Networked American Pool Video Game

Group Name: Pool-Maniac

CSEE 4840 Embedded System Design

Jiawan Zhang jz2492

Xunchi Wu xw2256

Yichen Liu yl2904

Yuhan Zhang yz2500

Zeshi Wang zw2221

1. Project Introduction

Pool, also more formally known as pocket billiards or pool billiards, is the family of cue sports and games played on a pool table having six receptacles called pockets along the rails, into which balls are deposited as the main goal of play.

In this project, we plans to design a 2D American Pool Video Game for two players following the basic American pool rules. Each player will play the game in difference machine (e.g. one plays in the FPGA and the other plays in the laptop) and communicate with the other through network. Our Pool-maniac is a video game played with a cue ball and fifteen object balls, numbered 1 through 15. One player must pocket balls of the group numbered 1 through 7 (solid colors), while the other player has 9 thru 15 (stripes).

This game will be implemented with VHDL and C language and shown on the VGA screen. Besides, a sound will be produced when the collision happens. The player who shoots in the black ball after finishing all his balls wins the game.

1.1. Rules

At the beginning of the game, the players could choose each group of ball to shoot. The one pocketed the first ball, he should pocket else in the group of that ball. And the other player should pocket the other ball group. The player who pocketed one ball can continue to shoot other balls. A player is entitled to continue shooting until he fails to legally pocket a ball of his group. After a player has legally pocketed all of his group of balls, he shoots to pocket the 8-ball and win the game.

1.2 The fouls

1) A shooter directly hits NO.8 ball with the cue ball, without finishing pocketing all his balls, is a foul.

2) When the cue ball is pocketed, it should be place back to the origin point, and the shooting turn switches.

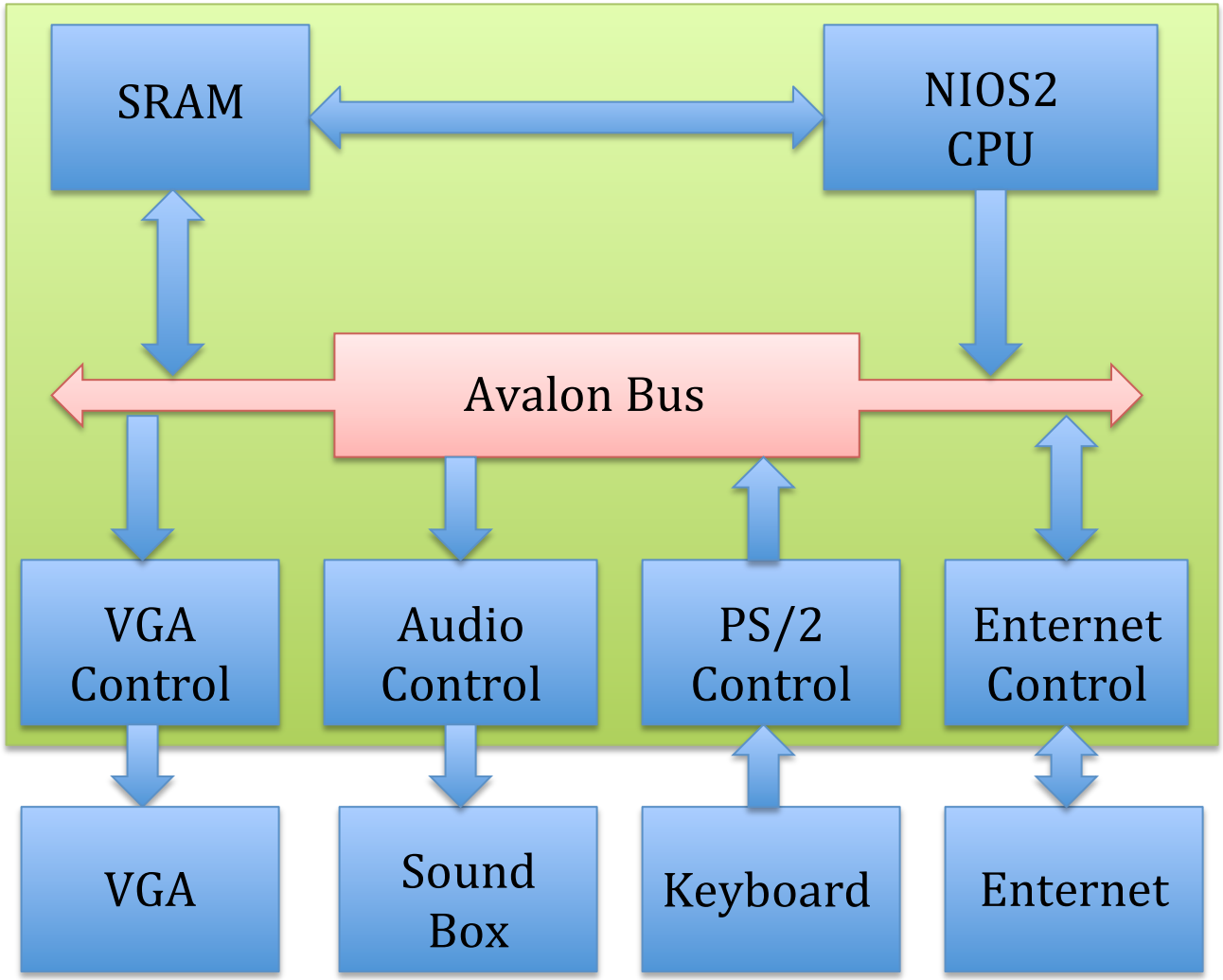
3) Once the ball case is set, a shooter cannot directly shoot opponent's balls. Otherwise it's a foul.

