DEVELOPMENT AND COMPARATIVE EVALUATION OF INNOVATIVE BUS-INTERCONNECT METHODOLOGY FOR PROGRAMMABLE CHIP-BASED SYSTEMS

Mohamed Muftah Eljhani

Department of Computer Engineering, Faculty of Engineering, University of Tripoli

Email: M.Eljhani@uot.edu.ly

Received 8 June 2023, revised 11 October 2023, accepted 2 January 2024

الملخص

أدى التعقيد المتزايد لتصميمات انظمة الشرائح القابلة للبرمجة System-on-Programmable-Chip (SoPC) إلى تحديات كبيرة في إنتاجية التصميم. حيث يؤثر إنشاء وتصميم واجهة SoPC بشكل كبير على أداء النظام واستهلاك الطاقة. تقدم هذه الورقة لمحة عامة عن ناقلات النظام الرقمي السائدة، وتستكشف العديد من هياكل ناقلات، وتقدم تركيبة ناقلات جديدة. بالإضافة إلى ذلك، تقدم وحدة تحكم في الناقلات تسهيل معاملات وحدة مسار البيانات. يعد تنفيذ هندسة الناقلات القائمة على ثلاث محاورTri-State Buses مفيدًا للتصميمات الكبيرة والمتكاثفة. ومع ذلك، نظرًا للقيود المفروضة على رقائق مصفوفة البوابات القابلة للبرمجة FPGA فيما يتعلق بثلاثية الناقلات الكبيرة، تم اقتراح هيكل الناقلات الجديد متعدد الإرسال ووحدة تحكم. يتضمن هذا البحث تصميم وتنفيذ ومحاكاة الوحدات الأساسية لنظام الناقلات القائم على الإرسال المتعدد باستخدام لغة وصف الأجهزة Verilog HDL. وإجراء المقارنات مع نظام الناقلات الثلاثي. أن الناقل المقترح في هذا البحث القائم على الإرسال المتعدد يحقق سرعة أعلى، وتبديد طاقة أقل، ومرونة معززة في التوقيت، وقدرات اختبار محسنة. الناقلات المقترحة مناسبة لـ FPGA وغيرها من الرقائق القابلة للبرمجة التي تتطلب ناقلات عالية السرعة ومنخفضة الطاقة. في تصميم SoPC، يتم تفضيل الناقلات القائمة على الإرسال المتعدد نظرًا لتسهيل استعمال الناقلات الثلاثية. علاوة على ذلك، تفضل الدوائر المتكاملة الخاصة بالتطبيقات الناقلات الداخلية القائمة على الإرسال المتعدد بسبب تحديات التوقيت واستهلاك الطاقة المرتبطة الناقلات القائمة على ثلاث حالات.

ABSTRACT

The increasing complexity of System-On-Programmable-Chip (SoPC) designs has led to significant challenges in design productivity. The instantiation and interface design of SoPCs significantly impact system performance and power consumption. This paper provides an overview of prevalent digital system buses, explores various bus architectures, and introduces a novel bus architecture. Additionally, it introduces a bus controller facilitating data path module transactions. Implementing tri-state-based bus architecture is beneficial for extensive designs with numerous blocks. However, due to limitations in Field Programmable Gate Array (FPGA) chips regarding tristate drivers for large buses, a new multiplexer-based bus structure and controller are proposed. This research involves designing, implementing, and simulating fundamental modules of the multiplexer-based bus system using Verilog hardware description language. Subsequently, comparisons with the tri-state-based bus system demonstrate that the multiplexer-based bus achieves higher speed, lower power dissipation, enhanced flexibility in timing, and improved testing capabilities. The proposed bus architecture is suitable for FPGAs and other programmable chips requiring a high-speed, low-power bus. In SoPC design, multiplexer-based buses are favoured due to easier Intellectual

Property integration compared to tri-state-based buses. Moreover, application-specific integrated circuits prefer internal multiplexer-based buses due to the timing and power consumption challenges associated with tri-state-based buses caused by capacitive loads on their nodes.

KEYWORDS: Multiplexer-Based Bus Structure; System-On-Programmable-Chip; FPGA Design; Field Programmable Gate Arrays; Verilog Hardware Description Language.

I. INTRODUCTION

Interconnection networks have been an essential component of electrical engineering from the early days of computers and telephones [1]. Because of the driving characteristics of metal oxide semiconductor (MOS) transistors, this has become even more important in the era of very large-scale integration (VLSI) circuitry [2]. Buses are the simplest type of connector, but because the power and space needed to operate them at maximum speed increase exponentially with the bus's capacitance, they are a poor choice in terms of density or power [3]. A bus is a group of digital communication lines used inside computer systems or within a single integrated circuit (IC) that facilitates data interchange and transfer between two or more modules in the system. Bus systems, which serve as an essential means of data transfer, can be tailored for particular uses, such as an I/O bus or memory bus. Essentially, the central processing unit (CPU) various components are connected by an internal system bus. The data bus of a reduced instruction set computer (RISC) has eight bits. Every instruction is fetched twice by the system, which is controlled by the state. When the state is zero, the higher eight bits of the instruction are sent to the responsive register, and the state is set to one. Since the status is 1, the bottom 8 bits of the instruction are delivered to the responsive register for the subsequent operation. Signal reset initiates the reset process. All registers are initialized to 0 and the system ends the current operation when reset is set to 1. The system remains in the reset state as long as the reset is high. The address is set to 0 and the data bus is maintained in a high-impedance state [4]. Modern computer buses can use bit serial and parallel connections, but the early ones were physically parallel electrical cables with many connections. Moreover, buses can employ point-to-point or multipoint approaches to concurrently connect two distinct components. System speed and power dissipation are significantly impacted by system-on-chip bus architectures. In order to achieve the desired design goals, system designers and the research community have concentrated on the problem of investigating, assessing, and creating SoPC communication architectures [5]. There are numerous alternatives to buses that have all been effectively employed in a variety of ASICs, FPGAs, chips, boards, and computers. Similar to buses, these solutions are not a magic bullet for all connectivity problems. By eschewing the set route and schedule of a typical bus, one might explore new design possibilities and infuse some elegance and originality back into an otherwise uninteresting endeavour [6]. Typically, the Electronic Design Automation (EDA) design flow starts with the schematic design entry in Verilog/VHDL hardware description language [7], proceeds through place and route tools and synthesis, and ends with programming the FPGA. Using the Verilog hardware description language, the proposed multiplexer bus system is created, simulated, and compared with the tristate system bus. It is then implemented on the Altera FPGA development board, Cyclone IV GX FPGA [8,9]. In earlier research, we used multiplexer and tri-state buses to design and simulate microprocessor system buses.

