GENERÁTOR HLÍDACÍCH OBVODŮ PRO KOMUNIKAČNÍ PROTOKOLY XILINX FPGA

Martin Straka
Informační technologie, 1. ročník, prezenční studium
Školitel: Doc. Ing. Zdeněk Kotásek, CSc.

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, Brno 612 66

strakam@fit.vutbr.cz

Abstrakt. Článek se zabývá návrhem nové metodologie založené na automatizovaném vytváření hlídacích obvodů pro testování správnosti komunikačních protokolů a tvoří první krok a nejnižší vrstvu našich cílů pro vývoj systémů odolných proti poruchám na bázi FPGA. Pro automatizované vytváření hlídacích obvodů je definován formální jazyk pro popis chováni protokolu a navržen nástroj pro generování hlídacího obvodu v jazyce VHDL na základě tohoto popisu. Evaluace nové metodologie je provedena na komunikačním protokolu LocalLink vyvinutý firmou XILINX. V závěru jsou zhodnoceny vlastnosti nově prezentované metodiky a možnosti jejího využití v disertační práci.

Klíčová slova. systémy odolné proti poruchám, komunikační protokol, hlídací obvod, FPGA, VHDL.

1 Úvod

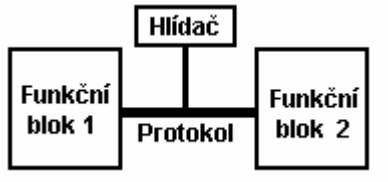
Odolnost proti poruchám je významnou metrikou pro číslicové obvody v mnoha odvětvích od automobilismu, přes síťové aplikace, po vesmírný výzkum. Běžnou technikou pro zlepšování spolehlivosti systému je replikace funkčních jednotek a vyhodnocování paralelně zpracovávaných dat. Tento přístup však přináší vysoké nároky na zdroje číslicového obvodu a zvýšenou spotřebu energie. Z hlediska vývoje zařízení odolného proti poruchám dále znamená vyšší nároky na návrh zařízení a složitější testování obvodu. Proto je vhodné hledat nové metody pro testování obvodu a metody pro návrh systémů odolných proti poruchám [1].

Nové možnosti v oblasti systémů odolných proti poruchám nabízejí programovatelná hradlová pole FPGA. Funkci těchto obvodů lze měnit nahráním nové konfigurace vnitřní struktury a u vyspělých FPGA technologií je možné provést tuto změnu pouze nad částí obvodu (parciální rekonfigurace) [2]. S využitím těchto technik lze implementovat pokročilé diagnostické postupy, které detekují chyby při zpracováni dat a vyhodnotí, zda jde o chyby trvalé či náhodné a v závislosti na výsledku provedou rekonfiguraci části obvodu pro zajištění nápravy [3]. Mnoho článků prezentuje mechanismy a principy testování s vyhrazením jistého počtu buněk pro účely rekonfigurace částí, kde byla detekována porucha [3]. Jiné techniky pro tvorbu systém odolných vůči poruchám využívají replikaci funkčních jednotek – tří modulová redundance. Další články se zabývají zabezpečením propojovací sítě v FPGA a detekcí poruch funkčních částí číslicového obvodu [5].

Cílem našeho výzkumu je navrhnout kompletní metodologii pro tvorbu systémů odolných proti poruchám v FPGA. Tato metodologie musí respektovat současné trendy při implementaci číslicových obvodů jako jsou opakované využití IP jader (Intellectual Property cores) a respektovat aktuální situaci, kdy mnoho bloků ve formě IP jader nemá vestavěné testovací prostředky (Built-In Self Test, BIST). První fáze naších cílů je popsána v tomto článku.

2 Motivace

Jedna z částí naší metodiky je zabezpečení síťových zařízení vyvinutých nad rodinou karet COMBO [4] v rámci výzkumných aktivit na Vysokém učení technickém v Brně. Příkladem těchto zařízení jsou systémy pro monitorování síťového provozu, systémy pro detekci nežádoucího síťového provozu a jiné [5]. Zmíněné zařízení nebyla z počátku vybavena žádnými diagnostickými a testovacími prvky a proto byl náš postup rozdělen do tří etap. První je návrh a realizace systému pro automatizovanou tvorbu hlídacích obvodů pro testování komunikačních protokolů [6]. Druhou etapou bude návrh prostředků průběžné a periodické diagnostiky pro tyto zařízení a třetí etapou vývoj strategií pro automatické opravy funkce systému po výskytu hardwarové poruchy. V této práci jsou prezentovány výsledky první etapy naší činnosti, tedy návrh a realizace systému pro automatizovanou tvorbu hlídacích obvodů. Tento systém zabezpečení obvodu vychází ze skutečnosti, že všechna IP jádra mají definované rozhraní a určitý protokol na tomto rozhraní. Tuto skutečnost demonstruje obrázek 1. Porušení tohoto protokolu značí, že došlo k poruše v hardwaru a na základě detekce této poruchy je možné vyvolat částečnou rekonfiguraci obvodu.



