

Facultatea: Automatica si Calculatoare

Specializare: Calculatoare si Tehnologia Informatiei

Disciplina: Structura Sistemelor de Calcul

Unitate de calcul pentru extragerea rădăcinii pătrate

**Îndrumător**

Prof. Butiri Alexandru

**Student**

Cozma Victoria

Grupa 30225

4 ianuarie 2021

**Cuprins**

1. Rezumat
2. Introducere
3. Fundamentare teoretică
4. Proiectare şi implementare
5. Rezultate experimentale
6. Concluzii
7. Bibliografie
8. **Rezumat**

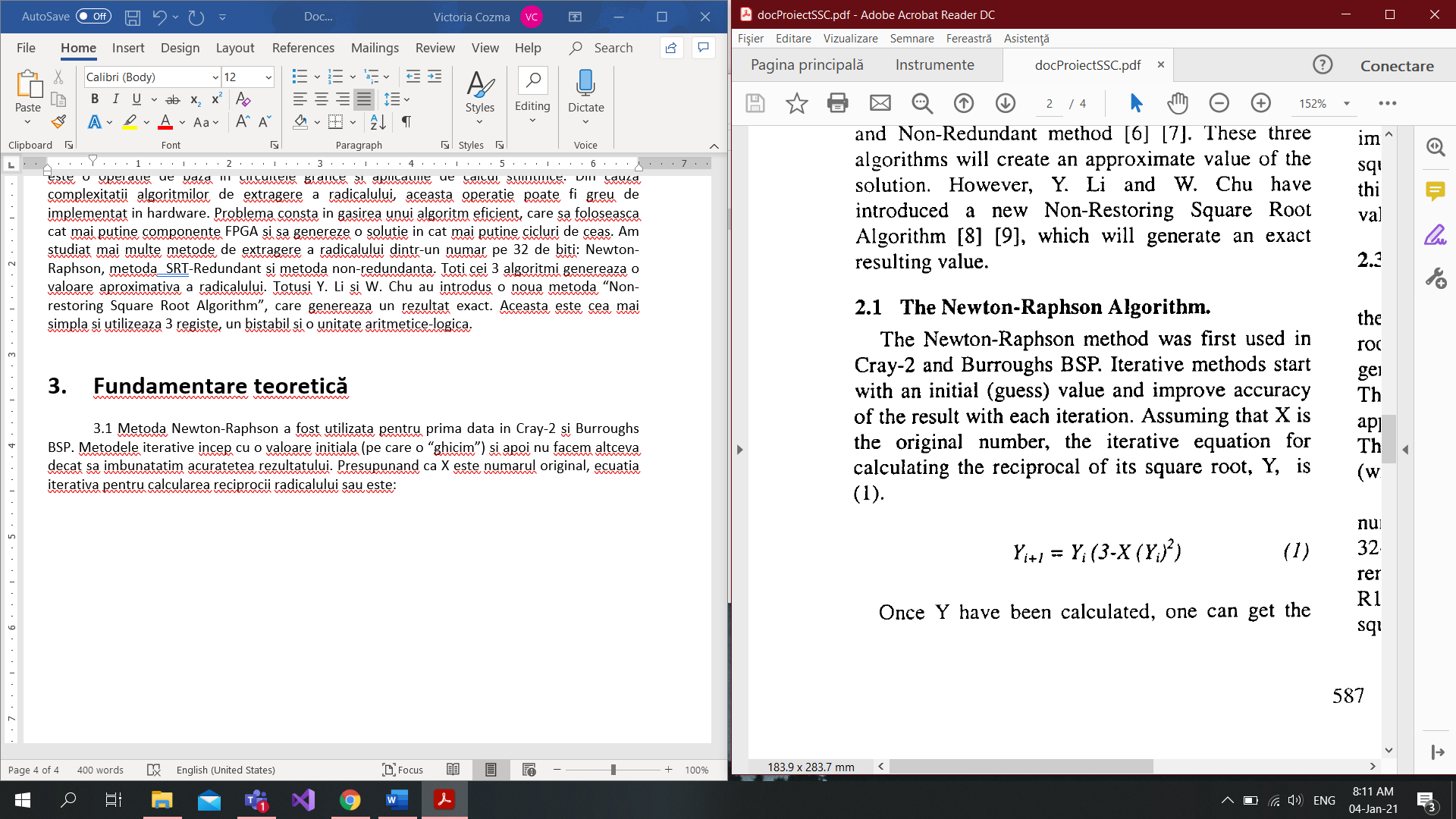
Scopul lucrării de faţă este să prezinte întreg procesul de extragere a radacinii patrate dintr-un numar intreg pe 32 biti. S-au analizat iniţial datele problemei pentru a se înţelege conceptele teoretice: cum sunt reprezentate numerele şi cum functioneaza algoritmul de extragere a radicalului. Ulterior, s-a studiat placa Basys 3 pentru afisaj. După parcurgerea în detaliu a acestor paşi, s-a trecut la rezolvarea problemei prin implementarea algoritmului. Acesta a fost “despicat” pe pasi. Pentru fiecare pas al algoritmului am creat cate o noua componenta si am utilizat modul structural de conectare a componentelor. Astfel, am reprezentat in cod VHDL urmatoarele componente: unitatea ALU, registru, registru de shiftare stanga cu 2 unitai, registru de shiftare dreapta cu 2 unitati, bistabil D. De asemenea, am construit si implementat automatul de stari pentru functionarea eficienta a algoritmului. Importanţa acestui proiect a fost de a înţelege diferiti algoritmi de extragere a radacinii patrate si de a-l implementa pe cel mai scurt si eficient.

