Program de analiză și optimizare post-sinteză pentru Proiectarea VLSI

Raport nr. 1

Sabina Nadejda Barila
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației
sabina-nadejda.barila@student.tuiasi.ro

Coordonator: Ș.l. dr. ing. Mircea-Călin Monor
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
mircea-calin.monor@academic.tuiasi.ro

***Rezumat*—Lucrarea de licență se încadrează în domeniul proiectării VLSI (Very Large Scale Integration), axându-se pe automatizarea etapelor de analiză și optimizare post-sinteză a circuitelor digitale.**

I. Introducere

Complexitatea tot mai mare a circuitelor integrate digitale impune dezvoltarea de unelte software (CAD/CAE) care să automatizeze etape din fluxul de proiectare, în special pentru optimizare și analiză temporală. Proiectul de față urmărește realizarea unui program de tip CAD/CAE care are ca intrări principale:

- Un fișier sursă Verilog HDL, descriind la nivel de poartă logică un circuit digital.
- O bibliotecă de sinteză (liste de celule/porți cu atribute de timp și consum).
- Un fișier de constrângeri de timp (setup, hold, perioadă de ceas, etc.).

Pe baza acestor informații, aplicația va:

- Construi o reprezentare internă a circuitului sub formă de graf (noduri = porți, arce = conexiuni).
- Identifica și analiza căile logice, cu accent pe căile critice.
- Încerca rezolvarea căilor critice prin metode de optimizare la nivel de poartă.

Relevanța proiectului constă în faptul că optimizarea căilor logice contribuie în mod direct la creșterea performanței circuitelor digitale. Soluțiile existente sunt adesea de tip „black-box“, fiind puțin flexibile pentru personalizări. Prin urmare, dezvoltarea unei biblioteci de porți în C++ și a unei tehnici dedicate pentru optimizare poate demonstra potențialul de a integra noi metode de rearanjare și, totodată, de a face procesul mai transparent, ușor de înțeles și de adaptat la diverse scenarii (inclusiv academice sau de cercetare).

II. Stadiul Actual

1) Analiza situației curente din domeniu

Literatura de specialitate și practica industrială arată că sinteza și optimizarea circuitelor digitale au fost studiate intens, atât prin algoritmi euristici, cât și prin metode formale. Cele mai cunoscute soluții se regăsesc în suita de instrumente EDA (Electronic Design Automation), cum ar fi:

- **Synopsys Design Compiler** – oferă un lanț complet de sinteză, dar este un tool comercial cu costuri ridicate și acces limitat la miezul algoritmilor.
- **Cadence Genus și Innovus** – platformă integrată pentru sinteză și P&R (Place and Route); similar, accesul la algoritmii interni este restricționat.
- **Yosys (open-source)** – suportă sinteza Verilog RTL pentru FPGA și ASIC, însă optimizările avansate legate de timing sunt parțial implementate; există potențial de extindere.

2) Soluții existente și limitările acestora

Majoritatea tool-urilor comerciale abordează optimizările de timp folosind algoritmi complecși, care deseori rămân un „secret comercial”. Astfel, personalizarea poate fi dificilă sau chiar imposibilă în afara configurării parametrilor disponibili în instrumentele respective.

În zona open-source, Yosys oferă un grad mai mare de transparență, dar optimizarea bazată pe constrângeri de timp rămâne relativ limitată față de soluțiile comerciale. În plus, integrarea unor noi modele de porți sau algoritmi de optimizare necesită cunoștințe avansate despre codul sursă al Yosys.

3) Identificarea golurilor de cercetare

- Extinderea capacităților de optimizare pentru timing în instrumente open-source.
- Integrarea unei abordări personalizate, pentru a ușura înțelegerea internă a modului de optimizare (rearanjare logică, restructurarea netlist-urilor).
- Transparența și lizibilitatea fluxului de optimizare, care, în mediul educațional, ajută studenții și cer-

cetătorii să înțeleagă în profunzime mecanismele de sinteză.

4) **Context local versus internațional**

La nivel internațional, comunitatea academică și industrială folosesc preponderent tool-uri comerciale consacrate, iar la nivel local (în mediul universitar românesc) se utilizează atât suite comerciale, cât și Yosys în proiecte de cercetare. Nevoia de instrumente didactice personalizate și transparente rămâne însă un subiect de interes permanent.

III. Obiective

1) **Scopul general al proiectului**

Realizarea unei reprezentări interne a circuitului descris la nivel de poartă logică, sub formă de graf, determinarea tuturor căilor care apar în proiect și analizarea lor din punct de vedere al timpului, crearea listei de căi critice și aplicarea unor metode de optimizare din același punct de vedere în încercarea de a elimina căile critice și generarea unei noi descrieri în Verilog HDL care să fie în concordanță cu modificările făcute.

2) **Obiective specifice**

- Dezvoltarea unei biblioteci de porți logice (AND, OR, NOT, XOR, MUX etc.) în C++, cu capacități de modelare a conexiunilor (intrări, ieșiri, întârzieri).
- Parcurgerea și interpretarea unui netlist Verilog pentru a construi o structură internă care să permită procesarea/analiza ulterioară.
- Crearea reprezentării sub formă de graf a circuitului.
- Modelarea constrângerilor de timp (setup time, hold time, frecvența de ceas, latențe etc.) și integrarea lor în algoritmul de optimizare.
- Implementarea algoritmilor de optimizare a căilor logice (de ex. backtracking, dynamic programming sau algoritmi euristici), astfel încât să reducă întârzierile totale pe calea critică.

