Laboratorul 4 – Operatii aritmetice in virgula fixa

Obiective

Acest laborator isi propune sa prezinte modul de implementare a operatiilor aritmetice in virgula fixa si a unor algoritmi pentru accelerarea executiei acestor operatii.

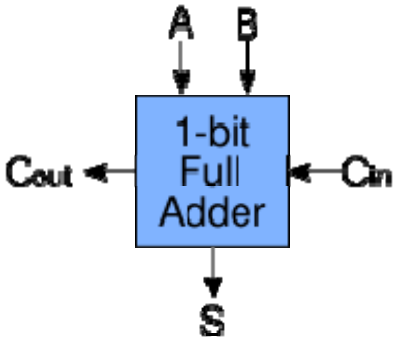
Proiectarea

De obicei, in sistemele numerice, numerele intregi sunt reprezentate binar in complement fata de 2 (numarul de biti variaza in functie cu precizia dorita).

Reprezentarea numerelor in complement fata de 2 prezinta avantajul de a putea decide daca un numar este pozitiv sau negativ analizand bitul cel mai semnificativ. Deasemenea, in cazul in care se doreste schimbarea semnului unui numar este suficient sa se adune o unitate la valoarea negata. Pe baza acestei propietati diferenta dintre doua numere in complement fata de 2 poate fi privita ca suma dintre primul si negatul celui de al doilea (suma cu transport initial „1”).

Sumatorul complet pe un bit

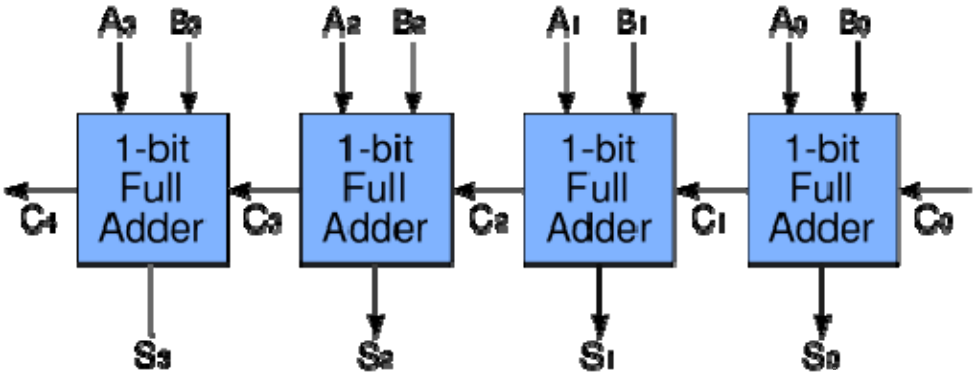
Un sumator complet pentru un singur bit este un circuit combinational cu 3 intrari si 2 iesiri:



Functiile logice ale iesirilor in functie de intrari sunt determinate pe baza tabelului urmator:

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
Cin	0	1	0	1	0	1	0	1
S	0	1	1	0	1	0	0	1
Cout	0	0	0	1	0	1	1	1

Mai multe astfel de sumatoare elementare pot fi inlantuite pentru a realiza o prelucrare a unor numere pe mai multi biti:



O astfel de arhitectura, desi are o structura extrem de simpla se poate dovedi ineficienta pentru insumarea unor numere mare (reprezentate pe multi biti) deoarece pentru calcularea rangului 2 este nevoie sa se astepte calcularea transportului generat de rangul 1 (si asa mai departe). Intarzierea se poate calcula destul de usor deoarece pentru obtinerea Cout pornind de la Cin sunt necesare doua niveluri de porti logice. Astfel pentru un sumator pe 32 de biti rezultatul este obtinut dupa 31x2+1=63 de niveluri de porti logice (intarzierea sumatorului este de 63 de ori mai mare decat intarzierea datorata unei porti logice elementare).

Sumatorul cu transport anticipat

Pentru a realiza o prelucrare mai rapida se utilizeaza arhitecturi care utilizeaza mai multe resurse in paralel pentru a reduce numarul de niveluri logice (scade numarul de niveluri dar pe fiecare nivel sunt mai multe porti logice).

Sumatorul cu transport anticipat se bazeaza pe generarea a doua semnale intermediare pentru fiecare pereche de biti de intrare. Semnalele sunt P si G (propagare si generare). P este „1” daca nivelul curent propaga transportul de la nivelul anterior, iar G este „1” daca nivelul curent genereaza transport.

