Implementation of UART with BIST Technique in FPGA

Bibin M C, Premananda B S

Abstract - Asynchronous serial communication is usually implemented by Universal Asynchronous Receiver Transmitter (UART), mostly used for short distance, low speed, low cost data exchange between processor and peripherals. UART allows full duplex serial communication link, and is used in data communication and control system. There is a need for realizing the UART function in a single or a very few chips. Further, design systems without full testability are open to the increased possibility of product failures and missed market opportunities. Also, there is a need to ensure the data transfer is error proof. This paper targets the introduction of Built-in self test (BIST) and Status register to UART, to overcome the above two constraints of testability and data integrity. The 8-bit UART with status register and BIST module is coded in Verilog HDL and synthesized and simulated using Xilinx XST and ISim version 14.4 and realized on FPGA. The results indicate that this model eliminates the need for higher end, expensive testers and thereby it can reduce the development time and cost.

Index Terms - UART, BIST, Error check, Status register, LFSR.

I. INTRODUCTION

Asynchronous serial Communication has advantages of less transmission lines, high reliability and long transmission distance. UART allows full-duplex communication in serial link, thus has been widely used in the data communications and control system. It is widely used in data exchange between Processor and peripherals. UART converts data from parallel to serial at transmitter with some extra overhead bits using shift register and vice versa at receiver. To the processor the UART appears as an 8-bit read/write parallel port [1], [2].

Basic UART communication needs only two signal lines (Receive, Transmit) to complete full-duplex data communication [2]. UART includes three modules namely, the baud rate generator, receiver and transmitter. The baud rate generator is used to produce a local clock signal which is much higher than the baud rate to control the UART receive and transmit; The UART receiver module is used to receive the serial signals at RXD, and convert them into parallel data; The UART transmit module converts the bytes into serial bits according to the basic frame format and transmits those bits through TXD.

When the transmitter is idle, the data line is in the high logic state. Otherwise when a word is given to the UART for asynchronous transmissions, "Start Bit" (logic low) is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the peripheral receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. After the Start Bit, the individual data bits of the word are sent, with the Least Significant Bit (LSB) being sent first. Each bit is transmitted for exactly the same amount of time as all of the other bits, and the receiver samples at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. When the entire data word has been sent, the transmitter adds a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the frame, it automatically discards the Start, Parity and Stop bits. If another word is ready for transmission, the Start bit for the new word can be sent as soon as the Stop bit for the previous word has been sent. Asynchronous data are "self-synchronizing" if there are no data to transmit, the transmission line is held idle.

In actual applications, usually only a few key features of UART are needed. Specific interface chip will cause waste of resources and increased cost [2]. Particularly in the field of electronic design, SoC technology is recently becoming increasingly mature. This situation results in the requirement of realizing the whole system function in a single or a very few chips. Integration of only core functions into a FPGA chip to achieve compact, stable and reliable data transmission avoids waste of resources and decrease cost.

Manufacturing processes are extremely complex, inducing manufacturers to consider testability as a requirement to assure the reliability and the functionality of each of their designed circuits [3], [7]. Testing of integrated circuits (ICs) is important to ensure a high level of quality in product functionality in both commercially and privately produced products. In the modern System-on-a-Chip (SoC) design, many cores are integrated into a single chip. Some of them are embedded, and cannot be accessed directly from the outside of the chip. Such SoC designs make the test of these embedded cores a great challenge. As ICs grow in gate counts, it is no longer true that most gate nodes are directly accessible by one of the pins on the package. This makes testing of internal nodes more difficult as they could neither no longer be easily controlled by signal from an input pin (controllability) nor easily observed at an output pin

Manuscript Received on July 2013
  Bibin MC, M-Tech, VLSI Design and Embedded Systems, VTU Extension Centre, UTL Technologies Ltd., Bangalore, India.
  Premananda B.S., Asst. Prof., VLSI Design and Embedded Systems, VTU Extension Centre, UTL Technologies Ltd., Bangalore, India.

(observe ability). Hence Internal Diagnostic Capabilities are to be introduced to test the embedded cores.

In this paper, internal diagnostic capabilities are built into UART by the introduction of Built-In-Self-Test (BIST) [3] and error simulation of data at receiver for any data corruption and thereby setting status flags [1]. The UART with status register and BIST module is coded in Verilog HDL and simulated using Xilinx tool ISESIM. The complete implementation and validation is done on Spartan 3A FPGA.

