistors) este folosită atât pentru scriere cât şi pentru citire. Pentru controlul căii de configurare sunt folosite semnalele Q şi /Q.

Logica este implementată cu ajutorul unor tabele LUT (lookup table) realizate din celulele de memorie, intrările funcțiilor controlând liniile de adresă.



Prin conectarea a 2^N celule de memorie SRAM prin intermediul unui multiplexor se obține un LUT care poate implementa orice funcție de N variabile. Cu toate că este nevoie de un număr mare de celule SRAM, uzual se construiesc LUT-uri cu 5 variabile de intrare. Pentru funcții cu mai mult de cinci variabile se folosesc multiplexoare pentru a forma LUT-uri de 6 sau 7 variabile.

Una sau mai multe tabele, combinate cu bistabile, formează un bloc logic configurabil. Aceste blocuri sunt aranjate într-un tablou bidimensional, segmentele de interconectare formând canale, similar cu rețelele de porți. Segmentele se conectează la pinii blocurilor logice din canale şi la alte segmente din blocurile de comutare prin intermediul tranzistoarelor de trecere controlate de celule ale memoriei de configurare.

O secvență de configurare pentru circuitele cu memorii SRAM constă dintr-un singur cuvânt lung de programare. Logica din circuit încarcă cuvântul de programare, pe care îl citeşte serial dintr-o memorie externă de fiecare dată când circuitul este alimentat. Biții acestui cuvânt setează valorile tuturor celulelor memoriei de configurare din circuit, setând astfel valorile tabelelor şi selectând segmentele care se vor conecta între ele. Circuitele cu memorii SRAM sunt reprogramabile. Ele pot fi actualizate în sistem, punând la dispoziția proiectanților noi opțiuni şi posibilități de proiectare.

Din această categorie de circuite FPGA fac parte cele ale firmelor Xilinx, Altera, AT&T.

Avantaje.

1. Necesită un proces de fabricație standard care permanent este optimizat.
2. Sunt reprogramabile, unele circuite şi în timpul funcționării.

Dezavantaje

1. Sunt volatile şi trebuie programate la fiecare conectare a tensiunii de alimentare a sistemului. Pentru circuitele FPGA pe bază de celule SRAM, în mod uzual configurația este memorată într-o memorie ROM externă din care se încarcă automat la inițializarea sistemului.

2. Timpul de rutare este mare şi deci sunt mai lente.

3. Consum mai mare de putere

4. Densitate de integrare mai mică

5. Au o securitate redusă a informației, deoarece ele trebuie configurate de fiecare dată la conectarea tensiunii de alimentare a sistemului iar codul de configurare poate fi citit.

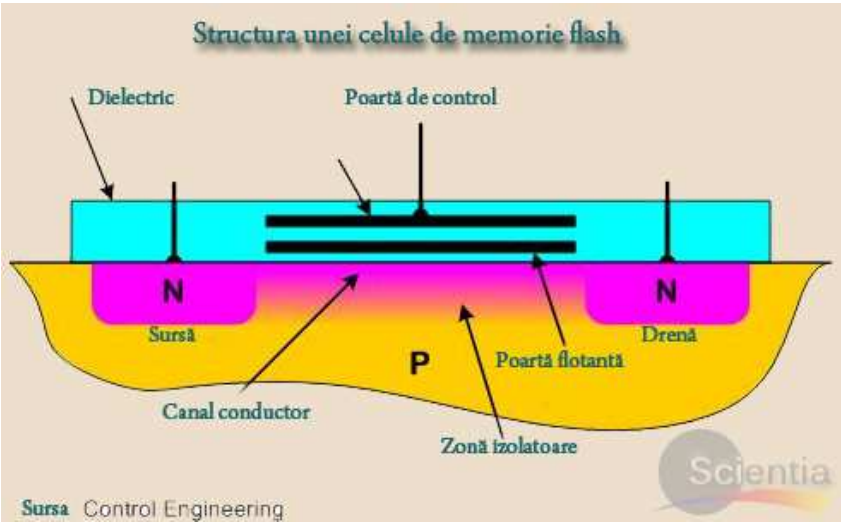
Totuşi avantajele tehnologiei SRAM sunt mai importante decât neajunsurile şi ele, de fapt, domină piața.

Circuite cu EEPROM/FLASH

Aceste circuite utilizează celule EEPROM/FLASH pentru fiecare element programat.

Memoria Flash este un tip de EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory), şi este o memorie nevolatilă, ceea ce înseamnă că nu pierde datele stocate după ce este deconectată de la sursa de alimentare.

Memoria flash este formată dintr-o matrice de celule de memorie care, la rândul lor, sunt formate din perechi de tranzistori (*MOSFET - metal oxide semiconductor field effect transistor)* ce au între ei un strat subțire de oxid izolator. Un tranzistor este numit poartă flotantă (*floating gate*), iar celălalt - poartă de control (*control gate*). Atunci când este stabilită o legatură între cele două porți, celula de memorie are valoarea 1, schimbarea valorii în 0 are loc prin intermediul unui proces numit Fowler-Nordheim tunneling. **Efectul tunel** rezultă din capacitatea unui obiect cuantic de a străbate o barieră de potențial la scară atomică, fapt care ar fi imposibil după legile mecanicii clasice.



Elimina dificultatiile de extragere din soclu si iradiere cu UV, aplicată la memoriile EPROM. Atat programarea cat si stergerea se fac electric. Prin perfectionarea tehnologiei si micsorarea grosimii stratului izolator al P_f exista posibilitatea programarii si stergerii electrice cu tensiuni mici aplicate intre drena si poarta. Polaritatea caderii de tensiune drena-poarta este inversata la stergere fata de programare.

Caracteristici:

- numar de stergeri si de reprogramari $>10^4$;
- durata informatiei memorate mai mare de zece ani.
- se pot rescrie in timpul functionarii.

Din această categorie de circuite FPGA fac parte cele ale firmelor, Altera, Actel, Lattice.

Tehnologia EEPROM/FLASH combină avantajele tehnologiilor precedente.

1. Sunt nevolatile
2. Reprogramabile
3. Folosesc un proces de fabricație standard.
4. Consum redus de putere
5. Securitatea informației

FPGA seria XC4000 (Xilinx)

Blocurile componente

Structura internă programabilă de utilizator include trei elemente majore configurabile:
- **Blocurile logice configurabile (CLBs)**, care furnizează elementele funcționale şi realizează structura logică proiectată;
- **Blocurile de intrare / ieşire (IOBs)**, care furnizează interfața între semnalele interne şi exteriorul circuitului (legătură realizată fizic prin intermediul pinilor);
- **Matrici de comutatoare programabile** pentru interconectarea blocurilor.

Blocurile logice configurabile (CLB)

CLB-urile implementează majoritatea funcțiilor logice proiectate.

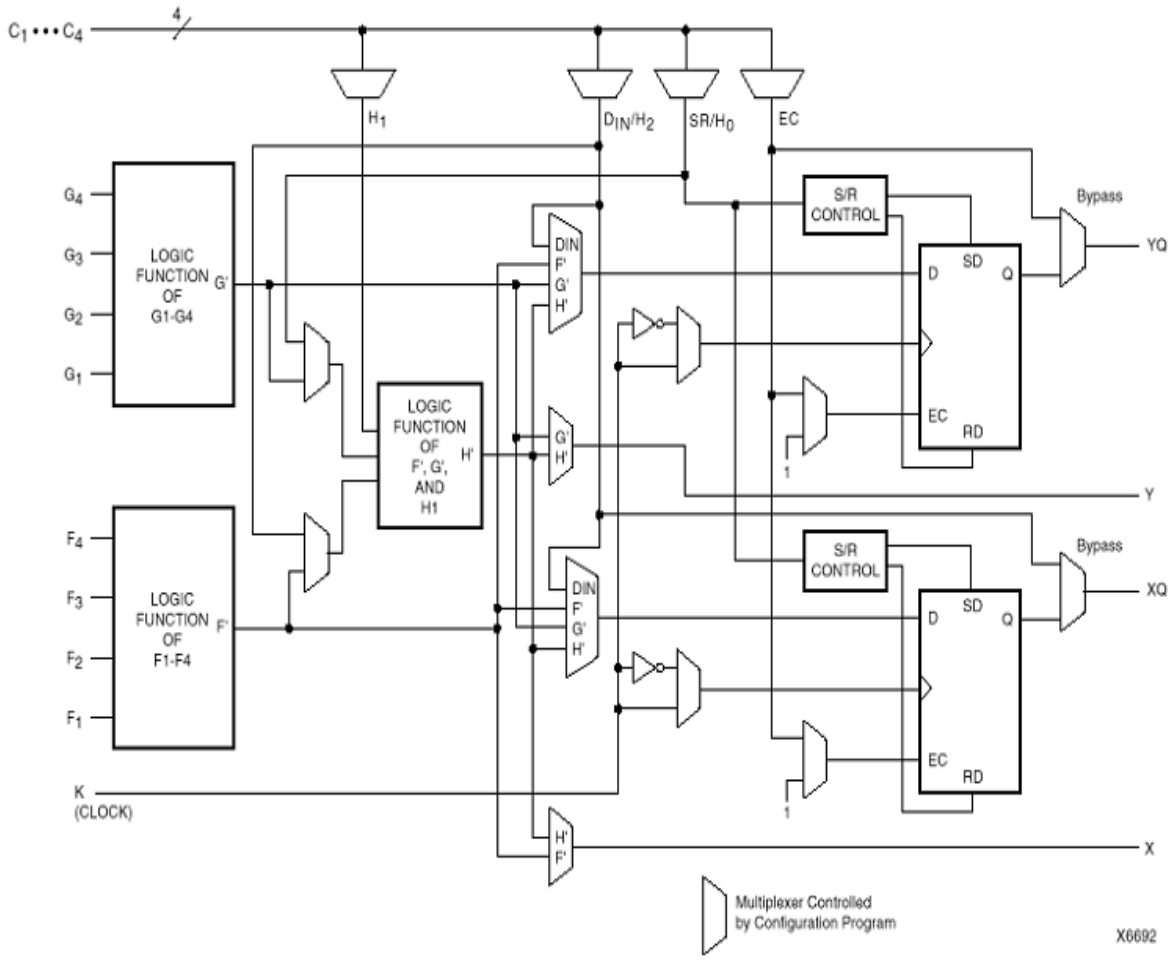
Elementele principale ale unui bloc configurabil sunt:

- F şi G generatoare de funcții cu 2x4 intrări. H generator de funcții suplimentar ce posedă 3 intrări dintre care două provin de la generatoarele de funcții F şi G, iar a treia provine din exterior de la unul din CLB-ulurile vecine. Astfel blocul configurabil poate implementa funcții logice cu până la 9 variabile de intrare. Prin implementarea funcțiilor logice, cu un număr mare de variabile într-un singur bloc, se reduce numărul de blocuri necesare pentru realizarea proiectului. De asemenea se reduc şi timpii de propagare, astfel că va crește capacitatea de implementare şi viteza de lucru. Generatoarele de funcții F' şi G' se pot utiliza ca celule de memorie RAM/ROM.

- două elemente de stocare numite registre (bistabili D), care se pot utiliza pentru stocarea rezultatelor date de generatoarele de funcții.

Elementele de registru sau generatoarele de funcții se pot utiliza şi independent. Intrările DIN, H1 sunt intrări directe de stocare. Ieşirile generatoarelor de funcții se pot utiliza ca ieşiri independente de ieşirile elementelor de stocare. Această flexibilitate măreşte resursele logice şi simplifică implementarea proiectelor. Astfel prin intermediul a 13 intrări şi 4 ieşiri este asigurat accesul la fiecare bloc configurabil. Intrările şi ieşirile respective legate la resursele programabile de interconectare vor realiza funcțiile logice.

Bistabilii interni pe lângă faptul că sunt ieşiri pentru rețeaua de interconexiuni, ieşirile combinaționale pot fi şi sursă de intrare pentru bistabili interni ai CLB pentru realizarea circuitelor secvențiale. Bistabilii sunt de tip D şi au tactul de înscriere pe front pozitiv prin semnalul comun de tact K. Validarea tactului se realizează prin semnalul EC (Enable Clock).

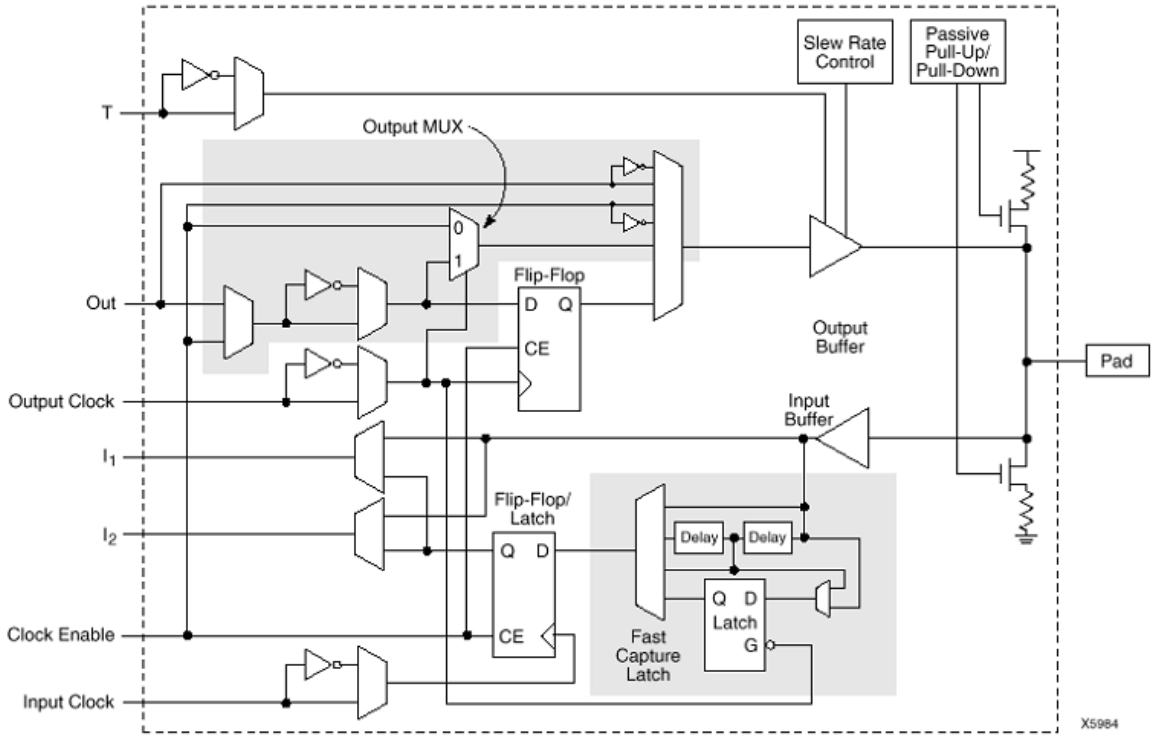


Circuitele XC4000 au caracteristici care permit integrarea unor sisteme complete. De exemplu, fiecare bloc CLB conține circuite care permit execuția eficientă a operațiilor aritmetice. Acestea implementează operații cu transport rapid pentru circuite de tip sumator. De asemenea, tabelele pot fi configurate ca celule RAM de tip R/W. Circuitele din seria XC4000E permit configurarea tabelelor ca memorii RAM cu porturi duale, cu un singur port de scriere şi două porturi de citire, existând posibilitatea ca blocurile RAM să fie sincrone. Fiecare circuit XC4000 conține planuri ŞI largi în jurul periferiei rețelei de blocuri logice pentru a facilita implementarea blocurilor de circuit cum sunt decodificatoarele de dimensiuni mari.

Blocurile de intrare / ieşire (IOBs)

Blocurile configurabile de intrare / ieşire realizează interfața între mediul exterior şi structura internă a circuitului FPGA. Fiecare IOB controlează un pin (pad) al circuitului integrat. Blocurile de intrare ieşire se pot configura ca şi port de intrare, port de ieşire sau port bidirecțional.

Diagrama simplificată a blocului IOB:



Căile I1 şi I2 furnizează semnalele de intrare în IOB. Acestea sunt conectate la un bistabil al cărui tact de înscriere poate fi pe front sau pe nivel logic. Semnalele I1 şi I2 se pot conecta la bistabilul de intrare al blocului IOB.

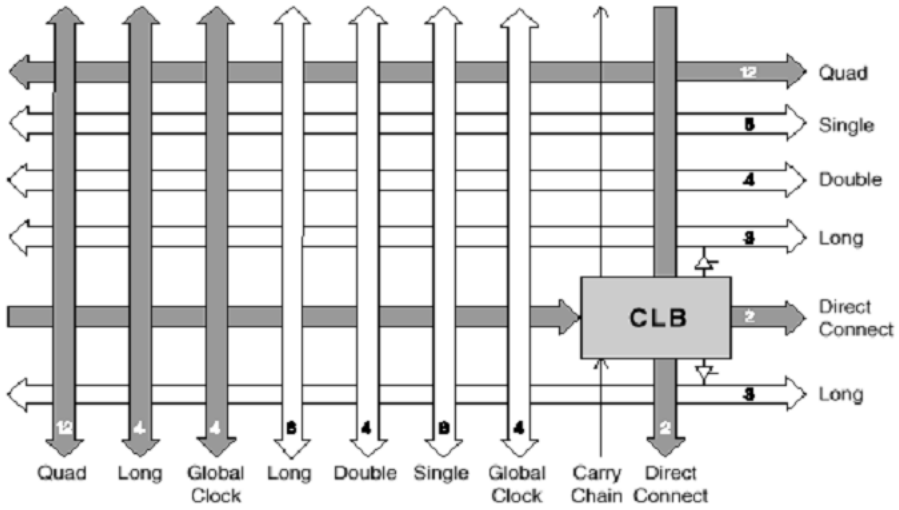
Semnalul de validare a bistabililor din IOB poate fi configurat astfel încât să fie comun sau separat pentru cele două registre. Acest semnal nu se poate inversa în interiorul IOB.

În mod opțional semnalele de ieşire se pot inversa în interiorul blocului de intrare / ieşire. Aceste semnale se pot conecta direct la ieşirea pinului sau la bistabilul de ieşire din IOB.

Slew Rate Control - Controlul vitezei de creştere a semnalului

Interconexiunile programabile

Toate conexiunile interne sunt compuse din segmente de metal cu puncte de cuplare programabile şi matrice de cuplare pentru realizarea legăturilor interne.



Există următoarele tipuri de conexiuni:
- Realizarea legăturilor între blocurile CLB este asociată rândurilor şi coloanelor matricei CLB;
- Realizarea legăturilor între blocurile IOB formează un cordon în jurul matricii CLB (VersaRing), acest cordon conectează pinii I/O cu blocurile logice interne;
- Conexiunile globale sunt compuse din rețele dedicate, proiectate pentru distribuirea rapidă a semnalelor de comandă şi control utilizate în proiect.

Se disting 5 tipuri de linii de interconectare: linii de lungime simplă, linii de lungime dublă, linii de lungime quad-dublă, linii de lungime octală, linii lungi.

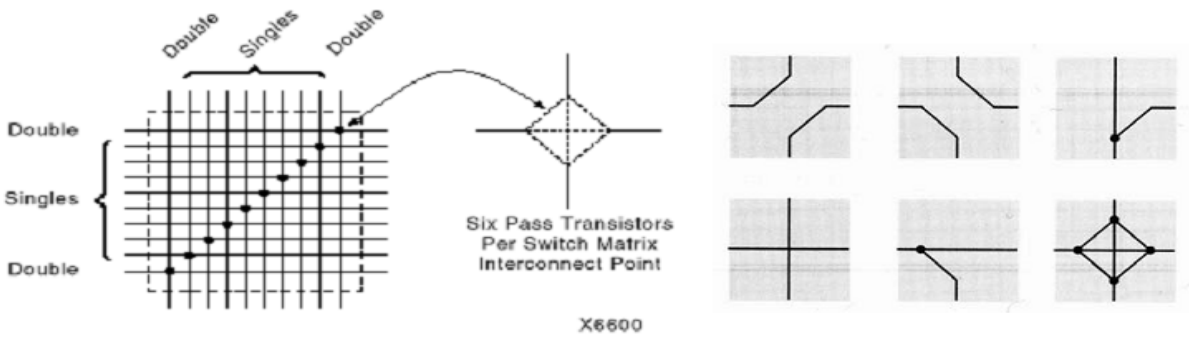
Întrările şi ieşirile în blocurile configurabile sunt distribuite în toate cele patru direcții, pentru ca realizarea conexiunilor să fie cât mai flexibilă. Poziția intrărilor şi ieşirilor CLB este interschimbabilă pentru evitarea congestiunilor, în timpul plasări şi realizări conexiunilor.

Liniile de simplă lungime permit interconectarea rapidă a blocurilor adiacente. Fiecărui bloc îi sunt asociate 8 linii verticale şi 8 linii orizontale de simplă lungime. Aceste linii sunt conectate în matrice programabile situate la intersecțiile fiecărei linii şi coloane. Liniile de simplă lungime introduc întârzieri în propagarea semnalelor ori de câte ori intră într-o matrice programabilă. Din această cauză nu sunt utilizabile pentru realizarea legăturilor la "distanță". Ele sunt utilizate în mod normal pentru a ghida semnalele într-o arie restrânsă.

Liniile de dublă lungime realizează o grilă din segmente de metal. Fiecare linie are lungime dublă în comparație cu liniile simple. Liniile duble sunt grupate în perechi şi intră în fiecare a doua matrice. Fiecărui CLB i se asociază patru linii verticale şi patru linii orizontale, pentru realizarea conexiunilor.

Liniile lungi asemănător liniilor de dublă lungime formează o grilă de metal ce acoperă toată aria circuitului. Aceste linii cu un fun-out ridicat şi timpi de propagare minimi pot conduce la semnale critice. La cele două linii orizontale fiecare CLB se poate conecta prin intermediul unui buffer 3-state. Astfel că aceste linii pot implementa bus-uri uni- sau bi-direcționale sau funcții cablate.

Matricea de cuplare programabilă. Liniile de simplă şi dublă lungime verticale şi orizontale se intersectează în aşa numita matrice de cuplare programabilă (programmable switcing matrix – PSM). Romburile de la intersecția a două linii reprezintă tranzistoarele de trecere utilizate pentru realizarea conexiunilor între linii
. Câteva posibilități de interconectare oferite de PSM:



Arhitecturi FPGA şi CPLD combinate

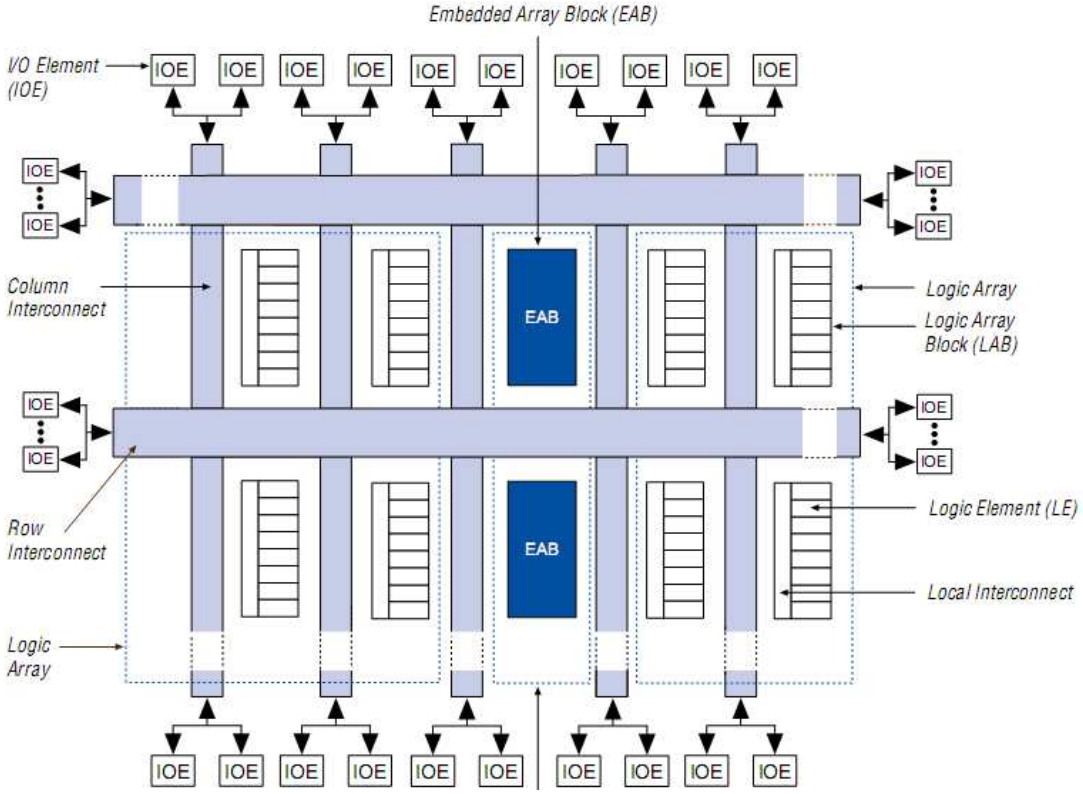
Tehnologia CPLD oferă performanțe asemănătoare tehnologiei FPGA pentru aplicații de complexitate medie, la un preț de cost mai scăzut. În acelaşi timp circuitele CPLD au o rețea de interconexiuni de lungime fixă (o proprietate a matricii de interconectare) ceea ce face ca sistemele numerice dezvoltate cu un circuit CPLD să prezinte întârzieri predictibile în totalitate

chiar din faza de proiectare. Întârzierile dintre fiecare două celule logice conținute în CPLD sunt fixe şi se cunosc. Aceasta se datorează faptului că structura de interconexiuni dintr-un CPLD este formată din linii conductoare de lungime constantă ce străbat structura circuitului pe toată lungimea şi lățimea acestuia.

În contrast cu CPLD-urile, FPGA-urile au o structură de interconexiuni formată din segmente care străbat circuitul, iar capetele acestora sunt conectate de o matrice de interconectare permițând semnalelor să ajungă de la o celulă logică la alta. Numărul de segmente necesare pentru a conecta două celule logice nu este nici fix şi nici predictibil, deci întârzierile nu se pot cunoaşte decât după ce se face asignarea şi plasarea celulelor (după implementare) .

Odată cu creşterea nivelului de integrare a circuitelor au apărut arhitecturi care combină avantajele CPLD şi FPGA. Un exemplu este familia de circuite FLEX (Flexible Logic Element matriX), firma ALTERA. Aceste arhitecturi sunt utilizate şi în circuitele integrate de tipul „sistem pe chip" – SOPC (system on programmable chip).

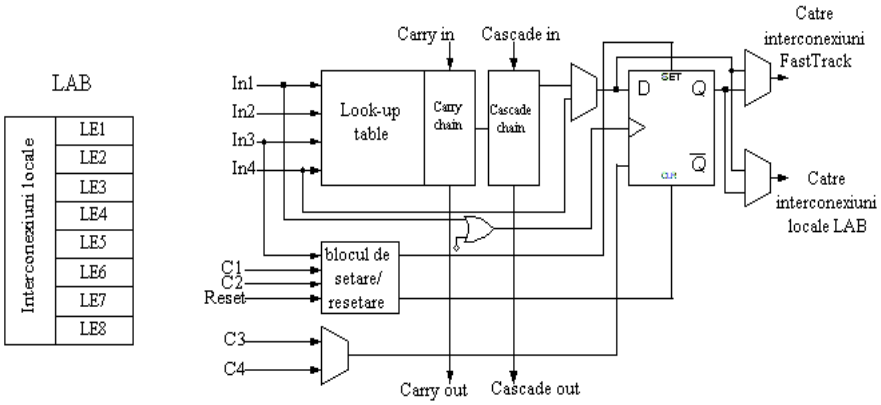
Schema de structură a circuitului FLEX10K:



Arhitectura generalã a circuitelor Altera, care se bazează pe tehnologia de programare EPROM constã dintr-o rețea de celule programabile, numite blocuri ale rețelei logice (Logic Array Block - LAB), interconectate printr-o resursă de rutare numitã rețea de interconectare programabilă (Programmable Interconnect Array - PIA).

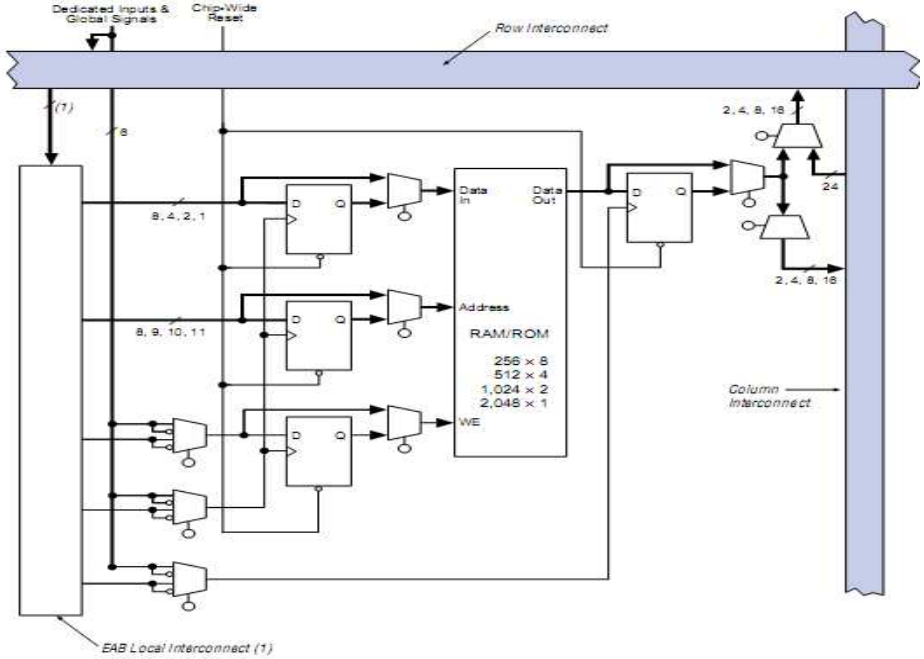
Interconexiunile sunt de două tipuri: locale (pentru conectarea elementelor logice din LAB) şi globale (pentru conectarea LAB-urilor).

Un bloc *LAB(Logic Arraz Blocks)* al circuitului *Flex10K* constă din 8 elemente logice (*LE*) şi interconexiuni locale prin care un element logic poate fi conectat cu oricare alt element logic din cadrul aceluiaşi bloc. Fiecare element logic constă dintr-o tabelă *LUT* cu patru intrări, care reprezintă un generator de funcții cu patru variabile, un bistabil programabil cu intrare de validare sincronă şi două tipuri de interconexiuni dedicate pentru fluxul de date: lanțul de transport şi lanțul de cascadare.



Interconexiunile locale sunt legate la interconexiunile globale FastTrack ale circuitului. Ca şi liniile lungi ale circuitului XC4000, fiecare conexiune FastTrack se extinde pe toată înălțimea sau lățimea circuitului. O diferență majorã între circuitele Flex10K şi FPGA este însă cã FastTrack conține numai linii lungi (tipic pentru CPLD), ceea ce permite configurarea simplă a circuitului. Toate liniile orizontale FastTrack sunt identice. De aceea, întârzierile de interconectare ale circuitului sunt mai predictibile decât cele ale altor circuite FPGA care utilizează segmente mai scurte, deoarece căile mai lungi conțin mai puține comutatoare programabile. Mai mult, conexiunile între liniile orizontale şi verticale trec prin buffere active, îmbunătățind în plus predictibilitatea.

Familia de circuite Flex 10K dispune în plus de blocuri SRAM încorporate (EAB – Embedded Array Block) de dimensiune variabilã. Structura unui EAB:



Blocurile SRAM pot fi configurate pentru diferite aranjamente: 256×8, 512×4, 1K×2, sau 2K×1. Aceste blocuri pot fi configurate şi pentru implementarea unui circuit logic complex, ca de exemplu un circuit de înmulțire.

Blocurile SRAM, în afară de modulul de memorie mai conțin câteva bistabile D sincrone şi multiplexoare programabile. Interconexiunea locala EAB primeşte 22-26 semnale de la rândul de interconexiune globală. Înscrierea datelor în modulul de memorie poate fi sincrona – de la bistabili, şi asincronă, direct de la interconexiunea locală. Semnalele de ieşire ale modulului de memorie pot fi transmise fie la rândul, fie la coloana de interconexiuni globale sincron – de la ieşirea bistabilului, asincron – direct de la modul.

Sisteme pe circuite integrate programabile - SOPC (System on programmable Chip)

Costul redus, performanța ridicată, densitatea mare a circuitelor programabile a făcut posibilă apariția circuitelor integrate, numite SOPC.

SOPC pot fi de 2 tipuri:

- cu structură uniformă şi posibilitate de reconfigurare a tuturor unităților din sistem (nuclee programabile - soft cores);
- cu unități fixe în care sunt realizate anumite funcții şi unități programabile (nuclee hard – hard cores). Inițial, nucleele hard erau destul de simple, în prezent ele reprezintă microprocesoare sau microcontrolere.

În SOPC cu nuclee programabile este posibilă realizarea părților componente a procesoarelor, memoriilor, dispozitivelor periferice. Resursele de proiectare permit formarea pe acelaşi circuit a componentelor virtuale – nucleelor soft, IP (Intellectual Property) a diferitor producători. Dar în aceste sisteme nu se atinge viteza maximă a nucleelor.

În SOPC cu nuclee hard, unitățile predefinite ocupă un spațiu de câteva ori mai mic, în comparație cu nuclee soft, deoarece ele nu conțin interconexiuni programabile şi sunt optimizate pentru îndeplinirea funcțiilor concrete. În acelaşi timp, se pierde flexibilitatea funcțională. Nucleele hard sunt fixate pe suprafața circuitului şi aceasta poate crea dificultăți în cazul plasării şi rutării componentelor programabile, neatingându-se performanța maximă.

În prezent se dezvoltă pe larg ambele tipuri de SOPC. De exemplu, firma Altera utilizează nucleul programabil Nios pe circuitele APEX20KE şi APEXII. Nivelul de integrare a acestor circuite este atât de înalt, că nucleul ocupă un spațiu foarte mic. Firma Xilinx a elaborat nucleul programabil Microblaze pentru familia VirtexII, care lucrează pe o frecvență de 125MHz.

Nucleele hard se bazează pe arhitectura RISC a procesoarelor firmelor ARM Limited, MIPS Tehnologies şi IBM Microelectronics, lucrează pe o frecvență de peste 200 MHz, au inclus principiul pipeline, îndeplinesc operații scalare au un consum de putere redus şi ocupă 2-3 mm^2. Arhitecturile standarde ale acestor procesoare permit utilizarea diverselor resurse CAD ce reduce complexitatea de proiectare şi micşorează timpul de lansare pe piață.

Exemple de SOPC cu nuclee programabile:
Familia APEX20K, Altera;
Familia Virtex, Xilinx;
Familia ProASIC, Actel;
Familia Delta39K, Cypress Semiconductor.

Exemple de SOPC cu nuclee hard fără procesoare încorporate:

Familia ESP (Embedded Standard Products), firma QuickLogic, cuurmătoarele subfamilii: QuickRAM, QuickPCI, quickPC, QuickDSP, QuickSD. Părțile programabile a acestor circuite sunt realizate cu antifuzibile şi au următoarele proprietăți: o singură programare, volum mai mic a interconexiunilor, viteză de lucru înaltă, sensibilitate mai mică la temperatură şi radiație.

Exemple de SOPC cu nuclee hard cu procesoare încorporate:

1. FPSLIC (Field Programmable System-Level Integration Chip), Atmel, cu următoarele blocuri:

- microcontroler AVR şi unitățile periferice;
- memorie SRAM;
- FPGA AT40K

2. Virtex4, Xilinx. Componente:

- procesoare PowerPC
- controlerul Ethernet MAC (Media Access Control) cu trei moduri de operare (10/100/1000 Mb/s)

- tranceivere seriale cu viteza cuprinsă între 622 Mb/s şi 11,1 Gb/s
- slice-uri dedicate procesării digitale a semnalelor DSP (XtremeDSP Slice) funcționând la 500 MHz. Acestea conțin:
- multiplicatoare pe 18x18 biți, dedicate,
- nivele de pipeline opționale pentru îmbunătățirea performanțelor,
- blocuri multiplicatoare-acumulatoare sau multiplicatoare-sumatoare,

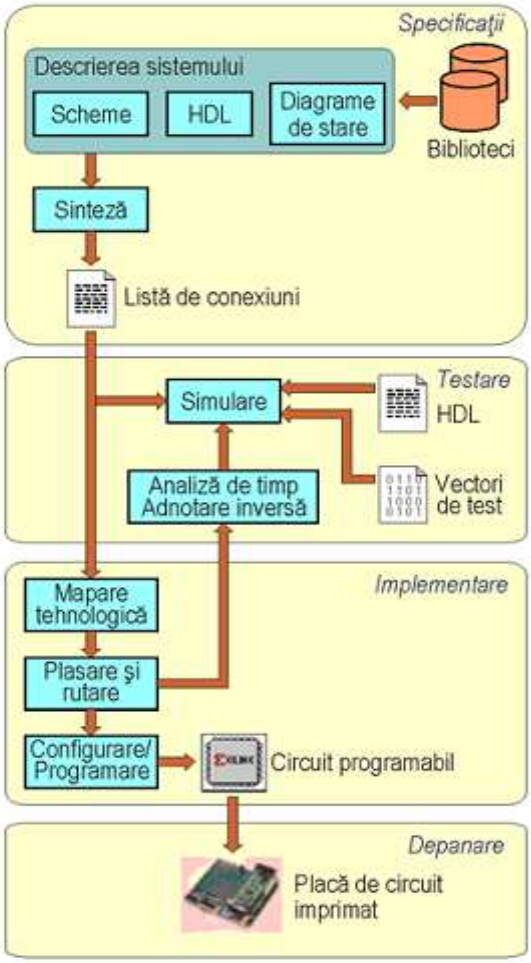
Procesul de proiectare cu circuite programabile

Etapele de proiectare

Pentru proiectarea sistemelor digitale utilizând circuite programabile, cum sunt circuitele FPGA şi CPLD, se utilizează pachete de programe de proiectare asistată de calculator (CAD – Computer Aided Design). Aceste pachete de programe asistă proiectantul în toate etapele procesului de proiectare. Astfel, majoritatea pachetelor CAD pentru circuitele programabile asigură următoarele funcții principale:

- Specificarea (descrierea) sistemului digital;
- Sinteza descrierii, deci transformarea acesteia într-o listă de conexiuni conținând porți elementare şi interconexiunile dintre ele;
- Simularea funcționării sistemului pe baza listei de conexiuni obținute, înainte de implementarea într-un anumit circuit;
- Implementarea sistemului într-un circuit prin adaptarea listei de conexiuni pentru a se utiliza în mod eficient resursele disponibile ale circuitului;
- Configurarea (programarea) circuitului pentru ca acesta să realizeze funcția dorită.

Etapele din cadrul procesului de proiectare a sistemelor digitale utilizând circuite programabile:



Descrierea sistemului

Principalele metode pentru descrierea sistemelor digitale:
- prin scheme logice,
- prin limbaje de descriere hardware (HDL – Hardware Description Language),
- prin diagrame de stare.

În mod tradițional, sistemele digitale sunt descrise prin scheme logice. Pentru aceasta se utilizează un editor schematic, care permite specificarea componentelor care trebuie utilizate şi a modului în care acestea trebuie interconectate.

Pe lângă schemele logice, o altă posibilitate pentru descrierea sistemelor digitale este cu ajutorul limbajelor de descriere hardware. Aceste limbaje sunt din ce în ce mai utilizate, fiind preferate pentru descrierea sistemelor cu complexitate mai ridicată, datorită următoarelor avantaje principale:

• Posibilitatea descrierii funcționale a sistemelor, aceasta fiind o descriere la un nivel mai înalt, fără detalierea structurii la nivelul componentelor simple sau a porților elementare. Astfel, timpul necesar pentru descrierea sistemelor complexe se reduce în mod semnificativ.

• Independența descrierilor HDL față de diferitele tipuri de circuite. În timp ce schemele logice sunt realizate cu componente de bibliotecă specifice unei anumite familii de circuite, descrierile HDL sunt complet independente de un anumit circuit, astfel încât aceeaşi descriere se poate utiliza pentru implementarea sistemului într-un anumit circuit FPGA, dar şi într-un alt tip de circuit programabil, de exemplu, într-o rețea logică programabilă.

• Posibilitatea modificării mai simple a descrierii HDL a unui sistem, datorită faptului că o asemenea descriere reprezintă în acelaşi timp o documentare a sistemului.

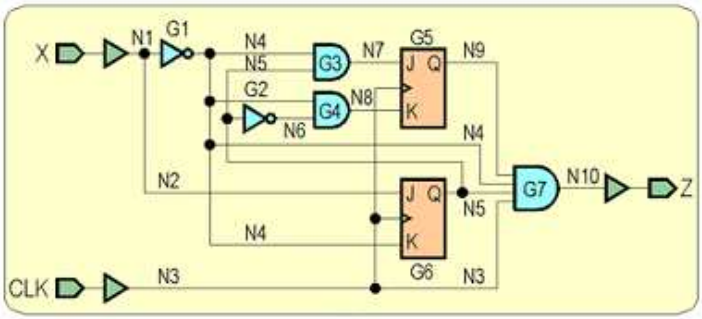
Există diferite limbaje de descriere hardware, dar mai utilizat este limbajul VHDL (VHSIC Hardware Description Language), VHSIC fiind acronimul pentru Very High Speed Integrated Circuit. Pe lângă acest limbaj, pentru proiectarea cu circuite FPGA se mai utilizează limbajul Verilog. Pentru proiectarea cu circuite CPLD, un limbaj utilizat în mod frecvent este ABEL (Advanced Boolean Expression Language). Limbajele VHDL şi Verilog sunt standardizate de institutul IEEE.

Sinteza sistemului

După descrierea sistemului digital, etapa următoare din cadrul procesului de proiectare este cea de sinteză a sistemului. Sinteza constă în translatarea schemei logice, a descrierii HDL sau a diagramei de stare într-o listă de conexiuni. Această translatare se realizează cu ajutorul unui program de sinteză din cadrul sistemului CAD. Lista de conexiuni (“netlist”) este o descriere compactă a sistemului digital sub formă textuală, în care sunt specificate componentele sistemului, interconexiunile dintre acestea şi pinii de intrare/ieşire. Această listă este prelucrată de celelalte componente ale sistemului CAD pentru realizarea etapelor următoare din cadrul procesului de proiectare.

Există diferite formate pentru listele de conexiuni, cel mai utilizat fiind formatul EDIF (Electronic Digital Interchange Format), acesta reprezentând un standard industrial. Pe lângă acest format standard, se pot utiliza diferite formate care sunt specifice anumitor producători de circuite. Un exemplu este formatul XNF (Xilinx Netlist Format), care este formatul propriu al firmei Xilinx, cel mai important producător de circuite programabile de tip FPGA şi CPLD. O altă posibilitate este utilizarea unui limbaj de descriere hardware ca format pentru lista de conexiuni. De exemplu, sistemul CAD poate utiliza o reprezentare structurală a sistemului proiectat într-un limbaj de descriere hardware specificat de proiectant.

Relația dintre schema logică a unui circuit simplu şi un format posibil al unei liste de conexiuni:




```
Component INV G1;
Component INV G2;
Component AND2 G3;
Component AND2 G4;
Component JKF G5;
Component JKF G6;
Component AND4 G7;

Net N1: X, G1.IN;
Net N2: X, G6.J;
Net N3: CLK, G5.C, G6.C, G7.IN4;
Net N4: G1.OUT, G3.IN1, G4.IN1,
        G7.IN2, G6.K;
Net N5: G6.Q, G2.IN, G3.IN2, G7.IN3;
Net N6: G2:OUT, G4:IN2;
Net N7: G3.OUT, G5.J;
Net N8: G4.OUT, G5.K;
Net N9: G5.Q, G7.IN1;
Net N10: G7.OUT, Z;
```

În prima parte a listei de conexiuni sunt declarate componentele din cadrul schemei, iar în a doua parte sunt specificate conexiunile dintre componente. Denumirile componentelor sunt G1..G7, iar denumirile conexiunilor sunt N1..N10. Aceste denumiri sunt fie cele specificate de proiectant, fie cele asignate în mod automat de sistemul CAD.

În circuitul ilustrat există două inversoare (G1 şi G2), două porţi ŞI cu două intrări (G3 şi G4), o poartă ŞI cu patru intrări (G7) şi două bistabile JK (G5 şi G6). Inversoarele au un pin de intrare IN, un pin de ieşire OUT, un pin de alimentare Vcc şi un pin de masă GND. Similar, porţile ŞI cu două intrări au doi pini de intrare IN1 şi IN2, un pin de ieşire OUT, un pin de alimentare şi un pin de masă. Bistabilele au doi pini pentru intrările de date J şi K, un pin pentru intrarea de ceas C şi un pin pentru ieşirea Q, pe lângă pinii de alimentare şi masă. Pentru simplitate, pinii şi semnalele de alimentare şi masă au fost omişi în această figură. O conexiune este indicată prin listarea tuturor pinilor care sunt conectaţi împreună. Semnalele de intrare X şi CLK sunt conectate la pinii de intrare cu aceleaşi nume ai circuitului, iar semnalul de ieşire Z este conectat la pinul de ieşire al circuitului.

Proiectantul poate specifica diferite criterii de optimizare de care să se ţină cont în procesul de sinteză. Exemple de asemenea opţiuni sunt:

- minimizarea numărul de porţi elementare necesare,
- obţinerea vitezei maxime de funcţionare a circuitului,
- minimizarea puterii consumate.

Proiectantul poate experimenta cu diferite criterii de optimizare pentru a obţine soluţia cea mai convenabilă pentru aplicaţia respectivă.

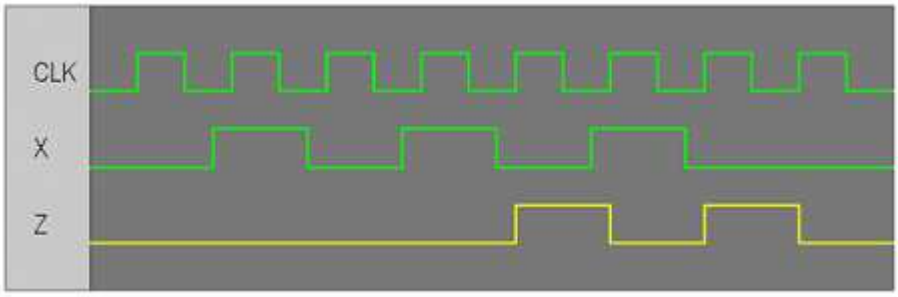
Simularea funcţională

În această etapă se utilizează un program simulator pentru verificarea funcţionării sistemului proiectat, înainte de implementarea acestuia într-un circuit programabil. Această verificare se referă doar la aspectele funcţionale ale sistemului, fără a se lua în considerare întârzierile semnalelor, care vor fi cunoscute numai după implementare. Pentru verificarea

funcțională proiectantul furnizează simulatorului mai multe combinații ale valorilor semnalelor de intrare, o asemenea combinație fiind numită ***vector de test***. De asemenea, proiectantul poate specifica valorile semnalelor de ieşire care trebuie generate de sistem pentru fiecare vector de test.

Simulatorul aplică pe rând câte un vector de test la intrările sistemului, determină semnalele de ieşire care sunt generate de sistem şi le compară cu valorile acestor semnale care au fost specificate de proiectant. În cazul în care apar diferențe, simulatorul afişează mesaje care indică diferențele apărute. Proiectantul va efectua modificările necesare ale descrierii sistemului pentru a corecta erorile apărute, va efectua sinteza descrierii modificate şi va executa din nou simularea funcțională. Aceste etape vor fi repetate până când sistemul va funcționa conform cerințelor.

