

Ministry of Higher Education and Scientific Research
La Manouba University
National School of Computer Science



System integration Report

Subject

UART PERIPHERAL INTEGRATED WITH NIOS 2

Directed By
Omar ROMDHANI
&
Ibrahim MAHDI

Supervised By :
Dr. Lobna Kriaa
&
Mr. Mohamed Masmoudi

Academic Year : 2019/2020

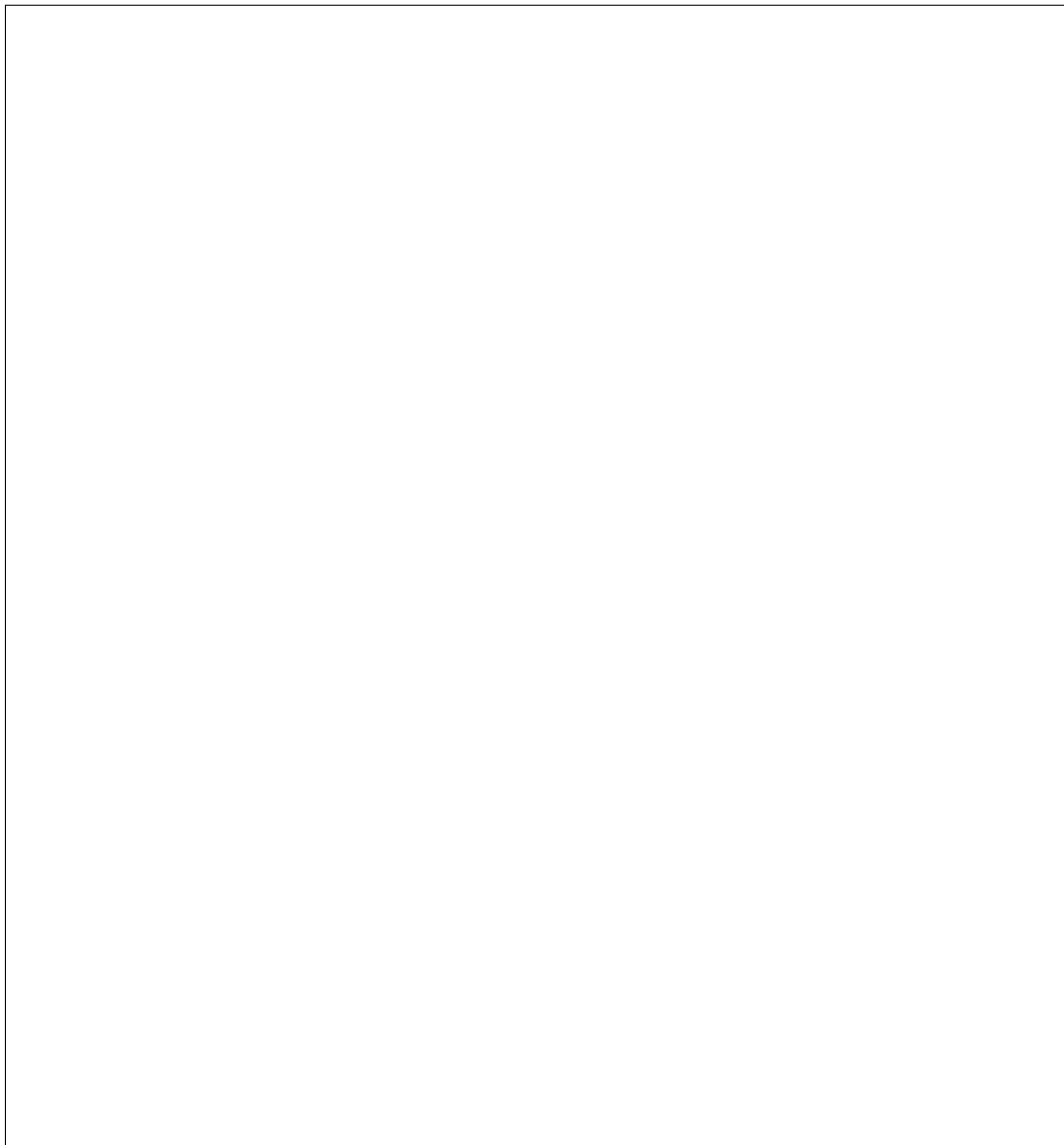
Abstract— This report gathers the design and implementation of an UART Peripheral. It also contains a description of how to integrate this peripheral with a Soft-core Processor, Nios 2, via an avalon interface.

