Automatizace návrhu spolehlivých systémů a její dílčí komponenty

Jakub Lojda
3. ročník, prezenční studium,
Školitel: Doc. Ing. Zdeněk Kotásek, CSc.

Fakulta informačních technologií Vysokého učení technického v Brně,
Centre of Excellence IT4Innovations
Božetěchova 1/2, 612 66 Brno, Czech Republic
Email: {ilojda, kotasek}@fit.vutbr.cz

***Abstrakt*—Vyšší úroveň integrace umožňuje implementovat stále složitější systémy. Vyšší integrace však zvyšuje riziko vzniku poruchy. Riziko je možno minimalizovat použitím technik odolnosti proti poruchám a maskování poruch. Vyšší složitost ale značně komplikuje vývoj takových systémů, který se do značné míry opírá o zkušenosti návrháře. Cílem našeho výzkumu je navrhnout metodu automatické konverze systémů neodolných na systémy odolné proti poruchám, která by uměla pracovat na téměř libovolné úrovni abstrakce. Tento článek je věnován dvěma podstatným částem výzkumu automatizace návrhu odolných systémů, tj. vkládání redundance a akceleraci vyhodnocení výsledků. Stěžejní částí článku je prezentace výsledků získaných během posledního roku výzkumu.**

***Klíčová slova*—Automatizace návrhu, HLS, vysokoúrovňová syntéza, odhad odolnosti, systém odolný proti poruchám.**

I. Úvod a Cíle výzkumu

Zvyšování integrace na čipu umožňuje realizovat složitější obvody, ale také vede na vyšší náchylnost k poruchám. Např. u hradlových polí dochází ke zvýšení jevů typu *Single Event Upset* (SEU). Obecným cílem našeho výzkumu je navrhnout metodu pro automatizovanou konverzi systému neodolného na systém odolný proti poruchám (OPP). Cílem je, aby metoda byla schopna pokrýt jak nové metody návrhu, např. vysokoúrovňovou syntézu, z angl. *High-Level Synthesis* (HLS), která pracuje na úrovni popisu algoritmu (např. C++), tak konvenční přístupy. Naše snaha vychází z obecného návrhu systémů OPP:

1) definovat zamýšlené parametry výsledného řešení,
2) vložit (modifikovat) architekturu pro dosažení OPP,
3) vyhodnotit zvažované parametry,
4) pokud řešení nesplňuje parametry, pokračovat bodem 2.

Tento článek pokrývá výzkum spojený s automatizací návrhu OPP a věnuje se především bodům 2 a 3. Sekce II cituje některé ze souvisejících výzkumů. Sekce III popisuje v úvodu využívanou platformu pro verifikaci OPP. Následují stěžejní Sekce IV a V, jež se zabývají zaváděním OPP do systémů vyvíjených pomocí HLS. Zavádění OPP je představeno v kontextu výzkumu dosavadního a také v kontextu posledního roku. Stěžejní je rovněž Sekce VI, jež je zaměřena na aktuální výsledky v oblasti urychlení a odhadu parametrů odolnosti. Sekce VII uvádí obecné cíle disertace a závěrečné zhodnocení.

II. Související práce

Následující část textu je věnována popisu aktuálních metod zavádějících OPP a rovněž akcelerujících vyhodnocení OPP.

A. Odolnost pro HLS

Autoři frameworku, který nazývají HLShield [2], představují přístup, při kterém je vstupní algoritmus označkován v profilovacím SW a následně zpracován upravenou variantou HLS, která podporuje vkládání redundance dle značek. Autoři příspěvku [14] představují manuální modifikaci algoritmu násobení matic pro získání spolehlivosti ve spojení se syntézou pomocí nástroje Vivado HLS. Na zvoleném algoritmu dokazují efektivitu jejich řešení, nicméně samotná modifikace kódu je ponechána na vývojáři.

