Checker for Communication Protocol between IP Cores Based on FPGA

Martin Straka and Zdenek Kotasek

Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, Czech Republic
{strakam,kotasek}@fit.vutbr.cz

Abstract. In the paper, the principles of a unit design which can be used for on-line communication protocol checking is presented. It is shown how the checker can be used to check the communication between IP cores implemented in FPGA. The communication must be precisely defined for this purpose, a formal approach was developed which allows to describe ambiguously the conditions which must be satisfied during the communication. From the description, the checker description in VHDL is generated (a compiler was developed for this purpose) and implemented into FPGA. The checker watches the communication and detects such states which do not satisfy protocol definitions. If such a situation appears, it is indicated that hardware implementation does not work properly. The methodology was verified on LocalLink communication protocol developed by Xilinx, Virtex 2 Pro FPGA was used for the implementation. Future research will be directed towards the development of fault tolerant systems design methodology in which the presented approach can be possibly used.

1 Introduction

For fault tolerant systems, different units which are able to detect faults must be developed. They can possibly check the function of units integrated into the design or communication protocols between them. The research described in this paper concentrates on the design of the checker for communication protocol testing.

Method for generating checker circuits from SEREs (Sequential Extended Regular Expressions) is described in [1]. Such sequences form the core of increasingly-used ABV (Assertion-Based Verification) languages. A checker generator capable of transforming assertions into efficient circuits allows the adoption of ABV in hardware emulation.

In [2], an original method to synthesize monitors from declarative specifications written in the PSL (Property Specification Language) was developed which enables standard. Monitors observe sequences of values on their input signals, and check their conformance to a specified temporal expression.

In [3], design methodology for fault tolerant systems implemented on SoC (System on Chip) is presented. In the paper, as an example, a complex fault

tolerant finite state machine has been mapped on the FPGA (Field Programable Gates Area) contained in the SoC. The fault identification has been obtained by using a checker permitting the identification of class of faults. When a fault is detected, an interrupt for the microcontroller is generated and the interrupt handling routine partially reprograms the FPGA to override the part of memory configuring the faulty block.

The development of test cases based on a formal model is an important issue for software and hardware testing including conformance testing of communication protocols and other reactive systems. A number of methods are known for the development of a test suite based on a specification given in the form of a FSM (Finite State Machine). In FSM-based on-line testing, one usually assumes that not only the specification, but also an implementation can be modeled as a deterministic FSM [4].

Protocols have grown larger and more complex with the advent of computer and communication technologies [5]. As a result, the task of conformance testing of protocol implementation has also become more complex. The study of DFT (Design For Testability) is a research area in which researchers investigate design principles that will help to overcome the ever increasing complexity of testing distributed systems [6]. Testability metrics are essential for evaluating and comparing designs. In [7], they introduce a new metric for testability of communication protocols, based on the detection probability of a default. They present two approaches for improved testing of a protocol implementation once those faults that are difficult to detect are identified.

2 Motivation for Research

Very often it is reported that FPGA based designs are constructed as fault tolerant designs with the possibility of recovering from errors by means of reconfiguration procedures. In our opinion, testing proper function of communication protocol can increase significantly the diagnostic quality of the design. It was decided that the checker will operate on different levels of detecting communication protocol faults:

1. a checker detecting an incorrect combination of control signals.
2. a checker detecting a correct combination of control signals.
3. a checker detecting a correct sequence of control signals.
4. a checker controlling data which can be part of frame.

The complexity of the checker will be different based on the type of communication protocol fault supposed to be detected by the checker. The complexity of the checker will influence the area required on the chip and communication speed. As an important aspect of the methodology we saw that the alternative of automated design of the checker should be available to a designer. For this purpose, we felt the need for a formal language by means of which the checker will be described together with the need for core generator to compile checker description into VHDL code. These ideas are presented later in the paper.

The paper is organized as follows. In the next section, formal definitions needed for the methodology presented in the paper are presented. In section 4, basic principles of the checker design methodology are described while the basic ideas of methodology implementation for LocalLink (LL) are explained in section 5. Conclusions and plans for future research are summarized in section 6.

3 Formal Definitions

Usually, to describe errors in communication protocols, formal models such as grammars, FSMs, or formal languages are used. As a result of our research a language was developed which allows to describe possible failures in communication protocol. The description is then used as an input to automatic generator which develops checker description in VHDL language. The main advantage of this approach is such that based on the language the checker can be generated automatically without the intervention of experienced designer.

When a communication protocol is checked, then not only the combinations of signals must be monitored but also their sequences. The checker behavior must therefore have features of sequential behavior which can be described by means of FSM. The definition of language for communication protocol errors detection therefore arises from the formal description of FSM.

