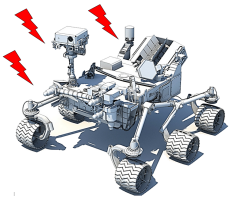
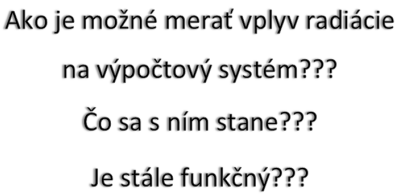
Excel@FIT 2015
http://excel.fit.vutbr.cz

Funkčná verifikácia robotického systému pomocou metodiky UVM

Stanislav Krajčir*





Abstrakt
Problém zabezpečenia odolnosti systémov proti poruchám patrí medzi veľmi aktuálne témy. Zariadenia, ktoré sú vysielané do vesmíru bývajú vystavené veľmi extrémnym podmienkam, ktoré môžu ovplyvniť ich funkčnosť. Z hľadiska efektívneho využívania zdrojov je dôležité mať možnosť overiť mieru odolnosti týchto zariadení proti poruchám ešte predtým, než im budú vystavené v skutočnom prostredí. Jedným zo spôsobov je využitie softvérovej simulácie. V rámci simulačného prostredia sa do zariadenia injektujú poruchy s cieľom overenia miery odolnosti daného zariadenia proti poruchám. Cieľom tejto práce je predstaviť možnosti automatizácie celého tohto procesu s využitím funkčnej verifikácie. Funkčná verifikácia umožňuje celý tento proces zrýchliť a zefektívniť. Jedná sa o inovatívne riešenie, pretože tento spôsob využitia funkčnej verifikácie nebol zatiaľ nikde realizovaný. V rámci práce je predstavený návrh a implementácia verifikačného prostredia. Verifikačné prostredie zohráva veľmi dôležitú úlohu v rámci celého procesu overovania odolnosti systémov proti poruchám. Výsledky tejto práce budú využité v rámci výskumu odolnosti systémov proti poruchám, ktorý prebieha na Ústave počítačových systémov Fakulty informačných technológií VUT.

Kľúčové slová: funkčná verifikácia, UVM, SystemVerilog, odolnosť systémov proti poruchám

*xkrajc06@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

[Motivácia] Systémy, ktoré sa vysielajú do vesmíru bývajú vystavené najrôznejším formám kozmického žiarenia. Cieľom rôznych výskumných organizácií je zabezpečenie správnej funkcionality týchto zariadení aj v takýchto extrémnych podmienkach. Hľadaním odpovedí na otázky, ktoré riešia úlohy tohto typu sa zaoberá oblasť zabezpečenia odolnosti systémov proti poruchám.

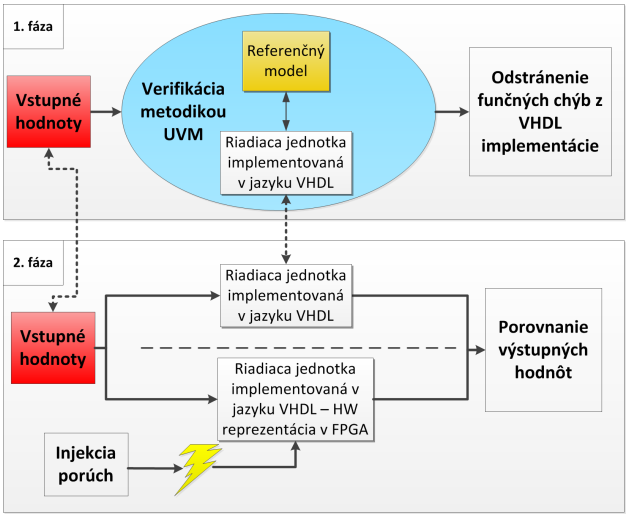
[Definícia problému] Kozmické žiarenie (protóny, neutróny a častice alfa), ktorému sú systémy vo vesmíre vystavené obsahuje veľké množstvo energie, ktoré je schopné ovplyvniť fungovanie elektromechanických zariadení. V rámci skúmania odolnosti systémov proti poruchám je veľmi dôležité overiť aké účinky môže žiarenie na zariadenie mať a aké chyby môže spôsobiť. Z hľadiska efektívneho využívania materiálov a finančných zdrojov je veľmi dôležité mať možnosť overiť mieru odolnosti systému proti poruchám predtým, než bude týmto poruchám systém vystavený v reálnom nasadení.

