EVOLUTIONARY DESIGN METHOD OF MULTIPLIERS USING DEVELOPMENT

Tomáš Kaplan
Master Degree Programme (2), FIT BUT
E-mail: xkapla02@stud.fit.vutbr.cz

Supervised by: Michal Bidlo
E-mail: bildom@fit.vutbr.cz

ABSTRACT

This work is focused on the techniques for overcoming the problem of scale in the evolutionary design of the combinational multipliers. The approaches to the evolutionary design that work directly with the target solutions are not suitable for the design of the large-scale structures. An approach based on the biological principles of development has often been utilized as a non-trivial genotype-phenotype mapping in the evolutionary algorithms that allows us to design scalable structures. The instruction-based developmental approach has been applied to the evolutionary design of the infinitely growing circuit structures.

1. ÚVOD

Tato práce se zabývá problematikou návrhu složitých kombinačních obvodů, kterými jsou kombinační násobičky, pomocí evolučních technik. Vzhledem k velmi častému a četnému používání operace násobení v nejrůznějších algoritmech je snahou nalézt takové zapojení obvodu, které bude ekonomicky a výkonnostně výhodné. V současnosti existuje nepřeberné množství algoritmů pro návrh takového obvodu, ovšem pomocí genetických algoritmů jsme schopni nalézt též řešení nekonvenční, případně kvalitnější. Z důvodu velké složitosti řešeného problému je namísto klasického genetického algoritmu použito lineární genetické programování a některé další evoluční techniky. Návrh nebude probíhat na úrovni jednotlivých hradel, ale na úrovni funkčních bloků, díky čemuž jsme schopni v určitých případech nalézt výrazně větší obvody.

2. KOMBINAČNÍ NÁSOBIČKA

V práci je představena základní varianta násobení bezznaménkových celých čísel. Metoda je založena na principu násobení na mřížce (lattice multiplication) [1]. Stejného principu využívají také násobičky pracující s Wallaceovým stromem [2], které jsou velmi často používány. Algoritmus lze rozdělit do třech hlavních částí. Nejdříve je nutné vytvořit vzájemné součiny jednotlivých cifer obou operandů. Následně provést sečtení vhodných částečných součinů do tzv. diagonálních součtů, a poté provést výsledné sečtení dříve vytvořených mezisoučtů a přenosů do vyšších řádů. Různými realizacemi jednotlivých částí jsme schopni docílit velmi odlišných vlastností celého obvodu.

3. EVOLUČNÍ ALGORITMY

Evoluční algoritmy jsou založeny na principech Darwinovy evoluční teorie [3]. Jedná se o stochastické algoritmy, které prohledávají stavový prostor řešeného problému a na základě daných kritérií hledají nejlepší možné řešení. Velmi často se tyto algoritmy používají na optimalizaci stávajících řešení a na evoluční návrh nových řešení.

Základním evolučním algoritmem je genetický algoritmus (GA) [3]. Algoritmus je založen na schopnosti boje o přežití mezi jedinci populace. Populace představuje množinu kandidátních řešení problému (chromozomů) a pomocí operátorů křížení a mutace se tato populace vyvíjí. Pro ohodnocení kvality jedince (řešení) slouží tzv. fitness funkce.

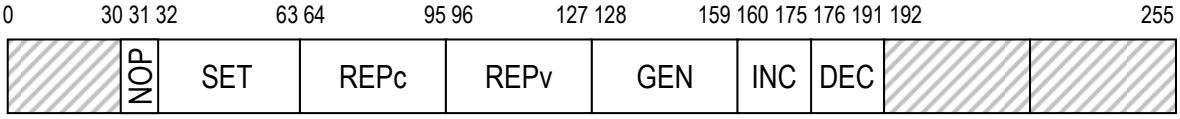
Variantou GA použitého v této práci je genetické programování (GP) [3]. V GP se nesnažíme hledat kandidátní řešení zakódované přímo do chromozomu, ale snažíme se najít program, kterým lze toto řešení popsat. Hledaný program je většinou zapsán ve specifickém jazyce, který je navržen přímo pro řešení daného problému.

Při řešení složitých problémů je zapotřebí využít některé pomocné techniky. Jednou z těchto technik je například development. Hlavním cílem developmentu je zajistit netriviální mapování genotypu na fenotyp (v tomto případě programu na výsledný obvod). V práci je použit development založený na instrukcích [4] a také vývoj za pomocí embrya [4]. Za embryo je zde považováno nějaké počáteční řešení problému případně i neúplné řešení, které se postupně vyvíjí.