Obrázek 1: Situace při zabezpečení komunikace dvou funkčních bloků.

3 Způsob řešení

Chyby, které se mohou vyskytnout v číslicových obvodech na sběrnici komunikačního protokolu je možné popsat různými způsoby. Typicky se používají formální modely popsané pomocí gramatik, stavových automatů nebo formálních jazyků [2]. Při kontrole protokolu je nutné hlídat nejen kombinace přenášených signálů, ale je nutné sledovat i jejich posloupnost. Hlídací komponenta musí tedy vykazovat sekvenční chování, které je možné popsat pomocí konečného automatu. Pro definici hlídacích obvodů jsme vytvořili jazyk, kterým je možné popsat správné nebo chybové stavy v komunikačním protokolu a umožnit tak automatické vytvoření hlídacího obvodu. Stavy protokolu popsané v tomto jazyce jsou následně analyzovány vytvořeným překladačem, který vytvoří výsledný popis hlídacího obvodu v jazyce VHDL. Takto získaný obvod disponuje rozhraním hlídaného protokolu a vnitřní architektura odpovídá konečnému automatu sestaveného na základě popisu v definičním jazyce. Následně lze provést syntézu do cílového FPGA.

Hlavní výhodou tohoto přístupu oproti přímé implementaci je možnost vygenerování výsledného obvodu na základě jednoduchého popisu bez účasti zkušeného VHDL návrháře.

4 Definice jazyka

Popis jazyka se skládá ze dvou částí. V první části jsou definovány vstupní symboly, které jednoznačně určují přechody mezi stavy automatu. Každý vstupní symbol je tedy definován jako podmínka nad výstupními signály hlídané komponenty. V druhé části je definována přechodová funkce automatu. Pro každý stav a terminální symbol je definován přechod do následujícího stavu. Počáteční stav automatu je označen jako *S0* a chyba protokolu je detekována přechodem do stavu *Serr*. Pokud není automat plně definovaný, setrvává v případě nepokryté kombinace vstupních signálů ve stejném stavu.

Vstupní symboly automatu jsou definovány jako konjunkce nebo disjunkce podmínek oddělených klíčovými slovy *OR* nebo *AND* a uzavřena do kulatých závorek. Každá podmínka obsahuje jedno porovnání (<, >, <=, >=, ==, <>) mezi signálem a celočíselnou konstantou.

$$cond0 \rightarrow Name\_signal == 1$$
$$cond1 \rightarrow Name\_signal >= 0$$

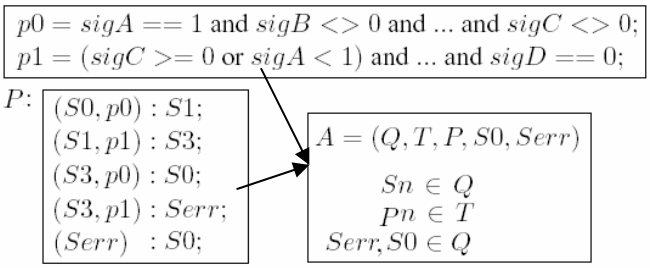
Takto definovaný výraz je vyhodnocen jako nový vstupní znak automatu. Syntaxe definice vstupního symbolu je zachycena následujícím příkladem:

$$p0 = cond0 \text{ and } cond1 \text{ or } (cond2 \text{ or } cond3) \text{ and } ... \text{ and } condn;$$

Chování konečného automatu je popsáno pomocí přechodové funkce, která je definována množinou přechodů ve tvaru stávající stav, vstupní symbol, nový stav nebo aktuální stav a nový stav :

$$(S0, p0) : S1;$$
$$(Serr) : S0;$$

Na základě výše uvedených pravidel je možné určit množinu stavů a vstupních symbolů a následně zkonstruovat konečný automat „A“. Příklad popisu jednoduchého hlídacího obvodu v definičním jazyce je uveden na obrázku 2.



