Application of DFT on UART Implmenatation

BIST Architecture

Madhura R[1], Sharanbsav A[2], Vaibhav Krishnan[3], A.G Sai Tejas[4], Akshita Deep[5]
Professor[1], Dept. of ECE
Dayananda Sagar College of Engineering
Shavige Malleshwara Hills, 91st Main Rd, 1st Stage, Kumaraswamy Layout
Bengaluru, India

Abstract *— Asynchronous serial communication is sometimes enforced by Universal Asynchronous Receiver Transmitter (UART), largely used for brief distance, low speed, low value information exchange between processor and peripherals. UART permits full duplex serial communication link, and is employed in electronic communication and system. there's a desire for realizing the UART perform a single chip. Further, style systems not full testability is hospitable the exaggerated chance of product failures and incomprehensible market opportunities. Also, there's a desire to confirm the info transfer is error proof. This targets the introduction of intrinsic self-take a look at (BIST) and standing register to UART, to beat the on top of 2 constraints of testability and information integrity. The 8-bit UART with standing register and BIST module is coded in Verilog HDL and synthesized and simulated using Vivado Hlx design edition and van be realized on FPGA if required. The results indicate that this model eliminates the requirement for higher finish, pricey testers and thereby it will scale back the event time and price.*

The expanding development of sub-micron innovation has brought about the trouble of testing. Design and test engineers must choose the option to acknowledge new obligations that had been performed by gatherings of experts in the earlier years. Design engineers who don't plan frameworks in light of full testability open themselves to the expanded chance of item disappointments and botched market chances. BIST is a plan strategy that permits a circuit to test itself. the test execution accomplished with the usage of BIST is demonstrated to be sufficient to balance the disincentive of the equipment overhead delivered by the extra BIST circuit. The procedure can give shorter test time contrasted with a remotely applied test and permits the utilization of minimal effort test gear during all phases of creation

Keywords *— UART, BIST, Error check, Status register, LFSR*

I. INTRODUCTION

A. Overview

The universal asynchronous receive/transmit technique is made public on mere information measure with a sampling procedure to comprehend economical communication among systems. A UART is an electrical circuit that plays the foremost necessary role in connecting the synchronous digital processors to the asynchronous devices by corporal punishment the communication procedure. It mainly handles the conversion between serial and parallel information. The incorporated LFSR worries among the sharing of signature bits along with information bits. The method presented for LFSR coming up with is made in logic block observer (BILBO). It offers advantages concerning fault coverage and detection of delay faults. BILBO offers quick testing in distinction to associate degree externally tailored analysis of faults and permits the adoption of economical to take a glance at devices' overall levels of fabrication. The UART ordinarily doesn't develop or receive the output signals used among different part of device. freelance interface instrumentation is accustomed switch the logic level signals of the UART to the external sign levels and also the different method around. External signals may be of the various fully completely different forms. samples of standards for voltage signs are RS-232, RS-422 and RS-485 from the EIA. Communication might even be simplex (unidirectional entirely, with no arrangement for the receiving device to send data back to the broadcasting device), full-duplex (both devices send and receive at identical times) or duplex (devices alternately transmits and receive). the start bit informs the receiver that a replacement data frame is inward. future bits, based on the code set selected, belongs to the information, the next one or two bits are consistently among the high logic level (standing and referred to as the stop bit(s). They inform the receiver that the information frame is finished.

B. Literature Survey

The first paper by T.D. Manoj Kumar Reddy, [2] focused on the implementation of UART on FPGA Spartan-3 using JTAG programmer and rs-422 transceiver. It basically focused on reduction of

FPGA resources and utilization. The communication over the UART was done in context to devices such as printer, scanner etc. and therefore the RS-323 was used. The UART can be used to control the process of breaking parallel data from the PC down into serial data that can be transmitted and vice versa for receiving data. The UART was interfaced using the NVRAM and computer using a 10MHz clock and power source of 28V supply. This 28 V was converted into 5V which was later converted to 3.3 V for the TTL logic.

The next paper by Soumya K and M Ramakrishna [3] focuses on the increased number of failures due to lack of testing and therefore introducing the concept of BIST to overcome testability and integrity. This was done by implementing the test pattern generator to the circuit under test to check the output for random inputs. The output was analyzed using the test response analyzer for the reliability. It focused on Verilog implementation for the 8-bit UART with status register.

The next work by D.Monica satyavathi, G.Anjaneyulu [4] displays the implementation of BIST technique in UART serial communication that was simulated and synthesized using Xilinx and model-sim 12.3 version. It takes into account the usage of error checking and thus this error checking at the receiver helps generate better efficiencies.

The work by Akshata Surendra, Sandra Benjamin and M Z Kurian [5] focuses on the reduction of failures in the nanotechnology based. this is basically based on the usage of LFSR in the place of external test pattern generator by generating random numbers and increase fault coverage. It also describes the BIST architecture taking LFSR into account.

The next work by Neeraj Pawar [7] establishes the need of MISR for fault recognition and durability as a part of increasing efficiency and thus refining the testing property within the UART. It compares the fault coverage in both normal mode of the LFSR with the fault coverage in the MISR mode.

Next work by V. Thirunavukkarasu, R.Saravanan and V.Saminadan [8] describes that generally BIST for UART consists of test patterns generated from a normal LFSR. But, bit swapping LFSR are different. BS-LFSR consists of shift registers, MUXs and XOR gates. The outputs from the BS-LFSR are similar to the normal LFSR, but the sequencing is different. There is a reduction of the power due to lesser switching states in BS-LFSR compared to normal LFSR, thus increasing the efficiency of the testing process.

The next work by Lee Zhi Yong [9] is integration of UART into existing RISC32 processor. An Interrupt Service Routine (ISR) was generated to handle the data received by UART. This was then synthesized on FPGA.

Another research by Mr.S.N.Shettennavar, Mr.B.N.Sachidanand, Mr.D.K.Gupta, Mr.V.M.Metigoudar [10] carried out the same BIST operation using an external test generator using ROM, binary counters, LFSR, and the cellular automaton. This included an extra benefit considering the CRC check done. This was done on the cost of the extra external hardware that was being used.

The work done by E. Raghuveera, K Hari Kishore, Shaik Shoukat Vali, and G. Siri Vennela [11] emerged with the BIST technology with the operation mode. Using this two processes were to be done. It can either just transfer the data serially using the normal UART operation or else can be used for the testing using the pseudo-random patterns or then check it using the MISR.

The work by P. Ramesh, Dr. D.N Rao, Dr. K. Srinivasa Rao [12] focuses on use of Multiple Polynomial LFSR since these provide patterns with higher randomness and perform better in detection of faults and the results were compared for the LFSR and multiple polynomials LFSR. This is done using low power ATPG method for efficient capture of power reduction scan testing and thus gives us better values using the test scans. The multiple Polynomial LFSR gave better results since the normal LFSR incurred little area overhead.

The next paper by Yogesh Kumar, Neeraj Pawar [13] focuses on the usage of expanded MISR for UART. It also discusses examination of error. Thus, for improving steadiness and the quickness of error detection of data, they embed the examining property within the universal asynchronous receive/transmit (UART) chip.

The next work by S Santhi Priya, G Ravikishore [14] explains that writing computer programs is sufficient to be advantageous than the extra equipment needed in BIST design. This system creates irregular test design naturally, so it can give less test time contrasted with a remotely connected test example and accomplishes considerably more profitability toward the end.

The work by V K Ivanov and E V Nosov [15] represents the basics of any serial transmission protocol implementation on the FPGA systems. It suggests that the developed communication protocol supported asynchronous oversampled signal transmission. It was also used for continuous transmission of the data over fiber optic cable that was not possible with the conventional UART systems.

The most recent work by Abhishek Kumar, Biswajit Pandey, and D M Akbar Hussain [16] discusses the energy efficiency of the UART systems. For this they have basically used the LVCMOS IO in place of the default IO standards and thus comparing the power consumptions between the two. They observed that observed that at any frequency level, the clock power, logic Power, Signal power are almost equal in both UART, i.e. default IO Standard based UART and LVCMOS IO Standard based UART.

