Universitatea	Tehnica din Cluj-Napoca
Catedra	de Calculatoare
Titlul proiectului	Unitate aritmetico-logica
Student	Ursache Vlad
Numarul grupei	30238-2
Indrumator	S.l. dr. ing. Lisman Florin
Data	22 decembrie 2022

Cuprins

Cuprins

1 Introducere

[1] Unitatea aritmetico-logica, ALU, este un circuit electronic digital complex care poate efectua operatii aritmetice și logice.
Orice sistem de calcul non-trivial contine o unitate atirmetico-logica (ALU). Implementarea acestuia consta in implementarea operatiilor de adunare, scadere, inmultire, impartire, operatii logice si de deplasare pe biti.
Solutia propusa este implementata pe o plcuta FPGA Basys 3. Aceasta este compusa dintr-un debouncer pentru butonul de Reset (al unitatilor secventiale), un sumator/scazator cu anticipare a transportului, un inmultitor prin metoda Booth, un impartitor fara refacerea restului partial, o unitate de operatii logice si de deplasare si un afisor pe 7 segmente. ALU-ul obtinut functioneaza pe 32 biti. Din motive de limitare hardware, vom lucra cu operanzi pe 6 biti (atatea switch-uri sunt prezente pe placuta). Testele din simulator sugereaza corectitudinea implementarii.
Aceste componente sunt bazate pe design-uri standard, non-triviale, adaugand complexitate proiectului.
Urmatoarele sectiuni vor descrie visceral fundamerarea teoretica, pasii de proiectare si implementare, rezultatele experimentale si concluziile.

2 Fundamentare teoretica

2.1 Sumator/scazator cu anticipare a transportului

Sumatoare cu anticipare a transportului pe 8 biti sunt cascadate cate 4 pentru a obtine un circuit care opereaza pe 32 de biti. Am ales aceasta configuratie din motive de eficienta. Calea de date este mai scurta.
Sumatorul cu anticiparea transportului are o viteza mai ridicata (pentru numar relativ mic de biti) decat sumatorul cu transport succesiv. Este asadar mai rapid/eficient. Alt avantaj semnificativ este reutilizarea componentei ca scazator aritmetic. Dezavantajul consta in spatiul suplimentar in circuitul integrat datorita complexitatii ridicate. Sunt folosite semnale aditionale si un generator de transport anticipat ([2] conform laboratorului 4: Circuite aritmetice combinationale).
Orice alt design putea fi folosit pentru adunare. Am fi putut folosi un sumator cu salvare a transportului, dar nu ar fi putut fi folosit la intregul lui potential pe doar 2 operanzi.

2.2 Inmultitor prin metoda Booth

Aceasta componenta secventiala urmeaza cu sfintenie algoritmul de inmultire prin metoda Booth. Organigrama de mai jos descrie in detaliu pasii implicati. Am folosit acest tip de inmultire pentru a putea lucra cu numere cu semn.

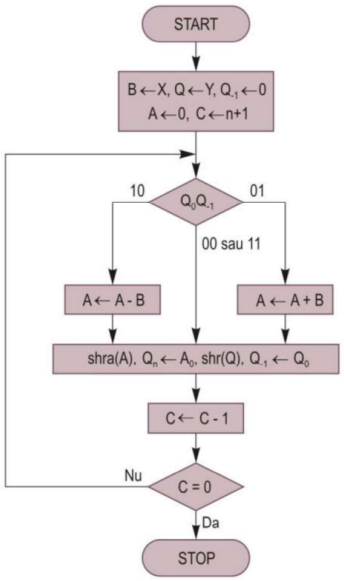

Figura 1: [2] Organigrama inmultire prin metoda Booth (laborator prof. Baruch)

2.3 Impartitor fara refacerea restului partial

Aceasta componenta secventiala urmeaza cu sfintenie algoritmul de impartire fara refacerea restului partial. In cazul impartirii la 0, utilizatorul este avertizat de acest lucru prin aprinderea unui LED. Organigrama de mai jos descrie in detaliu pasii implicati. Am folosit acest tip de impartire pentru eficientizarea operatiei, in detrimentul utilizatii impartirii cu refacerea restului partial ([2] conform Laboratorului 5: Circuite aritmetice secventiale).

