Self-checking synchronous controller design

S.M. Kia
S. Parameswaran

Abstract: Efficient models are introduced for totally self-checking/code disjoint (TSC/CD) and strongly fault-secure/strongly code disjoint (SFS/SCD) synchronous controller models. These models are based on two low-cost, modular, TSC edge-triggered and error-propagating CD flip-flops. Properties of the proposed synchronous controller models are proven. The design procedure for these models and their proper applications are explained.

1 Introduction

Current circuit complexity makes the determination of errors in a circuit arduous. Totally self-checking (TSC) circuits [1] use input–output coding to determine whether a circuit is operating accurately. The code disjoint (CD) property is utilised to design TSC circuits at the system level. CD circuits propagate the error through to indicate the error on the output. However, it is hard to devise circuits that are both TSC and CD.

A strongly fault-secure (SFS) circuit becomes a TSC circuit after a finite number of faults and until then will operate correctly (or is fault-secure). If, for the same faults, the circuit is CD and then becomes self-testing and still remains CD, then the circuit is said to be strongly CD (SCD). SFS circuits satisfy the goals of TSC.

Unordered codes [2] are used for input–output codings in circuits that are TSC or SFS. Techniques have been presented for the creation of combinational TSC circuits [3], SFS circuits [4] and SFS/SCD circuits [2] for a class of unordered codes.

1.1 Related works

As far as synchronous controller designs are concerned, there are no circuits that exhibit all the TSC properties. Nanya's sequential designs [5] add some extra circuitry to the clock of ordinary memories to make register files with self-testing load signals. Nanya's sequential circuits [5] with the proposed register file are based on a previous model [6] which does not consider the faults in storage elements. Other circuits [3, 7] also have not considered the faults in storage elements and have supposed that the clock is fault-free. Therefore, these methods could only produce proper SFS or TSC sequential circuits which do not need memories (asynchronous circuits). It is not easy to design cascaded hazard-free asynchronous circuits because of the fundamental mode of operation requirement.

© IEE, 1999
IEE Proceedings online no. 19990243
DOI: 10.1049/ip-cdt: 19990243
Paper first received 15th July 1997 and in revised form 21st August 1998
The authors are with the Department of Electrical and Computer Engineering, University of Queensland, St. Lucia, Queensland, 4072, Australia

In a self-checking system, any change of inputs or states must be checked for errors. A synchronised change of inputs and states is also necessary in a self-checking circuit to produce error-indicating output [5, 8]. For synchronous circuits to indicate errors, they must be edge-triggered synchronous systems. Level-triggered systems become asynchronous during the active clock level.

1.2 Contributions

The models are based on two CD flip-flops, DD_n-*FF* and TT_n-*FF* [8, 9]. An introduction to the controller model has been given previously [10].

Advantages of the proposed controller are that

(i) it is an edge-triggered synchronous circuit.

(ii) it is TSC/CD for input and state variables.

(iii) errors in state register are considered.

(iv) error in the clock input is considered.

(v) the cost of the checker for inputs and states is reduced by using self-checking flip-flops and SFS/SCD combinational circuits.

(vi) the cost of combinational logic is reduced by sharing product terms.

2 Fault model

The following faults are modelled in this paper: single stuck at 0 fault; single stuck at 1 fault; and multiple unidirectional faults.

Input–output coding techniques are used to detect faults automatically. It is assumed that the time interval between two faults is long enough for the proper cycle and input code to pass through the circuit [12]. Definitions of terms used in this paper are found elsewhere [1, 2, 4].

2.1 Note

The monotone functions of a combinational circuit with unordered codes in the input and output produce a two-level logic circuit with SFS properties [4]. A method is described elsewhere [2] for SFS/SCD combinational circuit design with systematic code input. This method [2] has two procedures called 'covering-nc-CD' and 'covered-nc-CD', which add a few lines to the output of the circuit to make it SFS/SCD. The method of removing untestable terms from an SFS equation to make a TSC circuit is presented elsewhere [3]. The TSC method [3] produces TSC/CD circuits if they are derived from SFS/SCD functions.

These methods of design for SFS, SFS/SCD, TSC and TSC/CD circuits are used for input and output combinational logic of the self-checking controller.

