Enhanced Moodle System with Assignment Self Checking

Mohammed Fadle Abdulla
Computer Science and Engineering Department,
Faculty of Engineering, University of Aden, Aden, Yemen

ABSTRACT. One of the main issues in designing a good course management system for the Distance Learning is the turnaround time, such as the time consuming in the correction process of the assignments to a course been assign with relatively large number of students. Here, the checking and correction of the homework submitted by the students often takes a lot of time and effort. This is especially true for digital logic design exercises, where a variety of functionally equivalent designs is possible. In this paper we present our vision to solve this problem for the Distance Learning system. We show how to apply a novel signature analysis techniques known as concurrent intermediate checking register CIC to check the circuits designed by the students, at predefined time instants called check points, which have the property that the original signature becomes equal to a function of the circuit responses. As the self test is run on the students' own workstation, the system provides immediate feedback about whether a circuit is working, and it allows to reduce the number of false solutions submitted by the students. We have verified the desired properties using selected high-level synthesis benchmarks.
KEYWORDS: Distance learning, Self_test, BIST, Verilog simulator, Moodle.

Introduction

Computer engineering curricula ensure students' acquisition of knowledge from this field. Students must acquire skills and working experience with the objects of the field. While programming tasks have become standard

parts of computer science courses, integration of circuit design technique in computer engineering courses lags behind, partly due to the expense and complexity of these tools compare to compilers [Mic07,HC03]. A course such as Computer Design introduces digital circuits, sequential circuits, boolean functions, memories, and their design, and the its assignments to the students should include design tasks of different digital circuits. Paper_and_pencil designs are known to be error prone and frustrating, thus bring motivation down. However, the using of the hardware design system, such as Pspice, Verilog, has a great impact in improving this situation. By simulation, the student can detect design errors and correct them. This increases motivation and improves the quality of submissions. And also the students acquire skills in working with the tools of the discipline. For the teacher, the advantages are improved readability of assignment and the possibility of automated correctness tests in case of many assignments especially for large number of students. The teacher can now concentrate on style and quality of the design.

The testing of the VLSI circuits the most complicated and time-consuming problems in digital design. Students should be able to test the correctness of their design at home. Student in the engineering faculties learns about design, but only truly dedicated students learn test. The next generation of graduated engineers involved with System-on- Chip (SoC) technology should be made better aware of the importance of test and be equipped with the skill in design_for_test (DFT) and Built_in self_test (BIST) [KMZ79,Lal97].

Another crucial problem arises in the environment of the distance learning, which is the time consuming in the checking and correction of the digital design work submitted by the students, especially with a relatively large number of students.

Many techniques have been proposed to ease the problem of correction of certain types of exercises for distance learning. The WebAssign system [Nor02,VPM04, B+99] which provide a module for automatic checking of students' homework solutions which are submitted as HTML document. It allows to check numerical values and answers to simple YES-NO or Multiple-Choice questions. Hades system [HC03, UW00, Hen00] tried to integrate a means to automate the checking of digital logic homework assignments based on conventional compaction technique using a scheme such as the multiple-input signature register MISR registers. In Hades approach, a large number of test patterns are applied to the circuit under test (CUT). The output responses of the CUT are compacted in a

LFSR register into a final signature which is then compared with a fault-free signature, which is obtained from a pre-process known as Fault Simulation [KMZ79, Lal97, Hen00].

Unfortunately, this system is not reliable due to the known problems of the conventional compaction techniques which based on the LFSR registers, such as:

(i) the delay time to get the final signature due to the large number of test patterns,

(ii) the high percentages of aliasing, which the test fails to detect the faulty circuit,

(iii) fault simulation of the fault free circuit is required to be done by the teacher to select the input seed for tests,

(iv) the large size of storage of the fault free signatures [Lal97, Bus99].

We propose a new and effective technique to enable the students to check the digital logic homework assignments fast and more efficient in terms of the time consuming, storage used and the correctness of the test. We integrated the use of Verilog simulator into web_based Course management system (CMS) developed with the Moodle system for the Faculty of engineering, Aden University. The Moodle is an open source software design to create a rich online learning environment [***07].

Teachers and students can access the CMS system server from any place with an internet connection. The teacher can encourage and in some cases require student participation, through different available activities provided by the CMS, such as, assignment, forums, chat, wiki, questionnaire, etc. This new teaching environment achieves

(i) the availability of course material independent of place and time,

(ii) learning according to the students‘ own needs,

(iii) up-to-date course material.

The new conception presented in this paper allows to improve the skills of students to be educated for digital hardware and SOC design in test related topics, as illustrated in the following section. The basic design of the proposed technique is provided in section 2. Section 3 describe the principle of creating the assignments for the students. The evaluation of the proposed technique is discussed in sections 4 and 5. Section 6 includes the final conclusion and future work.