[Existujúce riešenia] Existuje niekoľko využívaných prístupov k overeniu miery odolnosti systému proti poruchám. Jedným z nich je vytvorenie podobných podmienok ako sú vo vesmíre tu na Zemi. Zariadenie je v špeciálnej miestnosti vystavené týmto pod-

mienkam, pričom sa sleduje ich vplyv na jeho funkčnosť. Na základe získaných informácií sa zariadenie alebo jeho časti upravujú tak, aby boli schopné týmto extrémnym podmienkam odolať. Toto riešenie je však finančne nákladné. Druhým možným prístupom je vytvorenie softvérovej simulácie, v rámci ktorej sa do daného zariadenia umelo injektujú rôzne typy porúch. Tento spôsob je momentálne predmetom výskumu, ktorý prebieha na Ústave počítačových systémov (ÚPSY) Fakulty informačných technológii VUT. V rámci výskumnej skupiny Diagnostika bola na ÚPSY vytvorená riadiaca jednotka robotického systému určeného pre samočinný pohyb v bludisku [1],[2]. Táto riadiaca jednotka bola navrhnutá tak, že obsahuje mechanizmy, ktoré zabezpečujú odolnosť proti poruchám (napr. ide o mechanizmy *Triple Modular Redundancy – TMR*, *N-Modular Redundancy – NMR* a *Duplexné zabezpečenie*). Účelom tejto práce nie je presný popis zabezpečovacích mechanizmov. V prípade záujmu sú informácie o nich uvedené v publikácii [3]. V rámci simulačného prostredia sa do riadiacej jednotky injektujú poruchy a sleduje sa, ako na ne bude riadiaca jednotka reagovať. Tento spôsob overovania je z časového hľadiska zdĺhavý, pretože je potrebné manuálne spustiť celý simulačný proces a sledovať, či robot plní svoju funkciu a správne prechádza bludiskom zo štartovacej pozície do cieľovej pozície.

[Naše riešenie] Keďže sledovanie vplyvu porúch na riadiacu jednotku robota v rámci procesu overovania odolnosti jednotky proti poruchám v simulácii bolo časovo zdĺhavé, vzišiel návrh celý proces zautomatizovať, a tým pádom ho výrazne zrýchliť a zefektívniť. Pre realizáciu tohto návrhu sme sa rozhodli použiť prístup, ktorý na základe našich informácií zatiaľ nebol za týmto účelom nikdy použitý. Jedná sa o využitie funkčnej verifikácie. Výhody využitia tohto prístupu budú uvedené v ďalšej časti tejto práce. Cieľom tejto práce je vytvoriť verifikačné prostredie, pričom v rámci práce budú využité už existujúce časti výskumu z ÚPSY. Schematický princíp riešenia s využitím verifikačného prostredia je znázornený na obrázku 1.

Samotný proces je možné rozdeliť na dve hlavné fázy. **Prvá fáza** tvorí základ tejto práce (návrh a implementácia verifikačného prostredia - hlavný prínos autora článku). Jej cieľom je overiť korektnosť riadiacej jednotky robota. Za týmto účelom bude vytvorené verifikačné prostredie v jazyku SystemVerilog s využitím metodiky UVM (*Universal Verification Methodology*). Funkcionalita riadiacej jednotky bude overovaná voči funkcionalite tzv. referenčného modelu. Referenčný model je súčasťou verifikačného prostredia a implementuje rovnakú činnosť ako riadiaca jednotka. Jeho



Obrázok 1. Princíp fungovania verifikačného prostredia.