The paper is organized into 5 sections. Section 2 explains BIST technique and implementation possibilities. Section 3 explains the architecture of UART with introduction of BIST technique. Section 4 presents the results and section 5 provides the conclusion of work.

II. BIST TECHNIQUE

VLSI testing problems like Test generation problems, input combinatorial problems, gate to I/O pin ratio problems are discussed [3] and this motivated designers to identify reliable test methods in solving these difficulties. An insertion of special test circuitry on the VLSI circuit that allows efficient test coverage is the answer to the matter. This has been addressed by the need for design for testability (DFT) and hence the need for BIST. This is to specify test as one of the system functions and thus becomes self-test. BIST is an on-chip test logic that is utilized to test the functional logic of a chip, by itself. With the rapid increase in the design complexity, BIST has become a major design consideration in DFT methods and is becoming increasingly important in today's state of the art SoCs.

A properly designed BIST is able to offset the cost of added test hardware while at the same time ensuring the reliability, testability and reduces maintenance cost. BIST solution consists of a Test Pattern Generator (TPG), the circuit to be tested, a way to analyze the results, and a way to compress those results for simplicity and handling. Fig. 1 shows a BIST module composition. Generic BIST architecture components [7] are;

Circuit under Test (CUT): This is the portion of the circuit tested in BIST mode. It can be sequential, combinational or a memory. It is delimited by their Primary Input (PI) and Primary Output (PO).

Test Pattern Generator (TPG): It generates the test patterns for the CUT. It is a dedicated circuit or a microprocessor. The patterns may be generated in pseudorandom or deterministically. Normally, the pattern generator generates exhaustive input test patterns to the CUT to ensure the high fault coverage. For example, a CUT with 10 inputs will require 1024 test patterns.

Test Response Analysis (TRA): It analyses the value sequence on PO and compares it with the expected output.
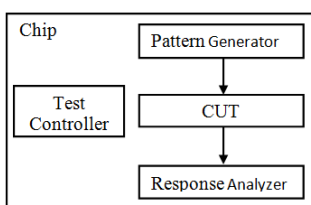

Fig. 1: A generic BIST module

BIST Controller Unit (BCU): It controls the test execution; it manages the TPG, TRA and reconfigures the CUT and the multiplexer. It is activated by the Normal/Test signal and generates a Go/No-Go. During BIST mode, it selects input from the pattern generator to CUT while during functional mode, selects primary inputs.

There are various approaches being used to generate test patterns for BIST [6], [7], viz. ROM, LFSR, binary counters, modified counters, cellular automation. A binary counter can generate an exhaustive but not randomized test sequences. Drawback of binary counters as the pattern generator is, it requires more hardware than typical Linear Feedback Shift Register (LFSR) pattern generator. Modified counters also have been successfully as test-pattern generators. However, they also require long test sequences. This method stores a good test pattern set in a ROM on the chip. However, drawback of this approach is relatively expensive in chip area. LFSR is used to generate pseudo-random test patterns. This normally requires a sequence of one million or more tests pattern in order to achieve high fault coverage. One of the advantages of LFSR is their compact and simple design and thus is currently the preferred BIST pattern generation method [6]. In this project, LFSR is used for test pattern generation.

III. PROPOSED UART ARCHITECTURE WITH BIST

The architecture proposes an 8-bit UART which operates at a baud rate of 9600 bps with a status register to monitor the correctness of every received data byte and enhance the testability of circuit by the introduction of BIST module. The hardware architecture of the 8-bit UART with Status register, incorporated with BIST module is explained in the following sections.