Modul în care pot fi vizualizate pe ecranul calculatorului semnalele de intrare şi de ieşire ale unui circuit:



Maparea tehnologică

Etapele următoare din cadrul procesului de sinteză realizează implementarea sistemului proiectat într-un circuit programabil (FPGA sau CPLD). Prima etapă din cadrul implementării este cea de mapare tehnologică. Această etapă constă dintr-o serie de operații care realizează prelucrarea listei de conexiuni şi adaptarea acesteia la particularitățile şi resursele disponibile ale circuitului utilizat pentru implementare. Operațiile executate în această etapă diferă în funcție de sistemul de proiectare. Cele mai obişnuite operații sunt:

- adaptarea la elementele fizice ale circuitului,
- optimizarea şi verificarea regulilor de proiectare (de exemplu, testarea depăşirii numărului pinilor de I/E disponibili în cadrul circuitului). În timpul acestei etape, proiectantul selectează tipul circuitului programabil care va fi utilizat, capsula circuitului integrat, viteza şi alte opțiuni specifice circuitului respectiv.

În urma execuției operațiilor din etapa de mapare tehnologică se generează un raport detaliat al rezultatelor tuturor programelor executate. Pe lângă mesaje de eroare şi de avertizare, se creează de obicei o listă cu resursele utilizate din cadrul circuitului.

Plasarea şi rutarea

Aceste operații sunt executate în cazul utilizării unui circuit FPGA pentru implementare.

Pentru proiectarea cu circuite CPLD, operația echivalentă este numită adaptare (“fitting”).

Plasarea este procesul de selectare a unor module sau blocuri logice ale circuitului programabil care vor fi utilizate pentru implementarea diferitelor funcții ale sistemului digital.

Rutarea constă în interconectarea acestor blocuri logice utilizând resursele de rutare disponibile ale circuitului.

Majoritatea sistemelor CAD realizează operațiile de plasare şi rutare în mod automat, astfel încât utilizatorul nu trebuie să cunoască detaliile arhitecturii circuitului utilizat pentru implementare.

Analiza de timp

Pachetele de programe CAD pentru proiectarea sistemelor digitale conțin de obicei un program numit analizor de timp, care poate furniza informații despre întârzierile semnalelor. Aceste informații se referă atât la întârzierile introduse de blocurile logice, cât şi la întârzierile datorate interconexiunilor. Analizorul poate afişa aceste informații în diferite moduri, de exemplu, prin ordonarea conexiunilor în ordinea descrescătoare a întârzierilor semnalelor. Proiectantul poate utiliza informațiile despre întârzierile semnalelor pentru a realiza o nouă simulare a sistemului, în care să se țină cont de aceste întârzieri. Această operație prin care se furnizează simulatorului informații detaliate despre întârzierile semnalelor se numeşte adnotare inversă ("back-annotation").

Anumite sisteme permit utilizatorilor experți plasarea şi rutarea manuală a unor porțiuni critice ale sistemului digital pentru a obține performanțe superioare.

Configurarea sau programarea circuitului

Operația de configurare se referă la circuitele programabile bazate pe memorii volatile SRAM (Static Random Access Memory) şi constă din încărcarea informațiilor de configurare în memoria circuitului. Operația de programare se referă la circuitele programabile bazate pe memorii nevolatile (cum sunt circuitele care conțin antifuzibile). Această operație se execută similar cu cea de configurare, dar informațiile de configurare sunt păstrate şi după întreruperea tensiunii de alimentare.

La sfârşitul operațiilor de plasare şi rutare, se generează un fişier care conține toate informațiile necesare pentru configurarea circuitului. Aceste informații se referă atât la configurarea blocurilor logice ale circuitului, cât şi la specificarea interconexiunilor dintre blocurile logice. Fişierul în care se înscriu aceste informații conține, în principiu, un şir de biți ("bitstream"), fiecare bit indicând starea închisă sau deschisă a unui comutator. Circuitele programabile conțin un număr mare de asemenea comutatoare, un comutator fiind realizat sub forma unui tranzistor sau a unei celule de memorie. Un bit de 1 din şirul de biți va determina închiderea unui comutator şi, deci, stabilirea unei conexiuni.

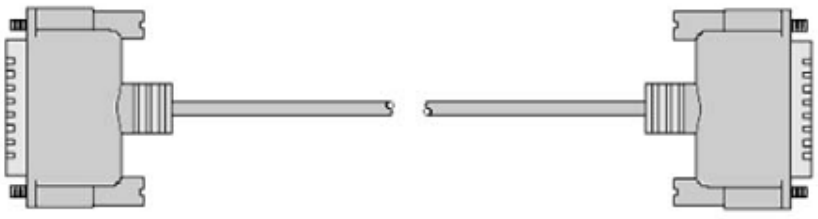
Biții din acest fişier de configurare sunt aranjați într-un anumit format pentru a realiza o corespondență între un bit şi comutatorul corespunzător.

Conținutul fişierului de configurare se transferă la circuitul programabil, aflat de obicei pe o placă de circuit imprimat împreună cu alte circuite. Comutatoarele circuitului se închid sau rămân deschise în funcție de valorile biților din şirul de configurare. După terminarea configurării, circuitul va funcționa conform descrierii sistemului digital care a fost implementat.

Din cauza memoriei volatile, circuitul trebuie configurat din nou după fiecare întrerupere a tensiunii de alimentare. Informațiile de configurare pot fi păstrate într-o memorie nevolatilă PROM (Programmable Read Only Memory), existând posibilitatea configurării automate a circuitului din această memorie nevolatilă la aplicarea tensiunii de alimentare.

Configurarea sau programarea se pot realiza utilizând interfața paralelă a calculatorului. Pentru aceasta, este necesar ca placa cu circuitul programabil să conțină un conector pentru interfața paralelă, pentru transfer utilizându-se un cablu paralel.

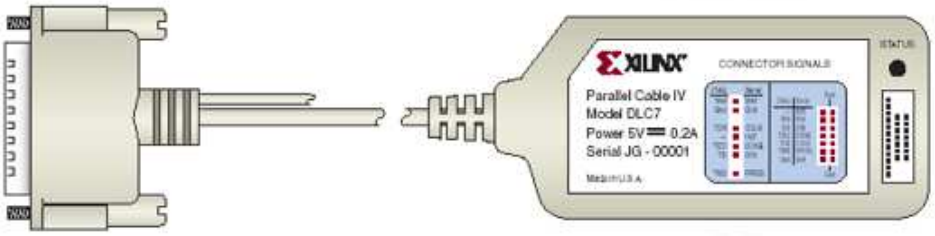
Exemplu de cablu paralel obişnuit, care conține conectori cu 25 de contacte DB25.



O altă posibilitate este utilizarea unui cablu special şi a unei metodologii de configurare propuse de organizația JTAG (Joint Test Advisory Group). Această metodologie, cunoscută şi sub numele de "Boundary-Scan", a fost standardizată de institutele IEEE (Institute of Electrical and Electronic Engineers) şi ANSI (American National Standards Institute) ca standardul 1149.1,

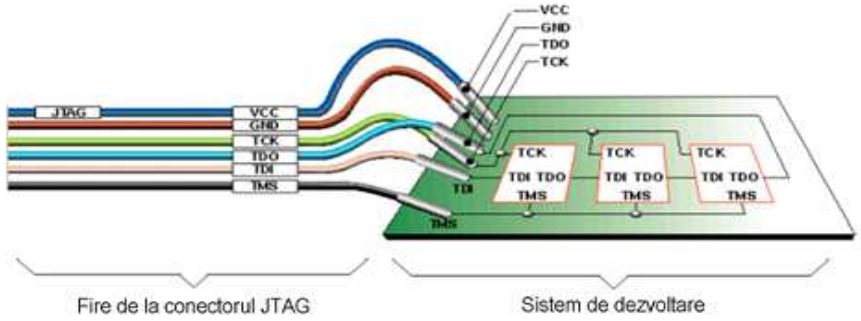
reprezentând un set de reguli de proiectare care facilitează configurarea sau programarea circuitelor, testarea şi depanarea acestora. Un capăt al cablului JTAG se conectează la interfaţa paralelă a calculatorului, iar celălalt capăt se conectează la un număr de 5 pini speciali de pe placa circuitului programabil. Informaţiile de configurare sunt preluate în paralel de la calculator şi sunt transferate serial (bit cu bit) la circuitul programabil. Un asemenea cablu permite şi testarea sistemului digital implementat prin citirea unor informaţii (valori ale semnalelor sau conţinutul unor locaţii de memorie) de la circuitul programabil în timpul funcţionării, transferul acestora la calculator şi vizualizarea lor pe ecran.

Exemplu de cablu JTAG al firmei Xilinx (Parallel Cable IV):



Există mai multe variante de cabluri JTAG produse de această firmă. Cablul MultiLINX poate fi conectat fie la interfaţa serială RS232 a calculatorului, fie la interfaţa USB, prin intermediul unui cablu serial sau al unui cablu USB. Dispozitivul MultiPRO poate fi utilizat atât ca şi cablu de configurare, cât şi ca programator pentru memorii PROM şi circuite CPLD CoolRunner II cu ajutorul unor adaptoare. Cablurile Parallel Cable III şi Parallel Cable IV se conectează la interfaţa paralelă a calculatorului. Cablul Parallel Cable IV permite o rată de transfer superioară comparativ cu cablul Parallel Cable III (de până la 5 MB/s faţă de 500 KB/s).

Exemplu deconectare a unui cablu JTAG în modul JTAG (sau “Boundary-Scan”) la un sistem de dezvoltare conţinând unul sau mai multe circuite programabile. Firele de legătură se conectează cu un capăt la pinii JTAG ai cablului, iar cu celălalt capăt la pinii JTAG corespunzători ai plăcii de dezvoltare. Un asemenea cablu poate fi utilizat fie pentru configurarea unui singur circuit programabil, fie a mai multor circuite conectate într-un lanţ “Boundary-Scan”. De menţionat că un cablu JTAG poate fi utilizat de obicei şi pentru configurarea circuitelor în alte moduri decât modul JTAG, cum sunt modurile “Slave Serial” sau “Slave Parallel”.



Depanarea sistemului

În această ultimă etapă a procesului de proiectare se verifică funcţionarea sistemului digital proiectat în condiţii reale. O funcţionare necorespunzătoare se poate datora nerespectării specificaţiilor de proiectare, a specificaţiilor circuitului utilizat pentru implementare, a unor aspecte legate de întârzierea semnalelor etc. Depanarea poate fi simplificată dacă circuitul se configurează astfel încât să conţină unele module speciale care permit citirea valorii unor semnale în timpul funcţionării şi transferul acestor informaţii la calculator, utilizând un cablu JTAG şi un program special pentru vizualizarea semnalelor dorite.

SISTEME RECONFIGURABILE DE CALCUL
NOTIUNI INTRODUCTIVE

Abrevieri frecvente in proiectarea hardware folosind circuite reconfigurabile

ASIC - Application **S**pecific **I**ntegrated **C**ircuit
- **CLB - C**onfigurable **L**ogic **B**locks
- **CPLD - C**omplex **P**rogrammable **L**ogic **D**evices
- **FPGA - F**ield-**P**rogrammable **G**ate **A**rrays
- **HDL - H**ardware **D**escription **L**anguage
- **IOB - I**nput/**O**utput **B**locks
- **LUT - L**ook-**U**p **T**able
- **PAL - P**rogrammable **A**rray **L**ogic
- **PLA - P**rogrammable **L**ogic **A**rray
- **PLD - P**rogrammable **L**ogic **D**evices
- **SoC - S**ystem-**O**n-**C**hip
- **VHDL - V**ery **H**igh **S**peed **I**ntegrated **C**ircuit **HDL**
- **VLSI - V**ery **L**arge **S**cale **I**ntegration
- **CAD** - Computer-aided design
- **EDA** – Electronic Design Automation
- **IP** – Intellectual Property

Cursul va trata urmatoarele probleme:

Istoric, tendinte si motivatia utilizarii circuitelor reconfigurabile

•Comparatie cu alte tipuri de proiectare hardware: ASIC si microprocessor.

· Trecere in revista a diferitelor tipuri de circuite reconfigurabile. Generalitati despre dispozitivele logice programabile

· Arhitecturi circuite de tip FPGA si CPLD

• Unelte de proiectare (*Computer-Aided Design* - CAD) pentru FPGA design

·Notiuni de limbaje de descriere hardware HDL (Hardware Description Language). Comparatie Verilog si VHDL cu prezentarea avantajelor si respectiv a dezavantajelor pe care le are fiecare tip de limbaj. Prezentare a notiunilor principale de programare VHDL si Verilog.

· Prezentare firme care produc circuite reconfigurabile: Actel, Atmel, Altera si Xilinx. Studii de caz pentru diferite familii de circuite.

· Prezentare circuite de la firma Xilinx din familiile Spartan, Virtex etc.

- Notiunea de *Soft Processor* cu exemplificare core de tip open source si notiuni introductive referitoare la procesoarele soft in varianta comerciala (tip Xilinx - MicroBlaze, Xilinx – PicoBlaze, Altera - NIOS II)

· Procesoare de tip hardware implementate in circuitele reconfigurabile

· Tendinte tehnologice (circuite cu granulatie mica si respectiv cu granulatie mare)

· Securitatea fisierului de configurare (bit-stream) – protejarea la copiere ilegala, rescriere sau alte atacuri asupra circuitelor reconfigurabile dintr-un produs.

LABORATORUL:

Lucrarile practice vor folosi placi de dezvoltare de la firma Digilent Basys2 sau Xess care au in componenta circuite de la firma Xilinx din familia de circuite Spartan II, Spartan II-E si Spartan 3E. Se vor avea in vedere urmatoarele probleme:

- Familiarizarea cu programarea FPGA si folosirea placilor de dezvoltare
- Etapele parcurse de la tema de realizat pana la programarea circuitului, testare si verificare a design-ului.
- Realizarea de proiecte folosind module VHDL, scheme si ierarhii
- Folosirea programelor de simulare (ISE Simulator sau ModelSim)
- Implementarea structurii de microprocessor soft (masina de stare) PicoBlaze pe un mediu hardware reconfigurabil

Sistemele reconfigurabile sunt sisteme care folosesc componente *hardware* care se pot *adapta (reconfigura)* la nivel logic pentru a rezolva probleme *specifice.*

Motivatia folosirii circuitelor reconfigurabile:

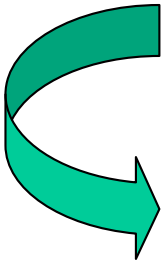
•Acceleratoare pentru realizarea aplicatiilor ce necesita calcul intensiv - în domeniul "high performance computing" (datorita **paralelismului** care este o caracteristica a unui astfel de sistem)

•Realizarea rapida a prototipurilor si a productie de serie mica

Calculatoarele Clasice sunt alcatuite din *hardware* si *software*

Procesorul este o functie fixa, determinata la momentul fabricatiei
➔hardware-ul este static
– software-ul (totalitatea programelor) este incarcat (instalat) dupa procesul de fabricare ➔ software-ul este dinamic

Sistemele de Calcul Reconfigurabile ➔ partea de hardware se poate modifica conform unei aplicatii, dupa procesul de fabricare



➔ **Hardware-ul este dinamic**

Embedded Systems

Sistemele integrate (embedded systems) includ aproape orice sistem de calcul altul decât computerele traditionale.

Reprezinta unul dintre domeniile cu cea mai rapida crestere.
Exemplele includ:

- telefoane mobile,
- dispozitive medicale,
- sisteme de alarmă,
- sistemele auto,
- playere audio portabile, etc.

Multe dintre cele mai interesante sisteme de calcul sunt sisteme de tip embedded:

- iPod-uri si iPhone-uri,
- robotii care calatoresc pe Marte,
- console pentru jocuri video

Embedded Systems

• Dispozitive cu functie fixa (de exemplu: MP3 decoder chip) realizeaza o singura functie care este definita la momentul procesului de fabricare

• Avantaj: *performante excelente pentru functia respectiva*

• Dezavantaj: nu este un dispozitiv flexibil, functia nu se poate modifica dupa procesul de fabricare.

• Dispozitive programabile (ex. ARM microprocessor)

– realizeaza orice functie care este definita dupa procesul de fabricare

•Avantaj: extrem de flexibil, se poate schimba functia de calcul in orice moment

•Dezavantaj: performante scazute, hardware-ul nu este gandit si dedicat unei functii specifice.

• Dispozitivele reconfigurabile – combina functiile fixe cu functii programabile

Avantaje: ofera *high performance ; hardware-ul se poate adapta in orice moment dupa procesul de fabricare pentru orice functie calculabila*

Scurt Istoric:

1960: G. Estrin at UCLA, “The Fixed Plus Variable Structure Computer”
1985: Firma Xilinx introduce circuitele de tip FPGA
– FPGA(field-programmable gate arrays = arii de porti logice probramabile) – sunt considerate a fi o culme a dispozitivelor logice programabile
– sunt programate prin scrierea celulelor de memorie static RAM
– volatilitatea acestora este considerata a fi un dezavantaj pentru implementare de functii logice
• Anii 90
– cercetatorii realizeaza ca volatilitatea circuitelor FPGA bazate pe SRAM este de fapt cheia pentru multe tipuri de aplicatii noi si incep sa lucreze cu sisteme de calcul bazate pe circuite de tip FPGA, aceste sisteme dezvoltate obtinand performante egale cu cele ale supercomputerelor.

In prezent:

Pentru piata de dispozitive hardware reconfigurabile

- Industria circuitelor de tip FPGA reprezinta unul dintre sectoarele cu cea mai rapidă creștere de pe piața semiconductoarelor

-Companiile noi (*start-up*) intra pe piata cu noi tipuri de dispozitive

-Sunt folosite pentru puterea lor de calcul atat in sisteme de tip *embedded* cat si pentru calculatoare de performanta inalta (*high performance)*

Ce este un FPGA?

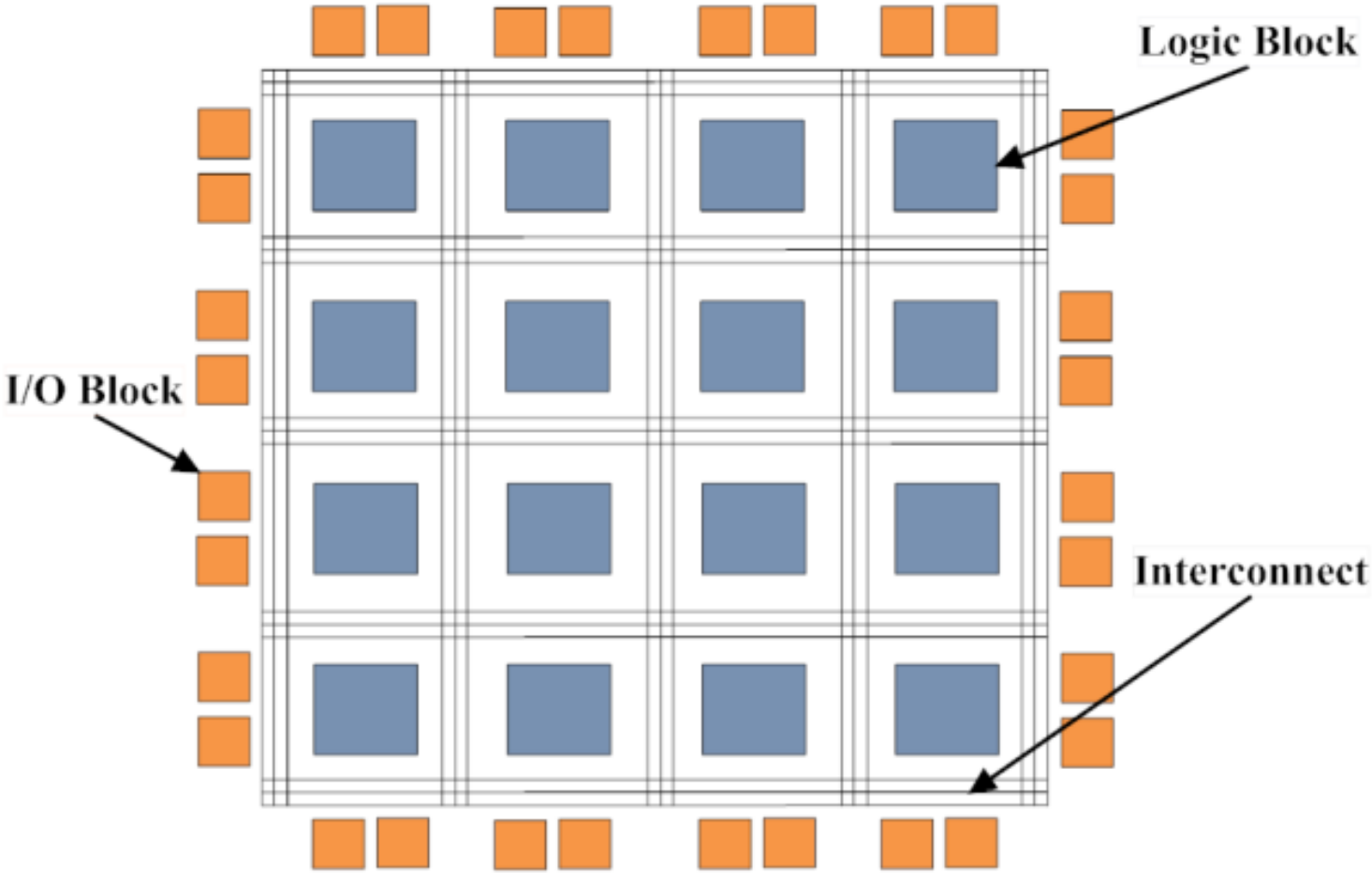
FPGA = Field-programmable gate array = Retea de porti logice programabile

Este un circuit reconfigurabil cu granulatie fină (a fine-grained reconfigurable fabric)

Este folosit în mai multe tipuri aplicatii, cum ar fi prototipizare si testare precum si ca accelarator software.

Este realizat din mai multe componente:
•blocuri logice – circuit de baza multiplicat in aria programabila
•blocuri I /O,
•canale de rutare (interconectare) si comutatoare programabile care realizeaza conectarea elementelor de logica la firele de legatura sau a firelor de legatura intre ele.
•mai multe blocuri de "hard". (sumatoare, memorii RAM, inmultitoare)

Reprezentare arhitectura FPGA



•Pentru a programa circuitul se foloseste o descriere hardware printr-un limbaj specific (HDL = hardware description language) sau o descriere la nivel de schema.

•Exemple de limbaje HDL >> Verilog si VHDL

•Se folosesc medii de dezvoltare/programare in functie de fabricantul de circuite (ex. Xilinx, Altera, etc)

Fluxul de proiectare:

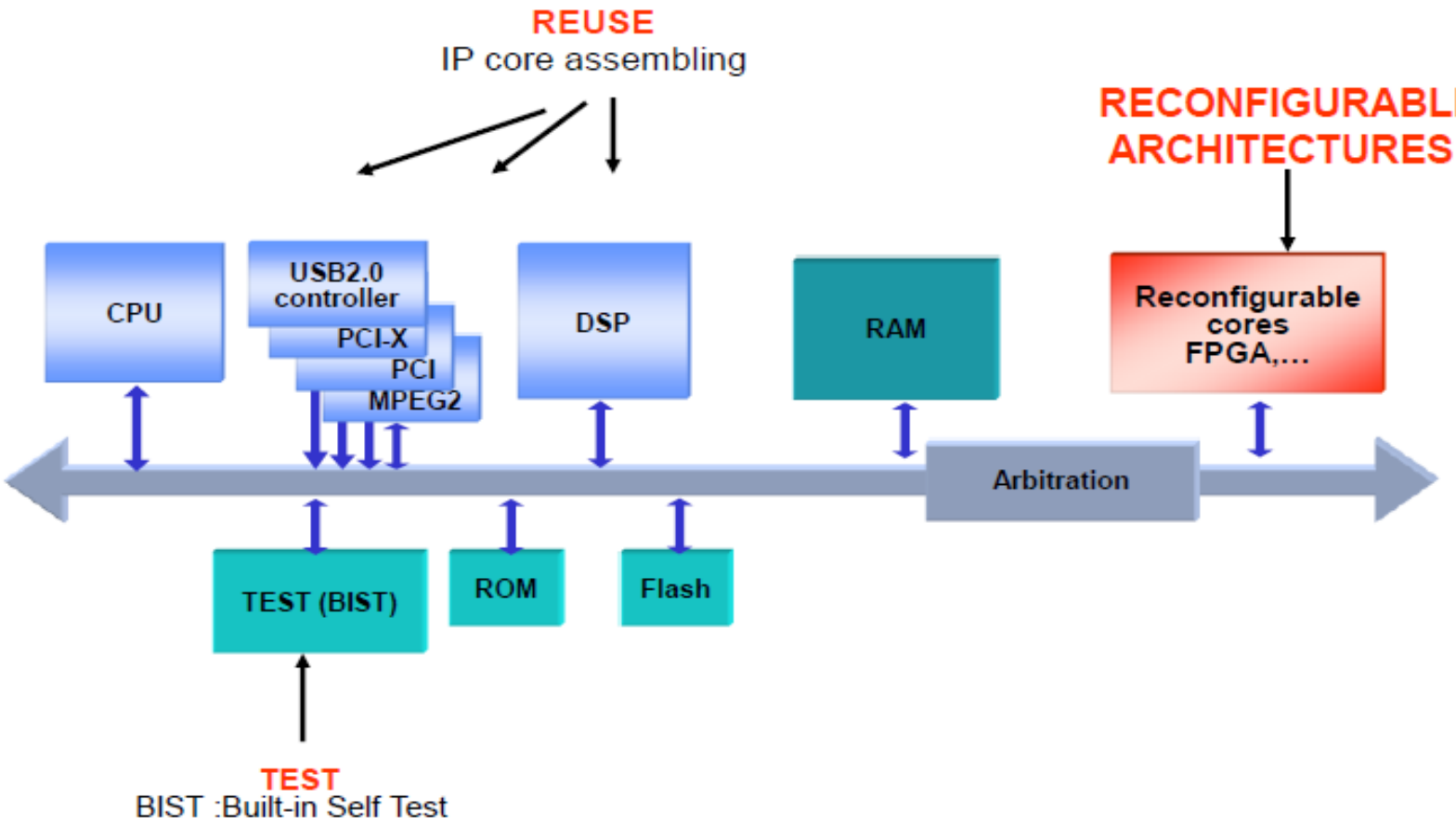
•Sinteza logicii – procesul de abstractizare a codului HDL in termeni de porti logice (se face si optimizarea in ceea ce priveste numarul de porti logice, aria si timpii de intarziere)

•Implementarea

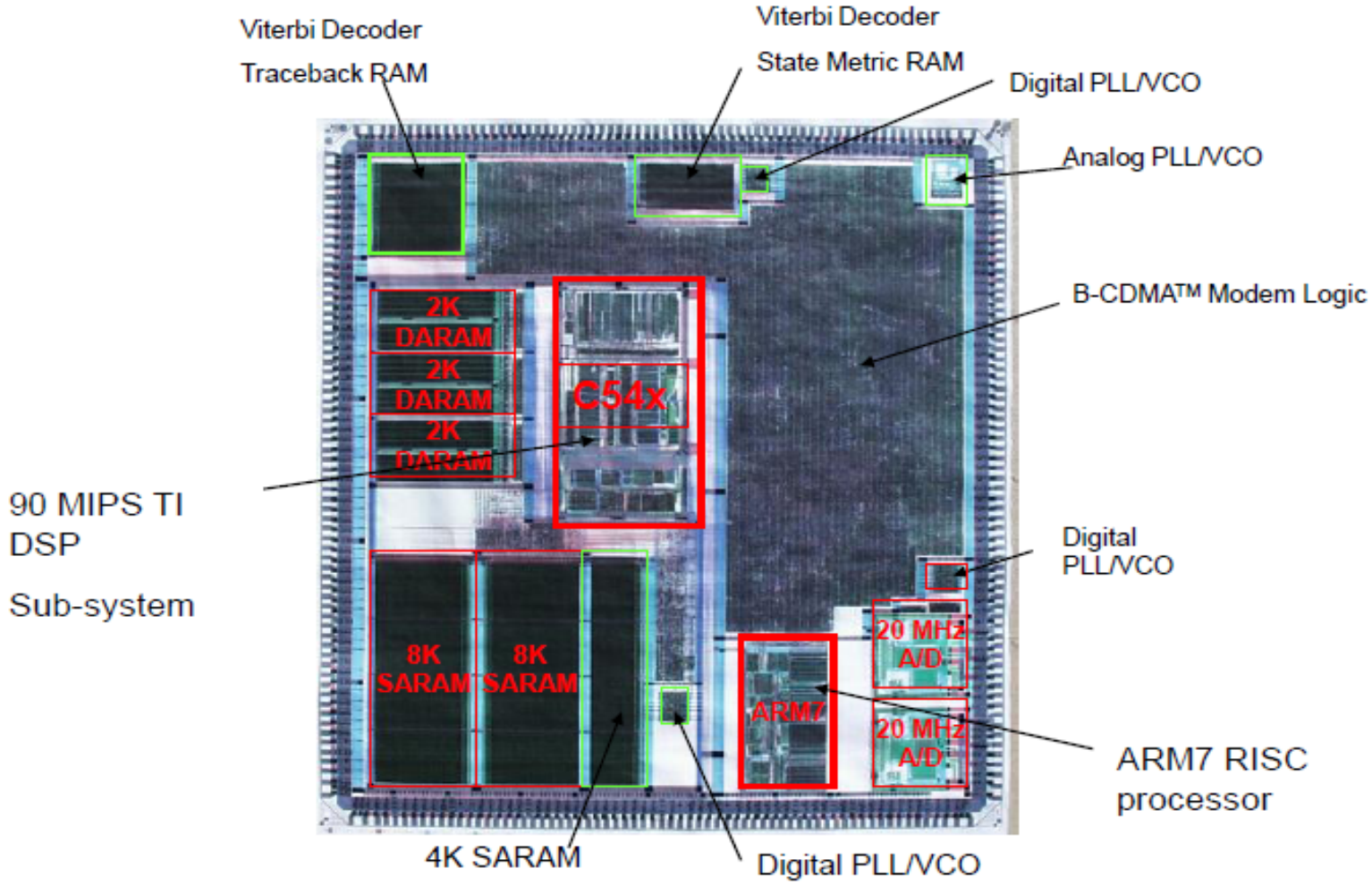
- Translatarea (intrarile, iesirile si constrangerile se pun intr-o structura logica)
- Maparea - se preia structura logica si se realizeaza primitive de low-level pentru pasul urmator (plasare si rutare)
- Plasarea si rutarea – se plaseaza in blocurile logice configurabile din circuit. Se realizeaza interconexiunile folosind canalele de rutare

•Realizare fisier de configurare (bitstream) care va programa circuitul FPGA

Platform based design



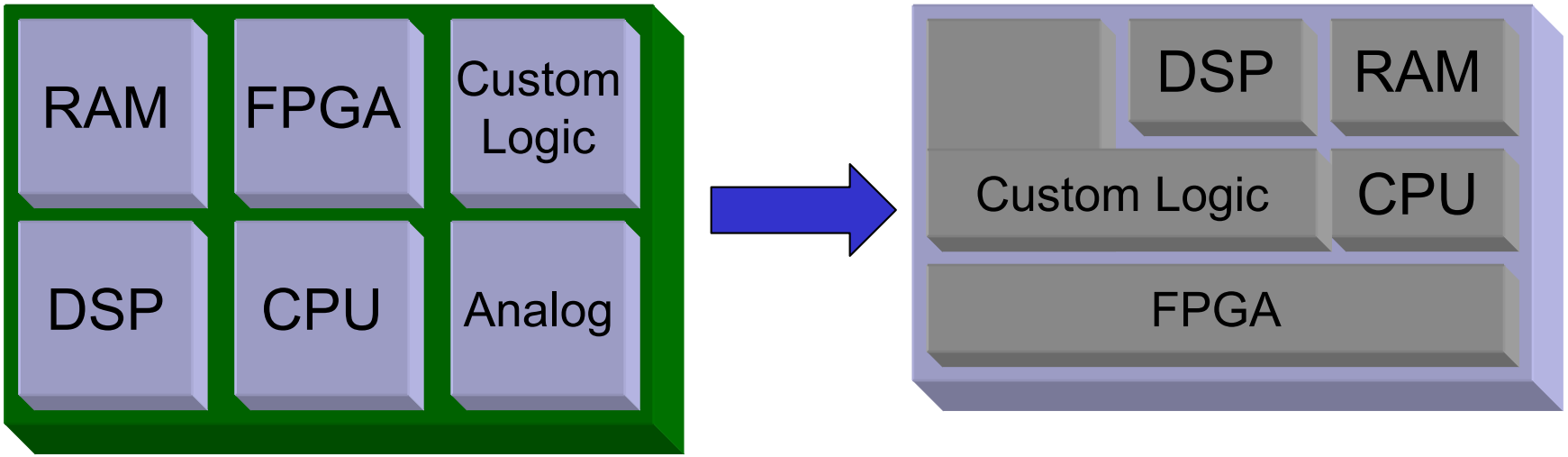
Platform based design :SoC example



B-CDMA(TM) SoC Layout

Logica Reconfigurabila (FPGA) in **System-On-A-Chip (SoC)**

SoC = un circuit integrat in care se afla incapsulate componentele de baza necesare pentru a defini un sistem de calcul. (intr-un singur circuit).



SoC Vs. Microcontroller Vs. Processor

- All implemented on a single chip package, differences include:
- Processor (CPU)
  - Is a single processor core;
  - Normally can be used for general purpose, but needs to be supported with Memories and IOs;
- Microcontroller (MCU)
  - Typically has a single processor core;
  - Has Memory blocks, basic IOs and other basic peripherals;
  - Mainly used for basic control purpose, such as embedded applications;

ARM-University Program

SoC Vs. Microcontroller Vs. Processor

- SoC – System on Chip
  - Can have a single or multiple powerful processor cores;
  - Has larger Memory blocks, a variety of IOs, and other peripherals;
  - Normally integrated with more powerful blocks, e.g. GPU, DSP, video/ audio encoder/ decoder;
  - Usually capable of running operating systems, e.g. Windows, Linux, iOS and Android;
  - Mainly used for advanced applications, such as the main chip of a digital device (smart phones, tablets).

ARM-University Program

Commercialized SoCs

- Benefiting from its power efficiency, SoCs have been widely used in mobile devices, such as smartphones, tablets and digital cameras.

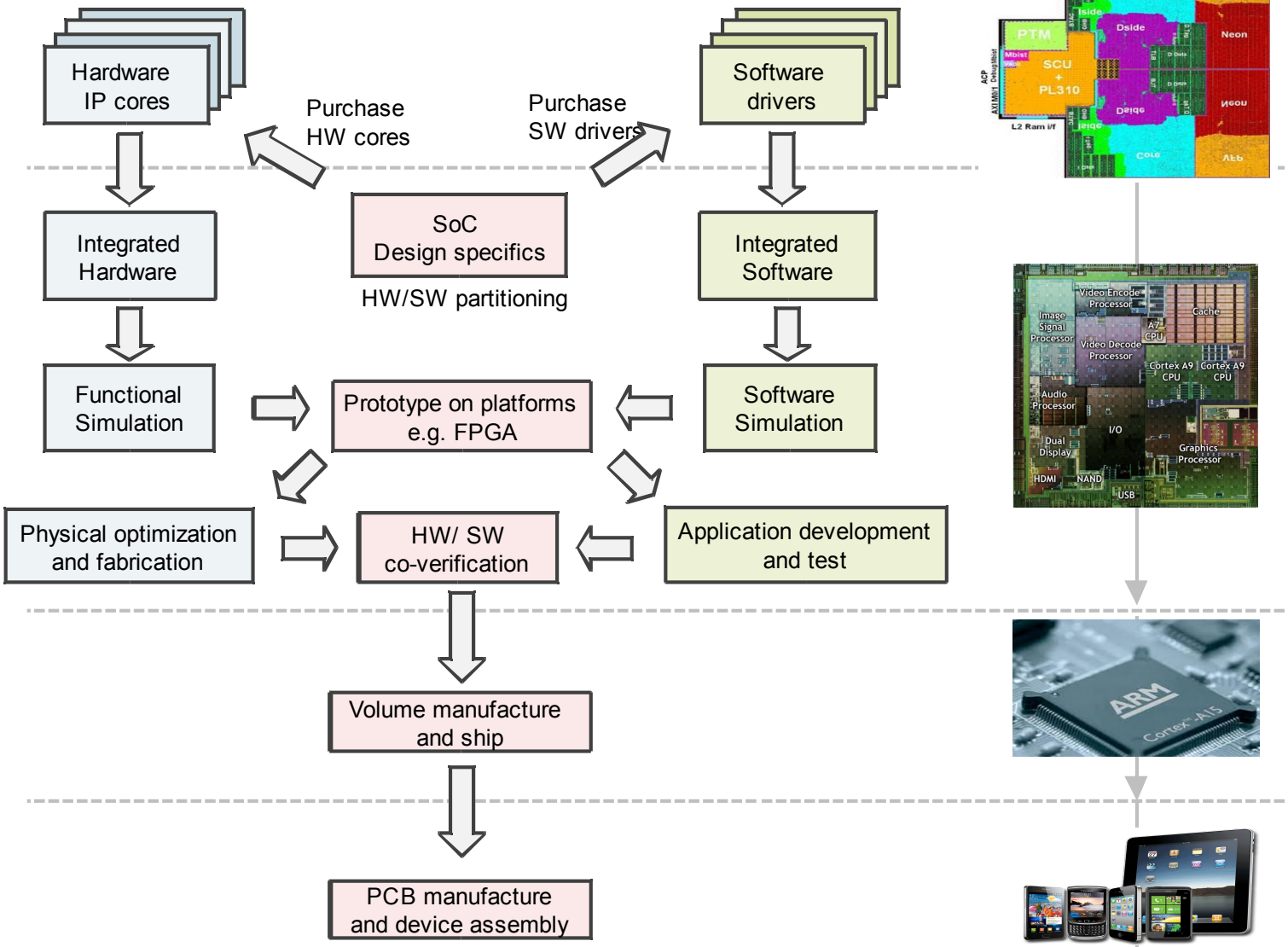
- A number of SoCs have been developed by a large eco-system of design companies, eg-, SnapdragonTM by Qualcomm®, Tegra® by Nvidia®, Ax by Apple®, OMAPTM by Texas Instruments, etc…

- Most mobile SoCs use ARM-based microprocessors since they deliver high performance with less power consumption.

ARM-University Program

SoC Design Flow

ARM-University Program



IP Vendors:
core design

Fabless Vendors:
SoC design

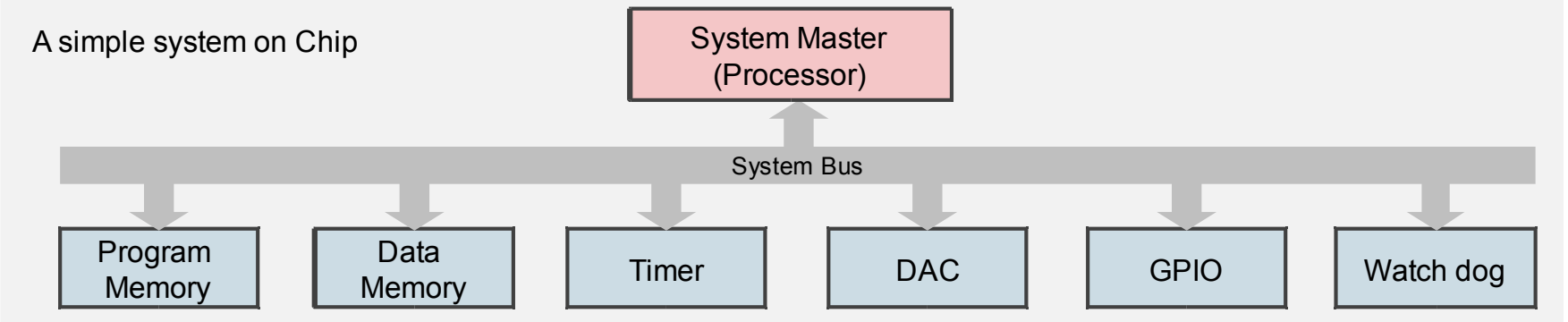
Foundries:
Chip fabrication

Device vendors:
Final products

What is inside of a SoC

ARM-University Program

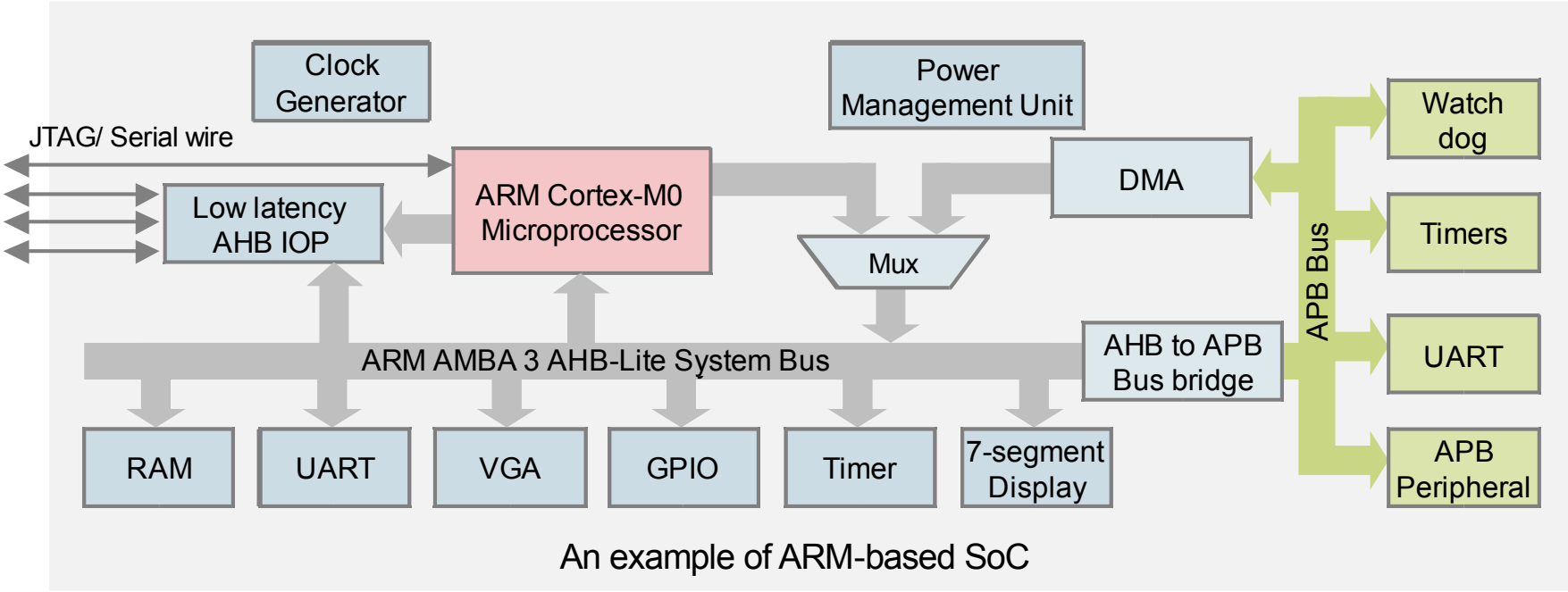
- The basic components of a SoC include:
  - A system master, such as a microprocessor or DSP;
  - System peripherals, such as Memory block, timer, external digital/ analog interfaces;
  - A system bus that connects master and peripherals together using a specific bus protocol.
- More sophisticated modules are integrated in modern SoCs, such as multicores, DSPs, GPUs, and multiple buses connected by bus bridges.



ARM-based SoC

ARM-University Program

- An basic ARM-based SoC usually consists of
  - An ARM processor, such as Cortex-M0;
  - Advanced Microcontroller Bus Architecture (AMBA), e.g. AMBA3 or AMBA4;
  - Physical IPs (or peripherals) from ARM or third parties;
  - Additionally, some SoCs may have a more advanced architecture, such as multi-bus system with bus bridge, DMA engine, clock and power management, etc…

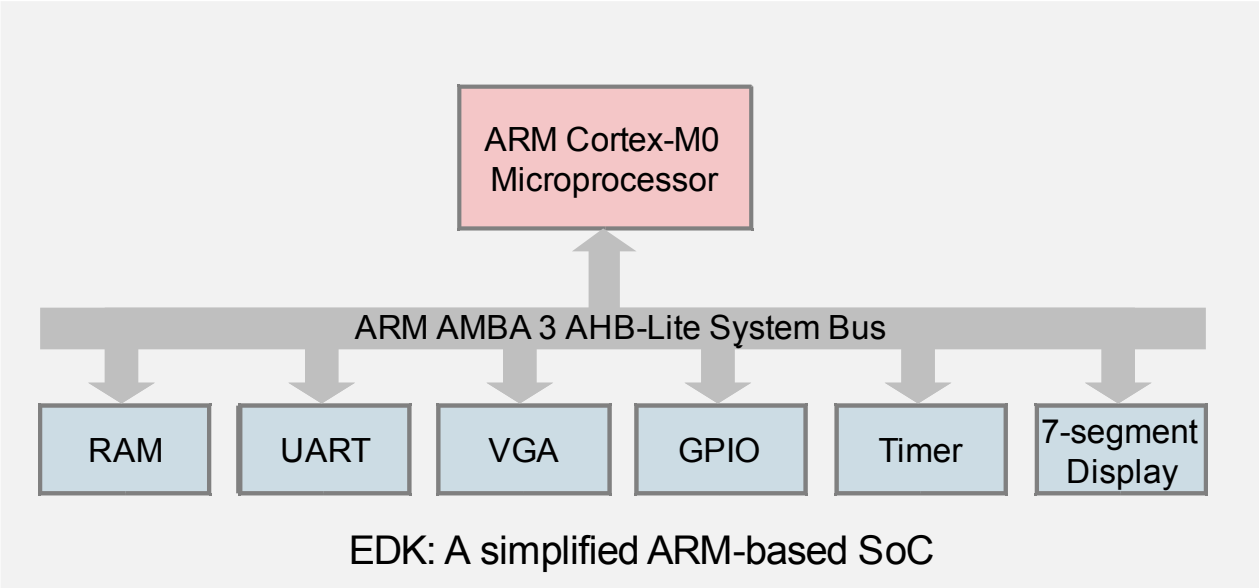


An example of ARM-based SoC

Design a Simple ARM-based SoC

- In this set of teaching materials we are going to design a simplified version of ARM-based SoC and prototype it onto a FPGA chip. The SoC will consist only some basic components:
  - An ARM Cortex-M0 microprocessor;
  - A single AHB Lite bus;
  - Customer-made physical IPs;

ARM-University Program



EDK: A simplified ARM-based SoC

Tehnici dezvoltare proiecte hardware:

- **Microcontroller**
  - **Avantaje:**
    - **Cost redus**
    - **Usor de folosit**
    - **Nu sunt orientate catre o aplicatie specifica**
  - **Dezavantaje**
    - **Lucreaza secvential**
    - **Opereaza cu date de latimi fixe**
- **ASIC**
  - **Avantaje**
    - **Performante foarte bune**
    - **Dezvoltate pentru productie de serie mare**

  Dezavantaje:
    - **Cost mare de realizare si proiectare**
    - **Durata mare pana la finalizare**

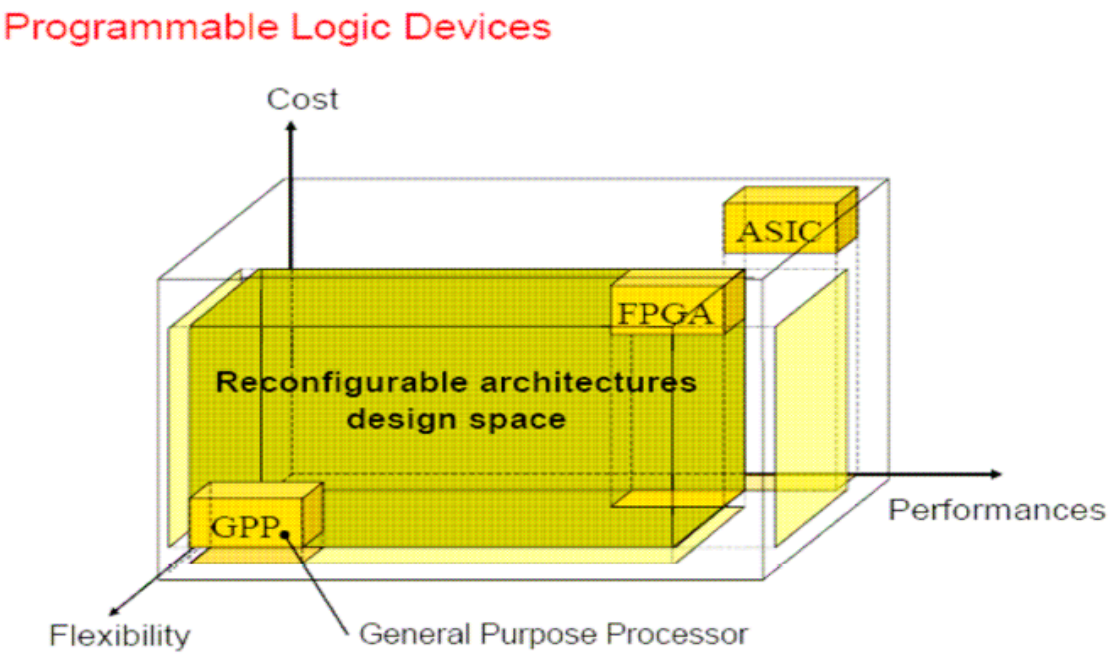
ASIC (application-specific integrated circuit)

Un ASIC (circuit integrat specific aplicaţiei) este un microcip proiectat pentru o aplicaţie specială, cum ar fi un anumit tip de protocol de transmisie. In contrast cu circuitele integrate generale, cum ar fi microprocesorul şi cipurile de memorie cu acces aleator , ASIC-urile sunt folosite într-o gamă largă de aplicaţii, inclusiv de control al emisiilor auto, monitorizarea mediului, şi asistenţi personali digitali (PDA-uri).

