Universitatea Tehnică din Cluj-Napoca

Catedra de Calculatoare

CIRCUITE DE ÎNMULȚIRE ZECIMALĂ

~STRUCTURA SISTEMELOR DE CALCUL~

Nume și prenume: Moșilă Luciana

Grupa: 30236

Profesor coordonator: Dragoș Florin Lișman

Ianuarie 2024

Cuprins

1. Rezumat

Proiectul propus a avut ca scop implementarea a două metode de înmulțire a două numere în BCD (Binary Coded Decimal) folosind limbajul VHDL în mediul de dezvoltare Vivado Design Suite, cu rezultatele vizibile pe placa de dezvoltare FPGA Basys 3. Într-o abordare inovatoare, proiectul a combinat două tehnici distincte de înmulțire, oferind astfel o soluție eficientă pentru manipularea numerelor BCD. Pentru implementare, s-au utilizat resursele și utilitățile oferite de mediul Vivado Design Suite, asigurând un cadru de dezvoltare corespunzător și ușor de gestionat. S-a realizat o descriere structurală a circuitelor cu ajutorul limbajului de descriere hardware VHDL. Proiectul ofera o soluție inovatoare și eficientă pentru manipularea acestor tipuri de date în cadrul sistemelor embedded.

2. Introducere

Sistemul de numerație zecimală reprezintă cea mai naturală și larg utilizată metodă de operare și gestionare a datelor numerice în interacțiunile dintre oameni și între om și calculator. Aritmetica zecimală este preferată în mediile de procesare a informațiilor numerice, cum ar fi aplicațiile științifice, financiare, economice și diverse programe online. Cu toate acestea, computerele generale din zilele noastre efectuează operații zecimale folosind aritmetica binară, deoarece informațiile binare pot fi stocate și manipulate eficient. Cu toate acestea, există motivații pentru trecerea la aritmetica zecimală, începând de la naturaleța operațiilor pentru ființa umană și până la discrepanțele dintre reprezentările binare și zecimale ale aceleiași valori. Valorile binare în virgulă mobilă pot doar să aproximeze anumite valori zecimale, ca în cazul reprezentării 0.1 în binar. Aceasta poate genera discrepanțe semnificative în rezultatele calculelor, iar în domenii precum instituțiile financiare, corespondența precisă între rezultatele generate de calculator și calculele manuale este esențială. Prin urmare, dispozitivele hardware care susțin aritmetica zecimală au devenit din ce în ce mai importante, oferind o soluție eficientă și precisă pentru operațiile zecimale într-un context de creștere a necesităților de putere de procesare. Astăzi, unitățile hardware pentru aritmetica zecimală sunt integrate ca parte esențială a procesoarelor generale, îmbunătățind semnificativ performanța operațiilor complexe precum înmulțirea în contexte în care algoritmii hardware anteriori erau lenti și iterativi.

Proiectul propune dezvoltarea, implementarea și testarea unui circuit logic de calcul dedicat efectuării operațiilor de înmulțire zecimală asupra numerelor de câte 4 cifre. Se vor examina diverse metode existente de înmulțire, iar sistemul proiectat va integra două dintre aceste tehnici, oferind utilizatorului posibilitatea de a alege metoda preferată pentru calcul. Proiectarea sistemului va adopta o abordare structurală, folosind limbajul de descriere hardware VHDL, iar implementarea va implica utilizarea unei plăci de dezvoltare FPGA. Prin intermediul acestei plăci, se va realiza implementarea circuitului proiectat, permitând utilizatorului să introducă numere prin intermediul switch-urilor plăcii, să selecteze metoda dorită și să

vizualizeze rezultatele operațiilor. Soluția propusă pentru implementare pe placa FPGA va integra tehnici precum "The Nine-Multiples-of-Multiplicand" și "The Repeated-Addition", furnizând astfel o abordare robustă și flexibilă pentru înmulțirea zecimală a numerelor de 4 cifre.

Tehnica "The Nine-Multiples-of-Multiplicand" constă în generarea și adunarea a nouă multiple ale multiplicandului, pe baza fiecărei cifre a multiplicatorului. Această metodă exploatează proprietățile algebrice ale sistemului de numerație zecimală, reducând înmulțirea la adunări repetate. Pe de altă parte, tehnica "The Repeated-Addition" se bazează pe principiul adunărilor repetate ale multiplicandului, în funcție de fiecare cifră a multiplicatorului. Această metodă simplifică operația de înmulțire la o serie de adunări succesive, fiind intuitivă și ușor de înțeles. Ambele tehnici oferă abordări diferite pentru efectuarea operațiilor de înmulțire și sunt integrate în soluția propusă pentru proiectul de calcul zecimal, aducând astfel contribuții specifice la eficiența și versatilitatea circuitului implementat.

Următoarele capitole ale lucrării vor concentra atenția asupra prezentării soluției adoptate pentru a satisface cerințele proiectului, detaliind mecanismele utilizate, etapele de dezvoltare, rezultatele obținute, concluziile și posibilele direcții de dezvoltare ulterioară ale sistemului construit. Capitolul 3, intitulat "Fundamentare teoretică", va furniza un rezumat concis al teoriei fundamentale care stă la baza dezvoltării proiectului, contextualizând progresul realizat până în acest moment. Partea centrală a lucrării, Capitolul 4, denumit "Proiectare și implementare", va reprezenta nucleul analizei, evidențiind fiecare etapă a procesului de realizare a proiectului și argumentând alegerile făcute în cadrul acestor etape.

3. Fundamentare teoretică

Înmulțirea zecimală este mai complexă decât înmulțirea binară datorită numărului crescut de cifre reprezentabile în sistemul de numerație zecimal (de la 0 la 9), pe când in baza 2, există doar două cifre binare, și anume 0 și 1. Accelerarea nevoii de procesare a datelor masive și ușurarea comunicării om-calculator a dus la evoluția sistemelor de calcul în aritmetică zecimală. Această evoluție este încă în progres, iar acest proiect își propune să materializeze prin implementare un astfel de sistem capabil de înmulțire a numerelor zecimale de 4 cifre pe o plăcuță de dezvoltare FPGA.

3.1. Soluții posibile ale proiectului

Metodele principale de înmulțire zecimală a numerelor cu semn sunt următoarele:

- "Repeated-addition method"
- "Nine-multiples-of-multiplicand method"
- "The Right-And-Left-Hand Components method"
- "Doubling-and-halving method"

3.1.1. **"Repeated-addition method"**

Această abordare reprezintă cea mai simplă metodă, totuși, din perspectiva timpului de execuție, este considerată cea mai lentă. Procesul începe prin urmărirea cifrelor înmulțitorului, începând cu cifra unităților, și calculează produsele parțiale prin adunări repetate ale deînmulțitului, decrementând cifra înmulțitorului până când aceasta atinge valoarea 0. După evaluarea sumei corespunzătoare unei cifre, grupul de registri care stochează produsul parțial este deplasat spre dreapta cu o poziție, urmând ca procesul să continue cu cifra următoare a înmulțitorului. Acest procedeu se repetă până când toate cifrele ale înmulțitorului sunt procesate. Pentru

a optimiza soluția, se poate considera înlocuirea sumatorului obișnuit cu un sumator-scazător în locul celui utilizat în proiect.

3.1.2.“Nine-Multiples-of-Multiplicand method”

Această tehnică presupune generarea a 9 multipli ai deînmulțitului la începutul operației și gruparea lor în registri speciali. Evaluarea fiecărei cifre a înmulțitorului determină selectarea registrului care contribuie la produsul parțial. Deși metoda celor 9 multipli este eficientă în ceea ce privește rezultatele obținute, atât în ceea ce privește costul, cât și timpul, devine mai laborioasă din cauza necesității de a calcula și încărca acești multipli în regiștrii speciali.

3.2. Metode și tehnologii utilizate

3.2.1.Limbajul utilizat

Pentru elaborarea acestui proiect, vom recurge la utilizarea limbajului de descriere hardware VHDL (Very High Speed Integrated Circuit Hardware Description Language). VHDL este un limbaj de programare specific pentru descrierea circuitelor digitale și arhitecturilor acestora. Având ca scop principal descrierea comportamentului și structurii sistemelor logice de calcul, VHDL va fi folosit pentru a realiza o descriere structurală detaliată a circuitului de înmulțire zecimală propus în acest proiect. Prin intermediul VHDL, vom defini modulele, interconexiunile și funcționalitățile circuitului, facilitând astfel simularea, sinteza și implementarea pe o placă de dezvoltare FPGA. Alegerea VHDL oferă un cadru eficient și precis pentru dezvoltarea sistemelor digitale, contribuind la realizarea proiectului cu acuratețe și robustețe.

3.2.2.Tehnologia utilizată

Instrumentul software utilizat este Vivado 2017.4 Design Suite, care oferă facilități și suport pentru dezvoltarea proiectului prin posibilitatea de scriere cod HDL, editor schematic, sinteză, simulare, implementare pe placa FPGA.

Placa FPGA care va fi utilizată este Basys 3 datorită componentelor și uneltelor ce asigură implementarea eficientă a proiectului pe placa de dezvoltare.

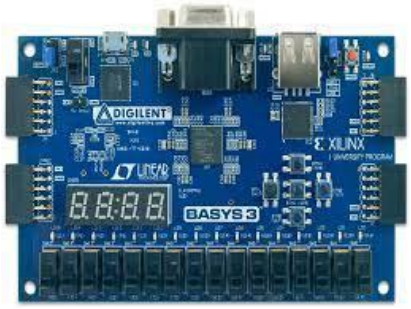


Fig. 1 Placa FPGA Basys3

3.3. Soluția propusă: Scenariu de urilizare

Utilizatorl trebuie sa actioneze butonul „U18” de fiecare data cand se va trece de la o etapa la alta. Etapele sunt:

Ledurile 8...3 indica etapa, iar ledurile 1 si 2 (LD2-X, LD1-Y) mentioneaza daca este o eroare prin introducerea unor date, invalide, in BCD pe switch-uri.

Inititializare(LD8 activ);

Introduceti metoda(LD7 activ si LD0(0 sau 1) , se va afisa pe SSD care dintre metode este selectata 1 sau 2);

Introduceti X (LD6 activ pe ssd se va vedea numarul in BCD ales);

Introduceti Y (LD5 activ pe ssd se va vedea numarul in BCD ales);

Rezultatul (LD3 activ).

Se poate realiza resetarea circuitului in orice moment prin butonul „T18”.

Pentru testarea circuitului, se efectuează următoarele operații:

1) Utilizatorul va alege ce metodă va fi utilizată pentru operația de înmulțire astfel: Switch-ul 0 (SW0) va alege una din metode. Cand Switch-ul nu este activ se va alege prima metoda “Repeated-addition method”, în timp ce activarea acestuia va selecta metoda “Nine-multiples-of-multiplicand method”.
2) In etapapele urmatoare se vor selecta inmultitorul si deinmultitul prin Switch-urile placii(SW15 ... SW0), structura datelor transmise catre circuit este urmatoarea: primele 4 switch-uri (de la stânga la dreapta) reprezintă codul BCD pentru cifra miilor, următoarele 4 switch-uri pentru cifra sutelor, următoarele 4 switch-uri pentru cifra zecilor și ultimele 4 pentru cifra unităților. În același timp, pe afișajul BCD 7 segmente vor apărea modificările efectuate.
3) Dupa efectuarea pasilor mentionati se va ajunge la etapa de afisarea rezultatelor. Pe SSD se pot vedea datele in functie de combinatia Switch-urilor 1 si 0.