Pentru un sumator pe 4 biti avem:

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1$$
$$C_3 = G_2 + P_2 \cdot C_2$$
$$C_4 = G_3 + P_3 \cdot C_3$$

Inlocuind recursiv, in expresiile de mai sus, obtinem:

$$C_1 = G_0 + P_0 \cdot C_0$$

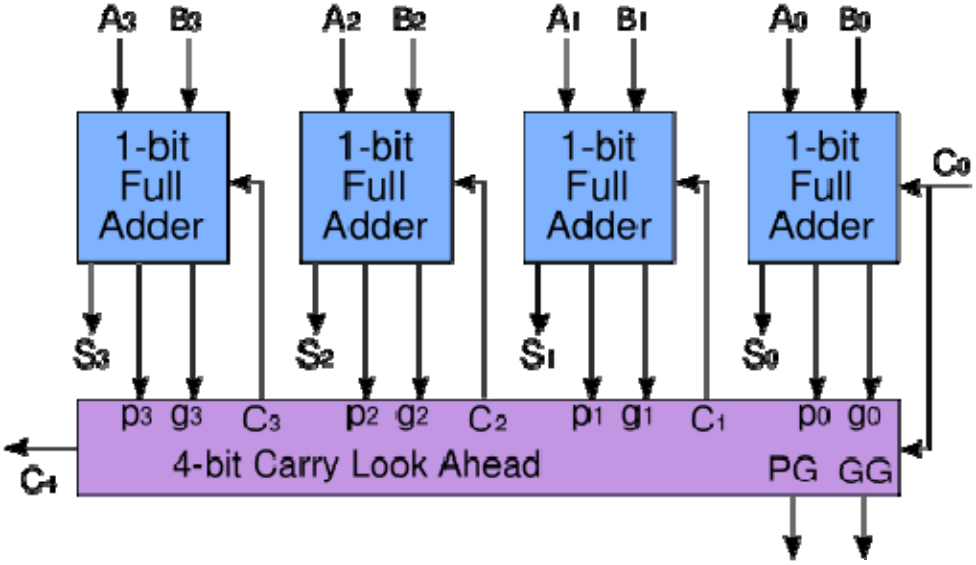
$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$

$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$

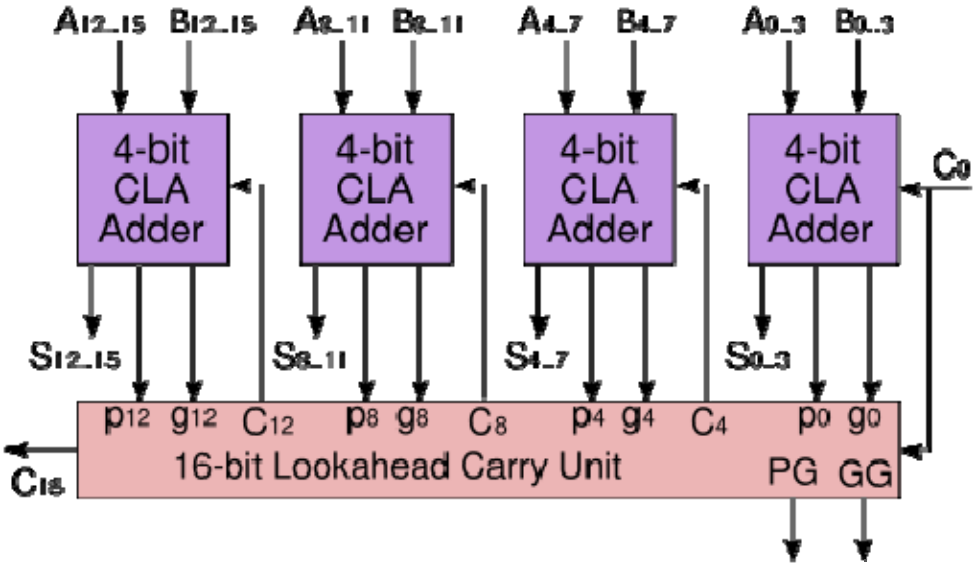
$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

$$G_i = A_i \cdot B_i \qquad P_i = A_i \oplus B_i$$

Se poate observa ca transportul la orice nivel poate fi determintat exclusiv pe baza P, G si C_0.



Mai multe unitati de acest fel pot fi folosite, utilazand acelasi principiu, pentru a obtine sumatoare de dimensiuni mai mari (pentru numere reprezentate pe mai multi biti):



$$C_4 = G_0 + P_0 \cdot C_0$$

$$C_8 = G_4 + G_0 \cdot P_4 + C_0 \cdot P_0 \cdot P_4$$

$$C_{12} = G_8 + G_4 \cdot P_8 + G_0 \cdot P_4 \cdot P_8 + C_0 \cdot P_0 \cdot P_4 \cdot P_8$$

$$C_{16} = G_{12} + G_8 \cdot P_{12} + G_4 \cdot P_8 \cdot P_{12} + G_0 \cdot P_4 \cdot P_8 \cdot P_{12} + C_0 \cdot P_0 \cdot P_4 \cdot P_8 \cdot P_{12}$$

Calcularea intarzierii unui astfel de sumator este un pic mai dificila la prima vedere:

- Pi si Gi se calculeaza cu 1 nivel logic
- Ci se calculeaza cu 3 niveluri logice
- Calculul transportului de intrare necesita 0, 4 si respectiv 5 si 5 niveluri logice
- Calculul sumelor partiale necesita 4, 7 si respectiv 8 si 8 niveluri logice

Ca urmare a acestor calcule se poate observa ca pentru un sumator pe 16 biti, implementat cu anticiparea transporului, intarzierea este de 8 ori mai mare decat intarzierea datorata unei porti logice (fata de 15x2+1=31 la sumatorul elementar).