We discovered that while tri-state-based bus implementation is helpful for large design applications with lots of design blocks, it can also make testing and synchronization more difficult. Large buses cannot be mounted on FPGA chips due to a lack of tristate drivers. As an alternative, bus structures based on multiplexers can be used by designers. Verilog hardware description language (HDL) is used in the design, implementation, and simulation of the fundamental modules of the proposed microprocessor bus system [10].

This document is divided into four sections. Section I has the introduction and literature review of the paper. Section II provides design methodology of the general bus structure. The design process and technique for the proposed multiplexer bus structure (parts A and B, respectively) and the tristate bus structure are shown together with simulation waveforms and design block diagrams. Section III contains the simulation and findings. Sections IV contain the paper's conclusion. Using a single bus structure is the simplest way to connect registers in a system. Since the bus may only be used for one transfer at a time, only two registers can actively use it at any given moment. All datapath registers are connected directly to the bus. Data is transferred between datapath registers via the bus, a subsystem in the design architecture. The design is structured so that each unit may process one whole word (8 bits) of data at a time in order to achieve high operation speed. Every bit of a word of data is sent in parallel, concurrently, via eight wires (one bit per wire), which act as a connecting link between data path registers, when the word is moved between registers. A bus is a common group of wires. It is crucial to make sure that only one register functions as a source register at a time and that other registers don't obstruct the process when data is sent from numerous source registers to multiple destination registers via a shared bus.

II. METHODOLOGY

The common tristate-based bus and the suggested multiplexer-based bus systems were created, put into practice, and simulated for comparison's sake. Modelsim and Quartus II Intel FPGA software tools are used to simulate and verify the design, which is created using the Verilog hardware description language. The data path module and the control unit module are the two primary components that make up any system. The datapath module (datapathx0, datapathx1, datapathx2, datapathx3) is used to store, manipulate, and move data between the system's data path registers. Building blocks for multiplexers, tristate buffers, and registers make up datapath modules. The data path module's operations are managed by the control unit module. The primary function of the state machine module, which serves as the brain of the design, is to provide a sequence of control signals that are used to manage and run the datapath module.

Tristate-based Bus System

Two primary elements comprise the tristate-based bus architecture: a control unit module and a data path module. There are nine sub-modules in the data path module. Datapathx0 through Datapathx3 are the four 8-bit registers in the system. The bus structure is implemented by connecting these registers through the use of tri-state drivers, as illustrated in Figure (1). Each register's data outputs (q) are linked to tri-state drivers. The drivers attach the contents of the respective register to the bus wires when they are activated by their enable signals. On the subsequent active edge of the clock, the register's contents will be altered if the enable input is set to 1.

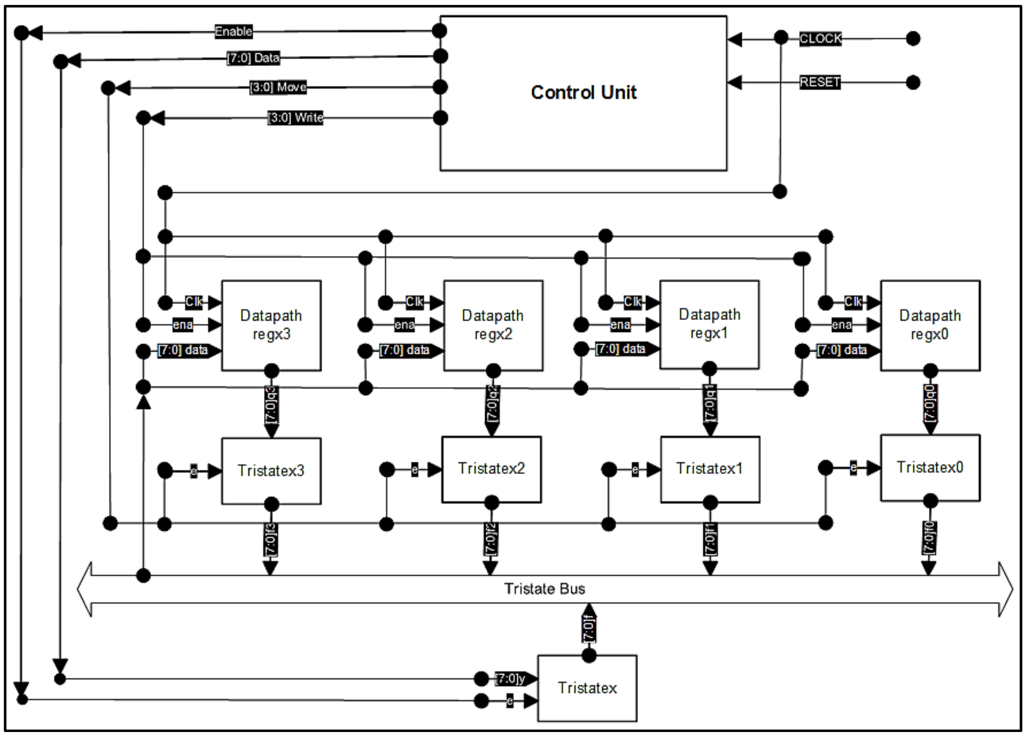


Figure 1: Tristate Bus Structure.

