Evoluční návrh elektronických obvodů

Lukáš Sekanina

Evoluční algoritmus je pojem, který zastřešuje třídu prohledávacích algoritmů, do níž patří zejména genetické algoritmy, evoluční strategie, evoluční programování a genetické programování [1]. Tyto algoritmy se inspirují biologickým fenoménem evoluce a používají k prohledávání množinu kandidátních řešení (tzv. populaci). Nové kandidátní jedince vytvářejí pomocí operátorů, jejichž činnost je inspirována genetikou (operacemi mutace a křížení).

Evoluční algoritmy jsou používány k řešení obtížných optimalizačních problémů již po několik desetiletí. S příchodem genetického programování začínají být evoluční postupy rutinně používány také při vypracovávání návrhu, kdy je namísto pouhé optimalizace několika koeficientů předem zvolených v systému automatizovaně budována celá struktura řešení. Mezi nejzajímavější výsledky evolučního návrhu z oblasti elektroniky a výpočetní techniky se řadí zejména různé analogové filtry, regulátory, číslicové obrazové operátory, antény, optické systémy a komunikační protokoly (viz shrnující publikace [2], [3] a [13]). U konkrétních evolučně navržených realizací bylo prokázáno, že fungují lépe (podle stanoveného kritéria) než nejlepší doposud známá řešení vytvořená konvenčními návrhovými technikami. Za použití evolučních technik je rovněž možné adaptovat či opravovat hardware za běhu aplikace.

V tomto článku bude nejdříve vysvětlen princip evolučního návrhu a poté představeny příklady evolučního návrhu v oblasti číslicových obvodů. Půjde o výsledky vytvořené skupinou EHW@FIT na Fakultě informačních technologií VUT v Brně. Nejdříve však definujme problém, který bude řešen, a přibližme konvenční metodu jeho vyřešení.

Konvenční a evoluční návrh obvodů

Předpokládejme, že je třeba s využitím elektronických obvodů realizovat logickou funkci o n vstupech, která je definována pomocí pravdivostní tabulky. Z pravdivostní tabulky je možné odvodit kanonický tvar v uvažovaném výpočetním modelu (např. výraz v běžné disjunktní formě, úplný binární rozhodovací diagram apod.). Cílem logické syntézy je obvykle najít takové vyjádření logické funkce v uvažovaném modelu, které minimalizuje počet hradel, zpoždění, spotřebu apod. V rámci zvoleného modelu existuje množina transformací, které je možné na logický výraz aplikovat, a tím měnit jeho tvar. Nikdy však nesmí dojít ke změně evaluace logické funkce, kterou výrazy reprezentují. Zvolený model a s ním spojené transformace určují prostor možných řešení. Konvenční algoritmy používané v rámci syntézy (např. Espresso, ABC apod.) jsou navrženy tak, že dokážou nalézt co nejvýhodnější řešení, ale pouze v rámci zvoleného modelu, tj. v předem vymezeném podprostoru možných řešení.

Evoluční návrh vychází z toho, jaké zdroje jsou dostupné pro implementaci na konkrétní platformě. Jde zejména o typ výpočetních prvků, jejich možné funkce, počty vstupů a výstupů a možnosti propojování. Podle konkrétní cílové platformy je pak vytvořena taková reprezentace obvodu, která by měla umožnit postihnout libovolný obvod z množiny všech vytvořitelných obvodů. Tím je definován prohledávací prostor pro evoluční algoritmus. V rámci tohoto prostoru mohou existovat velmi výhodné implementace požadované logické funkce, které nejsou dosažitelné v konvenčních modelech.

