International Conference on Mechanical and Industrial Engineering
ICMIE2022
November 15-17, 2022 Tripoli-Libya

Design and Comparative Analysis of An Alternative Approach of Bus-Interconnect for Systems on Programmable Chip

Mohamed Muftah Eljhani
Department of Computer Engineering
University of Tripoli
Tripoli, Libya
M.Eljhani@uot.edu.ly

***Abstract*—The expanding complexity of the System-on-Programmable-Chip (SOPC) has driven to the basic "design productivity gap" issues. SOPC design instantiation and interface encompasses a critical effect on system execution time and power dissipation. This paper overviews existing digital system buses which are commonly used in SOPC systems, discusses different bus architectures, and introduces a new bus architecture. Also, we present the design of bus controller which handles the transactions between data path modules. Tri-state-based bus implementation is useful for very large design applications with large number of blocks. Field Programmable Gate Arrays (FPGA) chips don't contain a sufficient number of tristate drivers to implement a system with large buses, the alternative we proposed a new multiplexer-based bus structure and bus controller. In this research work the basic modules of the proposed multiplexer-based bus system is designed, implemented, and simulated with verilog hardware description language (HDL), and placed and routed to Cyclone IV GX FPGA, and then compared with tri-state-based bus system and proved that the multiplexer-based bus is faster and has less power dissipation, additionally it is more flexible for timing and testing process. The proposed bus architecture can be used in FPGA and other programmable chips that needs high speed bus with low power consumption. In the SOPC design, most design applications use multiplexer-based bus, because Intellectual Property (IP) integration is easier than tri-state-based bus. Also, in the application specific integrated circuit (ASIC) design uses internal multiplexer-based bus because tri-state-based bus has issues with timing and power consumption due to the capacitive load by its nodes.**

Keywords—multiplexer-based bus structure, system-on-programmable-chip, FPGA design, field programmable gate arrays, verilog hardware description language

I. INTRODUCTION

Since the early days of computers and telephony, interconnection networks have been a critical part of electrical engineering [1]. This has become even more critical in the era of very large-scale integration (VLSI) circuitry because of the drive characteristics of metal oxide semiconductor (MOS) transistors [2]. Buses, although the simplest form of interconnect, are a poor choice from a density or power standpoint because the power and space required to drive them at maximum speed grow exponentially with the capacitance of the bus [3]. Buses in computer systems are a set of digital communication lines that allows two or more modules inside the system to transfer and exchange data, those communication lines are used inside a computer system or within a single integrated circuit (IC). Bus system can be optimized for a specific purposes, such as a memory bus and an I/O buses, they actually act as a vital medium for transferring data. Basically, internal system bus connects different modules within the central processing unit (CPU). In the reduced instruction set computer (RISC) the data bus is 8-bits, the system fetches every instruction twice, first the higher 8-bits and then the lower 8-bits, controlled by the state, when state is 0, the higher 8- bits of the instruction sent to the responsive register, and the state is set to 1. For the next operation, the lower 8-bits of the instruction is sent to the responsive register because the state is 1. Reset is triggered by signal rst. When rst is set to 1, the system will end the current operation and hold on in reset state as long as reset high, all registers are initialized to 0. The data bus hold in high-impedance state and the address is set to 0 [4]. Early computer buses were literally parallel electrical wires with multiple connections, but modern computer buses can use both parallel and bit serial connections. Buses can also connect two different components at the same time through the usage of point-to-point or multipoint technique. System-On-chip bus architectures have a significant effect on system speed and power dissipation. System designers, as well as the research community have focused on the issue of exploring, evaluating, and designing SoPC communication architectures to meet the targeted design goals [5]. The alternatives to buses are many, and all have been used successfully in various computers, chips, boards, ASICs, and FPGAs. These alternatives are no panacea, just as buses aren't a cure-all for every interconnection ailment. Avoiding the fixed routing and timetable of a standard bus can open up new avenues for design, and restore a bit of glamour and creativity to an otherwise mundane project [6]. The Electronic Design Automation (EDA) design flow typically follows a path from Verilog/VHDL hardware description language [7], or schematic design entry through synthesis

and place and route tools to the programming of the FPGA. The Proposed multiplexer bus system is designed, simulated, and compared with the tristate system bus using the verilog hardware description language, and implemented on the Altera FPGA development board, Cyclone IV GX FPGA [8], [9]. In previous research work we designd and simulated microprocessor system bus using tri-state and multiplexer buses, and found that tri-state-based bus implementation is useful for any large design application with a large number of design blocks, but at the same time, it can complicate synchronization and testing. FPGA chips do not have enough tristate drivers to mount large buses. Alternatively, designers can use bus structures based on multiplexers. The basic modules of the proposed microprocessor bus system are designed, implemented, and simulated using Verilog hardware description language (HDL) [10].