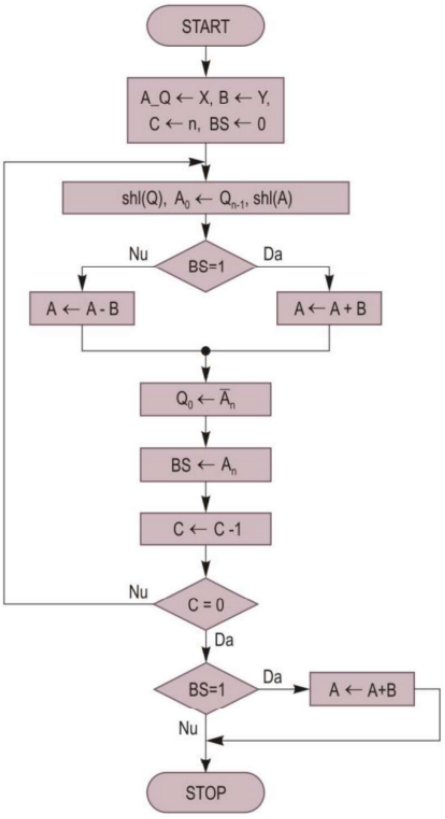

Figura 2: [2] Organigrama impartire fara refacerea restului partial (laborator prof. Baruch)

2.4 Unitate logica si de deplasare pe biti

Aceasta componenta este triviala. Face toate operatiile logice (not, and, or, xor) si de deplasare (shr, shl, sar, sal, ror, rol) obisnuite.

2.5 Debouncer

Aceasta componenta se asigura ca butonul de Reset al componentelor secventiale (inmultitor, impartitor si afisor pe 7 segmente). Introducerea generatorului de mono-impuls opreste efectul de flicker al afisorului la momentul resetarii.

2.6 Afisor pe 7 segmente

Pe acest display afisam rezultatele operatiilor introduse de la switch-uri. Datele de la intrarile componentei (rezultate) sunt transformate in semnale de iesire anod/catod, multiplexate in timp, astfel incat sa confere iluzia persistentei cifrelor pe ecran. Descrierea aprofundata nu face neaparat subiectul acestei discipline.

3 Proiectare si implementare

Proiectarea a inceput cu etapa de documentare. Operatiile fundamentale au ramas, iar cele nefezabile, au fost abandonate. Printre operatiile din urma se numara si algoritmul q_invsqrt folosit

pentru prima data in Quake 3. Din pacate, functioneaza doar in virgula flotanta.
Operatiile ramase au fost gandite sa fie cat mai eficiente, indiferent de complexitate. Asa se justifica existenta sumator/scazator-ului, inmultitorului si impartitorului. Fiecare dintre acestea au fost implementate cu ajutorul laboratorului.
ALU-ul poate fi utilizat prin introducerea primului operand pe primele 6 switch-uri din stanga. Urmatoarele 4 switch-uri codifica operatia de efectuat. Ultimele 6 switch-uri reprezinta cel de-al doilea operand. Pentru afisarea rezultatului corect, este necesara apasarea butonului de Reset in cazul cailor de date secventiale (inmultire si impartire).
Codurile operatiilor sunt:

```
-- 0000 add
-- 0001 sub
-- 0010 mul
-- 0011 div
-- 0100 not
-- 0101 and
-- 0110 or
-- 0111 xor
-- 1000 shr
-- 1001 shl
-- 1010 sar
-- 1011 sal
-- 1100 ror
-- 1101 rol
```

Figura 3: Codificarea operatiilor pe ALU

Mai multe detalii referitoare la componente regasim in sectiunea 2 a documentatiei.

4 Rezultate experimentale

O validare minimalista a operatiilor a fost realizata cu ajutorul unor bancuri de simulare. Au fost folosite atat numere mici, cat si numere mari sau edge case-uri.

