ACCELERATION OF TRANSISTOR-LEVEL EVOLUTIONARY DESIGN OF DIGITAL CIRCUITS USING ZYNQ

Vojtěch Mrázek
Master Degree Programme (2), FIT BUT
E-mail: xmraze06@stud.fit.vutbr.cz

Supervised by: Zdeněk Vašíček
E-mail: vasicek@fit.vutbr.cz

Abstract: The objective of this work is to develop a new method for evolutionary design of digital circuits at transistor-level suitable for HW acceleration. As a target platform, a system on chip Xilinx Zynq consisting of ARM and programmable logic is utilized. The proposed accelerator is more than 4x faster than CPU-based implementation with discrete simulation and more than 1000x faster than SPICE-based simulator.

Keywords: Cartesian genetic programming, transistor, Zynq, acceleration

1 ÚVOD

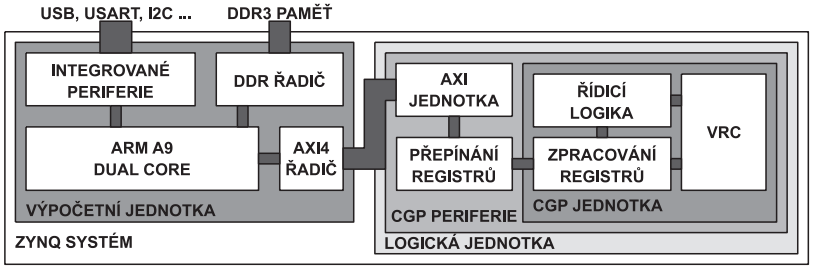
Návrh digitálních obvodů je velmi obsáhlá oblast, ke které můžeme přistupovat z více úrovní. Typicky se návrh provádí na úrovni hradel. Ukazuje se však, že nižší úroveň abstrakce může přinést úsporu v podobě nižšího příkonu nebo menší plochy na čipu. Příkladem může být obvod AND-NOR, který realizuje čtyřvstupou funkci $\overline{A \cdot B + C \cdot D}$. Pokud jej realizujeme na hradlové úrovni a nahradíme jednotlivá hradla v praxi běžně používanou ekvivalentní implementací na úrovni tranzistorů, bude nutné využít 16 tranzistorů. Ukazuje se však, že tutéž funkci jsme schopni na úrovni tranzistorů realizovat pouze za pomoci 8 tranzistorů.

V literatuře můžeme nalézt mnoho prací využívajících evolučních technik zabývajících se návrhem číslicových obvodů na úrovni hradel s využitím kartézského genetického programování [1]. Evolucí obvodů na úrovni tranzistorů se však mnoho prací nezabývá. Jedním z problémů jsou poměrně vysoké nároky na výpočetní výkon, neboť ve snaze garantovat funkčnost nalezených obvodů jsou autoři nuceni využít přesný simulátor typu SPICE [2]. Někteří autoři ve snaze urychlit evoluční návrh využívají velmi zjednodušený model tranzistoru a diskrétní simulaci, avšak evolucí navržená řešení se ukazují býti v praxi nefunkční [3]. Cílem této práce tedy je navrhnout funkční metodu, která bude využívat diskrétní simulaci obvodů v reprezentaci odvozené z kartézského genetického programování a kterou bude možné akcelerovat v HW.

2 NAVRŽENÁ PLATFORMA

Aplikace byla implementována na vývojové desce Xilinx ZC702 s využitím SoC Zynq XC7Z020. Blokové schéma je znázorněno na obrázku 1. Systém Zynq se skládá ze dvou částí. První část představuje tzv. výpočetní jednotka, která obsahuje dvoujádrový procesor ARM Cortex A9 a základní periferie nutné k běhu. K této jednotce je připojena externí 1 GB DDR3 paměť sloužící jako operační paměť programů. Výpočetní jednotka je propojena přes vysokorychlostní AXI4 rozhraní s druhou částí – logickou jednotkou, ve které je implementována hardwarová akcelerační jednotka umožňující simulaci kandidátního řešení na úrovni tranzistorů a komunikuje pomocí sady 32b registrů.