B. Vyhodnocení odolnosti

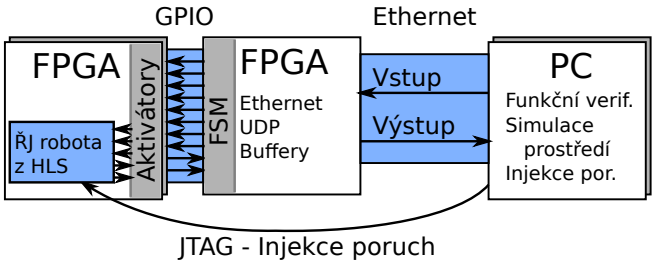
Vyhodnocení dopadu poruch je aktuálním problémem a není možné vyjmenovat všechny publikace na toto téma. Z vybraných se vyhodnocením OPP zabývají např. autoři článku [11], jenž je zaměřen na test implementací na úrovni RTL a netlist. Autoři příspěvku [13] navrhli nástroj pro injekci poruch, který ale vyžaduje modifikaci cílové implementace. Podporují mnoho syntetizovatelných modelů poruch, autoři navržené řešení implementovali v jazyce VHDL. Autoři v [10] se rovněž zaměřují na rychlost verifikace a představují řešení, kde je injekce poruch řešena přímo na čipu FPGA.

III. Verifikační prostředí s řadičem robota

V úvodu akademického roku ještě nebyl dokončen nástroj pro akceleraci vyhodnocení citlivých bitů (bude prezentován dále). Z tohoto důvodu jsme v úvodu práce využívali verifikační prostředí, jenž slouží pro finální, důkladné, ohodnocení systému i se zvážením mechanické části, což je ovšem časově mnohem náročnější. Proto přikládáme stručný popis tohoto verifikačního prostředí, jenž bylo v rámci naší skupiny již dříve publikováno v [12]. Verifikační prostředí je zaměřeno na vyhodnocení elektromechanických aplikací a je složeno ze dvou částí: 1) elektronického přípravku (kit s FPGA, do kterého jsou injektovány poruchy, zde ML506 založený na FPGA řady Virtex 5); 2) mechanického prostředí, které je simulováno na PC (zde aplikace Player/Stage [3]), implementace verifikace a injektoru poruch [15], rovněž běžících na PC. Schéma experimentálního prostředí ukazuje Obrázek 1. Pomocí injektoru poruch je možno cíleně vkládat poruchy do využitých bitů *Look-up* tabulek (LUT) a to zároveň pouze do těch, které jsou součástí specifikovaného bloku, tj. výhradně do verifikovaného obvodu.

IV. Vkládání redundance: Současný stav výzkumu

V našem předchozím výzkumu jsme se zabývali návrhem prostředků pro vkládání redundance do systémů popsaných na úrovni algoritmu a syntetizovaných pomocí HLS před průběhem samotné syntézy a bez zásahu do samotného procesu syntézy. Pro vkládání byla navržena metoda umožňující cílit redundanci za pomocí modifikovaných datových typů a s



Obrázek 1: Struktura experimentálního verifikačního prostředí.

nimi sémanticky spjatých modifikovaných operací nad těmito typy. Tyto prostředky jsme pracovně nazvali Redundantní Datové Typy (RDT). Pro každou metodu OPP je navržen příslušející RDT, který ji vkládá do algoritmu na místa, v nichž je instanciována proměnná daného datového typu, případně na místa, na kterých probíhají operace s instancí daného RDT. Tím je zajištěna automatická modifikace sémantiky operací, přičemž pro její dosažení je třeba modifikovat pouze onen datový typ proměnné. RDT jsou parametrizovatelné, každý RDT zahrnuje minimálně jeden parametr, jenž obsahuje jméno původně využitého datového typu, jehož funkci RDT zastoupí. RDT je pak možno v algoritmu používat ekvivalentním způsobem, jako typ původní, přičemž pomocí RDT je zajištěna jeho odolná implementace ve výsledné realizaci.

