Seminar 7 - Logică digitală

Ștefan-Alexandru Jura

26 Aprilie 2024

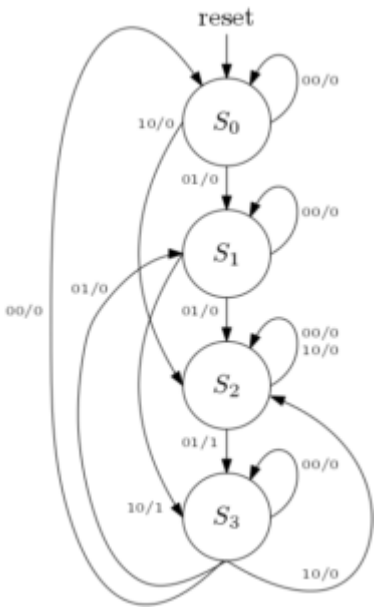
1 Recapitulare.

1.1 Circuite combinaţionale.

Sunt circuite fără memorie, fără feedback, există legătură doar de la intrări la ieşiri. Se implementează cu porţi logice (AND, OR, XOR etc).
Circuite combinaţionale complexe: Multiplexor, Codificator, Decodificator, Sumator (vezi exemplu implementare ALU + determinarea latenţei).
Funcţii logice: porţi logice, metode de minimizare (Veich-Karnaugh, Quine McKlusgey), implicanţi primi, implicanţi primi esenţiali.
Reprezentarea numerelor în virgulă fixă în sistemele de calcul: semn-mărime, complement de 1, complement de 2; standardul IEE754 pentru reprezentarea numerelor în virgulă mobilă.

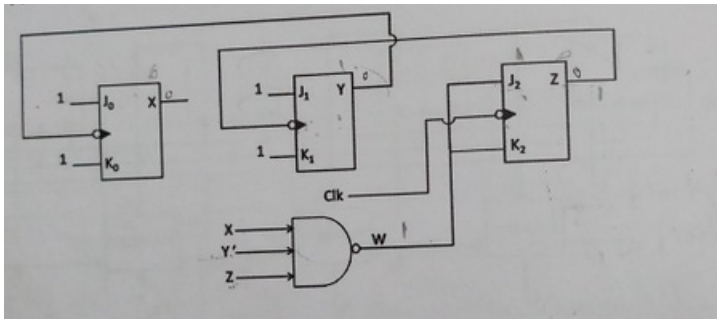
1.2 Circuite secvenţiale.

Sunt circuite cu memorie, cu feedback, există legătură de la intrări la ieşiri şi de la ieşiri la intrări.
Diferenţa dintre flip-flop şi latch: flip-flop-ul comută pe frontul crescător al semnalului de clock (ţin minte prin asocire **F**lip-flop - **F**ront), iar latch-ul pe palier.
Tipuri de FF-uri: SR, JK, D şi T. Cel mai folosit este cel de tip D - ieşirea va avea valoarea intrării.
Semnalul de reset care aduce circuitul într-o stare cunoscută - sincron (coordonat cu semnalul de clock; daca semnalul de clock este activ, atunci valoarea reset-ului contează) şi asincron (indiferent de semnalul de clock, dacă semnalul de reset este assert-uit, atunci acesta va avea efect).
Circuite de memorare - regiştrii, numărătoare.
Design-ul circuitelor secvenţiale - folosind automate cu stări finite, care sunt de 2 tipuri: Moore (ieşirea depinde doar de starea curentă) şi Mealy (ieşirea depinde atât de starea curentă, cât şi de intrare).
Exemplu întrebări de tip grilă examen.
1) Se dă automatul cu stări finite:



Întrebări:
1. Ce tip de automat este?
2. Câte intrări are?
3. Câte ieşiri are?
4. Dacă mă aflu în starea S1 şi dau intrarea 01, care va fi ieşirea şi starea următoare?

2) Se consideră următorul circuit digital. Iniţial, toate ieşirile X, Y şi Z ale flip-flop-urilor au valoarea 0 logic. Care vor fi valorile lor după aplicarea a 5 ciclii de tact?