implementácia je nezávislá na implementácii riadiacej jednotky, vychádza iba z rovnakej špecifikácie funkcionality. Na základe vstupných hodnôt, ktoré budú aplikované ako na vstup riadiacej jednotky, tak aj na vstup referenčného modelu, budeme overovať korektnosť jednotky za účelom odstránenia funkčných chýb z jej implementácie . Funkčná verifikácia disponuje tzv. analýzou pokrytia (*coverage analysis*), ktorá nám poskytuje informácie o tom, ktoré časti riadiacej jednotky boli počas verifikácie preverené. Na základe tejto informácie vieme z množiny vstupných hodnôt vybrať iba tie hodnoty, pomocou ktorých je dosiahnuté úplné pokrytie riadiacej jednotky. Ako metriku pokrytia je možné zvoliť napr. pokrytie kódu. Táto vyselektovaná množina vstupných hodnôt je pre nás veľmi dôležitá, nakoľko bude tvoriť základ pre druhú fázu. Prvá fáza bude bežať kompletne v softvérovom simulačnom prostredí.

Druhá fáza priamo nadväzuje na výsledky prvej fázy a bude v budúcnosti realizovaná v rámci výskumu na ÚPSY. Samotná realizácia druhej fázy nie je predmetom tejto práce. Nakoľko však priamo nadväzuje na výsledky tejto práce je tu uvedená z dôvodu, aby čitateľ získal celkový prehľad o riešení problematiky, ktorý je predmetom výskumu na ÚPSY. Jej cieľom je verifikácia riadiacej jednotky, ktorá bude umiestnená v hardvérovom obvode FPGA (*Field-Programmable Gate Array*), do ktorej budú umelo injektované poruchy. Ako referenčný model bude využitá riadiaca jednotka robota, ktorú sme si pripravili v prvej fáze. Teda táto riadiaca jednotka bude odladená a nebude obsahovať funkčné chyby. Vstupy bude tvoriť vyselektovaná množina vstupných hodnôt o ktorých vieme, že poskytujú úplné pokrytie. Tieto vstupné hodnoty budú privedené na vstup referenčnej riadiacej

jednotky, ktorá bude fungovať v softvérovej simulácii a na vstup riadiacej jednotky, ktorá bude umiestnená v obvode FPGA, teda bude realizovaná v hardvéri. Do hardvérovej jednotky zároveň budú injektované poruchy a výstupy medzi týmito dvomi jednotkami budú porovnávané. Na základe porovnania bude možné určiť, či injektovaná porucha ovplyvnila funkčnosť riadiacej jednotky.

[Prínos práce] Tvorba verifikačného prostredia zohráva veľmi dôležitú úlohu v rámci celého procesu overovania odolnosti systémov proti poruchám na ÚPSY. Hlavné prínosy sú nasledujúce:

1. Odstránenie funkčných chýb z implementácie riadiacej jednotky robota. Toto nám zaručí, že riadiaca jednotka je odladená a vykonáva požadovanú funkcionalitu korektne.
2. Vyselektovanie množiny referenčných vstupov na základe analýzy pokrytia. Referenčné vstupy budú vybraté iba za predpokladu, že ich aplikáciou na verifikovanú jednotku bolo dosiahnuté úplné pokrytie kódu. Referenčné vstupy budú následne využité pre overenie korektnosti riadiacej jednotky po injekcii porúch.
3. V prípade detekcie chyby v rámci druhej fázy procesu bude možné určiť konkrétny vstupný stimul, ktorý spôsobil poruchu a bude možné určiť presné miesto, kde chyba v rámci riadiacej jednotky, ktorá bude umiestnená v obvode FPGA, vznikla. V rámci simulácie (opätovné použitie fázy 1, ale len pre tento vybraný vstup) bude následne možné identifikovať jednoznačnú cestu, ktorou daný vstupný stimul prešiel od vstupného rozhrania až do miesta spôsobenia poruchy. Nakoľko v rámci druhej fázy bude ako referenčný model vystupovať odladená implementácia riadiacej jednotky, tak budeme jednoznačne vedieť, že chyba, ktorá bola detekovaná, je spôsobená samotnou injekciou. Celý proces overovania odolnosti riadiacej jednotky proti poruchám bude jednoducho reprodukovateľný, čím sa zvýši jeho efektívnosť.