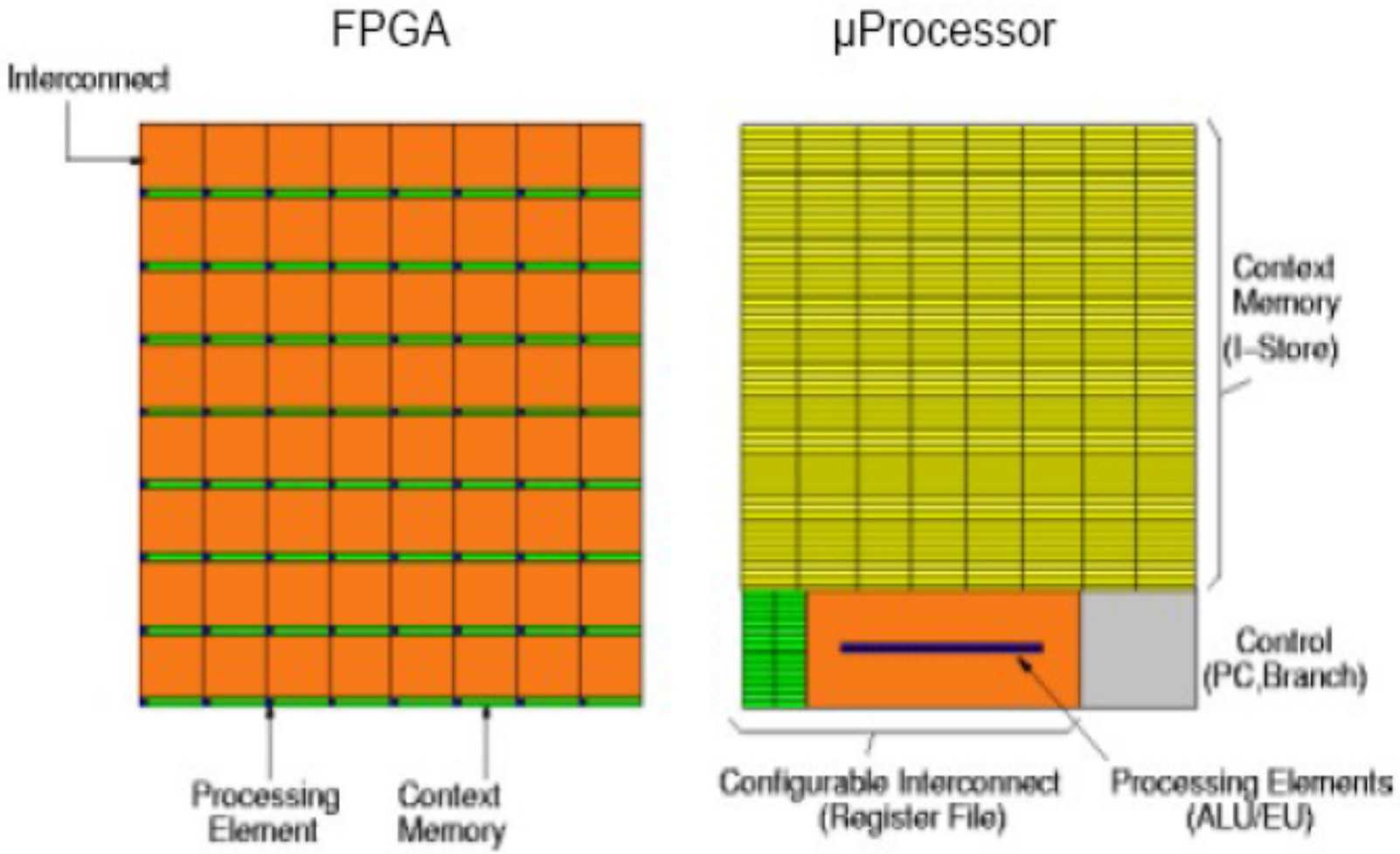
ASSP (application-specific standard product)

Este un dispozitiv semiconductor cu circuite integrate (IC), pentru o aplicaţie specifică şi vândut la mai mult de un utilizator (şi, prin urmare, "standard"). Ca un ASIC (circuit integrat de aplicatii specifice), ASSP este pentru o aplicaţie specială, dar acesta este vândut către orice număr de companii. (Un ASIC este proiectat şi construit la comanda pentru o companie anume.)

- **FPGA**
  - **Avantaje:**
    - **lucreaza paralel – se pot implementa sisteme de calcul de mari performante – High Performance Computing**
    - **Foarte bune pentru prototipizare si productie de serie mica**
  - **Dezavantaje: cost mediu spre mare**

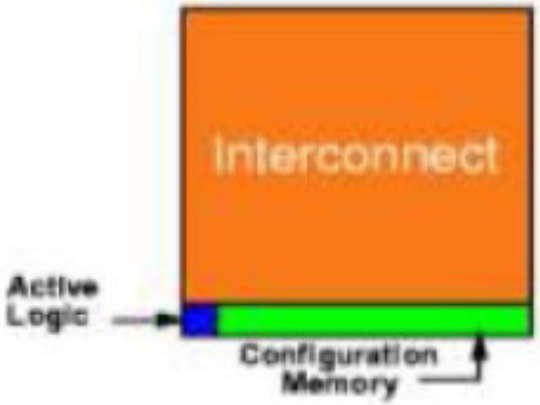


FPGAs : Area comparison (vs. processors)



FPGAs : Area comparison

Total Area ~ Active logic + Configuration memory + interconnect



Cost of the use of a reconfigurable technology instead of standard logic

Function	Area (λ^2)	
LUT MUX + ff	20K	(generous, closer to 10K)
Programming Memory	80K	(240K typical unencoded)
Interconnect	700K	(for $N_p = 2048$)

Principalii producatori de circuite de tip FPGA si CPLD sunt Xilinx, Altera, Lattice, Atmel, Actel,,.

www.**xilinx**.com/ - *Xilinx*

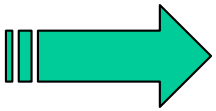
www.**altera**.com/ - *Altera*

www.**latticesemi**.com/ - Lattice

www.**actel**.com/ - *Actel*

www.**atmel**.com/ - *Atmel*

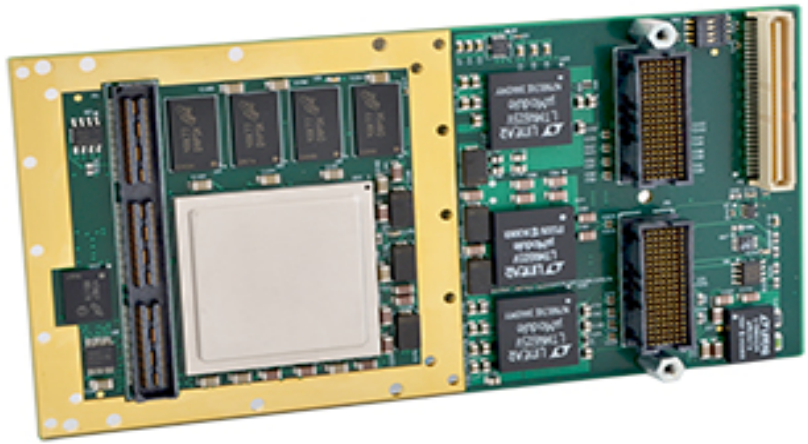
Exista firme care folosesc in produsele lor comerciale circuite FPGA



Exemplu

High-performance computing - https://www.acromag.com

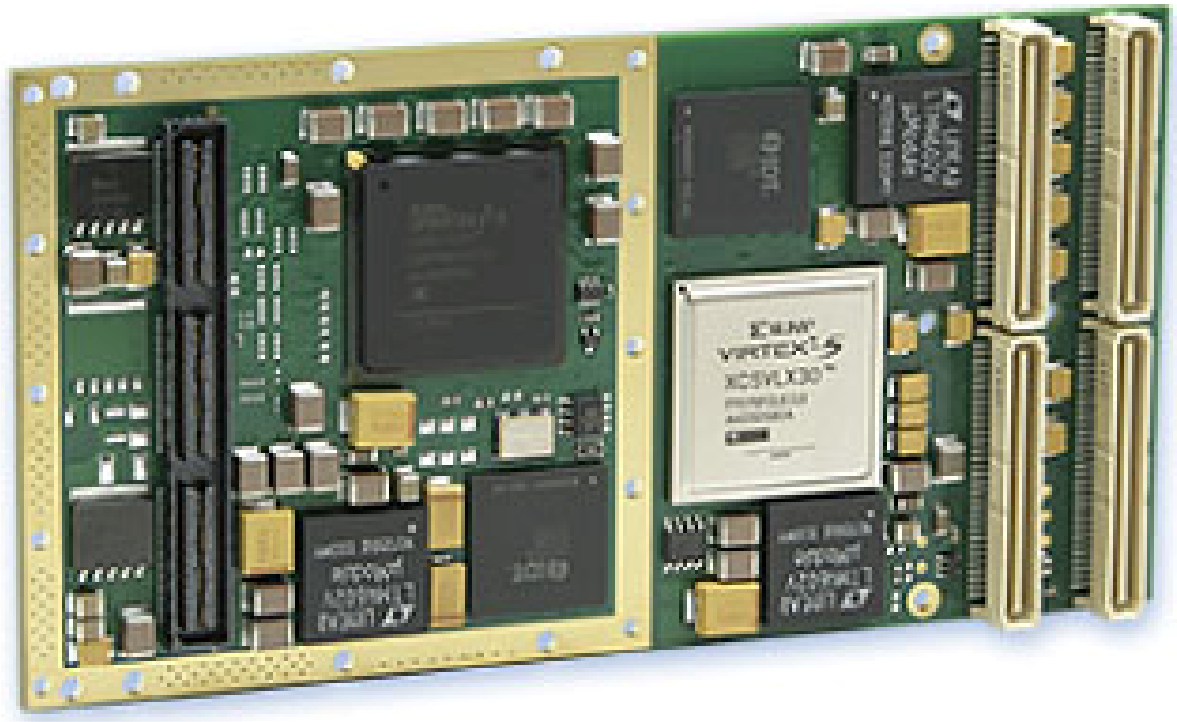
Acromag's **high-performance XMC FPGA modules** feature a user-customizable Artix®-7, Xilinx® Kintex-7, Virtex-6, Virtex-5 or Spartan-6 Xilinx FPGA. These modules allow you to develop and store your own instruction sets in the FPGA for a variety of adaptive computing applications. Select from several models with up to 410K logic cells optimized for logic, DSP, or PowerPC. The DDR3 SDRAM and a PCI Express interface rapidly move data.



With Acromag's Artix-7 FPGA modules, you can greatly **increase DSP algorithm performance for faster throughput using multiple channels and parallel hardware architectures**. Free up CPU cycles by offloading algorithmic-intensive tasks to the **FPGA co-processor**.

These modules are ideal for **high-performance** customized embedded systems. Optimize your system performance by integrating high speed programmable logic with the flexibility of software running on **MicroBlaze™ soft processors**.

Acromag's Engineering Design Kit provides software utilities and example VHDL code to simplify your program development and get you running quickly. A JTAG interface enables on-board debugging. Additional Xilinx tools help finish your system faster. Maximize FPGA performance with Vivado® or ISE® Design Suite. And with ChipScope™ Pro tools, you can rapidly debug logic and serial interfaces



Although there is no limit to the uses for Acromag's FPGA I/O boards, several applications are ideal for this new technology. Common uses include **sonar, radar, hardware simulation, automated test equipment, protocol conversion, in-circuit diagnostics, military servers, telecommunication, and digital signal processing**. Typical applications for conduction-cooled models include manned and unmanned vehicles, battleground SIGINT and communications systems, deployment on tanks, or any system where ambient or forced air can't provide adequate cooling.

Site firma Xilinx – prezentare companie

Facts At-a-Glance

- Founded 1984
- ~3,000 employees worldwide
- 20,000 customers worldwide
- 2,000+ patents
- Inventor of the field programmable gate array (FPGA)
- Pioneer of fabless manufacturing model
- Headquarters in San Jose, Dublin and Singapore
- Publicly traded company since 1990 (NASDAQ: XLNX)
- $1.7B in revenues in calendar year 2009

Programmability: An Imperative for Today's Leading Global Electronics Companies

More than at any other time, global economics favor programmable chips over costly application-specific integrated circuits (ASICs) and application-specific standard products (ASSPs). In today's challenging economic climate, programmable platforms have become strategically essential for world-class system companies to effectively compete. The costs and risks associated with application-specific devices can only be justified for a short list of ultra-high volume commodity products.

Programmable platforms have become the only viable means for today's companies to meet increasingly stringent product requirements – cost, power, performance, and density – in a business environment characterized by spiraling complexity, shrinking market windows, fickle market demands, capped engineering budgets, escalating ASIC and ASSP non-recurring engineering costs, and increased risk.

For Xilinx, the programmable imperative represents a two-fold commitment. First, to increase performance, densities and system-level functionality, while driving down cost and power consumption, at each manufacturing process node with every new generation of FPGAs. Secondly, to provide simpler, smarter programmable platforms and design methodologies that free up engineers to focus on end product innovation and differentiation.

FEATURED Devices XILINX

[Virtex-6 FPGAs]

Lowest Power High-Performance FPGAs

- Breakthrough price/ performance value
- Most flexibility, high-bandwidth I/O
- Cost reduction path with EasyPath™-6 FPGAs

[Spartan-6 FPGAs]

Lowest Cost, Lowest Power FPGAs

- Breakthrough low cost with transceivers
- Revolutionary feature enhancements
- Lowest power with small packaging

[EasyPath-6 FPGAs]

Fast Cost Reduction for Virtex-6 FPGAs

- Conversion-free
- Lowest total product cost
- Identical functionality with Virtex®-6 FPGAs

FEATURE Comparison Table

Features	Virtex-6	Virtex-5	Spartan-6	Extended Spartan-3A
Logic Cells	Up to 760,000	Up to 330,000	Up to 150,000	Up to 53,000
User I/Os	Up to 1200	Up to 1200	Up to 576	Up to 519 I/O
I/O Standards Supported	Over 40	Over 40	Over 40	Over 20
Clock Management Technology	PLL	DCM + PLL	DCM + PLL	DCM
Embedded Block RAM	Up to 38 Mbits	Up to 18 Mbits	Up to 4.8 Mbits	Up to 1.8 Mbits
Embedded Multipliers for DSP	Yes (25 x 18 MAC)	Yes (25 x 18 MAC)	Yes (18 x 18 MAC)	Yes (18 x 18 MAC)
Multi-Gigabit High Speed Serial	6.5 Gbps, beyond 11 Gbps	3.75 Gbps, 6.5 Gbps	3.125 Gbps	No
PCI Express® Technology	Gen 1, x8, hard Gen 2, x8, hard	Gen 1, x8, hard Gen 2, x8, soft	Gen 1, x1, hard	No
MicroBlaze™ Soft Processor	Yes	Yes	Yes	Yes

Pentru aplicatii:

Xcell Journal

Quarterly programmable logic magazine

- See how your peers employ FPGAs in their designs
- Learn new design techniques and tool tricks
- Latest application notes, tool and IP releases

In 2009, Xilinx introduced the Virtex®-6 family of 40nm FPGAs for compute intensive, high-speed, high-density SoC applications, and the Spartan®-6 family of 45nm FPGAs for applications where size, power, and cost are key considerations.

Each family serves as the foundation for programmable targeted design platforms that enable software and hardware designers alike to leverage open standards, common design methodologies, development tools, and run-time platforms. Xilinx targeted design platforms integrate five key elements, delivered by Xilinx and its robust network of ecosystem partners, and supported by field applications engineering and design services teams with in-depth expertise in domain and market-specific applications.

- Xilinx Virtex®-6 and Spartan®-6 FPGAs
- Design environments supporting and integrating industry-proven methodologies
- Scalable boards and kits adopting the industry standard FPGA mezzanine connector (FMC)
- Socketable IP cores
- Robust reference designs

Markets Served

Xilinx solutions enable the world's most innovative applications throughout a broad set of end markets:

- Aerospace/Defense
- Automotive
- Broadcast
- Consumer
- Data Process/Storage
- Industrial/Scientific/Medical (ISM)
- Wired
- Wireless

Modern FPGA

VIRTEX 5

- 65nm technology, 40-nm gate length (Poly)
- 1.6nm oxide thickness (16 Angstrom)
  - ~5 atomic layers
- Triple-Oxide Technology
  - 3 oxide thicknesses for optimum power and performance
- 1.0 Vcc core
  - Lower dynamic power
- 12 layer copper
- Strained silicon transistor
  - Maximum performance at lowest AC power

Over 1 Billion Transistors

Source: MPSOC'06 Keynote, Ivo Bolsen, Xilinx

Today's Programmable Imperative

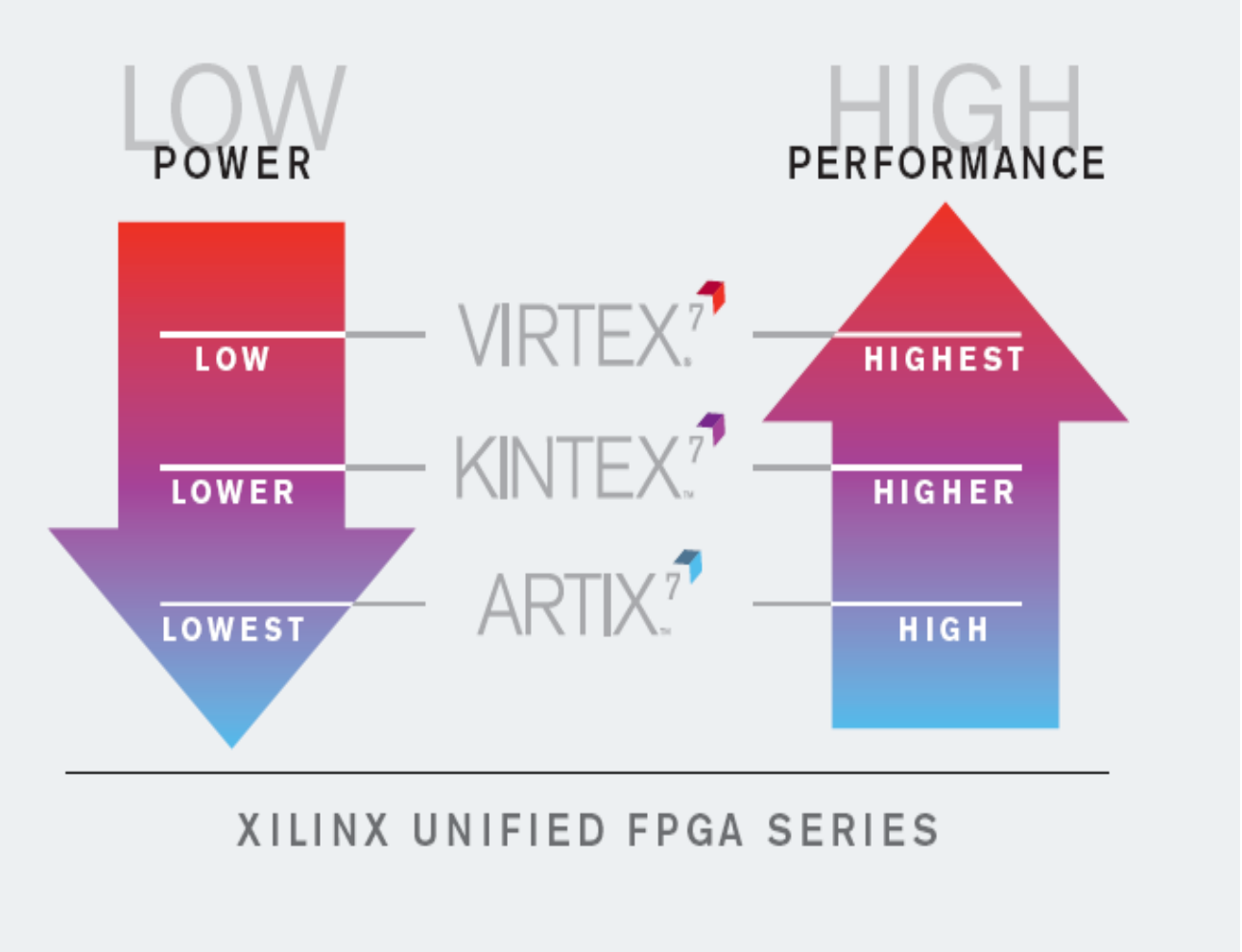
- Industry dynamics are driving the need for systems that consume much less power
- Insatiable bandwidth requirements call for higher system-level performance
- Competitive pressure is forcing customers to explore new options for managing performance/cost tradeoffs
- Companies are struggling to increase productivity without sacrificing innovation and differentiation

Xilinx 7 Series FPGAs: Addressing the Programmable Imperative

- Cutting power consumption in half, allowing FPGAs to be used in new applications and providing more "useable performance"
- Raising system-level performance by setting new benchmarks in logic density, I/O bandwidth, and signal processing
- Providing unmatched performance per dollar
- Offering a unified architecture that reduces customers development time and enables innovation and differentiation

http://www.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf

XILINX 7 SERIES FPGAS OFFERS BREAKTHROUGH POWER, PERFORMANCE AND DRAMATICALLY REDUCED DEVELOPMENT TIME



http://www.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf

7 SERIES FPGA FAMILY COMPARISON

MAXIMUM CAPABILITY	ARTIX-7 FAMILY	KINTEX-7 FAMILY	VIRTEX-7 FAMILY
Logic Cells	352K	478K	1,955K
Block RAM	12Mb	34Mb	65Mb
DSP Slices	700	1,920	3,960
Peak DSP Performance (symmetric FIR)	504 GMACS	2,450 GMACS	5,053 GMACS
Transceiver Count	4	16	88
Peak Transceiver Speed	3.75Gbps	10.3125Gbps	28.05Gbps
Peak Serial Bandwidth (full duplex)	30Gbps	800Gbps	2,784Gbps
PCI Express® Interface	Gen1 x4	Gen2 x8	Gen3 x8
Memory Interface	1,066Mbps	2,133Mbps	2,133Mbps
I/O Pins	450	500	1,200
I/O Voltage	1.2V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V
Packaging Options	Low cost wire bond	Low cost lidless flip chip and High performance flip chip	Highest performance flip chip
Target Applications	▪ Handheld portable ultrasound ▪ Digital SLR lens control module ▪ Software defined radio	▪ Wireless LTE infrastructure ▪ 10G PON OLT line card ▪ LED backlit and 3D video displays ▪ Medical imaging ▪ Avionics imaging ▪ Video-over-IP	▪ 400G and 100G line cards ▪ 100G OTN muxponder ▪ 100GE line card ▪ 300G bridge ▪ Terabit switch fabric ▪ RADAR ▪ ASIC emulation ▪ High-performance computing ▪ Test and measurement

Prezentare site Altera:

Altera Corporation (NASDAQ: ALTR) is the pioneer of programmable logic solutions, enabling system and semiconductor companies to rapidly and cost effectively innovate, differentiate, and win in their markets. Altera offers FPGAs, CPLDs, and ASICs in combination with software tools, intellectual property, and customer support to provide high-value programmable solutions to over 12,000 customers worldwide. Altera was founded in 1983 and has annual revenues in 2009 of US$1.20 billion. Altera is headquartered in San Jose, California, and employs approximately 2,600 people in 19 countries.

Forging the next evolution in electronic design, Altera® reprogrammable solutions deliver fast time to market and an advantage over costly, high-risk ASIC development and inflexible ASSPs and digital signal processors. Altera offers value to a much broader market than was previously addressed by programmable logic products.

By maintaining strong, long-standing partnerships with industry-leading technology suppliers, such as foundry partner Taiwan Semiconductor Manufacturing Company (TSMC), Altera customers are assured of quality and on-time delivery. Altera enhances its own place-and-route design software with tools from best-in-class EDA vendors. With the assistance of a world-class distribution network, Altera services customers around the world. This highly successful business model allows Altera to focus on its core competency: the development and deployment of leading-edge programmable technology that provides maximum value to customers.

Altera FPGAs

Altera offers customers a broad spectrum of FPGA solutions, geared towards diverse markets and applications. Altera's main FPGA series are listed in Table 1.

Table 1. Altera FPGA Series

Cyclone FPGAs	Arria FPGAs	Stratix FPGAs
You'll find the architecture of a low-cost Cyclone® series FPGA is ideal for high-volume, cost-sensitive applications. You can use a Cyclone FPGA alone or as a digital signal processor and can also provide a cost-effective embedded processing solution. A Cyclone FPGA offers a wide range of density, memory, embedded multiplier, and packaging options in a customer-defined FPGA feature set optimized for low-cost applications. You are likely to use a Cyclone series FPGA in a low-cost, power-sensitive automotive or consumer solution. ▶Learn more	An Arria® series FPGA is optimized for cost- and power-sensitive transceiver-based applications. An Arria FPGA has a rich feature set of functions (memory, logic, and DSP) combined with superior signal integrity. An Arria series FPGA features on-chip transceivers that allow you to integrate more functions and maximize system bandwidth (up to 16 transceivers supporting 3.75 Gbps). Because it is cost-optimized, an Arria series FPGA is more likely to be used to keep overall system costs low while meeting the digital signal processing (DSP) needs of new wireless standards such as 3G and long-term evolution (LTE). ▶Learn more	A Stratix® series FPGA is a high-density device that enables you to deliver high-performance, state-of-the-art products to market faster with lower risk and higher productivity. The latest Stratix series FPGA includes on-chip transceivers (up to 24 transceivers supporting 11.3 Gbps) that allow the transfer of data in and out of the FPGA at high frequencies. A Stratix series FPGA also simplifies the challenges of signal integrity by providing transceivers with best-in-class jitter characteristics. You'll often find a power and performance optimized high-density Stratix series FPGA at work in broadcast applications. ▶Learn more

http://www.altera.com/products/fpga.html

An Altera FPGA Provides Solutions for Many Market Segments and Applications

An Altera® Cyclone, Arria, and Stratix FPGA meets many of the requirements necessary for your next-generation FPGA design. These FPGAs provide market solutions and are a key element in designing for the following end market and application areas, as shown in Table 2.

Table 2. Altera End Market and Application Areas

End Markets		General Applications
• Automotive • Broadcast • Computer and Storage • Consumer • Display • Industrial	• Medical • Military • Test and Measurement • Wireless • Wireline	• Digital signal processing (DSP) • Embedded processing • ASIC prototyping • Memory interfaces

Altera Devices

Home > Products > Devices

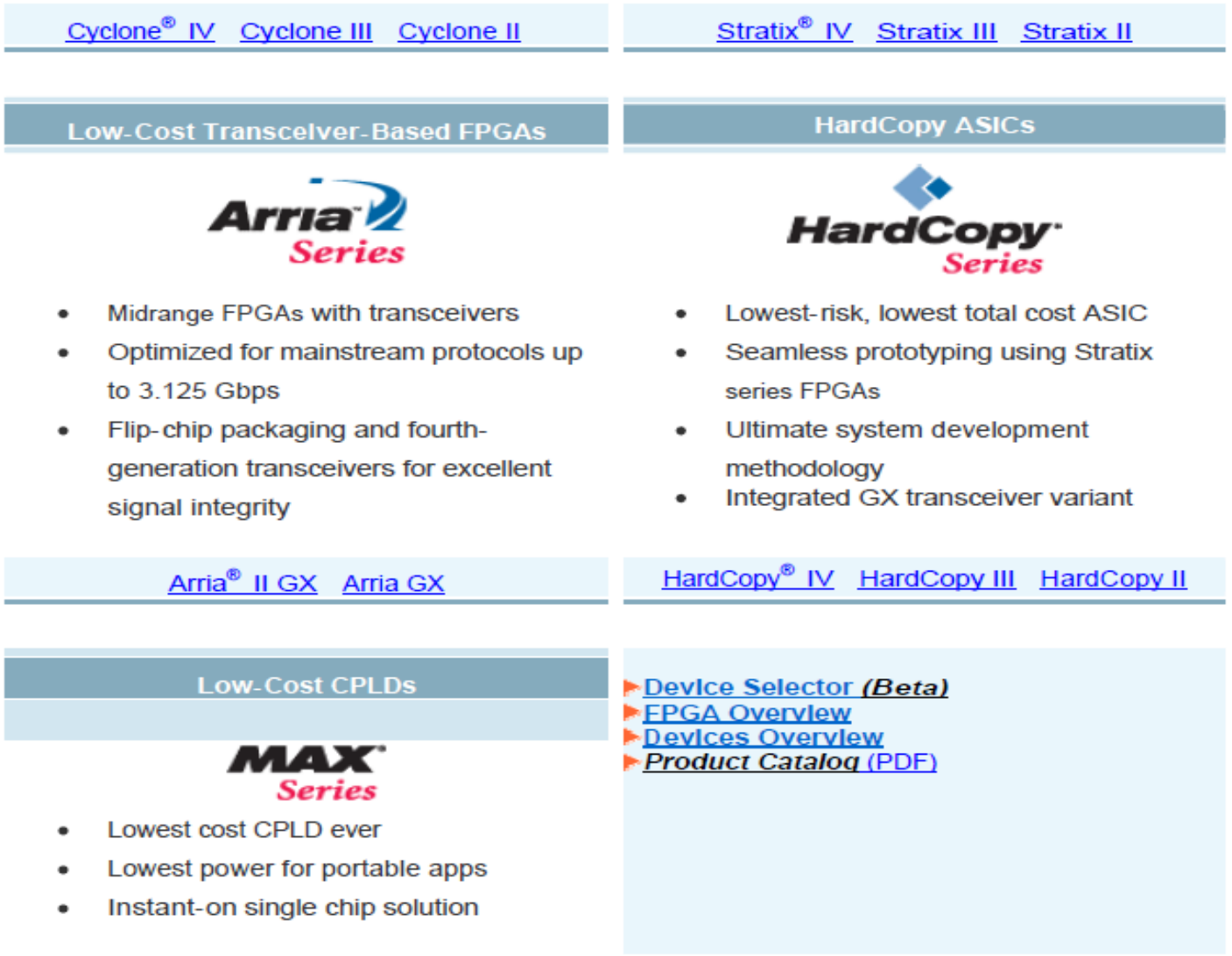
Low-Cost FPGAs	High-End FPGAs
Cyclone Series	Stratix Series
• Built from the ground up for low cost • Low power consumption • Integrated GX transceivers variant	• High-density, high-end FPGAs • Integrated GX transceiver variant • Design entire systems-on-a-chip

- Cyclone® V FPGAs
- Cyclone IV FPGAs
- Cyclone III FPGAs
- HardCopy® V ASICs
- MAX® V CPLDs
- MAX II CPLDs

- Stratix® V FPGAs
- Stratix IV FPGAs
- Stratix III FPGAs

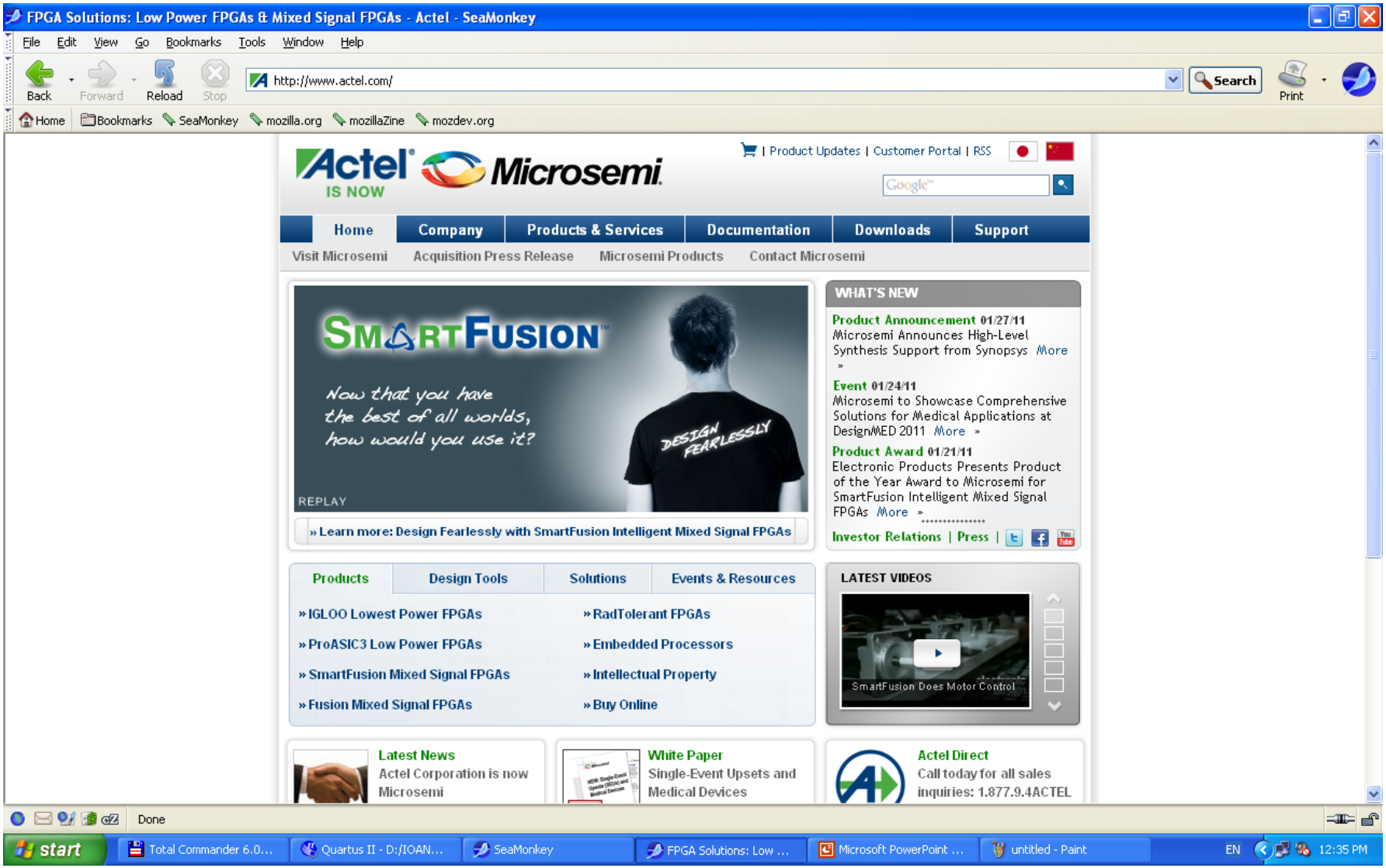
- Arria® V FPGAs
- Arria II FPGAs
- Arria GX FPGAs

- MAX® V CPLDs
- MAX II CPLDs



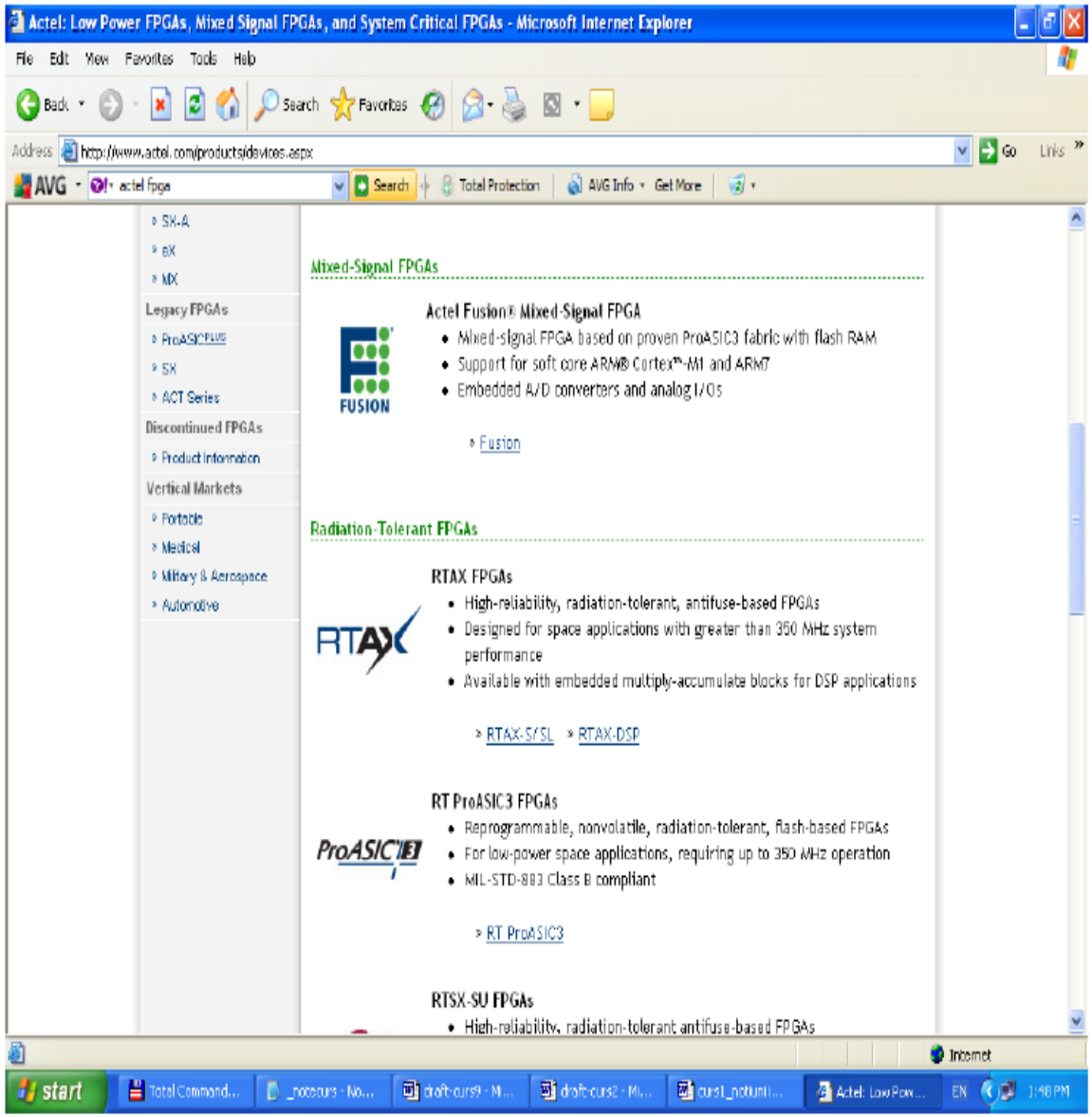

Automotive - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.altera.com/end-markets/auto/aut-index.html
Go Links
AVG Search Total Protection AVG Info Get More
ALTERA
Download Center Literature
Sign In/register myAltera Account
Products End Markets Technology Training Support About Altera Buy Online
Search
Automotive
Broadcast
Computer & Storage
Consumer
Industrial
Medical
Military
Test & Measurement
Wireless
Wireline
Automotive Quality
Standards & Qualifications
Automotive Resources
Consortia
Customer Successes
Glossary of Key Terms
Questions & Answers
Literature
Development Kits
The automotive industry is using programmable logic devices (PLDs) after discovering how they can contribute to the success of both system suppliers and automakers (original equipment manufacturers). With best-in-class quality, cost structures suitable for high-volume consumer applications, and increasing system performance, PLDs are entering the automotive market faster than any other semiconductor category. Unlike ASSP or microcontroller solutions, PLDs provide the flexibility and faster time-to-market that is increasingly becoming a key requirement in the automotive industry.
PLDs, which have already found a home in the automotive infotainment and
Next Steps
Get Reference Designs
Get IP Cores
View Webcast
Read Success Stories
Buy Now
Buy Paris Dev Kit
Support
View Knowledge Database
Use Troubleshooter
Join the Altera Forum
Join the Nios Forum
Documentation
View Literature
Get Solution Sheet (PDF)
Get Auto Handbook (PDF)
http://www.altera.com/end-markets/end-index.html
Internet
start
1:44 PM

Firma ACTEL – MICROSEMI http://www.actel.com/



- IGLOO Lowest Power FPGAs
- ProASIC3 Low Power FPGAs
- SmartFusion Mixed Signal FPGAs
- Fusion Mixed Signal FPGAs

- RadTolerant FPGAs
- Embedded Processors
- Intellectual Property



Firma Atmel http://www.atmel.com/products/fpga/default.asp

Field Programmable Gate Array Devices

Device Family	Summary Benefit	Applications	Technologies	Key Parameter
AT40K FPGAs (5 Volts) Details	Speed up processor-based system performance while lowering power, part count and cost.	Massive register counts (1,024 to 6,400 registers) make them ideal for use as re-configurable DSP co-processors.		5k to 40k Gates 5V
AT40KAL FPGAs (3.3 Volts)	Fast, flexible, distributed 10 ns FreeRAM™ without using valuable logic resources	High-density, compute-intensive DSP and other fast logic designs		5k to 50k Gates 3V

The AT40KAL family are FPGAs with the ability to implement Cache Logic design, where part of the FPGA can be reprogrammed without loss of register data, while the remainder of the FPGA continues to operate without disruption. This is ideal for building adaptive filters, variable coefficient multipliers and other designs where the datapath can change to increase system performance.

AT40KAL FPGAs (3.3 Volts)

Overview | Parametric | Tools

Click on Down/Up arrows to sort.

Download to Excel:

Devices	F.max (MHz)	Max I/O Pins	Memory	Registers	Speed	Usable Gates	Vcc (V)	Packag
AT40K05AL	100	128	2048	496	-1	5K - 10K	3.3	LQFP 144 TQ 100 PLC 84
AT40K10AL	100	192	4608	954	-1	10K - 20K	3.3	LQFP 144 TQ 100 PLC 84
AT40K20AL	100	256	8192	1520	-1	20K - 30K	3.3	LQFP 144 TQ 100 PLC 84
AT40K40AL	100	384	18432	3048	-1	40K - 50K	3.3	LQFP 1

FPGA Integrated Development Systems (IDS)

A tool for creating fast, predictable designs with ATA6625 AT40K, AT40KAL, and AT6000 series FPGAs using HDL Planner for VHDL and Verilog Entry. This tool can be used with other popular synthesis tool environments. The IDS is available as a standalone tool, or integrated into system designer software as a complete package for FPSLIC/FPGA.

FPSLIC - Field Programmable System Level Integrated Circuits

Resurse bibliografice:

[1] Prof. Sherief Reda Division of Engineering, Brown University Spring - Reconfigurable Computing

[2] Prof. Russell Tessier - Reconfigurable Computing

[3] D**r. Gilles SASSATELLI- Lecture on Reconfigurable Technologies**

[4] Marco Platzner Computer Engineering Group - **Reconfigurable Computing**

[5] ARM-University Program

1. Sisteme reconfigurabile de calcul.

1.1. Introducere.

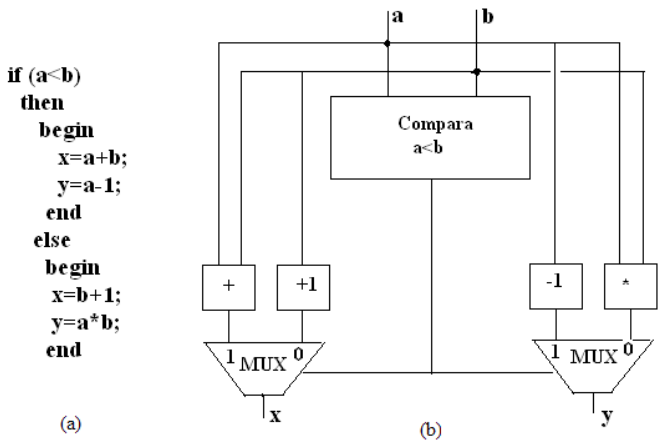
În prezent, cerințele privind calculul de înalta performanță, pentru rezolvarea problemelor din cele mai multe domenii ale activității social-economice, au devenit extreme de presante. Astfel, se pot menționa problemele de optimizare pe scară largă, simularea în fizica şi ştiinta pământului, bioinformatica, prelucrarea semnalelor etc. În acest context, procesoarele programate prin software, cu toate perfecționările aduse de proiectanți, nu oferă performanța necesară. Spre exemplu, viteza de execuție a instrucțiunilor prezintă limitări, iar procesoarele universale cu nuclee multiple, pentru o performanță ridicată, impun ca algoritmii de prelucrare să aibe fire de execuție caracterizate printr-o granularitate grosieră, ceea ce face ca schimburile de date între firele de execuție să fie relativ rare.

Actualmente, dezvoltarea tehnologiei mijloacelor automate de calcul oferă mai multe căi de implementare/execuție a algoritmilor*: procesoare programate prin software*; *structuri hardware fixe*, materializate prin circuite integrate specifice aplicației (ASIC); *structuri hardware reconfigurabile*, bazate pe arii de porți reprogramabile (FPGA).

Procesoarele programate prin software au avantajul unei mari flexibilități, prin posibilitatea de a descrie algoritmii cu ajutorul unui set de funcții primitive, numite instrucțiuni. Descrierea algoritmului poate fi mai mult sau mai puțin concisă, in funcție de arhitectura setului de instrucțiuni, de expresivitatea acestora. Dezavantajele acestui mod de implementarea a algoritmilor sunt legate de o limitare a vitezei datorită operării secvențiale, de setul fix de instrucțiuni, de execuția interpretativă a acestora etc. La baza procesoarelor programate prin software se află "maşina" von Neumann (vN), care va face obiectul paragrafului 1.2.

Structurile hardware fixe (ASIC) sunt caracterizate printr-o viteză relativ mare de operare, obținută prin exploatarea paralelismului intrinsec al algoritmului. În acelaşi timp, ele sunt extrem de eficiente sub aspectul indicatorului cost/performanță, în aplicațiile care reclamă un număr mare de circuite ASIC, similare ca tip. Pe de altă parte, nu pot fi modificate după fabricație şi, în general, au un cost rezidual ridicat.

Pornind de la posibilitățile de implementare ale algoritmilor: prin software sau prin hardware (fix/reconfigurabil), se conturează două modalități de efectuare a calculelor: *temporală* şi *spațială*. Pentru exemplificare se presupune execuția următorului fragment de program pe un calculator convențioanal şi într-o structură ASIC (fig.1.1.).



