Design automation of self checking circuits

S. M. Kia, S. Parameswaran
Department of Electrical and Computer Engineering
University of Queensland
St. Lucia, Queensland, 4072, Australia

Abstract

In this paper we explain the steps of the CAD tools developed for self checking circuits. The CAD tools developed are used to design Strongly Fault Secure, Strongly Code Disjoint (SFS/SCD) and Totally Self Checking, Code Disjoint (TSC/CD) circuits. Self checking combinatorial and sequential synchronous circuits including shift registers, counters, adders and checkers are designed, using these tools. The output of these CAD tools is given in structural level VHDL which can be synthesized via commercial tools.

Key words: Design Automation, CAD, Totally Self Checking Circuits, Strongly Fault Secure Circuits, VHDL.

1 Introduction

Self checking circuits give an indication before producing any error on the output. Two circuits which are currently used for self checking circuits are Totally Self Checking (TSC) [1] and Strongly Fault Secure (SFS) [2] circuits. TSC circuits give the indication of error after the first fault occurrence. The SFS circuits may not give an indication of error after the first occurrence of a fault. But, the SFS circuits continues to operate correctly until a certain number of faults, before giving an indication of error.

For cascading the self checking circuits at system level, the error propagation or Code Disjoint (CD) [1] property is quite important. There is no guarantee that TSC circuits are CD and thus may have to use checkers between succeeding circuits. Due to the difficulty in making TSC circuits CD, a method for making SFS combinational circuits Strongly Code Disjoint (SCD) [3] has been developed in [4]. The SCD property ensures the error propagation in the SFS circuits. They have been further extended in order to create asynchronous circuits [5].

These TSC and SFS methods have been applied for combinational parts of synchronous circuits. The authors have introduced two low cost flip flops ($DD_n - FF$ and $TT_n - FF$) which are TSC and CD. These FFs use two rail codes which is suitable for TSC/CD and SFS/SCD synchronous circuits. Internal next state equations, output equations, circuit diagram and symbolic representation of $DD_n - FF$ and $TT_n - FF$ are shown in Figure 1 and Figure 2.

The related methods of application and design of $DD_n - FF$ and $TT_n - FF$ are given in [6] and [7].

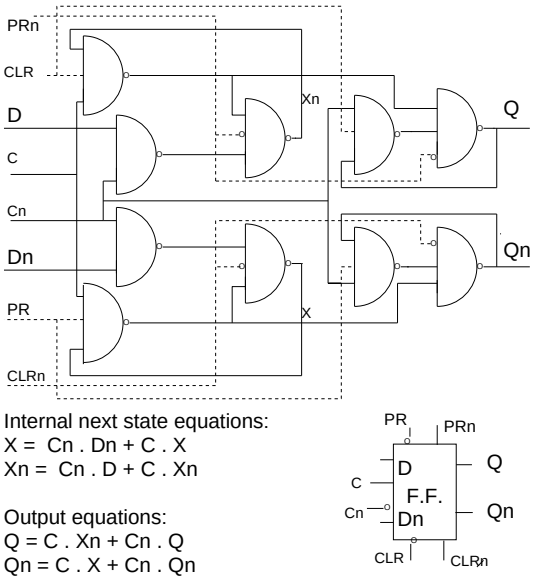

Figure 1: The equations, circuit diagram and symbolic representation of a $DD_n - FF$

When a whole self checking system is implemented using self checking modules (combinational logic and synchronous circuits, adders, registers etc), the level of complexity increases. Thus it is critical to be able to analyze and simulate such circuits before manufacturing. The ability to explore the design alternatives is an added necessity in a self checking design environment.

In this paper the authors introduce the algorithms of CAD tools which can design the modules of TSC, SFS/SCD and TSC/CD combinational and sequential circuits and checkers needed for self checking designs. These modules are then manually put together to build the total system. The outputs are presented in structural VHDL.

The design procedure is explained in section 2. The flow charts and examples are shown in section 3. In section 4 conclusions are given.

2 Self checking processor design

The design procedure of the self checking processors with our CAD tool is done in the following order.

