Funkční verifikace jako nástroj pro sledování vlivu poruch na elektro-mechanický systém

Jakub Podivínský
3. ročník, prezenční studium
Školitel: Zdeněk Kotásek

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno
ipodivinsky@fit.vutbr.cz

***Abstrakt*—Náplní tohoto článku je představení práce zabývající se využitím techniky funkční verifikace jako nástroje pro ověřování metodik pro zajištění odolnosti proti poruchám v systémech založených na FPGA. V tomto článku jsou představeny cíle disertační práce vycházející z aktuálního stavu poznání v řešené oblasti. Představen je také návrh řešení, jehož součástí je rozdělení procesu ověřování do tří fází: (1) klasická funkční verifikace, (2) funkční verifikace využívající FPGA a injektor poruch a (3) sledování vlivu poruch na mechanickou část. V této práci bude věnována pozornost zejména první a druhé fázi, které byly předmětem zkoumání v uplynulém roce. Jak již název článku napovídá, budou zde prezentovány experimenty využívající verifikační prostředí z první a druhé fáze pro sledování vlivu poruch na elektronickou část experimentálního elektro-mechanického systému.**

***Klíčová slova*—Funkční verifikace, odolnost proti poruchám, elektro-mechanický systém, řídicí jednotka robota.**

I. Úvod a motivace

Číslicové systémy hrají důležitou roli v našem každodenním životě, setkáváme se s nimi téměř v každé situaci a jejich výskyt se stále rozšiřuje. Řada z těchto číslicových systémů je využívána v prostředích, kde může docházet ke zvýšenému výskytu poruch a tyto poruchy mohou mít nedozírné následky ve formě finančních a materiálních ztrát, ale mohou ohrozit i lidské zdraví a životy. Může se jednat o různé aplikace v leteckém či vesmírném průmyslu, nebo o systémy využívané v lékařství, jejichž selhání může ohrozit zdraví pacientů.

Naše práce je zaměřená na FPGA (Programovatelná hradlová pole) využívající SRAM paměť, protože tyto obvody jsou stále populárnější a to především díky jejich vysokému výkonu a rekonfigurovatelnosti, která zajišťuje jejich vysokou flexibilitu. Obvody FPGA jsou složeny z konfigurovatelných logických bloků (CLB), které jsou propojeny konfigurovatelnou propojovací sítí. Konfigurace těchto prvků je uložena v SRAM paměti ve formě tzv. *bitstreamu*. Nevýhodou z hlediska spolehlivosti je vyšší náchylnost FPGA na poruchy způsobené nabitými částicemi. Tyto částice mohou způsobit inverzi bitu v *bitstreamu* a tím způsobit změnu chování celého obvodu. Tyto poruchy jsou nazývány *Single Event Upset* (SEU) [1]. Pro eliminování vlivu poruch se používají techniky jako *předcházení poruchám* nebo *odolnost proti poruchám* [2]. Odolnost proti poruchám (FT) je vlastnost systému, která říká, že tento systém je schopen vykonávat svou činnost i v přítomnosti poruchy. Existuje řada metodik pro zajištění odolnosti proti poruchám u systémů založených na FPGA a další jsou předmětem výzkumu na různých pracovištích [3].

Důležitou činností je testování, ověřování a porovnávání vlastností těchto metodik pro zajištění odolnosti proti poruchám. Existují nejrůznější přístupy, některé jsou spíše na teoretické úrovni, jako například simulační metoda pro emulaci SEU představená v [4]. Jiné přístupy naopak používají injekci poruch přímo do FPGA. Jako příklad poslouží speciální deska vyvinutá pro tyto účely prezentovaná autory v [5]. Ve všech případech se jedná o vyhodnocování odolnosti proti poruchám se zaměřením pouze na elektronickou část, v naší práci se chceme zaměřit také na vyhodnocování vlivu poruch na mechanickou část, kterou elektronické systémy v řadě případů řídí.

