ISSN 2347 - 3983
Volume 8. No. 7, July 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter95872020.pdf
https://doi.org/10.30534/ijeter/2020/95872020

An Efficient Debugging Architecture for DTG Based FIR Filter Using I^{2}C Protocol in DSP Processor

A Murali, K Hari Kishore
Research Scholar, Department of ECE, Koneru Lakshmaiah Education Foundation,
Vaddeswaram, Guntur, A.P, India
Professor, Department of ECE, Koneru Lakshmaiah Education Foundation,
Vaddeswaram, Guntur, A.P, India
Email: amurali3@gmail.com

ABSTRACT

Nowadays, the debugging process in the Field Programmable Gate Array (FPGA) platform is more essential to manufacture the hardware device without error. Normally, the Digital Signal Processor (DSP) unit contains a different kind of processing blocks such as Finite Impulse Response (FIR) filter, Fast Fourier Transform (FFT), Arithmetic Unit, etc. Every processing block is operated with analog and digital types of the signal which are transferred in various types of bus protocols such as Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I^{2}C). In this research work, the reconfigurable insertion technique is used in the FIR filter to debug the entire architecture with less area. Additionally, the protocol debugging also proceed in this work to improve the system performances. In case of any bug in the UART protocol, the data is transferred through 2CI bus protocols which reduce the time period of the entire architecture. In Spartan 6 FPGA device, I^{2}C bus protocol occupied 754 LUTs, 539 flip flops, 427 slices which are too less than other protocols. According to the power consumption, the UART protocol consumes 0.0135W which is better than other protocols.

Key words: Field Programmable Gate Array, Finite Impulse Response filter, Inter-Integrated circuit, Universal Asynchronous Receiver Transmitter, FSM controller, Reconfigurable buffers, Debug Instrument.

1. INTRODUCTION

In recent days, Field Programmable Gate Arrays (FPGAs) have become one of the core components for high-reliability computing systems. In modern days, so many errors have occurred while designing the architecture in the FPGA platform. In the DSP processor, the FIR filter is one of the major blocks which is used for digital communication. While communicating the digital information, the bug has occurred and it affects the user information. So, the debugging process needs to be performed on the FPGA platform. Design difficulty doubles every 18 months compared to design efficiency, which doubles every 39 months [1-2]. It is due to the enormous effort that has been made to validate complex designs. Virtual prototyping is typically used for device testing, but when the design is complex, virtual prototyping suffers from speed problems. Due to hardware invisibility, only small signals accessible at the FPGA pins[3-4] can be controlled. In order to address this problem, FPGA's vendors have introduced Integrated Logic Analyzer (ILA) cores embedded in the design that can be activated on the basis of certain preset conditions and provide a small debug window. Nonetheless, in addition to the restricted window, debugging is still difficult because user input is required to figure out the problems. For debugging the Register Transfer Level (RTL) module, bus protocols are helped to transfer the data. If the error has occurred in the bus protocols, the wrong data only reached to the RTL module [5-6]. The difficulty of debugging, therefore, increases with the design complexity which increases the design cycle time. It is a practical necessity to test and debug any embedded design before physical deployment. Despite the recent methodologies in the FIR filters design algorithm, there are still some issues faced by design engineers. Some of the issues are smaller filter order, low power consumption, minimization of Stop Band Ripple (SBR), Pass Band Ripple (PBR) [7]. The framework consists of comprehensible information about the design internal process that was normally used for the debugging task. The software was mostly used for the HLS blocks debugging process and validation. Although, once the design is mapped to an FPGA, this is difficult to find the bugs [8, 9]. Many existing methods have been applied for the FPGA debugging process that has the limitations of more circuit area and more debugging time. Moreover, the bugs are identified only in the RTL modules, not in bus protocols [10]. In this research work, debugging architecture is designed for verifying the DTG-FIR filter and identifies the error in bus protocol it seems.

The Major contribution of this research paper is given below,

- The debugging process is applied in bus protocol which helps to identify and rectify the error in UART and I^{2}C protocols.

- In case of any bug in any one of the bus protocols, alternative bus protocols help to transmit the data.
- The Reconfigurable buffer insertion technique is used to debug the multiple internal blocks of FPGA. The proposed reconfigurable buffer insertion technique occupied less hardware utilization compared to conventional debugging.
- FSM controlled based debugging architecture is used to maintain the common buffer for all the taps of the filter which helps to improve the system speed.

The paper is followed as a Literature survey is provided in section 2, the existing model problems and solution is given in section 3, the proposed explanation of the bus protocols are explained in section 4, Results and discussion is given in section 5, and conclusion and future scope are provided in section 6.

2. LITERATURE SURVEY

Hardware debugging circuits requires signal tracking to record and analyze circuit behavior. Recent methodologies in the Hardware debugging in FPGA circuits were surveyed in this section.

Habib ulhasan khan and Dianagohringer [11] proposed FPGA debugging process using rule based inference system with MATLAB software. This proposed architecture has undergone the Design Under Test (DUT) for transferring the memory data to the respective unit blocks. With the help of MATLAB, the stored information has performed the debug process using the rule based method. The error correction process was performed by the debug method but the decision-making process was handled by the user. Due to the usage of the inference system model, the data loss was too less. But, the debugging process took more time to identify the error and rectified it.

Eddie Hung *et al.* [12] presented coupling based instrument based FPGA debugging with user circuitry. Generally, the designers create the prototypes model for making the FPGA model with a possible combination of logical blocks that undergone the testing process. In this work, the trace buffer model was used to improve the observability of the FPGA devices and the debugging circuitry was enhanced with efficient operation speed. For the debugging process, the individual buffer was inserted for all the internal modules which increase the entire system area.

Goeders and Wilton [14] designed a debugging architecture that automatically records the circuit behavior and relates them to the source code. The developed architecture allows the designer to debug the HLS circuit's in-context with the source code. Several signals tracking method has been applied to the HLS that allow much longer execution trace to be captured. The signal tracking methods like signal compression, Cycle-by-Cycle tracking of dynamic changes in the signal, and offline signal restoration were used. The developed architecture is compared with an embedded logic analyzer to perform signal tracking shows that the length of execution trace is faster for the developed architecture. The circuit area and debugging time is required to be reduced.

