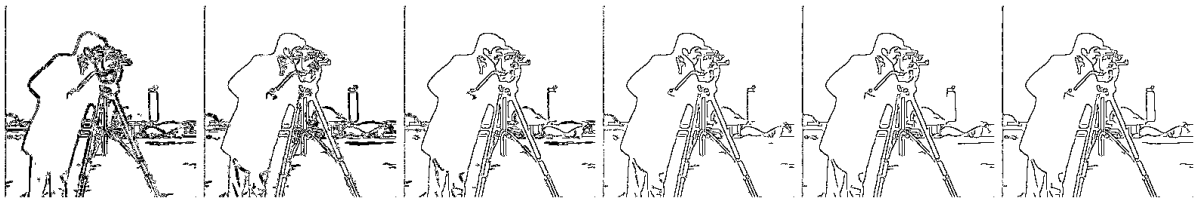
Excel@FIT 2015
http://excel.fit.vutbr.cz

Evoluční návrh pro aproximaci obvodů

Petr Dvořáček*



Abstrakt
V posledních letech klademe stále větší důraz na energetickou úspornost integrovaných obvodů. Můžeme vytvořit aproximační obvody, které nesplňují specifikovanou logickou funkci, a které jsou cíleně navrženy ke snížení plochy, zpoždění a příkonu. Tyto přibližně pracující obvody lze využít v mnoha aplikacích, kde lze tolerovat chyby, obzvláště ve zpracování obrazu. Tato práce popisuje evoluční návrh aproximačních obvodů pomocí kartézského genetického programování (CGP). Díky paralelnímu výpočtu fitness a různým akceleracím byla urychlena evaluace 8-bitové násobičky více než 170 krát oproti standardnímu přístupu. Pomocí inkrementální evoluce byly vytvořeny různé aproximační násobičky a použity v procesu potlačení nemaximálních hodnot v detekci hran.

Klíčová slova: kartézské genetické programování — aproximační počítání — optimalizace

Přiložené materiály: *N/A*

*xdvora0n@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Aproximační počítání se jeví jako nové perspektivní řešení, pomocí kterého lze vytvářet výkonné a energeticky nenáročné systémy za cenu vyšší chybovosti. Existuje mnoho aplikací, ve kterých můžeme tolerovat chyby. Lidské smysly, zejména zrak a sluch, nemusí chyby rozpoznat, což je řadu let využíváno v ztrátové kompresi multimediálních dat. Další toleranci k chybám můžeme pozorovat v oblasti získávání znalostí z databází, kde je prakticky nemožné získat prokazatelně optimální výsledek. Najdeme ji také v aplikacích, kde je člověk očekává, ku příkladu v předpovědi počasí, či v klasifikaci.

Aproximační výpočetní systémy můžeme nalézt ve všech úrovních softwaru i hardwaru. Byly představeny přibližné sekvenční a aritmetické obvody (např. SRAM [1], sčítačky [2], či násobičky [3]) a složitější obvody (např. perceptron, FFT, či kosinová transformace [4]). Byly navrženy procesory s volitelným stupněm aproximace, ba i programovací jazyky podporující aproximační počítání.

V průběhu vývoje vznikly dva hlavní proudy k aproximaci obvodů. První z nich je aproximace založená na porušení správného časování obvodu, například snížením napájecího napětí nebo zvýšením vstupní frekvence, což vede ke vzniku chyb [1]. Další aproximační metodou je funkční aproximace. Jedná se o cílené změny funkční specifikace obvodu za účelem snížení příkonu [4]. Tato práce se zaměřuje právě na funkční aproximaci.

Existuje několik konvenčních metod, které nám dovolí vytvářet aproximační obvody. Jednou ze základních aproximací je přiřazení konstanty nebo vstupu na výstup, či zanedbání nejméně významných bitů. Další možností je transformace plně funkčních aritmetických obvodů na jednodušší. Sčítání lze nahradit za logickou operaci OR, násobení může být zaměněno bitovým posuvem. Těchto poznatků a několika dalších bylo využito v systému ABACUS [5]. Dalším algorit-

mem je SALSA [4]. Jedná se iterativní metodu, která aproximuje obvod a současně garantuje zachování požadované míry kvality.

Evoluční algoritmy vychází z biologické teorie evoluce. Hlavní hnací silou evoluce je přírodní výběr. Nadprůměrně kvalitní jedinci mají více potomků, a proto se jejich genetická informace objevuje v dalších generacích častěji než u méně kvalitních jedinců. Postupem času získáváme jedince, jenž jsou dokonalejší než jedinci na počátku evoluce. Evoluční techniky pracují analogicky a byly úspěšně použity k návrhu obvodů [6].

