CHECKERS DESIGN FOR COMMUNICATION PROTOCOLS BASED ON FPGAS

Martin Straka
Doctoral Degree Programme (2), FIT BUT
E-mail: strakam@fit.vutbr.cz

Supervised by: Zdeněk Kotásek
E-mail: 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 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 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 methodology was verified on LocalLink communication protocol developed by Xilinx, Virtex 2 Pro and Virtex4 FPGAs were used for the implementation.

1. INTRODUCTIONS

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 Programmable Gate Array) 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 service routine partially reprograms the FPGA to override the part of memory configuring the faulty block.

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:

- The check of protocol control signals and their correct combinations.
- The check of correct sequences of control signals and evaluation of transitions between communication protocol states.
- The check of contents of data.

The complexity of the checker will be different based on the type of communication protocol fault supposed to be detected by the checker. 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 and basic principles of the checker design are presented. Basic ideas of methodology implementation for LocalLink (LL) are explained in section 4. Conclusions and plans for future research are summarized in section 5.

3. FORMAL LANGUAGE AND CORE GENERATOR

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 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* $Serr \in Q$. *Furthermore* $Q \cap T = 0$.

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

Definition 3. *An input automata symbols are defined as conjunction or disjunction of conditions, formally defined as a p = Ci (and C_{i+1})*, where p∈ T and i=1,2,..M, where M = ∑(control signals in checking protocol).*

Definition 4. *A transition function which is represented by a set of transitions in the form and is formally defined as a P : Q x T → Q.*

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:

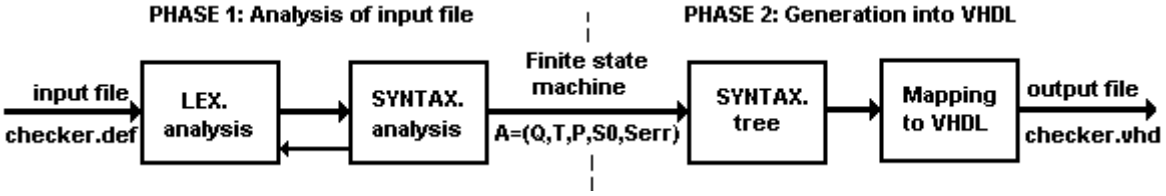

Figure 1: Phasis of core generator processing

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.

4. EVALUATION OF METHODOLOGY AND RESULTS

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 shown in Figure 2. Detailed specification of LL protocol is available in [4].

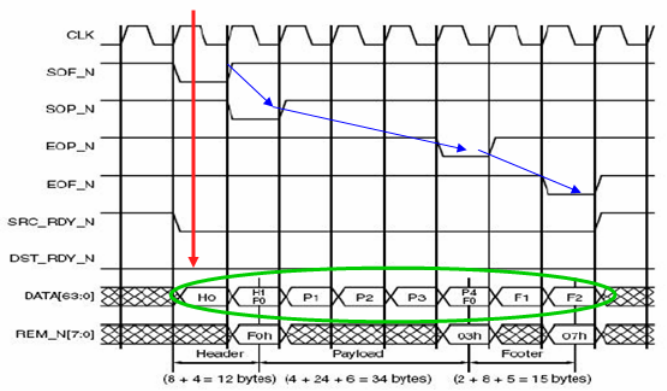

Figure 2: LocalLink Protocol Timing Diagram

The first, LL protocol specification the following correct signal combinations can be derived, SRC_RDY_N and DST_RDY_N being every active. 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. The last part covers data monitoring transported by means of the protocol, checking the conditions rules describing the contents of data. An example: the first transported byte must contain 0xAB (Start-of-Frame Delimiter), the ninth byte must have the value which is lower than 124 (the width of the word is 4 bytes). The principle of the evaluation and results for LL are demonstrated in Figure 3.

