VÝPOČETNÍ DEVELOPMENT ZALOŽENÝ NA INSTRUKCÍCH

Michal Bidlo
Informační technologie, 3. ročník, denní studium
Školitel: Lukáš Sekanina

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 61266 Brno, Česká republika

bidlom@fit.vutbr.cz

Abstrakt. Příspěvek pojednává o specifické oblasti evolučního návrhu využívajícího development založený na instrukcích. Tato technika se ukázala být vhodným kandidátem v oboru evolučního návrhu číslicových obvodů s ohledem na škálovatelnost cílových řešení. Jsou diskutovány dvě oblasti aplikací tohoto přístupu: (1) evoluční návrh generických řadicích sítí a (2) evoluční návrh generických kombinačních násobiček. V prvním případě bylo evolucí nalezeno několik programů, které jsou schopny konstruovat řadicí sítě s lepšími vlastnostmi v porovnání s konvenční metodou. Druhá aplikace představuje vůbec první experiment evolučního návrhu, kdy byl automaticky nalezen algoritmus pro konstrukci generických násobiček s využitím developmentu. V závěru jsou diskutovány přínosy těchto technik a nastíněna struktura dizertační práce.

Klíčová slova. Evoluční algoritmus, instrukce, program, development, číslicový obvod.

1 Úvod

Biologií inspirované algoritmy se v poslední době stále více uplatňují při řešení mnoha komplexních úloh v různých oblastech vědy. Evoluční návrh patří k relativně mladému a rozvíjejícímu se odvětví, které nám dosud umožnilo získat mnoho zajímavých výsledků. Jako příklad uveďme kreativní evoluční návrh nebo formy umělého života [1], číslicové obvody [2], neuronové sítě [3], analogové elektronické obvody nebo počítačové programy [4]. Přestože bylo aplikováno několik odlišných metod evolučních algoritmů, bylo možné takto získat pouze relativně jednoduchá řešení.

Problém škálovatelnosti patří pravděpodobně k nejvýznamnějším aspektům, které znesnadňují proces evolučního návrhu s využitím současných technologií. S narůstající složitostí cílového systému (např. v podobě počtu vstupních a výstupních veličin nebo počtu komponent potřebných k jeho implementaci) vzrůstá délka chromozomu reprezentujícího kandidátní řešení během evolučního procesu, což má za následek zvětšení vyhledávacího prostoru a je tudíž velmi náročné navrhnout efektivní prohledávací (návrhový) algoritmus.

Dosud bylo vyvinuto několik odlišných přístupů snažících se problém škálovatelnosti překonat. Patří mezi ně *evoluce na úrovni funkčních bloků* (složitějších celků tvořících stavební kameny cílového systému -- viz např. [5]), *inkrementální evoluce* (viz např. Toressenova metoda *rozděl a panuj* [6]) a tzv. *development* (postupný *vývin* cílového systému podle určitých pravidel - inspirován biologickými principy ontogeneze, viz např. [7, 8, 9]).

Cílem tohoto příspěvku je prezentace přístupu realizace developmentu založeného na instrukcích, v kombinaci s evolučními algoritmy, který umožňuje konstrukci generických číslicových obvodů různých tříd. Funkčnost metody bude demonstrována na aplikacích návrhu libovolně velkých řadicích sítí a kombinačních násobiček.