Kartézské genetické programování

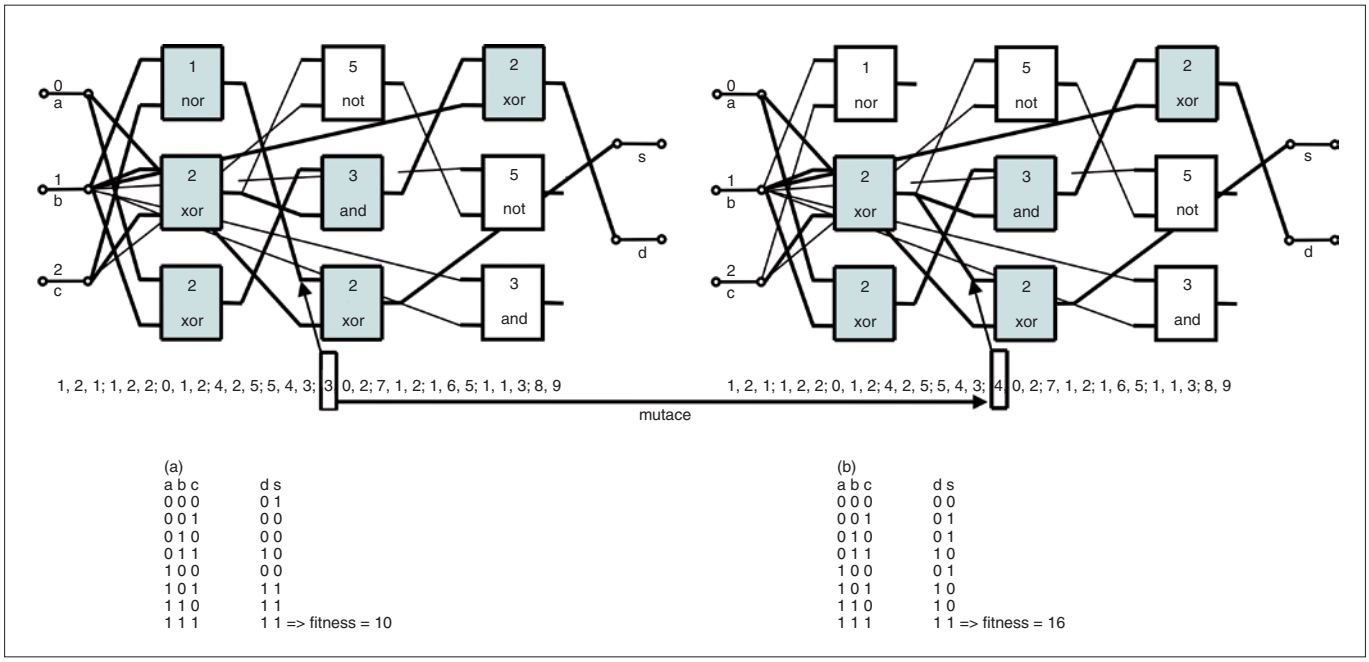
Kartézské genetické programování (*Cartesian Genetic Programming* – CGP) je vhodné pro navrhování grafových struktur, a tedy i např. schémat číslicových obvodů [4]. Jde o variantu genetického programování, u které jsou kandidátní řešení reprezentována pomocí orientovaných grafů. Kandidátní obvod je modelován jako pole programovatelných prvků (uzlů grafu) o velikosti $n_c \times n_r$ (počet sloupců × počet řádků). Označme počet primárních vstupů obvodu n_i a počet primárních výstupů obvodu n_o. Každý z uzlů, který může mít až n_t vstupů, realizuje právě jednu funkci vybranou z množiny dostupných funkcí F. Vstupy uzlu, který se nachází v j-tém sloupci, mohou být připojeny buď na primární vstupy obvodu, nebo na výstupy uzlů umístěných až v L předchozích sloupcích. Uzly stejného sloupce se nesmí propojovat. Primární výstupy mohou být připojeny k libovolnému uzlu obvodu.

Na *obr. 1a* je příklad instance CGP. Každý kandidátní obvod je zakódován pomocí řetězce celočíselných hodnot, který se nazývá genotyp (nebo chromozom). Způsob kódování je následující: Každému primárnímu vstupu obvodu je přiřazen index z intervalu $0, \ldots, n_i - 1$. K výstupům uzlů jsou postupně rovněž přiřazeny indexy, a to po sloupcích, s počáteční hodnotou n_i pro nejlevější horní uzel. Každý uzel obvodu je kódován pomocí $n_t + 1$ celočíselných hodnot. Prvních n_t hodnot určuje indexy uzlů, ke kterým budou připojeny vstupy uvažovaného uzlu. Poslední hodnota určuje kód logické funkce uzlu. Na konci chromozomu je n_o hodnot definujících indexy uzlů, ke kterým jsou připojeny primární výstupy obvodu. Základní vlastností uvedeného kódování je, že zatímco délka chromozomu je vždy konstantní, velikost zakódovaného obvodu (fenotypu) je variabilní. Množina všech chromozomů určuje prohledávací prostor, ve kterém pracuje evoluční algoritmus.

