GENETIC PROGRAMMING FOR DESIGN OF DIGITAL CIRCUITS

Michal Hejtmánek
Master Degree Programme, FIT BUT
E-mail: xhejtm03@stud.fit.vutbr.cz

Supervised by: Zbyšek Gajda
E-mail: gajda@fit.vutbr.cz

This article deals with a presentation of a method which is used for an automated circuit design. This method can design any digital circuit of several hundreds of gates. It will be shown an improvement of this method which increases success and effectivity of the method.

1 ÚVOD

Standardní inženýrské metody návrhu digitálních obvodů jsou obvykle založeny na matematických modelech, které vymezují typy a způsob použití hradel. Například metody využívající *booleovské* nebo *Reed-Mullerovy logiky* nejsou schopny využít hradel *XOR* [5].

Naproti tomu automatizovaný návrh pomocí *evolučních algoritmů* pracuje se zadanou množinou hradel libovolných typů zcela obecně a může tak převyšovat možnosti lidského návrháře v mnoha ohledech (škálovatelnost, složitost, efektivita, rychlost, cena, atd.).

2 EVOLUČNÍ ALGORITMY

Evoluční algoritmy (EA) patří do *stochastických metod moderní optimalizace* [4]. Na rozdíl od ostatních metod EA rozvíjí celou populaci kandidátních řešení současně a po vzoru biologické evoluce se v ní snaží působením selekčního tlaku a modifikacemi nejlepších jedinců vyvinout optimální řešení.

Pro návrh digitálních obvodů se nejčastěji využívají dvě varianty EA lišící se především strukturou jedince představujícího kandidátní řešení a způsobem jeho modifikace [3]:

Genetické programování (GP) obvykle pracuje se syntaktickými stromy nebo grafy libovolné délky, kterými kóduje proměnné i funkce. Diverzita populace je udržována především vzájemným křížením úspěšných jedinců, jenž lze doplnit i mutací. Obvykle pracuje s populací tisíců jedinců tisíce generací. Původně bylo GP navrženo pro automatizaci návrhu programů a symbolickou regresi.

Kartézské genetické programování (CGP) vzniklo později jako podmnožina GP, zaměřená přímo na vývoj obvodů. Používá chromozom pevně zvolené délky, kódující mřížku uzlů $m \times n$ *acyklicky orientovaného grafu*, která více odpovídá struktuře rekonfigurovatelných obvodů *FPGA*. Diverzita populace je v CGP udržována většinou pouze mutací a běžně se zde pracuje s populací několika desítek jedinců po milióny generací.

3 GENETICKÉ PROGRAMOVÁNÍ

Křížení v GP probíhá vytvořením kopií dvou vybraných rodičů, následovaným výměnou jejich náhodně zvolených podgrafů. Pravděpodobnost nalezení lepšího řešení touto cestou je zřejmě stejně velká jako v CGP, které místo křížení provádí mutaci (nahrazení náhodně vybraného podgrafu nově vygenerovaným). Tím se CGP zbavuje potřeby párování rodičů a nutnosti udržovat obrovskou populaci pro zajištění dostatečně pestrého počtu vzájemně vyměnitelných podgrafů. Další výhodou CGP je umožnění vývoje obvodu přímo na FPGA, kde dosahuje 20-50 násobné urychlení oproti běžnému vývoji na PC [5].

Bohužel, pravděpodobnost nalezení lépe fungujícího podgrafu, ať už náhodným výběrem nebo náhodným generováním, se postupem vývoje stále zmenšuje. Z průběhu testů, jejichž výsledky jsou shrnuty v kapitole 6, lze říci, že v průměru 95% času vývoje stráví algoritmus hledáním řešení splňujícího i posledních 5% pravdivostní tabulky. Čím je obvod lepší, tím je i množina jeho pozitivních změn menší, což často bývá příčinou *uváznutí vývoje v lokálním extrému*.

Rozšíří-li se kvalitní obvod v populaci příliš, stane se tak na úkor konkurence, ze které by se časem mohl vyvinout obvod lepší. V takovém případě je velmi nepravděpodobné, že se jen z něj skokově vyvine zcela odlišný a přece kvalitní obvod s širším potenciálem pro další vývoj ihned konkurenceschopný obvodu již rozšířenému, jenž se do současné podoby dlouhý čas vyvíjel.

Tyto problémy vedou k limitu oběma metodami navrhnutelných obvodů na několik set hradel a exponenciální nárůst velikosti pravdivostní tabulky s každým obvodovým vstupem omezuje (při testování všech možných kombinací vstupních hodnot) rozumný návrh na 8 vstupů [5].