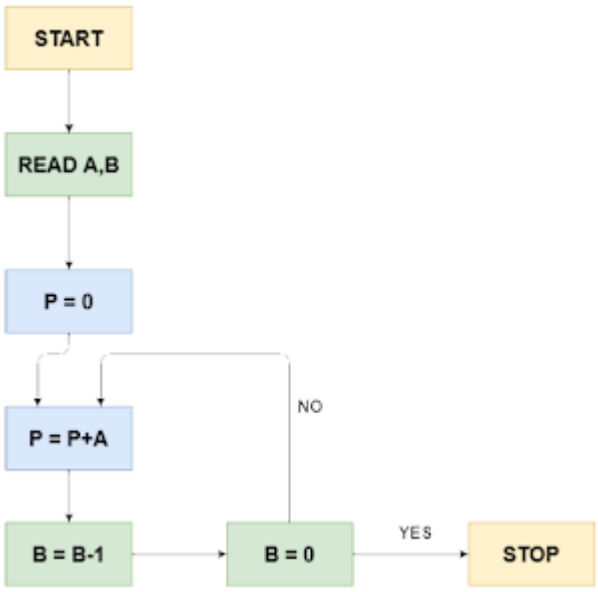
2 Model de problemă.

2.1 Privire de ansamblu. Descrierea algoritmului propus.

Să se implementeze un înmulţitor pe 16 biţi care înmulţeşte 2 numere în complement de 2, în manieră secvenţială. Se va implementa atât calea de date, cât şi calea de control.

Soluţie:

Algoritmul propus se bazează pe înmulţire prin adunare repetată. La fiecare ciclu de clock, se realizează o adunare a primului operand (combinaţional, folosind un sumator, precum RCA - Ripple Carry Adder) şi a produsul parţial stocat în acumulator. De exemplu, dacă avem $4 \cdot 5 = 20$, înseamnă că avem $4+4+4+4+4$, adică $8+4+4+4$, sau $12+4+4$. La fiecare semnal de clock, stocăm în acumulator suma dintre primul operand şi suma anterioară, stocată în acumulator, iar noua sumă este stocată în acumulator (de aici caracterul secvenţial si necesitatea semnalului de clock - pentru încărcarea registrului). Operaţia are loc până când conţinutul numărătorului în care este încărcat B devine 0, adunându-l, astfel, pe A de B ori, adică chiar operaţia $A \cdot B$. Schema logică (sau ordinograma) aferentă acestui algoritm este:



Având în vedere că operanzii sunt în complement de 2, strategia este următoarea:
• Se citesc operanzii dintr-un bus de date, câte un operand per ciclu de clock, deoarece aceştia vor fi stocaţi în câte un registru. Sumatorul de tip RCA realizează adunarea numerelor fără semn, deci, prin intermediul unei logici combinaţionale, se aduc operanzii la valoarea absolută (se transformă din complement de 2 în semn mărime, cu bitul de semn 0, adică număr pozitiv). Semnul rezultatului final (pozitiv sau negativ) va fi dat de operaţia EXOR între biţii de semn ai celor 2 operanzi. Reamintim tabelul de adevăr pentru funcţia EXOR:

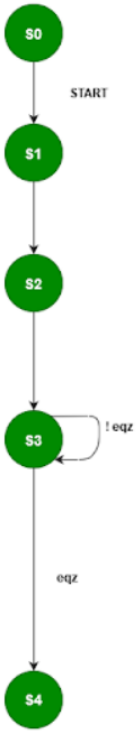
Input		output
A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

şi se observă că EXOR între biţii de semn ne dă semnul rezultatului final. De exemplu, dacă avem 2 numere pozitive (bitul de semn al amândurora este 0), rezultatul este tot pozitiv (bitul de semn final este 0), daca unul este pozitiv şi altul negativ, rezultatul este negativ (bitul de semn este 1).
• După încărcarea valorilor absolute în regiştrii (A în registrul A, B în numărător), se urmează paşii descrişi anterior şi, la final, în funcţie de rezultatul EXOR-ului, se stabileşte semnul rezultatului, care poate fi fie negativ (deci se aduce în complement de 2), fie pozitiv (rămâne în semn mărime, fiind număr pozitiv).

