WMT - A “Whac-A-Mole” like game

CSEE 4840 Embedded System Design

March 2013

Si Wang: sw2778@columbia.edu

Shuting Yang: sy2489@columbia.edu

Lingchuan Mei: lm2908@columbia.edu

Jian Lu: jl3946@columbia.edu

I. High level overview of the project

The main goal of this project is to deliver a "Whac-A-Mole" like game. There are two output devices, which are VGA display and audio output. The user will be using a mouse as an input device to move the hammer across the screen to hit the moles. This design report will provide the general idea of the project and the details of the implementation.

In the game, every round will last three minutes. The user get directed to the higher level if he or she misses less than 5 moles in the game, otherwise the game is finished. Every time the mole is hit, 5 credits are added to the user's score. The corresponding music is played while the user hits or misses the mole. This game will also keep track of the highest score of the users. Once the user breaks the score record, the audio will be playing cheering music.

II. A high-level block diagram of all hardware components and how they interact.

a. Overall block diagram

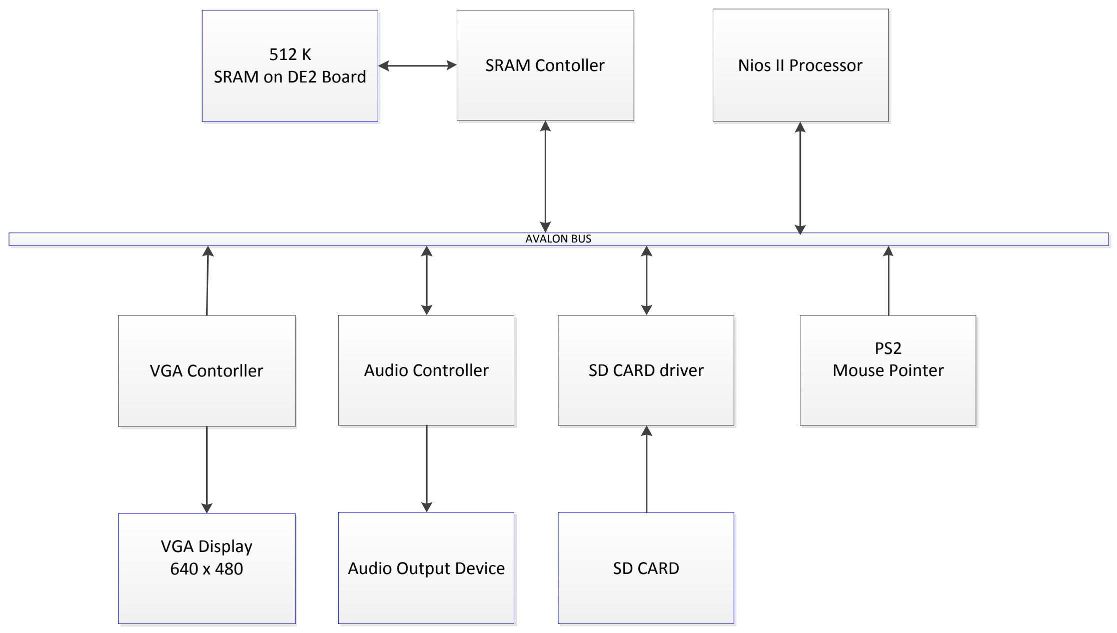


Figure 1 The overall block diagram of the project

b. Interaction between all components

1. Nios II & SRAM

SRAM is here acting as the memory of the whole system. So every interaction between Nios II and all the peripherals are using memory as the intermedia.

Nios II will read Mouse input and the status of the game stored in the SRAM. While reading from the SRAM, Nios II also writes data into the memory where peripherals read their input variables. For example, VGA controller will read the variable which represents the position and shape of the hammer. Audio controller will read the variable which represent the current playing sound.