4.1 Testare sumator/scazator

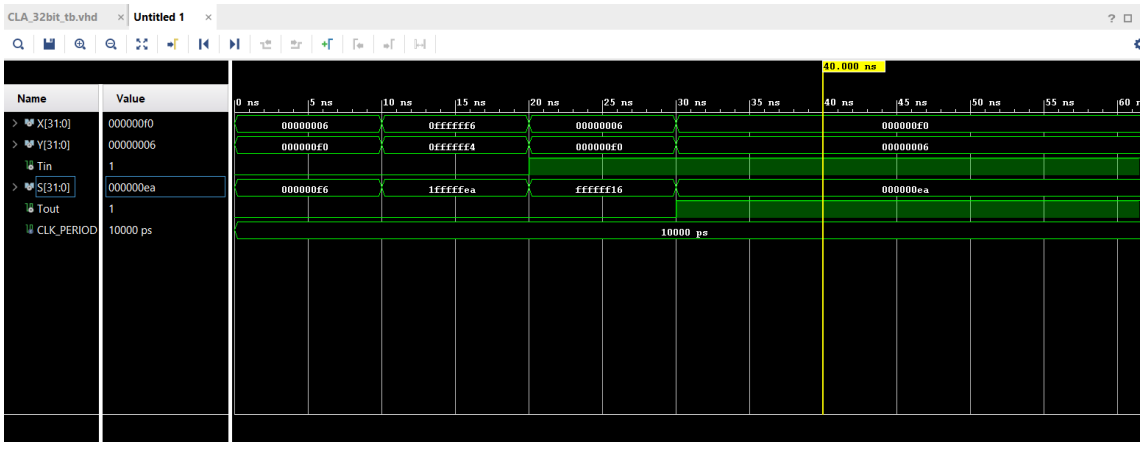

Figura 4: Banc de test pentru sumator/scazator

4.2 Testare inmultitor

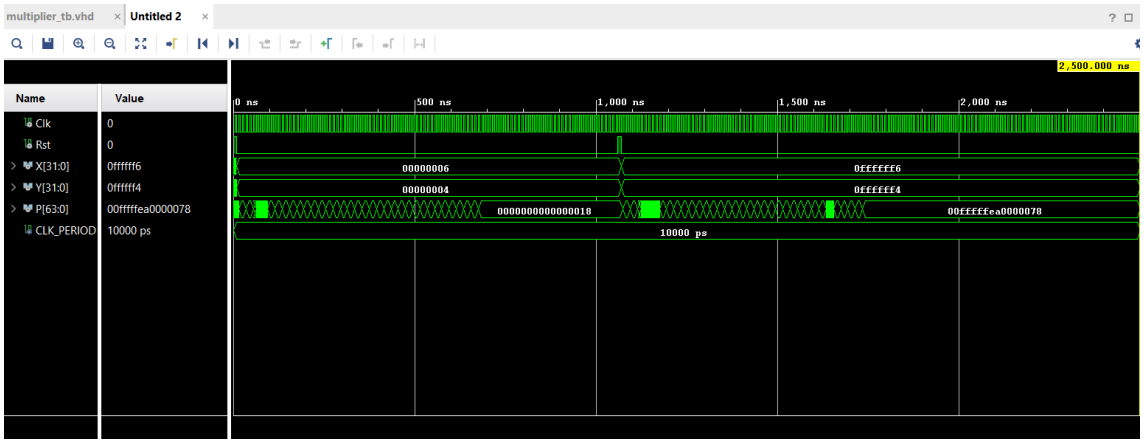

Figura 5: Banc de test pentru inmultitor

4.3 Testare impartitor

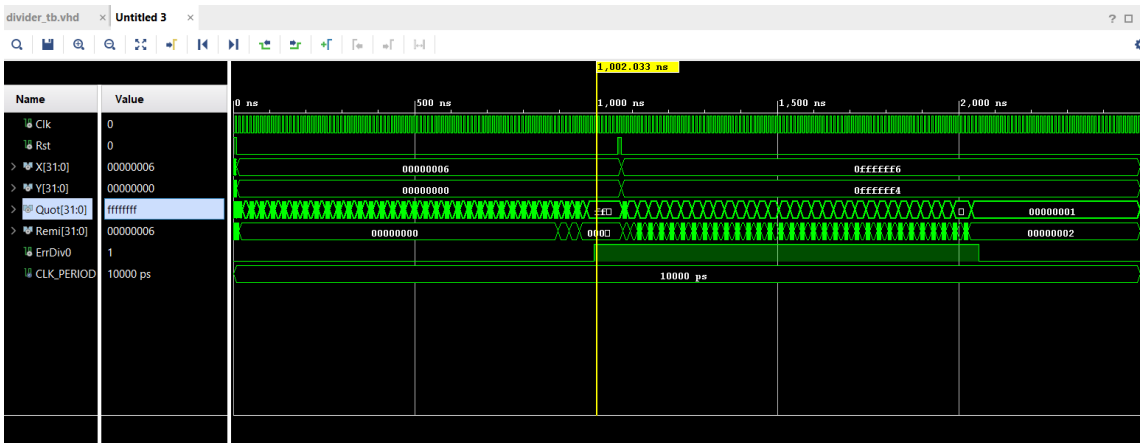

Figura 6: Banc de test pentru impartitor

4.4 Mentiuni testare

Operatiile logice si de deplasare pe biti sunt triviale. Testarea lor se poate face on-site.
Singurul pericol al utilizarii acestui ALU este overflow-ul (in special la adunare si inmultire). Acest aspect este acoperit de limitarea hardware (insuficiente switch-uri pentru modelarea unui numar pe 32 biti fara a deveni foarte greu de utilizat).

5 Concluzii

Am invatat cum opereaza un sistem de calcul cu valori numerice cu semn. In cele din urma, pana si cel mai dificil state-machine functioneaza, iar proiectul este 100% operational.
