Tank Game Design

Zixiang Zheng zz2642 & Wenzhe He wh2443

1. Project Overview

We will be designing a classic 1985 tank game 'Battle City'. Our goal is to imitate as much as possible, to bring up the original taste of the game.

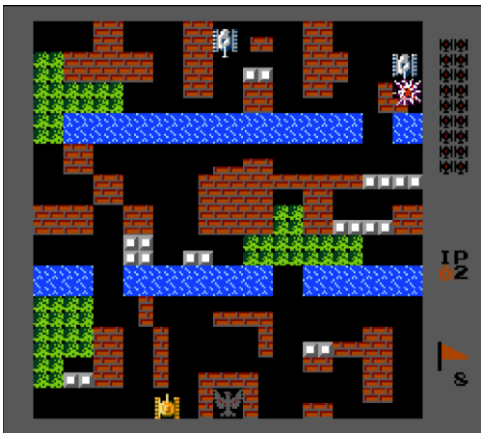


Fig.1 NES 1985 'Battle City'

The original tank game has 35 different stage, we will try to implement one stage and design another one stage by our own. Fig.1 shows the stage 8 of the game. Because the game was firstly deployed on NES. So, we will use joystick to control the game.

2. Game information.

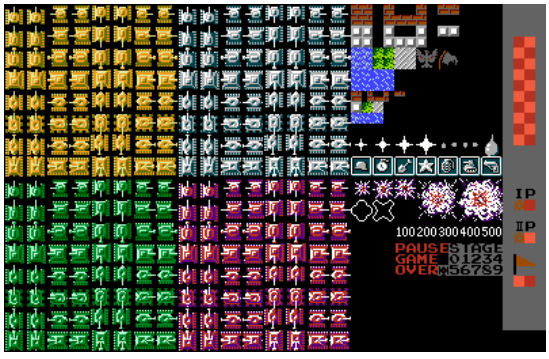


Fig.2 General Sprite of the 'Battle City'

Figure 2 shows the general sprite of the game. It is important to know that only have of the tank tiles

are actually used in the game.

(a) Color information

The total number of colors used in tank game is about 21. But it is important to notice that each tile contains no more than 4 colors. Which means that each pixel can be storage in 2 bits. Then use a color table to store the RBG information of the 21 different color.

(b) Map Information

There are 6 different blocks in the game. We will at least finish first 5 of them. We named them our own. Each block 16*16 contains with four identical 8*8 basic block.

Table 1 Block Information

Image (8*8)	Name	Tank pass	Bullet pass	Note
	Blank	Yes	Yes	
	Brick	No	No	
	Stone	No	No	Only level-4 player can break
	Grass	Yes	Yes	Tank and bullet will display under it
	Water	No	Yes	Water will display
	Ice	Yes	Yes	Tank cannot stop immediately

(c) Tank information

As we mentioned, there are only limit tank used in the game. In this part we will list all of them. Each tank used two images to display, this is because when tank is moving, the track is not move. As shown in figure, the track of to tank is actually 1 pixel different, so when tank is moving, they are displayed alternately.

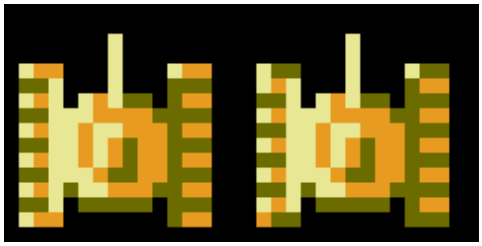


Fig. 3 Two Image of tank moving upward

Now we will list all tank and their image used in this game.

(1) Player 1 Tank

Table 2 Player 1 tank information

Tank image (16*16 pixel *8)	Feature	Bullet Speed	Bullet per round	Note
	Level 1	Slow	1	
	Level 2	Fast	1	
	Level 3	Fast	2	
	Level 4	Fast	2	Can destroy ‘Stone’

We call above tank color ‘Yellow’.

(2) Player 2 Tank

Table 3 Player 2 tank information

Tank image (16*16 pixel *8)	Feature	Bullet Speed	Bullet per round	Note
	Level 1	Slow	1	
	Level 2	Fast	1	
	Level 3	Fast	2	
	Level 4	Fast	2	Can destroy ‘Stone’

We call above tank color ‘Green’.

