Lab Exercise 6: AHB UART Peripheral

# Overview

In this lab, we design and implement an AHB UART peripheral to communicate with a PC or laptop. The work includes:

* Hardware design and implementation:
* Design and implement the UART hardware mechanism to interface with a PC or laptop.
* Design and implement the AHB bus interface for the peripheral.
* Prototype the hardware onto a FPGA.
* Software programming:
* Program the Cortex-M0 processor; access the UART peripheral using assembly language.
* 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.

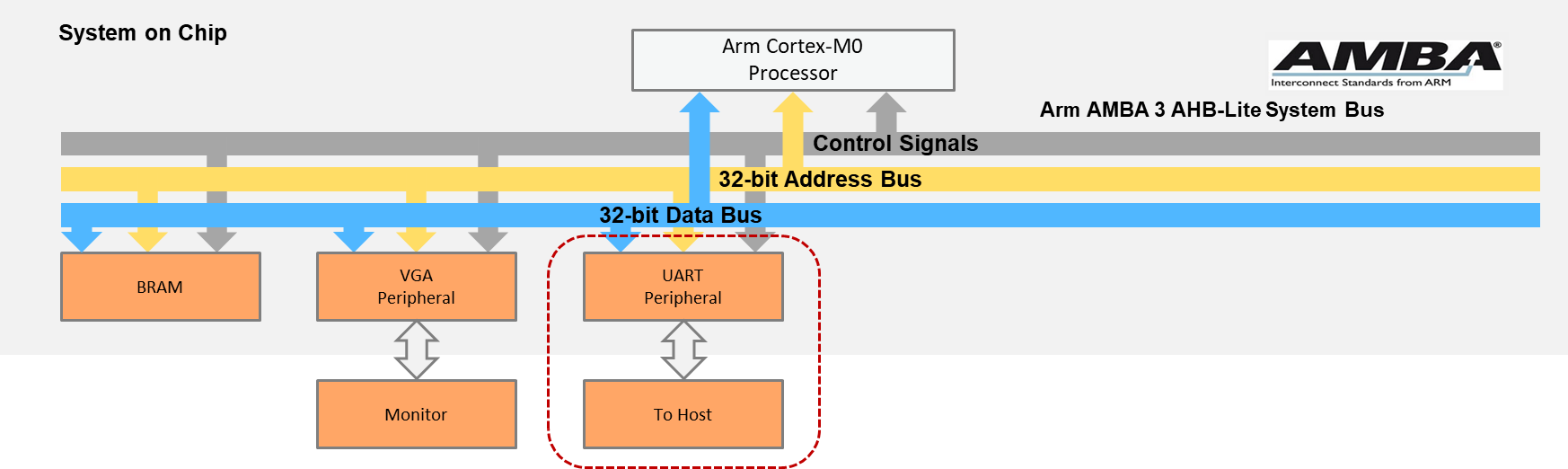
# Details

## Hardware

### system components

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



**SoC Peripherals**

### file structure

The files that will be used for the peripheral are listed below:

FILES USED BY VGA PERIPHERAL

|  |  |
| --- | --- |
| **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 |

AHBUART.v

baudgen.v

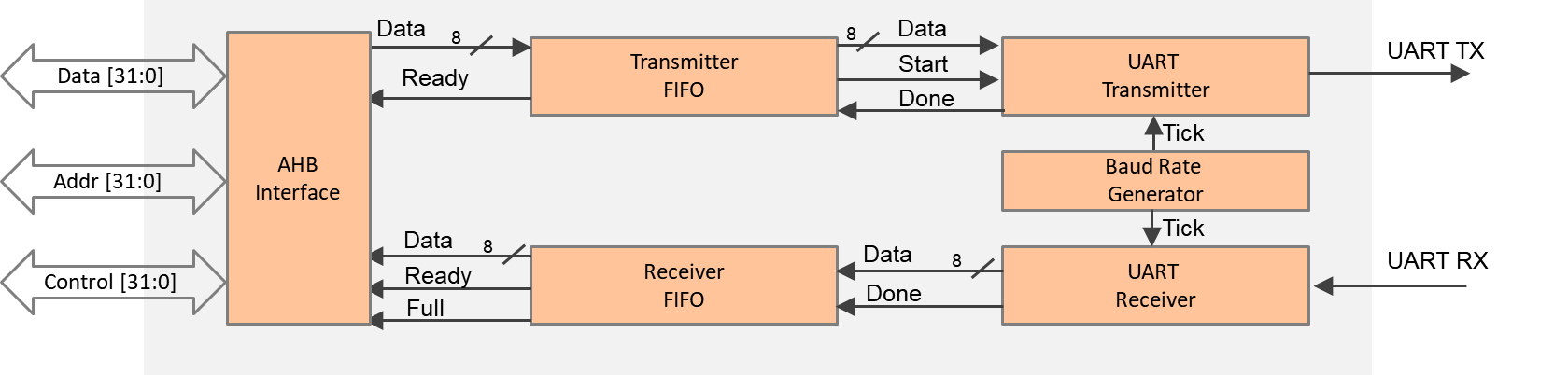
uart\_rx.v

uart\_tx.v

fifo.v

fifo.v

### block diagram



**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

### 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

The main code should be 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:



**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

Here are some extra things that you can do:

* 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.