Mudd Adventure

A 3D Raycasting Game

CSEE 4840 Embedded Systems

Project Design

3/27/2014

Mingrui Xu(mx2151)
Wei Cao (wc2467)
Bowen Dang (bd2384)
Shijie Hu (sh3251)

Table of Contents

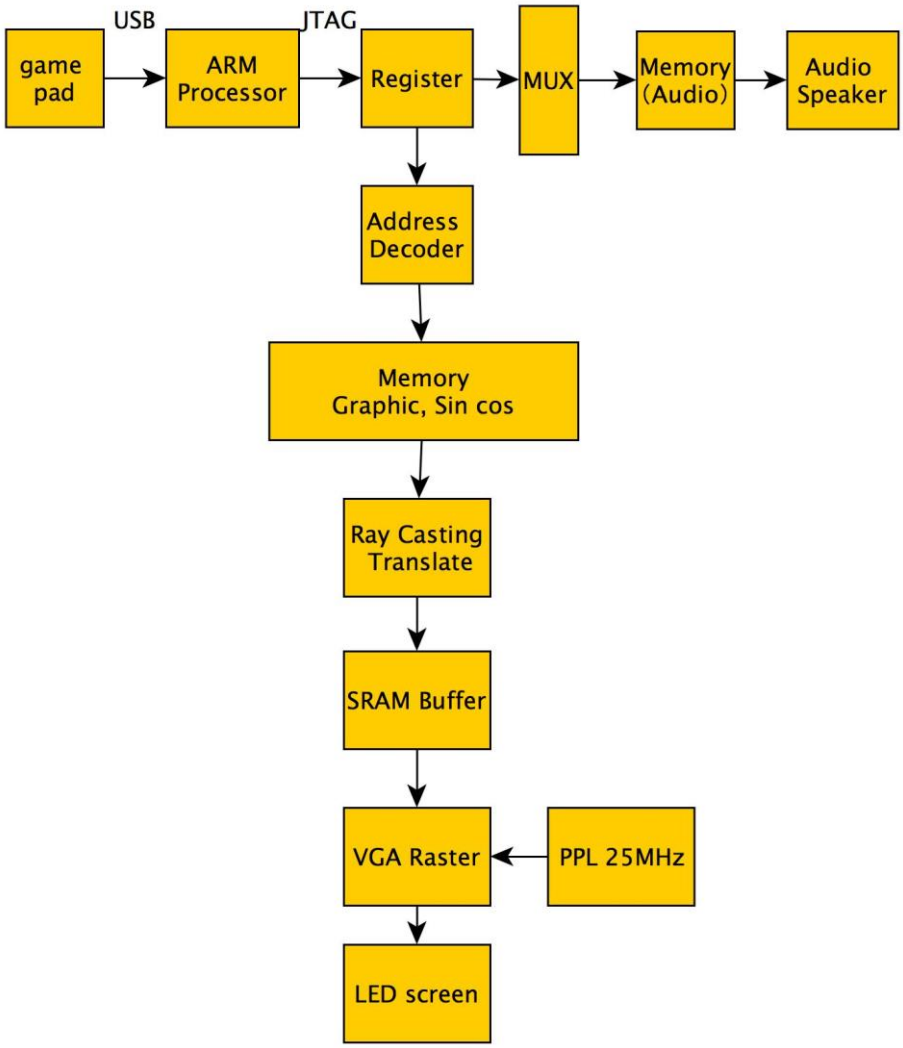
Introduction

Our project will design a virtual 3D adventure game by using Raycasting technique. A gamepad will be the game controller as a input. The 3D world will be displayed on the VGA output. Also the music or even some sound effects will also sent out from the output speaker during the game.

Essentially, hardware implementation is the most important and challenging part as well as the hardware-software communication. The hardware part will realize the 3d rendering, data processing and calculation, VGA display, sound output and input controller. Software will simply update the position information of the player in the virtual map and send the updated data to the hardware part.