Pro účely experimentální implementace byl zvolen jazyk C++ v kombinaci s využitím tzv. systému *šablon*, které pro jazyk C++ umožňují efektivně realizovat koncept RDT. Koncept RDT pro syntézu systémů OPP jsme prezentovali v [9]. Jeho podrobnější vyhodnocení s využitím na řadiči robota je prezentováno v [8]. V minulosti jsme zkoumali i možnost vyhodnocení *důležitosti* jednotlivých operací v obvodové realizaci za pomocí HLS a RDT, jež se ovšem v publikaci [6] ukázala jako náročná na analýzu a interpretaci výsledků a rovněž na časové zdroje.

V. Vkládání redundance: Aktuální výsledky

Aktuální výsledky výzkumu v oblasti vkládání redundance spočívají především ve zlepšení vlastností systémů generovaných s využitím RDT a jsou publikovány v [7]. V následujícím experimentu je zkoumán vliv míry redundance a volby majoritní funkce v hlasovacím členu na výslednou odolnost systému. Ke dříve prezentovanému RDT *triple*, jenž implementuje známou architekturu TMR, jsme přidali rovněž RDT *quadruple* (4MR) a *quintuple* (5MR). Dále jsme přidali varianty s bitovou majoritou, jejichž názvy končí *_bit*. Využíváme platformy pro finální verifikaci, která zvažuje též vlivy poruchy na řízenou mechanickou část.

Redundance v obvodu ve smyslu nadbytečného využití zdrojů nemusí nutně implikovat jeho OPP. Z tohoto důvodu jsme zaměřili pozornost na definici jednotky *Intenzity Injektáže Poruch* (IIP), která je vztažena k velikosti obvodu. Velikost obvodu zde reprezentujeme počtem bitů bitstreamu, do kterých násobné poruchy injektujeme. Výsledná jednotka je tedy $injekce/s/bit$, tj. počet injekcí během jedné sekundy na jeden bit bitstreamu. Pro naše experimenty jsme zvolili IIP rovnu $2,6e$-6. Tuto volbu jsme učinili na základě experimentu s jednotkou *quintuple*. Kritériem byla výše střední doby do poruchy, tj. *Mean Time To Failure* (MTTF), a procentuální zastoupení poruchových běhů. Z důvodu rozlišení pozorovaných veličin bylo pro tuto největší z jednotek třeba volit kompromis mezi nejvyšším MTTF a nejnižší procent. chybovostí běhů.

A. Vliv míry redundance a majoritní funkce

Pro každý RDT jsme vytvořili ŘJ robota, v jejímž popise jsou všechny proměnné ošetřeny daným RDT. Počet verifikačních běhů jsme stanovili pro každou z jednotek u na $0.1 \times bity\_LUT_u$, tak, aby rozsáhlejší jednotka byla testována důkladněji. Injekce poruch probíhaly náhodně do všech bitů LUT jednotek dle *uniformního* rozdělení a dle zvolené IPP = $2.0e$-6 inj/s/bit. Počty běhů, testovaných bitů, výsledky procent. selhání a MTTF každé jednotky uvádí Tabulka I.

Tabulka I: Přehled parametrů testů společně s výsledky procent. selhání běhů a MTTF.

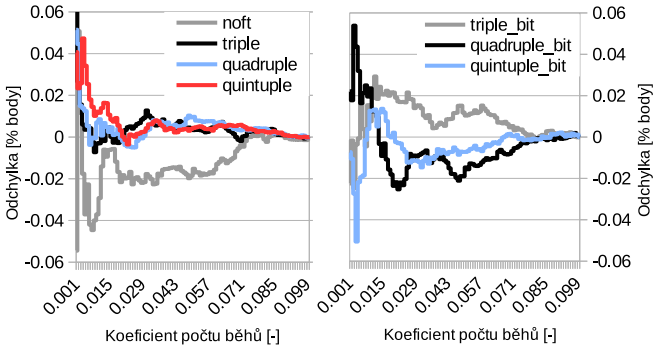
RDT aplikovaný na ŘJ robota		Parametry testů: bity LUT [b]	Počet běhů [-]	Intenz. poruch [inj/s/bit]	Obdržené výsledky: Selhané běhy[%]	MTTF [s]
noft (bez RDT)		19392	1940	2e-6	21.24	131.05
Slovní maj.	triple	48704	4871	2e-6	18.99	139.40
	quadruple	73216	7322	2e-6	20.88	138.14
	quintuple	122880	12288	2e-6	21.34	141.06
Bitová maj.	triple_bit	24480	2448	2e-6	18.91	128.68
	quadruple_bit	26784	2679	2e-6	20.87	132.88
	quintuple_bit	37632	3764	2e-6	25.05	130.08