2 Evoluční algoritmy a evoluční návrh

Evoluční algoritmy jsou stochastické vyhledávací systémy využívající principů Darwinovy evoluční teorie. Pod tímto označením však obvykle nalezneme celou řadu modifikací různých biologií inspirovaných technik, z nichž zřejmě nejznámější jsou genetické algoritmy [12], genetické programování [10], evoluční strategie [13, 14] a evoluční programování [15]. Tyto typy evolučních algoritmů se navzájem liší zejména ve způsobu reprezentace *kandidátních řešení* (potenciálních řešení daného problému) a jejich zpracování v průběhu umělého evolučního procesu. Kandidátní řešení (tzv. *fenotypy*) jsou zakódována do c*hromozomů* představujících prvky vyhledávacího prostoru algoritmu (tzv. *genotypy*). Chromozom se též obvykle nazývá *jedinec*. Podoba vyhledávacího prostoru závisí na řešeném problému, kterým může být například optimalizace parametrů číslicového filtru, optimalizace tvaru lopatky parní turbíny, návrh počítačových programů, kombinačních logických obvodů atd. Míra úspěšnosti kandidátního řešení je měřena pomocí tzv. *fitness funkce*, která pro daný problém vyhodnotí výsledek řešení s využitím informací obsažených v chromozomu. Chromozomy podstupují modifikace *genetickými operátory* (typicky křížení a mutace – inspirováno v biologii) zajišťujícími výměnu genetických informací mezi chromozomy a tvorbu nových kandidátních řešení (*potomků*). Evoluční proces sestává z posloupnosti generací, v pruběhu kterých jsou pro aplikace genetických operátorů upřednostňováni kvalitní jedinci, kteří mají větší šanci „přežít“ do dalších generací a vyprodukovat tak více kvalitnějších potomků. V důsledku tohoto procesu, reprezentujícího tzv. *selekční tlak*, dochází k postupnému zlepšování kvality hledaných řešení.

Evoluční algoritmy bývají v posledních letech hojně využívány k řešení náročných problémů v různých oblastech. Jejich aplikace je možné rozdělit podle účelu využití evolučního algoritmu na (1) e*voluční optimalizace* a (2) *evoluční návrh*. Zásadní odlišnost mezi těmito přístupy je následující: Evoluční optimalizace využívá znalostí o již hotovém fungujícím systému, z něhož je extrahována množina parametrů ovlivňujících kvalitu jeho funkce. Evoluční algoritmus je v tomto případě využit k nalezení vhodné konfigurace těchto parametrů tak, aby daný systém fungoval co nejefektivněji, tj. evoluce je využita k optimalizaci daných vlastností již známého systému. Cílem evolučního návrhu je však vlastní *design* samotného systému. Na jeho počátku je známa pouze požadovaná funkce systému, množina komponent, ze kterých má být cílový systém sestaven, a pravidla jejich možného propojení. Evoluční algoritmus je v takovém případě využit k *návrhu* systému s využitím těchto znalostí. Do jisté míry lze oba přístupy kombinovat, např. v případě návrhu logických obvodů je možné klást důraz na řešení s menšími počty hradel, nižším zpožděním apod. Je zřejmé, že evoluční návrh je mnohem náročnější jak pro vývojáře (inženýra), který musí navrhnout vhodný evoluční algoritmus a reprezentaci kandidátních řešení, tak pro evoluční algoritmus samotný z důvodu enormního rozsahu vyhledávacího prostoru a jeho vlastností s ohledem na dostupné výpočetní prostředky.

Poznamenejme však, že techniky prezentované v tomto článku neuvažují explicitně principy optimalizace na žádné úrovni činnosti návrhového systému. Smyslem tohto přístupu je „ponechat evoluci volné ruce“ při návrhu potupů pro konstrukci cílových objektů. Řešení získaná z široké škály experimentů jsou následně vyhodnocena s ohledem na jejich kvalitu a nejlepší z nich jsou případně dále analyzována již bez použití evoluce.

3 Biologický a výpočetní development

Development (ontogeneze) je původem biologický proces, během kterého dochází k *vývinu* zárodečné buňky (tzv. *zygoty*) v mnohobuněčný organismus. Základním principem developmentu je stavba organismu a sebeorganizace. Podstatnou částí ontogeneze je proces konstrukce, který je dán interakcemi mezi geny, proteiny, buňkami a prostředím, které buňky obklopuje. Výsledkem tohoto procesu je vyspělý organismus. V průběhu vývinu zygoty můžeme pozorovat tyto základní fáze [16] (poznamenejme, že tyto dílčí procesy neprobíhají zcela odděleně, ale obvykle se překrývají): buněčné dělení, formování tvaru organismu, morfogeneze, buněčná diferenciace, růst.