Základní varianta prohledávacího algoritmu, která je využita v CGP, používá populaci o velikosti l jedinců (obvykle $l = 5$) a jediný genetický operátor – mutaci, který pracuje tak, že náhodně vybere h genů (tj. h hodnot z chromozomu) a náhodně vygeneruje jejich nové hodnoty (*obr. 1b*). Pomocí mutace vznikají nová kandidátní řešení problému. Počáteční populace je vygenerována náhodně nebo ji tvoří předem připravená řešení problému. Kvalita každého kandidátního řešení je ohodnocena prostřednictvím účelové funkce (tzv. funkce *fitness*). V případě návrhu malých kombinačních obvodů je např. vypočteno, kolik je schopen kandidátní obvod vyprodukovat správných bitů pro všechny možné kombinace vstupních signálů. Například pro obvod, který má tři vstupy a dva výstupy, bude maximální hodnota *fitness* $2 \cdot 2^3 = 16$. Nová populace je vytvářena tak, že je ze staré populace vybrán jedinec s nejlepší hodnotou *fitness*, který je vložen do nové populace spolu se svými $l - 1$ mutanty. V CGP se používá jedno důležité pravidlo: Jestliže existuje více „nejlepších“ jedinců se stejnou hodnotou *fitness*, použije se jako rodič ten, který nebyl rodičem v předchozí generaci. Tímto je podporována genetická diverzita populace. Evoluce končí nalezením dostatečně kvalitního jedince nebo vyčerpáním povoleného počtu generací.

Kartézské genetické progarmování, pracující metodou „generuj a testuj“, přechází od jednoho řešení ke druhému pomocí genetického operátoru mutace. Obecně není garantováno, že rodič a potomek mají stejnou evaluaci, a představují tak logicky správnou implementaci zadané pravdivostní tabulky (tj. že realizují požadovanou funkci). K výslednému logicky správnému řešení se přichází postupným vylepšováním existujících řešení v důsledku selekčního tlaku vytvářeného funkcí *fitness*.

Jedním z prvních použití CGP byl evoluční návrh kombinačních sčítaček a násobiček [4], [5]. Bylo prokázáno, že CGP dokáže najít násobičky (pro dvou- až čtyřbitové operandy), které obsahují v průměru o 18 % méně dvouvstupových hradel než násobičky získané konvenčními algoritmy. Velkou výhodou CGP je, že umožňuje pracovat s komponentami, které nejsou podporovány v současných návrhových metodách. Bylo-li by třeba do obvodu zahrnout např. polymorfní hradla (viz odstavec Evoluční návrh polymorfních