PAPER	METHOD EMPLOYED	ADVANTAGES	DRAWBACK
Implementation and Customization of UART	Implementation of UART on FPGA Spartan-3 using JTAG programmer and rs-422 transceiver	UART used for breaking parallel data down into serial data that can be transmitted and received	Spartan 3 is an old model and its usage in the present time is almost obsolete
NOVEL IMPLEMENTATION OF UART WITH BIST TECHNIQUE IN FPGA	Introduction of BIST to overcome testability and integrity related issues.	Test pattern generator used to test to check the output for random inputs generated.	The presence of status register may introduce many programming errors.
UART implementation with multiple input signature register for full fault coverage	Establishes the need of MISR along with UART	Usage of MISR for fault recognition and durability	Each input polynomial encounters a different divisor polynomial.
Performance of Low Power BIST Architecture for UART"	Usage of BS-LFSR in place of normal LFSR	A reduction of the power due to lesser switching states in BS-LFSR	BS-LFSR consists of shift registers, MUXs and XOR gates i.e. extra hardware.
UART DESIGN, INTEGRATION AND SYNTHESIS ON FPGA	Integration of UART into existing RISC32 processor	An Interrupt Service Routine (ISR) was generated to handle the data received by UART	Due to introduction of the ISR the rate of transmission can be delayed.
Implementation of UART with BIST Technique	BIST operation using an external test generator	Extra benefit considering the CRC check done while transmission of data	It was at the cost of the extra external hardware that was being used.
Power Reduction Testing Techniques of BIST, LFSR & ATPG for Low Power Circuits	The use of Multiple Polynomial LFSR.	The multiple Polynomial LFSR gave better results since the normal LFSR incurred little area overhead	Each input polynomial encounters a different divisor polynomial.
Implementation of built in self-test (BIST) enabled UART using FPGA for fault detection	The usage of expanded MISR for UART could be observed.	Improved steadiness and the quickness of error detection of data, as they embed (UART) chip.	The trade between size and efficiency can be seen since there is increase in size
Serial communication protocol for FPGA-based systems	The developed communication protocol for asynchronous oversampled signal transmission	It was also used for continuous transmission of the data over fiber optic cable	It was basically based on only the usage of the optic cable transmission
Low voltage Complementary Metal Oxide Semiconductor Energy efficient UART design on Spartan6 FPGA	The usage the LVCMOS IO in place of the default IO standards	LVCMOS is the simplest I/O standard - it requires no termination, and consumes no static power	Signal power are almost equal in both i.e. UART and LVCMOS IO UART

Table 1: Comparison of previous works.

C. Objective

This project revolves around the incorporation of built-in testing features in serial data transmission via UART.To increase testing speed and make the system more robust it is required to implant the testing property inside the UART chip.. Structuring starts by actualizing the UART module, numerous information signature register (LFSR) and an analyzer utilizing Verilog HDL. The modules are at that point, collected improved for leased equipment prerequisite, mimicked on the test system, orchestrated and tried on Field-Programmable Gate Array (FPGA) to confirm the plan.

Asynchronous Serial data transmission is typically executed by Universal Asynchronous Receiver Transmitter (UART), utilized for short separation, low speed, minimal effort information trade among processor and peripherals. There is a requirement for understanding the UART work in a solitary or a not many chips due to VLSI Testing issues like test design age, input combinatorial issues, and I/O pin proportion issues. Structuring frameworks without full testability are available to the expanded chance of item disappointments and botched market chances.

It is a need to guarantee the information transfer is error proof. This project presents Status Register and BIST (Built-In-Self-Test) to ensure the testability and information trustworthiness. With Implementation of BIST, costly analyzer prerequisite and testing methodology beginning from circuit level to handle level testing are limited. LFSR exhibits the capabilities of external analyzer via consequently producing pseudo arbitrary examples to give great flaw inclusion to UART module. Subsequently generally speaking creation cost is diminished.

Synthesis of the UART module on Field Programmable Gate Array (FPGA) with complete timing analyses and resource usage information.

D. Methodology

There are two sorts of structure approach are accessible, Top-down plan system and Bottom-up structure approach. In top-down plan system, the top-level portrayal of a chip is first characterized then parceled into lower level portrayals. For base up structure procedure, the leaf hubs are first characterized. The leaf hubs are then incorporated to frame a more elevated level model of the chip. This process is rehashed until the top degree of the chip is reached. Since advanced framework frequently utilizes the reflection ideas to improve the structure procedure, hence top-down plan approach is utilized in this undertaking. Top-down structure system process. This procedure will keep on rehash until the framework configuration meets the prerequisite on usefulness. In the event that the plan doesn't meet the necessity, the structure stream must be rehashed.

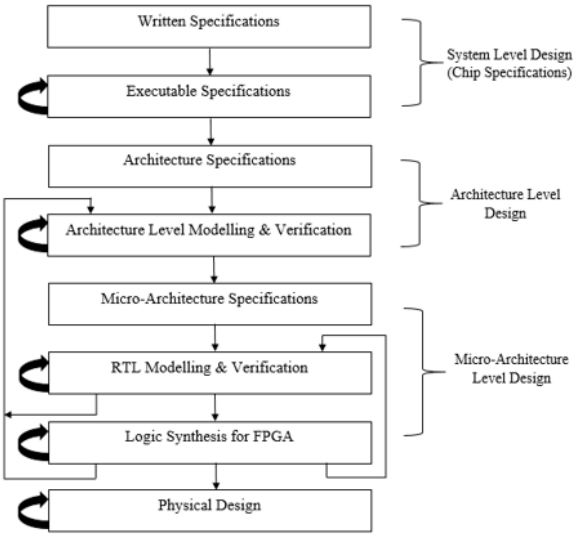


Fig 1: Top down approach of methodology.

II. BLOCK DIAGRAM AND WORKING PRINCIPLE

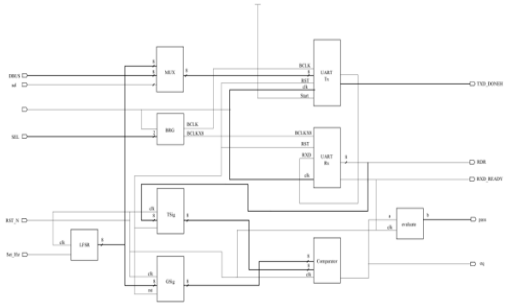


Fig 2: Block Diagram

A component that permits a machine to test itself is called BIST. It can produce designs dependent on an assortment of calculations, each centered around a specific kind of hardware or deficiency type.

Engineers design BIST to meet requirements such as:

- High Reliability
- Low repair cycle times. Main Purpose of BIST is to reduce the complexity, and thereby decrease the cost and to reduce the use of external test equipment.
  BIST reduces the cost in two ways:
- educes test-cycle duration
- The complexity of the test setup will be reduced

A. INTRODUCTION TO UART:

UART (Universal Asynchronous Receiver Transmitter) is an asynchronous serial communication device used for knowledge exchange between the processor and its peripherals. UART permits full-duplex communication within the serial link, so it has been widely utilized in information communication and system applications.

UART correspondence needs just two signs (Transmit, Receive) to finish full-duplex information correspondence. UART incorporates three modules to be specific, the baud rate generator, receiver, and transmitter. The baud rate generator is utilized to deliver a nearby clock signal which is a lot higher than the baud rate to control the UART get and transmit. The UART receiver module is utilized to get the sequential signs at RXD, and convert them into equal information; The UART transmitter module changes over the bytes into sequential bits as per the fundamental edge group and transmits those bits through TXD. Initially transmitter is idle; the TX out line is in the high logic state. In any case when a word is given to the UART for offbeat transmissions, "start bit" (rationale low) is added to the start of each word that will be transmitted. The beginning piece is utilized to alarm the fringe collector that an expression of information is going to be sent and to constrain the check in the beneficiary into synchronization with the check in the transmitter. After the beginning piece, the individual information bits of the word are sent, with the Least Significant Bit (LSB) being sent first. Each piece is transmitted for the very same measure of time as the entirety of different bits and the recipient tests at the wire at roughly part of the way through the period allotted to each piece to decide whether the bit is a 1 or a 0. At the point when the whole information word has been sent, the transmitter includes an equality bit that the transmitter produces. The Parity Bit might be utilized by the collector to perform basic blunder checking. At that point in any event one Stop Bit is sent by the transmitter to show current information is transmitted effectively and next information can be stacked for transmission.