4) Any balls pocketed on a foul remain pocketed, regardless of whether they belong to the shooter or the opponent. The penalty is the switch of shooting turn.

5) If a player jumps an object ball off the table on the break shot, it is a foul and the incoming player has the option of a. accepting the table in position and shooting, or b. taking cue ball in hand behind the head string and shooting.

2. Block Diagram

The top-level block diagram of our project is shown below. We use the Avalon bus with SOPC builder in the Quartus for the communication of each block. The whole block diagram includes SRAM, Nios2 CPU, VGA control, Audio control, PS2 control and Enthernet control. The real hardware devices include VGA, Audio chip, keyboard and Enternet. The details of the implementation and the memory needed for each block and will be provided later.



2.1 VGA block

The VGA block displays the interface of the game. The interface has 5 parts: pool table, balls, pool cue, score board and strength bar. It has 4 levels (Level 0 to Level 3). The data for Level 1 to Level 3 is stored in a ROM in FPGA. The data is stored in 3

separate ROMs, one for each level. The data will be stored in MIF files and loaded into memory bits instead of storing it in logic cells.

Level 0 contains the background (a single color rectangle), the frame of pool table (table without the six pockets), and the strength bar. Since all of them are made with single rectangles, each of which can be determined by 4 points and 1 color signals, so just give the singles as CONSTANT in VGA control vhdl file.

Level 1 contains the frame of score board and the pockets of the pool table. The frame of score board needs 30*150 pixels. The 4 pockets on the corners need 35*35 pixels. The other 2 pockets need 24*24 pixels.

Level 2 contains balls. There are 16 balls, and each ball needs 14*14 pixels.