Khan, *et al*. [15] developed an intrusive FPGA-in-the-Loop (FIL) debugging method using a rule-based inference system. The FPGA in-loop is used to debug with a cycle-accurate lossless debugging system with an unlimited trace window in an embedded design. The HDL debugging data has been used in this method for a runtime correspondence analysis. The golden reference generated through a high-level model and post-synthesis implementation simulation results were used to speed up the debugging process. The pattern extraction is used to analyze the generated golden reference and the received debugging data. The error is difficult to identify in the system and debugging time is needs to be reduced.

3. PROBLEM STATEMENT

Normally, FPGA has a different model that is helped to design the various designs. Different kinds of prototypes and DSP processors were also possible to design in the FPGA platform. Due to the easy handling and flexible characterization, the FPGA platform is frequently used in the society than the ASIC design. For designing the hardware setup module, FPGA is one of the main platforms to identify and rectify the errors. While performing the FPGA debugging the following problem has been created. i.e.

- In recent years, identifying the errors in FPGA systems is too difficult. Most of the debugging architecture takes 50% of the embedded design cycle.
- The conventional architecture occupied more area, more testing time, more power due to the more number of logical elements present in the modules.
- Most of the conventional architecture is performed in the debugging process in the main module of the architecture. If there is any bug in the bus protocols, the entire system operation has collapsed.
- Transmission bus protocols also lost more data during the communication from one module to another module.

Solution:

To conquer the above mentioned problems, the reconfigurable buffer insertion technique is used in this proposed method. Mainly, this work concentrates bus protocols such as UART and I^2C. In case of any error occurred in any one of the bus protocols, alternative protocols help for data transferring. There is no need to wait for transferring the data until the bug protocol needs to be debugged. This process helps to operate the FIR filter with efficient and perfect information. With the help of the FSM controller, the common buffer has designed to verify all the internal modules of the FIR filter.

4. DTG-FIR ARCHITECTURE

Initially, the DTG-FIR filter architecture is designed in the Verilog language. The block diagram of DTG-FIR is shown in Figure 1, which contains register, an address generator, FIR filter, and Shift register. Based on the block diagram, the entire architecture is simulated and the FIR results are verified.

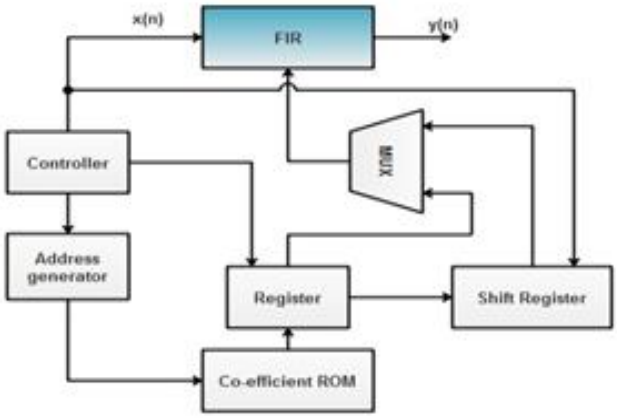

Figure 1: Block diagram of the DTG- FIR filter architecture

This FIR filter is undergone the debugging process, the basic flow of the debugging block diagram is shown in Figure 2, which contains the following modules such as input test vector, circuit under test, output response, expected correct response, and comparator.

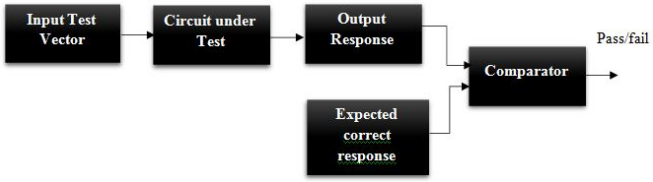

Figure 2: Fundamental flow of FPGA debugging

The processing inputs are generated from any sources or randomly which are stored in the register. The respective architecture is designed to perform the operation which contains the N number of modules. Each module needs to perform the design under test to avoid the error present in the respective modules. Likewise, all the modules have performed the circuit under test operation which produces the output. During the test case, if it is any bug has happened, the delivered output is entirely wrong. Initially, the bug needs to be identified and rectified in FIR filter architecture. Moreover, in this work, the bugs are checked in the bus protocols (UART, I^2C) which also generate wrong results in the output response terminal. The designer should know the input values, architecture operation, and the respective output values. So, the designer can say the expected response of the FIR filter architecture. With the help of the comparator, the output response results and expected response results are comparing. If it is a mismatch, the comparator generated the fail results in the comparator terminal. So, the bus protocols bugs are debugged using the error correction unit. In this case, the comparator produces the pass results in the output terminal. The process of bus protocols in the FIR filter explanation is given below.

4.1 Different Bus protocols in FIR filter architecture

UART is a serial communication protocol used for the sharing of data between computers and peripherals. UART is a low-speed, short-distance, low-cost protocol. A large number of hardware devices require RS232 interface, Serial Peripheral Interface (SPI) or I^2C for serial communication. They include most of the equipment used in laboratories, quality and process controllers, various sensors, measuring instruments such as scales, and a lot more. UART / SPI / I2C is used on more advanced devices due to its higher transfer rate and built-in support for many microcontrollers, microprocessors and DSPs. Use a deep-memory technique, the mixed-signal oscilloscope (MSO) simplifies the debugging of the various physical-and command-layer problems that users normally experience. Notwithstanding the standard being so common, it's not easy to debug RS232, SPI, and I2C connections in the runtime setting.

The various aspects such as hardware and cabling, communication settings, and protocols should be considered as any mismatch here can cause a problem. In this research, a reconfigurable debugging architecture is proposed for UART/SPI/I^2C interface with the DSP processor (FIR filter). The debugging architecture uses a single common buffer source to trace out and trigger into UART, I^2C, SPI protocol which is interfaced with the FIR filter. An FSM based controller circuit executes instruction sets to debug communication protocol based on its activation by analyzing transmission signals. Also, the proposed FSM circuit controls the required buffer size and read/write operation based on the debugging requirement. DSP processors require different communication standards on-chip for multi signal processing. Debugging these standards on-chip is a difficult process as it requires large buffer sizes to debug or test each standard. To design a common trace buffer using the FSM controller this reduces computational complexity and device resources.