The proposed model has two major modules viz. UART and BIST. Further in the UART, we have transmitter, receiver, and baud rate generator. Baud rate generator works at 50 MHz and further reduced as required for the operations in transmitter and receiver to achieve baud rate of 9600 bps. BIST has a control register, pattern generator and a comparator, as shown in fig. 2.
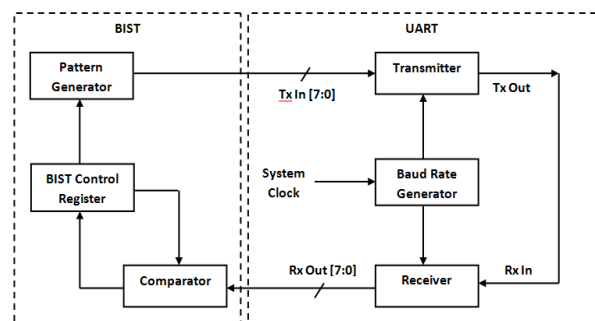

Fig. 2: UART with BIST architecture

A. UART Transmitter

The transmitter accepts parallel data from peripheral/processor, makes the frame of the data and transmits the data in serial form on the Transmitter Output (TXOUT) terminal (fig. 3). The baud rate generator output will be the clock for UART transmitter.
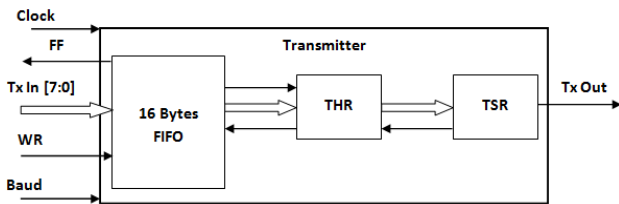
Fig. 3: UART Transmitter

Data is loaded from the parallel inputs TXIN0-TXIN7 into the Transmitter FIFO by applying logic high on the WR (Write) input. FIFO is 16-byte register. If FIFO is full, it sends FIFO Full (FF) signal to peripheral as shown in fig. 4.
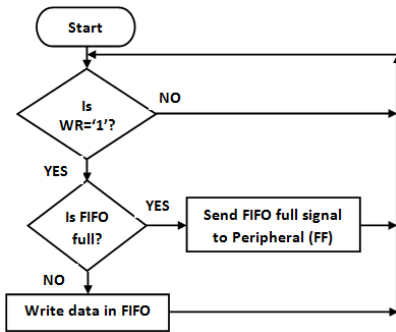

Fig. 4: Transmitter flowchart – Input to FIFO

When FIFO contains some data, it will send the signal to Transmitter Hold Register (THR), which is an 8-bit register. At the same time, if THR is empty it will send the signal to FIFO, which indicates that THR is ready to receive data from FIFO. If Transmitter Shift Register (TSR) is empty, it will send the signal to THR and it indicates that TSR is ready to receive data from THR. TSR is an 11-bit register in which framing process occurs. In frame, start bit, parity bit and one stop bit will be added as shown in fig. 6. Now data is transmitted from TSR to TXOUT serially. Fig. 5 is the flowchart explaining transmission of serial data from FIFO to transmitter output.
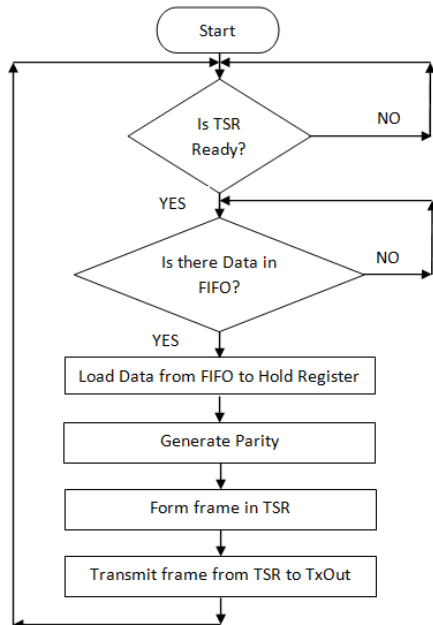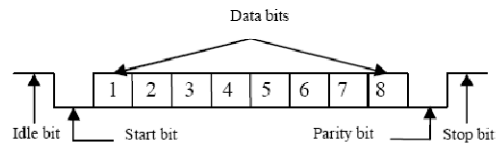