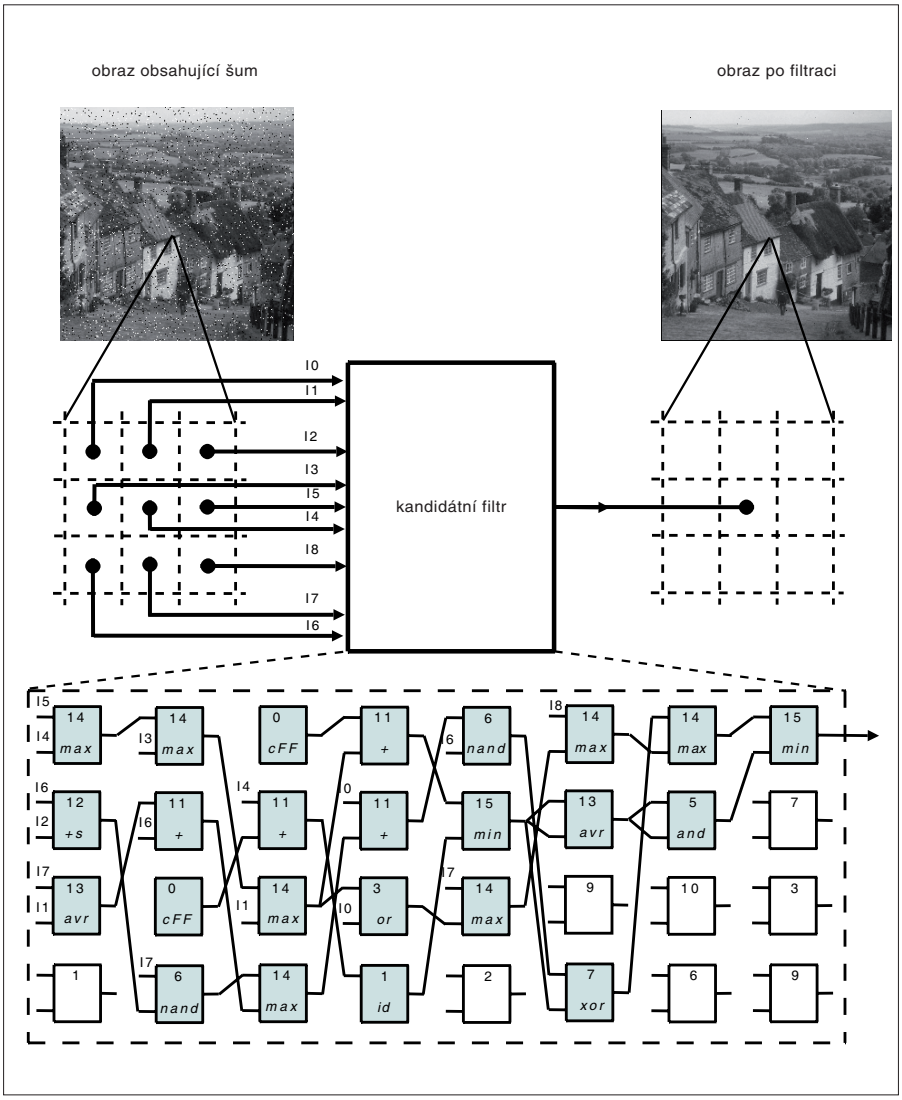
Obr. 1. a) obvod reprezentovaný pomocí CGP (s parametry: $n_c \times n_r = 3 \times 3$, $n_i = 3$, $n_o = 2$, $n_t = 2$, $L = 3$) a jeho pravdivostní tabulka, b) obvod po mutaci jednoho genu pracuje jako úplná jednobitová sčítačka (fitness = 16)

obvodů), která mění logickou funkci v závislosti na úrovni napájecího napětí, je evoluční návrh v současnosti jedinou technikou, jak získat kvalitní řešení.

Za hlavní nevýhodu evolučního návrhu je považována skutečnost, že popsaná metoda funguje jen pro relativně jednoduché obvody. Jestliže se totiž bude zvyšovat počet vstupů obvodu a současně bude požadováno ověření činnosti obvodu pro každý možný vstupní vektor, bude se doba evaluace kandidátního řešení prodlužovat exponenciálně. Dalším problémem je prodlužování chromozomu, které je způsobeno požadavkem na vytváření velkých obvodů. Dlouhé chromozomy implikují velké prohledávací prostory, ve kterých evoluční algoritmy nedokážou efektivně nacházet inovativní implementace. Pro obvody s přibližně deseti až patnácti vstupy již evoluce nenajde rozumné řešení vůbec. Proto byly zavedeny různé techniky, které umožňují zmíněným problémům se vyhnout. Na následujících příkladech budou některé z nich demonstrovány na reálných úlohách.

Evoluční návrh obrazových operátorů

K potlačení šumu v obraze nebo k detekci hran se používají lineární i nelineární číslicové filtry, které pracují tak, že novou hodnotu filtrovaného obrazového bodu (pixelu) počítají na základě předešlé (potenciálně poškozené) hodnoty pixelu a jeho nejbližších sousedů. Na *obr. 2* je ukázán příklad filtru, jenž používá devět pixelů na vstupu, které spolu tvoří tzv. filtrační masku o velikosti 3 × 3 pixely. Jestliže budou pixely reprezentovány na osmi bitech, lze takový filtr chápat jako číslicový obvod s devíti osmibitovými vstupy a