3 SFS/SCD and TSC/CD synchronous controller circuits

The general forms of TSC and SFS/SCD synchronous machines (controllers) with TSC/CD flip-flops are shown in Figs. 1 and 2. The *DD_n-FF* [11] or *TT_n-FF* [11] are used as elements of the state register. However, any flip-flop circuit that can satisfy the TSC/CD property can be used in these models.

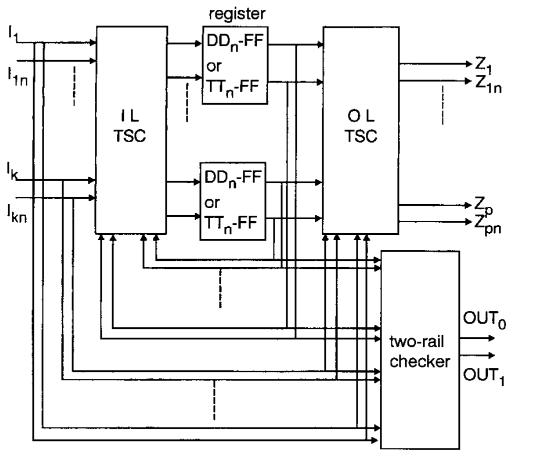

Fig. 1 *Structure of full checker TSC/CD controller with DD_n-FFs or TT_n-FFs*

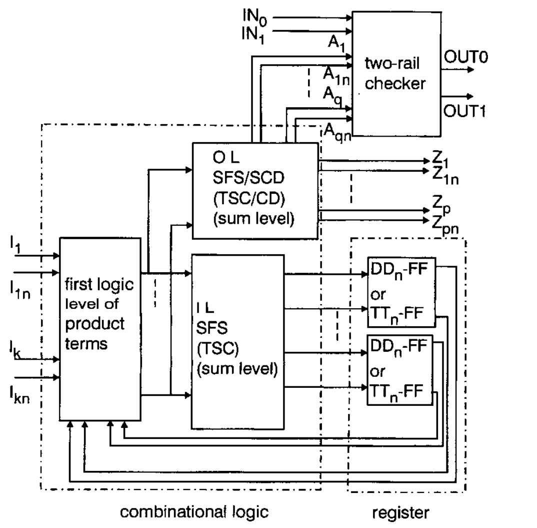

Fig. 2 *Structure of added line checker, TSC/CD and SFS/SCD controller with DD_n-FFs or TT_n-FF_s*

The next-state register made with *DD_n-FF* s produces a low-cost register. The *DD_n-FFs* are not error indicators, and therefore the output logic must be CD or a checker should be used. We explain below how to design the CD output logic for systematic state code assignments.

In the case of *TT_n-FFs*, the errors in the flip-flop become a stuck-at fault. Any fault at the state register made up of *TT_n-FFs* will be a stuck-at fault for the next state and automatically propagated. Therefore, for non-systematic state code assignments the state register with *TT_n-FFs* can be used without any need for the checker.

3.1 Controller models

Mealy or Moore model considerations are common for any synchronous controller. The input, output and state assignment codings should be designed to be self-checking. Input–output coding possibilities are as follows:

Case 1: both input and state assignment codewords are nonsystematic (Mealy or Moore model).

Case 2: input codewords can occur systematically during each state, but state assignment codewords are non-systematic (Mealy or Moore model).

Case 3: a combination of input codewords and state assignment codewords can occur systematically (Mealy or Moore model).

Case 4: a combination of input codewords and state assignment codewords do not occur systematically, but state assignment codewords occur systematically (Moore model).

Two models are shown in Figs. 1 and 2; either one or a combination of them can give a proper solution for the four different cases.

In Fig. 1, the self-checking controller is designed with input logic (IL) followed by a state register, which is followed by output logic (OL). The input signals are directly fed to the output. The input and output logic in Fig. 1 are considered to be SFS or TSC, and a checker is used for input codewords. This checker also checks the state assignment codewords if *DD_n-FFs* are used.

In Fig. 2, it is considered that the combinational logic is designed to be SFS/SCD or TSC/CD by adding extra lines to the output. The checker in Fig. 2 checks only the adding extra lines. Both input logic and output logic share the product terms (Fig. 2).

The product terms of both models can be shared. if the input codewords are checked by other circuits, the controller does not need to be CD for input codewords. If either input or output logic is CD for input codewords, then the controller is CD for input codewords. The self-checking synchronous controller circuits for the four cases are as follows.

