Vyhodnocení efektivity SAT řešičů pro obvodový SAT (BP, DP)

Cílem práce je provést experimentální vyhodnocení efektivity (rychlosti) dostupných open-source řešičů problému splnitelnosti booleovské formule (SAT) pro instance získané transformací z obvodu (netlistu), tj. pro tzv. „circuit-SAT“. Tyto instance jsou svojí povahou specifické. Jsou sice „lehké“ (spíše se blíží 2-SATu), ale objevují se v nich těžké části. SAT řešiče se proto pro ně mohou chovat jinak, než pro běžné zkušební instance. Účinnost dostupných SAT řešičů pro tyto instance zatím nebyla dostatečně zkoumána.

Jedná se o čistě experimentální práci. Programování pravděpodobně nebude zapotřebí (kromě psaní skriptů pro spouštění experimentů). Generátor instancí je k dispozici. Naučíte se pracovat s výpočetním clusterem CESNET MetaCentrum (OS Linux).

Výstupem budou příslušné statistiky a doporučení.

Evaluation of SAT-Solvers Efficiency for Circuit-SAT

The aim of this work is to conduct an experimental evaluation of the efficiency (mainly the speed) of available open-source SAT solvers for circuit-SAT instances. These are instances produced by a transformation of a logic circuit to SAT (CNF). These instances are specific to some extent. In general they are “easy to solve”, however, there sometimes appear difficult parts. Thus, standard SAT-solvers may behave differently for these instances.

It is just an experimental work. Most probably, programming will not be needed (except of writing scripts).

The outputs of the work will be respective statistics and recommendations.

Určování míry podobnosti logických obvodů (BP, DP)

Vytvořte nástroj pro stanovování míry podobnosti logických obvodů. Vstupem budou dva obvody popsané logickou sítí (netlist), které jsou funkčně ekvivalentní, ale jejich struktura je odlišná. Cílem práce je navrhnout metriku "podobnosti" takovýchto logických sítí a implementovat algoritmus pro výpočet míry podobnosti. Proveďte rešerši stávajících řešení (zejména těch založených na grafových algoritmech) a vyberte vhodné kandidáty pro implementaci, případně navrhněte vlastní algoritmus. Berte v potaz zejména škálovatelnost těchto algoritmů. Zvolený algoritmus (algoritmy) implementujte a otestujte s použitím dostupných obvodů (dodá vedoucí práce). Výsledky porovnejte s jinými řešeními, např. s přístupem založeným na kontrole funkční ekvivalence podobvodů a s přístupy založenými na jiných grafových algoritmech (např. SimRank).

Measuring the Similarity of Logic Circuits

Create a tool for computing of similarity of two logic circuits. The input will be two functionally equivalent (but structurally different) circuits described by a netlist. Design a metric to compute similarity of these two circuits and implement an algorithm computing it.

Deeper knowledge of logic circuits design is not needed; it is rather a graph problem.

Randomizace algoritmů v systému ABC (BP, DP)

ABC je interaktivní open-source nástroj pro logickou syntézu vytvářený na University of California, Berkeley. Je v něm implementována řada syntézních algoritmů (jako je minimalizace, mapování na technologii, FPGA mapování, ...). Všechny tyto algoritmy jsou plně deterministické, tj. při opakovaném spouštění produkují stejné výsledky. Na druhou stranu jsou však citlivé např. na pořadí vstupů syntetizovaného obvodu, což svědčí o přítomnosti algoritmů, do kterých by mohlo být možné vložit prvek náhodného výběru, bez narušení funkčnosti. Cílem diplomové práce je vybrané algoritmy randomizovat, tj. do míst, kde se algoritmus rozhoduje mezi „lokálně ekvivalentními“ volbami, zavést náhodný výběr. Výsledné randomizované algoritmy otestujte na standardních zkušebních úlohách a porovnejte s původními deterministickými.

Randomization of Algorithms in ABC

ABC is an interactive open-source tool from UC California, Berkeley. Numerous logic synthesis and optimization algorithms implemented in it (minimization, technology mapping, etc.). All these algorithms are almost fully deterministic, i.e., they generate equal results when run repeatedly. The aim of the thesis work is to introduce randomness into some selected algorithms and perform the experimental evaluation.

Programming language: C. Detailed knowledge of the logic synthesis problematics is not essentially necessary. Which is necessary, it the ability to understand the code.

Dekompozice logických funkcí založená na binárních rozhodovacích diagramech (DP)

Vytvořte nástroj (resp. "framework") pro dekompozici logických funkcí založený na dekompozici binárních rozhodovacích diagramů (BDD). Je možné (resp. velice vhodné) použít části dostupných zdrojových kódů podobných nástrojů. Navrhněte heuristiky pro efektivní řízení dekompozice, analyzujte různé možnosti. Vytvořený nástroj otestujte na standardních zkušebních úlohách, získané výsledky porovnejte s alternativními řešeními.

BDD-based Decomposition of Logic Functions

Design a tool for decomposition of logic functions based on binary decision diagrams (BDDs). You may (should) re-use available source code from similar existing tools (BDS). Design heuristics for effective decomposition control, analyze different possibilities. The designed tool should be tested using standard benchmark circuits.