Hardware

The overall block is showed in figure 1

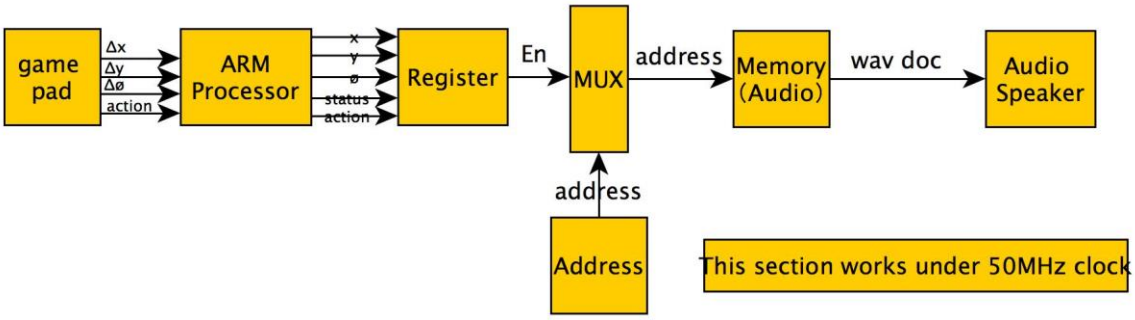


Input peripheral is a game pad and output peripherals are audio speaker and LED screen.

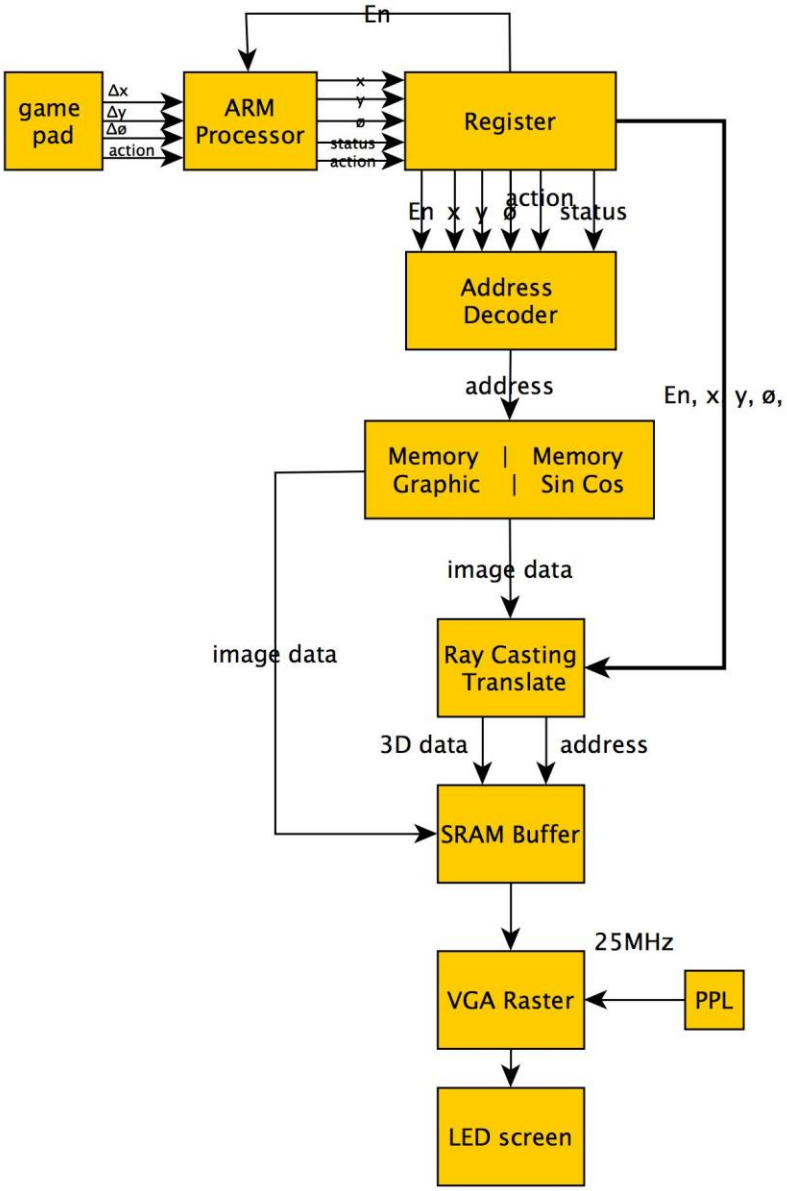
Game pad communicate with ARM processor under USB 2.0 protocol, and ARM communicate with FPGA (the main register) under JTAG protocol.

According to the flow chart above, the game pad output position shift informations, which are send to ARM processor, then ARM send current position information and other control information to FPGA main register.