	SW1	SW0
PH	1	0
PL	0	0
X	0	1
Y	1	1

PH: bitii cei mai semnificativi ai produsului rezultat

PL: bitii cei mai putini semnificativi ai produsului rezultat

X,Y operanzii utilizati

4. Proiectare și implementare

4.1. Arhitectura sistemului

O abordare integrată pentru implementarea a două metode distincte de înmulțire zecimală, evidențiind arhitectura generală a circuitului care cuprinde metodele de înmulțire alese. Structura globală a circuitului include registre specializate destinate stocării datelor introduse prin switch-urile plăcii FPGA, precum și unitatea de control, un automat de stare, care generează semnalele esențiale pentru activarea componentelor din sistem, asigurând buna funcționare a întregului circuit.

Având în vedere natura secvențială a circuitului și implementarea sa pe o placă de dezvoltare FPGA, unde intrările sunt gestionate prin intermediul butoanelor plăcii, am inclus un circuit de debounce pentru a preveni eventualele instabilități ale sistemului. De asemenea, circuitul dispune de componente logice pentru detecția erorilor, afișarea datelor pe un display BCD cu 7 segmente și gestionarea intrărilor de activare a registrelor menționate anterior.

În ceea ce privește circuitul de detecție a erorilor, acesta verifică dacă valorile introduse pentru deînmulțit și înmulțitor sunt valide, identificând orice cifră BCD care depășește valoarea 1001_2, adică cifra 9, pentru a asigura corectitudinea datelor introduse în sistem.

RTL:

If X(15 downto 12)>"1001" or X(11 downto 8)>"1001" or X(7 downto 4)>"1001" or X(3 downto 0)>"1001" then

errorX <= ‘1’

Else errorX <= ‘0’

If Y(15 downto 12)>"1001" or Y(11 downto 8)>"1001" or Y(7 downto 4)>"1001" or Y(3 downto 0)>"1001" then

errorY <= ‘1’

Else errorY <= ‘0’

RTL:

If metoda = ‘0’ then

P <= P1

Else

If metoda = ‘1’ then

P <= P2

Else

P <= 0;

Unitatea de comandă a sistemului reprezintă un automat de stare cu număr finit de stări, aceasta fiind responsabilă pentru generarea de semnale responsabile pentru activarea regiștrilor și circuitelor din sistem.

Ca intrări, acest automat are intarea de tact, o intrare de reset, o intrare care marchează finalul operației de înmulțire, o intrare care semnalează o eroare apărută în funcționare și o intrare pentru trecerea la următorul pas.

Ieșirile acestui circuit reprezintă un vector de 8 biți, un singur bit fiind activ la un moment dat, cel corespunzător stării curente a automatului. Acesta are un număr de 8 stări:

- Idle – starea de start a sistemului

- selectMetoda – în această stare se face selecția metodei. Automatul activează ieșirea pentru scrierea în registrul care stochează metoda introdusă
- introdX - în această stare se introduce deînmulțitul. Automatul activează ieșirea pentru scrierea în registrul care stochează deînmulțitul.
- introdY - în această stare se introduce înmulțitorul. Automatul activează ieșirea pentru scrierea în registrul care stochează înmulțitorul.
- calcul – această stare marchează faptul că circuitul corespunzător metodei alese este in proces de calculare a rezultatului operației de înmulțire.
- Stop – această stare marchează finalul operației.

Tranzițiile dintre stările automatului sunt prezentate în figura următoare:

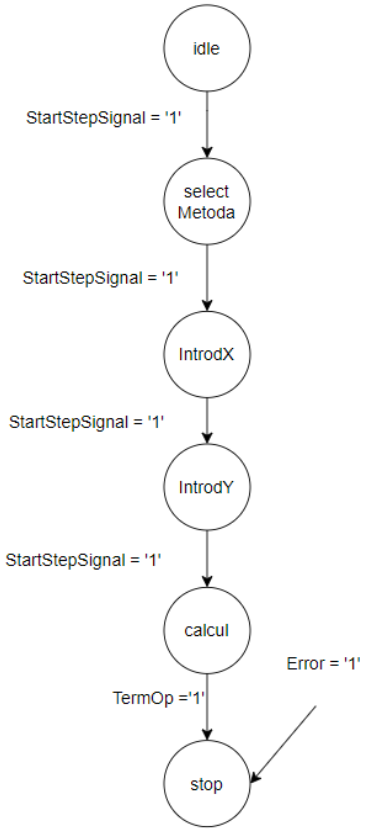


Fig.2 Diagrama de tranziții pentru FSM al sistemului general(https://app.diagrams.net/)

4.1.1.Metoda „Repeated-Addition”

Schema unui înmulțitor zecimal care implementează această metodă este prezentată în figura următoare:

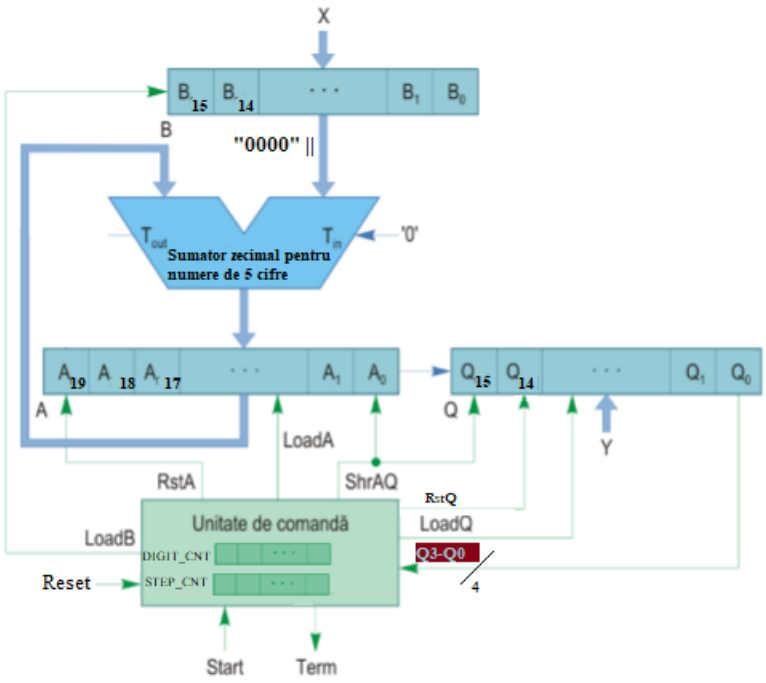


Fig.3 Inmultitorul zecimal[1]

Componentele acestui circuit sunt următoarele:

- B – registru utilizat pentru stocarea deînmulțitului
- A și Q – registre de deplasare cu încarcare paralelă și intrare serială utilizate pentru stocarea produsului parțial
- Sumator zecimal pentru numere zecimale de 5 cifre utilizat pentru calculul produsului parțial
- Unitate de comandă – automat cu stări finite care generează semnale responsabile pentru încarcare paralelă în registrewe, shiftare, terminare operație

Întrucât se înmulțesc două numere zecimale de 4 cifre, registrul B pentru stocarea deînmulțitului va fi de 16 biți (4 cifre x 4 biți necesari pentru reprezentarea unei cifre în cod BCD). Registrul Q va fi de asemenea de 16 biți. Inițial în acest registru se stochează înmulțitorul. După evaluarea unei cifre a

înmulțitorului, conținutul registrului va fi deplasat spre dreapta. Registrul acumulator (A) va avea nevoie de 20 de biți, întrucât pe primele 4 poziții va fi înscris transportul, ca cifră zecimală, rezultat în urma adunărilor repetate a deînmulțitului. Necesitatea acestei reprezentări este dată de faptul că adunarea repetată a deînmulțitului poate genera overflow mai mare decât 1. (De exemplu, adunarea repetată a valorii 9999 de 5 ori este 49995, ce trebuie reprezentat pe 20 biți). În acest sens, sumatorul zecimal este folosti pentru adunarea a două numere zecimale de 5 cifre, astfel că un operand este registrul Acumulator, iar celălalt este deînmulțitul, la care i se concatenează în față cifra 0.

Organigrama acestui circuit este redată în figura următoare:

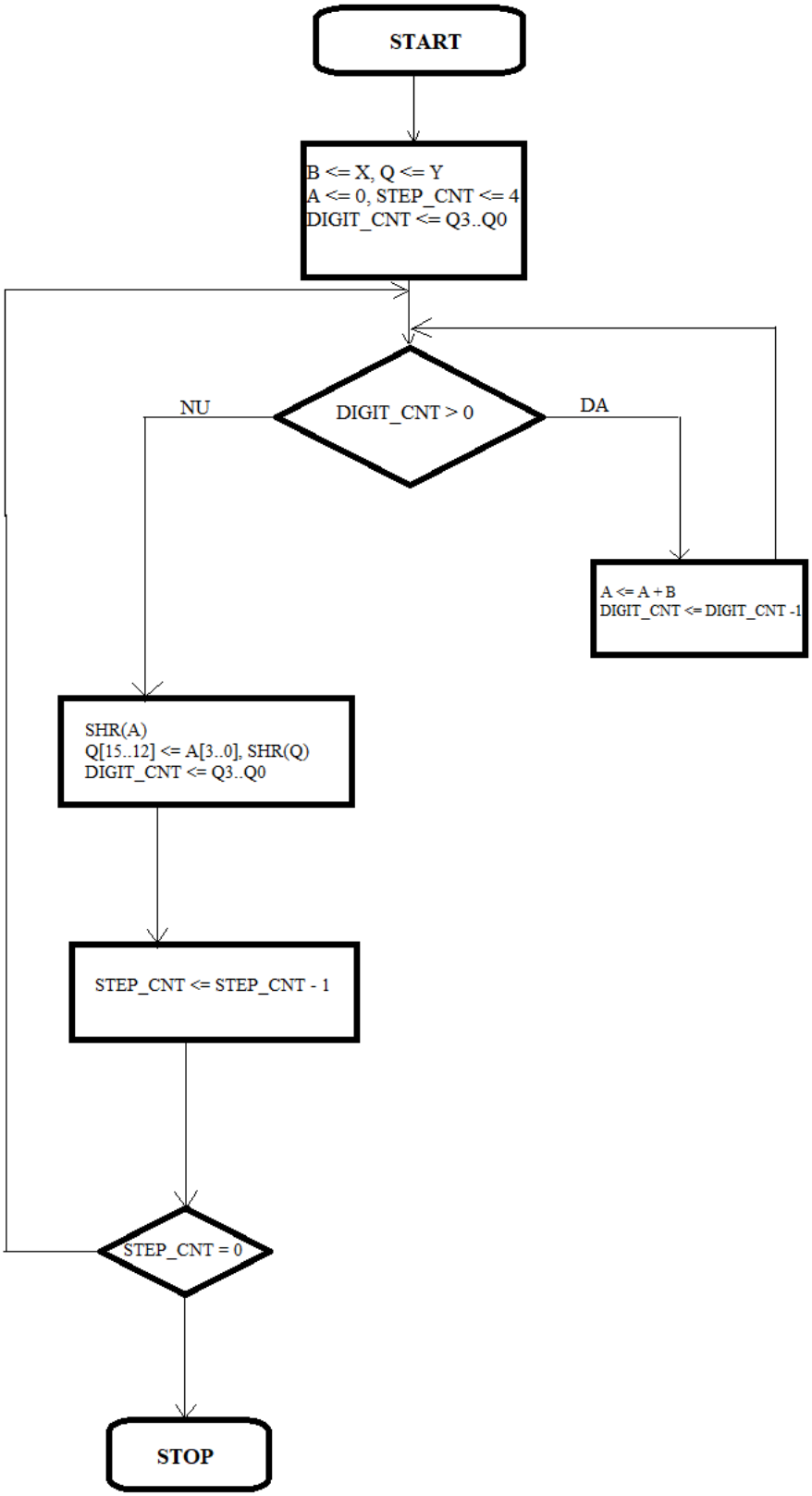


Fig.4 Organigrama algoritmului de înmulțire zecimală prin metoda “adunărilor repetate”