2.2 Sinteza dispozitivului de înmulţirea. Calea de date şi calea de control.

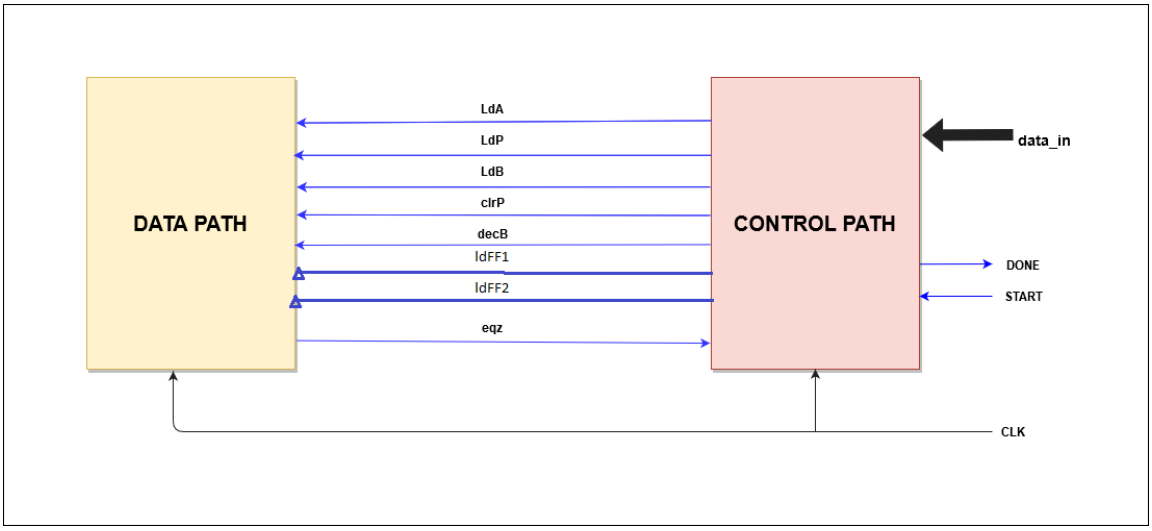
Pentru implementarea hardware a algoritmului, trebuie să ţinem cont de 2 elemente: calea de date şi calea de control.
Calea de date conţine toate elementele necesare stocării valorilor şi realizarea operaţiilor asupra acestora: regiştrii, numărătoare, flip-flop-uri, elemente de logică combinaţională etc. Elementele folosite pentru sinteza înmulţitorului sunt: un registru pentru stocarea deînmulţitului A, un counter pentru stocarea înmulţitorului B, cu direcţia de numărare în jos, un acumulator care păstrează produsele parţiale, un registru care stochează rezultatul final, 2

flip-flop-uri de tip D pentru stocarea biţilor de semn ai celor 2 operanzi şi logica combinaţională aferentă acestor elemente.
Calea de control ajută la controlul semnalelor pentru elementele secvenţiale din calea de date. De exemplu, semnalul de load pentru un numărător nu se activează mereu, ci doar la momente bine definite de timp, în funcţie de impulsurile de clock şi de alte semnale. Calea de control se sintetizează folosind un *automat cu stări finite.*
Pentru înmulţitorul propus, calea de control este implementată folosind un automat cu 4 stări:

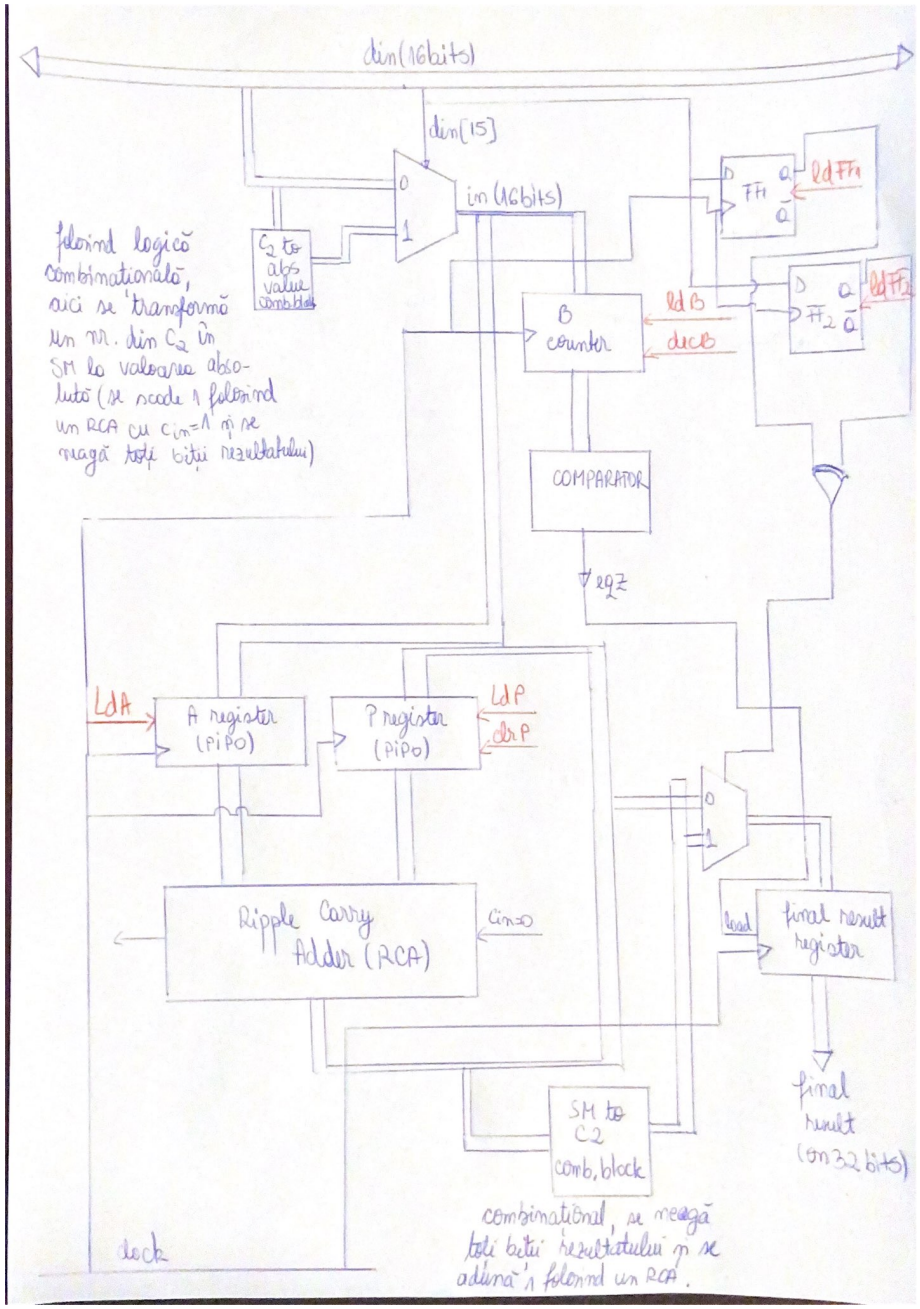


Calea de control are 8 semnale:
• ldA - comandă încărcarea registrului A;
• ldB - comandă încărcarea numărătorului;
• ldP - comandă încărcarea acumulatorului;
• clrP - încarcă acumulatorul cu valoarea 0 (pe 16 biţi);
• done - semnifică finalizarea înmulţirii;
• decB - semnal pentru decrementarea conţinutului numărătorului B;

• ldFF1 şi ldFF2 - semnale pentru încărcarea flip-flop-urilor cu biţii de semn ai celor 2 operanzi.
Iniţial, în starea S0, toate semnalele sunt setate pe 0. (presupunem că toate se activează pe 1 logic). Apoi, în starea S1, se activează semnalul ldA şi semnalul ldFF1, adică se realizează încărcarea în registrul A a operandului A şi în flip-flop-ul D a bitului de semn al acestui operand. Apoi, în starea S2, se activează semnalele ldB ¸ ldFF2 şi clrP, care încărcă conţinutul numărătorului, încarcă în flip-flop bitul de semn şi setează valoarea acumulatorului la 0. În starea S3, se activează ldP, care comandă încărcarea produsului parţial în acumulator şi semnalul decB, care arată că s-a realizat o operaţie de adunare din cele B totale. În funcţie de valoarea eqz (care este 1 dacă s-a ajuns cu valoarea counter-ului B la 0, adică s-a finalizat înmulţirea, altfel este 0), se rămâne în starea curentă (se continuă realizarea adunărilor), sau se trece în starea S4, unde se activează semnalul de done, ce semnifică finalizarea operaţiei de înmulţire. Ca şi model black-box, relaţia dintre cele 2 unităţi arată astfel:



Pentru implementarea fiecărei dintre cele 2 unităţi, se folosesc metodele cunoscute. Pentru unitatea de control, se implementează FSM-ul prin realizarea codificării stărilor (numerică, one-hot sau one-cold), tabelul de tranziţii, tabelul ieşirilor şi tabelul pentru intrările flip-flop-urilor. Pentru calea de date, design-ul este conform schemei de mai jos:



folosind logică combinațională, aici se transformă un nr. din C_2 în SM la valoarea absolută (se scade 1 folosind un RCA cu $C_{in}=1$ și se neagă toți biții rezultatului)

combinațional, se neagă toți biții rezultatului și se adună 1 folosind un RCA.

Codul sursă şi testbench-ul pentru design-ul acestui înmulţitor este disponibil la link-ul acesta.

Bibliografie

[1] David A. Patterson, John L. Hennessy (2014) *Computer Organisation and Design: The Hardware/Software Interface, Fifth Edition*, Elsevier.

[2] Mircea Vlăduţiu (2012) *Computer Arithmetic: Algorithms and Hardware Implementations*, Springer.

[3] Oana Boncalo (2024) *Curs Logică Digitală.*