**Design and Implementation of Asynchronous FIFO Using Intel FPGA**

[a] Gopika C Rajendran

M.Tech VLSI Design, School of Electronics Engineering

Vellore Institute of Technology, Vellore – 632014, Tamil Nadu, India.

[b] Sreethi S

M.Tech VLSI Design, School of Electronics Engineering

Vellore Institute of Technology, Vellore – 632014, Tamil Nadu, India.

[c] Jakka Naga Vineetha

M.Tech VLSI Design, School of Electronics Engineering

Vellore Institute of Technology, Vellore – 632014, Tamil Nadu, India.

[d] Dr. Sivasankaran K

Professor Grade 1, School of Electronics Engineering

Vellore Institute of Technology, Vellore – 632014, Tamil Nadu, India.

E-mail: [gopika.crajendran2022@vitstudent.ac.in](mailto:gopika.crajendran2022@vitstudent.ac.in) , [sreethi.s2022@vitstudent.ac.in](mailto:sreethi.s2022@vitstudent.ac.in) , [jakkanaga.vineetha2022@vitstudent.ac.in](mailto:jakkanaga.vineetha2022@vitstudent.ac.in) , [ksivasankaran@vit.ac.in](mailto:ksivasankaran@vit.ac.in)

**Abstract**

Asynchronous "First in First Out" (FIFO) is manufactured in a single device and it comes under a special type in buffers which is high speed in nature. These high-speed buffers make up as juncture between many building blocks or components. Asynchronous FIFO uses distinguished read and write pulses to read and write data, in order to ensure compliance with the cross-zone pathway principle, which enables seamless information conversion between two wide range of possible clock zone. It utilizes dual port Static Random Access Memory array with customized read and write ports. It has wide range of applications in the domain of digital signal processing, communication and makes use in high functioning disc controllers, inter-process communication systems for data transfer between two uncorrelated clock domains. In this study, goal is to achieve power optimization by following design considerations such as generation of full and empty flag. The design of asynchronous FIFO is done using Verilog HDL and verifies its functionality through Model Sim. The design is implemented on Intel FPGA board.

**Keywords:** asynchronous FIFO, synchronization, binary to gray code converters, counters

**Introduction**

FIFOs transfer data from clock domain to the same clock domain or any other clock domain. In this work, the asynchronous FIFO is originally conceived and implemented using a gray code standard method to synchronization. A special type of handshake is used in circuits to handle secure communication of information across control modules [[1](file:///C:\\Users\\DELL\\Downloads\\new%20intro.docx" \l "one)]. The "last operation" approach is used to reduce switching activity and, as a consequence, power usage in the circuit [[2](file:///C:\Users\DELL\Downloads\new%20intro.docx#two)]. For every other edge of the read clock, which is the timer that is followed by the write process, data is written. The breadth and depth of the read and write operations, respectively, govern the number of information bits that may be stored [[3](file:///C:\Users\DELL\Downloads\new%20intro.docx#three)]. The essence of a FIFO’s operation is driven by the read enable and writes enable control signals. The main goal of this project is to assess FIFO for usefulness, effectiveness, quality, and scope at the system level [[4](file:///C:\Users\DELL\Downloads\new%20intro.docx#four)].

Gray code pointers that are synchronised into the target clock by an additional bit are used in this study to build a secure FIFO across several clock domains. These pointers give us an idea of the full or empty condition. Verilog HDL is used for sequencing. Quartus Prime Version 13.0 is used for simulation and execution. EP4CE11529C7 from Cyclone IV E Family FPGA Board is used to implement the design.

**Related Works**

There are myriad ways to fabricate a FIFO correctly. FIFOs are for real simulated using a combo of methods, but synthesis is one of the principal courses of action. The only technique for creating, manufacture, and testing secure FIFOs over diverse clock domains uses gray-coded pointers that are contemporary to every new clock sphere before checking if the FIFO is complete or devoid. A trivial approach in digital microchip design is to use queues to process this data from a effectively separate clock zone. The preciseness of regular label printers, which primarily use ARM as the control core, does not meet the needs of freight forwarding and cognitive cache. Thermal print heads are controlled by ARMs and FPGAs in the boundless majority of industrial label printers today. This creates a different clock zone issue. Data transmission technology can ensure that data is not lost. To gain print speed and ensure system-level stability, security, and efficiency for industrial label printers, this paper presents an FPGA-based controller architecture. They are typically used to quickly and easily transfer information between two independent clocks. Every FIFO has a unique design, but this research describes a novel one that employs a signal called the "last operation" to easily produce the FIFO full and FIFO empty flags as well as a grey code integration to minimize potential switching activities, which in turn prevents power consumption in the circuit.

With the use of the SPICE simulator, designs through using CMOS are done to gauge the functioning of these components. A FIFO hardware implementation structure was created using the Self-Timed asynchronous components. Compared to the synchronous model, the performance has been accelerated. System-On-Chips (SOCs) are silicon chips with an explosive growth of processing elements (PEs). The SOC components coupled via contact logic have trouble stimulating communication between synchronous and asynchronous cores. To get around this, either the communication signals are implacably synced to each module's available clock with a tolerable limit of synchronization failure, or the global clocks of each simultaneous module are lagged or extended to prevent communication signals from easily exceeding the local clock's setup and hold time restriction. Request and Acknowledge signals make up a significant portion of asynchronous communication. The address counter and storage array circuit components are connected by these two signals to start mutually exclusive actions.

**Materials and Methods**

**Objective**

The objective of this work is to design the architecture of the asynchronous FIFO (Cypress Asynchronous FIFO [CY7C421]) which is an architecture developed by the Infineon Technologies. The notion to select this architecture is to analyze and optimize power. This objective is verified by using different software.

**Understanding the Specifications**

In this study, the asynchronous FIFO is designed having 512 words deep and 9-bit width data. It incorporates dual port SRAM array which stacks up to 512 memory locations and each memory location stores 9 bits data.

Where, is access time, = 20 ns and is read recovery time, = 10 ns

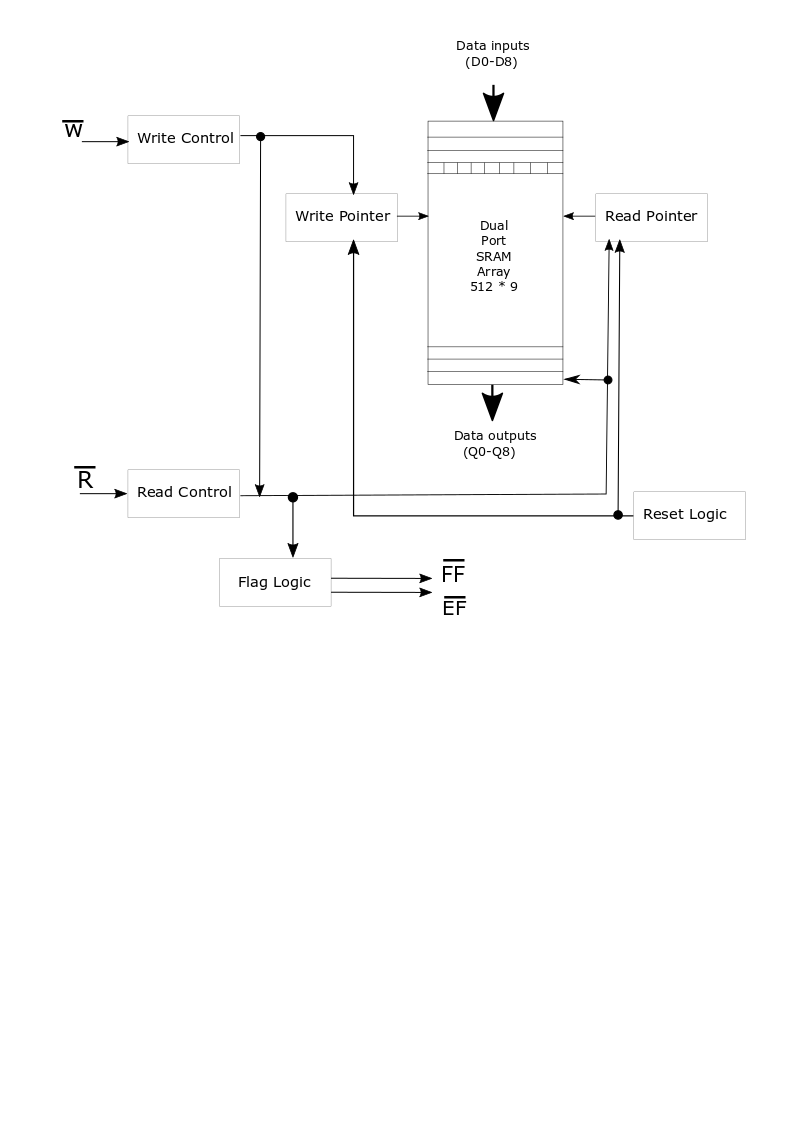
Therefore, maximum read frequency = 33.33 MHz.