This paper consist of seven sections which includes the abstract in section I. An introductory part in section II. A brief introduction is given on the general bus structure in section III. Section IV illustrates the design methodology and procedure of the tristate bus structure and the proposed multiplexer bus structure, (section A and B respectively), showing the design block diagrams and simulation waveforms. Simulation and results are in section V. The conclusion and references of the paper are in sections VI and VII respectivally. The simplest way to interconnect registers in a system is to use a single bus structure, all datapath registers are connected directly to the bus, because the bus can be used for only one transfer at a time, so only two registers can actively use the bus at any given time. In the design architecture, the bus is a subsystem that used to transfer data between datapath registers. To achieve high speed of operation, the design organized so that all its units can handle one full word (8-bit) of data at a given time. When a word of data is transferred between registers, all its bits are transferred in parallel simultaneously, that is, the bits are transferred over 8-wires, one bit per wire, those wires serves as a connecting path between data path registers. This common set of wires is called a bus. If common bus are used to transfer data from multiple source registers to multiple destination registers, it is very important to ensure that only one register acts as a source register at any given time and other registers do not interfere the operation.

II. METHODOLOGY

For the comparison reason, the common tristate-based bus and the proposed multiplexer-based bus systems were designed, implemented, and simulated. The design is simulated and verified using Modelsim and Quartus II Intel Altera software tools, and designed by Verilog hardware description language. Generally, each system consists of two main modules, the data path module and the control unit module. Datapath module is used to store and manipulate data and to transfer data from one data path register of the system to another (datapathx0, datapathx1, datapathx2, datapathx3. Datapath modules comprise registers, tristate buffers, and multiplexers building blocks. The control unit module controls the operation of the data path module. The state machine module is the control core of the design, it acts as a brain and its main task is to generate a series of control signals to control and operate the data path module.

A. *Tristate-based Bus System*

The tristate-based bus architecture is consisting of two main modules data path module and control unit module. The data path module is consisting of nine sub-modules. The system contains four 8-bit registers, datapathx0 to datapathx3 “Fig. 1”, shows how these registers are connected using tri-state drivers to implement the bus structure. The data outputs q of each register is connected to tri-state drivers. When selected by their enable signals, the drivers place the contents of the corresponding register onto the bus wires. If the enable input is set to 1, then the contents of the register will be changed on the next active edge of the clock.

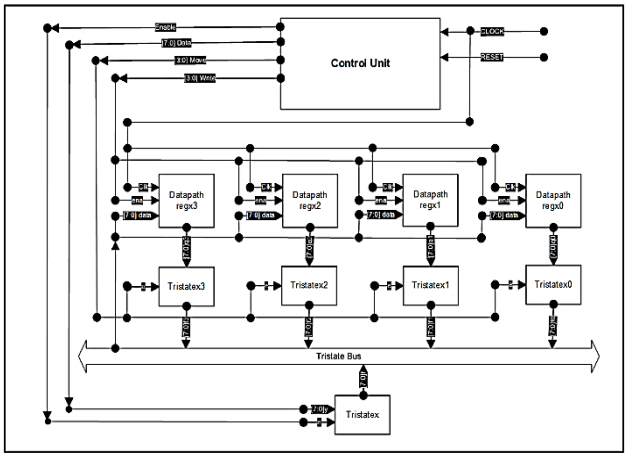

Fig. 1. Tristate Bus Structure

The enable input on each register is labeled ena, which stands for enable. The signal that controls the ena input for registers is denoted as [3;0] Write, while the signal that controls the associated tri-state driver is called [3:0] Move. These signals are generated by the control unit module. In addition to four registers, there is other module block that connected to the bus. The circuit diagram shows how 8-bits of data from an external source that is placed on the same bus, using the control input signal e that is generated by control unit module named Enable. It is essential to ensure that only one circuit block attempts to place data onto the bus wires at any given time. The control unit must ensure that only one of the tri-state drivers enable signals, datapathx0, datapathx1, datapathx2, datapathx3. are asserted at a given time. The control unit also produces the signals [3:0] Write, which determine when data is loaded into each register. In general, the control unit perform a number of functions, such as loading resisters with data and transferring the data stored in one register into another register. The control circuit is synchronized by a clock input, which is the same clock signal that controls the four data path registers. “Fig. 2”, shows the simulation waveforms of the tristate bus module.

