Universitatea Transilvania din Brașov

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Ing. Alexandru DINU

Contribuții la automatizarea verificării funcționale a sistemelor digitale

Contributions to the automation of functional verification of digital systems

REZUMAT / ABSTRACT

Conducător ştiințific

Prof.dr.ing. Petre Lucian OGRUȚAN

BRAȘOV, 2022

Universitatea Transilvania din Brașov

D-lui (D-nei) .....

COMPONENȚA

Comisiei de doctorat

Numită prin ordinul Rectorului Universității Transilvania din Braşov
Nr. ..... din .....

PREŞEDINTE:	Conf.dr.ing. Carmen GERIGAN Universitatea Transilvania din Brașov
CONDUCĂTOR ŞTIINȚIFIC:	Prof.dr.ing. Petre Lucian OGRUȚAN Universitatea Transilvania din Brașov
REFERENȚI:	Prof.dr.ing. Gheorghe ȘTEFAN Universitatea Politehnica din București
	Prof.dr.ing. Dan PITICĂ Universitatea Tehnică din Cluj-Napoca
	Prof.dr.ing. Mihai ROMANCA Universitatea Transilvania din Brașov

Data, ora şi locul susținerii publice a tezei de doctorat: 16.09.2022, ora 10:00, sala N.II.1

Eventualele aprecieri sau observații asupra conținutului lucrării vor fi transmise electronic, în timp util, pe adresa alexandru.dinu@unitbv.ro.

Totodată, vă invităm să luați parte la şedința publică de susținere a tezei de doctorat.

Vă mulțumim.

CUPRINS

CONTENT

LISTA DE ACRONIME

AMBA – Advanced Microcontroller Bus Architecture

ANN – Artificial Neural Network

API – Application Programming Interface

ASIC – Application Specific Integrated Circuit

AVM – Advanced Verification Methodology

AXI - Advanced eXtensible Interface

BCL – Base Class Library

BN – Batch Normalization

CI – Circuit integrat

CDTG – Coverage Driven Test Generation

CDV – Coverage Driven Verification

CRC – Cyclic Redundancy Check

CSV – Comma-Separated Values

CPU – Central Processing Unit

CR – Carriage Return

CRC - Cyclic Redundancy Check

CSV – Comma-Separated Values

CUDA – Compute Unified Device Architecture

DL – Deep Learning

DNN – Deep Neural Networks

DUT – Design Under Test

DUV – Design Under Verification

eRM – e Reuse Methodology

FC – Fully Connected

FPGA – Field Programmable Gate Array

FSM – Finite State Machine

FV – Formal Verification

GEP – Generation Expansion Planning

GPU – Graphics Processing Unit

IA – Inteligență Artificială

IP – Internet Protocol

IPv4 – Internet Protocol Version 4

ÎBR – Învățarea bazată pe recompense

JTAG – Joint Test Action Group

KNN – K-Nearest Neighbor

LF – Line Feed

LFSR - Linear Feedback Shift Register

LR – Logistic Regression

LSB – Least Significant Bit

LUT – Look-Up Table

ML – Machine Learning

MDS – Multi-Dimensional Scaling

MDT – Metode care folosesc Dependența Temporală

MDV – Metric Driven Verification

MOS – Mean Opinion Score

MSB – Most Significant Bit

MESV – Marcarea Extinsă a Stărilor Vizitate

MOEA – Multi-Objective Evolutionary Algorithm

MSV – Marcarea stărilor vizitate

NB - Naïve Bayes

NSGA-II – Non-dominated Sorting Genetic Algorithm II

OVM – Open Verification Methodology

PC – Personal Computer

PCIe - Peripheral Component Interconnect Express

PVT – Process-Voltage-Temperature

QoE – Quality of Experience

RAM – Random Access Memory

ReLU - Rectified Linear Unit

RF – Random Forests

RCP – Rata de preicții Corect Pozitive

RFP – Rata de prediții Fals Pozitive

RN – Rețele neurale

ROC - Receiver Operating Characteristic

RTL – Register Transfer Logic

SFMI – Secțiune Feroviară de Mare Interes

SGA – Simple Genetic Algorithm

SGD – Stochastic Gradient Descent

SGTD – Semi-Gradient Temporal-Difference

SPEA2 – Strength Pareto Evolutionary Algorithm 2

SoC – System on Chip

SRAM – Static-RAM

SVA – SystemVerilog Assertions

SVM – Support Vector Machines

TEE – Testarea Echipamentelor Electronice

TF-IDF – Term Frequency – Inverse Document Frequency

TLM - Transaction-Level Modelling

TTL – Transistor-Transistor Logic

TTM – Time-To-Market

UAL – Unitate aritmetico-logică

UART – Universal Asynchronous Receiver/Transmitter

USB – Universal Serial Bus

UVM – Universal Verification Methodology

VA – Valoare de Aleatorizare

VHDL – VHSIC Hardware Description Language

VHSIC - Very High Speed Integrated Circuit

VMM – Verification Methodolody Manual

d.p.d.v. – din punct(ul) de vedere

eng. – în engleză

INTRODUCERE

1.1 ASPECTE GENERALE

În societatea actuală, utilizarea sistemelor digitale se răspândește din ce în ce mai mult. Dacă în 1981, se implementa IPv4, considerându-se că adresele IP disponibile vor fi suficiente pentru toate dispozitivele care se vor conecta la internet [1], în 2006 deja s-a lansat IPv6 (urmând să fie adoptat practic la o dată ulterioară), care permite conectarea la Internet a unui număr mult mai mare de dispozitive. Dinamismul societății, reflectat și în creșterea accelerată a numărului dispozitivelor electronice introduse pe piață, se bazează, printre altele, și pe munca desfășurată de companiile producătoare din industria electronică. În domeniul electronic, activează o multitudine de entități care contribuie într-un mod diversificat la dezvoltarea dispozitivelor electronice. Pentru a-și asigura competitivitatea în situația actuală, actorii industriali fac eforturi deosebite de a scurta perioada care se scurge între conceperea unui produs dezvoltat pe bază de procesor și apariția sa pe piață (eng. Time-To-Market - TTM).

Dezvoltarea unui ASIC durează cel puțin un an (numai în fabrica de semiconductoare, timpul minim de manufacturare este de 6 luni [2]) . Acest proces implică munca a zeci de oameni și, în cele mai multe cazuri, a mai multe companii. Având în vedere aceste aspecte, automatizarea unor etape din procesul de proiectare poate conduce la economisirea unor resurse importante. Una dintre tehnologiile cele mai promițătoare din punctul de vedere al automatizării realizării sistemelor digitale în general și a circuitelor integrate în particular este inteligența artificială, aceasta fiind integrată încă din 2017 de o mare parte a întreprinderilor [3].

În cadrul procesului de proiectare logică a unui circuit integrat, verificarea funcțională este începută în paralel cu proiectarea circuitului, și continuă până când sunt atinse în simulare toate situațiile funcționale descrise în planul de verificare. Dat fiind faptul că verificarea funcțională este o etapă foarte consumatoare de timp în procesul de dezvoltare a dispozitivelor electronice (aceasta ocupând aproximativ 70% din resursele alocate părții de proiectare logică digitală (eng. *front*-end) a unui circuit [4]), inginerii din domeniu depun eforturi deosebite pentru a reduce timpul necesar acesteia, printr-o automatizare judicioasă a realizării verificării.

Având în vedere aceste eforturi globale, în această teză de doctorat se urmărește dezvoltarea de metode de automatizare a verificării funcționale, pornindu-se de la cea mai folosită metodologie de verificare din industrie: Universal Verification Methodology (UVM). Realizându-se calcule complexe, și folosindu-se tehnologii care și-au demonstrat eficiența într-o gamă largă de aplicații din industrie, se poate realiza o economie considerabilă a timpului de dezvoltare a sistemelor digitale, totodată crescându-se calitatea procesului de realizare a acestora.

Un proces complementar verificării, care este de asemenea abordat pe parcursul dezvoltării circuitelor integrate este validarea. Dacă verificarea funcțională presupune testarea faptului că DUT-ul funcționează conform specificațiilor, validarea verifică însăși fezabilitatea acestora. Una dintre modalitățile cele mai întâlnite de validare este realizarea unui prototip al circuitului de fabricat folosind un dispozitiv a cărui structură cuprinde o matrice de porți programabile (eng. *Field*

Programmable Gate Array - FPGA). Dispozitivele de tip FPGA reprezintă o categorie importantă a sistemelor digitale a cărei aplicabilitate este complementară folosirii circuitelor integrate. Fabricarea CI este rentabilă în cazul în care se produce o serie de sute de mii sau chiar milioane de produse, dat fiind faptul că prețul pentru realizarea măștilor de fabricare și costul celorlalte etape se împarte la mai multe dispozitive. În cazul în care se produce o serie mai puțin numeroasă de sisteme digitale, costurile de fabricație specifice CI devin prea mari pentru a putea fi transferate fiecărui dispozitiv vândut. În acest caz, funcționalitatea viitorului sistem digital poate fi implementată pe FPGA. Aceste dispozitive au avantajul de a putea fi reconfigurate, funcționalitatea acestora putând fi actualizată în raport cu avansul tehnologic. Prețul acestor dispozitive este însă semnificativ mai mare decât prețul unui cip dintr-o serie de milioane de bucăți de CI. O variantă intermediară de implementare a funcționalității sistemelor digitale, care oferă un cost mai scăzut față de implementările bazate pe FPGA și un timp redus de fabricație în comparație cu un CI este tehnologia ASIC structurat (eng. *Structured ASIC*) [5].

Prin configurarea dispozitivului de tip FPGA conform codului circuitului de realizat care modelează legăturile logice la nivel de regiștri (eng. *Register Transfer Logic* – RTL), sistemul digital va putea fi testat în cadrul dispozitivului real în care a fost proiectat să funcționeze. Astfel, având în vedere aspectele legilor fizice care poate nu au fost reproduse fidel de simulatoare (și deci au ascuns anumite probleme ale circuitului a cărui funcționalitate a fost simulată), specificația sistemului digital poate fi actualizată și completată în urma concluziilor extrase din funcționarea prototipului implementat pe FPGA. De asemenea, deși accesul la semnalele interne ale unui proiect implementat pe FPGA este dificil și, din acest motiv, validarea nu are același potențial de detectarea a erorilor ca și verificarea funcțională, dispozitivele FPGA pot evidenția și erori de funcționalitate ale DUT-ului în raport cu specificația acestuia. Având în vedere contribuția importantă a procesului de validare la obținerea unui DUT cu o funcționare optimă, pe lângă verificarea funcțională s-a abordat și acest aspect în cadrul tezei. În acest scop, s-a creat un sistem software (eng. *framework*) prin care valori de interes ale semnalelor DUT-ului implementat pe FPGA sunt preluate, procesate și analizate în vederea creării unui model de referință care poate fi folosit pentru detectarea unor probleme de funcționalitate ale DUT-ului.

1.2 TEMA ȘI OBIECTIVELE TEZEI

Tema tezei de față este verificarea funcțională a sistemelor digitale, iar subiectul tezei este crearea unor sisteme software de automatizare a verificării funcționale și a validării folosind tehnici de inteligență artificială. Principalele obiective urmărite în teza de față sunt:

1. Analiza potențialului de automatizare a procesului de verificare funcțională
2. Obținerea modelelor de referință pentru sisteme digitale folosind învățarea automată. Aceste modele pot fi obținute atât pe baza unui DUT simulat cât și pe baza unui proiect digital care funcționează în cadrul unui dispozitiv cu FPGA
3. Generarea automată de secvențe de stimuli care să aducă DUT-ul într-o stare dorită într-un timp optim
4. Generarea automată de secvențe de stimuli prin care să se atingă valoarea maximă de acoperire a situațiilor de funcționare avute în vedere la un moment dat

Pentru atingerea obiectivelor principale, menționate a fost necesară stabilirea mai multor obiective operaționale, cu rolul de ghidare a muncii de cercetare și de măsurare a succesului diferitelor abordări de automatizare. Aceste obiective sunt numerotate astfel încât să sugereze cărui obiectiv principal îi aparțin:

1.1. Analiza metodelor de proiectare și verificare a circuitelor integrate
1.2. Determinarea proceselor a căror automatizare are un potențial ridicat pentru îmbunătățirea calității sau diminuarea timpului de verificare funcțională al unui circuit integrat
1.3. Studiul inițiativelor de automatizare ale verificării aparținând mediului academic și industriei
1.4. Studiul metodelor de implementare ale inteligenței artificiale
2.1. Realizarea a minim trei medii de verificare folosind metodologia UVM și biblioteca UVM1.2
2.2. Dezvoltarea unor configurații optime de rețele neurale pentru a antrena un model cu rol de clasificare a seturilor de date
2.3. Determinarea algoritmilor optimi de învățare automată care pot determina corectitudinea unui transfer serial conform protocolului UART
2.4. Realizarea unui sistem complet cu rolurile de a prelua a datele de pe placa cu FPGA Spartan 3E prin intermediul protocolului UART, de a extrage informațiile de interes, de a procesa datele pe calculator și de a crea o bază de date gata de utilizat pentru antrenarea modelelor de inteligență artificială
2.5. Antrenarea unor modele de inteligență artificială cu rol de clasificare pe baza datelor extrase de la un modul de calcul al votului majoritar implementat pe placa cu FPGA Spartan 3E și compararea performanțelor acestora.
2.6. Definirea structurii unui sistem de monitorizare în timp real al modulelor implementate pe FPGA
3.1. Configurarea simulatorului ModelSim® pentru a putea fi utilizat pentru verificarea funcțională
3.2. Crearea unei scheme eficiente de recompense pentru antrenarea unui agent cu rolul de a învăța comportamentul unei unități aritmetico-logice
3.3. Conceperea unui sistem *software* de control al simulatorului ModelSim® și de citire și procesare a rezultatelor obținute în urma simulărilor
3.4. Integrarea unui algoritm de tipul *semi-gradient temporal-difference*(0) – SGTD(0) în sistemul software folosit pentru antrenarea unui agent de învățare bazată pe recompense
3.5. Compararea performanțelor modelelor antrenate folosind învățarea bazată pe recompense cu performanțele abordării clasice de verificare în vederea aducerii unui DUT într-o stare dorită
4.1. Construirea unui set de versiuni de algoritmi genetici cu rolul de a crește valoarea de acoperire al unor elemente de interes din verificare având la bază descrierea algoritmului genetic simplu (eng. Simple Genetic Algorithm [6])
4.2. Integrarea a diverse tipuri de algoritmi genetici în sistemele folosite pentru automatizarea atingerii valorii maxime de acoperire
4.3. Alegerea unei valori optime pentru coeficientul de mutație folosit de către algoritmii genetici

4.4. Compararea performanțelor abordărilor dezvoltate în cadrul acestei teze care au la bază algoritmi genetici cu performanțele obținute de versiunile mono-obiectiv ale algoritmilor SPEA2 și NSGA-II

4.5. Compararea performanțelor algoritmilor genetici cu performanțele abordării clasice de verificare în ceea ce privește obținerea valorii maxime de acoperire funcțională

Pe lângă obiectivele tehnico-științifice ale tezei, este urmărit și un obiectiv de ordin didactic: crearea unei experiențe deosebite și utile pentru studenții care învață despre verificarea circuitelor integrate, prin folosirea la clasă a unor mijloace des întâlnite în mediul industrial: lucrul în echipe în care fiecare membru are roluri asemănătoare cu cele din industria circuitelor integrate; utilizarea unor platforme on-line pentru administrarea sarcinilor personale, urmărirea progresului echipei; semnalizarea și rezolvarea în echipă a problemelor apărute pe parcursul proiectului. Atingerea acestui obiectiv, alături de pregătirea tehnică dobândită în cadrul tezei, reprezintă un plus de experiență important și necesar pentru a începe o carieră didactică.

De asemenea, pentru a valida rezultatele cercetărilor și înainte de susținerea publică a teze s-a stabilit ca obiectiv publicarea unor părți din cercetările efectuate în publicații cu expunere internațională, indexate în baza de date *Web Of Science*.

1.3 SUMAR AL CONȚINUTULUI LUCRĂRII

Structura acestei lucrări a fost realizată în concordanță cu obiectivele propuse.

Capitolul 2, **2 VERIFICAREA FUNCȚIONALĂ A CIRCUITELOR INTEGRATE**, conține informații relevante despre desfășurarea verificării conform abordărilor uzuale din industrie. De asemenea, sunt evidențiate oportunități de automatizare ale verificării funcționale. Capitolul se încheie cu prezentarea rezultatelor unei selecții de lucrări relevante din domeniul automatizării verificării circuitelor integrate și cu o listă a algoritmilor care au furnizat sau au potențial de a furniza rezultate bune în acest domeniu.

Capitolul 3, **3 MODALITĂȚI PRACTICE DE AUTOMATIZARE A VERIFICĂRII FOLOSIND INTELIGENȚA ARTIFICIALĂ**, prezintă conceptele pe care se bazează o bună parte din aplicațiile dezvoltate în domeniul inteligenței artificiale: învățarea automată, învățarea în profunzime și învățarea bazată pe recompense. În continuare sunt prezente trei studii care reprezintă o parte importantă a nucleului acestei teze. Primul studiu are în vedere realizarea unor modele de referință cu rol de clasificare pentru un proiect digital. Printre rezultatele acestui studiu, se numără modele care au reușit performanța de a avea peste 99% acuratețe în ceea ce privește determinarea tipului de transfer (corect sau eronat) care a ajuns la porturile unui proiect digital. Cel de-al doilea studiu are în vedere antrenarea unui agent de învățare bazată pe recompense. Acest agent mai întâi învață care sunt particularitățile mediului în care evoluează (fiecare stare a mediului fiind echivalentă cu o stare a DUT-ului la verificarea căruia va contribui), iar apoi este capabil să aleagă singur stimulii pe care trebuie să îi furnizeze DUT-ului astfel

încât să îl aducă în stările dorite de inginerii de verificare pentru a-i testa funcționalitatea. Acest studiu este prefațat de o parte de teorie solid fundamentată matematic care prezintă algoritmul folosit pentru antrenarea agenților. Cel de-al treilea studiu din acest capitol are în vedere generarea unor secvențe de date care să atingă valoarea maximă de acoperire a elementelor de interes în cadrul unui singur test de verificare. S-a demonstrat faptul că atât în ceea ce privește generarea secvențelor de date pentru ca DUT-ul să ajungă într-o anumită stare, cât și în ceea ce privește generarea secvențelor de date care să atingă o valoarea maximă a acoperirii funcționale, metodele dezvoltate în cadrul acestei lucrări depășesc în performanțe modalitatea clasică de generare constrâns-aleatorie a stimulilor folosită pe scară largă în industria verificării circuitelor integrate.

Capitolul 4, **4 VALIDAREA SISTEMELOR DIGITALE**, conține informații despre preluarea datelor de pe o placa Spartan 3E cu FPGA care sunt transmise unui calculator folosind protocolul serial Universal Asynchronous Receiver/Transmitter (UART). Apoi, se prezintă procesul dezvoltat pentru a se recompune și a se interpreta pe calculator datele provenite de pe placa cu FPGA, urmând ca acestea să fie stocate în format tabelar. Ulterior datele colectate sunt folosite pentru a antrena modele care învață dependența între intrările și ieșirea DUT-ului. Astfel, se probează posibilitatea de a crea modele de referință *software* pentru modulele *hardware* implementate pe FPGA, propunându-se ca aceste modele să fie folosite pentru depanarea sau monitorizarea în timp real a funcționalității dispozitivelor electronice.