(a) soluția temporală/software. (b) soluția spațială.

Fig.1.1. Fragment de program pe un calculator vN (a) şi într-o structură ASIC (b).

Timpul total de execuție pe un calculator vN este: $3*t_{instrucțiune}$, care poate cuprinde mai multe cicluri de ceas. Structura ASIC efectuează calculele într-un interval de timp egal cu întârzierea cea mai mare în propagarea semnalului de la intrare la ieşire.

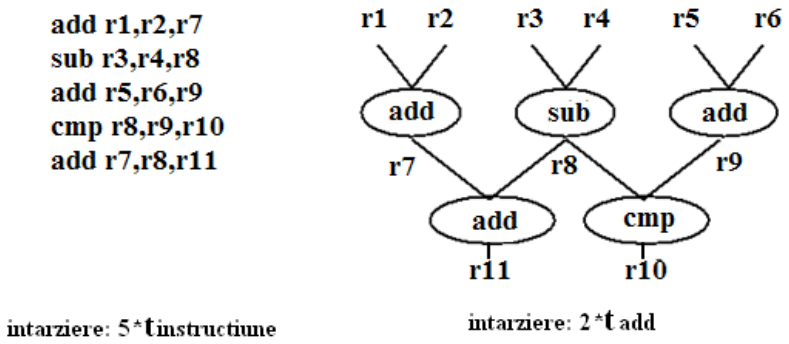
Structurile hardware reconfigurabile acoperă intervalul plasat între implementările algoritmilor prin software şi cele prin structuri ASIC, fiind mai performante decât primele şi mai flexibile decât cele din urmă, datorită utilizării dispozitivelor reprogramabile.

Pentru ilustrare, se presupune un şir de date: r1, r2, r3, r4, r5, r6, asupra cărora se execută următoarele operații:

$$r7 \leftarrow (r1 + r2);\ \ r8 \leftarrow (r3 - r4);\ \ r9 \leftarrow (r5 + r6);\ \ r10 \leftarrow (r8 \neq r9);\ \ r11 \leftarrow (r7 + r8).$$

Ca şi în cazul soluției ASIC, calculele se desfasoară după cum se arată în figura 1.2. În cazul sistemelor reconfigurabile, ansamblele de circuite care implementeaza funcțiile: **add**, **sub**, **cmp** sunt reconfigurabile static sau dinamic.

Cheia accelerarii calculului rezidă în abilitatea de a extrage paralelismul/concurența din descrierea algoritmului.



(a) soluția temporală/software. (b) soluția spațială.

Fig. 1.2. Modalități de efectuare a calculelor.

1.1. Calculatorul von Neumann.

Într-un studiu [1] asupra automatizării calculului, apărut în 1947, matematicianul von Neumann a demonstrat că un calculator, cu o structură relativ simplă, poate executa orice program, care descrie un algoritm de calcul, sub controlul unei unități de comandă, fără a fi necesare modificări de natură hardware.

Structura calculatorului vN (fig.1.3.) posedă o *memorie*, care constă în cuvinte binare cu lungime constantă, memorie utilizată pentru stocarea programului şi a datelor, o *unitate de comandă*, care înglobează un *contor al programului* (CP), pentru controlul execuției programului, şi o *unitate aritmetică-logică* (UAL), pentru calcule aritmetice şi logice. Între componentele funcționale ale unui calculator vN se regăsesc şi *unitățile de intrare/ieşire* (UI/E).

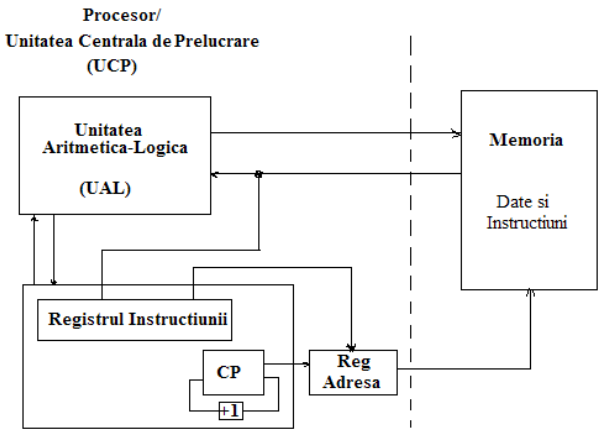


Fig.1.3. Structura calculatorului von Neumann.

Un algoritm de calcul este codificat, cu ajutorul *setului de instrucțiuni* al calculatorului, sub forma unui *program* ale cărui instrucțiuni sunt *executate secvențial*.

Calculatoarele vN moderne, în special cele de tip RISC (Reduced Instruction Set Computer), posedă, în cadrul UCP, un număr de cel putin 32 registre generale (RG), cu un port de intrare şi două porturi de ieşire. Operațiile aritmetice-logice se execută cu operanzi extraşi din RG, rezultatul fiind stocat tot în RG. Singurele instrucțiuni, care privesc memoria, sunt de Încarcare (Load) a unui RG şi Stocare (Store) a conținutului unui RG în memorie.

O analiză a derulării în timp a instrucțiunilor va evidenția 5 faze posibile: *Citeşte instrucțiunea* (Instruction Fetch – **IF**); *Decodifică instrucțiunea/Citeşte registrele generale* (Instruction Decode – **ID**); *Execută instrucțiunea/Calculează adresa* (Instruction Execute – **EX**); *Memorează rezultatul în Memoria de date* (Store Memory – **M);** *Scrie rezultatul în registrul general* (Write back - **WB**). Astfel, o instrucțiune se derulează în urmatoarele cicluri: **IF**, **ID**, **EX, M** şi **W.**

Avantajul principal al calculatorului vN constă în *flexibilitatea* de a executa orice program bine codificat. *Dezavantajele*, care se pot evidenția, constau: în *viteza de execuție scăzută*, datorită operării secvențiale, a diferenței mari între ciclul procesorului şi ciclul de lucru al memoriei, şi în *ineficiența utilizării resurselor* pe parcusul derulării instrucțiunilor. Perfecționările aduse, pentru a contrabalansa dezavantajele menționate, vizează : utilizarea unui *ceas mai rapid*, *operarea în bandă de asamblare* (BA), *memorii cache*, *citirea anticipată a instrucțiunilor* etc.

Arhitectura unui calculator se referă la modul în care programatorul percepe calculatorul pentru programarea acestuia. Calculatoarele convenționale, care operează secvențial, sub controlul *fluxului de instrucțiuni*, se pot caracteriza prin două arhitecturi : von Neuman si Harvard. Arhitectura vN posedă o singură memorie pentru instrucțiuni şi date, în timp ce arhitectura Harvard dispune de memorii separate pentru program şi date, memorii care pot fi accesate în paralel. În figura 1.4. se prezintă modalitățile de execuție a instrucțiunilor în calculatoarele actuale: secvențială, în banda de asamblare, pe arhitectută von Neumann şi în banda de asamblare, pe arhitectură Harvard.

La *execuția secvențială*, dacă se notează: t_{ciclu} = durata ciclului de execuție, de regulă egală cu perioada ceasului, pentru execuția unei instrucțiuni este necesar un timp $t_{instrucțiune} = 5*t_{ciclu}$, iar pentru execuția a 4 instrucțiuni $20*t_{ciclu}$.

In *condițiile benzii de asamblare şi ale arhitecturii vN*, o instrucțiune necesită un timp de execuție egal cu $5*t_{ciclu}$. În mod ideal (fără hazarduri) 4 instrucțiuni în secvență se execută în $8*t_{ciclu.}$.

Dacă prima instrucțiune din secvență este Load, atunci cea de-a patra instrucțiune trebuie să fie întarziată cu un ciclu, deoarece apare un hazard structural cauzat de existența unei singure

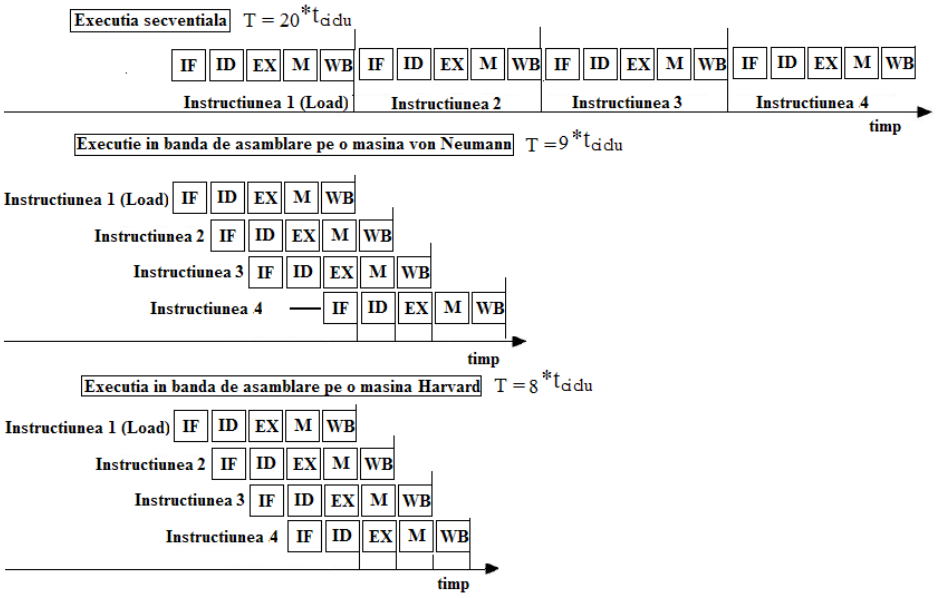


Fig. 1.4. Modalități de execuție a instrucțiunilor în calculatoarele actuale.

memorii: instrucțiunea 1 încearca să citească data din memorie, în timp ce instrucțiunea 4 urmează să fie citită. Astfel, în cazul dat, cele 4 instrucțiuni se execută în $9*t_{ciclu.}$. Hazardul structural este înlăturat, în cazul de față, dacă se utilizează o maşina Harvard în care conflictele de acces la memorie pentru instrucțiuni şi date sunt absente. Cele 4 instrucțiuni se execută în $8*t_{ciclu.}$.

1.2. Noi paradigme de calcul.

Cerințele tot mai ridicate, privind performanțele sistemelor convenționale de calcul, în legătură cu creşterea vitezei de operare, reducerea puterii disipate şi miniaturizarea, au condus, în ultimele decenii, la elaborarea a noi paradigme de calcul, facilitate de progresele înregistrate în plan tehnologic.

În legatură cu dezvoltatrea tehnologiei circuitelor electronice, s-a constatat o periodicitate de 10 ani în ceea ce priveşte apariția unor noi familii de circuite, cu noi funcționalități şi cu performanțe superioare. În figura 1.5. se prezintă diagrama concepută de T. Makimoto [2], din care rezultă că, începând cu anul apariției circuitelor integrate TTL, 1957, s-au succedat 5 "valuri" tehnologice: TTL, MSI/LSI, µP/memorii integrate, ASIC, FPGA. În prezent se consideră că are loc o trecere la *Sistemele integrate pe o Pastilă* (SoC) şi la *calculul reconfigurabil de granuralitate grosieră*. Influența tehnologiei asupra structurilor de calcul şi asupra programării acestora s-a concretizat prin: *structuri fixe/structuri variabile* şi prin *programări*: *rigidă, procedurală şi structurală*.

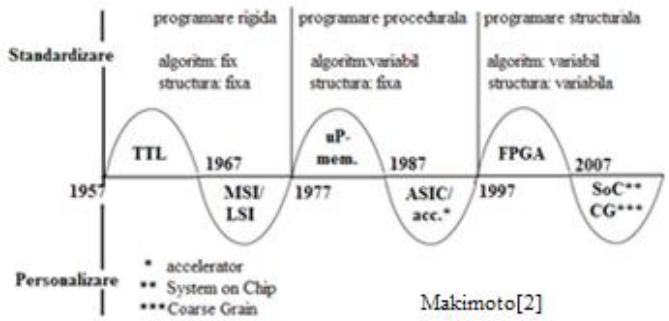


Fig. 1.5 Diagrama lui T. Makimoto, privind evoluția circuitelor electronice.

Având în vedere că modelul convențional al maşinii von Neumann, orientat pe *fluxul de instrucțiuni*, prezintă limitări serioase, nu numai în ceea ce priveşte viteza de operare [3], [4], cercetările în domeniu s-au orientat către modele neconvenționale de maşini şi, in special, către maşina bazată pe *fluxul de date/data stream* [5]. În ultimul deceniu, datorită progreselor întregistrate în domeniul circuitelor reprogramabile, de tip FPGA, dezvoltările de noi echipamente de calcul, în condiții industriale, au avut în vedere soluții de genul *maşini von Neuman în cooperare cu acceleratoare* [6] *non von Neumann* (fig.1.6.).

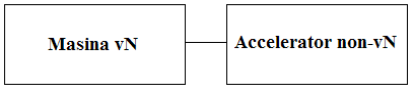


Fig.1.6. Modelul de bază care, în condiții industriale, înlocuieşte modelul von Neumann (vN).

O clasificare a paradigmelor de calcul, în conformitate cu stadiul actual al tehnologiei şi al cunoştintelor în acest domeniu, este prezentată în Tabelul 1.1, în care se regăsesc termeni ce vor fi explicați în cele ce urmează.

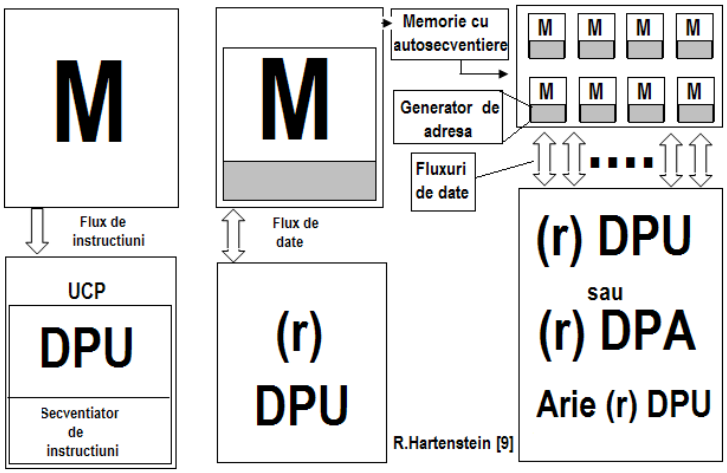
Tab.1. 1. **Clasificarea paradigmelor de calcul** (Hartenstein[9])

<table>
<tr><th colspan="3">Platforma</th><th>programul sursă rulat</th><th>paradigma maşinii</th></tr>
<tr><td colspan="3">Hardware</td><td>(neprogramabilă)</td><td rowspan="2">(lipseşte)</td></tr>
<tr><td rowspan="3">morphware</td><td>granularitate fină</td><td>rGA (FPGA)</td><td rowspan="2">configware</td></tr>
<tr><td rowspan="2">granularitate grosieră</td><td>(r)DPU, (r)DPA</td></tr>
<tr><td>procesor reconfigurabil “data stream”</td><td>flowware & configware</td><td rowspan="2">“antimaşina”</td></tr>
<tr><td colspan="3">procesor “data stream” (cablat)</td><td>flowware</td></tr>
<tr><td colspan="3">procesor bazat pe fluxul de instrucţiuni</td><td>software</td><td>maşina von Neumann</td></tr>
</table>

Noţiunea **morphware** (“Soft Hardware”) a fost introdusă la sfârşitul deceniului 9, în cadrul cercetărilor privind *Arhitecturile Polimorfice de Calcul*, desfăşurate sub egida Agenţiei pentru Proiecte Avansate de Cercetare, în domeniul Apararii (DARPA). Spre deosebire de de hardware-ul clasic, morphware-ul poate fi reconfigurat cu ajutorul unui cod structural (cod de configurare), numit **configware**, stocat in memoriile RAM “ascunse” ale structurii FPGA şi complet diferit de software-ul clasic.

Componentele hardware **rGA**, **(r)DPU**, **(r)DPA** semnifică: *Aria de Porţi reconfigurabilă* (FPGA), *Unitatea de Prelucrare a Datelor nereconfigurabilă/reconfigurabilă* şi *Aria de Prelucrare a Datelor nereconfigurabilă/reconfigurabilă*.

În timp ce Unitatea Centrală de Prelucrare (UCP) convenţională posedă un Contor al Programului (CP), care furnizează adresa pentru citirea instrucţiunii următoare, în vederea execuţiei acesteia, (r)DPU şi (r)DPA nu posedă CP, procesul de execuţie fiind amorsat de sosirea datelor, sub acţiunea unor generatoare de adrese, localizate în memorie (fig.1.7).



(a) (b)

Fig.1.7.Structura maşinii convenţionale (a) şi structura maşinilor bazate pe fluxul/fluxurile de date (b).

Flowware-ul, asociat cu **Fluxul de date**/**Data stream**, în ariile sistolice de prelucrare a datelor, defineşte planificarea sosirii datelor la porturile de intrare ale (r)DPA şi ale plecării rezultatelor din porturile de ieşire ale acesteia (fig.1.8.). Fluxul de date implică o execuţie controlată de transportul datelor, în contrast cu execuţia comandată de fluxul de instrucţiuni.

"Antimaşina" [7], [8] este bazată pe fluxul de date şi utilizează DPU, fără secvenţiator, sau DPA, fără secvenţiatoare. Secvenţiatoarele, corespunzătoare modelului "antimaşinii ", se găsesc în memorie (nu la nivelul UCP), sub forma de generator/generatoare de adrese. La o maşina convenţională nucleul [9], [10] îl constituie UCP, în jurul căruia "gravitează" fluxul de instrucţiuni, în timp ce, în cazul "antimaşinii" nucleul îl reprezintă DPU/DPA în jurul cărora "gravitează" fluxul/fluxurile de date, ceea ce constituie o explicaţie a termenului de "antimaşină". In cazul "antimaşii" fluxurile de date trebuie să fie programate, pentru a se preciza care din date şi la ce moment de timp trebuie să sosească la un port DPU sau la anumite porturi ale DPA.

Un astfel de program-sursa, orientat pe fluxul de date, poarta numele de *flowware*, dupa cum s-a arătat mai sus.

In *calculul paralel*, *granularitatea unui algoritm* semnifică raportul între timpii de calcul şi cei necesari comunicaţiilor.

*Paralelismul cu **granularitate fină*** este caracterizat prin task-uri individuale mici, cu coduri de execuție de dimensiuni reduse si cu timpi de execuție, de asemenea, mici. Datele sunt transferate relativ frecvent între procesoare şi constau în unul sau câteva cuvinte de memorie.

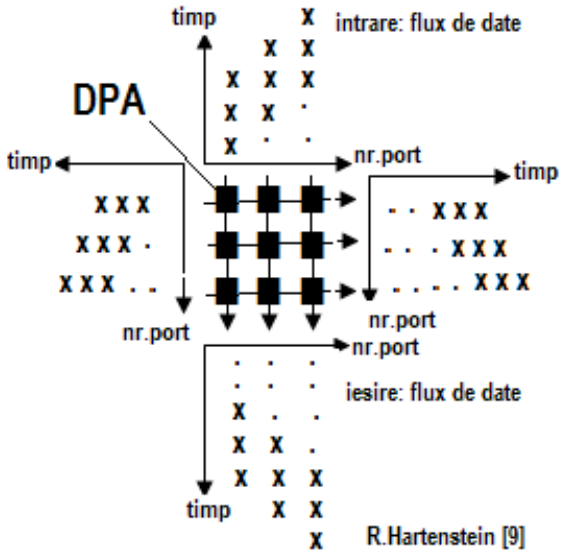


Fig.1.8. Fluxurile de date şi porturile de I/E pentru o structură DPA

*Paralelismul cu **granularitate grosieră*** se manifestă în cazul comunicațiilor de date cu frecvență relativ mică, după efectuarea unor mari cantități de calcule.

Granularitatea fină prezintă un mare potențial de paralelism şi de accelerare a calculului, însoțită de o regie (overhead) apreciabilă datorată, atât frecvenței ridicate a comunicațiilor, cât şi sincronizării acestora.

În *calculul reconfigurabil şi calculul de înalta performanță* termenii de mai sus trebuie asociați şi cu lățimea căii de date. Utilizarea unor unități de prelucrare pe un bit, de exemplu, ca în Blocurile Logice Configurabile, din FPGA, se situează în zona calculului cu granularitate fină sau în cea a reconfigurării cu granularitate fină. Utilizarea unor căi de date cu lățime mare, de exemplu de 32 de biți, ca în cazurile UCP-microprocesor sau a unităților DPU, a ariilor DPA, controlate de fluxul de date, este o caracteristică a calculului cu granularitate grosieră sau a reconfigurării cu granularitate grosieră [11].

Bibliografie.

[1] Goldstein, H. von Neumann, J. Burks, A.: Report on the mathematical and logical aspects of an electronic computing instrument; report, Princeton Institute of Advanced Study, 1947.

[2] Makimoto , T. Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing . 10th International Conference, FPL 2000 Villach, Austria, August 27–30, 2000 Proceedings.

[3] Backus, J.: Can programming be liberated from the von Neumann style?; Communications of the ACM, August 1978, 20(8).

[4] Arvind et al.: A critique of Multiprocessing the von Neumann Style; Proc. ISCA 1983.

[5] Bobda, C.: Introduction to Reconfigurable Computing Systems; Springer-Verlag, 2007.

[6] Hartenstein R. (invited chapter): Basics of Reconfigurable Computing; in: J. Henkel, S. Parameswaran (editors): DesigningEmbedded Processors. A Low Power Perspective; Springer Verlag, March 2007.

[7] Hirschbiel, A. et al.: A Flexible Architecture for Image Processing; Microprocessing and Microprogramming, vol 21, 1987.

[8] Weber, M. et al.: MOM - Map Oriented Machine; in: E. Chiricozzi (ed.): Parallel Processing and Applications, North-Holland, 1988.

[9] Hartenstein, R.. Data-stream-based Computing, Enabling Technology for Reconfigurable Computing. ENE, UnB, Seminar, Brasilia. Nov. 22, 2002.

[10] Hartenstein, R.. The von Neumann Syndrome. Invited paper, Stamatis Vassiliadis Symposium „The Future of Computing“, Delft, The Netherlands, Sept. 28, 2007.

[11] Petrescu, I. Contribuții la realizarea unor structuri de calcul performante, bazate pe circuite reconfigurabile. Lucrare de doctorat. Universitatea Politehnica București, 2009.

2. Algorithms for VLSI Processor Arrays

H. T. Kung and Charles E. Leiserson
Department of Computer Science
Carnegie-Mellon University

2.1. Introduction

"And the smooth stream in smoother numbers flows"

--Alexander Pope

We are interested in high-performance parallel algorithms that can be implemented directly on low-cost hardware devices. By performance, we are not refering to the traditional operation counts that characterize classical analyses of algorithms, but rather, the throughput obtainable when a special purpose peripheral device is attached to a general purpose host computer. This implies that time spent in I/O, control, and data movement as well as arithmetics must all be considered. The cost of the device must be measured in how well it can be implemented using LSI technology and must be sensitive to what the technology can do cheaply, and what is expensive.

LSI technology has made one thing clear. *Simple* and *regular* interconnections lead to cheap implementations and high densities, and high density implies both high performance and low overhead for support components. The two-dimensional array structure consisting of mesh-connected processors enjoys this desirable property. Therefore, we are interested in designing parallel algorithms which have simple and regular data flows so that they can be executed efficiently on such processor arrays. We are also interested in using *pipelining* as a general method for implementing these algorithms in hardware. By pipelining, processing may proceed concurrently with input and output, and consequently overall execution time is minimized. Pipelining plus multiprocessing at each stage of a pipeline should lead to the best-possible performance. In this section, we demonstrate simple and regular multiprocessor networks that are capable of pipelining some important matrix computations with optimal speed-up.

Most of the results reported here are based on a paper by H. T. Kung and C. E. Leiserson, which is to be presented at the Symposium on Sparse Matrix Computations and Their Applications in Knoxville, TN, November 2-3, 1978.

2.2. The Basic Components and Structures

The single operation common to the problems considered in this section is the so-called inner product step, $C \leftarrow C + A \times B$. We postulate a processor which has three registers R_A, R_B, and R_C. Each register has two connections, one for input and one for output. Fig. 2.2.1 shows two types of geometries for this processor. Type (a) geometry will be used for matrix-vector multiplication and solution of triangular linear systems (Sections 2.3 and 2.6), whereas type (b) geometry will be used for matrix multiplication and LU-decomposition (Sections 2.4 and 2.5).

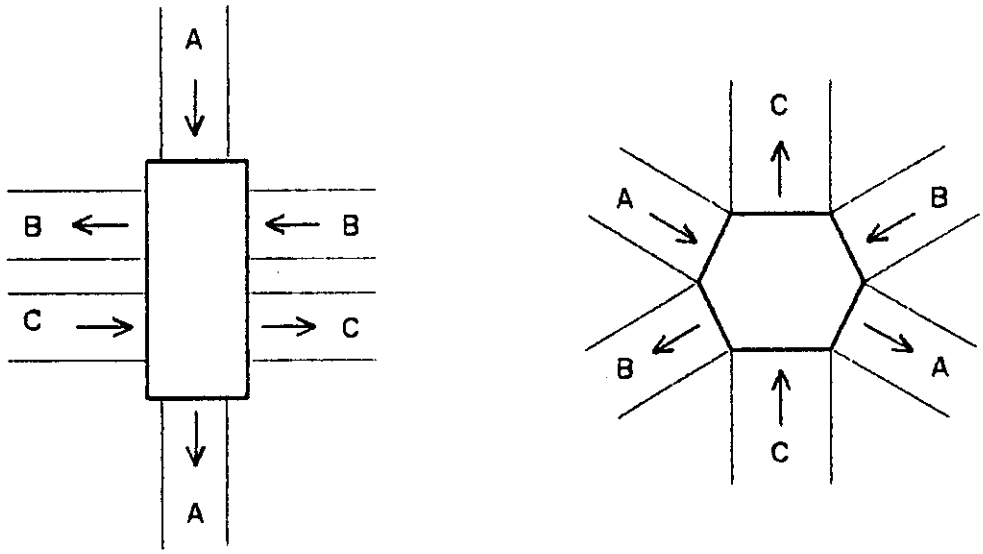


Fig. 2.2.1. Geometries for the inner product step processor.

The processor is capable of performing the inner product step. We shall define a basic time unit in terms of this processor. At time t, the processor shifts its inputs into R_A, R_B, and R_C, and computes $R_C \leftarrow R_C + R_A \times R_B$. At time t+1, the new value of R_C together with the input values for R_A and R_B are available as outputs. All outputs are latched and the logic is

clocked so that when one processor is connected to another, the changing output of one during the time interval between t and t+1 will not interfere with the input to the other during this time. This is not the only processing element we shall make use of, but it will be the work horse. These special processors will be specified later when they are used.

The basic network organization we shall adopt for internal communication is the mesh-connected processor scheme. (See Fig. 2.2.2.) All connections from a processor are to neighboring processors. The most widely known system based on this organization is the ILLIAC IV. If diagonal connections are added in one direction only, we shall call the resulting scheme hexagonally mesh-connected or hex-connected for short. We shall demonstrate that linearly connected and hex-connected processors are natural for matrix problems.

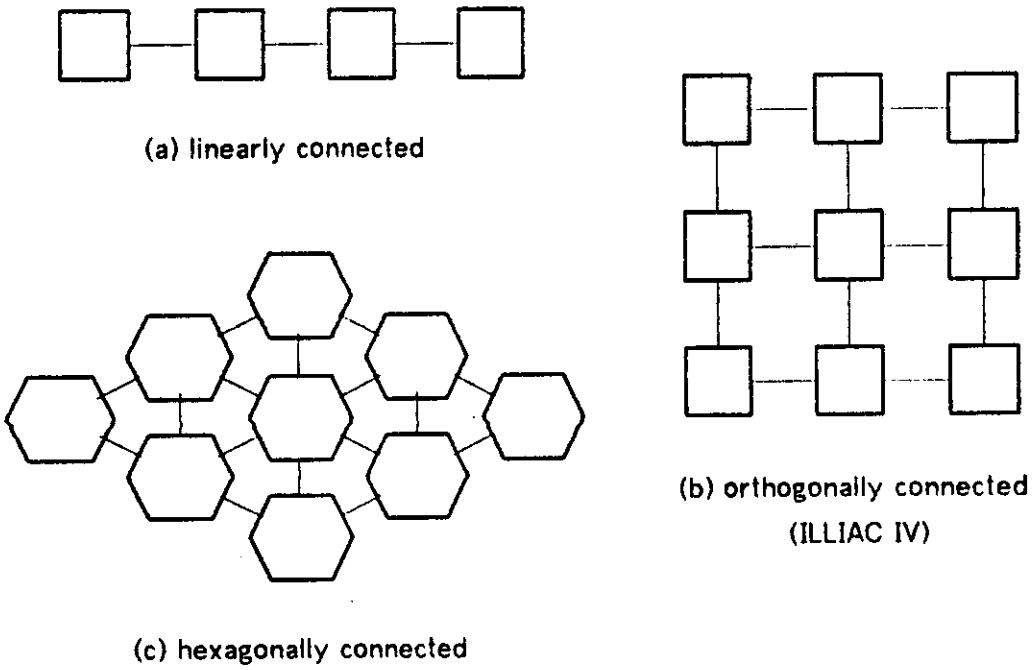


Fig. 2.2.2. Examples of mesh-connected processors.

When an input path to a processor lies on an edge of the device, we shall sometimes designate it as an external input connection from the host memory. Alternatively, we may let the input have a fixed value such as zero. An output data path will either go to the host memory or be ignored.

2.3. Matrix-Vector Multiplication

We consider the problem of multiplying a matrix with a vector. Let $A = (a_{ij})$ be an nxn band matrix with band width $w = p+q-1$, and $x = (x_1,...,x_n)^T$, $y = (y_1,...,y_n)^T$ be n-vectors such that $Ax = y$. (See Fig. 2.3.1 for the case when $p = 2$ and $q = 3$.)

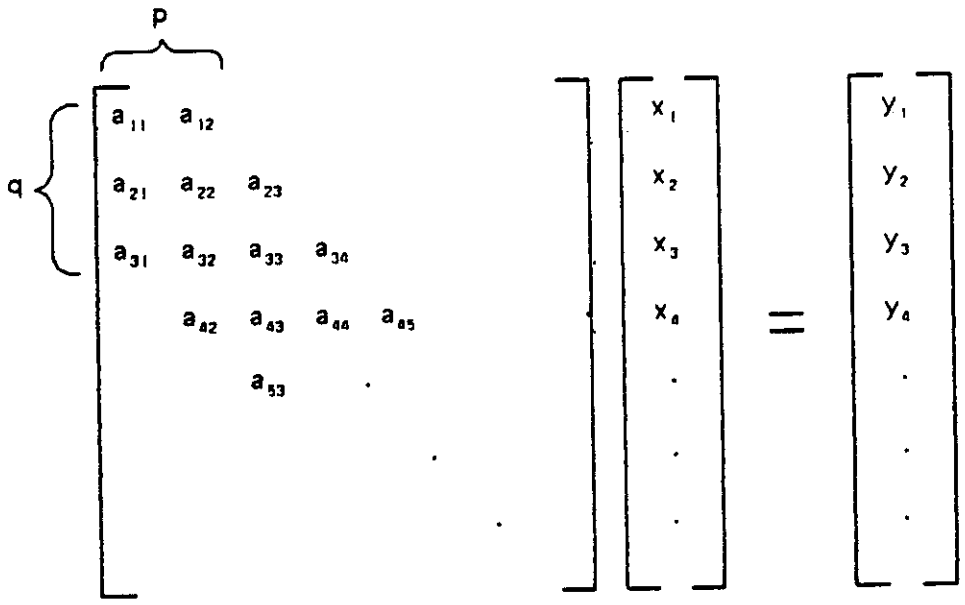


Fig. 2.3.1. The matrix-vector multiplication when the matrix is a band matrix with $p = 2$ and $q = 3$.

Suppose A and x are given. The following algorithm computes the product $y = Ax$ by pipelining the computation through w linearly connected processors. Before giving the code for each processor, we illustrate the algorithm for the band matrix-vector multiplication problem in Fig. 2.3.1. For this case the linearly connected network has four processors. See Fig. 2.3.2.

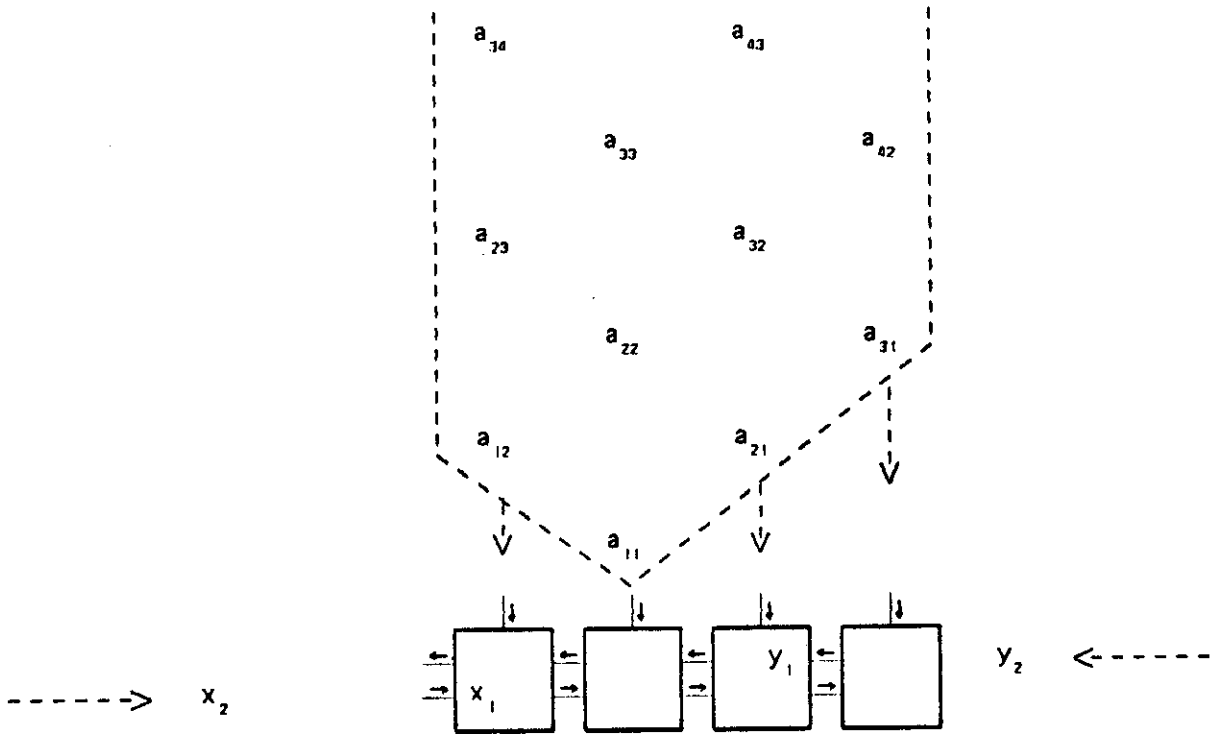


Fig. 2.3.2. The linearly connected network for the matrix-vector multiplication problem shown in Fig. 2.3.1.

The general scheme of our pipelining algorithm can be viewed as follows. The y_i, which are initially zero, keep moving to the left while the x_i are moving to the right and the a_{ij} are moving down. It turns out that each y_i is able to accumulate all its terms, namely, $a_{i,i-2}x_{i-2}$, $a_{i,i-1}x_{i-1}$, $a_{i,i}x_i$ and $a_{i,i+1}x_{i+1}$, before it leaves the network. Fig. 2.3.3 illustrates the first seven steps of the algorithm. Note that when y_1 and y_2 are output they have the correct values. Observe also that at any given time alternating processors are idle. (Indeed, it is possible to use w/2 processors in the network for a general band matrix with band width w. We did not do so for the sake of clarity.)

We now specify the algorithm more precisely. Assume that the processors are numbered by integers 1, 2, . . ., w from the left end processor to the right end processor. Each processor has three registers, R_A, R_x and R_y, which will hold entries in A, x and y, respectively. Initially, all registers contain zeros. Each step of the algorithm consists of the following operations, but for odd numbered time steps only odd numbered processors are activated and for even numbered time steps only even numbered processors are activated.

1. *Shift.*

   - R_A gets a new element in the band of matrix A.

   - R_x gets the contents of register R_x from the left neighboring node. (The R_x in processor 1 gets a new component of x.)

   - R_y gets the contents of register R_y from the right neighboring node. (Processor 1 outputs its R_y contents and the R_y in processor w gets zero.)

2. *Multiply and Add.*

   $R_y \leftarrow R_y + R_A \times R_x$.

Using the processor postulated in section 2.2, we note that the three shift operations in step 1 can be done simultaneously, and that each step of the algorithm takes a unit of time. Suppose the bandwidth of A is $w = p+q-1$. It is readily seen that after w units of time the components of the product $y = Ax$ start shifting out from the left end processor at the rate of one output every two units of time. Therefore, using our network all the n components of y can be computed in $2n+w$ time units, as compared to the O(wn) time needed for the sequential algorithm on a single processor.

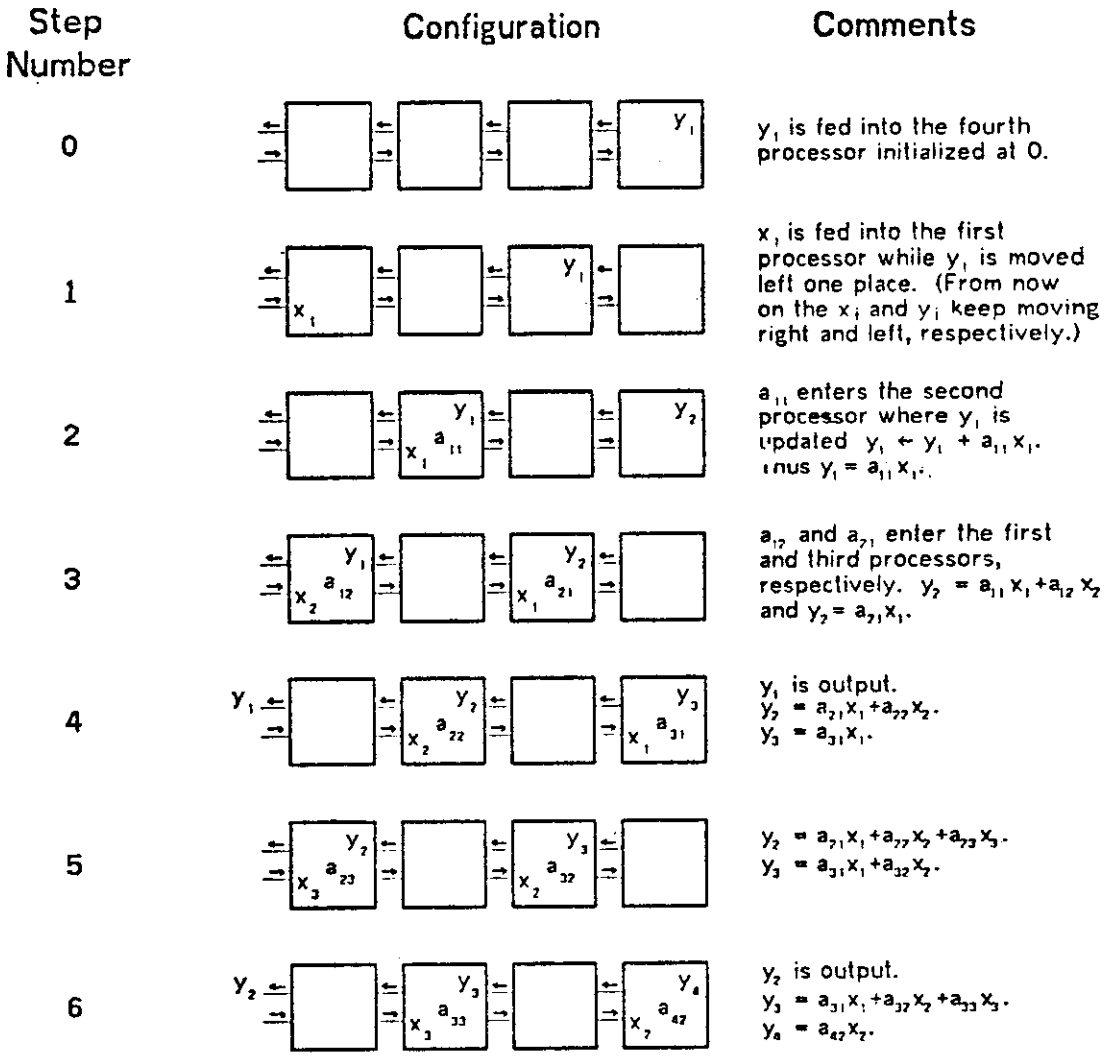


Fig. 2.3.3. The first seven steps of the matrix-vector multiplication algorithm.

2.4. Matrix Multiplication

This section considers the problem of multiplying two matrices. Let A and B be nxn band matrices of bandwidth w_1 and w_2, respectively. We show that a network of $w_1 w_2$ hex-connected processors can compute the product C = AxB in $3n+\min(w_1,w_2)$ units of time. The algorithm uses the same principle as the one in Section 2.3. We illustrate the general scheme by considering the matrix multiplication problem depicted in Fig. 2.4.1.

The diamond shaped interconnection network for this case is shown in Fig. 2.4.2, where processors are hex-connected and data flows are indicated by arrows. The nonzero elements in A, B and C move through the network in three directions, as indicated in the figure. Initially, the c_{ij} are all zeros. One can easily see that with the type (b) inner product processors described in Section 2.2, each c_{ij} is able to accumulate all its terms before it leaves the network.

Suppose that Fig. 2.4.2 describes the configuration at time 1. Then, for example, c_{11} gets $a_{11}b_{11}$ at time 2 and $a_{12}b_{21}$ at time 3, while c_{21} gets $a_{21}b_{11}$ at time 3 and $a_{22}b_{21}$ at time 4. (Note that approximately only one third of processors in the network are active at a given time. Indeed, it is possible to use about $(w_1 w_2)/3$ processors in the network for multiplying two band matrices with band widths w_1 and w_2.)

$$\begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ a_{31} & a_{32} & a_{33} & a_{34} \\ & a_{42} & & \ddots \\ & & & \ddots \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & \\ b_{21} & b_{22} & b_{23} & b_{24} \\ & b_{32} & b_{33} & b_{34} \\ & & b_{42} & \ddots \\ & & & \ddots \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & & \ddots \\ & & & \ddots \end{bmatrix}$$

Fig. 2.4.1. Matrix multiplication.

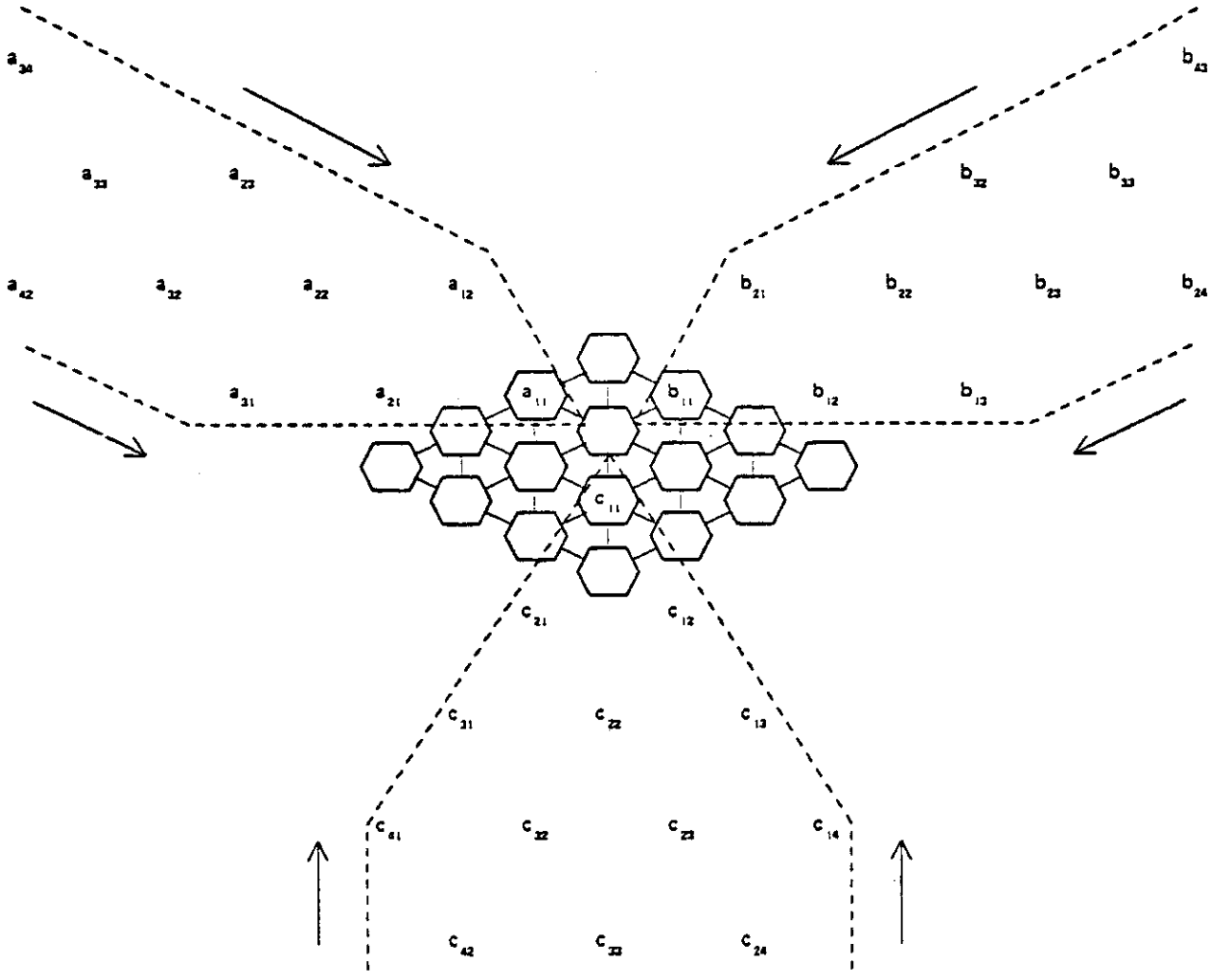


Fig. 2.4.2. The network for the matrix multiplication C = AxB shown in Fig. 2.4.1.

2.5. The LU-Decomposition of a Matrix

The LU-decomposition of a given a matrix A is the problem of computing lower and upper triangular matrices L and U such that A = LU. (Cf. Fig. 2.5.1.)

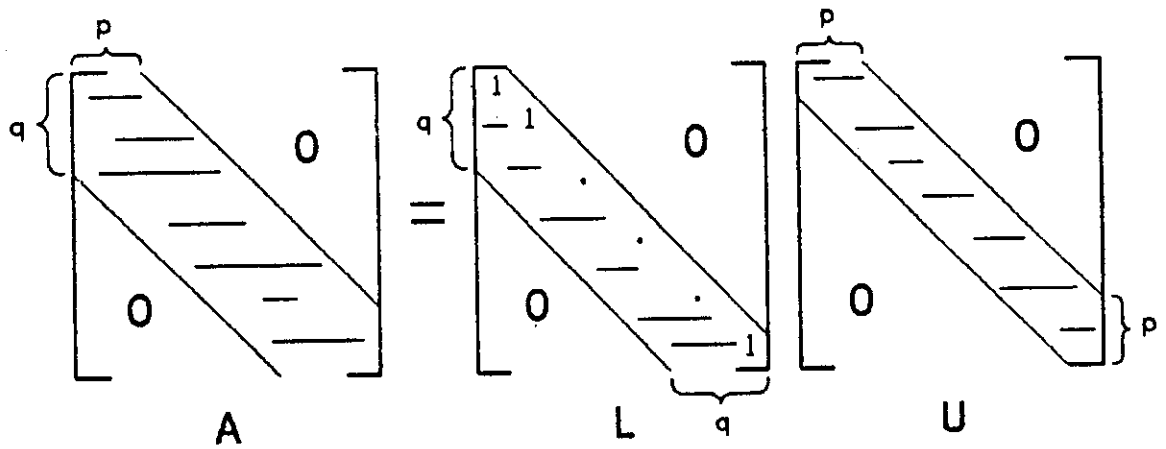


Fig. 2.5.1. The LU-decomposition of a matrix.