Where, is write strobe width, = 15 ns and is write recovery time, = 10 ns

Therefore, maximum write frequency = 40 MHz.

**Block Diagram and Explanation**

The Figure 1 gives insights into the various blocks which we have made use of. The design includes many sub-module blocks and a FIFO memory which has 512 memory address locations and each address storing 9 bits wide data. RAM with two ports offers simultaneous access to read and write data at several locations. This doubles the efficiency. There are two separate pointers for read and write. When write signal is triggered it gives signal to write pointer which points to the adjacent memory location of FIFO. The data is further written in the memory. Similarly, when read signal enables it gives information to read pointer and points to the current location to be read from the FIFO memory and 9 bits wide data is read. The position of these pointers determines the empty and full flag condition. The reset logic will reset both the pointers to first location of memory.

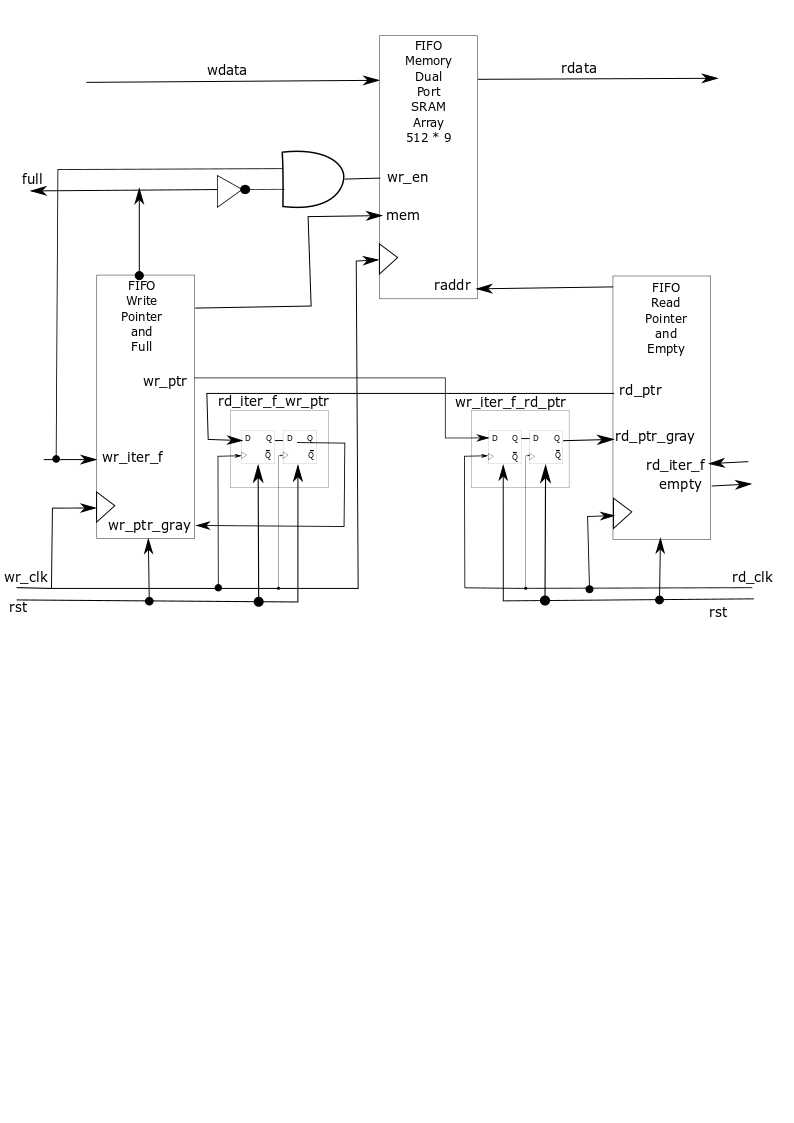


**Figure 1.** Architecture of Asynchronous FIFO

**Experimental Procedure**

The design of several sub-modules is given in figure 2 and its RTL implementation is given in Figure 5. The read operation (Figure 6), write operation (Figure 7), empty condition, full condition, memory (Figure 8), binary to gray converter (Figure 9 and Figure 10), and counter logic are among the sub-blocks. The Verilog Hardware Description Language is being used in the design of each sub-module. It starts with first resetting the fifo, where both the pointers point to the first location of the memory. When the first rising edges is detected, write clock is activated and gives the information to write pointer and data is written to the next location of memory with 9 bits wide data. It must be taken care that data must be stable for setup time before falling edge and should stay valid till hold time. Empty flag is raised to high only when reset is high during positive edge of clock. The pointer always increases by one.

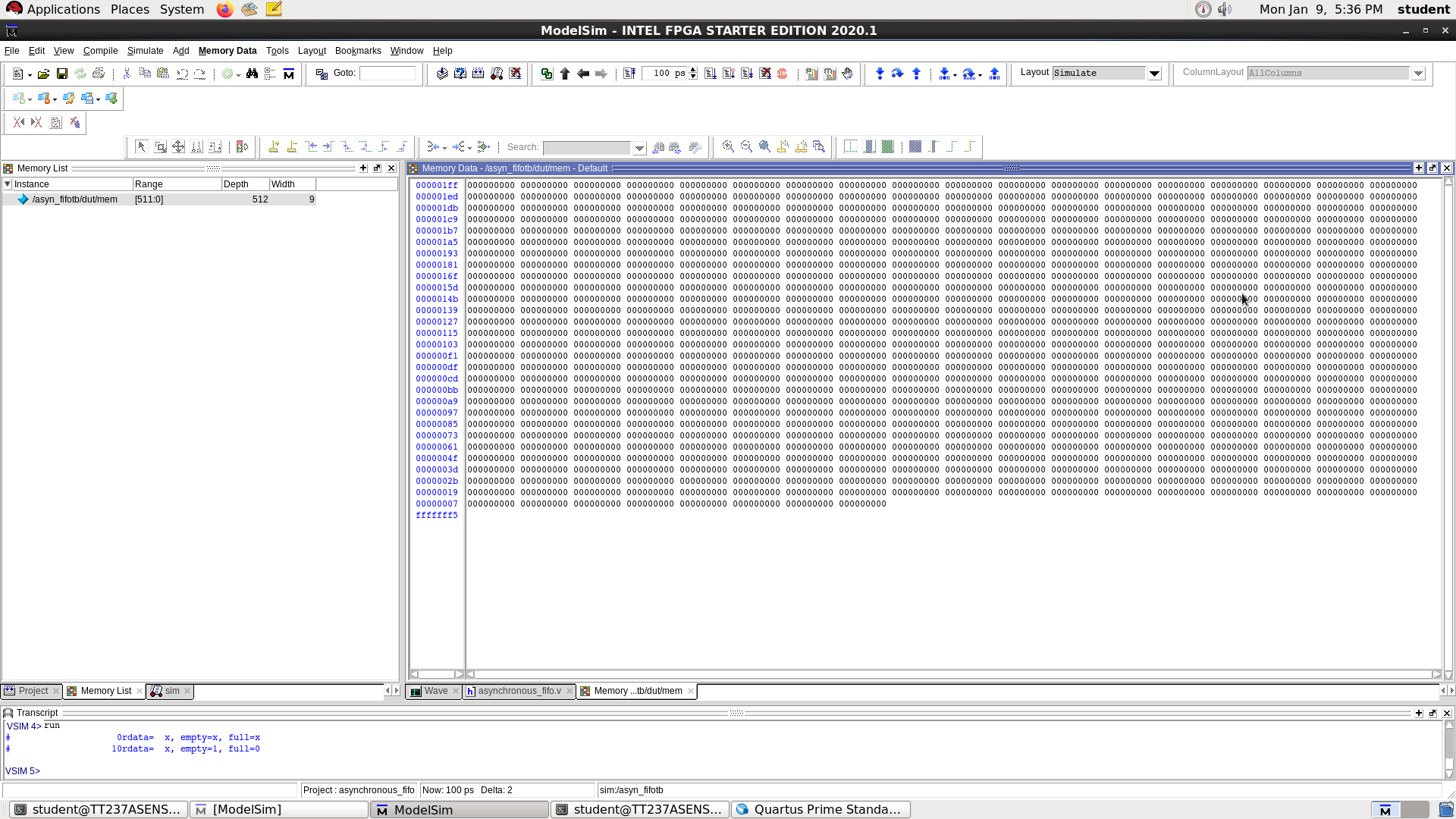
When the next rising edge is detected, read clock is activated and gives the information to read pointer and data is read from the memory location the read pointer is pointing to. Empty flag is raised lower while rest is high during the positive edge of clock. Next important block designed is that of synchronizing circuit. Synchronization is done to eliminate instability at the output. This is done in order to remove different bit change in the output. Binary to grade code converters is implemented to check only one-bit change at the output and thus reducing indeterminate value at the output side. Empty condition is generated in the read clock domain which is synchronized to write pointer and both the pointers are equal including the extra bit. Full condition is generated in the write clock domain which is synchronized to read pointer. Other than the extra bit all the remaining bits are equal.