1 Teacher/Student Perspectives

Using the CMS environment, the assignment creation is meant to be flexible and the teacher have wide range of possible assignments which can be configured in only a few minutes. He can specify the following information for an assignment, namely, (1) the assignment *title, description, and file,; (2)assignment type* (offline online text, upload file), *(3) grades, scales, (4) available_from* (date the assignment is visible to students), *(5) Due_date* (sets the date the assignment is due), (6) *Prevent-late-submissions* option (an additional date indicating the limit for late submissions), and (7) *submission method.*

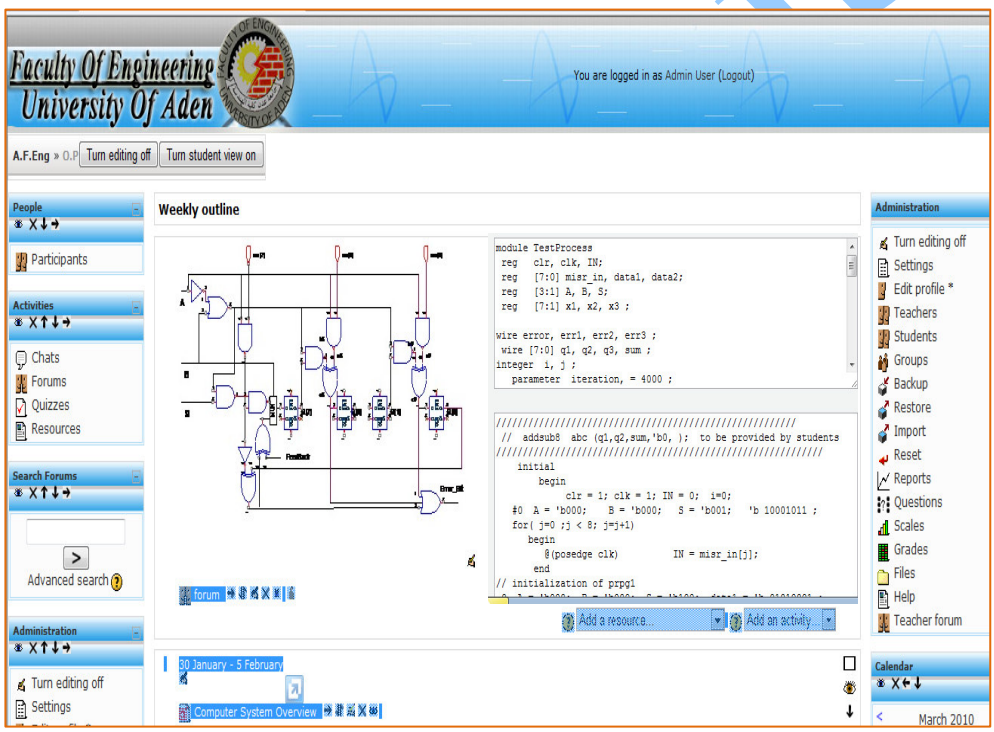

Figure 1. Preparing an assignment in CMS

Teachers are able to personalize the exercises for each student through different set of testing parameters. The teacher can adjust the working period where the students can access the assignment and submit their design files. During the working period, a student can access the assignment and testing template files, develop the solution and running the

pretest process and correcting the design and then submit it to the system server. He also can repeatedly improve his solution and submit it again until the end of the working period is reached.

The students are not asked to carry out boring measurements, but instead, they are asked to develop designs to the given assignment, and they themselves have to test and carry out testing process to find the correctness of their design.

As the self test is run on the student's own PC, the system provides immediate feedback about whether a circuit is working with very low percentage of aliasing and hence it allows to reduce the number of false solutions submitted by the student. Figure 1 illustrates the preparation of a Verilog testing template to be sent with an assignment for the students to solve and test.

2 Basic Idea

Let $t = (t_1, \ldots t_n)$ be a sequence of n test patterns applied to the circuit under test (CUT). Let $r=(r_1, \ldots, r_n)$ be the outputs responses of the CUT. Let $r^*=(r^*_1, \ldots, r^*_n)$ be the fault-free output responses of the CUT. Assume that the outputs are compacted into m-bit signature S using MISR.

Let S_j be the content of the signature register after the responses $r_1, r_2, \ldots, r_j$ have been compacted. We refer to S_n as the signature of the CUT. Conventionally, the signature S_n is compared with the signature of the fault-free circuit G_n (also called the golden signature); the CUT is declared faulty if the signature of the CUT is different from the golden signature.

In this paper, we shall refer to S_i, $1 \leq i < n$, as partial signatures of CUT; we can also talk of a golden partial signature (GPS) in the same spirit. A faulty design can render one or more responses r_j to be different from the correct responses r^*_j.