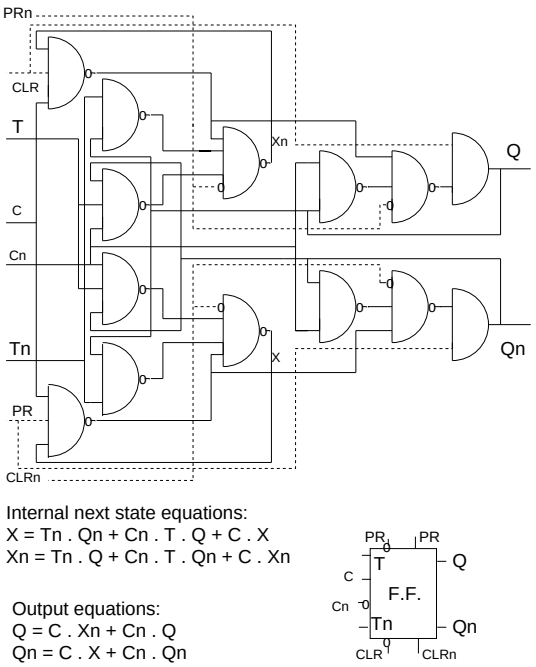

Figure 2: The equations, circuit diagram and symbolic representation of a $TT_n - FF$

Figure 3 depicts the procedure.

1. The introduction of basic elements (e.g. logic gates) and components (e.g. TSC/CD Flip-Flops).

2. The construction of commonly used circuits are held in a library of modules. For example, adders, counters, shift registers and two rail checkers are built in this stage. As far as possible these circuits are TSC/CD in order to have low design cost. We have a single bit designed of these (predefined) circuits in the library.

3. The analysis and generation of the combinational circuits for different input-output mappings. These circuits are designed with SFS/SCD or TSC/CD rules and can be used with different unordered codes[1].

4. The analysis and generation of the sequential circuits for controllers. These modules are clocked and they are SFS/SCD or TSC/CD utilizing the two rail code for state assignments.

5. The construction of the whole system with the generated parts. The subsystems previously designed are connected together manually, in order to arrive at the final design.

[1] Unordered codes are used for self checking circuits. Two rail and Berger coding can be done by the CAD tools. Other kinds of unordered code such as $m - out - of - n$ codes can be given too.

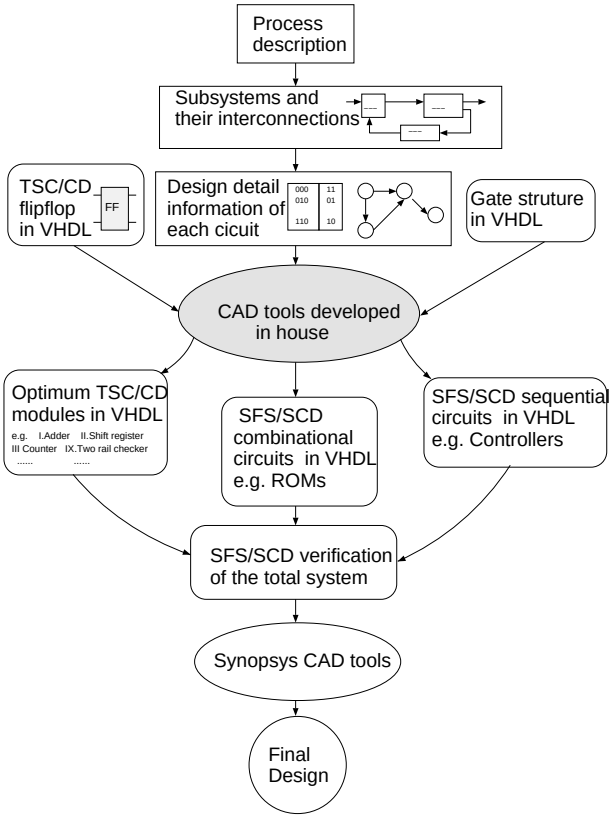

Figure 3: Procedure of automated self checking system design with VHDL structure

Presently the CAD tools developed in this project find the equations and create separate VHDL descriptions for each module. The input to the CAD tools differ according to the circuit. The output of the CAD tools are modules such as combinational logic, synchronous sequential logic, adders, registers, etc, written in VHDL structural description. These descriptions are put together to create a total VHDL description of a system which is then put through commercial design tools to create the final system.

3 Algorithms of developed CAD tools

In the system developed at the University of Queensland, two analytical CAD tools generate combinational and synchronous sequential logic circuits.

The combinational circuits can be made with variety of unordered codes. The unordered codes that are used for self checking circuit design, generate the SFS circuits [2]. The method in [5] is used to generate TSC circuits. The transformation of an SFS circuit to an SFS/SCD is based upon the method in [4]. By applying the method of [5] on SFS/SCD circuits, the TSC/CD combinational circuits are built. Predefined component generators are also available which create TSC/CD adders, decoders and checkers. These predefined component generators generate circuits for any bit widths prescribed by the user. Non predefined

combinational circuits are created in two level logic format (either And–Or format or Nand format). The synthesis of these circuits will be given in Algorithm 1.