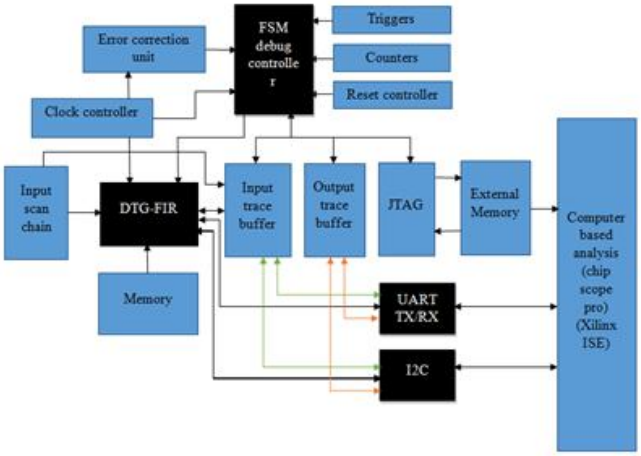

Figure 3: Debugging Architecture for DTG-FIR Filter with Serial Communication protocols

Generally, simulation is one of the basic processes in the VLSI implementation. This simulation should be performed before starting the debugging process. To effortlessly examine the corresponding structure bugs, a wide scope of the upgrade is applied to plan and checking the normal output against the simulated outputs. The significant bit of simulation is that recreation can be prepared in programming without the requirement of hardware. Simulation gets many structure mistakes; particularly function errors, misconception interface necessities, inaccurate detail, and different mistakes, which

can be distinguished through a basic test. The debugging architecture of the DTG-FIR filter is shown in fig.3. The most common debugging technique is to insert trace buffers in DTG based FIR filter and monitor those results in Chip scope pro analyzer. One trace buffer is inserted at input ports to trace FIR input and another buffer at the output port. Triggers are applied to switch internal logic core and results are traced by output buffers. Those traced signals are transferred to the Logic analyzer through JTAG port.

Apart from trace buffers, other modules should have inserted with core logic i.e.: counter module, timer module, error-correcting module, etc. The error-correcting unit replaces error with debugged logic values. The clock signals are used to control the error correction unit module. In this research work, three different bus protocols are used to transfer the data from PC to FPGA and vice versa such as UART, SPI, and I^2C. For both transmitter and receiver operation, parallel to serial and serial to parallel converter has used with respective baud clock generator. Based on the signal type, the protocol is selected to transfer the data. With the help of the error correction unit, the bugged error is solved in the bus protocols which help to give the proper input to the FIR module. FSM controller helps to use a reconfigurable buffer insertion technique with a common buffer. The detailed explanation of the entire bus protocols and FSM controller are explained below.

4.1.1 Implementation of UART

The proposed UART architecture consists of the basic UART sub modules, which are the receiver, transmitter and baud rate generator called the Prescaler. In addition, this design has internal buffers in both the receiver and the transmitter. Since this architecture operates in the serial clock domain and interacts with the parallel clock domain of the processor, we implement buffers using asynchronous FIFOs. The asynchronous FIFO architecture ensures smooth data transmission between two separate clock domains. Figure 4, below, displays the overall UART block diagram [5].

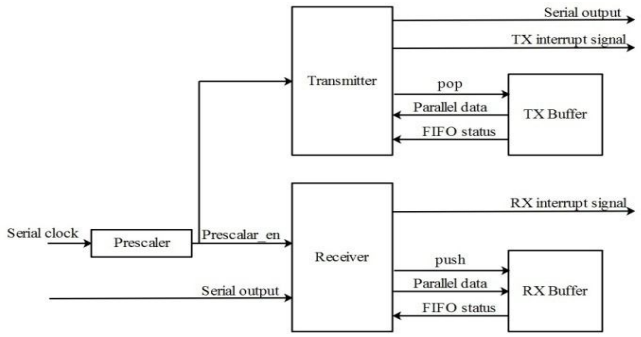

Figure 4: Overall block diagram of UART protocol

4.1.2. Implementation of I^2C:

Transfer from and to the master device is serial and is split into 8bit packets. All these basic specifications render implementing the I2C protocol very easy, even with cheap microcontrollers that do not have any unique I2C hardware controllers. You need 2 free I / O pins and a few basic I2C routines to send and receive commands. The original I2C guidelines set a fixed frequency of 100 kHz for the clock. Such as Quick mode was later expanded to 400 kHz. There is also a high-speed mode that will go up to 3.4 MHz and an ultra-fast mode of 5 MHz, too. Connection from and to the master computer is serial and is separated into 8bit packets. Both such basic specifications make it really straightforward to incorporate an I2C protocol, even for low-cost microcontrollers that do not have a special I2C hardware controller. It requires two free I / O pins and a few basic I2C routines to send and receive commands. The original I2C guidelines set a fixed clock frequency of 100 kHz. It was later improved to 400 kHz in quick mode. There is also a high-speed mode that can reach up to 3.4 MHz and there is also an ultra-fast 5 MHz mode.

UART, I2C and SPI are the most widely used serial protocols for both inter-chip and intra-chip low / medium bandwidth data transmissions. Depending on the type of signal, the bus protocols can pick and transmit data without failure. This useful knowledge allows designers to make deliberate and precisely designed architectural decisions. For a thorough comparative analysis, both protocols are implemented as general-purpose IP implementations, integrating all the requisite features needed by current ASIC / SoC applications according to a recent market review of a large number of commercial I2C and SPI devices [6].

4.2 FSM process

In the debugging module, the FSM controller helps to decide the communication protocol where the input signal has transferred. The block diagram of the FSM processing model is shown in Figure 5.