Capitolul 5, **5 VERIFICAREA FUNCȚIONALĂ DISTRIBUITĂ ÎNTR-O ABORDARE DIDACTIC**, oferă detalii despre o modalitate de predare modernă, care include diverse aspecte întâlnite în industrie, a noțiunilor de verificare a circuitelor integrate. Mediul de lucru în echipă și tematica laboratorului au constituit elemente atractive pentru studenți care i-au stimulat pentru a realiza niște proiecte cuprinzătoare și le-a trezit interesul pentru acest domeniu. În urma prezentării proiectelor în fața unor companii din domeniu, studenții au putut să intre în legătură cu reprezentanții companiilor în vederea realizării de stagii de practică sau a angajării după finalizarea studiilor.

Capitolul 6, **6 CONCLUZII ȘI CONTRIBUȚII ORIGINALE**, cuprinde concluziile principale extrase din studiile realizate în cadrul tezei de doctorat, menționează cele mai importante rezultate obținute, prezintă câteva posibilități de continuare a studiilor și enumeră lucrările publicate.

2 VERIFICAREA FUNCȚIONALĂ A CIRCUITELOR INTEGRATE

2.1 IMPORTANȚA VERIFICĂRII

Pe parcursul ultimelor zeci de ani, plăcile electronice cu componente discrete au fost înlocuite treptat de circuitele integrate (CI). Acest lucru a dus la creșterea exponențială a raportului dintre funcționalitățile realizate de CI și dimensiunea acestora. Prin integrarea unui număr mare de componente în același cip s-a îngreunat, de asemenea, posibilitatea de detectare a erorilor din dispozitivele electronice.

Trecerea de la circuite discrete la circuite integrate a încurajat realizarea unor sisteme digitale cu funcții deosebit de complexe. Aceste funcționalități complexe pot fi însă viciate chiar și de greșeli minore care pot apărea pe parcursul realizării proiectelor digitale. De aceea, diversitatea de funcționalități pe care le poate conține un sistem digital în ziua de astăzi este pusă în valoare doar atunci când greșelile de implementare *hardware* lipsesc sau sunt atât de neînsemnate încât pot fi compensate din modul de realizare al componentei *software* a proiectului. Dacă în programare, repararea unei greșeli se face relativ ușor, printr-o actualizare *software* a funcționalității defectuoase, în proiectarea digitală lucrurile se prezintă într-un mod total diferit, dat fiind modul de fabricare al circuitelor integrate. Pentru a realiza în fabrică un circuit integrat este necesară aplicarea unor procedee tehnologice costisitoare, pe parcursul unei perioade îndelungate de timp: depunere de material, litografie, diverse tratamente chimice (ex. oxidare, corodare) ale substratului de siliciu, crearea unor măști care vor fi apoi folosite pentru a crea traseele din siliciu prin intermediul fotolitografiei, etc. De aceea nu este ieșit din comun ca fondurile necesare realizării unui circuit integrat să ajungă până la zeci de milioane de euro [7]. Dat fiind faptul că realizarea circuitelor integrate este un proces atât de costisitor, este critic ca proiectul logic care va fi transpus în siliciu să nu conțină erori. Acest lucru este urmărit atât prin efectuarea verificării funcționale a circuitelor integrate cât și prin validarea acestora, folosindu-se plăci de prototipare (de exemplu, dispozitive de tip FPGA). Ajustarea măștilor sau a altor componente realizate în fabricile de circuite integrate în situațiile în care proiectul unui cip a fost trimis în fabrică având probleme funcționale este foarte costisitoare, dacă nu imposibilă.

Pe de altă parte, validarea (care este realizată, în multe cazuri, prin transferul proiectului într-un dispozitiv FPGA) confirmă fezabilitatea conceptului care stă la baza modulului, solicitând modificări în specificație dacă este cazul. De asemenea, procesul de validare al circuitului este folosit și pentru a detecta erori funcționale [8].

În cele din urmă, după ce circuitul integrat a fost fabricat, acesta este testat pentru a se elimina cipurile care au erori din cauza viciilor de fabricație. Partea comună pentru toate cele trei procese -

verificare, validare și testare- este reprezentată de existența unui mecanism de generare a stimulilor care sunt transmiși DUT-ului.

Atât pe dispozitivul cu FPGA, cât și în circuitul integrat deja realizat, accesul la semnalele interne ale dispozitivului electronic este extrem de redus. Mai mult, preluarea unei cantități suficiente de valori ale semnalelor de pe un dispozitiv fizic și transferul lor pe unitatea computerizată a bancului de probe unde s-ar putea afla programele de verificare ale corectitudinii acestora sunt niște operațiuni dificile. Principala problemă constă în viteza de transfer a datelor dintre sistemele electronice, care trebuie să depășească viteza datelor vehiculate în cadrul DUT-ului (aceasta din urmă fiind de multe ori echivalentă chiar cu una din frecvențele de funcționare ale circuitului). În cazul dispozitivelor FPGA, chiar dacă valorile datelor pot fi preluate la viteze mari și transferate în memoria RAM, apare problema insuficienței memoriei necesare pentru stocarea unei mari cantități de date care poate fi necesară pentru evaluarea complexă a diferite funcționalități. De aceea, atunci când se folosește un dispozitiv FPGA, este mai potrivită analiza corectitudinii funcționării unor semnale direct pe placa unde acestea au fost generate [9]. Un astfel de sistem este prezentat în cadrul capitolului 0 al acestei teze. Având în vedere limitările prezentate mai sus, verificarea funcțională a proiectului RTL al unui CI rămâne cel mai potrivit candidat dintre cele trei procese menționate (verificare, validare, testare) pentru detectarea erorilor de funcționalitate.

2.2 REALIZAREA VERIFICĂRII FUNCȚIONALE A CIRCUITELOR INTEGRATE

Verificarea funcțională a circuitelor integrate are ca scop descoperirea problemelor introduse involuntar de către proiectanți în proiectul digital al cipului. Aceasta se realizează prin simulare sau prin verificarea prin formule (eng. *formal verification* - FV) a codului RTL al sistemului digital.

În teza de față accentul se pune pe verificarea dinamică sau verificarea prin simulare. Verificarea dinamică poate utiliza și elemente specifice verificării prin formule (de ex. aserțiuni) pe lângă alte metrici prin care se măsoară calitatea modulului verificat (ex. elemente de verificare a datelor – eng. *checkers*) și a verificării în sine (ex. elemente de acoperire – eng. *coverage*). Pentru ca DUT-ul să poată fi stimulat și pentru a i se putea citi ieșirile, se creează un mediu de verificare în jurul acestuia. Componentele mediului de verificare sunt configurate astfel încât să poată comunica cu proiectul digital conform protocoalelor pe care acestea le implementează pentru fiecare interfață.

Una din principalele provocări pe care le întâlnesc inginerii de verificare este faptul că aceștia trebuie să creeze secvențele de date care vor fi trimise DUT-ului în scopul aducerii acestuia în toate situațiile de funcționare menționate în specificație. Metrica folosită pentru a se evalua în ce măsură au fost simulate funcționalitățile DUT-ului până la un anumit moment se numește acoperire funcțională (eng. *functional coverage*). Metoda folosirii acestei metrici ca o condiție de încheiere a verificării este cunoscută sub numele de verificare ghidată de acoperire (eng. *Coverage Driven Verification* - CDV). Din acest punct de vedere, conform verificării clasice ale cărei principii de bază sunt descrise în Metodologia Universală de Verificare (Universal Verification Methodology – UVM) [10], inginerii au la dispoziție două abordări:

- realizarea de teste direcționate, care să implementeze scenarii specifice de verificare descrise în specificația circuitului electronic; prin aceste scenarii, se poate configura

mediul de verificare astfel încât să se atingă cazuri limită ale funcționării DUT-ului; dezavantajul acestei abordări este faptul că este greu, și de multe ori imposibil, ca inginerii care construiesc planul de verificare să își imagineze toate situațiile limită în care ar putea ajunge să funcționeze circuitul integrat;

- realizarea de teste cu date generate aleatoriu, dar care sunt constrânse în limitele valide de valori conform specificației (eng. *constrained random generation*); acest mod de realizare a verificării are avantajul de a aduce proiectul digital într-o multitudine de situații de funcționare cu un minim efort de configurare; principalul dezavantaj al acestei abordări este timpul necesar rulării miilor de simulări care trebuie să aducă DUT-ul în toate situațiile de funcționare de interes; mai mult, este posibil ca testele cu date generate constrâns-aleatoriu să nu poată realiza toate scenariile dorite sau descrise în specificație, în aceste cazuri fiind necesare fie adăugarea de constrângeri suplimentare în generarea datelor, fie apelarea la scrierea de teste direcționate.

Verificarea clasică prin simulare presupune combinarea celor două metode de mai sus. Mai întâi se începe cu crearea testelor unde datele sunt generate aleatoriu, iar apoi, în funcție de scenariile existente în planul de verificare care nu au fost atinse, inginerii de verificare intervin pentru a direcționa evoluția simulărilor viitoare pentru atingerea scopurilor dorite. În cadrul acestei teze se urmărește degrevarea inginerilor de necesitatea de a crea o mare parte din testele direcționate. Un mod de a realiza acest lucru este exploatarea conceptului de generare a testelor în scopul îmbunătățirii procentului de acoperire funcțională (eng. *coverage driven test generation* – CDTG) abordat atât de entități academice cât și de companii din industrie [11-13]. Acest concept presupune citirea valorilor de acoperire funcțională de către simulator, realizarea corelațiilor între stimulii transmiși și evoluția gradului de acoperire funcțională și generarea automată de noi stimuli prin care să se ruleze și scenariile de interes care încă nu au fost atinse în simulare. Folosind cuvintele cheie "corelații între stimuli" și "generarea automată" se poate intui faptul că un candidat potrivit pentru a contribui la această sarcină este inteligența artificială care are ca punct forte detectarea corelațiilor existente între date. De aceea, s-a urmărit în această teză aplicarea metodelor de inteligență artificială pentru realizarea unor modele de referință *software* ale circuitelor și generarea automată a stimulilor care, odată trimiși DUT-ului, îmbunătățesc procentul scenariilor de verificare atinse în simulare. În acest scop s-au utilizat învățarea automată (eng. *machine learning*), învățarea profundă (eng. *deep learning*), învățarea ghidată de recompense (eng. *reinforcement* learning) și algoritmii genetici.

Integrarea inteligenței artificiale în verificarea circuitelor integrate are mai multe avantaje:

- automatizarea procesului de verificare
- realizarea mai rapidă a sarcinilor
- creșterea calității verificării, prin atingerea în simulare a unui număr crescut de scenarii de funcționare ale proiectului digital

Aceste avantaje sunt evidențiate în studiile realizate și prezentate în cadrul tezei. Folosirea inteligenței artificiale are însă și o serie de neajunsuri: este necesară rularea unui set inițial de simulări care consumă timp. În situații complexe, antrenarea unor modele de IA care să beneficieze de o

acurateţe minimă necesară (această valoare depinde de la caz la caz, în cadrul lucrării de față, valorile de acurateţe de peste 97% sunt considerate acceptabile) poate dura foarte mult, acest timp adăugându-se la perioada necesară rulării unui număr mare de simulări. De aceea, înainte de a antrena un model de inteligență artificială, trebuie evaluată complexitatea funcționalităților DUT-ului de verificat. De exemplu, dacă unele dintre acestea pot fi concis reprezentate matematic, atunci verificarea prin formule poate reprezenta soluția optimă.

2.3 VERIFICAREA FUNCȚIONALĂ PRIN SIMULARE

Verificarea funcțională este o etapă complexă în proiectarea CI, fiind "partea leului" din cadrul proiectării circuitelor integrate" [14]. Aceasta implică echipe numeroase de ingineri, proporționale cu complexitatea ASIC-ului dezvoltat. Conform [14], există trei tipuri de metodologii de verificare: verificarea funcțională dinamică, verificarea funcțională statică și verificarea funcțională hibridă.

Verificarea se realizează prin simularea comportamentului DUT-ului (pentru a se vedea dacă acesta funcționează conform specificațiilor) și prin determinarea funcționalității modelului matematic al DUT-ului (prin verificarea formală).

Pe baza specificației funcționale a modulului, atât proiectantul ("designer-ul"), cât și inginerii de verificare creează un model al DUT-ului. Proiectantul creează un model sintetizabil, iar inginerii de verificare creează un model mai abstract al DUT-ului, nesintetizabil, numit "model de referință", după cum se poate vedea și în *Fig. 1*. Zona portocalie aparține mediului de verificare, care conține partea de generare, modelul de referință și structurile de monitorizare a datelor, iar zona verde aparține proiectului de verificat.

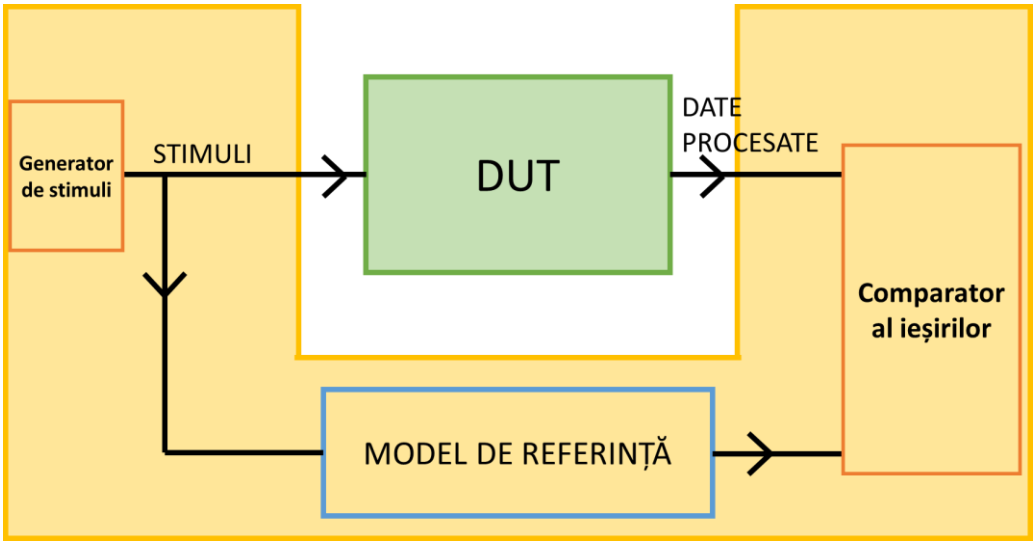

Fig. 1 Principiul verificării unui circuit folosind un model de referință

Dacă cele două modele, furnizându-li-se aceiași stimuli, funcționează identic, se certifică faptul că DUT-ul a fost proiectat bine.

Verificarea unui circuit se termină atunci când s-au simulat toate situațiile în care cipul poate ajunge la un moment dat, acesta funcționând fără eroare în toate aceste cazuri.

2.4 DIRECȚII DE ÎMBUNĂTĂȚIRE A PROCESULUI DE VERIFICARE FUNCȚIONALĂ

Verificarea, validarea și testarea se realizează după anumite norme care, de-a lungul timpului, cu concursul marilor actori industriali din domeniu, au fost îmbunătățite și uniformizate. În ziua de astăzi, toate cele trei procese amintite mai sus se bazează în principal pe utilitare software. Pentru ca puterea computațională să fie eficient exploatată, trebuie urmărite câteva aspecte "cheie" în cadrul verificării funcționale: realizarea corelațiilor între stimuli furnizați la intrările unui circuit și valorile de acoperire sau funcționalitățile atinse ale DUT-ului, detectarea și simularea situațiilor "limită" în care poate intra circuitul, detectarea similitudinilor între scenariile de verificare și reducerea numărului de teste redundante.

2.5 STADIUL ACTUAL CU PRIVIRE LA AUTOMATIZAREA VERIFICĂRII CIRCUITELOR INTEGRATE

În lucrarea de față au fost analizate mai multe lucrări care au ca scop automatizarea procesului de dezvoltare al circuitelor integrate, accentul punându-se în special pe verificare. Printre rezultatele prezentate de colectivele de cercetare se află: corelarea tranzacțiilor (eng. *test vectors*) cu aserțiunile digitale activate de acestea [13], predicția evoluției unei simulări pe baza stărilor DUT-ului atinse la începutul acesteia [15], corelarea rezultatelor de acoperire funcțională cu stimulii de intrare care le-au generat în scopul antrenării unor modele de învățare supravegheată care au ca intrări rezultatele de acoperire funcțională și ca ieșiri valorile stimulilor [16], creșterea nivelului de acoperire funcțională pentru elementele care au cele mai mici valori [17], detectarea părților din proiectul digital care au fost prea puțin stimulate în decursul procesului de verificare [18], implementarea unui generator de cod pentru mediile de verificare folosind mai multe limbaje utilizate în verificarea ASIC-urilor [19], predicția nivelului de acoperire ale unor funcționalități ale DUT-ului ținând cont de caracteristici funcționale corelate cu acestea [20], generarea automată a elementelor de măsurare a completitudinii verificării [21], clasificarea și gruparea testelor pe baza datelor din rapoartele de simulare [22]. Interesul acordat acestui domeniu demonstrează că automatizarea procesului de verificare a circuitelor integrate are avantaje importante, dintre care cele mai importante sunt reducerea timpul de realizarea a circuitelor integrate și obținerea unei siguranțe mai ridicate cu privirea la lipsa problemelor funcționale.

Pe baza activităților de cercetare prezentate în cadrul acestui capitol au fost publicat articolele:

A. Dinu, P. L. Ogrutan, "Opportunities of using artificial intelligence in hardware verification," *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2019, pp. 224-227, doi: 10.1109/SIITME47687.2019.8990751.

și

A. Dinu, P. L. Ogruţan, "Coverage fulfillment methods as key points in functional verification of integrated circuits," *2019 International Semiconductor Conference (CAS)*, 2019, pp. 199-202, doi: 10.1109/SMICND.2019.8923695.

3 MODALITĂȚI PRACTICE DE AUTOMATIZARE A VERIFICĂRII FOLOSIND INTELIGENȚA ARTIFICIALĂ

3.1 INTELIGENȚA ARTIFICIALĂ ÎN VERIFICARE

Inteligența artificială reprezintă un concept prin care se urmărește realizarea de acțiuni specifice omului cu ajutorul calculatorului. Printre sarcinile specifice algoritmilor de inteligență artificială se regăsesc clasificarea datelor, intuirea unor tendințe în evoluția acestora, recunoașterea unor repere, etc. Verificarea funcțională presupune folosirea unui număr mare de seturi de date pentru verificarea funcționalităților unui DUT. Analiza datelor, clasificarea acestora, și generarea lor astfel încât să se simuleze toate situațiile funcționale de interes ale DUT-ului reprezintă sarcini foarte potrivite pentru motoarele de inteligență artificială.