Each register has an enable input designated with the letter ena, which stands for enable. The signal represented by [3;0] governs the ena input for registers. Write while the corresponding tri-state driver's control signal is called [3:0]. Proceed. The control unit module generates these signals. There is an additional module block attached to the bus in addition to the four registers. Using the control input signal e, which is produced by the control unit module named Enable, the circuit diagram illustrates how eight bits of data from an external source that is connected to the same bus are used. Making sure that only one circuit block is attempting to send data onto the bus wires at any given time is crucial. Datapathx0, Datapathx1, Datapathx2, and Datapathx3 are tri-state drivers that enable signals. The control unit has to make sure that at any one time, only one of these drivers is asserted. The signals [3:0] Write, which regulate when data is put into each register, are likewise generated by the control unit. The control unit generally carries out a variety of tasks, including data loading into resisters and data transfer from one register to another. Clock input: The same clock signal that drives the four data path registers also synchronizes the control circuit. The simulated waveforms of the tristate bus module are displayed in Figure (2).

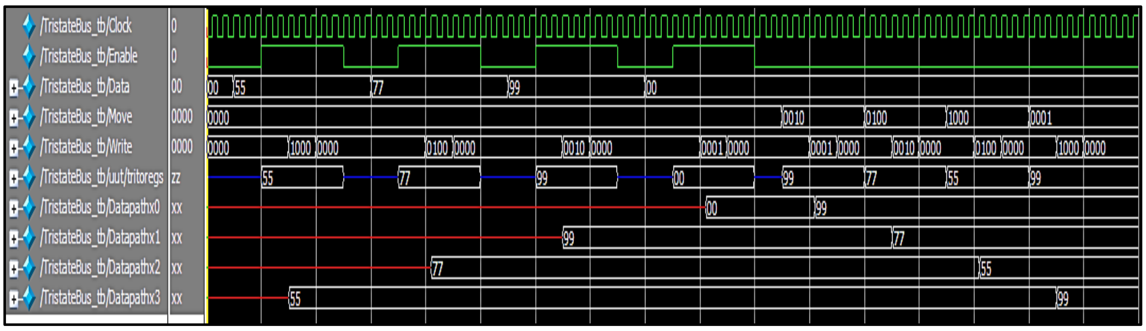


Figure 2 : Tristate Bus Simulation Waveforms.

Data supplied from the outside can be loaded from the bus into data path registers using the control signals designated [3:0] Input. The data is loaded into datapathx3 when [3:0] Write = 1000; datapathx2 when [3:0] Write = 0100; datapathx1 when [3:0] Write = 0010; and datapathx0 when [3:0] Write = 0001, respectively; etc. Such actions are carried out by the finite state machine (FSM) that implements the control. The load values for the data paths regx3 (55 hex), regx2 (77 hex), regx1 (99 hex), and regx0 (00 hex) are as follows. Data path registers filled with data, data path regx1 contents moved to data path regx0, data path regx2 to data path regx1, data path regx3 to data path regx2, and data path regx0 to data path regx3. Overall, data path registers loaded with data.

Multiplexer-based Bus System

The data path module and control unit module are the two primary modules that make up the suggested multiplexer-based bus design. There are six smaller modules that make up the data path module. Datapathx0 through Datapathx3 are the four 8-bit registers in the system. In Figure (3), the bus topology is implemented by connecting these registers via multiplexers rather than tristate buffers. Every register's data output (q) is linked to a 4to1 multiplexer. The multiplexer loads the contents of the matching register onto a 2to1 multiplexer that is attached to the bus when it detects their [1:0] e signals. Each data path register has an enable input designated by the letter ena, which stands for enable. [3;0] Write is the signal that governs the ena input for registers, while [1:0] Move is the signal that governs the related multiplexer output. The control unit module is the source of these signals.