Fig. 5: Transmitter flowchart – FIFO to TXOUT

Fig. 6: UART Frame Format

B. UART Receiver

The received serial data is available on the RXIN pin. The received data is applied to the sampling logic block. The receiver timing and control is used for synchronization of clock signal between transmitter and receiver. The receiver sampling is 16 times to that of the transmitter baud rate. In the architecture of UART receiver (fig. 7), initially the logic line (RxIn) is high.
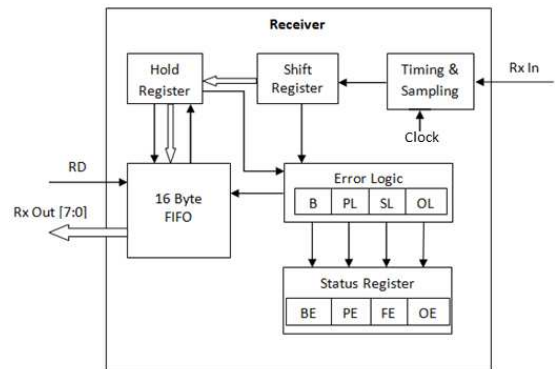

Fig. 7: UART Receiver

Whenever it goes low sampling and logic block will take 4 samples of that bit and if all four are same it indicates the start of a frame. After that remaining bits are sampled in the same way and all the bits are send to Receiver Shift Register (RSR) one by one where the entire frame is stored. RSR is a 12 bit shift register. Fig. 8 shows the receiver logic.
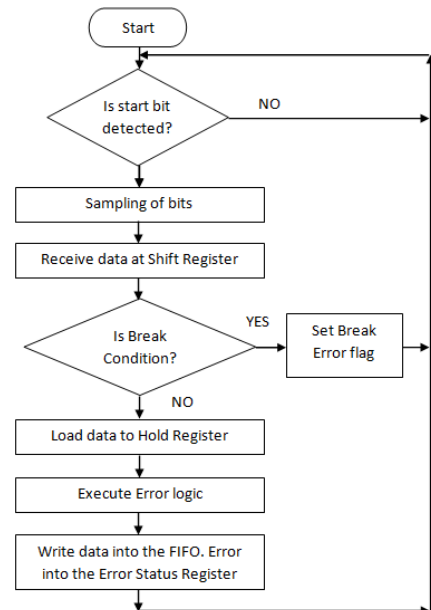

Fig. 8: Receiver flowchart (Input to FIFO)

Now if the Receiver Hold Register (RHR) is empty it sends signal to RSR so that only the data bits from RSR goes to RHR which is an 8 bit register. The remaining bits in the RSR are used by the error logic block. Then, if receiver

FIFO is empty it send the signal to RHR so that the data bits goes to FIFO. When RD signal is asserted the data is available in parallel form on the RXOUT0-RXOUT7 pins.

The status register is implemented with flags for error logic operations performed on the received data. The error logic block handles 4 types of errors: Parity error (PE), Frame error (FE), Overrun error (OE), Break error (BE). If the received parity does not match with the parity generated from data bits, PE bit will be set which indicates that parity error occurred. If receiver fails to detect correct stop bit or when 4 samples do not match frame error occurs and FE bit is set. If the receiver FIFO is full and other data arrives at, RHR overrun error occurs and OE bit is set. If the RXIN pin is held low for long time than the frame time then there is a break in received data and break error occurs and BE bit is set. Reading of data from receiver is explained by means of flowchart in fig. 9.
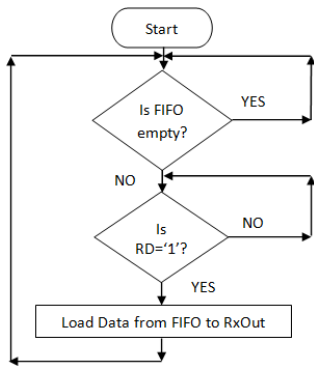

Fig. 9: Receiver flowchart (FIFO to Output)

C. BIST Pattern Generator

LFSR is used to generate pseudo-random test pattern for the BIST. A LFSR is a shift register where the input is a linear function of two or more bits (taps). It consists of D flip-flops and linear exclusive-OR (XOR) gates.

The bits contained in selected positions in the shift register are combined in some sort of function and the result is fed back into the register's input bit. The selected bit values are collected before the register is clocked and the result of the feedback function is inserted into the shift register during the shift, filling the position that is emptied as a result of the shift.

The bit positions selected for use in the feedback function are called "taps". The list of the taps is known as the "tap sequence". The largest state space possible for such an LFSR will be 2^{n-1}, all possible values except the zero state. All zero is not allowed in LFSR, as it will always produce 0 in spite of how many clock iteration. Because each state can have only once succeeding state, an LFSR with a maximal length tap sequence will pass through every non-zero state once and only once before repeating a state.