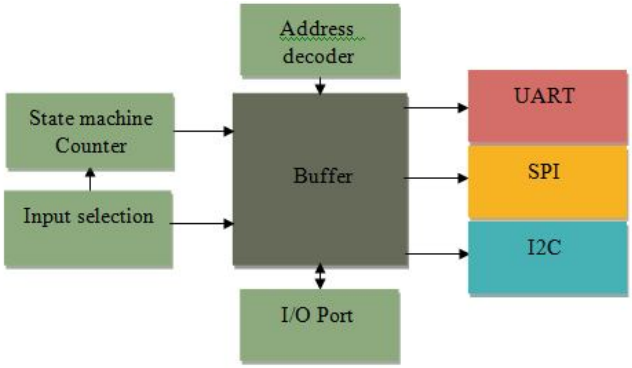

Figure 5: FSM processing diagram

Initially, the bus protocols are set as the state machine to transfer the information from one module to another module. If state machine is 1, the UART protocol is enabled to perform their operation. If state machine is 2 and 3, SPI and I^2C bus protocols are enabled to transfer the intermediate data. Based on the input selection module, the bus protocols are selected and processed their operation. For performing the UART, state

machine 4 is allocated for memory which contains the 10 bit (start bit, stop bit, 8 bit data) as well as SPI and I^2C have allocated in the state machine 5 and 6 memory. State machine 5 and 6 undergone the process of finding the master and slave. SPI and I^2C bus protocols contain the memory size of 16 bit or 32 bit information based on the ADC model. This entire process is performed in the state machine counter. Address decoder is used to assign the memory for storing the data in buffer. Based on the input selection, the bus protocols are selected and the outputs are delivered in the I/O port terminal. The experimental results are explained in the below section.

5. EXPERIMENTAL SETUP

The proposed DTG-FIR filter architecture was experimented using 4GB RAM with 3.30 GHz, i3 processor, and 500GB hard disk. The architecture has been implemented using Verilog language in Xilinx 14.4 software. By using Xilinx Vivado 14.4, the filter architecture was integrated with the Integrated Logic Analyzer. By using the Chip Scope analyzer, the trigger value has applied to the different taps which helped to select the serial communication protocols. This research work is implemented and verified in Spartan 6 FPGA devices.

5.1 Results and Discussion

In this research work, the DTG-FIR filter has implemented and verified the performance parameters. The main parameters of the FIR filter are input and the co-efficient value. Both input and co-efficient are generated randomly and it should multiply and accumulate the multiplication results. Here, 16 tap FIR filters have implemented in the Spartan 6 FPGA device. Figure 6, shown the DTG-FIR filter waveform which is taken from Modelsim software.

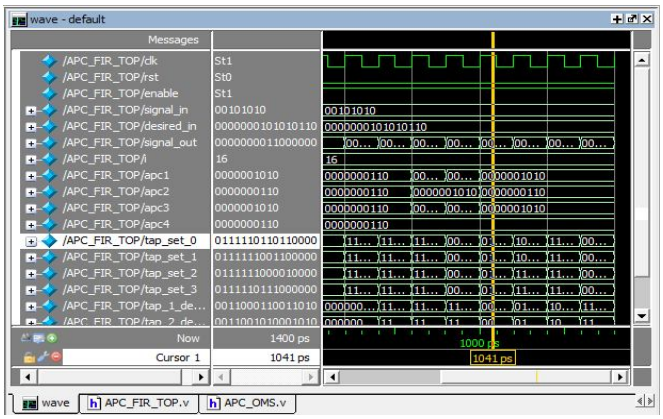

Figure 6: The waveform of DTG-FIR filter architecture

The waveform contains control signals, input ports, and output ports. Clock, enable and reset signals are represented as 1 to control the entire filter architecture. The DTG-FIR filter architecture input is generated randomly which is representing as signal_in. This input has to perform the filter operation with co-efficient which also generated randomly. The co-efficient data is represented as desire in. These two values are performed in the DTG-FIR filter architecture for each clock cycle. Each clock cycle output is stored in the signal out register. From this simulation, it is cleared that the proposed architecture has simulated perfectly without any execution error. After the simulation, the same Verilog code is undergone for the implementation process. From that implementation process, the FPGA performances are evaluated. The Verilog code is synthesized in the Spartan 6, Virtex 4, Virtex 5, and Virtex 6 FPGA device that results are given in Table 1.

Table 1: FPGA performance for Spartan 6 target devices

Target device	Method	LUT	Flip flop	Slices	Frequency (MHz)	Power(W)
Virtex 4	MBI	984/10944	874/10944	447/5472	95.68	0.017
	RD-UART	811/10944	854/10944	443/5472	264.21	0.0587
	RD-I^2C	701/10944	514/10944	405/5472	301.77	0.0188
Virtex 5	MBI	874/12480	887/12480	354/3120	101.58	0.025
	RD-UART	654/12480	354/12480	547/3120	267.36	0.015
	RD-I^2C	641/12480	341/12480	333/3120	354.21	0.058
Virtex 6	MBI	777/46560	987/93120	985/11640	125.62	0.087
	RD-UART	875/46560	655/93120	648/11640	288.54	0.0697
	RD-I^2C	677/46560	601/93120	554/11640	369.32	0.087
Spartan 6	MBI	1043/5720	876/11440	478/1430	71.242	0.017
	RD-UART	875/5720	592/11440	463/1430	245.02	0.0135
	RD-I^2C	754/5720	539/11440	427/1430	328.12	0.018

In this comparison, four different architectures are implemented and the results are compared in the table. In the conventional method, More Buffer Insertion (MBI) technique has used to debug the filter architecture. Due to the usage of more buffers in the architecture, the hardware components have increased rapidly. To overcome this issue and debug the error is the bus protocols, Reconfigurable Debugging (RD) technique is used in all serial communication protocols. Among the three protocols, I^2C occupied less hardware utilization and the power consumption is less in UART.

5.2 Comparative Analysis

In this section, the proposed method is compared with different conventional debugging models such as FPGA-in-the-loop (FIL) [14], Incremental Trace Buffer [15], High Level Synthesis (HLS) [8], [18] and Signal Tracing Technique (STT) [13]. All the conventional architectures were used more number of buffers for debugging each RTL module. Due to the more number of buffers, the delay and the entire system area have increased. More logical elements have used in conventional methods which increase the area of the entire architecture. Moreover, all the conventional methods are failed to identify the bugs in the bus protocols during the transmission. But, the proposed architecture can identify and rectify the error which occurred in the bus protocols. The comparison of the proposed method with the existing design is given in Table 2, Table 3 and Table 4. The pictorial representation of the comparison is shown in Figure 7, Figure 8 and Figure 9.