V posledních letech probíhá výzkum evolučních algoritmů v aproximaci obvodů. V publikaci [2] je použito kartézské genetické programování (CGP) k aproximaci a rovněž je prezentována fitness funkce vhodná k aproximaci aritmetických obvodů. V článku [7] je ukázána vícekriteriální aproximace podle různých chybovostních metrik. V publikaci [3] je představena heuristická inicializace aproximačních obvodů a jejich evoluční doaproximaci.

Dosavadní akcelerace CGP se především zaměřovaly na simulaci obvodu. Byla ukázána možnost paralelní simulace [6], či předkompilace chromozomu [8]. Bohužel žádný článek se nevěnuje rychlosti výpočtu fitness hodnoty pro aproximaci obvodů.

Cílem tohoto článku je představit nový paralelní výpočet fitness funkce, který umožní významně urychlit celý evoluční návrh. Optimalizovaný algoritmus použijeme v návrhu aproximačních 8-bitových násobiček. Následně demonstrujeme použitelnost aproximačních obvodů v Cannyho detektoru hran.

Zbytek textu je členěn následovně. V kapitole 2 jsou podrobněji popsány metody evolučního návrhu aproximačních obvodů. Teorie Cannyho detekce hran se nachází v kapitole 3. Kapitola 4 popisuje navržené metody. Experimentální vyhodnocení je představeno v kapitole 5. Článek uzavírá kapitola 6.

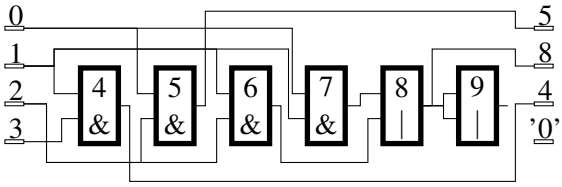
2. Evoluční návrh obvodů

2.1 Kartézské genetické programování

Pomocí evolučního návrhu byly vyvinuty různé inovativní řešení, které mohou být lepší než konvenční řešení navržené člověkem. CGP patří mezi nejpoužívanější algoritmus v evolučním návrhu číslicových obvodů [6], [9].

2.1.1 Reprezentace obvodu v CGP

Kandidátní obvod bývá nejčastěji reprezentován acyklickým orientovaným grafem. Uzly grafu jsou uspořádány do matice o n_c sloupcích a n_r řádcích. Každý uzel představuje určitou operaci g z konečné množiny operací Γ. V evolučním návrhu obvodů uzly většinou reprezentují binární hradla se specifickou Booleovou operací. Parametr l-back l definuje propojitelnost uzlů mezi jednotlivými sloupci grafu. Pro $l = 1$ je propojitelnost minimální, protože se propojují uzly mezi sousedními. Pro $l = n_c$ je propojitelnost maximální, protože se mohou propojovat libovolné sloupce. Zároveň musí být určen počet primárních vstupů obvodu n_i a počet primárních výstupů n_o.



Obrázek 1. Propojení obvodu reprezentovaného CGP s parametry: $n_i = n_o = 4$, $n_c = l = 6$, $n_r = 1$, $\Gamma = \{0^{\&}, 1^{|}\}$. Chromozom: $(1,3,0)(0,2,0)(1,2,0)$ $(0,1,0)(7,6,1)(8,8,1)(5,8,4,'0')$. Převzato z [3].

Zapojení obvodu je zakódováno chromozomem konstantní délky. Každé hradlo (uzel grafu) je tvořeno trojicí (i_1, i_2, α), kde α představuje kód funkce z Γ, a kde i_1 a i_2 jsou indexy libovolného uzlu předchozích sloupců nebo se jedná o zapojení na primární vstup obvodu. Chromozóm dále obsahuje n_o genů, které označují uzly nebo primární vstupy anebo logické konstanty 0 a 1 připojené na primární výstup. Logické konstanty mohou být vhodné při vývoji aproximačních obvodů [3]. Příklad zakódování CGP je uveden na obrázku 1.

