Lab Exercise 5: AHB vga Peripheral

# Overview

In this lab, we need to design and implement an AHB VGA peripheral to display images onto a VGA monitor. Work includes:

* Hardware design and implementation:
* Design and implement the VGA hardware mechanism to interface with a VGA monitor.
* Design and implement the AHB bus interface for the peripheral.
* Prototype the hardware onto an FPGA.
* Software programming:
* Program the Cortex-M0 processor and access the VGA peripheral using assembly language.
* Demonstrate the SoC:
* Display texts and images on a VGA display.
* Analyze the behavior of the peripheral using an on-chip hardware debugging tool.

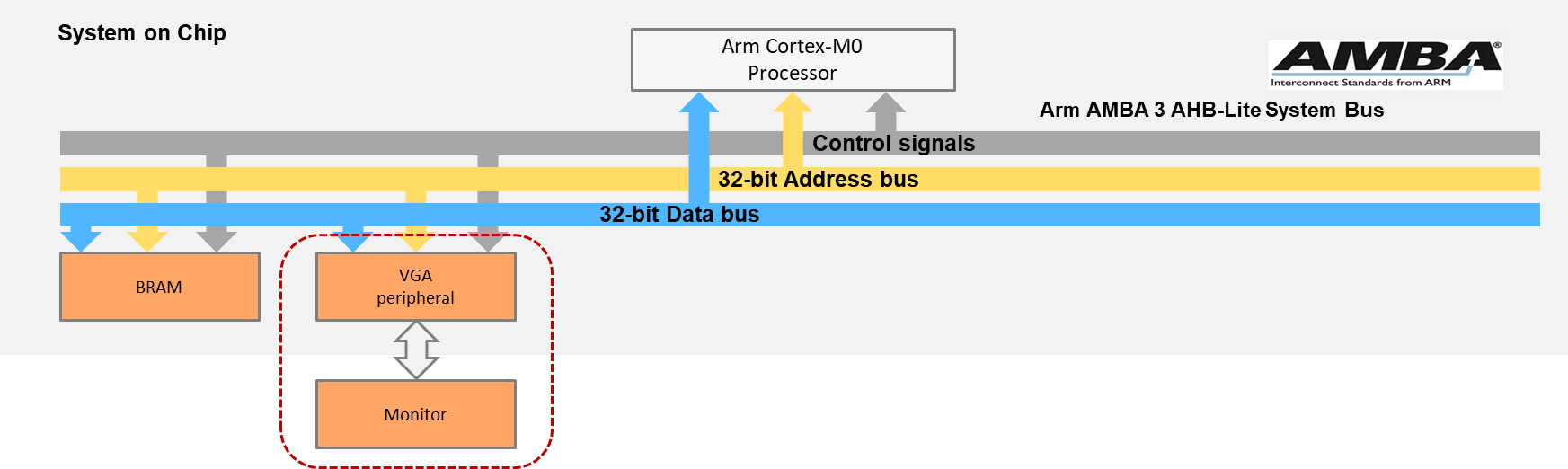
# Details

## Hardware

### system components

The hardware components of the SoC include:

* An Arm Cortex-M0 microprocessor
* An AHB-Lite system bus
* Two AHB peripherals
  + A VGA peripheral to interface with a VGA monitor



**SoC Peripherals**

### file structure

The VGA peripheral is more complicated than the other modules. This is because we separate the screen into two regions, and the text region uses hardware logic to dynamically generate fonts’ pixels depending on the input character. Nevertheless, some of the modules, such as font ROM, have been given to you and you can directly use them in your design.