3.2 REALIZAREA DE MODELE DE REFERINȚĂ CU ROL DE CLASIFICARE

Realizarea și testarea modelelor de inteligență artificială

În cadrul acestui studiu se urmărește antrenarea unui model de inteligență artificială care să învețe corelațiile dintre parametrii de configurare și semnalele unui DUT (atât intrări cât și ieșiri). Aceste corelații sunt de folos pentru a obține informații despre funcționalitatea unui DUT doar indicând ce stimuli sunt de interes, și cum este configurat DUT-ul, fără a mai fi necesară rularea unei simulări. Deși aceste observații nu sunt suficiente pentru verificarea unui proiect digital, prin folosirea unui model de referință se pot descoperi seturile de stimuli care au capacitatea să aducă un DUT într-o anumită stare. Stimulii obținuți sunt utilizați ulterior în cadrul simulărilor pentru a avansa în procesul de verificare. Această abordare are cel puțin două avantaje foarte importante:

- Având în vedere faptul că există o tendință din ce în ce mai concretă ca producătorii de simulatoare să ofere servicii bazate pe calculul în *cloud (eng. cloud computing* - oferă posibilitatea clienților de a rula simulări de la distanță pe serverele companiei producătoare a utilitarelor de simulare) în loc să furnizeze clienților programele pe care aceștia să le instaleze pe servere locale, plata utilizării programelor se va face în funcție de timpul folosirii acestora. Așadar, cu cât se reduce numărul de simulări necesare (prin prezicerea unor seturi de stimuli care pot activa funcționalitățile dorite ale unui DUT), se va putea realiza o economie semnificativă de către companiile care realizează verificarea funcțională.
- În cadrul verificării funcționale, una din cele mai consumatoare de timp activități este rularea simulărilor. Dacă numărul acestora este diminuat, prin reducerea redundanței stimulilor, întregul proces se va desfășura mai rapid, ceea ce este un lucru extrem de dorit în mediul industrial competitiv al circuitelor integrate.

Acest deziderat se poate atinge după realizarea unei baze de date din care se pot extrage indicații exacte despre funcționarea unui DUT. Crearea unei astfel de bază de date presupune rularea unui set inițial de simulări, care are rolul de a extrage datele necesare pentru a caracteriza funcționalitățile de interes ale unui proiect digital. Ca studiu de caz este propus un transmițător-receptor (eng. *transceiver*) care preia date seriale folosind protocolul UART și le retrimite sub formă de date paralele.

S-a urmărit antrenarea unor modele care să poată prezice, în diferite situații, dacă datele sunt recepționate corect sau eronat de către monitor. Astfel, parametrii folosiți ca date de intrare sunt:

- valoarea de *baud rate* folosită de către monitor la recepționarea datelor
- deviația procentuală între viteza de *baud rate* folosită la transmitere și viteza de *baud rate* folosită la recepție
- frecvența semnalului de ceas furnizat DUT-ului și celorlalte componente de verificare
- răspunsul comparatorului de date TRANSMISIE_RX; acesta compară datele transmise cu cele recepționate și semnalează, printr-un bit, dacă ele sunt identice sau nu
- valoarea datei transmise; aceasta poate să fie egală sau nu cu data recepționată de către monitor, în funcție de parametrii menționați mai sus

În primul rând s-a avut în vedere analiza distribuției datelor folosite pentru antrenarea modelelor de inteligență artificială. După ce au fost generate date suplimentare, astfel încât valorile acestora să fie uniform distribuite, s-a trecut la antrenarea modelelor realizate conform a două abordări: prin realizarea unor rețele neurale, strat cu strat, folosind biblioteca PyTorch [23] și prin configurarea și folosirea algoritmilor de inteligență artificială disponibili în biblioteca scikit-learn[24].

Tabelul 1 Rezultatele obținute de către cele mai performante cinci modelele antrenate cu și fără valoarea datei transmise ca parametru de intrare

Se folosește data transmisă	Numele modelului-Numărul de iterații de antrenare-Rata de învățare-procentul de date de test	Timpul de antrenare(s)	Numărul de iterații	Rata de învățare	Acuratețea [%]
Nu	ANNv16-600-0.02-0.3	261.9950755	600	0.02	96.056869
	ANNv11-400-0.01-0.3	856.2789841	400	0.01	96.041159
	ANNv4-600-0.02-0.3	267.5667813	600	0.02	96.001885
	ANNv11-300-0.01-0.3	623.28969	300	0.01	96.001885
	ANNv16-400-0.01-0.3	180.027879	400	0.01	96.001885
Da	ANNv11-500-0.01-0.3	880.4907887	500	0.01	98.672532
	ANNv4-500-0.01-0.3	241.5506263	500	0.01	98.601838
	ANNv11-600-0.01-0.3	1063.711563	600	0.01	98.601838
	ANNv2-600-0.01-0.3	224.3682716	600	0.01	98.570419
	ANNv2-400-0.01-0.3	143.9144161	400	0.01	98.224806

În *Tabelul 1* se pot vedea performanțele obținute de cele mai performante modele bazate pe rețele neurale, iar în *Tabelul 2* se pot vedea rezultatele obținute de cele mai performante modele bazate pe implementările algoritmilor de inteligență artificială testați din biblioteca scikit-learn.

În ceea ce privește *Tabelul 1*, se observă că adăugarea unui nou câmp de caracterizare a datelor de intrare (valoarea datei transmise) a crescut acuratețea modelelor antrenate. Acest rezultat poate fi interpretat logic din următorul punct de vedere: chiar dacă există o diferență mai mare de viteză de *baud rate* între transmițător și receptor, dacă se transmit șiruri de biți în care nu apar multe alternanțe 0->1->0 (de exemplu, 00001111 sau 11111100), erorile de transmisie a datelor nu vor mai fi atât de vizibile. Aceasta este posibil deoarece, dacă în contul unui bit va fi eșantionată valoarea bitului vecin cu o valoare similară, eroarea din procesul de recepționare va fi "ascunsă".

Tabelul 2 Măsurători de performanță ale modelelor de învățare automată realizate folosind algoritmi din biblioteca scikit-learn

Date normalizate	Algoritmul și valoarea parametrului schimbat	Timpul de antrenare [s]	Timpul de predicție [s]	Acuratețea	Sensibilitatea	Precizia	Scorul F1	Aria delimitată de ROC
NU	Naïve Bayes -distribuție Bernoulli	0.016	0	85.9%	85.9%	86.7%	85.9%	86.2%
DA	SVM (C= 200000)	175.29	0.905	97.3%	97.3%	97.3%	97.3%	97.3%
DA	SVM (C= 500000)	358.65	0.671	97.4%	97.4%	97.4%	97.4%	97.4%
DA	Clasificatorul K Valori apropiate (K = 10)	0.078	0.640	97.8%	97.8%	97.8%	97.8%	97.8%
NU	Clasificatorul KValori apropiate (K = 20)	0.469	1.576	86.2%	86.2%	86.7%	86.2%	86.4%
DA	Clasificatorul KValori apropiate (K = 20)	0.078	0.703	97.3%	97.3%	97.3%	97.3%	97.4%
NU	Regresia Logistică (c=1,10,100,1000,10000, 100000; calcul = 'lbfgs')	0.167	0.002	54.2%	54.2%	54.0%	52.2%	53.3%
DA	Regresia Logistică (c=1,10,100,1000,10000, 100000; calcul = 'lbfgs')	0.047	0.000	92.6%	92.6%	92.6%	92.6%	92.6%
NU	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=10)	0.336	0.029	98.7%	98.7%	98.7%	98.7%	98.7%

Tabelul 2 (continuare)

DA	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=10)	0.156	0.016	98.6%	98.6%	98.6%	98.6%	98.6%
NU	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=20)	0.487	0.044	99.3%	99.3%	99.3%	99.3%	99.3%
DA	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=30)	0.172	0.016	99.3%	99.3%	99.3%	99.3%	99.3%
NU	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=40)	0.557	0.033	99.3%	99.3%	99.3%	99.3%	99.3%
DA	Clasificator de tip Colecții de Arbori Decizionali (adâncime maximă=40)	0.156	0.016	99.3%	99.3%	99.3%	99.3%	99.3%

Se observă că algoritmul reprezentând colecții de arbori decizionali a depășit în performanță toți ceilalți algoritmi atunci când a avut o adâncime de minim 8 noduri și a avut cele mai bune rezultate atunci când a avut o adâncime mai mare de 10 noduri. De asemenea, faptul că în majoritatea cazurilor metricile acuratețe, sensibilitate (eng. *recall*), precizie, scorul F1 și aria de sub graficul ROC au valori asemănătoare indică faptul că datele sunt uniform distribuite în spectrul de valori posibile.

Concluzii

În cazul de față, cea mai bună acuratețe (considerată în cazul de față o metrică de referință) a fost obținută de clasificatorul de tip Colecții de Arbori Decizionali (eng. *Random Forests*): 99,3%. În cazul modelelor bazate pe rețele neurale cea mai bună acuratețe obținută a fost de 98,67%. Totuși comparația între valori nu este făcută asupra unor situații în totalitate similare deoarece în cazul rețelelor neurale toate datele de intrare au fost normalizate, pe când atunci când la baza modelelor la au stat algoritmii implementați în biblioteca *scikit-learn*, datele de intrare nu au fost normalizate (cu excepția datelor folosite pentru antrenamentul algoritmilor SVM și NB cu distribuție polinomială). În schimb, o concluzie evidentă este faptul că modelele bazate pe rețele neurale necesită mult mai mult timp de antrenare față de modelele bazate pe clasificatorul de tip Colecții de Arbori Decizionali.

Unul din principalele avantaje ale realizării modelelor de referință ale DUT-urilor folosind învățarea supravegheată este faptul că efectul stimulilor asupra funcționalității de interes a unui circuit poate fi evaluat mai repede, urmând ca doar datele de interes să fie trimise la intrările proiectului RTL în simulare. De asemenea, extinzând cercetarea prezentată, chiar și în cazul în care modelele de referință sunt realizate pe baza unui DUT care conține erori, acestea pot fi detectate dacă datele cu care lucrează modelele de IA sunt trimise unor verificatoare (eng. *checkers*) *software*. În acest mod, mutându-se o parte din verificarea realizată prin simulare în programele care comunică cu mediile de referință, verificarea este din nou accelerată.

De asemenea, având în vedere faptul că, folosind modele de referință create folosind inteligența artificială, situațiile funcționale dorite se ating mult mai repede și, în perspectiva că producătorii de simulatoare vor muta oferirea serviciilor în *cloud*, studiul prezentat în cadrul acestui subcapitol poate contribui la diminuarea costurilor asociate folosirii infrastructurii externe de simulare.

Pe baza activităților de cercetare prezentate în cadrul acestui capitol a fost publicat articolul:

A. Dinu, G. M. Danciu, Ș. Gheorghe, "Level up in verification: learning from functional snapshots," *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*, 2021, pp. 1-4, doi: 10.1109/EMES52337.2021.9484129.

3.3 O MODALITATE ORIGINALĂ DE MODELARE A COMPORTAMENTULUI CIRCUITELOR DIGITALE FOLOSIND ÎNVĂȚAREA BAZATĂ PE RECOMPENSE

Introducere

În acest capitol se prezintă rezultatele obținute prin abordarea învățării bazate pe recompense (eng. *reinforcement learning* - ÎBR) pentru a genera secvențe de stimuli necesare în verificarea eficientă și rapidă a circuitelor integrate. Motivul principal pentru alegerea acestei abordări este faptul că soluția problemei propuse se obține în urma transmiterii la intrarea DUT-ului a unei succesiuni de stimuli. Prin urmare, acțiunile sunt ordonate cronologic, iar numărul de pași necesar până a se ajunge la soluția dorită nu este cunoscut apriori. Determinarea unui agent să învețe care stare este mai valoroasă (prin implementarea unei scheme de recompense) și cum să atingă rapid o țintă propusă este o sarcină adecvată pentru ÎBR.

Obiectivul urmărit în cadrul acestui capitol este de a demonstra modul în care poate fi creat un model software al unui proiect digital utilizând învățarea bazată pe recompense. Acest model poate fi utilizat pentru a prezice stimulii necesari pentru aducerea rapidă a DUT-ului într-o starea dorită, pe parcursul simulărilor.

Conceptul de lucru

În cadrul acestui capitol se prezintă implementarea unor agenți de ÎBR care aleg valorile care vor fi transmise operandului B și determină operatorul folosit de o unitate aritmetică logică (UAL) la fiecare operație, pentru ca aceasta să genereze rezultatele dorite. UAL folosită în cadrul acestui studiu lucrează numai cu numere întregi. Valoarea pentru primul operand al UAL (operandul A), cu excepția primei iterații când este stabilită aleatoriu, copiază valoarea rezultatului operației anterioare efectuate de UAL. Valoarea operandului A poate avea un număr de biți parametrizabil dar, pentru ca simulările să se termine mai repede, în majoritatea situațiilor din lucrarea curentă operandul A este declarat pe 8 biți. Valoarea operandului B poate să fie egală întotdeauna cu "2", sau poate fi modificată luând valorile "2", "3" sau "4", în funcție de acțiunile întreprinse de agentul ÎBR. UAL poate efectua cele patru operații matematice de bază: adunare, scădere, înmulțire și împărțire (se ia în considerare doar partea întreagă a rezultatului). Dacă, la adunare și înmulțire, rezultatul depășește valoarea maximă, se păstrează numai cei 8 biți mai puțin semnificativi (LSb) ai rezultatului (MSb se pierde) și se activează ieșirea "Depășire". Dacă, la scădere, se obține o valoare negativă, se stochează numai cei 8 LSb, iar ieșirea "Depășire" este de asemenea activată. Pe de altă parte, în timpul antrenării agentului ÎBR, ieșirea "Depășire" nu este luată în considerare. Astfel, DUT-ul obținut nu efectuează operații

matematice adevărate, dar are o funcționalitate bine definită. UAL descrisă mai sus a fost reprezentată în Fig. 2.

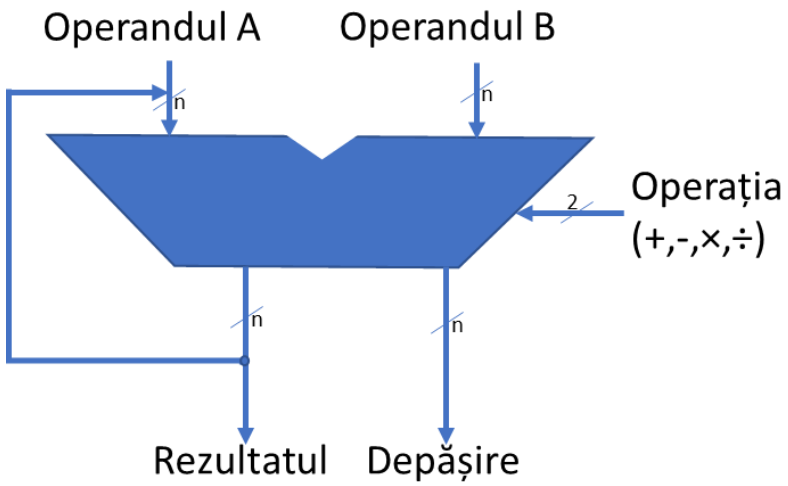

Fig. 2 Structura UAL utilizată ca DUT cadrul acestui capitol

Pentru a trece de la o stare la alta, agentul ÎBR trebuie să furnizeze proiectului digital sau modelului de referință (realizat în software) al DUT-ului valoarea operandului B și codul operației care trebuie efectuată.

Implementarea mecanismului de antrenare

Ecosistemul creat pentru a folosi metoda de ÎBR în automatizarea generării de stimuli

Ideea de bază a lucrului prezentat în cadrul acestui capitol este de a crea o comunicare lucrativă între mediul de simulare care înglobează funcționarea DUT-ului și agentul de ÎBR care trebuie să învețe cum să controleze DUT-ul pentru a atinge obiectivele stabilite. La nivel de concept, această comunicare este realizată după cum se vede în Fig. 3, unde caseta albastră reprezintă mediul de simulare al DUT-ului, iar casetele verzi reprezintă componentele dezvoltate în Python. Agentul folosit în ÎBR furnizează DUT-ului stimulii care îi vor determina următorul pas al funcționării sale. Acești stimuli caracterizează "acțiunea" pe care DUT-ul trebuie să o îndeplinească. După fiecare execuție a testului pe baza stimulilor primiți de la agent, ecosistemul software preia valoarea rezultatului operației din raportul generat. Acest rezultat reprezintă următoarea stare în care ajunge agentul. Pe baza depărtării logice (depărtarea logică între două stări este definită ca numărul minim de operații necesar cunoscut pentru a ajunge dintr-o stare în alta) dintre această stare și ținta stabilită, mediul Python calculează "recompensa" pe care o primește agentul pentru ultima acțiune efectuată. Astfel, agentul învață care este efectul acțiunilor sale și care sunt cele mai convenabile acțiuni, având în vedere starea actuală.

În faza de inferență, agentul deja antrenat este capabil să selecteze cea mai scurtă cale care îl aduce din starea inițială în starea țintă.

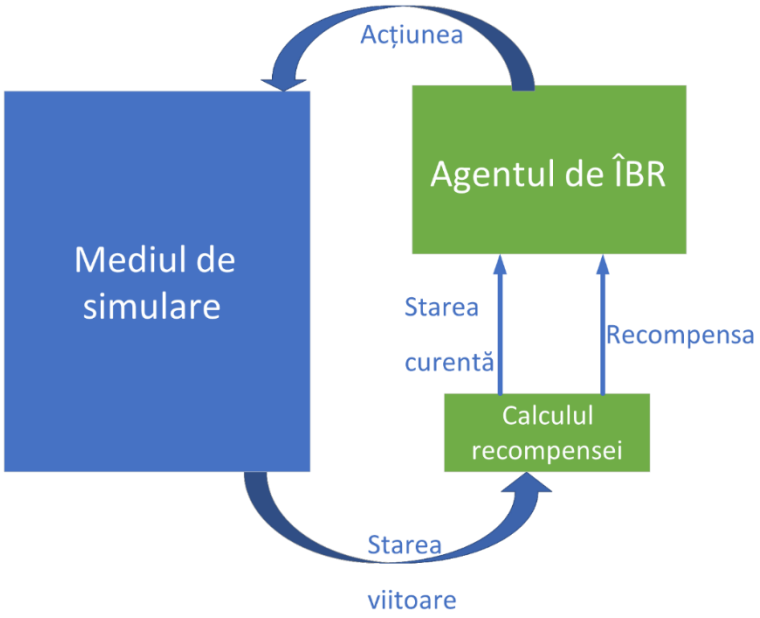

Fig. 3 Comunicarea dintre agent și DUT atât în faza de antrenare, cât și în cea de inferență.

Procesul de antrenare

Procesul de antrenare reprezintă pasul necesar pentru a conferi modelului de ÎBR o imagine corectă asupra comportamentului DUT-ului. Acesta depinde de câțiva parametri importanți care influențează în mod direct performanța procesului de antrenare prin învățarea bazată pe recompense: numărul de iterații de antrenare, schema de recompensare, coeficientul de învățare și factorul de actualizare.

