CSEE 4840
EMBEDDED SYSTEMS

"WhaC-A-Mole" Project Design

Spring 2016

Group Members

Astha Agrawal	*aa375*
Jai Sharma	*js4773*
Aditya Bagri	*aab2234*
Georgios Charitos	*gc2662*

Due date: 3-24-16

1) System Block Diagram

The highest level in our design will be based on the Processor System Block Diagram(Fig1.1). Most Processors are based on this particular system block.

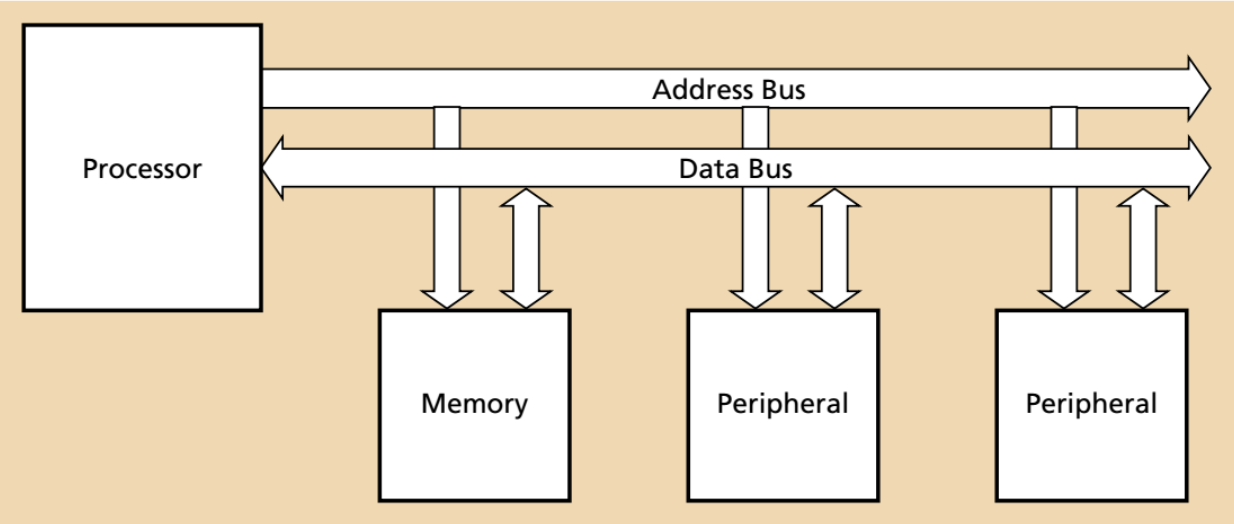

Figure 1.1: *Processor System Block Diagram*

Below we outline in detail the blocks our design includes :

- **SRAM Memory**: The memory will be provided by Linux so we do not need to implement it like we did in Lab1.
- **SRAM Controller**:The memory controller is also included in Linux.
- **Nios II Processor**: The SoCKit board provides the Altera Nios II processor.
- **Avalon Bus**:SoCKit board includes the Altera Avalon bus which we will use as Data and Address bus.
- **VGA Display**: The monitor for our project as a peripheral
- **VGA Display Controller** : We will have to create a controller for the VGA Display on the screen.
- **Mouse/Controller**: A mouse or a videogame controller will be used as a second peripheral.
- **Audio Output**: An audio Output will be required for the different sounds of the game.
- **Audio Controller**: The audio output will be controlled by its separate controller which we will design.

All the components and the way they are connected are shown in Fig1.2.