Můžeme vidět, že aplikace *triple* snížila počet selhání, což je očekávaný stav. Pro *quadruple* se počet selhání snížil oproti nezabezpečené variantě, ale oproti *triple* došlo ke zhoršení. Předpokládáme, že tento fenomén je způsoben tím, že 4MR zabírá více plochy, zatímco pro bezporuchovou funkčnost je vyžadována funkčnost tří jednotek (tj. majorita). Tento výsledek potvrzuje fakt, že sudý počet jednotek v nMR parametry ve skutečnosti zhoršuje. Toto naznačuje i MTTF, které je pro *triple* a *quadruple* téměř ekvivalentní. Nicméně, pro *quintuple* dochází ke zvýšení procentuálního počtu selhání. Vyšší MTTF, které je s tímto údajem v rozporu ale naznačuje, že se je toto měření pravděpodobně ovlivněno vyšším rozptylem hodnot Jednotka s *triple_bit* dosáhla nejlepšího výsledku procentuálního zastoupení selhaných běhů, tj. $18.91\,\%$. V kontrastu, MTTF se snížilo. Pro tento jev opět předpokládáme vliv rozptylu MTTF. Pro *quadruple_bit* můžeme pozorovat obdobný fenomén, jako u *quadruple*. Pro *quintuple_bit* se procentuální zastoupení chybných běhů zvýšilo a MTTF snížilo. V tomto případě nepřinesla bitová majoritní funkce zlepšení, nicméně pokud zvážíme spotřebované zdroje (a to i u všech ostatních jednotek), jedná se stále o výrazně účinnější variantu při spotřebované ploše na čipu oproti slovní majoritě.

B. Počet verifikačních běhů

Protože jsme počet běhů zvolili čistě empiricky na základě předchozích zkušeností, uvádíme zde ještě retrospektivní vyhodnocení. Předpokládejme, že vyšší počet verifikačních běhů vede na přesnější výsledek a rovněž, že aritmetický průměr těchto výsledků konverguje k ideální hodnotě. Za ideální hodnoty tedy prohlasme původně vypočtené hodnoty procentuálního selhání jednotek. Za pomocí detailních záznamů pak následně simulujeme stav ohodnocení pro případ, že bychom jako počet verifikačních běhů volili koeficienty počtu běhů od 0.001 do $0.099 \times bity\_LUT_u$, namísto původních $0.1 \times bity\_LUT_u$. Referenční hodnota byla odečtena od hodnot obdržených výše popsaným způsobem. Takto jsme obdrželi graf na Obrázku 2.

Jak je možno pozorovat, od koeficientu 0.073 téměř nedochází ke změnám výsledků. Usuzujeme tedy, že počet běhů $0.1 \times bity\_LUT_u$ je pro naše ohodnocení obvodů dostačující.



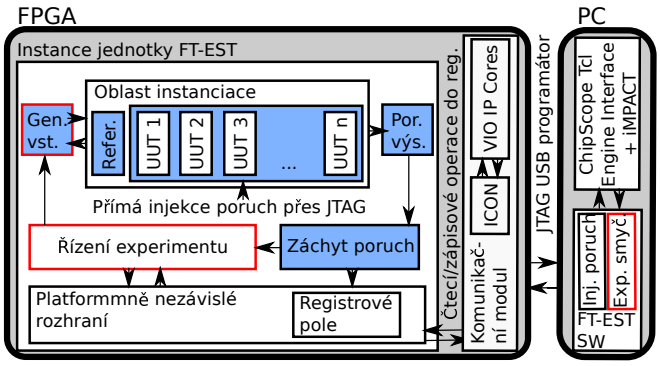
Obrázek 2: Zpětně vypočtené odchylky v procentních bodech selhaných běhů v závislosti na počtu testovaných běhů.