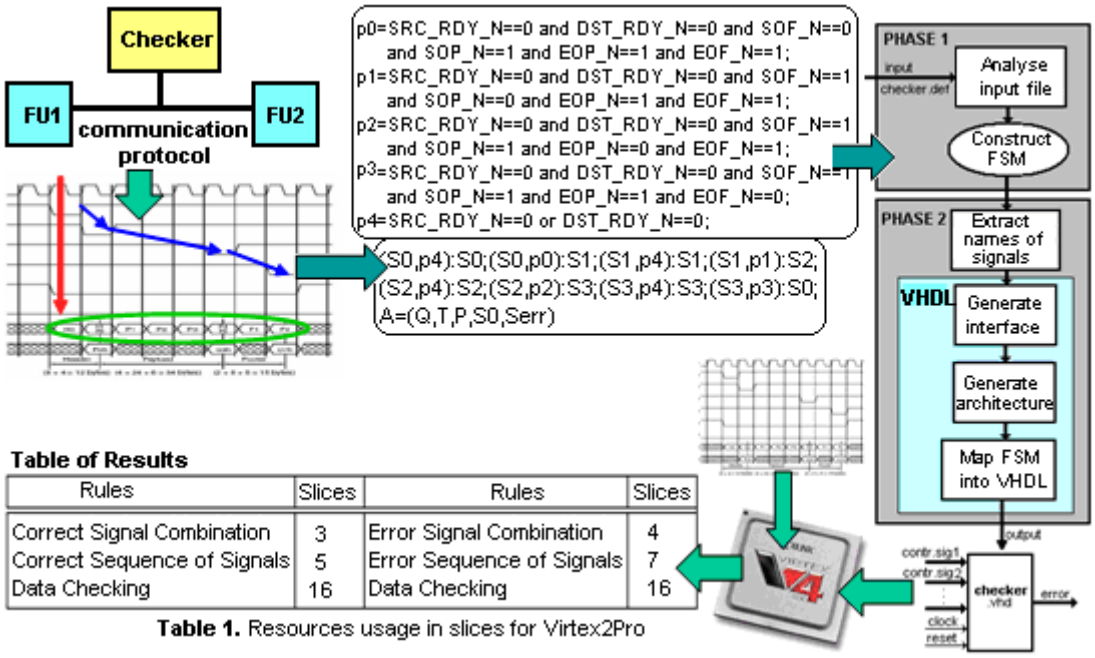

Table of Results

Rules	Slices	Rules	Slices
Correct Signal Combination	3	Error Signal Combination	4
Correct Sequence of Signals	5	Error Sequence of Signals	7
Data Checking	16	Data Checking	16

Table 1. Resources usage in slices for Virtex2Pro

Figure 3: Principle of the methodology for LocalLink.

The types of rules were derived from the LL protocol specification developed for FPGA Virtex4 and Virtex2Pro. For each of three types of rules for which checkers were synthesized, different numbers of slices were involved into the design as the result of the synthesis. The right column of the Table 1 gives the numbers of slices needed to detect error states of the protocol. In both cases, power requirements are very low.

5. CONCLUSION

A methodology was developed which can be used for automatic development of communication protocol checkers. The following tasks had to be solved during the work on the topic: 1. the development of formal tool for the formal description of properties that are supposed to be checked - the properties are derived from protocol definitions and rules. 2. the implementation of the generator which can be then used for compiling the description into VHDL code, 3. the synthesis of the checker into Virtex FPGA, 4. experimenting with all these tools, comparing the results (in terms of checker complexity) gained for different sets of properties. Our approach of generating hardware checkers is different from those based on the utilization of PSL (Property Specification Language). While PSL based methodologies allow to describe formally the properties to be checked and add this description to the HDL source code of the component to be synthesized (and develop the checker together with the component), our methodology can be used in situations where the design is finished and it is still expected that certain properties should be checked. As an example, precisely defined communication protocol can serve for which a checker is needed. In our future research, we have an intension to compare our results with those based on the usage of PSL.

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] Xilinx Inc. 2100 Logic Drive. LocalLink Interface Specification. San Jose, (2006).