New Hybrid-Based Self-Test Strategy for Faulty Modules of Complex Microcontroller Systems

Mohamed H. El-Mahlawy, Sherif Hussein, and Gouda I. Mohamed

***Abstract*—In this paper, a new hybrid test strategy, called hybrid-based self-test (HYBST), is presented to test complex digital circuits such as microcontrollers. This test strategy integrates the signature multi-mode hardware-based self-test (SM-HBST) with the software-based self-test (SBST). In this test strategy, the microcontroller is divided into a number of main modules, and then test subroutines are used to functionally test each module, based on its instruction set architecture (ISA). The ISA is used to generate test subroutines that represent test pattern generators (TPGs) and part of the test controller. The SM-HBST represents the other part of the test controller and the test response compaction (TRC). The experimental results illustrate the superiority of the HYBST in the memory utilization, test application time, testing of internal modules of the microcontroller, and testing of general-purpose input-output (GPIO) pins of the microcontroller. In addition, an integrated test solution for fault diagnosis of the circuit boards including random logic integrated circuits (ICs) and microcontroller chips is presented to indicate a real practical test strategy.**

***Index Terms*—Testing of digital circuits; Built-In Self-test for digital circuits; Testing of microcontroller circuits; Software-based self-test; Hardware-based self-test; Hybrid-based self-test.**

Original Research Paper
DOI: 10.7251/ELS1822059E

I. Introduction

With the advent of complex integrated circuits (ICs), the stream data of bits at the available test points of the printed circuit boards (PCBs) or the complex ICs on the circuit under test (CUT) become very complex. Testing the CUT for correct operation after manufacturing is an important issue. It is to apply proper test patterns to the CUT, and the test response generated from the CUT is analyzed to locate the faulty source components. Several testing approaches achieve this objective. Most of them fall into two main categories: in-circuit testing (ICT) and functional testing [1-5]. The ICT requires a costly special kind of the test fixture (bed of nails) [2]-[3], [6]. The functional tester verifies that the CUT board performs the functions it was designed for. Only CUT inputs and CUT outputs need to be tested with inexpensive test fixture (edge connector) [5], [7]-[8].

Microcontrollers are considered an important part of the electronic system. The complexity of microcontrollers with poor accessibility makes their test process a difficult task using external automatic test equipment (ATE) [2]-[3], [6]. Therefore, empowering chip to test itself looks the suitable solution for the microcontroller testing. The built-in self-test (BIST), considered a mechanism of the hardware based self-test (HBST), provides significant advantages not only for processor module but also for other peripherals found in the microcontroller [6]. [8]-[9]. The BIST adds special hardware overhead to the circuit design in the chip level, the board level, and the system level to realize self-test operations. This hardware overhead is the test pattern generator (TPG), the test response compactor (TRC), and the BIST controller. The required test patterns generated from the TPG are applied to the CUT board, and the test responses are compacted using the TRC for fault diagnosis so that the CUT board can be replaced and returned to the service [10]-[11].

The easiest way to test a small circuit is to apply all possible test patterns, called exhaustive testing. This testing is not practical for large circuits [1], [12]-[13]. The more common approach for testing is to use computer algorithms for automatic test pattern generation (ATPG) [2]-[3], [5], [14]-[15]. These algorithms are effective at finding sequences of test patterns that can detect all detectable hardware faults [14]-[15]. Pseudorandom testing is widely used in testing of digital circuits, whose test patterns can be generated by simple hardware circuits [1]-[3], [5]. Signature analysis is a TRC technique that detects errors in stream data of bits, caused by hardware faults. It compacts the test response for each output node of the CUT board into a signature [4], [6], [9]-[11]. After all required input test patterns are applied, the reference (good) signature is generated. This signature is compared with the corresponding measured signature. When they are different, a fault is detected. The signature analyzer (SA) requires the storage of fewer bits. The aliasing probability of an n-stage signature analyzer approaches 2^{-n} [16].

The signature analysis performs the hot functional test for conventional digital random logic ICs, and memory devices [9]. By determining a unique signature for each node in the CUT board, the fault detection and then fault location (fault diagnosis) can be achieved. In addition, another test strategy was presented to functionally test single-shot (SS) circuit on the PCB [17]. It can test the SS circuit by measuring the time duration. The time

Manuscript received 18 September 2017. Received in revised form 21 February 2018, 4 April 2018, and 28 July 2018. Accepted for publication 16 August 2018.

Mohamed H. El-Mahlawy is with Electrical Engineering department, faculty of Engineering and Technology, Future University in Egypt (FUE), Cairo, Egypt (corresponding author: +201121455800; email: mohamed.elmahlawy@fue.edu.eg).

Sherif Hussein is with Computers department, military technical college, Cairo, Egypt (email: sherif.i.morsy@gmail.com)

Gouda I. Mohamed is with Computers department, military technical college, Cairo, Egypt (e-mail: gisalama@mtc.edu.eg).

duration is considered the signature of its proper functionality. The HBST is limited to properly test digital CUT including a microcontroller chip. Adding hardware parts of that HBST have a negative impact on the circuit area and performance degradation. The alternative to the HBST is the software-based self-test (SBST) that promises an attractive and non-intrusive test solution for embedded systems [18].

In the SBST strategy, no extra test hardware is required. Fig. 1 illustrates the concept of the embedded SBST strategy, where the test program is resided in the flash memory of a microcontroller. During the application of the testing, the on-chip test generation program emulates a TPG to generate required test patterns, applied to main modules of a microcontroller. In addition, the on-chip test application program collects the test response and stores them in the memory. The stored test response is compacted into a signature using the TRC program. Test response can later be unloaded and analyzed by an external ATE. At the final stage, the external ATE will provide a decision about the microcontroller under test.

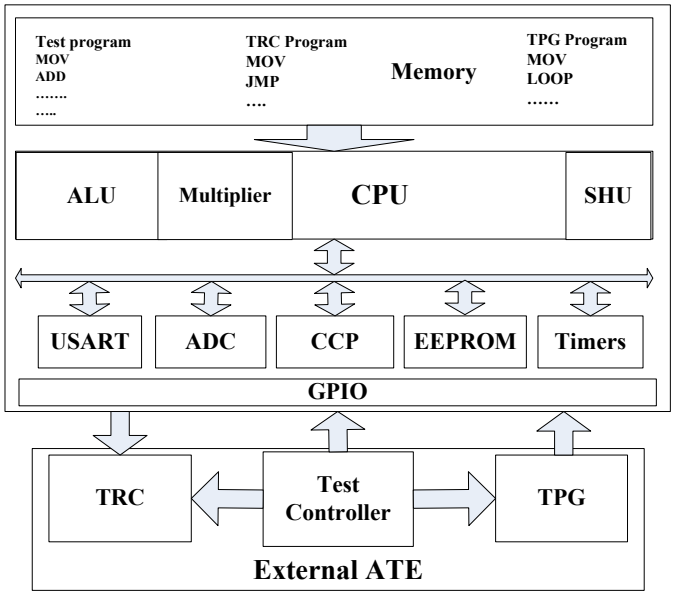


Fig. 1. Block diagram of the SBST strategy.

The SBST strategy is classified in two different categories. The first category is functional in nature [19]. The functional SBST strategy is either based on functional fault models [20]-[23] or based on the checking experiment principle without assuming any fault model [24]. It was found that these approaches are not exactly suited for embedded processor cores and that they achieve low fault coverage. The second category is structural in nature and requires a structural fault driven test development.

L. Chen [25] presented a structural SBST strategy that targets processor components using pseudo-random pattern sequences. This strategy is not considered the regular structure of critical components within a processor and hence leads to large self-test code, large memory requirements, and excessive test application time, even when applied to a small processor model. N. Kranitis [26] presented a structural SBST strategy for testing a processor in an embedded system, based on the divide-and-conquer strategy and the instruction set architecture (ISA) of the processor. For every component and its operations within the processor, deterministic test patterns are generated to detect structural faults. In addition, N. Kranitis [27] introduced a hybrid-SBST strategy that combines deterministic test patterns and random test patterns to test commercial processor cores.

The objectives of an effective SBST strategy are to increase fault coverage, reduce self-test code, reduce memory utilization, and reduce test application time. Therefore, the structural SBST strategy is more efficient than the functional SBST strategy in terms of fault coverage without system modifications, the size of the self-test code, memory utilization, and test application time [18], [26]-[28].

Most of recent researches utilize the SBST strategy for embedded processors and large microprocessors. However, the researches for testing of microcontrollers with small memories, used in industrial applications, are limited. N. Kranitis [29] proposed a low-cost SBST methodology for *Reduced Instruction Set Computer* (RISC) processor cores with the aim of producing small test code sequences. It is based on two phases to test the functional modules such as the *register file*, the *arithmetic-logic unit* (ALU), the *shifter*, and the *multiplier* in the first phase. Then, the control and hidden modules are tested in the second phase. In these phases, a few deterministic test patterns (not ATPG) are used to utilize small test application time. This methodology achieves 92% of the *single stuck-at* fault coverage on the architecture of the Plasma/MIPS model using the fault simulator. This test strategy, proposed in [29], is limited only to test functional modules of the processor with high fault coverage. However, the control and hidden models are tested with low fault coverage. This proposal cannot test microcontrollers with small memories, used in industrial applications.

Dattatraya [30] proposed an expert work for fault diagnosis of the Philips 89v52RD2 microcontroller. He proposed checking experiments for every fault, based on the intelligent diagnostic assessment and management of testing process. The knowledge base consists of a procedural description of the test, and uses the knowledge about troubleshooting process. The results obtained are validated using experts and testing equipment under the same input patterns for automation in knowledge acquisition and updating process. This strategy is limited to test a microcontroller with high fault coverage, and needs huge alteration to get the proper knowledge base. In addition, the authors in [30] did not illustrate the memory utilization, test application time, and the fault coverage for the Philips 89v52RD2 microcontroller under test. Therefore, the authors did not prove the applicability of this test strategy.

The authors in this paper implemented the SBST strategy to test two different families of the PIC microcontrollers (PIC16F87X – PIC18F4X2) [6], [31]. It was found that the SBST needs large space of memory for the instruction set code that emulates the TPG, the TRC, and the test controller to realize self-test operations. In addition, the SBST strategy cannot test most modules in the microcontroller like timers, general-purpose input-output (GPIO) pins and Capture/ Compare/PWM (CCP)

modules without the external ATE. Therefore, the SBST strategy is limited to test microcontrollers with small memories. The necessity to introduce a test strategy for testing microcontrollers with small memories is highly required.

In this paper, the new test strategy for testing microcontrollers, called hybrid-based self-test (HYBST), is presented. It integrates the HBST strategy and the SBST strategy. Due to the diversity of the digital CUT boards in the practical field, the signature multi-mode hardware-based self-test (SM-HBST) strategy is highly required. Therefore, the HYBST strategy integrates the SM-HBST strategy and the SBST strategy. Based on divide-and-conquer strategy, the microcontroller is structurally divided into a number of main modules and test subroutines are constructed to exhaustively test each of these modules. Generation of these test subroutines requires knowledge of the ISA of the microcontroller. The exhaustive testing guarantees the detection of all detectable combinational faults, detected by single-pattern test generator [1], [12]-[13]. This leads to achieve high fault coverage without performance degradation and without fault simulator. Test subroutines, embedded in the microcontroller memory, generate test patterns for each module in a microcontroller chip, and the test response is then propagated to GPIO pins of a microcontroller chip to be compacted by the external SM-HBST. Test subroutines are used to emulate TPG and part of the emulated test controller, running in the microcontroller itself. The SM-HBST outside a microcontroller chip represents the other part of the test controller and the TRC. To realize a real practical test strategy, the merging of the presented HYBST strategy with the developed software part of the SM-HBST is applied as a fault diagnosis solution for electronic digital boards that contains both conventional random logic ICs and a microcontroller chip. It is evaluated on two different families of Microchip microcontrollers; PIC18F4X2 and PIC16F87X [31].

This paper is organized into six sections. The presented section introduces the previous published works. Section II describes the basic concept of the microcontroller test strategy. Section III describes the design and implementation of the SM-HBST strategy. Section IV presents testing of microcontroller modules, and states the comparisons between the HYBST strategy and the SBST strategy using two different families of microcontrollers. Section V presents fault diagnosis of the digital circuit board including random logic ICs and a microcontroller chip, and then section VI illustrates the experimental results of the whole test strategy. Finally, the last section concludes the presented paper.

II. Basic Concept on the Microcontroller Test Strategy

The microprocessor, considered a powerful computing component, may use other components such as memory, timers, and communication peripheral environment. On the other hand, the microcontroller is designed to include the microprocessor and its components in a single integrated circuit. Microcontrollers are popular with industrial developers. The increasing logic-to-pin ratio of the microcontroller poses very serious problems in testing at the board level. These problems lead to an increasingly long test pattern generation, long test application time, and low fault coverage. In addition, the stream data bits at the available test points of the CUT board are large so detecting a hardware fault becomes difficult as well as locating the source faulty node (nodes). The SM-HBST strategy is limited for complex digital circuits such as microcontrollers that have heterogeneous components with poor accessibility.

In this paper, the comparison criteria of testing performance between different test strategies are based on the following:

1) Memory utilization (Data memory – Flash memory) is considered to reduce hardware overhead and to leave the largest space of the available memory for the application program of that microcontroller.
2) Test application time (required number of clock cycles to finish the test).
3) Testability of microcontroller modules reflects the percentage of fault coverage. When the number of tested modules increases, the fault coverage increases.

The SBST strategy cannot test all internal microcontroller modules especially timers, GPIO pins and CCP module. If the SBST strategy can test other modules in a microcontroller, then it will need an external ATE to load measured signatures from microcontroller memory for fault detection. The SBST strategy cannot be applied to test microcontrollers with small memories because it needs large space of memory for the software code to emulate TPG, TRC and test controller of the BIST system. To enable this test strategy to test digital CUT boards including a microcontroller chip, more effort is needed to achieve the criteria of the testing performance, and another test strategy is required to handle this challenge.

In this paper, a new test strategy that combines both the SM-HBST strategy and the SBST strategy is called the hybrid-based self-test (HYBST) test strategy. The HYBST divides the test operation between a microcontroller chip and the SM-HBST. The emulated TPG and part of the emulated test controller are running in the microcontroller itself using test subroutines. In addition, the TRC and the other part of the test controller are running in the SM-HBST outside the microcontroller. Fig. 2 illustrates the block diagram of the HYBST strategy.