The synchronous circuits use two rail codes. Synchronous circuits are divided into two sections: the first, predefined circuits such as shift registers, counters etc; and the second controllers. In the first category, a simple generator generates optimized TSC/CD circuits given the necessary bit width. In the case of the controller, a more elaborate algorithm is used to design the circuit. The controller algorithm will be given in "Algorithm 2". Self checking Controllers have a state register which is constructed with $DD_n - FF$s or $TT_n - FF$s, and self checking combinational circuitry.

The models for TSC/CD and SFS/SCD synchronous circuits are presented by the author in [8]. Details of the self checking system design are given by the author in [9].

The steps of transaction with these two CAD tools are summarized in the following two sections. The terms used in the following sections are due to mentioned design methods.

3.1 Combinational logic circuits

Algorithm 1

1. Read the input output table.
2. Read the number of input output bits from the input output table.
3. Read the desired method of coding for inputs and outputs, i.e. 1- Two rail code 2- Berger code 3- As it is [2].
4. Create a new input output table with desired coding and find the number of bits for input output coding.
5. Read the desired method of design, i.e. 1- TSC 2- SFS/SCD.
6. Separate the implicants for each output variable.
7. According to the selected option follow the TSC(a) or SFS(b) methods.
   (a) TSC procedure:
      i. Extract the testing implicants for each variable.
      ii. Check every variable in every implicant for each function to be tested by testing implicants. If any variable is not tested, eliminate that variable. Continue this process for all variables of the functions which are left over from last elimination.
      iii. Simplify the repeated implicants and implicants which are subset of bigger implicants and calculate the number of product terms and size of each product term.

   (b) SFS/SCD procedure:
      i. Follow the covering-nc-CD procedure: Find every immediately covering non codeword which covers at least two codewords. If outputs of all covered codewords are the same, make them different by setting new values to don't care conditions and if there is no don't care value, add new two rail codes.
      ii. Follow the covered-nc-CD procedure: Find every immediately covered non codeword which is covered by at least two codewords. If outputs of all covering codewords are the same, make them different by setting new values to don't care conditions and if there is no don't care value, add new two rail codes.
      iii. Set any left don't care values to a two rail code, e.g. 01 or 10.
      iv. Calculate the number of product terms and the size of each product term.
8. Print the output functions if required.
9. If TSC/CD was selected, use the SFS/SCD design and apply the TSC method else go to the next step.
10. Write the VHDL description of circuit structure as follows.
    (a) Read the name of the file for VHDL description of the circuit.
    (b) Read the desired hardware i.e. 1- AND-OR 2-NAND.
    (c) Put the entity information in the file.
    (d) Put the necessary component in the file.
    (e) Introduce the interconnection points as signals in the file.
    (f) Make a NAND or AND gate for every new product term (do not repeat).
    (g) Make a NAND or OR gate for every output function.
    (h) Make the input, output and intermediate line connections.
    (i) Print the ending message including number of gates and the sum of their fan-in.

Any non meaningful input will result in an error message being displayed.

3.1.1 Combinational circuit design example

Given below is the input to Algorithm 1. The first line gives the program information about the number of inputs and outputs. In this case there are three inputs and four outputs.

[2] Any other kind of unordered code like $m - out - of - n$ is directly given in the input table and no more coding is required.

3 4
000 0011
001 0110
010 1100
011 1001
100 1010
101 0101
110 0011
111 1010

We select the Berger coding for the input which adds two more bits to the input codes. The 2-out-of-4 coding is chosen for the output of the circuit as shown in the above input-output table. The SFS/SCD functions of outputs (Z) are found due to inputs (a) with no added extra output lines as follows:

$Z_1 = a_2a_4 + a_2a_3a_5 + a_1a_4 + a_1a_2a_3$
$Z_2 = a_3a_4 + a_2a_4 + a_1a_3a_5$
$Z_3 = a_4a_5 + a_3a_4 + a_1a_4 + a_1a_2a_5 + a_1a_2a_3$
$Z_4 = a_4a_5 + a_2a_3a_5 + a_1a_3a_5 + a_1a_2a_5$

Then by demanding the TSC/CD functions the equations program generates the following:

$Z_1 = a_2a_4 + a_2a_3 + a_1a_4$
$Z_2 = a_3a_4 + a_2a_4 + a_1a_3a_5$
$Z_3 = a_4a_5 + a_3a_4 + a_1a_4 + a_1a_2$
$Z_4 = a_5$

The procedure is continued and the VHDL description of the circuit is given with the size information (10 gates which the sum of their fan-in is equal to 25) of the circuit. Shared terms have been taken into account.