Table 2: Comparative analysis of ZED performances

Target Device	Designs	LUT	Flip flop	Slices
ZED (xc7z020-484)	HLS [8]	7685	4948	1542
	FIL[14]	3962	854	825
	RD-UART	1541	844	815
	RD-I^2C	1125	798	775

Table 3: Comparative analysis of Stratix 4 performances

Target FPGA	Designs	LUT	Flip flop	Slices
Stratix 4 (EP4SGX110)	ITB [15]	3054	691	720
	RD-UART	2988	687	714
	RD-I^2C	2541	612	674

Table 4: Comparative analysis of Stratix 5 performances

Target FPGA	Designs	LUT	Flip flop	Slices
Stratix 5 (5sgxea7n2f45c2)	STT [13]	3301	1206	1420
	RD-UART	3214	1154	1358
	RD-I^2C	3001	1048	1264

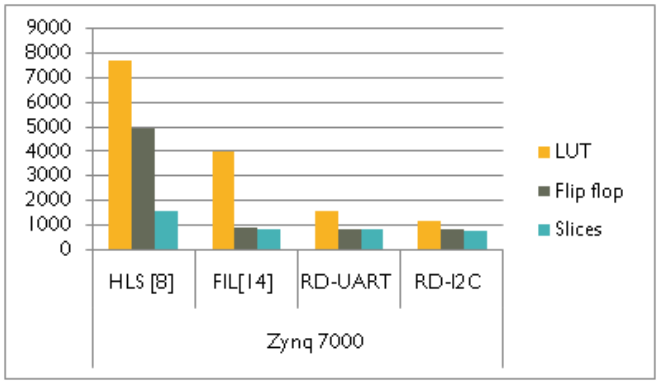

Figure 7: Comparison of Zynq 7000 FPGA performance for different methods

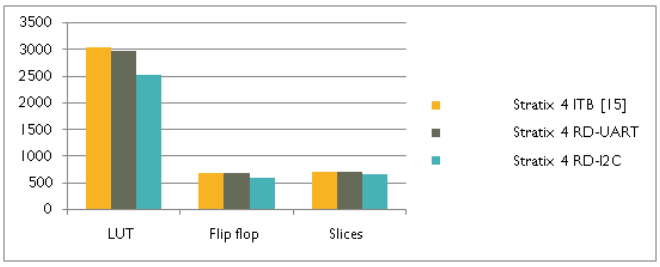

Figure 8: Comparison of Stratix 4 FPGA performance for different methods

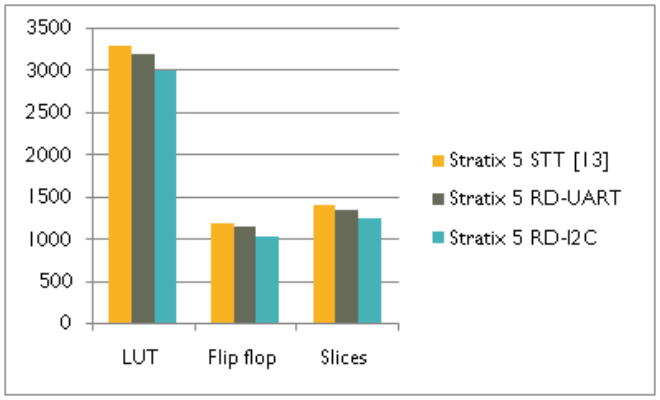

Figure 9: Comparison of Stratix 5 FPGA performance for different methods

5.3 FPGA based Serial Communication Protocol Debugging

The process of connecting input port and buffer connection detail is shown in Figure 10, which contains the net name, source instance, source component, and base type. Clock signals and trigger data signals are major blocks in the debugging process [19-36]. For the synchronization block, the clock is a major element to control the entire architecture. Trigger data helps to set and reset the respective tap window and enable the output values in the output simulation window. Trigger representation is given in Table 5.

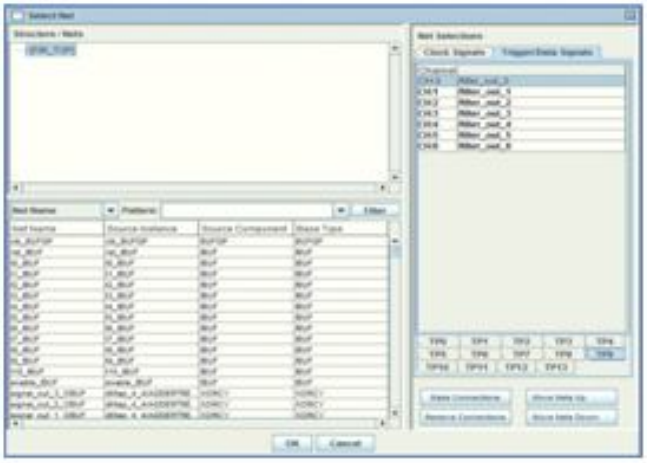

Figure 10: Net selection process

Table 5: Trigger name and respective functions

Trigger name	Function
1TP0	Reset
TP1-TP8	Tap selection
TP9	Filter output
TP10	UART
TP11	I^2C
TP13	Serial communication protocol output

Among the 14 trigger signals, TP0 is used to reset the design structure and helps to process the interior input terminal. TP1 – TP8 represented as the tap selection process. Every trigger is used to select the respective taps to perform the filter output. Based on the tap values, the filer output is generated which is triggered in TP9. After that, major blocks of this research work are triggered in TP10 and TP11. The UART and I^2C bus protocols are used to transmit the data from one place to another place. In this research work, the bug can be identified in the protocols itself. If the wrong input is given to the filter module, the FIR filter architecture also produces the wrong output. In this kind of situation, the error can be checked and debugged in the bus protocols.

