Implementing a Self-Checking PROFIBUS Slave

Margrit Reni Krug*
Marcelo Lubaszewski*
José Manuel Martins Ferreira**
Gustavo Ribeiro da Costa Alves**

margrit@inf.ufrgs.br, luba@iee.ufrgs.br
jmf@fe.up.pt, galves@def.isep.ipp.pt

*PPGC – UFRGS – Instituto de Informática
Av. Bento Gonçalves, 9500 – Campus do Vale – Bloco IV
Caixa Postal 15064 – CEP 91501-970 Porto Alegre – RS - Brasil

** *Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto - Portugal*

Abstract

This work presents the study and preliminary results of the high level implementation of a self-checking Profibus slave. From an existing VHDL description of the device, a test strategy was studied and implemented, so that the whole circuit has embedded test structures capable to perform at-speed test of the slave. In this paper, we show the used test strategies and implementation results achieved from a synthesis process in a FPGA environment.

1 Introduction

In a world of great and hard competition, it is mandatory for a company to supply the best products and services to its customers. To make this task possible, a good information system is crucial. A system capable to get and process data that come from factory ground is even more critical, since these information contribute to taking several decisions, considering economical, technicial and safety factors. To accomplish this communication between the equipments of a factory and the control system, fieldbuses are used. Due to the great importance of the information that is present in this sort of bus, it is necessary to guarantee the media reliability and avalability. A solution that presents a good tradeoff between cost and efficiency is the integrated self-test approach.

Nowadays, the dimension of the systems make it difficult the test process. For this problem, the solution has been the insertion of test structures into the circuit, rather than test the complex system by external means [D&T97]. Since the fieldbuses are complex systems used in crucial processes which claim for security, the self-checking capability is of great interest. In this work we address the problems related to the self-checking design of an entire slave block of a fieldbus, regarding the description language, time restrictions and area overhead.

This paper presents in section 2 a brief description of industrial buses. In section 3, the Profibus struture, self-chek concept is discussed; in the section 4, test strategies for FSMs are studied. Experimental results are presented in section 5. In section 6, an strategy to fault tolerant slave is presented, and, finally, in section 7, the conclusions and future works are given.

2 Industrial Buses

An industrial bus, also known as fieldbus, is a protocol for the communication between automation and control equipments of an industrial plant, such as sensors or actuators and a microprocessor.

Several types of fieldbuses exist in the market: Profibus [PRO98], Foundation Fieldbus

[FOU96], CAN [CAN98], InterBus [INT98], etc. Each one has its characteristics and advantages, but a standard protocol capable to provide all industry requirements is not avaliable yet.

The fieldbus came on as a solution for the problem of joining and operating instruments of factory ground. The main requirements for this protocol are: a) it should be capable to operate instruments from different manufaturers having different features b) it should work in a distributed system.

This protocol implements some techniques for safe communication, ensuring the transmission of message packages (frames). Some test techniques, such as to verify whether a certain node of the net is active (that is answering to requests and receiving information) are accomplished by most of the protocols.

However, the functionality of the bus, that is, if the protocol itself is meeting to its requirements, is not guaranteed. What is detected now is if some device tied up in the fieldbus is not simply working. If the fieldbus itself fails, stopping its activities completely or even presenting a random behavior, it does not exist a mechanism that comes to identify the fault, not even to signal the error.

The behavior of permanent or random fault in the fieldbus can cause considerable losses, and possibly irreparable damages.

Permanent faults occur in components affected by mechanical rupture or some other waste phenomenon. Random faults are caused by disturbances of external signals, in general, due to radiations or electrical source floating. This paper aims at proporing a test technique that can detect those faults in a particular fieldbus described next, the Profibus.

3 Profibus

The Profibus is divided in a master/slave structure. Master devices are called active elements, because any communication request is accomplished through them.

The slave Profibus is a passive element of the architecture, because it can only answer to requests and never taking the initiative of a communication. It is modeled using finite state machines (FSM).

A slave's Profibus FSM consists on just two states: offline and passive-idle. When a station is found in the offline state it is unable to communicate with any other station that is tied up to the bus. This is due to the fact that the initialization of the whole set of necessary variables to the communication process was not correctly done.

After the initialization of the variables, the FSM transitions from the offline state to the passive-idle state, which represents the operational state. The machine stays in this state until there is a restart request or till a fatal error occurs.

In the passive-idle state the slave station is permanently listening to the messages passing through the bus to check if they are addressed for itself or if they are broadcast messages.

4 The Self-Checking Capability

It is common knowledge that the solution to test problems must be based on the adoption of a hierarchical and structured strategy, taken into account early in the integrated circuit design. Furthermore, it is important that this same strategy can be employed for design validation, manufacturing, maintenance and on-line tests, thus being applicable to all phases of a circuitš lifetime[LUB92].