Key words : UART, VHDL, Qsys, Nios2, Soft-core, avalon

Résumé— Ce rapport rassemble la conception et l'implémentation d'un périphérique UART en VHDL. Il contient également une description de l'intégration de ce périphérique avec un processeur Soft-core Nios2 à travers un inteface avalon.

Mots clés : UART, VHDL, Qsys, Nios2, Soft-core, avalon

Appreciation and signature of the supervisor

A large, empty rectangular box with a thin black border, intended for the supervisor's appreciation and signature. It occupies the lower half of the page.

Acknowledgement

We would like to express our sincere gratitude to our supervisors without whom it would have not be possible to complete the work.

We would like to express our gratitude to **Mr. Mohamed MASMOUDI** as well as **Ms. Lobna KRIAA** for their priceless help and their valuable and constructive instructions during the planning and development of my project.

We are highly grateful for their guidance and their continuous supervision also for providing us with all the adequate documentation in the project's field.

We would like to extend our recognition and gratitude to them as well as all people who contributed with their competence in realizing this project.

Contents

Introduction	1
1 Preliminary Study	2
1.1 UART Basics	2
1.2 UART data frame	2
2 Design	3
2.1 Input/Output signals	3
2.2 Peripheral State Machine	4
3 Test and Validation	5
3.1 Manual test	5
3.2 Test bench	6
4 Integrating Peripheral with Nios2	7
4.1 UART interfaces	7
4.2 Linking the Peripheral with the Nios2	9
Netography	10

List of Figures

2.1	Proposed UART inputs/outputs	3
2.2	Finite State Machine	4
3.1	manual test of data transmission	5
3.2	manual Test bench	6
4.1	Signals	7
4.2	manual Test bench	8
4.3	System Contents	9

Glossary of Acronyms

ASIC *Application-specific integrated circuit*

FSM *Finite State Machine*

ENSI *Ecole Nationale des Sciences de l'Informatique*

VHDL *Very High Speed Integrated Circuit Hardware Description Language*

UART *Universal asynchronous receiver-transmitter*

General introduction

In the last years, the evolution of technologies contributed in a rapid increase in the size of data which revealed the necessity of using powerful tools to process these data, Application-specific integrated circuits (ASIC) are the best in terms of performance, but these circuits may take a lot of time to get constructed, and once it is created, it is not possible anymore to modify its circuit.

This revealed the necessity of a more flexible solution. FPGAs are particularly useful for prototyping ASICs or processors. An FPGA can be reprogrammed until the ASIC or processor design is final and bug-free and the actual manufacturing of the final ASIC begins.

In this field, and as computer science engineering students, we realized this project to get familiarized with this technology and to learn more about it.

Chapter 1

Preliminary Study

Introduction

The purpose of this project is to create an UART (Universal Asynchronous Receiver-Transmitter) device. This chapter highlights the basics that we need to create our UART device.

1.1 UART Basics

UART is used to transmit and receive serial data, it uses two wires Tx and Rx where Tx is used to send serial data and Rx is used to receive serial data. The UART bus is used to send or receive data from one device to another. The Tx wire of the first device is connected to the Rx wire of the second device.

Since the data will be transmitted on one wire, we will be sending multiple bits in sequences, and in order to receive these consequences, the two devices should use the same baud rate. The baud rate is the rate at which information is transferred in a communication channel. In the serial port context, "9600 baud" means that the serial port is capable of transferring a maximum of 9600 bits per second .

1.2 UART data frame

The UART data frame contains 4 Fields:

- The starting bit : Indicates the beginning of the data transmission.
- The data field : Contains the data that we want to send and it can be represented on 5 to 9 bits.
- the parity bit : Is used to control the data transmission .
- stopping bit : Indicates the end of the transmission.