Pe parcursul activităților de cercetare, au fost modificați în mod constant atât parametrii generali menționați mai sus, cât și alți parametri specifici pentru algoritmii utilizați, pentru a se găsi cele mai bune configurații de antrenare a DUT-urilor abordate.

Următorul pas după antrenarea unui model este folosirea acestuia pentru a atinge obiectivul pentru care a fost antrenat (în acest caz, obiectivul fiind reprezentat de obținerea unui rezultat dorit). Procesul de exploatare al agentului antrenat poartă numele de inferență. În cadrul etapei de inferență, s-au folosit trei abordări: inferența cu Marcarea Stărilor Vizitate (MSV), inferența fără MSV și inferența cu Marcarea Extinsă a Stărilor Vizitate (MESV).

După ce se stabilește contextul general de antrenare (se creează o nouă instanță a mediului în care evoluează agentul de ÎBR), un agent evoluează pe parcursul a mai multe iterații. La fiecare iterație, agentul pornește de la o stare aleasă aleatoriu și întreprinde acțiuni succesive până când ajunge la una dintre stările terminale. În timpul deplasării de la o stare la alta, acesta înregistrează cât de bine a progresat (având în vedere recompensa primită pentru fiecare acțiune) și actualizează un vector de *ponderi* cu valori pentru fiecare stare (în cazul algoritmului SGTD(0)) sau actualizează o matrice de *ponderi* cu valori pentru fiecare pereche stare-acțiune (în cazul algoritmului Q-learning).

Procesul de inferență

Inferența presupune a oferi unui agent deja antrenat posibilitatea de a recrea o situație întâlnită în procesul de antrenare sau de a crea o situație nouă, reunind mai multe experiențe similare din trecut. În sens larg, inferența este utilizată pentru a face predicții asupra datelor, pe baza parametrilor deja

învățați [25]. În cazul învățării prin întărire, inferența este etapa în care un agent de ÎBR deja format va ajunge la ținta pentru care a fost antrenat într-un mod eficient, independent de starea sa inițială, deoarece a învățat care sunt acțiunile care, în raport cu starea agentului la un moment dat, sunt cele mai favorabile pentru apropierea de țintă.

Pașii generali ai procesului de inferență utilizat în cadrul acestui studiu pot fi văzuți în caseta de text de mai jos.

Conceptul de inferență

1. se preia în mediul software modelul deja antrenat
2. se instanțiază mediul care îi permite agentului să își realizeze acțiunile
3. se aduce agentul într-o stare inițială
4. se inițializează contorul de pași cu 0 (numărul de pași necesar pentru a ajunge în starea finală reprezintă o metrică pentru performanța agentului)
5. până când se atinge starea terminală dorită:
    a. se selectează o nouă acțiune în conformitate cu politica agentului dobândită de acesta în urma antrenamentului
    b. se actualizează starea agentului în funcție de ultima sa acțiune
    c. se incrementează valoarea numărului de pași efectuați

Rezultate

Algoritmul testat în cadrul acestui studiu este SGTD(0) (forma cu semi-gradient a metodei diferenței temporale). În continuare, dacă nu se specifică o altă regulă, toate procesele de antrenare ale căror rezultate sunt prezentate în continuare au considerat ca stări terminale dorite valorile "61", "62" sau "63" și ca stările terminale periculoase valorile "1", "2" și "3".

În cadrul proiectului actual au fost utilizate mai multe scheme de recompensă pentru a afla care configurație oferă cele mai bune rezultate. *Tabelul 3* prezintă atât performanțele obținute prin utilizarea fiecărei scheme de recompensă, cât și superioritatea ÎBR față de situația în care datele de intrare ale UAL se generează în mod aleatoriu (această situați fiind exponentul modalității clasice de generare a stimulilor din verificarea funcțională).

Uneori este posibil ca procesul de antrenare să nu reușească să creeze agenți de înaltă calitate. Pentru a recupera acești agenți, am introdus mecanismul de marcare a stărilor vizitate (MSV), o abordare similară fiind utilizată și în implementări ale altor algoritmi cum ar fi [26-28]. Mecanismul MSV presupune ca, pe parcursul procedeului de inferență, de fiecare dată când o stare este vizitată, valoarea sa să fie actualizată la "-100", ceea ce înseamnă că agentul nu ar trebui să se mai întoarcă niciodată în acea stare.

Se știe, de asemenea, că în cadrul procesului de antrenare, stările dintr-un grup au avut aceeași pondere, deci au avut o anumită valoare. Ideea că stările consecutive au aceeași valoare poate fi utilizată și la inferență. Prin urmare, în cazul în care o stare (diferită de cea finală) a fost deja vizitată, se poate presupune că starea finală nu este încă apropiată, iar vecinii apropiați ai stării vizitate

Tabelul 3 Comparație între agenții antrenați folosind diferite scheme de recompensă (în cadrul proceselor de inferență pentru agenții ÎBR s-a folosit și abordarea MSV, descrisă mai jos)

	Numărul de pași necesari pentru a atinge ținta pentru agenții antrenați cu:				
starea de pornire	prima schemă de recompensă	a doua schemă de recompensă	a treia schemă de recompensă	operațiune aleatorie la fiecare pas	doar adunare sau scădere
248	5	55	5	141	120
34	4	21	4	303	15
144	12	28	3	92	26
50	12	19	9	114	37
0	3	13	18	136	20
90	10	>256	10	102	13
45	11	19	11	35	78
255	2	>256	2	218	3
15	6	25	6	56	66
69	17	22	7	195	21
199	10	29	6	105	37
Numărul mediu de pași:	8.36	mare	7.36	136.09	39.64

(adăugând și scăzând -1 din valoarea stării vizitate) pot primi, de asemenea, valoarea "-100" pentru a evita vizitarea lor în viitor în cadrul procesului de inferență curent.

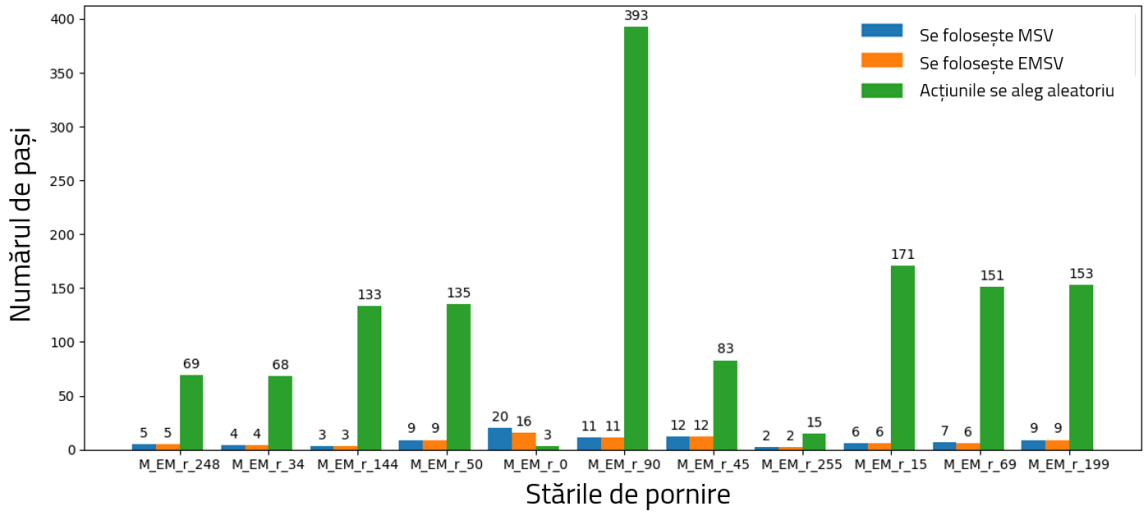

Fig. 4 Rezultatele proceselor de inferență asupra unui model de agent antrenat folosind a treia schemă de recompensare, în care mecanismele MSV și MESV au fost aplicate pe rând, fiind totodată comparate cu abordarea care presupune alegerea aleatorie a acțiunilor până la atingerea stării terminale dorite. Ultimul număr din descrierea fiecărui grup de bare reprezintă starea inițială

Această abordare este denumită mecanismul extins de marcare a stărilor vizitate (MESV) și se aplică atunci când stările vecine care vor fi marcate ca fiind vizitate nu fac parte din stările terminale dorite.

Rezultatele aplicării metodei MSV, a metodei MESV și a metodei aleatorii pentru selectarea acțiunilor în procesul de inferență sunt prezentate în *Fig. 4*. Se remarcă faptul că atunci când starea de pornire a fost egală cu 0 și 69, MESV a depășit mecanismul MSV. De asemenea, nu există nicio situație în care MSV să fi avut rezultate mai bune decât MESV.

Pentru compararea performanțelor agenților obținuți, s-au parcurs o serie de pași. La etapa de inferență, agenții sunt plasați în toate stările mediului în care acesta evoluează (în acest caz, acesta este o cerință rezonabilă, deoarece există doar 256 de stări) și se numără pașii necesari pentru ca agentul să ajungă la starea finală dorită pentru fiecare situație în parte. În *Fig. 5* se poate observa că agenții 2, 3 și 4, care au fost configurați în mod similar, au obținut rezultate mai bune decât ceilalți agenți. Agentul 8 a obținut cea mai slabă performanță, deoarece de cele mai multe ori nu a reușit să atingă starea terminală dorită, atunci când a pornit de la celelalte stări (majoritatea proceselor sale de inferență sunt marcate cu 400 de pași).

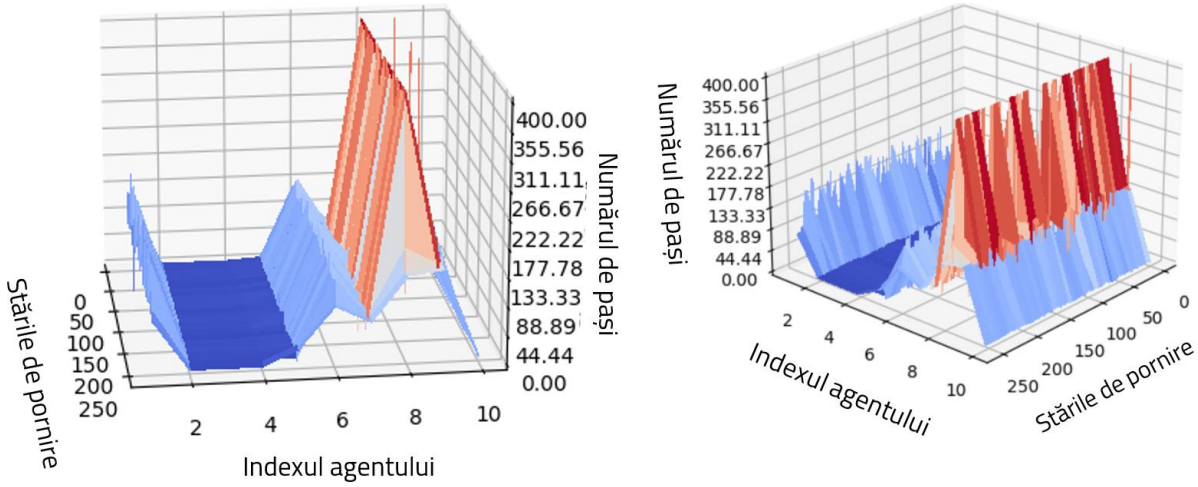

Fig. 5 Grafic care indică performanța agenților antrenați (același grafic este observat din două poziții)

Concluzii

În acest capitol, au fost antrenați agenți de ÎBR pentru ca aceștia să învețe ce intrări trebuie să dea DUT-ului pentru ca acesta să se apropie de starea țintă. S-a dovedit că folosirea ÎBR poate conduce la generarea unor secvențe de stimuli care pot aduce DUT-ul într-o stare prestabilită mai repede decât folosind acțiuni aleatorii. Spre deosebire de generarea aleatorie a stimulilor, generarea stimulilor folosind un agent bine antrenat a dus la scăderea cu 2,2 ori a numărului mediu de pași necesari pentru a atinge una din stările terminale ale DUT-ului atunci când mediul a conținut 4096 de stări. În cazul unui mediu având 256 de stări, generarea aleatorie a datelor a obținut un număr mediu de 76,36 de pași pornind din 12 stări diferite, iar cel mai bun agent antrenat a obținut un număr mediu de 4,18 pași atunci când a pornit din aceleași 12 stări. Astfel, s-a obținut o creștere a performanței de peste 18 ori, atunci când agenții de ÎBR au fost utilizați.

Pe baza activităților de cercetare prezentate în cadrul acestui capitol a fost publicat articolul:

G. Ștefan and D. Alexandru, "Controlling hardware design behavior using Python based machine learning algorithms," *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*, 2021, pp. 1-4, doi: 10.1109/EMES52337.2021.9484105.

3.4 UTILIZAREA ALGORITMILOR GENETICI PENTRU AUTOMATIZAREA PROCESULUI DE ACOPERIRE FUNCȚIONALĂ

Introducere

Acoperirea funcțională este cel mai important factor care indică progresul procesului de verificare funcțională a unui circuit. În multe cazuri, este dificilă stabilirea stimulilor de intrare necesari pentru a accesa un anumit comportament al DUT-ului. Prin urmare, se caută modalități de automatizare a generării de stimuli pentru a atinge toate funcționalitățile de interes cu mai puțin efort uman și într-un timp mai scurt. Având în vedere literatura de specialitate în acest domeniu, cea mai avantajoasă modalitate de automatizare a generării stimulilor în scopul creșterii gradului de acoperire funcțională în verificarea codului RTL al unui proiect digital este utilizarea algoritmilor genetici.

În cadrul acestui capitol, se prezintă mai multe abordări bazate pe algoritmi genetici. Acestea sunt testate în timpul procesului de verificare a trei module realizate în limbajul Verilog. Pentru aceasta, s-a dezvoltat un sistem software care facilitează comunicarea între algoritmii genetici și simulatorul Modelsim®. Algoritmii genetici generează stimuli care sunt folosiți în simulare. Apoi, din rapoartele de simulare sunt extrase valorile de acoperire obținute care sunt apoi analizate, ordonate și sortate astfel încât să se formeze noi stimuli, mai performanți. Inginerilor de verificare le rămân, în principal, sarcinile de a configura algoritmii, de a determina condiția de oprire a acestora și de a alege modul de aplicare a procesului de combinare încrucișată și al mutației.

Pentru a evalua abordările dezvoltate în cadrul cercetării curente, sunt utilizate patru proiecte digitale (DUT-uri) cu care se comunică prin intermediul mai multor medii de simulare construite corespunzător obiectivelor de atins.

Infrastructura de antrenare

Prin intermediul limbajului Python s-a realizat un program prin care algoritmii genetici pot furniza stimuli DUT-ului. Informațiile relevante din rapoartele de simulare sunt citite, de către acești algoritmi, realizându-se astfel o buclă de reacție negativă vizibilă și în *Fig. 6*.

Blocurile colorate în albastru reprezintă funcționalități ale codului realizat în limbajul Python. Programul începe cu generarea unei populații inițiale de stimuli de intrare. Fiecare grup de stimuli este evaluat fiind trimis către mediul de simulare, sub formă de fișiere text cu conținut standard. Mediul de simulare încorporează mediul de verificare, DUT-ul și alte structuri necesare pentru efectuarea simulărilor. Procesul care comandă funcționarea ecosistemului software inițiază o simulare de fiecare dată când informațiile din fișierele text de intrare sunt înlocuite cu noi stimuli de către un generator de date a cărui funcționare se bazează pe algoritmi genetici. După fiecare simulare, valoarea de acoperire funcțională este corelată cu setul de stimuli care a generat-o. Dacă valoarea de acoperire este mai mare decât maximul obținut până atunci, setul de stimuli care a generat-o este considerat mai performant.

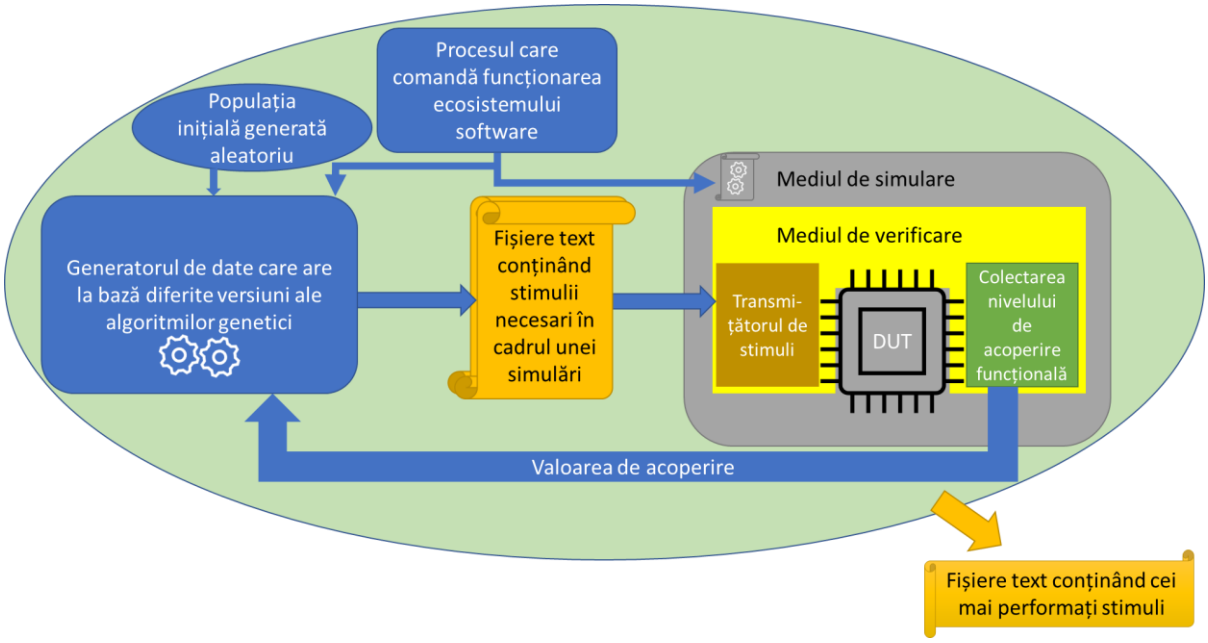

Fig. 6 Schema ecosistemului de programare proiectat pentru automatizarea procesului de verificare funcțională folosind algoritmi genetici

Descrierea ecosistemului de programare dezvoltat bazat pe algoritmi genetici

Pe baza Algoritmului Genetic Simplu (eng. *Simple Genetic Algorithm - SGA*) abordat în [6] au fost dezvoltați pașii din *Fig. 7* care reprezintă etapele parcurse în cadrul algoritmilor genetici dezvoltați în cadrul acestei cercetări.

În fiecare generație, se calculează scorurile indivizilor care o compun, pentru a stabili care dintre ei vor deveni părinți (pasul 2). Aceasta înseamnă că pentru fiecare individ este inițiată o simulare ModelSim®, în care DUT-ul primește stimulii corespunzători conținutului acestuia. La sfârșitul simulării, valoarea de acoperire obținută este asociată cu individul care a generat datele pentru rularea simulării.