Existing off-line test techniques cover one part of the test necessary for integrated circuits and boards. These techniques, in the form of Built-In Self-Test (BIST) schemes [AGR93a],[AGR93b], have already reached an important level of maturity regarding the fault coverage, the surface overhead and the test application time.

The other part of the tests necessary for circuits and boards is related to the on-line testing capability, a feature of great importance for systems where poor functioning can, for example, lead to a disaster.

The self-checking circuit implementation, for making on-line testing possible, is based on the encoding of functional block outputs and on the verification of these outputs by specific checkers [CAR68].The circuit error indication is supplied by a global checker to the level immediately higher in the test hierarchy (figure 1).

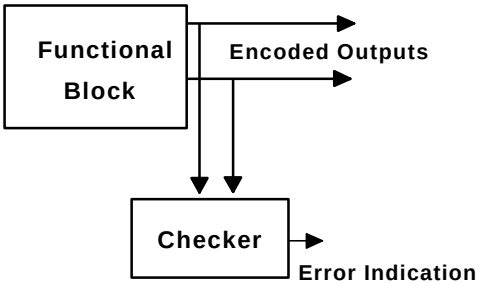

Figure 1 - A self-checking circuit

The goal to be self-checking circuits is often called TSC (totally Self-Checking). The first erroneous output of a functional block has provoke an error indication on its checker outputs. In [CAR68] is introduced the basic ideas and Anderson [AND71] defined the TSC property for functional blocks and for checkers. Smith and Metze [SMI78] define SFS (Strongly Fault Secure) circuits and Nicoladis et al [NIC84] defined SCD (Strongly Code Disjoit) checkers, wich are respectively the largest classes of functional blocks and of checkers allowing the TSC goal to be ensured.

In finite state machines, several faults can occur, among them illegal and incorrect state transition. The first one refers to a transition that is not specified in the machine. In the second one, the transition is specified in the machine, but it happened in an wrong time [ROC95]. Therefore it is very important to embed self-checking techniques in the slave controllers of the fieldbuses, to guarantee that the transition of the states happens in a safe way.

5 FSM On-Line Test Strategy

A typical fault in a FSM is a wrong state transition, that may cause a transition to a new state that does not exist or that is incorrectly selected [LEE96]. An approach capable to detect these single-state transition faults is a parity checker associated with the FSM as shown in figure 2 [SHE94]. This technique was implemented in our Profibus slave.

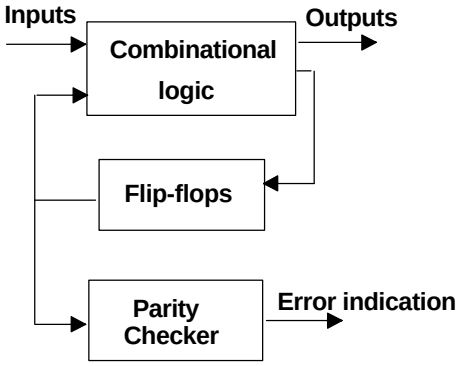

Figure 2 - Parity checker

In this approach, a parity code is associated with each state of the FSM, and a parity checker verifies the state transition to detect possible incorret behaviors. This technique was presented in [SHE94], and its main advantage is that the circuit is capable to keep working as a sequential circuit during test, avoiding scan propagation, very common in this kind of logic. The state code is defined so that fault masking is considered. The detailed coding method is presented in [SHE94] and it is based on definitions of distinguishable machines [CHE92], [LEE76], [CHE90].

This technique makes it possible the on-line test of the FSM, since it does not interfere in the normal operation of the machine.

6 Experimental Results

The slave block of the Profibus was firstly described using two VHDL [ALT95] modules. The first one represents the connection of data of the frame level, called FHD3FRM and, the second one represents the level of physical connection, named PHY_5.

The technique of parity check was implemented in each one of the Profibus slave machines.

Table 1: Synthesis results for the original and self-checking description the using parity check technique

Block	Device	Number of LCs	Percent of device used	Clock Frequency (MHz)
Phy_5	EPF10K20RC240-4	179	15%	18.41
Fhd3frm	EPF10K20RC240-4	363	31%	10.35
Phy_5	EPF10K20RC240-4	248	21%	14.81
Fhd3frm	EPF10K20RC240-4	511	44%	5.13

The four descriptions of the Profibus slave, the original and the self-checking one, were synthesized using the ALTERA tool [ALT95]. Results in terms of number of logic cells, frequency of operation and device used are shown in Table 1. In this table, white rows represent the original descriptions, and gray rows apply to the self-checking descriptions. With these results, it is possible to compare the cost of the insertion of self-checking structures in a real application.

The increase in terms of logic cells is of 40% and of 38,54%, respectively for FHD3FRM and PHY_5.

In terms of clock frequency, the penalties were also important. This decrease may compromise the performance of some applications. However, one should consider the advantages of using a n on-line test technique to increase the safety of the entire system.