2.1.2 Fitness funkce

Výpočet fitness hodnoty definujeme sumou absolutních diferencí (SAD), která je pro aproximační obvody vhodnější než suma Hammingových vzdáleností [2]. Cílem evoluce je minimalizovat chybu:

$$f = \sum_{j=1}^{K} |y(j) - t(j)| \tag{1}$$

kde y je výsledek kandidátního řešení a t je očekávaný výsledek pro danou vstupní kombinaci. Hodnota K odpovídá počtu vstupních kombinací plně funkčního obvodu, tedy $K = 2^{n_i}$. Takže s rostoucím počtem vstupů roste exponenciálně doba evaluace kandidátního řešení.

2.1.3 Prohledávací algoritmus

1. Vytvoří se $(1+\lambda)$ náhodných obvodů.
2. Každý obvod se ohodnotí fitness funkcí f.
3. Obvod s nejnižší fitness hodnotou je označen za rodiče. Pokud existuje více jedinců s nejnižší

fitness hodnotou, vybere se za rodiče ten jedinec, který jim nebyl v předchozí iteraci (generaci).

4. Z rodiče je vygenerováno λ potomků, na které se aplikuje nanejvýš h validních bodových mutací.
5. Kroky 2–3 se opakují tak dlouho, než je vyčerpán počet generací n_g, či nalezen patřičný obvod.

2.2 Optimalizace algoritmu

Kvůli exponenciálnímu růstu výpočtu fitness je třeba se zaměřit na urychlení simulace kandidátního řešení.

Nejpoužívanější akcelerační technika spočívá ve využití 64 bitových registrů a logických operací procesoru [6]. Díky nim můžeme paralelně simulovat 64 testovacích vektorů.

V publikaci [8] byla představena metoda předkompilace chromozomu do binárního strojového kódu. Interpretace chromozomu, který obsahuje instrukce skoku, probíhá pouze jednou a to při jeho kompilaci. Pak se spouští rychlý kód, jenž neobsahuje instrukce skoku. Kromě toho kompilovaný kód nemusí ohodnocovat nepoužitá hradla.

Další technika využívá přeskočení evaluace jedinců [10], u kterých dojde k neutrálním mutacím. Což jsou mutace, které nemají vliv na výslednou fitness hodnotu. K neutrálním mutacím může dojít tehdy, mutujeme-li nepoužitá hradla. Na obrázku 1 se jedná o hradlo s indexem 9. Takže nové kandidátní řešení má stejnou fitness hodnotu jako jeho rodič.

2.3 Použití CGP v aproximaci složitějších obvodů

Evoluční přístup pomocí CGP selhává v návrhu složitějších obvodů, jimiž jsou kupříkladu přibližné mediány o 25-ti vstupech. Ohodnocení velkých obvodů je jednak pomalé, a jednak algoritmus s náhodnou inicializací populace má tendence uváznout v lokálních extrémech. Proto se využívá různých heuristik ke zlepšení škálovatelnosti použití [6].

V článku [3] byla ukázána metoda heuristická inicializace (HI) obvodů. Jedno hradlo z obvodu C_i je nahrazeno propojkou, která spojuje výstup odebraného hradla s jedním z jeho vstupů. Celý postup se zopakuje pro všechna hradla a všechny jejich vstupy. Výsledkem heuristické inicializace je takový obvod C_{i+1}, který má nejmenší chybu. Obvodem C_{i+1} je pak inicializována počáteční populace CGP.

Z plně funkčního obvodu C_0 o n_n hradlech postupně vyvineme pomocí HI obvody obsahující $n_n - 1$, $n_n - 2, \ldots 2, 1$ hradel. Z několika běhů CGP získáme nejlépe aproximovaný obvod o k hradlech. Na tento obvod pak aplikujeme HI, abychom získali menší obvod obsahující $k - 1$ hradel.

3. Cannyho detektor hran

Detekce hran patří mezi nejčastější operace počítačového vidění. Cannyho hranový detektor je vícekrokový algoritmus, který byl realizován, aby co nejlépe vyhověl jednoznačnosti, kvalitě a přesnosti detekce [11]. Algoritmus se pak skládá z těchto částí:

1. Eliminace šumu
2. Výpočet velikosti a směru gradientu
3. Nalezení lokálních maxim
4. Dvojité prahování s hysterezí

K odstranění šumu se používá konvoluce s Gaussovým filtrem typicky o velikosti 5x5 a parametrem σ, který určuje sílu rozmazání. Výpočet velikosti a směru gradientu je založen na detekci hran podle první derivace, kterými jsou například Sobelovy operátory (S_H, S_V). Velikost a směr gradientu lze získat dle vztahů 2 a 3.