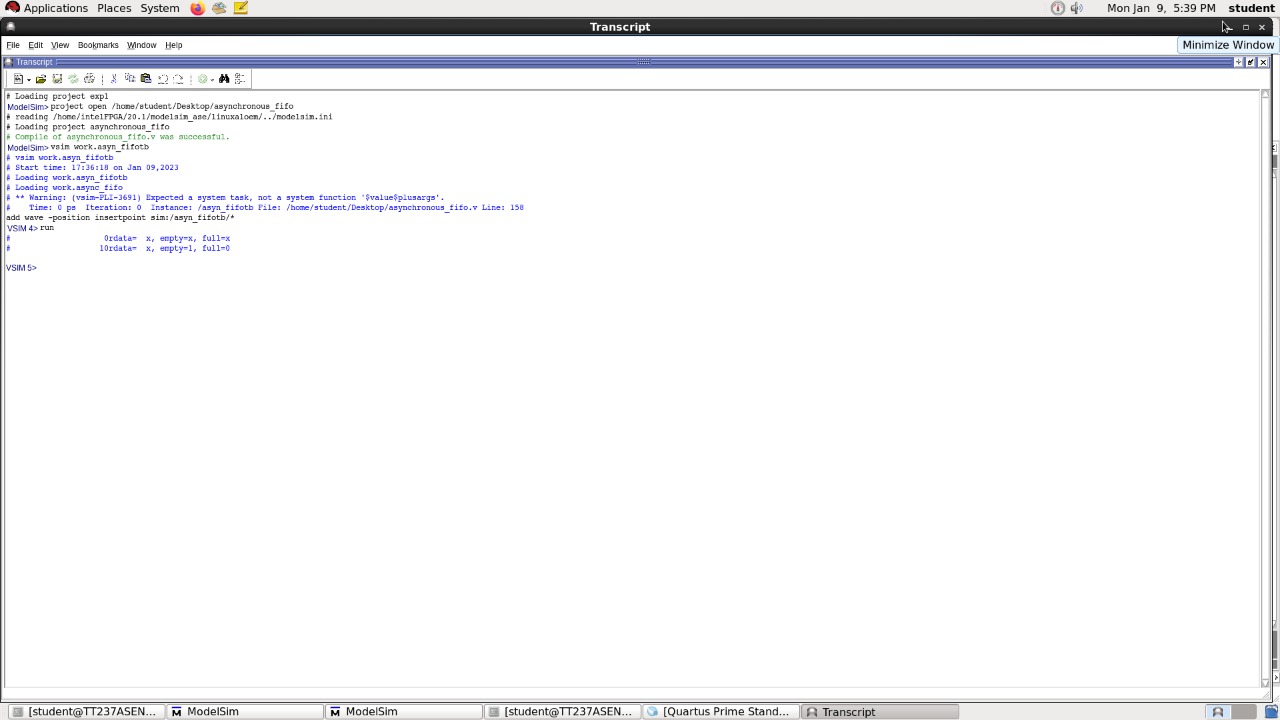
**Figure 2.** Design Level Architecture

**Results and Discussions**

The 512 memory spaces of the memory are allocated initially to 0 when the empty condition is being triggered. Each of the memory is obtained to be 9 bits depth as was designed and can be confirmed with reference to the Figure 3.

**Figure 3.** Initial memory locations of the Asynchronous FIFO

The empty condition is triggered and the full condition is set to zero as is evident in the Figure 4.



**Figure 4.** Empty condition obtained in the testbench

The Asynchronous FIFO, whose design was generated in the ModelSim, is implemented in the Quartus Prime 20.1 and the various results obtained are as follows. The top-level abstraction as obtained from the INTEL Quartus Prime 20.1 is as shown in Figure 11. The implementation on the FPGA board with indication of memory, read and write clocks are as given in Figure 12 and Figure 13.

**Table 1.** Flow Summary

|  |  |
| --- | --- |
| Details | Status |
| Flow Status | Successful | |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition | |
| Revision Name | asynch\_fifo\_test | |
| Top-level Entity Name | asynch\_fifo\_test | |
| Family | Cyclone IV E | |
| Device | EP4CCE115F29C7 | |
| Timing Models | Final | |
| Total logic elements | 270/114,480 (<1%) | |
| Total registers | 180 | |
| Total pins | 28/529 (5 %) | |
| Total virtual pins | 0 | |
| Total memory bits | 0 / 3,981,312 (0 %) | |
| Embedded Multiplier 9-bit elements | 0 / 532 (0 %) | |
| Total PLLs | 0 / 4 (0 %) | |

The above report gives us an idea as to how utilised the device is. From the report it is evident that less than 1% of the device is being made use of, in terms of logic elements.