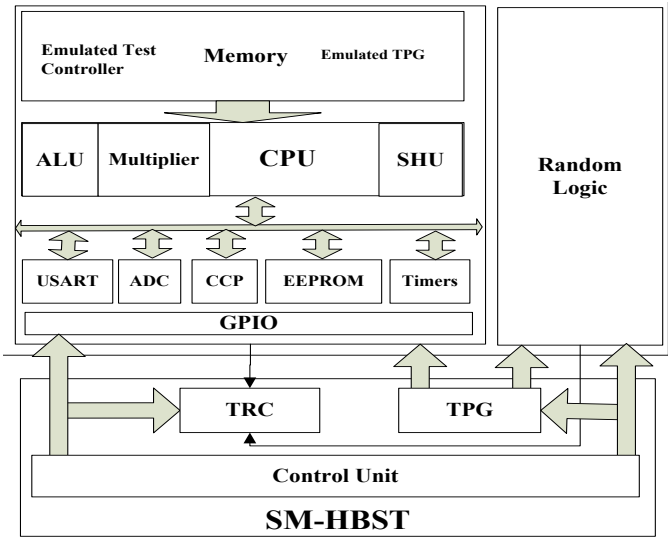


Fig. 2. Block diagram of the HYBST strategy.

Test development of the HYBST is divided into four main phases to construct test subroutines of a microcontroller. The first phase includes information extraction and modules identification of a microcontroller. The second phase is the instruction selection strategy depending on observability and controllability. The third phase is the operand selection, and the last one is the test routine development.

A. Information extraction phase

The information extraction phase shows the features of a microcontroller. According to the divide-and-conquer strategy, the microcontroller is divided into main modules. These modules are the processor, the memory, timers, the pulse width modulation (PWM), GPIO pins, the universal synchronous asynchronous receiver transmitter (USART) and the analog-to-digital converter (ADC). The information on every module is collected to be effectively tested. The ADC module is tested by [32]-[34], and it is not presented in this research. The memory of the microcontroller can be divided into the random access memory (RAM), the electrically erasable programmable read only memory (EEPROM), and the flash memory. In addition, the processor can be divided into an ALU and a multiplier. The effectiveness of this phase is evaluated on certain microcontroller families (PIC16F87X – PIC18F4X2). Table I presents the key features of two microcontrollers.

Table I
Key Features of the Microcontroller

Key Features	PIC16F877	PIC 18F452
FLASH Memory (14-bit Word)	8k Word	16k Word
Data Memory	368 Byte	1536 Byte
EEPROM (Byte)	256	256
I/O Ports	5 I/O Port	5 I/O Port
Timers	3	4
Capture/Compare/PWM Modules	2	2
Serial Communication	USART	USART
Multiplier	-	8 × 8
ISA (instruction set architecture)	35 instruction	75 instruction
Operating speed in Million instruction per second (MIPS)	5 MIPS	10-15 MIPS

B. *Instruction selection phase*

Based on the ISA of the microcontroller [31], it is found that every module M performs a set of operations O_M. $I_{M,O}$ is denoted to the set of microcontroller instructions that, during execution, enable the same control signals and cause module M to perform operation O. It is evident that, for each module M, there is at least one microcontroller instruction that, during its execution, causes module M to perform operation O, i.e. $I_{M,O} \neq \emptyset$. After identification of the set $I_{M,O}$ for every module operation, an instruction I was selected from the set $I_{M,O}$.

These instructions, which belong to the same set $I_{M,O}$:

1) Have different observability properties. When operation O is performed, the outputs of module M conduct internal microcontroller registers with different observability characteristics.
2) Have different controllability properties. When operation O is performed, the internal microcontroller registers with different controllability characteristics conduct the inputs of module M.

After identification of the set $I_{M,O}$, select an instruction $I \in I_{M,O}$ according to the following criteria:

Criterion 1: Discard instructions $\in I_{M,O}$ that when operation O is performed, the outputs of module M do not propagate to internal registers of the microcontroller. This means that the faulty component output cannot be propagated. For example, instructions *CALL* (call subroutine), *RETFIE* (return from interrupt), *RETLW* (Return with literal in *W* register (accumulator)) and *SLEEP* (go into standby mode) don't propagate to internal microcontroller registers.

Criterion 2: Between instructions I_A and $I_B \in I_{M,O}$, if I_A requires a smaller instruction sequence to propagate the outputs of module M to GPIO pins, I_A is ranked higher priority than I_B. It means that I_A is more easily observed than I_B, and it should be preferred over I_B. For example, instruction XORWF (Exclusive-OR *W* register and *f* (Register file address (0x00 to 0x7F))) is easily observed over XORLW (Exclusive-OR literal and *W* register) because it can be used to directly transfer the output to external ports.

Criterion 3: If instructions I_A and $I_B \in I_{M,O}$ have the same priority based on *criterion* 2, another criterion is required. Therefore, if I_A requires a smaller instruction sequence to generate a specific test pattern at the internal register of the microcontroller, I_A is ranked higher than I_B. For example, instruction INCF (Increment *f*) has higher priority over INCFSZ (Increment *f*, skip if zero) because it uses less clock cycles when it is executed.

At the end of this phase, test instructions are selected, based on the above three criteria. They are considered the main foundation of embedded test subroutines that test microcontroller modules. After the O_M and the $I_{M,O}$ set are identified, the number of these instructions are reduced. It should be noted that module M executes operation O during its instruction execution but the module outputs are not propagated to GPIO pins. It is not included in the $I_{M,O}$ set according to *Criterion* 1. If part of the test response of a module is not driven to a well accessible internal register (that is the case of flag outputs, driving status register or special function registers), an extra instruction sequence is required to propagate to accessible registers and then to GPIO pins. Table II and Table III illustrate the instruction reduction of the CPU module of microcontrollers (PIC18F452 – PIC16F877).

C. *Operand selection phase*

Operand selection phase chooses the appropriate test patterns to use it with test subroutines in order to get high fault coverage. The presented test strategy in this paper is based on exhaustive testing for test pattern generation to achieve high structural fault coverage for each module of the microcontroller. It detects all

detectable combinational faults detected by single-test pattern without using the fault simulator [1], [5], [6], [12]-[13].

TABLE II
INSTRUCTION REDUCTION OF THE CPU MODULE OF THE PIC16F877

Module *M*	Operation *O* that can be executed by this module	$I_{M,O}$ used to test this module according to *Criteria* 1, 2 & 3
CPU ALU	ADDWF, ANDWF, CLRF, CLRW, COMF, DECF DECFSZ, INCF, INCFSZ, IORWF, MOVF, MOVWF NOP, RLF, RRF, SUBWF SWAPF, XORWF, BCF, BSF, BTFSC, BTFSS, ADDLW, ANDLW, CALL, CLRWDT, GOTO, IORLW, MOVLW, RETFIE, RETLW, RETURN, SLEEP, SUBLW, XORLW	CLRF , CLRW, MOVLW, BCF, ADDWF, SUBWF, XORWF, IORWF, ANDWF, COMPF, DECF, INCF, MOVWF, MOVF, BSF, ADDLW, BTFSS, GOTO, RETURN
	35 instruction	19 instruction

The next sections present the fourth phase. It describes the design and implementation of the SM-HBST strategy, presented in section III. In addition, test subroutines that test internal modules of a microcontroller are presented in section IV. These test subroutines have been completely implemented using the previous SBST strategy with two different compaction techniques and the HYBST strategy. Test subroutines are developed for each of the microcontroller modules based on the above three criteria using both assembly and *C* programming languages.

TABLE III
INSTRUCTION REDUCTION OF THE CPU MODULE OF OF THE PIC18F452

Module *M*	Operation *O* that can be executed by this module	$I_{M,O}$ used to test this module according to *Criteria* 1, 2 & 3
CPU ALU	ADDWF, ADDWFC, ANDWF, CLRF, COMF, CPFSEQ, CPFSGT, CPFSLT DECF, DECFSZ, DCFSNZ, INCF, INCFSZ, INFSNZ IORWF, MOVF, MOVFF, MOVWF, NEGF, SETF, SUBFWB, SUBWF, SUBWFB, SWAPF, TSTFSZ XORWF, BCF, BSF, BTFSC BTFSS, BTG, BC, BN, BNC, BNN, BNOV, BNZ, BOV BRA, BZ, CALL, CLRWDT, DAW, GOTO, NOP, POP PUSH, RCALL, RESET, RETFIE, RETLW, RETURN SLEEP, ADDLW, ANDLW, IORLW, MOVLB, MOVLW, RETLW, SUBLW, XORLW, TBLRD*, TBLRD*+, TBLRD*-, TBLRD+*, TBLWT*, TBLWT*+, TBLWT*-, TBLWT+*	CLRF , MOVLW, MOVWF, XORWF, SUBWF, ANDWF, IORWF, ADDWF, COMPF, SWAPF, XORLW, ADDLW, ANDLW, SUBLW, IORLW, DECF, INCF, BSF, BNZ, RETURN
	68 instruction	20 instruction

III. DESIGN AND IMPLEMENTATION OF THE SM-HBST

In this section, the FPGA-based design and implementation of the SM-HBST for testing the digital CUT is presented. The SM-HBST is responsible of generating the required test patterns to the CUT through the TPG, compacting the test response from the target test node on the CUT through the TRC, and controlling the test cycle of the presented self-test strategy through the control unit (CU). The main block diagram of the SM-HBST, shown in Fig. 3, is composed of the TPG, the TRC, and the CU. In addition, the schematic diagram of the FPGA-based design is illustrated in Fig. 4. The SM-HBST has four main test modes; pseudorandom test mode (PRT mode), deterministic test mode (DET mode), hybrid test mode between the PRT and the DET mode (HPDT mode), and single-shot test mode (SS mode).

A. *Design of the PRT mode*

In the PRT mode, the TPG tests the digital CUT as a test stimulus. It stimulates all nodes in the CUT. The TPG, based on the PRT mode, is called pseudorandom TPG (PRTPG). The simplified block diagram that shows the basic test operation of a CUT in the PRT mode is shown in Fig. 5. Test patterns, generated from the PRTPG, are applied to the CUT and the test response is captured by the TRC every clock cycle. The TRC compacts all bits of a test response, generated from a stimulated node into a measured signature. The measured signatures are stored and compared to the reference (good) signatures for fault diagnosis. The TRC in the SM-HBST, shown in Fig. 3, has two blocks; the SA and the edge detection compactor (EDC). The SA is a linear feedback shift register (LFSR) as a compactor circuit [1], [6], [10]-[11]. The SA, designed in the SM-HBST, is the 23-bit whose primitive polynomial equals to $1 + x^5 + x^{23}$ with aliasing probability 0.000012 % (2^{-23}). Therefore, the probability of detecting error bits of a test response, clocked to the 23-stage SA, equals to 99.9999 %. The EDC will be discussed later in section *C*.

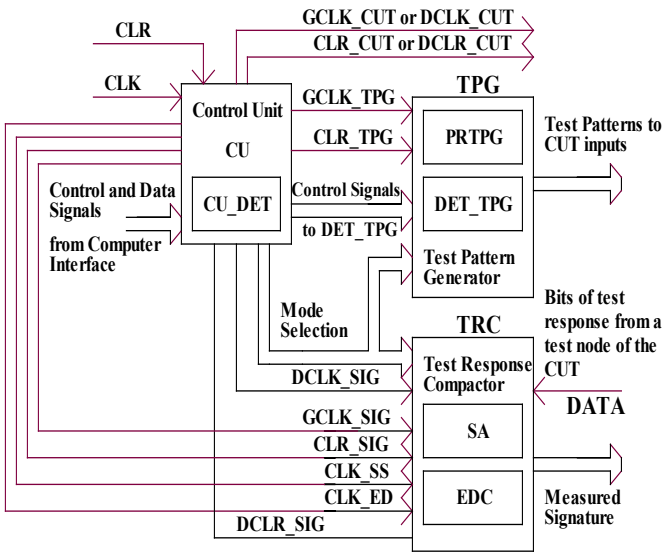


Fig. 3. Main block diagram of the SM-HBST.

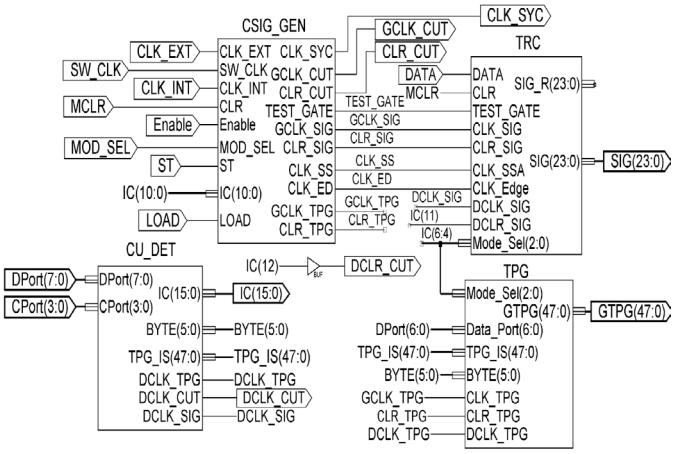


Fig. 4. Schematic diagram of the FPGA-based design of the SM-HBST.

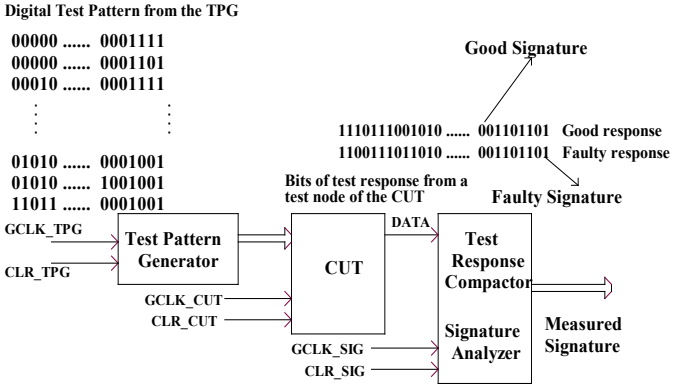


Fig. 5. Simplified block diagram of the test operation in the PRT mode.

The CU, shown in Fig. 3, is implemented with two modules *CSIG_GEN* and *CU_DET* in the FPGA-based design, shown in Fig. 4. Either the internal clock (CLK_INT) or the external clock (CLK_EXT) synchronizes the main clock of the module CSIG_GEN that controls each part of that test scheme. Both internal and external clock are utilized to synchronize the test operation. The selection of the clock is based on the control signal *SW_CLK* that switches between the *CLK_INT* and the *CLK_EXT*, shown in Fig. 4. The presented SM-HBST board unit has the TPG with forty-eight outputs that stimulate all nodes in the CUT with at most forty-eight inputs. To increase the test capability of large CUT inputs (more than forty-eight inputs), two or more SM-HBST board units can be used. The clock *CLK_SYC*, generated from module *CSIG_GEN* of the first SM_HBST board unit, is designed to synchronize the second SM_HBST board unit by feeding the *CLK_EXT*.

Fig. 6 shows the timing diagram of the test operation in the PRT mode. The required clocks and clear signals for the test operation are properly asserted. All clocks during the test gate are three-phase clocks, shown in Fig. 7. By applying known input test patterns from the bus *GTPG*(47:0) to the CUT, an unique signature can be generated at each node in the CUT. The PRTPG is clocked by *GCLK_TPG* and the SA is clocked by *GCLK_SIG*. The test patterns, generated from the PRTPG, are asserted at the rising edge of the *GCLK_TPG*, and the SA is asserted at the falling edge of the *GCLK_SIG*. The *GCLK_CUT* clocks the practical digital CUT either at the rising edge trigger or at the falling edge trigger. The required clear signals of each clock signal; *CLR_TPG*, *CLR_SIG*, and *CLR_CUT* are designed for proper timing operation to start the test gate as shown in Fig. 7(a). They are properly asserted at the starting of every test gate to provide the proper initialization of sequential circuits in the CUT. In addition, the closing of the test gate is shown in Fig. 7(b).