Once the L and U factors are known it would be relatively easy to solve a linear system Ax = b or to invert A. In the following we describe a parallel LU-decomposition algorithm using a hex-connected network.

We assume that A is either a symmetric positive-definite or irreducible diagonally dominant matrix. It is well-known that under this assumption the L and U matrices can be obtained by Gaussian elimination without pivoting. We show the rather surprising fact that Gaussian elimination enjoys the same data flow as matrix multiplication and that all the processors except one perform the same inner product step. In fact, the same matrix multiplication network in Section 2.4 can be used to compute L and U matrices, provided that the processor at the top now computes minus the reciprocal of an input and the orientation of the other boundary processors is properly altered. More precisely, at the special processor at the top, the data from the south processor is passed unchanged to the north, minus its reciprocal is computed and sent to the southwest processor, and a numerical value "1" is sent to the southeast processor. The processors on the left hand "upper" side are rotated 120 degrees clockwise and always receive "0" from their northwest external input connections. Similarly, the processors on the right hand "upper" side are rotated 120 degrees counterclockwise and always receive "0" from their northeast external input connections. (Of course, it is not necessary to actually input "0" for

these processors; we did so for the sake of uniformity.)

Suppose that $L = (l_{ij})$ and $U = (u_{ij})$. Then Gaussian elimination computes the entries in L and U using the following procedure:

$$l_{ij} = m_{ij} \text{ for } i > j, \ 1 \text{ for } i = j \text{ and } 0 \text{ for } i < j,$$

$$u_{ij} = a_{ij}^{(i)} \text{ for } i \leq j \text{ and } 0 \text{ for } i > j,$$

$$a_{ij}^{(0)} = a_{ij},$$

$$m_{ik} = - a_{ik}^{(k)} / a_{kk}^{(k)},$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} + m_{ik} \, a_{kj}^{(k)}$$

To illustrate our results, we consider a band matrix A with $p = 4$ and $q = 4$. When elements in the band of matrix A are fed into the lower edge of the hex-connected network as shown in Fig. 2.5.2, the elements of L and U are output from the upper edge. Fig. 2.5.3 shows an enlargement of the configuration after eight steps of the algorithm have been executed. The flow of data on the network is indicated by arrows in Fig. 2.5.3. The hexagons denote the standard processors which perform the inner product step just like the corresponding processors in the matrix multiplication network (cf. Fig. 2.4.2). The processor at the top denoted by a circle performs the reciprocal and negation operations. As in the matrix multiplication algorithm, each processor only operates once every three time steps. We will not give a formal correctness proof for the algorithm here. But for understanding the algorithm the reader is advised to view the LU-decomposition as the inverse problem of multiplying a lower triangular matrix with 1's on the diagonal to an upper triangular matrix. Then the algorithm of this section can simply be regarded as one which undoes the matrix multiplication algorithm of Section 2.5. Having realized this, one should be able understand also why the two algorithms use the same network and enjoy the same data flow pattern. The idea of using the same network for both the forward and backward problems seems to be general. It will be used again in Section 2.6.

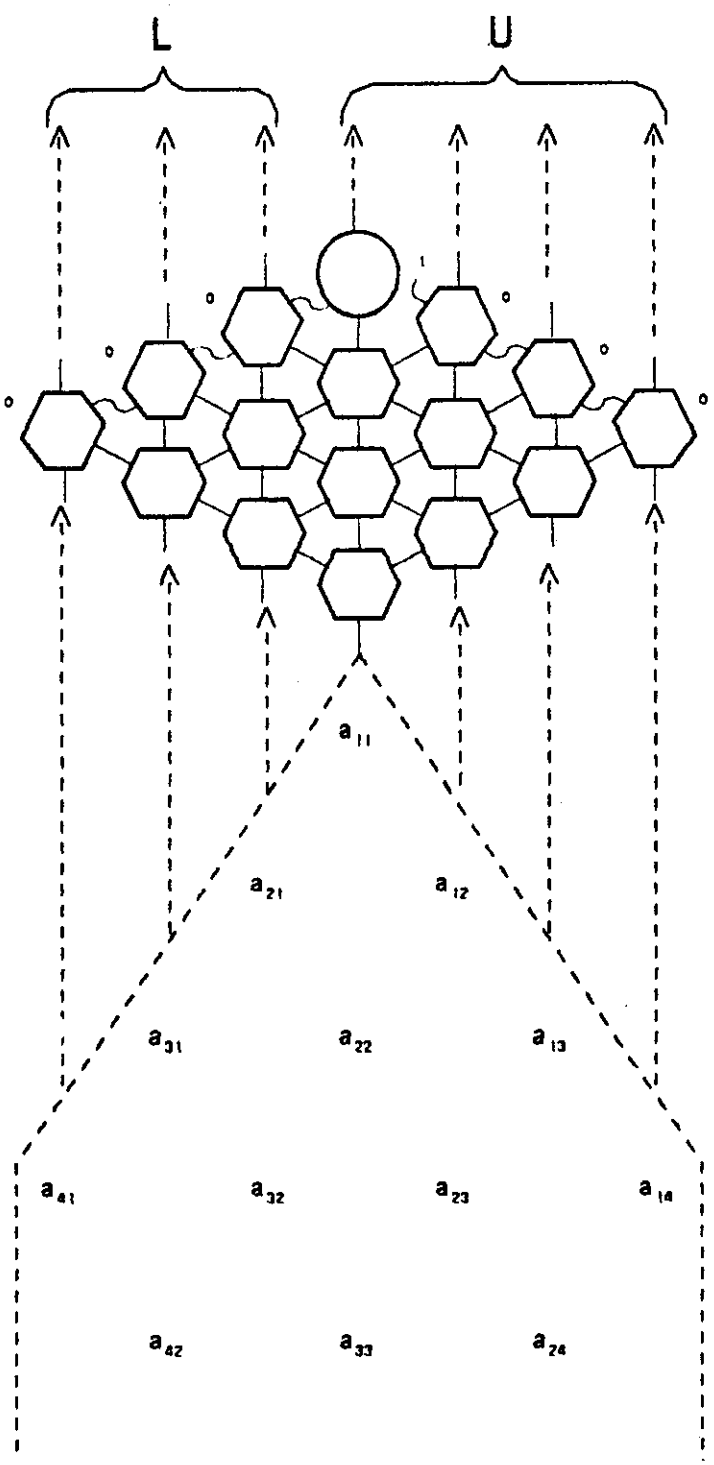


Fig. 2.5.2. The hex-connected network for pipelining the LU-decomposition of a band matrix with p = 4 and q = 4.

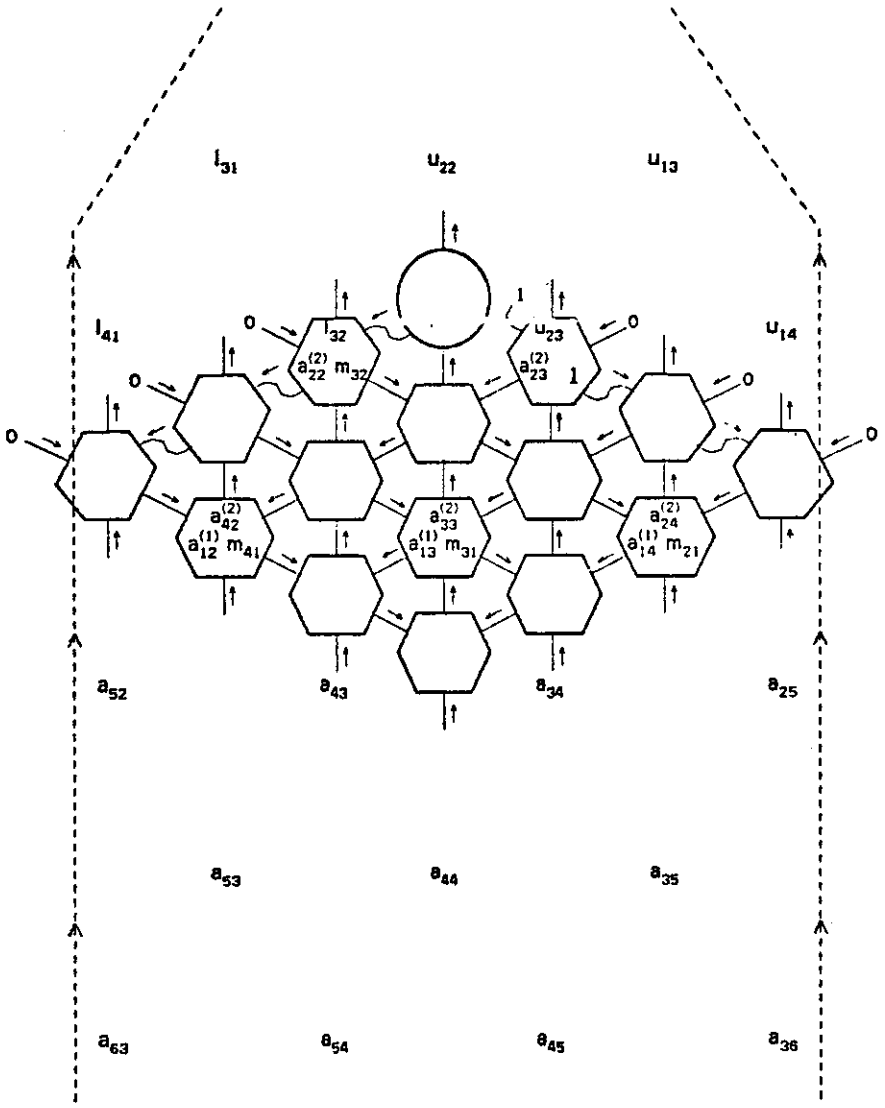


Fig. 2.5.3. LU-decomposition after the first eight steps.

It is readily seen that if matrix A is nxn, then using the network shown in Fig. 2.5.2 the L and U matrices can be computed in 3n+4 units of time. In general, if A is an nxn band matrix with band width w = p+q-1, then with a network of no more than pq hex-connected processors, the LU-decomposition of A can be done in 3n+min(p,q) units of time. (It is possible to reduce the number of required processors to about pq/3.) In particular if A is an nxn dense matrix, then n^2 hex-connected processors can compute the L and U matrices in 4n units of time, including I/O time.

2.6. Triangular Linear Systems

Suppose that we want to solve a linear system $Ax = b$. Then after having done with the LU-decomposition of A (e.g., by methods described in Section 2.5), we still have to solve two triangular linear systems $Ly = b$ and $Ux = y$. This section concerns itself with the solution of triangular linear systems.

Let $A = (a_{ij})$ be a nonsingular nxn band lower triangular matrix with band width $w=q$. Suppose that A and an n-vector $b=(b_1,...,b_n)^T$ are given. The problem is to compute $x=(x_1,...,x_n)^T$ such that $Ax=b$. (See Fig. 2.6.1 for the case when $q=4$.)

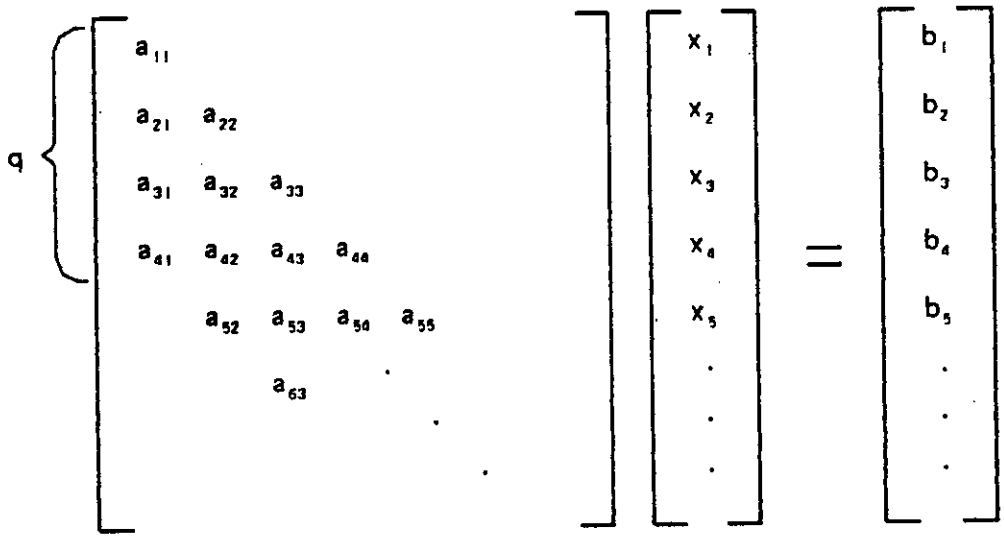


Fig. 2.6.1. The band (lower) triangular linear system with $q=4$.

We show that this problem can be solved by the algorithm and network almost identical to those used for band matrix-vector multiplication in Section 2.3. (Note that the linear system problem can be regarded as the inverse of the matrix-vector multiplication problem.) We illustrate our result by considering the linear system problem in Fig. 2.6.1. For this case, the network and the general scheme of the algorithm are described in Fig. 2.6.2.

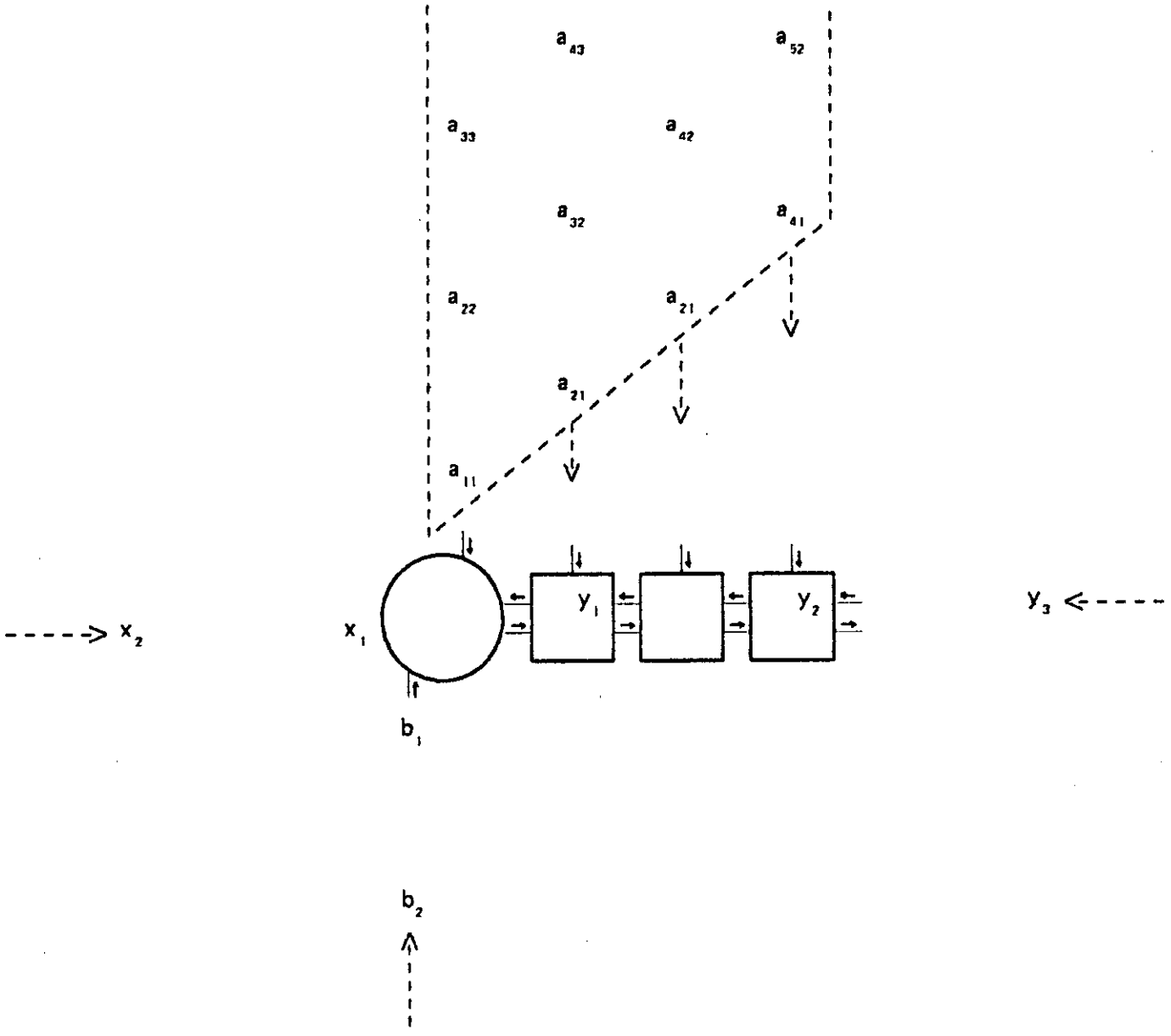


Fig. 2.6.2. The linearly connected network for solving the linear system shown in Fig. 2.6.1.

The y_i, which are initially zero, keep moving to the left while the x_i, a_{ij} and b_i are moving in the network, as indicated in Fig. 2.6.2. The left end processor is special in that it performs $x_i \leftarrow (b_i - y_i)/a_{ii}$. Each y_i accumulates inner product terms in the rest of the processors as it moves to the left. At the time y_i reaches the left end processor it has the value $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{i,i-1}x_{i-1}$, and, consequently, the x_i computed by $x_i \leftarrow (b_i - y_i)/a_{ii}$ at the processor will have the correct value. Fig. 2.6.3 demonstrates the the first ten steps of the algorithm.

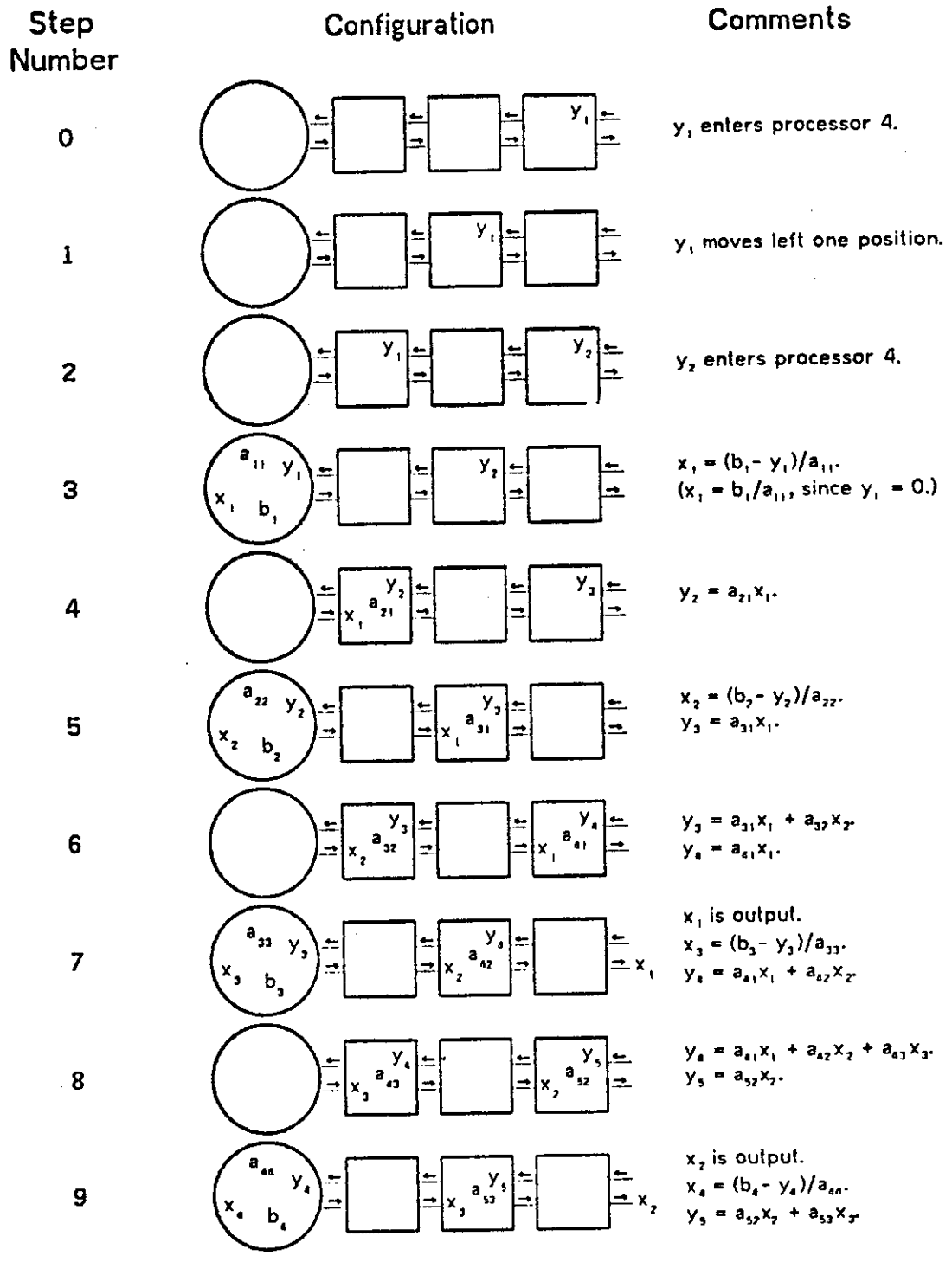


Fig. 2.6.3. Solving a lower band triangular system with $q = 4$.

One can check that the computed x_1, x_2, x_3 and x_4 all have correct values. With this network we can solve an nxn band triangular linear system with band width $w=q$ in $2n+q$ units of time.

2.7. Applications and Comments

2.7.1 Variants of the Algorithms

Rather than the basic algorithms presented above it is their variants that will be used mostly in practice. No attempt is given here for listing all the possible variants; it is important that the reader understands the basic principles of the algorithms so that he can construct appropriate variants for his specific problems.

We first note that although most of our illustrations are done for band matrices all the algorithms work for the regular nxn dense matrix. In this case the band width of the matrix is w = 2n - 1. If the band width of a matrix is so large that a corresponding algorithm requires more processors than a given network provides, then one should decompose the matrix and solve each subproblem on the network.

One can often reduce the number of processors required by an algorithm if the matrix is known to be sparse. For example, the matrices derived from differential equations by using finite differences or finite elements approximations are usually "sparse band matrices." These are band matrices whose nonzero entries appear only in a few of those lines in the band which are parallel to the diagonal. In this case by introducing proper delays to each processor for shifting its data to its neighbors, the number of processors required by the algorithms in Sections 2.3 and 2.6 can be reduced to the number of those diagonal lines which contain nonzero entries. This variant is useful for performing iterative methods involving sparse band matrices.

It is possible to use our algorithms and networks to slove some nonnumerical problems when appropriate interpretations are given to the addition (+) and multiplication (x) operations. For example, some pattern matching problems can be viewed as matrix problems with comparison and Boolean operations.

2.7.2. Convolution and Discrete Fourier Transform

There are a number of important problems which can be formulated as matrix-vector multiplication problems and thus can be solved rapidly by the algorithm in Section 2.3. The problems of computing convolutions and discrete Fourier transforms are such examples. If a matrix has the property that the entries on any line parallel to the diagonal are all the same, then

the matrix is a Toeplitz matrix. The convolution problem is simply the matrix-vector multiplication where the matrix is a triangular Toeplitz matrix (cf. Fig. 2.7.1).

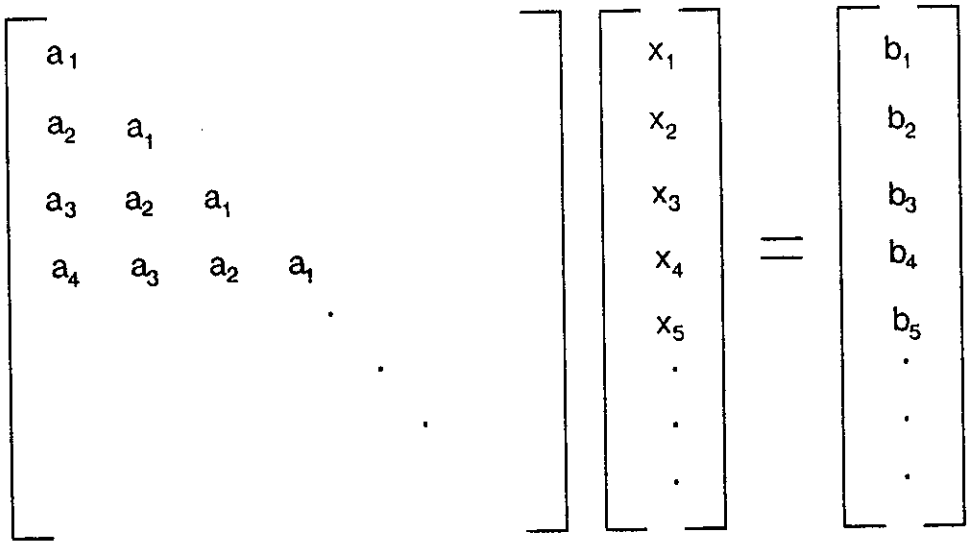


Fig. 2.7.1. The convolution of vectors a and x

On the other hand the n-point discrete Fourier transform is the matrix-vector multiplication, where the (i,j) entry of the matrix is $\omega^{(i-1)(j-1)}$ and ω is a primitive n^{th} root of unity (cf. Fig. 2.7.2).

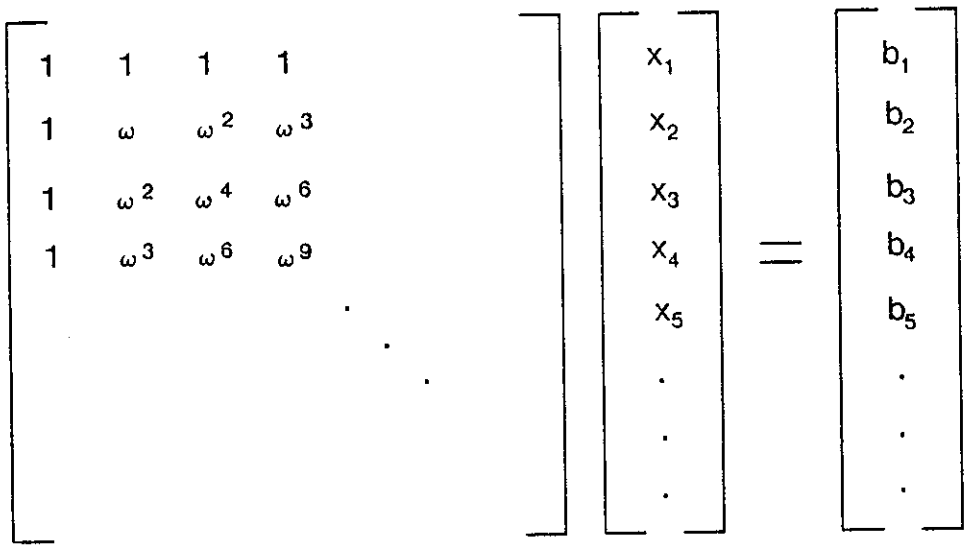


Fig. 2.7.2. The discrete Fourier transform of vector x.

Therefore using a linearly connected network of size O(n) both the convolution of two n-vectors and the n-point discrete Fourier transform can be computed in O(n) units of time, rather than O(n log n) as required by the sequential FFT algorithm. Moreover, note that for the convolution problem each processor has to receive an entry of the matrix only once, and this entry can be shipped to the processor through horizontal connections and stay in the processor during the rest of the computation. For the discrete Fourier transform problem each processor can in fact generate on-the-fly the powers of ω it requires. As a result, for these two problems it is not necessary for each processor in the network to have the external input connection on the top of the processor, as depicted in Fig. 2.3.2.

In the following we describe how the powers of ω can be generated on-the-fly during the process of computing an n-point discrete Fourier transform. The requirement is that if a processor is i units apart from the middle processor then at time i + 2j the processor must have the value of $\omega^{j^2 + ij}$ for all i, j. This requirement can be fulfilled by using the algorithm below. We assume that each processor has one additional register R_t. All processors except the middle one perform the following operations in each step, but for odd (respectively, even) numbered time steps only processors which are odd (even) units apart from the middle processor are activated. For all processors except the middle one the contents of both R_A and R_t are initially "0".

1. *Shift.* If the processor is in the left (respectively, right) hand side of the middle processor then

   - R_A gets the contents of register R_A from the right (respectively, left) neighboring processor.

   - R_t gets the contents of register R_t from the right (respectively, left) neighboring processor.

2. *Multiply.*

   $R_A \leftarrow R_A \times R_t$.

The middle processor is special; it performs the following operations at every ever numbered time step. For this processor the contents of both R_A and R_t are initially "1".

1. $R_A \leftarrow R_A \times R_t^2 \times \omega$.

2. $R_t \leftarrow R_t \times \omega$.

2.7.3. The Common Memory Access Pattern

Note that all the algorithms given in this section retrieve and store elements of the matrix in the some order. (See Fig. 2.3.2, 2.4.2, 2.5.2, and 2.6.2.) Therefore, we recommend that matrices be always arranged in memory according to this particular ordering so that they can be accessed efficiently by any of the algorithms.

2.7.4. The Pivoting Problem

In Section 2.5 we assume that the matrix A has the property that there is no need of using pivoting when Gaussian elimination is applied to A. What should one do if A does not have this nice property? (Note that Gaussian elimination becomes very inefficient on mesh-connected processors if pivoting is necessary.) This question motivated us to consider Givens' transformation for triangularizing a matrix, which is known to be a numerically stable method. It turns out that, like Gaussian elimination without pivoting, the orthogonal factorization based on Givens' transformation can be implemented naturally on mesh-connected processors, although a pipelining implementation appears to be more complex.

2.8. Concluding Remarks

Research in interconnection networks and algorithms has been traditionally motivated by large scale array computers such as ILLIAC IV (see, for example, Kuck[5] and Stone [3]). The results presented in this section were, however, motivated by the advance in integrated circuit technology, though they are certainly applicable to parallel array computers. We have shown that many basic matrix computations can be done very efficiently by special purpose multiprocessors, which may be built cheaply using the current technology. The common feature of our algorithms is that their data flows are very *simple* and *regular*, and they are *pipeline algorithms*. We have discovered that some data flow patterns and interconnection schemes are fundamental for matrix computations. For example, the two-way flow on the linearly connected network is common to

both matrix-vector multiplication and solution of triangular linear systems (Sections 2.3 and 2.6), and the three-way flow on the hexagonally mesh-connected network is common to both matrix multiplication and LU-decomposition (Sections 2.4 and 2.5). A practical implication of this fact is that one device may be used for solving many different problems. Moreover, we note that almost all the processors needed in any of these devices are the inner product processor postulated in Section 2.2. A careful design for this processor is desirabe since it is the work horse for all the devices presented.

For the important problem of solving a dense system of n linear equations in 0(n) time on nxn mesh-connected processors, we have improved upon the recent results of Kant and Kimura [13]. The basis of their results is an theorem on determinants which was known to J. Sylvester in 1851. Their algorithm requires that the matrix be "strongly nonsingular" in the sense that every square submatrix is nonsingular. It is sufficient for our algorithms in Section 2.5 that the matrix be symmetric positive-definite or irreducible diagonally dominant.

We end this section by noting that processor communication will likely continue to dominate the cost of parallel algorithms and systems. Communication paths inherently take more space and energy than processing elements. We regard the problem of minimizing communication costs as fundamental. We hope the results of this section have demonstrated that the communication problem in parallel algorithms is not only tractable but also interesting. We expect that a large number of algorithms having small communication costs will be discovered in the future.

Personal supercomputer for only $100!

Clive Maxfield
10/24/2012 11:15 AM EDT

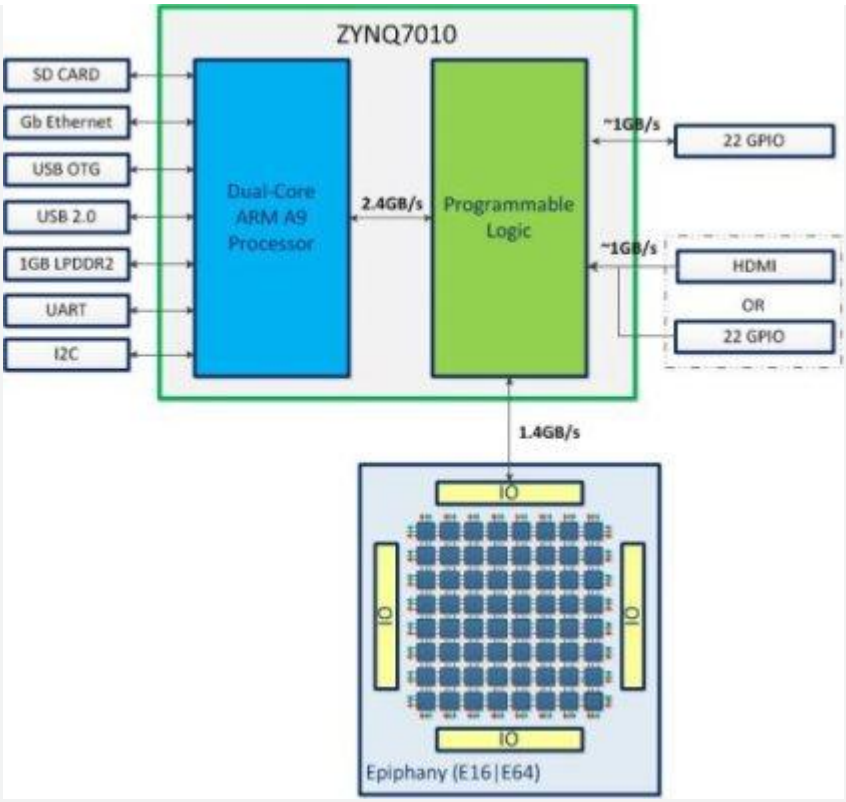
Some time ago I wrote a blog **From RTL to GDSII in Just Six Weeks** about a guy Andreas Olofsson who left his job, formed a company called **Adapteva**, and – working in his basement and living off his pension fund – single-handedly invented a new computer architecture. Andreas designed his own System-on-Chip (SoC) from the ground up – Including learning how to use all of the EDA tools – then took the device all the way to working silicon and a packaged prototype... and that's when things really started to get interesting!

The chip that Andreas designed is called the **Epiphany**. This is an array of processor cores, each equipped with its own local memory and a single-precision floating-point engine. Everything is designed so as to offer optimum performance while consuming as little power as possible. Epiphany is extremely scalable – The Epiphany-III (implemented at the 65nm node) boasts an array of 16 processors, while the Epiphany-IV (implemented at the 28nm node) features an array of 64 processors.

The end result is that, when operating at peak performance, running at 800MHz, the Epiphany-IV offers 100 Gflops of raw computing power while consuming only 2W. This means that, at 50Gflops/Watt, the Epiphany-IV is 50 to 100X more efficient than anything else out there.

Well, I just heard from Andreas. His current project is to create an open source personal supercomputer platform that anyone can buy for only $100, and that can be used to implement the most compute-intensive tasks like embedded and robotic vision, software-defined radios, and … well, almost anything really.

This supercomputer, which is called the **Parallella**, is based on a combination of the **Zynq-7000 All Programmable SoC** from **Xilinx** and the **Epiphany** from **Adapteva** as illustrated in the block diagram below.



The Zynq-based Parallella personal supercomputer

Initially there will be two versions of this little beauty -- the version equipped with an Epiphany E16 (16 cores) will cost only $100, while the version equipped with an Epiphany E64 (64 cores) will cost only $199. I'm told that, even when running flat out, the Parallella equipped with an Epiphany E64 will consume as little as 5W!

The guys and gals at Adapteva are currently using a Zynq evaluation board to extensively prototype the user experience of the Parallella boards. In our chat earlier, Andreas told me: *"The user experience of running Ubuntu (one of the more popular flavors of Linux) on the Zynq is fantastic!"* The picture below shows Andreas' Zynq evaluation board with an Epiphany daughter card plugged in via one of the FMC connectors.

.

A Zynq development board with an Epiphany daughter card

Are you familiar with **Kickstarter.com**? This is a funding platform for creative projects -- everything from films, games, and music to art, design, and technology. If people like a particular project, they can pledge money to make it happen. It's only if the project succeeds in reaching its funding goal that the backers' credit cards are charged -- if the project falls short, no one is charged.

The point is that Andreas and the folks at Adapteva have set Parallella up as a Kickstarter project. If you are interested, you can **click here** to learn more and -- if you wish -- make a pledge. Pledges can be as little as $15 or as much as $10,000 or more.

In order to proceed, they need to raise $750,000 by the Kickstarter deadline of Saturday 27 October at 6:00 p.m. Eastern Daylight Time. I personally have every confidence that if they get the money they will succeed. After all, this project is led by the man who single-handedly designed a silicon chip in his basement.

It's not often you get a chance to really "make a difference" in this world. **I just pledged $99 myself**. For this, when the project succeeds, I will receive my own Epiphany E16-based Parallella loaded with all of the development tools required to implement almost any project of my dreams. What say you? Are you with me?

If you found this article to be of interest, visit **Programmable Logic Designline** where – in addition to my **Max's Cool Beans** blogs – you will find the latest and greatest design, technology, product, and news articles with regard to programmable logic devices of every flavor and size (FPGAs, CPLDs, CSSPs, PSoCs...).

Comments

: 10/24/2012 3:58 PM EDT

Its very interesting, but i'm skeptical of the usefulness. The thing is that the cores are going to be starved for data. Maybe you can pick a few specific applications where this may not be the case, but in general you dont just process the same data over and over. If you look at the architecture you have coherency problems and bandwidth problems. If you were to analyze many applications, many of the cores would just be idle waiting for data input or output. Also the program(s) running on the cores need to be relatively small. I mean all cores can see what the others are doing, but how do you manage that? Hence the result really expensive super computers ...

Reply

10/24/2012 4:08 PM EDT

Agreed, bandwidth CAN be a killer, but there are plenty of applications that require a massive amount of processing per byte. Here are some of the applications we think the Parallella would be great at:

face detection
face recognition
finger print matching
object tracking
pattern matching
optical flow
content based image retrival
signature verification
optical character recognition
automated optical inspection
number plate recognition
stereo vision
gesture recognition
people counter
remote sensing
velocity moments
visual world
image stabilization
iris matching
object classification
video analytics
manufacturing inspection
augmented overlay
synthetic aperture radar

hyperspectral imaging
IR imaging
smart stream compression
large focal array sensor imaging
fractal compression
optical flow
autonomous flight
landmine detection
GNU radio
cognitive radio

Complete list here:
http://www.adapteva.com/white-papers/104-parallel-computing-projects-for-next-summer/

10/25/2012 3:33 AM EDT

The first WANT-NOW app for this beast should ofcourse be a FPGA sim,synthesis and routing tool!

(Does anyone work to do that with CUDA yet?)

Whoever comes first, let me know and I'll throw my money at you! :)

Reply

10/25/2012 1:15 PM EDT

Check out my blog on this and related topics at All programmable Planet:
http://www.programmableplanet.com/author.asp?section_id=2141&doc_id=253083

10/24/2012 3:59 PM EDT

Someone just emailed me to say: "If you consider 16 or 64 cores a SuperComputer then what is this one with 144 that is shipping now? http://www.greenarraychips.com

There is more to this than just core count, like interconnections. Can we make a 4D-HyperCube like we can with the XMOS (decedents of Imos Transputers)? http://www.xmos.com/resources/xkits?category=XK-XMP-64+Development+Board

I replied "I think the main point here is that a lot of today's really compute-intensive tasks require floating point capability -- to the best of my knowledge, products like Green Arrays and XMOS don't support floating-point."

Reply

10/24/2012 5:19 PM EDT

Max, Thank you for the really kind article! Just want to clarify that I really only designed the first chip myself. The last three chips were a complete team effort, with Roman Trogan being in charge of chip design and Oleg Raikhman in charge of verification and programming tools integration. I supported them from time to time, but spent most of my time failing at fundraising, selling, and marketing..

Reply

10/24/2012 7:36 PM EDT

This is a very interesting project. Best of luck to you Andreas!

Reply

10/24/2012 11:38 PM EDT

Can this be used for finite element numerical simulations: there are many simulation tools for semiconductors, materials, meteology, geology... they run very slow even on multi-core PC.

10/25/2012 10:04 AM EDT

With the right software, we numerical simulations could be a great fit. The challenge right now is that the software infrastructure for parallel programming still needs a lot of work. That's one of the driving reasons for starting this project. Ironically, the challenge of boot strapping ubiquitous parallel programming is a serial process.

10/25/2012 9:46 PM EDT

Thanks for the explanation, I will try to understand it as a layman of Computer engineeing: are you saying that some commercial simulation tools still can't run on this supercomputer? Such as Ansys, Silvaco...these are popular simulation tools for semicopnductor.Is it possible to make them run in the near future?

10/25/2012 3:40 AM EDT

This is excelent and a great bang for the buck IMHO. This is whether you are a believer in this kind of multicore approach or not. At the very least you can see the board as a Zync-7000 development board as well, which the cheapest I could find was around 300 bucks (albeit a stronger sibbling of this FPGA, SoC, whatever...). As a (big) bonus you have this nice parallel core (the Epiphany)that you can play with, and who knows what kind of applications can be devised that can make a very good use of it. The sky (imagination) is the limit! :-)

10/25/2012 10:09 AM EDT

Thank you. Yes, we got lucky with our choice of the Zynq, it has generated an incredible amount of really positive interest.(not even related to the goal of this project:-)) I guess that's what they call "fortuitous serendipity".

Reply

10/25/2012 4:52 AM EDT

This is very interesting.
I myself have just finished developing a 64-processor chip targeted at Ethernet packet inspection and filtering.
The processor cores are optimised hardware implementations of the "Berkeley Packet Filter" processor.
Ref:
http://en.wikipedia.org/wiki/Berkeley_Packet_Filter
http://www.tcpdump.org/papers/bpf-usenix93.pdf

The 64-processor cores are implemented on a Xilinx Virtex-6 FPGA and makes good use of its DSP48E1 primitives and on-chip block-rams to achieve single-cycle operation for most instruction op-codes.

This allows 4x10Gbps of Ethernet packets to be inspected, analysed and filtered at full-line rate on the chip.

This means you can now replace a full rack of servers with a single PCIe card.

Here is the finished product:

http://www.telesoft-technologies.com/images/docs/DX-OEM-GEN-MK-DS-33862-02-MPAC-IP-6010-4x10GbE.pdf

This product has applications in:
Cyber security
Network intrusion detection (IDS)
Lawful intercept
Virus Signature Detection
etc.

10/25/2012 11:17 AM EDT

This is very interesting. I have an assortment of platforms: Arduino Uno, Raspberry Pi, Altium NanoBoard and have just ordered an Arduino Due.

To me this is just as exciting as the January '75 Popular Electronics article introducing the Altair 8800. I ordered one right away and nothings been the same since.

My interests have included machine vision and the platforms I have now, except maybe the NanoBoard, are totally inadequate.

As soon as I figure out how I will cough up the $99 donation.

10/26/2012 7:29 AM EDT

Big processing power at 5watts power consumption. Initially there will be lot of requirement for the applications in the mobile plate form. Later on desk top systems also.Probably after its launch this will be tuned up further with feed back from the users.

10/26/2012 2:11 PM EDT

The Adapteva cores look interesting. I think I would like to implement similar, very minimalist architecture in FPGA on my Altium NanoBoard

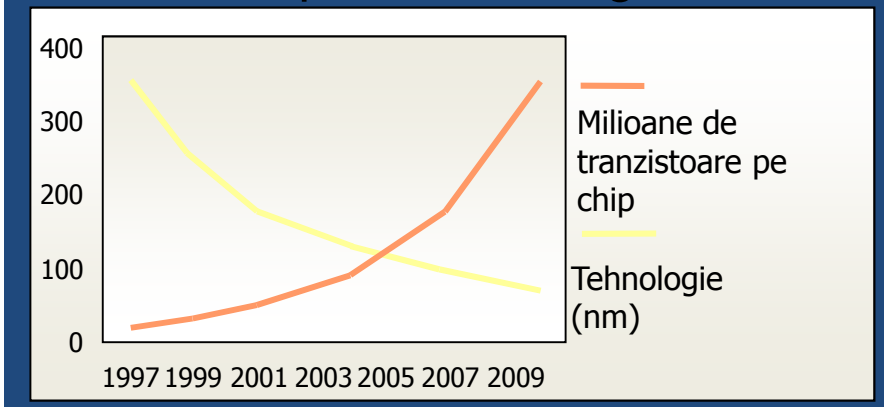
Sisteme Incorporate

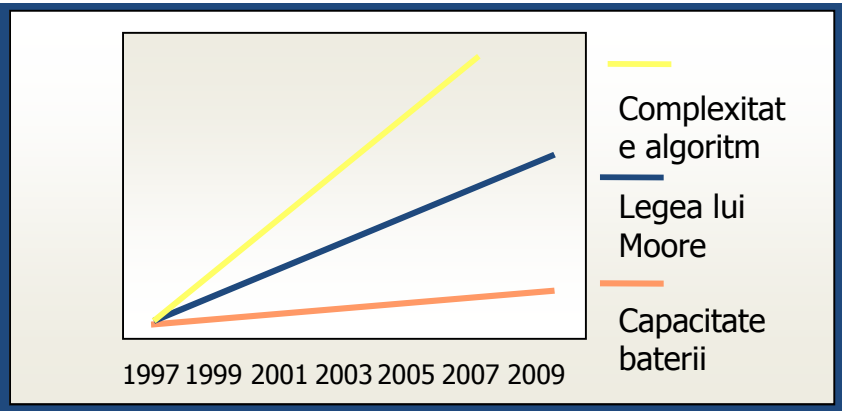
Cursul 11

Sisteme Reconfigurabile

De ce avem nevoie de reconfigurabilitate?