Principy ontogeneze bývají mnohdy využívány buď samostatně nebo v kombinaci s jinými biologií inspirovanými metodami. V oblasti počítačového inženýrství je development primárně chápán jako mapovací proces z genotypu na fenotyp v evolučním algoritmu. Genotypy představují vhodně zakódovaná kandidátní řešení do chromozomů evolučního algoritmu a fenotypy reprezentují původní instance řešení daného problému, které jsou předmětem vyhodnocení pomocí fitness funkce. V případě aplikace modelu vývinu v kombinaci s evolučním algoritmem obsahuje genotyp *předpis* pro konstrukci cílového řešení (fenotypu). Jelikož je development ve skutečnosti velice komplikovaný proces, bývají jeho modely podstatně zjednodušeny, případně simulují pouze některé jeho části. Z tohoto důvodu mnohdy nelze v případě umělých systémů (jejich částí a procesů v nich probíhajících) inspirovaných těmito principy jednoznačně identifikovat korespondenci s jednotlivými fázemi developmentu v biologii. Jak ukazují dosavadní experimenty, míra shody modelů developmentu s původními biologickými principy závisí z velké části na povaze řešeného problému.

4 Development založený na instrukcích

V roce 1992 představil John R. Koza originální metodu založenou na principech přírodního výběru zvanou genetické programování [10], která představuje v současné době jednu z nejznámějších a nejpoužívanějších evolučních technik pro řešení problémů z mnoha oblastí [11]. Základní myšlenkou genetického programování je využití evoluce k automatické tvorbě počítačových programů. Ačkoliv se princip genetického programování liší od ostatních běžně používaných evolučních technik (zejména z pohledu reprezentace kandidátních řešení a genetických operátorů), klíčová myšlenka automatického programování může být využita prakticky v kterémkoliv evolučním algoritmu. Jak je zřejmé, pro specifický problém je možné zvolit libovolnou vhodnou množinu instrukcí a zavést libovolnou jejich interpretaci a pravidla vykonávání takto vytvořeného programu v závislosti na konkrétních rysech dané aplikace. Na základě tohoto principu je možné specifokovat model umělého vývinu pro potřeby evolučního návrhu, využívající vhodně zvolené instrukce, který nazveme *development založený na instrukcích.*

Pro účely tohoto příspěvku budeme tento přístup chápat jako evoluční návrh *programu* tvořeného aplikačně specifickými instrukcemi, který představuje předpis pro konstrukci vybrané třídy kombinačních obvodů – v tomto případě řadicích sítí a násobiček. Poznamenejme, že pro experimenty, jejichž výsledky jsou prezentovány v kapitolách 5 a 6, byl použit jednoduchý genetický algoritmus [1]. Jelikož fenotypy genetického algoritmu představují kombinační logické obvody, jejich vyhodnocení (výpočet fitness) je prováděno jako součet korektně zpracovaných bitů na výstupu obvodu pro všechny možné jeho binární vstupní (testovací) kombinace. Například, pro kombinační násobičku 4x4 bity, obsahující 4+4=8 vstupů a 8 výstupů, je nejvyšší možná fitness hodnota rovna součinu počtu testovacích vektorů a počtu výstupů obvodu, tj. $2^8 \cdot 8 = 2048$.