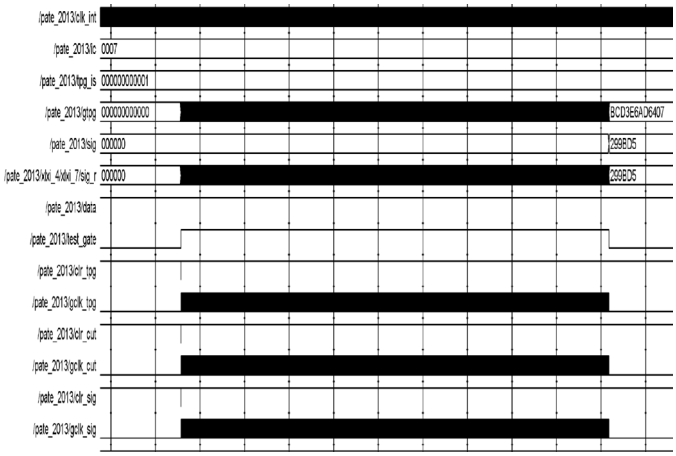
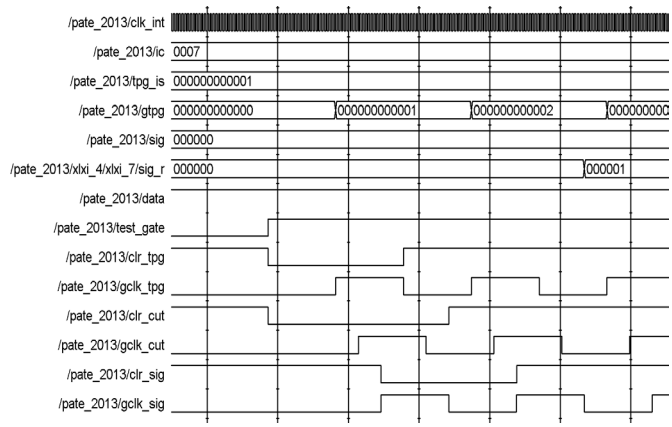


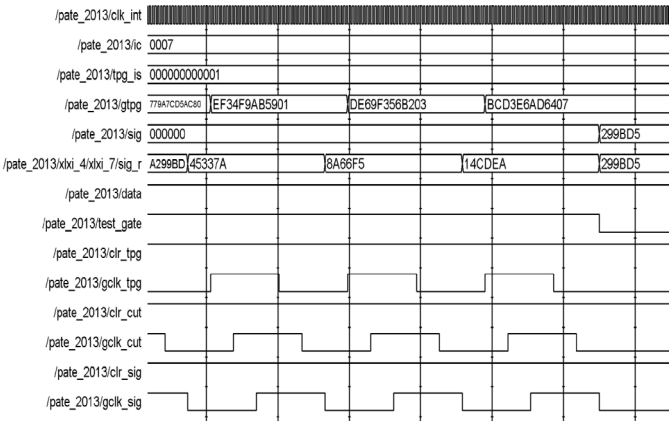
Fig. 6. Timing diagram of the test operation in the PRT mode.

The SA processes the output bits of a target node every clock cycle during the test gate interval. The test gate is controlled by the control signal *TEST_GATE*. After certain clock cycles, the test gate is closed, and the final signature is generated from the bus *SIG*(23:0). Three-phase clocks provide the test pattern enough time to propagate through the integrated circuits in the CUT either at the rising edge trigger or at the falling edge trigger of the *GCLK_CUT*, before the acquisition of the test response (received through the input signal, DATA, showed in Fig. 3, Fig. 4, and Fig. 5). The assertion of the proper timing ensures the stability of the signature generation for the proper test operation. The incorrect signature will accurately indicate an incorrect waveform of the target node as long as the error appears in the bits of the test response. The error appears only if the TPG generates the test patterns that detect hardware faults and the SA generates the incorrect signature. Data compaction is achieved by probing the target node asserted at the falling edge of *GCLK_SIG* during the *TEST_GATE*. The input binary sequence may be in different lengths but at the end of the test gate only the signature is the residue of the SA. The outputs of the SA has proper hexadecimal signature “299BD5” when the DATA sets HIGH.

The system has master clear (*MCLR*). The control signal, *Enable*, is set HIGH to enable the test operation. The switching between multiple opening and the single opening of the *TEST_GATE* is based on the control signal *MOD_SEL*. When the *MOD_SEL* sets LOW, the *TEST_GATE* is opened once to calculate a new signature, and the control signal *ST*, shown in Fig. 4, is asserted for every new signature. The multiple opening of *TEST_GATE* is used, when *MOD_SEL* sets HIGH. The module *CU_DET* in the *CU* produces the programmable code



(a) Starting test gate.



(b) Closing test gate.

Fig. 7. Timing waveforms of the starting and closing test gate interval.

IC(15:0) to configure the test scheme. It is considered the initial condition code to the target chip for proper test operation. Table IV illustrates the control code *IC*(3:0) to produce the test clocks and the phase delay between the three-phase clocks. Table V illustrates the control code *IC*(6:4) for the selection of the test operation modes. The control code *IC*(10:7) illustrates the required number of clock cycles inside the *TEST_GATE*.

Table IV
Generated frequency and the phase delay according to *IC*(3:0) or *N*

N	three-phase clocks		*Delay*	*N*	three-phase clocks		*Delay*
	Period	Frequency			Period	Frequency	
1	240 ns	4.17 MHz	40 ns	9	1200 ns	833.3 kHz	200 ns
2	360 ns	2.78 MHz	60 ns	10	1320 ns	757.6 kHz	220 ns
3	480 ns	2.08 MHz	80 ns	11	1440 ns	694.4 kHz	240 ns
4	600 ns	1.67 MHz	100 ns	12	1560 ns	641.0 kHz	260 ns
5	720 ns	1.39 MHz	120 ns	13	1680 ns	595.2 kHz	280 ns
6	840 ns	1.19 MHz	140 ns	14	1800 ns	555.6 kHz	300 ns
7	960 ns	1.04 MHz	160 ns	15	1920 ns	520.8 kHz	320 ns
8	1080ns	0.93 MHz	180 ns				

Table V
Test Modes According to *IC*(6:4)

IC(6:4)		Test Modes
“000”	Mode 0	Pseudorandom testing only
“001”	Mode 1	Deterministic testing only
“010”	Mode 2	Programming deterministic testing for TPG
“011”	Mode 3	Hybrid the deterministic testing with pseudorandom testing
“100”	Mode 4	Single-Shot testing in microsecond range
“101”	Mode 5	Single-Shot testing in millisecond range

B. Design of the DET mode

Some inputs of the CUT board need the specific binary states; not pseudorandom binary signals. The TPG, based on the *DET* mode, is the DET_TPG whose outputs are 48-bit. The DET_TPG that tests the CUT board as a test stimulus generates deterministic test patterns with arbitrary test length. These test patterns are calculated from algorithmic methods that support the detection of different fault models [2]-[3], [5], [14]-[15]. The DET_TPG retrieves these test patterns and generates all required control signals to automatically transfer these test patterns to the CUT inputs. In the *DET* mode, the start, stop, and all control signals of the deterministic test cycle are generated from the data port *DPort*(7:0) and the control port *CPort*(3:0). These ports are applied to the *CU_DET* module by the personal computer (PC), shown in Fig. 4. These ports generate the required control signals for the proper test operation in the DET mode.

The *CU_DET* module generates the *DCLK_TPG* to synchronize and control the test pattern rate of the DET_TPG at the rising edge. In addition, it generates the *DCLK_CUT* that clocks the CUT board either at the rising edge trigger or at the falling edge trigger before the receiving of the test response of the *DATA* by the SA, clocked by the *DCLK_SIG* at the falling edge. In this case, the number of clock cycles inside the test gate depends on the required test patterns to test the CUT board in the *DET* mode. In the *DET* mode, *IC*(11) is the signal that clears the SA at the beginning of the test gate, and the *IC*(12) is used to clear the memory element in the CUT board in the *DET* mode. When the test gate is closed, the proper signature is generated. After that the next three bits *IC*(15:13) are used to automatically transfer the signature to the PC through the status port.

The *CU_DET* module has three sub-modules. They are the decoder of the *CPort*(3:0), the initial condition port (ICP) for the generation of *IC*(15:0), and the initial seed port (ISP) of the PRTPG (*TPG_IS*(47:0)). Table VI illustrates the truth table of the decoder of the *CU_DET*. Each state of *CPort*(3:0) generates a specific control signal. The data port *DPort*(7:0) is used to write the command in the ICP and the data in both the ISP (*TPG_IS*(47:0)) and the DET_TPG (*GTPG*(47:0)).

Table VI
Truth Table of the Decoder of the CU_DET module.

CPort(3:0)	*Control Signal*	*Function*
“0000”	BYTE(0)	Latch the *GTPG*(7:0) and *TPG_IS*(7:0)
“0001”	BYTE(1)	Latch the *GTPG*(15:8) and *TPG_IS*(15:8)

CPort(3:0)	*Control Signal*	*Function*
"0010"	BYTE(2)	Latch the *GTPG*(23:16) and *TPG_IS*(23:16)
"0011"	BYTE(3)	Latch the *GTPG*(31:24) and *TPG_IS*(31:24)
"0100"	BYTE(4)	Latch the *GTPG*(39:32) and *TPG_IS*(39:32)
"0101"	BYTE(5)	Latch the *GTPG*(47:40) and *TPG_IS*(47:40)
"0110"	BYTE_L	Latch the *IC*(7:0)
"0111"	BYTE_H	Latch the *IC*(15:8)
"1000"	Reserved	Reserved
"1001"	Reserved	Reserved
"1010"	DCLK_TPG	Clock the DET_TPG
"1011"	DCLK_CUT	Clock the CUT
"1100"	DCLK_SIG	Clock the SA
"1101"	DCLK_IC	Clock the *IC*(15:0)
"1110"	DCLK_IS	Clock the *TPG_IS*(15:0)
"1111"	Reserved	Reserved

Each 48-bit deterministic test pattern needs six latches to sequentially store six data bytes of *DPort*(7:0). From Table VI, there are six control signals *BYTE*(5:0) that control the latching data and the *DCLK_TPG* clocks the latched data to simultaneously transfer to *GTPG*(47:0) through the DET_TPG in the TPG module. This sequence is repeated each test pattern generation. In the same way, the *DCLK_IS* clocks the latched data to simultaneously transfer to *TPG_IS*(47:0). In addition, the *DCLK_SIG* clocks the SA in the TRC module, and the *DCLK_CUT* clocks the CUT board outside the SM-HBST. Finally, there are two control signals *BYTE_L*, and *BYTE_H* that control the latching data and the *DCLK_IC* clocks the latched data to simultaneously transfer to *IC*(15:0). The programmable code *IC*(15:0) controls the presented test architecture. The problem of the glitch-free affects the control of the test cycle in the *DET* mode. Different delays in the decoder of the *CU_DET* generate glitches in the outputs of control signals. To eliminate these glitches, the signal *DPort*(7) is used to disable the decoder during the changing states of the *CPort*(3:0). This situation eliminates glitches and provides stable test operation in the *DET* mode.

Some inputs of the CUT board need a fixed binary state, and the other inputs need pseudorandom binary signals. The hybrid between the PRT mode and the *DET* mode, called *HPDT* mode, divides inputs of the CUT board into two sets. The first set needs a fixed binary state as a single deterministic test pattern, generated from the *DET_TPG*. The other set needs pseudorandom binary signals, generated from the *PRTPG*. Therefore, test patterns applied to board inputs are the concatenation of the *PRTPG* and *DET_TPG* through the multiplexer selections. This mode increases the ability of the SM-HBST to test different circuit topology.

C. *Design of the SS mode*

The microcontroller oscillator uses quartz crystal for its operation. The frequency of such oscillator is precisely defined and very stable that makes it ideal for time measurement. If it is necessary to measure time between two events, it is sufficient to count pulses generated by this oscillator. In practice, pulses generated by the quartz oscillator are applied either directly or via a prescaler to increment the number stored in the timer register. In the *TRC* module of the SM-HBST, the edge detection compactor (*EDC*) measures the time interval of the stimulated pulse, generated from at least one basic timer module of the microcontroller. The time measurement of the stimulated pulse is considered the measured signature of the output of the single-shot (*SS*) circuit. The simplified block diagram that illustrates the testing application of an *SS* circuit is shown in Fig. 8. In the SS mode, a sequence of deterministic test patterns, generated from *DET_TPG*, is utilized to trigger the SS circuits for the proper pulse generation. The testing criterion of the *EDC* is used to generate the measured signature, based on the edge detection of the *SS* signal. In addition, the interaction between the *EDC* of the SM-HBST and the testing of the timer module in the microcontroller is presented in section IV (part D).

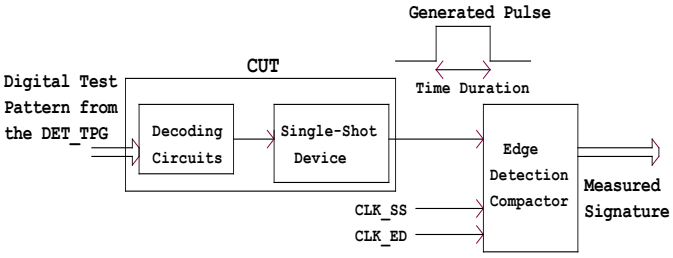


Fig. 8. Digital testing of the SS circuit in the SS mode.

The schematic diagram of the *EDC* and the timing waveform of it are shown in Fig. 9 and Fig. 10, respectively. The *CU* in Fig. 3, and the *CSIG_GEN* module in Fig. 4 generate the required test signals to control the *EDC* in the *SS* mode by two clocks; *CLK_SS*, and *CLK_ED*. The *CLK_SS* (1 MHz in μs range and 1 kHz in ms range) is used to measure the time duration of the stimulated pulse. The *CLK_SS* clocks the 23-bit binary counter (*SS_Counter* shown in Fig. 9). The *CLK_ED* (5 MHz in μs range and 1 MHz in ms range) is used to generate two synchronized pluses; *Edge_1* and *Edge_2*. They are generated either at the rising edge or at the falling edge of the stimulated pulse (*DATA*) of the *SS* circuit. When both edges of the pulse are asserted, the *Edge_1* is generated and the falling edge of it triggers the *Edge_2*, shown in Fig. 10. The *SS* counter starts the counting by the *CLK_SS* and the *ST_STOP* signal generated from *Edge_1* and *Edge_2*. *Edge_1* is a latch pulse for the *SS_LATCH* cell. The assertion of *Edge_1* latches the bus *SSCO*(23:0) to the output bus *SS_SIG*(23:0), and then to output bus of the *TRC* (*SIG*(23:0)). The assertion of *Edge_2* is to clear the *SS_counter*.