Definition 3.1 *A deterministic Finite State Machine is an initialized complete deterministic machine that can be formally defined as a 5-tuple* $A = (Q, T, P, S0, Serr)$*, where* Q *is a finite set of states,* $S0$ *is the initial state and* $S0 \in Q$*,* T *is a finite set of input symbols,* P *is a next state (or transition) function:* $P : Q \times T \rightarrow Q$ *and* $Serr$ *is the finite state and* $Serr \in Q$*. Furthermore* $Q \cap T = \emptyset$*.*

Definition 3.2 *A condition is formally defined as a* $C(i) = Sig \times Oper \times Int$*, where* Sig *is a name of control signal,* $Oper \in (<, >, <=, =, ==, <>)$ *is a comparison operator between controlled signal and* $Int \in N$ *numeric constant.* $i \in 1, 2, 3, ...$

Definition 3.3 *An input automata symbols are defined as conjunction of conditions, formally defined as a* $p(n) = \bigwedge_{i=1}^{X} C(i)$*, where* $n \in 1, 2, 3, ..., N$ *and* $p(n) \in T$ *and* $X = \sum(control\ signals\ in\ checking\ protocol)$

Definition 3.4 *A transition function which is represented by a set of transitions in the form and is formally defined as a* $P(n) : Q \times T \rightarrow Q$*, where* $n \in 1, 2, 3, ..., N$

The language for description behavior of communication protocol is represented by the following LL1 grammar:

1. $program \rightarrow @\ conditions\ \#\ states\ \$$
2. $program \rightarrow e$
3. $states \rightarrow (\ state\ states$
4. $states \rightarrow e$
5. $state \rightarrow ID\_state\ \ transition\ )\ :\ ID\_state\ ;$
6. $transition \rightarrow ,\ INPUT\_SYM$
7. $transition \rightarrow e$
8. $conditions \rightarrow INPUT\_SYM\ =\ expression\ \ expressions\ ;\ conditions$
9. $conditions \rightarrow e$
10. $expressions \rightarrow AND\ expression\ \ expressions$
11. $expressions \rightarrow OR\ expression\ \ expressions$
12. $expressions \rightarrow e$
13. $expression \rightarrow SIGNAL\_NAME\ compare$
14. $compare \rightarrow == INT$
15. $compare \rightarrow <> INT$
16. $compare \rightarrow > INT$
17. $compare \rightarrow >= INT$
18. $compare \rightarrow < INT$
19. $compare \rightarrow <= INT$

4 Automated Checker Design

Core generator is a program for automated development of checker structure based on the description provided in formal language. By means of the formal language the conditions of communication protocol are described. The process of generating checker consists of two phases, see Figure 1:

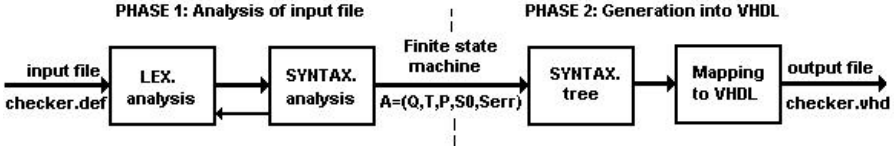
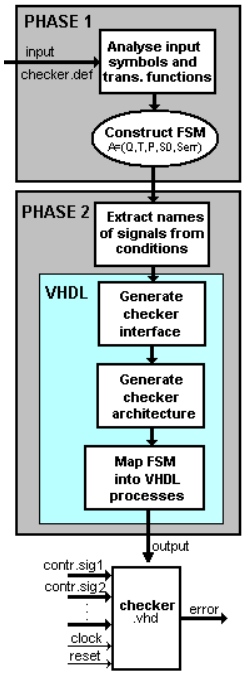


Fig. 1. Phasis of core generator processing

PHASE 1: The input file is analyzed, the conditions which must be satisfied together with transition functions are transformed into FSM description.

PHASE 2: The transitions reflected by FSM description are mapped into VHDL processes.



As the first step of the input file analysis, the symbols of the files are analyzed together with conditions assigned to them. The set containing all input symbols is created and the syntax analysis of conditional statements is performed. For each conditional statement a syntax tree is formed which is then used during mapping the conditions onto the description in VHDL language. As the result of the analysis, an FSM is constructed, $A = (Q, T, P, S0, Serr)$.
The second phase starts with creating the interface of the checker. The names of signals are extracted from transition conditions. The conditions are then mapped onto VHDL processes. The interface signals are the input to the process, the output of the process is the only signal, whose name reflects one of input symbols. The contents of the process is generated from the syntax tree developed in the first phase of the analysis. The mapping of FSM into VHDL is performed by means of two processes. One of them operates as a register in which current state is stored and the second process describes the combinational logic reflecting transition conditions.

5 Evaluation of Methodology

The proposed approach for generating checker structure was tested on LocalLink (LL) communication protocol developed by Xilinx company which is used especially for FPGA components interconnection. The LL protocol has been integrated to many IP Cores.

The LL is based on synchronous point-to-point communication protocol which transfers data in the form of packets. To the LL advantages generic data width of transferred data belongs which is a very important aspect for stream processing applications. Additionally, LL offers upstream and downstream flow control, efficient link bandwidth utilization and optional parity checking. The LL interface contains six control signals, data bus and signals identifying the number of valid bytes available in the last data word. The example of LL communication protocol is shows in Figure 2. Detailed specification of LL protocol is available in [8].