Unlike conventional testing schemes, where only the final signature S_n is compared with G_n, comparing multiple intermediate signatures with the respective partial golden signatures will improve the confidence of the testing scheme. Conventional signature schemes suffer from aliasing, which occurs when the signature of the faulty CUT matches the golden signature. The concept of partial signature matching reduces the probability of aliasing. However the storing and comparing process of many partial signatures is expensive, and unattractive approach. In this paper we present a solution to this problem, based on the notation of *check points*. The

essential idea is that there are several time instants ***i*** at which the golden partial signature can be derived indirectly from the responses of the CUT.

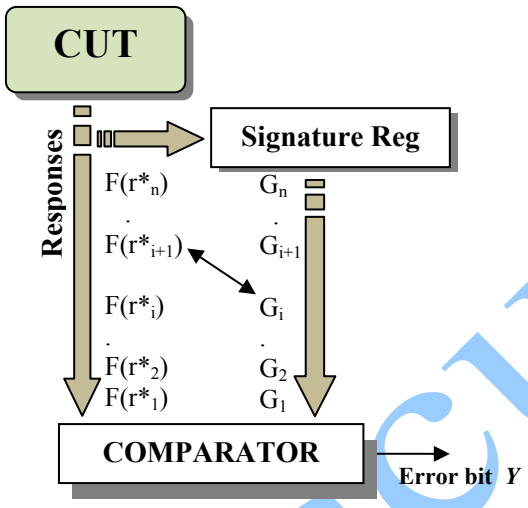

Figure 2. Generating the Error bit Y

Given a string A, let the notation $\tilde{A}$ refer to the rotation of A by 1 bit to the right. Let F_i be a Boolean function which maps an m-bit input to an m-bit output by flipping the j^{th} bit position in the input iff the binary representation of ***i*** contains a 1 in the j^{th} position, $1 \leq j < m$. There are 2^m possible such functions which we denote F_0, F_1,…, F_{2m-1} . Our scheme is based on the observation that there are several instances ***i*** such that, for a given F_x, $F_x(r^*_i) = \tilde{G}_{i-1}$

We refer to such an instance ***i*** as a fault free check point or a golden check point GCP_i. A check point CP_i is called a faulty check point if $F_x(r_i) \neq \tilde{S}_{i-1}$, i.e. $CP_i \neq GCP_i$. The signal which computes $F_x(r_i) \neq \tilde{S}_{i-1}$ at check points is called an error bit. The error bit is LOW at a good check point instance and HIGH at a faulty check point. There must be at least one faulty check point to declare that the CUT is faulty. Availability of these nearly free check points is the key to our scheme. Figure 2 illustrates the idea behind a check point.

The error bit can be easily implemented through a simple modification to the MISR register. The modified register is called a

"Concurrent Intermediate signature Comparison Register" (CIC) and is illustrated in Figure 3. The reader may wonder whether such check points will always exist for a CUT; our experiments indicate that they do. The error bit Y at the time instant ***i*** is defined as $Y_i = \begin{cases} 0 & if \quad F(r_i) = S_{i-1} \\ 1 & otherwise \end{cases}$

The database consists of the indicator bits ***X*** at the time instant ***i***, defined as

$$X_i = \begin{cases} 1 & if \quad \text{i is a } check\ po\text{int instant} \\ 0 & otherwise \end{cases}$$