$$|G| = |S_H| + |S_V| \tag{2}$$

$$\Theta = \tan^{-1}(S_H / S_V) \tag{3}$$

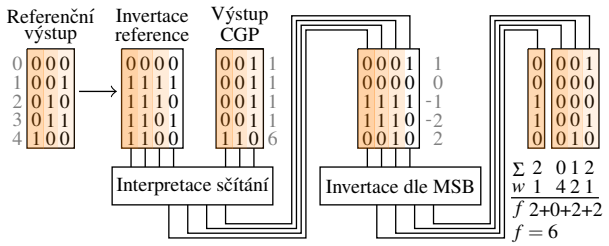
Základem nalezení lokálních maxim je rozdělení hran do čtyř skupin podle jejich směru a to na hrany pod přibližným úhlem 0°, 45°, 90° a 135°. Cannyho hranový detektor využívá dva prahy T_l a T_h. Prahování s hysterezí se rozumí, že silné hrany ($> T_h$) jsou ponechány. Slabé hrany jsou také ponechány, pokud jsou napojeny s nějakou silnější hranou. Velmi slabé hrany ($< T_l$) jsou potlačeny a nevyskytují se ve výsledném řešení.

4. Navržené metody pro evoluční aproximaci

V podkapitole 4.1 si nejdříve ukážeme možnost akcelerace výpočtu fitness hodnoty. Akcelerované CGP pak použijeme v algoritmu k vývoji aproximačních 8-bitových násobiček. Tyto aproximované násobičky využijeme v Cannyho hranovém detektoru a jejich konkrétní použití je popsáno v podkapitole 4.2.

4.1 Paralelní výpočet fitness funkce

Odezvou paralelní simulace obvodu (viz podkapitola 2.2) je n_o 64-bitových vektorů. Tyto vektory jsou seřazeny od nejvíce významných bitů (MSB) po nejméně významné (LSB). Fitness funkce vhodná pro aproximační počítání je definovaná vzorcem 1. Před výpočtem musíme provést tzv. transpozici bitů. Tím je myšlen převod n_o 64-bitových vektorů na 64 celočíselných hodnot, na které lze aplikovat vzorec 1. Princip transpozice vektorů byl použit v publikacích [2] a [3].



Obrázek 2. Paralelní výpočet fitness funkce SAD na 5-ti bitových vektorech, kde Σ značí *popcnt* a w je váha vektoru. Šedou barvou označeny celočíselné hodnoty.

Problematickou částí dosavadního přístupu je právě v procesu transpozice vektorů, jelikož je potřeba mnoha bitových posuvů, aritmeticko-logických operací a skokových instrukcí, které zabraňují jejich zřetězenému zpracování v CPU. Akcelerace lze dosáhnout interpretací logických obvodů na vektorové úrovni registrů.

Odčítání realizujeme následovně. Před zahájením evoluce CGP invertujeme referenční výstup specifikace obvodu C_r. Invertovaný referenční výstup C_r^- se v průběhu simulace sečte s kandidátním výstupem C_o. Operace sčítání je implementována, jako interpretace sčítačky s postupným přenosem na bitové úrovni. Nejdříve se vypočte hodnota výstupu s_0 a příznak přenosu c_0 podle Booleových vzorců 4 a 5, které znázorňují poloviční binární sčítačku. Potom se postupně získají jejich j-té hodnoty, jak je uvedeno ve vztazích 6 a 7. Operace $+$ značí logický součet, $\cdot$ je logický součin a $\oplus$ znamená logický exkluzivní součet.

$$s_0 = C_r^-(0) \oplus C_o(0) \tag{4}$$
$$c_0 = C_r^-(0) \cdot C_o(0) \tag{5}$$
$$s_j = C_r^-(j) \oplus C_o(j) \oplus c_{j-1} \tag{6}$$
$$c_j = C_r^-(j) \cdot C_o(j) + (C_r^-(j) \oplus C_o(j)) c_{j-1} \tag{7}$$

Problémové je také získání absolutní hodnoty. Po odečtení jsou některé hodnoty v bitových vektorech kladné a některé záporné. Na nejvíce významném vektoru s_{n_o} je uložen příznak záporného čísla, podle kterého se invertuje zbytek vektorů $s_j = s_j \oplus s_{n_o}$. V jednotlivých vektorech j se určí počet jedniček *popcnt*, který se následně vynásobí váhou vektoru 2^j. Získání absolutní hodnoty můžeme zapsat rovnicí 8. Celý postup paralelní simulace je znázorněn na obrázku 2.