**Table 2.** Power Analyzer Summary (for comparatively high toggle rate)

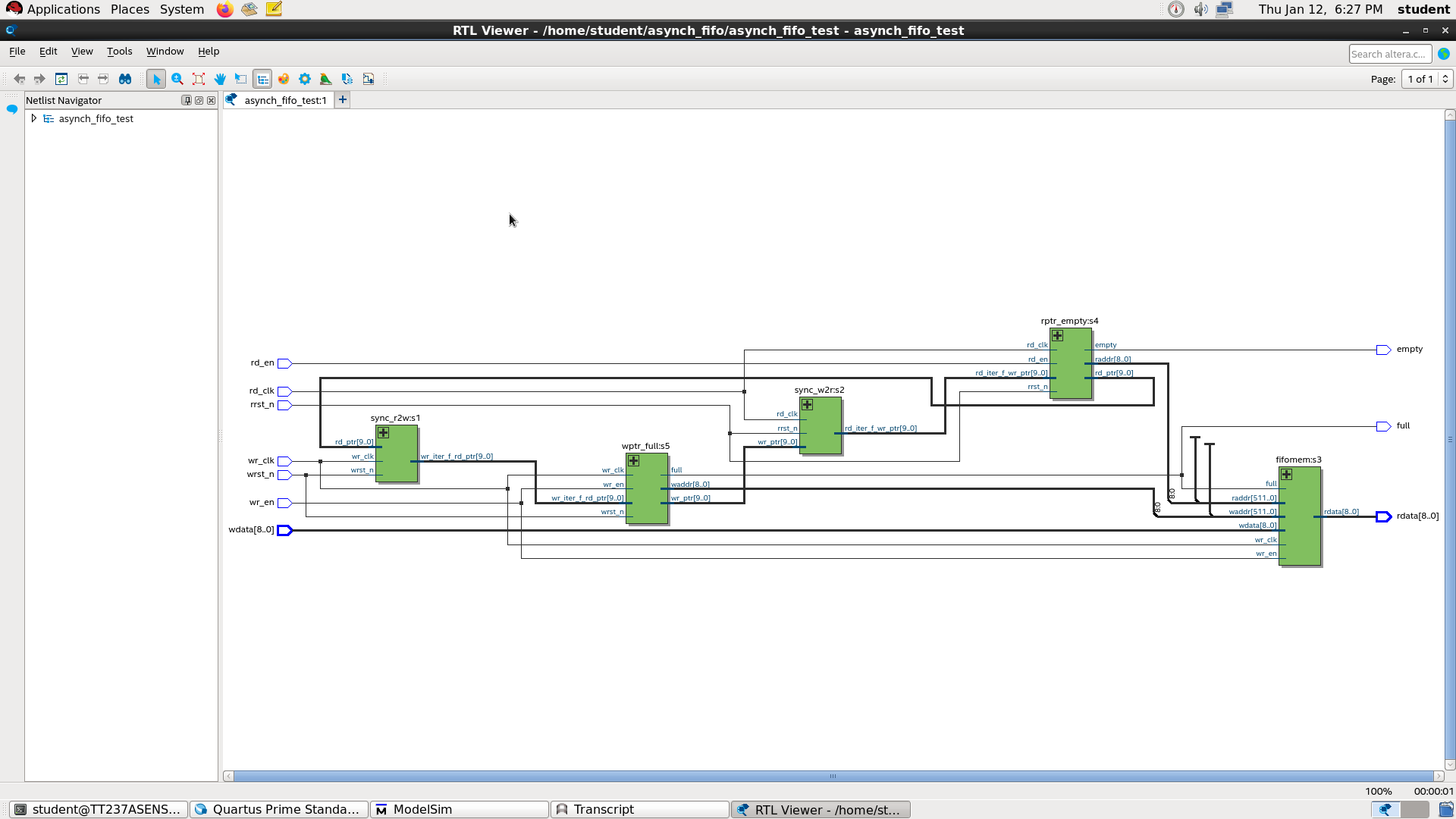
|  |  |
| --- | --- |
| Details | Status |
| Power Analyzer Status | Successful | |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition | |
| Revision Name | asynch\_fifo\_test | |
| Top-level Entity Name | asynch\_fifo\_test | |
| Family | Cyclone IV E | |
| Device | EP4CCE115F29C7 | |
| Power Models | Final | |
| Total Thermal Power Dissipation | 312.61 mW[[1]](#footnote-1)[a] | |
| Core Dynamic Thermal Power Dissipation | 32.45 mW | |
| Core Static Thermal Power Dissipation | 99.00 mW | |
| I/O Thermal Power Dissipation | 181.16 mW | |

**Table 3.** Power Analyzer Summary (for comparatively low toggle rate)

|  |  |
| --- | --- |
| Details | Status |