4.1.2.Metoda “The Nine-Multiples-of-Multiplicand”

Un înmulțitor zecimal care implementează această metodă este prezentat în figura urmatoare:

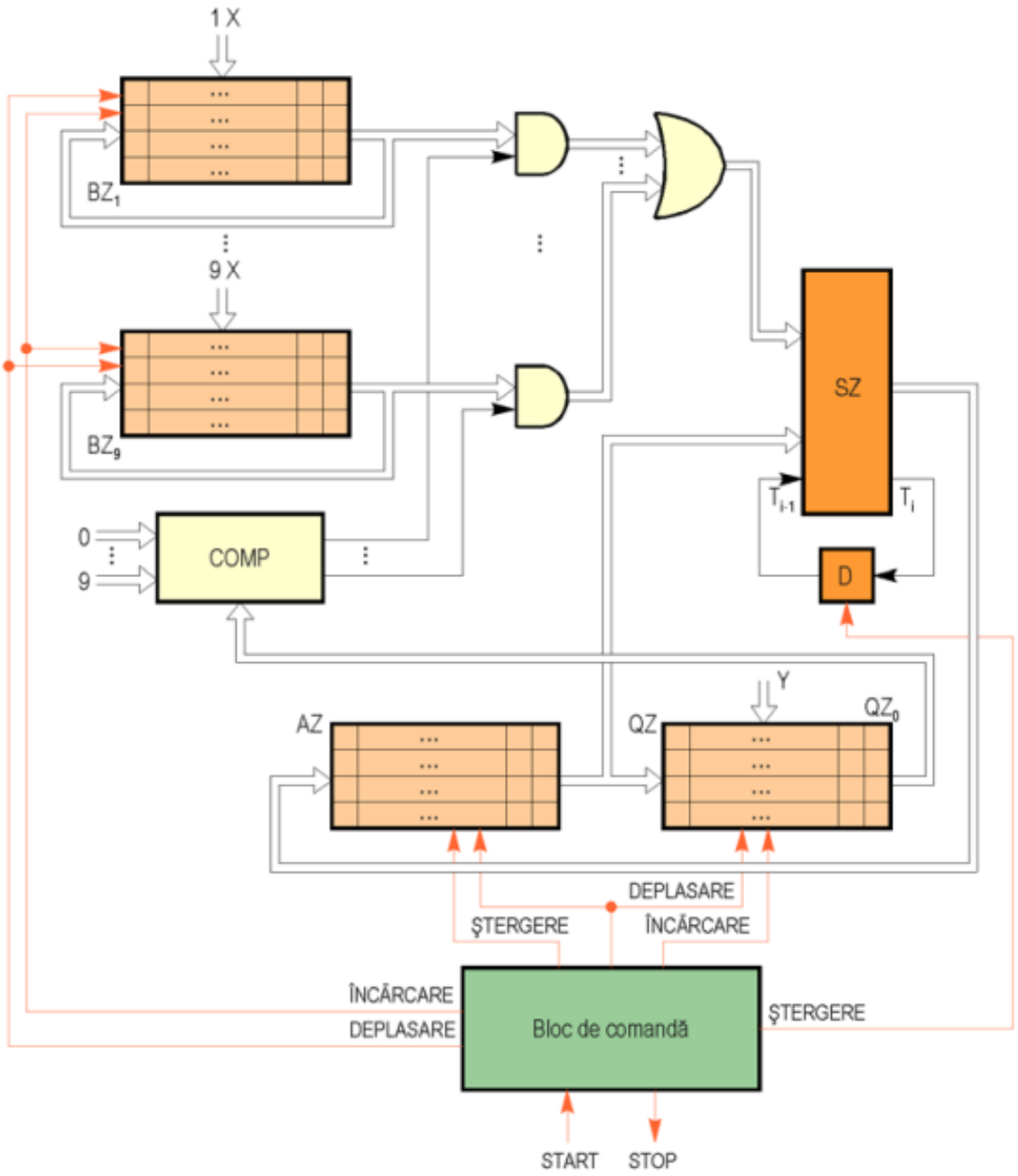


Fig.5 Diagramă bloc pentru circuitul pentru metoda “The Nine-Multiples-of-Multiplicand” [2]

Componentele circuitului sunt următoarele:

- AZ – grup de regiștri utilizați ca acumulator pentru stocarea produsului parțial
- QZ – grup de regiștri utilizați pentru stocarea înmulțitorului
- BZ1 ... BZ9 – grup de regiștri utilizați pentru stocarea multiplilor deînmulțitului
- SZ – sumator zecimal pentru calcularea produsului parțial
- D – latch pentru stocarea carry digit provenit de la sumatorul zecimal
- COMP – circuit de comparație care compară la fiecare pas cifra curentă a înmulțitorului , QZ_i, cu indexul regiștrilor BZ_i. Utilizând o serie de porți ȘI, informația registrului activ BZ_i este transmis la intrarea sumatorului zecimal.

Această metodă constă în generarea celor 9 multiplii ai deînmulțitului și salvarea acestora în niște regiștri speciali. Acești regiștri sunt de 20 de biți, adică de 5 cifre zecimale. Sumatorul zecimal utilizat este pentru adunarea a două numere de 5 cifre, întrucât un operand provine de la unul din regiștrii amintiți anterior, iar celălalt provine din registrul acumulator. Registrul acumulator este un regisutru de deplasare dreapta cu încărcare paralelă sincronă și reset sincron. În acest regisutru se vor stoca produsele parțiale și la final, împreună cu registrul Q, va forma rezultatul înmulțirii. Registrul Acumulator este pe 20 de biți, o cifră zecimală în plus fiin adăugată pentru salvarea transportului de la sumatorul zecimal. În rezulatul final, primele 4 poziții din registrul acumulator nu vor fi luate în considerare. Registrul Q este un registru de deplasarea dreapta cu încărcare paralelă și reset sincron. Inițial acesta conține înmulțitorul și la fiecare pas va fi deplasat spre dreapta împreună cu registrul acumulator, după evaluarea unei cifre a înmulțitorului și efectuarea operațiilor de adunare la produs parțial.

Organigrama acestui circuit este redată în figura următoare:

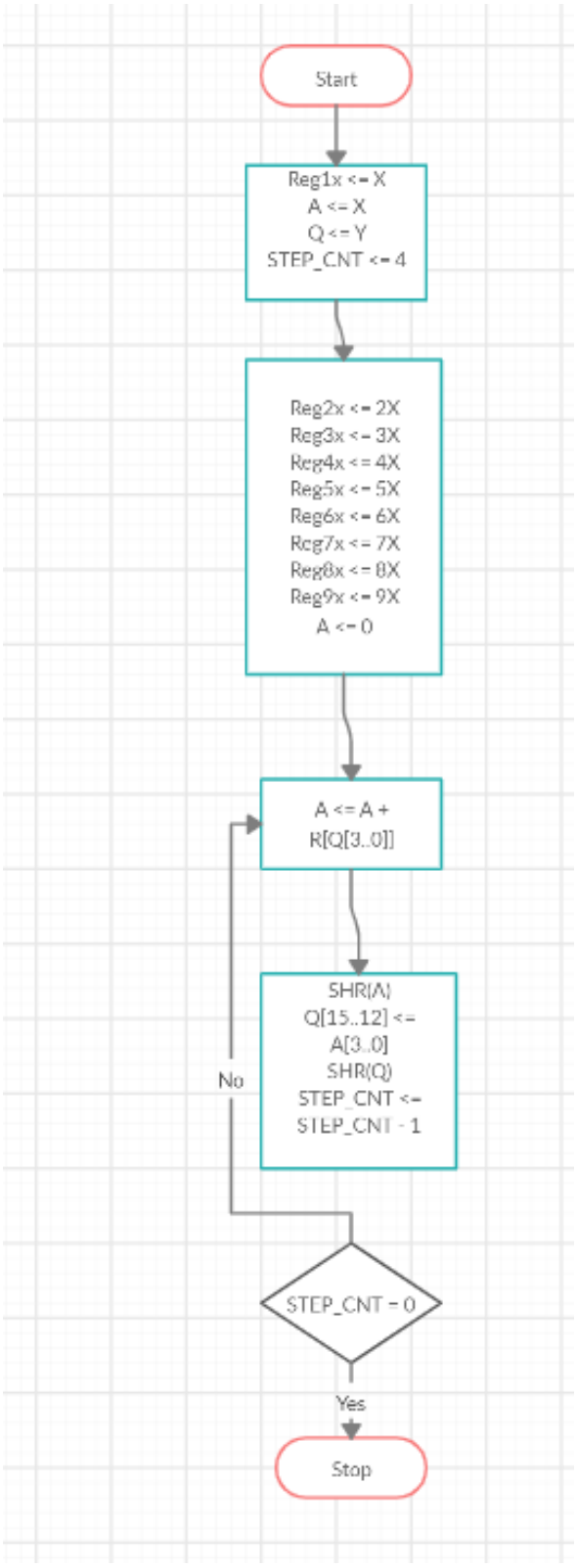


Fig. 6 Organigrama algoritmului de înmulțire zecimală prin metoda "celor 9 multiplii ai deînmulțitului"

4.1.3 Sumatorul zecimal

Reprezentarea numerelor zecimale se face cu ajutorul codului BCD. Sumatorul zecimal adună două cifre BCD în parallel și generează o sumă tot în codul BCD. Deoarece cifrele BCD pot lua valori doar între 0 și 9, pentru o sumă care depășește valoarea 9, se generează un bit de transport și rezultatul se corectează prin adunarea valorii 6. Sumatorul zecimal este construit din două sumatoare binare pe 4 biți și o logică pentru detectarea erorii de depășire și corectare. În implementarea adoptată, am ales construcția sumatorului binary pe 4 biți prin anticiparea transportului, pentru eficientizare.

