Design, Implementation and Test of a Networks-on-Chip (NoC) Router using VHDL

CSE 215 Electronic Design Automation
Research Project Specification Document

Ain-Shams University
Cairo-Egypt
By: Dr. Haytham Azmi

Posted May 5, 2020

Contents

1 Introduction

A router is a networking module that buffers and forwards data (in packets format) across a network toward their destinations, through a process known as routing. Routers use headers and forwarding tables to determine the best path for forwarding the packets according to design requirements, and they use communication protocols to communicate with each other and configure the best route between any two nodes. Routers are not only used in computer networks applications, but also have been integrated in system-on-Chip (SoC) based designs to form what is know as Networks-on-Chip (NoC) applications. With SoC designs that have hundreds of Intellectual Property (IP) cores[1], interconnection using a single shared bus is not sufficient anymore. The NoC is a new paradigm that provides an integrated solution for achieving efficient inter-module communication [1]. Unlike computer networks, NoC has shorter communication delay and limited silicon area which put more limitations on the NoC-based system design and require different approaches to overcome design challenges specially in router design [2]. A router block diagram is shown in Figure 1. Once packets

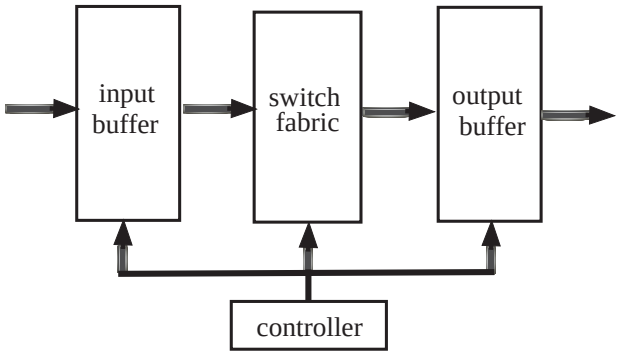

Figure 1: Router block diagram.

arrive at the input of the router, they are buffered at the input buffer, then the controller reads the packet header and configures the switch fabric to direct the packet to the appropriate output buffer. The output buffer is a group of FIFOs designed to handel multiple access requests at the output ports. Routing tables and timing synchronization are done in the controller part. In this project, we apply the concepts and design methodologies studied in Electronic Design Automation (CSE 215) course to design, implement, and test a simple router using VHDL language. The following sections explain in details the project objective, description, modules, and the project report outlines.

[1] An IP core is a block of logic or data that is used in making a field programmable gate array (FPGA) or application-specific integrated circuit (ASIC) for a product.

2 Project objective

The aim of this project is to design, implement and test a simple router using VHDL. The project main concern is the design of a synthesisable VHDL code for the router and its test bench using design methodologies studied in class.

3 Project description

Routers can be classified according to two different criteria [3]: the type of the switch fabric (SF) and the location of buffers and queues within the router. In this section, the functional description of one router type is explained. The output-queuing router with a point to point switch fabric is chosen as an example for simplicity. Digital systems usually contain two main blocks: datapath, and

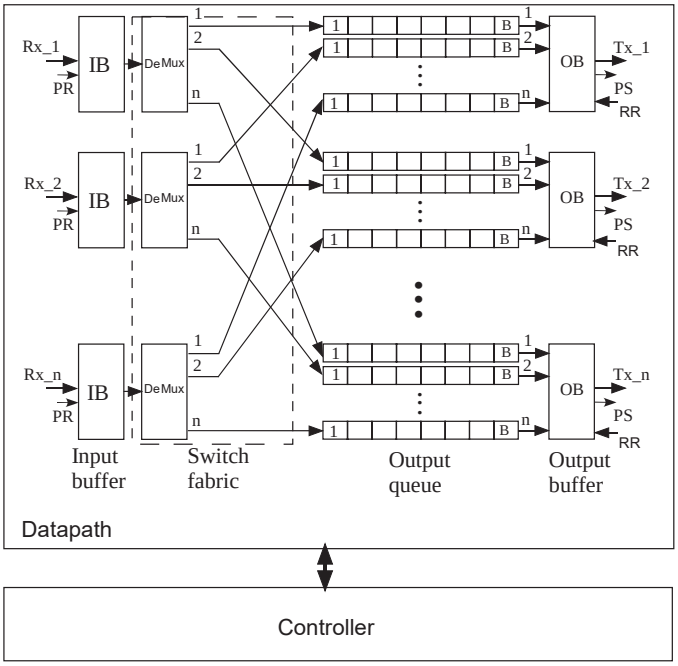

Figure 2: Output queuing router architecture.