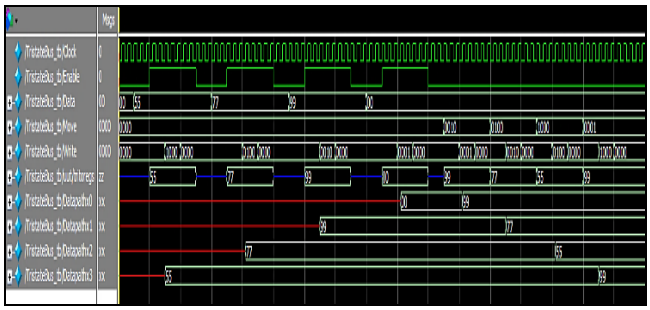

Fig. 2. Tristate Bus Simulation Waveforms

The control signals named [3:0] Write allow the externally supplied data to be loaded from the bus into data path registers. When [3:0] Write = 1000, the data is loaded into datapathx3; When [3:0] Write = 0100, the data is loaded into datapathx2; When [3:0] Write = 0010, the data is loaded into datapathx1; and When [3:0] Write = 0001, the data is loaded into datapathx0; The finite state machine (FSM) that implements the control performs those operations. The data path regx3 loaded with 55 hex, data path regx2 loaded with 77 hex, data path regx1 loaded with 99 hex, and data path regx0 loaded with 00 hex. After all, data path registers loaded with data, the contents of data path regx1 moved to data path regx0, data path regx2 moved to data path regx1, data path regx3 moved to data path regx2, and data path regx0 moved to data path regx3.

B. Multiplexer-based Bus System

The proposed multiplexer-based bus architecture is consisting of two main modules named data path module and control unit module. The data path module is consisting of six sub-modules. The system contains four 8-bit registers, datapathx0 to datapathx3 “Fig. 3”, shows how these registers are connected using multiplexers instead of tristate buffers to implement the bus structure. The data outputs q of each register is connected to 4to1 multiplexer. When selected by their [1:0] e signals, the multiplexer places the contents of the corresponding register onto 2to1 multiplexer that is connected to the bus. The enable input on each data path register is labeled ena, which stands for enable. The signal that controls the ena input for registers is denoted as [3;0] Write, while the signal that controls the associated multiplexer output is called [1:0] Move. These signals are generated by the control unit module.

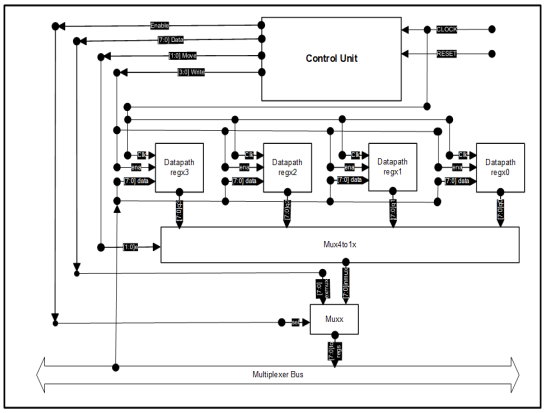

Fig. 3. Multiplexer Bus Structure

In addition to four registers, there is other module block that connected to the bus. The circuit diagram shows how 8-bits of data from an external source that is placed on the same bus, using the control input signal Enable that connected to the sel signal of the 2to1 mux that is generated by control unit module. It is essential to ensure that only one circuit block attempts to place data onto the bus wires at any given time. The control unit must ensure that only one of the enable signals, datapathx0, datapathx1, datapathx2, datapathx3. are asserted at a given time. The control unit also produces the signals [3:0] Write, which determine when data is loaded into each register. The control circuit is synchronized by a clock input, which is the same clock signal that controls the four data path registers. “Fig. 4”, shows the simulation waveforms of the multiplexer bus module.