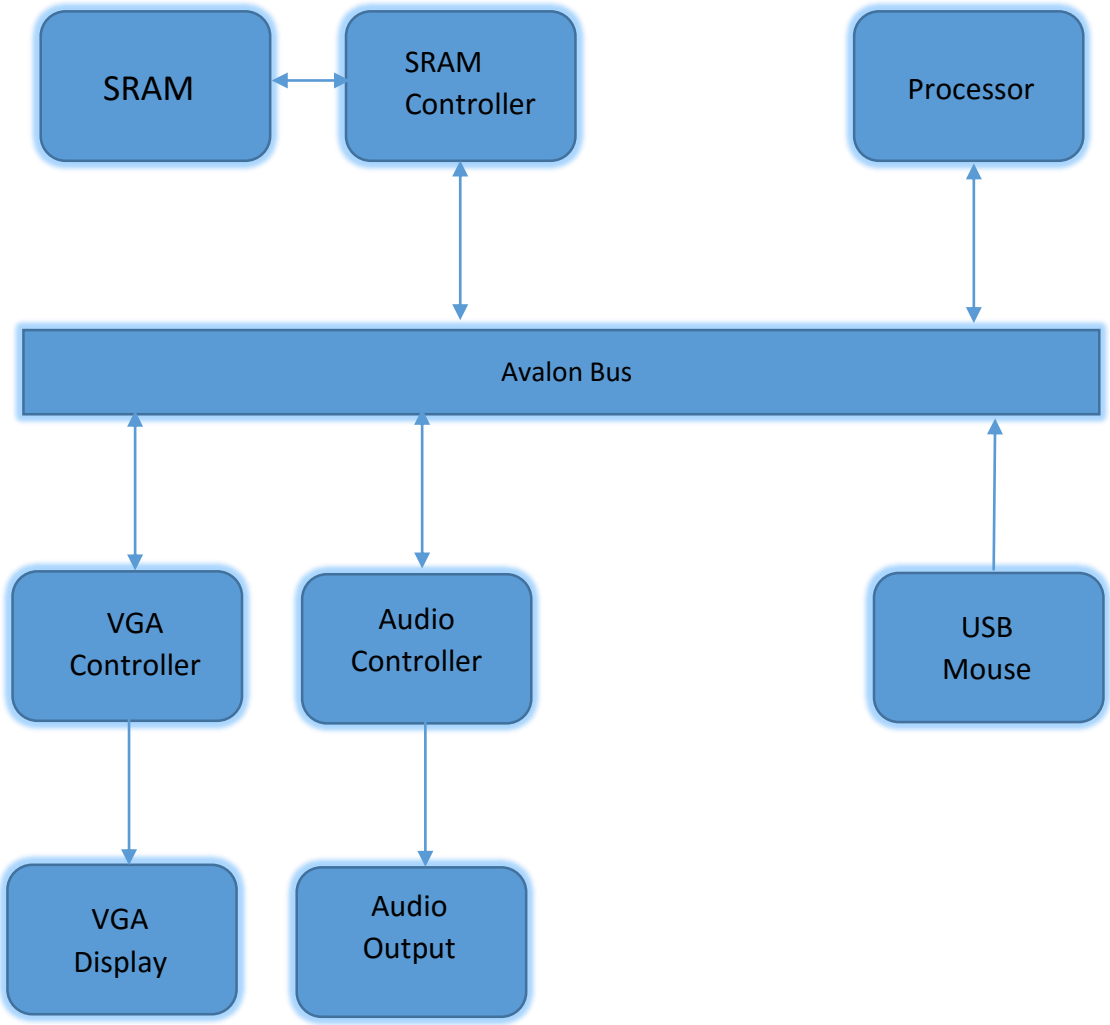

Figure 1.2:*Project Block Diagram*

2) Hardware Components

Processor & SRAM

VGA controller reads the variable that represents the position and shape of the bat that is used to hit the moles. The audio controller reads the variable that represents the current playing audio.

Variables used that are stored in registers.

Variable	Function
Mouse	Read
Score	Write
Level	Write
Bat	Write
Mole	Write
Time	Write
Sound Track	Write

Table 2.1: *Processor Variables*

SRAM: This is the memory of the whole system. It is hosted on the Linux environment. It is the intermedia between the processor and the peripherals. SRAM stores the status of the game. Audio data is stored in the SRAM.

The Processor: This is used to either read or write data to the SRAM via the Avalon Bus.

VGA Display Controller

The VGA controller will perform a function similar to the VGA_LED_Emulator we used in Lab3. It will send and receive a number of variables through input ports to and from the memory. The output ports determine the image on the 640x480 display. Table 2.2 names and describes the ports for these variables.