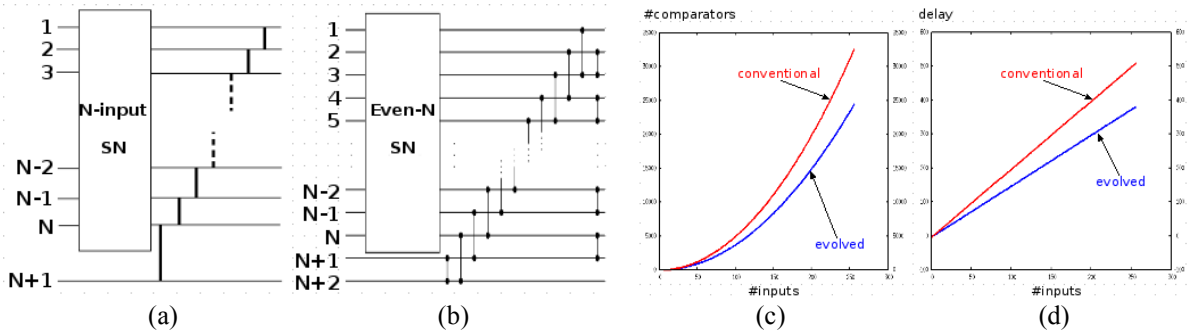
5 Návrh libovolně velkých řadicích sítí

Úspěšná metoda zaožená na aplikaci developmentu, založeného na instrukcích, pro evoluční návrh generických řadicích sítí byla představena v [9]. Řadicí síť je kombinační obvod, obsahující N vstupů a N výstupů, který realizuje uspořádání libovolné vstupní sekvence hodnot do neklesající posloupnosti. Základním stavebním blokem řadicí sítě je tzv. komparátor obsahující dva vstupy a dva výstupy, který porovnává vstupní hodnoty a v případě, že tyto nejsou uspořádány v požadované relaci, provede na výstupech jejich záměnu. Řadicí síť je tvořena konečnou posloupností komparátorů, jejichž vstupy jsou propojeny s primárními vstupy sítě nebo s výstupy jiných komparátorů. Vhodné propojení a uspořádání komparátorů je klíčové pro (1) optimalizaci počtu komparátorů a tedy i ceny řadicí sítě a (2) pro optimalizaci zpoždění sítě, které má zásadní vliv na rychlost řazení.

Systém pro návrh generických řadicích sítí prezentovaný v [9] pracuje s jednoduchými instrukcemi typu COPY, respektive MODIFY, které provádí kopírování určité posloupnosti již

existujících komparátorů řadicí sítě, respektive modifikaci zapojení komparátorů. Na počátku je návrhářem specifikováno tzv. *embryo*, které představuje triviální instanci řadicí sítě. Cílem návrhového procesu je nalezení vhodného programu tvořeného uvedenými instrukcemi, jehož aplikací na embryo vznikce větší funkční řadicí síť, která je použita jako vstup stejného programu pro vytvoření další instance řadicí sítě atd. Řadicí síť může tedy „růst" teoreticky nekonečně, prostřednictvím iterativní aplikace tohoto programu. Obecná konvenční technika konstrukce řadicích sítí je znázorněna na obrázku 1a. Zobecněná metoda nalezená evolučním algoritmem v kombinaci s developmentem založeným na instrukcích je na obrázku 1b. V obou případech probíhá konstrukce řadicích sítí podobným způsobem: větší řadicí síť je tvořena z menší sítě přidáním vhodného uspořádání komparátorů. Proto je adekvátní porovnat výsledné řadicí sítě navržené evolucí právě s touto konvenční technikou. Výsledek srovnání z pohledu počtu komparátorů, respektive zpoždění v závislosti na počtu vstupů řadicími sítí, ukazuje obrázek 1c, respektive 1d. Je zřejmé, že výsledné řadicí sítě navržené evolucí vykazují lepší vlastnosti s konvenčním řešením [9]. Poznamenejme, že algoritmus konstrukce řadicích sítí, zobrazený na obrázku 1b, byl formálně dokázán jako obecný [17].

Pro srovnání uveďme, že zcela odlišná technika pro návrh generických řadicích sítí s využitím developmentu založeného na přepisovacích systémech byla představena v [20]. Ačkoliv se v některých případech podařilo nalézt obecné řešení, které je z pohledu ceny nebo zpoždění sítě výhodnější než konvenční algoritmus, kvalit nejlepšího návrhu zmíněného na obrázku 1 se však dosáhnout nepodařilo.



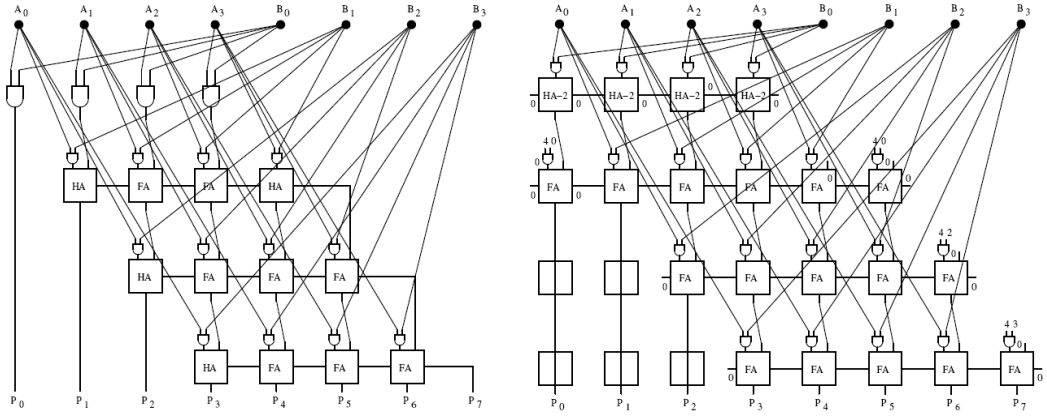
(a) (b) (c) (d)