Obrázek 2: Příklad popisu jednoduchého hlídacího obvodu.

Jazyk pro popis hlídacího obvodu je schopen popsat libovolný automat, který kontroluje signály protokolu. Je tedy možná hlídat protokoly, jejichž chování lze popsat pomocí regulárního jazyka. Zvolený jazyk je schopen kontrolovat nejen chyby protokolu, ale je schopen popsat i některé poruchy na úrovni funkce. Například je možné sledovat, zda přenášená hodnoty odpovídají předpokládanému rozsahu.

5 Konstrukce překladače

Generátor je program pro automatizované vytváření hlídacích obvodů na základě popisu ve formě definovaného jazyka. Proces generování se skládá ze dvou fází a je zachycen na obrázku 3. Nejprve je generátorem provedena analýza vstupního souboru a na základě formálního popisu je zkonstruován

deterministický konečný automat. Na fázi analýzy poté navazuje fáze mapování podmínek a automatu do obvodové realizace. Výsledkem mapování je komponenta popsaná v jazyce VHDL, která má rozhraní hlídané komponenty a výstupní chybový signál *Error*.

Analýza vstupního souboru začíná u vstupních symbolů a podmínek jim přiřazených. Při analýze je postupně vytvářena množina všech vstupních symbolů a je prováděna syntaktická analýza podmíněných výrazů. Ke každému podmíněnému výrazu je vytvořen syntaktický strom, který je následně využit ve fázi mapování na popis obvodu v jazyce VHDL.

Obrázek 3: Fáze generování hlídacího obvodu.

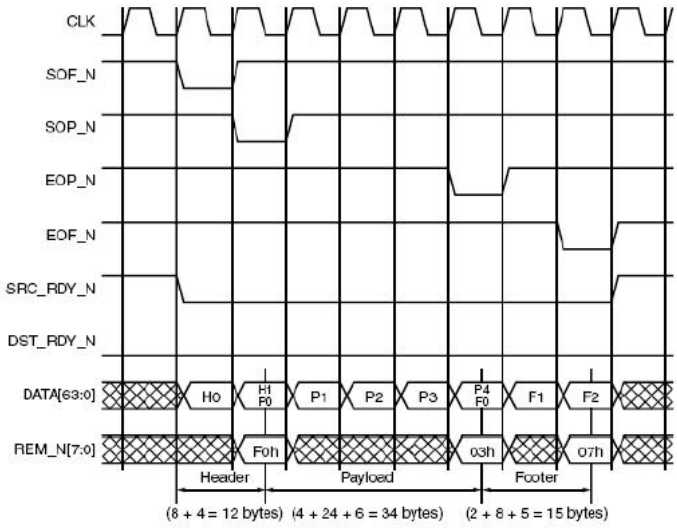
Po úspěšném sestavení množiny vstupních symbolů *T* jsou čteny jednotlivé přechody s cílem sestavit konečný automat. Z definice přechodu je vždy vybrán zdrojový a cílový stav. Oba jsou automaticky přidány do množiny všech stavů automatu *Q*. Současně se pro každý přechod automatu zkontroluje, zda je použit dříve definovaný vstupní symbol množiny T. Pokud vstupní symbol nebyl definován, je hlášena chyba a běh generátoru se ukončí. Analýzou stavů automatu je ukončeno čtení vstupního souboru. Známy jsou množina vstupních symbolů *T,* množina stavů automat *Q* i množina všech přechodů P. Počáteční stav je označen podle definičního jazyka jako *S0* a do množiny koncových stavů je přidán pouze stav *Serr,* který signalizuje výskyt poruchy. Po analýze vstupního souboru je tedy zkonstruovaný automat *A = (Q,T,P,S0,Serr)*.