At the point when the beneficiary has gotten the entirety of the bits in the edge, it consequently disposes of the Start, Parity and Stop bits. On the off chance that another word is prepared for transmission, the Start bit for the new word can be sent when the Stop bit for the past word has been sent. Nonconcurrent information are "self-synchronizing" if there are no information transmit, the transmission line is held inert.

B. TEST PATTERN GENERATOR

While designing any system, testing the correctness of the system's working is crucial to ensure that it is able to operate efficiently and accurately. While manual testing is a good way to ensure that the system is working properly, it proves to be extremely time consuming and impractical when the system needs to work on a large amount of data. Thus, in BIST method, we make use of a test pattern generator which automatically generates pseudo-random test patterns, which are then fed to the system, and then the final output (or intermediate, as required), is checked to verify whether the system is working properly or not. This eliminates the need to perform manual testing and, with the right amount of test pattern generation, proves to be a very accurate and efficient way of testing the system.

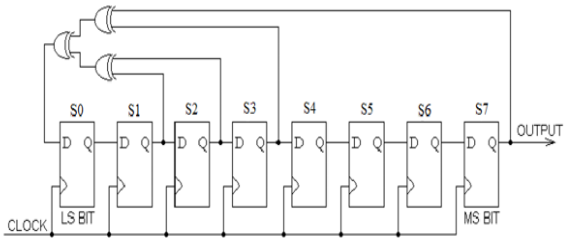


Fig 3: Test pattern Generator.

In our project, we make use of a Linear Feedback Shift Register Fig. 3 to generate various pseudo-random sequences based on the equation which governs the feedback given to the register. The shift register generates an output at each positive edge of the clock which is given to it. This output is sent to the UART transmitter, and at the same time, the

output is also sent to be stored in the Golden Signature memory, which is used later on for testing the correctness of the UART transmission.

C. TEST SIGNATURE

As mentioned before, as the number of chips on IC's and the amount of data to be fed and processed by a system keeps increasing, it becomes increasingly difficult, and presently, quite impossible to employ manual testing in the working process of the system. Hence, we employ BIST techniques, making use of an Automatic Test Pattern Generator to allow the system to test itself without any user intervention. However, for this, the test output of the system needs to be stored for further analysis. The storage area for these outputs is called a test signature.

In our project, we have made use of a Static RAM as the storage area for these outputs. It has been configured to store multiple 8-bit data at various addresses, which can then be accessed individually further on to check whether the system is working properly or not. By editing the code, we can vary the amount of test pattern outputs which can be stored at a time in this memory.

Fig 4: Test Signature Block

D. GOLDEN SIGNATURE

This is also a part of the response analysis of the system when it is operated in test mode. Here, the ideal output which should be generated by the system for a given test pattern is stored, and multiple outputs pertaining to various pseudo-random sequences are stored at a time in this memory.

Like the Test Signature, we use another Static RAM which acts as the Golden Signature in our project, which is able to store multiple 8-bit data elements, the number of which can be controlled by making changes to the underlying code. Depending on the size of the Golden and Test signatures, we can increase the fault coverage of the system while maintaining its efficiency at the same time. In this, the output of the LFSR is stored, which is nothing but the input given to the UART transmitter, when operated in test mode.

Fig 5: Golden Signature Block

E. COMPARATOR

To verify whether the system is working properly or not, the test signature needs to be compared with the golden signature. Hence, we make use of an 8-bit comparator which compares respective individual elements of the test and golden signatures, and generates a signal indicating whether they are equal or not. Based on this equality, the system gives an output indicating if it is working properly (if equality is satisfied) or not (if equality fails at any time).

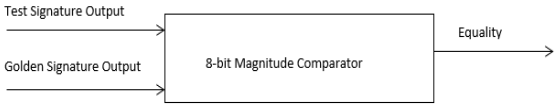


Fig 6: 8-bit comparator

F. UART TRANSMITTER

The main purpose of UART communication is to transmit data serially. But most of the data being used in practical applications is transmitted in a parallel fashion. Thus, to enable serial communication, it is the job of the transmitter to ensure that whatever data is acquired by it is transmitted to the receiver in a serial fashion.

For this purpose, we use two registers, namely a hold registers and a shift register. The input acquired by the transmitter (which could either be user input in normal mode, or test input from the test pattern generator) is loaded into the hold register at once. In the hold register, start and stop bits are added on either sides of the data, and when the transmitter is ready to send data, hold register sends its data to the shift register from where it is sent to the receiver bit-by-bit, i.e. serially.

To ensure that there is no problem of the synchronization of the transmitter and the receiver, a baud rate generator is used to control the baud rate of transmission, thus eliminating the need to accurately synchronize transmitter and receiver clocks at the same time. This leads to a reduction in the chance of any error occurring in the transmission. Additionally, the hold register can be configured to add an additional parity bit, which can be used for error detection purposes.

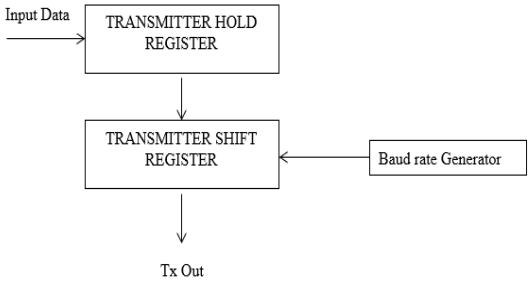


Fig 7: UART Transmitter

G. UART RECEIVER

The Receiver's function is the exact opposite of that of the transmitter. It must ensure that the serial data is received and converted into parallel form correctly. The serial data is first sent to the receiver shift register, where the start, stop and parity bits (whichever applicable) are then removed to obtain the original data fed for transmission. The shift register is synchronized accurately with the transmitter with the help of a baud rate generator, thus eliminating the need for synchronizing transmitter and receiver clocks simultaneously

This data is then fed to the receiver hold register and stored there, and from there it can be sent to the required destination in parallel. A 1-bit signal indicates whether the receiver hold register contains data to be sent to the destination or not.

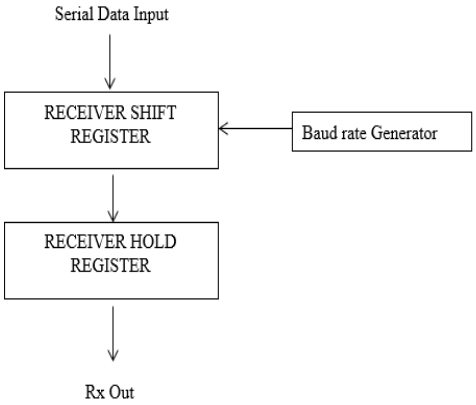


Fig 8: UART Receiver

H. MULTIPLEXER

To control the operating mode of the system, we make use of a 2:1 multiplexer as shown in the figure. A select line MUX_SEL is used to control the switching between the respective modes. When it is set to 1, the multiplexer allows the data coming from the test pattern generator the LFSR, and passes it onto the UART transmitter. This represents the testing mode of the system. If MUX_SEL=0, the user data or data coming from a previous source is allowed to pass onto the UART transmitter, thus facilitating the normal operation of the system.

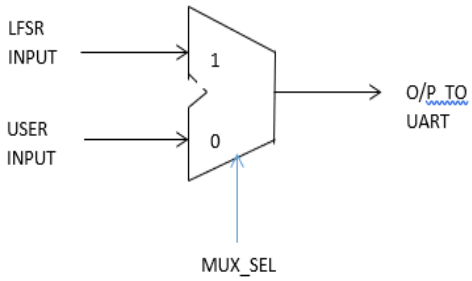


Fig 9: Multiplexer

I. BAUD RATE GENERATOR

The baud rate generator is a component present within the UART system, responsible for synchronizing the activities of the UART transmitter and receiver, without them needing the help of any external clock. This is done with the help of the pulses generated at a particular frequency by the baud rate generator, and the rate of pulse generation is measured in terms of baud rate.