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





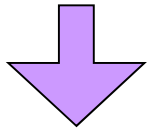
- Complexitate crescuta a algoritmilor
- Limitari puternice a capabilitatilor bateriilor

Se doreste
si performanta
si flexibilitate

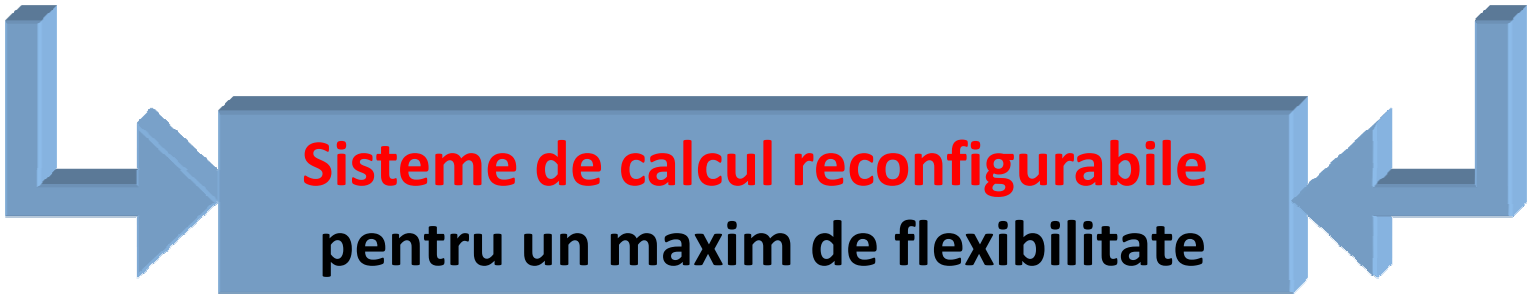
Constrangeri severe pentru consumul de energie

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)

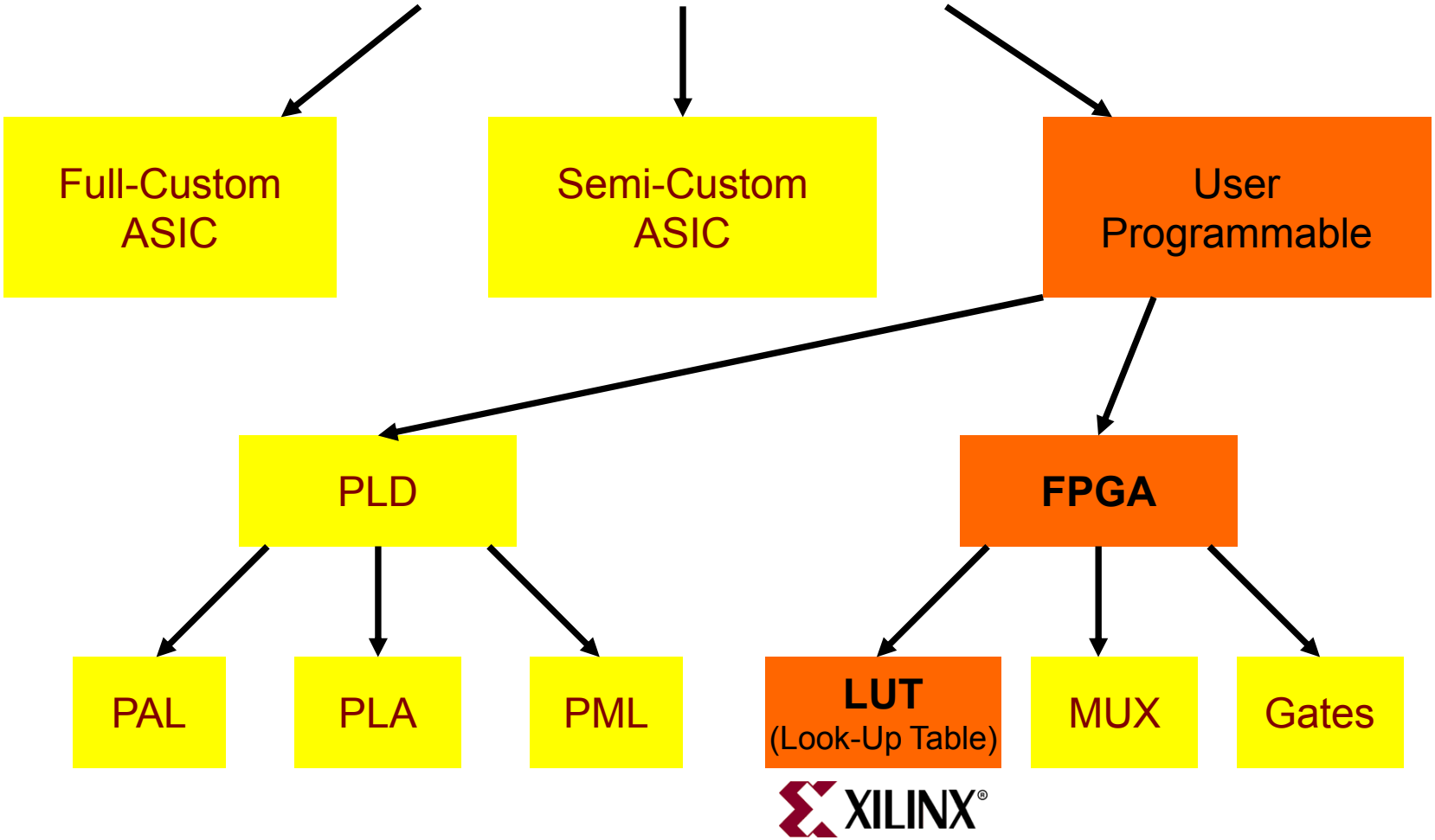


Se pot obtine imbunatatiri semnificative daca functionalitatea procesoarelor embedded este extinsa cu functii specifice aplicatiei



Tipuri de circuite integrate

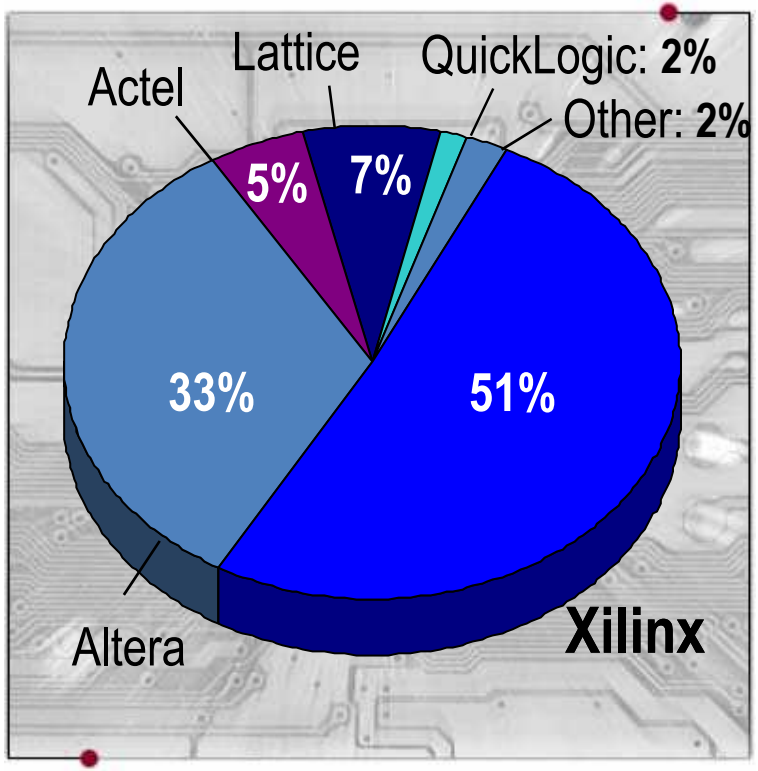
Integrated Circuits



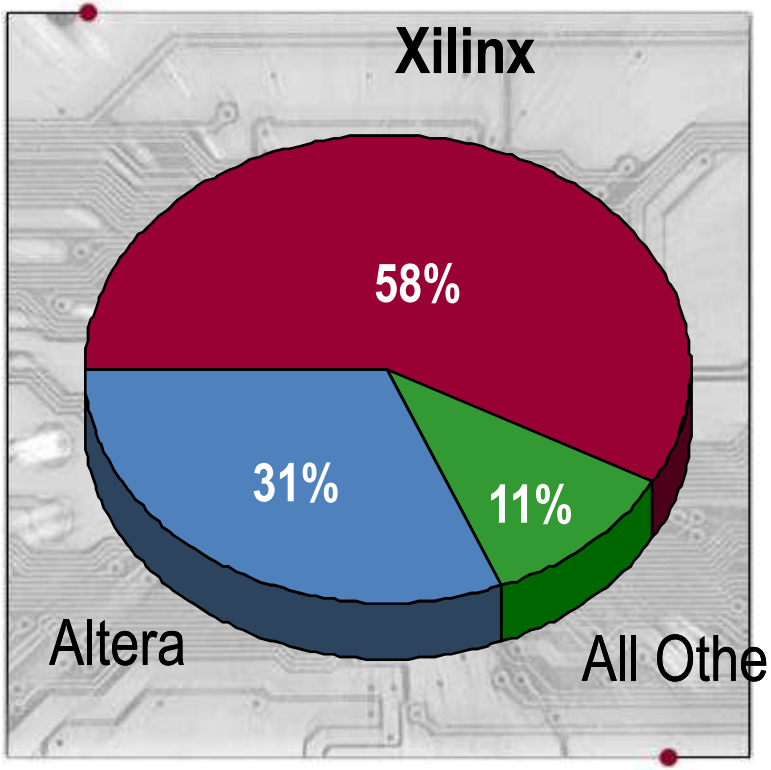
The Programmable Marketplace

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

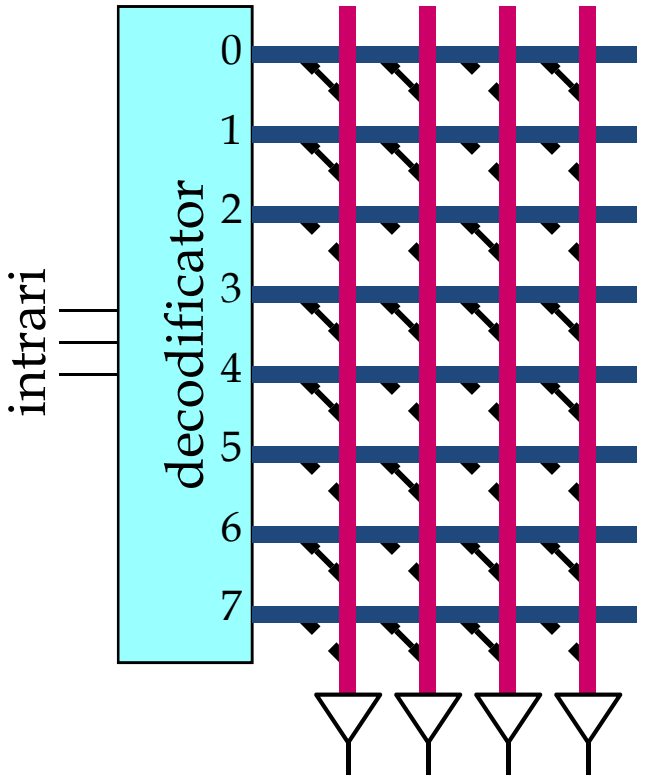
- reconfigurat de fiecare data de catre proiectanti
- nu se proiecteaza un layout fizic al componentelor. Proiectarea are ca rezultat un bitstream cu care se configureaza dispozitivul.

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

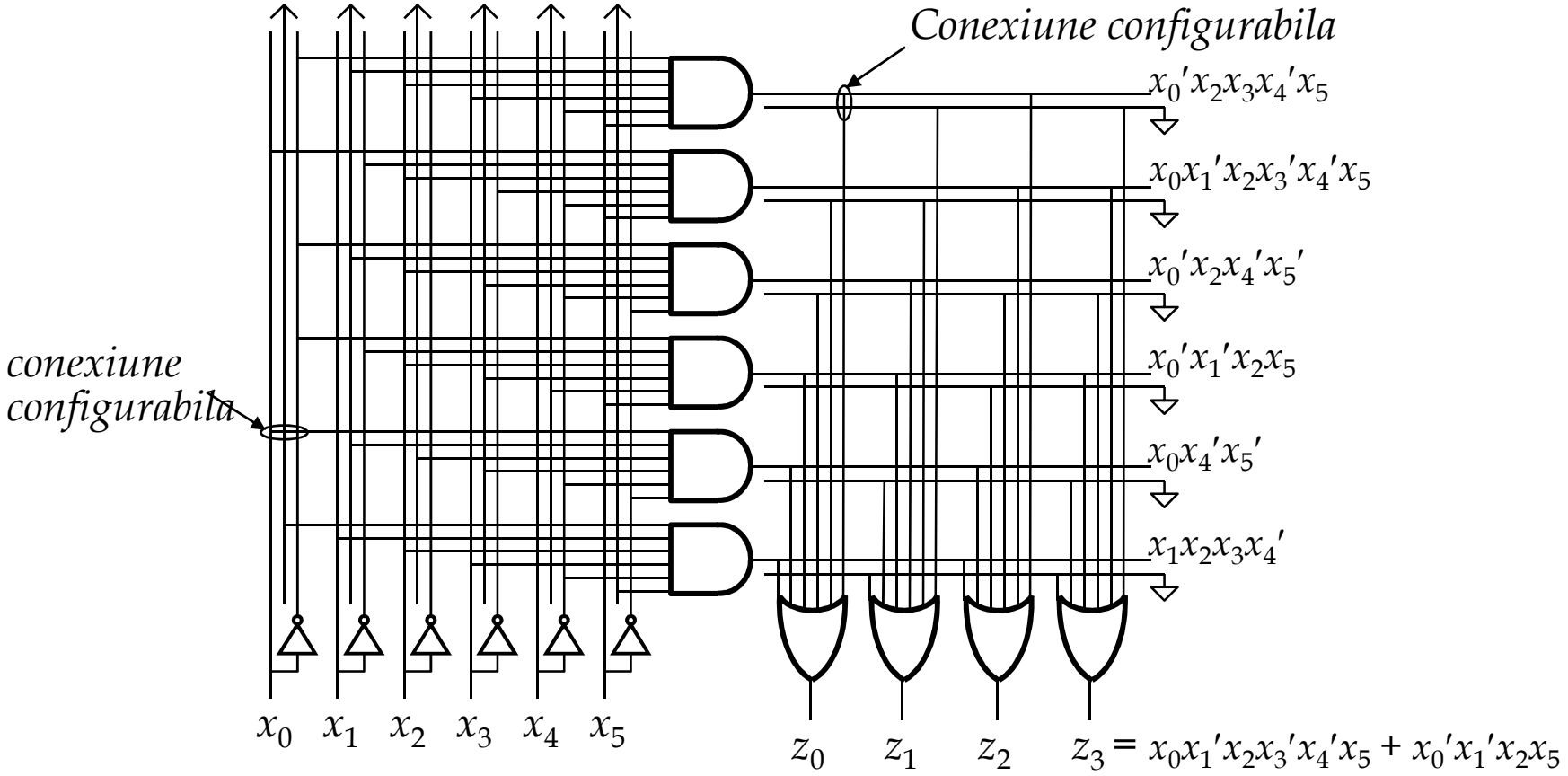


- Unele PROM-uri sunt scrise prin eliminarea conexiunilor
  - » Tensiune mare aplicata pe linia si coloana pe care se vrea marcarea unui 0 logic
  - » Curentul mare aplicat legaturii linie-coloana determina “arderea” legaturii
- Alte memorii pot fi arse si reprogramate (EPROM, EEPROM)

Logica in RAM/ROM

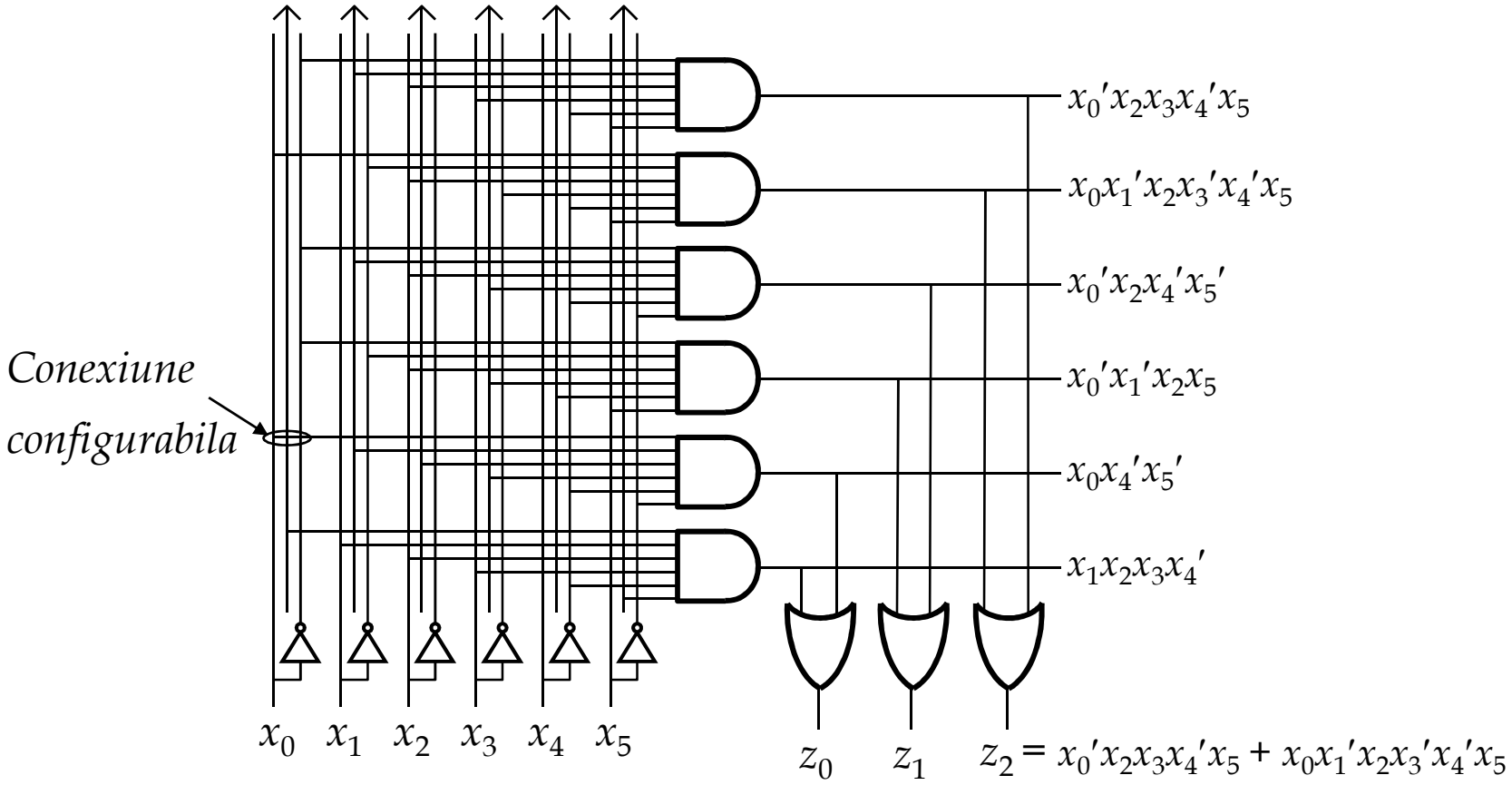
- Orice memorie RAM poate implementa functii logice reconfigurabile
  - Stocheaza tabela de adevar a functiei logice in memorie
  - Exemplu: folosesc RAM de 4 biti pentru a simula o poarta SI cu doua intrari stocand 0 la adresele 00, 01 si 10 si 1 la adresa 11
  - cu 2^m cuvinte de 1 bit se poate implementa 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)

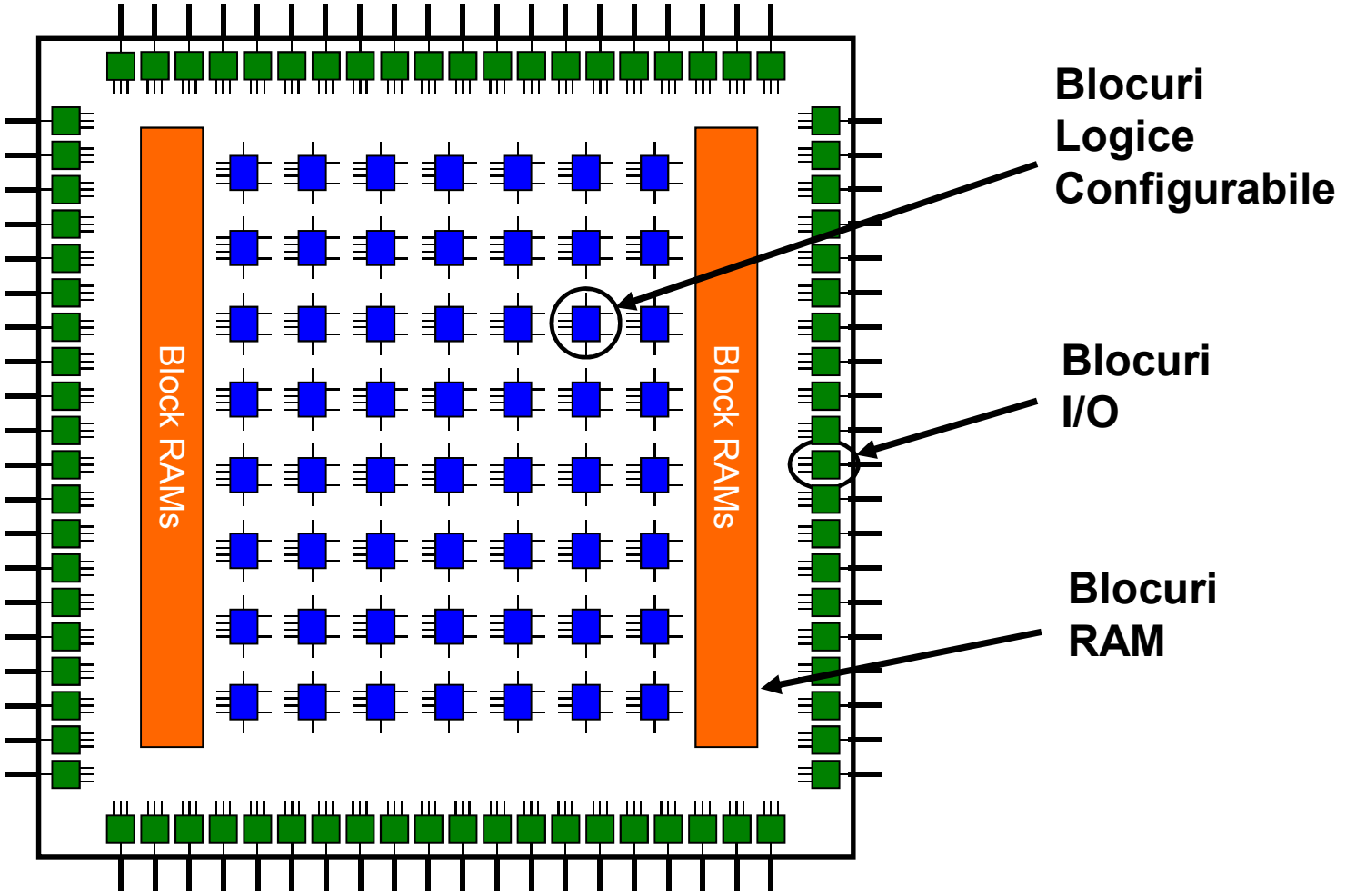


- PAL este similar cu PLA dar are un plan SAU fix.
- Mai usor de programat (si mai ieftin de produs)
- Dezavantaj: numar limitat de termeni generati la iesire

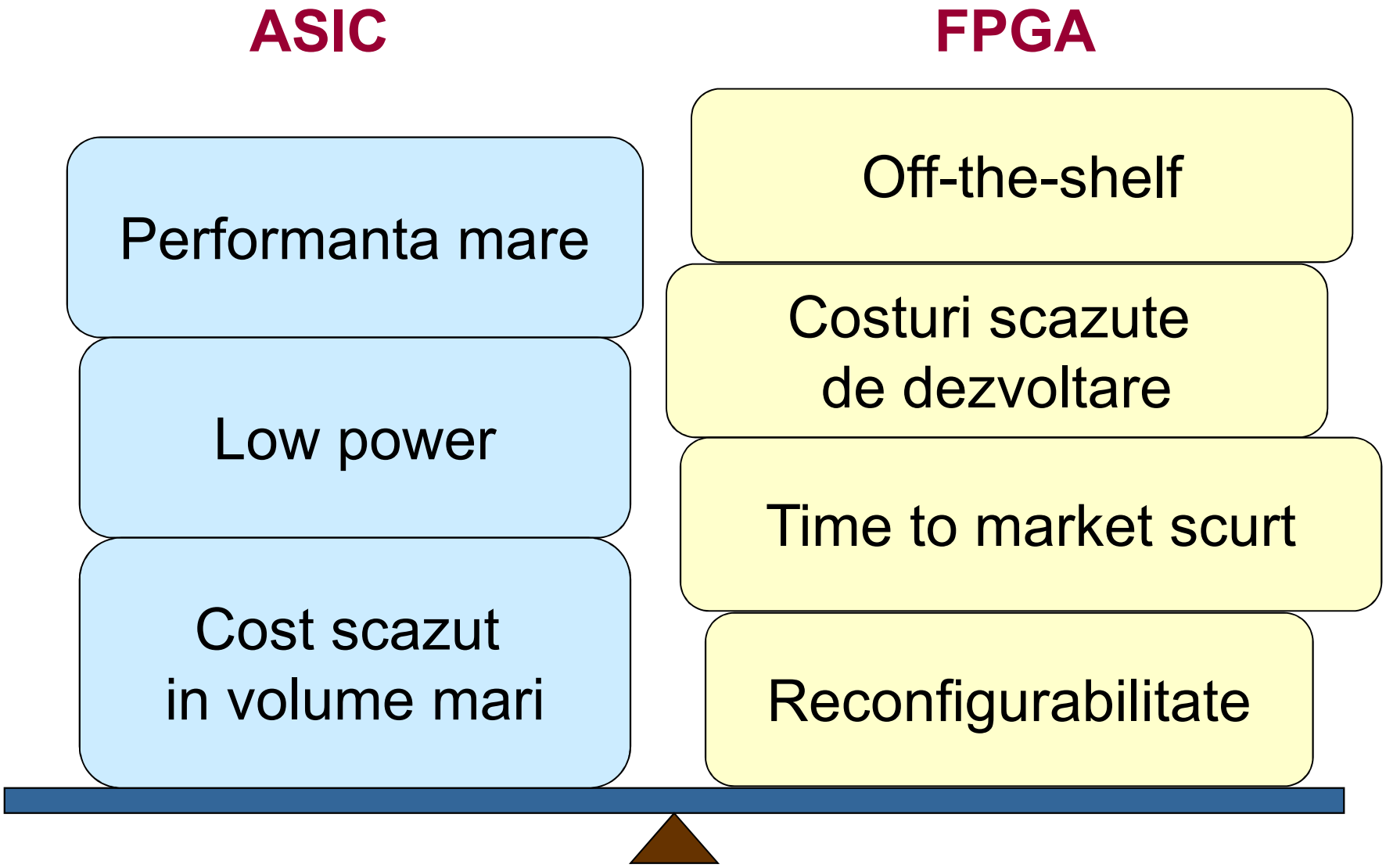
Field Programmable Gate Array (FPGA)

- 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?

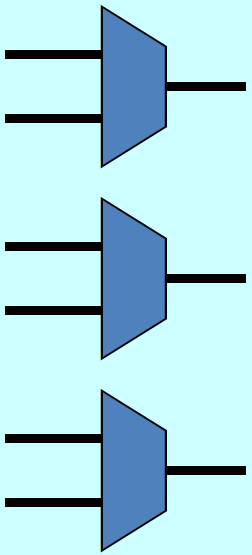


Alte avantaje FPGA

- 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

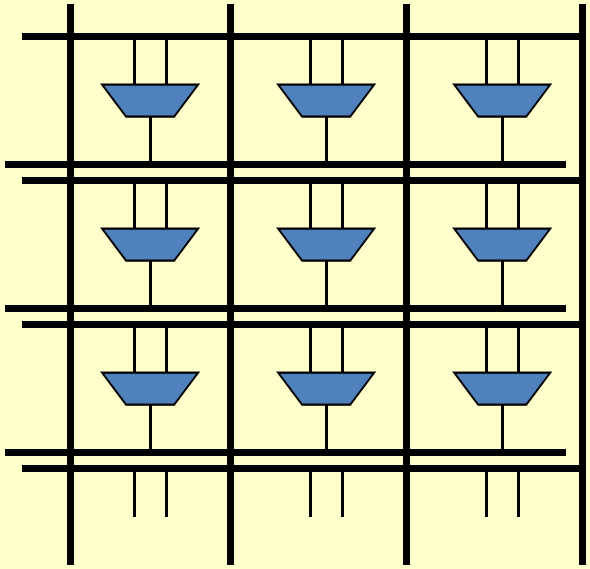
Latima de banda computationala

Fixa



CPU

“Fara limite”



RC

Producatorii majori de FPGA

FPGA SRAM

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

} **Aproape 60% din piata**

- Atmel
- Lattice Semiconductor

FPGA Flash

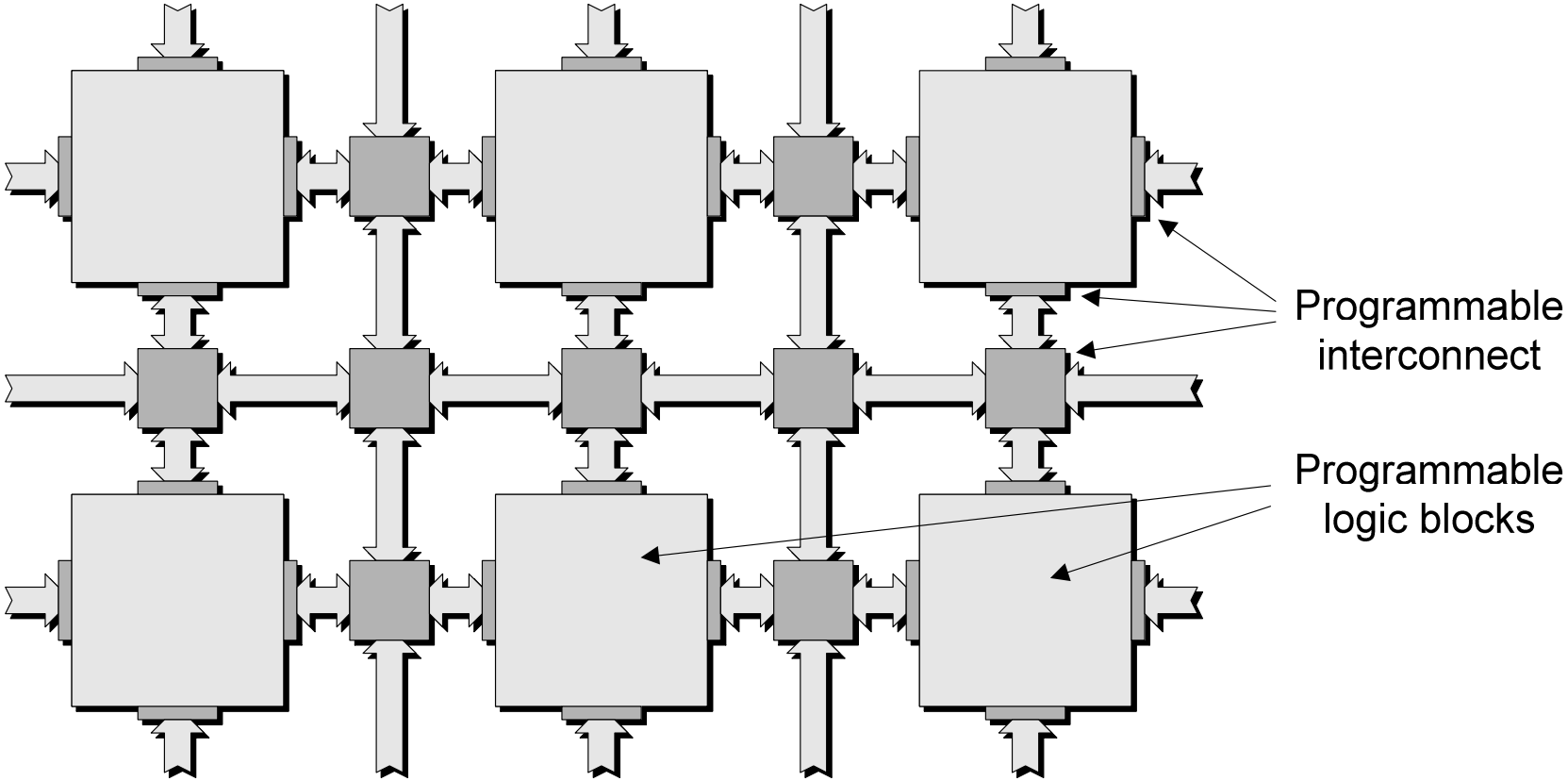
- Actel Corp.
- Quick Logic Corp.

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**

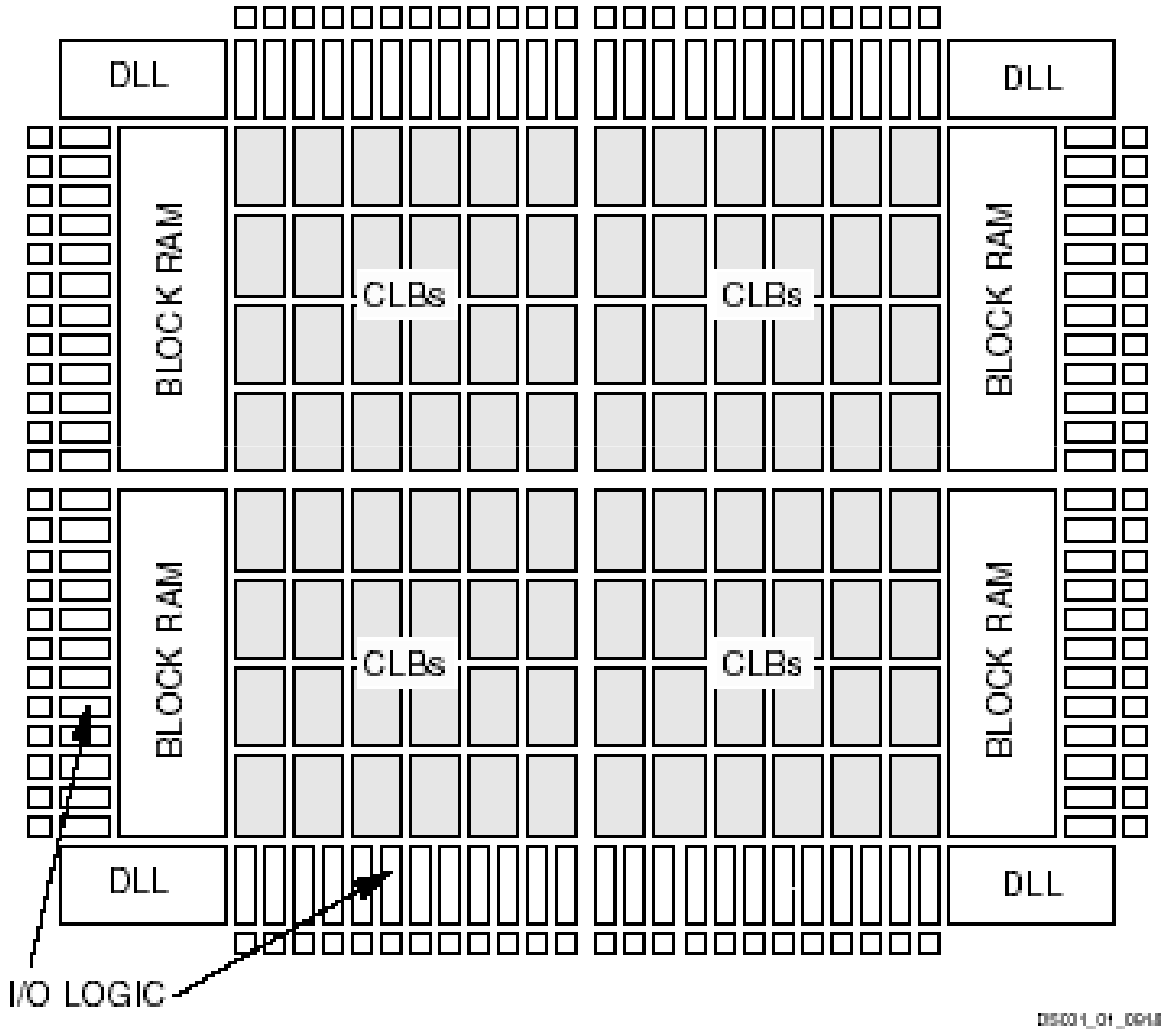
Prezentare

Structura generala a unui FPGA

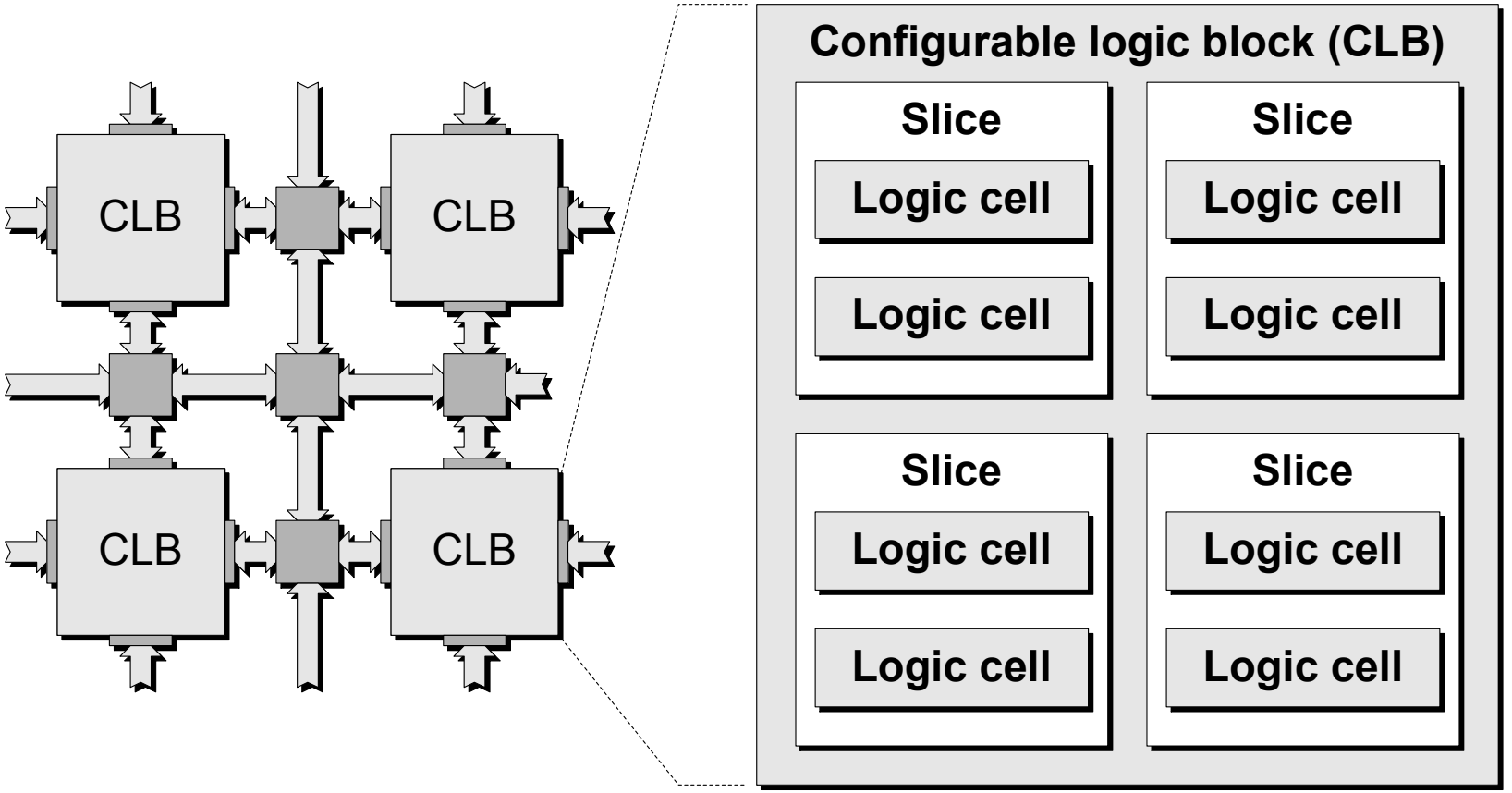


The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Xilinx FPGA Block Diagram

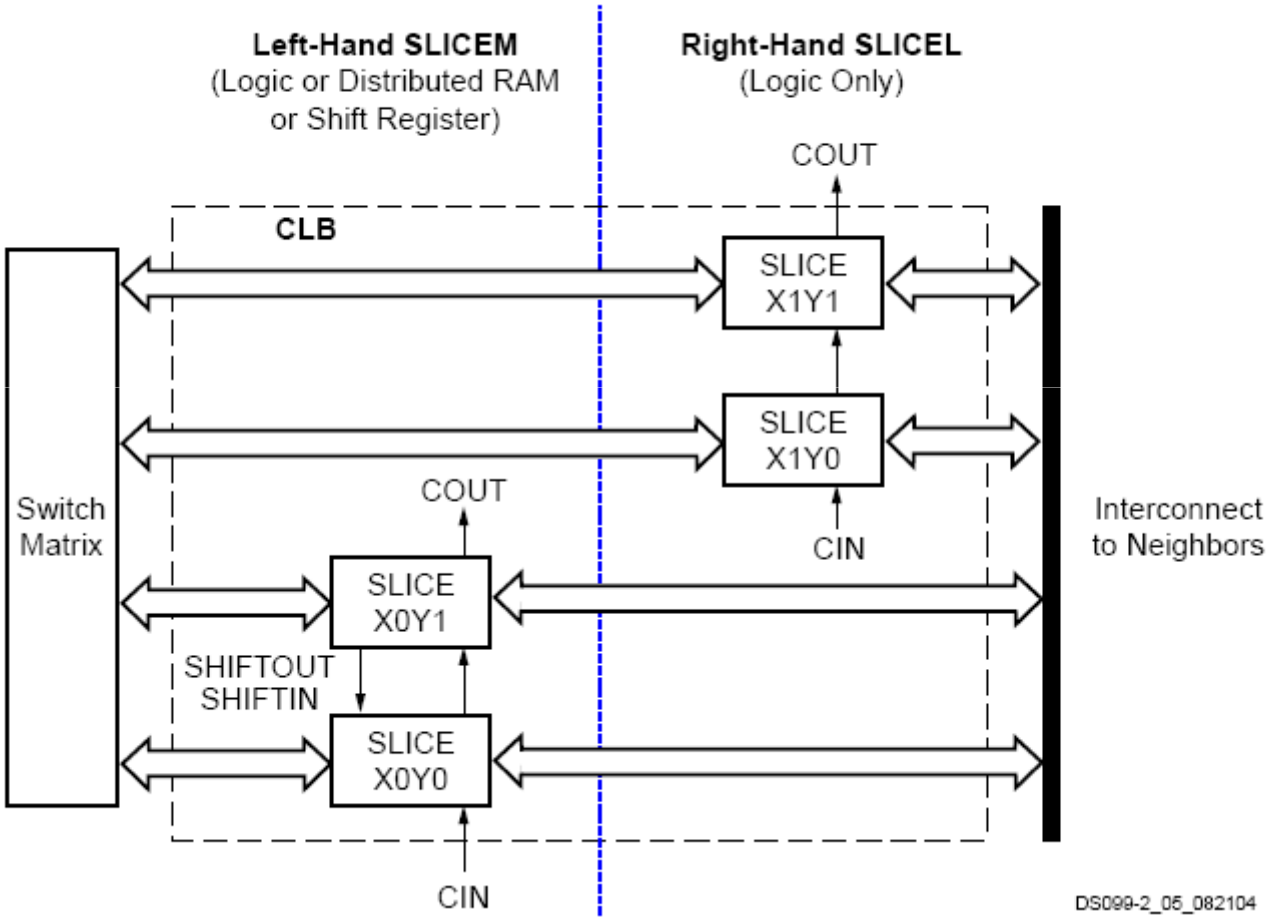


Xilinx CLB

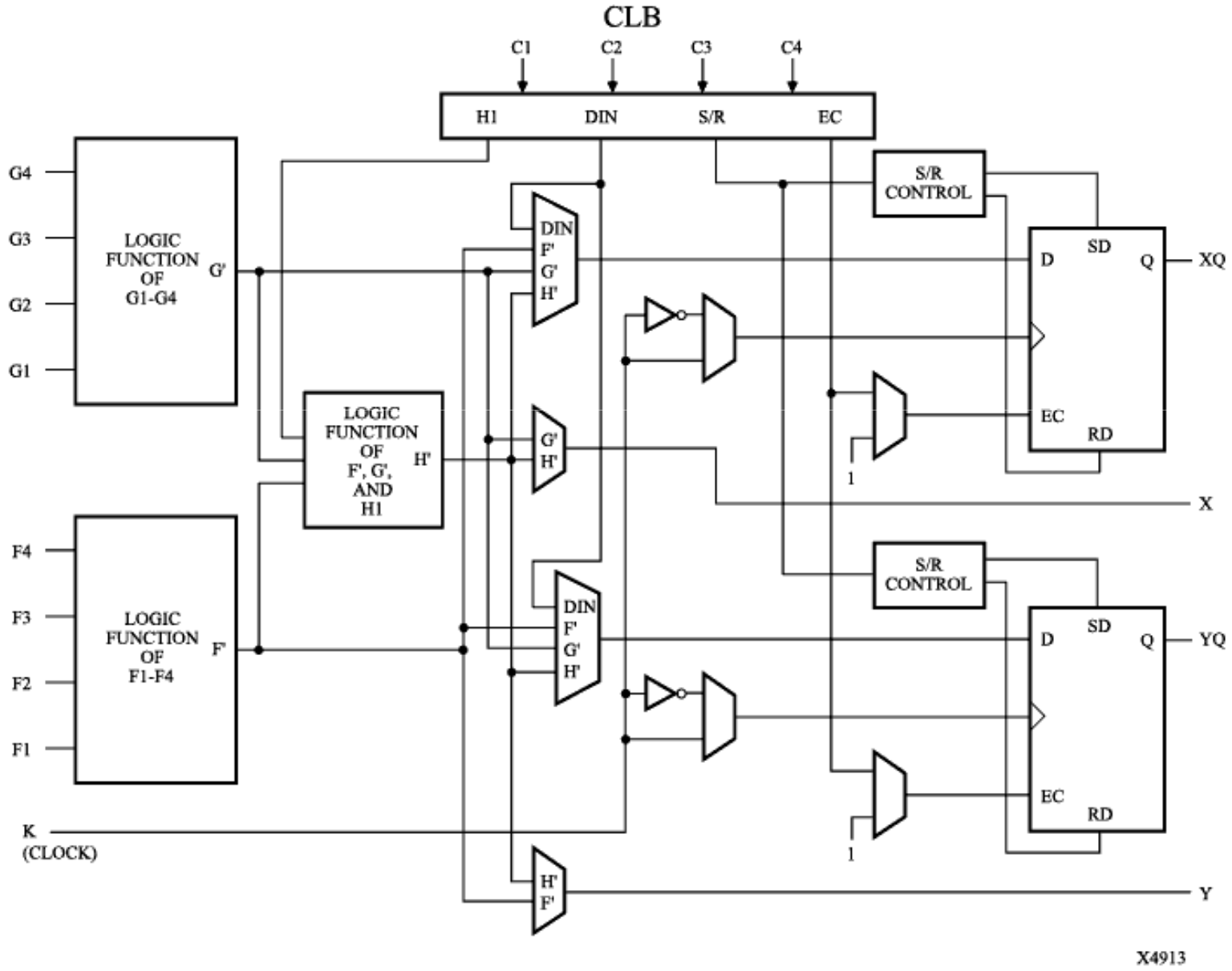


The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Configurable Logic Block CLB