The first, LL protocol specification the following correct signal combinations can be derived, SRC_RDY_N and DST_RDY_N being every active:

1. Every frame must start with SOF_N signal and no other signal is allowed to be active.
2. Each frame must contain a header at the beginning. Thus, if SOP_N is active, no other signal is allowed to be active.
3. Each frame must contain a footer. Thus, if EOP_N is active, no other signal is allowed to be active.

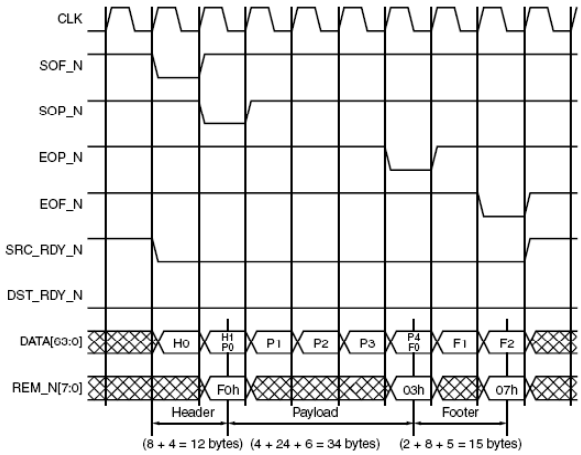


Fig. 2. Local Link Protocol Timing Diagram

4. Each frame must be accomplished with EOF_N signal, no other signal is allowed to be active.
5. If data is transported, no other signal except of SRC_RDY_ N and DSC_RDY_N signals is allowed to be active.
6. Any others activity or next frame.

This list of rules can be easily rewritten based on formal definitions as follows:

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

This approach is limited and can detect only the basic faults caused by forbidden combinations of signals in the protocol interface.

The second type of rules considers the sequences of control signals. For the LL protocol the following transition rules can be applied:

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

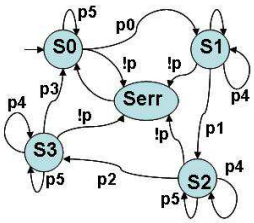


Fig. 3. Finite state machine for the detection of faults on LocalLink

6 Conclusions and Plans for Future Research

For all types of rules checker structure was generated and correct behavior was tested on COMBO6X card with FPGA Virtex2 Pro for network traffic. Synthesis to Virtex2 Pro FPGA was also performed to obtain basic parameters of generated circuit. For all generated circuits, the maximal frequency was higher than 200 MHz and does not affect maximal frequency of IP cores. FPGA logic utilization was different for all types of rules and types of diagnostic levels.

The checker design methodology presented in this paper is the first step towards the development of the methodology for fault tolerant systems design. For the checker design, PSL (Property Specification Language) can be used as well. So far, it was used for verification purposes only, although some indications of its possible use for diagnostic purposes appeared recently. Therefore, the aspects of both methodologies, i. e. that one based on our approach and PSL based approach will be studied and compared first.

The final objective of the FTS (Fault Tolerant System) design methodology will aim at the development of such approaches which will allow short availability and long lifetime parameters to be gained. It will become important in such applications in which high tolerance parameters will be required. It is expected that the methodology will be able to distinguish between permanent and transient errors for which different strategies of fault detection and subsequent reconfiguration must be used.

Acknowledgements

This work was supported by the Research Project No. MSM 0021630528 - Security-Oriented Research in Information Technology and by GACR project No. 102/05/H050 - Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems (Grant Agency of the Czech Republic).

References

1. Boule, M., Zilic, Z.: Efficient automata-based assertion-checker synthesis of seres for hardware emulation. In: 12th Asia and South Pacific Design Automation Conference (ASP-DAC 2007). McGill University, Montreal, Quebec, Canada. (2007).
2. Morin-Allory, K., Borrione, D.: Proven correct monitors from PSL specifications. In: Proceedings of the conference on Design, automation and test in Europe. Leuven, Belgium, (2006) p.1246-1251
3. Pontarelli, S., Cardarilli, G.C., Malvoni, A., Ottavi, M., Re, M., Salsano, A.: System-on-Chip Oriented Fault-Tolerant Sequential Systems Implementation Methodology. In: Proceedings of the 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. (2001).
4. Dorofeeva, R., El-Fakih, K., Maag, S., Cavalli, A., Yevtushenko, N.: Experimental Evaluation of FSM-Based Testing Methods. In: Lecture Computer Science. (2000).
5. Vuong, T. S., Loureiro, A., Chanson, S. T.: A framework for the design for testability of communication protocols. In: 6th International IFIP Workshop on Protocol Test Systems, Pau, France (1993) p.415–438
6. Petrenko, A., Dssouli, R., Koenig, H.: On evaluation of testability of protocol structures. In: 6th International IFIP Workshop on Protocol Test Systems, Pau, France. (1993).
7. Chung, A., Huang, T.: A New Metric for the Testability of Communication Protocols. In: Tnternational Conference on Engineering, Information Technology, e-Business, and Applications - CSITeA02.(2002).
8. Xilinx Inc. 2100 Logic Drive. LocalLink Interface Specification. San Jose, (2006).