1. **Introducere**

Tema proiectului este una actuala, intrucat extragerea radacinii patrate dintr-un numar este o operatie de baza in circuitele grafice si aplicatiile de calcul stiintifice. Din cauza complexitatii algoritmilor de extragere a radicalului, aceasta operatie poate fi greu de implementat in hardware. Problema consta in gasirea unui algoritm eficient, care sa foloseasca cat mai putine componente FPGA si sa genereze o solutie in cat mai putine cicluri de ceas. Am studiat mai multe metode de extragere a radicalului dintr-un numar pe 32 de biti: Newton-Raphson, metoda SRT-Redundant si metoda non-redundanta. Toti cei 3 algoritmi genereaza o valoare aproximativa a radicalului. Totusi Y. Li si W. Chu au introdus o noua metoda “Non-restoring Square Root Algorithm”, care genereaza un rezultat exact. Aceasta este cea mai simpla si utilizeaza 3 registe, un bistabil si o unitate aritmetice-logica.

1. **Fundamentare teoretică**
   1. Metoda Newton-Raphson

Metoda a fost utilizata pentru prima data in Cray-2 si Burroughs BSP. Metodele iterative incep cu o valoare initiala (pe care o “ghicim”) si apoi nu facem altceva decat sa imbunatatim acuratetea rezultatului. Presupunand ca X este numarul original, ecuatia iterativa pentru calcularea reciprocii radicalului sau este:



Odata ce am aflat Y, putem calcula si radicalul prin inmultirea cu X. Acest algoritm converge cvadric.