controller. The datapath block consists of modules that can be implemented using combinational logic, while the controller block is usually implemented using finite state machine (FSM). Figure 2 shows the architecture of an n-ports output queuing router. Packets arrive at the input of the router asynchronously with a packet ready signal indicator (PR). The controller reads the packet header, then configures the switch fabric to direct the packet to the corresponding queue. There are n queues for each output port. These queues serve as FIFO buffers that are synchronized with both clock edges. Finally, packets are served in a

random service probability depending on the target destination availability. A round robin scheduler algorithm serves backlogged queues one after another in a fixed order at the output. An active high Receive Ready (RR) signal indicates the target destination availability. Packets are sent to the next hop with Packet Sent (PS) signal for synchronization purposes. The packet format is shown in Figure 3. The first three bits represents the destination port number, the following four bits contain the sent data, and the last bit is reserved for parity check.

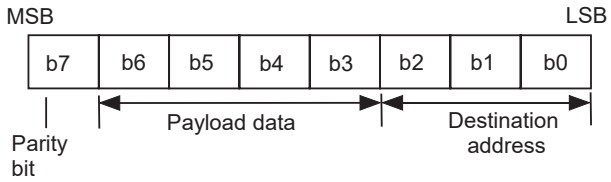

Figure 3: Packet format.

As shown in Figure 2, a simple router datapath consists of four main modules: input buffer, switch fabric, output queue, and output buffer. In the following subsections, we explore various design options for each internal module.

3.1 Input Buffer

The implementation of the input buffer can be done in many different ways. The input buffer simply is a register consists of 8 flip-flops. The main function of the input buffer is to store the packet once it arrived. It's for the designer to decide how will this module synchronize with the output world. For example, designers can add extra modules to synchronize with the incoming bits or extract the clock from the incoming packet.

3.2 Switch Fabric

The switch fabric (SF) unit is responsible of providing full connectivity between the inputs and outputs of the switch with reasonable hardware and delay [3]. In that sense, the input and output ports share the SF as a common communication resource, and, as such, conflicts may arise that have to be resolved using some contention-resolving technique or arbitration protocol. Reference [4] reviews and models five protocols and explains how they can be implemented in hardware.

A simple SF can be implemented as a set of demultiplexers as shown in Figure 2. These demultiplexers are controlled from the controller module. The controller module reads the first 3 bits in the incoming packet, then set the connection to output queue according to destination address.

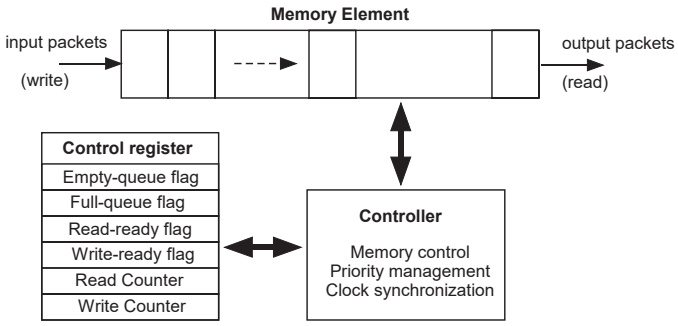

Figure 4: FIFO building blocks

3.3 Output Queue

Queues are usually implemented as First-in/First-out (FIFO) buffers. FIFO implementation has many design options and varieties. Starting from synchronous or asynchronous design, using one-port memory or dual-port memory, whether all arrived packets are served with equal priority or there is an arbitration rule, to the type of scheduler at the output (i.e., static priority, round robin, etc.)