D. BIST Operation

For BIST, UART is set in an internal loop back mode (fig. 2). This is used to test both the transmitter and receiver of the UART. This will loop-back the serial data and transmit the data back to the receiver.

For the BIST, the test pattern is generated by LFSR as mentioned in the last section and the pattern is loaded to the FIFO of the UART transmitter. Each test byte is then padded with start, parity and stop bits and sent from transmitter and is looped back to receiver. The receiver will extract the data from frames received and loads to receiver FIFO. During this the status register flags will be set according to the error check. Then the Tx FIFO is compared with Rx FIFO to verify the transmitted data and received data are same. If FIFOs are with same data then BIST pass and Bit-0 of BIST control register is set to '1', else '0'. 8-bit BIST control register is defined as below. The register helps in identifying the operation in which BIST failure occurred.

Table I: BIST Control Register

B7	When set, BIST starts
B6	Sets when LFSR pattern generation is over and Tx FIFO is loaded
B5	Sets when Tx test pattern is transmitted
B4	Sets when Rx FIFO is loaded with received data
B3	Sets when comparison starts
B2	Sets if any error flag is set in status register
B1	No function
B0	1 if BIST pass, 0 if BIST fails

IV. SIMULATION RESULTS

The verilog HDL coding and simulation of the design are done in Xilinx tool ISim 14.4. The operating clock frequency used for simulation is 50 MHz. The baud rate set is 9600bps. Data word length is 8-bits.

A. Simulation Results of Transmitter

The fig. 10 shows the serial transmission of data. Data transmitted is "10101010". This 8-bit data is loaded to transmit shift register and start, stop & parity bits are added to form the frame inside TSR and sent to TXD. When the reset is 0 and transmit is 1, the transmitter starts transmitting the data. i.e. the data starts shifting out from the transmitter shift register. Since the desired baud rate is 9600bps, the bits are shifted out on TxD line at the interval of 50MHz/9600=5208 clock cycles. Similarly all the bits are sent. The serial transmission is observed at TXD pin along with frame format (1 logical low start bit, 8-bit data (LSB to MSB), parity bit and finally logical high stop bit).

Fig. 10: Simulation result of UART transmitter

B. Simulation Results of Receiver

The UART receiver converts the serial data into parallel form and makes it available at RxData[7:0]. The Serial data is received at RXD pin. Each bit is sampled and the sampled bit is saved into receive shift register. From this, the frame bits viz. start, parity and stop bits are discarded in RSR and written to receive FIFO, RxData. The 8-bit data simulated is "11111111". Further received data will be stored in the

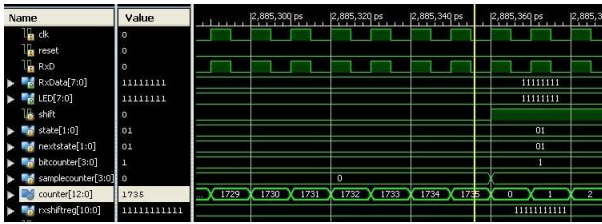remaining FIFO locations. Fig. 11 shows the reception of serial data.

Fig. 11: Simulation result of UART receiver

C. Simulation Results of BIST Operation

The fig. 12 shows the simulation result of BIST module. The pattern generated by LFSR is transmitted from transmitter and received in receive FIFO.
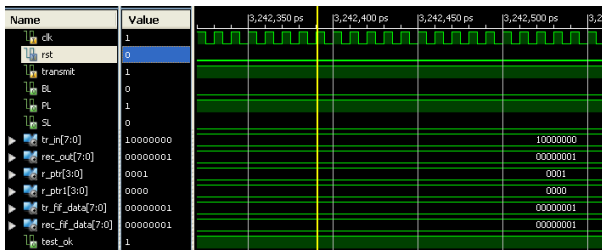

Fig. 12: Simulation result of BIST module

The transmit FIFO is compared with the receive FIFO after the reception of each word. The comparison starts after the delay of transmitter for sending the data and receiver for reception of the data. In this simulation, the first byte sent is "00000001" which matches in both FIFO; hence the test status bit is set '1'.