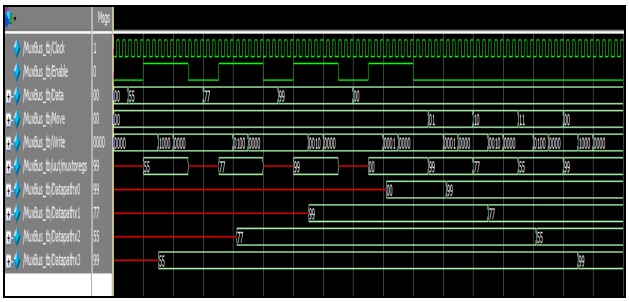

Fig. 4. Multiplexer Bus Simulation Waveforms

The control signals named [3:0] Write allow the externally supplied data to be loaded from the bus into data path registers. When [3:0] Write = 1000, the data is loaded into datapathx3; When [3:0] Write = 0100, the data is loaded into datapathx2; When [3:0] Write = 0010, the data is loaded into datapathx1; and When [3:0] Write = 0001, the data is loaded into datapathx0; The finite state machine (FSM) that implements the control performs those operations. Inertially, we loaded data path registers with same values as in the tristate bus to compare the results of the proposed multiplexer-based bus with the tristate-based bus. The data path regx3 loaded with 55 hex, data path regx2 loaded with 77 hex, data path regx1 loaded with 99 hex, and data path regx0 loaded with 00 hex. After all, data path registers loaded with data, the contents of data path regx1 moved to data path regx0, data path regx2 moved to data path regx1, data path regx3 moved to data path regx2, and data path regx0 moved to data path regx3.

III. SIMULATION AND RESULTS

After design and simulate the multiplexer bus sub-modules individually, we instantiated, simulated and verified the system as top-level module. Then the system was compared to the tristate bus system to evaluate the functionality, speed and power dissipation of the proposed multiplexer bus system against the tristate bus system that is designed and implemented in the design specially for this purpose. “Fig. 5”, shows the simulation waveforms of multiplexer bus and tristate bus systems.

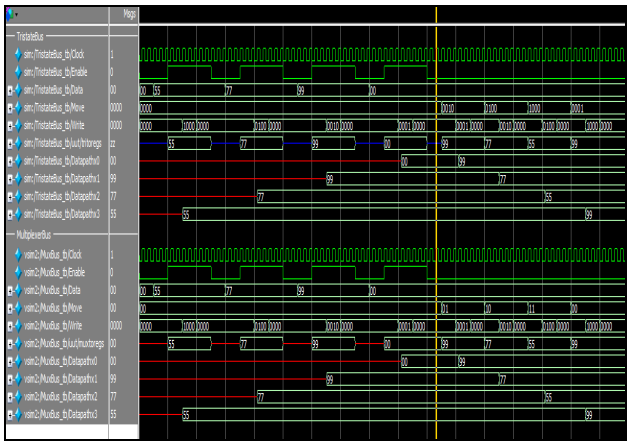

Fig. 5. Simulation Waveforms of Tristate Bus and Multiplexer bus Systems

The resulting simulation waveforms show that both systems are used the same data set, and both are working identical, except that the first module used tristate bus while the second module used multiplexer bus to connect data path registers together.

A. Register Transfer level

Register-transfer-level (RTL) used in verilog hardware description language to create high-level representations of both tristate bus and multiplexer bus modules from lower-level representation. "Fig. 6", shows register transfer level (RTL) model for tristate bus module, and "Fig. 7", shows RTL model for multiplexer bus. "Fig. 6, 7", show how data is transformed as it is passed from register data path to another register data path in the design. The loading and transforming of the data to registers data path is performed by the combinational logic part that exists between all registers.