Audio Section



Graphic Section



X,Y and ø are used to build the 3D map. En signal selects between explore mode and fighting mode, referring to separated music and scenes. Also the feedback of En will decide how the controller works. There will be a delay between the scene change. Status signal refers to the attacks and hp of player, shown in the screen as 2D digits. Action signal decides what to display in the screen when the player will act when facing the enemy in the game.

Memory

SDRAM provided is 2×256K×16bits storage space.

Memory is divided into three main parts.

1. Graphic Spirits part, which include 2D Graphic image data.
   We choose 24 bits for a single pixel's RGB value. So every pixel will have a 3byte RGB data.
   We choose five 64*64 pixels tile for display. The storage space required would be
   Storage Space =64 ×64 pixels ×5 × 24bits/pixel= 61,440 byte ≈ 61K
   The VGA LED screen has 480×640=307200 pixels≈3.1M pixels
   So the framebuffer required storage space would be
   3.1M×24bits/8 bits per byte≈ 10M byte

2. sin or cos look up table. This table is used for ray casting transform.
   For ray casting module, hardware cannot calculate sinusoids value. We build a look up table for store the sinusoid value.

   total for sinusoids = 10×2 × 360 angles /4 × 4 bytes each = 7230bytes ≈7M

3. Audio document. This part contains two waveform audio documents which showed be played during the game. The two waveform documents are small and the total storage space is enough for all data.

VGA

VGA will display 3D graphics on LED screen by drawing one pixel after the other. At each drawing, VGA controller will read column and row index (address) and RGB data from the framebuffer. When the last pixel in that row is drawn, the pointer will set to be zero and draw the following line.

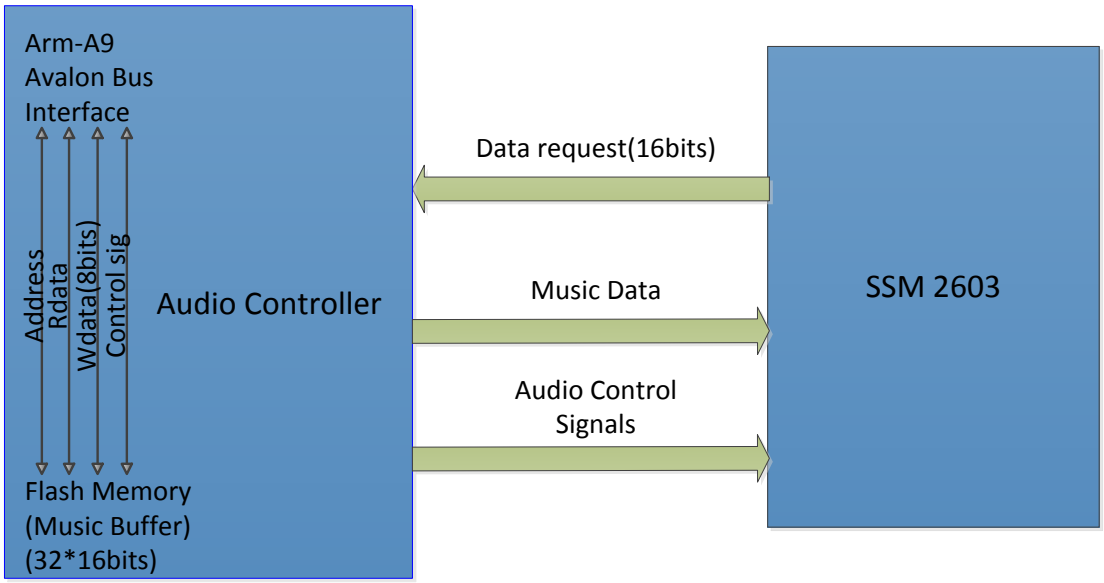
VGA is supposed to work under 25.175MHz while the board provides a 50MHz clock. So it is proper to use PPL to generate a 25MHz clock for VGA.

Timing

FPGA parts works under 50MHz clock. The VGA pixel scanning frequency is 25MHz, and frame frequency should be no less than 25Hz.
One Frame time/# of pixels= 1/25/(480×640)=1.3×10(-7) s→7.69MHz
3D ray casting block works under 50MHz clock, so every pixel has about 6 clock periods to do transform.