7 Evolving to Fault Tolerant Slave

An other technique used to test FSMs is called Monitoring Machine. In this approach an auxiliary sequential circuit, called monitoring machine, operates in lock-step with the main machine, such that any faulty in either of the two machines is immediately detected [PAR96].

The monitoring machine can be applied to FSMs with pre-encoded states and facilitates on-line detection of errors, then it also applies description of slave PROFIBUS.

We are now working on a proposed of a different new implementation of the monitoring machine. In this version, the monitoring machine not only monitores the main Profibus FSM but it is also capable to replace this FSM in case of fault. The capability to distinguish wich of the two machines is faulty is achived by parity check, in much the same way as described in the previous section.

8 Conclusions and Future Works

This work presented an implementation of a self-checking Profibus slave. Two modules inside a VHDL description were identified, studied, and re-defined so that self-checking structures could be inserted. New VHDL descriptions were implemented and synthesized, generating a complete at-speed self-checking Profibus slave**.** Results in terms of area overhead and clock frequency for both descriptions were presented and compared.

In the near future we intend to study the fault coverage for all blocks, to compare the implementation of different self-checking strucutres to study the power consumption for the self-checking implementation and to define the best test schedule.

Since the fault tolerance extension is an on going activity, the experimental referring results of the implementation of the monitoring machine technique will also be demonstrated in future publications.

9 References

[AGR93a] AGRAWAL, V. D. , KIME, C.R., SALUJA, K. K., **A Tutorial on Built-In Self_Test** - Part 1: Principles. IEEE Design and Test of Computers, n. 10, v(1), 1993.

[AGR93b] AGRAWAL, V. D., KIME, C. R., SALUJA, K. K., **A Tutorial on Built-In Self_teste** - Part2: Applications. IEEE Design and Test of Computers, n. 10, v(2), 1993.

[ALT95] **ALTERA DATA BOOK** 1995. Altera Corporation, march, 1995**.**

[AND71] ANDERSON, D.A., **Design of Self-Checking Digital Networks Usign Coding Techniques**, CSL Univ. Of Illionois, Urbana, Report 527, September 1971.

[CAN98] CAN- Controler Area Network - http//www.nrtt.demonco.co.ukl.

[CAR68] CARTER, W.C. And SCHNEIDER, P. R., **Design of Dynamically Checker Computers**, 4th IFIP Congress, Edinburgh, Scotland, v(2), pp. 878-883, 1968.

[CHE90] Cheng, K.T.;Jou, J.Y, **Functional Test Generation for Finite State Machines**, International Test Conference, **Proceedings**..., 1990, pp.162-168.

[CHE92] Cheng, K.T.;Jou, J.Y., A **Functional Fault Model for Sequential Machines**, IEEE Transactions Computer-Aided Design, V(11), 1992, pp. 1065-1073.

[D&T97] **Built-in Self-Test for Design**. D&T Roundtable. IEEE Design and Test of Computers, n. 3, v. 14, july-setember 1997.

[FOU96] Foundation Fieldbus. **Technical Overview, revision 1.0**, September 1996.

[INT98] INTERBUS, http://www.interbusclub.com, 1998

[LEE76]Lee, S.C., **Digital Circuits and Logic Design**, Prentice-Hall, Englewood Cliffs, N.J., 1976.

[LEE96] LEE, D. And YANNAKAKIS, M., **Principles and Methods of Testing Finite State Machines - A Survey** . Procedings of the IEEE, n. 8, v(84), August 1996.

[LUB92] LUBASZEWSKI, M. And COURTOIS, B., **On the Design of Self-Checking Boundary Scannable Boards**, International Test Conference, proceedings..., pp. 372-381, September, 1992.

[NIC84] NICOLAIDIS, M., JANSCH, I. And COURTOIS, B., Strongly Code Disjoint Checkers, Proc. Fault Tolerant Comp. Symp., pp. 16-21, 1984.

[PAR96] PAREKHJI, R. A., VENKATESH, G. And SHERLEKAR, S. D., **Monitoring Machine Based Synthesis Technique for Concurrent Error Detection in Finite State Machines**. Journal of Eletronic Testing: Teory and Applications, n. 8, pp. 179-201, 1996.

[PRO98] PROFIBUS. **Technical Overview**. http://www.profibus.com, 1998.

[SHE94] SHEU, M.l.; Lee, C.L., **Simplifying Sequential Circuit Test Generation**, IEEE Design and Test of Computers, fall, 1994, pp.28-38.

[SMI78] SMITH, J.E. And METZE, G., **Strongly Fault Secure Logic Networks**, IEEE Trans. On Computer, v(27), n 6, June 1978.

[ROC95] ROCHET, R..; LEVEUGLE, R. And SAUCIER, G. **Efficiency Comparison of Signature Monitoring Schemes for FSMs**. International Conference ASP-DAC'95, CHDL'95 and VLSI'95, procedings..., Japan, 1995.