The baud rate generator makes use of the system clock for its operation, and a select line has been given to choose between 4 different baud rate values (2400,4800,9600.19200). It generates 2 pulses, bclk and bclkx8, which are used for synchronization purposes. Bclk controls the rate at which data bits are transmitted by the transmitter, while bclkx8,whose frequency is 8 times greater than bclk, is sent to the receiver to be used for splitting the incoming data bits into 8 parts, so that the mid-point of each bit pulse can be measured for minimizing the possibility of transmission errors.

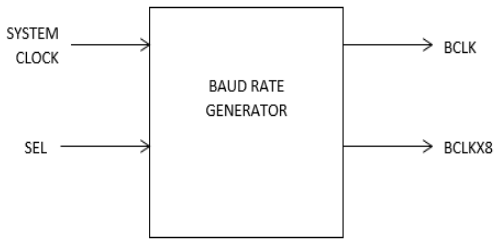


Fig 10: Baud Rate Generator

III. SOFTWARE DESCRIPTION

A UART (Universal Asynchronous Receiver Transmitter) is regularly a bit of equipment on microcontrollers or PCs that changes information among sequential and equal structures. It is utilized for sequential correspondence of information and incorporates a transmitter (which is basically a corresponding to the sequential converter), a collector (which is basically sequential to the equal converter), and both are timed independently. UART is regularly associated with the transfer of information between devices. At the point when an information byte is kept in touch with the transmitting information register of a UART by the information transport, the UART will change over it to sequential structure begins to transmit the information on the sequential line. UARTs are much of the time used with correspondence measures like RS-485 or RS-232. Information arrangement and transmission speeds (known as baud rate) are configurable in the UARTs and the electric flagging levels and strategies are normally dealt with by the driver circuits that are outside to the UART. UARTs can work in both Full-duplex and Half-duplex methods of correspondence.

In this postulation, structure to Semi-custom format is performed for the UART (both in Full duplex and Half-duplex mode). Verilog HDL (Hardware

Description Language) is utilized here for Register Transfer Level (RTL) coding.

A. Vivado Design Suite HLx Editions - Accelerating High Level Design

The new Vivado® Design Suite HLx editions supply design teams with the tools and methodology needed to leverage C-based design and optimized reuse, IP sub-system reuse, integration automation and accelerated design closure. When coupled with the UltraFast™ High-Level Productivity Design Methodology Guide, this unique combination is proven to accelerate productivity by enabling designers to work at a high level of abstraction while facilitating design reuse.

Starting with Vivado 2019.1, Dynamic Function eXchange is included at no additional cost within all Vivado Editions, including WebPack. Prior versions may require a "Partial Reconfiguration" license, depending on the Edition.

1) Accelerating High Level Design

- Software-defined IP Generation with Vivado High-Level Synthesis
- Block-based IP Integration with Vivado IP Integrator
- Model-based Design Integration with Model Composer and System Generator for DSP

2) Accelerating Verification

- Vivado Logic Simulation
- Integrated Mixed Language Simulator
- Integrated & Standalone Programming and Debug Environments
- Accelerate Verification by >100X with C, C++ or System C with Vivado HLS
- Verification IP

3) Accelerating Implementation

- 4X Faster Implementation
- 20% Better Design Density
- Up to 3-Speedgrade Performance Advantage for the low-end & mid-range and 35% Power Advantage in the high-end,

One of the key differences between a processor and an FPGA is whether the processing architecture is fixed. This difference directly affects how a compiler for each target works. With a processor, the computation architecture is fixed, and the job of the compiler is to determine how to best fit the software application in the available processing structures. Performance is a function of how well the application maps to the capabilities of the processor and the number of processor instructions needed for correct execution. In contrast, an FPGA is similar to a blank slate with a box of building blocks. The job of the Vivado® HLS compiler is to create a processing architecture from the box of building blocks that best fits the software program. The process of guiding the Vivado HLS compiler to create the best processing architecture requires fundamental knowledge about hardware design concepts.

B. What is an FPGA?

An FPGA is a type of integrated circuit (IC) that can be programmed for different algorithms after fabrication. Modern FPGA devices consist of up to two million logic cells that can be configured to implement a variety of software algorithms. Although the traditional FPGA design flow is more similar to a regular IC than a processor, an FPGA provides significant cost advantages in comparison to an IC development effort and offers the same level of performance in most cases. Another advantage of the FPGA when compared to the IC is its ability to be dynamically reconfigured. This process, which is the same as loading a program in a processor, can affect part or all of the resources available in the FPGA fabric. When using the Vivado® HLS compiler, it is important to have a basic understanding of the available resources in the FPGA fabric and how they interact to execute a target application. This chapter presents fundamental information about FPGAs, which is required to guide Vivado HLS to the best computational architecture for any algorithm.

C. Vivado High-Level Synthesis

The Xilinx® Vivado® High-Level Synthesis (HLS) compiler provides a programming environment similar to those available for application development on both standard and specialized processors. Vivado HLS shares key technology with processor compilers for the interpretation, analysis, and optimization of C/C++ programs. The main difference is in the execution target of the application. By targeting an FPGA as the execution fabric, Vivado HLS enables a software engineer to optimize code for throughout, power, and latency without the need to address the performance bottleneck of a single memory space and limited computational resources. This allows the implementation of computationally intensive software algorithms into actual products, not just functionality demonstrators.

D. UART ON FPGA

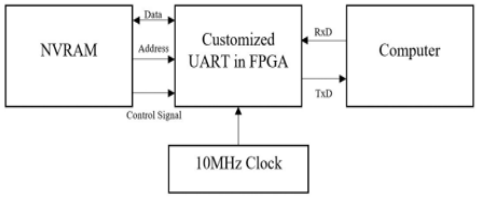


Fig 11: UART implementation on FPGA

It primarily comprising of FPGA and Level converter and RS-422 handset, NVRAM and the PC, the plan of the UART is embedded in Xilinx SPARTAN-3 FPGA.VHDL is utilized for the plan in Xilinx instrument and equipment programming should be possible by utilizing JTAG Software engineer. Reference 10 MHz clock is utilized for the UART plan. By utilizing the choice lines baud pace of UART is chosen. TX and RxD are the yields of the transmitter and beneficiary segments of the UART. These single-finished signs are can be changed over into differential by utilizing MAX1490 RS-422 handset. The single-finished signs from FPGA are of 3.3V can be changed over into 5V by utilizing a level converter. FPGA what's more, NVRAM works at 3.3V and RS-422 handset works at 5V. Aside from UART structure some inner logic feasibility of the plan. UART configuration is then dumped on both the FPGA sheets, at that point the information is traded sequentially between both the sheets utilizing the RS-232 link, though the equal information that will be transmitted is created by the LFSRs when the prior byte is effectively transmitted. can be actualized to create the signs for composing information into the NVRAM for each 5 ms. A portion of the signs, for example, chip Enable (CE) Write Enable (WE), Output Enable (OE), Address, Data can be produced utilizing some interior rationale executed in FPGA.

These signs are utilized for composing the NVRAM UART information out organization is as appeared in fig 4. The grouping is Idle condition, one beginning piece, eight information bits(LSB first) trailed by a couple of stop bits., the UART is created in the FPGA, the single-finished signs Tx also, Rx are changed over into 5v and took care of to the RS-422 handset also, the yields are differential signals and imparted to the outside world.RS-422 to USB is utilized for imparting with the PC. With the utilization of Hyper terminal, the correspondence can be checked on the PC. By choosing parameters like Baud rate, information bits, equality, number of stop bits, stream control correspondence can be built up. The Received information can be caught in a document and can be seen by utilizing HEX perky programming device.