Generování testu pro kombinační obvody s ohledem na implementaci standardními buňkami (DP)

Cílem práce je navrhnout a naimplementovat algoritmus pro automatické generování testu (ATPG) pro kombinační obvody, s ohledem na finální implementací pomocí standardních buněk. Poruchový model je zde odlišný od klasického (trvalá 1/trvalá 0); uvažují se poruchy přímo na úrovni tranzistorů.
Klasické strukturní ATPG algoritmy pro tento poruchový model jsou známy (resp. rozšíření stávajících algoritmů je přímočaré). Tato práce by se měla zaměřit na využití principů ATPG založených na splnitelnosti booleovské formule (SAT-based ATPG) pro tento účel.
1) Nastudujte principy ATPG založených na splnitelnosti booleovské formule (SAT-based ATPG).
2) Nastudujte principy generování testu pro standardní buňky (Cell-Aware testing).
3) Společně s vedoucím práce navrhněte způsob, jak využít ATPG založené na splnitelnosti booleovské formule pro generování testu pro standardní buňky, případně proveďte rešerši, zda takové přístupy již neexistují.
4) Výše zmíněné naimplementujte a otestujte.

Komprese testu pro číslicové obvody s Illinois-Scan architekturou (DP)

Navrhněte a implementujte algoritmus pro kompresi testu pro tzv. „Illinois-scan“ dekompresní architekturu pro testování číslicových obvodů. Zde jsou testovací vektory doručovány do „scan-chainů“ paralelně, což způsobuje ztrátu pokrytí poruch. Původně navržený algoritmus se zaměřoval pouze na řazení klopných obvodů (DFF) ve „scan-chainech“, což je v současném návrhovém postupu nerealistické. Uvažujte tedy předepsané řazení DFF a soustřeďte se na výpočet vhodných testovacích vektorů, za účelem maximalizace pokrytí poruch a minimalizace délky testu. Použijte dostupné nástroje pro generování testovacích vektorů (ATPG).
Výsledný nástroj otestujte a algoritmus experimentálně vyhodnoťte z hlediska pokrytí poruch a délky testu.

Test Compression Based on the Illinois-Scan Architecture

Design and implement a test compression algorithm for the Illinois-scan decompression architecture used for testing digital circuits. Here the test vectors are delivered to multiple scan-chains in parallel, which may cause the fault-coverage loss. The originally proposed algorithm focused mainly on the flip-flops (DFFs) reordering in the scan-chains, which is infeasible in the contemporary digital design flow. Therefore, assume a pre-defined DFFs ordering, and focus on the test patterns generation instead, in order to maximize the fault coverage and minimize the test length. For this purpose, use available test generation tools (ATPGs).

Verify the functionality of the developed tool and perform experimental evaluation of the proposed algorithm, in terms of the fault coverage and the test length.

Implementace Packet-based komprese (BP, DP)

Cílem práce je naprogramovat stávající (publikovaný) algoritmus pro generování komprimovaných testovacích vektorů způsobem "Packet-based compression". Jedná se o čistě implementační práci, bez nutnosti hlubších znalostí problematiky. Výsledný program bude sloužit pro srovnání s nově vyvíjenými algoritmy. Preferovaný jazyk: C++ (je možné výhodně použít stávající framework).
Možno dále pokračovat jako VýLet/DP vytvořením nových algoritmů.

Implementation of Packet-based compression

The aim of this work is to implement the test patterns generation algorithm based on the "Packet-based compression" principle. It is a purely implementation work, no deeper knowledge of the problematic is required. Preferred language: C++ (it is possible to use an existing framework).

Implementace algoritmů pro výpočet řiditelnosti a pozorovatelnosti (BP, DP)

Ve stávajícím frameworku pro logickou syntézu (LogSynth) naprogramujte aplikaci pro výpočet řiditelnosti a pozorovatelnosti signálů v logickém obvodu. Jedná se o implementaci známých algoritmů (COP, SCOAP, algebraický přístup, PREDICT). Výsledky jednotlivých algoritmů srovnejte na standardních zkušebních obvodech.
Programovací jazyk: C++. Hlubší znalost problematicky číslicového návrhu není nutná.

Paralelní optimalizace logických obvodů (DP)

Vytvořte nástroj pro optimalizaci logických obvodů s možností využití více CPU jader. Logický obvod (netlist) bude rozdělen na části, které budou paralelně optimalizovány (resyntetizovány) jedním, případně více optimalizačními algoritmy. Výsledný obvod pak bude vytvořen opětovným složením těchto částí. Uvažujte paralelizmus ve smyslu rozdělení netlistu a paralelizmus ve smyslu aplikace různých optimalizačních algoritmů. Nástroj důkladně otestujte na standardních zkušebních úlohách a ověřte jeho efektivitu a škálovatelnost.
Doporučený jazyk: C++, MPI.

Reimplementace databáze publikací (DP)

Přeprogramujte stávající webovou aplikaci pro správu publikací a konferencí (webová databáze) do jazyka podporovaného ICT FIT, tj. JavaScript/TypeScript, Ruby, Rust, Java/Groovy nebo Clojure.
Původním jazykem je PHP (Nette Framework) a JavaScript.
Použitá databáze má být PostgreSQL nebo SQLite (původní je MySQL).
Dále vytvořte důkladnou dokumentaci.
Výslednou aplikaci důkladně otestujte, proveďte funkční a uživatelské testy.
Vzhledem k pravděpodobně náročnosti je zde možnost skupinového projektu.

Vylepšení webové aplikace pro vizualizaci pokročilých iterativních metod (BP, DP)

Do stávající webové aplikace pro vizualizaci pokročilých iterativních metod (simulované ochlazování, genetický algoritmus, tabu prohledávání) naimplementujte některá vylepšení a rozšíření. Jedná se zejména o:

- Podporu skriptování
- Agregaci dat
- Exaktní řešiče problémů
- Nové problémy k řešení
- Nové způsoby řízení algoritmů
- Oprava menších „bugů“
- Vylepšení uživatelského rozhraní

Proveďte důkladné funkční a uživatelské testy.
Vzhledem k rozsáhlosti tématu je zde možnost skupinového projektu. Ale lze se pochopitelně omezit jen na nějakou část.