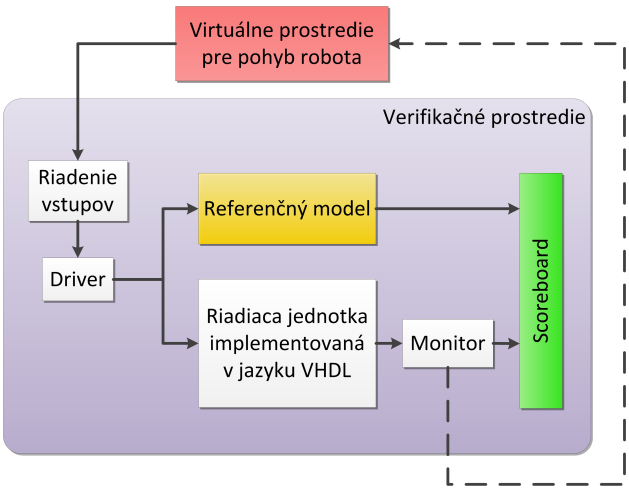
Táto práca pokrýva prvé dva body z vyššie uvedených prínosov (prvá fáza - tvorba verifikačného prostredia). Na tvorbu verifikačného prostredia priamo nadväzuje tretí bod (druhá fáza), ktorý bude v budúcnosti realizovaný v rámci výskumu na ÚPSY.

2. Teoretický úvod

Jedným z aktuálne najviac využívaných prístupov pre verifikáciu hardvérových systémov je **funkčná verifikácia**. Je založená na simulácii, pričom využíva ďalšie prídavné techniky za účelom zefektívnenia verifikačného procesu, medzi ktoré patria napr. generovanie náhodných vstupných stimulov, verifikácia riadená pokrytím, verifikácia založená na formálnych tvrdeniach a samo-kontrolné mechanizmy. Pre tvorbu verifikačných prostredí sa využíva programovací jazyk **SystemVerilog** [4], ktorý vznikol rozšírením jazyka Verilog o konštrukcie, ktoré umožňujú zvýšiť efektívnosť procesu simulácie a verifikácie. Za účelom zjednodušenia tvorby verifikačných prostredí a zabezpečenie ich jednoduchej prenositeľnosti a znovupoužiteľnosti boli v priebehu času vytvorené rôzne verifikačné metodiky napr. *Verification Methodology Manual – VMM, Open Verification Methodology – OVM* a *Universal Verification Methodology – UVM*. Tieto metodiky boli vytvorené konzorciom veľkých nadnárodných spoločností zaoberajúcich sa výrobou, predajom a verifikáciou hardvérových obvodov. Ide o spoločnosti Mentor Graphics, Cadence Design Systems, Synopsis a Accellera. Metodika UVM [5] je najnovšou metodikou a stavia na úspechoch nasadenia metodík VMM a OVM.

3. Návrh verifikačného prostredia

Verifikačné prostredie pre robotickú jednotku je navrhnuté pomocou metodiky UVM. Zjednodušená schéma návrhu verifikačného prostredia je uvedená na obrázku 2.



Obrázok 2. Návrh verifikačného prostredia.

Verifikačné prostredie je zostavené z niekoľkých UVM komponentov, ktorých zjednodušený popis je nasledovný:

- *Riadiaca jednotka* – riadiaca jednotka robota, implementovaná v jazyku VHDL, ktorá je verifikovaná. Je prevzatá z [1].
- *Referenčný model* – model, ktorý vykonáva rovnakú funkciu ako riadiaca jednotka robota. Bude

implementovaný v jazyku C na základe návrhu.