3.2 Synchronous circuits

Some extra points related to sequential synchronous design are considered below.

- Both functions of internal states and outputs of the circuit must be considered.
- Only two rail codes are used.
- The internal states or outputs functions can be TSC, SFS, TSC/CD or SFS/SCD.
- With two rail codes, the covering and covered nc-CD procedures in SFS/SCD design are identical.
- The Mealy and Moore model can be selected.
- One of two Flip-Flop kinds ($DD_n - FF$ or $TT_n - FF$) can be selected for a state register.
- The product terms of the combinational logic are shared between internal states and outputs.

The algorithm is as follows:

Algorithm 2

1. Read the synchronous circuit table.
2. Read the desired kind of Flip-Flop for state register, i.e. 1-$DD_n - FF$ 2-$TT_n - FF$
3. Read the number of bits and combinations of input, state and next states from the synchronous circuit table.
4. If $TT_n - FF$ is selected, change the next state table accordingly.
5. Read the desired method of design, i.e. 1- TSC 2- SFS/SCD.
6. Read the desired model of the synchronous circuit, i.e. 1- Mealy 2- Moore.
7. For Moore model read if the output is to be extended as a Mealy model or not.
8. If Moore model output is to be extended as a Mealy model, put equal outputs in the table for all input assignments. From this point the extended Moore to Mealy model will be considered as a Mealy model.
9. Separate the implicants for each next state variable.
10. According to the selected method follow the TSC (a) or SFS/SCD (b) methods for the next state table.
    (a) TSC procedure for next state table:
    Follow the TSC procedure for the next state functions as the TSC procedure for outputs of the combinational circuit (step "a" in the combinational circuit algorithm).
    (b) SFS/SCD procedure for next state table:
    Follow (i) for Mealy and (ii) for Moore model.
        i. Do not change the next state table. It is already SFS.
        ii. Follow the SFS/SCD procedure for the next state functions as the SFS/SCD procedure for outputs of the combinational circuit (step "b" in the combinational circuit algorithm).
11. Print the next state functions with new added functions if wanted.
12. For Mealy model implicants will be made according to input and state assignments. For Moore model, implicants will be made only according to state assignments.
13. Separate the implicants for each output variable.
14. According to the selected method follow the TSC (a) or SFS/SCD (b) methods to derive the output table.

(a) TSC procedure for output table:
Follow the TSC procedure for the output functions as the TSC procedure for outputs of the combinational circuit (step "a" in the combinational circuit algorithm).

(b) SFS/SCD procedure for output table:
Follow the SFS/SCD procedure for the output functions as the SFS/SCD procedure for outputs of the combinational circuit (step "b" in the combinational circuit algorithm).

15. Print the output functions with new added output functions if required.

16. If TSC/CD was selected, use the SFS/SCD design and apply the TSC method else go to the next step.

17. Write the VHDL description of circuit structure (see step 10 of the combinational circuit algorithm).

18. Display the ending message including the number of gates and the sum of their fan-in and the number of Flip-Flops.

3.2.1 Synchronous circuit design example

Suppose the input file name for our synchronous circuit design is 'SYNexample' with the following input-output coding relations.

```
8 6 5 6
010101 010110 011010 011001 101001 101010 100110
100101
010101 010110 101001 101010 011010
1 1 1 1 2 2 2 2
2 2 3 3 3 3 2 2
4 4 4 4 4 4 4 4
5 5 4 4 4 4 5 5
2 2 2 2 2 2 2 2
Mealy 4
0101 0101 0101 0101 0101 0101 0101 0101
0101 0110 0110 0101 0101 0110 0110 0101
1001 1001 1001 1001 1001 1001 1001 1001
0101 0101 0101 0101 0101 0101 0101 0101
1001 1001 1001 1001 1001 1001 1001 1001
```

The first line shows that there are 8 input combinations using 6 input variables and 5 state assignment combinations using 6 state variables. The next two lines give the input assignments and state assignments. The next states are given by their numbers in the next five lines. For example, the second line of these five lines is

```
2 2 3 3 3 3 2 2
```

which gives the eight possible next states under eight possible input combinations if the present state is 2. The controller is a Mealy model with 4 output variables. The output table is given in the last 5 lines. Note that two rail codes are used.