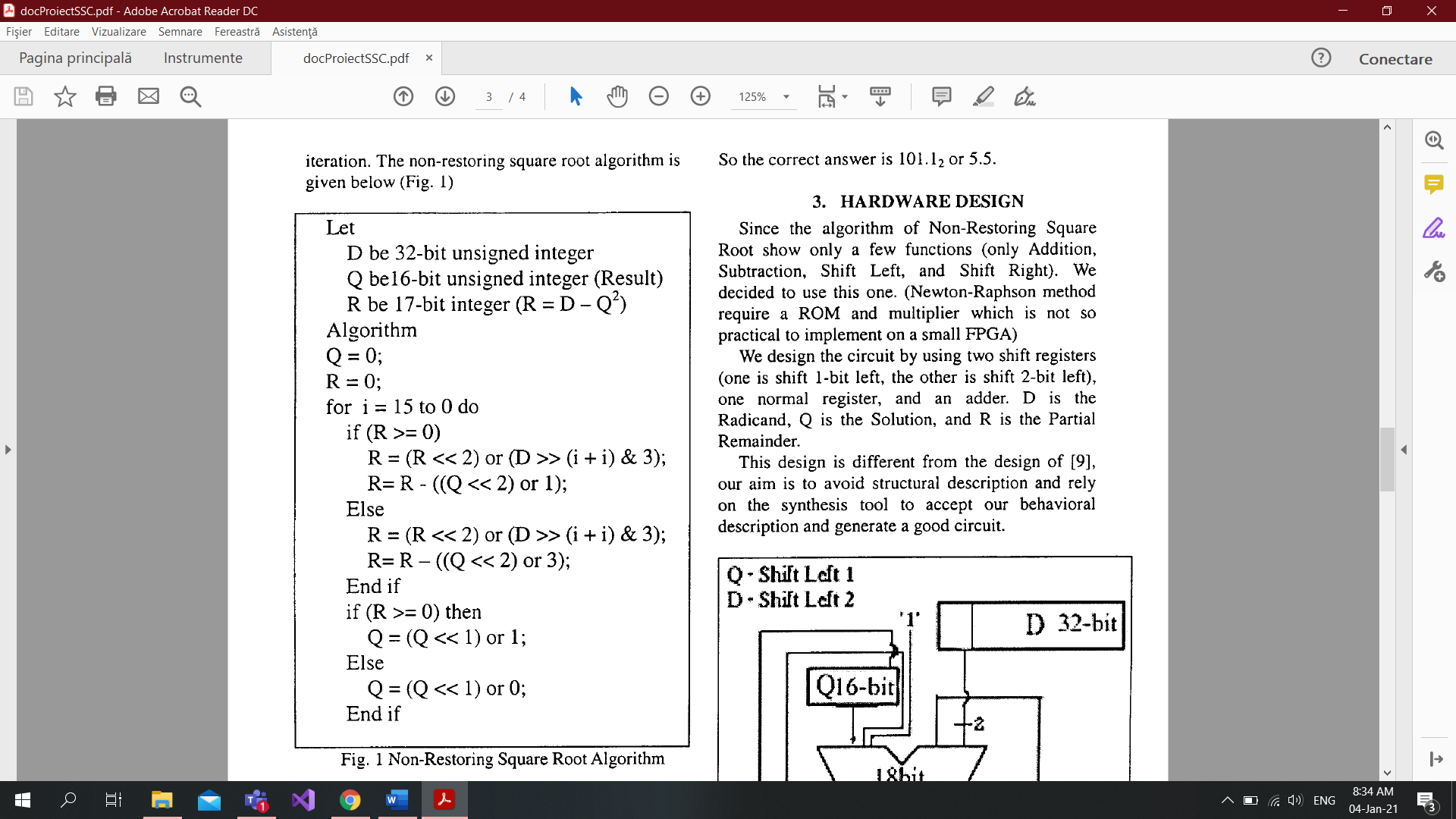
Algoritmul are nevoie de un senerator de seminte (seed) pentru generarea Y0(valoarea initiala aleatoare a rezultatului), utilizand oe memorie ROM, de exemplu. Numarul de intrari in memoria ROM si lungimea fiecarei intrari influenteaza precizia operatiei si numarul de iteratii. Pentru a obtine un rezultat pe 24 biti, memoria ROM trebuie sa aiba 64 cuvinte x 6 biti. In acest caz, vom avea nevoie doar de 2 iteratii pentru o precizie de 24 biti. La fiecare iteratie se vor realiza operatii de inmultire, adunare si scadere. Pentru a mari viteza de efectuare a inmultirii, se poate implementa metoda arborelui Wallace, pentru a produce un produs partial si a propaga transportul. Din cauza ca inmultitorul necesita un numar mare de porti logice, algoritmul nu este eficient in implementarea sa pe placa FPGA. De asemenea, este complicat sa extragem restul radacinii patrate.