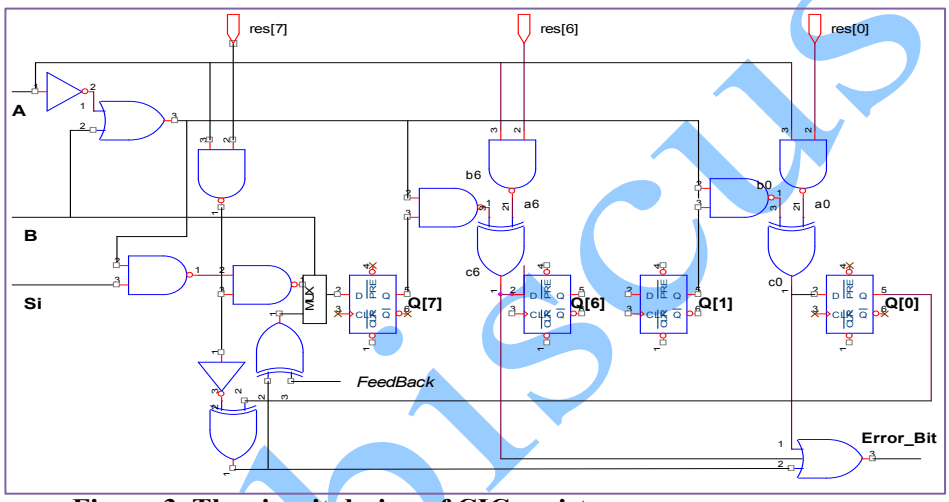

Figure 3. The circuit design of CIC register

The CUT is faulty if the logical AND of bits ***X*** and ***Y*** is 1 at any time. A check point at instant ***i*** becomes faulty if $S_{i-1} \neq F(r_i)$. We now present arguments about the effectiveness of our scheme in reducing aliasing. If a check point falls between two faulty responses, it is bound to reveal the malfunction of CUT, which may otherwise go undetected. This result is stated as the following: " *If, at time instant i, the partial signature* $S_i \neq G_i$ *and* r_h *is the next faulty response, then check point* CP_j, $I \leq j < i+h$ *must be faulty"*. Assume that a faulty free CP $_{i+j}$ exists at time instant $i+j$, where $0 \leq j < h$. This implies $S_{i+j-1} = F(r_{i+j})$. Further r_{i+j} is a correct response by assumption. Therefore, the partial signature $S_{i+j-1} = G_{i+j-1}$, for all $i \leq j < i+h$. However, we have a contradiction for i+j.

2.1 Responses Functions

We define an auxiliary Boolean function that maps a *b*-bit input to a *b*-bit output by flipping *j* bits of the input, $0 \leq j < b$. There can be 2^b such functions and we refer to them as $\boldsymbol{H_0, H_1, \ldots, H_{2b-1}}$. The function H_i flips the j^{th} bit position of the input if and only if the binary representation of *i* contains a 1 in the j^{th} position. The function $\boldsymbol{H_0}$ is the identity function and leaves the input unchanged. Given a binary string ***B***, let the function *LeftRotate(B)* refer to the rotation of ***B*** by 1 bit to the left. For reasons that will become clear later, the signature function *F* is selected as $F\left(r^*_{i+1}\right) = LeftRotate(H_x(r^*_{i+1}))$

Where *x* is determined as explained later; the fault free (golden) check point GCP_{i+1} will thus refer to a time instant ***i*** where $G_i = F(r^*_{i+1})$. We denote by γ_M the set of all golden check points of a *CUT*.

$$\gamma_M = \{ \boldsymbol{GCP_{i+1}} \mid \boldsymbol{G_i = LeftRotate(r^*_{i+1})} \quad \boldsymbol{1 \leq i < n} \}$$

Figure 3 gives the Verilog modules of the proposed CIC register to generate the error status ***Y*** for the CUT. An MISR procedure is used to compact the blocks, as explained earlier. At the same time, we compare $\boldsymbol{r^{b-1}_i}$, $\boldsymbol{r^{b-2}_i}$, $\boldsymbol{\ldots r^1_i}$ with the signature register bits $\boldsymbol{S_{b-2}, S_{b3}, \ldots, S_0}$. Another EXOR routine is used to compare $\boldsymbol{S_{b-1}}$ with r^0_i. The values $r^{b-1}_i \oplus \boldsymbol{S_{b-j-1}}$, $1 \leq j < b$ and $r^0_i \oplus \boldsymbol{S_{b-1}}$, are OR-ed together to generate the bit ***Y***. (Replacing the *OR* process by *NAND* one will implement the function $\boldsymbol{H_2^m{}_{-1}}$). Figure 4 gives the module of an 8-bit adder/subtractor circuit.

```
//-------------------------------------------------------------------------------------
//  Concurrent Intermediate Signature Comparison (CIC)
//-------------------------------------------------------------------------------------
 module  CIC8  (res,clk,clrb,A,B,S,q,ErrorBit,x,Si) ;
   input  Si,A,B,S,clk,clrb;
   input  [7:0] x , [7:0] resp;    //responses to be compacted, x for polynomail
   output error_bit , Fback, [7:0] q;
   output [7:0] RegB,  [7:0] aIn, [7:0] bIn, [7:0] cIn ;
 // ----------   controlling feedback
    RegB  <=  q   &  x ;
    Fback = RegB[7] ^ RegB[6] ^RegB[5]^RegB[4] ....RegB[0];
 //-------------------------------------------------------------------
   always @(posedge clk or  posedge clrb)
   if  (clrb)   RegB <=1;
   else    RegB  <= {Fback, RegB[6:0]};
 //-------------------------------------------------------------------
   aIn  <=  res[7:0]  &   8{A}
```

```
    not  (ab,A);   or (B1,ab,B);
    bIn  <= {q[0],q[7],q[6],q[5],q[4],q3],q[2],q[1] } &  8{B1} ;
    xor  #0 (last_rq,res[7],q[0]);
     cIn  <=  aIn  ^  bIn;
    or  (cl,A,B,S) ;
    and   (clock,clk,cl);
     ErrorBit = cIn[6] | cIn[5] | cIn[4] | cIn[3] | cIn[2] | cIn[1] | cIn[0] |  { res[7] ^q[0]};
    flop  fo (cIn[0],clock,clrb,q[0],) ,  f1 (cIn[1],clock,clrb,q[1],),
         f2 (cIn[2],clock,clrb,q[2],) ,  f3 (cIn[3],clock,clrb,q[3],),
         f4 (cIn[4],clock,clrb,q[4],) ,  f5 (cIn[5],clock,clrb,q[5],),
         f6 (cIn[6],clock,clrb,q[6],) ,  f7 (cIn[7],clock,clrb,q[7],);
    endmodule
//-------------------------------------------------------------------------------------------------------------
```