Below is the list of the variables that are stored in the memory

Variable	Nios II operation
Mouse(3 bytes)	read
SD card data	read
Level(1 byte)	write
Hammer(3 bytes)	write
Background(1 byte)	write
Mole(3 bytes)	write
Score(1 byte)	write
Timer(2 bytes)	write
Sound_track_number(1 byte)	write
Data into the Audio controller(unknown)	write
SD_sound_play(1 byte)	write

2. SRAM and Audio controller
   Every clk cycle, Audio controller will read the audio data stored in the SRAM and play it using the audio output device
3. SRAM and SD controller
   Every clk cycle, SD will read the SD_sound_play variable stored in the SRAM. If it’s 1 then, the SD card driver will output the audio file according to the Sound_track_number.
4. SRAM and VGA controller
5. SRAM and Mouse pointer

III. An analysis of the memory requirements of the system and each piece and in which memory you intend to house it.

IV. Critical path

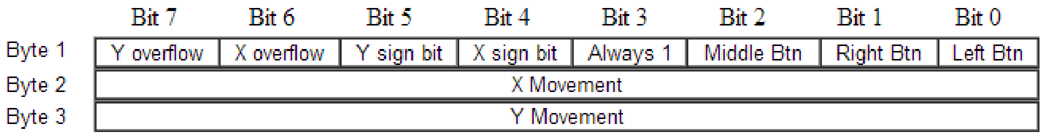
V. Detailed discussion of each peripheral

a. Nios II processor

The Nios II processor will read data will read data or write data to the SRAM through Avalon bus and SRAM controller.

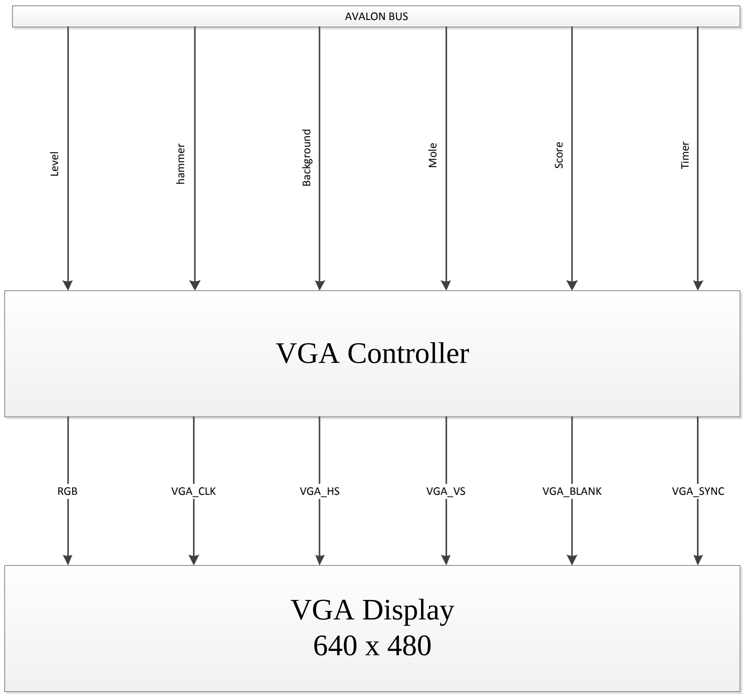
b. Mouse pointer

The mouse sends out a 3 byte data package at a time. The last 3 bits of the signal the first byte shows which button was pressed in during the last operation. The second byte represents the x-axis increment/decrement since last data transmission. The third byte represents the y-axis increment/decrement since the last data transmission. Both of them are nine bit counters where the sign bit is stored in Byte 1. The 9 bit two’s complement number will record the movement since last time. The value of the two counters range from -255 to 255. If the increment/decrement has exceeded the range, the overflow bits in Byte 1 will be set.



	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

c. VGA controller & VGA display



The VGA controller controls all the variables that determines the image on the 640 x 480 display. Here, the VGA has the following inputs and output in our system.