As shown in Fig. 4, the main building blocks of a buffer are: memory element, control register, and controller. The memory unit consists of a set of words. Number of these words (buffer length) is defined by designers to match target traffic characteristics. The size of each word equals the packet width (assuming fixed packet size). Inaccurate estimation of the buffer length may cause poor system performance if the burstiness ratio exceeds the buffer capacity. On the other hand, if the buffer length is much greater than the actual number of bursty packets, excessive use of the silicon area will occur. Therefore, the modeling process is very important for choosing the appropriate buffer length.

The controller mainly manages the data flow inside the buffer to serve as a FIFO, in this experiment, and synchronize the internal system clock with the external interfaces. The control register consists of a set of flags to indicate the status of the FIFO and pointers (counters) to write/read to/from any memory address.

There are many proposed IP cores for various FIFO designs. [5] However, for NoC-based system, many design parameters must be considered such as synchronization problem, complexity of the controller, and the area constrains. It is for designers to choose the appropriate design that matches target application constraints and requirements.

3.4 Output Buffer

Just like the input buffer, the output buffer is a register consists of 8 flip-flops. The main function of the output buffer is to pick the packet from the output queue and deliver it to the output port according to the chosen schedular

algorithm. Again, it's for designers to decide which schedular technique they can use (round robin, processor sharing, etc.).

3.5 Controller

The controller module is designed to:

1. Synchronize router internal modules with the incoming packets.
2. Read packet header and configure the switch fabric.
3. Control data flow inside the router and apply round-robin schedular at the output ports.

4 Project Modules

4.1 Module No. M-ROU-01

Objective: **To write VHDL code for 8-bit Register with Positive-Edge Clock, Asynchronous Reset, and Clock Enable.**
The VHDL code should follow pin definitions and design parameters mentioned

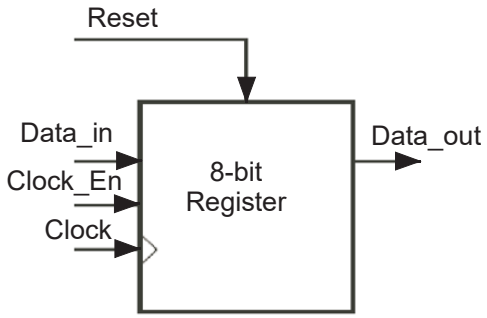

Figure 5: Input buffer.

in Table 1.
Pin Description:

Table 1: Input buffer

IO pin	**Description**
Datat_in [7:0]	Data Input
Clock	Positive-Edge Clock
Data_out [7:0]	Data Output
Clock_En	Clock Enable (active High)
Reset	Asynchronous Reset (active High)

1. Data is transferred from input port to output port at the clock positive edge if and only if $Clock\_En = 1$ and $Reset = 0$.
2. If Reset = 1, then output port is set to "00000000".
3. If $Clock\_En = 0$, then output port keeps its current value.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.2 Module No. M-ROU-02

Objective: **To write VHDL code for 1-to-4 8-bit DeMUX Using Case Statement.**

The VHDL code should follow pin definitions and design parameters mentioned

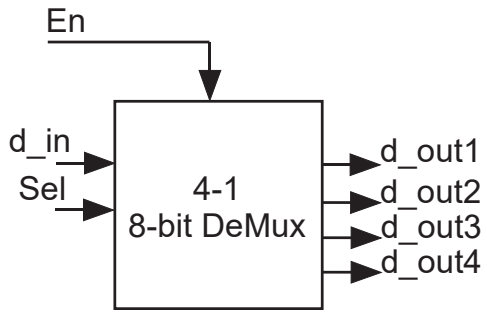

Figure 6: Demultiplexer.

in Table 2.