Obrázek 1: (a) obecná konvenční metoda konstrukce řadicích sítí – straight-insertion sort, (b) zobecnění nejlepšího algoritmu konstrukce řadicích sítí nalezeného evolucí v kombinaci s developmentem založeným na instrukcích, (c) srovnání počtu komparátorů řadicích sítí vytvořených konvenční a evoluční metodou, (d) srovnání zpoždění řadicích sítí vytvořených konvenční a evoluční metodou

6 Návrh generických kombinačních násobiček

Systém pro evoluční návrh generických kombinačních násobiček s využitím developmentu založeného na instrukcích byl představen v [18]. Cílem je evolučním algoritmem nalézt množinu nezávislých programů, které sdílí množinu interních proměnných vývojového systému a parametr specifikující bitovou šířku vyvíjené násobičky. Instrukční soubor obsahuje operace přiřazení proměnné, inkrementace a dekrementace proměnné, jednoduchý cyklus a instrukci pro generování stavebních bloků obvodu. Návrh je inspirován strukturou konvenčních kombinačních násobiček, jejichž některé části se liší od struktury zbylé části obvodu a tvoří tak nepravidelné struktury obvodu (první úroveň AND hradel a druhá úroveň sčítaček s AND hradly, viz obrázek 2a). Tyto odlišné části a zbývající dvě úrovně obvodu je možné vytvořit třemi různými algoritmy parametrizovanými bitovou šířkou operandu (4 bity). Z tohoto důvodu byl do systému zaveden *vliv prostředí* reprezentujícího externí informaci pro řízení vývinu (inspirováno biologickým developmentem, kde prostředí, reprezentující určité vnější podmínky, ovlivňuje jistým způsobem vývin organismu). Prostředí je v tomto případě reprezentováno konečnou posloupností celočíselných hodnot určujících, který program z hledané množiny programů se má v daném okamžiku provést pro vytvoření určité části

násobičky. Obvodu na obrázku 2a odpovídá posloupnost aplikace těchto algoritmů specifikovaná prostředím (0, 1, 2, 2), tj. V prvním kroku je aplikován program 0, který vytvoří první úroveň AND hradel, dále program 1 pro vytvoření kompozice sčítaček a následně dvakrát program 2 za účelem vytvoření zbylé části obvodu, která sama o sobě vykazuje vysoký stupeň regularity.. Stejný tvar prostředí byl použit i pro evoluční návrh, jehož výsledek ukazuje obrázek 2b. Podoba prostředí je určena návrhářem na počátku procesu vývinu. Optimalizované výsledky tohoto systému odpovídají struktuře konvenční násobičky (viz obrázek 2a), lze tedy hovořit o automatickém znovuobjevení tohoto principu. Poznamenejme, že metoda prezentovaná v [18] představuje první případ evolučního návrhu generických násobiček s využitím developmentu. Tento model byl dále rozšířen a zdokonalen pro evoluční návrh generických násobiček s uchováním přenosu, které jsou efektivnější z hlediska zpoždění v porovnání s klasickými kombinačními násobičkami [19].



(a) (b)

Obrázek 2: Kombinační násobička 4×4 bity: (a) konvenční řešení, (b) struktura vyvinutá programy nalezenými evolucí, která je po optimalizaci komponent s nulovými vstupy totožná s konvenční násobičkou

7 Diskuze a obsah dizertační práce

Jak je patrné z výsledků uvedených experimentů, development založený na instrukcích prokázal schopnosti návrhu vybraných tříd reálných generických obvodů. V oblasti řadicích sítí se podařilo objevit algoritmy a obvodové struktury, které byly ještě donedávna neznámé, a které přináší inovaci z pohledu efektivity implementace a funkce řadicích sítí. Vůbec poprvé se podařilo navrhnout obecné algoritmy pro konstrukci násobiček s využitím evolučních technik. Ačkoli v tomto případě nebylo dosaženo žádné inovace v porovnání s konvenční metodou návrhu této třídy obvodů, představují dosažené výsledky významný přínos v oboru evolučního návrhu z důvodu ověření zavedení externího řízení developmentu (prostředí), což může být vhodnou inspirací pro další výzkum nejen pro oblast číslicových obvodů.