Fig 12: UART data out format

IV. FLOWCHART & ALOGORITHM

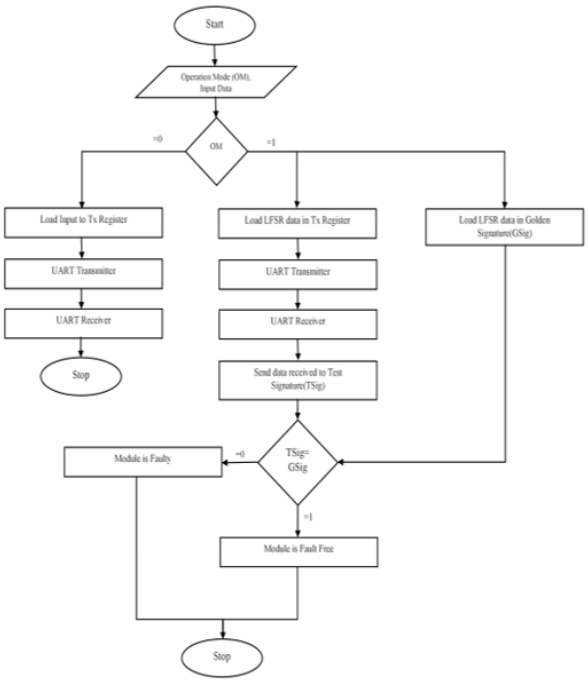


Fig 13: Flow chart of the UART module

1. The working begins with entire system being reset to its initial state with a negative edge triggered reset value.
2. Then, the system clock is set (manually by the user or by the incoming data itself) and the select line of the multiplexer is set depending on which mode the system should work inside. regular or testing mode.
3. If the mode is set to regular by making the select line value 0, the system takes in incoming data from the user or another system directly and sends it to the UART system for transmission.
4. If the mode is set to testing by making the select line value 1, data sent to the UART system is acquired from the test pattern generator, whose timing is controlled by feeding the receiver ready signal as the clock.
5. A few samples of test pattern generator outputs are stored in the golden signature component.
6. During normal operation, the data is directly transmitted through the UART and the received data is stored in a register, ready for

parallel transmission to other systems as and when required.

7. If test mode is used, the UART transmits the samples which are generated by the pattern generator and stored in the golden signature, and the output of the UART is stored in the test signature.
8. The multiple values stored in the golden signature and test signature respectively are then compared, and if all of them are found to be equal, then it is indicated through a value that the system is working properly, otherwise the system has failed.
9. Based on the comparison, a message is also displayed whether the system has passed or failed in its working.

A. TEST PATTERN GENERATOR ALGORITHM:

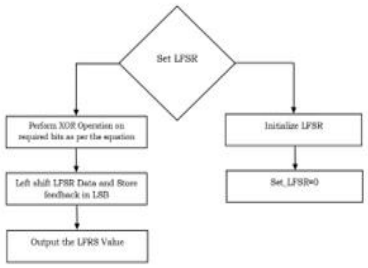


Fig 14: Flow chart of the Test pattern Generator

The test pattern generator is made by using an 8-bit Linear Feedback Shift Register, and its algorithm is as follows:

1. Clock input is given to the LFSR to control the time period of each cycle, and a set value is also given to initialize the LFSR as and when required.
2. Initially set value is made 1, to initialize the LFSR and set its output to 11111111.
3. After that, set value is made 0, and the working of the LFSR begins.
4. A feedback value is calculated for each cycle of the LFSR, with the help of a system equation which performs XOR operations on particular bits of the value stored in the register.
5. By modifying the above-mentioned equation, different pseudo-random value combinations can be generated.
6. At the end of each cycle, a left shift is performed on the register value, and the feedback is stored in the Least Significant Bit (LSB).
7. A new feedback is generated in each cycle, thus leading to the generation of the required pseudo-random sequences, controlled by the LFSR equation.
8. A new sequence is generated within each cycle.

B. UART:

The UART is made of 3 components, the transmitter, the receiver and the baud rate generator.

4) UART TRANSMITTER:

Fig 15: Flow chart of the UART Transmitter

1. The transmitter is set to its initial state during the initial reset phase of the entire system.
2. The system clock is fed to the transmitter to control the rate at which data is stored in the transmitter shift register (TSR).
3. To control the rate at which data is sent out from the TSR, a signal from the baud rate generator is used, which is used to synchronize the transmitter and receiver independent of the system clock.
4. If the system is in reset state, the TSR value is made 11111111.
5. After reset phase, the TSR is loaded with the incoming data by initializing a load signal value to 1.
6. The TSR value is then transmitted by creating another temporary register, which sends the data serially(bit-by-bit) and this operation is controlled by a start bit and a shift bit.
7. When start=1, the temporary register is initialized as 1 as the MSB, followed by the entire TSR value.
8. Then, start becomes 0 and shift becomes 1, due to which serial data transmission starts.
9. When shift=1, the temporary register undergoes logical right shift, and the LSB of each cycle which is removed during the shifting process is sent as serial output to the UART receiver.
10. The start and shift signal values are controlled with the help of a Finite State Machine (FSM).
11. The FSM makes use of the rising edge of the baud rate signal as its clock, and it uses a 4-bit counter to control its working.
12. Three states of the system are defined: IDLE as 0, SYNC as 1, and T_DATA as 2.

13. Initially, when reset signal is given, the FSM is set to IDLE state.
14. When reset signal is removed, the FSM state transforms from the present state to the next state. The next state is determined as follows:
15. If both, start transmission bit and loadstar bit are 0, the next state is set to IDLE, the 4-bit counter is reset, and the start and shift bits are set to 0.
16. If the above condition becomes false, the next state is set to SYNC, which initializes the transmitter for the synchronization process. At this time, the counter is maintained in the reset phase, however the loadstar bit is set to 1.
17. During the SYNC state, the next state is set to T_DATA, which stands for transmit data. In this state, counter is still maintained in the rest state, but the start bit is set to 1.
18. In the T_DATA state, initially the next state is set to T_DATA as well, to allow all the bits to be transmitted. The counter is made to increment in each cycle in this state, start bit is set to 0 and the shift bit is set to 1.
19. With each count, a bit of the values stored in the shift register is transmitted as mentioned previously.
20. When count value exceeds 8, it means that all bits have been transmitted, and so the next state is set to IDLE, the count value is reset to 0, and a TXD_DONE bit is set to 1,

C. UART RECEIVER:

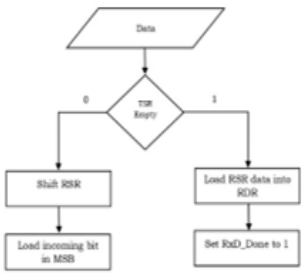


Fig 16: Flow chart of the UART Receiver

1. The receiver is set to its initial state during the initial reset phase of the entire system.
2. The system clock is fed to the receiver to control the rate at which data is stored in the output shift register (RDR).
3. To control the rate at which serial data is received by the receiver and stored in a register (RSR), a signal from the baud rate generator is used, which is used to synchronize the transmitter and receiver independent of the system clock.
4. While storing the received data in the RSR, to ensure that integrity of the data is maintained, a signal (bclkx8) from the baud rate generator is used, whose frequency is 8 times more than the baud rate signal sent to the transmitter.
5. This signal is used to divide the received bit into 8 parts. This is done so that we can attain the mid value of the bit pulse by using the fourth part.
6. The mid value is desired because it depicts the bit value most accurately.
7. When the shift bit is set to 1, the receiver shift register (RSR) undergoes a right shift, and the value obtained from the above splitting process is stored in the MSB.
8. The shift signal value is also controlled with the help of a Finite State Machine (FSM).
9. The FSM makes use of the rising edge of the baud rate signal (bclkx8) as its clock, and it uses a 3-bit and a 4-bit counter to control its working.
10. Three states of the system are defined: IDLE as 0, START_DETECTED as 1, and R_DATA as 2.
11. Initially, when reset signal is given, the present state of the FSM is set to IDLE.
12. When reset signal is removed, the FSM state goes from the present state to the next state. The next state is determined as follows:
13. Initially, if received data (RXD) is 0, it indicates a start bit, so the next state is set to START_DETECTED, both the counters are reset, and the shift bit is set to 0.
14. If the above condition is found to be false, the next state is set to IDLE, to maintain the receiver in its usual dormant state.
15. During the START_DETECTED state, the 3-bit counter is made to increment. This is used to model the bit splitting process into the working. The 4-bit counter is maintained in the reset state.
16. If RXD becomes 1 before the mid value of that bit pulse is reached, it indicates a faulty start bit, and the next state is again set to IDLE.
17. However, if RXD maintains it state till its mid position, it indicates that start bit is correct, hence the next state is set to RECEIVE_DATA, to start the reception process, and then the 3-bit counter is cleared.
18. In the RECEIVER_DATA state, the 3-bit counter is made to increment again. This is done for obtaining the mid-point of the next bit, which is 8 parts away this time.
19. When the 3-bit counter obtains its maximum value, shift signal is set to 1 to start shifting the data into the RSR, the 3-bit counter is reset to 0 and the 4-bit counter starts incrementing.