3.1.1 Case 1: The monotone function of the next-state logic and output logic produces SFB functions which can be changed to TSC functions. A checker (self-exercised) is required for input codewords. if *DD_n-FFs* are used for the state register, then the checker should also check the state assignment codewords. If TTn-FFs are used for the state register, then the state assignment codewords do not have to be checked (as explained above).

3.1.2 Case 2: In this case, *TT_n-FFs* are used for state register. If the controller is a Mealy machine, the SFB (TSC) output logic can be made SCD (or CD) for input codewords by adding extra lines to the output according to Pagey *et al.*'s method [2]. The input logic remains SFS (TSC). There is no need for the CD property for state assignment codewords because *TT_n-FFs* are used. if the controller is a Moore machine, the SFS (TSC) input logic can be made SCD (or CD) for input codewords by adding extra lines according to Pagey *et al.*'s method [2], and the output logic remains SFS (TSC) (Fig. 2).

3.1.3 Case 3: In this case, *DD_n-FFs* are used for the state register. If the controller is a Mealy machine, the

SFS (TSC) output logic can be made SCD (or CD) for a combination of input codewords and state assignment codewords by adding extra lines to the output according to Pagey *et al.*'s method [2], and the input logic remains SFS (TSC). If the controller is a Moore machine, the SFS (TSC) input logic equations can be made SCD (or CD) for a combination of input codewords and state assignment codewords by adding extra lines according to Pagey *et al.*'s method [2], and the output logic remains SFS (TSC) (Fig. 2).

3.1.4 Case 4:

A checker (self-exercised) is required for input codewords in this Moore machine. The SFS (TSC) output logic function (according to states) can be made SCD (CD) by adding extra lines to the output according to Pagey *et al.*'s method [2], and the input logic remains SFS (TSC) (Fig. 1).

The following example shows how to produce a TSC/CD controller for a simple code.

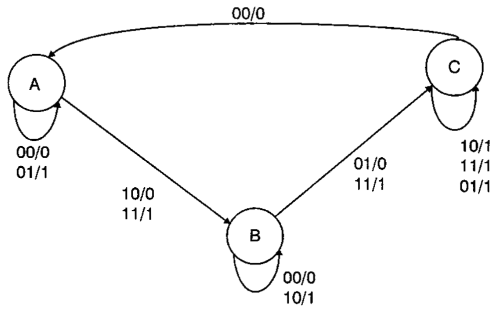

Fig. 3 *State-output flow diagram of synchronous machine S*

3.2 Example

Consider the state flow diagram of a synchronous machine S, which is shown in Fig. 3. The state diagram shows three stages; *A*, *B*, and *C*. The inputs and outputs are not coded and contain two bits for input and one bit for output. The simple next-state table without unordered coding is shown in Table 1.

Table 1: Simple next state function of S (not coded)

State	Inputs = I_1I_2			
Y_1Y_2	0 0	0 1	1 1	1 0
0 0 (A)	0 0 (A)	0 0 (A)	0 1 (B)	0 1 (B)
0 1 (B)	0 1 (B)	1 1 (C)	1 1 (C)	0 1 (B)
1 1 (C)	0 0 (A)	1 1 (C)	1 1 (C)	1 1 (C)

Table 2: Next-state functions of S (excitation table for DD$_n$-FFs)

State	Inputs = $I_1I_{1n}I_2I_{2n}$			
$Y_1Y_{1n}Y_2Y_{2n}$	01 01	01 10	10 10	10 01
01 01 (A)	01 01 (A)	01 01 (A)	01 10 (B)	01 10 (B)
01 10 (B)	01 10 (B)	10 10 (C)	10 10 (C)	01 10 (B)
10 10 (C)	01 01 (A)	10 10 (C)	10 10 (C)	10 10 (C)

For state assignment, two-rail code pairs Y_1, Y_{1n} and Y_2, Y_{2n} are used. The next-state table is shown in Table 2. Input codewords are systematic, but state assignment codewords are not systematic. Since the input codewords are systematic in all three possible states, the controller circuit is designed as in Case 2 in Section 3.1.

The monotone (SFS) functions for the *DD$_n$-FF* excitation circuit can be derived directly from Table 2. The excitation functions for the *TT$_n$-FF*s can be derived from Table 3 (although the next-state table and the excitation table for the *DD$_n$-FF* are the same, this is not the case when considering *TT$_n$-FF*s). Table 3 has been derived from *TT$_n$-FF* state transitions [11].