D. Synthesis Report

Table II shows the design implementation summary of the proposed architecture

Design Utilization Summary	
Selected Device	3s500efg320-4
Number of Slices	513 out of 4656 11%
Number of Slice Flip Flops	626 out of 9312 6%
Number of 4 input LUTs	819 out of 9312 8%
Number used as logic	815
Number used as Shift registers	4
Number of IOs	16
Number of bonded IOBs	16 out of 232 6%
Number of GCLKs	1 out of 24 4%
Timing Summary	
Speed Grade	-4
Minimum period	10.592ns (Max. Frequency: 94.411MHz)
Minimum input arrival time before clock	6.232ns
Maximum output required time after clock	7.661ns
Maximum combinational path delay	No path found

V. CONCLUSION

The architecture of UART that support 8-bit data word length at 9600 bps baud rate for serial transmission of data with the addition of status register for detecting errors in data transfer and BIST which allows to test the circuit itself, is introduced. Working of UART has been tested using Xilinx ISE simulator, which is implemented on FPGA. With error checking status register, we can detect the different types of errors occurred during communication and hence correct them.

With the implementation of BIST, expensive tester requirements and testing procedures starting from circuit or logic level to field level testing are minimized. The LFSR replaces the function of the external tester features such as a test pattern generator by automatically generating pseudo random patterns to give good fault coverage to the UART module. Although the additional BIST circuit increases the hardware overhead and design time, it eliminates the need to acquire high-end testers. The reduction of the test cost helps in the reduction of overall production cost.

REFERENCES

[1] Naresh, Vatsalkumar and Vikaskumar Patel, "VHDL Implementation of UART with Status Register", *in the proceedings of International Conference on Communication Systems and Network Technologies, IEEE Computer Society*, 11-13th *May 2012,* DOI: 10.1109/CSNT.2012.164, pp.750-754.

[2] Fang Yi-yuan and Chen Xue-jun, "Design and Simulation of UART Serial Communication Module Based on VHDL", *in the proceedings of 3rd International Workshop on Intelligent Systems and Applications (ISA), IEEE, May 2011*, DOI: 10.1109/ISA.2011.5873448, pp.1-4.

[3] Mohd Yamani Idna Idris, Mashkuri Yaacob and Zaidi Razak, "A VHDL Implementation of UART Design with BIST Capability", *in the proceedings of Malaysian Journal of Computer Science*, *June 2006,* Vol. 19(1), pp. 73-86.

[4] Dr. Garima Bandhawarkar Wakhle, Iti Aggarwal and Shweta Gaba, "Synthesis and Implementation of UART using VHDL Codes", *in the proceedings of International Symposium on Computer, Consumer and Control*, *IEEE June 2012*, DOI: 10.1109/IS3C.2012.10.

[5] Norhuzaimin J and Maimun H.H, "The design of high speed UART", *in the proceedings of Asia-Pacific Conference on Applied Electromagnetics, APACE 05, IEEE, 20-21st Dec. 2005,* DOI: 10.1109/APACE.2005.1607831, pp.5-8.

[6] Dr. T.V.S.P.Gupta, Y. Kumari and M. Ashok Kumar, "UART realization with BIST architecture using VHDL", *in the proceedings of International Journal of Engineering Research and Applications, February 2013,* Vol. 3, Issue 1, ISSN: 2248-9622, pp.636-640.

[7] Shikha Kakar, Balwinder Singh and Arun Khosla, "Implementation of BIST Capability using LFSR Techniques in UART", *in the proceedings of International Journal of Recent Trends in Engineering, May 2009,* Vol 1, No. 3.

[8] Sybille Hellebrand, Birgit Reeb and Steffen Tarnick, "Pattern Generation for a Deterministic BIST Scheme", *in the proceedings of IEEE/ACM International Conference on Computer-Aided Design, ICCAD-95, Digest of Technical Papers, November 1995,* DOI: 10.1109/ICCAD.1995.479997, pp. 88-94.

[9] Mahat N.F, "Design of a 9-bit UART module based on Verilog HDL", *in the proceedings of 10th IEEE International Conference on Semiconductor Electronics (ICSE), 19-21st Sept. 2012,* DOI: 10.1109/SMElec.2012.6417210, pp. 570-573.