De asemenea, pentru a evalua performanța abordărilor dezvoltate cu privire la unul dintre DUT-uri (care constă în administrarea unei secțiuni de cale ferată dintr-o grădină zoologică), obiectivul principal de acoperire a fost abordat, de asemenea, folosind doi dintre cei mai populari și mai utilizați algoritmi genetici [29]: Versiunea a doua a algoritmul genetic cu sortare ne-dominată II (eng. *Non-dominated Sorting Genetic Algorithm II* - NSGA-II) și Algoritmul evolutiv cu indicatori de dominare 2 (eng. *Strength Pareto Evolutionary Algorithm 2* - SPEA2).

Abordările bazate pe algoritmi genetici prezentate în lucrarea de față (numerotate în formatul V2.3, V3.1, etc pentru o evidență mai bună a acestora) au fost testate pe mai multe obiective de acoperire aferente DUT-urilor avute în vedere, pentru a se verifica performanța acestora în diferite situații. S-au utilizat mai mulți indici pentru a compara performanța abordărilor utilizate: numărul de ordine (indexul) primului individ (primul set de stimuli) care a obținut valoarea maximă de acoperire, indexul generației (iterației) în care cel puțin 50% din numărul de indivizi ai populației inițiale (n_pop) au obținut valoarea maximă de acoperire și indicele generației în care un număr de indivizi egal cu n_pop au obținut valoarea maximă de acoperire.

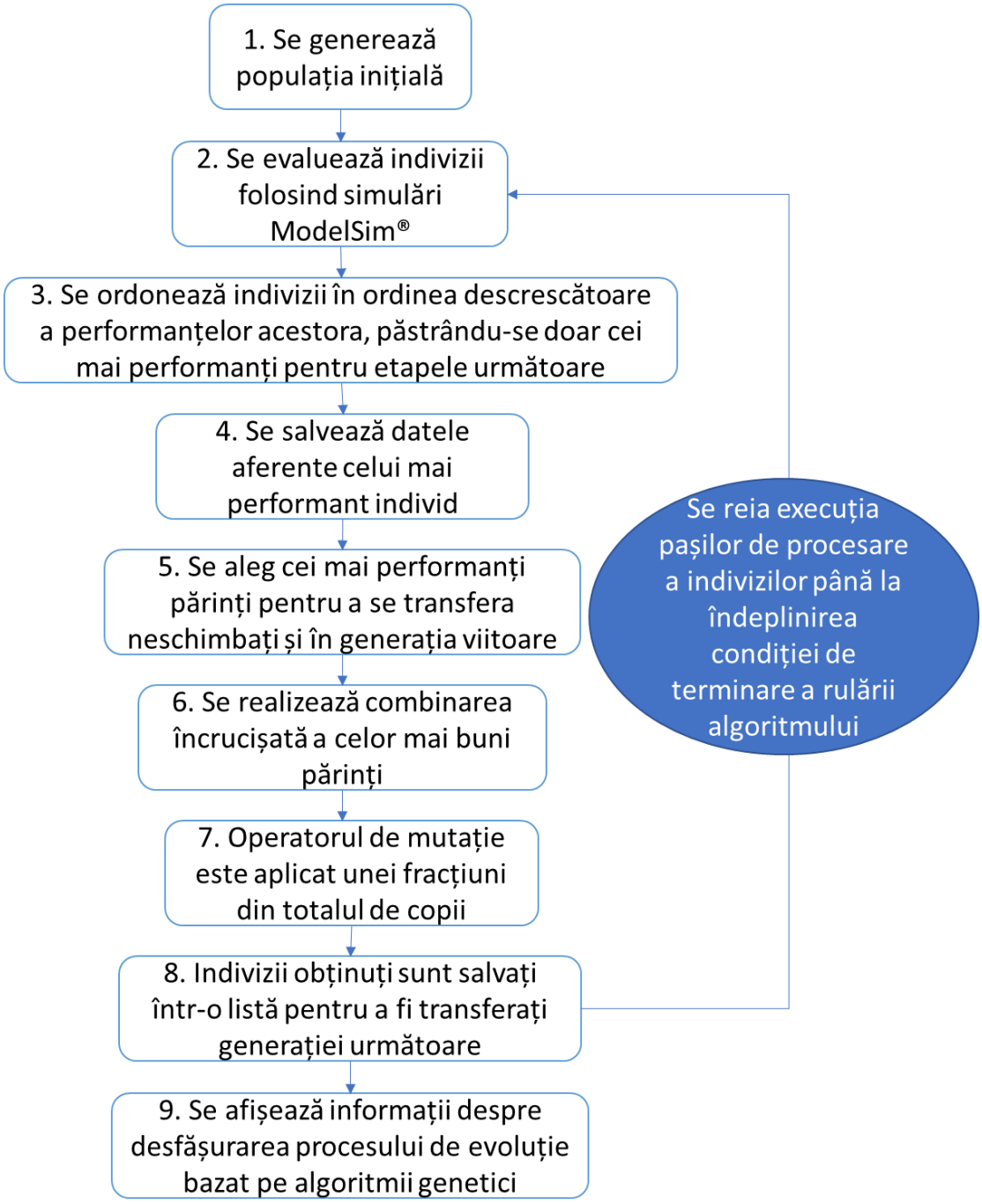

Fig. 7 Diagrama logică a versiunii de algoritmi genetici implementată în cadrul studiului curent

Rezultate

Tabelul 4 conține rezultatele obținute atunci când s-a urmărit acoperirea semnalului care vehiculează valori de luminozitate unui regulator integrat într-o lampă inteligentă. Pentru a evalua oportunitatea aplicării metodelor bazate pe algoritmi genetici, s-au rulat și 400 de simulări care au folosit stimuli generați constrâns-aleatoriu (acestea reprezentând exponentul abordării clasice din metodologiile de verificare). Se poate observa că majoritatea abordărilor bazate pe algoritmii genetici au depășit scorurile obținute prin simulări aleatorii.

Tabelul 4 Rezultatele obținute atunci când regulatorul lămpii inteligente a fost stimulat cu valori furnizate de programe bazate pe algoritmi genetici

Criteriul	Câmpul de valori al semnalului de luminozitate a fost împărțit în									
	15 intervale					20 intervale				
	V3.2	V3.3	V3.4	V.3.5	stimuli aleatorii	V3.2	V3.3	V3.3	V3.5	stimuli aleatorii
acoperire maximă atinsă	100	100	100	100	93.3	90	95	100	85	85
părinți/generație	20	20	20	20	400	20	20	100	20	400
prima iterație care conține un set de date care a obținut o acoperire maximă	12	4	9	5	1	7	18	9	4	1
prima generație în care s-au obținut cel puțin n_pop/2 elemente cu acoperire maximă	n/a	15	22	13	n/a	n/a	28	21	28	n/a
numărul de seturi de stimuli care au atins o acoperire maximă	1	144 (2)	60(1)	11(1)	7	2 (1)	43 (2)	173 (2)	17(4)	3

(1) toate seturile de date au fost obținute prin mutarea aceluiași set de date de stimuli

(2) stimulii generați pot fi grupați în două mari categorii pe baza similitudinilor dintre valorile lor

(3) stimulii generați pot fi grupați în trei mari categorii pe baza similitudinilor dintre valorile lor

(4) stimulii generați pot fi grupați în patru mari categorii pe baza similitudinilor dintre valorile lor

În *Fig. 8* poate fi observată evoluția nivelului maxim de acoperire atins de fiecare generație din cadrul a două procese de antrenare distincte realizate folosind abordarea V3.3. Aceste imagine arată comportamentul specific algoritmilor genetici: prin efectuarea de combinații multiple între indivizi, se obține o valoare de acoperire din ce în ce mai mare(într-o situație general reprezentativă, pe măsură ce se succed iterațiile de antrenare, se obțin indivizi cu un rezultat mai mare al funcției de evaluare a gradului de atingere a obiectivului – eng. *fitness function*) . Indivizii care au obținut scoruri ridicate sunt transferați în generația următoare și, de asemenea, sunt utilizați în procesul de combinare încrucișată, cu scopul de a genera alți indivizi și mai performanți. Atunci când algoritmii genetici beneficiază de un proces de antrenare corespunzător cu dificultatea obiectivului de îndeplinit, indivizii evoluează pe parcursul mai multor generații până când se atinge valoarea de acoperire maximă posibilă. În cazul de față, abordarea V3.3 a obținut cel mai mare scor (acoperire de 95%), atunci când populația inițială a avut doar 20 de indivizi. De asemenea, cu portocaliu este reprezentat numărul de secvențe care pot atinge nivelul maxim de acoperire la un moment dat.

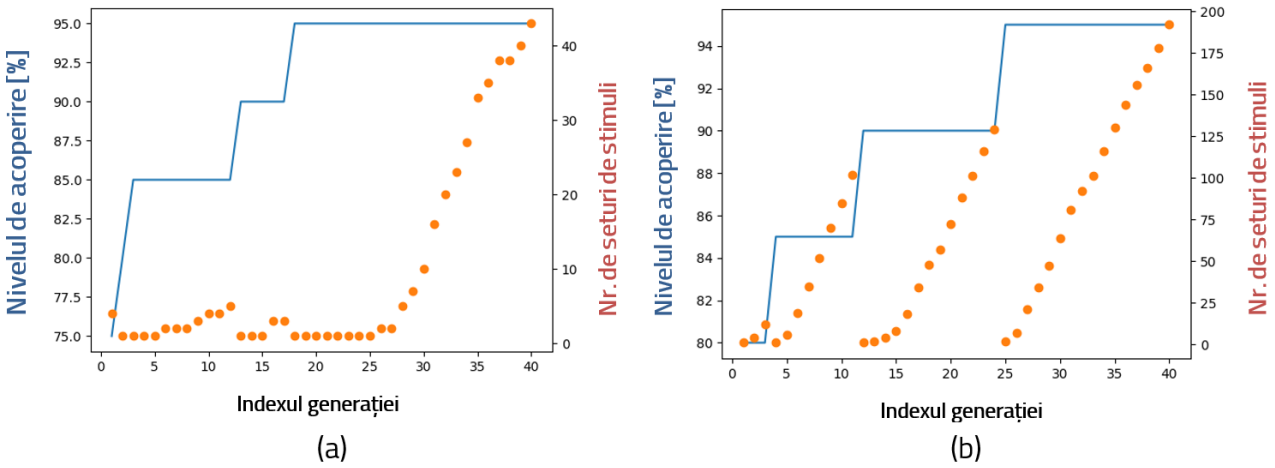

Fig. 8 Două exemple de evoluție a unei populații de 20 de indivizi de-a lungul a 40 de generații în cazul abordării V3.3. (a) reprezintă descrierea vizuală a datelor notate în Tabelul 4; (b) reprezintă un alt exemplu de evoluție obținut folosind abordarea V3.3.

În altă situație s-a avut în vedere semnalul pe 10 biți prin care un regulator ambiental primește valori de la un senzor de luminozitate. Valorile legale ale luminozității sunt cuprinse între 0 și 900. Doar abordările V3.2 și V3.3 prezentate anterior au fost testate pentru DUT-ul curent, deoarece aceste versiuni au oferit cele mai bune rezultate și în cazul regulatorului pentru lampa inteligentă. Rezultatele obținute au fost notate în *Tabelul 5*.

Tabelul 5 Rezultatele obținute în cadrul testării abordărilor bazate pe algoritmi genetici care au avut ca obiectiv atingerea valorii de acoperire de 100% pentru semnalul de luminozitate al regulatorului ambiental

Criteriul	15 intervale			20 intervale			
	V3.2	V3.3	stimuli aleatorii	V3.2	V3.3	V3.2	stimuli aleatorii
acoperire maximă atinsă	100	100	93.3	90	100	95	85
părinți/generație	20	20	400	20	20	200	400
prima iterație care conține un set de date care conduce la o acoperire maximă	12	9	1	13	29	3	1
prima generație în care s-au obținut cel puțin n_pop/2 elemente cu acoperire maximă	23	14	n/a	n/a	35	n/a	n/a
numărul de seturi de stimuli pentru a obține o acoperire maximă	10(1)	128(1)	15	1	30(1)	2	1

(1) toate seturile de date au fost obținute prin mutarea aceluiași set de date de stimuli

După cum poate fi văzut în *Tabelul 5*, abordarea V3.3 a obținut din nou cele mai bune rezultate în ceea ce privește valoarea maximă atinsă a acoperirii. În cazul simulărilor aleatorii, nici una din cele 400 de secvențe de stimuli care au fost transmise către DUT, nu a atins nivelul maxim de acoperire atins de abordările bazate pe algoritmi genetici pentru aceeași categorie. Ca o performanță neobișnuită, V3.3 a reușit chiar să atingă obiectivul de acoperire de 100 %, atunci când valoarea rezultatului a fost împărțită în 20 de intervale: aceasta înseamnă că fiecare stimul din secvențele respective de 20 de elemente a atins un interval de valori diferit.

Dacă exemplul anterior de automatizare a creșterii acoperirii funcționale s-au avut în vedere valorile unor semnale, în acest caz sunt de interes stările de funcționare în care ajunge un DUT proiectat pentru administrarea accesului a șase trenuri pe un tronson de cale ferată cu o singură linie.

Performanța obținută de fiecare abordare testată poate fi observată în *Tabelul 6*. Rezultatele obținute cu ajutorul abordărilor dezvoltate, care fac obiectul prezentei lucrări, sunt comparate cu rezultatele obținute de algoritmii NSGA-II și SPEA2.

Tabelul 6 Comparație între abordările de automatizare a creșterii valorii de acoperire bazate pe algoritmi genetici și o secvență de 800 de seturi de stimuli generați aleatoriu

Criteriul	**Abordări propuse**				**Abordarea clasică**	**Algoritmi genetici consacrați**				
	V3.2	**V3.21**	**V3.3**	**V3.6**	**stimuli aleatorii**	**NSGA-II clasic [30]**	**NSGA-II clasic-modificat [31]**	**NSGA-II modifi-cat [31]**	**SPEA2[2] [32] I**	**II**
acoperire maximă atinsă	100	100	100	100	100	100	100	100	100	100
părinți / generație	20	20	20	20	800	20	20	20	20	20
prima iterație care conține un set de date care conduce la o acoperire maximă	4	5	2	5	1	15	2	9	6	3
timpul scurs până când se ajunge la primul set de date cu performanțe ridicate [minute]	6m	6m	4m	11m	38m	24m	3m	12m	21m	11m
prima generație în care s-au obținut cel puțin n_pop[1]/2 elemente cu acoperire maximă	12	12	11	13	n/a	27	n/a	n/a	11	15
timpul scurs până când s-au obținut n_pop/2 elemente cu acoperire maximă [minute]	19m	17m	24m	26m	n/a	44m	n/a	n/a	38m	53m
numărul de seturi de stimuli care asigură o acoperire maximă[3]	49(1)	176(1)	146(1)	205 (1)	1	45(1)	3(3)	2(2)	285(2)	212(3)

[1] *n_pop* reprezintă numărul inițial de indivizi pe populație și este egal cu 20 în implementările curente.

[2] în cazul algoritmului SPEA2, același algoritm a fost rulat de două ori și, din cauza caracterului aleatoriu al procesului, s-au obținut rezultate diferite.

[3] în paranteze sunt notate numărul de grupuri de secvențe de stimuli care pot fi formate pe baza similitudinii indivizilor;

Se observă faptul că toate abordările dezvoltate în cadrul cercetării actuale au generat cel puțin o secvență care a dus la atingerea valorii maxime de acoperire, nu mai târziu de a cincea iterație.

Concluzii

Algoritmii genetici oferă rezultate favorabile dacă parametrii lor sunt setați în mod corespunzător. Există două moduri de a înțelege care ar trebui să fie valorile corespunzătoare ale parametrilor pentru un caz specific de utilizare a abordărilor bazate pe algoritmi genetici. Primul constă în consultarea literaturii de specialitate cu privire la valorilor utilizate în mod obișnuit în practică. Prin consultarea experienței altor ingineri de verificare sau a cercetătorilor universitari se creează o bază solidă. Apoi, odată ce sunt găsite mai multe valori adecvate pentru parametrii de configurare, acestea trebuie testate pe cazul specific de verificare care trebuie abordat. În acest fel, se va selecta cel mai eficient algoritm pentru a crește performanța procesului de verificare funcțională din mai multe abordări bine configurate. În cercetarea de față s-au abordat aceste două etape, în primul rând prin consultarea altor inițiative conexe din industrie și prin utilizarea experienței altor ingineri de verificare (de exemplu, utilizarea valorii de 5% pentru pragul ratei de mutație) și, în al doilea rând, prin configurarea infrastructurii create cu ajutorul algoritmilor genetici (Fig. 7) în mai multe moduri (ex. Tabelul 6).

Un mare avantaj al utilizării algoritmilor genetici este că se pot obține mai multe seturi de stimuli care ating acoperirea maximă, acestea fiind derivate din seturile de stimuli performante deja create.

O contribuție importantă prezentată în cadrul acestui capitol este realizarea unui sistem complet de comunicare dintre un regulator de proces software și un emulator hardware (reprezentat de mediul de simulare). Programul construit cu ajutorul limbajului Python urmează mai mulți pași ușor de înțeles care oferă un mecanism eficient pentru automatizarea acoperirii funcționale în comparație cu verificarea clasică bazată pe generarea constrâns-aleatorie. Sistemul proiectat poate fi dezvoltat în continuare și utilizat în alte abordări de automatizare a verificării funcționale.

Pentru fiecare proiect asupra căruia s-au testat metodele propuse în cadrul acestui capitol, generarea stimulilor prin intermediul algoritmilor genetici au redus timpul de atingere a valorii de 100% pentru acoperirea funcțională față de cazurile când au fost utilizate generări aleatorii ale stimulilor. De asemenea, folosind algoritmii genetici, s-a crescut considerabil procentul de secvențe de intrare care ating 100% acoperire funcțională în raport cu modalitatea de generare constrâns-aleatorie a datelor de intrare. Rezultatele acestor implementări originale confirmă faptul că algoritmii genetici pot depăși generarea constrâns-aleatorie a stimulilor, care este o componentă importantă a modului clasic de verificare funcțională.

Pe baza activităților de cercetare prezentate în cadrul acestui capitol au fost publicate articolele:

A. Dinu, G.M. Danciu, P.L. Ogrutan. Cost-Efficient Approaches for Fulfillment of Functional Coverage during Verification of Digital Designs. *Micromachines* **2022**, *13*, 691. https://doi.org/10.3390/mi13050691
și

G. M. Danciu, A. Dinu. Coverage Fulfillment Automation in Hardware Functional Verification Using Genetic Algorithms. *Appl. Sci.* **2022**, *12*, 1559. https://doi.org/10.3390/app12031559

4 VALIDAREA SISTEMELOR DIGITALE

Întrucât în momentul de față circuitele sistemelor digitale conțin un număr din ce în ce mai mare de porți logice, iar funcționarea lor este din ce în ce mai complexă, devine indispensabilă și folosirea altor tehnici avansate de evaluare a acestora (de exemplu folosirea emulatoarelor și a prototipării pe FPGA) în completarea efortului depus în cadrul verificării funcționale [22].