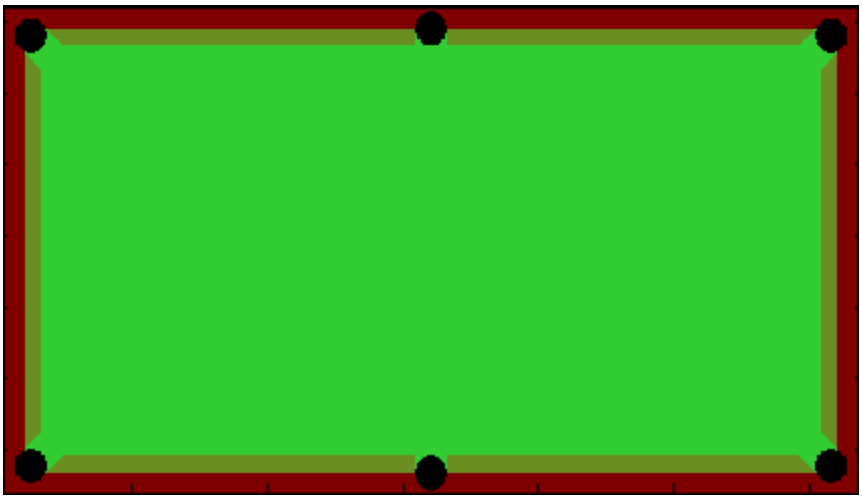
Level 3 contains the pool cue. The pool cue needs a 300*300 matrix to store the pattern of the pool cue for different angles in the first quadrant. The patterns for the rest quadrants will be given by mirror.

For Level 1 and Level 2, each pixel needs 5 bits, which represent 31 colors and the transparent color. For Level 3, each element in the matrix needs 6 bits to represent the angles of the pool cue.

Therefore, the total ROM memory needed for VGA part is as following table.

	Score board	pockets	balls	Pool cue	Total size for VGA
Num of data	30*150 = 4500	35*35*4 + 24*24*2 = 6052	14*14*16 = 3136	300*300 = 90k	
Num of bits for data	5	5	5	6	
Total size (bit) (k = 1024)	21.98k	39.56k	15.32k	527.35k	604.21k

Followings are the pattern for pool table and balls for this game (generated with Matlab).



2.2 Audio block

For the audio block, the player can choose if the sound will be turned on or turned off, which is implemented by the CPU to choose whether the sound mode is on. Besides, it plays the corresponding sound when the pool cue or the table hits the ball, the collision happens between different balls and the ball hits the bag.

To implement the audio module, the music will be generated through the FM synthesis using the DE2 on board WM8731 audio CODEC based on lab 3. The basic FM equation is: $x(t)=sin(\omega_c t+Isin(\omega_m t)$, where $x(t)$ is the amplitude at time t, ω_c is the carrier frequency (the fundamental tone we hear), ω_m is the modulating frequency, and I is the modulation depth. When the audio controller receives the CPU command, it will make the sound box play a corresponding sound.

The audio control block has a ROM to store the sampling sound and it can be accessed at different rates decided by ω_c. Additionally, each sound will last about half second and we will use a counter to control this time. The clock frequency is

50MHz, so the counter needs to count 25M so that the sound will last half second. We plan to use a 5000Hz sample frequency of the 16-bit data. Since the sound need to last for half second at a 50MHz clock frequency, the size of the data for each sound is about 2500×16 bits.

2.3 Keyboard

The game uses PS/2 keyboard to get the operation signals from the players. The keyboard in our project is used to let the player control the state of the pool cue.

Left and right arrow key of the keyboard will be used to rotate the pool cue around the cue ball to set hitting direction. The left arrow key will make the pool cue rotate anticlockwise around the cue ball, while the right arrow key will realize the clockwise rotation. Besides, we will try to hold the space key to adjust strength, and release it to hit the ball. There will be a bar to show the current strength. While holding the space key and the bar will increase gradually and if the bar reaches the maximum strength, it will fall back to zero and increase again if keep pressing. The basic PS2 driver in lab2 can still be utilized here and the keyboard interface for our game will be implemented in C program.

2.4 Enthernet block

The network design will be similar to that of the lab 2, while the main difference is that we will only use one DE2 board with two players operating on two seperate computers meanwhile. Everytime one player kicks a ball using the space key, the communication between two computers and DE2 board will take place to send one packet containing two data fields: the angle of the pool cue, the strength of the hit. The current X & Y pixel location of every 16 balls thus can be caculated using the designed algorithms by two computers after receiving the packet, by which means the data fields of the packet to send will be greatly simpilified.