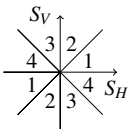
$$fit = popcnt(s_{n_o}) + \sum_{j=0}^{n_o-1} 2^j * popcnt(s_{n_o} \oplus s_j) \tag{8}$$

Předchozí práce [8] nezahrnuje JIT kompilaci (zkr. JIT z angl. Just-in-time compilation) fitness funkce, která je v případě SAD náročná. Předkompilaci fitness funkce můžeme implementovat za pomocí referenční instrukční sady Intelu x64 [12] spolu s reverzním inženýrstvím již zkompilovaných úseků kódu. Aplikováním předkompilace fitness funkce eliminujeme skokové instrukce. Další akcelerace můžeme dosáhnout, využijeme-li zásobník procesoru na mezivýpočty sčítanců s_j.

4.2 Aproximační násobení v Cannyho detektoru hran

Nejdříve se musíme zamyslet, ve které části algoritmu použijeme aproximační násobičky. Byla představena hardwarová akcelerace Gaussova filtru nevyžadující násobení [13]. Konvoluci Sobelových operátorů lze řešit sčítáním, odčítáním a bitovým posuvem. V hysterezi se nic nenásobí. Vylučovací metodou nám zůstala detekce lokálních maxim, jinými slovy potlačení nemaxim.

Operace tan^{-1} se v hardwaru implementuje dosti náročně. Proto vypočteme úhly podle vertikálních a horizontálních složek Sobelových filtrů, jak je znázorněno na grafu 3.



Obrázek 3. Získání čtyř složek přibližných orientací úhlů podle velikosti Sobelových složek.

Proces ztenčení je založen na získání velikosti gradientu (magnitudy) sousedních pixelů, které se nachází na normále od daného směru hrany. Pokud jsou sousední magnitudy menší než magnituda daného pixelu, potom se jedná o lokální maximum. Výpočet normály můžeme realizovat váhováním. První část detekce lokálního extrému pro první směr je znázorněna vztahem 9, kde horní indexy odpovídají souřadnicím obrázku. Úpravou této nerovnice získáme vztah 10 s násobením, které budeme aproximovat. Před použitím aproximačních obvodů je nutné normalizovat magnitudy a Sobelovy složky na hodnoty v rozmezí od $0-255$, protože budeme vyvíjet násobičky přijímající dva osmibitové vektory.

$$|G^{x-1,y-1}|\frac{S_V^{x,y}}{S_H^{x,y}} + |G^{x,y-1}|(1-\frac{S_V^{x,y}}{S_H^{x,y}}) \leq |G^{x,y}| \tag{9}$$
$$|G^{x-1,y-1}|S_V^{x,y} - |G^{x,y+1}|(S_V^{x,y} - S_H^{x,y}) \leq |G^{x,y}|S_H^{x,y} \tag{10}$$

5. Experimentální vyhodnocení

5.1 Akcelerace

Cílem této části práce je urychlit algoritmus CGP. Budeme zkoumat tři možné implementace fitness funkce. První akcelerační metodou (A1) je výpočet SAD pomocí transpozice. Druhá z nich (A2) využije paralelního výpočtu fitness funkce SAD. Třetí urychlovací technika (A3) ohodnocuje obvod pomocí předkompilovaného paralelního výpočtu SAD. Akcelerační techniky porovnáme podle počtu evaluací za sekundu, viz vztah 11, kde *time* odpovídá času jednoho běhu CGP.

$$eps = \frac{n_g * \lambda}{time} \tag{11}$$

Pro každou akcelerační metodu algoritmus CGP využíval následující optimalizace: neohodnocování nepoužitých hradel, přeskakování neutrálních mutací a předkompilovanou 64-bitovou paralelní simulaci obvodu. Pro každou násobičku a každou akcelerační metodu bylo provedeno celkem 10 nezávislých běhů, podle kterých byla spočtena průměrná hodnota *eps*. Nastavení parametrů CGP bylo zvoleno pro všechny běhy následovně: $\Gamma = \{$BUF, INV, AND, OR, XOR, NAND, NOR, XNOR$\}$, $n_r = 1$, $n_c = 40$, $l = 40$, $\lambda = 4$, $h = 2$. Ohodnocovaly se násobičky 4x4, 5x5, 6x6, 7x7 a 8x8 se specifickým počtem generací n_g. Tento experiment probíhal na 2.3 GHz procesoru Intel Core i3-2350M.