Inmultirea

Inmultirea binara este asemanatoare celei zecimale. Ea se realizeaza prin insumarea produselor partiale dintre operandul A si fiecare digit din operandul B. Rezultatul inmultirii are un numar dublu de biti (fata de operanzi). Deoarece fiecare digit din B reprezinta o putere a lui 2, inmultirea este asigurata prin deplasarea la stanga a lui A. Algoritmul prevede urmatorii pasi:

- Se memoreaza cei 2 operanzi
- Pentru ficare digit (bit) din B:
    - Daca e 1 se aduna valoarea lui A (deplasata) la un acumulator (initial 0)
    - Se deplaseaza A la stanga cu o pozitie

Se observa faptul ca pentru a inmulti 2 numere pe N biti este nevoie de N-1 deplasari si (in cel mai defavorabil caz cand toti bitii din B sunt 1) de N adunari.

Algoritmul lui Booth

Pornind de la cazul anterior (cem mai ineficient), se observa ca, pentru un numar care contine o grupare de biti 1 inlantuiti (de exemplu numarul 15 reprezentat pe 4 biti = 1111), inmultirea poate fi privita ca o diferenta de produse partiale cu puteri ale lui 2 (inmultirea cu 15 este de fapt diferenata dintre inmultirea cu 16 si cu 1). Deoarece inmultirea cu o putere a lui 2 este echivalenta cu o deplasare la stanga (cu un numar de biti egali cu puterea), pentru cazul anterior

(al inmultirii unui numar reprezentat pe 4 biti cu 15), in loc de 3 deplasari si 4 adunari este nevoie de o deplasare si 2 adunari (conform celor prezentate anterior, pentru numerele in complement fata de 2, scaderea este echivalenta cu adunarea cu negatul operandului si transport initial 1).

Algoritmul lui Booth foloseste aceasta proprietate si reduce astfel numarul de operatii necesare.

Se analizeaza perechile de biti din B. Daca perechea este „00” sau „11” de trece mai departe (se deplaseaza o copie a lui A si se analizeaza urmatoarea pereche din B). Daca perechea este „01” se adauga prosusul partial corespunzator. Daca perechea este „10” se scade produsul partial corespunzator.

High-Radix

O alta metoda de a creste viteza de prelucrare (folosita mai ales in cazul operatiilor cu numere mari – cum ar fi criptarea RSA) in detrimentul resurselor utilizate, este High-Radix. Aceasta metoda presupune pseudo codificarea numerelor in baze superioare (un numar codat binar poate fi considerat ca fiind codat in hexazecimal daca se grupeaza seturi de 4 biti).

Presupunem inmultirea a doua numere reprezentate pe 32 de biti. Folosind algoritmul clasic va fi nevoie de 31 deplasari la stanga si maxim 32 de adunari.

Daca grupam cei 32 de biti in 8 grupuri de cate 4 (pseudo-reprezentamin hexazecimal), ca trebui sa operam 7 deplasari si maxim 8 adunari. Desigur deplasarile au aceeasi complexitate (numarul de pozitii cu care se face deplasarea este irelevant pentru intarzierea operatiei). Pentru a inmulti 2 grupuri de cate 4 biti avem nevoie insa de o metoda rapida. In acest caz se foloseste un LUT (LookUp Table), care pentru orice combinatie de perechi de 4 biti, intoarce rezultatul inmultirii. Se obtine astfel o intarziere mica (data de logica de selectie) in detrimentul resurselor folosite (LUT-ul poate fi comparat cu o memorie ce are latimea cuvantului de 8 biti si adancimea de 256 de cuvinte).

Impartirea

Impartirea a doua numere reprezentate in binar se face asemanator cu situatia in care ele sunt reprezentate in zecimal. Se incearca sa se scada impartitorul din deimpartit (incepand cu pozitia cea mai semnificativa).

Sa analizam impartirea lui 27 la 5 (rezultatul este in mod evident 5 si rest 2). Reprezentat in binar avem 00011011 impartit la 0101.

- Se incearca sa se scada 0101 din 00011, dar rezultatul este negativ si deci, in acest caz, cel mai semnificativ bit al rezultatului este 0.
- La pasul urmator se incearca sa se scada 0101 din 0110 si se obtine un bit 1 pentru cat si rest 0001

- La pasul urmator se incearca sa se scada 0101 din 0011 si se obtine iar 0
- La pasul urmator se incearca sa se scada 0101 din 0111 si se obtine un bit 1 si rest 0010

Rezultatul final este: Catul 0101 si rest 0010

Implementarea

Utilizand resursele din cadrul laboratorului se urmareste integrarea unei unitati aritmetice (implementate conform algoritmilor prezentati) intr-o interfata ce va permite testarea operatiilor introducand de la tastatura cei doi operanzi si pentru operatia curenta rezultatul va fi afisat pe monitor (impreuna cu cei 2 operanzi pentru a permite verificarea corectitudinii).