VI. Odhad spolehlivosti: Aktuální výsledky

Pro návrh systémů OPP je nutné mít možnost relativně rychlé verifikace obvodu. Často se stává, že aplikovaná metoda nevede správným směrem, jakým by se vývoj měl ubírat. Z tohoto důvodu je žádoucí možnost obvod otestovat s nižší přesností výsledku s cílem urychlit odhad. Závěrečná verifikace je poté provedena s vyšší přesností, případně s využitím odlišného verifikačního nástroje.

A. Struktura nástroje FT-EST

Pro tyto účely byl během posledního roku vyvinut nástroj, který umožňuje akceleraci odhadu spolehlivosti. Nástroj jsme nazvali Fault Tolerance ESTimation (FT-EST) framework. Výsledky z této části byly publikovány v [4]. Akcelerace spočívá mj. v možnosti autonomního testu, generování stimulů v rámci FPGA a možnosti paralelního spouštění testů. Míra akcelerace je pak dána velikostí testovaného obvodu. Nástroj zohledňuje možnost spouštět testy automatizovaně bez nutnosti zásahu návrháře, nicméně možnost modifikace parametrů testu je ponechána. Struktura nástroje je shrnuta na Obrázku 3.



Obrázek 3: Zjednodušená architektura nástroje FT-EST; části zvýrazněné modře jsou dynamicky generovány; části vyznačené červeně, které určují význam experimentu, volí vývojář.

Jednotka *generování vstupů* generuje stimuly do *oblasti instanciace*. V oblasti instanciace se nachází samotné instance testované jednotky, z angl. *Unit Under Test* (UUT) a také jednu instanci referenční jednotky, jež není předmětem injektáže poruch. Odtud jdou výstupní porty UUT do jednotky *porovnání vstupů* a následně jednotky *záchytu poruch*. Jednotka *řízení experimentu* zajišťuje řízení průběhu experimentu v HW části. Oddělený *komunikační modul* umožňuje snazší přenos frameworku na jiné platformy. SW část obsahuje opět komunikační modul a dříve prezentovaný injektor poruch [15], který umožňuje specifikaci injektáže do konkrétního bloku na FPGA. Výsledky o tom, který bit je pro funkčnost kritický jsou poté zaznamenávány do záznamového souboru na PC.

B. Vliv aplikace RDT na jednotlivé operace

Rozhodli jsme se využít FT-EST k odhalení slabin metody RDT. Pro tyto účely jsme zvolili tři jednoduché algoritmy. Algoritmy realizují součet dvou 16 bitových ne-znaménkových čísel a odečet dvou znaménkových 16 bitových čísel, přičemž výsledek je reprezentován rovněž na 16 bitech. Dále realizaci výpočtu CRC-8 s výstupem 8 bitů a vstupní šířkou dat 32 bitů. Na každý z těchto popisů jsme aplikovali metodu RDT a to na všechny proměnné, které byly v popisu využity vč. těch, které slouží jako vstupní parametry (tj. po syntéze jako rozhraní obvodu). Během vyhodnocení jsme k obvodům přidali ještě externí hlasovací člen, jenž byl implementován ve VHDL. Pro férové srovnání byla injektáž poruch prováděna rovněž do externího hlasovacího členu. Syntézu jsme provedli i pro obvody bez RDT. Porovnání výsledků je v Tabulce II.

Tabulka II: Využité a citlivé bity vč. jejich procent. zastoupení.