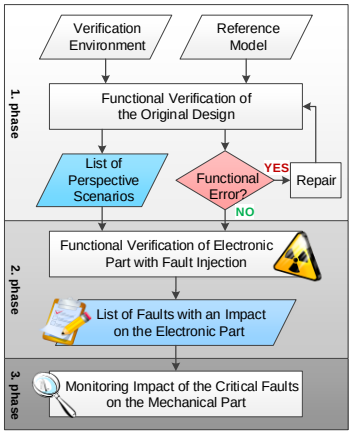
II. Cíle disertační práce a návrh řešení

V rámci své disertační práce bych rád nalezl odpověď na několik otázek: *Jaké budou výsledky FT metodik na reálných systémech? Lze spoléhat na to, že ne všechny poruchy v elektronické části systému se projeví na chování řízené mechanické aplikace? Lze využít funkční verifikaci pro ověřování vlivu poruch na FT systémy?* Na základě těchto otázek byly formulovány cíle disertační práce:

- Navrhnout a vytvořit platformu, která bude založená na technologii FPGA a bude sloužit k testování FT metodik a k sledování vlivu poruch nejen na výstup elektronické části, ale také na řízenou mechanickou aplikaci. Bude využita technika funkční verifikace a injektor poruch [6] vyvinutý týmem doc. Kotáska. Jádrem platformy bude experimentální elektromechanická aplikace.
- V návaznosti na vytvořenou testovací platformu navrhnout proces ověřování FT metodik s přihlédnutím ke specifikům elektromechanických systémů. Tyto postupy budou využívat navrženou a vytvořenou testovací platformu. Proces bude navržen na základě experimentování s vytvořenou platformou. Součástí bude popis činností před zahájením procesu ověřování. Proces bude ověřen a demonstrován na dalším experimentálním systému. Tím dojde k zobecnění získaných výsledků.

V rámci dosavadní práce byl navržen proces ověřování metodik pro zajištění odolnosti proti poruchám, který je aktuálně rozdělen do tří fází znázorněných na Obrázku 1. V *první fázi*

proběhne klasická funkční verifikace experimentálního sytému tak, aby byl zajištěn jeho soulad se specifikací. Výstupem této fáze je odladěný experimentální systém a sada vygenerovaných verifikačních scénářů, pro které byla verifikace prováděna. Ve *druhé fázi* je prováděna funkční verifikace s využitím FPGA, tak aby řídicí elektronika běžela přímo na FPGA a bylo do ní možné injektovat poruchy. V této fázi jsou využity dříve získané verifikační scénáře, pro které je jisté, že se experimentální systém bez výskytu poruch chová správně. Můžeme si tak být jisti, že případné nekorektní chování způsobila injektovaná porucha a ne chyba v implementaci. V poslední *třetí fázi* dochází k detailnímu ověřování situací, kdy se injektovaná porucha projevila na výstupu řídicí elektroniky a je vyhodnocován projev této poruchy na mechanickou část. Pro každou fázi je potřeba specifické verifikační prostředí, v tomto článku bude pozornost věnována první a druhé fázi a budou představena použitá verifikační prostředí pro náš experimentální elektromechanický systém. Jako experimentální systém používáme robota pro hledání cesty v bludišti a jeho řídicí jednotku.

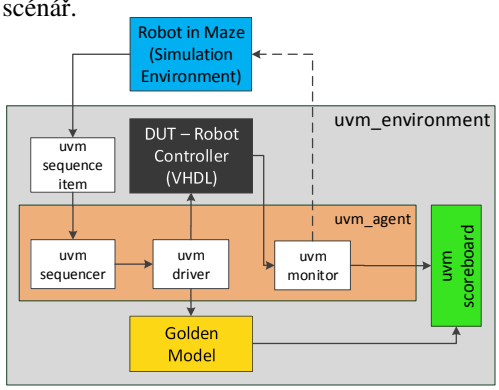


Obrázek 1. Tři fáze procesu ověřování odolnosti proti poruchám.