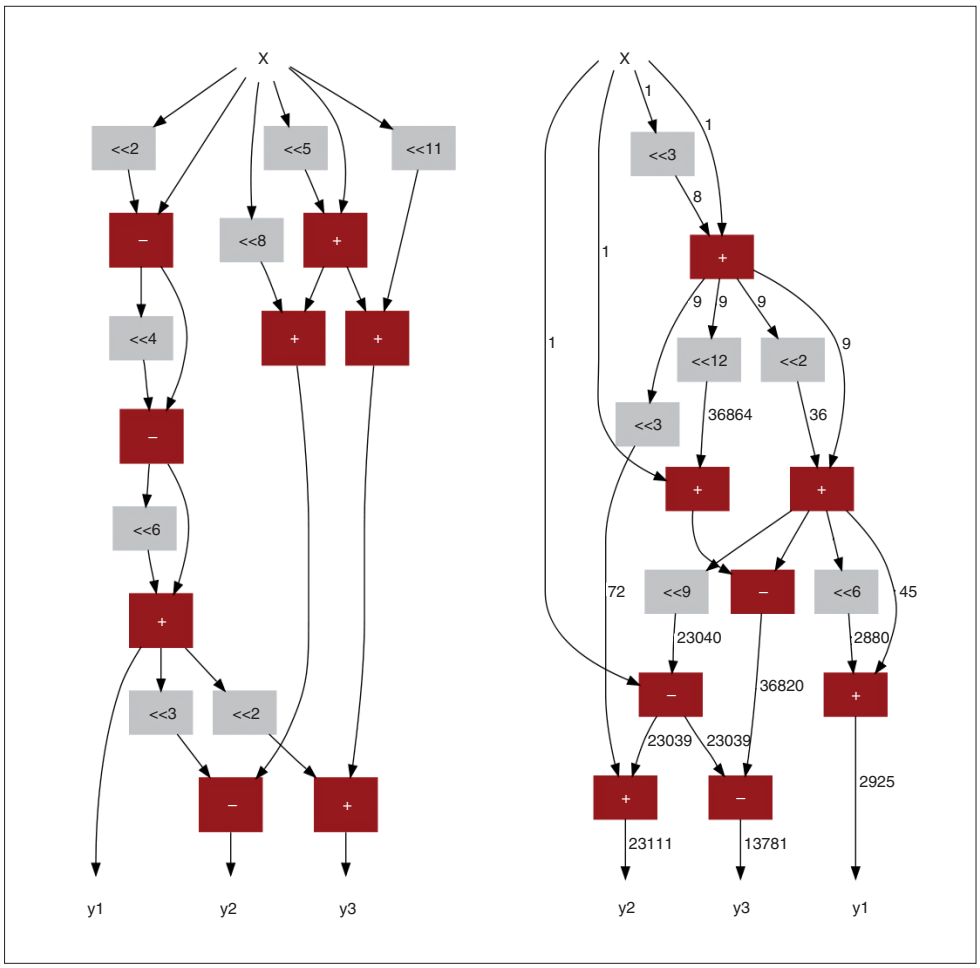


Obr. 2. Evoluční návrh obrazového filtru pomocí CGP

osmibitovým výstupem. Evoluční návrh nelineárního filtru, který má potlačit např. výstřelový šum, může být proveden pomocí CGP. Je však třeba udělat několik modifikací. Zaprvé nebude kandidátní filtr reprezentován na úrovni hradel, ale na úrovni funkčních bloků, kterými jsou typicky obvody počítající průměr, minimum a maximum dvou osmibitových hodnot apod. Reprezentace na úrovni funkčních bloků umožňuje pracovat s relativně krátkým chromozomem (tj. malým prohledávacím prostorem), který by při práci na úrovni hradel několikanásobně narostl. Druhá modifikace vychází z toho, že v rámci evaluace kandidátního filtru není možné ověřit odezvu filtru pro všechny možné vstupní kombinace. Proto je v průběhu evoluce použit vhodně zvolený trénovací obraz o velikosti např. 128 × 128 pixelů. Cílem evoluce je minimalizovat odchylku ideální verze trénovacího obrázku od obrázku filtrovaného kandidátním filtrem. Funkčnost a obecnost evolučně navrženého filtru je na konci evoluce ověřena za použití sady testovacích obrázků.

V pracích [6], [7] bylo prokázáno, že evolučně navržené filtry pro různé typy šumu (např. sůl a pepř, dávkový výstřelový šum) a detektory hran dosahují ve srovnání s konvenčními řešeními (mediánové filtry, Sobelův detektor apod.) minimálně stejné vizuální kvality výsledku, tedy poměru PSNR (*Peak Signal to Noise Ratio)* změřeného na sadě obrázků. Navíc tyto filtry, pokud by byly implementovány na čipu, zaberou podstatně méně plochy než konvenční řešení. Typickým příkladem je evolučně navržený filtr určený pro potlačení šumu typu sůl a pepř (s intenzitou 5 až 10 %), který dosahuje o 10 % vyšší hodnoty PSNR a potřebuje o 25 % menší plochu na čipu než standardní mediánový filtr. Jeden z vytvořených filtrů je v ČR chráněn užitným vzorem UV20017/2009.