Pin Description:

Table 2: Demultiplexer

IO pin	Description
d_in [7:0]	Data Input
*d_out*1 [7:0]	Data Output - port 1 ; d_out1 = d_in ONLY when Sel ="00"
*d_out*2 [7:0]	Data Output - Port 2 ; d_out2 = d_in ONLY when Sel ="01"
*d_out*3 [7:0]	Data Output - Port 3 ; d_out3 = d_in ONLY when Sel ="10"
*d_out*4 [7:0]	Data Output - port 4 ; d_out4 = d_in ONLY when Sel ="11"
Sel [1:0]	Selector
En	Enable

1. Data is transferred from input port to the appropriate output port according to selector value if $En = 1$.
2. If $En = 0$, then, all output ports keep their current value.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.3 Module No. M-ROU-03

Objective: **To write VHDL code for Dual-Port Block RAM with Different Clocks.**

The VHDL code should follow pin definitions and design parameters mentioned

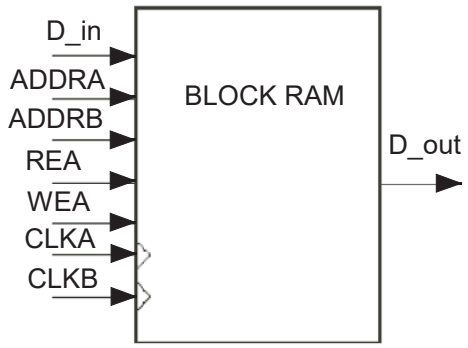

Figure 7: Dual-port RAM.

in Table 3.

Pin Description:

Table 3: Dual-port RAM

IO pin	**Description**
d_in [7:0]	Data Input
d_out [7:0]	Data Output
WEA	Write enable (active high)
REA	Read enable (active high)
ADDRA [2:0]	Write port (A) address bus
ADDRB [2:0]	Read port (B) address bus
CLKA	Clock signal for port A
CLKB	Clock signal for port B

1. Port A (write) is synchronized with CLKA and port B (read) is synchronized with CLKB.

The delivery has to contain:

Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.4 Module No. M-ROU-04

Objective: **To write VHDL code for a Gray Counter.**
The VHDL code should follow pin definitions and design parameters mentioned

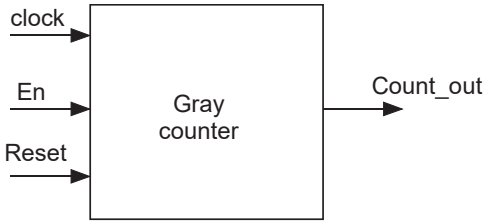

Figure 8: Gray Counter.

in Table 4.
Pin Description:

Table 4: Gray counter

IO pin	**Description**
clock	Counter Clock (rising edge)
Reset	Reset (active high)
En	Enable (active high)
Count_out [3:0]	Counter output

1. A counter that counts up with the rising edge of each clock when it's enable pin is set to '1'. The output is gray code.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.5 Module No. M-ROU-05

Objective: **To write VHDL code for a Gray to Binary Converter.**
The VHDL code should follow pin definitions and design parameters mentioned

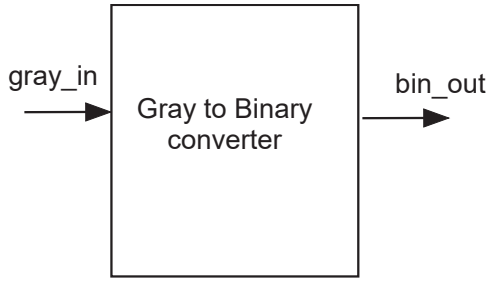

Figure 9: Gray to binary converter.

in Table 5.
Pin Description:

Table 5: Gray to Binary Converter

IO pin	Description
gray_in [3:0]	Gray code input
bin_out [3:0]	Binary code output

1. A Simple Gray to binary converter.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.6 Module No. M-ROU-06

Objective: **To write VHDL code for a FIFO controller.**
The VHDL code should follow pin definitions and design parameters mentioned

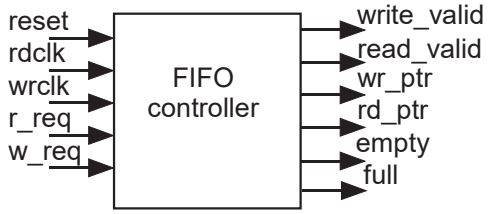

Figure 10: FIFO controller.