Port Name	IN/OUT	Function
LEVEL	IN	Passes information about the game level.The background also is asociated with the level since it will have a different colour for different levels
BAT	IN	The bat represents the cursor's position on the screen
TIME	IN	Timer starting at 3min for each level
SCORE	IN	Score depending on the number of stikes
MOLE	IN	Mole's position on the screen
VGA_CLK	OUT	VGA Display CLK
VGA_SYNC_n	OUT	VGA Synchronizer
VGA_HS	OUT	VGA Horizontal Synchroniser
VGA_VS	OUT	VGA Vertical Synchroniser
VGA_BLANK_n	OUT	VGA Display Blank Signal
VGA_R	OUT	VGA pixel red
VGA_G	OUT	VGA pixel green
VGA_B	OUT	VGA pixel blue

Table 2.2: *VGA Controler Ports*

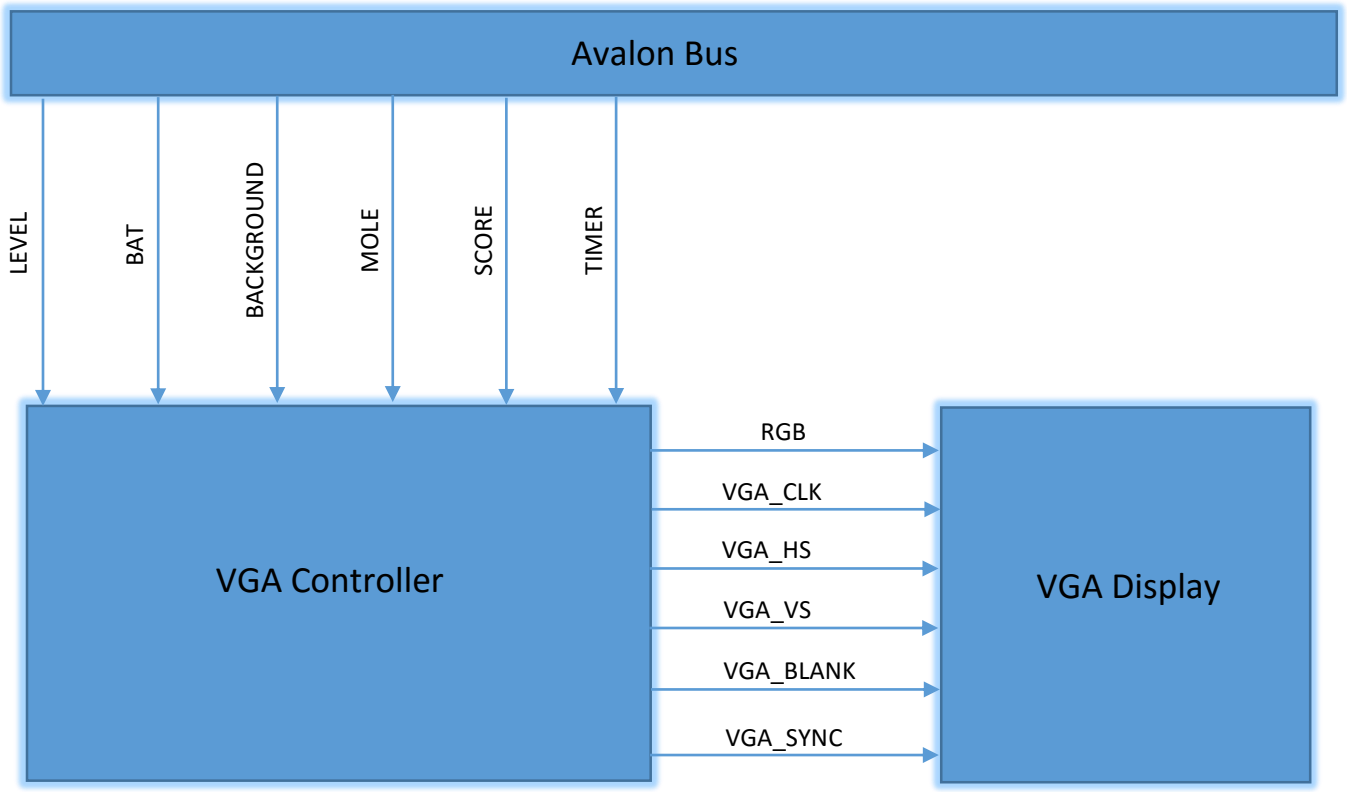

Figure 2.1: *VGA Display Controller Block with ports*

Audio Controller

The Audio:

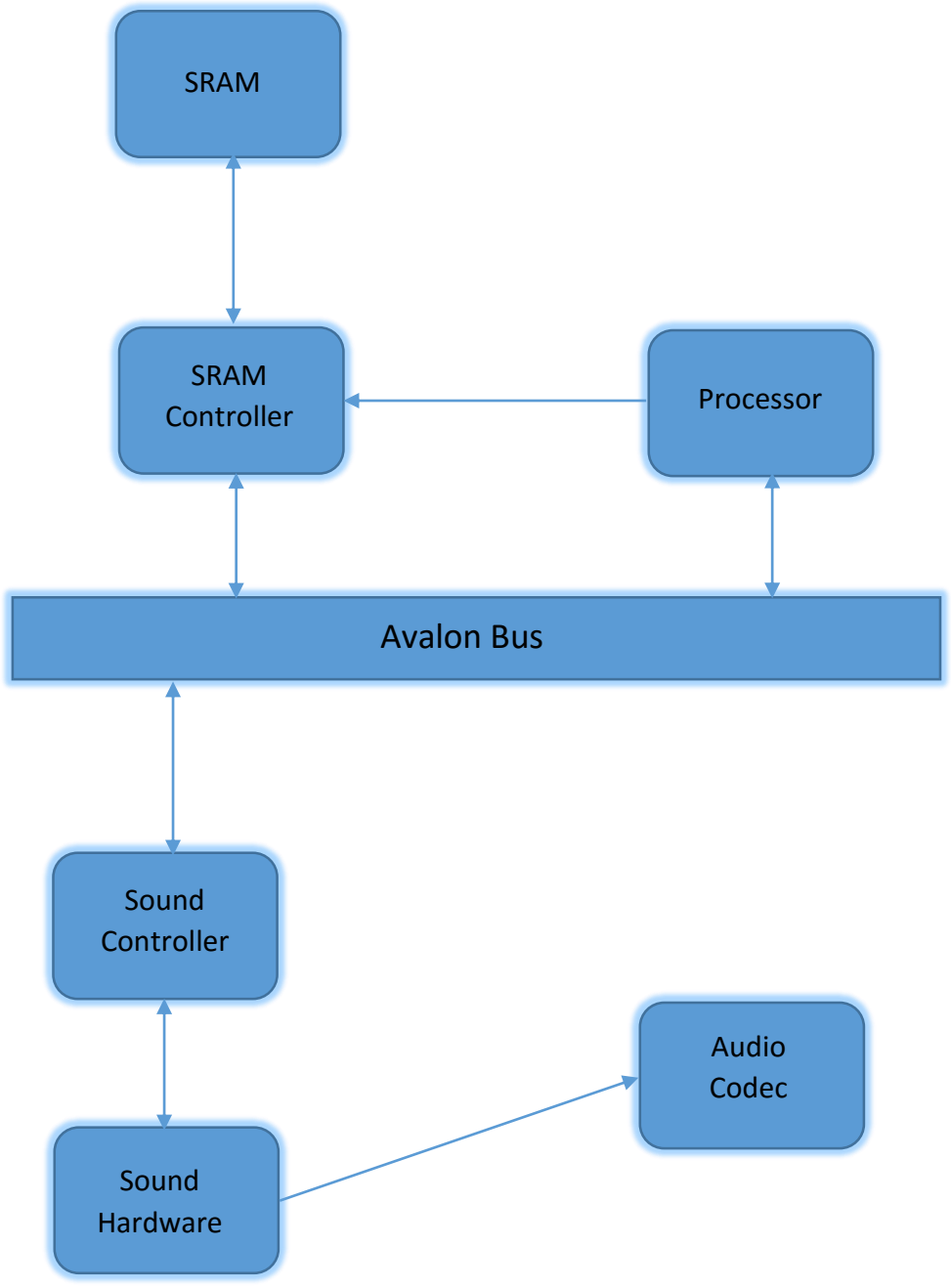

Figure 2.2:*Audio Controller Block ports*

- The processor controls SRAM to transfer audio files to audio peripherals.
- Audio Codec is the audio function chip on the SoCKit board.
- The Sound Hardware consists of the SystemVerilog code that is used to control the chip by providing it clocks and transferring audio data into it.
- The Sound controller maps the audio part into the bus-memory-processor system.
- The processor receives the C code that decides when to play what audio.