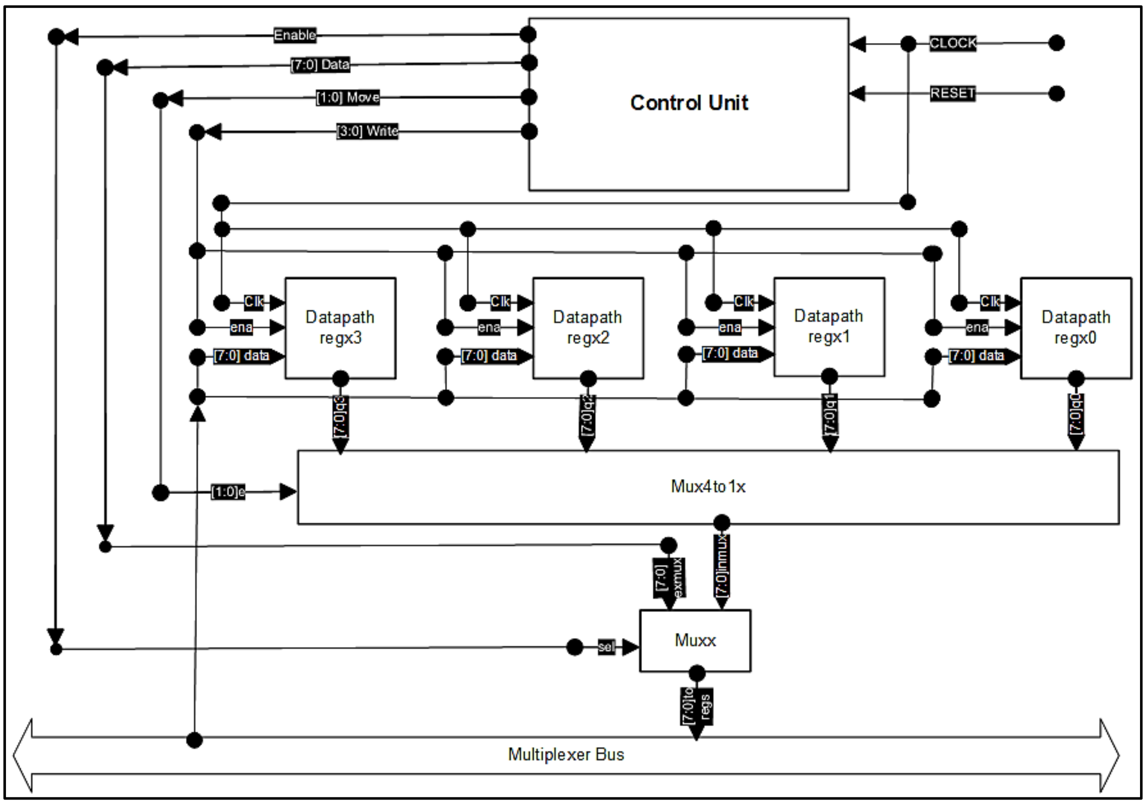


Figure 3: Multiplexer Bus Structure.

Apart from the four registers, there's an additional module block attached to the bus. The circuit diagram illustrates how to use the control input signal Enable, which is coupled to the sel signal of the 2 to1 mux produced by the control unit module, to place eight bits of data from an external source on the same bus. It is crucial to make sure that, at any given moment, only one circuit block is attempting to load data onto the bus wires. Only one of the enable signals—datapathx0, datapathx1, datapathx2, or datapathx3—must be asserted at any one time, according to the control unit. Additionally, the control unit generates the [3:0] Write signals, which govern when information is fed into each register. The clock signal used to synchronize the four data path registers also synchronizes the control circuit. The multiplexer bus module's simulated waveforms are displayed in Figure (4).

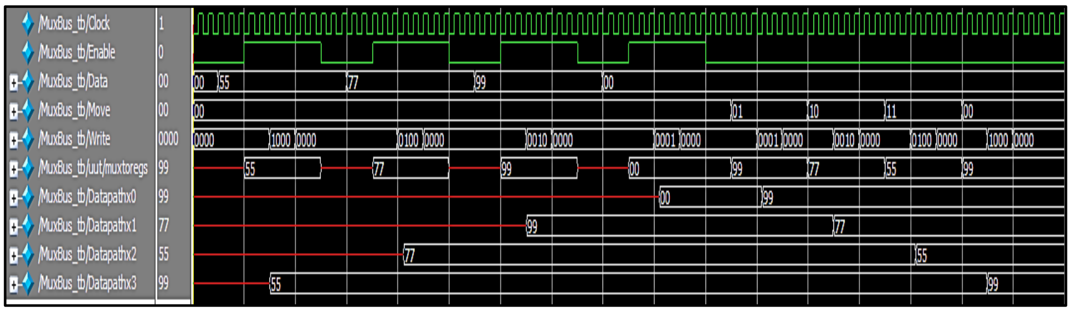


Figure 4: Multiplexer Bus Simulation Waveforms.

The externally supplied data can be loaded from the bus into data path registers by using the control signals designated [3:0] Write. The data is loaded into datapathx3 when [3:0] Write = 1000; datapathx2 when [3:0] Write = 0100; datapathx1 when [3:0] Write = 0010; and datapathx0 when [3:0] Write = 0001. These operations are carried out by the finite state machine (FSM) that implements the control. To compare the performance of the proposed multiplexer-based bus with the tristate-based bus, we initialized data path registers with the same values as in the tristate bus. A total of 00 hex was loaded into data path regx0, 99 hex into data path regx1, 77 hex into data path regx2, and 55 hex into data path regx3. Data path registers were finally filled with data, and data path regx1's contents were transferred to data path regx0, data path regx2 was sent to data path regx1, data path regx3 was transferred to data path regx2, and data path regx0 was transferred to data path regx3.

Control Unit

A CPU in a computer includes a bus control unit. The control unit in a computer typically steps through the instruction cycle sequentially. This includes retrieving the command, retrieving the operands, decoding the command, arranging input/output processes, carrying out the command, and finally writing the outcomes back to memory. The control unit modifies its behaviour in order to carry out the subsequent instruction correctly when it is inserted. Thus, the control unit, which in turn controls the computer, is directly controlled by the instructions.

Bus Control Design

The system's control was built to input data, transfer it to the data path for upload into the register, and switch between registers. In order to regulate the input based on the control state, the control is linked to the data path. The necessary signals, which specify when data is fed into each register, are also generated by the control unit. The control unit generally performs a variety of tasks, including data loading into resisters and data transfer from one register to another. An equal clock signal that governs four registers serves as the clock input, synchronizing the control circuit. The data path is divided into three modules: the first module is for the 4-bit register, the second is for the 4-bit MUX

4 to 1, the third is for the MUX 2-1. The control module comprises four outputs (data, write, move, and enable) and two inputs (clock and reset).

Finite State Machine of Bus Control

The control was created using a state machine methodology, the control relies on a predetermined set of states to function. Thirteen distinct states were identified and created. Figure (5), shows the subsequent possible states. These states specify how the control will act and function in various conditions and inputs.

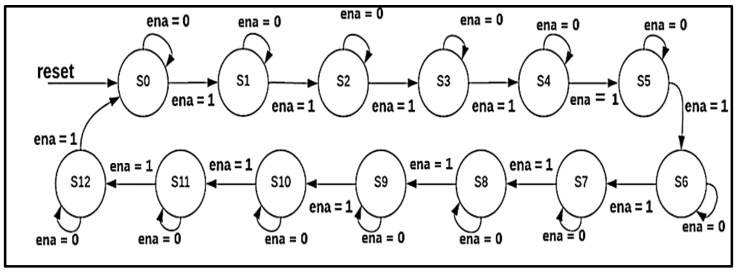


Figure 5: Bus Control FSM State Diagram.