Schema bloc a unui sumator zecimal este redată în figura următoare:

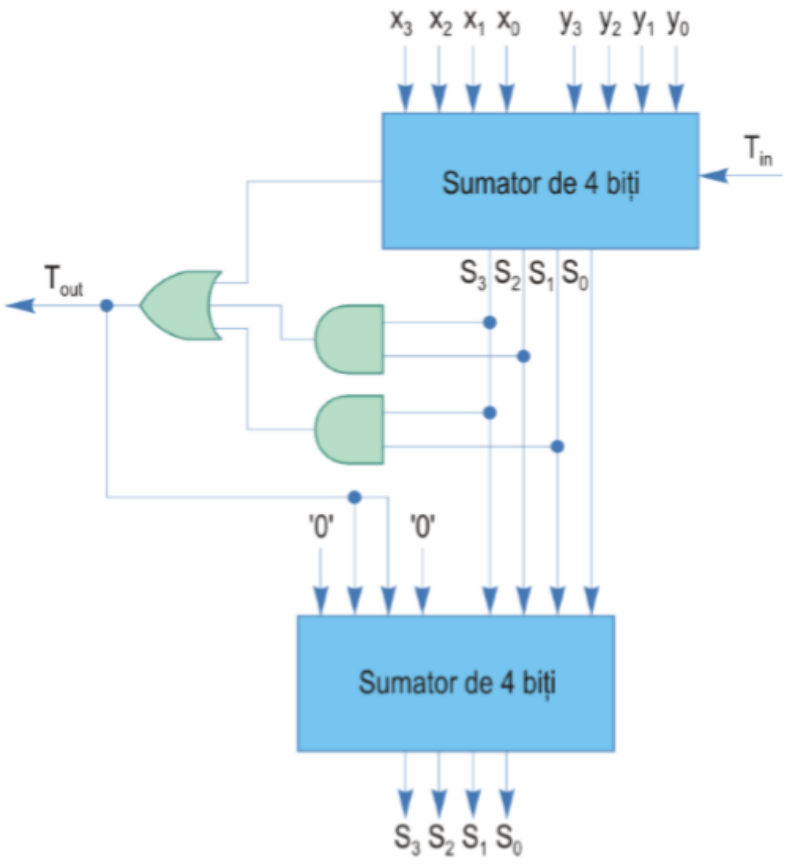


Fig.7 Schema bloc pentru un sumator zecimal [1]

X3 – X0 reprezintă o cifră zecimală reprezentată în cod BCD, Y3-Y0 reprezintă a doua cifră zecimală. S3-S0 reprezintă suma dintre cele două numere, iar Tin și Tout reprezintă transportul de intrare, respective de ieșire, care pot fi utilizați pentru cascadarea sumatoarelor pentru a realiza un sumator pentru numere cu mai multe cifre.

RTL:

$BCD(S_3S_2S_1S_{0+}) <= BCD(X_3X_2X_1X_0) + BCD(Y_3Y_2Y_1Y_0)$

$If(BCD(S_3S_2S_1S_0) >= 10)$ then

Tout <= '1'

Else

Tout <= '0'

4.1.4Sumatorul zecimal pentru numere de 4 cifre

Acest sumator ne permite să calculăm suma a două numere de câte 4 cifre, fiind construit prin interconectarea a 4 sumatoare zecimale de o cifră.

RTL:

BCD(S[15..0]) <= BCD(X[15...0]) + BCD(Y[15..0])

If(BCD(S[15..0]) >= 10000) then

Tout <= '1';

Else

Tout <= '0';

4.1.5 Registru de deplasare dreapta și încărcare paralelă cu resetare sincronă

Această componentă este definită ca un registru generic de n biți care are următoarele modalități de funcționare sincrone:

- Încărcare paralelă cu datele de pe intrarea D, dacă semnalul Load este setat pe 1 logic
- Resetare dacă semnalul Rst este setat pe 1 logic
- Shiftare spre dreapta cu 4 poziții dacă semnalul CE este setat pe 1 logic. Pe primele 4 poziții se introduc datele de pe intrarea SRI.

Abordarea metodei de shiftare cu 4 poziții este preferată datorită necesității manipulării datelor ca cifre zecimale exprimate în cod BCD.

Schema unui astfel de circuit este următoarea:

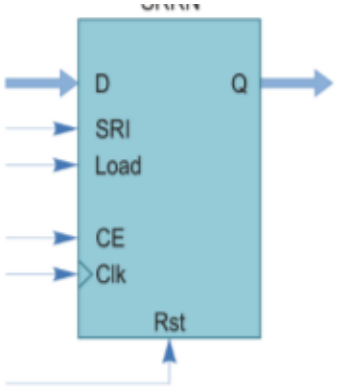


Fig.8 Simbolul registrului de deplasarea dreapta cu resetare sincronă și încărcare paralelă ([1])

RTL: IF (CLK = '1') THEN

IF(Rst = '1') THEN

Q <= 0

ELSE

IF(Load = '1') THEN

Q <= D

ELSE

IF(CE = ‘1’) THEN

Q <= SRI || Q(n-1..4)

4.1.6 Bistabil cu resetare sincronă

Această componentă este definită ca un bistabil cu date generice pe n biți. Funcționalitatea acestui tip de bistabil este dată de resetarea asincronă sau de încărcarea sincronă a datelor de pe intrare, dacă semnalul ChipEnable (CE) este setat la 1 logic.

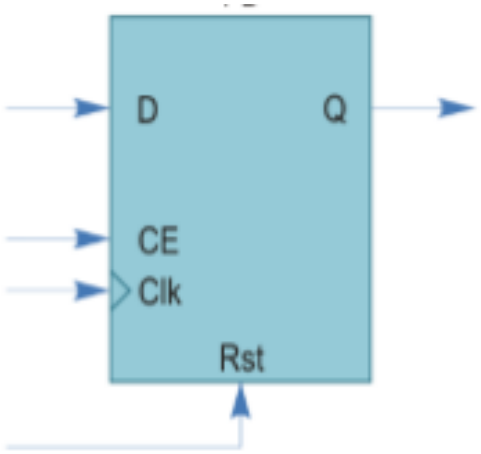


Fig. 9: Simbolul unui bistabil cu resetare sincronă ([1])

RTL: IF(CLK = ‘1’) THEN

IF(Rst = ‘1’) THEN

Q <= 0

ELSE

IF(CE = ‘1’) THEN

Q<= D

4.2 Unitatea de comandă a circuitului de înmulțire prin metoda “adunărilor repetate”

Această componentă este un automat cu stări finite care generează semnale pentru registrele și bistabilele folosite.

Ca intrări, acest FSM are un semnal de Start, care pune în funcțiune automatul, un semnal de reset pentru reînceperea funcționării, semnalul de tact și o intrare pe 4 biți pe care va intra ultima cifră a înmulțitorului, ce reprezintă chiar numărul de adunări care trebuie efectuate într-un pas.

Ca ieșiri, automatul oferă următoarele rezultate:

- LoadB – semnal de chip enable pentru bistabilul B, unde se stochează deînmulțitul
- RstB – semnal de reset pentru bistabilul B
- LoadA – semnal de încărcare paralelă pentru registrul A, care reprezintă acumulatorul și ajută la calcularea produsului parțial
- RstA – semnal de reset pentru registrul accumulator A
- LoadQ – semnal de încărcare paralelă pentru registrul Q, care inițial se va încărca cu valoarea înmulțitorului și ulterior va reține produsul parțial și rezultatul final
- RstQ – semnal de reset pentru registrul Q
- shrAQ – semnal de shiftare dreapta care se leagă atât la registrul A, cât și la registrul Q, pentru shiftarea produsului parțial
- Term – semnal ce indică terminarea operației de înmulțire

În definirea automatului, se identifică următoarele stări:

- Idle – starea inițială a automatului
- Init – starea în care se realizează inițializarea sistemului: se încarcă regiștrii și bistabilele cu valorile inițiale
- InitCifră – starea în care se inițializează contorul pentru numărul de adunări ce trebuie efectuate

- testCifră – starea în care se testează numărul de adunări care mai trebuie efectuate
- OperațiiProdParțial – starea în care se adună deînmulțitul la produsul parțial existent
- Shiftare – starea în care se shiftează conținutul regiștrilor A și Q
- testCnt – starea în care se verifică numărul de cifre rămase ale înmulțitorului
- stop – starea finală în care ajunge automatul după terminarea operației de înmulțire

Ieșirile automatului în funcție de stare sunt redate în următorul tabel:

	loadA	RstA	loadB	RstB	loadQ	RstQ	shrAQ	Term
Idle	0	1	0	1	0	1	0	0
Init	1	0	1	0	1	0	0	0
initCifră	0	0	0	0	0	0	0	0
testCifra	0	0	0	0	0	0	0	0
OperațiiProdParțial	1	0	0	0	0	0	0	0
Shiftare	0	0	0	0	0	0	1	0
testCnt	0	0	0	0	0	0	0	0
Stop	0	0	0	0	0	0	0	1

Fig.10 Ieșirile unității de comandă în funcție de stările automatului

Tranzițiile între stările automatului sunt prezentate în figura următoare:

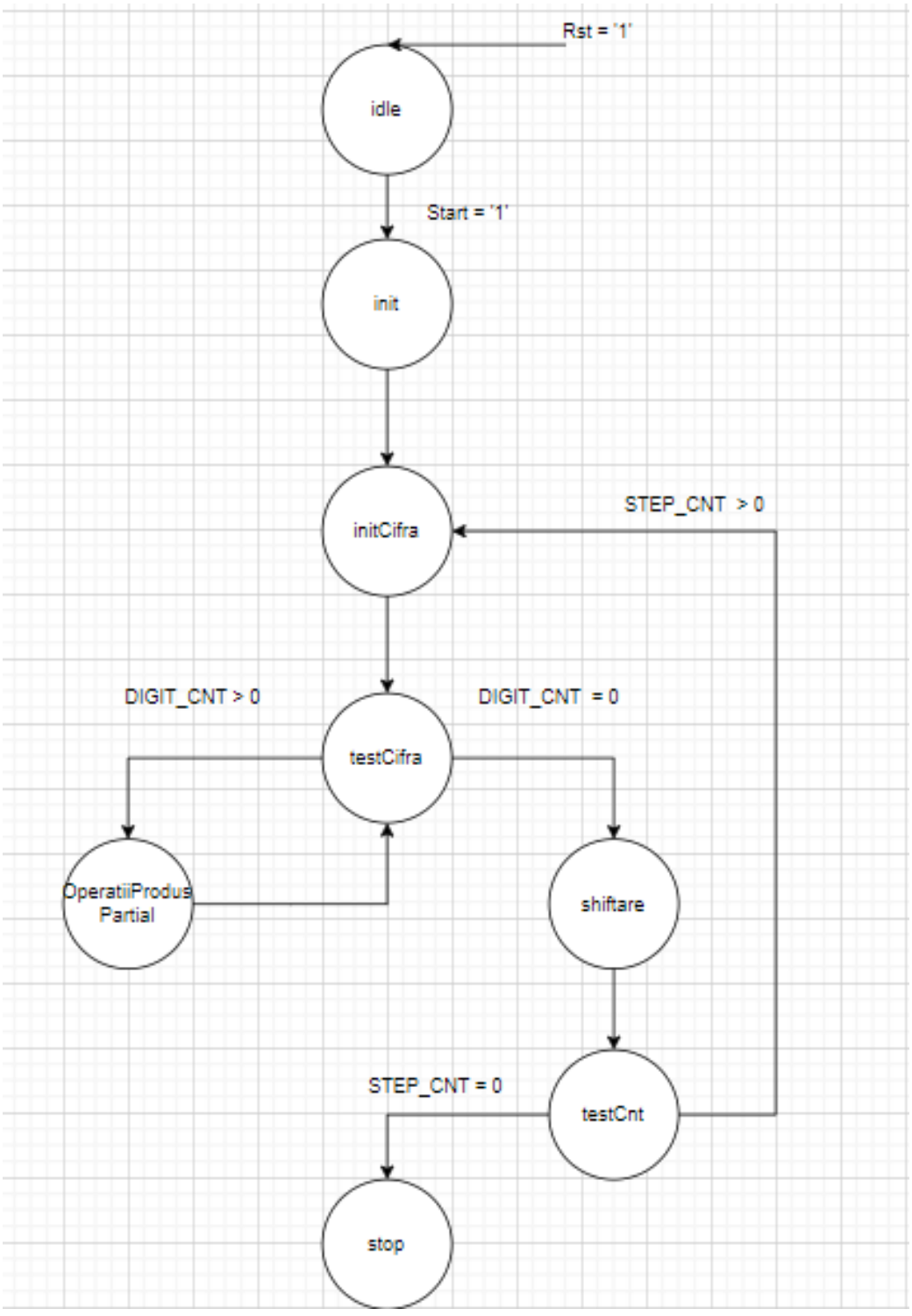


Fig.11 Diagrama de tranziții pentru FSM din metoda adunărilor repetate (https://app.diagrams.net/)

4.3 Unitatea de comandă a circuitului de înmulțire pentru metoda "celor 9 multiplii ai deînmulțitului"

Această componentă este un automat cu stări finite care generează semnale pentru registrele și bistabilele utilizate.