D. *FPGA Implementation of the SM-HBST*

All modules of the presented design are connected and the timing simulation of the complete design is achieved to verify the proper operation of the chip design before the implementation on the FPGA chip (Xilinx - X3S200FT256-4). Other cells are used to assist the interface between the inputs and the outputs of the whole chip and the interface circuitry. The debouncer of the *MCLR*, and the other debouncers of the push-bottom switches are used to provide the required *test mode* selection and the required clock selection. In addition, the module that displays a signature of the presented design through the seven-segment

display is used. The device utilization summary and the timing summary report, generated from the FPGA implementation, are presented in Table VII.

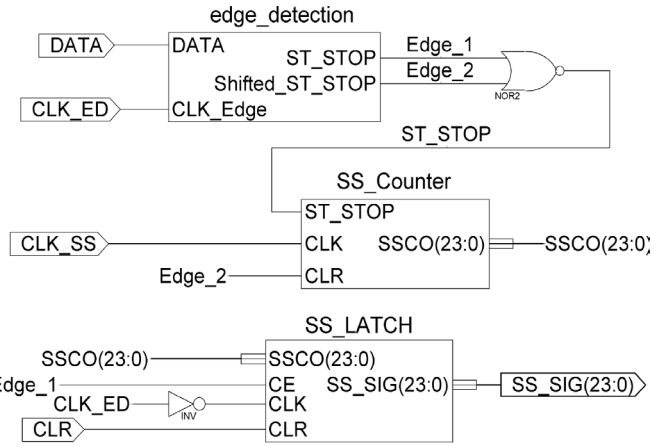


Fig. 9. Schematic diagram of the EDC of the SM-HBST.

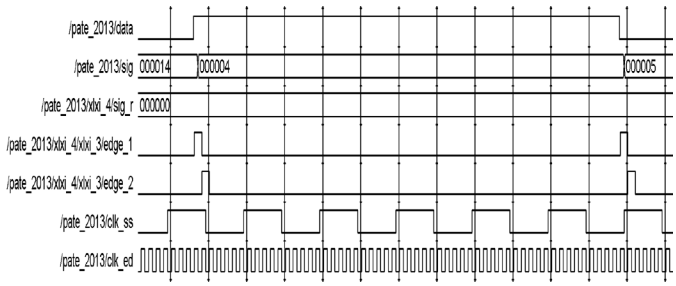


Fig. 10. Timing waveforms of the testing of the SS circuit in the SS mode.

Table VII
FPGA Utilization Summary and Timing Summery of the Spartan-3 (xc3s200-4ft256)

Logic Utilization	Used	Available	Utilization
Number of occupied Slices	411	1,920	21%
Number of Slice Registers	491	3,840	12%
Number of 4 input LUTs	439	3,840	11%
Number of bonded IOBs	115	173	66%
Number of GCLKs	8	8	100%
Number of DCMs	4	4	100%
Total equivalent gate count for design: 34,373			
Timing Summary: Speed Grade -4			
Minimum period: 17.928ns (Maximum Frequency: 55.779MHz)			
Minimum input arrival time before clock: 11.375ns			
Maximum output required time after clock: 16.976ns			
Maximum combinational path delay: 11.600ns			

IV. Testing of Internal Microcontroller Modules

The complexity of microcontrollers that have heterogeneous components with poor accessibility makes their test process a difficult task. In this section, the test process of the internal microcontroller modules is presented. The flowchart of the complete test program of the HYBST strategy for Microchip PIC microcontrollers is illustrated in Fig. 11. The test program asks first, if the system is going to operate either in the normal mode or in the test mode. If the normal mode is chosen, the system will do the predefined industrial application, and if it operates in the test mode, the system will be prepared to operate in the test mode. In the test mode, the microcontroller receives an input selection from one of its ports (*PORTA*), used to test a specific module. This input selection is sent from the SM-HBST. Other microcontroller ports are set output ports to propagate the test response to the SM-HBST.

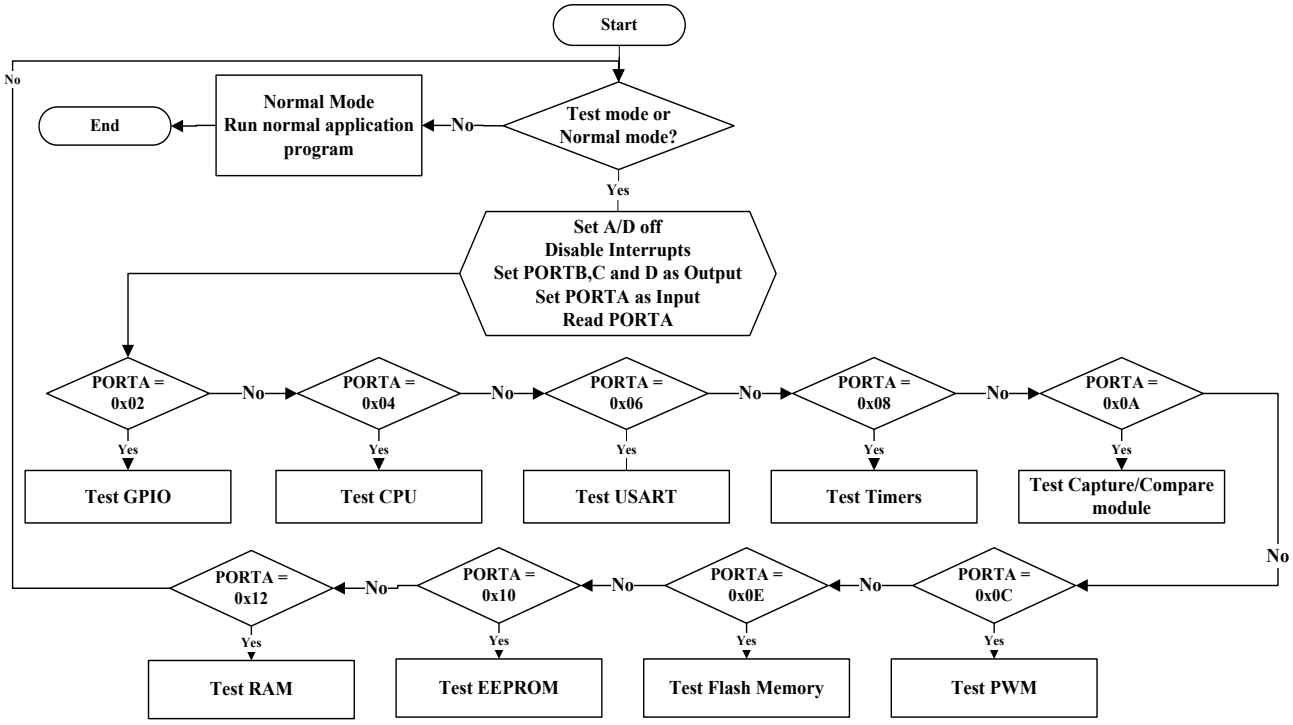


Fig. 11. The flowchart of the complete test program of the HYBST strategy for Microchip PIC microcontrollers.

The test program is composed of several test subroutines, stored in the flash memory of the microcontroller. Test subroutines are based on the ISA of the microcontroller. The emulated TPG of the HYBST provides the required test patterns for testing each internal module of the microcontroller. Test response of each internal module is propagated to the SM-HBST for test response compaction through GPIO pins of the microcontroller. Test subroutines were written in *mikroC* compiler and were simulated in Proteus 7 professional.

A. GPIO test

GPIO pins of the microcontroller allow observing and controlling modules of the microcontroller. Some pins are multiplexed with alternate functions. For all ports, the data direction register, called the TRIS register, controls the directions of I/O pins (either input or output). In this test, GPIO pins (PORTB, PORTC and PORTD) are set output ports. Then, test subroutine sends exhaustive test patterns to these ports. It is noted that some of microcontroller ports are used in other tests, shown in the next sections. Other ports like PORTA and PORTE operate in the input mode to help in running test subroutines and swapping the operation between the test mode and the normal mode. The GPIO test is designed and implemented using the HYBST strategy only. Reference signatures of the GPIO test, based on the SM-HBST, are shown in Table VIII (Last two columns of the table).

Table IX shows the statistics of the GPIO test for the HYBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 and PIC18F452. GPIO pins cannot be tested using the SBST strategy.

Table VIII
Reference signatures of the CPU test and GPIO test

Port	PIN	*signatures of the CPU test*		*signatures of the GPIO test*	
		PIC 16F877	PIC 18F452	PIC 16F877	PIC 18F452
PORTD	0	4F461D	3C2EAD	FFB597	98A6B8
	1	207353	F2C053	4B4474	30B502
	2	16CCD1	86F82A	15C0F5	C40F5E
	3	F5C88A	9AA92E	803D50	8077CC
	4	589ED2	4BDB14	4E4EAF	A0DC17
	5	19F564	243FD2	53F0E0	9A5D2C
	6	449194	8EF7CD	00E9F9	CB508C
	7	AD4496	EB5A81	3F4772	7F1705
PORTB	0			77B646	6E592F
	1			B3FF0A	EF6287
	2			38714E	5D5B1D
	3			046D41	1AD23C
	4			188AE6	DF0D0E
	5			352BC7	6C503A
	6			FD55AC	5F26C3
	7			A5F29A	F341F5
PORTC	0			51F32A	DC2A3F
	1			132938	371FEC
	2			7E08EF	442883
	3			AA24AD	3EEEF0
	4			725B95	9AD2DA
	5			667634	746187
	6			E22CDD	D352F1
	7			ADE6ED	D4FE96

Table IX
Statistics of the GPIO Test for Both PIC16F877 and PIC18F452

GPIO test	Unit	HYBST	
		PIC16F877	PIC18F452
RAM utilization	Byte	16 4.34 %	21 1.36 %
Flash Memory utilization	Word	37 0.45 %	56 0.17 %
Clock cycles	Clock cycle	5,232	6,224

B. CPU test

The central processing unit (CPU) is the brain of the microcontroller. It is responsible for fetching the correct instruction for instruction decoding and then executing it. The CPU executes the instruction to control the microcontroller operation. It controls the address bus of the program memory, the address bus of the data memory, and the accesses to the stack. The CPU is structurally divided into small sub-modules. In this paper, the test subroutine structurally divides the CPU into the ALU, the shift unit (SHU), and the 8*8 multiplier. The ALU performs arithmetical and logical operations, and controls status bits, found in the STATUS register. The result of some instructions forces status bits to a value depending on the state of the result.

In the CPU test, the test subroutine is designed and is then implemented to functionally test the CPU of the microcontroller for both the HYBST strategy and the SBST strategy. The CPU test is based on the divide-and-conquer strategy and the exhaustive testing. The CPU instructions are selected based on the above three criteria, presented in section II. Instruction set $I_{M\text{-}CPU,O}$ from Table II and Table III can test the CPU module. Moreover, the instructions are not randomly chosen, but they are carefully crafted in order to test the desired sub-modules of the CPU. Faulty sub-modules are detected by applying the proper instruction, and then the results of this instruction must be directly sent to the SM-HBST for signature generation and comparison. The control unit is already tested during the CPU test.

Some selected instructions are used to test status bits after arithmetic and logic operations. It is found from the extracted information that microcontroller families have three basic operations. They are word-oriented file register operations, literal and control operations, and bit-oriented file register operations. Some instructions in $I_{M\text{-}CPU,O}$ do the same function with different arguments. Only one form of these instructions is used. Therefore, thirteen instructions are used to test the full capabilities of the CPU (the PIC16F877 has not a multiplier). For example, the $I_{M\text{-}CPU,O}$ set has ADDWF and ADDLW instructions. Both instructions make an addition but the first instruction adds the working register to any other register and the second instruction adds literal to the working register (the working register is the accumulator in the microprocessors).

Reference signatures of the CPU test are taken by the SM-HBST through PORTD port, shown in Table VIII (Middle two columns of the table). Table X and Table XI compare between the HYBST strategy and the SBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 and PIC18F452 to finish the target test. The CPU test process is outlined in List 1.

From Table X and Table XI, the HYBST strategy in the CPU test achieves a significant amount of reduction in the memory utilization and the test application time. In the SBST strategy, the test application code in the flash memory is responsible for generating the test patterns as the TPG and compacting the test response using either emulated LFSR or emulated MISR as the TRC for signature generation. Either the emulated LFSR or the emulated MISR is a group of instructions that increases the size of the test application code, shown in Table X and Table XI. Every single shift of the binary states of either the emulated LFSR or the emulated MISR consumes large clock cycles. The MISR simultaneously compacts the test response of the selected module, and the LFSR individually compacts the test response of each output. Therefore, the TRC code as the MISR is executed 3328 times (thirteen instructions * 256), and the TRC code as the LFSR is executed 26624 times (thirteen instructions * 256 * 8). Therefore, the test application time in the case of the LFSR is greater than the test application time in the case of the MISR. In addition, the on-chip test application code collects the test responses and stores them in the data memory after being compacted into signatures using the TRC code. The usage of the data memory during the CPU test will delay the test process and hence the number of clock cycles is dramatically increased in the case of the SBST strategy.

Table X
Statistics of the CPU Test for PIC16F877

CPU test	Unit	HYBST	SBST with TRC using			
			MISR		LFSR	
RAM utilization	Byte	16 4.34%	40	10.86%	84	23.36%
Flash Memory utilization	Word	49 0.59%	2235	27.28%	2431	29.67%
Clock cycles	Clock cycle	20,500	41,645,103		76,081,267	

Table XI
Statistics of the CPU Test for PIC18F452

CPU test	Unit	HYBST	SBST with TRC using			
			MISR		LFSR	
RAM utilization	Byte	21 1.36%	47	3.05%	93	6.05%
Flash Memory utilization	Word	92 0.28%	3934	12.00%	4788	14.61%
Clock cycles	Clock cycle	25,576	42,365,535		82,430,191	

```
Extract information about CPU modules (ALU – SHU – Multiplier if
exist), then
for (each CPU module M_CPU)
{
    for (every operation Є O_M-CPU)
    {
        Determine I_M-CPU,O
        Select I Є I_M-CPU,O, using controllability and observability criteria
        Apply Exhaustive test patterns for all I_M-CPU,O
        Send test response to PORTD pins
        Acquire and compact test response using the SM-HBST
    }
}
Evaluate compacted test response using the SM-HBST.
```

Listing 1. CPU test process

In the HYBST strategy and during the CPU test, the instructions in $I_{M\text{-}CPU,O}$ use two operands (registers). One of these registers is the working register and the other register is the PORTD, connected to the SM-HBST. In GPIO test, the exhaustive test patterns are applied to the working register only and the result on any instruction is saved in the register of the PORTD. The result of every instruction in the CPU test is directly sent to the PORTD, connected to the SM-HBST for signature generation. The hardware-based TRC in the SM-HBST consumes only single clock cycle every single shift of the binary states, shown in Fig. 7(b) (section III). In addition, the number of memory access is reduced by using the SM-HBST and there is no need to access the data memory of the microcontroller during the execution of the hardware-based TRC to store the generated signatures. Therefore, the memory access that consumes large clock cycles is not required. For 8*8-multiplier test, the exhaustive test patterns are applied to the working register only and the register of the PORTD takes the same value. The result of the multiplication is sent to the PORTD. Therefore, the HYBST strategy that uses the hardware-based TRC in the SM-HBST consumes a small number of clock cycles compared to the software-based TRC of the SBST strategy during the CPU test.