- *Riadenie vstupov* – získava hodnoty zo senzorov z virtuálneho prostredia pohybu robota (ďalej v texte len virtuálne prostredie) a transformuje ich na vstupy pre verifikačné prostredie,
- *Driver* – vstupné hodnoty preposiela na vstup referenčného modelu a na vstup riadiacej jednotky robota,
- *Monitor* – výstupy z riadiacej jednotky robota odosiela do jednotky scoreboard,
- *Scoreboard* – porovnáva výstupy z jednotky monitor (verifikovanej jednotky) a referenčného modelu a kontroluje, či sa zhodujú.

Verifikačné prostredie bude komunikovať s virtuálnym prostredím pre pohyb robota. Virtuálne prostredie slúži pre znázornenie pohybu robota pri priechode bludiskom. Na vstup verifikačného prostredia bude odosielať informácie o polohe robota v bludisku a vzdialenosti od okolitých stien bludiska (hodnoty zo senzorov). Verifikačné prostredie bude poskytovať virtuálnemu prostrediu spätnú väzbu, na základe ktorej zobrazuje priechod robota bludiskom.

3.1 Návrh referenčného modelu

Hlavnou časťou verifikačného prostredia je referenčný model. Referenčný model bude implementovaný v jazyku C, a to na základe špecifikácie, z ktorej vychádzala aj implementácia riadiacej jednotky robota vo VHDL, ktorá je prevzatá z [2]. Postup operácií, ktoré bude referenčný model vykonávať je nasledovný:

1. Načítanie mapy bludiska, štartovacej a cieľovej pozície,
2. Výpočet aktuálnej pozície,
3. Výpočet vektora prekážok,
4. Výpočet cesty v bludisku.

Mapa bludiska môže byť ľubovoľného rozmeru. Skladá sa z jednotlivých políčok, pričom každé políčko je reprezentované pomocou logickej hodnoty 1 a 0. Políčko, ktoré reprezentuje stenu, je označené logickou hodnotou 0 a políčko, ktoré reprezentuje cestu, je označené logickou hodnotou 1. Na obrázku 3 je znázornený príklad mapy o veľkosti 8 x 8 políčok spolu s jej binárnou reprezentáciou.

Výpočet pozície: Aktuálna, štartovacia a cieľová pozícia sa vypočítava pomocou vyhodnotenia vzdialenosti od troch kontrolných bodov, ktoré sú rozmiestnené na fixných pozíciách v bludisku. Pozícia týchto bodov je známa. Na základe vzdialenosti od týchto bodov vieme pomocou Pythagorovej vety vypočítať presnú pozíciu robota.

1	1	1	1	1	1	1	1
1	0	1	1	0	0	0	1
1	0	0	0	0	1	0	1
1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1
1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1
1	1	1	1	1	1	1	1

Obrázok 3. Mapa bludiska a jej bitová reprezentácia.

Výpočet vektora prekážok: Pre vyhodnotenie prekážok sa využívajú štyri senzory, pričom každý senzor je umiestnený práve na jednej strane robota. Výpočet vektora prekážok spočíva v porovnaní hodnoty získanej zo senzora s referenčnou hodnotou. V prípade, ak hodnota získaná zo senzora je menšia, než referenčná hodnota, tak na danom políčku sa nachádza prekážka. V prípade, ak získaná hodnota je väčšia, než referenčná hodnota, dané políčko je prázdne a robot sa môže na toto políčko presunúť. Vektor prekážok je reprezentovaný pomocou štyroch bitových hodnôt (štyri svetové strany).

Výpočet cesty v bludisku: Samotný výpočet bude realizovaný pomocou algoritmu „Ľavou rukou po stene“. Tento algoritmus vychádza z pravidla, ktoré tvrdí, že pokiaľ sa budeme počas hľadania východu z bludiska držať vždy ľavou rukou steny, tak nezablúdime. Buď nájdeme východ z bludiska alebo sa vrátime späť na počiatočnú pozíciu. Referenčný model teda v každom kroku vyhodnotí svoju aktuálnu pozíciu, porovná ju s cieľovou pozíciou a v prípade, ak sa nenachádza v cieli, určí nasledujúcu súradnicu pohybu tak, aby sa vždy držal ľavou rukou steny bludiska.