3.2. Algoritmul Radix-2 SRT-Redundant si algoritmul Non-redundant

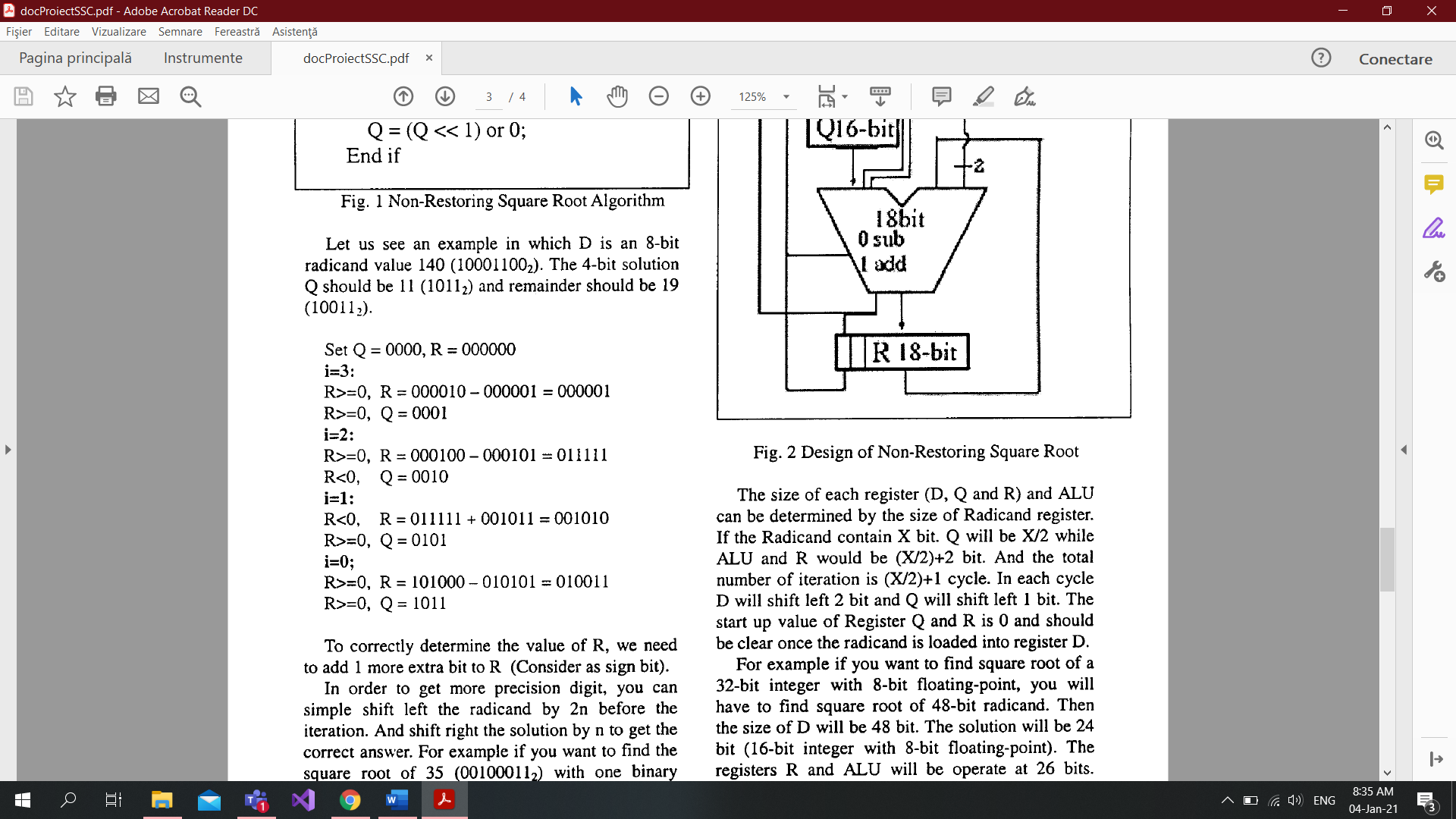
Aceste 2 metode de extragere a radicalului sunt similare. Ambele folosesc relatia de recursivitate. La fiecare iteratie, vom avea o shiftare si o adunare. Determinarea functiei recursive este foarte complicata, in special pentru algoritmul Radix-2. De asemenea, implementare lor pe FPGA nu adminte generarea unui rezultat per ciclu de ceas. Mentionam si ca aceste 2 metode pot genera erori la calcularea celui mai ptin semnificativ bit din rezultat.

3.3. Algoritmul fara restabilire (Non-restoring)

Acest algoritm foloseste reprezentarea in complement fata de 2 a rezultatului. La fiecare iteratie, algoritmul poate genera valoarea exacta (chiar si a celui mai putin semnificativ bit). Nu sunt necesare calcule complicate, ca in cazul algoritmilor precedenti. De asemenea, restul exact se genereaza instant(cu o mica corectie daca e negativ). Sa presupunem ca numarul initial este un intreg fara semn pe 32 de biti (notat cu D31..0). Radacina patrata a numarului o vom nota cu Q15..0. R este restul ( R = D – Q^2), care va fi notat cu R16..0. Pseudocodul pentru algoritmul dat este prezentat mai jos:

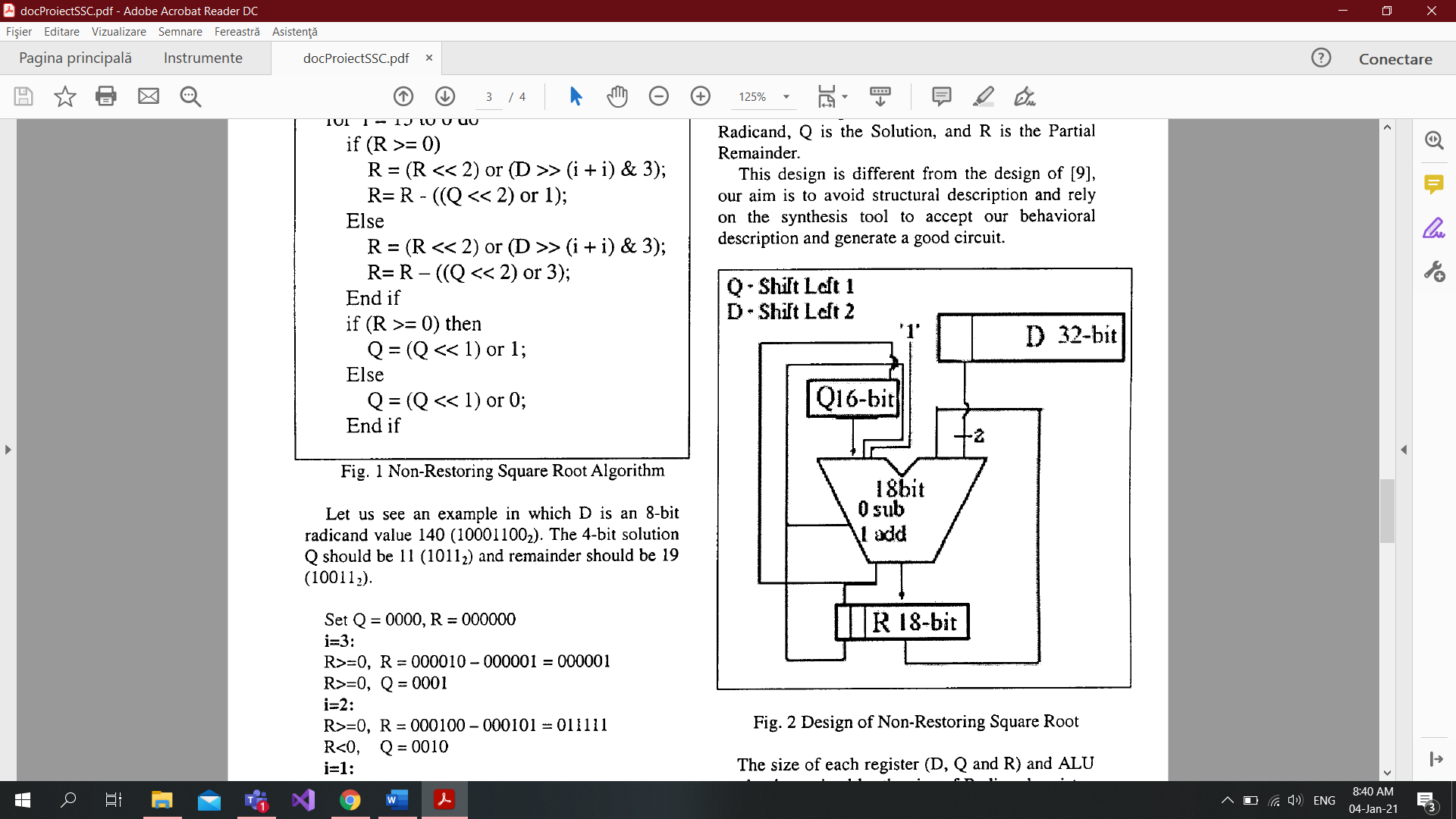


In continuare vom analiza un exemplu concret, unde D este un numar pe 8 biti cu valoarea 140(10001100). Solutia Q ar trebui sa fie 11(1011), iar restul ar trebui sa fie 19(10011).