Fáze mapování začíná vytvořením rozhraní hlídací komponenty. Pro jednotlivé vstupní symboly jsou ze všech podmínek vybírány použité signály a z nich je vytvořeno rozhraní komponenty. Po vytvoření entity se generuje architektura komponenty. Nejprve jsou mapovány podmínky na VHDL procesy. Vstupem procesu jsou signály rozhraní, výstupem je jediný signál, jehož název odpovídá vstupnímu symbolu definovanému v popisu jazyka pro danou podmínku. Obsah procesu je generován na základě syntaktického stromu vytvořeného v první části analýzy. Optimalizace je ponechána na syntezátoru, který v této oblasti poskytuje velmi dobré výsledky. Samotné mapování konečného automatu je realizováno prostřednictvím dvou procesů. Jeden proces pracuje jako registr, který uchovává aktuální stav a druhý popisuje kombinační logiku pro přechod do následujícího stavu.

6 Experimentální výsledky

Navržený systém pro generování hlídacích obvodu byl otestován na protokolu LocalLink (dále jen LL) [7], který byl vyvinut firmou *Xilinx* pro propojování funkčních komponent v FPGA. Pro daný protokol je dostupných mnoho bloků ve formátu IP jader, které není možné modifikovat a nemají integrované BIST nebo jiné offline či online testy, a proto je vhodné ověřit a zabezpečit jejich chováni vygenerovaným obvodem.

Protokol LL je synchronní point-to-point protokol s generickou datovou šířkou rozhraní, určeny pro přenos dat ve formě rámců. Na obrázku 4 je vidět časový diagram protokolu. Dalšími jeho vlastnostmi jsou oboustranná kontrolu toku dat, efektivní využití dostupné šířky pásma a volitelné kontrola parity, kterou je u některých IP jader (u kterých je tato funkce implementována) možno využit k zabezpečeni dat. Detailní specifikace protokolu je v [7].



Obrázek 4: Časový diagram protokolu LocalLink.

Pro evaluaci byly zvoleny dvě možné úrovně diagnostiky komunikace a byla sledována náročnost vygenerovaného obvodu na zdroje FPGA. Nejjednodušší zabezpečení spočívalo ve sledování kombinací řídicích signálů protokolu a detekci správných stavů odpovídajících specifikaci protokolu – hlídání sekvencí řídicích signálů.
Následující pravidla jsou platná při aktivních signálech SRC_RDY_ N a DSC_RDY_N:

1. Každý rámec musí začínat signálem SOF_N a žádný jiný signál nesmí být aktivní.
2. V každém rámci musí být na začátku hlavička, proto je-li aktivní SOP_N, nesmí být aktivní žádný jiný.
3. Každý rámec musí obsahovat patičku, proto se signálem EOP_N nesmí být aktivní žádný jiný signál.
4. Každý rámec musí být ukončen signálem EOF_N a žádný jiný signál nesmí být aktivní.
5. Jsou li přenášena data, nesmí být aktivní žádný signál kromě SRC_RDY_ N a DSC_RDY_N.

Odpovídající zápis v definičním jazyce ukazují následující pravidla:

$$
\begin{aligned}
p0 &= SRC\_RDY\_N == 0 \; and \; DST\_RDY\_N == 0 \; and \; SOF\_N == 0 \\
&\quad and \; SO\_N == 1 \; and \; EOP\_N == 1 \; and \; EOF\_N == 1 \\
p1 &= SRC\_RDY\_N == 0 \; and \; DST\_RDY\_N == 0 \; and \; SOF\_N == 1 \\
&\quad and \; SO\_N == 0 \; and \; EOP\_N == 1 \; and \; EOF\_N == 1 \\
p2 &= SRC\_RDY\_N == 0 \; and \; DST\_RDY\_N == 0 \; and \; SOF\_N == 1 \\
&\quad and \; SO\_N == 1 \; and \; EOP\_N == 0 \; and \; EOF\_N == 1 \\
p3 &= SRC\_RDY\_N == 0 \; and \; DST\_RDY\_N == 0 \; and \; SOF\_N == 1 \\
&\quad and \; SO\_N == 1 \; and \; EOP\_N == 1 \; and \; EOF\_N == 0 \\
p4 &= SRC\_RDY\_N == 0 \; and \; DST\_RDY\_N == 0 \; and \; SOF\_N == 1 \\
&\quad and \; SO\_N == 1 \; and \; EOP\_N == 1 \; and \; EOF\_N == 1 \\
p5 &= SRC\_RDY\_N == 0 \; or \; DST\_RDY\_N == 0
\end{aligned}
$$

$$
\begin{aligned}
&(S0, p5) : S0; (S0, p0) : S1; \\
&(S1, p5) : S1; (S1, p1) : S2; (S1, p4) : S1; \\
&(S2, p5) : S2; (S2, p2) : S3; (S2, p4) : S2; \\
&(S3, p5) : S3; (S3, p3) : S0; (S3, p4) : S3; \\
&A = (Q, T, P, S0, Serr)
\end{aligned}
$$