4.1 MOTIVAȚIE ȘI PREZENTARE SUCCINTĂ A CONȚINUTULUI

În industria sistemelor digitale, validarea este prima verificare "fizică" a conceptului unui DUT. Astfel, validarea pe FPGA este folosită concomitent cu verificarea funcțională a DUT-ului, și are rolul atât de a descoperi eventualele probleme din codul RTL, cât și de a proba faptul că funcționalitatea cipului este realist proiectată și partea mecanică (perifericele cipului sau elemente controlate de cip) funcționează corespunzător specificației.

Cu toate acestea, descoperirea surselor erorilor atunci când modulul de pe FPGA nu funcționează conform specificațiilor reprezintă de multe ori o mare provocare. Faptul că se descoperă o problemă, dar accesul la semnalele interne este deosebit de limitat (putând fi citite valorile acestora doar folosind pini exteriori sau analizoare logice) face ca adevărata cauză a erorii descoperite să necesite eforturi deosebite pentru a fi complet descrisă.

În acest capitol se descrie implementarea unei succesiuni de operații prin care se poate realiza un model de referință care să fie folosit în procesul de semnalare a erorilor care se găsesc în proiectele folosite pentru a configura dispozitivele FPGA. Mai întâi a fost creat un generator de date cuprinzând un registru de deplasare cu reacție lineară (eng. *Linear Feedback Shift Register* – LFSR), folosind limbajul VHDL. Pentru a se asigura că acesta funcționează în mod optim, generatorul de date a fost testat în simulare, înainte de a fi încărcat pe placa cu FPGA, Spartan 3E. Datele generate prin intermediul modulului LFSR implementat pe FPGA au fost grupate în perechi și operațiunea de vot majoritar a fost aplicată pentru fiecare trei perechi succesive. Mai departe, datele au fost trimise de pe placă spre exterior prin intermediul protocolului UART, folosindu-se un adaptor de tip FT232R. Informațiile transmise au fost recepționate pe calculator și analizate pentru a recompune informațiile și a elimina datele transmise eronat.

În continuare, s-a analizat modul de distribuire al datelor (care au fost stocate sub formă tabelară în fișiere cu extensia *.csv*) în cadrul intervalelor de valori posibile. Constatându-se că valorile acestora urmează anumite tipare de generare, ordinea datelor a fost schimbată și, în acest mod, s-a uniformizat distribuirea datelor (această condiție fiind necesară în vederea unei antrenări eficiente a modelelor de inteligență artificială). Apoi, folosind datele obținute (care conțineau șase valori generate de algoritmul LFSR și rezultatul algoritmului de vot majoritar aferent acestora) și stocate în fișiere tabelare, au fost antrenate atât modele de învățare automată (eng. *machine learning* - ML) cât

și modele de învățare profundă (eng. *deep learning* - DL) pentru a se obține modele de referință ale funcționalității proiectului integrat pe FPGA care a constat în realizarea operației de vor majoritar prin compararea a trei perechi de date. Aceste modele pot fi folosite ulterior pentru a evalua dacă FPGA-ul continuă să funcționeze corect iar, într-o abordare ulterioară a problematicii, ar putea să spună și care dintre funcționalitățile plăcii cu FPGA au fost deteriorate. În final, s-a exemplificat cum ar putea fi folosit modelul de referință obținut pentru semnalarea problemelor din modulele digitale exact în momentul în care acestea apar.

4.2 VERIFICAREA PROIECTELOR IMPLEMENTATE PE DISPOZITIVELE DE TIP FPGA

Accesul la semnalele modulelor implementate pe FPGA, necesar pentru stabilirea corectitudinii funcționării acestora, se poate face în diverse moduri. Cel mai la îndemână exemplu este folosirea analizoarelor logice dezvoltate de producătorii de dispozitive de tip FPGA (ex. analizorului logic ChipScope™ realizat de compania Xilinx, actualmente achiziționată de AMD). Penalizarea pentru această abordare, dacă este implementată de către utilizator prin mijloace proprii (fără a folosi utilitare special concepute de producătorii de plăci în acest scop), se reflectă în timpul necesar pentru configurarea acestui mecanism de salvare a datelor în memoria RAM și transferul lor către unitatea de procesare. O altă opțiune este utilizarea tabelelor cu registre de deplasare (eng. *Shift-register LUTs*) care însă furnizează un spațiu de stocare mai redus [33]. În [34], unde se folosesc, de asemenea, memoriile de tip Look-up Table (LUT), se propune o abordare a depanării orientată spre programare, unde comenzile pentru procesarea informațiilor se transmit folosindu-se o consolă. O altă abordare a depanării proiectelor digitale configurate pe FPGA este prezentată în [35], unde sunt abordate implementări software ale procesoarelor grafice (GPU). În acest studiu, structura FPGA se configurează pentru a funcționa ca un GPU. Această abordare se remarcă printr-o complexitate ridicată, aplicarea acesteia fiind oportună în cadrul proiectelor de dimensiuni mari.

O altă modalitate de depanare, larg răspândită și în rândul studenților, este rularea codului descărcat pe FPGA și prin intermediul unui simulator (ex. Questa® Advanced Simulator oferit de Siemens EDA, SimVision™ oferit de Cadence Design Systems, VCS® oferit de compania Synopsys, etc.). Dacă pe FPGA circuitul nu funcționează conform așteptărilor, se încearcă recreerea aceleiași situații în simulare, unde mai multe semnale sunt ușor vizibile, și problema poate fi caracterizată mai complet.

4.3 REALIZAREA UNEI APLICAȚII DE TRANSFER A DATELOR DE LA FPGA LA CALCULATOR

Acest capitol are rolul de a oferi indicații asupra pașilor necesari pentru a transfera date de la un dispozitiv cu FPGA la un calculator, oferind o idee mult simplificată despre procesele realizate în cadrul analizoarelor logice comerciale descrise amintite în capitolul 0. Pentru transferul datelor de la placa cu FPGA (în cazul de față fiind vorba despre placa Spartan 3E [36]) s-a utilizat un modul logic de UART, care a fost conectat la un convertor FT232R.

Pentru a se putea recepționa datele pe computer, s-a folosit utilitarul Putty, versiunea 0.73 (64-bit). De asemenea, pentru a se putea separa informațiile recepționate, acestea au trebuie despărțite octet cu octet, acest lucru realizându-se prin aplicarea în Microsoft Excel.

4.4 APLICAȚIE DE VERIFICARE A PROIECTELOR IMPLEMENTATE PE FPGA FOLOSIND INTELIGENȚA ARTIFICIALĂ

Pentru o generare uniformă a datelor pentru proiectul prezentat în acest capitol s-a folosit algoritmul LFSR. Acest algoritm fost implementat pe FPGA împreună cu celelalte elemente necesare calculului de vot majoritar și transmiterii datelor spre exterior în mod serial (*Fig. 9*).

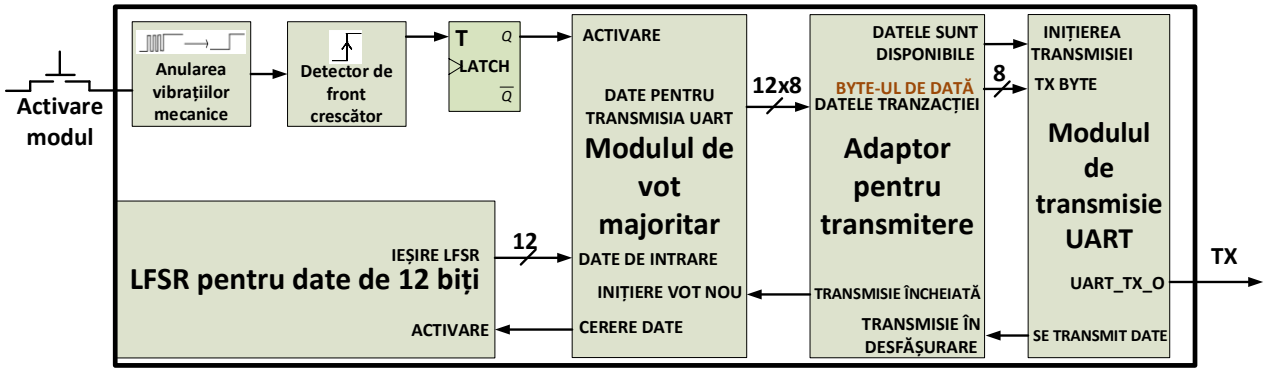

Fig. 9 Schema generatorului de date implementat pe FPGA

Modulul de vot majoritar reprezintă nucleul proiectului implementat pe FPGA. Ieșirea modulului de vot majoritar constă în cele șase numere folosite pentru vot și rezultatul votului: 2 ,dacă poziția din stânga perechii a câștigat (în binar 2 scriindu-se "10", deci cifra 1 fiind pe poziția câștigătoare) sau 1 dacă cele mai multe numere mari au fost pe a doua poziție a perechilor de numere. Implementarea algoritmilor de inteligență artificială, ca element demonstrativ, a avut rolul de a prezice rezultatul unui vot oarecare având la dispoziție cele șase numere de la intrare.

În cazul de față, având la dispoziție un fișier cu aproximativ 32800 de linii, s-au antrenat (folosindu-se un laptop care are ca nucleu procesorul Intel®Core™i7-6498DU cu o frecvență de bază de 2.50 GHz) mai mulți clasificatori disponibili în biblioteca scikit-learn. Cele mai bune rezultate ale procesului de antrenare sunt notate în *Tabelul 7*.

Tabelul 7 Rezultatele obținute în urma antrenării modelelor care au la bază algoritmi furnizați de biblioteca scikit learn

Algoritm	timpul de antrenare [s]	timpul de inferență [s]	acuratețea	precizia	sensibilitatea (recall)	scorul F1	aria delimitată de ROC
Gaussian Naïve Bayes	0.014	0.002	84.1%	84.0%	84.0%	84.0%	84.0%
SVM (C= 100)	9.055	1.391	92.9%	92.6%	92.6%	92.6%	92.6%
SVM (C= 200000)	457.027	0.36	97.5%	97.5%	97.5%	97.5%	97.5%
SVM (C= 500000)	1134.641	0.311	97.6%	97.6%	97.6%	97.6%	97.6%
KNN (se consideră cele mai apropiate20 valori)	0.013	0.697	93.6%	93.5%	93.5%	93.5%	93.4%

Tabelul 7 (Continuare)

Algoritm	timpul de antrenare [s]	timpul de inferență [s]	acuratețea	precizia	sensibilitatea (recall)	scorul F1	aria delimitată de ROC
Logistic Regression (calcul = sag)	0.12	0.001	84.1%	84.0%	84.0%	84.0%	84.0%
Logistic Regression (calcul = saga)	0.07	0.002	84.1%	84.0%	84.0%	84.0%	84.0%
Colecții de Arbori Decizionali (adâncimea maximă=10)	0.319	0.017	94.0%	93.4%	93.4%	93.4%	93.4%
Colecții de Arbori Decizionali (adâncimea maximă=20)	0.345	0.019	94.6%	94.3%	94.3%	94.2%	94.2%

În *Tabelul 7* se poate observa că valorile metricilor acuratețe, precizie, scor F1, sensibilitate și arie delimitată de ROC sunt foarte asemănătoare. Acest lucru denotă o bună distribuție a datelor folosite pentru antrenare în spectrul de valori posibile. Cel mai performant algoritm în situația de față s-a dovedit a fi algoritmul de tip mașini cu suport vectorial (SVM), atunci când valoarea parametrului C a fost cel puțin 1000, având astfel o acuratețe între 94,9% și 97,6%.

A doua modalitate de creare a modelelor de referință a fost reprezentată de proiectarea rețelelor neurale folosind biblioteca PyTorch. Astfel, au fost dezvoltate unsprezece versiuni de rețele neurale derivate dintr-o arhitectură cu opt straturi și șapte versiuni de rețele neurale derivate dintr-o arhitectură având paisprezece straturi de neuroni. Rezultatele procesului de antrenare sunt vizibile în *Tabelul 8*.

Rețeaua care a oferit cele mai bune rezultate este cea denumită v15. Când antrenamentul acestei rețele a ajuns la 600 de iterații, acuratețea rețelei a fost egală cu 99,749%, depășind astfel și performanțele algoritmilor de învățare automată care au folosit funcții oferite de biblioteca *scikit learn*.

Prin antrenarea unor rețele neurale și configurarea acestora pentru a atinge o performanță ridicată, s-au obținut modele de referință care emulează funcționalitatea DUT-ului cu diferite grade de precizie. Selectând cel mai performant model de referință obținut și furnizându-i aceiași stimuli ca și proiectului digital implementat de pe FPGA, acesta poate fi folosit pentru a verifica corectitudinea funcționării acestuia. Această verificare se poate realiza încontinuu, de exemplu prin configurarea unei alte părți din FPGA cu rețeaua neurală reprezentând modelul de referință al DUT-ului. Stabilirea corectitudinii funcționării DUT-ului poate fi făcută fie de către o unitate centrală externă, fie de către o logică existentă pe FPGA. Acest sistem original de procesare de date este reprezentat în *Fig. 10*.

Tabelul 8 Primele zece cele mai performante configurații de rețele neurale antrenate

Numele modelului	Timpul de antrenare [s]	Numărul de iterații de antrenare	Variabila aleatorie (e.g. *seed*)	Procentajul de date de test din total (%)	Rata de învățare	Acuratețea (%)
Configurația v15-600-0.02-0.4	980.4	600	3245	40	0.02	99.74916
Configurația v18-600-0.01-0.4	349.5	600	3245	40	0.01	99.74156
Configurația v18-600-0.01-0.3	443.0	600	3245	30	0.01	99.7365
Configurația v18-400-0.01-0.4	229.9	400	3245	40	0.01	99.73396
Configurația v18-500-0.01-0.4	287.5	500	3245	40	0.01	99.73396
Configurația v15-600-0.01-0.3	967.0	600	3245	30	0.01	99.70609
Configurația v15-500-0.01-0.3	813.8	500	3245	30	0.01	99.68582
Configurația v18-400-0.01-0.3	313.2	400	3245	30	0.01	99.67569
Configurația v18-300-0.005-0.4	174.5	300	3245	40	0.005	99.65795

Concluzii

Spre deosebire de capitolul 0 unde rezultatele cele mai bune au fost obținute de algoritmi de învățare automată, în acest studiu s-au obținut performanțe mai ridicate folosindu-se rețelele neurale dezvoltate utilizând biblioteca PyTorch. Acest lucru evidențiază avantajele de a avea mai multe metode pentru rezolvarea unei probleme complexe, și a alege în final soluția cea mai performantă, care poate să difere de la un set de date la altul.

Modalitatea de pre-procesare a datelor a fost implementată într-o manieră originală, fiind descriși toți pași de la generarea datelor în hardware până la transmiterea, refacerea și stocarea acesteia în calculator sub formă de bază de date în format *.csv*. O cerință deosebit de importantă este faptul că

atunci când se colectează datele pentru a se antrena un model, trebuie să fim convinși că acele date sunt corecte. Altfel, se introduc probleme de funcționalitate direct în modelul de referință.

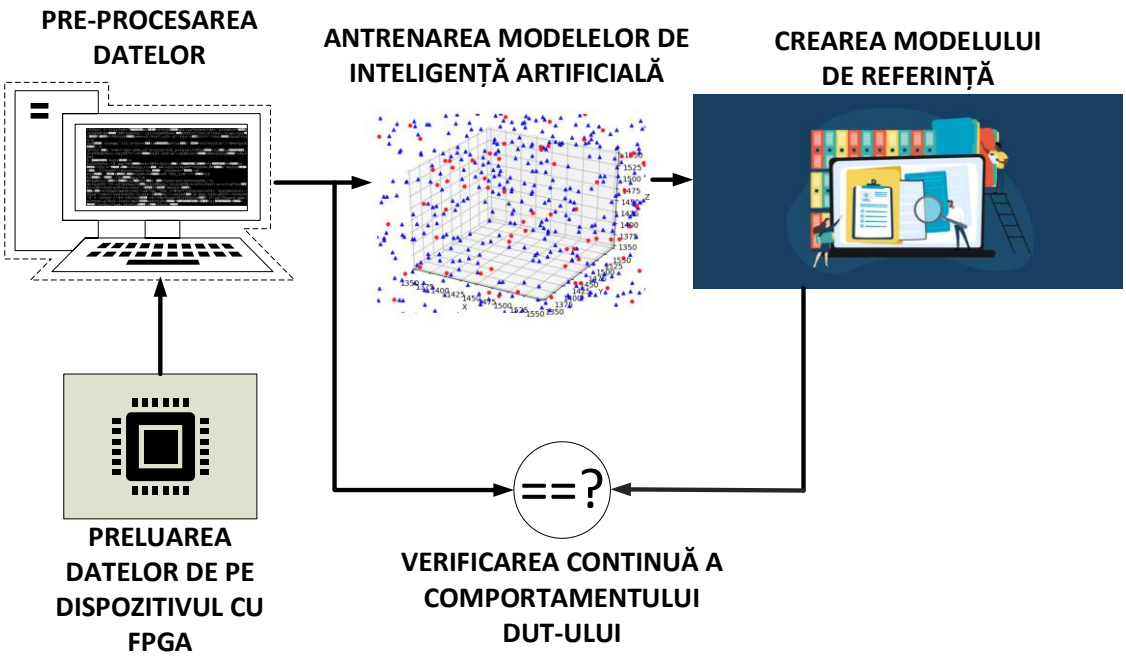

Fig. 10 Sistemul de monitorizare a funcționalității proiectului configurat pe FPGA

Cea mai însemnată limitare în realizarea sistemului de depanare prezentat este transmiterea datelor spre unitatea de procesare (ex. spre server sau calculator). Dacă datele sunt generate cu o frecvență mai mare decât viteza de transmitere a acestora spre calculator (ex. mai mare decât viteza de baud rate în cazul de față), unele eșantioane de date vor fi pierdute.

Modelul de referință obținut în cadrul acestui studiu poate fi folosit pentru a verifica continuu dacă proiectul digital implementat pe FPGA continuă să funcționeze în parametri doriți, conform sistemului original reprezentat în *Fig. 10*. Dacă sunt descoperite probleme în timpul monitorizării DUT-ului, gradul avansat de automatizare al procesării informației prezentat mai sus împreună cu flexibilitatea caracteristică dispozitivelor cu FPGA facilitează înlăturarea rapidă a defectelor din componenta cu funcționalitate eronată. Astfel, fiabilitatea dispozitivelor electronice se îmbunătățește și schimbările din cadrul specificațiilor sistemelor electronice se implementează într-un timp mai redus.

Pe baza activităților de cercetare prezentate în cadrul acestui capitol au fost publicat articolele:

A. Dinu, Ș. Gheorghe, G.M. Danciu, P.L. Ogrutan (2021). Debugging FPGA projects using artificial intelligence. *Science and Technology*, *24*(3), 299-320.

și

A. Dinu, G.M. Danciu, P.L. Ogrutan, "Efficient analysis of digital systems' supplied data," *2020 International Symposium on Electronics and Telecommunications (ISETC)*, 2020, pp. 1-4, doi: 10.1109/ISETC50328.2020.9301139.