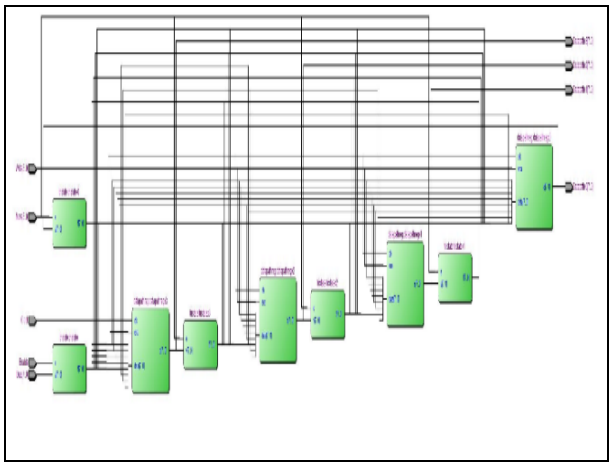

Fig. 6. Register Transfer Level for Tristate Bus Module

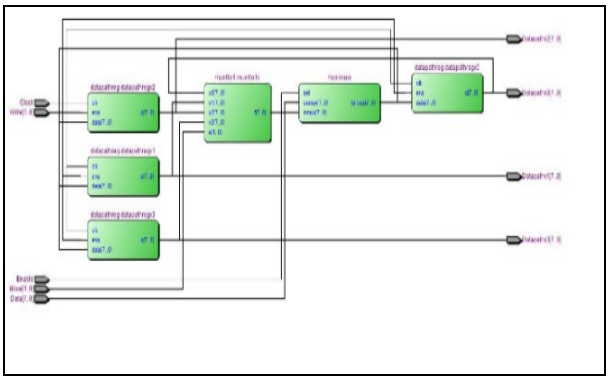

Fig. 7. Register Transfer Level for Multiplexer Bus Module

"Fig. 6", and "Fig. 7", show that the proposed multiplexer-based bus module has less register transfer level than the tristate-based bus, which means that it has less FPGA chip resources to implement the bus system.

B. Clock To Output Times

Clock to output times is time analyzer used by Time Quest application under Intel-FPGA Quartus II software tool. The minimum time required to obtain a valid output at an output pin that is fed by a register after a clock signal transition on an input pin that clocks the register. This time always represents an external pin-to-pin delay. In the Quartus® Prime software, you can specify the required minimum time to clock output for the entire project and/or any clock signal, any register driving an output or bidirectional pin, or any output or bidirectional pin driven by a register. You can also specify a point-to-point minimum requirement between a clock and a register, a clock and an output pin, or a register and an output pin.

The time test for both tristate bus module in "Fig. 8", and multiplexer bus module in "Fig. 9", show that the multiplexer bus module has less time delay than the tristate bus module. In tristate bus module the datapathx0 output has 8.639x10-9 seconds, datapathx1 output has 8.444x10-9 seconds, datapathx2 output has 9.085x10-9 seconds, datapathx3 output has 8.154x10-9 seconds. In multiplexer bus module the datapathx0 output has 7.218x10-9 seconds, datapathx1 output has 7.311x10-9 seconds, datapathx2 output has 7.081x10-9 seconds, datapathx3 output has 6.915x10-9 seconds.

Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	Datapathx0[*]	Clock	8.639	8.617	Rise	Clock
1	Datapathx0[0]	Clock	8.639	8.617	Rise	Clock
2	Datapathx0[1]	Clock	7.576	7.492	Rise	Clock
3	Datapathx0[2]	Clock	8.182	8.165	Rise	Clock
4	Datapathx0[3]	Clock	8.514	8.413	Rise	Clock
5	Datapathx0[4]	Clock	8.629	8.520	Rise	Clock
6	Datapathx0[5]	Clock	7.731	7.628	Rise	Clock
7	Datapathx0[6]	Clock	7.892	7.829	Rise	Clock
8	Datapathx0[7]	Clock	7.689	7.650	Rise	Clock
2	Datapathx1[*]	Clock	8.444	8.412	Rise	Clock
1	Datapathx1[0]	Clock	8.444	8.412	Rise	Clock
2	Datapathx1[1]	Clock	7.030	6.936	Rise	Clock
3	Datapathx1[2]	Clock	7.077	6.988	Rise	Clock
4	Datapathx1[3]	Clock	7.061	6.978	Rise	Clock
5	Datapathx1[4]	Clock	6.878	6.765	Rise	Clock
6	Datapathx1[5]	Clock	7.038	6.953	Rise	Clock
7	Datapathx1[6]	Clock	8.342	8.275	Rise	Clock
8	Datapathx1[7]	Clock	7.259	7.200	Rise	Clock
3	Datapathx2[*]	Clock	9.085	9.078	Rise	Clock
1	Datapathx2[0]	Clock	7.582	7.489	Rise	Clock
2	Datapathx2[1]	Clock	7.501	7.395	Rise	Clock
3	Datapathx2[2]	Clock	7.243	7.178	Rise	Clock
4	Datapathx2[3]	Clock	9.085	9.078	Rise	Clock
5	Datapathx2[4]	Clock	7.909	7.843	Rise	Clock
6	Datapathx2[5]	Clock	7.908	7.849	Rise	Clock
7	Datapathx2[6]	Clock	7.586	7.498	Rise	Clock
8	Datapathx2[7]	Clock	7.980	7.860	Rise	Clock
4	Datapathx3[*]	Clock	8.154	8.082	Rise	Clock
1	Datapathx3[0]	Clock	7.328	7.226	Rise	Clock
2	Datapathx3[1]	Clock	7.693	7.589	Rise	Clock
3	Datapathx3[2]	Clock	7.005	6.917	Rise	Clock
4	Datapathx3[3]	Clock	7.001	6.916	Rise	Clock
5	Datapathx3[4]	Clock	7.055	6.955	Rise	Clock
6	Datapathx3[5]	Clock	7.761	7.726	Rise	Clock
7	Datapathx3[6]	Clock	7.017	6.923	Rise	Clock
8	Datapathx3[7]	Clock	8.154	8.082	Rise	Clock