The waveform parameters are explained below,

- rst_IBUF – Reset signal
- From t0_IBUf to t7_IBUF – Taps
- t8_IBUF – UART protocol
- t10_IBUF – I^2C protocol

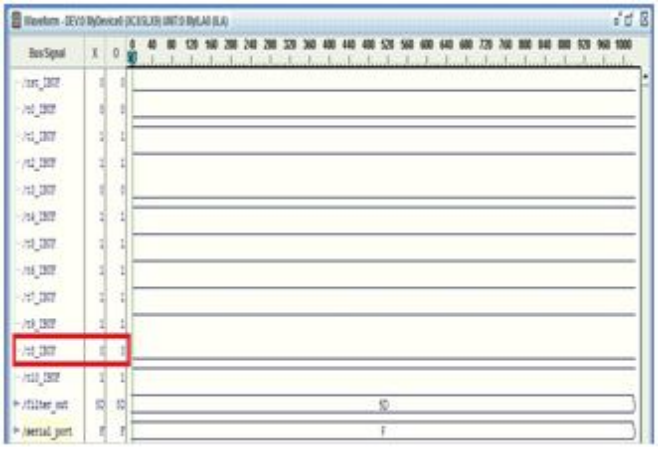

Figure 11: UART error signal

Generally, the UART protocol has affected by two kinds of errors such as synchronization mismatch and data mismatch. In synchronization mismatch, the start and stop bits are needed to be checked and the bug can be rectified. If the start and stop bit has changed from the input data, it has been inverted and given to the FIR filter terminal. Moreover, intermediate data also should be check to identify the error in the UART protocols. Based on the error correction code, the errors have been solved and the FIR filter architecture produces the error-free output. When the t8_IBUF trigger is 0, the UART protocol is enabled and helps for the data communication. The debugged output is shown in Figure 11.

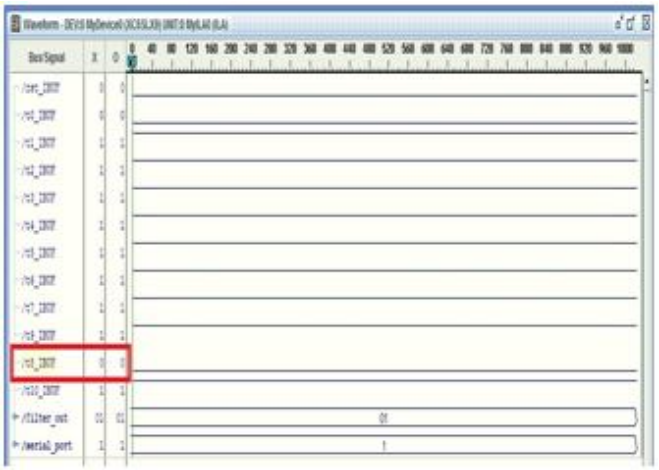

Figure 12: UART error-free output

Serial port output (F) is generated wrongly that caused the wrong FIR filter output. The error free UART output is shown in Figure 12. For certain cases, an error has occurred in UART protocol. At that time, the data can be transferred to the I^2C bus protocol is shown in Figure 13, which can be enabled when the t10_IBUF is triggered to zero. For all the waveform windows, the FIR filter outputs are delivered in the filter_out variable and the serial port indicates the bus protocol operation.

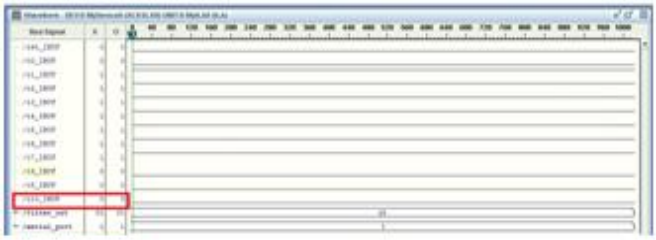

Figure 13: I^2C output waveform

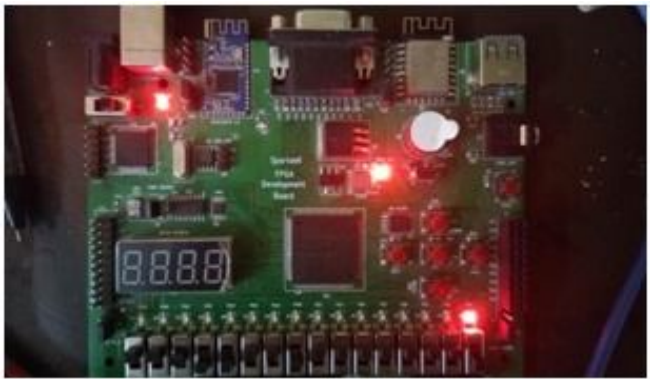

Figure 14: UART protocol hardware output

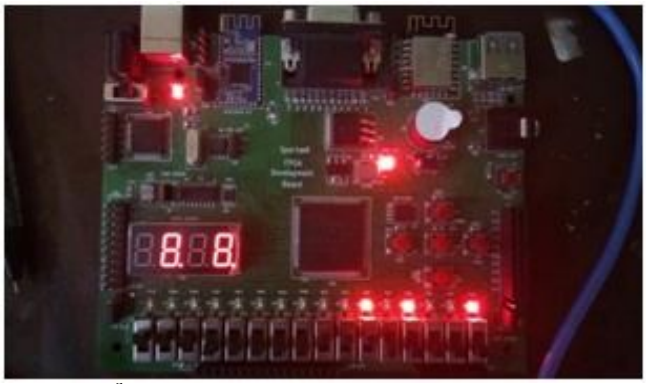

Figure 15: I^2C protocol hardware output

The respective hardware outputs are shown in Figure 14 and Figure 15. For I^2C protocols are working in 5 tap operation mode. 4^{th} and 5^{th} tap input is considered as 3, and 4. Similarly, the co-efficient is considered as 4 and 5. These input and co-efficient have performed the filter operation. The different kind of tap input is given to the serial port to perform the filter operation. After processing the filter operation, the perfect output is delivered to the filter output terminal without any bug. Finally, due to the usage of the RD technique in serial communication protocols, the hardware utilization of entire architecture has improved and operated at high speed.