III. SIMULATION AND RESULTS

The system is instantiated, simulated, and verified as a top-level module after creating and simulating each multiplexer bus sub-module separately. The functionality, speed, and power consumption of the suggested multiplexer bus system were then assessed in comparison to the tristate bus system, which is specifically designed and constructed for this purpose. The tristate bus and multiplexer bus simulation waveforms are displayed in Figure (6).

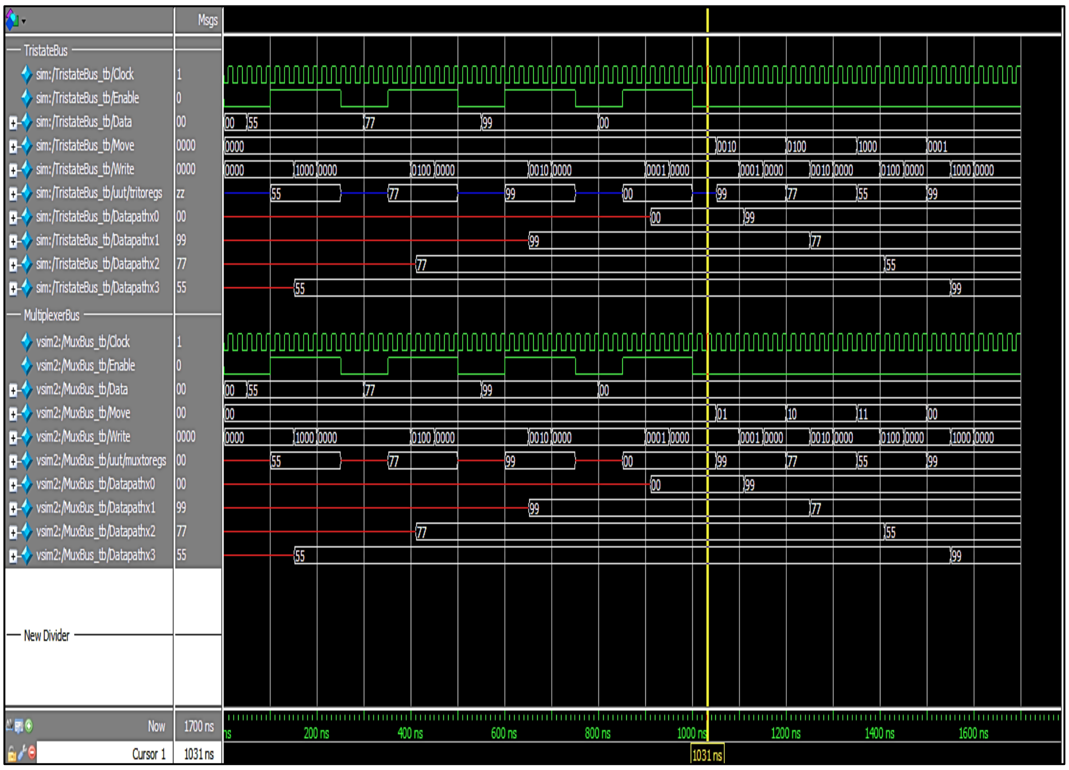

Msgs
TristateBus
sim:/TristateBus_tb/Clock
sim:/TristateBus_tb/Enable
sim:/TristateBus_tb/Data
sim:/TristateBus_tb/Move
sim:/TristateBus_tb/Write
sim:/TristateBus_tb/uut/tritoregs
sim:/TristateBus_tb/Datapathx0
sim:/TristateBus_tb/Datapathx1
sim:/TristateBus_tb/Datapathx2
sim:/TristateBus_tb/Datapathx3
MultiplexerBus
vsim2:/MuxBus_tb/Clock
vsim2:/MuxBus_tb/Enable
vsim2:/MuxBus_tb/Data
vsim2:/MuxBus_tb/Move
vsim2:/MuxBus_tb/Write
vsim2:/MuxBus_tb/uut/muxtoregs
vsim2:/MuxBus_tb/Datapathx0
vsim2:/MuxBus_tb/Datapathx1
vsim2:/MuxBus_tb/Datapathx2
vsim2:/MuxBus_tb/Datapathx3
New Divider
Now 1700 ns
Cursor 1 1031 ns
200 ns
400 ns
600 ns
800 ns
1000 ns
1200 ns
1400 ns
1600 ns
1031 ns

Figure 6: Simulation Waveforms of Tristate Bus and Multiplexer Bus Systems.

The simulation waveforms that are produced demonstrate that both systems are using the same data set and operating in the same way, with the exception that the second module connected data path registers using a multiplexer bus, whereas the first module used a tristate bus.

Register Transfer level

In the Verilog hardware description language, register-transfer-level (RTL) is used to convert lower-level representations of tristate bus and multiplexer bus modules into high-level representations. The register transfer level (RTL) model for the multiplexer bus is depicted in Figure (8), whereas the RTL model for the tristate bus module is shown in Figure (7). Data transformation is demonstrated in Figures (7,8) when data is transferred from one register data path to another in the architecture. The combinational logic component, which is present between every register, loads and transforms the data into the registers' data channel.