3) Software Components

VGA DISPLAY DRIVER

The project's software will communicate with the VGA controller through the VGA display driver which will be compiled into the kernel. In Fig3.1 we show the Linux Operating System Structure:

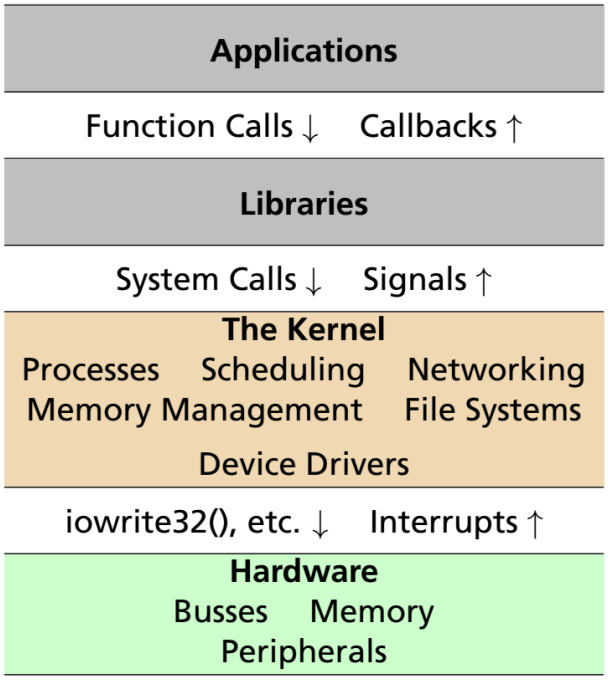

Figure 3.1: *Linux Operating System Structure*

The top level C program will communicate with the VGA driver through ioctl requests. The driver will call functions to "talk" to the VGA controller device. Interrupts from other peripherals like the mouse are also handled by the driver.

USB Mouse

We will create a C program in order to create a path from the mouse peripheral to the processor. The software will use libusb and threads in order to send a data package to the processor. A similar work was done in Lab2 but it involved communication through the keyboard. There is a simple data exchange between the PC and the mouse. The PC asks the mouse if there is new data available each time, the mouse will send the data if it is available, otherwise it will send a NACK (No Acknowledge) to tell the PC that there is no data available. Data sent to the PC have the structure below in Fig 3.2.

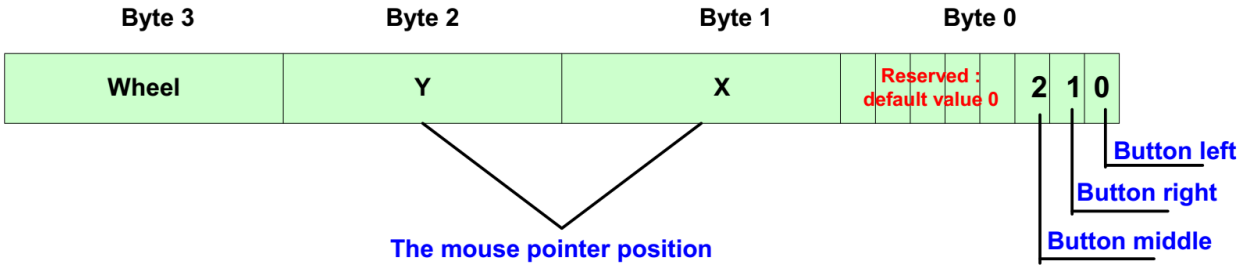

Figure 3.2: *Mouse Date Packet sent via USB*

- **Byte 3** has information about the wheel which is not used in our application.
- **Byte 2** is the y position of the cursor.
- **Byte 1** is the x position of the cursor.
- **Byte 0:**
  - Bits 0-2 are the left, right and middle button respectively.
  - Bits 7-3 are all 0.

Audio Controller

The sound data are arrays of 16-bit sample values, if sample values are aligned in sequence, they will appear as sound wave. Audio codec makes sounds by aligning sample values in sequence and convert them into analog signals.

There are two types of sounds that we have to deal with: the background music and the sound effects for particular situation such as hit/miss a mole, mole popping in/out and so on. Since the audio to play is of 16 bit we cannot let that register overflow and so we have to add these two types in such a way that the scale of audio to play remains at 1. How we achieve this is as: when no sound effects are triggered, the audio to play is simply the background music playing at

volume scale 1. When sound effects are triggered, we cannot simply add the sound effects to the background music and so we add them in a weighted fashion wherein the sound effects are multiplied by a fraction that is larger than the fraction multiplied to the background music, and such that the scaled sum is 1.

REFERENCES

[1] *ATMEL*: *AVR270: USB Mouse Demonstration*