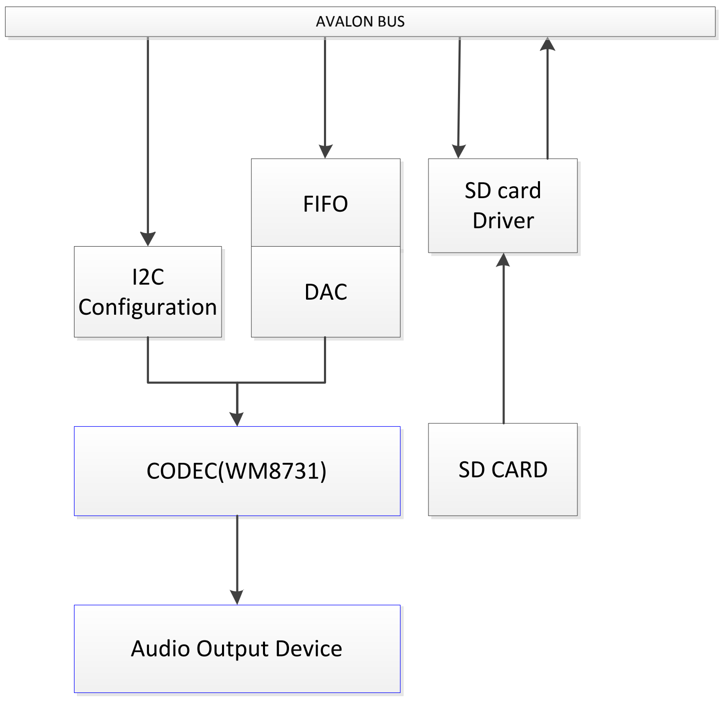
Port name	Input/Output	Description
Level	input	Current level of the game. Also determines the background of the game. Each level has a different background.
Hammer	input	Contains two pieces of information. One is the position of the hammer; One is the shape of the hammer(normal/hit)
Mole	input	Contains two pieces of information. The position of the moles and the

		shape of the moles(out/hit)
Score	input	Used for display the total score on the display
Timer	input	Used for display the count-down of the 3 minutes game period
VGA_CLK	output	The clock of the VGA display.
VGA_HS	output	The horizontal sync signal of the VGA display
VGA_VS	output	The vertical sync signal of the VGA display
VGA_BLANK	output	The blank signal of the VGA display
VGA_SYNC	output	
RGB	output	The color of the current pixel

d. SRAM controller & 512k SRAM on board

Signal name	Input/Output	Description
SRAM_DQ	Input and output	The data transfer between SRAM controller and SRAM on the de2 board
SRAM_ADDR	output	The operation target address
SRAM_UB_N, SRAM_LB_N	output	--
SRAM_UB_N, SRAM_LB_N	output	--
SRAM_OE_N	output	--

e. Audio controller & SD card driver



The audio feature of this game is designed to play fun music during the game, victory music when you passed a level, and sound effects of the hammer hitting the moles.

The audios are read from SD card and received by audio controller via the Avalon Bus. The controller is mainly consists of the WM8731 audio CODEC, and the interface between the CODEC and the FPGA. Thus to implement this, we need a good understanding of the WM8731 principles. There are two functions need to be realized in the interface design. Firstly, we need to set the configurations of WM8731 according to the sampling rate and the chip clock frequency, which can be sent from the FPGA to the WM8731 CODEC through an I2C interface. Secondly, certain local clock frequency will be generated according to the configuration, and data flow will be buffered in a FIFO, feed into the CODEC, this will be realized by a DAC controller. At last, the CODEC will send data to the audio device to produce audio output. This part may require a lot of work.

The game music may require more than 512k storage so we will be using a SD card to store the music. The use of SD card will require SD card driver and FAT16 file system driver, which will be realized by software. With the SD card drivers, the data can be accessible by audio controller through Avalon Bus.

VGA controller:
SRAM controller: SRAM is the interface between Avalon Bus and the 512K SRAM on the DE2 board. All the data need to be written into the SRAM or read from the SRAM will be controlled by the SRAM controller. Each operation will

be either read or write, and the SRAM controller will pass the read or write signal along with the address into the SRAM.