Algoritmus	metoda OPP	bity LUT [b]	Počet injekcí [-]	Počet poruch [-]	Citlivé bity [%]
Součet	bez (simplex)	4288 b	4288	890	20.76 %
Součet	TMR	8320 b	8320	225	2.70 %
Odečet	bez (simplex)	4288 b	4288	178	4.15 %
Odečet	TMR	8320 b	8320	278	3.34 %
CRC-8	bez (simplex)	4800 b	4800	1658	34.54 %
CRC-8	TMR	6592 b	6592	879	13.33 %

Výsledky naznačují obecnou funkčnost konceptu RDT, ale zároveň indikují místa, kde by mělo dojít ke zlepšení. Pro operaci odečtu došlo k procentuálnímu snížení citlivých bitů, ale z celkového pohledu je jejich počet navýšen, což není dobrý stav. Jeho řešení bude předmětem dalšího vývoje.

C. Vliv pokrytí na přesnost odhadu

V rámci experimentální činnosti s nástrojem FT-EST jsme vyhodnotili rovněž zrychlení odhadu vlivem neúplného pokrytí bitů na přesnost odhadu. Výsledky z Tabulky III ukazují, že pro zvolené obvody byl v případě 30% pokrytí poruch (tj. přibližně trojnásobného zrychlení odhadu) rozdíl v odhadech procent. zastoupení kritických bitů maximálně 3.6% bodů.

Tabulka III: Odchylka pro různé rychlosti odhadu dle pokrytí.

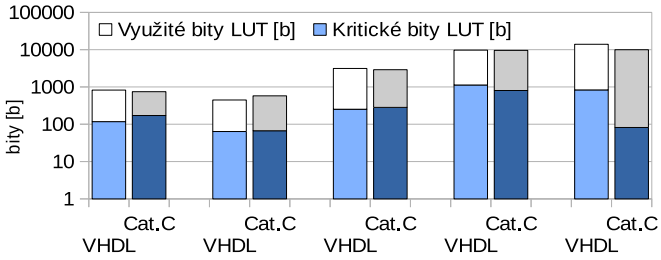
Pokrytí SEU [%]	Rozsah odchylky při odhadu [% body] Součet simplex	Součet TMR	Odečet simplex	Odečet TMR	CRC-8 simplex	CRC-8 TMR
60 %	1.63 – -1.63	0.46 – -0.40	0.70 – -0.89	0.63 – -0.46	1.71 – -1.94	1.1 – -0.97
30 %	2.57 – -2.47	0.98 – -0.78	1.91 – -1.20	0.91 – -1.1	3.38 – -3.64	1.99 – -2.46
10 %	6.06 – -5.36	1.74 – -1.50	2.61 – -2.52	1.71 – -1.9	6.29 – -6.21	4.26 – -3.78
5 %	11.0 – -9.6	2.58 – -1.98	4.71 – -3.69	3.15 – -2.62	9.63 – -9.54	5.78 – -5.14
1 %	16.6 – -16.1	6.91 – -2.7	12.2 – -4.15	8.68 – -3.34	21.7 – -19.96	18.5 – -11.8

Výsledky naznačují, že i tímto postupem je možno odhad akcelerovat a snížit tak čas potřebný k automatickému návrhu, resp. ve stejném čase prozkoumat více konfigurací a obdržet tak kvalitnější výsledek systému OPP.

D. Vliv použití HLS na spolehlivost

V publikaci [5] prezentujeme vliv použití HLS na spolehlivost syntetizovaného obvodu. Cílem experimentu bylo potvrzení předpokladu, že změna jazyka popisu výrazně neovlivní výslednou spolehlivost, pakliže porovnáváme systémy funkčně

ekvivalentní. Z testovací sady *ITC'99* [1] bylo vybráno 5 obvodů, jejichž VHDL implementace byla manuálně převedena do jazyka C++. Převod byl uskutečněn bez úmyslného vkládání optimalizací. Protože jsme záměrně vybrali implementačně nenáročné obvody, posun úrovně abstrakce popisu nebyl příliš patrný přímo z kódu, který odpovídal téměř 1:1 kódu VHDL. Výsledky byly syntetizovány pomocí nástroje Catapult C a Xilinx ISE. Původní VHDL varianty byly rovněž syntetizovány. Z výsledků zobrazených v grafu na Obrázku 4 plyne, že využití C++ pro popis nemělo výrazný vliv na měřené parametry.