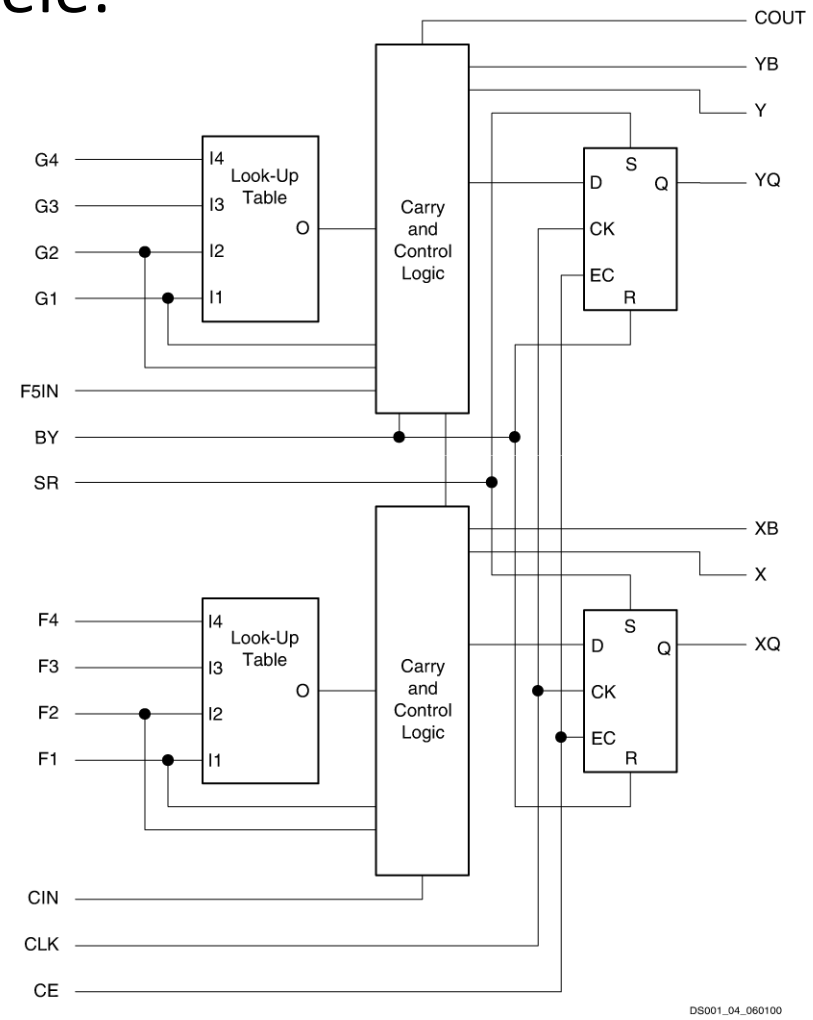


CLB Slice


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

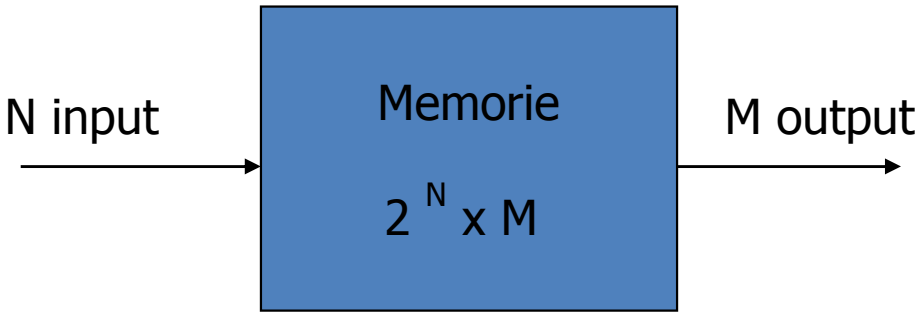
- Fiecare slice contine urmatoarele:
  - LUT cu patru intrari
    - Orice functie logica cu 4 intrari,
    - sau RAM 16bit x 1
    - sau Registru Shift pe 16 biti
  - Carry & Control
    - Logica aritmetica rapida
    - Logica pentru inmultire
    - Logica de multiplexare
  - Elemente de stocare
   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



Functiile Logice

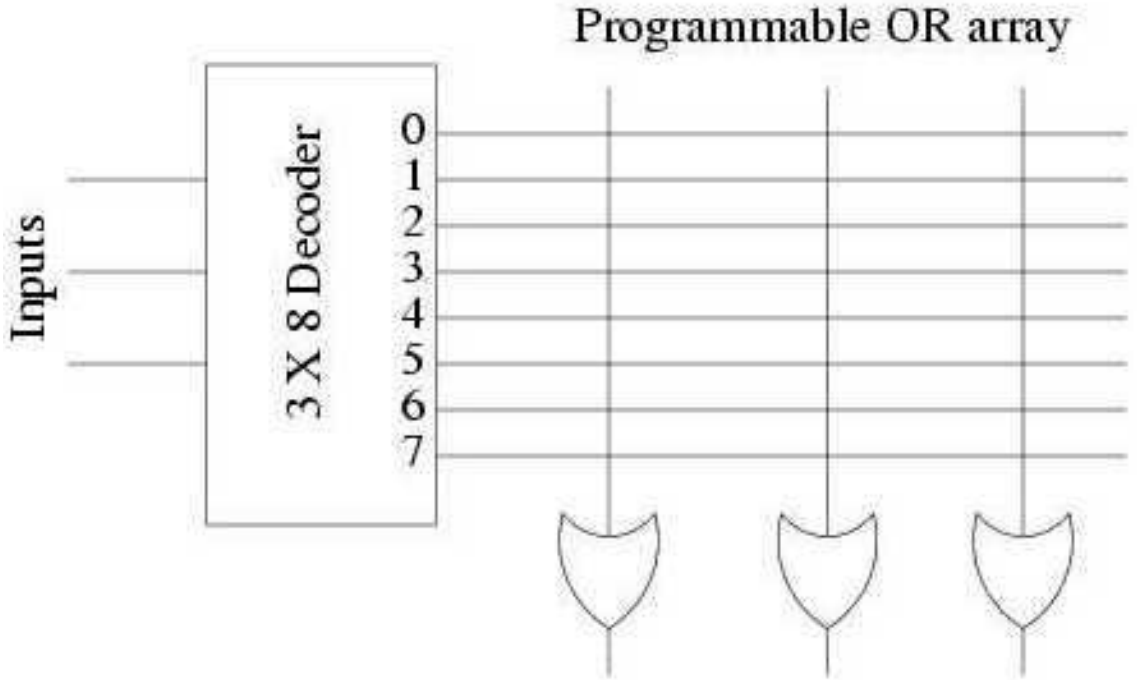
- Implementate folosind Look-up Table (LUT)
- Un LUT cu k intrari corespunde unei memorii de 2^K x 1 bit
- Un LUT cu k intrari poate implenta orice functie logica cu k intrari si o singura iesire function

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

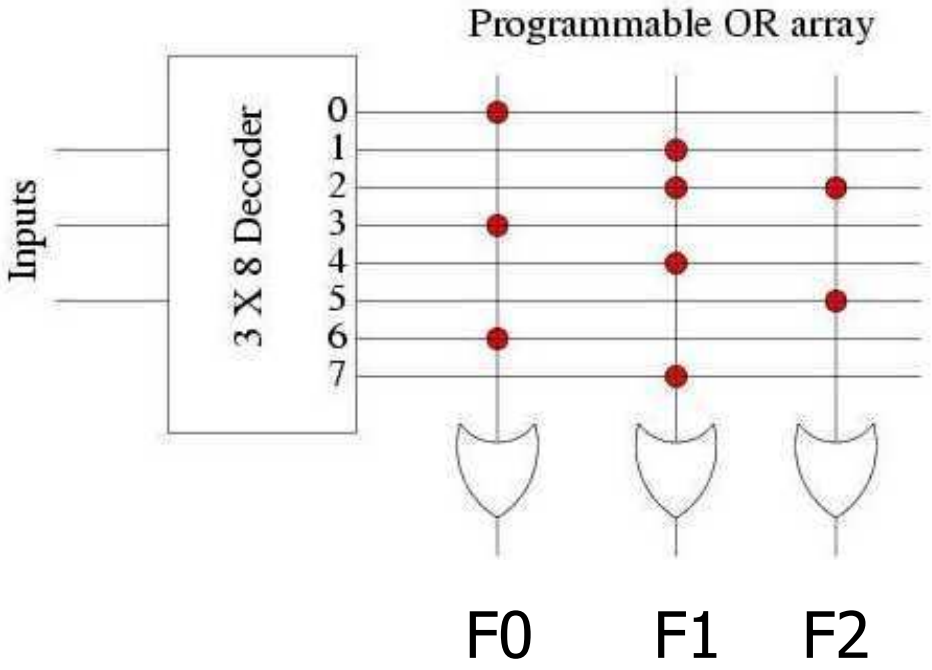
Diagrama logica a unui LUT 8x3



Suma de mintermeni

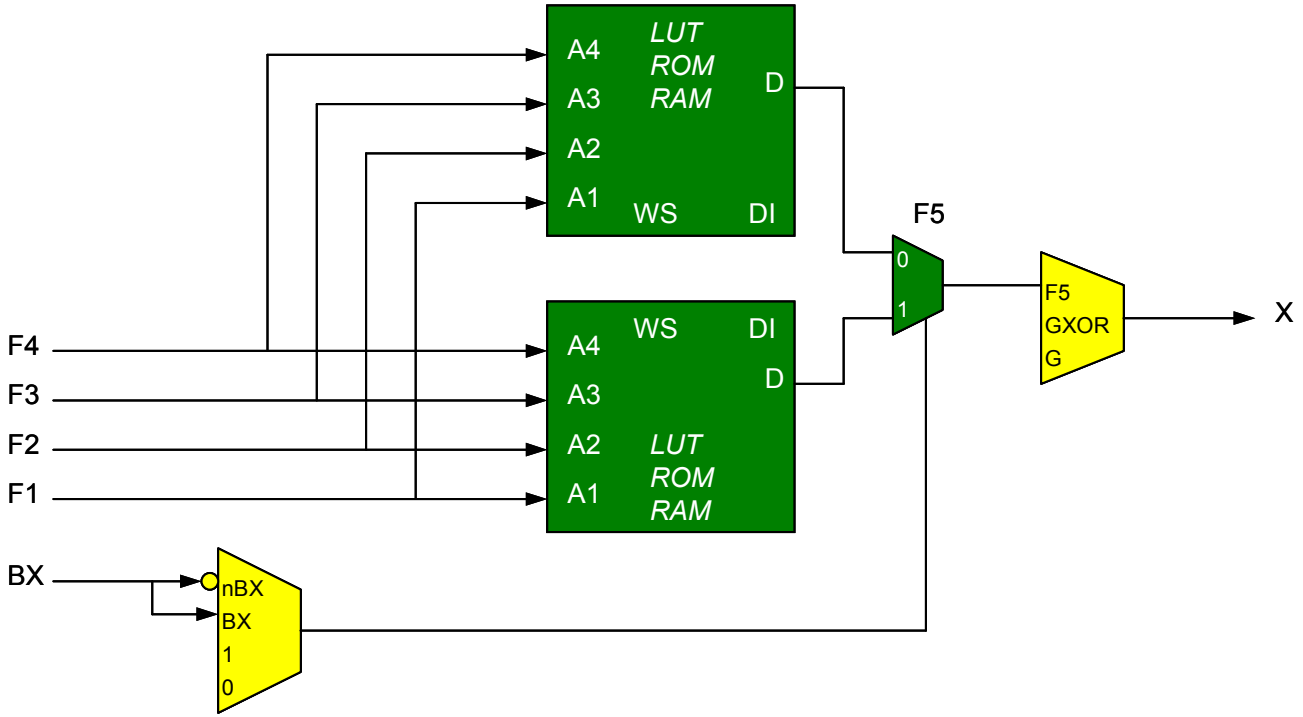
Implementarea unei functii logice folosind LUT

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

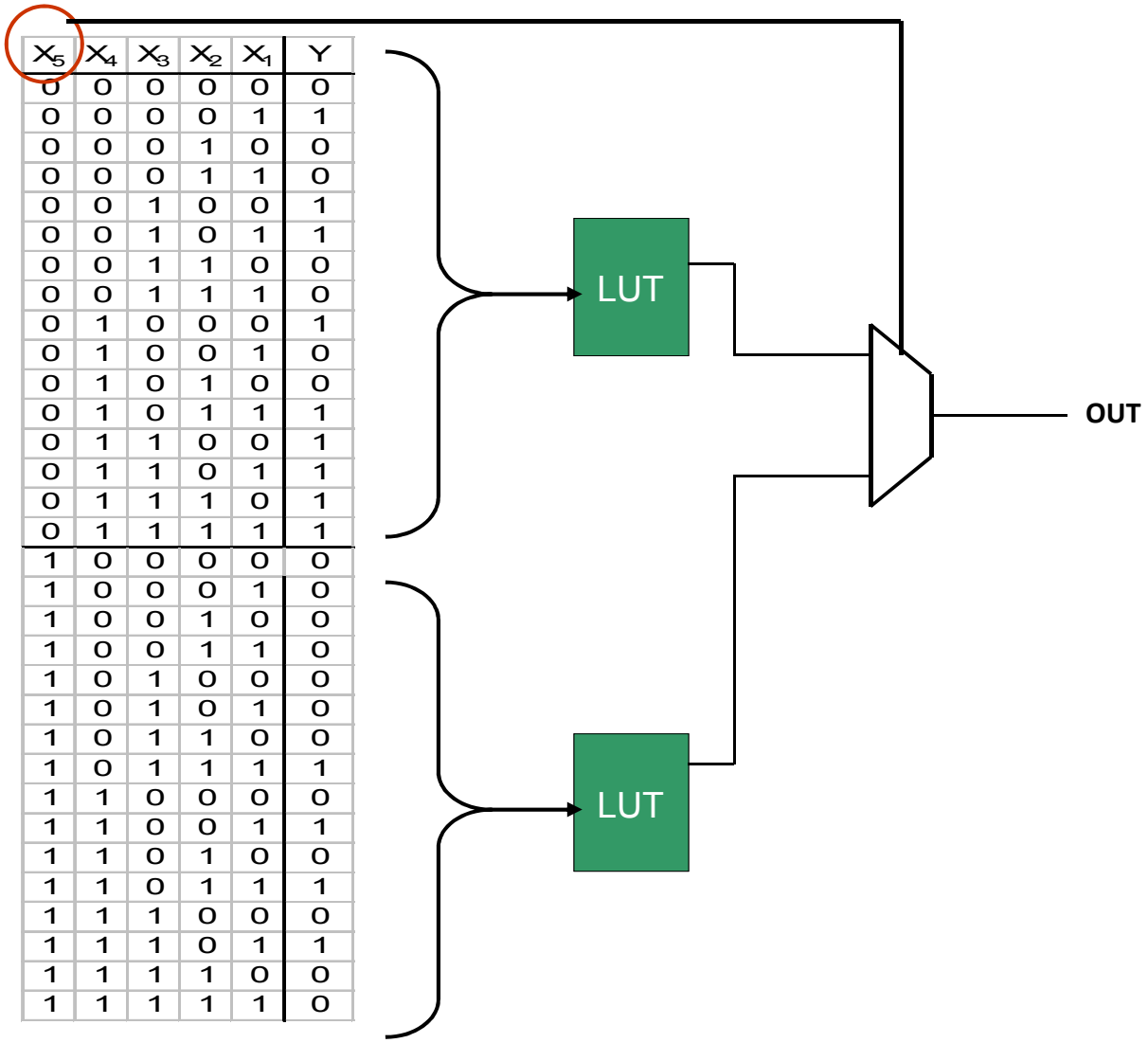


Implementarea unei functii logice cu 5 intrari folosind 2 LUT

- 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

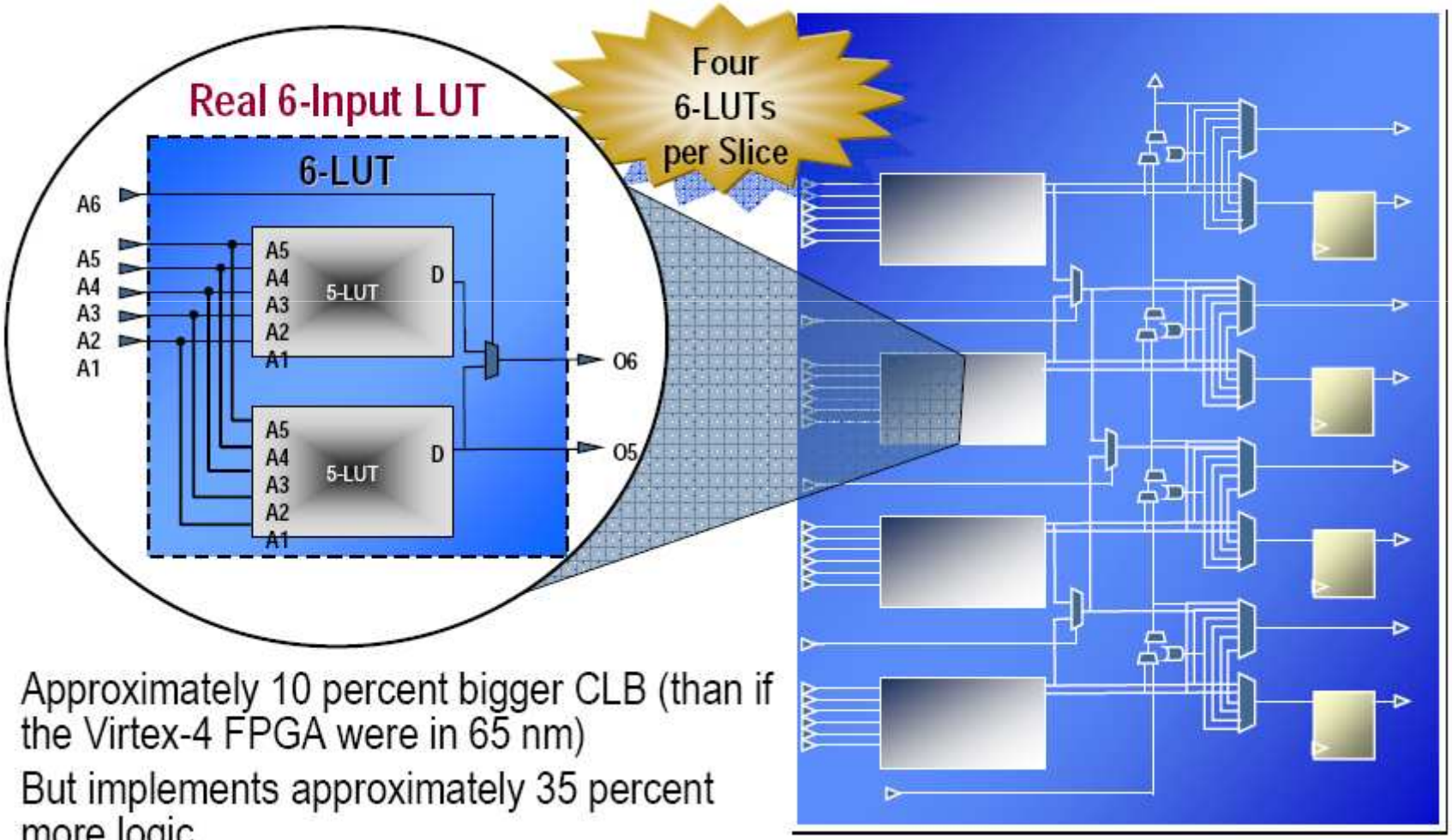


Implementarea unei functii logice cu 5 intrari folosind 2 LUT



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

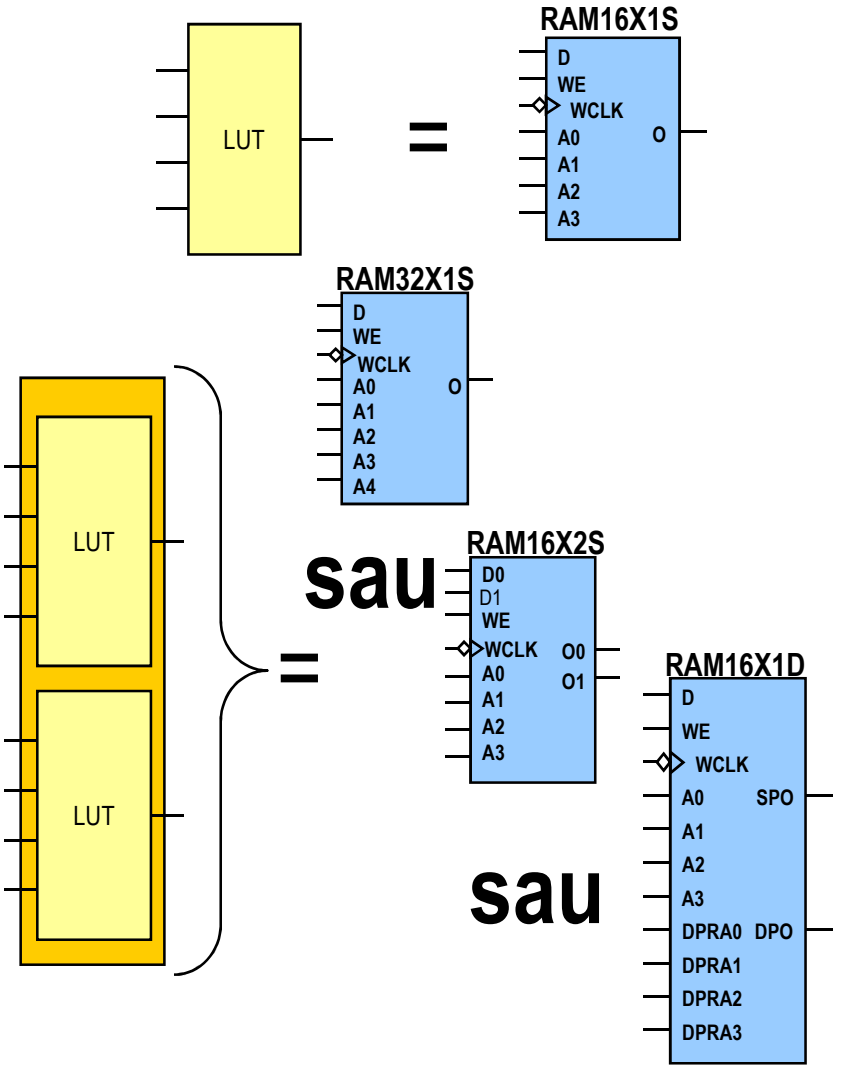
Exemplu Virtex



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

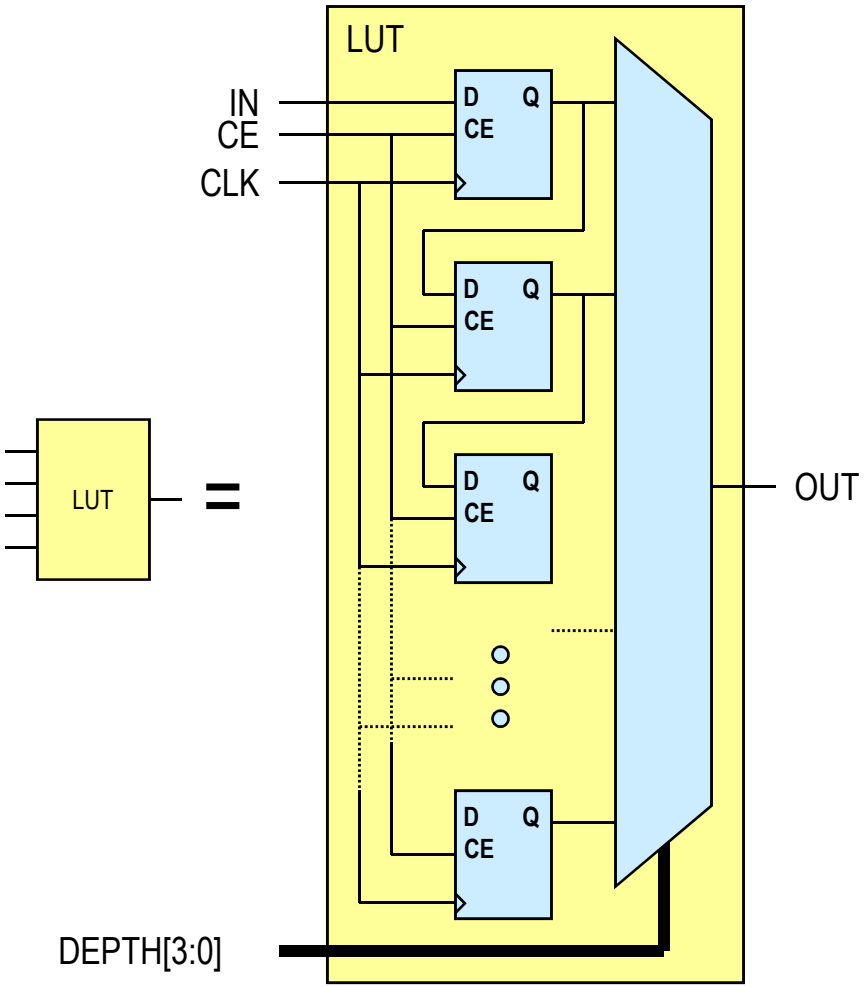
RAM Distribuit

- 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

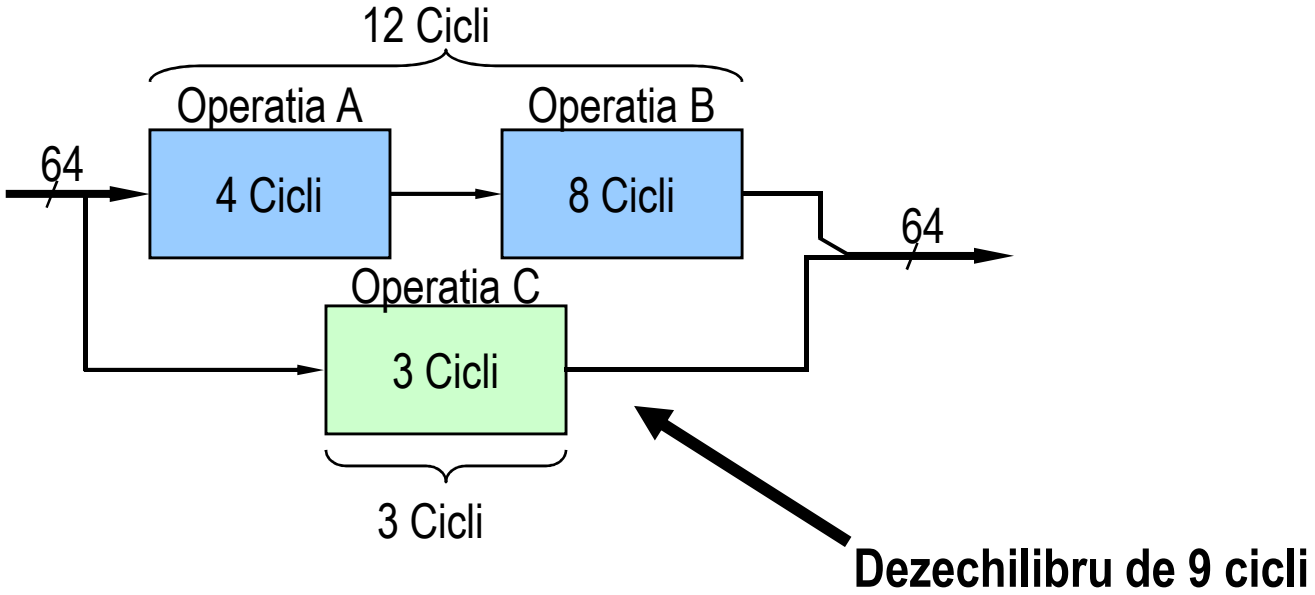


Registru Shift

- 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



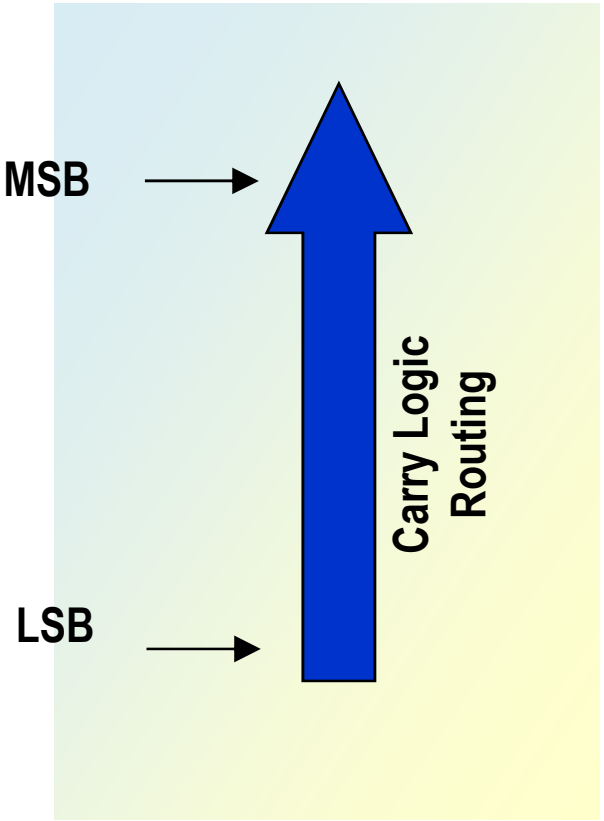
Registru Shift



- 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



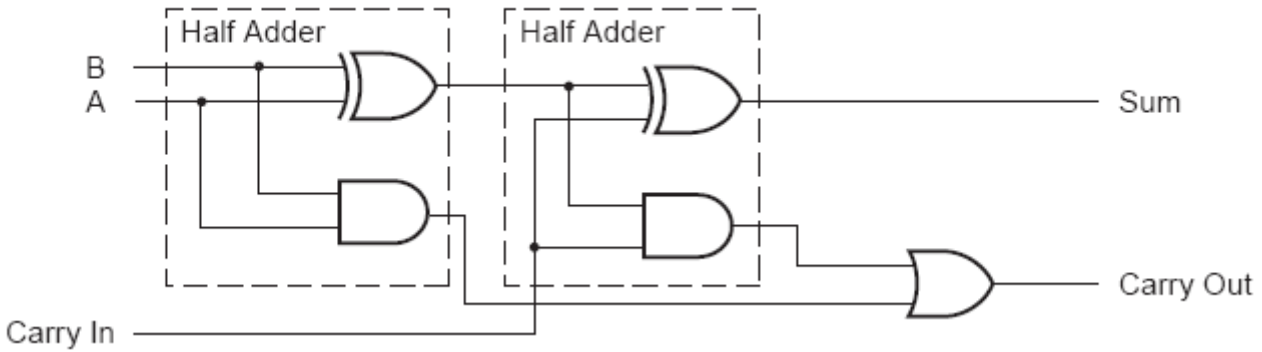
Accesarea Carry Logic

- 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

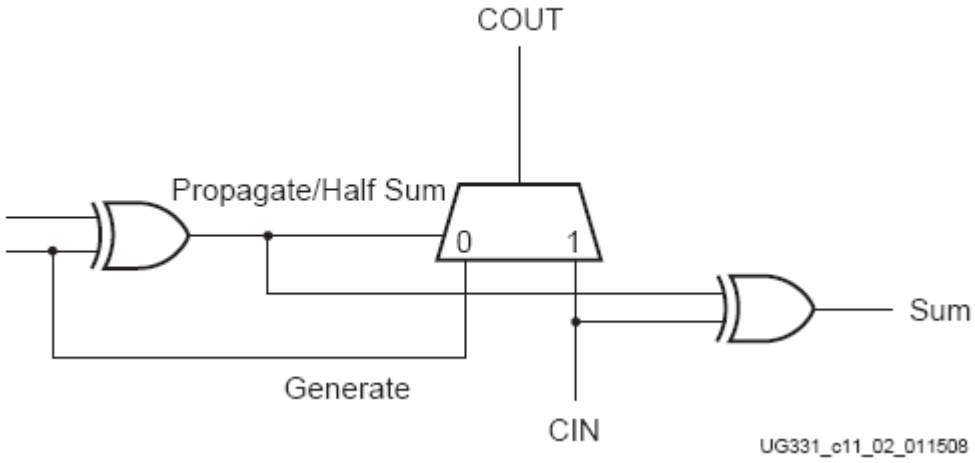
Implementarea Carry Logic

- Carry Look-Ahead

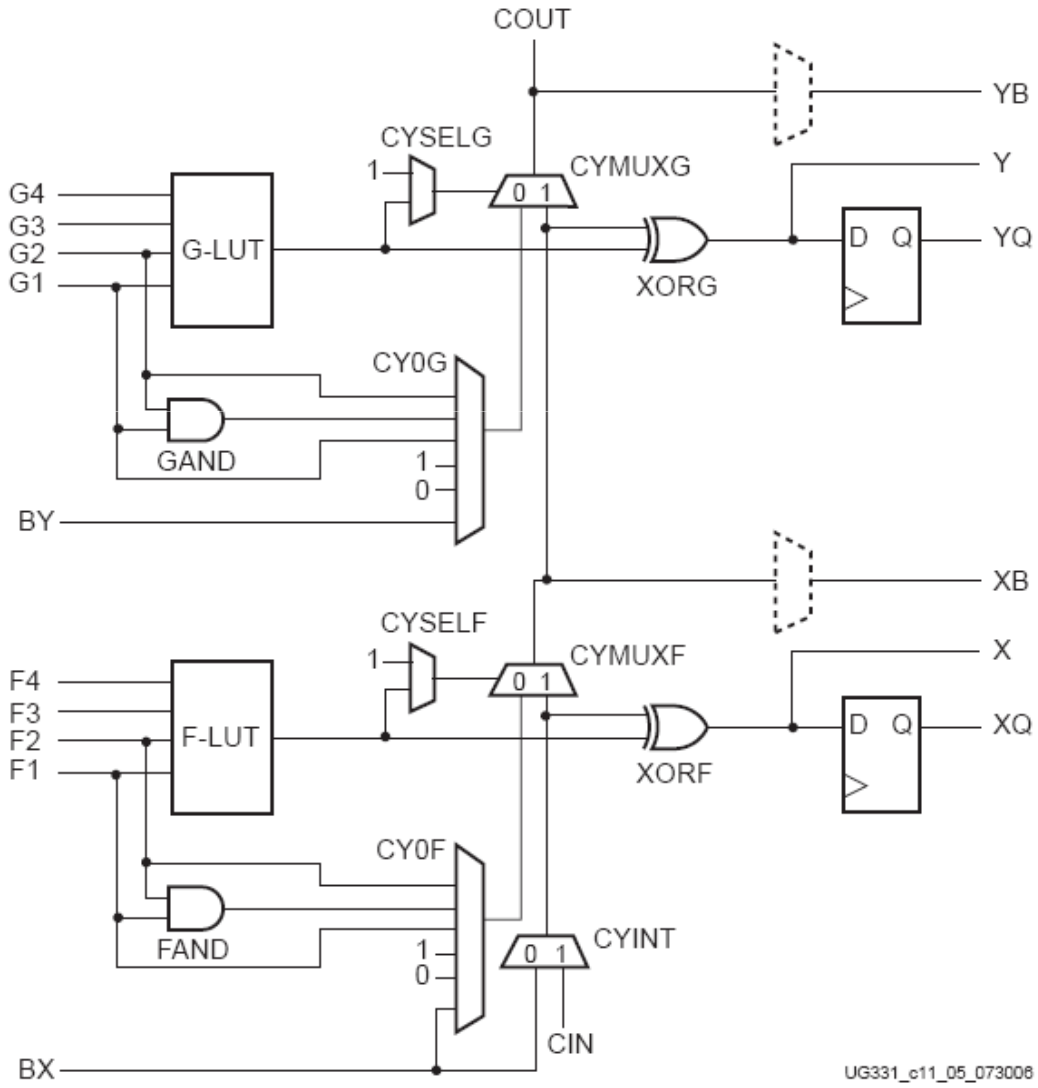
A	B	Propagate	Generate
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Avantaje:

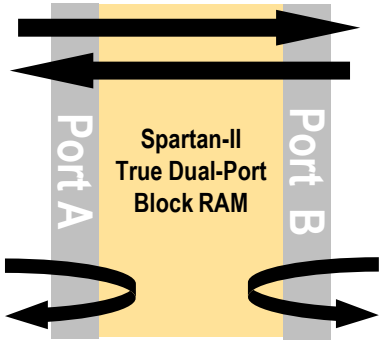
1. Foloseste un singur LUT (trei intrari)
2. Carry se propaga rapid – un singur MUX



Structura unui slice cu logica adaugata



Block RAM



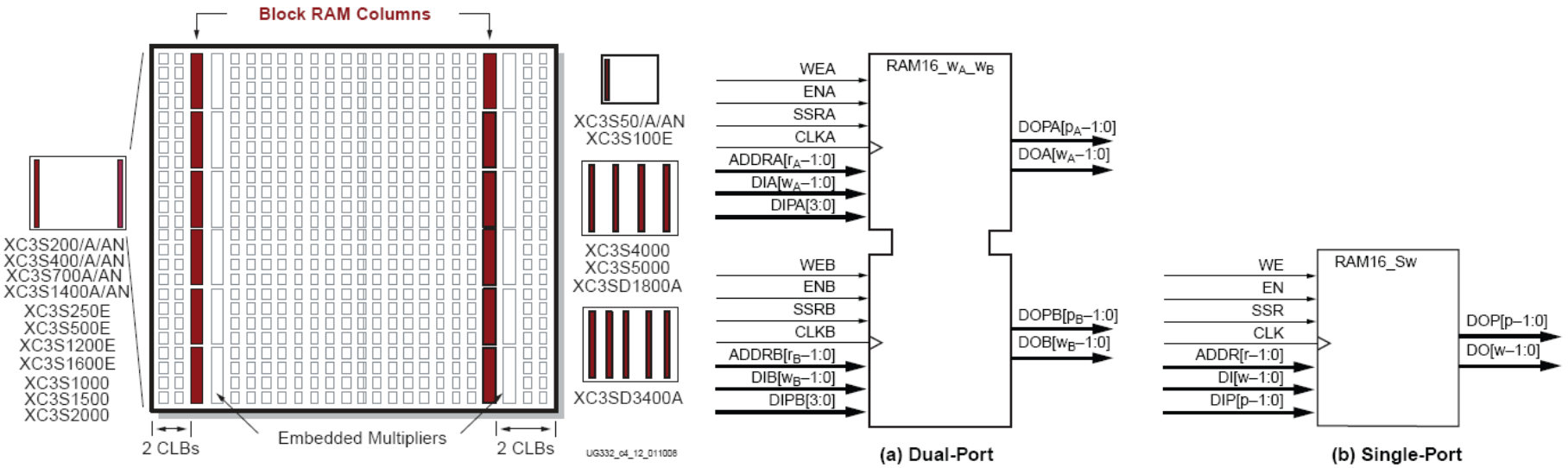
- Implementare eficienta a memoriei intre CLB si porturile IO
  - Blocuri dedicate
- Toata logica de control este implementata in celula de RAM
  - De la 4 la 104 blocuri de memorie
    - 18 kbiti pe bloc
  - Blocurile se pot cascada pentru implementarea unei memorii mai mari
- Poate functiona ca single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

Cantitatea de RAM familia Spartan 3

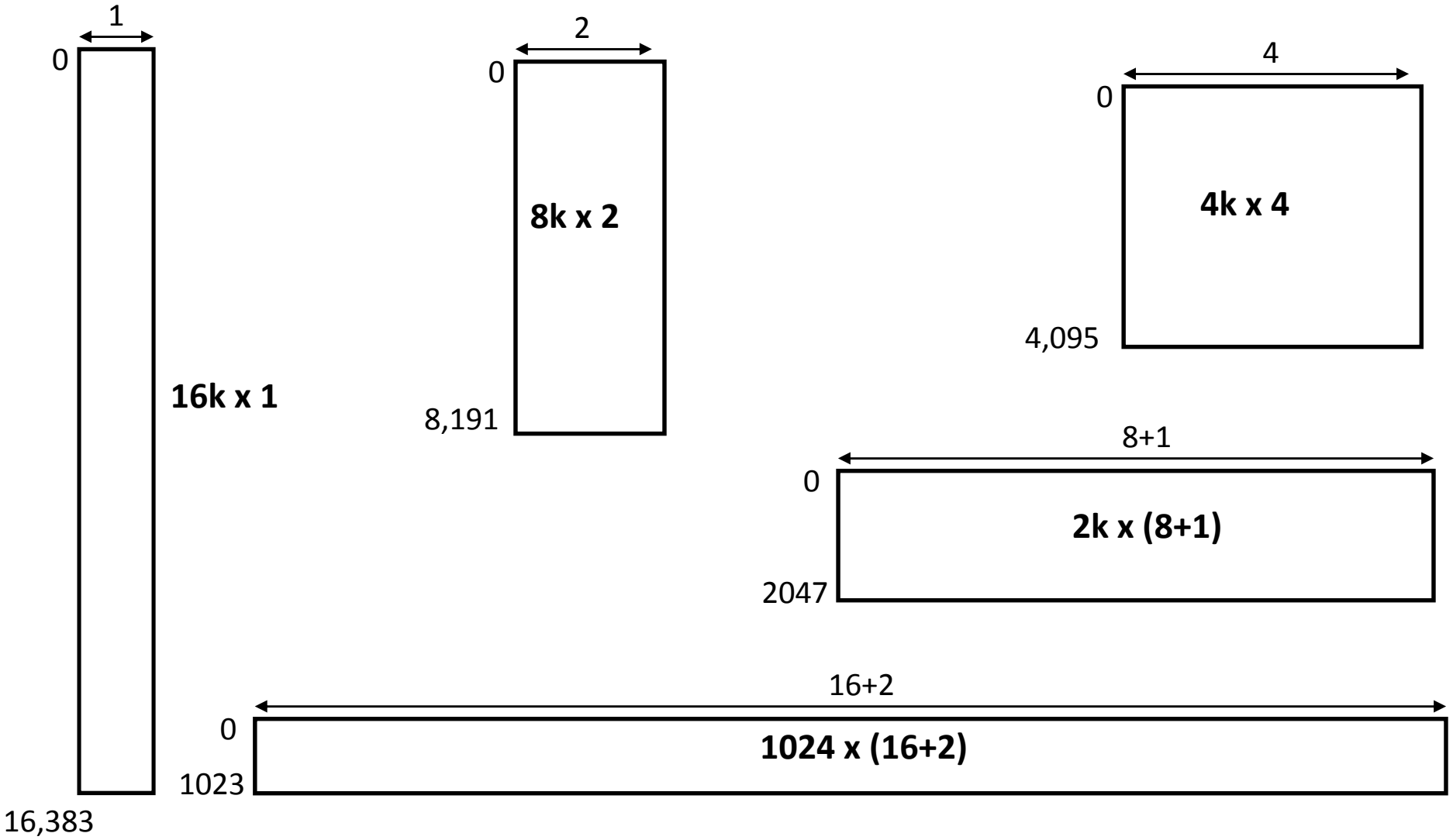
Device	Total Number of RAM Blocks	Total Addressable Locations (bits)	Number of Columns
XC3S50	4	73,728	1
XC3S200	12	221,184	2
XC3S400	16	294,912	2
XC3S1000	24	442,368	2
XC3S1500	32	589,824	2
XC3S2000	40	737,280	2
XC3S4000	96	1,769,472	4
XC3S5000	104	1,916,928	4

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)