Table 3: Excitation table of S for TT$_n$-FFs

State	Inputs = $I_1I_{1n}I_2I_{2n}$			
$Y_1Y_{1n}Y_2Y_{2n}$	01 01	01 10	10 10	10 01
01 01 (A)	01 01	01 01	01 10	01 10
01 10 (B)	01 01	10 01	10 01	01 01
10 10 (C)	10 10	01 01	01 01	01 01

The monotone (SFS) functions for *TT$_n$-FF*s excitation circuit can be written from Table 3 as follows:

$$T_1 = Y_{1n}Y_2I_{1n}I_2 + Y_{1n}Y_2I_1I_2 + Y_1Y_2I_{1n}I_{2n}$$
$$T_{1n} = Y_{1n}Y_{2n}I_{1n}I_{2n} + Y_{1n}Y_{2n}I_{1n}I_2 + Y_{1n}Y_{2n}I_1I_2$$
$$+ Y_{1n}Y_{2n}I_1I_{2n} + Y_{1n}Y_2I_{1n}I_{2n} + Y_{1n}Y_2I_1I_{2n}$$
$$+ Y_1Y_2I_{1n}I_2 + Y_1Y_2I_1I_2 + Y_1Y_2I_1I_{2n}$$
$$T_2 = Y_{1n}Y_{2n}I_1I_2 + Y_{1n}Y_{2n}I_1I_{2n} + Y_1Y_2I_{1n}I_{2n}$$
$$T_{2n} = Y_{1n}Y_{2n}I_{1n}I_{2n} + Y_{1n}Y_{2n}I_{1n}I_2 + Y_{1n}Y_2I_{1n}I_{2n}$$
$$+ Y_{1n}Y_2I_{1n}I_2 + Y_{1n}Y_2I_1I_2 + Y_{1n}Y_2I_1I_{2n}$$
$$+ Y_1Y_2I_{1n}I_2 + Y_1Y_2I_1I_2 + Y_1Y_2I_1I_{2n}$$

After simplifying the functions by Diaz *et al.*'s method [3], the following TSC functions are found which are used to synthesise the input logic circuit:

$$T_1 = Y_{1n}Y_2I_2 + Y_1I_{1n}I_{2n}$$
$$T_{1n} = Y_{2n} + I_1I_{2n} + Y_{1n}I_{2n} + Y_1I_2$$
$$T_2 = Y_{2n}I_1 + Y_1I_{1n}I_{2n}$$
$$T_{2n} = Y_{2n}I_{1n} + I_{1n}I_2 + Y_{1n}I_{1n} + Y_2I_2 + Y_2I_1$$

The monotone (SFS) functions for output logic can be found from output Z, Z_n values due to inputs and states of machine S, which are shown in Table 4.

Table 4: Output Z, Z$_n$ values due to inputs and states of machine S

State	Inputs = $I_1I_{1n}I_2I_{2n}$			
$Y_1Y_{1n}Y_2Y_{2n}$	01 01	01 10	10 10	10 01
01 01 (A)	01	10	10	01
01 10 (B)	01	01	10	10
10 10 (C)	01	10	10	10

Functions Z and Z_n in Table 4 are SFS (TSC) but are not CD for input codewords. To design the SFS/SCD output logic circuit, the systematic combination of codes must be considered for code spaces (also referred to as density properties [2]). Therefore, it is necessary to have all possible code words occurring in the circuit. In our example, the four possible code words of input are used.

To design SFS/SCD output logic circuit, we add the last term as shown in Table 5, where the state $Y_1Y_{1n}Y_2Y_{2n}$ = 10 01 has been added. As the outputs are still unknown, they are left as 'dd'. This addition is only in the output table and not in the next-state table. If the added state occurs in the next-state logic (under error conditions), the circuit outputs an error.

In Table 5 there is one two-rail pair for the output Z Z_n. Using Pagey *et al.*'s method [2], we add another bit pair (for the CD property) (The added term in the output table is not really necessary for the TT_n-FF state register. However, Pagey *et al.*'s method [2] makes the circuit easy to design with the added state.) This addition of bits is shown in Table 6.