6. CONCLUSION AND FUTURE SCOPE

In recent years, the debugging process has been applied to the DSP processor unit which contains a different kind of processing blocks. All the processing blocks are doesn't contain unique characterization. Some of the blocks are processing the analog signal as an input and some of the modules are operated in digital signals. So, bus protocol is one of the major ports to transfer the information from one processing unit to another processing unit. The bug may be present in bus protocols also which affects the system performances and data loss has occurred. So, this research work concentrates on identifying and rectifies the bug in the UART, I^2C, and SPI bus protocols. All the types of signal input can transfer to the DSP processing unit which designed with the FIR filter module. If the bug is identified in the UART protocol, the data can transfer through the SPI protocol. Reconfigurable buffer insertion technique helps to reduce hardware utilization. I^2C bus protocol occupied 754 LUTs, 539 flip flops, 427 slices which are too less than other protocols. SPI protocol worked in the operating frequency of 330.12 MHz. According to the power consumption, the UART protocol consumes 0.0135W which is better than other protocols. In the future, different optimization techniques will be implemented to verify MpSoC architecture.

REFERENCES

1. Anumothu, Murali, Kishore, K.H. and Reddy, D.V., 2016. Integrating FPGAs with trigger circuitry core system insertions for observability in debugging process. *Journal of Engineering and Applied Sciences*, *11*(12), pp.2643-2650.
2. Pandey, B., Pandey, N., Kaur, A., Hussain, D.A., Das, B. and Tomar, G.S., 2019. Scaling of Output Load in Energy Efficient FIR Filter for Green Communication on Ultra-Scale FPGA. *Wireless Personal Communications*, *106*(4), pp.1813-1826. https://doi.org/10.1007/s11277-018-5717-2
3. Eslami, F. and Wilton, S.J., 2018. Rapid triggering capability using an adaptive overlay during FPGA debug. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *23*(6), pp.1-25.
4. Hung, E. and Wilton, S.J., 2012. Scalable signal selection for post-silicon debug. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *21*(6), pp.1103-1115.
5. Mahat, N.F., 2012, September. Design of a 9-bit UART module based on Verilog HDL. In *2012 10th IEEE International Conference on Semiconductor Electronics (ICSE)* (pp. 570-573). IEEE. https://doi.org/10.1109/SMElec.2012.6417210
6. Oudjida, A.K., Berrandjia, M.L., Tiar, R., Liacha, A. and Tahraoui, K., 2009, December. Fpga implementation of i 2 c &spi protocols: A comparative study. In *2009 16th IEEE International Conference on Electronics, Circuits and Systems-(ICECS 2009)* (pp. 507510). IEEE.
7. Aguirre, M.A., Tombs, J.N., Baena-Lecuyer, V., Mora, J.L., Carrasco, J.M., Torralba, A. and Franquelo, L.G., 2005. Microprocessor and FPGA interfaces for in-system codebugging in field programmable hybrid systems. Microprocessors and Microsystems, 29(2-3), pp.75-85. https://doi.org/10.1016/j.micpro.2004.06.009
8. Monson, J.S. and Hutchings, B.L., 2018. Enhancing debug observability for HLS-based FPGA circuits

through source-to-source compilation. Journal of Parallel and Distributed Computing, 117, pp.148-160.
9. Farooq, U., Chotin-Avot, R., Azeem, M., Ravoson, M. and Mehrez, H., 2018. Novel architectural space exploration environment for multi-FPGA based prototyping systems. Microprocessors and Microsystems, 56, pp.169-183.
10. Kourfali, A. and Stroobandt, D., 2019. In-circuit fault tolerance for FPGAs using dynamic reconfiguration and virtual overlays. Microelectronics Reliability, 102, p.113438. https://doi.org/10.1016/j.microrel.2019.113438
11. Khan, H.U.H. and Göhringer, D., 2017, April. FPGA Debugging with MATLAB Using a Rule-Based Inference System. In *International Symposium on Applied Reconfigurable Computing* (pp. 106-117). Springer, Cham.
12. Hung, E., Goeders, J.B. and Wilton, S.J., 2014, April. Faster FPGA debug: Efficiently coupling trace instruments with user circuitry, International Symposium on Applied Reconfigurable Computing (pp. 73-84). Springer, Cham.
13. Goeders, J. and Wilton, S.J., 2016. Signal-tracing techniques for in-system FPGA debugging of high-level synthesis circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36(1), pp.83-96. https://doi.org/10.1109/TCAD.2016.2565204
14. Podlubne, A. and Göhringer, D., 2019. Intrusive FPGA-in-the-loop debugging using a rule based inference system. Microprocessors and Microsystems, 64, pp.185-194.
15. Hung, E. and Wilton, S.J., 2013. Incremental trace-buffer insertion for FPGA debug. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(4), pp.850-863.
16. Chella Santhosh, K. Hari Kishore, G. Pavani Lakshmi, G.Kushwanth, P. Rama Krishna Dharma Teja, R. S. Ernest Ravindran, Sree Vardhan Cheerala, M. Ravi Kumar **"**Detection of Heavy Metal Ions using Star-Shaped Microfluidic Channel" International Journal of Emerging Trends in Engineering Research, ISSN: 2347-3983, Volume-7 Issue-12, Page No: 768-771, December 2019. https://doi.org/10.30534/ijeter/2019/067122019
17. B. Srikanth, M. Siva Kumar, J.V.R. Ravindra, K. Hari Kishore **"**Double Precession Floating Point Multiplier using Schonhage-Strassen Algorithm used for FPGA Accelerator" International Journal of Emerging Trends in Engineering Research, ISSN: 2347-3983, Volume-7 Issue-11, Page No: 677-684, December 2019. https://doi.org/10.30534/ijeter/2019/437112019
18. Radhika Rani Chintala, Lakshmi Sri Ram Janjanam, Sai Kousik G, Sai Pawan S ''FPGA Implementation of Katan Block Cipher for Security in Wireless Sensor Networks'' International Journal of Emerging Trends in Engineering Research , ISSN: 2347-3983, Volume-7 Issue-11 Page No: 492-497, December 2019. https://doi.org/10.30534/ijeter/2019/157112019
19. Mahesh Madavath, K Hari Kishore **"**RF Front-End Design of Inductorless CMOS LNA Circuit with Noise Cancellation Method for IoT Applications" International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-8, Issue No: 6, Page No: 176-183, April 2019.
20. K.Sarath Chandra, K Hari Kishore **"**Electrical Characteristics of Double Gate FINFET under Different Modes of Operation" International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-8, Issue No: 6S, Page No: 172-175, April 2019.
21. P.Ramakrishna, M. Nagarani, K Hari Kishore **"**A Low Power 8-Bit Current-Steering DAC Using CMOS Technology" International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-8, Issue No: 6S, Page No: 137-140, April 2019.
22. Avinash Yadlapati, K Hari Kishore **"**Implementation of Asynchronous FIFO using Low Power DFT" International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-8, Issue No: 6S, Page No: 152-156, April 2019.
23. P Ramakrishna, K Hari Kishore **"**Implementation of Low Power and Area Efficient 7-Bit Flash Analog to Digital Converter" Journal of Computational and Theoretical Nanoscience, ISSN: 1546-1955, Volume-16, Issue No: (5-6), Page No: 2213-2217, June 2019.
24. Mahesh Madavath, K Hari Kishore **"**Design and Analysis of Receiver Front-End of CMOS Cascode Common Source Stage with Inductive Degeneration Low Noise Amplifier on 65 nm Technology Process" Journal of Computational and Theoretical Nanoscience, ISSN: 1546-1955, Volume-16, Issue No: (5-6), Page No: 2628-2634, June 2019. https://doi.org/10.1166/jctn.2019.7942
25. K Hari Kishore, Fazal Noorbasha, Katta Sandeep, D. N. V. Bhupesh, SK. Khadar Imran, K. Sowmya "Linear convolution using UT Vedic multiplier" International Journal of Engineering and Technology(UAE), ISSN No: 2227-524X, Vol No: 7, Issue No: 2.8, Page No: 409-418, March 2018.
26. K Hari Kishore, B. K. V. Prasad, Y. Manoj Sai Teja, D. Akhila, K. Nikhil Sai, P. Sravan Kumar "Design and comparative analysis of inexact speculative adder and multiplier" International Journal of Engineering and Technology(UAE), ISSN No: 2227-524X, Vol No: 7, Issue No: 2.8, Page No: 413-426, March 2018. https://doi.org/10.14419/ijet.v7i2.8.10472
27. G Siri Vennela, K Hari Kishore, E Raghuveera "High Accurate and Power Efficient ECG-Based Processor for Predicting Ventricular Arrhythmia" Journal of Advanced Research in Dynamical and Control Systems, ISSN No: 1943-023X, Vol No: 10, Issue No: 2, Page No: 1180-1121, May 2018.
28. Avinash Yadlapati, K Hari Kishore "Low Power Synthesis for Asynchronous FIFO using Unified Power Format (UPF)" International Journal of Engineering and