Techniky inspirované biologickými principy ontogeneze, implementované v umělých modelech, poskytují užitečné mechanismy pro vyšetřování různých vlastností a chování těchto systémů. Zdá se však, že z praktického hlediska není striktní přejímání principů biologie nejvhodnější pro řešení reálných (v našem případě obvykle technických) problémů. Volba vhodných technik s ohledem na reprezentaci cílových řešení je tak pravděpodobně silně aplikačně specifickou záležitostí. Otázkou však zůstává zejména to, jak navrhnout optimální model pro řešení konkrétního problému a zda jsou současné technologie schopny poskytnout vhodnou platformu pro efektivní realizaci takového systému.

V průběhu uplynulých let byly provedeny sady experimentů ověřujících různé typy modelů developmentu pro oblast evolučního návrhu číslicových obvodů. Jako nejvýhodnější se ukázal být právě development založený na instrukcích, mezi jehož přednosti patří zejména:

1. Výpočetní úplnost. V obecném případě jsou instrukce považovány za prostředky určitého programovacího jazyka (např. Assembler, C apod), jehož cílem je univerzálnost s ohledem na uplatnění v různých oblastech.
2. Snadná přizpůsobitelnost. Množinu instrukcí je možné volit libovolně, jednotlivé instrukce mohou být interpretovány specifickým způsobem s ohledem na charakter řešeného problému, cílovou architekturu, evoluční algoritmus atd.
3. Reprezentace programů. Programy tvořené instrukcemi z daného instrukčního souboru, které jsou předmětem evoluce, je možné reprezentovat jako lineární strukturu, případně jako syntaktický strom, v závislosti na možnostech instrukčního souboru. Tím je zaručena snadná aplikace genetických operátorů, což má za následek zvýšení efektivity evolučního procesu.
4. Škálovatelnost. Experimenty prokázaly možnosti realizace specifických systémů developmentu založeného na instrukcích za účelem návrhu generických řešení různých problémů (škálovatelnost cílových objektů). Problém škálovatelnosti představuje zásadní překážku evolučního návrhu složitých systémů a development byl původně zaveden za účelem jeho překonání. Je však též třeba studovat vlastnosti jednotlivých modelů s ohledem na možnosti řešení problému škálovatelnosti v daných aplikacích a development založený na instrukcích představuje jeden z možných přístupů.
5. Aplikovatelnost. Development založený na instrukcích může být využit v mnoha oblastech, které vyžadují algoritmický přístup k řešení problémů.

S ohledem na tyto aspekty bude další výzkum zaměřen právě na development založený na instrukcích, na detailnější studium jeho vlastností s ohledem na různé aplikace (např. úplnost/redundance instrukčního souboru, minimální délka programu apod.) a na experimenty související s uplatněním tohoto modelu v oblasti návrhu číslicových obvodů. Dizertační práce tedy bude mít následující strukturu. Úvod bude zaměřen na popis základních biologických principů, které se staly inspirací pro výpočetní development v kombinaci s technikami evolučního návrhu. Přehled současného stavu této problematiky bude zahrnovat různé přístupy a vybrané dosavadní experimentální výsledky v oblasti evolučního návrhu využívajícího development. Budou též diskutovány výhody a nevýhody konkrétních vývojových modelů. Hlavní část práce, jejíž náplň je stručně shrnuta v tomto příspěvku, se zabývá specifickým modelem ontogeneze v oboru evolučního návrhu číslicových obvodů, který je nazván development založený na instrukcích. Jeho principy, aplikace a experimentální výsledky představují hlavní přínos autora této práce. V závěru budou diskutovány různé vlastnosti tohoto modelu, zhodnocení dosažených výsledků a možnosti budoucího výzkumu v této oblasti.

8 Závěr

V příspěvku byly shrnuty principy specifického modelu ontogeneze založeného na instrukcích, který se ukázal být jednou z vhodných technik pro evoluční návrh číslicových obvodů. Výzkum byl zaměřen zejména na návrh generických obvodových struktur. Byly demonstrovány aplikace evolučního návrhu libovolně velkých řadicích sítí a kombinačních násobiček a provedeno zhodnocení z hlediska přínosu nových poznatků. Nakonec byla nastíněna podoba budoucí dizertační práce zaměřené na development založený na instrukcích aplikovaný v oblasti evolučního návrhu obvodů.