Tabulka 1. Porovnání akceleračních technik.

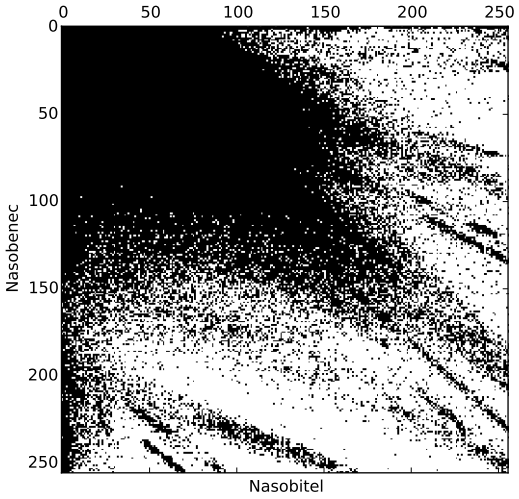
Násobička	$n_g * 10^6$	A1 [*eps*]	A2 [*eps*]	A3 [*eps*]
4x4	10	54 912	**277 923**	226 222
5x5	5	12 052	168 688	**186 781**
6x6	1	2 683	61 673	**129 541**
7x7	0.1	602	17 644	**53 930**
8x8	0.01	135	4 782	**23 190**

Výsledky experimentu jsou zobrazeny v tabulce 1. Můžeme vidět, že pro násobičky 4x4 bity je metoda A3 neefektivní. Pro větší násobičky (5x5 a víc) je však vhodnější použít kompilovanou fitness funkci (A3). Důvod nalezneme především v eliminaci skokových instrukcí při kompilaci. Návrh 8-bitové násobičky byl urychlen **170** krát oproti původní metodě (A1).

5.2 Vývoj aproximačních násobiček pro hranový detektor

V prvním scénáři (S1) se budou aproximovat 8-bitové násobičky podle úplné specifikace. Tedy definujeme všech 2^{16} možných vstup-výstupních kombinací. Ve druhém scénáři (S2) se budou aproximovat 8-bitové násobičky podle částečné specifikace. Takže definujeme pouze některé vstup-výstupní kombinace násobičky.

Je zřejmé, že násobička v Cannyho hranovém detektoru nevyužívá všechny možné kombinace operandů, ale pouze část, jak je znázorněno na obrázku 4. Hodnoty s černou barvou značí, že došlo k násobení daných prvků. Částečnou specifikaci definujeme právě podle nich. Budeme sledovat relativní chybovost násobičky, počet chyb v detekovaném obrázku a rychlost evaluace daného obvodu.