Druhou částí procesu kontroly je monitorování dat přenášených protokolem a kontrola splnění definovaných podmínek a pravidel pro obsah datové části. Pro příklad, první přenášený bajt rámce musí obsahovat hodnotu 0xAB (Start-of-Frame Delimiter) nebo devátý bajt rámce musí mít hodnotu

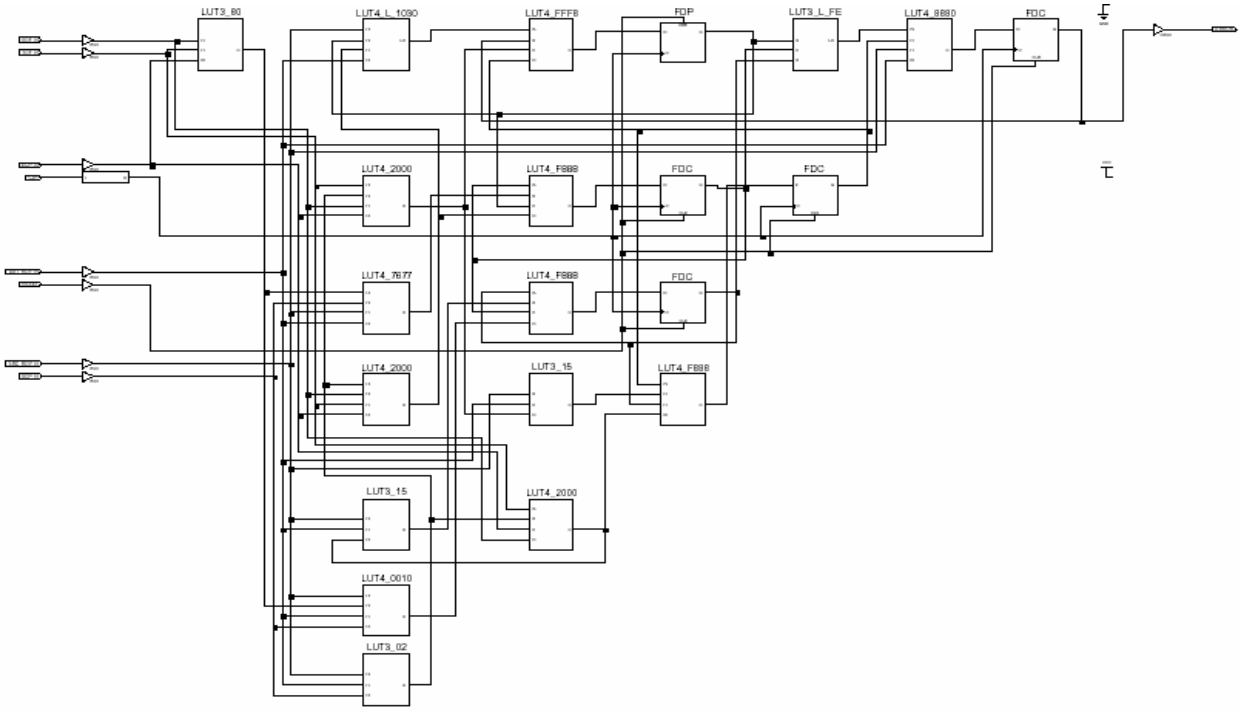
nižší než 124 (při šířce přenášeného slova 4 bajty). Odpovídající zápis v definičním jazyce ukazují následující pravidla:

$$p0 = DATA_0[7\ downto\ 0] == 0xAB;$$
$$p1 = DATA_2[7\ downto\ 0] < 124;$$

Výše uvedená pravidla byla zpracována generátorem obvodu a výsledná implementace ve VHDL byla syntetizována pro FPGA Virtex II Pro, které je hlavním výpočetním prvkem na rodině karet COMBO. Správná funkce všech vygenerovaných obvodů byla ověřena jak v simulacích tak při testování v hardwaru. Výsledky syntézy pro výše definované tři stupně zabezpečení jsou shrnuty v tabulce 1, schéma obvodu demonstruje obrázek 5. Využití zdrojů FPGA je různé v závislosti na počtu definovaných pravidel a typu pravidel. V tabulce v levé sloupci je použitý typ pravidel, v pravém sloupci je počet využitých *Sliců* v FPGA.