C. USART test

The USART module is known as the *serial communication interface* (SCI). The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as personal computers. In addition, it can be configured as a half duplex synchronous system that can communicate with peripheral devices. The test subroutine of the USART test is designed and is then implemented to test the functionality of the USART module. First, it sets the baud rate of the USART module to 1200 bps. Then, the test patterns (0x00 – 0xFF – 0x33 – 0xCC – 0x0F – 0xF0) are sent to the transmitter (TX) of the USART module and loop it back again through MAX232 chip outside the microcontroller to receive it through the receiver (RX) of the USART module. These received test patterns are propagated to the PORTD. In addition, the signatures on each pin of the PORTD, a signature on TX pin, and a signature on RX pin are measured using the SM-HBST (RX pin and TX pin is multiplexed with the PORTC). Reference signatures, generated by the SM-HBST, are shown in Table XII, and the USART test process is outlined in List 2.

Table XII
Reference signatures of the USART

Ports	PIN	PIC16F877	PIC18F452
PORTD	0	8FE6ED	470BC1
	1	024B37	BD5481
	2	370115	70EC7A
	3	BAACCF	8AB33A
	4	28555F	75D341
	5	A5F885	8F8C01
	6	90B2A7	4234FA
	7	1D1F7D	B86BBA
PORTC	RX	731F53	8AABC8
	TX	731F53	8AABC8

```
Configure USART, then
Set baud rate to 1200 bps, Data to 0x00, and PORTD to 0
Loop
 {
      Send data through transmitter pin
      Check for received data from receiver pin
      If (received data = sent data) then
         Send received data to PORTD.
      Else
         Send received data to PORTD, set error, and exit loop
}
Change data and continue loop until test patterns are sent through
transmitter pin
Acquire and compact test response using the SM-HBST
Evaluate compacted test response using the SM-HBST
```

Listing 2. USART test process

Table XIII and Table XIV compare between the HYBST strategy and the SBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 and PIC18F452 to finish the USART testing. From Table XIII and Table XIV, the HYBST test strategy in the USART test achieves a significant amount of reduction in the memory utilization and the test application time.

Table XIII
Statistics of the USART Test for PIC16F877

USART test	Unit	HYBST	SBST with TRC using	
			MISR	LFSR
RAM utilization	Byte	23 6.25%	40 10.86%	89 24.18%
Flash Memory utilization	Word	213 2.60%	2292 27.97%	2528 30.85%
Clock cycles	Clock cycle	70,852	176,043	242,219

Table XIV
Statistics of the USART Test for PIC18F452

USART test	Unit	HYBST	SBST with TRC using	
			MISR	LFSR
RAM utilization	Byte	28 1.82%	45 2.92%	96 6.25%
Flash Memory utilization	Word	354 1.08%	4002 12.21%	4876 14.88%
Clock cycles	Clock cycle	67,416	164,095	232,715

D. Timer test

From *information extraction* phase, PIC microcontroller families have at least one basic timer module. It can be used as timers/counters. These timers have different sizes (8 bits or 16 bits) and different prescalers. This test subroutine is designed and implemented to functionally test the timers based on two different prescalers. For each timer, special function registers (SFRs) are configured for timer operation using the internal clock cycles. The initial value and the prescaler (1:1, 1:2 and 1:4) are set. The timer is started for counting and PORTB pins are set to HIGH state from LOW state. When the overflow of the timer is occurred, then PORTB pins are deactivated to LOW state.

During this test, the on-time is measured as a signature in the SS mode of the SM-HBST. This test subroutine, based on the HYBST strategy, cannot be implemented in the case of the SBST strategy because the timers must be externally tested through the GPIO pins. The timer test process is outlined in List 3. Reference signatures, generated by the SM-HBST in mode SS (mode 4 or mode 5), are shown in Table XV. Table XV illustrates the measured on-time pulse in µs range (mode 4) for both PIC16F877 (has three timers) and PIC18F452 (has four timers) and compares them with the expected pulse duration from the calculations based on the internal clock cycles of the corresponding microcontroller [6].

```
Configure Timer presale
Enable timer interrupt
Start timer
Set GPIO pin to high
Loop
{
   Watch timer interrupt to check if over flow or not
   If (timer over flow)
     Exit loop
}
Reset GPIO pin to low
Acquire on-time pulse from a GPIO pin using the SM-HBST

Repeat this process but using different prescalers and apply it to all timers
```

Listing 3. Timer test process

Table XVI shows the statistics of the timer test for the HYBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 and PIC18F452. Timer modules cannot be tested using the SBST strategy.

Table XV
Reference Signatures of the Timers and the CCP

Timer/ Prescale	PORT PIN / PIC	PIC16F877	Expected	PIC18F452	Expected
Timer test signatures					
TIMER 0/ (1:2)	PORTB 0	006930 µs	006895 µs	006906 µs	006895 µs
TIMER 0/ (1:4)	PORTB 1	013828 µs	013791 µs	013720 µs	013791 µs
TIMER 1/ (1:1)	PORTB 2	003506 µs	003447 µs	003520 µs	003447 µs
TIMER 1/ (1:2)	PORTB 3	006933 µs	006895 µs	006825 µs	006895 µs
TIMER 2/ (1:1)	PORTB 4	003466 µs	003447 µs	003479 µs	003447 µs
TIMER 2/ (1:4)	PORTB 5	013787 µs	013791 µs	013801 µs	013791 µs
TIMER 3/ (1:1)	PORTB 6			003519 µs	003447 µs
TIMER 3/ (1:2)	PORTB 7			006826 µs	006895 µs
CCP test signatures					
TIMER2	PWM1	E70FD2		9F5EB5	
TIMER2	PWM2	11C8AC		272632	
TIMER1	COMP	006873 µs	006895 µs	003446 µs	003447 µs
TIMER1	COMP	006873 µs	006895 µs	003446 µs	003447 µs

Table XVI
Statistics of the Timer Test for PIC16F877 and PIC18F452

Timer test	Unit	HYBST	
		PIC16F877	PIC18F452
RAM utilization	Byte	16 4.34%	23 1.49%
Flash Memory utilization	Word	98 1.19%	244 0.74%
Clock cycles	Clock cycle	14,680	17,796

E. Capture/Compare/PWM (CCP) test

Both microcontroller families contain two CCP modules, used together with the timers tested in timer test. The CCP modules are identical in operation, with the exception of the operation of the special event trigger. Different CCP modes

depend on timers in the microcontroller. Each CCP module can operate in the following modes:

- *Capture* mode: The CCP module captures the value of Timer1 when an external event occurs in CCPx pin.
- *Compare* mode: The register in the CCP module stores a number (16-bit), compared to the value in Timer1. The result of the comparison may generate an event that may include a change in the CCPx pin.
- *Pulse width modulation* (*PWM*) mode: The CCP module and Timer2 make up a PWM modulator whose output is located in CCPx pin.

The test subroutine of the CCP test is designed and implemented to functionally test CCP modules in *Compare* and *PWM* modes only, based on the HYBST strategy. CCP modules are not tested in all modes because timers were fully tested before in the timer test. First, CCP modules are configured to operate in the *PWM* mode where *PWM1* and *PWM2* are configured to work at frequency 5 kHz with 50% duty cycle. After that, CCP modules are configured to operate in the compare mode. Timer1 is started to count, and its value is then compared with CCP modules until it reaches these known values. Reference signatures of CCPs, based on the SM-HBST in both modes of operations, are taken from PORTC.CCP1 pin and PORTC.CCP2 pin, shown in Table XV. The CCP module test process is outlined in List 4.

Table XVII shows the statistics of the CCP test for the HYBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 and PIC18F452. The CCP module cannot be tested using the SBST strategy.

```
Configure CCPx registers to work in compare mode
Initialize the value of timer1 and enable interrupt
Start timer
Set PORTC.CCPX to HIGH
Loop
 { Watch timer1 until reaching compare value
      If (timer1 = compare value)
                Exit loop  }
Reset PORTC.CCPX to LOW
Evaluate CCPx pin output using the SM-HBST
Repeat this process for CCPx
Configure CCPx registers to work in the PWM mode
Initialize PWMx duty cycle to 50% of 5 KHz clock
Start PWMx
      Watch PWMx output using the SM-HBST
Stop PWMx
```

Listing 4. CCP module test process.

TABLE XVII
STATISTICS OF THE CCP TEST FOR PIC16F877 AND PIC18F452

CCP test	Unit	HYBST	
		PIC16F877	PIC18F452
RAM utilization	Byte	20 5.43%	25 1.62%
Flash Memory utilization	Word	356 4.34%	624 1.90%
Clock cycles for PWM	Clock cycle	8,828	6,776
Clock cycles for COMPARE	Clock cycle	52,292	52,264

F. Memory Test

Microcontrollers have three main memory organization; the flash memory (program memory), the EEPROM and the data memory. Each memory block has its own bus as in Harvard architecture [6], so that access to each block can occur during the same clock cycle. The data memory can further be broken down into the general-purpose RAM and Special Function Registers (SFRs). The SFRs are used to control the peripheral modules found in the microcontroller. The RAM are used to store data that microcontroller needs during its normal operation. This RAM can be divided into smaller banks, also.

1. *Flash Memory Test*: The Flash memory is an important module in a microcontroller chip, because it stores the application program and the test program. Microcontroller modules are going to be tested after the application program and the test program are correctly downloaded. This test is divided into two steps. In step 1, run a C++ program, written on Visual Studio 2010 package, on a personal computer. This program reads the hexadecimal (*Hex*) file words of the application program and the test program, generated from the *mikroC* compiler. Then, the *Hex* file is compacted based on the MISR with primitive polynomial $x^8 + x^6 + x^5 + x^4 + 1$, for the reference signature generation [1], [6]. The reference signature will be stored in the last location in the EEPROM. In step 2, the CPU of the microcontroller will read the data word of the flash memory and the data word is compacted using the same MISR. After that, the generated measured signature will be compared with the reference one, stored in the EEPROM. If both signatures are the same, then the application program and the test program is successfully downloaded and the flash memory is successfully tested. This test subroutine is designed and is then implemented for both HYBST strategy and SBST test strategy.

2. *RAM Test*: J. V. De-Goor and Z. Al-Ars introduced many functional fault models (FFMs) for memories like static faults and dynamic faults [35]-[36]. Based on divide-and-conquer strategy, the RAM module is divided into smaller banks and each bank is individually tested. In the case of the PIC16F877, the RAM can be divided into four smaller bank and twelve banks for the PIC18F452. Then, each bank is individually tested using March test algorithms [37] because of their simplicity and linearity with the memory size. The March test can be defined as a sequence of March elements, where a March element is a sequence of memory operations sequentially performed on all memory cells. In a March element, the way from one cell to the next one is specified by the address order. For some March elements, the address order can be chosen as increasing or decreasing. In a March element, it is possible to perform a write 0 (W0), a write one (W1), a read zero (R0) and a read one (R1). The zero and one after read operations represent the expected values of the read on the output. An example of a March element is ↑ (R0; W1), where all memory cells are accessed in an increasing address order while performing R0 then W1 on each cell, before continuing to the next cell. By arranging a number of March elements one after the other, a March test is constructed.

The RAM test is designed and implemented based on March AB algorithm. March AB {↕(W0); ↑(R0;W1;R1;W1;

R1); ↑(R1;W0;R0;W0;R0); ↓(R0;W1;R1;W1;R1); ↓(R1;W0;R0;W0;R0); ↕(R0)} was introduced by S. Carlo, A. Bosio, G. Natale, and P. Prinetto [37]. Their March test targets realistic memory static linked faults and dynamic unlinked faults in SRAMs and has a test length with a complexity of 22*n*. Here the test subroutine, based on March test AB, is constructed. (In addition, it is possible to extend this test subroutine to any March test [38].) The test response from the RAM is propagated through the PORTC (GPIO pins) to the SM-HBST for signature generation.

Table XVIII contains reference signatures of the RAM test using March AB, generated by the SM-HBST. The data bus of each RAM bank is propagated to PORTC. The data bus is 8-bit with eight same signatures on each pin of PORTC. For simplicity, only single signature will be presented for each RAM bank.

Table XVIII
RAM test Signatures Based on March AB

Module	RAM Bank	PIC 16F87X	PIC 18F4X2	RAM Bank	PIC 18F4X2
RAM	1	0438F8	EFF5D9	7	08E472
	2	B300AE	511ECC	8	B2691B
	3	CFB46B	A83A68	9	C3162B
	4	379AAD	BA8630	10	A5A126
	5		A33AFC	11	DFB602
	6		9EC7E8	12	93B255

Table XIX and Table XX compare between the HYBST strategy and the SBST strategy in terms of memory utilization and the total number of clock cycles for both PIC16F877 (368 byte divided into 4 parts) and PIC18F452 (1536 byte divided into 12 parts) to finish RAM test using March AB. From Table XIX and Table XX, the HYBST strategy in the RAM test using March AB achieves the reduction in memory utilization and the test application time except in the RAM utilization in Table XX.

Table XIX
Statistics of the RAM Test Using March AB for PIC16F877

RAM test	Unit	HYBST		SBST with TRC using MISR		SBST with TRC using LFSR	
RAM utilization	Byte	16	4.34%	16	4.34%	16	4.34%
Flash Memory utilization	Word	200	2.44%	336	4.10%	336	4.10%
Clock cycles	Clock cycle	279,668		531,399		532,039	

Table XX
Statistics of the RAM Test Using March AB for PIC18F452

RAM test	Unit	HYBST		SBST with TRC using MISR		SBST with TRC using LFSR	
RAM utilization	Byte	23	1.49%	21	1.36%	21	1.36%
Flash Memory utilization	Word	302	0.92%	528	1.61%	528	1.61%
Clock cycles	Clock cycle	1,221,368		2,071,891		2,071,891	

3. *EEPROM Test:* The EEPROM is read and is written during normal operation. This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the SFRs. There are four SFRs used to read and to write this memory. These registers are EECON1, EECON2 (not a physically implemented register), EEDATA and EEADR. User program can use the EEPROM to store the important data during the application run and to read it again after the end of the application. First, the test subroutine reads the data from EEPROM locations. Then, the test subroutine stores the data into a temporary location before testing this module.