The SFS/SCD design of the controller with $DD_n - FFs$ ($TT_n - FFs$ is also possible) is made by adding a pair of lines to the output lines. The TSC/CD process simplifies the SFS/SCD equations [5]. The TSC/CD next state (Y), output (Z) and added output (Z_A) equations found by the developed CAD tools are given below:

$$
\begin{aligned}
Y_1 &= y_4y_5x_3 + y_3y_6 + y_1x_3\\
Y_2 &= y_4x_4 + y_4y_6 + y_5x_4 + y_2y_3\\
Y_3 &= y_4y_5x_3 + y_3y_6 + y_1\\
Y_4 &= y_4x_4 + y_4y_6 + y_2x_4 + y_2y_3\\
Y_5 &= y_6x_1 + x_1x_4 + y_5x_4 + y_3\\
Y_6 &= y_4y_6x_2 + y_4x_2x_3 + y_4y_5x_3
\end{aligned}
$$

$$
\begin{aligned}
Z_1 &= y_3y_6 + y_2y_3 + y_1y_6 + y_1y_4\\
Z_2 &= y_2y_4 + y_1y_3y_5\\
Z_3 &= y_2y_4y_5x_5 + y_2y_3y_6 + y_1y_4y_6\\
Z_4 &= y_2y_4x_6 + y_2y_4y_6 + y_5x_6 + y_1y_3 + y_3y_5 + y_1y_5
\end{aligned}
$$

$$
\begin{aligned}
Z_{A1} &= y_6x_2x_4x_5 + y_6x_2x_3x_6 + y_6x_1x_3x_5 + y_6x_1x_4x_6 +\\
&\quad y_2y_4y_5x_2x_4x_6 + y_2y_4y_5x_2x_3x_5 + y_2y_4y_5x_1x_3x_6 +\\
&\quad y_2y_4y_5x_1x_4x_5 + y_3x_2x_4x_5 + y_3x_2x_3x_6 + y_3x_1x_3x_5 +\\
&\quad y_3x_1x_4x_6 + y_1x_2x_4x_5 + y_1x_2x_3x_6 + y_1x_1x_3x_5 + y_1x_1x_4x_6\\
Z_{A2} &= y_6x_2x_4x_6 + y_6x_2x_3x_5 + y_6x_1x_3x_6 + y_6x_1x_4x_5 +\\
&\quad y_2y_4y_5x_2x_4x_5 + y_2y_4y_5x_2x_3x_6 + y_2y_4y_5x_1x_3x_5 +\\
&\quad y_2y_4y_5x_1x_4x_6 + y_3x_2x_4x_6 + y_3x_2x_3x_5 + y_3x_1x_3x_6 +\\
&\quad y_3x_1x_4x_5 + y_1x_2x_4x_6 + y_1x_2x_3x_5 + y_1x_1x_3x_6 + y_1x_1x_4x_5
\end{aligned}
$$

By following the CAD steps, the circuit design in the VHDL description language is made with 3 $DD_n - FFs$, 69 gates which the sum of their fan-in is equal to 272. The block diagram of the TSC/CD controller is shown in Figure 4. The SFS/SCD implementation has the same structure as in Figure 4 except for a larger combinational part. Other controller structures can be implemented.

The ZA1 and ZA2 output lines support the error propagation of the controller. These two lines can be used for the cascading of self checking circuits by combining them to the same lines of previous stages.

3.2.2 A Large controller design example

As an example of a large circuit, the self checking structure of the controller of a 32 bit sample processor, DP32 given in [10], is designed. The next state diagram is derived with 22 states. The inputs are decoded to get 7 input two rail code pairs. Five pairs of two rail codes are used for state assignments. The outputs are decoded to 24 pairs of two rail codes. The next state and output table of the controller has 128 columns and 22 rows.