IV. Metodologie

1) **Resurse utilizate**

- Hardware : laptop persoanl (ACER NITRO5) cu un compilator C++ (GCC).
- Software : IDE pentru C++ (Visual Studio Code), sistem de versionare (Git) pentru urmărirea evoluției proiectului, un simulator Verilog (ModelSim sau Icarus Verilog) pentru verificarea corectitudinii netlist-urilor.

2) **Metode și algoritmi**

- Organizarea unei biblioteci de porți logice și pe parse-ul fișierelor de intrare.
- Reprezentare sub formă de graf – fiecare nod în graf va fi o poartă logică (AND, OR, NOT, NAND, etc.) cu atribute (timp de propagare, număr de intrări), iar arcele vor lega ieșirile porților de intrările altor porți.
- Analiza statică de timp (STA) – se va implementa o metodă de parcurgere a grafului pentru determinarea timpilor de întârziere pe fiecare cale. Se pot folosi metode de topological sorting pentru rețele aciclice sau, dacă există bucle secvențiale (flip-flop-uri), se va compartimenta analiza.
- Tehnici de optimizare – se vor încerca transformări structurale (înlocuirea unei secvențe de porți cu o poartă echivalentă mai rapidă, duplicarea unor ramuri etc.) și, dacă e posibil, retiming la nivel elementar (repoziționarea flip-flop-urilor).
- Generarea noului Verilog HDL – după aplicarea optimizărilor, codul structural trebuie regenerat, astfel încât să reflecte noile conexiuni și instanțe de porți.

V. Activități desfășurate până în prezent

- Documentare: a fost analizată literatura de specialitate și modul de lucru al tool-urilor comerciale.
- Definirea formatelor de intrare: s-au studiat fișierele netlist Verilog la nivel gate-level, cu scopul de a construi un parser minimal care să identifice porțile și conexiunile.
- Crearea unei structuri de date inițiale: s-a început proiectarea claselor C++ (`Gate`, `Net`, `Connection`) și a unui modul simplificat de parcurgere pentru construcția grafului.
- Înțelegerea fișierelor de constrângeri: s-a stabilit cum vor fi mapate constrângerile (perioada de ceas, timp de setup, timp de hold) în structurile interne.
- Testare preliminară: s-au rulat exemple mici (circuit de adunare pe 4 biți) pentru a vedea cum se construiește graful și cum se calculează întârzierile.

VI. Rezultate corelate cu obiectivele semestrului I

- În semestrul I, obiectivele stabilite împreună cu îndrumătorul au vizat:
  - Documentarea și analiza literaturii – finalizată.
  - Conceperea bibliotecii de porți – în progres, cu o parte din clase deja funcționale.
  - Implementarea unui parser minim – parțial realizată, având suport pentru porți de bază.
- Ce se preconizează să fie obținut în etapa următoare:
  - Extinderea parser-ului pentru a suporta mai multe tipuri de porți și structuri Verilog.
  - Implementarea completă a metodelor de analiză statică de timp (AST) pentru identificarea căilor critice.
  - Integrarea unei prime versiuni a algoritmului de optimizare (cel puțin pentru un subset de circuite) și evaluarea performanțelor pe câteva exemple.

VII. Concluzii preliminare

- **Sinteza progresului actual**

Proiectul se află într-un stadiu incipient de implementare, însă pașii preliminari, precum definirea bibliotecii de porți logice și crearea unui parser simplu, sunt deja realizați. Stadiul actual al lucrării confirmă fezabilitatea

ideii, iar studiul bibliografic arată că există un interes continuu pentru unelte flexibile și transparente de optimizare logică.

- **Pași următori pentru îndeplinirea obiectivelor proiectului**
  - Finalizarea parsării netlist-urilor complexe din Verilog.
  - Implementarea și rafinarea algoritmului de optimizare, incluzând retiming sau alte tehnici avansate (dacă timpul permite).
  - Validarea rezultatelor prin compararea cu unelte de sinteză consacrate (ex. Yosys) pentru un set de circuite de test.
  - Documentarea detaliată a performanței obținute și a limitărilor identificate.

BIBLIOGRAFIE

[1] D. D. Gajski, N. D. Dutt, A. Wu, and S. Lin, *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[2] W. Wolf, *Modern VLSI Design: IP-Based Design*, 4th ed. Pearson, 2008.

[3] L. Benini and G. D. Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Springer, 1998.

[4] Synopsys, *Synopsys Design Compiler User Guide*, 2020.

[5] Cadence, *Cadence Genus Synthesis Solution Datasheet*, 2021.

[6] C. Wolf, "Yosys open synthesis suite," http://www.clifford.at/yosys/.

[7] M. Soeken and R. Drechsler, *Formal Verification and Synthesis of Boolean and Reversible Circuits*. Springer, 2018.