și

A. Dinu, G.M. Danciu, P.L. Ogruțan, "Debug FPGA projects using machine learning," *2020 International Semiconductor Conference (CAS)*, 2020, pp. 173-176, doi: 10.1109/CAS50358.2020.9268007.

5 VERIFICAREA FUNCȚIONALĂ DISTRIBUITĂ ÎNTR-O ABORDARE DIDACTICĂ ORIGINALĂ

Această teză a fost îmbogățită prin adăugarea unui capitol cu valențe didactice: predarea disciplinei de verificare a circuitelor integrate, folosind mijloace moderne, regăsite frecvent în industrie. Succesul obținut la sfârșitul unui semestru de activități didactice, demonstrat de organizarea unei sesiuni de prezentare a proiectelor studenților în fața companiilor din domeniu, a arătat că studenții pot realiza proiecte de interes pentru industrie, folosind mijloace specifice acesteia într-un cadru școlar.

5.1 AVANTAJELE DISTRIBUIRII SARCINILOR ÎN REALIZAREA UNUI CIRCUIT INTEGRAT

Proiectantul unui circuit poate depune mai mult efort pentru conturarea unei funcționalități complexe și mai puțin efort pentru o altă funcționalitate pe care a implementat-o cu mai multă ușurință. Acesta, dacă ar trebui să efectueze și verificarea funcțională a circuitului, va investi mai multă energie pentru a testa în continuare aspecte ale funcționalității mai complexe, și va depune mai puțin efort pentru a verifica partea care "oricum a fost simplă de implementat, sunt șanse minime să fie ceva greșit acolo". Inginerii de verificare însă, vor depune efort pentru a verifica complet fiecare funcționalitate, neținând cont de efortul depus la implementare. Astfel de verificare obiectivă este benefică în descoperirea de greșeli chiar în părțile simple din proiect care au fost tratate poate prea puțin sau nu au fost înțelese suficient de proiectant. Mai mult, specificațiile unui circuit integrat sunt scrise de către arhitect, care nu este direct implicat nici în proiectare, nici în verificare. Această delimitare între arhitecți, proiectanți și ingineri de verificare în cadrul unui proiect de realizare a unui circuit digital este deosebit de importantă pentru o desfășurare optimă a verificării funcționale deoarece presupun o responsabilitate distribuită. De aceea, s-a urmărit o abordare distribuită în cadrul activităților didactice proiectate, unde studenții au avut roluri bine stabilite în cadrul procesului de verificare.

5.2 TEMA ȘI CONFIGURAȚIA ALEASĂ PENTRU ACTIVITĂȚILE DE LABORATOR

Tema laboratorului, la care au participat 39 de studenți, a reprezentat conceperea, proiectarea și verificarea funcționalității unui circuit integrat. Atât tema, cât și instrumentele de predare au fost dezvoltate astfel încât să trezească interesul studenților, iar cunoștințele dobândite să poată fi folosite în activitatea profesională.

Ca instrumente de lucru, pentru scrierea mediului de verificare, studenților li s-a predat limbajul SystemVerilog [37], cel mai folosit limbaj de verificare în industrie [38]. Pentru simularea proiectelor, s-a folosit platforma on-line EDAPlayground care conferă acces la simulatoare comerciale în mod gratuit, dar cu anumite limitări. Pentru a se putea comunica cu cadrul didactic sau între membrii unei echipe, s-a folosit platforma on-line Azure DevOps. Aceasta a permis atât definirea sarcinilor de efectuat în cadrul proiectului de către fiecare student, cât și urmărirea problemelor găsite în procesul de verificare.

5.3 REZULTATELE OBȚINUTE ÎN URMA DESFĂȘURĂRII ACTIVITĂȚILOR DIDACTICE

În cadrul activității didactice prezentate, s-au format 7 echipe de câte 5 studenți și o echipă de 4 studenți. Fiecare echipă și-a conceput propriul circuit, care a fost documentat de către arhitect, a fost implementat de către proiectant în limbajul Verilog și a fost verificat de către ceilalți studenți din echipă care au primit titulatura de ingineri de verificare.

Funcționalitățile concepute de studenți pentru cele opt proiecte didactice au fost :

- Controlul unei prize inteligente
- Managementul unei intersecții cu trei semafoare
- Administrarea comenzilor dintr-un restaurant
- Controlul ventilării unei încăperi
- Convertor numeric serial la BCD
- Controlul unei lămpi inteligente
- Dispozitiv de control al ambientului
- Gestionarea intrării unei parcări

Șapte dintre circuitele de mai sus împreună cu mediile de verificare și specificațiile aferente au fost prezentate în fața a șapte companii în domeniul verificării circuitelor integrate din Brașov. Prezentarea proiectelor a fost interactivă, studenții oferind răspunsuri pe parcursul prezentărilor la întrebările venite din partea reprezentanților companiilor. Din impresiile transmise de către reprezentanții companiilor, s-a extras o frază care arată că rezultatele obținute și cunoștințele acumulate de studenți pe timpul semestrului sunt apreciate în mediul industrial: "Încă o data felicitări pentru efort si pentru rezultate. Foarte multă munca pentru un singur om sa ajute si sa coordoneze toți studenții".

După sesiunea de prezentare a proiectelor, studenții care au fost interesați să colaboreze cu firmele din domeniu au fost invitați să ofere datele de contact reprezentanților companiilor, într-un formular centralizat de către cadrul didactic. Astfel, cel puțin doi studenți au obținut un stagiu de practică ("după materia pe care am avut-o cu dumneavoastră semestrul acesta și după obținerea unui stagiu de practică in acest domeniu a început să mă atragă mai mult domeniul de verificare, de Verilog".) în respectivele companii, și știm că un alt student este deja angajat cu normă întreagă.

Nu în ultimul rând, sesiunea de prezentare a proiectelor a fost benefică pentru consolidarea relației Universității cu companiile. Reprezentații industriei au primit informații suplimentare în legătură cu cunoștințele studenților, iar studenții au aflat lucruri noi despre companiile care pot accede (după prezentarea lucrărilor de către cursanți, fiecare companie și-a prezentat domeniul de activitate). De asemenea încrederea studenților în calitatea procesului de învățare și, implicit încrederea în Universitate, a crescut. Relațiile dezvoltate în urma sesiuni de prezentare a proiectelor s-au reprezentat în Fig. 11.

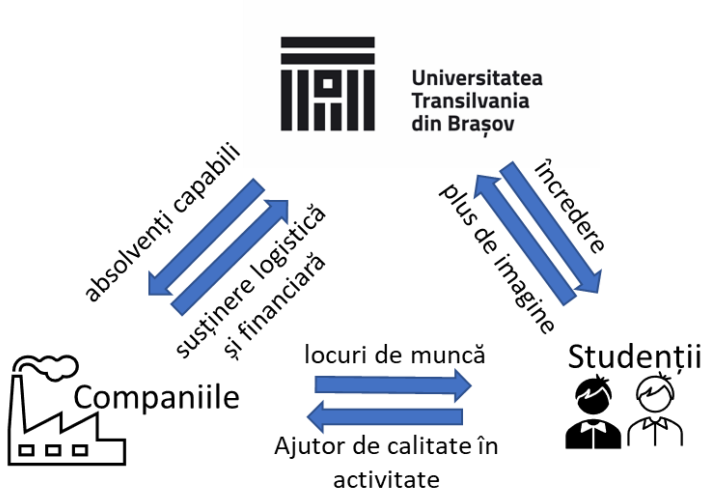

Fig. 11 Tipurile de relații care pot fi îmbunătățite prin susținerea proiectelor studenților în fața companiilor din domeniu

5.4 Concluzii

Faptul că studenții au fost lăsați să își construiască singuri temele de proiect, colaborarea avută prin intermediul platformei Azure DevOps și crearea unor roluri pentru studenți asemănătoare cu cele avute de inginerii din domeniu în industrie au reprezentat câteva elemente originale al activităților didactice apreciate de studenți, care au reușit să îi motiveze pentru a realiza niște proiecte de calitate, alegând să lucreze la proiecte chiar și în afara orelor dedicate materiei respective.

Desfășurarea activităților didactice susținute în cadrul materiei "Testarea echipamentelor electronice" a avut în vedere îmbogățirea studenților cu o parte din aptitudinile necesare angajaților din câmpul muncii. Folosindu-se tehnologii de top, s-au dezvoltat în cadrul activităților practice proiecte de verificare funcțională pe care studenții, însușindu-și cunoștințele de bază necesare, au avut încrederea și dorința de a le prezenta în fața reprezentaților unor companii din domeniu. Organizarea lucrului pe echipe, accentul pus pe implementarea practică a noțiunilor teoretice învățate, managementul întregului proiect folosind platforma Azure DevOps furnizată de Microsoft sunt doar câteva din lucrurile care au atras atenția și implicarea studenților, făcându-le munca mai plăcută și mai captivantă. S-a demonstrat că pe parcursul activităților didactice, studenții și-au însușit elemente cheie de comunicare academică, au dat dovada că pot participa în proiecte tehnice cu succes dacă beneficiază de sprijinul adecvat și pot reprezenta într-un mod pozitiv grupul din care fac parte. În final, se poate afirma că succesul unei activități didactice reprezintă o simbioză între dorința și efortul studenților de a învăța, prestanța cadrului didactic care trebuie să reprezinte un sprijin de încredere disponibil de câte ori este nevoie, disponibilitatea tehnologiilor de vârf din industrie și a infrastructurii solide prin care acestea sunt accesate și crearea unor legături sănătoase cu mediul comercial. În aceste condiții, din ce în ce mai mult va fi pe buza studenților unul din motoarele care determină schimbarea în bine a activităților didactice: satisfacția studenților exprimată ca în propoziția: "Sincer, au fost cursul și laboratorul meu preferat"(extras din chestionarul completat de studenți la sfârșitul activităților didactice).

6 CONCLUZII ȘI CONTRIBUȚII ORIGINALE

6.1 CONCLUZII GENERALE

Există numeroase oportunități de a automatiza procesul de verificare funcțională a circuitelor integrate utilizând inteligența artificială. Tranzacțiile generate pentru acoperirea funcționalităților încă nesimulate ale proiectului de verificat, acoperirea martorilor de completitudine a verificării, generarea automată a codului aferent unor componente ale mediului de verificare, predicția completitudinii verificării, generarea martorilor de completitudine, generarea aserțiunilor digitale și detectarea erorilor din simulare sunt doar câteva din sarcinile potrivite pentru a fi trimise spre execuție motoarelor de inteligență artificială. Din acest motiv, atât mediul academic, cât și Industria fac eforturi semnificative pentru a integra IA în procesul de verificare al circuitelor integrate. Cu toate acestea, modalitățile de automatizare ale verificării nu au atins încă un nivel de maturitate, nefiind încă stabilite de nicio metodologie sau standard industrial. În cadrul acestei lucrări s-a oferit o privire de ansamblu asupra modalității clasice de abordare a verificării funcționale, dar și asupra posibilităților de automatizare a acestui proces.

Din punct de vedere practic, în teza de față s-a realizat automatizarea procesului de verificare funcțională din mai multe puncte de vedere.

Verificarea unui modul digital reprezintă etapa cea mai consumatoare de timp din dezvoltarea logică a unui circuit. O bună parte din acest timp este consumată de simulări lungi, în care se generează stimuli care se trimit circuitului pentru a-l aduce pe acesta în toate situațiile de funcționare posibile. În capitolul 0 s-a arătat cum, pe baza datelor generate pe parcursul simulării sistemelor digitale, se pot crea modele de referință care să implementeze cu fidelitate anumite funcționalități ale unui proiect digital al unui circuit. Creând un model software al unui circuit și furnizându-i stimulii care vor putea fi primiți de către modulul electronic în funcționarea reală, verificarea acestuia se poate face mai rapid, înlăturându-se timpul necesar pentru realizarea simulărilor. În această teză, modelele de referință au fost reprezentate atât de algoritmi de învățare automată cât și de rețele neurale create de la zero. În cazul modelelor bazate pe rețele neurale, acuratețea maximă obținută în prezicerea corectitudinii unui transfer de date a fost de peste 98,6%, iar în cazul modelelor rezultate în urma aplicării algoritmilor de învățare automată acuratețea a atins valoarea de 99,3%.

Conform principiilor verificării, este necesar ca toate funcționalitățile unui modul să fie testate în simulare, pentru ca eventualele probleme existente să nu ajungă să fie transferate în siliciu. În unele situații, recrearea unor scenarii de funcționalitate în simulare este un proces anevoios, de durată și necesită o bună cunoaștere a modulului verificat de către ingineri. Pentru a scurta timpul simulării tuturor scenariilor de funcționalitate propuse ale unui DUT, în capitolul 0 s-a realizat un nou tip de modele de referință al circuitelor digitale față de cele din capitolul 0. Aceste modele, dezvoltate conform procedeului de învățare bazată pe recompense, au fost capabile să genereze o serie de stimuli pentru ca circuitul de verificat (reprezentat în cazul de față de o unitate aritmetico-logică) să

poată ajunge într-o stare prestabilită într-un număr minim de pași. S-a demonstrat faptul că automatizarea propusă în capitolul 0 poate întrece în performanțe generarea aleatorie a stimulilor (care actualmente este folosită într-o mare măsură în procesul clasic de verificare a circuitelor integrate), aducând un DUT în starea dorită chiar și de peste 18 ori mai repede decât în cazul utilizării abordării clasice. Studiul integrării învățării bazate pe recompense în procesul de automatizare a circuitelor integrate a excelat prin multitudinea de teste făcute în detectării celor mai potrivite valori pentru parametrii de configurare ai algoritmilor. Una din contribuțiile originale în acest capitol este dezvoltarea metodei de marcare a stărilor vizitate prin care, în cadrul procesului de inferență, se încearcă recuperarea și exploatarea agenților slabi antrenați. Acest mecanism (împreună cu versiunea extinsă a sa) a reușit să facă să atingă starea terminală agenți care în mod normal nu ar fi atins-o, deoarece ar fi intrat în bucle infinite de vizitare a anumitor stări. De asemenea, acest mecanism a obținut întotdeauna performanțe cel puțin la fel de bune cu situațiile în care nu s-a aplicat mecanismul (E)MSV și în cazul agenților bine antrenați.

Cel mai important indicator al progresului obținut în verificarea funcțională este acoperirea funcțională. Generându-se constrâns-aleatoriu o mulțime de stimuli care apoi se trimit la intrările circuitului de verificat în cadrul simulărilor rulate, valoarea de acoperire funcțională crește. Cu cât aceasta se apropie mai mult de 100%, cu atât acoperirea crește din ce în ce mai greu deoarece situațiile de funcționare ale circuitului de verificat încep să se repete în cadrul simulărilor, iar situațiile limită este puțin probabil să fie "nimerite" de stimulii generați constrâns-aleatoriu. În capitolul 0 este propusă o modalitate automată de a construi secvențe de stimuli pentru a atinge valoarea maximă de acoperire pentru diferite obiective avute în vedere atunci când se verifică un DUT. Aceste secvențe sunt construite pe parcursul mai multor iterații ale unor proceduri *software* euristice bazate pe funcționarea algoritmilor genetici. S-a dovedit în cadrul acestei teze faptul că generarea stimulilor folosind algoritmi genetici prezintă avantaje importante față de generarea constrâns-aleatorie a stimulilor atunci când proiectul de verificat este suficient de complex. În cele mai multe cazuri testate, generarea constrâns aleatorie nu a reușit să atingă valoarea maximă de acoperire, pe când cele mai bune abordări dezvoltate în cadrul acestui studiu au atins-o prin intermediul mai multor secvențe de date generate. De asemenea, atunci când s-a putut genera în mod aleatoriu o secvență de stimuli care a obținut 100% acoperire, raportul dintre numărul de secvențe performante generate prin intermediul algoritmilor genetice și cele generate de abordarea clasică a fost de 205:1. De asemenea, abordările dezvoltate în cadrul acestei teze se remarcă prin faptul că folosesc versiunea gratuită a simulatorului ModelSim® și totuși beneficiază de elementele importante în desfășurarea verificării: o distribuție bună a generării valorilor aleatorii, colectarea de acoperire funcțională și folosirea componentelor din biblioteca UVM1.2. Acest lucru a fost făcut posibil compensând din implementările *software* limitările impuse de simulator și realizând un studiu intens asupra modalității de încărcare a fișierelor din biblioteca UVM1.2 în proiectele concepute, fără ajutorul unui script adițional (ex. un program care folosește utilitarul *Makefile*, rulând în sistemul de operare *Linux*).

Dacă verificarea funcțională are în vedere eliminarea problemelor logice din proiectul digital al circuitelor integrate astfel încât acestea să ajungă să funcționeze exact cum este scris în documentele de specificație, validarea constă în demonstrarea fezabilității specificației circuitului de proiectat. Una din cele mai răspândite metode de validare a unui proiect digital este încărcarea acestuia (total sau parțial) pe un dispozitiv de tip FPGA. Astfel, se poate observa dacă toate presupunerile făcute în

specificație legate de un circuit au corespondent în funcționarea reală a acestuia. Pe de altă parte, în anumite limite, prin intermediul validării se poate măsura și corectitudinea funcționării DUT-ului. În capitolul 0 din această teză se prezintă pe larg un sistem de depanare realizat prin intermediul unui dispozitiv cu FPGA care are ca finalitate crearea unui model de referință al logicii circuitului dezvoltat. Sunt parcurse toate etapele, de la generarea stimulilor, la transferul acestora pe calculator, pre-procesarea fișierelor obținute, extragerea informațiilor de interes și antrenarea cu acesta a unor modele realizate conform noțiunilor de inteligență artificială. Acest studiu a avut ca finalitate realizarea unui model de referință care să copieze fidel funcționalitatea DUT-ului pentru a putea fi folosit apoi la depanarea unor versiuni ulterioare ale acestuia. În fluxul de procesare al informațiilor dezvoltat în cadrul capitolului 0 se remarcă modalitatea originală de pre-procesare a datelor care presupune înlocuirea unor caractere speciale, care îngreunează analiza datelor, cu alte caractere neutre. O idee originală privind continuarea acestui proiect este încărcarea modelului de referință rezultat alături de logica propriu-zisă a circuitului pe FPGA, pentru a se realiza o monitorizare în timp real a funcționalității DUT-ului și a se semnala prompt devierile de funcționalitate care pot apărea din cauza unor probleme nedescoperite ale modulului digital sau din alte cauze interne sau externe modulului electronic.