4. Implementácia

Na základe návrhu, ktorý bol uvedený v kapitole 3 bolo implementované verifikačné prostredie a referenčný model.

4.1 Implementácia verifikačného prostredia

Verifikačné prostredie bolo implementované v jazyku SystemVerilog pomocou metodiky UVM. Toto verifikačné prostredie beží v simulačnom programe ModelSim verzie 10.0 od spoločnosti Mentor Graphics s knižnicou UVM verzie 1.1d. Jednotlivé zložky verifikačného prostredia boli implementované pomocou komponentov, ktoré ponúka knižnica UVM. Ide napr. o komponenty *uvm_env, uvm_test, uvm_agent, uvm_scoreboard, uvm_monitor,* atď.

4.2 Implementácia referenčného modelu

Referenčný model bol na základe návrhu uvedenom v kapitole 3.1 implementovaný v jazyku C. Referenčný model funguje ako konzolová aplikácia. Aplikácia požaduje päť vstupných hodnôt: mapu bludiska, počiatočnú súradnicu x, počiatočnú súradnicu y, cieľovú súradnicu x a cieľovú súradnicu y. Výstupom programu je informácia o aktuálnej pozícii v bludisku pri jeho prehľadávaní. Program ukončí korektne svoju činnosť, keď sa robot presunie na cieľovú pozíciu. V rámci procesu výpočtu sa využivajú nasledujúce premenné:

- X – výsledná súradnica v dimenzii X,
- Y – výsledná súradnica v dimenzii Y,
- APP_X – súradnica v dimenzii X pred normalizáciou,
- APP_Y – súradnica v dimenzii Y pred normalizáciou,
- BOX_SIZE – konštanta, ktorá slúži na prevod (normalizáciu) vypočítanej hodnoty súradnice na hodnotu reprezentujúcu políčko v bludisku,
- DIST_A – vzdialenosť od bodu A, ktorého poloha je fixná,
- DIST_B – vzdialenosť od bodu B, ktorého poloha je fixná,
- DIST_C – vzdialenosť od bodu C, ktorého poloha je fixná,
- X_MAX – konštanta, určujúca maximálnu vzdialenosť v dimenzii X,
- Y_MAX – konštanta, určujúca maximálnu vzdialenosť v dimenzii Y,

Nasledujúce implementačné kroky zodpovedajú jednotlivým krokom návrhu:
1) Výpočet aktuálnej pozície robota: Výpočet je realizovaný pomocou Pythagorovej vety. Na základe tejto vety boli odvodené nasledujúce 2 rovnice:

$$APP_X^2 + APP_Y^2 = DIST_B^2 \tag{1}$$

$$(X_{MAX} - APP_X)^2 + APP_Y^2 = DIST_C^2 \tag{2}$$

Po úprave vyššie uvedených rovníc je možné vyjadriť hodnoty APP_X a APP_Y, ktoré reprezentujú súradnice X a Y. Nakoľko sa pohybujeme po jednotlivých políčkach, je potrebné získané hodnoty súradníc previesť na hodnotu, ktorá je vyjadrená v políčkach mapy. Toto vyjadrenie zabezpečíme vydelením získanej hodnoty súradnice hodnotou konštanty BOX_SIZE.