Obrázek 4: Graf porovnání úrovně kritických bitů pro jednotky syntetizované z VHDL a z C++ pomocí HLS.

VII. Cíle disertační práce a závěr

Cílem disertační práce je navrhnout metodu pro automatizaci návrhu systémů OPP, pro jeho dosažení byly stanoveny tři hlavní podcíle: 1) vybudovat prostředky pro automatické vkládání redundance; 2) zvolit vhodnou variantu ohodnocení odolnosti; 3) navrhnout metodu automatické volby zabezpečení ve vztahu k vlastnostem dané komponenty (příp. části algoritmu) při zohlednění optimalizačních parametrů.

Pro podcíl 1) byly navrženy a implementovány prostředky pro selektivní vkládání OPP do algoritmu zapsaného v jazyce C++, jenž jsme nastínili v [9]. Prostředky využívají možnosti moderních programovacích jazyků vytvářet, případně modifikovat, definice datových typů a s nimi spojených operací. Pro náš výzkum jsme prostředky implementovali v C++. Dále jsme ověřili funkčnost představené metody rovněž ve vztahu k nastavení zvolených parametrů syntézy [8]. V publikaci [6] jsme ověřili možnost použití selektivního zabezpečení dílčích částí algoritmu k sestavení koeficientu *důležitosti* konkrétních instancí operací v algoritmu. Dále jsme se zabývali rozšířením o nové RDT, jež využívaly různé úrovně redundance a odlišný typ majoritní funkce [7]. V publikaci [5] jsme zkoumali vliv použití HLS a tedy i odlišného jazyka popisu na spolehlivost syntetizovaného obvodu. Výsledky naznačují velmi dobrou použitelnost HLS a rovněž nezávislost dosažené spolehlivosti na volbě jazyka vyšší úrovně abstrakce.

Pro účely podcíle 2) byl vyvinut framework FT-EST pro odhad během vývoje, jenž je popsán v [4]. V téže publikaci jsme prezentovali rovněž možnost snížení pokrytí poruch za účelem nadále až trojnásobné akcelerace odhadu (tj. bez započítání akcelerace dosažitelné použitím FT-EST) při minimálním vlivu na jeho přesnost.

V rámci podcíle 3) bude zkoumána možnost využití existujících algoritmů pro průzkum stavového prostoru s důrazem na minimalizaci počtu ohodnocení obvodů. Algoritmy vyberou z dostupných metod vkládání redundance (např. TMR, 5MR, *kódy oprav chyb*) a provedou ohodnocení takto získané pracovní verze obvodu. Mezi rozhodující parametry zprvu zařadíme samotnou spolehlivost při statickém omezení implementačního prostoru. S využitím konfigurovatelnosti FT-EST bude možno přiřadit různým typům výstupních dat (příp. operačních transakcí) odlišnou důležitost. V další fázi by mohlo být jistě zajímavé zakomponovat další možnosti omezení, např. možnost omezení spotřeby elektrické energie. V rámci bodu 3) bychom dále rádi realizovali metodu vkládání OPP pro některý konvenční jazyk popisu HW realizace (pravděpodobně VHDL) a i na tomto jazyce ukázali možnosti automatické volby OPP.

Poděkování

Tato práce byla podporována Ministerstvem školství, mládeže a tělovýchovy z Národního programu udržitelnosti (NPU II), projektu IT4Innovations excellence in science – LQ1602, projektem řešeným na VUT v Brně pod číslem FIT-S-17-3994 a projektem JU ECSEL SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), grantová dohoda č. 783119.

Reference

[1] Corno, F.; Reorda, M.; Squillero, G.: RT-level ITC'99 benchmarks and first ATPG results. *Design Test of Computers, IEEE*, ročník 17, č. 3, červenec 2000: s. 44–53, ISSN 0740-7475, doi:10.1109/54.867894.