| Power Analyzer Status | Successful | |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition | |
| Revision Name | asynch\_fifo\_test | |
| Top-level Entity Name | asynch\_fifo\_test | |
| Family | Cyclone IV E | |
| Device | EP4CCE115F29C7 | |
| Power Models | Final | |
| Total Thermal Power Dissipation | 241.98 mW[[2]](#footnote-2)[a] | |
| Core Dynamic Thermal Power Dissipation | 0.11 mW | |
| Core Static Thermal Power Dissipation | 98.79 mW | |
| I/O Thermal Power Dissipation | 143.08 mW | |

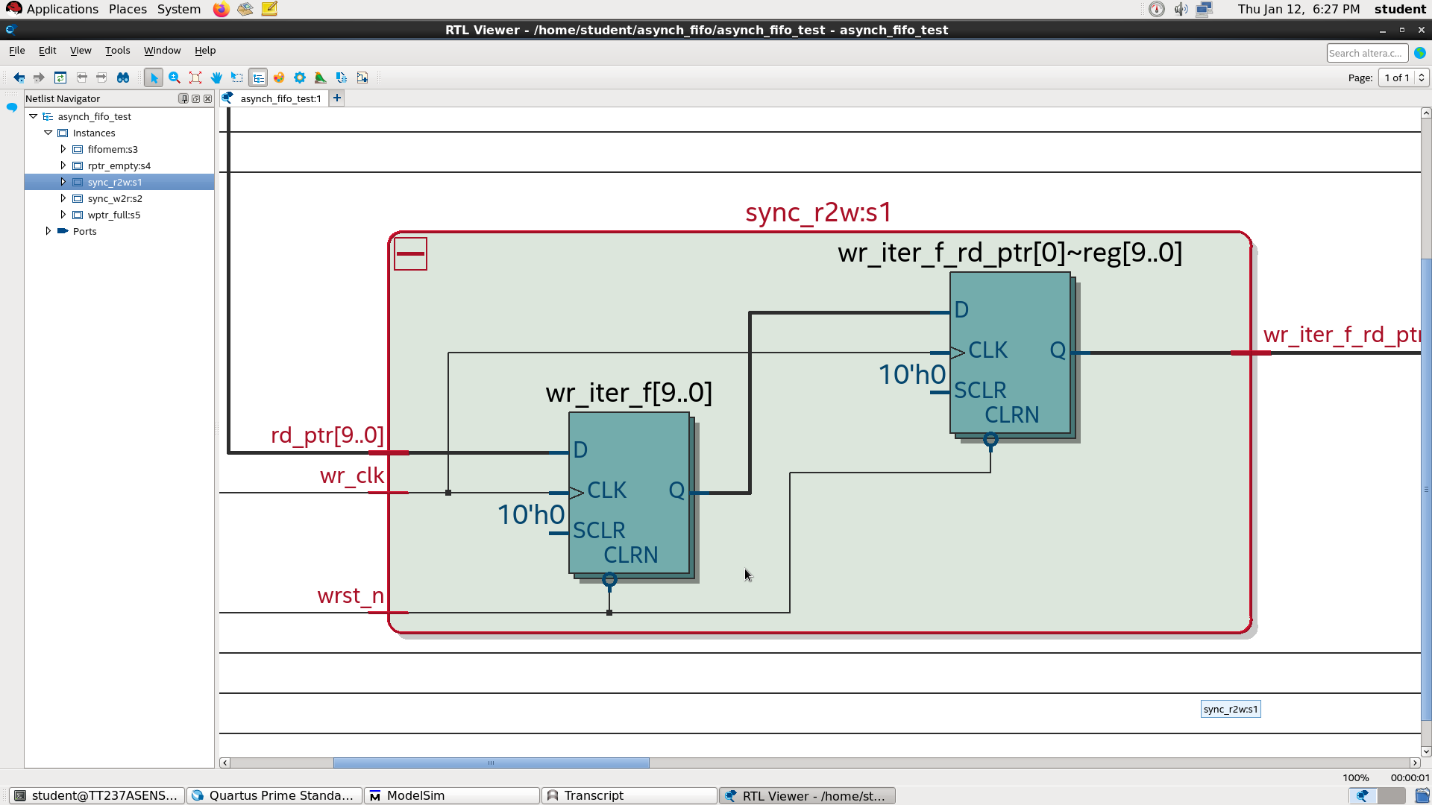
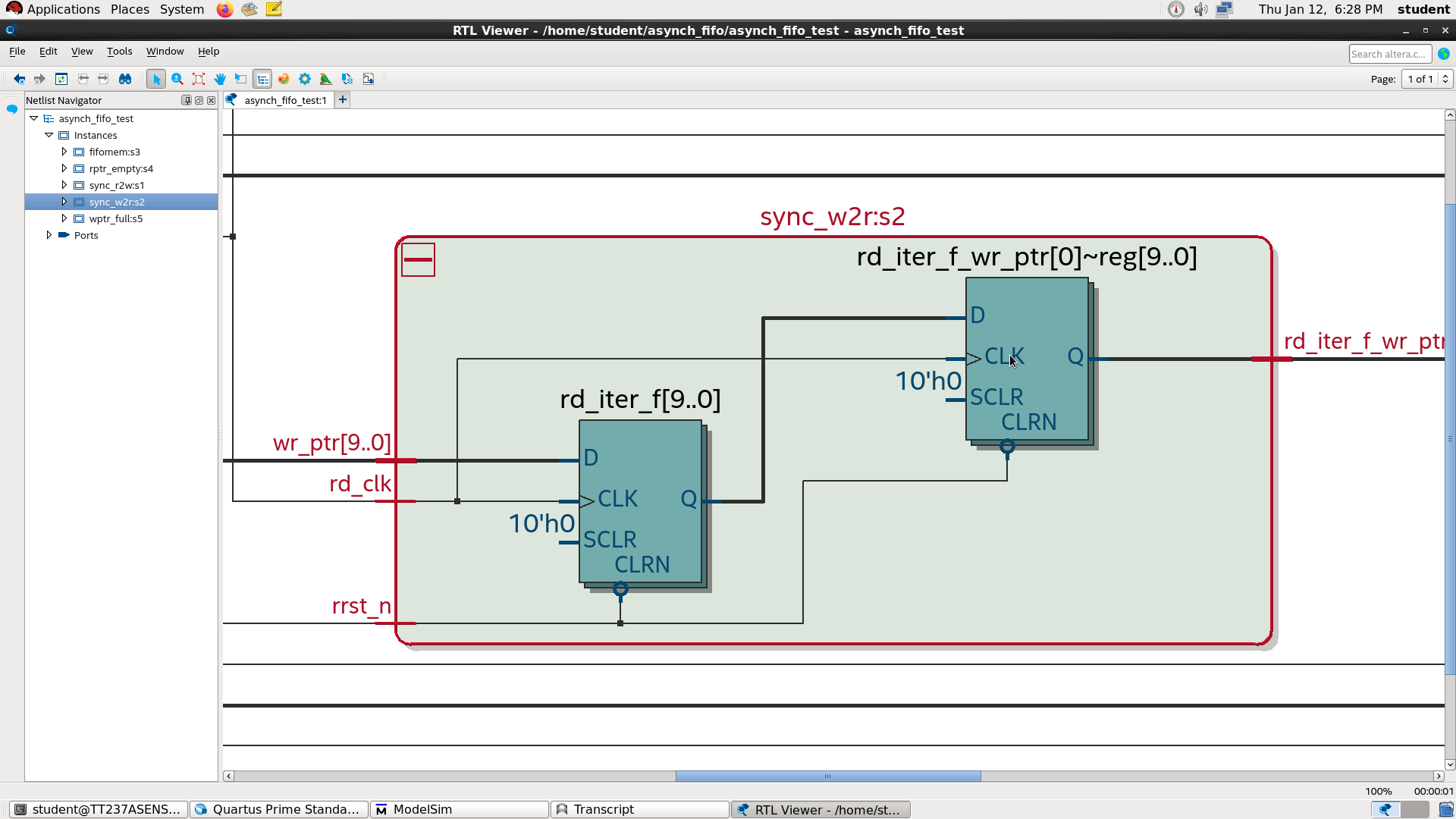
The above results are power analyzer reports which helps us to analyse the thermal power and estimate the power budget during the process of designing. This helps us to confirm that the designed model does not exceed the calculated values.