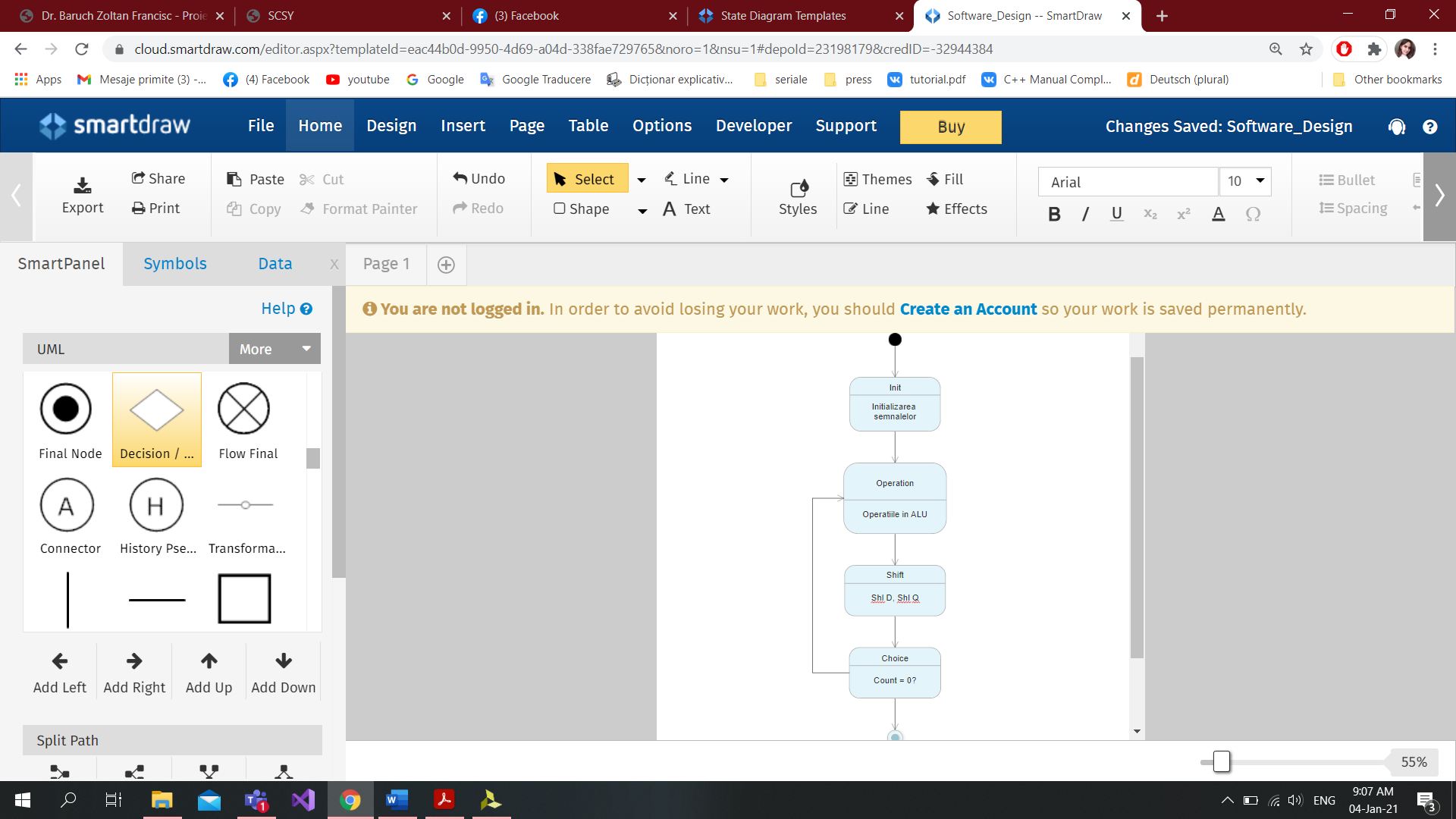
3.4. Schema bloc

In acest proiect, am ales sa dezvolt si sa implementez algoritmul Non-restoring, pentru care am folosit urmatoare schema-bloc:



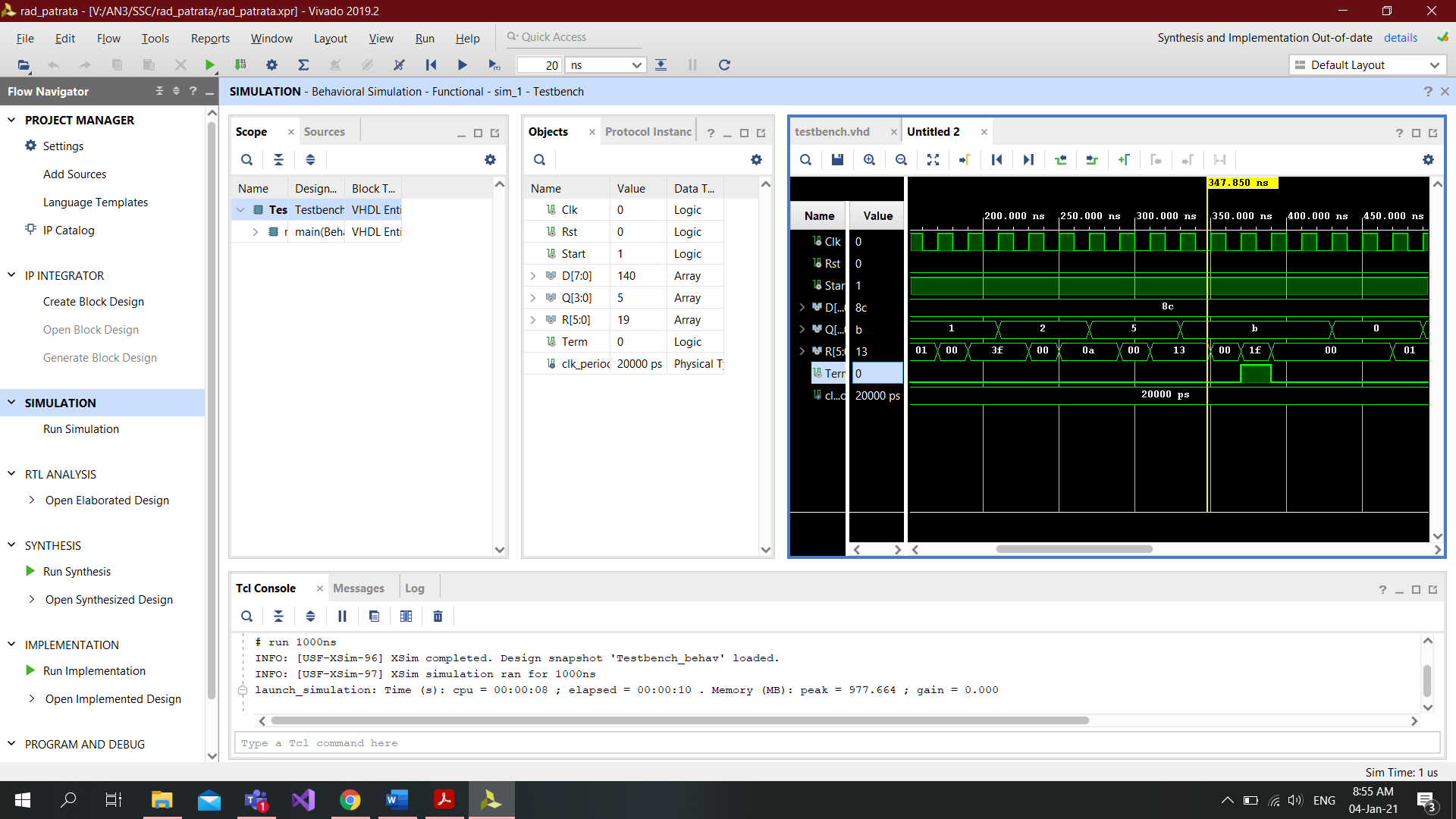
Marimea fiecarui registru(D, Q si R) si ALU depinde de marimea numarului initial din care se extrage radacina patrata. Daca D contine X biti, atunci Q va contine X/2 biti, in timp ce ALU si R vor fi de marime (X/2) + 2 biti. Numarul total de iteratii va fi de (X/2) + 1 cicluri de ceas. In fiecare ciclu de ceas, registrul D va fi shiftat la stanga cu 2 biti, iar registrul Q – la stanga cu un bit. Registrul R si Q trebuie initializate cu 0 odata ce se incarca valoarea initiala in registru D.

Pentru functionarea corecta a circuitului, am implementat urmatoarea diagrama de stari:

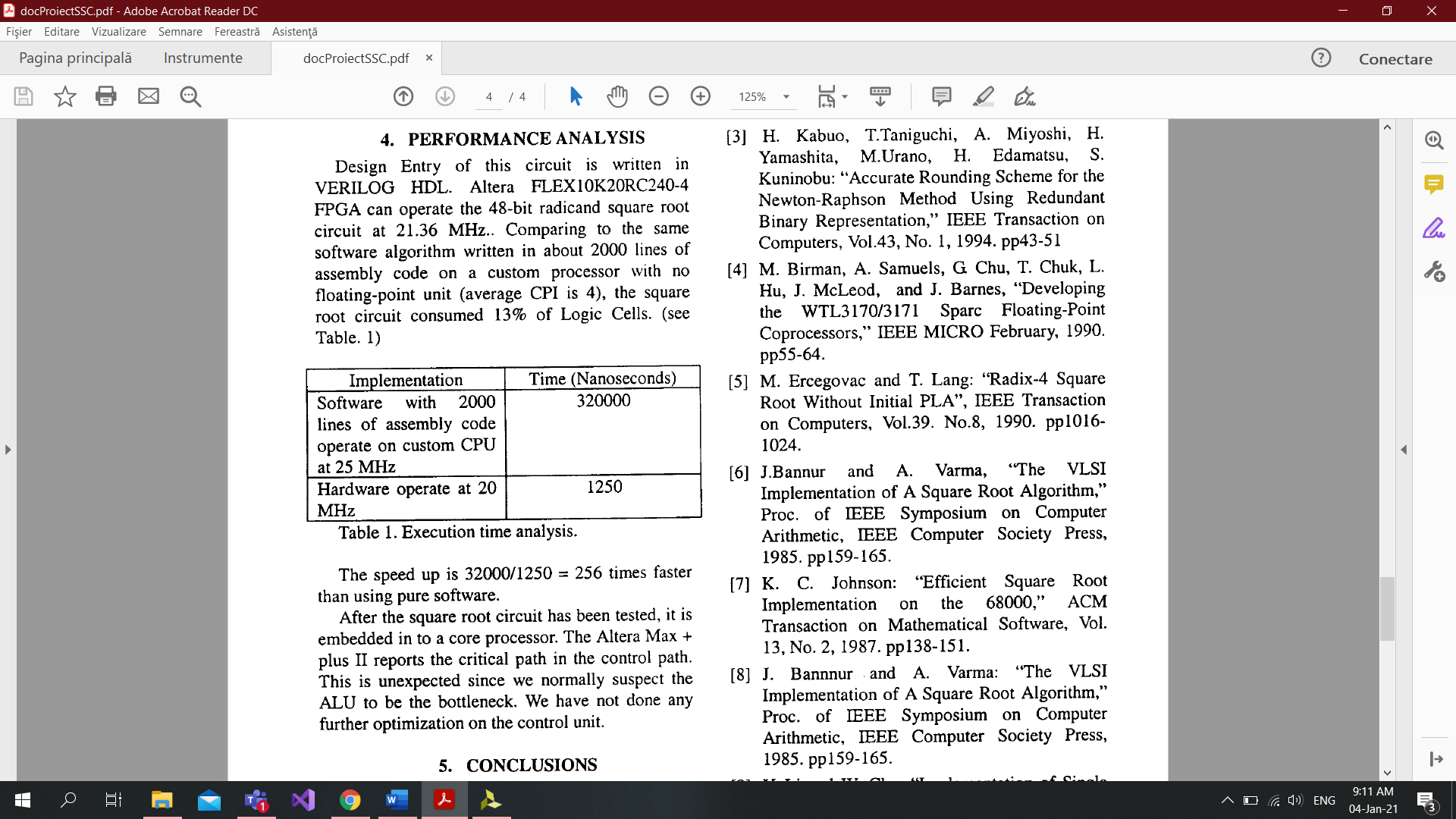


1. Rezultate experimentale

Algoritmul a fost implementat cu succes in cod VHDL. Acesta a fost testat in simulator. In continuare vom prezenta rezultatele simularii pentru valorile de intrare din exemplul anterior, unde D este un numar pe 8 biti cu valoarea 140(10001100). Solutia Q ar trebui sa fie 11(1011), iar restul ar trebui sa fie 19(10011). Pentru a ilustra acest exemplu, am creat un modul de test.



Din curiozitate, am comparat eficienta programului scris in VHDL cu acelasi algoritm scris in Assembly (in aproximativ 2000 linii de cod). Rezultatele sunt prezentate in tabelul de mai jos:



1. Concluzii

In prezent, exista multi algoritmi care permit calcularea radacinii patrate. Sistemele ce folosesc frecvent operatia de extragere a radicalului necesita implementarea algortmilor potriviti, care ar imbunatati performanta circuitelor printr-un timp scurt de generare a solutiei si printr-un spatiu mic.

Algoritmul Non-restoring poate fi implementat foarte simplu, iar rezultatul este obtinut rapid (comparativ cu algoritmul Newton-Raphson si metoda non-redundanta). Marele avantaj al acestui algoritm consta in utilizarea a doar 4 componente, ceea ce permite integrarea circuitului la scala mare. Pentru a creste si mai mult performanta circuitului, acesta poate fi imbuntatit prin utilizarea tehnicii pipeline.

1. Bibliografie
2. J. Hennesey and D.Patterson “Computer Architecture A Quantitative Approach”, Inc. 1996. Appendix A: Computer Artihmetic by D. Goldberg.
3. C. Ramamoorthy, J. Goodman and K. Kim: “Some properties of iterative Square-Rooting methods Using High-Speed Multiplication”, IEEE Transaction on computers, Vol C-21, N.8, 1972, pp837-847
4. H. Kabuo, T. Taniguchi : “Accurate Rounding Scheme for the Newton-Raphson Method Using Redundant Binary Representation,” IEEE Transaction on computers, 1994, pp43,51
5. M Birman, A. Samuels, G.Chu “Developing the WTL3170/3171 Sparc Floating-Point Coprocessors”, IEEE MICRO February 1990, pp55-64
6. M.Ercegovac and T.Lang: “Radix-4 Square Root Without Initial PLA”, IEEE Transaction on Computers, 1990, p1016-1024