***Intro to System-on-Chip Design Course***

**LAB 6**

**AHB UART Peripheral**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc83817516)

[1.1 Lab overview 1](#_Toc83817517)

[1.1.1 Hardware design and implementation: 1](#_Toc83817518)

[1.1.2 Software programming: 1](#_Toc83817519)

[1.1.3 Demonstrate the SoC: 1](#_Toc83817520)

[2 Learning Objectives 1](#_Toc83817521)

[3 Requirements 2](#_Toc83817522)

[4 Project files 2](#_Toc83817523)

[5 Hardware 3](#_Toc83817524)

[5.1 Overview of the SoC hardware 3](#_Toc83817525)

[5.2 UART Peripheral 3](#_Toc83817526)

[5.2.1 UART Peripheral Block diagram 3](#_Toc83817527)

[5.2.2 UART Peripheral Memory map 4](#_Toc83817528)

[6 Software 4](#_Toc83817529)

[6.1 Main code tasks 4](#_Toc83817530)

[7 Hardware Debugging 6](#_Toc83817531)

[7.1 On-chip debugging 6](#_Toc83817532)

[8 Extension work 6](#_Toc83817533)

[8.1 Extra tasks for this lab: 6](#_Toc83817534)

# Introduction

## Lab overview

In this lab, we will implement an AHB UART peripheral and write simple program for the processor to communicate with a PC or laptop. The steps to do this include:

### Hardware design and implementation:

The processor, bus interface, on-chip memory and peripheral hardware are written in Verilog and provided for you, with some modification/additions needed to make it work. The SoC will be implemented in an FPGA.

### Software programming:

The program targeted at the Cortex-M0 processor is written in assembly language and will be used to access the UART peripheral. The program is provided for you. You will need to compile it in Keil to generate a *code.hex* file which will be copied to the FPGA project directory.

### Demonstrate the SoC:

* Send/receive information to/from a host (PC or laptop).
* Analyze the behavior of the peripheral using an on-chip hardware debugging tool.

# Learning Objectives

* Implement a simple SoC which consist of Cortex-M0 processor, AHB-Lite bus and AHB peripherals (Program memory and LED, VGA, UART) on an FPGA.
* Modify and compile an assembly code to receive characters entered on the keyboard through UART which is then displayed on a VGA.

# Requirements

This lab requires the following hardware and software:

* **Hardware:**
  + **Diligent BASYS 3** FPGA board connected to computer via **MicroUSB cable.** A constraints file for this board is also provided.
* **Software**
  + Xilinx Vivado
  + Keil uVision
  + TeraTerm

# Project files

You will need the files from the previous lab along with the following files which are provided with this Lab:

|  |  |
| --- | --- |
| **File name** | **Description** |
| AHBUART.v | The top module of the UART peripheral, includes the AHB interface |
| baudgen.v | Generate system ticks for a fixed transmission baud rate, e.g., 19200 bps |
| uart\_rx.v | UART receiver; receives eight sequential bits and translates them to 8-bit parallel data |
| uart\_tx.v | UART transmitter; sends 8-bit data in sequential bits |
| fifo.v | A FIFO to buffer the data to be sent and data that has been received |

# Hardware

## Overview of the SoC hardware

The hardware components of the SoC include:

* An Arm Cortex-M0 microprocessor
* An AHB-Lite system bus
* Three AHB peripherals
  + A BRAM module
  + A VGA peripheral
  + A UART peripheral to interface with a host

Diagram

Description automatically generated

Figure :SoC Peripherals

## UART Peripheral

The UART peripheral is used to interface the AHB bus with other systems which receives and transmits serial data.

### UART Peripheral Block diagram

Diagram

Description automatically generated

**UART Peripheral Block Diagram**

* UART transmitter
  + Reads data (in byte) from the transmitter FIFO
  + Converts a single byte data to sequential bits
  + Sends bits to the tx pin, clocked in a fixed rate provided from the baud generator
* UART receiver
  + Receives the sequential bits from the rx pin using the clock generated from the baud generator
  + Reassembles the bits to a single byte
  + Writes the received byte to the receiver FIFO
* UART FIFO
  + Temporally buffers the data to be sent or the data that has been received

### UART Peripheral Memory map

The default memory map of the peripherals is listed below:

MEMORY MAP OF PERIPHERALS

|  |  |  |  |
| --- | --- | --- | --- |
| **Peripheral** | **Base address** | **End address** | **Size** |
| BRAM | 0x0000\_0000 | 0x00FF\_FFFF | 16MB |
| VGA | 0x5000\_0000 | 0x50FF\_FFFF | 16MB |
| UART | 0x5100\_0000 | 0x51FF\_FFFF | 16MB |

PERIPHERAL REGISTERS

|  |  |  |
| --- | --- | --- |
| **Register** | **Base address** | **Size** |
| Data | 0x5100\_0000 | 4 bytes |
| FIFO status | 0x5100\_0004 | 4 bytes |

* Data register

Used for both input and output data

* FIFO status register
  + Bit0: Rx FIFO empty

If empty, the processor cannot read from the FIFO.

* + Bit1: Tx FIFO full

If full, the processor must wait before writing to the FIFO.

# Software

## Main code tasks

The main code is written in assembly and should perform the following:

* Initialize the interrupt vector.
* Display a string (e.g., “TEST”) at the console region (same as previous lab).
* Plot four pixels at the four corners of the image region (same as previous lab).

Then, it should repeat the following:

* Wait until a character is received (the receive FIFO is not empty).
* Print the received text to both UART and VGA.

An example of the demo:

A picture containing text, monitor, electronics, indoor

Description automatically generated

Figure :Demo Example

# Hardware Debugging

## On-chip debugging

Use the on-chip debugging tool to sample and analyze the signals at run-time. Suggested signals are as follows.

Towards AHB bus:

* HADDR[31:0]
* HWDATA[31:0]
* HRDATA[31:0]
* HWRITE
* HREADY
* HSIZE[2:0]
* HTRANS[1:0]
* HRESP

Towards the UART:

* FIFO\_tx\_empty
* FIFO\_rx\_empty
* UART\_tx
* UART\_rx
* Tx\_data[7:0]
* Rx\_data[7:0]

# Extension work

## Extra tasks for this lab:

* Add configuration registers to the UART peripheral, whereby the processor can configure the peripheral by modifying its configuration registers, for example,
  + Change the UART baud rate.
* Add parity bit for the UART transmission.
* Send/receive files to/from a host using UART.