20. As long as the 4-bit counter value is less than 8, it keeps incrementing and shift signal is maintained at high state.
21. When 4-bit count value becomes 8, it means that all bits have been received, so both counters are cleared.
22. At this point, if received value RXD is 0, it means that Stop bit has not been detected, so the next state is set to IDLE.
23. When RXD=1, it indicates that Stop bit has been detected, so the ladder value becomes 1 to load the RSR value into the output register, and an RXD_READY signal is also set to 1, to indicate that data has been received successfully and it is available for further use.

D. BAUD RATE GENERATOR:

1. The baud rate generator is given system clock for its timing purposes, and it is also given a select value to choose a particular baud rate from a set of different rates.
2. Few registers are defined as follows: mod3, count4 and mod8.
3. These are used for the generation of appropriate baud rate signals for the UART transmitter and receiver.
4. Mod3 acts as a modulo-3 counter, which is incremented in each clock cycle and then reset to 0 when its value becomes 3.
5. The value of mod3 is assigned to a clock divider variable CLK_DIV, which is then used as a clock for the incrementing process of count4 register.
6. By selecting a particular bit of the count4 register, we can generate baud rate signals of the appropriate frequency. For our purpose, we use 9600 baud-rate, and hence we select count4[1].
7. The value acquired from the above process is stored in a variable BCLKX8, which is used in the receiver.
8. BCLKX8 is also used as a clock in the incrementing process of mod8, and we assign mod8[2] to an output variable BCLK, which is used as the baud-rate signal in the transmitter.

E. GOLDEN SIGNATURE:

The golden signature is an SRAM which stores a set of pseudo-random sequences generated by the LFSR.

1. Negative edge trigger reset of the system also clears the golden signature. The RXD_READY signal generated by UART receiver acts as clock for this component.
   A base address as well as the depth of the signature (how many 8-bit data values it can store) is also defined
2. Initially, the write enable signal is set to 1 and read enable is set to 0.
3. The write enable signal is maintained in the high state for 4 clock cycles.
4. During each clock cycle, value obtained from LFSR is stored in a particular address, and then the address value is incremented by 1. Thus, 4 8-bit values get stored in the signature.
5. Then, write enable is set to 0 and read enable is set to 1. This is maintained for additional 4 clock cycles. The address value is also set to its base value.
6. In this period, during each cycle, data is read from a particular address of the signature and sent to the comparator, and then the address is incremented by 1.

F. TEST SIGNATURE:

The test signature is an SRAM which stores a set of values received from the UART receiver.

1. Negative edge trigger reset of the system is used to clear the golden signature. The RXD_READY signal generated by UART receiver acts as clock for this component.
   A base address as well as the depth of the signature (how many 8-bit data values it can store) is also defined.
2. Initially, the write enable signal is set to 1 and read enable is set to 0.
3. The write enable signal is maintained in the high state for 4 clock cycles.
4. During each clock cycle, value obtained from LFSR is stored in a particular address, and then the address value is incremented by 1. Thus, 4 8-bit values get stored in the signature.
5. Then, write enable is set to 0 and read enable is set to 1. This is maintained for additional 4 clock cycles. The address value is also set to its base value.
6. In this period, during each cycle, data is read from a particular address of the signature and sent to the comparator, and then the address is incremented by 1.

G. COMPARATOR:

1. The timing of comparator operation is controlled by given RXD_READY as clock to the comparator.
2. Individual values obtained from the golden signature and test signature are compared.
3. If both are found to be equal, a pass signal is set to 1, indicating correct working of the system, and if unequal, it is set to 0.
4. Thus, if all the respective values of the 2 signatures are not equal, pass signal becomes 0, indicating fault in system operation, and if they are all equal, then pass

signal maintains its value at 1, indicating correct working of the system.

V. RESULTS AND SIMULATIONS

The figure shows the simulation results for one of the trials runs we performed.

The signals like clk, dbus, din, Sc_in, gsig, tsig, rdr. etc are observed in the simulator.

The system was operated at a frequency of 10MHz generating clock pulses of 100ns time period

In it, DBUS at the top is the data being sent by the user or another device during normal operation, whereas Sc_in is the data incoming from the test pattern generator (LFSR). SEL has been set as 1 to select 9600 baud rate and mux_sel has been set high so that the system operates in test mode.

Sc_in is being sent through the Mux and hence din, which is UART input, is equal to Sc_in. The data goes through the various UART stages and finally gets stored in the Receiver Data Register (RDR). Due to serial communication, a delay of 1 clock cycle is present in the above simulation, and hence din value is obtained at RDR in the next clock cycle.

Whenever transmitter finishes transmitting, Txd_doneh signal becomes 1, and at the center of this high pulse, the receiver sets the Rxd_ready signal to High state (1) as well, indicating that the receiver is ready to receive the data.

Gout and tout are the golden signature and test signature values respectively. For every 8 clock cycles, they are able to store and then send out 4 different data elements, and as seen in the figure, they are all equal. Hence the eq signal which checks for equality goes and remains in the high state, due to which the pass signal also becomes 1, indicating correct working of the system.

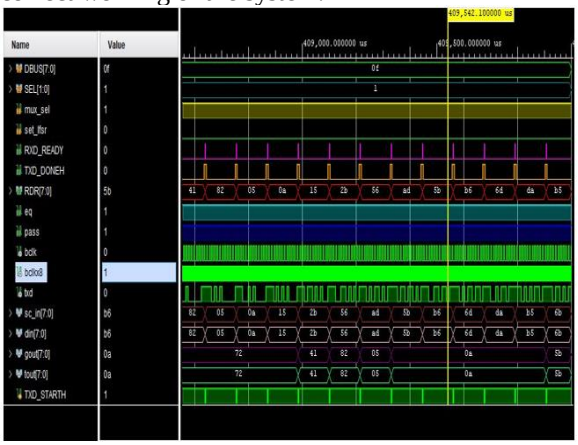
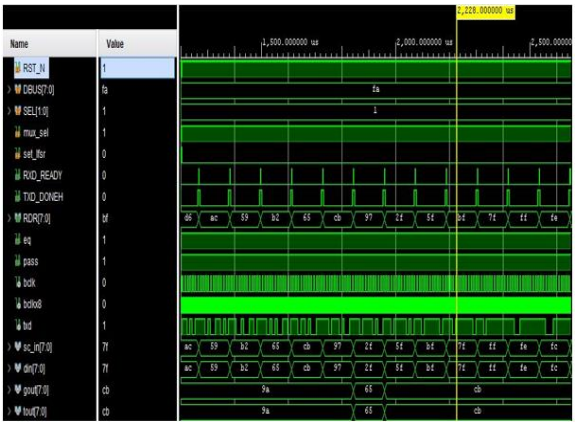


Fig 17: Simulation Results of the UART module



18: Simulation Result for different test pattern

The above figure shows another simulation result which we obtained by changing the test pattern generation equation, and the test and golden signature capacities.

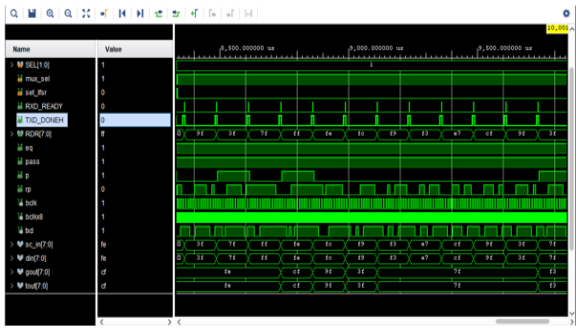


Fig 19: Test case Simulation with parity checker.