Chapter 2

Design

2.1 Input/Output signals

In this section we are going to define our Input/Output signals.
The following figure indicates the inputs and outputs of our proposed peripheral.

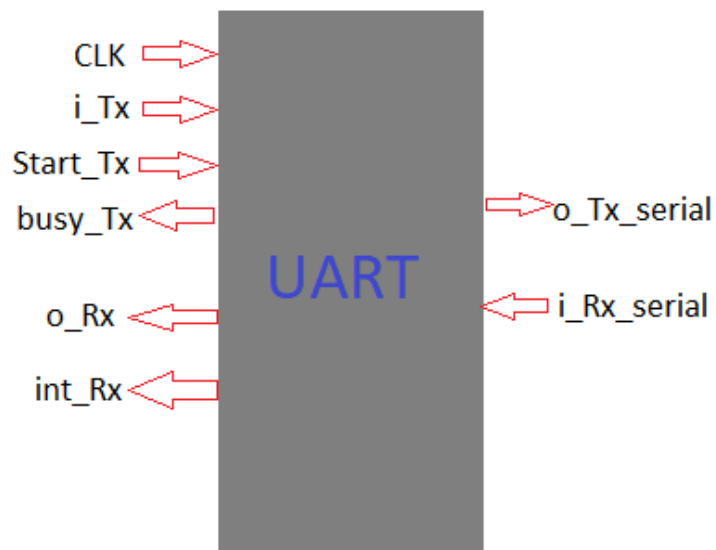


Figure 2.1 – Proposed UART inputs/outputs

The figure 2.1 represents the inputs and outputs signals that we used to create our UART device.

The following table define each signal, whether it is an input or an output, the number of bits of each signal and a brief description.

Signal	Mode	bits	Description
clk	in	1 bit	The system clock
i_Tx	in	8 bits	Data to transmit
start_Tx	in	1 bit	Starts off the transmission
busy_Tx	out	1 bit	Indicates whether the device is in transmission mode or not
o_Rx	out	8 bit	Received data
int_RX	out	1 bit	interruption that indicates the end of reception
o_Tx_serial	out	1 bit	Data serial transmission wire
i_Rx_serial	in	1 bit	Data serial reception wire

2.2 Peripheral State Machine

After identifying the system input/output signals, we define the finite state machine of our system. For both transmission and reception we can define 4 states:

- idle : The system is idle.
- starting : The peripheral is sending a starting bit or is receiving the starting bit.
- busy : Sending or receiving the data.
- done : Sending or receiving the stop bit.

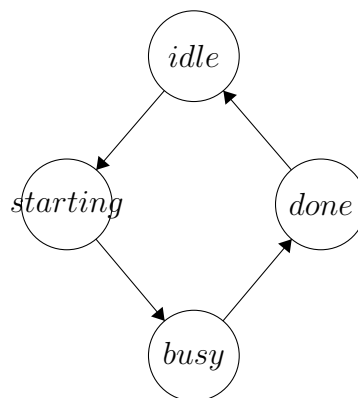


Figure 2.2 – Finite State Machine

Chapter 3

Test and Validation

In this chapter we will indicate the tests that we made to insure the peripheral functionality. We made two type of test :

- Manual tests (simulating and verifying).
- Automatic test (test bench).

3.1 Manual test

After developing the device, we test it using model sim by setting some signals to different values and follow the signals behavior.