Daca nu ne referim la debouncer si SSD, proiectul este 100% original (codul, nu conceptul).
Marele dezavantaj, overflow-ul la adunare si inmultire este acoperit de limitarile hardware, deci nu constituie neaparat o problema.
Aceste limitari hardware fac dificila (sau poate chiar nejustificata) introducerea unui flag de overflow, intrucat afisorul pe 7 segmente este deja depasit. De numarul switch-urilor nu mai vorbim.

6 Bibliografie

[1]https://en.wikipedia.org/wiki/Arithmetic_logic_unit
[2]https://users.utcluj.ro/ baruch/ro/pages/cursuri/structura-sistemelor-de-calcul/laborator.php

A Rezumat

Design-ul elaborat poate fi consultat direct din proiect sau in imaginea de mai jos. Inarmati cu schema elaborata a proiectului, codul vorbeste de la sine. Din motive de simplitate si evitare a confuziilor legate de ierarhii, codul nu va fi atasat in anexa, ci alaturi de documentatie.

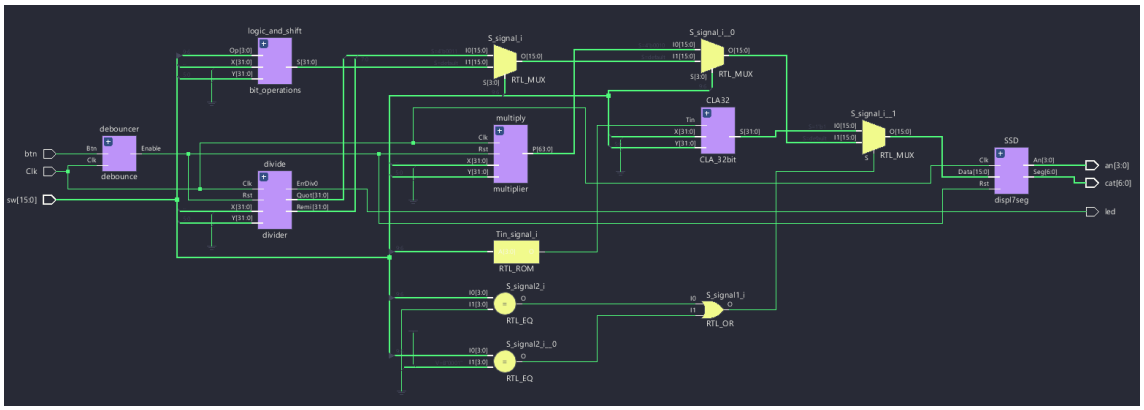

Figura 7: Design elaborat