Table 5: Output table with added code word for SFS/SCD design

State	Inputs = $I_1I_{1n}I_2I_{2n}$			
Y_1Y_{1n} Y_2Y_{2n}	01 01	01 10	10 10	10 01
01 01 (A)	01	10	10	01
01 10 (B)	01	01	10	10
10 10 (C)	01	10	10	10
10 01 (–)	dd	dd	dd	dd

Table 6: Final output table with extra added outputs A_1, A_{1n} for SFS/SCD design

State	Inputs = $I_1I_{1n}I_2I_{2n}$			
Y_1Y_{1n} Y_2Y_{2n}	01 01	01 10	10 10	10 01
01 01 (A)	01, 10	10, 10	10, 01	01, 01
01 10 (B)	01, 01	01, 10	10, 10	10, 01
10 10 (C)	01, 10	10, 10	10, 01	10, 10
10 01 (–)	10, 01	01, 10	01, 01	01, 10

If Diaz *et al.*'s method [3] is followed for output logic functions, the circuit for TSC/CD design is produced. The TSC/CD equations seem simpler than SFS/SCD equations, but this is not always the case. For example, in terms of circuit hardware cost, especially in Mealy model controllers, the SFS/SCD circuit may give a better solution.

3.3 Different input–output coding

Two-rail codes were used for inputs, states and outputs in the above example. It is possible to use other kinds of unordered codes for inputs and outputs, but two-rail codes need to be used for state assignments because of the two-rail TSC/CD flip-flops.

4 Conclusions

We have introduced models and design methods for self-checking edge-triggered synchronous controllers. Conventional circuits were TSC or SFS only, and did not consider the faults in memories and on the input clock. Our proposed method considers the faults in memories and input clock, and reduces the checker cost by producing SFS/CD or TSC/CD circuits. The methodology is achieved using the two low-cost DD_n-FFs and TT_n-FFs for the state register.

5 References

1 ANDERSON, D.A., and METZE, G.: 'Design of totally self checking check circuits for m-out-of-n codes', *IEEE Trans., Computers*, 1973, **C-22**, pp. 263–269

2 PAGEY, S., SHERLEKAR, S.D., and VENKATESH, G.: 'A method for the design of SFS/SCD circuits for a class of unordered codes', *J. Electron. Testing: Theory Appl.*, 1991, **2**, pp. 261–277

3 DIAZ, M., AZEMA, P., and AYACHE, J.M.: 'Unified design of self-checking and fail-safe combinational circuits and sequential machines', *IEEE Trans., Computers*, 1979, **C-28**, pp. 276–281

4 SMITH, J.E., and METZE, G.: 'Strongly fault secure logic networks', *IEEE Trans., Computers*, 1978, **C-27**, pp. 491–499

5 NANYA, T.: 'Error secure/propagating concept and its applications to the design of strongly fault-secure processors', *IEEE Trans., Computers*, 1988, **C-37**, pp. 14–24

6 NANYA, T., and KAWAMURA, T.: 'A note on strongly fault-secure sequential circuits', *IEEE Trans., Computers*, 1987, **C-36**, pp. 1121–1123

7 ÖZGÜNER, F.: 'Design of totally self-checking asynchronous and synchronous sequential machines'. Proceedings of international symposium on *Fault tolerant computing*, June 1977, pp. 124–129

8 KIA, S.M., and PARAMESWARAN, S.: 'Synchronous TSC/CD error indicator for self checking systems'. DILLON, T.S., and FORWARD, K.E. (Eds.).: 'IEEE Pacific Rim international symposium on *Fault tolerant systems*', 1993, pp. 156–160 ((CRL Publishing Ltd., London, Melbourne University, Melbourne, Australia, December, 1993), pp. 156–160)

9 KIA, S.M., PARAMESWARAN, S., and HARRISON, H.B.: 'Design of TSC and SFS/SCD synchronous circuits with TSC/error propagating flip-flops'. Proceedings of IREE/IEAust 12 Australian *Microelectronics* conference, IREE, Marriott Surfers Paradise Resort, Queensland, Australia, October 1993, pp. 75–80

10 KIA, S.M., PARAMESWARAN, S., and ZORIAN, Y.: 'Novel architectures for TSC/CD and SFS/SCD synchronous controllers', 12th IEEE symposium *VLSI test*, April 1994, pp. 138–143

11 KIA, S.M., and PARAMESWARAN, S.: 'New designs for self checking flip flops', *IEE Proc. Comput. Digit. Tech.*, 1998, **145**, (2)

12 WAKERLY, J.: 'Error detecting codes self-checking circuits and applications' (North Holland, New York, 1978)