Figure 4. A Verilog module of an 8_bit CIC register

The test template provided to the students has two objects of the CIC register for generating input test vectors and the student can change their setting of the feedback configuration. In this mode, the student has to specify the initial state, to set up the feedback structure, and to specify the length of the test sequence. In the test generation mode, the student can choose a target fault and run the test for different purposes.

3 Solving an Assignment

The basic idea of our test approach is the use of the modified linear feedback shift register, namely, the CIC register for both, as patterns generation, as signature register, and as a concurrent comparator. In the context of this paper, we use the proposed test technique for other reasons. Most importantly, it is very simple to write a Verilog code or a testbench circuit based on CICs. Also, there is no need to write specific input stimuli for each assignment. Instead, the CIC generators and CIC signature registers are initialized with suitable seed values and the simulation is run for a predefined number of clock cycles. The proposed approach ensure that the checking of the partial signature is done within the test process, and there is no need for further comparison of the final signature with the fault free one.

As an example, the Verilog test template used for the self test of a simple adder/subtructor circuit is shown in Figure 5. It consists of the CIC based generators and a CIC based signature register. The generator and signature registers are connected to the CUT, as well as clock and reset signals. In order to guarantee correct interfaces for the circuits, the CIC

design template, a description of the assignment, and an empty (interface only) circuit are provided to the students, who then have to design and test the missing circuit logic.

```
// ------------------------------------------------------------
//AddSub.v    8-bit  adder/subtractor
//------------------------------------------------------------
  module addsub8 (Ain,Bin,sum,Cout,Sign);
    input Sign;
    input [7:0] Ain, Bin;
    output [7:0] sum;
    output Cout;
      reg [7:0] sum;
      reg [7:0]  ExOrB;
      reg Cout;
  always @(Ain, Bin, Sign)
     begin              // if Sign=0 then A+B, else  A-B
       ExOrB = Bin ^ { 8 {Sign}} ;
       {Cout,sum} = Ain + ExOrB + Sign;
     end
 endmodule          // addsub
//------------------------------------------------------------
```

Figure 5. A module for an 8-bit adder/subtractor

3.1 Personalized Assignments

After solving the assignment by designing the circuit, the student updating the Verilog test template code received from the teacher by inserting the code of his logic between the test pattern generators and the signature register can be and start the test process. The proposed CIC register design can be made to work as a random test patterns or as a signature register with immediate comparator. Having no any faulty check point, the circuit is working correctly. The student can now submit the solution (the design files) to the faculty server. By using different initial seed values for the CIC registers, the teacher is able to personalize the exercise for each student . For a simple circuit like the adder presented above, the main advantage of the self test is the immediate feedback to the student whether the design is already correct. However, for larger designs and especially for sequential circuits with complex state machines, where a variety of solutions is possible, the self test also greatly reduces the work required to check the students' solutions (and it still provides feedback to the student). The Verilog