Ca intrări, acest FSM are un semnal de Start, care pune în funcțiune automatul, un semnal de reset pentru reînceperea funcționării și semnalul de tact.

Ca ieșiri, automatul oferă următoarele rezultate:

- LoadA – semnal de încărcare paralelă pentru registrul A, care reprezintă acumulatorul și ajută la calcularea produsului parțial
- RstA – semnal de reset pentru registrul accumulator A
- LoadQ – semnal de încărcare paralelă pentru registrul Q, care inițial se va încărca cu valoarea înmulțitorului și ulterior va reține produsul parțial și rezultatul final
- RstQ – semnal de reset pentru registrul Q
- shrAQ – semnal de shiftare dreapta care se leagă atât la registrul A, cât și la registrul Q, pentru shiftarea produsului parțial
- LoadX – un semnal de 9 biți, fiecare bit corespunde semnalului de încărcare paralelă sinronă pentru un registru care memorează un multiplu al deînmulțitului
- RstX – semnal de 9 biți, fiecare bit corespunde semnalului de reset sincron pentru un registru care memorează un multiplu al deînmulțitului
- MuxInit – este un semnal de selecție pentru un multiplexor 2:1 care arată un termen pentru sumatorul zecimal. În partea de inițializare a multiplilor deînmulțitului, acest semnal este setat la 1 logic (pe multiplexor va fi selectat conținutului registrului ce salvează deînmulțitul), urmând ca în partea de computație a produselor parțiale și a rezultatului final, acest semnal va fi setat la 0 logic. (multiplexorul selectează registrul ce corespunde ultimei cifre a înmulțitorului).
- Term – semnal ce indică terminarea operației de înmulțire

În definirea automatului, se identifică următoarele stări:

- Idle – starea inițială a automatului. În această stare se face și inițializarea registrului 1X care stochează deînmulțitul și resetarea registrului Acumulator
- Init – starea în care se realizează inițializarea sistemului: se încarcă registrul Q și în registrul Acumulator se salvează valoarea lui X.
- Init2X – în această stare sumatorul zecimal va avea la ieșire dublul deînmulțitului și această valoare se va salva în registrul 2X
- Init3X - în această stare sumatorul zecimal va avea la ieșire triplul deînmulțitului și această valoare se va salva în registrul 3X
- Init4X - în această stare sumatorul zecimal va avea la ieșire valoarea de 4 ori mai mare a deînmulțitului și această valoare se va salva în registrul 4X
- Init5X - în această stare sumatorul zecimal va avea la ieșire valoarea de 5 ori mai mare a deînmulțitului și această valoare se va salva în registrul 5X
- Init6X - în această stare sumatorul zecimal va avea la ieșire valoarea de 4 ori mai mare a deînmulțitului și această valoare se va salva în registrul 6X
- Init7X - în această stare sumatorul zecimal va avea la ieșire valoarea de 4 ori mai mare a deînmulțitului și această valoare se va salva în registrul 7X
- Init8X - în această stare sumatorul zecimal va avea la ieșire valoarea de 4 ori mai mare a deînmulțitului și această valoare se va salva în registrul 8X
- Init9X - în această stare sumatorul zecimal va avea la ieșire valoarea de 4 ori mai mare a deînmulțitului și această valoare se va salva în registrul 9X
- Adun – în această stare se adună valoarea din registrul corespunzător ultimii cifre a înmulțitorului, adică cel care stochează valoarea (ultima cifră curentă a înmulțitorului stocată în registrul Q) x (deînmulțit) cu valoarea din registrul acumulator, rezultatul este salvat în registrul

acumulator, este de asemenea salvat și transportul în poziția cea mai semnificativă a registrului acumulator.

- Shiftez – starea în care se shiftează conținutul regiștrilor A și Q
- testC – starea în care se verifică numărul de cifre rămase ale înmulțitorului
- stop – starea finală în care ajunge automatul după terminarea operației de înmulțire

Ieșirile automatului în funcție de stare sunt redate în următorul tabel:

	LoadA	RstA	LoadQ	RstQ	shrAQ	MuxInit	LoadX	RstX	Term
Idle	0	1	0	1	0	0	000000001	111111110	0
Init	1	0	1	0	0	1	000000000	000000000	0
Init2x	1	0	0	0	0	1	000000010	000000000	0
Init3x	1	0	0	0	0	1	000000100	000000000	0
Init4x	1	0	0	0	0	1	000001000	000000000	0
Init5x	1	0	0	0	0	1	000010000	000000000	0
Init6x	1	0	0	0	0	1	000100000	000000000	0
Init7x	1	0	0	0	0	1	001000000	000000000	0
Init8x	1	0	0	0	0	1	010000000	000000000	0
Init9x	1	0	0	0	0	1	100000000	000000000	0
Adun	1	0	0	0	0	0	000000000	000000000	0
Shiftez	0	0	0	0	1	0	000000000	000000000	0
testC	0	0	0	0	0	0	000000000	000000000	0
stop	0	0	0	0	0	0	000000000	000000000	1

Fig.12: Ieșirile unității de comandă în funcție de stările automatului

Tranzițiile între stările automatului sunt prezentate în figura următoare:

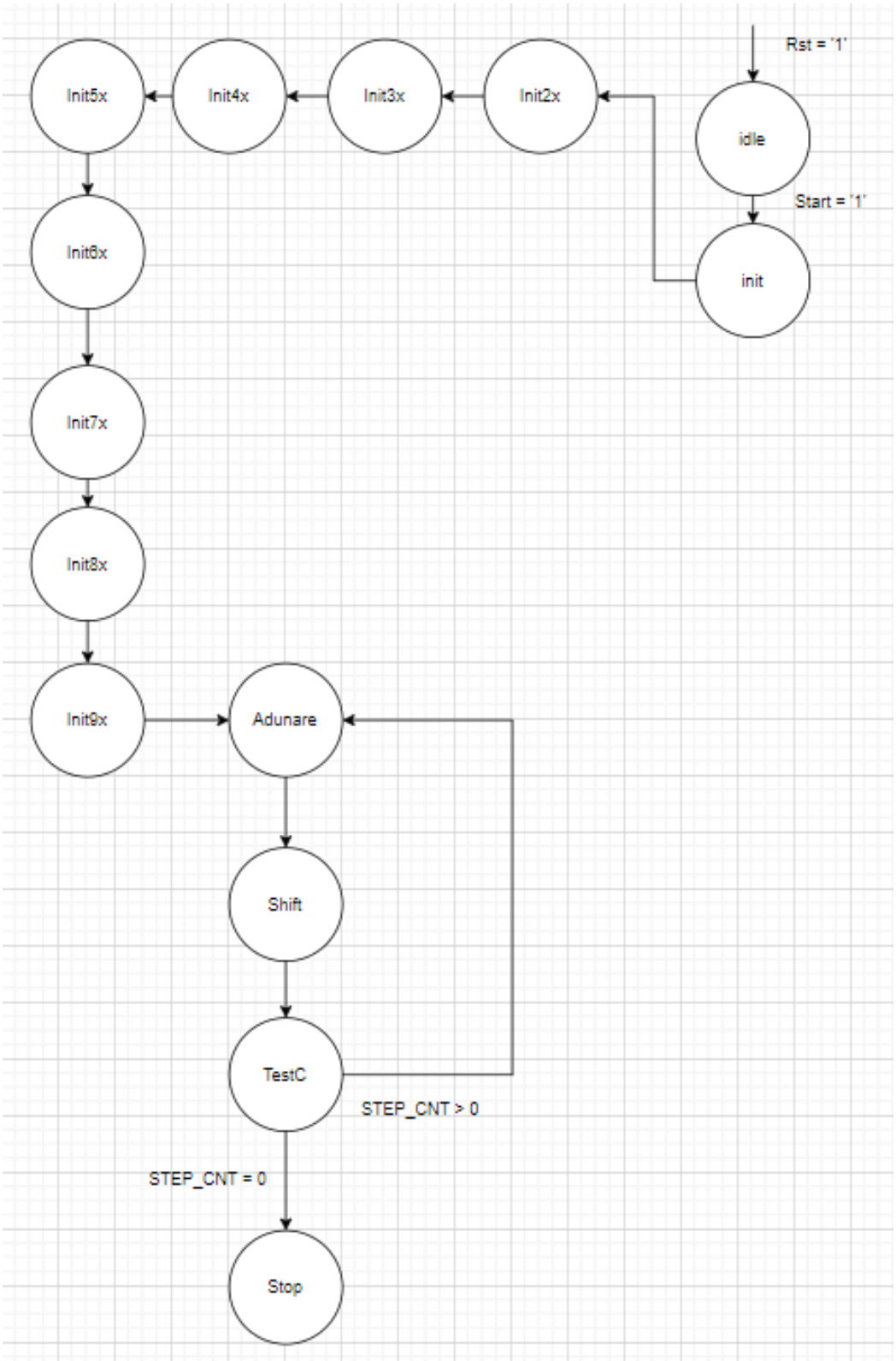
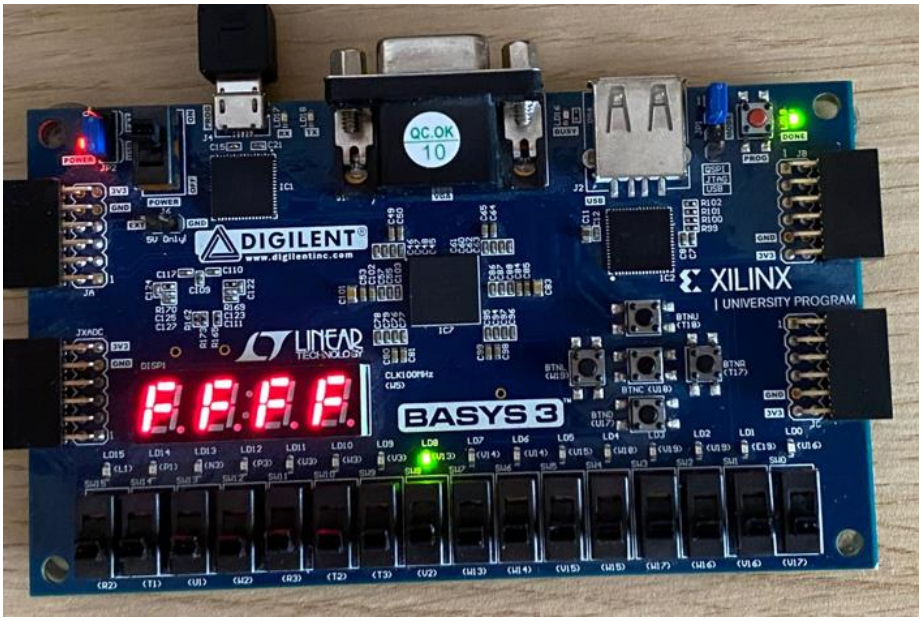
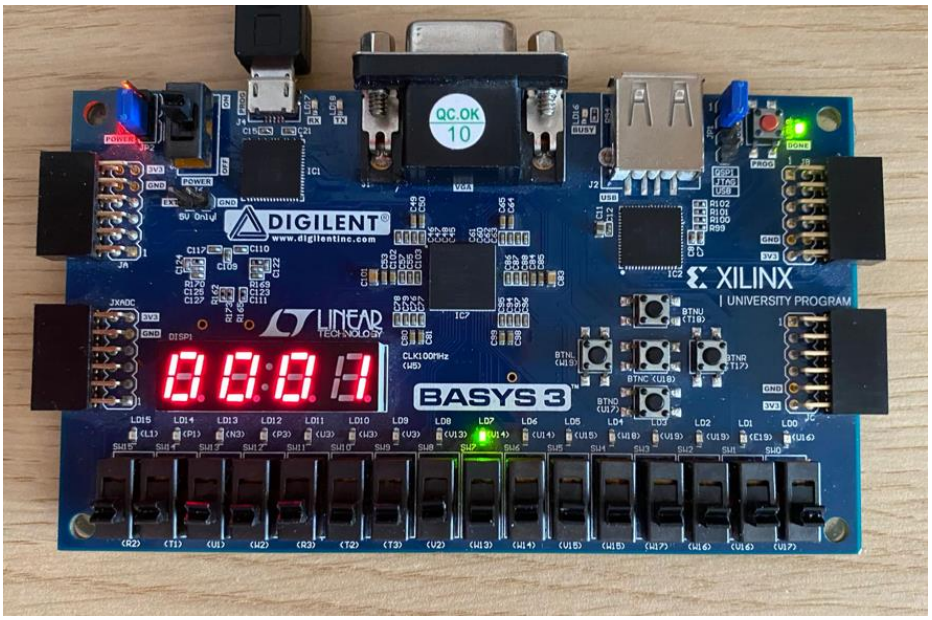