The thermal dissipation power gives an idea of the cooling requirements of the designed model. We can also make sure that the power that dissipates as heat is within the limits of the design.

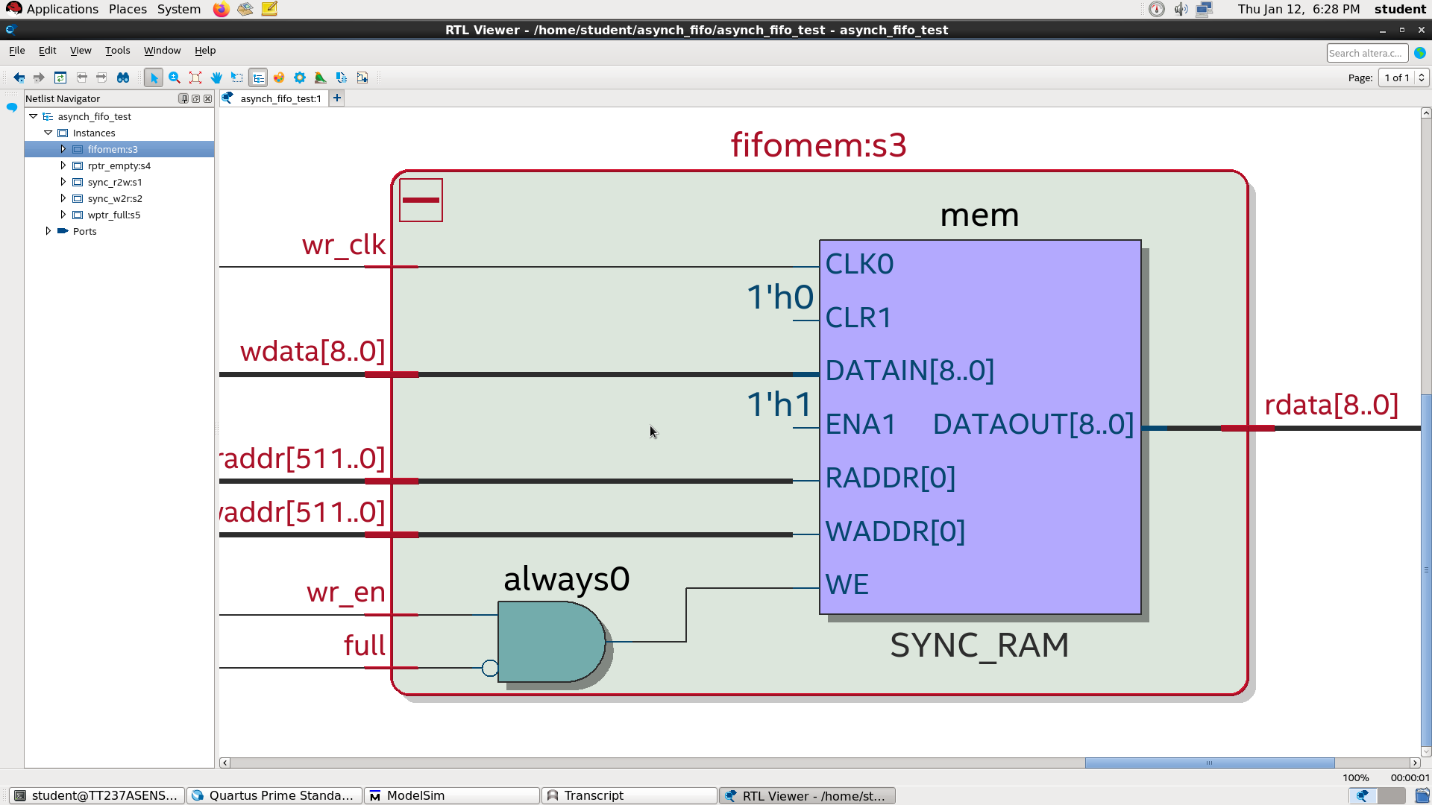
**Table 4**. Timing Analyzer Summary

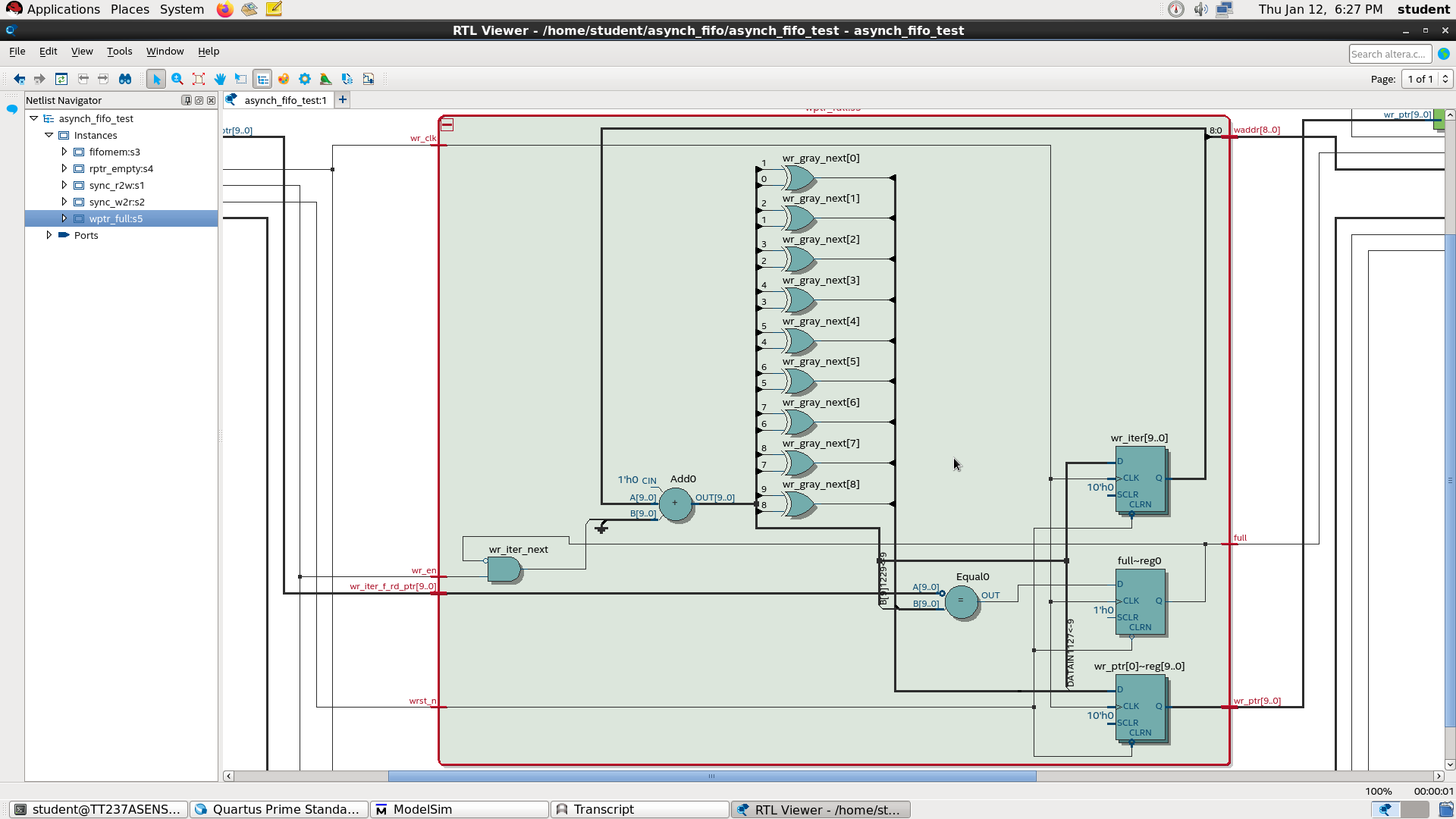
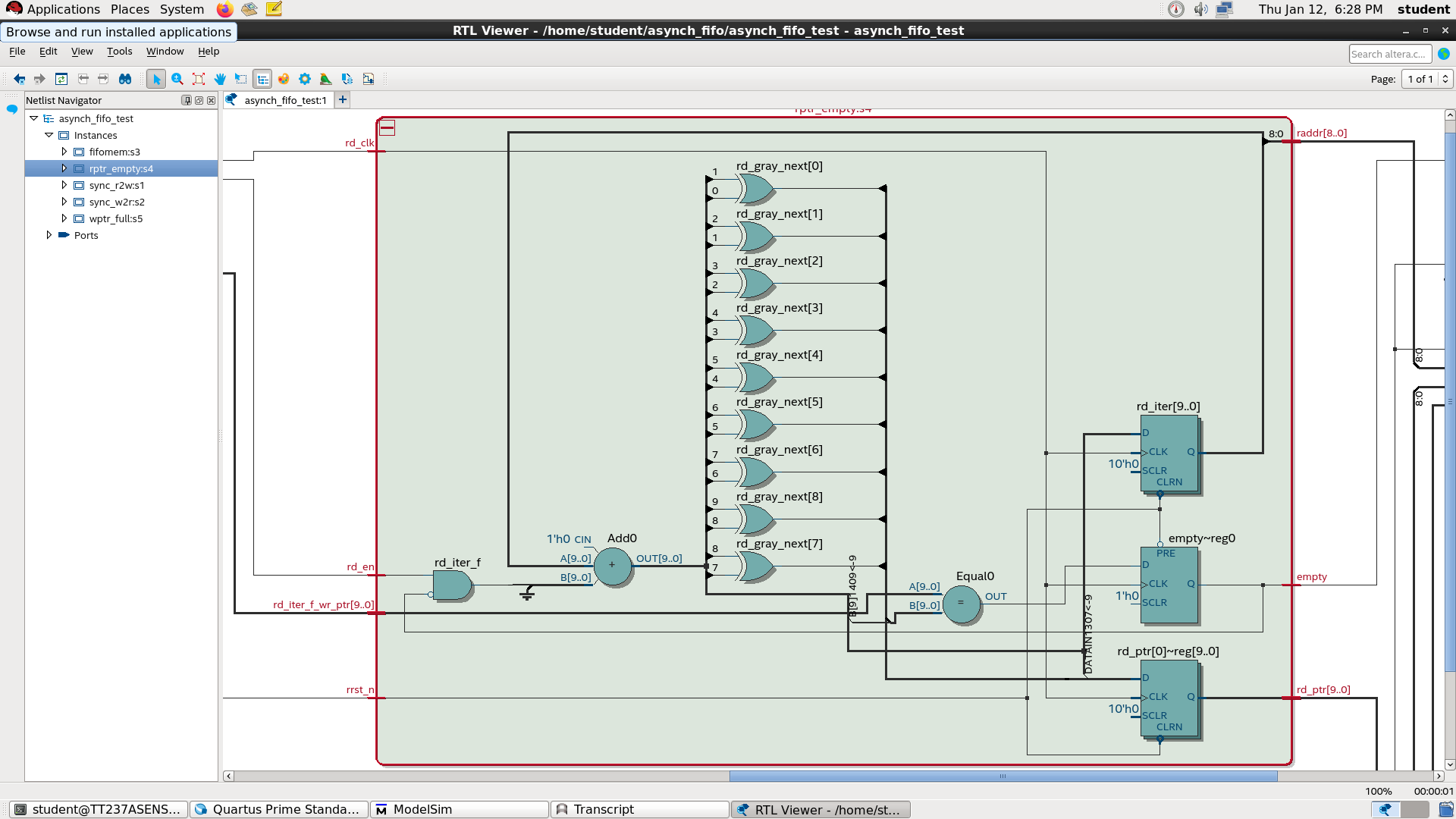
|  |  |
| --- | --- |
| Details | Status |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition | |
| Timing Analyzer | Legacy Timing Analyzer | |
| Revision Name | asynch\_fifo\_test | |
| Device Family | Cyclone IV E | |
| Device Name | EP4CCE115F29C7 | |
| Timing Models | Final | |
| Delay Models | Combined | |
| Rise/Fall Delays | Enabled | |