Pro evoluční návrh filtrů byl vytvořen akcelerátor v programovatelném hradlovém poli FPGA Xilinx Virtex II Pro (100 MHz), který umožňuje urychlit evoluční návrh 44krát oproti implementaci běžící v procesoru Celeron 2,4 GHz [7]. Tento akcelerátor pracuje s polem 8 × 4 programovatelných bloků, přičemž délka chromozomu je 384 bitů. Trénovací obrázky o velikosti 128 × 128 pixelů jsou uloženy v externích pamětech SRAM. Genetické operace jsou prováděny procesorem PowerPC, který je vestavěn v FPGA. Za jednu sekundu dokáže akcelerátor ohodnotit až 6 000 kandidátních filtrů. Jeden běh evoluce je dokončen v průměru do 10 s.



Obr. 3. a) konvenční realizace násobičky pro koeficienty (23111, 13781, 2925) podle [9], b) evolučně vytvořené řešení, které má o jednotku menší zpoždění a obsahuje o dva posuvy méně

Evoluční návrh testovacích obvodů

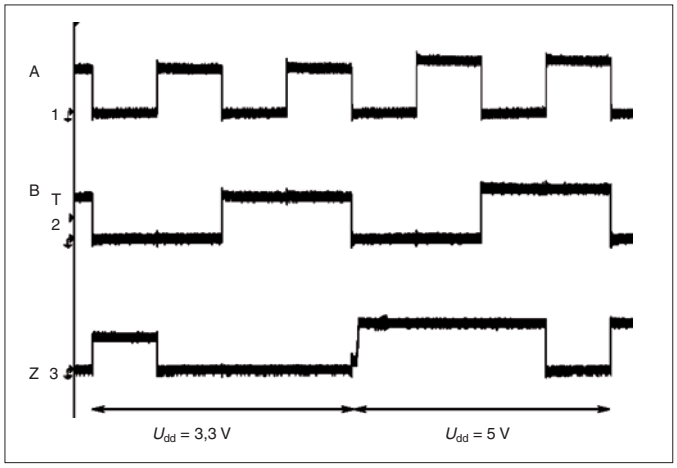
Následující příklad ukazuje, že je evoluční přístup schopen vytvářet obvody skládající se (po syntéze) z milionu i více hradel. Jde o evoluční návrh testovacích obvodů s předem definovanou mírou testovatelnosti. Testovatelnost obvodu je odhadována a následně numericky vyčíslována (např. v rozsahu 0 až 100 %) za použití metod analýzy testovatelnosti, které mají za úkol detekovat obtížně testovatelné části obvodu. Protože analýza testovatelnosti vlastně představuje výpočetně nepříliš obtížnou analýzu určitých strukturních vlastností grafu, kterým je obvod modelován, může být provedena s kvadratickou časovou složitostí, a tudíž relativně rychle i pro velké obvody.

Cílem evolučního návrhu je automaticky vygenerovat testovací obvody na úrovni meziregistrových přenosů a s požadovanou průměrnou řiditelností a pozorovatelností, což jsou indikátory testovatelnosti. Vstupem pro evoluční návrh je uživatelem zadaný počet primárních vstupů a výstupů obvodu, počet a typ prvků obvodu a požadavek na průměrnou řiditelnost a pozorovatelnost. Výstupem jsou syntetizovatelné obvody s požadovanou úrovní testovatelnosti a s požadovaným počtem hradel. Metoda byla použita při návrhu sady testovacích obvodů s různým počtem hradel a s různými parametry pozorovatelnosti a řiditelnosti, která je k dispozici zájemcům na internetu [8]. Nejsložitější z navržených obvodů obsahuje po syntéze 1,2 milionu hradel. Tento obvod je rovněž největším obvodem, který byl (podle vědomí autorů) vytvořen pomocí evolučního návrhu.