Since the large delay (20 *ms*) is required between the written data cycle of the EEPROM and the read data cycle of the EEPROM, therefore the required number of clock cycles is large to deal with the SM-HBST. Therefore, the EEPROM test subroutine was implemented based on a simple marsh algorithm with low time complexity. It is the modified algorithmic test sequence (MATS) algorithm that detects all combination of stuck-at faults (SAF) in RAMs and has a test length with a complexity of 4*n* [39]-[40]. MATS test sequence is {↑(W0); ↑(R0,W1); ↑(R1)} where the EEPROM is internally tested and the data results are sent for validation.

The measured signature in the HYBST strategy is calculated inside the microcontroller like the SBST strategy. Therefore, from Table XXI and Table XXII, there is no improvement in the memory utilization and no reduction in the total test application time for both PIC16F877 and PIC18F452 to finish the EEPROM test.

Table IV XXI
Statistics of the EEPROM Test using MATS for PIC16F877

EEPROM test	Unit	HYBST		SBST with TRC using MISR		SBST with TRC using LFSR	
RAM utilization	Byte	22	5.97%	22	5.97%	22	5.97%
Flash Memory utilization	Word	190	2.32%	190	2.32%	190	2.32%
Clock cycles	Clock cycle	15,493,952		15,493,951		15,493,951	

Table XXII
Statistics of the EEPROM Test using MATS for PIC18F452

EEPROM test	Unit	HYBST		SBST with TRC using MISR		SBST with TRC using LFSR	
RAM utilization	Byte	27	1.75%	27	1.75%	27	1.75%
Flash Memory utilization	Word	316	0.96%	316	0.96%	316	0.96%
Clock cycles	Clock cycle	15,445,035		15,445,035		15,445,035	

G. Comparisons between the HYBST and the SBST

Test subroutines were individually investigated for testing all microcontroller modules in the previous sections. In this section, all test subroutines are merged together in one complete test program. The effectiveness of the complete test program using the HYBST strategy and the SBST strategy will be evaluated on two Microchip® microcontrollers (PIC16F877A and PIC18F452). Table XXIII and Table XXIV show the comparison of the complete test program between the HYBST strategy and the SBST strategy, based on the emulated LFSR and the emulated MISR as the TRC. These tables present the statistics of the complete test program such as the memory utilization (data memory, and flash memory), the test application time (the number of clock cycles taken to finish the complete test), and the testability of microcontroller modules (the number of tested modules in a microcontroller).

Table XXIII
Statistics of the Complete Test Program for PIC18F452

Microchip® PIC18F452	Unit	HYBST		SBST with TRC using			
				MISR		LFSR	
RAM utilization	Byte	30	1.95%	47	3.06%	94	6.12%
Flash Memory utilization	Word	2326	7.09%	5592	17.06%	6682	20.4%
Clock cycles	Clock cycle	21,112,792		70,023,259		112,101,595	
Tested Modules		All Passed		Timers, GPIO and CCP Failed			

Table XXIV
Statistics of the complete test program for PIC16F877

Microchip® PIC16F877	Unit	HYBST		SBST with TRC using			
				MISR		LFSR	
RAM utilization	Byte	27	7.33%	44	11.95%	93	25.27%
Flash Memory utilization	Word	1663	20.3%	3568	43.55%	3897	47.57%
Clock cycles	Clock cycle	25,600,168		78,348,171		114,734,099	
Tested Modules		All Passed		Timers, GPIO and CCP Failed			

Performance enhancement of the HYBST strategy is shown in Table XXV. The comparisons between the HYBST strategy and the SBST strategy based on the emulated LFSR and the emulated MISR as the TRC present the superiority of the presented HYBST strategy over the SBST strategy. It achieves a significant amount of reduction in the memory utilization and the test application time. Besides, the fault coverage of the HYBST strategy is greater than the other SBST strategies.

Table XXV
Performance Enhancement Based on the HYBST Test Strategy

Microcontroller	PIC 18F452		PIC 16F877	
	SBST with TRC using		SBST with TRC using	
	LFSR	MISR	LFSR	MISR
RAM reduction	68.08%	36.17%	70.96%	38.63%
Flash memory reduction	65.19%	58.40%	57.32%	53.40%
Clock cycle reduction	81.16%	69.84%	77.68%	67.32%

From Table XXV, the superiority of the HYBST strategy in the memory utilization is achieved due to the reduction of the RAM utilization by 70.96% and 38.63% for PIC16F877 and by 68.08% and 36.17% for PIC18F452. In addition, the HYBST strategy reduces flash memory utilization by 57.32% and 53.40% for PIC16F877 and by 65.19% and 58.40% for PIC18F452. The superiority of the HYBST strategy in the total number of clock cycles is achieved due to the reduction of the test application time by 77.68% and 67.32% for PIC16F877 and by 81.16% and 69.84% for PIC18F452.

From Table XXIII, Table XXIV, and Table XXV, the complete test program in the PIC18F452 utilizes larger memory space than in the PIC16F877 because the PIC18F452 has more features than the PIC16F877 especially in timers and the multiplier that needs more effort to test them. Since the PIC18F452 achieves from 10-15 MIPS, and the PIC16F877 operates in 5 MIPS [6], [31], therefore the complete test program of the PIC18F452 consumes lower test application time because of its higher performance rate. In addition, the HYBST strategy can test all modules and the SBST strategy cannot test timers, GPIO pins, and CCP modules because the SBST strategy cannot properly test GPIO pins of the microcontroller without using the SM-HBST strategy.

In the next section, the integrated test strategy of the microcontroller testing on the printed circuit board (PCB) for fault diagnosis indicates a real practical test strategy.

V. Fault Diagnosis of the Circuit Board Including Microcontroller Chip

Fault diagnosis of the CUT board including a microcontroller chip for correct operation is an important issue. It is required to apply the proper test patterns to the inputs of the CUT board and to analyze the test response for fault diagnosis. During the fault diagnosis, the faulty source components are located so that the CUT board is returned back to the service. The testing of a microcontroller chip in the board level requires integrating test strategies that test random logic integrated circuits (ICs) based on the SM-HBST strategy and test a microcontroller chip based on the HYBST strategy. The following integrated test strategy consists of two main phases; the *CUT preparation and preprocessing* phase and the *testing and fault diagnosis* phase.

A. CUT preparation and preprocessing phase

In this phase, the schematic diagram of the CUT should be defined. To clarify the idea of this phase, a simple CUT board is selected, shown in Fig. 12. It consists of three main parts. The first part is random logic part that consists of one decoder (74LS138 - U7), and two 4-Bit magnitude comparators (74LS85 - U3, U5). The second part is the microcontroller part that consists of the microcontroller chip (PIC16F877 or PIC18F452), and the integrated circuit (MAX232 - U8). Both the application program used in the normal mode and the test program used in the test mode are loaded to the microcontroller. The third part is the design for testability (DFT) part that enables the CUT board to operate in two operation modes; normal mode and test mode. The DFT part has three multiplexer ICs (74LS157 - U4, U6, U9) that partition the microcontroller part from the random logic part in the test mode. The DFT part enables the testing of each part in the CUT board with its own test strategy to reduce the testing difficulty, and to assist the fault isolation. In the normal mode, the CUT board does its normal operation according to industrial applications.

The CUT board layout is generated from the circuit layout tool (ORCAD layout), shown in Fig. 13. The *layout netlist file* that defines a component list and a complete description of the connectivity nets between the components on the PCB is generated. From the *layout netlist file*, different related files are generated. These files will be very useful in the next phase to locate the faulty source components. The *Node file* is constructed from the *layout netlist file*. It holds node number, node name such as *U04P04-74LS157* (it means IC number in

the schematic diagram U04, pin number in the IC 04, and IC name 74LS157), node type (input node or output node), and the status of the node test. The *Dependency file* is also constructed, and holds output nodes such as *U04P04-74LS157* and input nodes affect this output node in the same IC {*U04P01-74LS157*, *U04P02-74LS157*, *U04P03-74LS157*, *U04P015-74LS157*}. After constructing dependency file, the *connectivity file* will be created from the *layout Netlist file*. It holds the output nodes, connected to input nodes of other ICs. It is like the output node *U04P04-74LS157*, connected to the input node *U05P01-74LS85*. These files are related to each other. When the output node, *U04P04-74LS157*, in the node file is chosen, the related information to this node appears in the dependency file and in the connectivity file.

This phase ends by capturing reference signatures for all nodes of the golden CUT board with the aid of the SM-HBST strategy. The capturing reference signatures can be achieved for the random logic part and DFT part according to section III, and then the microcontroller part according to section IV. This phase is executed only once for every new CUT board.

B. Test and fault diagnosis phase

This test phase achieves the testing of the CUT board in two different directions. First direction is to test random logic part and the DFT part. The SM-HBST generates required test pattern signals, propagated through these ICs, and then it detects the stream binary data, generated from a target node on the CUT board by the test probe. The measured signature of that node, based on the test strategy in section III, is generated. The other direction is to test the microcontroller part. The SM-HBST generates required test signals to select the test routines of the target microcontroller module. The stream binary data, generated from a target node on the microcontroller part, is also captured by the SM-HBST to generate the measured signature of that node, based on test strategy in section IV.

After the measured signatures of every node in the CUT board are generated, they are automatically compared with the corresponding reference signatures, captured using the previous phase. If the measured signature of a node is equal to a corresponding reference one, the test status flag of that node is set true (good node), and if this signature is not equal to a corresponding reference one, the test status flag of that node is set false (bad node).

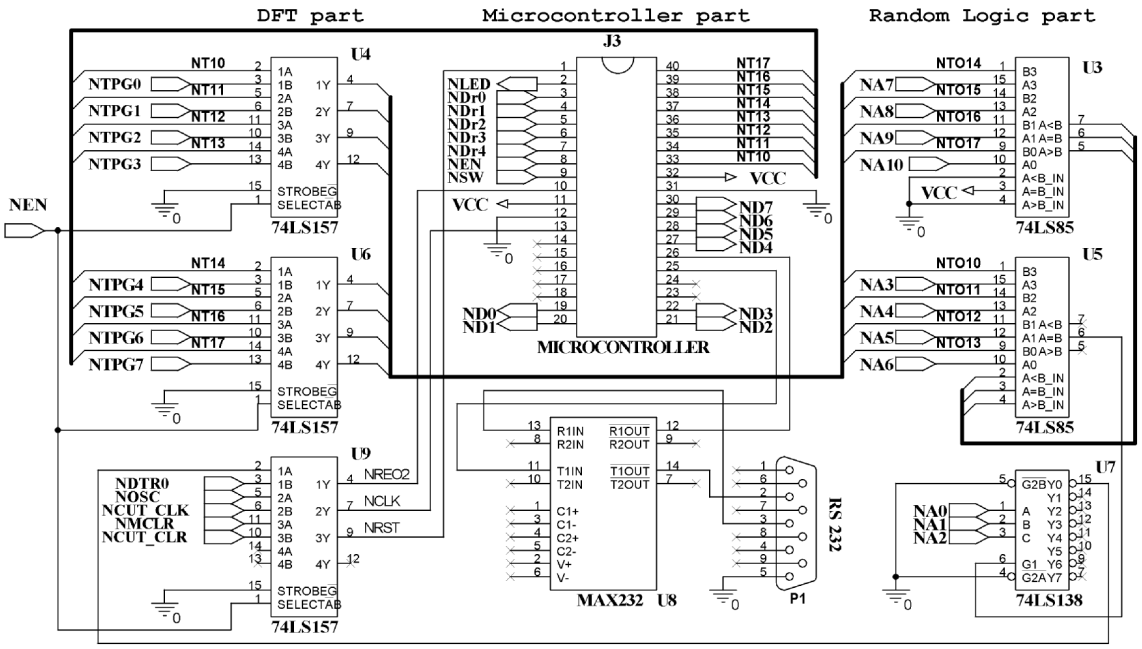
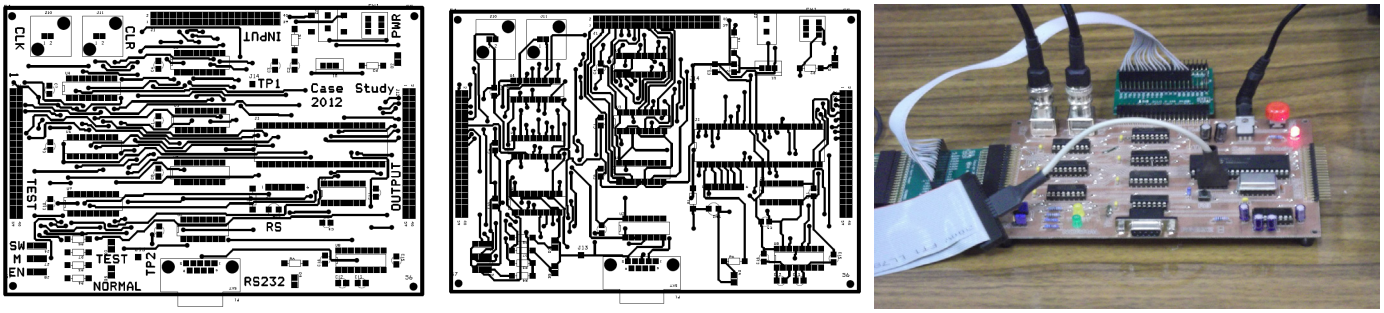


Fig. 12. Schematic diagram of the CUT.



(a) The CUT layout fornt plane. (b) The CUT layout back plane. (c) The printed circuit board of the CUT.

Fig. 13. The CUT board.

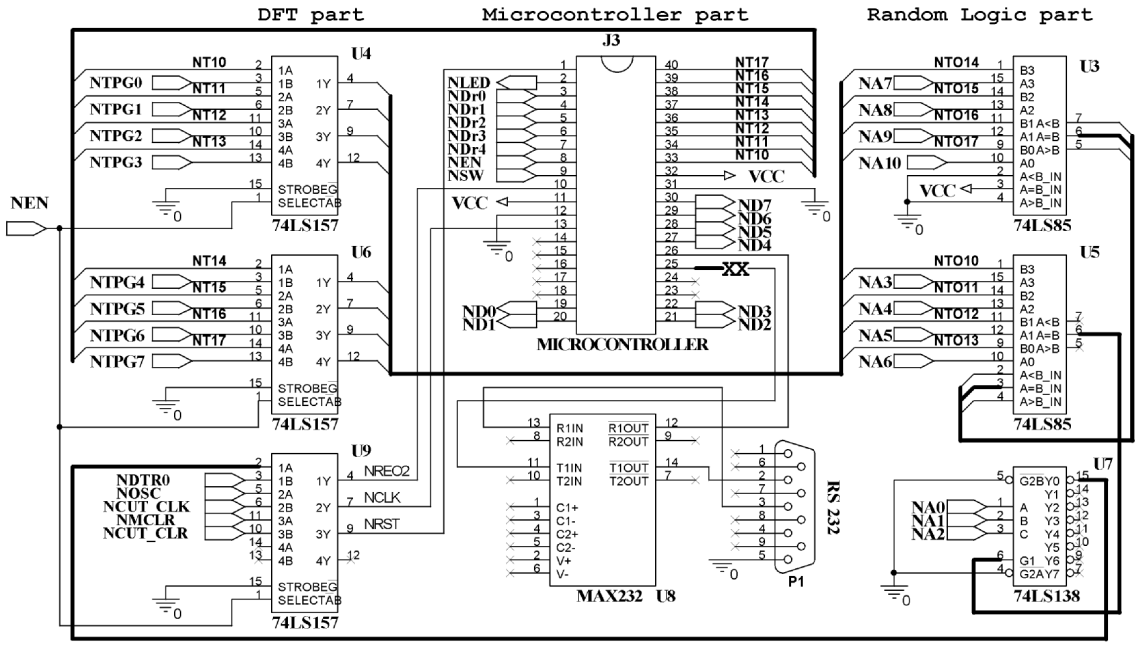


Fig. 14. Schematic diagram of the faulty CUT propagation.