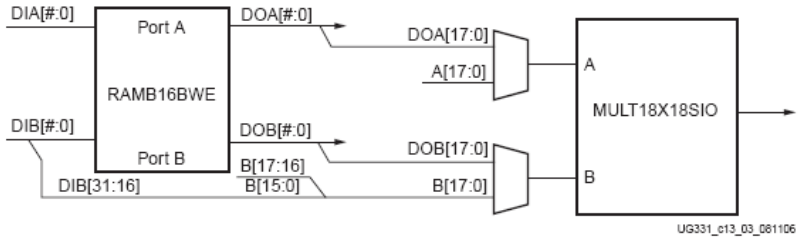


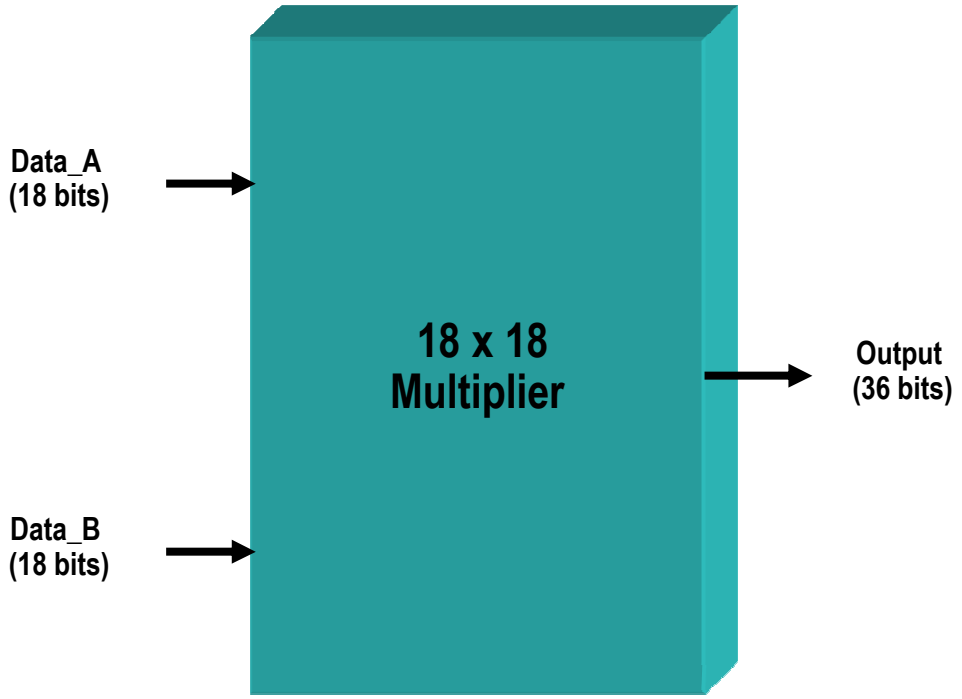
Block RAM Port Aspect Ratios



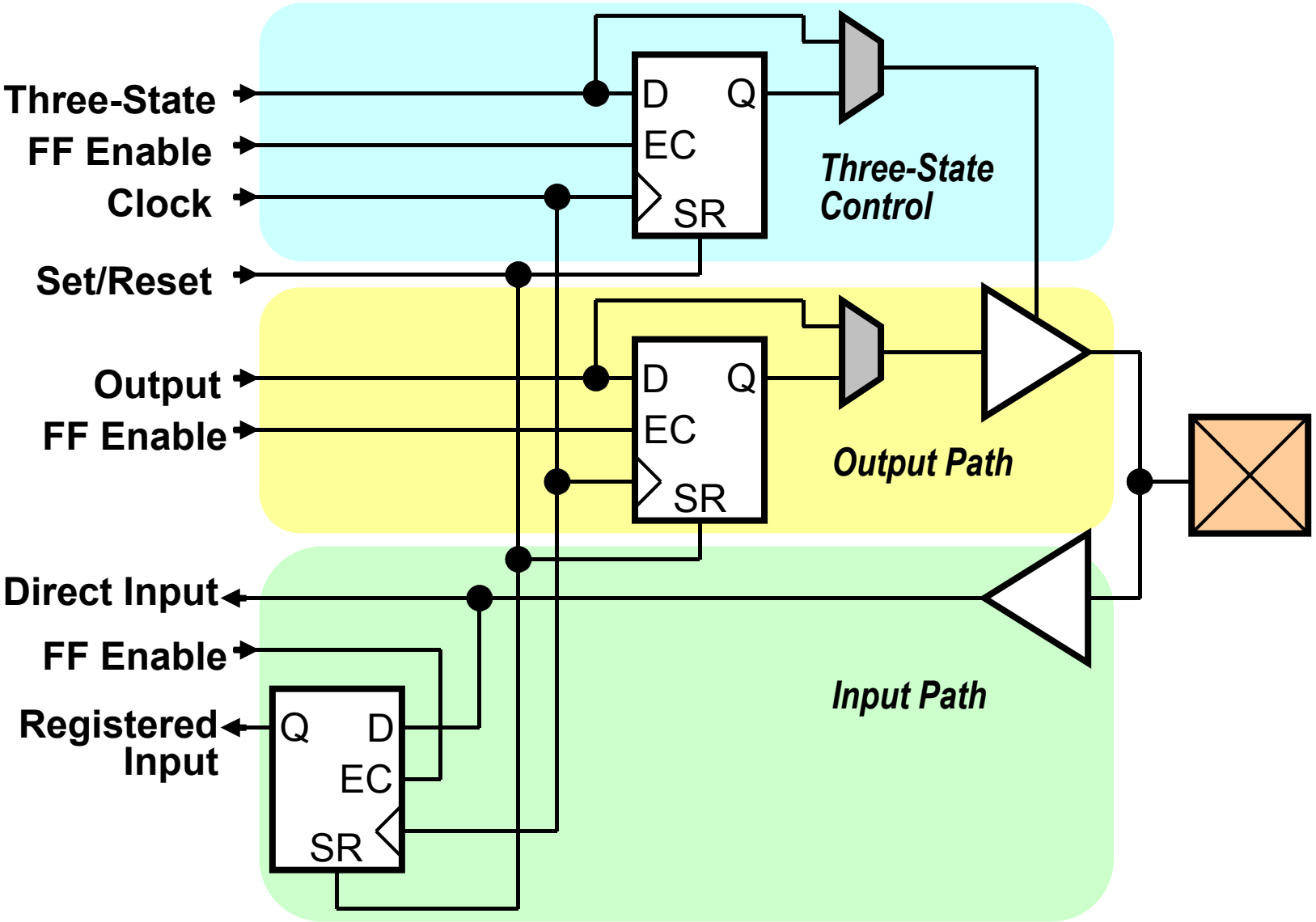
18 x 18 Embedded Multiplier

- 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





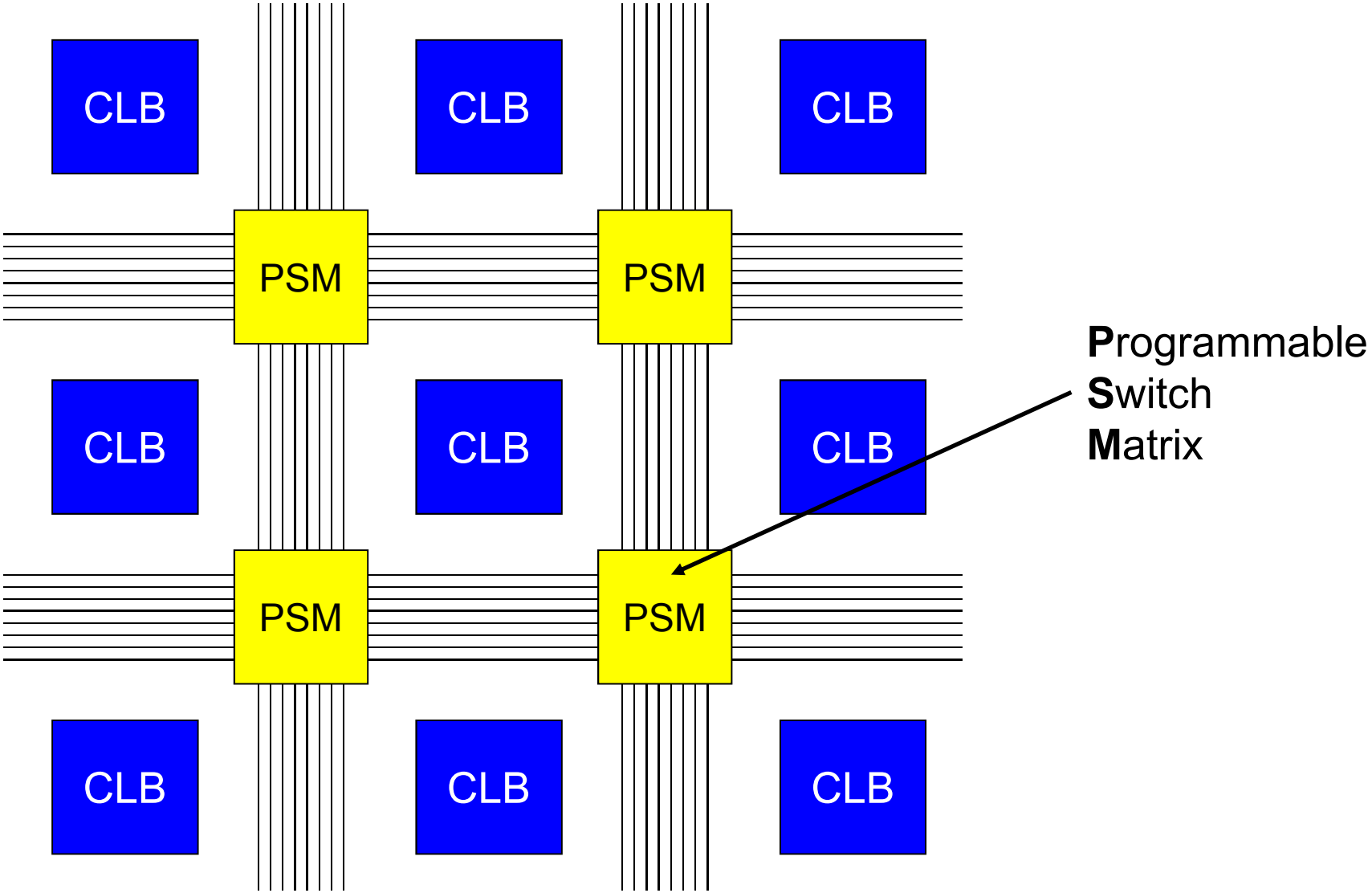
Structura de baza a unui bloc I/O



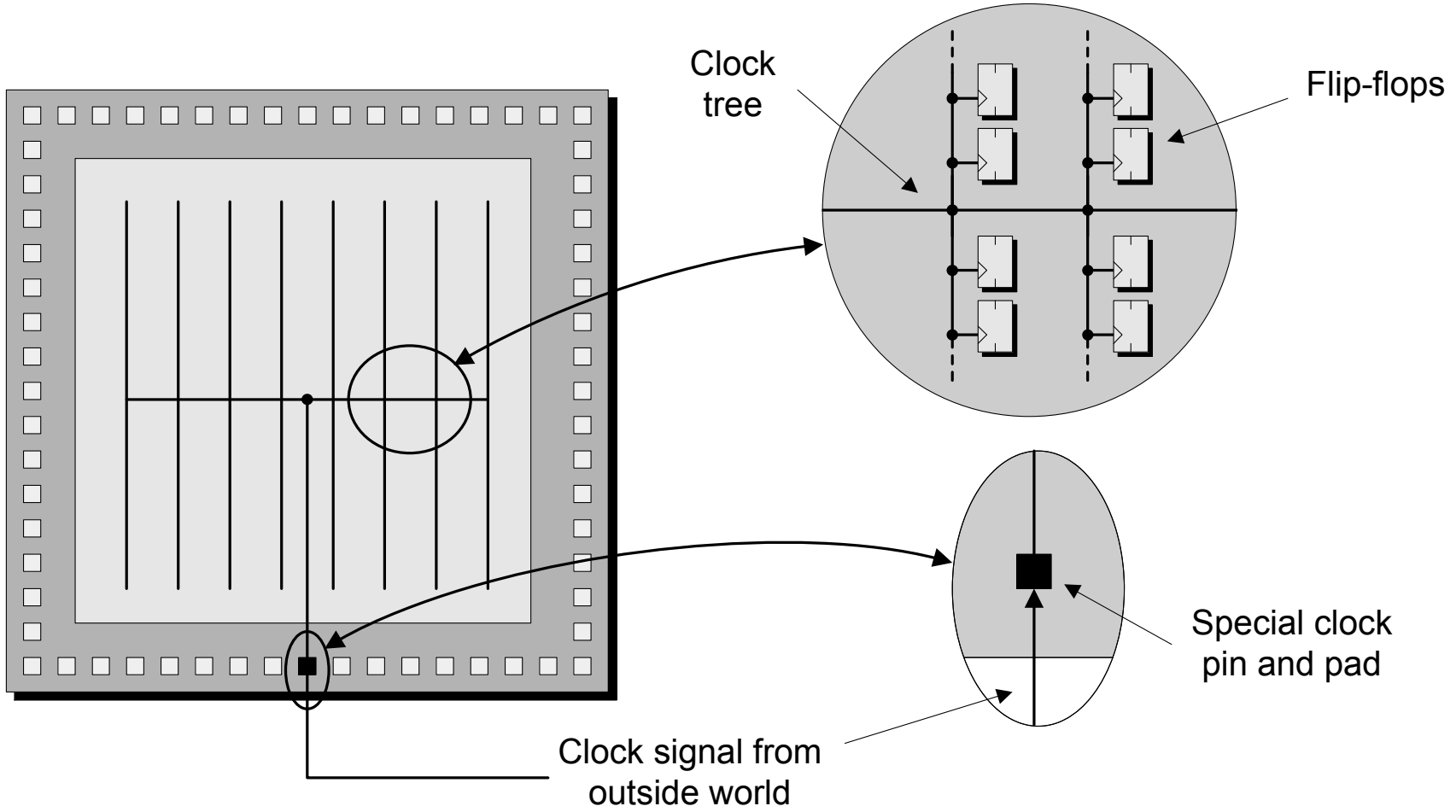
Funtionalitatea unui bloc I/O

- 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

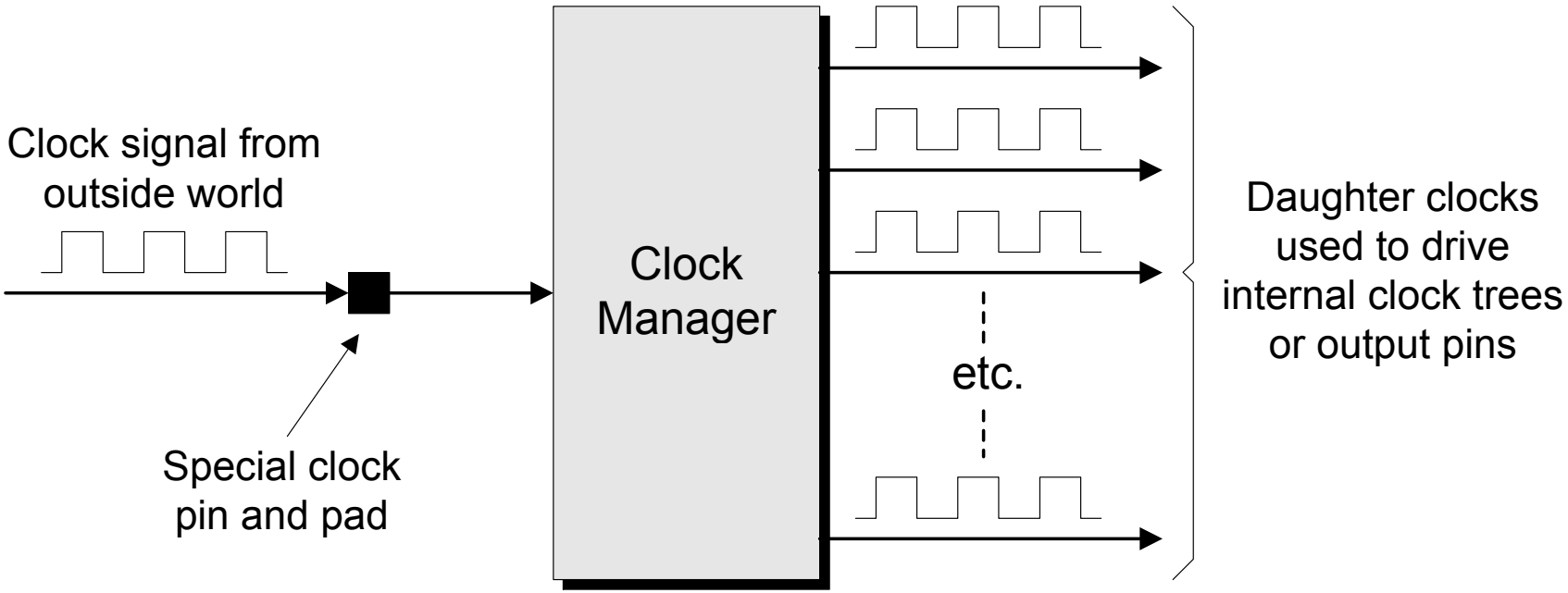


Clock Tree



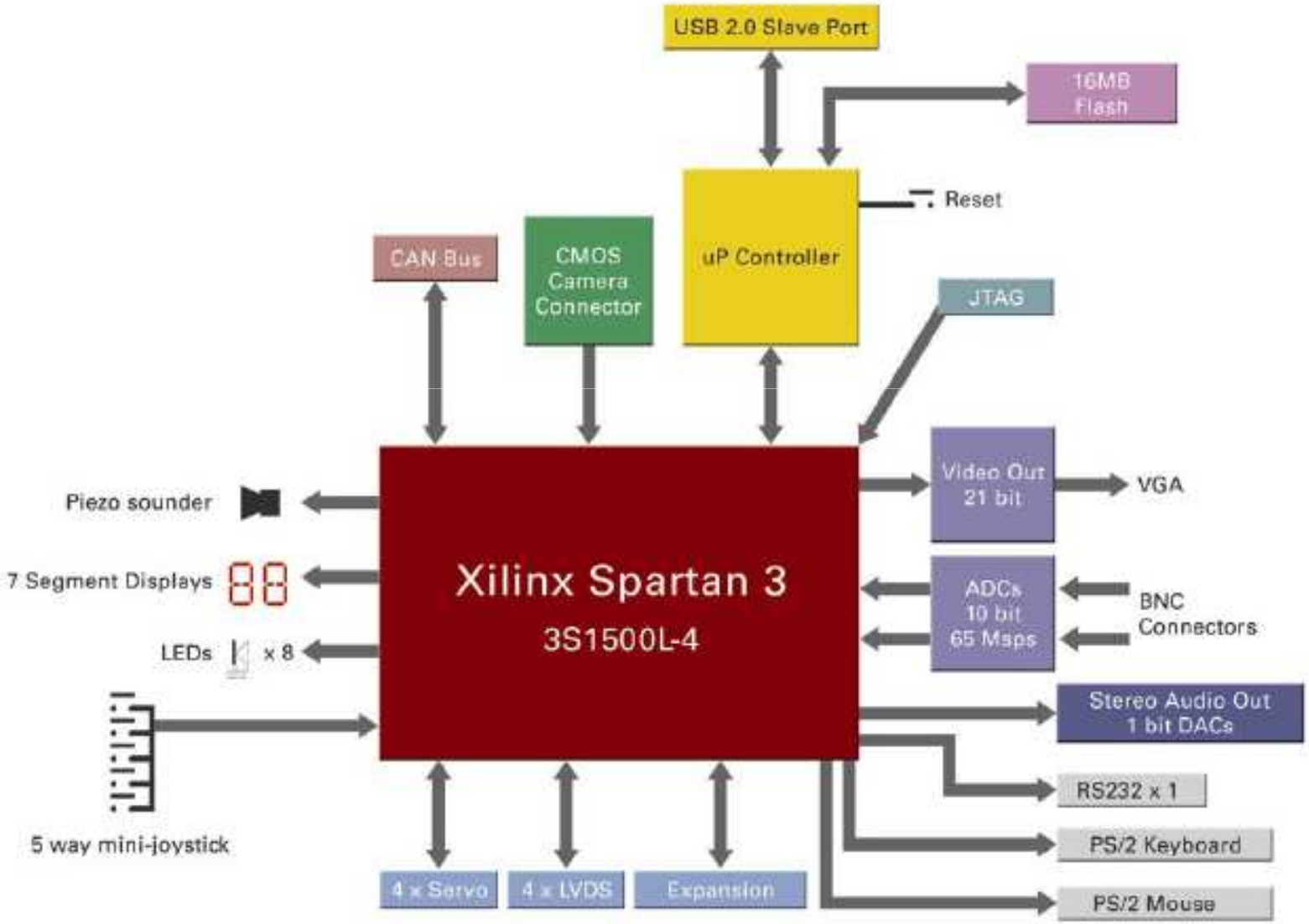
The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Digital Clock Manager (DCM)



The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Exemplu de sistem cu FPGA



FPGA Design Flow

Design flow (1)

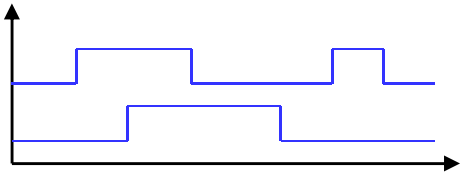
Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

Specificatii (descrierea functionalitatii)

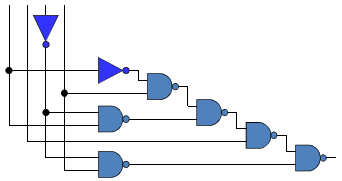
```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

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)

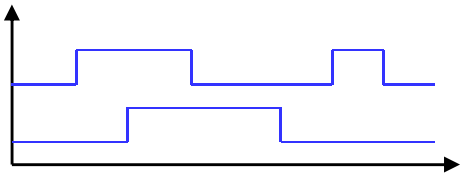
Simulare Functionala



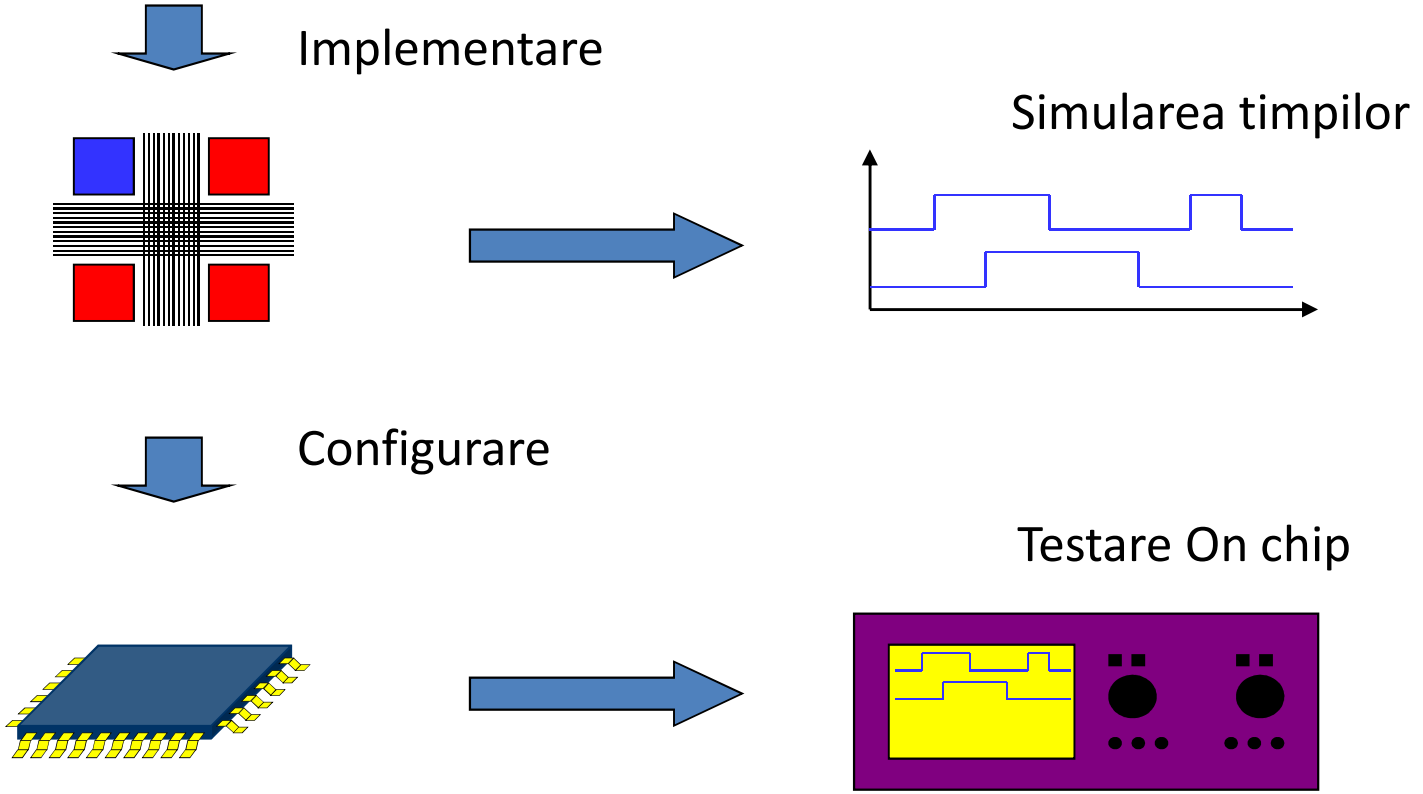
Sinteza



Simulare Post-synthesis



Design flow (2)



Sinteza

Tool-uri de Sinteza

Synplify Pro

Xilinx XST

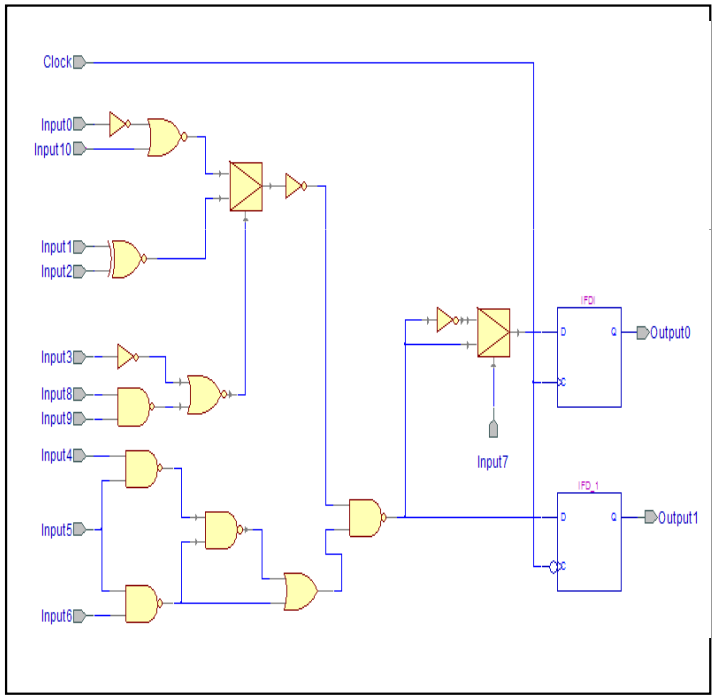
… si altele

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;
```

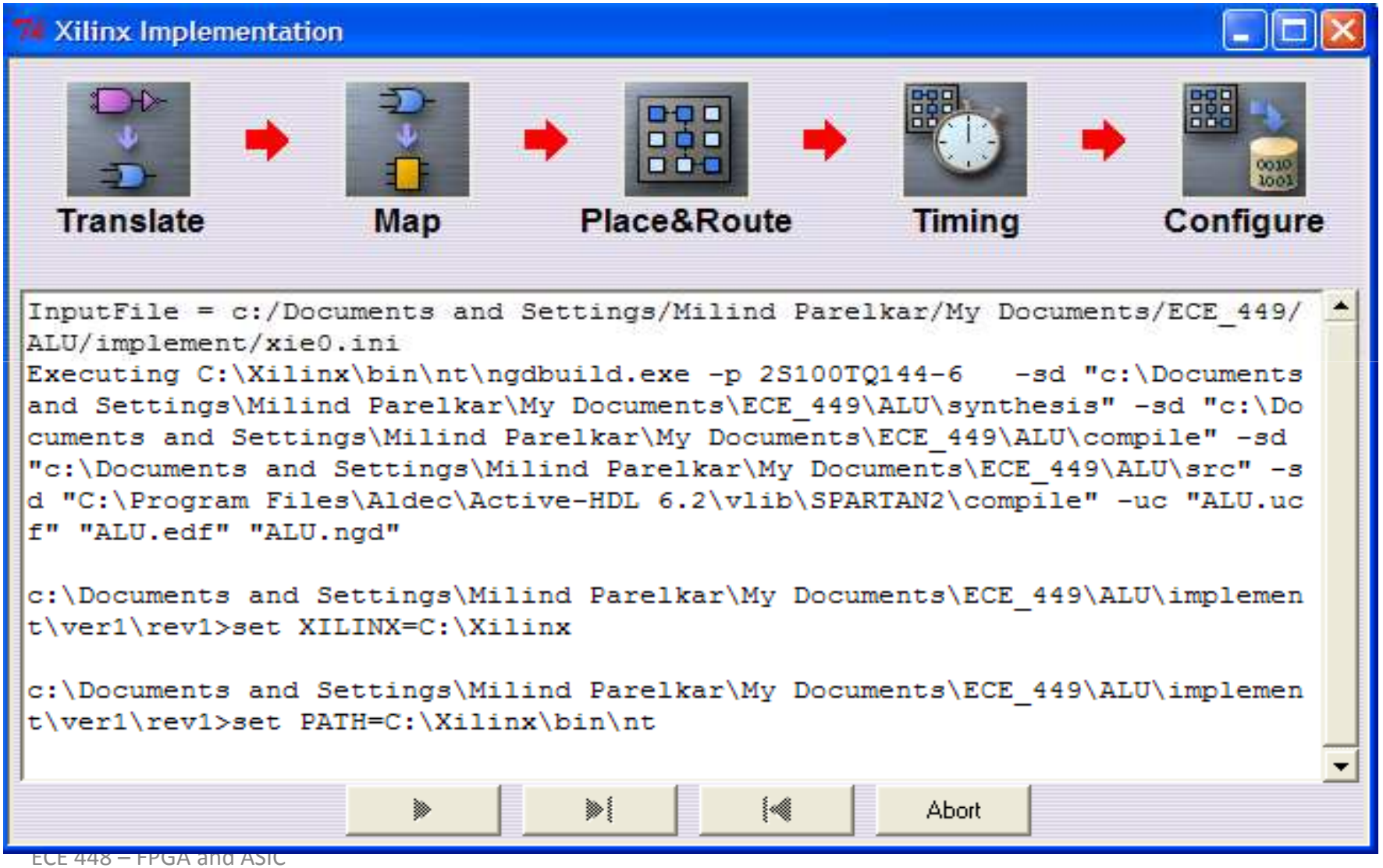
Netlist Circuit



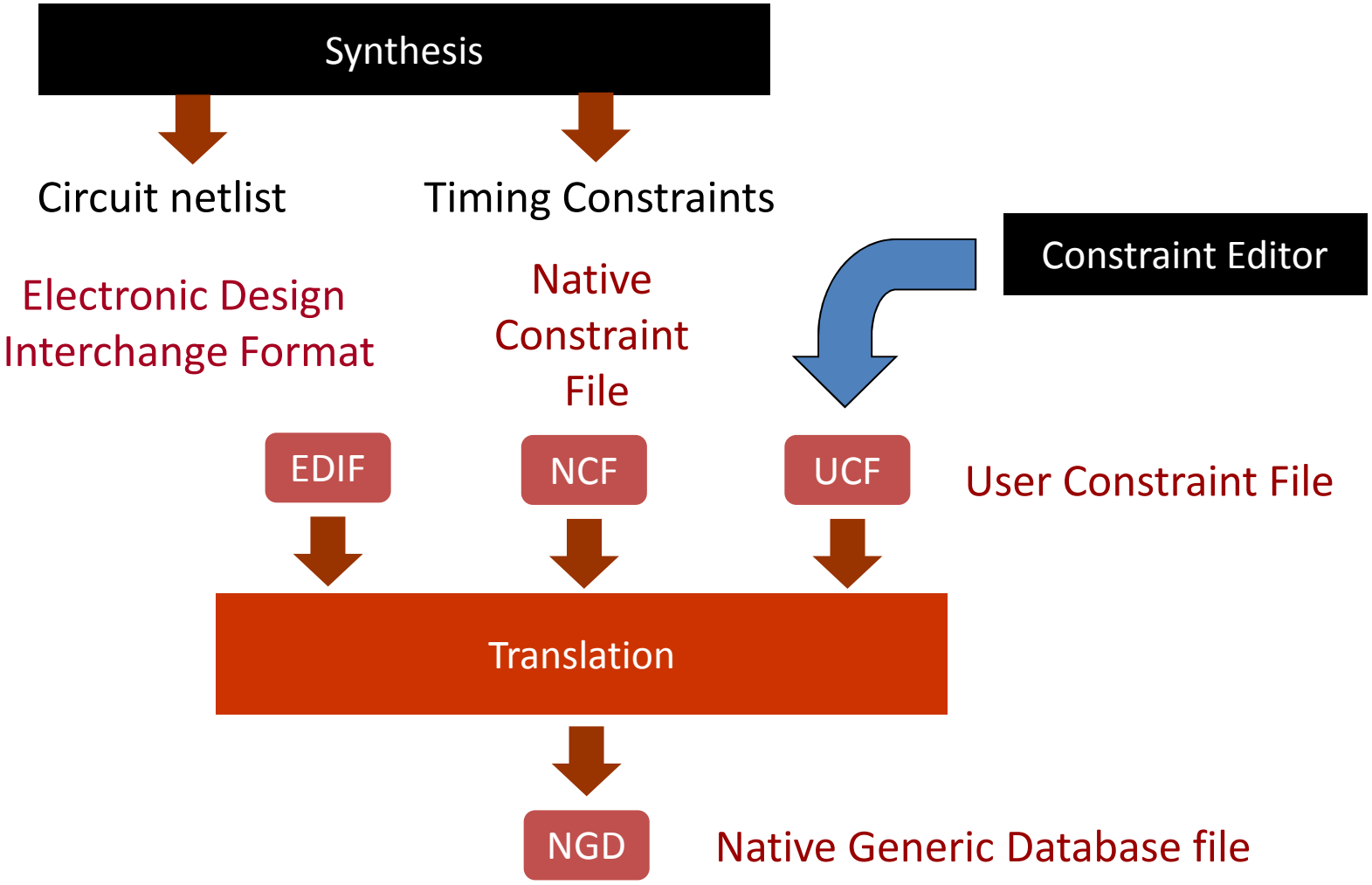
Implementare

Implementare

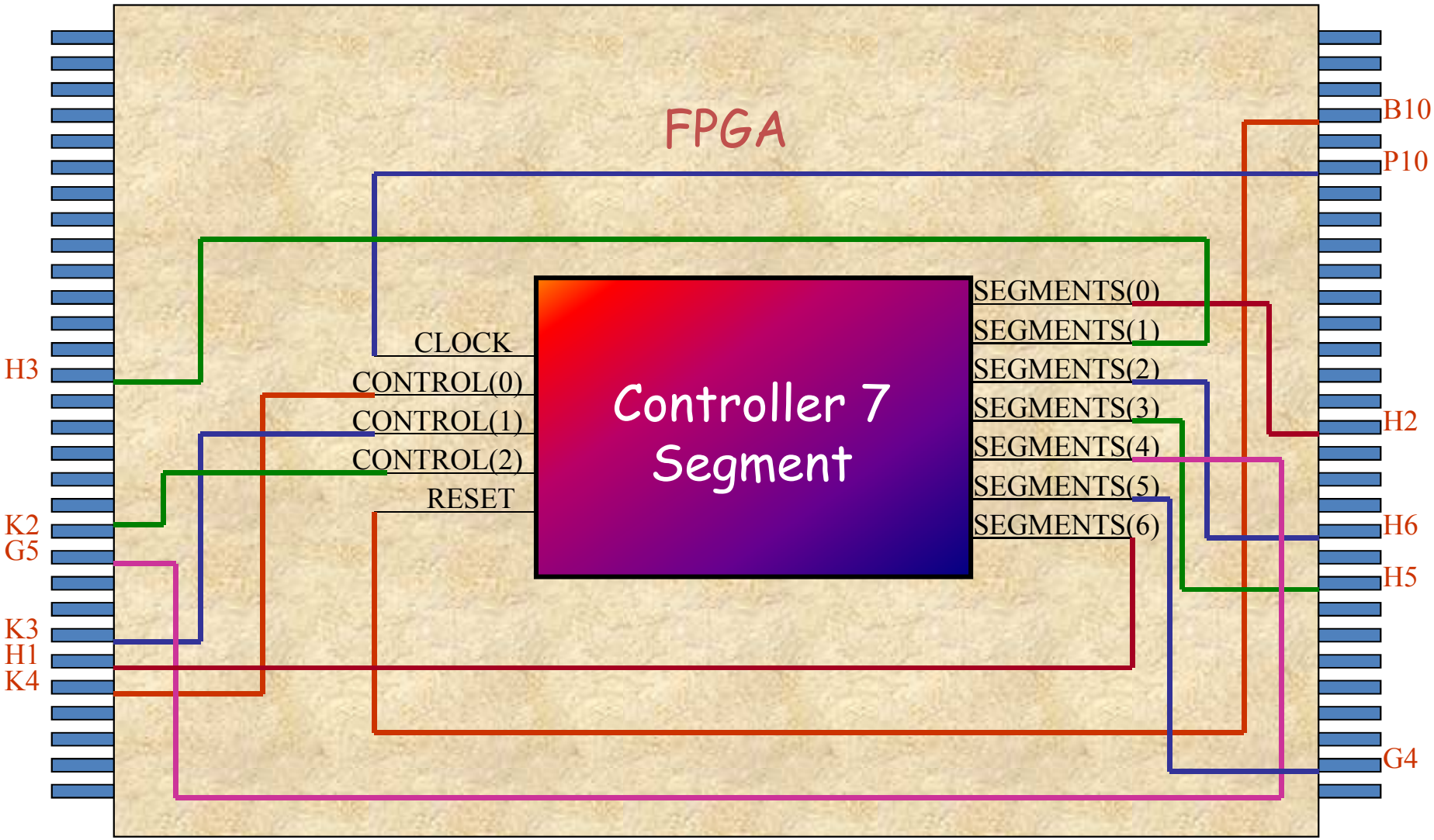
- Dupa siteza intregul proces de implementare este realizat de tool-uri FPGA proprietare


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

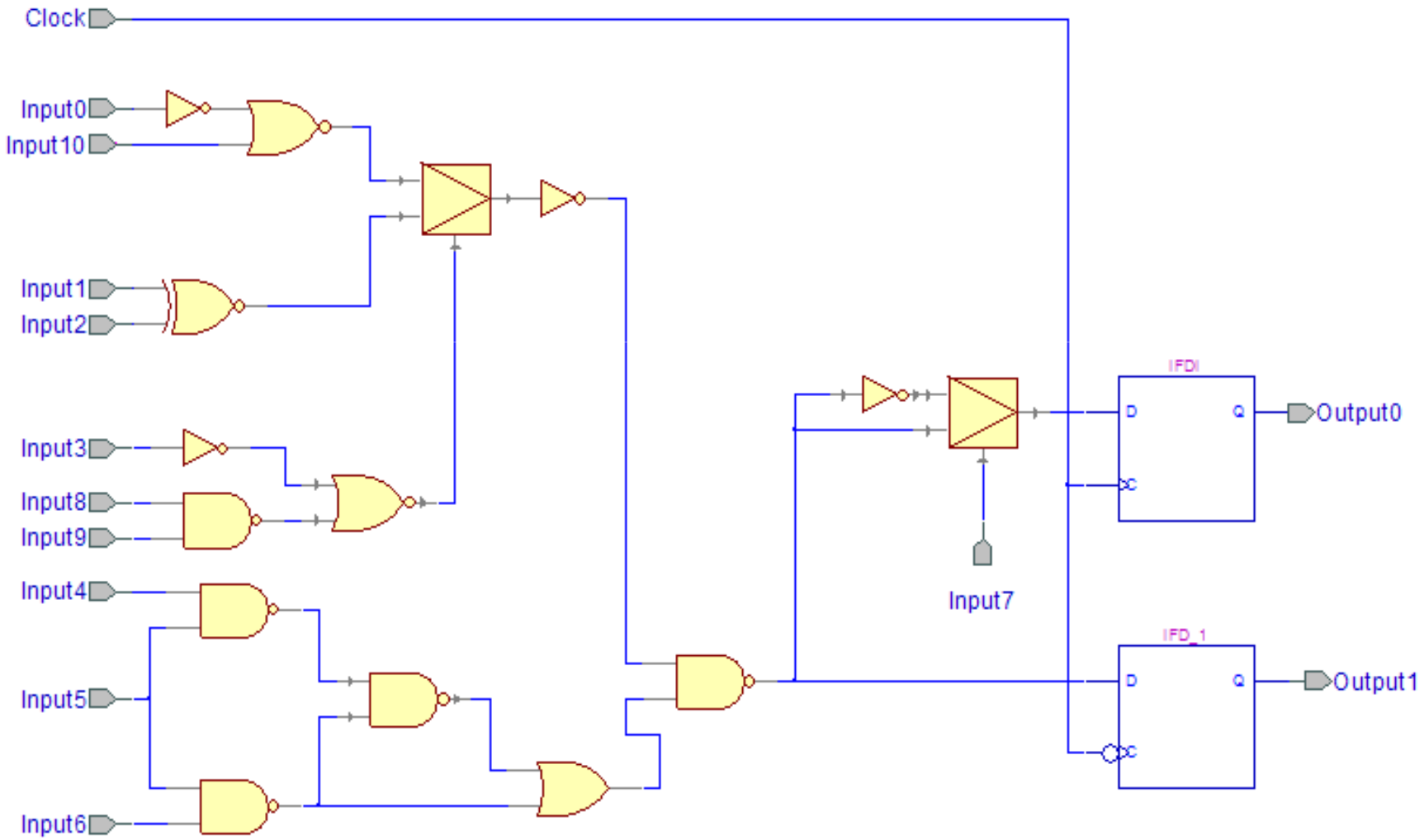
Translatie


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

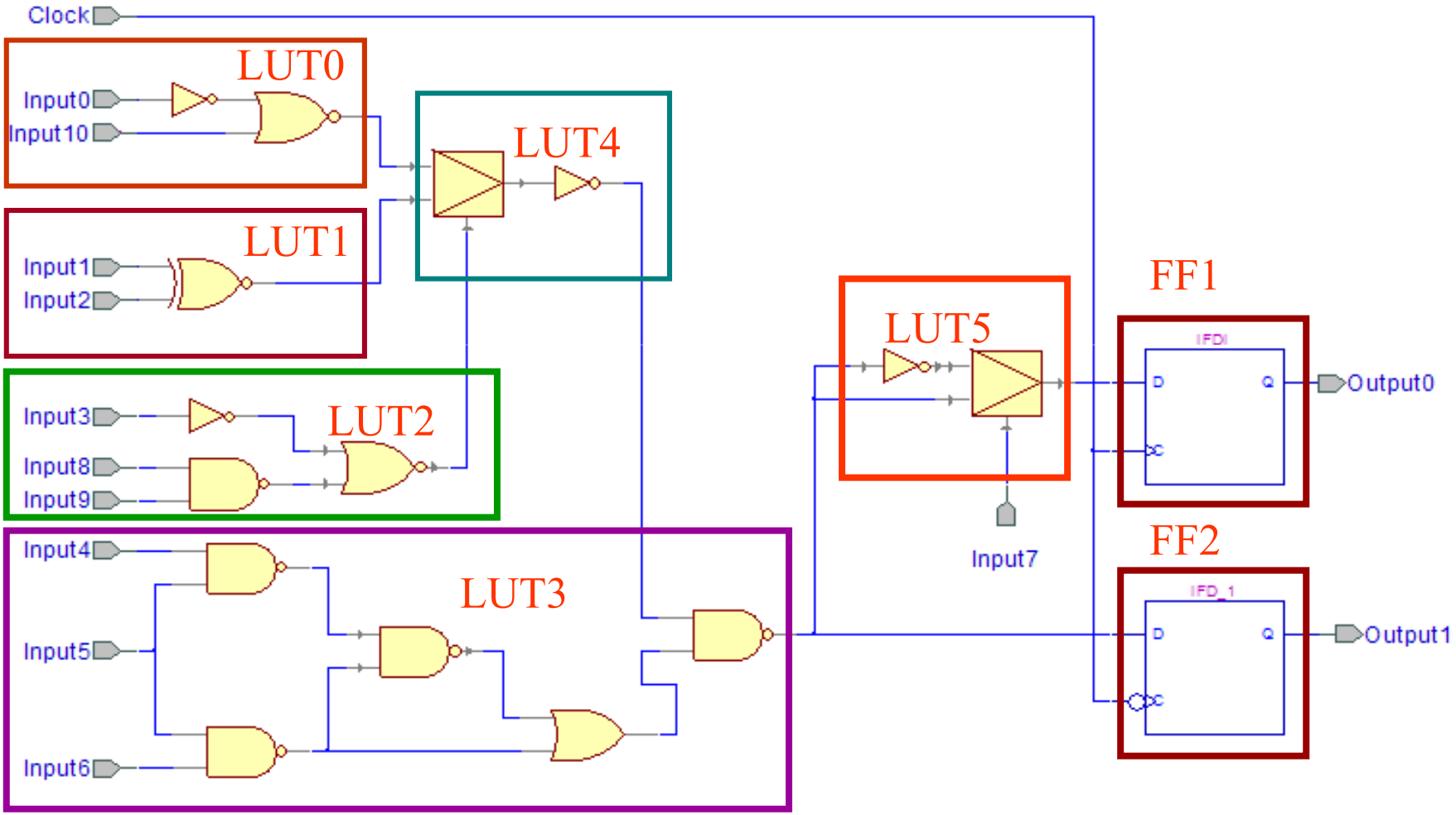
Asocierea Pinilor



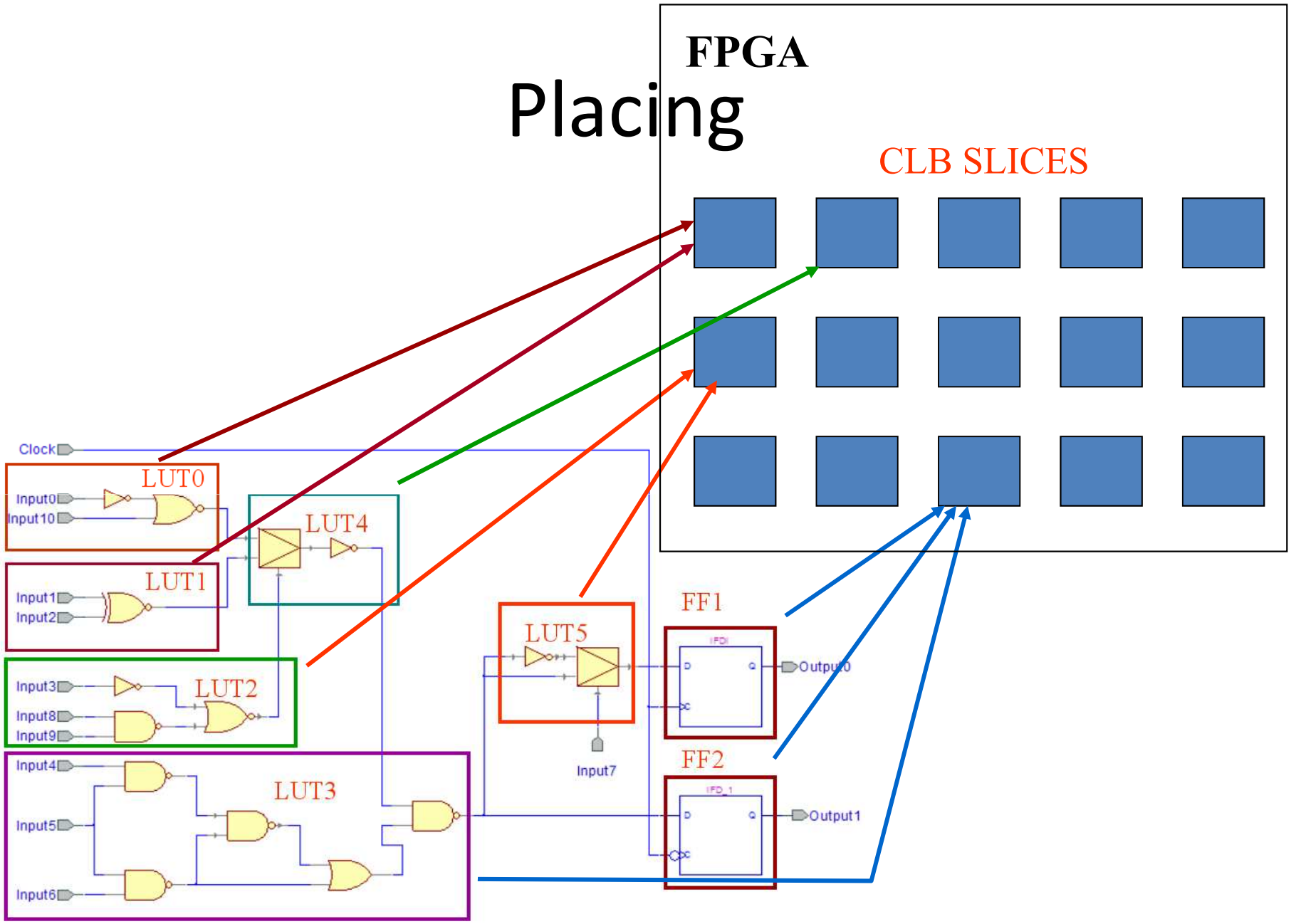
Netlist Circuit



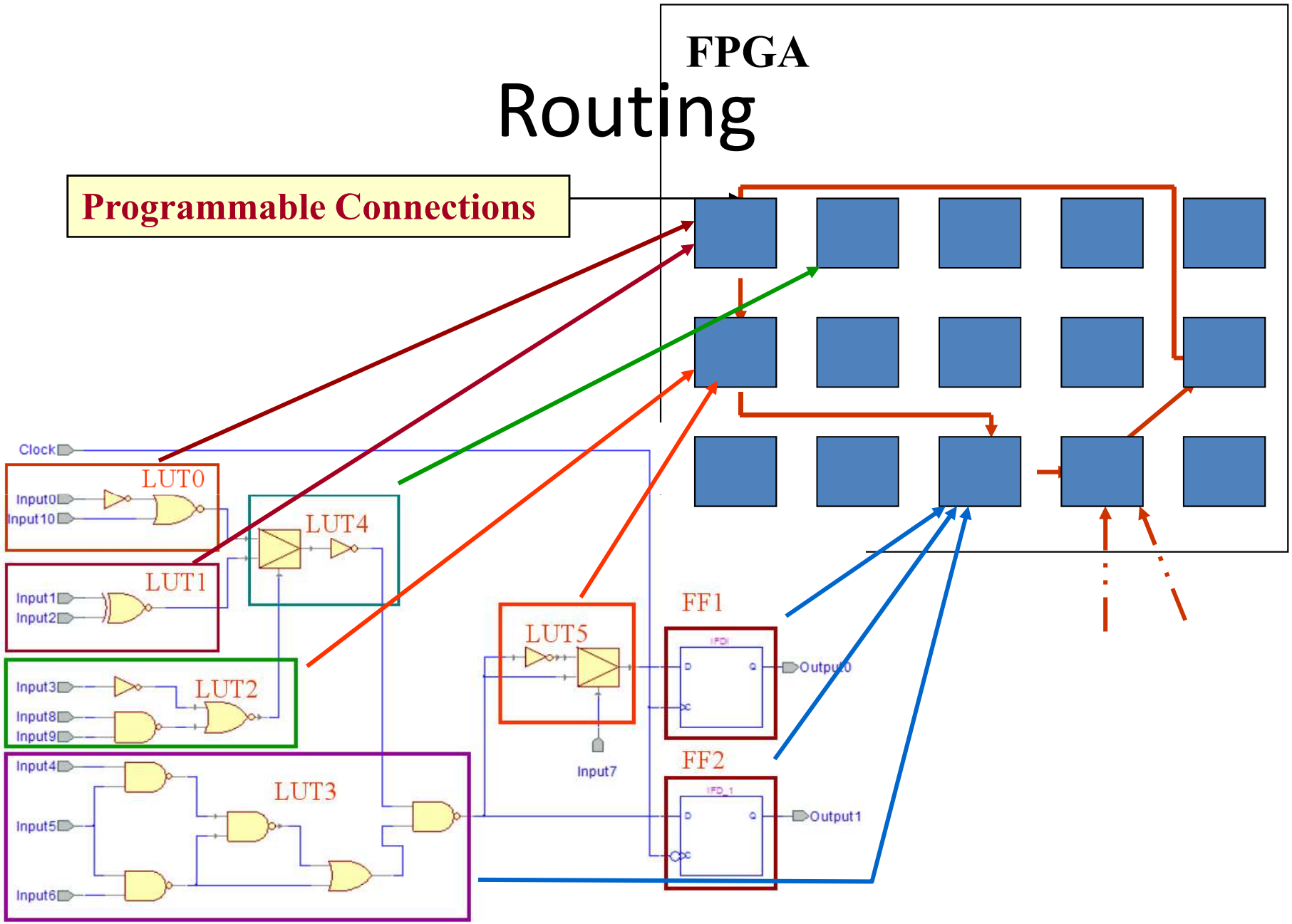
Mapare


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.