module to run a test iterations to check for the CPs of the adder circuit is shown in Figure 6.

```
// ------------------------------------------------------------------------------------------------------
//  Test Template with three CICs registers for the test patterns
//  generation and the signature compaction and comparison
//------------------------------------------------------------------------------------------------------
module TestTemplate
 reg   clr, clk, IN;
 reg   [7:0] misr_in, data1, data2;
 reg   [3:1] A, B, S,   [7:1] x1, x2, x3 ;

wire error, err1, err2, err3 , [7:0] q1, q2, q3, sum ;
integer  i, j ;
  parameter  iteration, = 4000 ;
  event  start_of_test,  start_load ;
CIC8   cic    (sum,clk,clr,A[3],B[3], S[3],q3,err3,x3,err2) ,
       prpg1  (data1,clk,clr,A[1],B[1], S[1],q1,err1,x1,err3) ,
       prpg2   (data2,clk,clr,A[2],B[2], S[2],q2,err2,x2,IN);
flop ff (err1,clk,clr,error,);
//-------------------------------------------------------------------------
 //     addsub8  abc (q1,q2,sum,'b0, );     solution to be provided by students here
//-------------------------------------------------------------------------
 initial
     begin
        clr = 1; clk = 1; IN = 0;  i=0;
        A = 'b000;   B = 'b000;  S = 'b001;   misr_in = 'b 10001011 ;
         for( j=0 ;j < 8; j=j+1)
           begin       @(posedge clk)    IN = misr_in[j];      end
// initialization of prpg1
       #0  A = 'b000;  B = 'b000;  S = 'b100;  data1 = 'b 01010001 ;
       for( j=0 ;j < 8; j=j+1)
           begin     @(posedge clk)    IN = data1[j];      end
// initialization of  prpg2
        #0  A = 'b000; B = 'b000; S = 'b010  data2 = 'b 11001011 ;
       for( j=0 ;j < 8; j=j+1)
          begin      @(posedge clk)      IN = data2[j];     end
  end
initial
     begin
       #0  x1 = 'b 0110000;  x2 = 'b 0101001;  x3 = 'b 0001010;
       #0  data1 = 'b 01010001; data2 = 'b 11001011; misr_in = 'b10001011;
       #strobe("NOTE :\n {p1:prpg1  m:misr   xor : feedback ……..
       #strobe ("\n S= %b  \n  p1=%d[%b]   p2 =%d[%b] m3=%d[%b] ……
       #strobe(" xor1=%d[%b]   xor2=%d[%b]  xor3 = %d[%b] …….
```

```
    #strobe("\n iter(clocks)   time  P1  P2  ……..
    end
  always #10  clk=~clk;    // always  #10  misr_in = sum = q1 + q2;
  always  @(start_of_test)
  begin        A = 'b001;  B = 'b111;  S = 'b001;      end
 always @(posedge clk)
  begin
     i = i +1;      if (i > iteration)    $finish;
    if (1)
      begin  $strobe(" %d %d  ……[%b]%d   %d",I,$time,q1,q1,q2….        end
      if  (i==25)    $strobe("  …..  STARTING  ….. ");
  end
 initial
  begin       #500  ->  start_of_test;       end
endmodule
//-------------------------------------------------------------------------------------------------
```

Figure 6. Test template which can be accessed by students

3.2 More Faults Checking

Because the simulator uses a discrete-event based simulation engine, the simulation includes gate-delays and allows timing checks on the circuits. For example, one of the homework assignments asks the students to design a carry-select adder from given 8-bit adder and multiplexer blocks. Therefore, the correct solution to the exercise does only differ in the timing, but not the functional behavior, from a simpler ripple-carry architecture. However, by running the self test at a clock-speed that exposes the gate-delays, we can easily detect the slower (and wrong) design, without the need for additional timing checks or intricate waveform comparisons.

4 Alarm Check points ACPs

We have observed that the compaction of a faulty solution yields many instants similar to that of the basic check points of the original fault free circuit but occur at different time instants. These instants belong only to the faulty circuit. We call these instances as Alarm check points *(ACPs)*. An ACP occurs at time instant ***i*** if $F(r^*_i) \neq G_{i+1}$ *and* $F(r_i) = S_{i+1}$

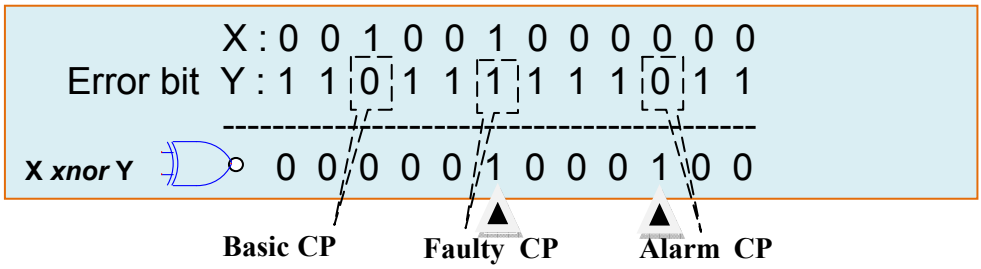

Figure 7. Concept of *Faulty* and *Alarm* check point

While the set of the basic CPs for a particular design can be determined by simulating the original circuit, no simulations are required to determine the Alarm Check Points. Testing of a faulty designs has shown that even a single error response compacted into the CIC register will generate an adequate number of such Alarm check points. We show that ACPs have many superior properties over basic CPs. A single ACP is sufficient to declare a faulty design prepared by the student.

Alarm check point can be easily detected by a simple modification to the procedure for detecting basic check point, namely, replacing the ***AND*** condition by a logical ***XNOR*** function as shown in Figure 7. The merits of the Alarm check point concept are high. Any single ACP represents a defect in the student design. The total number of the ACPs and their time instances can be used for the diagnosis of the actual type of the fault (stuck_at , delay, etc) imposed in the circuit design.