When the fault is applied to the pin U03P06-74LS85 in the random logic part, it propagates from the source faulty node (pin) to the multiplexer U9 pin 2 (U09P02). In the test mode, this fault cannot propagate to the microcontroller part (input PORTE), shown in Fig. 14. The node U09P02 is considered the output of the random logic part in the test mode. The SM-HBST generates required test patterns in the test mode 0, and the measured signatures of every node in the random logic part are captured. All output nodes that have *false* test status are stored in a temporary file as bad nodes. If this file has no bad nodes, then this CUT board is fault-free, and if this file has at least one bad node, then this CUT board is faulty. The bad nodes of the CUT board shown in Fig. 14 are shown in Table XXVI with the shadow rows. They are {*U03P06*, *U05P05*, *U05P06*, *U05P07*, *U07P07*, *U07P08*, *U07P09*, *U07P10*, *U07P11*, *U07P12*, *U07P13*, *U07P14*, *U07P15*}.

The dependency file and connectivity file, generated from the previous phase, are utilized to track each bad node in the temporary file to get the source faulty node (nodes). For example, the bad node U07P15-74LS138 in the temporary file has the dependency set, DU07P15 {U07P01, U07P02, U07P03, U07P04, U07P05, U07P06}. If the node U07P06-74LS138 that belongs to DU07P15 is connected to the node U05P06-74LS85 in the temporary file, skip the node U07P15-74LS138 and another bad node in the temporary file is tracked. This search processes until the source faulty node U03P06-74LS85 is determined. This node has the dependency set, DU03P06 {U03P01, U03P02, U03P03, U03P04, U03P09, U03P10, U03P11, U03P12, U03P13, U03P14, U03P15}. All nodes in the DU03P06 have no connection with any bad node. Therefore, the node U03P06-74LS85 is selected as the source faulty node. If the measured signature of the source faulty node is varying from test cycle to another test cycle (flashing), and the measured signature of any node in the temporary file is flashing, a short circuit (bridging fault) among the nodes in the temporary file may be occurred.

TABLE XXVI
REFERENCE AND MEASURED SIGNATURES OF TESTED NODES OF THE RANDOM LOGIC PART

No.	Node Name	Reference (Good) signature		Measured signature		Status
226	U03P05-74LS85	B0B300	0 - 0	B0B300	0 - 0	True
227	U03P06-74LS85	25FA0A	0 - 0	299BD5	0 - 0	False
228	U03P07-74LS85	BCD2DF	0 - 0	BCD2DF	0 - 0	True
241	U04P04-74LS157	82775E	0 - 0	82775E	0 - 0	True
244	U04P07-74LS157	413BAF	0 - 0	413BAF	0 - 0	True
246	U04P09-74LS157	A09DD7	0 - 0	A09DD7	0 - 0	True
249	U04P12-74LS157	D04EEB	0 - 0	D04EEB	0 - 0	True
251	U05P05-74LS85	8044B1	0 - 0	0B3009	0 - 0	False
252	U05P06-74LS85	4D5644	0 - 0	EF8624	0 - 0	False
253	U05P07-74LS85	E48920	0 - 0	CD2DF8	0 - 0	False
273	U06P04-74LS157	682775	0 - 0	682775	0 - 0	True
276	U06P07-74LS157	B413BA	0 - 0	B413BA	0 - 0	True
278	U06P09-74LS157	DA09DD	0 - 0	DA09DD	0 - 0	True
281	U06P12-74LS157	ED04EE	0 - 0	ED04EE	0 - 0	True
292	U07P07-74LS138	680659	0 - 0	3E6B17	0 - 0	False
294	U07P09-74LS138	5147C1	0 - 0	69A611	0 - 0	False
295	U07P10-74LS138	D1F135	0 - 0	5385D7	0 - 0	False
296	U07P11-74LS138	1F0722	0 - 0	CB4EF9	0 - 0	False
297	U07P12-74LS138	A1990C	0 - 0	6A5924	0 - 0	False
298	U07P13-74LS138	8A29C0	0 - 0	9B24BD	0 - 0	False
299	U07P14-74LS138	A29B21	0 - 0	D29304	0 - 0	False
300	U07P15-74LS138	33CA26	0 - 0	036E91	0 - 0	False
321	U09P04-74LS157	299BD5	0 - 0	299BD5	0 - 0	True
324	U09P07-74LS157	000000	0 - 0	000000	0 - 0	True
326	U09P09-74LS157	299BD5	0 - 0	299BD5	0 - 0	True
329	U09P12-74LS157	299BD5	0 - 0	299BD5	0 - 0	True

VI. Experimental results

This section presents the complete application of this hybrid test strategy, applied to the CUT board in Fig. 14. In the normal mode, the main input signals (*NA0* to *NA10*), the master clear (*NMCLR*), and the main clock oscillator (*NOSC*) are propagated through the random logic part and the microcontroller part. The presented hybrid test strategy does not depend on the embedded application of the CUT board. It is applicable for any CUT board including a microcontroller chip whatever its industrial applications are.

In previous section, the fault was applied to the pin U03P06-74LS85 in the random logic part. The reference (good) signatures and the corresponding measured signatures of the CUT board were captured and then stored in the database according to test mode 0. In this section, the other fault is applied to the input pin J03P26-RX-PIC18F452 in the microcontroller part.

To properly test the CUT board, it is required to go through various modes and to connect the SM-HBST with the CUT board. First, it is required to know some control signals for the operation modes of the CUT board. The control input signal, *NEN*, is connected to all multiplexers in the DFT part. It allows the switch between the normal mode and the test mode. These multiplexers can switch between main inputs of the CUT board in the normal mode and the required test signals, generated from the SM-HBST in the test mode. Multiplexers (U4 and U6) switch the output port PORTB (pin 33 to pin 40) of the microcontroller and the test pattern inputs (*NTPG0* to *NTPG7*), generated from the SM-HBST (*GTPG*(7:0)). The signal input, *NSW*, enables the microcontroller operation in the normal mode. Multiplexer U9 switches between the output signal generated from the output of decoder U7 and the input signal *NTDR0* that enables the microcontroller part in the test mode. In addition, multiplexer U9 switches between the master clear (*NMCLR*) of the CUT, and the test clear (*NCUT_CLR*), generated from the SM-HBST (*CUT_CLR*). It also switches between the clock oscillator (*NOSC*) of the CUT board, and the test clock (*NCUT_CLK*), generated from the SM-HBST (*GCUT_CLK*).

Signal *NEN, NSW*, and signal *NTDR0* are connected to the port PORTE (pin 8 to pin 10) of the microcontroller as an input port. These signals, shown in Table XXVII, partition the CUT into two main parts (the random logic part and the microcontroller part) in the test mode. The inputs of the random logic part (*NTPG0* to *NTPG7*, and *NA0* to *NA10*) are connected to test signals, generated from the SM-HBST (*GTPG*(18:0)). The SM-HBST generates the required test patterns and the measured signatures of target nodes according to test mode 0, explained in section III. In the microcontroller part, the microcontroller receives an input selection (*NDr0* to *NDR4*) from port PORTA (pin 3 to pin 7) as an input port. This input selection is stimulated from the SM-HBST as the deterministic test patterns (*GTPG*(42:47)) according to test mode 3 shown in Table V. These deterministic test patterns are used to select the proper microcontroller module for testing. Other microcontroller ports are set to be output ports (PORTB, PORTC, and PORTD) to propagate the test response to the SM-HBST for signature generation.

Table XXVII
Control signals for modes of the CUT operation

Control signals			Operation
NEN	*NSW*	*NTDR0*	
0	0	X	Disable the microcontroller operation in the normal mode.
0	1	X	Enable the microcontroller operation in the normal mode.
1	X	1	Enable switching of the multiplexers in test mode and enable the testing of the microcontroller part.
1	X	0	Enable switching of the multiplexers in test mode and disable the testing of the microcontroller part.

The applied fault to the pin J03P26-RX-PIC18F452 in the microcontroller part is examined. The fault is the open-circuit in the output track on the CUT board between pin J03P26-RX-PIC18F452 and pin U08P11-MAX232. It affects the *USART Receive* part of the serial port connection. This leads to a number of bad nodes in the serial port of the microcontroller part. The USART test routine, presented in section IV, sends test patterns to the USART transmitter pin J03P25-TX-PIC18F452 and loops it back again through MAX232 chip to receive it from USART receiver pin J03P26-RX-PIC18F452 (a short circuit through RS232).

Based on the CUT board shown in Fig. 14, Table XXVI shows some tested nodes that include the reference and the measured signatures of the random logic part, and Table XXVIII shows some tested nodes that include the reference and the measured signatures of the microcontroller part. Every tested node in Table XXVI has a single reference signature, a single measured signature and a single test status. However, in Table XXVIII, some external tested nodes of the microcontroller have multiple reference signatures, multiple measured signatures, and different test status according to the testing of the selected module of the microcontroller.

Table XXVIII
Reference and measured signatures of tested nodes of the microcontroller part

No.	Node Name	Reference Signature		Measured Signature		Status
73	J03P19-RD0-PIC18F452	98A6B8	PORTS	98A6B8	PORTS	True
74	J03P20-RD1-PIC18F452	30B502	PORTS	30B502	PORTS	True
75	J03P21-RD2-PIC18F452	C40F5E	PORTS	C40F5E	PORTS	True
76	J03P22-RD3-PIC18F452	8077CC	PORTS	8077CC	PORTS	True
81	J03P27-RD4-PIC18F452	A0DC17	PORTS	A0DC17	PORTS	True
82	J03P28-RD5-PIC18F452	9AD2DA	PORTS	9AD2DA	PORTS	True
83	J03P29-RD6-PIC18F452	CB508C	PORTS	CB508C	PORTS	True
84	J03P30-RD7-PIC18F452	7F1705	PORTS	7F1705	PORTS	True
95	J03P19-RD0-PIC18F452	3C2EAD	CPU	3C2EAD	CPU	True
96	J03P20-RD1-PIC18F452	F2C053	CPU	F2C053	CPU	True
97	J03P21-RD2-PIC18F452	86F82A	CPU	86F82A	CPU	True
98	J03P22-RD3-PIC18F452	9AA92E	CPU	9AA92E	CPU	True
99	J03P27-RD4-PIC18F452	4BDB14	CPU	4BDB14	CPU	True
100	J03P28-RD5-PIC18F452	243FD2	CPU	243FD2	CPU	True
101	J03P29-RD6-PIC18F452	8EF7CD	CPU	8EF7CD	CPU	True
102	J03P30-RD7-PIC18F452	EB5A81	CPU	EB5A81	CPU	True
103	J03P19-RD0-PIC18F452	470BC1	USART	00F4BF	USART	False
104	J03P20-RD1-PIC18F452	BD5481	USART	00F4BF	USART	False
105	J03P21-RD2-PIC18F452	70EC7A	USART	00F4BF	USART	False
106	J03P22-RD3-PIC18F452	8AB33A	USART	00F4BF	USART	False
107	J03P27-RD4-PIC18F452	75D341	USART	00F4BF	USART	False
108	J03P28-RD5-PIC18F452	8F8C01	USART	00F4BF	USART	False
109	J03P29-RD6-PIC18F452	4234FA	USART	00F4BF	USART	False
110	J03P30-RD7-PIC18F452	B86BBA	USART	00F4BF	USART	False
111	J03P25-TX-PIC18F452	8AABC8	USART	8AABC8	USART	True
112	J03P26-RX-PIC18F452	8AABC8	USART	299BD5	USART	False

In Table XXVIII, nodes 73 to 76 and nodes 81 to 84, generated from PORTD with PORTS status, test the GPIO module. Nodes 95 to 102, generated from PORTD with CPU status, test the CPU module, and nodes 103 to 110, generated from the PORTD with USART status, test the USART module. Shaded rows in Table XXVI and Table XXVIII show bad nodes, resulted from the target fault in the microcontroller part besides the applied fault in the random logic part.

Since the USART transmitter generates proper test patterns from node J03P25-TX-PIC18F452 according to the USART test, therefore the measured signature (8AABC8) is similar to the reference signature (8AABC8). Due to the open-circuit between pin J03P25-TX-PIC18F452 and pin U08P11-MAX232, the measured signature of pin U08P11-MAX232 (299BD5) is different from the reference signature (8AABC8). These patterns return back again through MAX232 chip from node U08P12-MAX232 and the microcontroller receives them from USART receiver pin J03P26-RX-PIC18F452. Therefore, the measured signature of pin J03P26-RX-PIC18F452 (299BD5) will be different from the reference signature (8AABC8). The test status of node J03P25-TX-PIC18F452 is a good node, but the test status of node J03P26-RX-PIC18F452 is a bad node. In addition, nodes 103 to 110, generated from PORTD, are bad nodes in the USART test. However, the nodes generated from PORTD in the GPIO test and the CPU test, are good nodes. Therefore, the source faulty node of the microcontroller part is node J03P26-RX-PIC18F452 of the USART module in the microcontroller.

Discussion: In this paper, three main test strategies are discussed. The first one is the HBST strategy. All testing system will be outside the CUT board on the external ATE. It costs large hardware overhead and large test application time to stimulate the target fault in the CUT board, besides small DFT circuitry is used to increase the controllability and observability of the CUT board. Increasing the controllability and observability of the CUT board assists the external ATE to properly apply the TPG and to acquire the test response. The CUT board including a microcontroller chip is a heterogeneous system with poor accessibility, so its embedded modules need large application time to properly apply test patterns. Therefore, the SBST strategy is considered the second test strategy to resolve the test cost of the HBST. The SBST strategy is mainly used to test large microprocessors.

In the SBST strategy, the testing system including the TPG, TRC, test controller, and testing evaluation will be emulated using the software code inside the memory of the microprocessor system. The cost of the testing system is based on the hardware utilization, test application time, and the fault coverage. This microprocessor system has large memory, and the usage of the testing system inside this large memory does not utilize much space in it. Therefore, the SBST is an efficient test strategy for large microprocessors. However, when the SBST strategy is applied to the microcontroller with limited memory space, the cost of the hardware utilization is increased. No authors proposed solution for this issue. The authors in this paper propose a new hybrid test strategy to solve this issue, called the HYBST.