Fig. 8. Tristate Bus Module Clock to OutputTimes

Minimum Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	Datapathx0[*]	Clock	7.218	7.097	Rise	Clock
1	Datapathx0[0]	Clock	7.786	7.710	Rise	Clock
2	Datapathx0[1]	Clock	7.427	7.354	Rise	Clock
3	Datapathx0[2]	Clock	7.374	7.291	Rise	Clock
4	Datapathx0[3]	Clock	7.794	7.683	Rise	Clock
5	Datapathx0[4]	Clock	7.373	7.255	Rise	Clock
6	Datapathx0[5]	Clock	7.218	7.097	Rise	Clock
7	Datapathx0[6]	Clock	8.889	8.864	Rise	Clock
8	Datapathx0[7]	Clock	8.034	7.915	Rise	Clock
2	Datapathx1[*]	Clock	7.311	7.207	Rise	Clock
1	Datapathx1[0]	Clock	8.711	8.657	Rise	Clock
2	Datapathx1[1]	Clock	7.311	7.207	Rise	Clock
3	Datapathx1[2]	Clock	8.136	8.097	Rise	Clock
4	Datapathx1[3]	Clock	7.827	7.869	Rise	Clock
5	Datapathx1[4]	Clock	7.454	7.323	Rise	Clock
6	Datapathx1[5]	Clock	7.911	7.798	Rise	Clock
7	Datapathx1[6]	Clock	7.966	7.938	Rise	Clock
8	Datapathx1[7]	Clock	7.347	7.251	Rise	Clock
3	Datapathx2[*]	Clock	7.081	6.993	Rise	Clock
1	Datapathx2[0]	Clock	7.656	7.570	Rise	Clock
2	Datapathx2[1]	Clock	7.847	7.732	Rise	Clock
3	Datapathx2[2]	Clock	7.081	6.993	Rise	Clock
4	Datapathx2[3]	Clock	7.230	7.096	Rise	Clock
5	Datapathx2[4]	Clock	7.587	7.455	Rise	Clock
6	Datapathx2[5]	Clock	7.253	7.164	Rise	Clock
7	Datapathx2[6]	Clock	7.214	7.091	Rise	Clock
8	Datapathx2[7]	Clock	7.263	7.172	Rise	Clock
4	Datapathx3[*]	Clock	6.915	6.829	Rise	Clock
1	Datapathx3[0]	Clock	7.256	7.192	Rise	Clock
2	Datapathx3[1]	Clock	6.915	6.829	Rise	Clock
3	Datapathx3[2]	Clock	6.995	6.869	Rise	Clock
4	Datapathx3[3]	Clock	6.938	6.852	Rise	Clock
5	Datapathx3[4]	Clock	7.743	7.663	Rise	Clock
6	Datapathx3[5]	Clock	7.117	7.032	Rise	Clock
7	Datapathx3[6]	Clock	8.899	8.987	Rise	Clock
8	Datapathx3[7]	Clock	7.202	7.095	Rise	Clock

Fig. 9. Multiplexer Bus Module Clock to Output Times

C. *Power Play Power Analyzer*

Power play power analyzer used by Intel-FPGA Quartus II software tool. The power analyzer used to measure the thermal power dissipation for both tristate bus module in "Fig. 10", and multiplexer bus module in "Fig. 11", Processor power dissipation or processing unit power dissipation is the process in which computer processors consume electrical energy, and dissipate this energy in the form of heat due to the resistance in the electronic circuits [11]. The result shows that the multiplexer bus module has less thermal power dissipation than the tristate bus module.

PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Tue Feb 15 10:26:38 2022
Quartus II 64-Bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	MuxBus
Top-level Entity Name	MuxBus
Family	Cyclone IV GX
Device	EP4CGX15BF14A7
Power Models	Final
Total Thermal Power Dissipation	68.22 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	58.84 mW
I/O Thermal Power Dissipation	9.39 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig. 10. Tristate Bus Module Power Play Power Analyzer

The multiplexer bus module has total of 68.22x10-3 watts power dissipation, while the tristate bus module has total of 68.23x10-3 watts power dissipation.

PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Tue Feb 15 10:12:58 2022
Quartus II 64-Bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	TristateBus
Top-level Entity Name	TristateBus
Family	Cyclone IV GX
Device	EP4CGX15BF14A7
Power Models	Final
Total Thermal Power Dissipation	68.23 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	58.84 mW
I/O Thermal Power Dissipation	9.39 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig. 11. Multiplexer Bus Module Power Play Power Analyzer

IV. FPGA PLACE AND ROUTE

After verifying both tri-state and multiplexer-based bus modules individually, the submodules are instantiated, simulated and verified. The system was implemented and tested using Cyclone EP1C6Q240C8 FPGA evaluation platform, that designed specially to test the functionality of the bus system in hardware, "Fig. 12", For further advanced FPGA development tools, complete electronic design automation (EDA) Printed Circuit Board system is designed to test and verify the bus system.

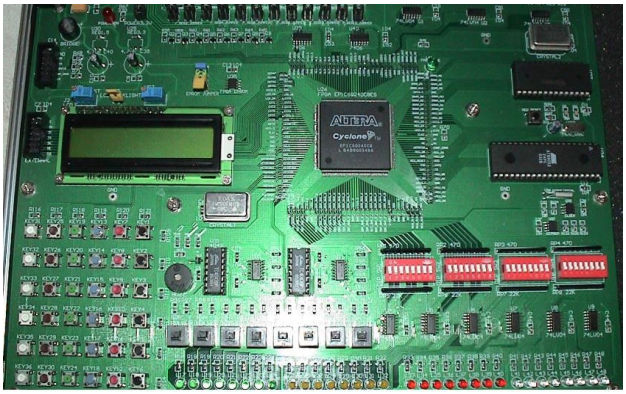

Fig. 12. Printed Circuit Board with Cyclone FPGA

V. CONCLUSION

The key contribution of this paper is to design, simulate, and implement efficient multiplexer- based bus structure that can be used on FPGA design that has limited tristate bus resources. The waveforms and results obtained from the simulation and testing processes illustrate that, compared to the tristate-based bus, the proposed multiplexer-based bus structure uses less hardware resources, has less time delay, and less thermal power dissipation. In future the study can extend by implement microprocessor system that use the multiplexer- based bus structure instead of tristate-based bus structure to communicate between its internal registers, arithmetic logic unit, and control unit.

REFERENCES

[1] V.E. Benes, "Mathematical Theory of Connecting Networks and Telephone Traffic," Academic Press, 1965.

[2] Foty, "MOSFET Modeling with Spice," Prentice Hall, 1996.

[3] Johnson and Graham, "High Speed Digital Design: a Handbook of Black Magic," Prentice Hall, 1993.

[4] Mohamed Eljhani and Veton Keposka, "Reduced Instruction Set Computer Design on FPGA", 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA, 25-27 May 2021, Tripoli- Libya.

[5] Nikil Dutt, Kaustav Banerjee, Luca Benini, Kanishka Lahiri, Sudeep Pasricha, "Tutorial 5: SoC Communication Architectures: Technology, Current

Practice, Research, and Trends", vlsid, pp.8, 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07), 2007.

[6] Altera Corporation, “Comparing IP Integration Approaches for FPGA Implementation”.

[7] The IEEE Standard Hardware Description Language based on the Verilog Hardware Description Language (IEEE Std 1364-2001).

[8] Micheal D. Ciletti, “Advanced Digital Design with the Verilog HDL “Prentice Hall ,2004.

[9] William Stallings, “Computer Organization and Architecture, Designing for Performance”, Prentice Hall, 2001.

[10] Esra Tellisi, Jumanah Mansur, and Mohamed Eljhani, “An Alternative Microprocessor Bus Structure Design on FPGA”, International Science and Technology Journal 2022.

[11] https://en.wikipedia.org/wiki/Processor_power_dissipation.