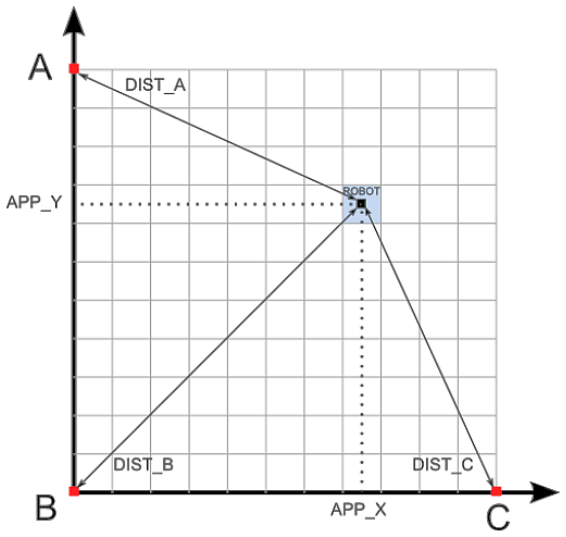
$$APP_X = \frac{X_{MAX}^2 - DIST_C^2 + DIST_B^2}{2.X_{MAX}} \tag{3}$$

$$X = \frac{APP_X}{BOX_{SIZE}} \tag{4}$$

$$APP_Y = \frac{Y_{MAX}^2 - DIST_A^2 + DIST_B^2}{2.Y_{MAX}} \tag{5}$$

$$Y = \frac{APP_Y}{BOX_{SIZE}} \tag{6}$$

Samotný výpočet pozície robota je ilustrovaný pomocou obrázku 4.



Obrázok 4. Výpočet pozície robota.

2)Výpočet vektora prekážok: je realizovaný pomocou porovnania vstupných hodnôt z jednotlivých senzorov, ktoré sa nachádzajú na všetkých štyroch stranách robota s referenčnou hodnotou DIST_MAX, ktorá určuje povolenú vzdialenosť robota od steny. V prípade, ak je získaná hodnota zo senzora menšia, než referenčná hodnota, tak v danom smere sa nenachádza žiadna prekážka a políčko je voľné. V opačnom prípade sa nachádza v danom smere prekážka a pohyb týmto smerom nie je možný.

3) Výpočet cesty v bludisku: využíva algoritmus pre hľadanie cesty „Ľavou rukou po stene“. Algoritmus je implementovaný tak, že robot sa pri každom pohybe snaží prejsť na políčko, ktoré sa nachádza na ľavej strane od jeho pozície. Tento princíp je realizovaný tak, že implementácia pohybu obsahuje štyri smery pohybu, ktoré robot môže absolvovať v nasledujúcom poradí cyklicky: východ ->sever ->západ ->juh ->východ ->... Táto cyklická závislosť pohybu zabezpečí, že robot sa vždy bude „držať ľavou rukou pri stene“. Ak je políčko naľavo voľné, presunie sa naň. Pokiaľ sa však na danom políčku nachádza prekážka, otočí sa smerom doprava a opäť sa pokúša prejsť na políčko, ktoré sa nachádza na ľavej strane od jeho pozície. Robot sa otáča do pravej strany dovtedy, kým

Algoritmus 1: Pseudokód hľadania cesty.

```
1: smer = VÝCHOD;
2: while nie som v cieli do
3:     smer = smer - 1 ; // otočenie doľava
4:     while cieľ pohybu == STENA do
5:         smer = smer +1 ; // otočenie doprava
6:     end while
7:     vykonaj pohyb (smer) ; // robot sa presunie na novú pozícu vo vypočítanom smere podľa hodnoty smer
8: end while
```

nie je políčko naľavo od jeho pozície voľné. Tento proces sa cyklicky opakuje, pokým sa robot nedostane na cieľovú súradnicu políčka. Princíp výpočtu cesty je demonštrovaný algoritmom 1.

5. Experimenty a overenie funkčnosti

Funkčnosť implementácie referenčného modelu bola overená prostredníctvom jednoduchého testovacieho prostredia implementovaného v jazyku C. Vstupné hodnoty boli nasledujúce:

- Počiatočná pozícia,
- Cieľova pozícia,
- Mapa bludiska,

V rámci overovania sa v každom kroku od počiatočnej pozície, až do cieľovej pozície priechodu bludiskom sledovala aktuálna pozícia robota a smer, ktorým sa robot pohol. Tieto hodnoty sa následne porovnávali s očakávanými hodnotami. Počiatočné a cieľové pozície boli zvolené tak, aby počet potrebných krokov, ktoré robot musí vykonať v rámci priechodu každého bludiska bol minimálne 5. V rámci overovania boli využité bludiská o rôznych rozmeroch (hodnoty sú uvedené v políčkach). Konkrétne údaje o vykonaných experimentoch sú uvedené v tabuľke 1.