Ačkoliv výsledky dosažené těmito modely nejsou v mnoha případech zdaleka optimální, představují významný přínos této problematice z důvodu možnosti nalezení nových algoritmů, jejichž vlastnosti a výsledky mohou být inspirací pro další zdokonalování vývojových systémů. Tyto skutečnosti indikují potřebu dalšího výzkumu v oblasti biologií inspirovaných modelů s využitím dosavadních výsledků a poznatků, který bude v našem případě zaměřen zejména na vývoj generických obvodových struktur a studium jejich vlastností s ohledem na spolehlivost, bezpečnost, adaptabilitu a další aktuální aspekty současné doby.

Poděkování

Výzkum byl proveden za podpory Ministerstva školství, mládeže a tělovýchovy České republiky v rámci projektu č. 0021630528 *Výzkum informačních technologií z hlediska bezpečnosti* a Grantové agentury České republiky v rámci projektu č. 102/05/H050 *Integrovaný přístup k výchově studentů DSP v oblasti paralelních a distribuovaných systémů.*

Literatura

[1] P. J. Bentley (ed.): *Evolutionary Design by Computers,* Morgan Kaufmann Publisher, San Francisco, 1999
[2] J. F. Miller, D. Job: *Principles in the evolutionary design of digital circuits – part I*, Genetic Programming and Evolvable Machines, 1(1), 2000, s. 8–35
[3] X. Yao: *Evolving artificial neural networks*, Proceedings of the IEEE, 87(9), 1999, s. 1423–1447
[4] J. R. Koza et al: *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann Publisher, San Francisco, 1999
[5] M. Murakawa et al: *Evolvable Hardware at Function Level*, In: Proc. of the Parallel Problem Solving from Nature IV, LNCS vol. 1141, Springer–Verlag Berlin Heidelberg New York, 1996, s. 62–71
[6] J. Toressen: *A scalable approach to evolvable hardware*, Genetic Programming and Evolvable Machines, 3(3), 2002, s. 259–282
[7] T. G. W. Gordon, P. J. Bentley: *Towards development in evolvable hardware*, In: Proc. of the 2002 NASA/DoD Conference on Evolvable Hardware, IEEE Computer Society Press, 2002, s. 241–250
[8] P. Haddow, G. Tufte: *Bridging the genotype–phenotype mapping for digital FPGAs*, In: Proc. Of the 3rd NASA/DoDWorkshop on Evolvable Hardware, IEEE Computer Society Press, 2001, s. 109–115
[9] L. Sekanina, M. Bidlo: *Evolutionary Design of Arbitrarily Large Sorting Networks Using Development*, Genetic Programming and Evolvable Machines 6(3),2005, s. 319–347
[10] J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992
[11] J. R. Koza, F. H. Bennett III., D. Andre, M. A. Keane: *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann pub., 1999
[12] J. H. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975
[13] I. Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973
[14] H.-P. Schwefel: *Numerical optimazaion of computer models* , Wiley, Chichester, 1981
[15] L. J. Fogel, A. J. Owens, M. J. Walsh: *Artificial Intelligence through Simulated Evolution*, John Wiley, 1966
[16] S. Kumar, P. J. Bentley (eds.): *On Growth, Form and Computers*, Elsevier Academic Press, 2003
[17] M. Bidlo, R. Bidlo, L. Sekanina: *Designing a Novel General Sorting Network Constructor Using Artificial Evolution*, In: TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY, roč. 15, Enformatika, Barcelona, 2006, s. 85–90
[18] M. Bidlo: *Evolutionary Development of Generic Multipliers: Initial Results*, In: Proc. of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems, IEEE Computer Society, Los Alamitos, 2007, s. 405–412
[19] M. Bidlo: *Evolutionary Design of Generic Combinational Multipliers Using Development*, Proc. of the International Conference on Evolvable Systems, ICES 2007, Lecture Notes in Computer Science vol. 4684, Springer-Verlag, Berlin Heidelberg, 2007, s. 77–88
[20] J. Hýsek: *Evoluční návrh s využitím přepisovacích systémů [diplomová práce]*, Ústav počítačových systémů FIT VUT v Brně, 2007