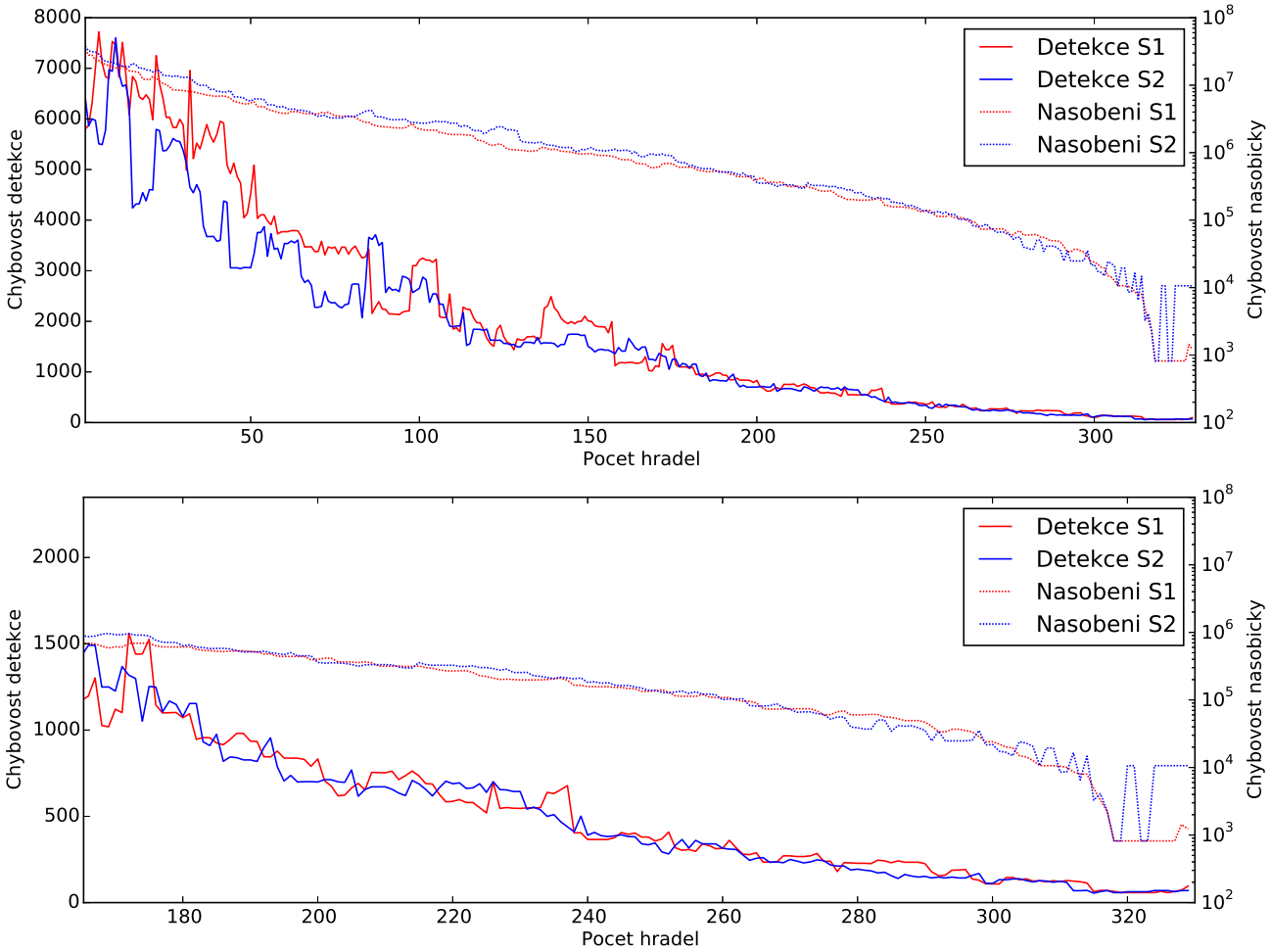


Obrázek 4. Využití operandů násobení v Cannyho detekci hran. Hodnoty brány z několika obrázků.

Pro oba aproximační scénáře byl využit algoritmus prezentovaný v podkapitole 2.3 spolu s akcelerovanou verzí výpočtu fitness funkce. Inicializace algoritmu 2.3 proběhla za pomocí plně funkční násobičky, která byla zkonstruována pomocí Wallaceova stromu [14]. Počet hradel plně funkční násobičky činil $n_n = 330$. Tato hodnota se po 50-ti běžích CGP dekrementovala. Nastavení CGP bylo zvoleno následovně: $\Gamma = \{$BUF, INV, AND, OR, XOR, NAND, NOR, XNOR$\}$, $n_r = 1$, $n_c = n_n + 3$, $l = n_c$, $\lambda = 4$, $h = 0.05 n_c$, $n_g = 500\,000$. Pro dvou týdenní běh byl využit školní server edesign.

Pomocí nalezených aproximačních násobiček byly následně detekovány hrany, dle kapitoly 4.2. Pro detekci byl vybrán standardní obrázek kameramana. Parametry detekce byly zadány následující $\sigma = 1.5$, $T_l = 15$ a $T_h = 30$.

Výsledky detekce pomocí přibližných násobiček o 255 hradlech si můžeme prohlédnout na obrázku 5 dole. Metoda S1 oproti plně funkčnímu řešení nedetekovala hrany komínu. Obě metody nedetekovaly kousek krajiny pod komínem. Další detektory využívající přibližné násobičky jsou prezentovány na úvodní straně, kde počet hradel činil 1, 66, 132, 198, 264 a 330 hradel. Vlastnosti těchto násobiček jsou ukázány v tabulce 2.



Obrázek 6. Porovnání chybovosti aproximačních násobiček a použití v detekci hran. Chybovost násobičky byla počítána dle vztahu 1. Chybovost detekce byla stanovena dle chybně určených pixelů vůči plně funkčnímu řešení.

Obrázek 5. Ztenčení hran pomocí aproximačních násobiček. Nahoře vlevo je plně funkční řešení, nahoře vpravo je řešení využívající násobičku o pěti hradlech. Dole jsou řešení využívající násobičky o 255 hradlech vytvořené metodami S1 (zleva) a S2.