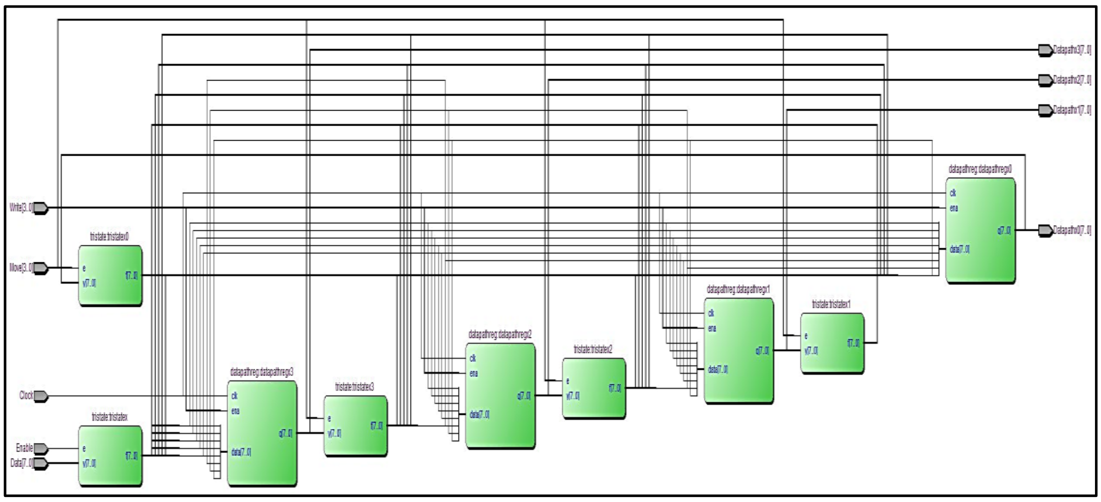


Figure 7: Register Transfer Level for Tristate Bus Module.

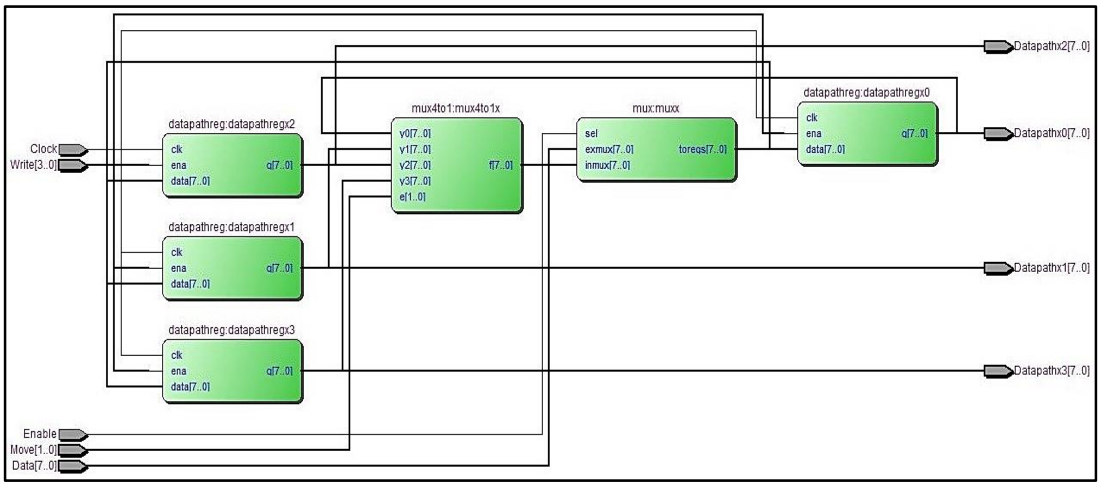
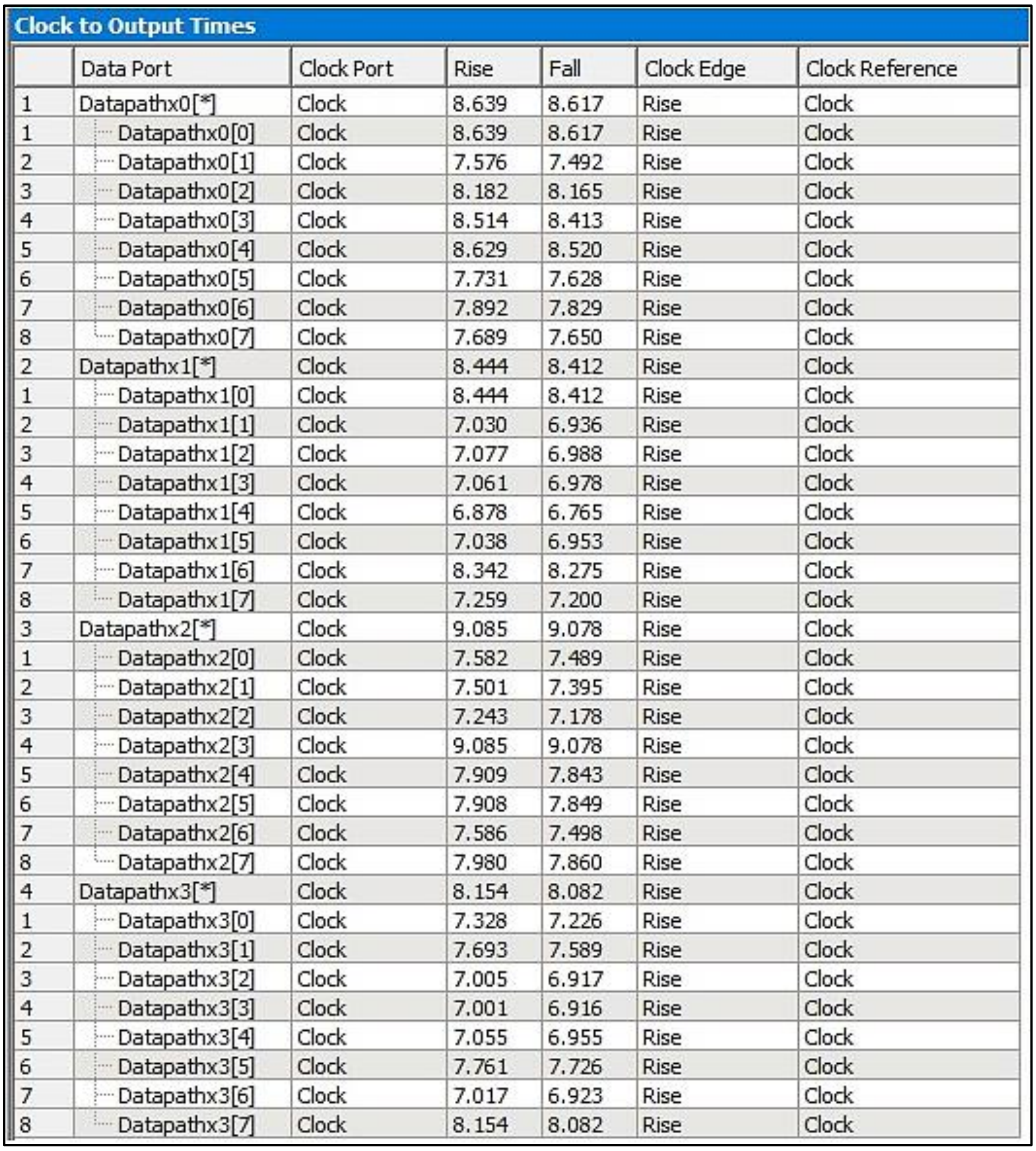