in Table 6.
Pin Description:

Table 6: FIFO controller

IO pin	**Description**
reset	reset signal (active high)
rdclk	input read clock
wrclk	input write clock
r_req	request read signal
w_req	request write signal
$write\_valid$	valid write output indication
$read\_valid$	valid read output indication
$wr\_ptr[3:0]$	write address
$rd\_ptr[3:0]$	read address
empty	FIFO empty flag
full	FIFO full flag

FIFO controller receives read request and write request from r_req and w_req signals respectively. Then, it checks the validity of read and write operation and generates valid-indication-signal on $read\_valid$ and $write\_valid$ ports respectively. Finally, controller outputs the corresponding read and write addresses. When FIFO is full, write operation is disabled, and when it is empty, read operation is disabled. FIFO controller sets the *empty* output port to high when the FIFO is empty and sets *full* output port to high when FIFO is full.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.7 Module No. M-ROU-07

Objective: **To write VHDL code for a FIFO**
The VHDL code should follow pin definitions and design parameters mentioned

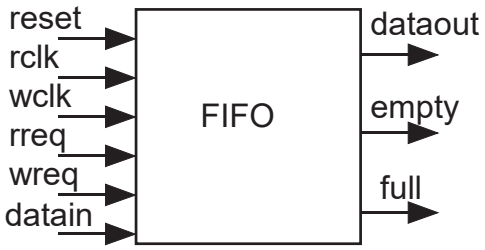

Figure 11: FIFO.

in Table 7.
Pin Description:

Table 7: FIFO controller

IO pin	**Description**
reset	reset signal (active high)
rclk	input read clock
wclk	input write clock
rreq	request read signal
wreq	request write signal
datain [7:0]	input data to FIFO
dataout [7:0]	output data from FIFO
empty	FIFO empty flag
full	FIFO full flag

FIFO receives read request and write request from *rreq* and *wreq* signals respectively. When FIFO is full, write operation is disabled, and when it is empty, read operation is disabled. FIFO *empty* output port is set to high when the FIFO is empty and *full* output port is set to high when FIFO is full.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.8 Module No. M-ROU-08

Objective: **To write VHDL code for a Round Robin Schedular**
The VHDL code should follow pin definitions and design parameters mentioned

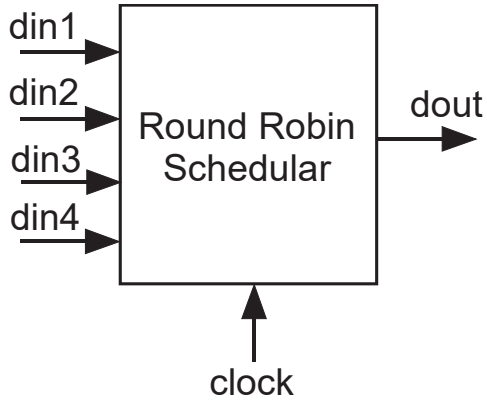

Figure 12: Round robin Schedular.

in Table 8.
Pin Description:

Table 8: Round robin Schedular

IO pin	Description
clock	module clock (rising edge)
din1 [7:0]	input data port 1
din2 [7:0]	input data port 2
din3 [7:0]	input data port 3
din4 [7:0]	input data port 4
dout [7:0]	output data

Round-robin scheduling is a scheduling algorithm which repeatedly runs through a list of users. In this module, we have 4 input ports and one output port. A round robin scheduler algorithm serves backlogged ports one after another in a fixed order at the output. That means in the first clock cycle, data from input port 1 is transferred to the output port. The next clock cycle port 2 is served and so on.