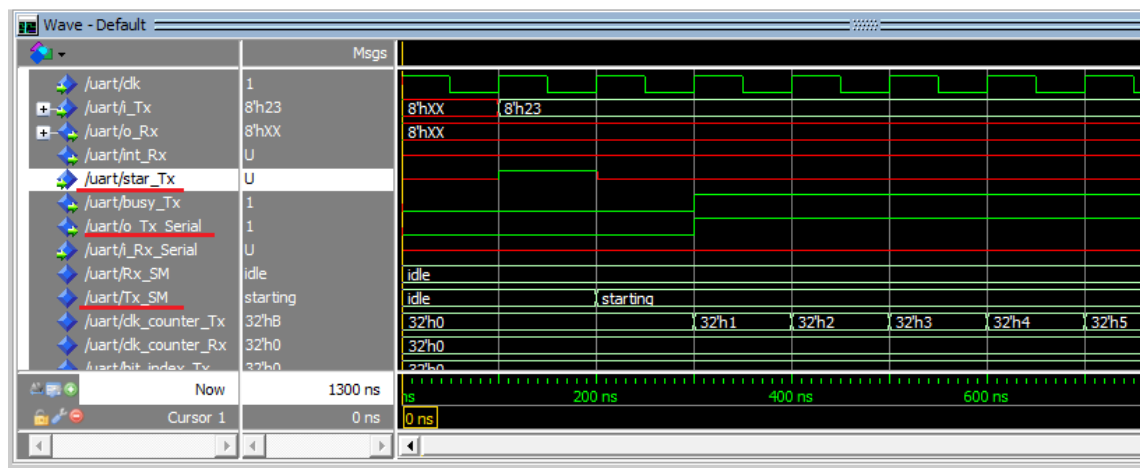


Figure 3.1 – manual test of data transmission

In the figure 3.1, we can see the signals behavior while changing and the Tx_state machine change from idle to starting .

3.2 Test bench

In addition to the manual test we created a test bench by writing a VHDL code as shown in the next figure:

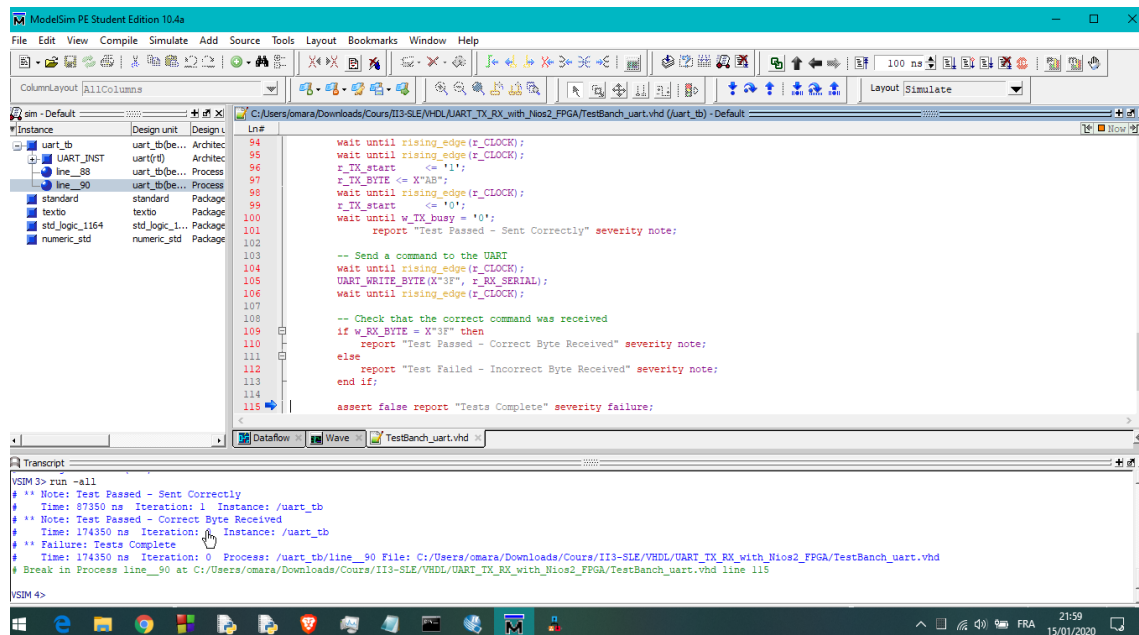


Figure 3.2 – manual Test bench

Using the test bench code, we simulated the transmission and the reception, and we verified that the peripheral is functioning correctly.