The first two simulation results were without an error checking mechanising, but the exact above simulation with a Parity checker as a error proof mechanism.

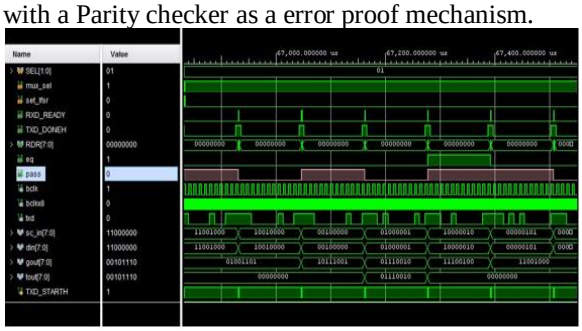


Fig 20: Failed Test case Simulation

The above test results show the test case of failure simulation, since the failure can be witness only on hardware in a practical used case, we have hence simulated a test case for Failure condition.

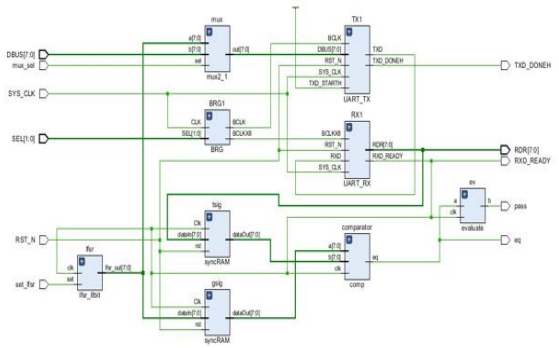


Fig 21: RTL schematic of the UART module without parity checker

The above figure shows the schematic obtained after performing RTL synthesis on the system. It depicts each and every component of the system and how they are interlinked with each other.

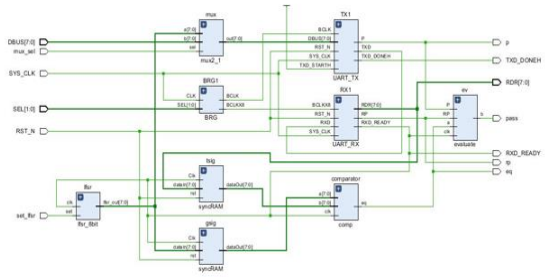


Fig 22: RTL schematic of the UART module with parity checker

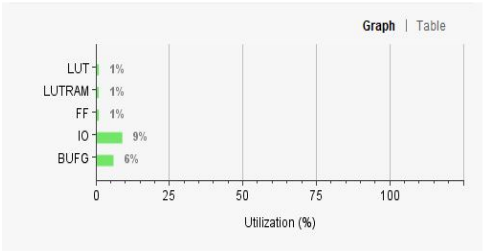


Fig 23: Graph of Design Utilization

The above figure shows the graph part of the design Utilization report with the components on the y-axis and the utilization percent on the x-axis.

Name	Slice LUTs (41000)	Slice Registers (82000)	Slice (10250)	LUT as Logic (41000)	LUT as Memory (13400)	Bonded IOB (300)	BUFGCTRL (32)
UART	82	109	32	66	16	28	2
BRG1 (BRG)	9	10	4	9	0	0	0
comparator (comp)	4	2	3	4	0	0	0
ev (evaluate)	1	2	1	1	0	0	0
gsig (syncRAM)	15	15	4	7	8	0	0
lfsr (lfsr_8bit)	1	8	1	1	0	0	0
mux (mux2_1)	4	0	2	4	0	0	0
RX1 (UART_RX)	13	29	11	13	0	0	0
tsig (syncRAM_0)	15	15	5	7	8	0	0
TX1 (UART_TX)	20	28	10	20	0	0	0

Table 2: Design Utilization table.

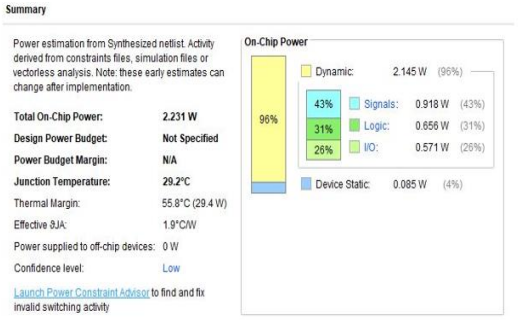


Fig 24: Power Utilization Report

VI. APPLICATIONS, ADVANTAGES AND LIMITATIONS

There have been upsets in semiconductor innovation. Presently multi-million transistors are being coordinated on a solitary chip named as SoC to understand the full usefulness of a framework. SoC offers tremendous points of interest, for example,

Improved execution, Decrease of cost, Expanded dependability Diminished force utilization Decreased framework size ,Brief timeframe to advertise

Typically, predesigned licensed innovation modules are sewed together in structuring and creating SoC to lessen time-to-advertise. UART is a regularly utilized segment in SoC for information correspondence. A miniaturized scale UART of minimal size is currently a consuming interest in IC structure industry. UART from various organizations, for example, NS16450A and NS16550A of National Semiconductor Organization (NSC) [SC26C198 and SC28L92 from Philips Company are accessible in the advertise. They are utilized as discrete segment in structuring electronic framework. Structure of UART chip has been talked about in some writing where it has been utilized as a stage to complete analyses to take care of various research issues. Yandel depict on-line testability issue while O'Neill et a feature issues identified with rapid sequential correspondence what's more, Yunshan and Marshall execute IFIS (If it Fails It Stops) testability procedure utilizing UART chip. The plan of the UART chip introduced in this paper claims uniqueness for its particularity highlight and utilization of the business standard Verilog HDL code. Verilog abbreviates the configuration pattern of a chip by proficiently depicting its conduct It is innovation autonomous. On the off chance that a specific IC manufacture process gets obsolete, it is conceivable to integrate another degree of plan by just changing the incorporating innovation record however utilizing the equivalent Verilog HDL code.

A. Applications of UART

- Transmitting and receiving UARTs must be set to the same bit, parity and stop bits in order for them to function properly.
- Serial ports used in personal computers and embedded systems use UARTs. These devices utilize the CPU to test the condition of information transmissions. Instead, the UART chip can be implemented to overcome the cost and storage issues in the device.
- Can be used in microcontrollers due to the requirement of only two wires for communication leading to a compact size.
- Due to advancement in technology the concern for serial data transmission speed has been considerably reduced making UART viable option for compact and cheaper designs.

B. Advantages of UART

- UARTs are straightforward and they just depend on two wires.
- They have an equality bit so as to check for mistakes in information parcels.
- UART is a generally utilized strategy. in the devices.

C. Disadvantages of UART

- The data frame must not contain more than 9 bits.
- The baud rates of the receiving and transmitting ends of UART are restricted to ten percent.
- Serial data transmission speed is less compared to parallel data transmission

Numerous applications depend on the equal execution of indistinguishable tasks; the capacity to design the FPGA's CLBs into hundreds or thousands of indistinguishable handling squares has applications in picture preparing, man-made consciousness (AI), server farm equipment quickening agents, undertaking organizing and car propelled driver help frameworks (ADAS).

A significant number of these application regions are changing rapidly as prerequisites develop and new conventions and norms are embraced. FPGAs empower producers to execute frameworks that can be refreshed when vital. A genuine case of FPGA use is fast pursuit: Microsoft is utilizing FPGAs in its server farms to run Bing search calculations.

The FPGA can change to help new calculations as they are made. In the event that necessities change, the plan can be repurposed to run recreation or displaying schedules in an HPC application. This adaptability is troublesome or difficult to accomplish with an ASIC.

Other FPGA utilizes incorporate aviation and safeguard, clinical gadgets, advanced TV, shopper hardware, modern engine control, logical instruments, cybersecurity frameworks and remote interchanges.