4. NAVRŽENÉ METODY

V následujících kapitolách budou představeny tři modely řešení. Modely jsou založeny na myšlence vygenerovat všechny částečné součiny najednou a vhodně je rozmístit do sloupců (diagonál). Dále pracovat na úrovni sloupců a hledat takové zapojení funkčních bloků, které povede ke správnému výslednému součtu. Výsledný součet diagonálních mezisoučtů je realizován pomocí sčítačky s postupným přenosem (ripple-carry adder), která je sice pomalým, ale levným řešením. Na rozdíl od přístupu prezentovaného v [4] neprobíhá adresování každého bloku na přesnou pozici, ale pracuje se zde se zásobníky, ze kterých lze vhodným průchodem sestavit finální obvod.

Pro možnosti developmentu založeného na instrukcích je vytvořena jednoduchá instrukční sada, která je uvedena na obrázku 1. Jednotlivé instrukce pracují s operandy, které mohou být vybírány z maximálně čtyř proměnných a z maximálně čtyř konstant, které slouží jako vstupní znalost algoritmu. Jako konstanta může být například vložena informace o počtu bitů operandu. Zakódování každé instrukce je provedeno na osmi bitech a obsahuje operační kód instrukce (3bity), rozšíření operačního kódu (1bit) a jeden případně dva operandy (2bity).



Obrázek 1: Instrukční sada zakódovaná na 8bitech.

Jednotlivé instrukce provádějí následující operace. NOP – prázdná instrukce. SET – nastaví hodnotu proměnné na hodnotu jiné proměnné nebo konstanty. REP – opakuje n-krát počet m následujících instrukcí. Hodnotu n a m lze nastavit pomocí konstanty nebo proměnné.

GEN – generuje jeden z osmi funkčních bloků na pozici danou hodnotou proměnné. INC – inkrementuje proměnnou o jedničku. DEC – dekrementuje proměnnou o jedničku případně ponechá nulu.

Pro ohodnocování obvodů vytvořených evolucí je možné použít dva základní přístupy. Jedním přístupem je ohodnotit obvody na základě struktury obvodu. Struktura obvodu lze odvodit ze znalosti základních bloků. Tento přístup je velice rychlý, ovšem není schopen najít obvody příliš odlišné od konvenčních řešení. Jiným přístupem je ohodnotit obvody na základě jejich funkčnosti. Vyzkoušení všech možných kombinací vstupních vektorů je pro větší operandy výpočetně náročné, oproti tomu je možné najít zajímavější obvody.

4.1. NÁSOBIČKA ZE ZÁKLADNÍCH BLOKŮ

Základními bloky pro tvorbu násobičky jsou zde poloviční jednobitová sčítačka (HA) a úplná jednobitová sčítačka (FA).

4.2. NÁSOBIČKA Z NETRADIČNÍCH BLOKŮ

Násobička tvořená z netradičních bloků je schopna vyřešit do jisté míry problém škálovatelnosti, protože namísto sestavování obvodů ze základních jednobitových bloků lze použít bloky složitější. Tyto bloky například mohou pracovat na více bitech. Dále je velice žádoucí nalézt násobičku vytvořenou z bloků, které jsou specifické pro určité typy architektur (například FPGA).

4.3. NÁSOBIČKA Z EMBRYA

Poslední metoda řeší úplně problém škálovatelnosti. Cyklický vývoj vycházející z embrya, které je tvořeno násobičkou 2x2 bity, je schopen každou další iterací vytvořit plně funkční složitější obvod. Takto nalezený obvod se může stát v dalším kroku embryem.

5. ZÁVĚR

Představené metody byly ověřeny na příkladech vytvořených ručně. Dále bylo prozatím docíleno znovunalezení násobičky 5x5bitů vytvořené pomocí základních bloků. Nyní je zapotřebí provést minimálně znovunalezení řešení pomocí dalších dvou metod. Dále je také nutné provést velké množství experimentů a porovnat nalezené řešení s řešeními konvenčními. Výzkum byl proveden v rámci grantu FIT-10-S-1 a výzkumného záměru MSM0021630528.

LITERATURA

[1] Broadbent, F.W.: Lattice Multiplication and Division, 1987, Arithmetic Teacher, vol. 34, no. 5, s. 28-31

[2] Wallace, C.S.: A suggestion for a fast multiplier, IEEE Trans. Electronic Computers, 1964, vol. EC-13

[3] Kvasnička, V., Pospíchal, J., Tiňo, P.: Evoluční algoritmy. Vydavatelství STU Bratislava, 2000

[4] Bidlo, M.: Evolutionary Design of Generic Structures Using Instruction-Based Development, PhD thesis, Department of Computer Systems FIT BUT, CZ, 2008