Rozmer bludiska	Priemerný počet vykonaných krokov	Počet vykonaných experimentov
3 x 6	5	2
5 x 8	9	10
8 x 8	21	20
10 x 12	38	10
16 x 16	99	10
Počet experimentov spolu		**52**

Tabuľka 1. Tabuľka s experimentami.

Spolu bolo vykonaných päťdesiatdva experimentov, pomocou ktorých bola overená funkčnosť referenčného modelu.

Experimenty na overenie funkčnosti verifikačného prostredia nie je možné v dobe publikovania tohto článku vykonať nakoľko komunikácia medzi jednotlivými komponetami implementovaného verifikačného prostredia nie je odladená a úplne funkčná.

6. Záver

[Zhodnotenie práce] Cieľom práce bolo popísať návrh a implementáciu verifikačného prostredia pomocou metodiky UVM pre riadiacu jednotku robotického systému. Verifikačné prostredie bolo kompletne implementované za účelom odstránenia funkčných chýb z implementácie riadiacej jednotky robota a vyselektovania množiny vstupných hodnôt, ktoré dosiahli úplné pokrytie kódu. V čase publikovania tohto príspevku nebola implementovaná komunikácia medzi jednotlivými komponentami verifikačného prostredia a komunikácia medzi verifikačným prostredím a virtuálnym prostredím, ktoré generuje a zasiela potrebné informácie na vstup verifikačného prostredia. Z tohto dôvodu nie je možné uviesť konkrétne hodnoty, ktoré by informovali o dosiahnutom pokrytí kódu.

[Prínos práce] Princíp využitia funkčnej verifikácie v oblasti odolnosti systémov proti poruchám je možné považovať za inovatívny, nakoľko podobný prístup nebol zatiaľ nikde publikovaný. Je dôležité poznamenať, že tento prístup bol navrhnutý generický tak, aby bolo možné verifikovať akýkoľvek číslicový obvod a pripraviť ho na následné testovanie odolnosti proti poruchám. V rámci práce bol tento princíp demonštrovaný na príklade riadiacej jednotky robota.

[Budúca práca] V rámci budúcej práce bude dokončená implementácia komunikácie medzi virtuálnym prostredím a verifikačným prostredím. Na základe tejto komunikácie bude možné zasielať potrebné hodnoty na vstup verifikačného prostredia (najmä informácie zo senzorov robota). Vďaka tomu bude možné vykonať verifikáciu za účelom odladenia riadiacej jednotky a vyselektovania vstupných hodnôt, ktoré budú dosahovať úplné pokrytie. Tým pádom budú pripravené všetky podklady k realizácii druhej fázy procesu zabezpečenia odolnosti systémov proti poruchám, ktorý sa bude realizovať v rámci výskumnej činnosti na ÚPSY.

Literatúra

[1] Jakub Podivínský, Ondřej Čekan, Marcela Šimková, and Zdeněk Kotásek. The evaluation platform for testing fault-tolerance methodologies

in electro-mechanical applications. In *17th Euromicro Conference on Digital Systems Design*, pages 312–319. IEEE Computer Society, 2014.

[2] Podivínský Jakub. VHDL návrh řídící jednotky robota určeného pro samočinný pohyb v bludišti, 2013. Diplomová práca. FIT VUT.

[3] J. Hlavička. *Číslicové systémy odolné proti poruchám*. ČVUT, Praha, 1992.

[4] IEEE standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language. *IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009)*, pages 1–1315, Feb 2013.

[5] UVM Cookbook, 2012. https://verificationacademy.com/cookbook/uvm.