The files used in this module are listed below:

FILES USED BY VGA PERIPHERAL

|  |  |
| --- | --- |
| **File name** | **Description** |
| AHBVGASYS.v | The top module of the VGA, includes the AHB interface |
| vga\_image.v | The frame buffer in the image region |
| vga\_console.v | Used to display characters in the console region |
| vga\_sync.v | Used to generate VGA synchronization signals |
| font\_rom.v | The ROM used to store the pixels of a character |
| dual\_port\_ram\_sync.v | A dual-port synchronized on-chip RAM |
| counter.v | A generic counter |

AHBVGASYS.v

vga\_console.v

vga\_image.v

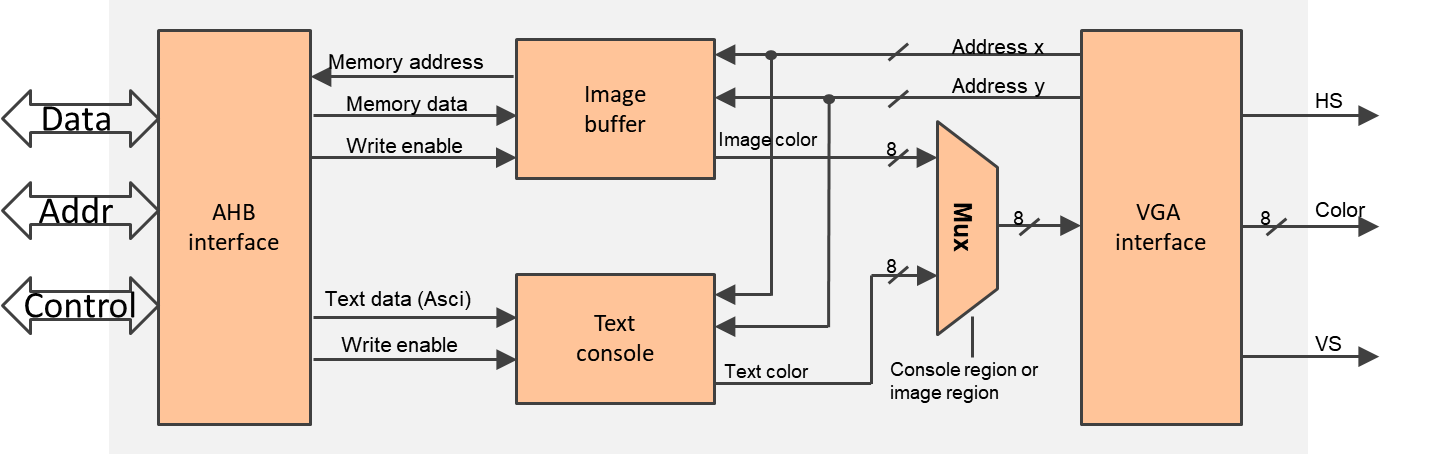
vga\_sync.v

font\_rom.v

dual\_port\_ram\_sync.v

counter.v

### block diagram



**VGA Peripheral Block Diagram**

* VGA interface
  + Generates synchronization signals to the VGA port
  + Is directly connected to external pins of the VGA port
  + Outputs the address of the current pixel
* Image buffer
  + Stores the color information of all pixels in the image region
  + Is implemented on a dual-port memory
* Text console
  + Displays texts in the text region
  + Is implemented on hardware logics

### memory map

The following table shows the suggested memory map for peripherals and registers:

MEMORY MAP OF PERIPHERALS

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

PERIPHERAL REGISTERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Register** | **Base address** | **End address** | **Size** |
| Console text | 0x5000\_0000 | 0x5000\_0000 | 4 byte |
| Image buffer | 0x5000\_0004 | 0x50FF\_FFFF | (16M-4) byte |

* Console text: 1 word (4 byte) space to print a character
* Image buffer: the rest of the space is used to store pixels in the image region.

## software

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

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

An example of the demo:



**Demo Example**

# HARDWARE debugging

## on-chip debugging

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

Towards AHB bus:

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

Towards the VGA interface:

* Horizontal\_Sync
* Vertical\_Sync
* VGA\_Color
* Address\_x
* Address\_y

# Extension work

Here are some extra things you can do:

* Draw lines, rectangles, or even a picture at the image region.
* Add configuration registers to the VGA peripheral, whereby the processor can configure the peripheral by modifying its configuration registers, for example
  + Change the image resolution.
  + Adjust the size of the text region and the image region.
* Display live videos at the image region.
* Change languages at the console region.