**Figure 5.** RTL Viewer of the top-level module of the designed Asynchronous FIFO

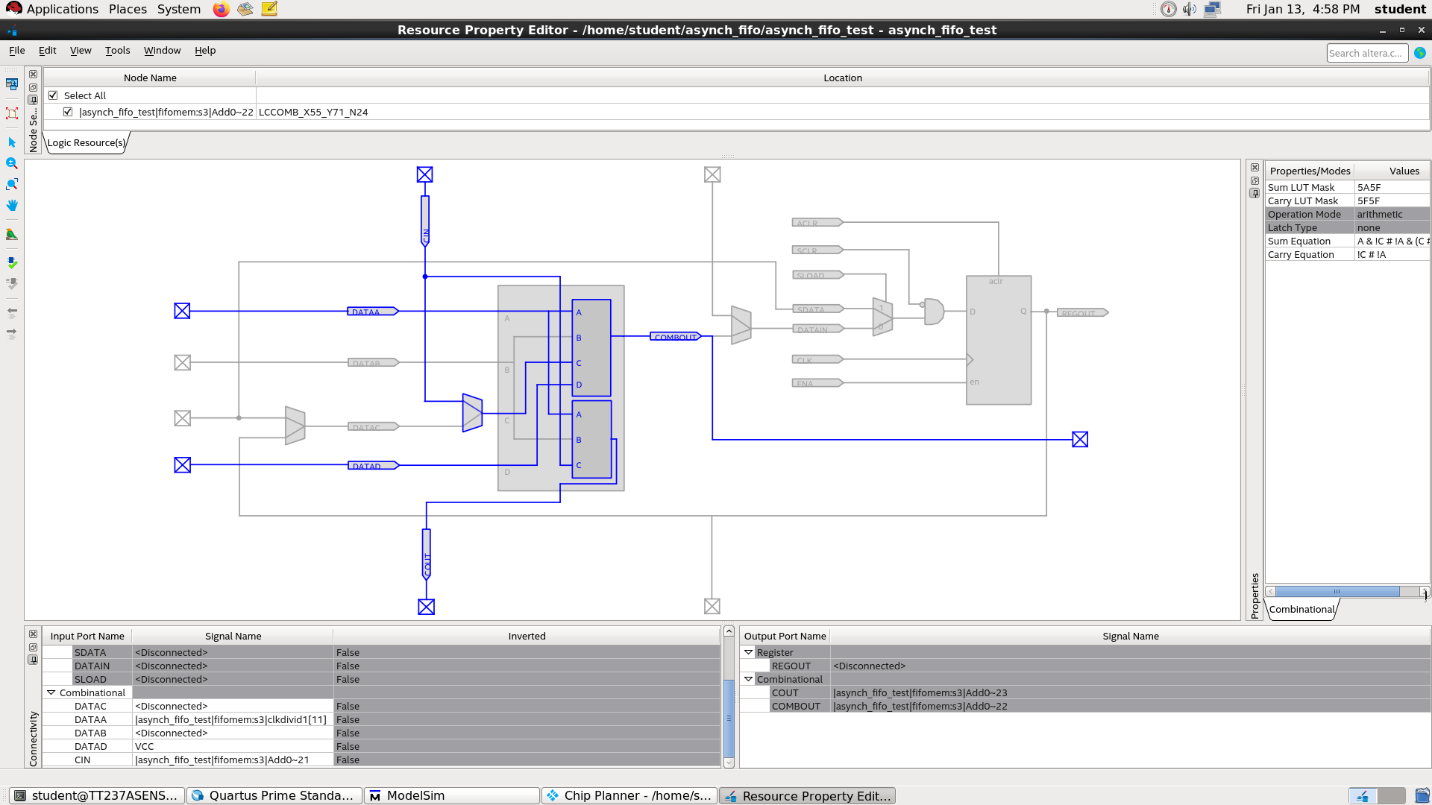
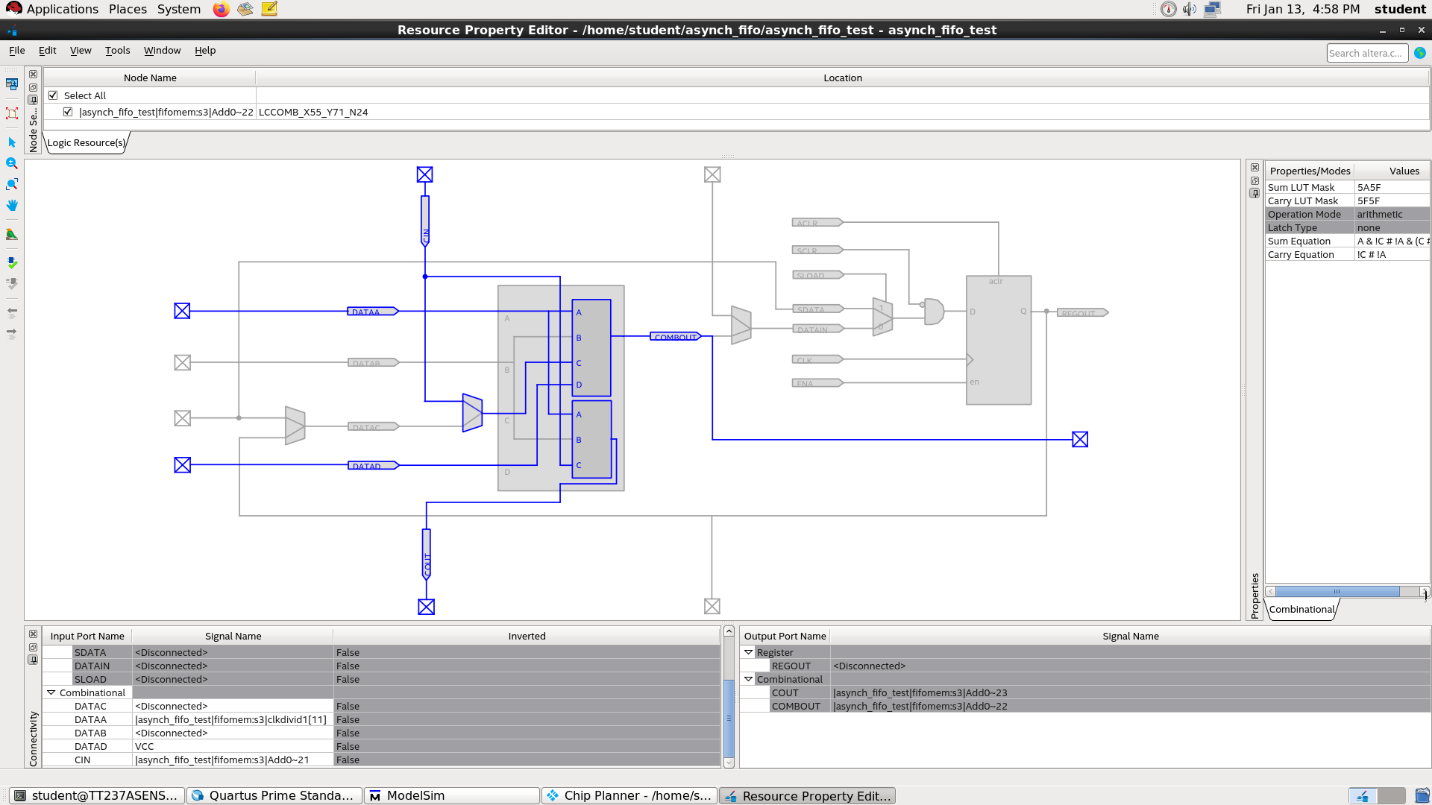
****Figure 6.** Expanded view of the write to read synchronisation logic