III. První fáze - verifikační prostředí pro řídicí jednotku robota

V první fázi navrženého procesu ověřování odolnosti proti poruchám je využívána klasická funkční verifikace založená na simulaci verifikovaného obvodu. Klasická funkční verifikace ověřuje, zda systém odpovídá specifikaci monitorováním jeho vstupů a výstupů v simulačním prostředí (např. *ModelSim*). V našem případě je verifikovaným obvodem (DUT) řídicí jednotka robota pro hledání cesty v bludišti. V první fázi využíváme verifikační prostředí, které je implementováno podle metodiky UVM (Universal Verification Methodology), což znamená, že odpovídá současným trendům a požadavkům. Toto verifikační prostředí je připraveno pro vyhodnocování jednoho verifikačního scénáře (jedno bludiště, počáteční a cílová pozice) a tvoří základ rozšiřujícího prostředí pro vyhodnocování množiny verifikačních scénářů.

Implementované verifikační prostředí je zobrazeno na Obrázku 2, kde jsou patrné základní komponenty odpovídající metodice UVM. Za pozornost stojí mechanická část, přesněji její simulace, která je řízena výstupy DUT a zároveň generuje nové vstupy pro DUT. V našem případě využíváme volně dostupné simulační prostředí pro simulaci robota Player/Stage [7]. Součástí verifikačního prostředí jsou, mimo DUT a simulaci mechanické části, také další komponenty. O produkování referenčních výstupů řídicí jednotky se stará referenční (*golden*) model, který byl implementován v rámci diplomové práce [8] v jazyce C/C++. Komponenta *sequence* získává data ze senzorů z robota v simulačním prostředí a transforumuje je na transakce pro DUT, které dále předává komponenta *driver* jako vstup DUT a referenčního modelu. Naopak *monitor* čte výstupní data z DUT (směr a rychlost pohybu robota) a předává je (1) zpět do simulačního prostředí robota, který provede zadaný pohyb, a (2) do komponenty *scoeboard* k porovnání s výstupem referenčního modelu. Výstupem tohoto verifikačního prostředí je informace o tom, zda DUT pro tento verifikační scénář respektuje specifikaci či nikoliv, a zároveň také informace o pokrytí kódu (*codecoverage*) pro daný verifikační scénář.



Obrázek 2. Verifikační prostředí pro jeden verifikační scénář.

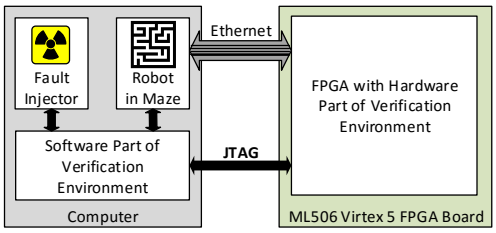
Představené verifikační prostředí není schopno automaticky vyhodnotit více verifikačních scénářů, tuto činnost zajišťuje rozšířený proces ověřování. Verifikační prostředí tvoří jednu z komponent toho procesu. Důležité je také generování bludiště, kdy pro každý verifikační běh je vygenerováno nové bludiště. Po každém běhu verifikačního prostředí je uložen ověřený verifikační scénář uložen spolu s informací o dosaženém pokrytí kódu. Po skončení celého procesu ověření množiny verifikačních scénářů jsou sloučeny jednotlivé dílčí informace o pokrytí kódu a výsledkem je jediný report.

IV. Druhá fáze - architektura testovací platformy

V druhé fázi je využíváno verifikační prostředí, které pro implementaci DUT využívá FPGA, což umožňuje injekci poruch přímo do reálného obvodu. Pro potřeby provozování verifikačního prostředí využívajícího FPGA jsme navrhli platformu složenou z několika komponent běžících na FPGA nebo na počítači:

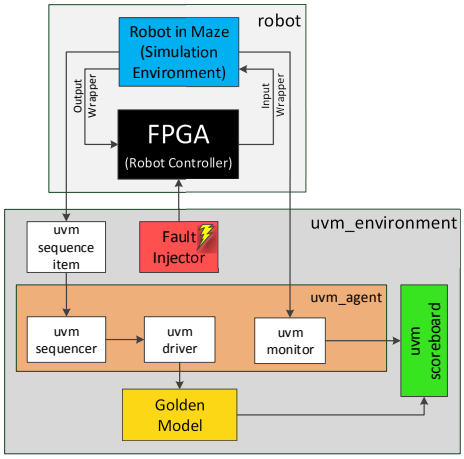
- SW část verifikačního prostředí běžící na počítači,
- SW simulační prostředí pro simulaci robota (Player/Stage) běžící na počítači,
- řídicí jednotka robota implementovaná na FPGA,
- externí injektor poruch [6] běžící na počítači, který umožňuje simulovat poruchy přímo v FPGA.