Table 1. The number of check points for faulty circuits (*stuck_at_0*)

CUT	*Fault_Free Circuit*			*Faulty circuit (stuck at 0)*		
	BCPs	*FCPs*	*ACPs*	*BCPs*	*FCPs*	*ACPs*
AddSub	*42*	**0**	**0**	*42*	**3**	**4**
DiffEq	*25*	**0**	**0**	*25*	**6**	**2**
CRT	*39*	**0**	**0**	*39*	**3**	**2**
ARF	*12*	**0**	**0**	*12*	**32**	**9**
Expon	*18*	**0**	**0**	*18*	**14**	**7**

Table 1 shows the number of the basic check points (BCPs), and faulty check points (FCPs) as well as the Alarm check points (ACPs) for some benchmarks circuits. The faulty free circuit was infected by a *stuck_at_zero* fault in a wire. There are sufficient amount of the fault check points and the Alarm check points.

Any fault in the CUT can always be detected if the compaction of the test responses of the CUT generates at least one Alarm check point. The percentage of the total faults detected for the CUT, (fault coverage FC), can be computed as

$$FC = \frac{Number\ of\ detectable\ faults}{Total\ number\ of\ faults} \times\ 100\%$$

Table 2 shows the percentage of fault coverage of *stuck_at _0/1* faults at the 32 primary inputs of the 8-bit adder/subtractor for different characteristic polynomials and with 3000 test patterns. FC_B, FC_A, and FC_T are the fault coverage due to the Basic check points, Alarm check points and due to both, respectively.

Table 2. Fault coverage of stuck_at faults in the 32 primary inputs of 8-bit adder/subtractor. (0,1,4,8) = $1+x+x^4+x^8$

Ch/s Polys	*Fault Coverage %*			
	BCPs	*% FC_B*	*% FC_A*	*% FC_T*
0,1,2,7,8	32	81.25	96	100
0,1,3,5,8	21	93.75	84.3	93.75
0,3,7,8	18	68.75	87.5	93.75
0,2,4,5,8	27	93.75	100	100

5 Evaluation

For a given b_bit CIC register, a CUT can have up to 2^b different sets of Basic check points. Since there are 2^b possible seeds for compacting a *b*-bit responses, each circuit can have also up to 2^b CPs for a given function $\boldsymbol{F_x}$. Therefore, each CUT can have up to $\boldsymbol{2^{2b}}$ different sets of the CPs.

Table 3 shows the simulated number of CPs for some for selected High-level synthesis benchmarks when tested for 4000 test patterns and different functions of the signature register. The benchmark circuits include the differential equation *DiffEq*, the Chinese Reminder Theorem for integers CRT, the AR filter *ARF*, and a circuit *Expon* to compute the exponential ex. It is clear that for test with ***n*** test patterns, there exists an initial seed for the CIC register, which leads to at least $\boldsymbol{n/2^b}$ CPs irrespective of the characteristic polynomial and the function.

Table 3 : The check points for selected benchmarks

CUTs	Number of Basic Check Points	
	Function F_0	Function F_{32}
AddSub	*42*	*16*
DiffEq	*25*	*20*
CRT	*39*	*18*
ARF	*12*	*6*
Expon	*18*	*35*

Figure 8 gives the number of basic check points obtained through simulation of 8-bit adder/subtractor circuit for different functions, namely, F_0, F_{22}, and F_{70} and for characteristic polynomial $1+x+x^4+x^8$ in 8-bit CIC register when the test patterns length was varied from 1 to 5000. Figure 9 gives the number of basic check points obtained through simulation of 8-it adder/subtractor circuit for different characteristic polynomials, namely, P1 = (0,2,4,8), and P2 = (0,1,2,6,8) in 8-bit CIC register when the test patterns length was varied from 1 to 5000.