Figure 8: Register Transfer Level for Multiplexer Bus Module.

The suggested multiplexer-based bus module has less register transfer level than the tristate-based bus, as shown in Figures (7,8), indicating that it requires fewer FPGA chip resources to implement the bus system.

Clock to Output Times

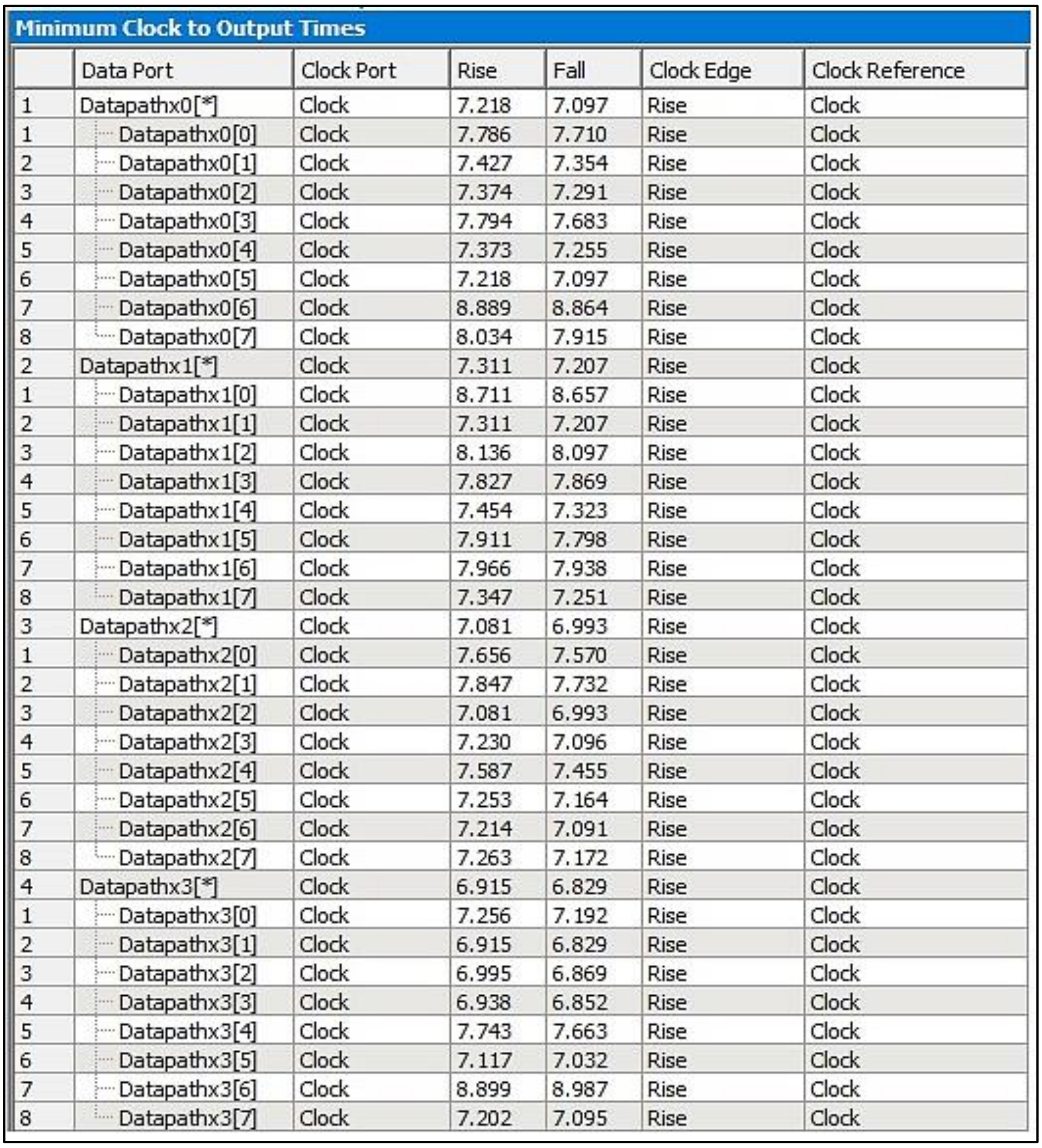
The Intel-FPGA Quartus II software tool's Time Quest application uses a timing analyser called clock to report times. The least amount of time needed following a clock signal transition on an input pin that clocks the register to produce a valid output at an output pin that is fed by the register. There is always an external pin-to-pin delay represented by this time.


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

Figure 9: Tristate Bus Module Clock to Output Times (in Nano seconds).