Celkovou architekturu této platformy zachycuje Obrázek 3. Mimo zmíněných komponent jsou zde patrné také komunikační kanály mezi FPGA a počítačem. Pro komunikaci mezi řídicí jednotkou robota a robotem je použito rozhraní Ethernet. Rozhraní JTAG je určeno pro programování FPGA a následně také pro injekci poruch do FPGA.



Obrázek 3. Struktura experimentální platformy.

Verifikační prostředí využívající FPGA (Obrázek 4) vychází z velké části z výše uvedeného klasického verifikačního prostředí. Zobrazené verifikační prostředí je rozdělené na dvě části, první částí je simulovaný robot komunikující s řídicí jednotkou na FPGA. Tato část pracuje zcela autonomně, mezi FPGA a simulačním prostředím proudí informace ze senzorů a pokyny k pohybu robota, v simulačním prostředí lze sledovat pohyb robota. Druhou částí je samotné verifikační prostředí, které převzalo většinu komponent z původního prostředí. Toto verifikační prostředí zde funguje jako pozorovatel, což znamená, že pouze pasivně sleduje komunikaci mezi FPGA a simulací robota a porovnává je s výstupy referenčního modelu a nijak do této komunikace nezasahuje.



Obrázek 4. Architektura verifikačního prostředí využívajícího FPGA.

Stejně jako v první fázi, i zde je třeba zajistit automatické vyhodnocování množiny verifikačních scénářů spolu s injekcí poruch. V tomto procesu již nedochází ke generování bludiště, jelikož ve druhé fázi se pracuje s množinou vygenerovaných a ověřených bludišť. Po naprogramování FPGA, spuštění simulace robota a verifikačního prostředí robot začíná procházet bludištěm. A právě zde nastává prostor pro injekci poruch do FPGA. Poruchy mohou být injektovány dle zvolené strategie, například jednonásobné poruchy do vybrané funkční jednotky, vícenásobné poruchy do jedné nebo více funkčních jednotek a podobně. Výstupem tohoto procesu je report shrnující provedené experimenty, pro každý běh je uložena také pozice injektované poruchy pro použití v další fázi.

Pro potřeby injekce poruch máme vyvinutý vlastní injektor poruch [6], který umožňuje injektovat poruchu do zadaného bitu bitstreamu. Tento injektor využívá rozhraní JTAG a částečnou dynamickou rekonfiguraci, kdy nejprve vyčte část používaného bitstreamu, modifikuje zadaný bit a uloží tento upravený bitsream zpět do FPGA. Pro zjištění relace mezi bitem bitstreamu a funkční jednotkou používáme nástroj RapidSmith , který umí analyzovat FPGA a najít seznam bitů bitstreamu odpovídající využitým LUT tabulkám v zadané oblasti FPGA. Díky rozmístění funkčních jednotek na FPGA pomocí nástroje PlanAhead víme, v které části FPGA se nachází jednotlivé funkční jednotky.

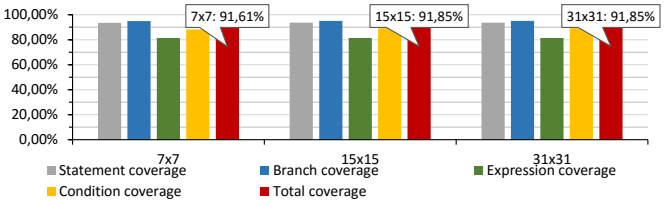
V. Experimentální výsledky

Doposud prováděné experimenty s představenými verifikačními prostředími korespondují s první a druhou fází navrženého procesu ověřování odolnosti proti poruchám. V první fázi jsme prováděli funkční verifikaci řídicí jednotky robota s využitím představeného verifikačního prostředí. Výstupem této fáze je (1) odladěná řídicí jednotka robota, (2) množina použitých verifikačních scénářů a (3) report informující o dosaženém pokrytí kódu. Pro experimenty v této fázi jsme použili tři rozměry bludiště: 7x7, 15x15 a 31x31 políček. Průměrný počet kroků, které musí robot vykonat, zachycuje Tabulka I.