[2] Fibich, C.; Horauer, M.; Obermaisser, R.: HLshield: a reliability enhancement framework for high-level synthesis. *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, červen 2017, s. 1–10, doi:10.1109/SIES.2017.7993378.

[3] Gerkey, B.; Vaughan, R.; Howard, A.; aj.: The Player/Stage Project. *přístupné z http://playerstage.sourceforge.net*, 2003.

[4] Lojda, J.; Podivinsky, J.; Cekan, O.; aj.: FT-EST Framework: Reliability Estimation for the Purposes of Fault-Tolerant System Design Automation. *Článek přijat k prezentaci na: Digital System Design (DSD), 2018, 21st Euromicro Conference.*

[5] Lojda, J.; Podivinsky, J.; Kotasek, Z.: Fault Tolerance Properties of Systems Generated with the Use of High-Level Synthesis. *Článek přijat k prezentaci na: 16th IEEE East-West Design & Test Symposium (EWDTS-2018).*

[6] Lojda, J.; Podivinsky, J.; Kotasek, Z.: Redundant Data Types and Operations in HLS and their Use for a Robot Controller Unit Fault Tolerance Evaluation. *2017 IEEE East-West Design & Test Symposium (EWDTS)*, září 2017, s. 359–364, doi:10.1109/EWDTS.2017.8110127.

[7] Lojda, J.; Podivinsky, J.; Kotasek, Z.; aj.: Majority Type and Redundancy Level Influences on Redundant Data Types Approach for HLS. *Článek přijat k prezentaci na: 16th Biennial Baltic Electronics Conference (BEC), 2018.*

[8] Lojda, J.; Podivinsky, J.; Kotasek, Z.; aj.: Data Types and Operations Modifications: A Practical Approach to Fault Tolerance in HLS. *2017 IEEE East-West Design & Test Symposium (EWDTS)*, září 2017, s. 273–278, doi:10.1109/EWDTS.2017.8110113.

[9] Lojda, J.; Podivínský, J.; Krčma, M.; aj.: HLS-based Fault Tolerance Approach for SRAM-based FPGAs. *Proceedings of the 2016 International Conference on Field Programmable Technology*, IEEE Computer Society, 2016, ISBN 978-1-5090-5602-6, s. 297–298.

[10] López-Ongil, C.; Garcia-Valderas, M.; Portela-García, M.; aj.: Autonomous Fault Emulation: A New FPGA-based Acceleration System for Hardness Evaluation. *IEEE Transactions on Nuclear Science*, ročník 54, č. 1, 2007: s. 252–261.

[11] Nidhin, T.; Bhattacharyya, A.; Behera, R.; aj.: Verification of Fault Tolerant Techniques in Finite State Machines Using Simulation based Fault Injection Targeted at FPGAs for SEU Mitigation. *2017 4th International Conference on Electronics and Communication Systems (ICECS)*, IEEE, 2017, s. 153–157.

[12] Podivinsky, J.; Cekan, O.; Lojda, J.; aj.: Verification of Robot Controller for Evaluating Impacts of Faults in Electro-mechanical Systems. *2016 19th Euromicro Conference on Digital System Design (DSD)*, IEEE, 2016, s. 487–494.

[13] Rudrakshi, S.; Midasala, V.; Bhavanam, S.: Implementation of FPGA based Fault Injection Tool (FITO) for Testing Fault Tolerant Designs. *IACSIT International Journal of Engineering and Technology*, ročník 4, č. 5, 2012: s. 522–526.

[14] dos Santos, A. F.; Tambara, L. A.; Kastensmidt, F. L.: Evaluating the Efficiency of using TMR in the High-level Synthesis Design Flow of SRAM-based FPGA. *2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS)*, únor 2017, s. 1–4, doi: 10.1109/LASCAS.2017.7948064.

[15] Straka, M.; Kastil, J.; Kotasek, Z.: SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems. *14th EUROMICRO Conference on Digital System Design*, IEEE Computer Society, 2011, ISBN 978-0-7695-4494-6, s. 223–230.