D. Features of the proposed UART

The proposed UART is having the following features:

• Independently controlled transmitter and receiver module.
• Programmable baud-rate generator.
• Compact size.
• False start-bit detection.
• Programmable word length, 1 start-bit, no parity

VII. CONCLUSIONS AND FUTURE WORK

The simulated waveforms validate the precision of the Verilog HDL usage to outline the characteristic and the working of the proposed UART system. The test is performed at 9600bps baud rate. The UART module makes use of high fault coverage which is the most important thing in a design to ensure the reliability of the design, which can be further increased with appropriate changes in the memory allocation and usage.

The outputs on the computer are also verified for different baud rates. When implemented on FPGA, it will have enough free macro cells available and could be reconfigured for any new application or additions for future design.

The current design of the UART is capable to handle the received data with parity checker as error detection mechanism. An error correction mechanism can be implemented in future to correct the data with error.

Talking about further improvements in the system design and application, we can use different frequency level such as Wi-Fi frequencies range of MHz or GHz to make more efficient UART. There is wide scope to implement a frequency generator using different IO standard such as LVDCI (Low Voltage Digitally Controlled Impedance), HSTL (High Speed Transistor Logic)

Apart from the points mentioned above, we can include the capabilities of Storing the data Dynamically and Internal clocking mechanism.

VIII. ACKNOWELDGEMENT

In this section, a brief acknowledgment has to be presented as a thanks giving to all the people who have directly or indirectly helped in the successful completion of the project.

IX. REFERENCES

[1] Liakot Ali, Roslina Sidek , Ishak Aris , Alauddin Mohd. Ali, Bambang Sunaryo Suparjo*," Design of a micro-UART for SoC application*", Science Direct, Computers and Electrical Engineering 30 (2004) 257–268, December 2004.

[2] T.D. Manoj Kumar Reddy, *"Implementation and Customization of UART in Xilinx FPGA*". International Journal of Combined Research & Development (IJCRD), eISSN: 2321-225X; pISSN: 2321-2241 Volume: 2; Issue: 1; January – 2014.

[3] M.RAMAKRISHNA –Assistant Professor, Dept of ECE, SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES." NOVEL IMPLEMENTATION OF UART WITH BIST TECHNIQUE IN FPGA", International Journal of Global Innovations -Vol.2, Issue. I, Paper Id: SP-V2-I1-005, ISSN Online: 2319-9245, May 2014.

[4] D.Monica satyavathi1, G.Anjaneyulu2 " *Implementation of BIST Technique in uart Serial communication*", Electronics and Communication Engineering Dept. M.V.G.R.College of Engineering Vizianagaram, India," IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 4, Issue 5, Ver. II (Sep-Oct. 2014), PP 21-29 e-ISSN: 2319 – 4200, Oct, 2014.

[5] AKSHATA SURENDRA SAVANT1, SANDRA BENJAMIN2 AND M Z KURIAN3 1M.Tech in VLSI and Embedded Systems, Department of ECE, SSIT, Tumakuru, Karnataka, India., "*Design and Simulation of UART and Built-in-self-test Architecture in Verilog-HDL",* International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353 Volume 14 Issue 2 –APRIL 2015.

[6] M. Mamatha, Vijaykumar R. Urkude M. Tech in VLSI at Vignan Institute of Engineering and Technology IIAssociate Prof., Dept. of Electronics and Comm. Engg. in Vignan Institute of Engg. and Technology, "*BIST Enabled UART for Real Time Interface Applications using FPGA*" International Journal of Research in Electronics and Communication Technology (IJRECT 2015), IJRECT All Rights Reserved 10, ISSN: 2348 – 9065, ISSN: 2349 - 3143, Vol. 2, Issue 4 Oct. - Dec. 2015.

[7] Neeraj Pawar, "*UART implementation with multiple input signature register for full fault coverage"*, SCHOOL OF VLSI DESIGN AND EMBEDDED SYSTEMS NATIONAL INSTITUTE OF TECHNOLOGY KURUKSHETRA-136119, SESSION 2014-2016.

[8] V. Thirunavukkarasu, R.Saravanan and V.Saminadan" *Performance of Low Power BISTArchitecture for UART*" International Conference on Communication and Signal Processing, April 6-8, 2016, India.

[9] LEE ZHI YONG, "*UART DESIGN, INTEGRATION AND SYNTHESIS ON FPGA"* REPORT SUBMITTED TO University Tunku Abdul Rahman, May 2016,

[10] M.S.N. Shettennavar, Mr.B.N. Sachidanand, Mr.D.K.Gupta, Mr.V.M.Metigoudar Assistant Professor, Dept. of Electronics Engineering, DKTE's Textile and Engineering Institute ,Ichalkaranji Maharashtra India." *Implementation of UART with BIST Technique* "International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 - 0056 Volume: 03 Issue: 09, Sep-2016.

[11] E. Raghuveera1, K Hari Kishore2, Shaik Shoukat Vali3, and G. Siri Vennela Department of ECE, K L University, Vaddeswaram, Guntur, A.P, India, "*VERILOG IMPLEMENTATION OF UART WITH BIST TECHNIQUE FOR TPG*", International Journal of Pure and Applied Mathematics, Volume 115 No. 7, 531-536 ,ISSN: 1311-8080 ,(ISSN: 1314-3395 url: http://www.ijpam.eu , Special Issue, 2017

[12] P. Ramesh, Dr. D.N Rao, Dr. K. Srinivasa Rao, Assoc Professor, ECE Department Joginpally BR Egg College, ECE Department DRK Institute of science and "*Power Reduction Testing Techniques of BIST, LFSR & ATPG for Low Power Circuits*" International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE) Vol 4, Issue 11, November 2017.

[13] Yogesh Kumar, Neeraj Pawar Assistant Professor, MITRC, Alwar, Rajasthan, India," *Implementation of built in self-test (BIST) enabled UART using FPGA for fault detection*". National Journal of Multidisciplinary Research and Development, ISSN: 2455-9040, Impact Factor: RJIF 5.22, Volume 3; Issue 1; January 2018

[14] S SANTHI PRIYA, G RAVIKISHORE Assistant Professor, Department of ECE, Vidya Jyothi Institute of Technology, Hyderabad, Telangana," *Design and analysis of UART based on BIST*", International Journal of Research, e-ISSN: 2348-6848, p-ISSN: 2348-795X, Volume 05 Issue 23, December 2018

[15] V K Ivanov, E V Nosov, "*Serial communication protocol for FPGA-based systems*", International Youth Conference on Electronics, Telecommunications and Information Technologies 11–12 July 2019, Congress Centre of Peter the Great, St. Petersburg Polytechnic University, Russian Federation, *Conf. Ser.* 1326 012044, 1st Oct 2109.

[16] Biswajit Pandey, Abhishek Pandey, DM Akbar Hussain, "*Low voltage Complementary Metal Oxide Semiconductor Energy efficient UART design on Spartan6 FPGA*", 11th International Conference on Computational Intelligence and Communication Networks, 2019.

Manuals-

[1] Tom Feist," *Vivado Design Suite*" Xilinx, WP416 (v1.1) June 22, 2012.

[2] Vivado Design Suite User Guide "*High-Level Synthesis*", Xilinx, UG902 (v2017.1) April 5, 2017

Websites-

[1] https://www.xilinx.com/products/design-tools/vivado.html

[2] https://microcontrollerslab.com/uart-communication-working-applications/

X. NOMENCLATURE AND ACRONYMS

Abbreviations:

UART	Universal Asynchronous Receiver Transmitter
BIST	Built in Self-Test
Mux	Multiplexer
DBUS	Input Data Bus
Sel	Select line
Rst_N	Negative Edge Reset
LFSR	Linear Feedback Shift Register
Gout	Golden signature output
Tout	Test signature output
P	Transmitter parity bit
Rp	Receiver parity bit
Rsr	Receiver shift register
Tsr	Transmitter shift register
Rdr	Receiver data register
Din	Input data to UART
Sc_in	Input from LFSR to mux
Txd_doneh	Signal which goes high to indicate successful data transmission
Rxd_ready	Data ready at receiver to be used
Eq	Equality
Pass	Signal indicating correctness of system
Bps	Bits per Second
Ms	Milliseconds