For the TSC/CD design of the circuit as a Mealy model, the circuit required about 3100 gates and 5 self checking Flip-Flops as state registers. After including the next state and output table information in a file, the total time including the latency for choosing different design options is less than one hour (45 minutes user time, 30 seconds system time) on a DEC 5000 series computer. Note that the circuit is first designed as SFS/SCD and then changed to TSC/CD. It is easy to examine different state assignments and models to

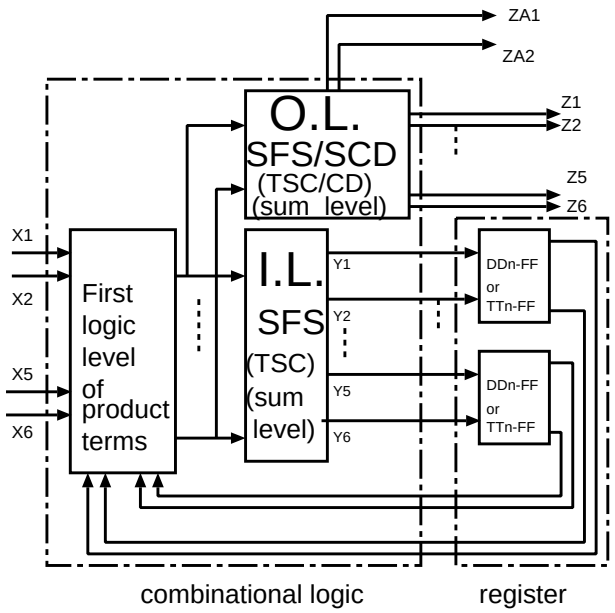

Figure 4: Block circuit diagram of TSC/CD (SFS/SCD) controller 'SYNexample'

get to an optimum design.

The complexity of the SFS/SCD design procedure increases exponentially with the number of inputs and outputs [4]. This exponential complexity favours the decomposing of circuits into smaller sections.

4 Conclusions

Self checking circuit CAD tools were introduced. Two algorithms were given which designed combinational circuits and synchronous controllers. Other generators are used for predefined components such as shift registers adders etc. Applications of the two algorithms were demonstrated through examples. The hard and time consuming design task of self checking circuits are accurately done in a short time. It is now possible to explore the design space to find an optimum solution. The designed circuit output is also given in VHDL which can then be synthesized using commercial tools.

References

[1] D. A. ANDERSON & G. METZE, "Design of totally self checking check circuits for m-out- of-n codes ," *IEEE Transactions on Computers* C-22 (Mar. 1973), 263–269.

[2] J. E. SMITH & G. METZE, "Strongly Fault Secure logic networks ," *IEEE Transactions on Computers* C-27 (June 1978), 491–499.

[3] M. NICOLAIDIS, I. JANSCH & B. COURTOIS, "Strongly code disjoint checkers," *Proc. of 14th Fault Tolerant Computing Symposium* FTCS-14 (1984), 16–21.

[4] S. PAGEY, S. D. SHERLEKAR & G. VENKATESH, "A method for the design of SFS/SCD circuits for a class of unordered codes ," *Journal of Electronic testing: Theory and Applications* 2 (1991), 261–277 .

[5] M. DIAZ, P. AZEMA & J. M. AYACHE, "Unified design of Self-Checking and Fail-Safe combinational circuits and sequential machines ," *IEEE Transactions on Computers* C-28 (Mar. 1979), 276–281.

[6] S. M. KIA & S. PARAMESWARAN, "Design of TSC and SFS/SCD synchronous circuits with TSC/error propagating Flip-Flops," in *Proc. 12th Australian Microelectronics Conference*, H. B. HARRISON, ed. #MICRO'93, IREE,I E Aust, Marriott Surfers Paradise Resort, Queensland, Australia, Oct. 1993, 75–80.

[7] S. M. KIA & S. PARAMESWARAN, "Synchronous TSC/CD error indicator for self checking systems," in *The 1993 Pacific Rim International Symposium on Fault Tolerant Systems*, T. S. DILLON & K. E. FORWARD, eds. #PRFTS'93, IEEE, CRL Publishing Ltd, London, Melbourne University, Melbourne, Australia, Dec. 1993, 156–160.

[8] S. M. KIA & S. PARAMESWARAN, "Novel architectures for TSC/CD and SFS/SCD synchronous controllers," in *12th IEEE VLSI TEST SYMPOSIUM*, Y. ZORIAN, ed. #VTS94, IEEE, Cherry Hill, New Jersey, April 1994, 138–143.

[9] S. M. KIA, "Novel circuits, methods and automation in self checking circuits," Department of Electrical and Computer Engineering, the University of Queensland , PhD thesis(to be submitted) , Australia, 1994.

[10] P. J. ASHENDEN, *The VHDL Cookbook*, Internet, Dept.of Computer Science, University of Adelaide, South Australia, 1990.