Evoluční návrh násobiček s konstantními koeficienty

V oblasti číslicového zpracování signálu je často zapotřebí násobit přicházející a potom částečně zpracovaný číslicový signál pevně zvolenou konstantou nebo několika konstantami. Protože jedním z úkolů návrháře je pokud možno minimalizovat plochu čipu, bývají v této úloze obecně použitelné násobičky, které jsou poměrně složité, nahrazeny speciálními násobičkami s konstantními koeficienty. Tyto násobičky jsou realizovány prostřednictvím operací sčítání, odčítání a posuvu (tj. násobení mocninou 2). Pro řešení této úlohy existuje velmi kvalitní heuristika [9].

Kartézské genetické programování, které pracuje na úrovni funkčních bloků, je rovněž možné použít k vypracování návrhu takových násobiček [10]. Tato úloha je zajímavá především tím, že je možné velmi rychle ohodnotit potenciálně poměrně složité kandidátní řešení (např. obvod obsahující desítky sčítaček, jeden šestnáctibitový vstup a dvacet šestnáctibitových výstupů). Obsahuje-li totiž obvod pouze uvedené lineární komponenty, stačí pro jeho evaluaci vypočítat výstup pro jedinou, vhodně zvolenou hodnotu na vstupu



Obr. 4. Chování polymorfního hradla NAND/NOR, které pro U_{dd} = 3,3 V realizuje funkci Z = A nor B a pro U_{dd} = 5 V realizuje funkci Z = A nand B

(např. $x = 1$). Ve funkci *fitness* je pak sečtena odchylka všech výstupů od požadovaných hodnot. Při dosažení plně funkčního řešení (hodnota *fitness* je nulová) je minimalizován počet komponent násobičky. V *obr. 3* je porovnáno řešení nalezené heuristikou s řešením získaným pomocí CGP pro násobičku s třemi výstupy. CGP se ukazuje jako kvalitní metoda zejména pro redukci zpoždění těchto násobiček.

Evoluční návrh polymorfních obvodů

Polymorfní obvody obsahují kromě běžných hradel i tzv. polymorfní hradla, která se vyznačují tím, že mohou měnit svoji logickou funkci v závislosti na teplotě, úrovni napájecího napětí nebo podle jiného externího stimulu [11]. Na Fakultě informačních technologií (FIT) VUT v Brně vznikl rekonfigurovatelný polymorfní čip REPOMO32, který obsahuje kromě běžných hradel i polymorfní hradla NAND/NOR, jejichž funkce je řízena úrovní napájecího napětí U_{dd} [12]. Pokud je U_{dd} = 3,3 až 3,8 V, hradlo pracuje jako NOR, pro U_{dd} = 3,9 až 5 V pracuje jako NAND (*obr. 4*). Ostatní hradla realizují jedinou logickou funkci pro celý rozsah napájecího napětí. K výrobě byla použita technologie AMIS 0,7 µm. REPOMO32 má čtyři vstupy, osm výstupů a obsahuje 32 konfigurovatelných prvků. Tento čip je možné použít k demonstraci použití polymorfní elektroniky a pro studium vlastností polymorfních obvodů.

Kartézské genetické programování, bylo použito pro evoluční návrh polymorfních obvodů na úrovni hradel. Typickým příkladem je násobička s konstantními koeficienty, která pro jednu úroveň napájecího napětí produkuje součin ax a pro druhou úroveň napájecího napětí vytváří součin bx, kde a a b jsou zadané konstanty a x je hodnota vstupního signálu. Tyto násobičky je možné použít pro elegantní realizace polymorfních rekonfigurovatelných filtrů FIR. Pomocí polymorfních hradel a evolučních technik byla dále navržena samočinně se testující sčítačka a vytvořena metoda pro redukci počtu testovacích vektorů [13].

Shrnutí

Členové týmu z FIT VUT v Brně pracovali i na dalších projektech. Například pomocí lineárního genetického programování byl objeven algoritmus pro tvorbu řadicí sítě s počtem vstupů $n + k$ z řadicí sítě, která má n vstupů. Navržený algoritmus vede k řadicím sítím s lepšími parametry než srovnatelná konvenční řešení. Ve spolupráci s Fakultou informatiky MU v Brně bylo lineární genetické programování využito k vytvoření protokolů pro amplifikaci bezpečnosti v bezdrátových senzorových sítích. Navržené protokoly umožní zvýšit počet zabezpečených linek ve srovnání s řešením, které je dosažitelné existujícími protokoly. Genetický algoritmus byl využit pro optimální výběr modulů, které se liší zpožděním a plochou, v úloze počáteční syntézy číslicového obvodu. Monografie [13] shrnuje současný stav evoluční elektroniky ve světě, včetně výsledků výzkumného týmu FIT VUT v Brně.