Audio output implementation

We use the audio controller and the analog SSM 2603 codec to implement the background music and sound effect output. The audio controller sends the interrupt signal and gets data from the Avalon bus. The simple schematic diagram is as follows:



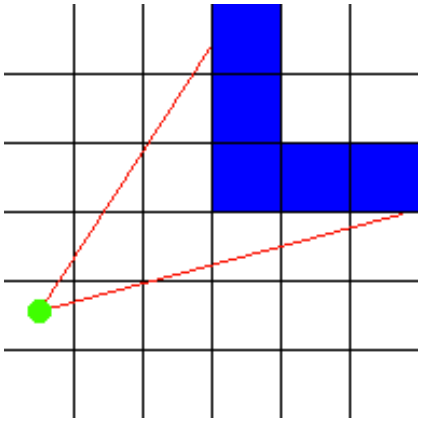
The buffer size is 32 * 16 bits. In the diagram, Address is the write address controlled by Arm-A9. Rdata and Wdata interface get data from and to Arm-A9. The control sig sends control signals from Arm-A9 to flash memory. The music data sends background music to SSM 2603 codec. When one music ends, the codec will ask for next one through data request. Sound effect audios are stored in on-chip block RAMs. The sound effect can play at the same time with background music controlling by Arm-A9. After the sound effect finished, a signal is sent and does not have effect on the playing of background music.

Software

Our project is basically a simple 3D shooting adventure game and the Raycasting is the key part of our 3D display rendering technique. This technique is used to create a 3D perspective on the monitor by utilizing a 2D map. So for the software aspect, making the map and controlling and updating the position of the player in the map are the main parts. Also, the software code will perform the Raycasting calculation algorithm and the details are explained in the next paragraph.

The basic idea of Raycasting technique is as follows: the map is a 2D square grid, and each square can either be 0 which means the place is vacate and the player is free to walk through, or a positive value which means the square is occupied by a wall.

For each x-coordinate value of the monitor screen, a ray will be sent out starting at the player location and with a direction that depends on both the player's looking direction, and the x-coordinate of the screen. Then, let this ray move forward on the 2D map, until it hits a map square that is a wall. When it hits a wall, the distance between the hit point and the player position will be calculated, and use this distance value to calculate how high this wall has to be drawn on the screen to make the 3D vision. The wall is higher when it is closer. If it is far, it should be relatively small. These are all 2D calculations.



The image above shows the overview of the process. The green dot is the location of the player. Two red lines are rays. Here, both rays hit the walls so their lengths will be saved for the 3D rendering calculation.

In the software code, the player position will be updated by receiving the signals from the gamepad controller. The motions include forward, backward, turning left

and right. When the software gets the motion command from the gamepad, the new distance calculations will be activated in the code after locating the new player location on the map. And the new dataset will be send to the corresponding SRAM part. Then new 3D will be showed out via VGA to the screen.

Taking the player position as the input, the whole Raycasting software code performs a loop to update the data every time according to the clock rate. When the Raycasting loop is done each time, the time duration of the current and previous frame can be calculated as well as the FPS value. The ARM Cortex A9 processor we use here can deliver devices capable of over 1GHz clock frequency. So the performance should be great. So actually we can just set the frame time as a constant value to determine the speed of moving or rotating and the FPS calculation will be unnecessary.

One more thing is the shooting command. As a shooting game, this is a key part. First we need to put enemies on the map as our targets. The enemies are called “sprites”. They are quite separate from the reycasting 3D rendering but their positions will still be stored in the software code. When the software receives the shooting command, the state will be updated if one target is hit. Then the signal will be sent to the hardware to update the display of the screen.

If everything goes well, our game will be pretty much similar as the famous game Wolfenstein 3D.

Milestone

April 1 (milestone 1)

Implement Raycasting algorithm on FPGA

Finish Drive for LED screen

Enable to read and write FPGA SDRAM

--write and read a 2D image from FPGA SDRAM and show it on LED screen.

April 8

Implement 3D transfer algorithm, read a 2D image from FPGA SDRAM, transfer it into 3D image, and show it on appointed position on LED screen.

Finish drive for Game pad

Add Audio section. Music will play when explore the map.

April 15 (milestone 2)

Enable game pad control over explore section. Use game pad to control the movement and view-point when exploring the map

Add 2D player's status image to the screen. Design 2D fighting section

April 22

Enable game pad control over fighting section. Use game pad to shoot the enemy.

Finish scene switch between fighting section and exploring section.

April 29(milestone 3)

Integrated audio section and graphic section.

Other changes and adjustment.