Chapter 4

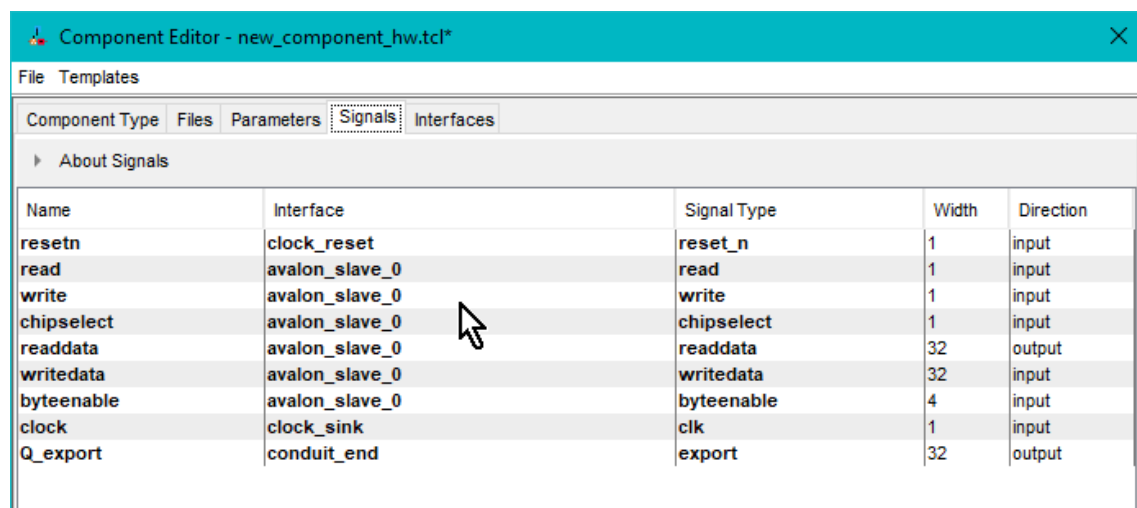
Integrating Peripheral with Nios2

Introduction

In this chapter we demonstrate that we managed to integrate our UART peripheral with Nios2 using Qsys

4.1 UART interfaces

First we started by adding a new component, which means adding the needed vhd files and adding the needed interfaces to ensure a communication throw an avalon interface.



Component Editor - new_component_hw.tcl*				
File Templates				
Component Type Files Parameters Signals Interfaces				
About Signals				
Name	Interface	Signal Type	Width	Direction
resetn	clock_reset	reset_n	1	input
read	avalon_slave_0	read	1	input
write	avalon_slave_0	write	1	input
chipselect	avalon_slave_0	chipselect	1	input
readdata	avalon_slave_0	readdata	32	output
writedata	avalon_slave_0	writedata	32	input
byteenable	avalon_slave_0	byteenable	4	input
clock	clock_sink	clk	1	input
Q_export	conduit_end	export	32	output