Evoluční návrh se ukazuje jako velmi zajímavá alternativa konvenčního návrhu, zejména jestliže existuje kvalitní a rychlý simulátor použitelný pro evaluaci kandidátních řešení, a dále v případech, kdy nelze specifikaci řešení vyjádřit přesně (např. u obrazového filtru), nebo kdy konvenční metody návrhu nejsou plně automatizovatelné a vyžadují experimentování. Za největší nevýhody evolučního návrhu je obvykle považována vysoká výpočetní náročnost evoluce, nemožnost garantovat nalezení dostatečně kvalitního řešení v každém běhu algoritmu a obtížná interpretovatelnost nalezeného řešení.

Poděkování

Tento výzkum proběhl v rámci výzkumného záměru MSM0021630528 – Výzkum informačních technologií z hlediska bezpečnosti.

Literatura:

[1] ZELINKA, I. et al.: *Evoluční výpočetní techniky – principy a aplikace*. BEN technická literatura, 2009.

[2] HIGUCHI, T. – LIU, Y. – YAO, X.: *Evolvable hardware*. Berlin, Springer, 2006.

[3] LOHN, J. D. – HORNBY, G. S.: *Evolvable hardware: Using evolutionary computation to design and optimize hardware systems*. IEEE Computational Intelligence Magazine 2006, 1(1), s. 19–27.

[4] MILLER, J. – JOB, D. – VASSILEV, V.: *Principles in the Evolutionary Design of Digital Circuits - Part I.* Genetic Programming and Evolvable Machines, 2000, 1(1), s. 8–35.

[5] VASSILEV, V. – JOB, D. – MILLER, J.: *Towards the Automatic Design of More Efficient Digital Circuits*. In: Proc. of the 2nd NASA/ /DoD Workshop on Evolvable Hardware, IEEE Computer Society, 2000, s. 151–160.

[6] SEKANINA, L.: *Evolvable components: From Theory to Hardware Implementations*. Berlin, Springer, 2004.

[7] VAŠÍČEK, Z. – SEKANINA, L.: *An Evolvable Hardware System in Xilinx Virtex II Pro FPGA*. International Journal of Innovative Computing and Applications, 2007, 1(1), s. 63–73.

[8] PEČENKA, T. – SEKANINA, L. – KOTÁSEK, Z.: *Evolution of Synthetic RTL Benchmark Circuits with Predefined Testability*. ACM Transactions on Design Automation of Electronic Systems, 2008, 13(3), s. 1–21.

[9] VORONENKO, Y. – PÜSCHEL, M.: *Multiplierless Multiple Constant Multiplication*. ACM Transactions on Algorithms, 2007, 3(2), s. 1–28.

[10] VAŠÍČEK, Z. – ŽÁDNÍK, M. – SEKANINA, L. – TOBOLA, J.: *On Evolutionary Synthesis of Linear Transforms in FPGA*. In: Evolvable Systems: From Biology to Hardware, LNCS 5216, Springer Verlag, 2008, s. 141–152.

[11] SEKANINA, L. – STAREČEK, L. – KOTÁSEK, Z. – GAJDA, Z.: *Polymorphic Gates in Design and Test of Digital Circuits*. Journal of Unconventional Computing, 2008, 4(2), s. 125–142.

[12] SEKANINA, L. – RŮŽIČKA, R. – VAŠÍČEK, Z. – PROKOP, R. – FUJCIK, L.: *REPOMO32 – new reconfigurable polymorphic integrated circuit for adaptive hardware*. In: Proceedings of 2009 IEEE Workshop on Evolvable and Adaptive Hardware, IEEE CIS, 2009, s. 39–46.

[13] SEKANINA, L. a kol.: *Evoluční hardware*. Academia, Praha, 2009.

Lukáš Sekanina,
Fakulta informačních technologií,
VUT v Brně
(sekanina@fit.vutbr.cz)

recenzovali: prof. Ing. Vilém Srovnal, CSc.,
FEL, VŠB-TU Ostrava;
doc. Ing. Ivan Zelinka, Ph.D.,
FAI, UTB ve Zlíně