Logická jednotka je složena z následujících částí. Vzhledem k omezenému počtu registrů a sekvenčnímu přístupu po 8 bitech k zápisu jednotka **přepínání registrů** vytváří ovládací strukturu. Obsahuje



Obrázek 1: Schéma vývojové platformy.

registry pro potvrzení zápisu, resetu celé periferie a podle obsahu řídicího registru chromozomu plní tento konfigurační řetězec. Konfigurační řetězec a ovládací bity jsou vstupem do jednotky **CGP**, která má na starost simulaci. K tomu je využit virtuální rekonfigurovatelný obvod **VRC**. Simulace je řízena pomocí stavového automatu. Simulaci předchází detekce aktivních prvků, která má za úkol vyhodnocováním elementů, které nemají vliv na výstup. Po detekci následuje 2^n vyhodnocení vstupního vektoru, kde n je počet vstupů požadovaného obvodu.

Virtuální obvod **VRC** obsahuje výpočetní elementy uspořádané do mřížky. Propojení jednotlivých elementů ve VRC je realizováno následovně: jelikož data mezi elementy jdou nejenom směrem k následujícímu sloupci, ale přenos je potřeba i v opačném směru, jsou sloupce propojeny jak od výstupu k vstupu, tak v opačném směru pomocí zpětné vazby. Mohou být totiž použity propojky nebo tranzistor může pracovat v inverzním režimu, kdy proud prochází od *drain* k *source*. **Element VRC** pracuje ve dvou režimech, v režimu detekce aktivních prvků a v režimu výpočtu. V režimu výpočtu může realizovat jednu z těchto funkcí: pMOS, nMOS, buffer (propojka) a spojení dvou signálů do jednoho.

3 VÝSLEDKY

Navržený systém byl implementován ve VHDL a syntetizován pomocí nástroje Vivado. Evoluční algoritmus pro ARM byl popsán v jazyce C a kompilován pomocí GCC. Důležitým parametrem indikujícím účinnost akcelerace je rychlost ohodnocení jednoho kandidátního řešení (tabulka 1). Ukazuje se, že v závislosti na velikosti VRC může logická jednotka pracovat na frekvenci 40 – 50 MHz podle velikosti VRC. Výpočetní jednotka je taktována na 667 MHz. Počet ohodnocených řešení v čase

Verze	**2 vstupy**		**3 vstupy**		**4 vstupy**		**5 vstupů**	
	t	**R**	**t**	**R**	**t**	**R**	**t**	**R**
NGspice – Xeon	49 000	-	60 000	-	100 000	-	237 000	-
CPU – Xeon	34	1.00	73	1.00	180	1.00	253	1.00
Zynq (ARM)	130	0.26	320	0.23	762	0.27	950	0.27
Zynq (ARM+FPGA)	41	0.83	44	1.65	49	3.67	54	4.68
Zynq (FPGA)	2	17.00	3	24.34	6	30.00	11	23.00

Tabulka 1: Průměrná doba evaluace jednoho kandidátního řešení t v μs a zrychlení R

však není konstantní, neboť počet taktů nutných k ohodnocení kandidátního řešení je ovlivňován přítomností zkratu, množstvím propojek apod. Pro vyhodnocení výkonnosti byl proto zvolen problém evolučního návrhu obvodu NAND, velikost VRC 10x8 a evoluce byla omezena na 500 000 generací. Nejprve byla otestována simulace pomocí simulátoru NGspice na procesoru Intel Xeon E5-2630 s frekvencí 2.30 GHz. Rychlost diskrétní simulace na stejném procesoru je více než 1000x vyšší, což ukazuje to, že použít vyšší abstrakci je pro EA vhodné. Stejná softwarová verze byla otestována na procesoru ARM na čipu Zynq. Na tomto čipu, ale s využitím hardwarové akcelerační jednotky, bylo provedeno další měření. Je vidět, že v každém případě došlo ke zrychlení, ale oproti procesoru Xeon se vyplatí přenášet simulaci do hardwaru až pro 3 a více vstupů. Nejen, že dojde k úspoře času, ale