We can define the minimum clock output time needed for the project as a whole, as well as any clock signal, any register that drives an output or bidirectional pin, or any output or bidirectional pin that is driven by a register, in the Quartus II program. A point-to-point minimum requirement can also be specified between an output pin and a register, an output pin and a clock, or a clock and an output pin. The multiplexer bus module in Figure (10) has shorter time delay than the tristate bus module, according to the time test conducted for both the tristate bus module in Figure (9) and the multiplexer bus module in Figure (10). The datapathx0, datapathx1 and datapathx2 outputs of the tristate bus module have 8.639×10^{-9} seconds, 8.444×10^{-9} seconds, 9.085×10^{-9} seconds, and 8.154×10^{-9} seconds, respectively. The datapathx0, datapathx1, datapathx2, and datapathx3 outputs in the multiplexer bus module are 7.218x10^{-9}, 7.311x10^{-9}, 7.081x10^{-9}, and 6.915x10^{-9} seconds, respectively.



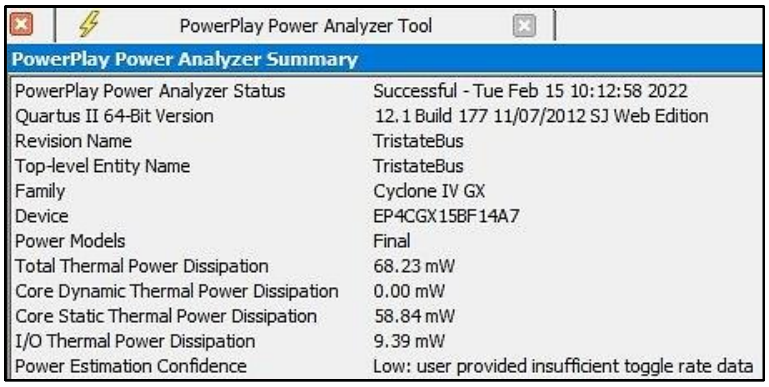
Minimum Clock to Output Times

	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

Figure 10: Multiplexer Bus Module Clock to Output Times (in Nano seconds).

Power Play Power Analyser

The software tool Intel-FPGA Quartus II uses a power play power analyser. The multiplexer bus module in Figure (12), as well as the tristate bus module in Figure (11), had their thermal power dissipation measured by a power analyser. The process by which computer processors use electrical energy and release it as heat as a result of resistance in the electronic circuits is known as processor power dissipation, or processing unit power dissipation [11]. The findings indicate that the tristate bus module dissipates more thermal power than the multiplexer bus module.

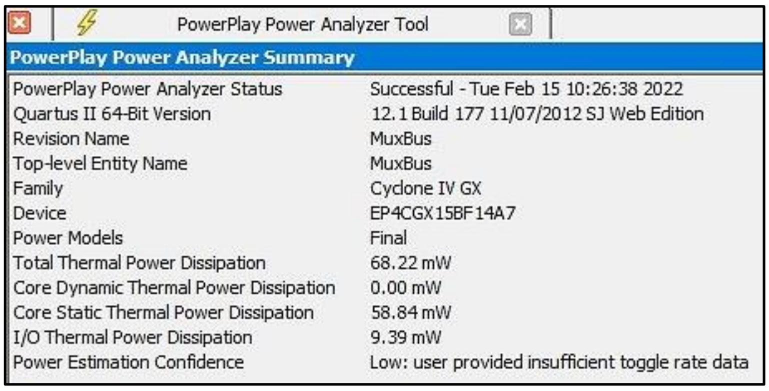


PowerPlay Power Analyzer Summary

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

Figure 11: Tristate Bus Module Power Play Power Analyser.

The combined power dissipation of the multiplexer bus module and the tristate bus module is 68.22x10-3 and 68.23x10-3 watts, respectively.



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

Figure 12: Multiplexer Bus Module Power Play Power Analyzer.

FPGA PLACE AND ROUTE

The submodules are created, simulated, and verified after the individual verification of the multiplexer-based and tri-state bus modules. The Cyclone EP1C6Q240C8 FPGA evaluation platform, which was created specifically to test the bus system's hardware capability, was used to develop and test the system (Figure 13). A full electronic design automation (EDA) printed circuit board system is designed to test and validate the bus system in order to provide more sophisticated FPGA development tools.

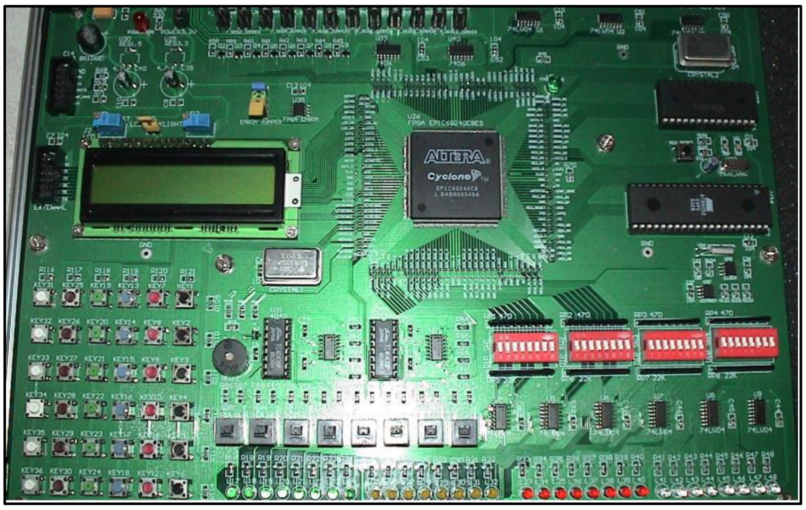


Figure 13: Printed Circuit Board with Cyclone FPGA.

IV. CONCLUSION

The design, simulation, and implementation of an effective multiplexer-based bus structure that can be applied to FPGA designs with constrained tristate bus resources constitute the paper's primary contribution. The suggested multiplexer-based bus architecture utilizes fewer hardware resources, has less time delay, and dissipates less thermal power than the tristate-based bus, as shown by the waveforms and results acquired from the simulation and testing processes. The study can be expanded in the future by developing a microprocessor system that uses buses between its internal registers, arithmetic logic unit, and control unit using a multiplexer-based bus structure rather than a tristate-based bus structure.

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.