Figure 4.1 – Signals

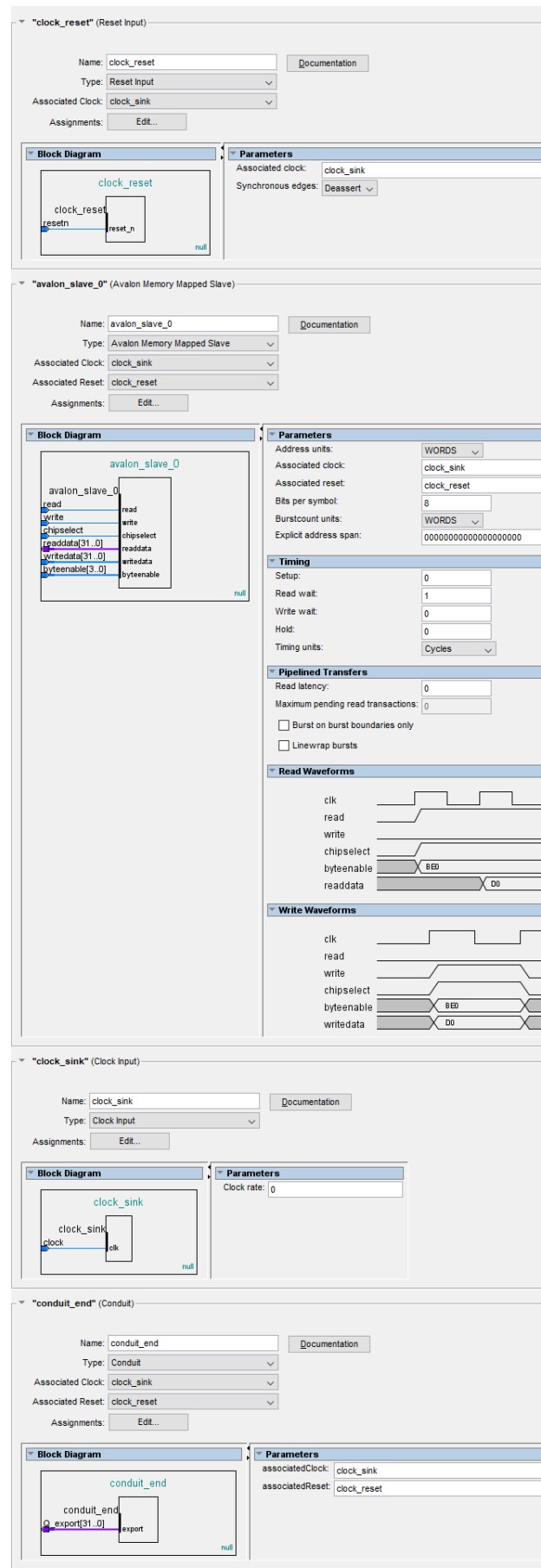


Figure 4.2 – manual Test bench

4.2 Linking the Peripheral with the Nios2

The following figure illustrates how to connect the UART peripheral with the other peripherals:

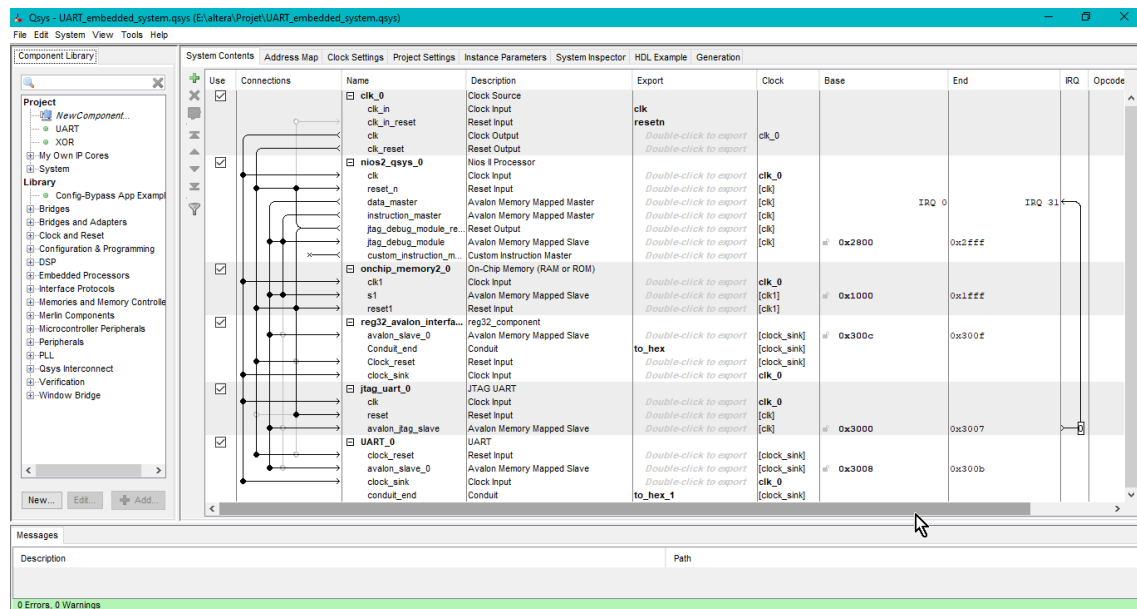


Figure 4.3 – System Contents

Netography

- [1] Making Qsys Components.
https://people.ece.cornell.edu/land/courses/ece5760/DE1_S0C/Making_Qsys_Components_15_0.pdf. Accessed: 1-1-2020.

[1]