The device will utilize standard UDP packets with IP headers specifically designed and the ad hoc network that exists in the lab.

Concerning the communication process involved, four different network layers will be accessed programmatically by the user: Link Layer, Network Layer, Transport Layer, and Application Layer.

Reliable Transfer

Since UDP protocol only transfers packets with best effort, without assurance of the arrival of every packet, it is possible that the packet loss and packet corruption will take place during the transmission.

In order to avoid packet loss in transmission, we will adopt the SEND-ACK protocol between the sender and receiver. That is to say, when a packet is sent, the sender will await an ACK from the receiver to acknowledge the receipt of the packet during a specified amount of time (for example 500 ms). If the expected ACK is not received by the sender within that period of time, it indicates that a packet loss has taken place in the transmission and the last packet will be sent again. Multiple failures to transmit the packet will lead to termination of the game. Upon receipt of this acknowledgment from the receiver, the sender and receiver can then compute the detailed X & Y pixel location of every 16 balls.

If a packet is received but is corrupted (based on an inconsistent checksum value), the receiver will not send an ACK and wait for the sender to resend the packet. If corruption of a given packet occurs multiple times, the game will terminate.

Application Layer

The Application Layer will be accessed to transmit data fields between two networked computers.

The two data fields to be transmitted between two clients are: **angle of the pool cue** and **strength of the hit**, both of which will be used later by the algorithms to compute the current X & Y pixel location of every 16 balls after each hitting.

Protocol of Application Layer

The protocols of the Application Layer used is shown below. Note that additional fields may be added during implementation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Type ID				Player ID				Angle of The Pool Cue								Strength of The Hit								Options or Filled							

3. Software Description

Basically we use C language to 1. read the input that the players made using the keyboard, specifically, shoot strength and the angle; 2. program the movement of the balls and 3.send back the coordinates of the balls to hardware and use VHDL to do the screen display.

While connecting with network, we send the parameter of hitting strength and angle via UDP to another hardware and the two devices compute the movement of the ball individually, but based on the same parameters.

The core algorithm mainly including two parts: 1. Detect ball collision and 2. give the reasonable movement trajectory of the ball. To detect ball collision, we first need to work out an algorithm to detect if ball collision happens. If not, which is the majority case, go directly to the next cycle, updating positions for the balls. Otherwise, based on the different collision mechanism to calculate moving directions and velocities of the balls. There are two types of collision: 1. the collision between the ball and the table edge and 2. the collision between two or more balls. In the first type of collision, we could use the simple mirror reflection to decide the bouncing angle with no energy loss. For the second case, we need to consider the different collision angles of the balls actually collide. The constant friction coefficient is introduced into the speed calculation. It helps to simulate the gradually decreasing speed of the moving balls.

4.Critical Path

For an American Pool game, the appropriate functionality of the algorithm is of great significance. Different strength and angle when hitting the cue ball will direct different result. The collide will also happen between different balls with different speed as well as between balls and table, and this is also important for the game. We need to design the algorithm of the collision carefully.

Besides, the movement of the ball should be design carefully. We should make the movement as real as possible, especially the move direction of the ball after collision and the speed of the ball. If the movements of the balls generate many unexpected actions, the joy of the game will be greatly decreased.

In addition, since there are two players will be in the game simultaneously and UDP is a connectionless transfer, which cannot assume that all packets will arrive without error across the network. We need to solve the problems when the packet lost during transmission and when a corrupted packet is received. If the synchronization problem of the packet occurs multiple times, the game will hard to continue. Thus, a robust mechanism is need to make sure this synchronous requirement.

5. Conclusion

In our project, a networked American Pool video game is designed. We store most of data in the DE2 board. The memory for the hardware needed is discussed above and the part of software will not occupy lots of memory. Thus, with carefully design for each block, like VGA, audio, PS/2 keyboard and Enternet, for a 512K SRAM our pool-manic game should work.