Tabulka I
Průměrný počet kroků robota

Velikost bludiště	7x7	15x15	31x31
Průměrný počet kroků	16	93	433

V průběhu této fáze jsme sledovali zejména vliv počtu vygenerovaných bludišť a velikost bludiště na dosažené pokrytí kódu. Provedli jsme funkční verifikaci pro 10, 100, 200 a 500 verifikačních scénářů pro každý rozměr bludiště. Narůstající počet verifikačních scénářů neměl téměř žádný vliv na dosažené pokrytí . Sloupcový graf 5 zobrazuje dosažené pokrytí kódu (jednotlivé dílčí složky a celkové pokrytí) pro 100 verifikačních běhů pro různé rozměry bludiště. Z tohoto grafu je patrné, že ani rozměr bludiště neměl podstatný vliv na dosažené pokrytí, navzdory původnímu předpokladu.



Obrázek 5. Pokrytí kódu pro 100 verifikačních běhů s bludišti různých velikostí.

Ve druhé fázi byly provedeny experimenty využívající verifikační prostředí s FPGA a injektor poruch. Experimenty byly prováděny s řídicí jednotkou robota bez aplikace metodik pro zajištění odolnosti proti poruchám, jejich cílem byla (1) analýza řídicí jednotky robota z pohledu spolehlivosti a (2) demonstrace funkčnosti vytvořené testovací platformy.

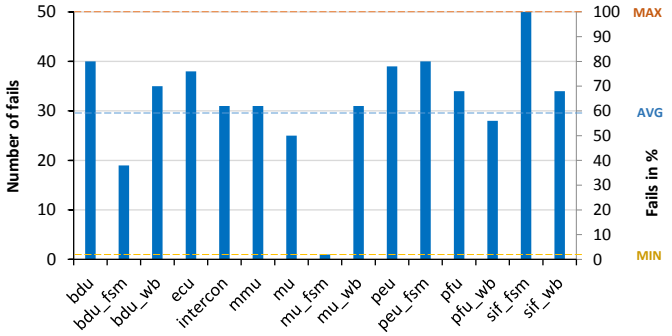
Bylo provedeno 50 verifikačních běhů pro každou funkční jednotku s verifikačními scénáři získanými během předchozí

fáze, přesněji se jednalo o bludiště o rozměrech 15x15 políček, která nám zajistila dostatek času pro injekci poruchy a následné sledování jejího vlivu před tím, než robot dorazil do cíle. Řídicí jednotka robota obsahuje 15 funkčních bloků, dohromady tedy bylo provedeno 750 verifikačních běhů. Tento počet byl zvolen především z časových důvodů, jelikož jeden krok robota trvá přibližně 5s, projití celého bludiště pak zabere cca 10minut. Pro injekci poruch jsme si zvolili strategii injekce jednonásobné poruchy do vybrané funkční jednotky během jednoho verifikačního scénáře. Tabulka II shrnuje získané výsledky, pro každý funkční blok je zde uveden počet poruch (z celkového počtu 50 injektovaných poruch), které způsobily neshodu mezi výstupem řídicí jednotky robota a referenčním modelem. Uvedeno je také procentuální vyjádření.

Tabulka II
EXPERIMENTÁLNÍ VÝSLEDKY: VLIV PORUCH NA VÝSTUP ELEKTRONIKY.

Funkční blok	Počet poruch [-]	Počet poruch [%]	Funkční blok	Počet poruch [-]	Počet poruch [%]
bdu	40	80	mu_wb	31	2
bdu_fsm	19	38	peu	39	78
bdu_wb	35	70	peu_fsm	40	80
ecu	38	76	pfu	34	68
intercon	31	62	pfu_wb	28	56
mmu	31	62	sif_fsm	50	100
mu	25	50	sif_wb	34	68
mu_fsm	1	2			

Výsledky experimentů shrnuje také Obrázek 6, ze kterého je jasně patrné, že některé funkční bloky jsou na poruchy citlivější než jiné. V grafu je zakreslena průměrná hodnota, která je přibližně 60%. Některé bloky se odchylují méně, jiné více. Největší odchylka od průměru je patrná u bloku *sif_fsm*, kde byla dosažena hodnota 100%, naopak u bloku *mu_fsm* se projevily pouze 2%. U těchto bloků jsme se rozhodli zopakovat experimenty s použitím většího počtu verifikačních scénářů. Pro každý tento blok jsme provedli dalších 225 verifikačních běhů. Získané výsledky se přiblížily průměrným hodnotám.