Počet hradel	Chyba SAD [10^3]	Chyba detekce	Relativní plocha	Logical effort
1	338 520	5 799	40	1.76
66	37 901	3 470	3 472	36.11
132	10 508	1 696	6 760	39.39
198	4 405	830	10 096	48.64
264	955	278	13 824	52.56
330	0	0	16 752	56.66

Tabulka 2. Porovnání vlastností aproximovaných násobiček dle metody S1. Chyba detekce byla brána dle chybně určených pixelů vůči plně funkčnímu řešení. Relativní plocha obvodu s relativním zpožděním (Logical effort) byla určena podle standardu *VSCLIB* [15].

Při podrobnější analýze zjistíme, že obě metody se chovají různě, jak je zobrazeno na obrázku 6. Například detekce hran pomocí přibližných násobiček, kde počet hradel $n_n = 217 \ldots 230$, dopadla pro metodu S2 hůře. Avšak detekce hran, kde $n_n = 15 \ldots 85$, dopadla pro metodu S2 podstatně lépe, i když měla hůře

aproximované násobičky.

Naprostá výhoda metody S2 je v rychlosti ohodnocení kandidátního řešení, jelikož se neohodnocují všechny případy násobení (kterých je 2^{16}), ale pouze část ($31594 \simeq 2^{15}$ případů fitness). Urychlení oproti metodě S1 je tedy dvojnásobné.

6. Závěr

Tato práce se zabývala evoluční aproximací obvodů pomocí CGP. Díky paralelnímu výpočtu fitness funkce byla doba evaluace aproximované 8-bitové násobičky urychlena 170 krát. Byla představena možnost aproximace podle částečné specifikace, která umožňuje aproximovat obvody ještě rychleji – záleží na počtu odebraných případů ohodnocení.

Použitelnost přibližných obvodů byla demonstrována v potlačení nemaximálních hodnot v Cannyho detektoru hran. Výsledné aproximované obvody zabíraly méně plochy a byly rychlejší.

V budoucí práci se můžeme zaměřit na evoluční aproximaci složitějších obvodů nebo na hledání dalších aplikací, které tolerují chyby.

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce prof. Lukáši Sekaninovi za vedení a odbornou pomoc při řešení této práce. Dále bych chtěl poděkovat doktoru Zdeňku Vašíčkovi za tipy ohledně akcelerace CGP.

Literatura

[1] Yunus Emre and Chaitali Chakrabarti. Quality-aware techniques for reducing power of jpeg codecs. *J. Signal Process. Syst.*, 69(3), 2012.

[2] Lukáš Sekanina and Zdeněk Vašíček. Approximate circuits by means of evolvable hardware. In *2013 IEEE International Conference on Evolvable Systems (ICES)*, pages 21–28. IEEE, 2013.

[3] Zdeněk Vašíček and Lukáš Sekanina. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation*, pages 1–13, 2014.

[4] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan. Salsa: Systematic logic synthesis of approximate circuits. IEEE, 2012.

[5] K. Nepal, Yueting Li, R.I. Bahar, and S. Reda. Abacus: A technique for automated behavioral synthesis of approximate computing circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*. IEEE, 2014.

[6] Lukáš Sekanina a kolektiv. *Evoluční hardware: Od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Academia, 2009.

[7] Zdeněk Vašíček and Lukáš Sekanina. Evolutionary design of approximate multipliers under different error metrics. In *17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 135–140. IEEE, 2014.

[8] Karel Slaný and Zdeněk Vašíček. Efficient phenotype evaluation in cartesian genetic programming. In *Proc. of the 15th European Conference on Genetic Programming*. Springer, 2012.

[9] Julian F. Miller. *Cartesian Genetic Programming*. Natural Computing Series. Springer Berlin Heidelberg, 2011.

[10] Brian W. Goldman and William F. Punch. Reducing wasted evaluations in cartesian genetic programming. In *Genetic Programming*, pages 61–72. Springer, 2013.

[11] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1986.

[12] Intel. *Intel®64 and IA-32 Architectures Software Developer's Manual*, 2014.

[13] Pei-Yung Hsiao a kolektiv. A parameterizable digital-approximated 2d gaussian smoothing filter for edge detection in noisy image. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006.

[14] C.S. Wallace. A suggestion for a fast multiplier. *Electronic Computers, IEEE Transactions on*, 1964.

[15] Vsclib standard cell library, 2013. http://www.vlsitechnology.org/html/cells/vsclib013/.