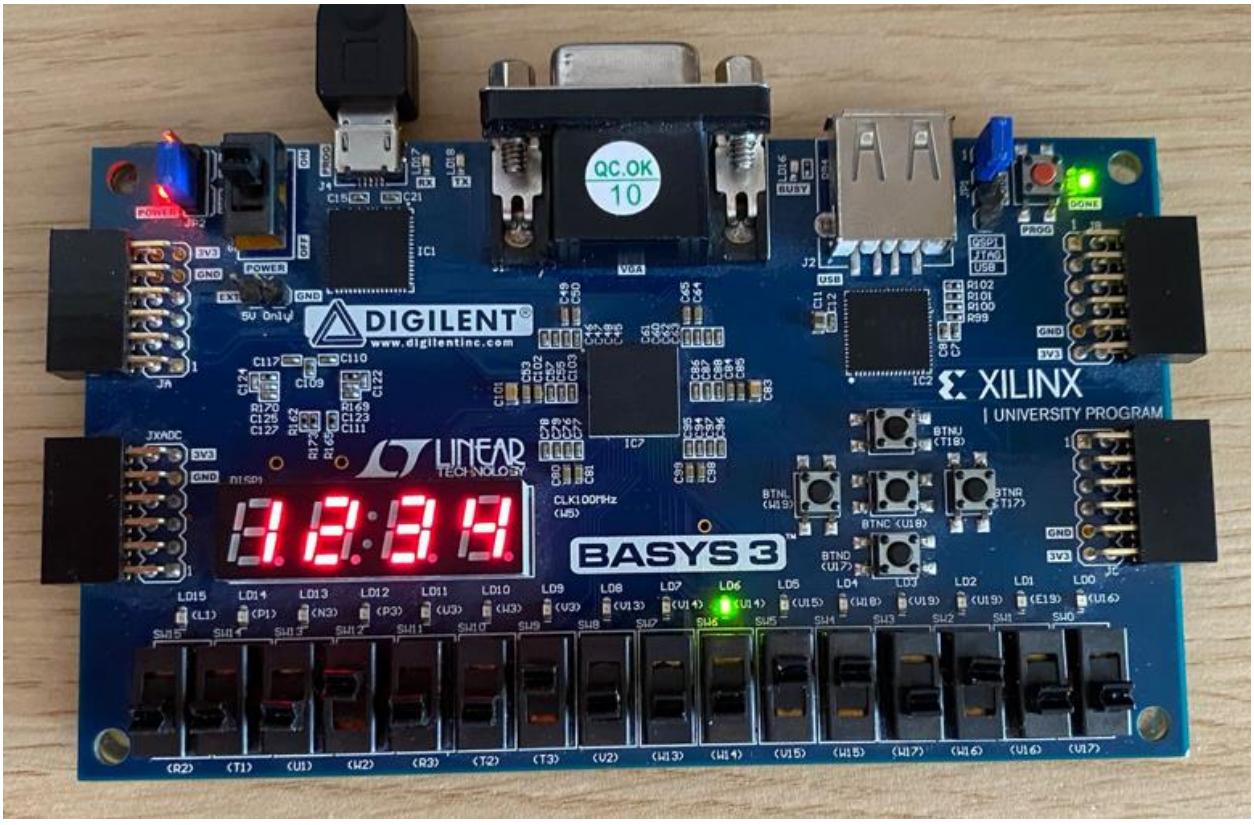
Fig. 13 Diagrama de tranziții pentru FSM din metoda celor 9 multiplii (https://app.diagrams.net/)

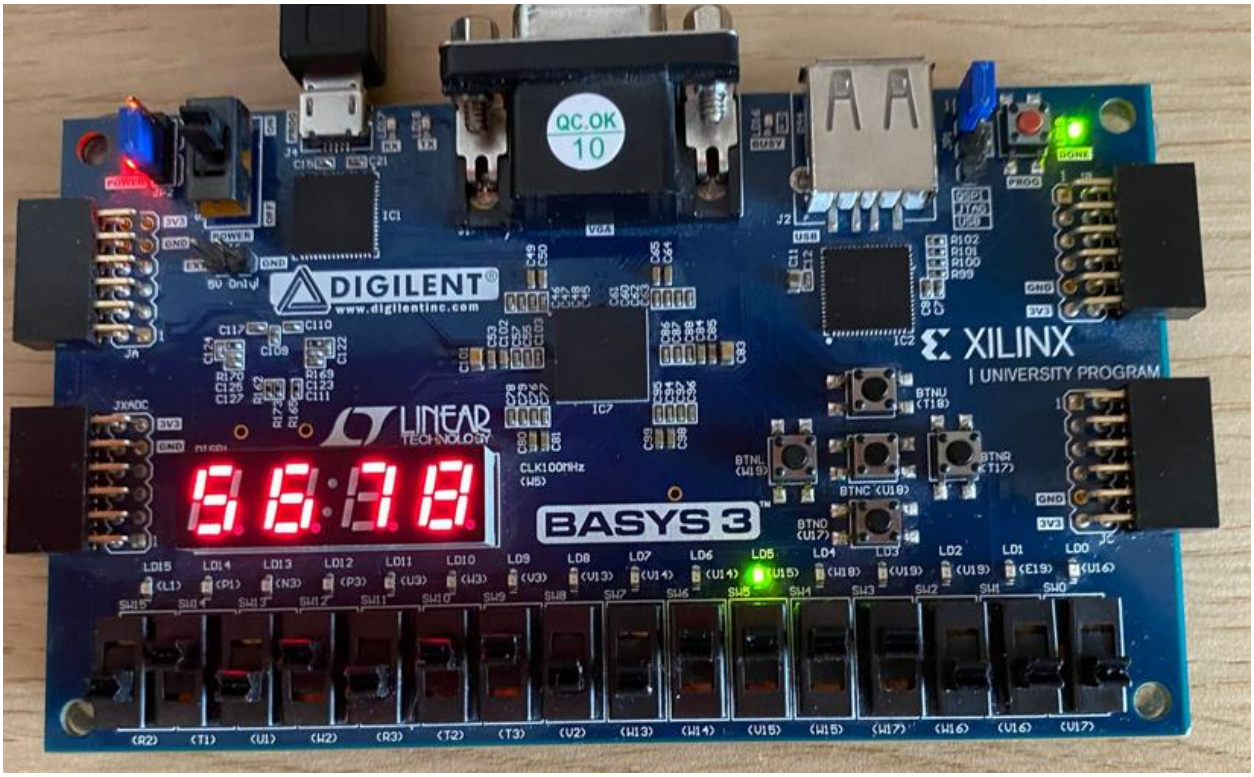
5. Rezultate experimentale

Metoda 1:

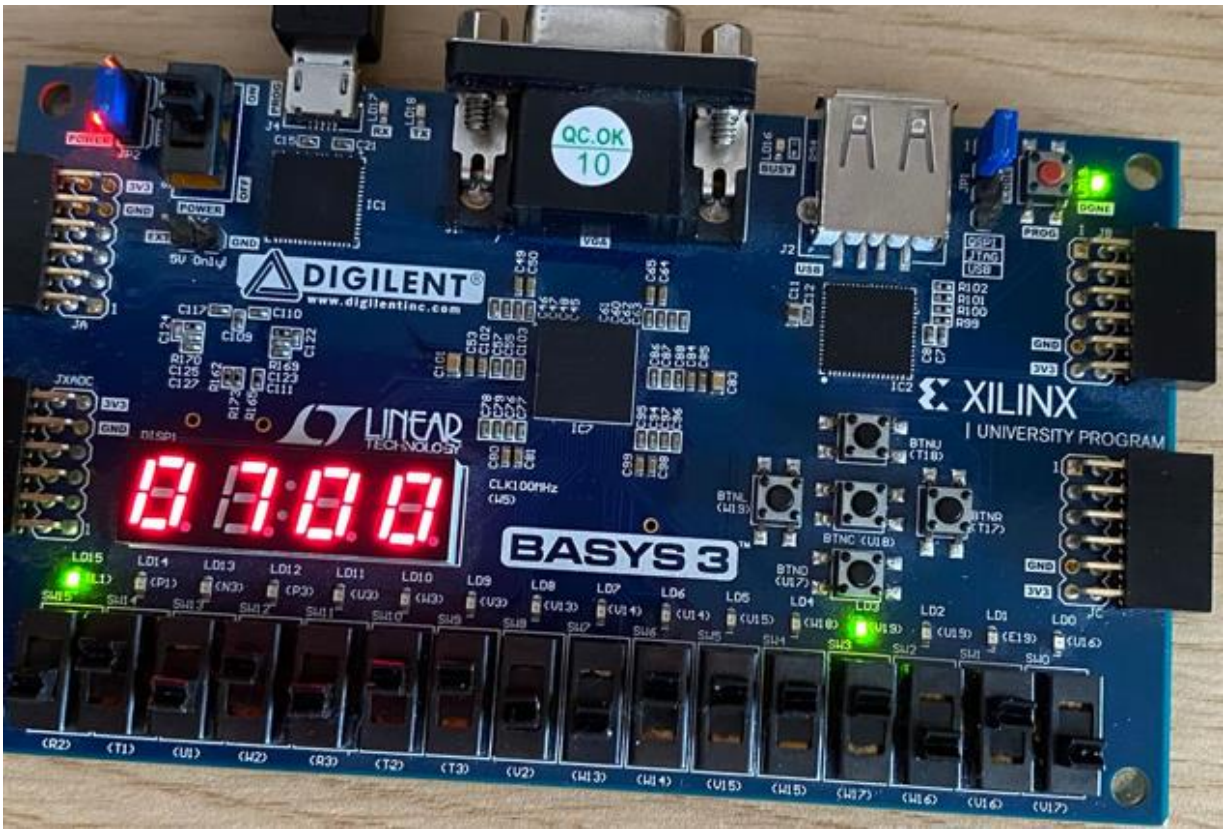


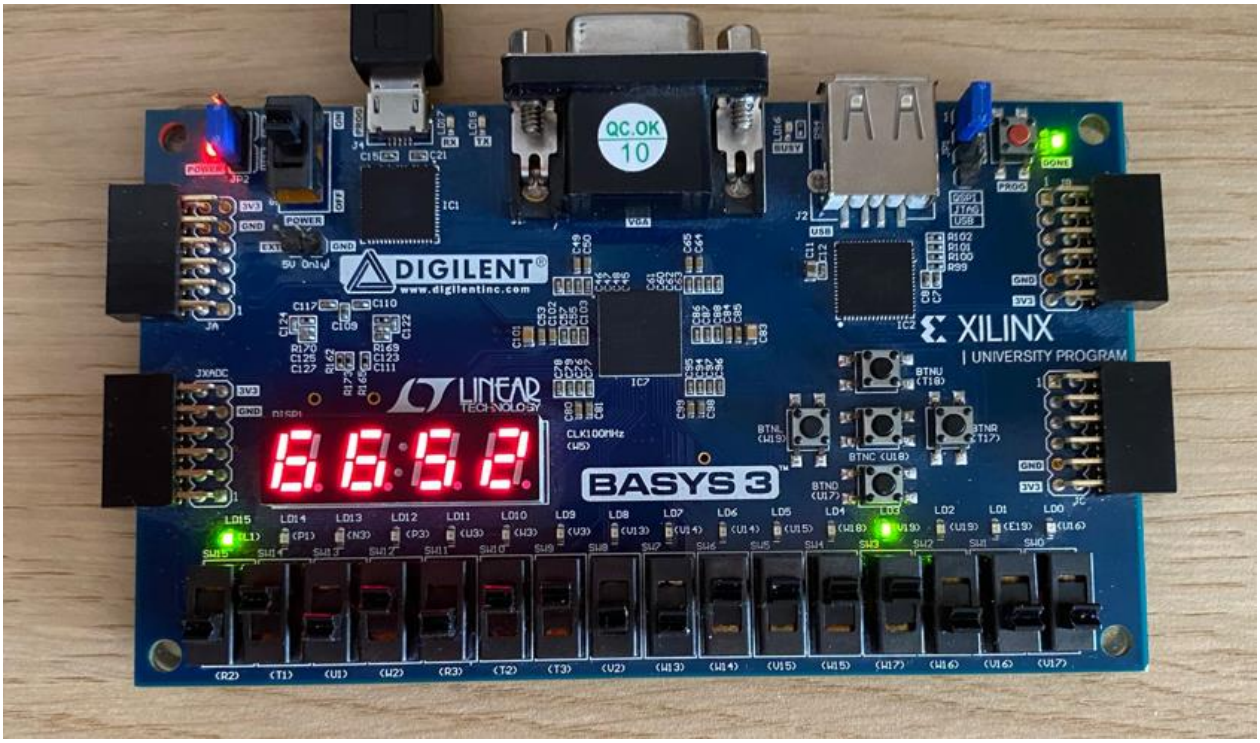



QC.OK
10
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
(W5)

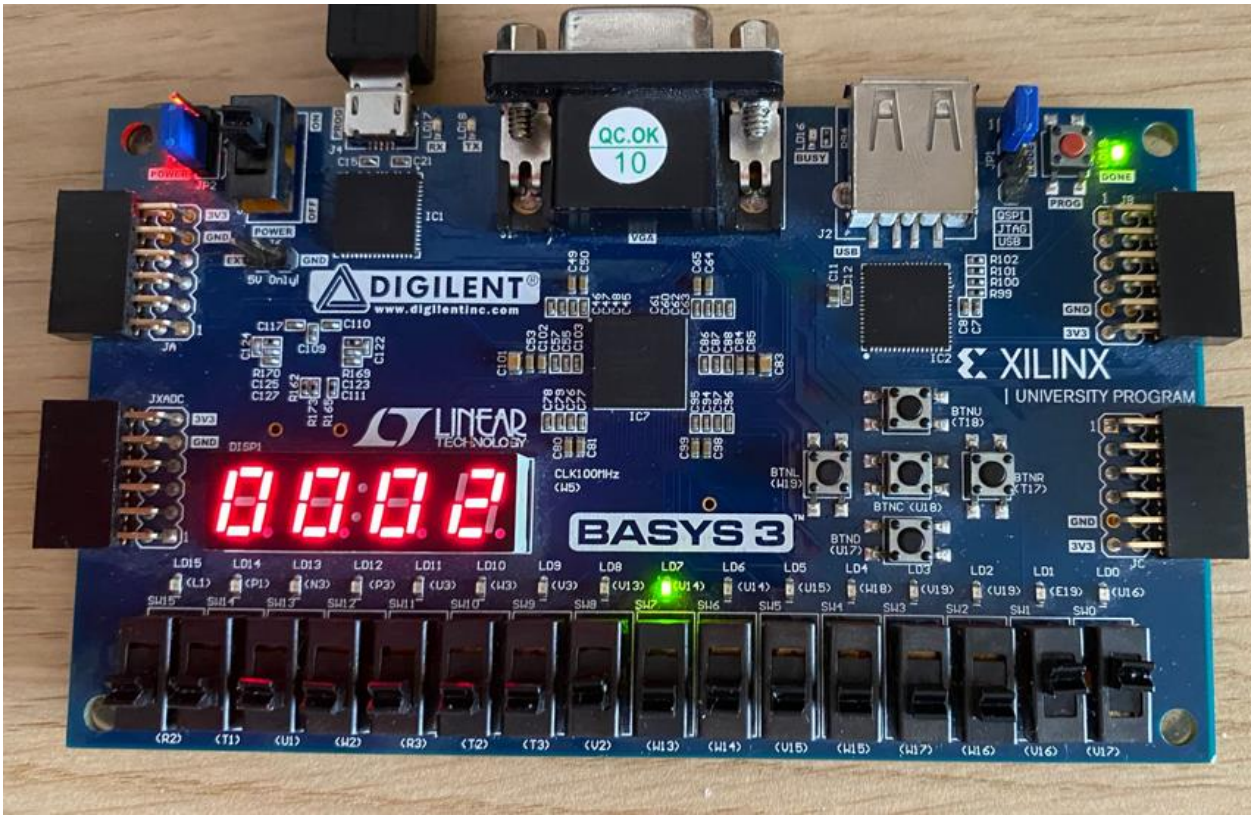

QC.OK
10
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
(W5)

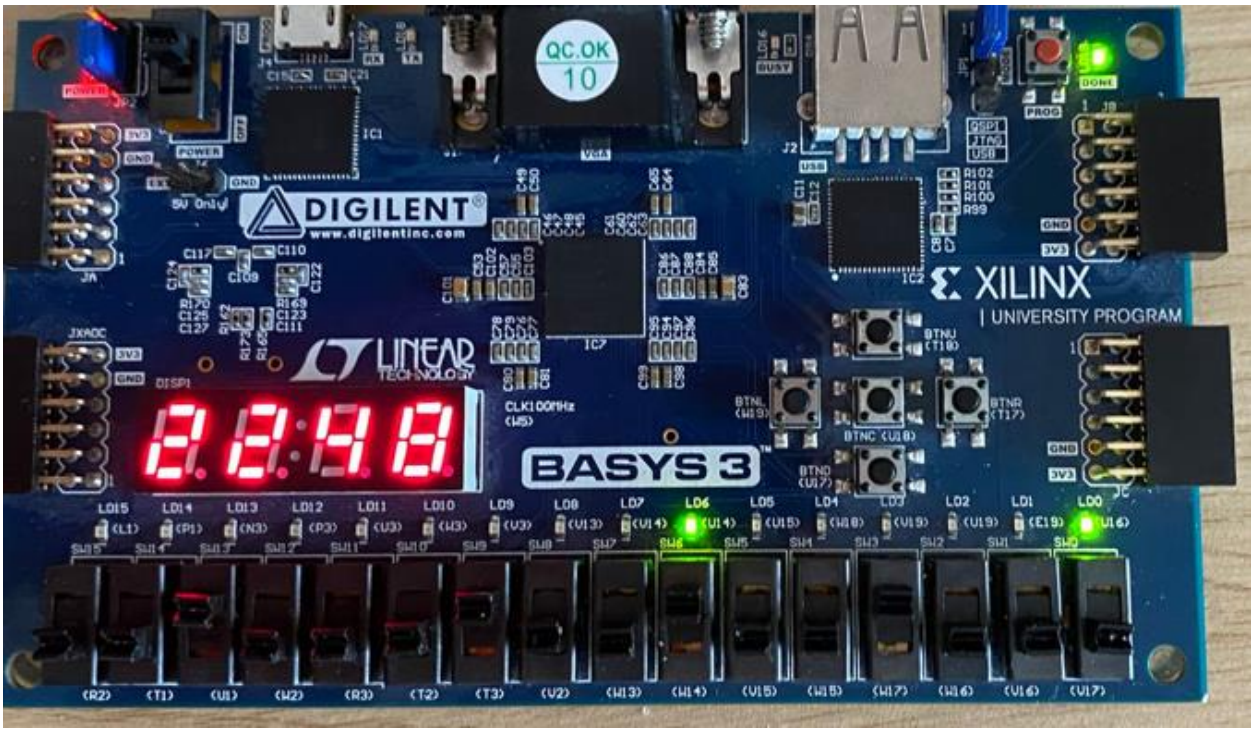
Rezultate metoda 1:

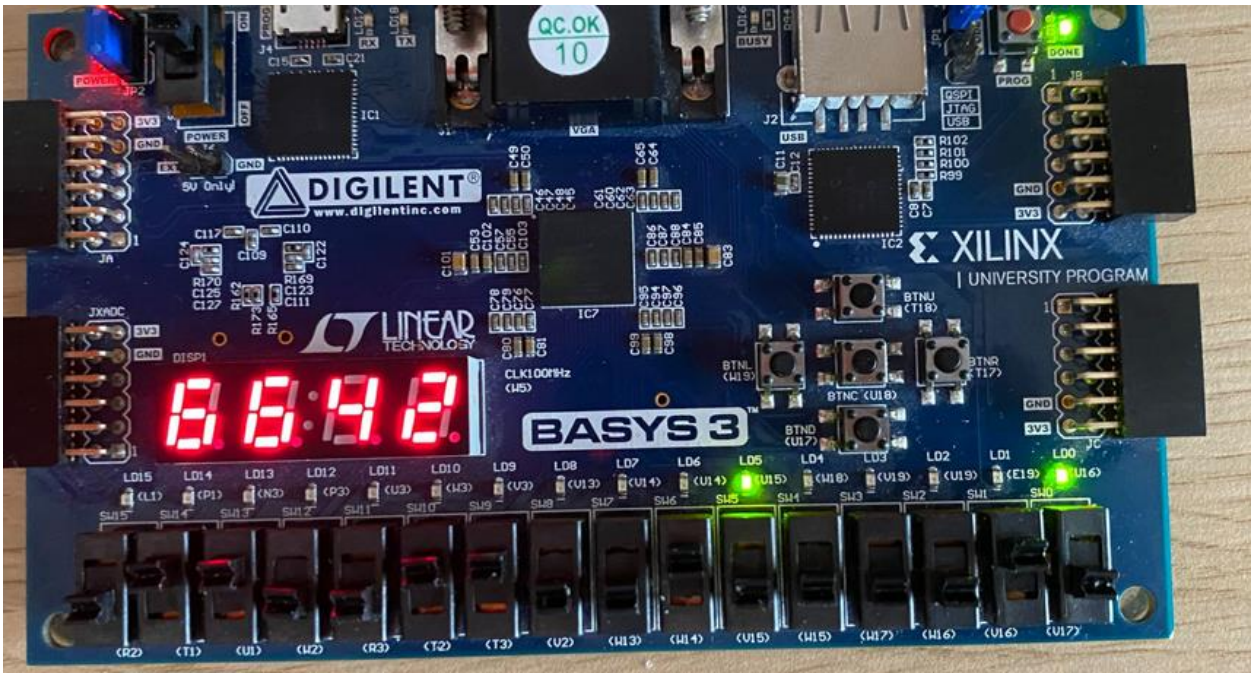




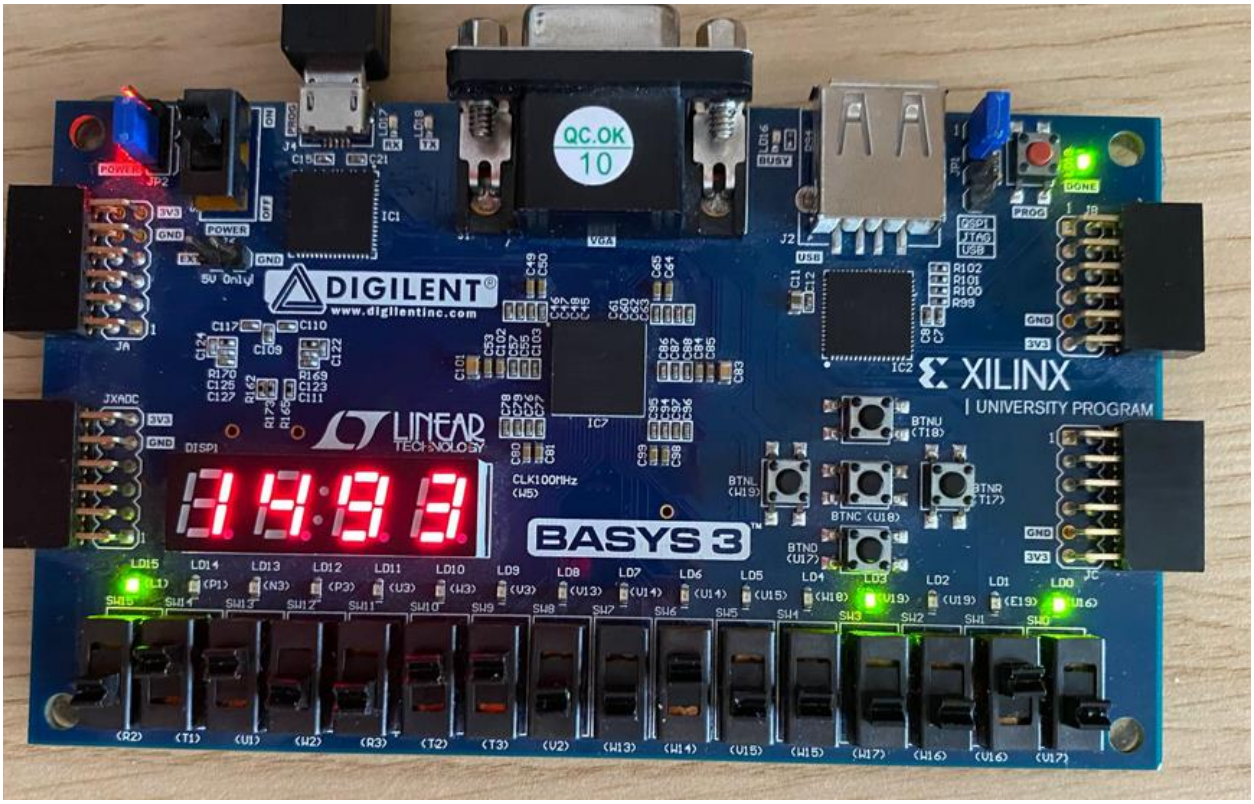
Metoda 2:

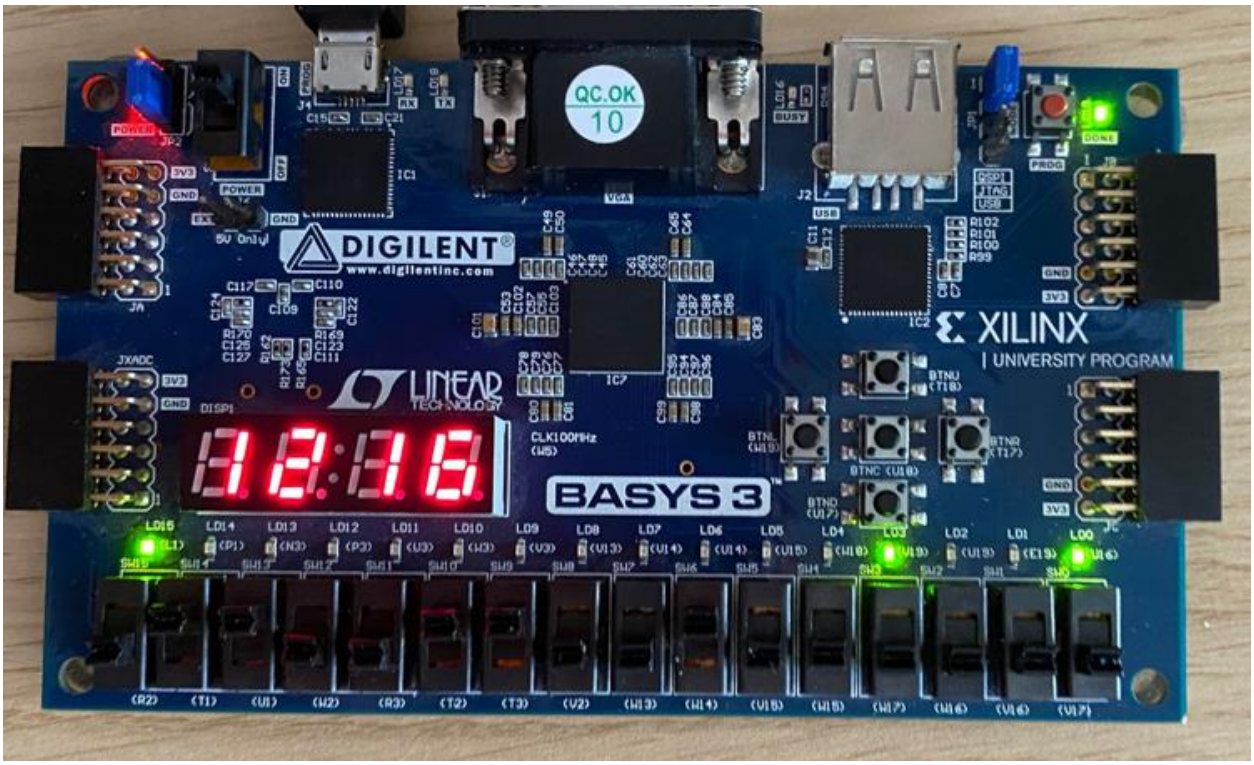

QC.OK
10
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
(R2) (T1) (U1) (W2) (R3) (T2) (T3) (V2) (W13) (W14) (V15) (W15) (W17) (W16) (V16) (V17)


QC.OK
10
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
(R2) (T1) (U1) (W2) (R3) (T2) (T3) (V2) (W13) (W14) (V15) (W15) (W17) (W16) (V16) (V17)



Rezultate metoda 2:




QC.OK
10
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR
TECHNOLOGY
BASYS 3
CLK100MHz
(W5)
BTNU
(T18)
BTNC (U18)
BTNL
(W19)
BTNR
(T17)
BTND
(U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
(R2) (T1) (U1) (W2) (R3) (T2) (T3) (V2) (W13) (W14) (V15) (W15) (W17) (W16) (V16) (V17)

6. Concluzii

Acest proiect a avut ca obiectiv proiectarea și implementarea a doua circuite distincte pentru înmulțirea zecimală a numerelor de patru cifre, folosind metodele "Adunărilor repetate" și "Celor 9 multiplii ai deînmulțitului". Utilizând limbajul de descriere hardware VHDL și mediul de dezvoltare Vivado Design Suite, fiecare metodă a fost implementată prin intermediul unei descrieri structurale, inclusiv o unitate de comandă specifică și regiștri speciali, precum accumulatori și regiștri pentru stocarea deînmulțitului și înmulțitorului.

De asemenea, a fost dezvoltat un sistem pe placa de dezvoltare FPGA Basys3, care integrează metodele. Utilizatorul are posibilitatea de a alege metoda dorită, de a introduce numere de patru cifre și de a observa rezultatele obținute. Proiectul a inclus și o unitate de comandă separată pentru gestionarea resurselor plăcii, permițând introducerea datelor, prin intermediul switch-urilor plăcii.

Compararea metodelor implementate s-a realizat în domeniul costului, consumului de timp și energie, având ca surse de date sinteza circuitelor. Avantajul evident al proiectului constă în dezvoltarea abilităților de proiectare a unui sistem hardware complex, identificarea și construirea unităților de comandă esențiale, precum și în aprofundarea cunoștințelor legate de algoritmi hardware pentru înmulțire. Proiectul aduce un plus în contextul real, având aplicații extinse în domeniile economic, financiar și tehnologic, contribuind la evitarea erorilor de calcul asociate algoritmilor care operează cu numere în baza de numerație 2.

În concluzie, acest proiect a rezolvat eficient problema înmulțirii zecimale a numerelor de patru cifre.. Avantajele includ abilitățile dobândite în proiectare și implementare, cu aplicabilitate semnificativă în domenii diverse. Dezavantajele pot include eventualele erori introduse în sistem și necesitatea unor optimizări ulterioare. Proiectul sugerează dezvoltări viitoare, precum implementarea unor metode mai eficiente sau extinderea funcționalității pentru a acoperi și operații de împărțire zecimală, consolidând astfel abordarea matematică a proiectului.

Bibliografie

[**1]** Baruch Z.F., "Circuite aritmetice combinaționale" – Îndrumător de laborator 2019/2020 (link: http://users.utcluj.ro/~baruch/ssc/labor/Aritm-Combinationala.pdf)

[**2]** Baruch Z.F., "Circuite aritmetice secvențiale" – Îndrumător de laborator 2019/2020 (link: http://users.utcluj.ro/~baruch/ssc/labor/Aritm-Secventiala.pdf)

[3] Baruch Z.F., "Structure of Computer Systems with Applications", U.T. Press 2003

[4] Creț O., Văcariu L., Nețin A., "Analiza și sinteza dispozitivelor numerice: Îndrumător de laborator", U.T. Press 2005

[5] G. Jaberipur, A. Kaivani, "Binary-coded decimal digit multipliers" în IET Comput. Digit. Tech., Vol. 1, No. 4, July 2007 (link: https://pdfs.semanticscholar.org/8719/ed34825f114e30044cdc676d26997cb80993.pdf)