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.9 Module No. M-ROU-09

Objective: **To write VHDL code for a 4-port router**
The VHDL code should follow pin definitions and design parameters mentioned

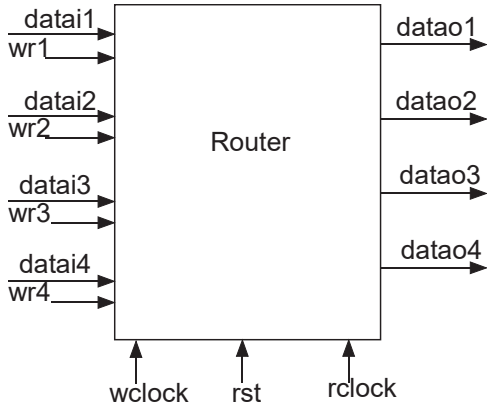

Figure 13: Round robin Schedular.

in Table 9.
Pin Description:

Table 9: Round robin Schedular

IO pin	Description
datai1,datai2,datai3,datai4 [7:0]	input ports
wr1, wr2, wr3, wr4	packet available indicator (active high)
datao1,datao2,datao3,datao4 [7:0]	output ports
wclock	arrival clock for input ports synchronization
rclock	service clock for output ports synchronization
rst	reset signal

Data arrived at the input port in a fixed packet length format (8 bits). The first two bits are the output port address. The router is to transfer input packets to the appropriate output ports. The router input ports are synchronized with an input clock source (wclock) whereas the output ports are synchronized with the scheduler clock source (rclock).

The delivery has to contain:
Synthesizable VHDL Code, total equivalent gate count for the design, and screen shot from Modelsim wave window.

4.10 Module No. M-ROU-10

Objective: **Design Verification for a 4-port router**

In this task, you are expected to write a test-bench to test the router functionality and analyze its performance.

The delivery has to contain:
Testbench VHDL Code, performance analysis and screen shot from Modelsim wave window.

5 Report outlines

The final project report will include a detail description of the design problem, background information, the design approach and detailed design information, test scenarios, cases and results, cost data, and conclusions and recommendations. The final project report should also describe how you demonstrated your design.

The report should contain the following sections:

1. **Introduction - [ILO: c2]**
   The introduction is an essential part of any report; it prepares the reader to read the main body of the report. This section should state the objectives, assumptions and desired outcomes of the project. Why we need NoC Routers, what are the different router types, and what are the main building blocks of your router. It should also introduce the reader to the basics of how to build a simple chip from specifications down to layout using CAD tools.

2. **Design Flow - [ILO: a1, a3, a6, a7]**
   In this section you discuss the basic integrated circuit subsystems and process flow. This includes an illustration of the digital design and verification flows and the steps that must be performed to complete the logic and physical synthesis. Support this chapter with figures that explain the flow and give examples of the CAD tools that could be used to complete the chip design flow.

3. **Literature Review - [ILO: a1, a6, a7, b4, c2, d1, d2]**
   In this section you survey the related implementations of NoC routers in the literature. Conducting a literature review involves collecting, evaluating and analyzing publications (such as books and journal articles) that relate to the HDL implementation of NoC Router. A good literature review doesn't just summarize sources – it analyzes, synthesizes, and critically evaluates to give a clear picture of the state of knowledge on the subject.

4. **Design Implementation - [ILO: a1, a2, a4, b3, c1]**
   This section should state the functional description of each element in the design and how it was implemented. Start by creating a block diagram of the design then provide a table that lists the design modules and the function of each module. This section discusses the implementation of each module in VHDL, the design challenges, and how the sequential elements and combinational logic are integrated and synthesized. USE Appendix A to include all VHDL Source codes

5. **Schedular Design and FSM Implementation - [ILO: a2, a4, a5, b1, b2, b3]**
   This section focuses on the design and implementation of the schedular module using FSM design. You should draw the FSM diagram, explain

its implementation type (mealy vs moore) and justify your choice. You should also discuss the differences between VHDL implementation styles (single process, 2-process, and 3-process styles). The section should report the timing analysis of this module. Finally, a synthesis analysis should be done to explain how this module is synthesized and to highlight the critical path in the circuit.