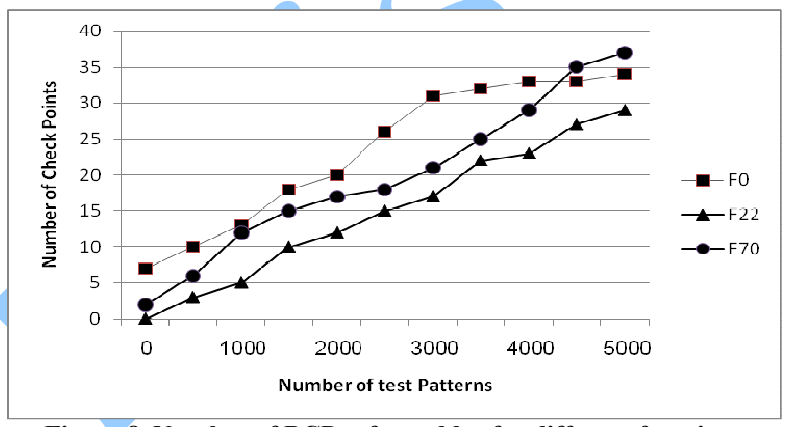

Figure 8. Number of BCPs of an adder for different functions

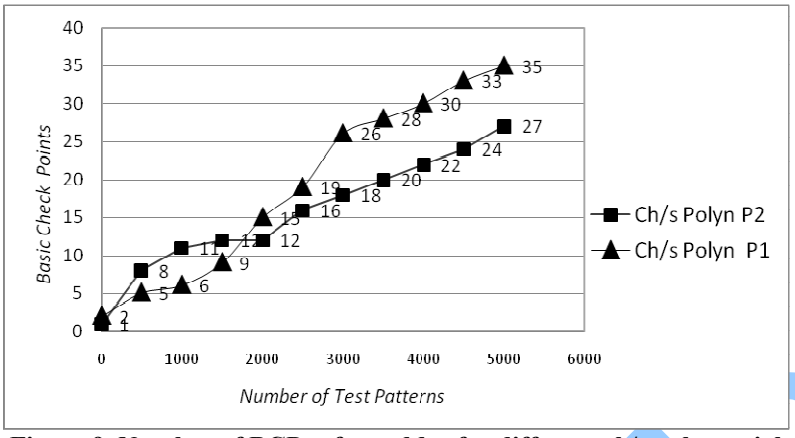

Figure 9. Number of BCPs of an adder for different ch/s polynomial

The proposed technique can improve the logic circuit assignments checking process through:

- Student approach constructive tasks like designing and testing digital circuits, and hence learn essential skills in DFT and BIST for their professional life.
- Checking of the Alarm CPs and intermediate signatures ensure the quick response to faulty circuit and reduce the turnaround time.
- Comparing multiple signature throughout the testing process ensure less aliasing probability,
- Also the register can be run at very high clock rates which enables a full speed test of the CUT, and also enable testing the delay faults as well,
- Ensure high percentage of fault coverage when combining both the approaches the Basic check points and the Alarm check points.

Conclusion

In this paper, we presented and implemented our vision to solve some problems in the distance teaching system of the circuit design courses. The new conception presented here allows to improve the skills of students to be educated for digital hardware and SoC related topics. We described an alternate and robust method for checking the student designs based on the notion of Check Points (CPs). Our approach relies on First, observing a finite number of CPs, which get generated during the course of testing, and Second, tracing the alarm check points in the defected design.

A benefit to our approach is that because of these two techniques, the detection of the faulty design becomes more reliable, faster, and requires less space. Students can use the same test template for training purposes. They can insert different possible faults, and watch how the faults change the circuit's behavior at different input patterns, how the test patterns can be generated to detect a given fault, or how the faults can be localized by test patterns.

The system is also more effective for large designs such as sequential circuits with complex state machines, where variety of solutions is possible, the system proposed greatly reduces the work required to check the students' solutions, and it also provide a good feedback to the student. Furthermore, our approach aims at considerably speeding up the duration of the correction process of designing assignments for the logic courses with large number of students especially, in distance education. We have verified the desired properties using selected High-level synthesis benchmarks.

References

[Bus99] **M.L. Bushnell** - *Increasing Test Coverage in a VLSI Design Course*, International Test Conference, Atlantic City, NJ, USA, 1999.

[B+99] **J. Brunsmann, A. Homrighausen, H.-W. Six, J. Voss** - *Assignments in a Virtual University The WebAssign-System*, Proc. 19th World Conference on Open Learning and Distance Education, Vienna, 1999

[Hen00] **N. Hendrich** - *A Java-based framework for simulation and teaching: Hades*, in Microelectronis Education, Proc. EWME-2000, 285-288, 2000

[HC03] **H. Han-Pang, L. Chiou-Hwa** - *Java-Based Distance Learning Environment for Electronic Instruments*, IEEE Transaction on Education, vol. 46,no. 1, FEB 2003

[KMZ79] **B. Koenemann, J. Mucha, G. Zwiehoff** - *Built-in Logic Block Observation Techniques (BILBO)*", Proc. International Test Conference, 37-41, 1979

[Lal97] **P. K. Lala** - *Digital circuit testing and testability*, Academic press, 1997

[Mic07] **T. Michael** - *Integrated Online Courseware for Computer Science Courses*, ITiCSE'07: Conference on Innovation and technology in computer science education, Dundee, Scotland, June 25-27, 2007

[Nor02] **H. Norman** - *Automatic Checking of Students, Designs Using Built-In Self Test Method*, Proceedings of the 4th European Workshop on Microelectronics Education, EWME 2002, Baiona, 23-24 May 2002

[VPM04] **M. Vladimir, M. Plamen, L. Mihail** - *Tutoring Tool for Logical Schemes Design*, International Conference on Computer Systems and Technologies – CompSysTech 2004

[UW00] **R. Ubar, H.-D. Wuttke** - *Action Based Learning System For Teaching Digital Electronics And Test*, 3rd European Workshop on Microelectronics Education (EWME 2000), Aix-en-Provence, Kluwer Academic Publishers, May 18-19, pp. 107-110,2000

[***07] http://moodle.org/, Jan 15, 2007