The cost of the testing system based on the HYBST strategy is composed of three parts. The first part is the SM-HBST design based on the FPGA implementation whereas the total gate equivalent of the presented design on the FPGA is 34,373 gate equivalent (equals 137,492 transistors) where the gate equivalent unit is 2-input NAND gate that equals four transistors. The second part is the used ICs in the DFT on the CUT board, and the third part is the hardware utilization in the microcontroller. The HYBST strategy is used to test the CUT board including the microcontroller chip. It is required to make a comparison between the number of transistors used in the testing system and the CUT. This discussion can be concluded by what the expected number of transistors in the microcontroller is.

From Table I and for the PIC18F452, the data memory (SRAM) used is 1536 byte (1536 x 8 = 12288 bits). Every bit of the SRAM cell needs six transistors. This leads to 73,728 transistors in the 12288 bits of the data memory. The excepted number of transistors in the periphery circuitry is around 2,224 transistors according the memory design steps in [41]. Therefore, the total required transistors in the data RAM of the PIC18F452 microcontroller is around 76,000 transistors. The Flash memory that has 16 kword (word = 14 bits) is around 267,376 transistors [41]. The used EEPROM memory has 256 byte. It is around 2,648 transistors [41]. Therefore, the total transistors in these three modules, only in the microcontroller chip, are 346,024 transistors. From this estimation of the expected number of transistors in the data memory (SRAM), the flash memory, and the EEPROM memory, it exceeds the total transistors in the FPGA implementation. If all microcontroller modules are considered, the expected number of transistors in the microcontroller will be greater than the CUT board.

The HYBST strategy is proposed to solve the testing cost of the HBST strategy and the SBST strategy in the hardware overhead, test application time and the fault coverage. If the test cost of the HYBST strategy is compared to the test cost of the SBST strategy for testing the microcontroller with small internal memory, the memory utilization and test application time are reduced besides increasing in the fault coverage. Adding FPGA circuit board increases the hardware overhead in the case of the HYBST strategy. However, the hardware overhead in the case of the SBST strategy is the memory utilization only. If the test cost of the HYBST is compared to the test cost of the HBST strategy for testing the microcontroller with poor accessibility, the hardware overhead is nearly the same but the test application time is reduced besides increasing in the fault coverage.

The presented test strategy in this paper has some limitations. The maximum frequency of the three-phase clocks (GCLK_TPG, GCLK_CUT, and GCLK_SIG) from Table IV is 4.17 MHz. Whereas, the maximum frequency of most microcontrollers is greater than 4 MHz, therefore, the proposed test strategy is not suitable for at-speed testing. In addition, and due to the small numbers of input modules in the microcontroller, the exhaustive testing is used to detect all combinational faults, detected by single-pattern test generators without fault simulator [1], [13]. Therefore, the proposed test strategy is not suitable for the detection of the delay fault, detected by two-pattern test

generators without fault simulator [12]. It is required to expand the following proposal to the two-pattern test generators, presented in [12]. This extension will increase the required hardware overhead in the memory utilization and will increase the required number of the clock cycles.

Finally, in the EEPROM test, the large delay (20 ms) between the written data cycle and the read data cycle increases the required number of clock cycles to deal with the SM-HBST, shown in Table XX and Table XXI. Therefore, the EEPROM test subroutine was implemented based on a simple marsh algorithm with low time complexity and low fault coverage. From Table XX and Table XXI, there is no improvement in the memory utilization and no reduction in the total test application time for both PIC16F87X and PIC18F4X2.

VII. Conclusion

The HBST strategy is limited to test complex digital circuits such as microcontrollers that have heterogeneous components with poor accessibility. In addition, the SBST strategy is efficient in the case of complex embedded processors. However, for microcontrollers with small internal memories, the SBST is limited in the memory utilization, and cannot test all its internal modules and its GPIO pins without the external ATE.

In this paper, the integrated test strategy solution of the CUT board including a microcontroller chip with small internal memories for fault diagnosis was presented. It targets both conventional random logic ICs and a microcontroller chip on the CUT board. Conventional random logic ICs is tested based on the SM-HBST strategy. Every pin (node) in the random logic ICs is stimulated by the TPG of the SM-HBST, and the test response from a tested pin (node) is compacted for signature generation. Measured signatures are generated according to different test modes and are automatically compared to get the source faulty pins (nodes).

In addition, the new hybrid-based self-test strategy, HYBST that tests the microcontroller circuits with small internal memories was presented. It combines both the SM-HBST strategy and the SBST strategy. Based on divide-and-conquer strategy, the microcontroller is divided into a number of main modules. The ISA of the microcontroller family is used to generate test subroutines as a part of the BIST scheme, which is the emulated TPG and part of the emulated test controller. The SM-HBST represents the other part of the BIST scheme as the test controller and the test response compaction. Each microcontroller module is exhaustively tested and all test patterns detect all combinational faults without fault simulator. The comparisons between the SBST strategy and the HYBST strategy show that the HYBST strategy is more suitable for testing of microcontroller chips with small internal memories. The HYBST strategy is superior in:

1) RAM utilization; where it reduces RAM utilization.
2) Flash memory utilization; where it reduces flash memory utilization.
3) Test application time; where it reduces total number of clock cycles.
4) Testing all internal microcontroller modules, and testing of GPIO pins using the SM-HBST.

Experimental results showed that the presented test solution offers a suitable test strategy for complete digital electronic boards, composed of digital random logic ICs and microcontroller chips with small internal memories. It indicates a real practical strategy with the aid of the SM-HBST, and reduces testing difficulty to achieve high fault coverage.

References

[1] M. H. El-Mahlawy, "Pseudo-Exhaustive Built-In Self-Test for Boundary Scan", Ph.D. dissertation, Kent University, U.K., 2000. Accessed on December 19th, 2018, available at: https://www.researchgate.net/profile/Mohamed_El-Mahlawy/publication/35962900_Pseudo-exhaustive_built-in_self-test_for_boundary_scan/links/58931820a6fdcc45530af29c/Pseudo-exhaustive-built-in-self-test-for-boundary-scan.pdf

[2] N. K. Jha and S. Gupta, *Testing of Digital Systems*, Cambridge University Press, 2003.

[3] A. Miczo, *Digital Logic Testing and Simulation*, John Wiley & Sons, 2003.

[4] M. El Said Gohniemy, S. Fadel Bahgat, Mohamed H. El-Mahlawy, and E. E. M. Zouelfoukkar, "A Novel Microcomputer Based Digital Automatic Testing Equipment using Signature Analysis," *IEEE International conference on Industrial Applications in Power Systems, Computer Science and Telecommunications*, pp. 140-144, Bari, Italy, May 13-16, 1996.

[5] M. Abramovici, M. A. Breuer, A. D. Friedman, *Digital systems testing and testable design*, Wiley-IEEE Press, New York, 1994.

[6] S. I. Morsy, "Hybrid Self-Test Solution in Embedded System on Chip", M.Sc. thesis, Military Technical College, Cairo, Egypt, 2012.

[7] M. H. El-Mahlawy, Ehab A. El-Sehely, Al-Emam S. Ragab, and Sherif Anas, "Design and Implementation of a New Built-In Self-Test Boundary Scan Architecture." *IEEE 15th International conference on Microelectronics*, pp. 27-31, Dec. 9-11, 2003.

[8] D.K. Sharma, Meerut, R.K. Sharma, B.K. Kaushik, and Pankaj Kumar, "Boundary scan based testing algorithm to detect interconnect faults in printed circuit boards," *Circuit World*, vol. 37, no, 3, pp. 27–34, 2011.

[9] M. H. El-Mahlawy, "Signature Multi-Mode Hardware-Based Self-Test Architecture for Digital Integrated Circuits," *IEEE International Conference on Electronics, Circuits, & Systems*, pp. 437-441, Dec. 6-9, 2015.

[10] M. H. El-Mahlawy, and Ahamed Seddik, "Design and Implementation of New Automatic Testing System For Digital Circuits Based on The Signature Analysis," *12th International Conference on Aerospace Sciences & Aviation Technology*, Military Technical College, Egypt, May 29-31, 2007.

[11] M. H. El-Mahlawy, Ahmed Abd El-Wahab, and Al-Emam S. Ragab, "FPGA Implementation of the Portable Automatic Testing System for Digital Circuits," *6th International Conference of the Electrical Engineering*, Military Technical College, Egypt, May 27-29, 2008.

[12] M. H. El-Mahlawy, Emad H. Khalil, Fawzy Ibrahim, and Mohamed. H. Abd El-Azeem, "Two-Test Pattern Capabilities of the LFSR/SR Generator in Pseudo-Exhaustive Testing based on Coding Theory Principles," *European Journal of Scientific Research*, vol. 140, no. 2, pp. 161-177, July 2016.

[13] Mohamed H. El-Mahlawy, and Winston Waller, "New Test Pattern Generators for the BIST Pseudo-Exhaustive Testing based on Coding Theory Principles," *Communications on Applied Electronics*, vol. 4, no. 8, pp. 29-44, April 2016.

[14] P. K. Lala, *Digital circuit testing and testability*, Academic Press, 1997.

[15] A. Krstic, and Kwang-Ting (Tim) Cheng, *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Publishers, 1998.

[16] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Bounds and analysis of aliasing errors in linear feedback shift register," *IEEE transactions on computer-aided design*, vol. 7, no. 1, pp 75-83, Jan. 1988.

[17] M. H. El-Mahlawy, "Signature-Based Self-Test Approach for Single-Shot Circuits on the Circuit Board Level," *4th International Japan-Egypt Conference on Electronics, Communications and Computers* (*JEC-ECC*

2016), pp. 38-42, May 31 - June 2, 2016.

[18] M. Psarakis, Dimitris Gizopoulos, Ernesto Sanchez and Matteo S. Reorda, "Microprocessor Software-Based Self-Testing," *IEEE Design & Test of Computers*, vol. 27, no. 3, pp. 4-19, May/June, 2010.

[19] J. Shen and J. A. Abraham, "Native Mode Functional Test Generation for Processors with Applications to Self-Test and Design Validation," *IEEE International Test Conference* (*ITC*), pp. 990-999, Oct. 18-23, 1998.

[20] Hunger and A. Gaertner., "Functional Charaterization of Microprocessors," *IEEE International Test Conference* (*ITC*), pp. 794-803, 1984.

[21] J. V. De-Goor, and O. Jansen, "Self-Test for the Intel 8085," *Microprocessing and Microprogramming*, vol. 29, no. 3, pp. 165-175, Oct. 1990.

[22] D. Brahme and J. A. Abraham, "Functional Testing of Microprocessors," *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 475–485, June 1984.

[23] S. M. Thatte and J. A. Abraham, "Test Generation for Microprocessors," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 429-441, June 1980.

[24] R. Velazco, C. Bellon, and H. Ziade, "Analysis of Experimental Results on Functional Testing and Diagnosis of Complex Circuits," *IEEE International Test Conference* (*ITC*), pp. 64-72, Sep. 1988.

[25] L. Chen and S. Dey, "Software-Based Self-Testing Methodology for Processor Cores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 369–380, March 2001.

[26] N. Kranitis, G. Xenoulis, A. Paschalis, D. Gizopoulos, and Y. Zorian, "Application and Analysis of RT-level Software-Based Self-Testing for Embedded Processor Cores," *IEEE International Test Conference* (*ITC*), vol. 1, pp. 431-440, Sept. 30-Oct. 2, 2003.

[27] N. Kranitis, A. Merentitis, G. Theodorou, D. Gizopoulos, and A. Paschalis, "Hybrid-SBST Methodology for Efficient Testing of Processor Cores," *IEEE Design & Test of Computers*, vol. 25, no. 1, pp. 64-75, Jan.-Feb. 2008.

[28] L. Terng, W. C. E. Stroud, and N. A. Touba, *System-on-Chip Test Architectures - Nanometer Design for Testability*, Elsevier, 2008.

[29] N. Kranitis, G. Xenoulis, D. Gizopoulos, A. Paschalis, and Y. Zorian, " Low-Cost Software-Based Self-Testing of RISC Processor Cores," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (*DATE'03*), Munich, Germany, March 3-7, 2003.

[30] D. V. Kodavade, and S.D. Apte, "An Intelligent Framework for Fault Diagnosis in 89c51RD2 Microcontroller Based System," *IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pp. 283- 286, Italy, Sep. 21-23, 2009.

[31] M. Predko, *Programming and Customizing PICmicro Microcontrollers*, 3rd edition, Toronto, Canada, McGraw-Hill Companies Inc., 2008.

[32] M. H. El-Mahlawy, Sherif Anis, Mahmoud E. A. Gadallah, and Emad A. El-Samahy, "FPGA-Based Implementation of the Digital Testing of Analogue Circuits," *European Journal of Scientific Research*, vol. 138, no. 4, pp. 256-286, April 2016.

[33] M. S. Saleh, Mohamed H. El-Mahlawy and Hossam E. Abou-Bakr Hassan, "Digital Signature Based Test of Analogue Circuits Using Amplitude Modulated Multi-Tone Signals" *IEEE 28th International conference on Microelectronics*, pp. 117-120, Dec. 17-20, 2016.

[34] A. Mousa, and M. H. El-Mahlawy, "Test Pattern Generator Optimization for Digital Testing of Analogue Circuits" IEEE 8th International Conference on Intelligent Computing and Information Systems (ICICIS 2017), pp. 118-126, Dec. 5-7, 2017.

[35] J. V. De-Goor and Z. Al-Ars, "Functional Memory Faults: A Formal Notation and a Taxonomy," *18th IEEE VLSI Test Symposium*, pp. 281-289, 30 Apr-04 May, 2000.

[36] S. Hamdioui1, A. J. van de Goor, and M. Rodgers, "March SS: A Test for All Static Simple RAM Faults," *IEEE International Workshop on Memory Technology Design and Testing*, pp. 95-100, 2002.

[37] S. D. Carlo, A. Bosio, G. D. Natale and P. Prinetto, "March AB, a State-of-the-Art March Test for Realistic Static Linked Faults and Dynamic Faults in SRAMs," *IET Computer & Digital Techniques*, vol. 1, no. 3, pp. 237-245, May 2007.

[38] M. S. Ragab, Mohamed H. El-Mahlawy and Emad A. El-Samahy, "Efficient Microcontroller System to Test an SRAM Chip Using Signature Analysis," *13th International Computer Engineering Conference* (*ICENCO*), Dec. 27-28, 2017.

[39] I. El-Kabbey, *Testing of Electronic Boards*, M.Sc. thesis, Military Technical College, Cairo, Egypt, 2008.

[40] M. H. El-Mahlawy, Mahmoud S. Hamed, Mohamed H. Abd-El-Zeem, and Issa Yossef, "FPGA Implementation of the BIST IP For SRAM Chips", *6th International Conference of the Electrical Engineering*, Military Technical College, Egypt, May 27-29, 2008.

[41] J. M. Rabaey, *Digital Integrated Circuits — A Design Perspective*, 2nd edition, Prentice Hall, 2003.