6. **Test and Simulation Results - [ILO: a1, a4, b3, b4]**
   This section should discuss the test strategies and Compare different strategies in testing the Router. This includes the implementation of test-bench files and the choices of various testcases and code coverage analysis. USE Appendix B to report VHDL Test Bench Source Code. USE Appendix C to report the Simulation Waveform Output

7. **Conclusion - [ILO: a5, b1, c1]**
   This section should discuss the design challenges during the digital design process and summarizes the implementation results of the design. It should also conclude what is the best implementation style for the FSM schedular and the test strategies used in the design. This part should also report the timing results and suggest possible directions for future work.

8. **Task Distribution List**
   Use the table below to explain what each student did in the project. If the workload is distributed equally among all team members, you may write "the project workload is distributed equally among all team members"

6 Evaluation Criteria

1. **90% and above:** Your report must be of outstanding quality and fully meet the requirements of the project specification and learning outcomes stated. You must show independent thinking and apply this to your work showing originality and consideration of key issues. There must be evidence of wider reading on the subject. In addition, your report should:

   - illustrate a professional ability of designing digital logic using VHDL,
   - express a deep understanding of the simulation and synthesis of logic circuits,
   - and applying, masterly, the learned knowledge in the report.

2. **76% - 89%:** Your project must be of good quality and meet the requirements of the project specification and learning outcomes stated. You must demonstrate some originality in your work and show this by applying new learning to the key issues of the project. There must be evidence of wider reading on the subject. In addition, your report should:

   - illustrate a Good ability of designing digital logic using VHDL
   - express a very Good understanding of the simulation and synthesis of logic circuits,
   - and applying most of the learned knowledge in the report.

3. **67% - 76%:** Your project must be comprehensive and meet all of the requirements stated by the project specification and learning outcomes. You must show a good understanding of the key concepts and be able to apply them to solve the problem set by the project specs. There must be enough depth to your work to provide evidence of wider reading. In addition, your report should:

   - illustrate a moderate ability of designing digital logic using VHDL,
   - express a good understanding of the simulation and synthesis of logic circuits,
   - and applying most of the learned knowledge, correctly, in the report.

4. **50% - 67%:** Your work must be of a standard that meets the requirements stated by the project specification and learning outcomes. You must show a reasonable level of understanding of the key concepts and principles and you must have applied this knowledge to the project problem. There should be some evidence of wider reading. In addition, your report should:

   - illustrate a fair ability of designing digital logic using VHDL,
   - express a fair understanding of the simulation and synthesis of logic circuits,
   - and applying some of the learned knowledge, correctly, in the report.

5. **Below 50%:** Your project is of poor quality and does not meet the requirements stated by the project specification and learning outcomes. There is a lack of understanding of key concepts and knowledge and no evidence of wider reading. In addition, your report would:

   - Illustrate an inability of designing digital logic using VHDL,
   - Failed to define the basics of the simulation and synthesis of logic circuits,
   - Failed to apply correctly the learned knowledge for proposing a valid solution.

7 Academic Misconduct

The University defines Academic Misconduct as "any case of deliberate, premeditated cheating, collusion, plagiarism or falsification of information, in an attempt to deceive and gain an unfair advantage in assessment". This includes attempting to gain marks as part of a team without making a contribution. The department takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy. If you are found guilty, you may be expelled from the University with no award. It is your responsibility to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer Magazine*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[2] A. Ivanov and G. De Micheli, "Guest editors' introduction: The network-on-chip paradigm in practice and research," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 399–403, Sep. 2005.

[3] F. Gebali, *Computer Communications networks: Analysis and Design*, 3rd ed. Victoria, B.C., Canada: Northstar Digital Design, inc., 2005.

[4] F. Elguibaly, "Analysis and design of arbitration protocols," *IEEE Transactions on Computers*, vol. 38, no. 2, pp. 1168–1175, 1989.

[5] LogicCore. (2004, Nov. 11,) Xilinx Technical Document, Asynchronous FIFO v6.1. Product Specification DS232. [Online]. Available: $http://www.xilinx.com/ipcenter/catalog/logicore/docs/async_fifo.pdf$