Teza de doctorat de față este întregită de prezența unui secțiuni referitoare la procesul didactic din cadrul materiei Testarea Echipamentelor Electronice care constă în formarea modernă a noilor generații cu privire la dezvoltarea și verificarea circuitelor integrate. Partea de automatizare a verificării prezentată în cadrul acestei teze nu are menirea de a înlocui elementul uman, ci de a-l degreva de sarcinile care nu țin de creativitate ci mai degrabă de un proces încercare-eroare (eng. *trial and error*) în care este necesară generarea unei mari cantități de date. În cazul acestei sarcini și a altor astfel de activități, puterea computațională a sistemelor de calcul din ziua de astăzi poate fi exploatată într-un mod deosebit de avantajos. Așadar, dat fiind faptul că eliminarea problemelor din proiectul digital al unui circuit, începând de la specificație și planul de verificare până la atingerea obiectivului de „zero erori”, este un proces condus de ingineri, este important ca studenții în domeniul ingineriei electrice și științei calculatoarelor să beneficieze de noțiuni fundamentale solide în acest domeniu. Activitățile didactice proiectate și prezentate în capitolul 0 au oferit studenților posibilitatea de a lucra într-un mediu asemănător celui din industrie, în echipă, folosind utilitare de administrare a sarcinilor de efectuat și urmărire a problemelor și primind roluri asemănătoare celor din companiile producătoare de circuite integrate (arhitect, proiectant, inginer de verificare). Studenților le-a fost stimulată creativitatea prin dezvoltarea unor specificații de proiect de la zero, cât și dorința de a lucra ceea ce a dus la realizarea unor proiecte interesante, calitative pentru nivelul de studenți de anul IV, care au putut fi prezentate în fața profesioniștilor din domeniu.

Prin urmare, automatizarea abordărilor clasice care compun verificarea funcțională scurtează timpul de realizare a unui circuit integrat. Întreaga cercetare din această teză este marcată de avantajele de care algoritmii de inteligență artificială dau dovadă într-o gamă largă de aplicații . Metodele de automatizare propuse au rolul de a ușura o parte din sarcinile pentru a căror rezolvare inginerii trebuie care lucrează în procesul de dezvoltare al circuitelor trebuie să apeleze mai degrabă la metode cu o importantă componentă aleatorie (de exemplu, constrângerea generării stimulilor), decât la experiența pe care au acumulat-o la locul de muncă. Considerăm că, așa cum s-a trecut de la plăcile cu componente discrete la circuitele integrate, așa cum s-a trecut de la programarea în limbaj mașină

la programarea folosind limbaje de nivel înalt, tot așa și modalitățile actuale de obținere a stimulilor pentru procesul de verificare funcțională (bazate în mare măsură pe generarea constrâns-aleatorie) vor fi înlocuite de metode automatizate care vor scurta considerabil procesul de verificare funcțională. Această teză se încadrează în tendința globală de automatizare a proceselor industriale, în general, și a verificării funcționale, în particular, care, după cum reiese din studiul lucrărilor din domeniu, este susținută de către toți marii actori din domeniul circuitelor integrate.

6.2 CONTRIBUȚII PERSONALE

În continuare sunt enumerate punctual contribuțiile principale ale acestei teze, făcându-se corespondența cu numerotarea obiectivelor operaționale rezultate din detalierea obiectivelor principale (prezente în capitolul 0).

1.1. Documentarea procesului de realizare a circuitelor integrate din punct de vedere logic.
1.2. Realizarea unui set de propuneri originale de îmbunătățire a variantei clasice de verificare funcțională
1.3. Realizarea unei analize ale inițiativelor de automatizare ale diferitelor componente ale verificării
1.4. Prezentarea unor noțiuni de bază referitoare la domeniul inteligenței artificiale
2.1. Obținerea a patru medii de verificare construite conform metodologiei UVM, folosite pentru a verifica patru proiecte digitale
2.2. Realizarea a 22 de configurații de rețele neurale pe baza cărora se pot antrena modele cu rol de clasificare
2.3. Testarea a cinci algoritmi de învățare automată în multiple configurații prin antrenarea a numeroase modele configurate diferit
2.4. Crearea unui sistem funcțional hardware-software de generare, transmitere, analiză și reținere în memoria calculatorului a datelor furnizate de o placa Spartan 3E cu FPGA.
2.5. Obținerea unor modele de referință performante pentru modulul de vot majoritar implementat pe placa cu FPGA Spartan 3E
2.6. Conceperea unui sistem original de urmărire în timp real a funcționalităților modulelor implementate pe dispozitive cu FPGA
3.1. Definirea unui set de pași care permit folosirea simulatorului ModelSim® pentru verificarea funcțională
3.2. Testarea mai multor scheme de recompensare și alegerea celei care a permis antrenarea eficientă a agenților de învățare bazată pe recompense
3.3. Obținerea unui sistem *software* original (Fig. 3) prin care s-a realizat o comunicare completă între algoritmii implementați în limbajul Python și simulatorul ModelSim®
3.4. Implementarea algoritmului SGTD(0) în sistemul *software* folosit pentru antrenarea agenților de ÎBR
3.5. Testarea modelelor antrenate folosind ÎBR și compararea acestora cu generarea constrâns-aleatorie a stimulilor

4.1. Implementarea mai multor variante de automatizare a generării stimulilor de date pentru a obține valoarea maximă de acoperire având ca nucleu algoritmii genetici
4.2. Dezvoltarea unor sisteme software de comunicare cu simulatorul ModelSim® în care stimulii sunt generați prin intermediul algoritmilor genetici
4.3. Definirea unei formule originale de modificare a coeficientului de mutație de-a lungul iterațiilor algoritmilor genetici
4.4. Integrarea unor versiuni ale algoritmilor NSGA-II și SPEA2 în sistemele *software* de antrenare dezvoltate în cadrul acestei teze și compararea acestora cu abordările bazate derivate din algoritmul genetic simplu
4.5. Demonstrarea superiorității abordărilor bazate pe algoritmii genetici față de generarea constrâns-aleatorie a stimulilor în procesul de acoperire

Un obiectiv secundar, atins în cadrul acestei teze, este realizarea unor activități didactice moderne de predare a disciplinei de verificare a circuitelor integrate. Scopul acestora a fost ca viitorii ingineri să fie pregătiți pentru a se integra în industria aflată într-o stare permanentă de evoluție, caracterizată de automatizare. Studenții au avut acces pe parcursul activităților didactice la servicii software accesate on-line (reîntâlnindu-se practic cu noțiunea de *cloud computing*), utilitare pentru a dezvolta o comunicare profesională eficientă între membrii unor echipe extinse în mai multe locații și au folosit cele mai utilizate limbaje și metodologii de verificare la ora actuală pentru a-și dezvolta propriile proiecte de verificare. Calitatea acestor proiecte a fost validată în cadrul unei sesiuni la care au participat companii de verificare din domeniu, atât din Brașov, cât și din București.

6.3 LUCRĂRI PUBLICATE

O parte din elementele dezvoltate pe parcursul studiilor întreprinse au fost publicate la conferințe și în cadrul unor reviste tehnice din România și din străinătate. Astfel, informațiile dezvoltate au fost validate de mai multe grupuri de cercetători, iar conținutul tezei a fost îmbunătățit semnificativ în urma observațiilor primite înainte de publicarea articolelor. În domeniul tezei s-au realizat următoarele publicații:

Reviste cu factor de impact:

1. **Alexandru Dinu**, Gabriel Mihail Danciu, Petre Lucian Ogruțan, *Cost-Efficient Approaches for Fulfillment of Functional Coverage during Verification of Digital Designs*, 2022 Micromachines, an MDPI Journal, FI 2.89, SRI 1.171, indexare WoS și PubMed

2. Gabriel Mihail Danciu, **Alexandru Dinu**, *Coverage Fulfillment Automation in Hardware Functional Verification Using Genetic Algorithms*, Applied Sciences, an MDPI Journal, FI 2.679, SRI 0.923, indexare WoS și PubMed, WOS:000755070200001

3. **Alexandru Dinu**, Ștefan Gheorghe, Gabriel Mihail Danciu, Petre Lucian Ogruțan, *Debugging FPGA projects using artificial intelligence,* 2021 Romanian Journal of Information Science and Technology, FI 0.63, SRI 0.273, WOS:000704174600003

Conferințe indexate WoS și IEEE:

1.**Alexandru Dinu**, Petre Lucian Ogruțan, *Coverage fulfillment methods as key points in functional verification of integrated circuits,* 2019 International Semiconductor Conference (CAS) , Sinaia, indexare WoS si IEEE, WOS:000514295300042

2.**Alexandru Dinu**, Petre Lucian Ogruțan, *Opportunities of using artificial intelligence in hardware verification,* 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), Cluj-Napoca, indexare WoS si IEEE, WOS:000564733700045

3.**Alexandru Dinu**, Gabriel Mihail Danciu, Petre Lucian Ogrutan, *Efficient analysis of digital systems' supplied data,* 2020 International Symposium on Electronics and Telecommunications, indexare WoS si IEEE, WOS:000612681000020

4.**Alexandru Dinu**, Gabriel Mihail Danciu, Petre Lucian Ogrutan, *Debug FPGA projects using machine learning,* 2020 International Semiconductor Conference (CAS), indexare WoS si IEEE, WOS:000637264600039

Conferințe indexate în BDI:

1. Stefan Gheorghe, **Alexandru Dinu**, *Controlling hardware design behavior using Python based machine learning algorithms*, 2021 16th International Conference on Engineering of Modern Electric Systems - ICEMES, indexare IEEE

2.**Alexandru Dinu**, Gabriel Mihail Danciu, Stefan Gheorghe, *Level up in verification: learning from functional snapshots*, 2021 16th International Conference on Engineering of Modern Electric Systems - ICEMES, indexare IEEE

Cărți:

1.Petre Lucian Ogrutan, **Alexandru Dinu**, *Inițiative noi și dificultăți ale trecerii la educația prin implicare a studenților la inginerie,* 2021, Editura Universității Transilvania din Brașov, ISBN 978-606-19-1420-3, 86 pagini

2. Gabriel Mihail Danciu, **Alexandru Dinu**, Alexandra Dobrinaș, *Structuri de date si algoritmi*, 2022, Editura Universității Transilvania din Brașov, ISBN 978-606-18-1483-8,

De asemenea, corespunzător activităților conexe tezei s-au realizat următoarele publicații:

1. Petre Lucian Ogrutan, Alina Luminita Machidon, **Alexandru Dinu**, *Is There a Link Between Creativity and Multiculturalism in Education?,* 2019 TEM Journal, indexare WoS, WOS:000468971700034

2. Iustin Constantin, **Alexandru Dinu**, Simona Coman, *Automatic light intensity control for a POV technology based display*, 2021 16th International Conference on Engineering of Modern Electric Systems - ICEMES, indexare IEEE

BIBLIOGRAFIE

[1] J. Hunt, *Advanced Guide to Python 3 Programming*. Springer, 2019.

[2] Sigenics, "Custom ASIC Cost Calculator," ed, 2016.

[3] J. Donahue. "Survey: 80 Percent of Enterprises Investing in AI, but Cite Significant Challenges Ahead." Teradata. https://www.teradata.com/Press-Releases/2017/Survey-80-Percent-of-Enterprises-Invest-in-AI (accessed July, 13, 2020).

[4] H. Kaeslin, *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*. Cambridge University Press, 2008.

[5] T. Tulbure, "A Dynamic Reconfigurable CPLD Architecture for Structured ASIC Technology," Berlin, Heidelberg, 2011: Springer Berlin Heidelberg, pp. 296-301.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[7] D. Gajera, "Process costing of microchip," Master of Science in Industrial Engineering, Department of Industrial and Management Systems Engineering, West Virginia University, 2006. [Online]. Available: https://researchrepository.wvu.edu/etd/4228

[8] S. Walters, "Computer-aided prototyping for ASIC-based systems," *IEEE Design & Test of Computers*, vol. 8, no. 2, pp. 4-10, 1991.

[9] F. Eslami, E. Hung, and S. J. Wilton, "Enabling effective FPGA debug using overlays: Opportunities and challenges," *arXiv preprint arXiv:1606.06457*, 2016.

[10] (2015). *Universal Verification Methodology (UVM) 1.2 User's Guide*.

[11] T. Blackmore, R. Hodson, and S. Schaal, "Novelty-Driven Verification: Using Machine Learning to Identify Novel Stimuli and Close Coverage," presented at the Design and Verification Conference and Exhibition United States (DVCON US 2021), Virtual, 2021.

[12] C. Ioannides and K. I. Eder, "Coverage-directed test generation automated by machine learning--a review," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 17, no. 1, pp. 1-21, 2012.

[13] F. Wang, H. Zhu, P. Popli, Y. Xiao, P. Bodgan, and S. Nazarian, "Accelerating coverage directed test generation for functional verification: A neural network-based framework," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018: ACM, pp. 207-212.

[14] R. Purisai, "How to choose a verification methodology," ed, 2004.

[15] M. Gad, M. Aboelmaged, M. Mashaly, and M. A. A. e. Ghany, "Efficient Sequence Generation for Hardware Verification Using Machine Learning," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 28 Nov.-1 Dec. 2021 2021, pp. 1-5, doi: 10.1109/ICECS53924.2021.9665495.

[16] M.-C. Cristescu and C. Bob, "Flexible Framework for Stimuli Redundancy Reduction in Functional Verification Using Artificial Neural Networks," in *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, 2021: IEEE, pp. 1-4.

[17] E. El Mandouh, A. Salem, M. Amer, and A. G. Wassal, "Cross-product functional coverage analysis using machine learning clustering techniques," in *2018 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*, 2018: IEEE, pp. 1-2.

[18] S. Sokorac, "Optimizing random test constraints using machine learning algorithms," in *Proceedings of the design and verification conference and exhibition US (DVCon)*, San Jose, CA, USA, 2017.

[19] Z. Xie, X. Wang, Z. Lian, Y. Luo, and Z. Hu, "A novel intelligent verification platform based on a structured analysis model," *Science China Information Sciences*, journal article vol. 56, no. 6, pp. 1-14, June 01 2013, doi: 10.1007/s11432-013-4817-6.

[20] R. Roy, C. Duvedi, S. Godil, and M. Williams, "Deep Predictive Coverage Collection," in *Proceedings of the design and verification conference and exhibition US (DVCon)*, San Jose, CA, USA, 2018.

[21] E. El Mandouh and A. G. Wassal, "Automatic generation of functional coverage models," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016: IEEE, pp. 754-757.

[22] E. El Mandouh and A. G. Wassal, "Accelerating the debugging of FV traces using K-means clustering techniques," in *2016 11th International Design & Test Symposium (IDT)*, 2016: IEEE, pp. 278-283.

[23] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026-8037, 2019.

[24] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825-2830, 2011.

[25] P. Mark. "Understanding AI: Inference." https://www.codemotion.com/magazine/dev-hub/machine-learning-dev/understanding-ai-inference/ (accessed April, 7, 2021).

[26] S. Pandey and V. Pande, "A node marking algorithm for partitioning wireless mesh networks," in *2011 IEEE Conference on Open Systems*, 2011: IEEE, pp. 363-368.

[27] A. Rodríguez, N. Botina, J. Gómez, and A. Diaconescu, "Improving data collection in complex networks with failure-prone agents via local marking," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 5081-5089, 2019.

[28] D. S. Hirschberg and L. L. Larmore, "Average case analysis of marking algorithms," *SIAM Journal on Computing*, vol. 15, no. 4, pp. 1069-1074, 1986.

[29] Y. Yusoff, M. S. Ngadiman, and A. M. Zain, "Overview of NSGA-II for optimizing machining process parameters," *Procedia Engineering*, vol. 15, pp. 3978-3983, 2011.

[30] R. Wojciech. "Implementation of NSGA-II algorithm in form of a Python library." https://github.com/wreszelewski/nsga2 (accessed May, 2, 2022).

[31] P. N. G. Bao, T. L. Quan, Q. T. Tho, and A. Garg. "Implementation of NSGA-II algorithm in form of a Python library." https://github.com/baopng/NSGA-II (accessed May 2, 2022).

[32] V. Pereira. "Project: Metaheuristic-SPEA-2." https://github.com/Valdecy/Metaheuristic-SPEA_2 (accessed May 4, 2022).

[33] R. Hale and B. Hutchings, "Enabling Low Impact, Rapid Debug for Highly Utilized FPGA Designs," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 27-31 Aug. 2018 2018, pp. 81-813.

[34] Y. Iskander, C. Patterson, and S. Craven, "High-level abstractions and modular debugging for FPGA design validation," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 1, pp. 1-22, 2014.

[35] R. Ma *et al.*, "Specializing FGPU for Persistent Deep Learning," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 8-12 Sept. 2019 2019, pp. 326-333.

[36] Xilinx. "Spartan-3E FPGA Starter Kit Board User Guide." https://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf (accessed July, 11, 2020).

[37] "IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)*, pp. 1-1315, 2018.

[38] H. Foster. "Part 10: The 2020 Wilson Research Group Functional Verification Study." Siemens. https://blogs.sw.siemens.com/verificationhorizons/2021/01/20/part-10-the-2020-wilson-research-group-functional-verification-study/ (accessed October 4, 2021).

ANEXE

Rezumatul tezei

Teza de doctorat *Contribuții la automatizarea verificării funcționale a sistemelor digitale* cuprinde un studiu aprofundat al modalităților de automatizare a verificării circuitelor integrate folosind inteligența artificială. Principalele realizări în urma activităților de cercetare sunt sistemele *software* create care au rolul de a transfera o parte din sarcinile comune din cadrul verificării funcționale algoritmilor de inteligență artificială. Lucrarea este structurată în șase capitole, dintre care primul conține o introducere în subiectul tezei alături de obiectivele acesteia. Capitolul doi conține informații relevante despre desfășurarea verificării conform abordărilor uzuale din industrie. Acesta este completat cu idei de automatizare a verificării funcționale și cu prezentarea rezultatelor unei selecții de lucrări relevante din domeniul automatizării verificării circuitelor integrate. În cadrul celui de-al treilea capitol se prezintă trei modalități de automatizare a procesului de verificare funcțională, demonstrându-se eficiența acestora în raport cu elemente caracteristice modalității clasice de verificare (ex. generarea aleatorie a stimulilor). În capitolul al patrulea se prezintă procesul dezvoltat pentru a se recompune și a se interpreta pe calculator datele provenite de pe placa cu FPGA, în scopul antrenării unor modele de inteligență artificială pe baza acestora. În al cincilea capitol se oferă detalii despre punerea la punct a unor activități didactice moderne care au rolul de a dezvolta competențe cheie pentru studenți, și a-i familiariza pe aceștia cu mediul de lucru pe care îl vor întâlni în companiile de profil. Ultimul capitol evidențiază concluziile tezei de doctorat și demonstrează atingerea obiectivelor propuse la începutul acesteia.

Abstract

The PhD thesis *Contributions to the automation of functional verification of digital systems* comprises an in-depth study of ways to automate the verification process of integrated circuits using artificial intelligence. The main achievements of the research are the software systems designed to transfer some of the common tasks in functional verification to artificial intelligence algorithms. The paper is structured in six chapters, the first of which contains an introduction to the subject of the thesis together with its objectives. Chapter two contains relevant information on the conduct of functional verification process according to common industry approaches. The chapter is complemented by a set of ideas related to functional verification automation and by the presentation of the results of a selection of relevant papers in the field of IC verification automation. The third chapter presents three ways of automating the functional verification process, demonstrating their effectiveness against the elements characteristic of the classical verification approach (e.g. random stimulus generation). The fourth chapter presents the process developed to recompose and interpret on the computer the data provided by FPGA board, to train artificial intelligence models based on them. The fifth chapter provides details of the development of modern teaching activities aiming to develop key skills for students, and to familiarize them with the working environment they will encounter in companies. The concluding chapter outlines the conclusions of the thesis and demonstrates the achievement of the proposed objectives.