Obrázek 6. Experimentální výsledky: vliv poruch na výstup elektroniky.

Závěr z našich experimentů je jednak zjištění, které funkční bloky jsou více či méně náchylné k poruchám, ale také ke zjištění, že jsme schopni pomocí funkční verifikace doplněné injekcí poruch do FPGA ověřovat vliv poruch na elektromechanický systém, přesněji na jeho elektronickou část (řídicí jednotka robota).

VI. ZÁVĚR A DALŠÍ PRÁCE

V této práci byl představen posun v řešení disertační práce na téma ověřování metodik pro zajištění odolnosti proti poruchám pomocí kombinace funkční verifikace a injekce poruch do FPGA. Byl zde představen návrh procesu pro ověřování odolnosti proti poruchám rozděleného do tří fází, kdy každá fáze vyžaduje specifické verifikační prostředí. V tomto článku bylo představeno verifikační prostředí odpovídající první (klasická funkční verifikace) a druhé (verifikační prostředí využívající FPGA) fázi navrženého procesu. Tato verifikační prostředí byla také předmětem experimentů, jejich výsledky byly v článku také stručně představeny. Představené výsledky byly průběžně publikovány na mezinárodních konferencích a v odborném časopise, výčet publikací bude součástí prezentace.

V budoucí práci se plánujeme zaměřit jednak na třetí fázi navrženého procesu, tedy na automatizaci sledování vlivu poruch na mechanickou část. Musíme vytvořit další verifikační prostředí, které bude schopné provádět tuto kontrolu automaticky. Díky používanému simulačnímu prostředí Player/Stage jsme schopni sledovat nejen informace ze senzorů robota, ale také informace o chování a pozici robota v bludišti. Současně plánujeme také aplikaci různých metodik pro zajištění odolnosti proti poruchám a jejich ověřování pomocí naší platformy. Plánujeme využít klasické ztrojení (TMR) spolu s naším řadičem rekonfigurace, on-line hlídací obvody a podobně.

PODĚKOVÁNÍ

Tato práce byla podpořena Ministerstvem školství mládeže a tělovýchovy z Národního programu udržitelnosti II (NPU II) v rámci projektu IT4Innovations excellence in science - LQ1602, projektem ARTEMIS JU č. 641439 (ALMARVI) a projektem VUT FIT-S-14-2297.

REFERENCE

[1] M. Ceschia, M. Violante, M. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin, and A. Candelori, "Identification and Classification of Single-event Upsets in the Configuration Memory of SRAM-based FPGAs," vol. 50, no. 6, 2003, pp. 2088–2094.

[2] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[3] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 37:1–37:34, Jan. 2015.

[4] C. Bernardeschi, L. Cassano, A. Domenici, and L. Sterpone, "Accurate Simulation of SEUs in the Configuration Memory of SRAM-based FPGAs," in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 115–120.

[5] M. Alderighi, F. Casini, S. d'Angelo, M. Mancini, S. Pastore, and G. R. Sechi, "Evaluation of Single Event Upset Mitigation Schemes for SRAM-based FPGAs Using the FLIPPER Fault Injection Platform," in *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*. IEEE, 2007, pp. 105–113.

[6] M. Straka, J. Kastil, and Z. Kotasek, "SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems," in *14th EUROMICRO Conference on Digital System Design*. IEEE Computer Society, 2011, pp. 223–230.

[7] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-robot and Distributed Sensor Systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[8] S. Krajcir, "Functional Verification of Robotic System Using UVM," Tech. Rep., 2015. [Online]. Available: http://www/study/DP/DP.php?id=15154