4 STÁRNUTÍ RODŮ

Pro řešení těchto obtíží jsem navrhl rozšíření obecného algoritmu o „stárnutí rodů“. Rodovou linii tvoří obvody, které od svého předka zdědily kořenovou (tedy nejvlivnější) část *fenotypu*. Každý obvod si počítá kolikrát se křížil a každý potomek jeho rodové linie v tomto počtu pokračuje. Jen obvody lepší než jejich rodič začínají počítat zase od nuly a u ostatních, po překročení nastavené meze, dojde k penalizaci. V obecném algoritmu působením selekčního tlaku nahrazují lepší obvody v populaci ty slabší. Přidáním stárnutí jsou to i stejně kvalitní ale novější *fenotypy* místo starších a dlouho bezvýsledně kombinovaných. Tím je zajištěno postupné vymření celých rodových linií mimo větví, které prokázaly postup ve vývoji. Takovéto „omlazení“ populace pak umožňuje vyvinout se i doposud méně konkurenceschopným obvodům, snižuje riziko uváznutí na jediném *fenotypu* a zvyšuje rychlost konvergence algorimu.

5 IMPLEMENTACE

Algoritmus obecného GP (i s rozšířením) navrhující libovolné sekvenční digitální obvody na *úrovni logických hradel* jsem implementoval v jazyce C++. Pro inicializaci počáteční populace jsem vybral metodu *Ramped Half-and-Half* a pro výběr podle *standardizované hodnoty fitness*, definované na intervalu $<0, \infty)$ tak, že menší hodnota je lepší, *turnajovou selekci* [1, 2]. Hodnota fitness funkce (reprezentující kvalitu obvodu) odpovídá počtu nesplněných bitů pravdivostní tabulky požadovaného obvodu a je vyhodnocována paralelně po 64 bitech [5]. Jako nahrazovací strategii jsem použil *steady-state* s *elitismem*. Obvod s více výstupy vzniká odděleným pseudo-paralelním vývojem jednotlivých obvodových výstupů, které jsou následně pomocí heuristiky skládány dohromady.

6 TESTY

Implementované metody jsem otestoval na obvodech přiměřené složitosti ve 30 běžích při velikosti populace 5 000 jedinců pro každý obvodový výstup a množinou dvouvstupových hradel AND, OR, XOR, NOT. Vývoj obvodu byl ukončen nalezením požadovaného zapojení nebo dosažením limitu 10 000 generací. Úspěšnost návrhu funkčního obvodu společně s průměrným počtem potřebných generací obou algoritmů shrnuje tabulka 1.

testované obvody	obecný algoritmus		vylepšený algoritmus	
	úspěšnost	prům. počet generací	úspěšnost	prům. počet generací
násobička 3×3	17%	4189	33%	6157
sčítačka $3+3$	93%	1175	97%	970
multiplexor 4 na 1	100%	70	100%	69

Tabulka 1: Statistika obvodů navržených testovanými algoritmy.

Vylepšený algoritmus dosáhl ve všech případech vyšší úspěšnosti, a pokud se blížila 100%, tak i rychlejší konvergence. Nejmenší urychlení návrhu přinesl multiplexoru, u kterého se ve většině případů v krátké době vývoje nemohl vliv stárnutí ještě projevit. Za prodloužením průměrné doby vývoje bitové násobičky stojí dvojnásobný počet úspěšných návrhů, ze kterých je tato hodnota počítána. Stárnutí tedy dokázalo vymíráním rodů pokračovat ve vývoji i v situacích, ve kterých původní algoritmus již uvázl.

7 ZÁVĚR

Provedené experimenty potvrzují výchozí předpoklad pozitivního vlivu stárnutí na prevenci uváznutí i rychlost konvergence.

Dále hodlám pokračovat ve vývoji dalšího vylepšení založeného na výměně různých podgrafů se stejnou nadřazenou strukturou (tj. stejného rodu). Díky tomu by se mohlo snížit enormní množství zmetků vznikajících při dosavadní náhodné výměně podgrafů, protože vzájemně vyměněné stavební bloky budou mít i v obou potomcích podobnou funkci jako v rodičích.

REFERENCE

[1] Kvasnička, V., Pospíchal, J., Tiňo, P.: *Evolučné algoritmy.* STU Bratislava, 2000.

[2] Banzhaf, W, Nordin, P., Keller, R. E., Francone, F. D.: *Genetic Programming An Introduction on the Automatic Evolution of Computer.* Morgan Kaufman Publishers, Inc, 1998. Dokument dostupný na URL http://books.google.cz/books?id=1697qefFdtIC (3.2008)

[3] Bidlo, M.: *Biologií inspirovaný vývin jako technika evolučního návrhu*, In: Kognice a umělý život VII, Opava, CZ, SLU, 2007, s. 43-53.

[4] Štefka, D.: *Alternativy k evolučním optimalizačním algoritmům*, [Diplomová práce], České vysoké učení technické Praha, 2005.

[5] Sekanina, L.: *Biologií inspirované počítače*, Vysoké učení technické v Brně, 2007. Dokument dostupný na URL http://www.fit.vutbr.cz/study/courses/BIN/ (11.2007).