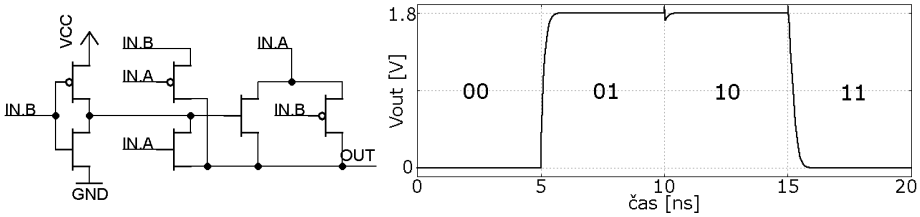
také Zynq má mnohem menší spotřebu (cca 1,5 W). Posledním změřeným parametrem byla rychlost akcelerace bez komunikace s výpočetní jednotkou. Lze vysledovat, že režie komunikace procesoru ARM s logickou jednotkou se pohybuje kolem 43 μs nezávisle na velikosti řešeného problému.

Využití zdrojů v závislosti na velikosti VRC a na počtu vstupů je ukázáno v tabulce 2. Nároky jsou velmi závislé zejména na počtu řádků. S rozšiřováním počtu řádků se totiž zvětšují multiplexory pro výběr vstupu a hlavně obvody pro zpracování zpětné vazby. Je vidět, že pro mřížku 7x4 by bylo možné CGP jednotku až šestkrát replikovat, čímž bychom dobu simulace 6x zkrátili. U mřížky velikost 8x6 by bylo možné umístit na čip obvod VRC třikrát.

Počet elementů	**2 vstupy**		**3 vstupy**		**4 vstupy**		**5 vstupů**	
	LUT	FF	LUT	FF	LUT	FF	LUT	FF
7x4	16 %	2 %	15 %	2 %	15 %	2 %	15 %	2 %
8x6	29 %	3 %	31 %	3 %	28 %	3 %	30 %	3 %
10x8	54 %	4 %	55 %	4 %	53 %	4 %	55 %	4 %

Tabulka 2: Nároky na LUT a Flip-Flop zdroje po implementaci

Navržená metoda a implementovaný akcelerátor byly ověřeny na sadě testovacích úkolů evolučního návrhu i optimalizace. Uveďme příklad pro hradlo XOR (obr. 2). Využívá 6 tranzistorů a je menší, než klasická řešení, která nemají vstup připojený k *source* tranzistoru. U tohoto obvodu fungovala i simulace využívající analogového simulátoru s tranzistory technologie TSMC 0.25μm.



Obrázek 2: Ukázka nalezeného obvodu XOR a jeho přesné simulace.

4 ZÁVĚR

Akcelerace pomocí HW jednotky se jeví jako slibná cesta pro zrychlení evolučního návrhu. Za cenu menšího příkonu a menší doby výpočtu se podařilo nalézt řešení stejně kvalitní jako v případě čistě softwarové implementace. Navržená metoda diskrétní simulace však má svá úskalí. V cca 20 % případů evoluce nalezne řešení, které pracuje korektně při frekvenci do 100 MHz. Jedním z možných rozšíření této práce je proto implementace technik umožňujících detekovat tato problematická řešení již v době evoluce a zamezit tak jejich vzniku. Další možností je implementovat podporu pro více jednotek a omezit režii danou komunikací, čímž bychom mohli dosáhnout dalšího urychlení.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory grantu GAČR 14-04197S.

REFERENCE

[1] Miller J.: *Cartesian genetic programming*. Berlin, DE: Springer Berlin Heidelberg, XXII. vydání, 2011, 344 s., ISBN 978-3-642-17309-7

[2] Walker, J.; Hilder, J.; Reid, D.; aj.: The evolution of standard cell libraries for future technology nodes. *Genetic Programming and Evolvable Machines*, ročník 12, č. 3, Springer US, 2011: s. 235-256, ISSN 1389-25

[3] Žaloudek, L.; Sekanina, L.: Transistor-level Evolution of Digital Circuits Using a Special Circuit Simulator. In *Evolvable Systems: From Biology to Hardware*, LNCS 5216, Springer Verlag, 2008, ISBN 978-3-540-85856-0, s. 320-331.