**Figure 7.** Expanded view of the read to write synchronisation logic

**Figure 8.** Expanded view of the memory logic **

******Figure 9.** Expanded view of the binary to gray converter

**Figure 10.** Expanded view of the gray to binary converter

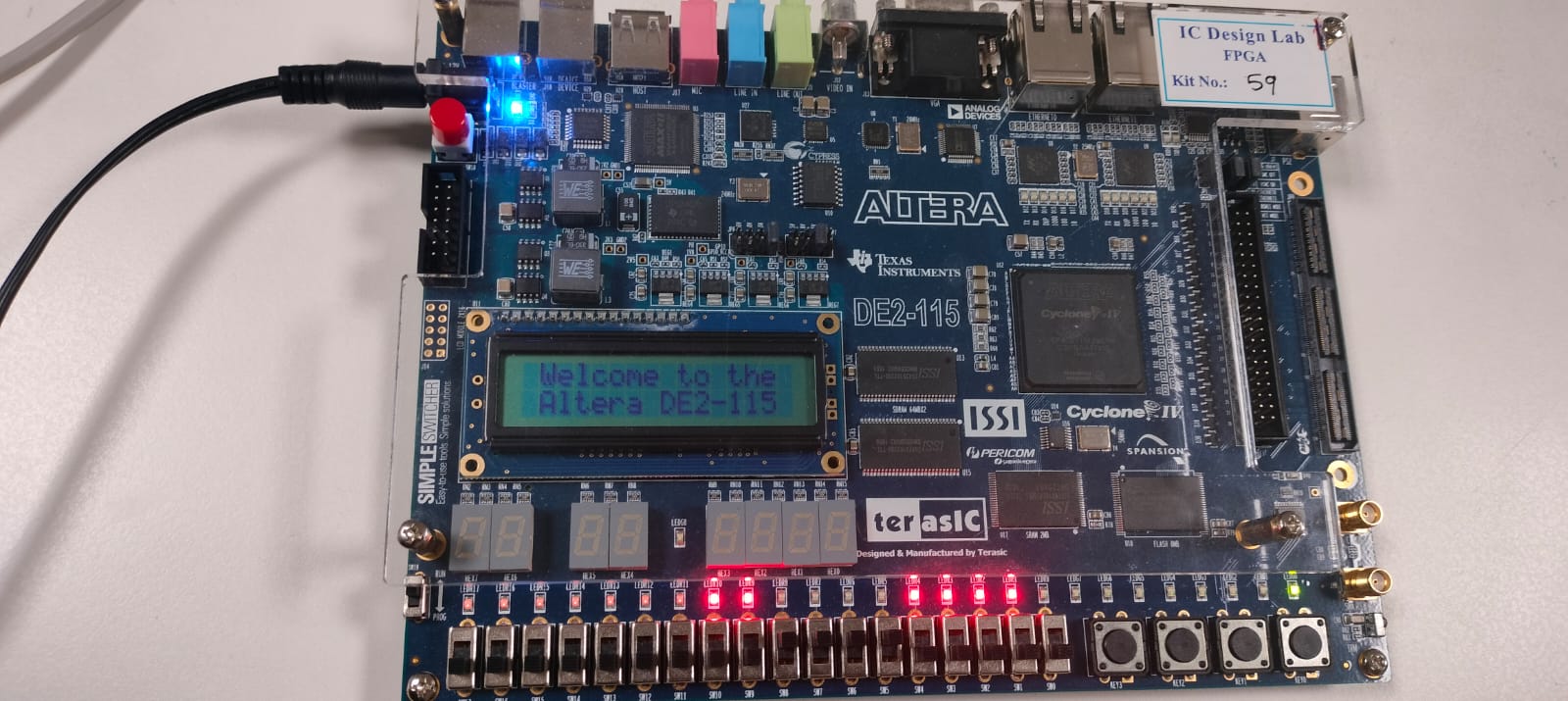
**Figure 10.** Expanded view of the gray to binary converter

**Figure 11.** Top Level Abstraction as obtained from INTEL Quartus Prime 20.1



Empty condition

**Figure 12.** Implementation of empty condition in the INTEL FPGA Board



Read clock and write clock

Memory

**Figure 13.** Implementation of memory writing in the INTEL FPGA Board

**Conclusion**

**The architecture of asynchronous FIFO given by the Cypress asynchronous FIFO is designed which simplify the task of data synchronization across various clock domains, synchronization is done to reduce the instability condition. Empty flag and full flag conditions are achieved by synchronizing to write pointer and read pointer respectively. Power is optimized and data loss is minimized. The functionality is verified using ModelSim simulator and Intel Quartus Prime.**

**Acknowledgments:**

**Our endeavor stands incomplete without dedicating our gratitude to everyone who has contributed a lot towards successful completion of this study. First, we offer our thanks to the parents for their blessings, we are indebted to God almighty for blessing us with his grace and taking our** **endeavor to successful culmination.**

We are very grateful to our guide Dr. Sivasankaran K, Professor Grade I, VIT Vellore for the support and guidance given to us and steering us to successful completion of this study.

We finally, thank all our well-wishers who had supported us directly indirectly during our project work.

**Conflict of Interest Statement:**

**The author’s in this paper has no conflict of interest statements.**

# **References**

Molnar, C. E., Jones, I. W., Coates, W. S., Lexau, J. K., Fairbanks, S. M., & Sutherland, I. E. (1999). Two FIFO ring performance experiments. *Proceedings of the IEEE*, *87*(2), 297-307.

Avinash Yadlapati and Hari Kishore Kakarla. (1816). Design and Verification of Asynchronous FIFO with Novel Architecture Using Verilog HDL, *Journal of Engineering and Applied Sciences, ISSN No,* 159-163.

Wang, S., Xu, Y., Tang, J., & Huang, S. (2021). Design of Asynchronous FIFO Controller Based on FPGA. *International Core Journal of Engineering*, *7*(1), 153-159.

Rao, P. R., & Nanda, M. Implementation and Verification of Asynchronous FIFO Under Boundary Condition.

Cummings, C. E. (2001, March). Synthesis and scripting techniques for designing multi-asynchronous clock designs. In *SNUG 2001 (Synopsys Users Group Conference, San Jose, CA, 2001) User Papers*.

Hauck, S. (1995). Asynchronous design methodologies: An overview. *Proceedings of the IEEE*, *83*(1), 69-93.

Yun, K. Y., & Donohue, R. P. (1996, October). Pausible clocking: A first step toward heterogeneous systems. In *Proceedings International Conference on Computer Design. VLSI in Computers and Processors* (pp. 118-123). IEEE.

Abdel-Hafeez, S., & Quwaider, M. Q. (2020, April). A one-cycle asynchronous FIFO queue buffer circuit. In *2020 11th International Conference on Information and Communication Systems (ICICS)* (pp. 388-393). IEEE.

Sharmaa, S. (2015, July). Implementation of an RTL synthesizable asynchronous FIFO for conveying data by avoiding actual data movement via FIFO. In *2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-7). IEEE.

**Tables**

[**Table 1.** Flow Summary 9](#_Toc124596518)

[**Table 2.** Power Analyzer Summary (for comparatively high toggle rate) 10](#_Toc124596519)

[**Table 3.** Power Analyzer Summary (for comparatively low toggle rate) 11](#_Toc124596520)

[**Table 4**. Timing Analyzer Summary 12](#_Toc124596521)

**Figures**

[**Figure 1.** Architecture of Asynchronous FIFO 6](#_Toc124596961)

[**Figure 2.** Design Level Architecture 7](#_Toc124596962)

[**Figure 3.** Initial memory locations of the Asynchronous FIFO 8](#_Toc124596963)

[**Figure 4.** Empty condition obtained in the testbench 8](#_Toc124596964)

[**Figure 5.** RTL Viewer of the top-level module of the designed Asynchronous FIFO 13](#_Toc124596965)

[**Figure 6.** Expanded view of the write to read synchronisation logic 13](#_Toc124596966)

[**Figure 7.** Expanded view of the read to write synchronisation logic 13](#_Toc124596967)

[**Figure 8.** Expanded view of the memory logic 14](#_Toc124596968)

[**Figure 9.** Expanded view of the binary to gray converter 14](#_Toc124596969)

[**Figure 10.** Expanded view of the gray to binary converter 15](#_Toc124596970)

[**Figure 11.** Top Level Abstraction as obtained from INTEL Quartus Prime 20.1 15](#_Toc124596971)

[**Figure 12.** Implementation of empty condition in the INTEL FPGA Board 16](#_Toc124596972)

[**Figure 13.** Implementation of memory writing in the INTEL FPGA Board 16](#_Toc124596973)

1. [a] milli Watt (unit of power) [↑](#footnote-ref-1)
2. [a] milli Watt (unit of Power) [↑](#footnote-ref-2)