Technology (UAE), ISSN No: 2227-524X, Vol No: 7, Issue No: 2.8, Page No: 7-9, March 2018.

29. Chella Santhosh, K. Hari Kishore, G. Pavani Lakshmi, G.Kushwanth, P. Rama Krishna Dharma Teja, R. S. Ernest Ravindran, Sree Vardhan Cheerala, M. Ravi Kumar **"**Detection of Heavy Metal Ions using Star-Shaped Microfluidic Channel" International Journal of Emerging Trends in Engineering Research, ISSN: 2347-3983, Volume-7 Issue-12, Page No: 768-771, December 2019. https://doi.org/10.30534/ijeter/2019/067122019
30. B. Srikanth, M. Siva Kumar, J.V.R. Ravindra, K. Hari Kishore **"**Double Precession Floating Point Multiplier using Schonhage-Strassen Algorithm used for FPGA Accelerator" International Journal of Emerging Trends in Engineering Research, ISSN: 2347-3983, Volume-7 Issue-11, Page No: 677-684, December 2019. https://doi.org/10.30534/ijeter/2019/437112019
31. Nadhindla Bala Dastagiri, Kakarla Hari Kishore, Vinit Kumar Gunjan and Shaik Fahimuddin, "Design of a Low-Power Low-Kickback-Noise Latched Dynamic Comparator for Cardiac Implantable Medical Device Applications", Lecture Notes in Electrical Engineering, ISSN No: 1876-1100, E-ISSN: 1876-1119, pp. 637-645, 2018.
32. Avinash Yadlapati, Hari Kishore Kakarla **"**Low-power design-for-test implementation on phase-locked loop design" Measurement and Control, ISSN: 0020-2940, Volume-52, Issue No: (7-8), Page No: 995-1001, June 2019.
33. Nan Jiang, Abdol Ghaffar Ebadi, Kakarla Hari Kishore, Qahtan.A.Yousif, Mohammad Salmani "Thermomechanical Reliability Assessment of Solder Joints in a Photo-voltaic Module Operated in a Hot Climate" *IEEE Transactions on Components, Packaging and Manufacturing Technology*, P-ISSN: 2156-3950, E-ISSN: 2156-3985, Vol No: 10, Issue No: 1, Page No: 160-167, January 2020.
34. Mahesh Madavath, Hari Kishore Kakarla, Azham Hussain, C.S. Boopathi **"**Design and Analysis of CMOS RF Receiver Front-End of LNA for Wireless Applications" Microprocessors and Microsystems (SCI), ISSN: 0141-9331, Volume-75, June 2020. Article number 102999. https://doi.org/10.1016/j.micpro.2020.102999
35. K Divya Madhuri, K Hari Kishore "Implementation of 4-bit Ripple Carry Adder by Adopting Sub threshold Adiabatic Logic for Ultralow-Power Application" Journal of Advanced Research in Dynamical and Control Systems, ISSN No: 1943-023X, Vol No: 12, Issue No: 6, Page No: 11-17, May 2020.
36. B Srikanth, M Siva Kumar, J V R Ravindra, K Hari Kishore **"**The enhancement of security measures in advanced encryption standard using double precision floating point multiplication model" Transactions on Emerging Telecommunications Technologies, ISSN: 2161-3915, Volume: 31, Feb 2020. https://doi.org/10.1002/ett.3948