Tabulka 1:Výsledky syntézy.

Typ pravidel	Zdroje v FPGA (slices)
Kombinace a sekvence signálů	9
Kontrola dat	15



Obrázek 5: Vygenerovaný hlídací obvod pro zabezpečení LocalLink.

Výsledky ukazují, že generované obvody jsou velmi málo náročné na zdroje FPGA a režie při použití zabezpečovacích obvodů je tedy minimální. Přitom je možné monitorovat libovolné chyby protokolu a zabezpečit správné chování komponenty. Frekvence všech vygenerovaných obvodů přesahovala 300 MHz a nijak nelimitovala výkon zabezpečených IP jader.

7 Předpokládaný obsah disertační práce

Doposud jsem se zabýval aplikací on-line testů, jejímž výsledkem je vytvoření hlídacího obvodu pro zabezpečení komunikace mezi dvěma funkčními bloky v FPGA, které komunikují přes specifický komunikační protokol. Předpokládaný směr řešení disertační práce je zabývat se využitím off-line testů při návrhu rekonfigurovatelných systémů v FPGA. Například pro síťové aplikace nad kartou COMBO6 [4] najít metodologii, která by rozpoznala či předpověděla prázdný interval v komunikaci, a v tomto intervalu by se naplánovaly a spustily sekvence off-line testů. Při výskytu chyby by se mohla následně vyvolala částečná rekonfigurace poškozeného bloku.

8 Závěr

Tato práce prezentuje novou techniku pro automatizovaný návrh obvodů pro testování komunikačních protokolů. V úvodu je rozebrán aktuální stav v oblasti návrhu systémů odolných proti poruchám na FPGA. V dalších kapitolách je představena nová metodologie, využívající formálního jazyka pro popis možných chyb protokolu a automatický nástroj pro generování popisu obvodu v jazyce VHDL. Výsledky získané zabezpečením protokolu LL ukazují na hlavní výhody této metody - nízkých nároků na zdroje hradlového pole i při detekci všech možných chyb protokolu. Výhodou jsou také maximální dosahované frekvence generovaných obvodů. Ty při použití cílové technologie Virtex II Pro ve všech případech přesahovaly 300 MHz a nijak nelimitovaly zabezpečované obvody.

Poděkování

Výzkum je podporován projektem financovaného Grantovou Agendou České Republiky pod číslem 102/05/H050 „Integrovaný přístup k výchově studentů DSP v oblasti paralelních a distribuovaných systémů“ a projektu č. MSM 0021630528 – „Výzkum informačních technologií z hlediska bezpečnosti“. Rád bych zde taky poděkoval Jiřímu Tobolovi za cenné podklady a informace tykající se návrhu generátoru hlídacích obvodů.

Literatura

[1] P. Kubalik, P. Fiser, and H. Kubatova, Fault tolerant system design method based on self-checking circuits, in *Proceedings of the 12th IEEE International Symposium on On-Line Testing (IOLTS06)*. Corno, Italy: IEEE Computer Society, 2006, pp. 185-186.

[2] S. Mitra, W.-J. Huang, N. R. Saxena, S.-Y. Yu, and E. J. Mc-Cluskey, Reconfigurable architecture for autonomous self-repairs, in *IEEE Design and Test of Computers*, vol. 21, no. 3, pp. 228-240, 2004.

[3] V. Lakamraju and R. Tessier, Tolerating operational faults in cluster-based FPGAs, in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays (FPGA00)*. New York, NY, USA: ACM Press, 2000, pp. 187-194.

[4] J. Novotny, O. Fucik, and R. Kokotek. Schematics and PCB of COMBO6 card. Technical Report 14/2002, CESNET, 2002. available at http://www.cesnet.cz/doc/techzpravy/2002/combo6/combo6.pdf.

[5] W. Xu, R. Ramanarayanan, and R. Tessier, Adaptive fault recovery for networked reconfigurable systems, in *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM03)*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 143-154.

[6] R. Lai, A survey of communication protocol testing, *Journal of Systems and Software*, vol. 62, no. 1, pp. 21-46, 2002.

[7] Xilinx Inc. 2100 Logic Drive. *LocalLink Interface Specification*. San Jose, September 2006.