(3) Enemy normal Tank

Table 4 Enemy normal tank information

Tank image (16*16 pixel *8)	Feature	Bullet Speed	Movement Speed	Note
	Level 1	Slow	Slow	
	Level 2	Fast	Fast	
	Level 3	Fast	Slow	
	Level 4	Fast	Red	Need 4 bullet to destory

We call above color ‘White’.

The level-4 tank will first display in ‘Green-White color’. This means that it will swipe between ‘Green’ and ‘White’ so that it looks like ‘light green’ so that it will be different with player’s ‘Green’ tank. After 1 hit, it will change to ‘Yellow-White’, then it will change to ‘Green -Yellow’, then it will change to ‘White’. Then after another bullet, it will be destroyed.

(4) Emery Gadget tank

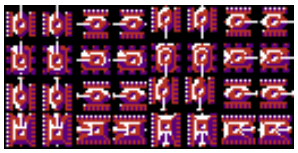


Fig.4 Gadget tank image

We called above color ‘red’.

Emery Gadget tank is display in ‘Red-white color’. When it is hit, the player bullet will deal same damage then there will be a random Gadget display on map. The red color will disappear after the hit.

(d) Gadget Information

Although there are 7 icon of the Gadget, our team only know first 6 of them. So, we will at least finish 4 of them.

Table 5 Gadget information

Image (16*16)	Name	Function
	invincible	Player tank become invincible for several second
	Freeze	Emery tank will stop moving and shooting for several sec.
	Reinforce	The bass wall will repair and become stone for several sec.
	Upgrade	Upgrade player tank by one level.
	Nuke	Destroy all existing tank
	Life	Add one additional life

(e) Important Game logic

(1) Player can use arrow button to control the movement of tank and press ‘A’ to shoot.

(2) Brick wall will lose 8*16 pixel after player shoot. Brick wall will lose 4*16 after emery shoot.

(3) There are at most 4 emery tanks in the game. When the number is less than 4, it will appear in top

left, top middle, top right in order. The remain enemy tank of the stage will display on the top right of the game.

(4) Player remain life will display on the middle right of the screen. If the life remain is 0 and tank is destroyed, game over will appear on screen.

(5) Just to make it simple, all bullet can be destroyed by another bullet. All tank can deal damage to another tank.

(6) If tank turn turned in the middle of 8 pixels, it will automatedly be adjust to fix point to avoid bug.

(7) The size of the battle filed is 208*208, the size of whole game is 256*224

3. Block diagram

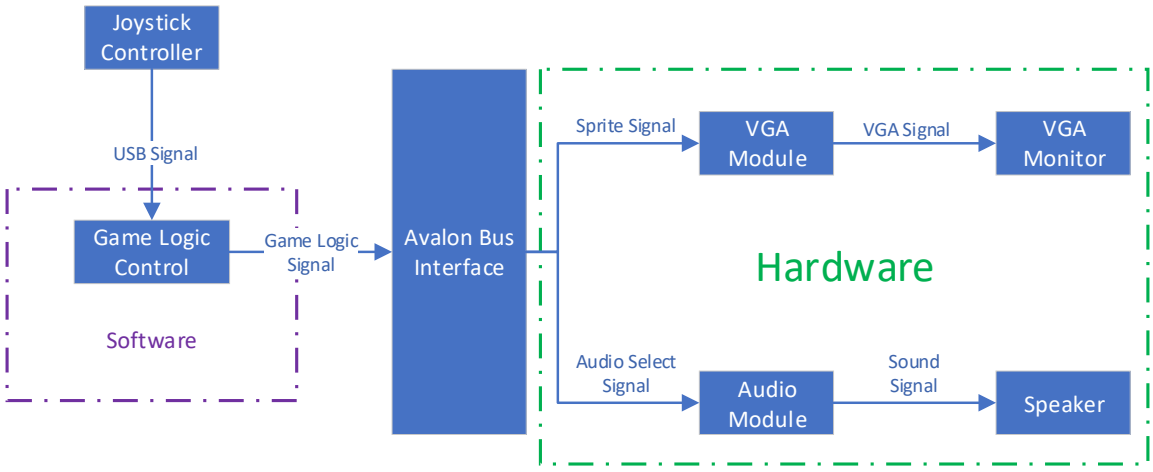


Fig.5 Block Diagram

4. Hardware Implementation.

(a) VGA Module

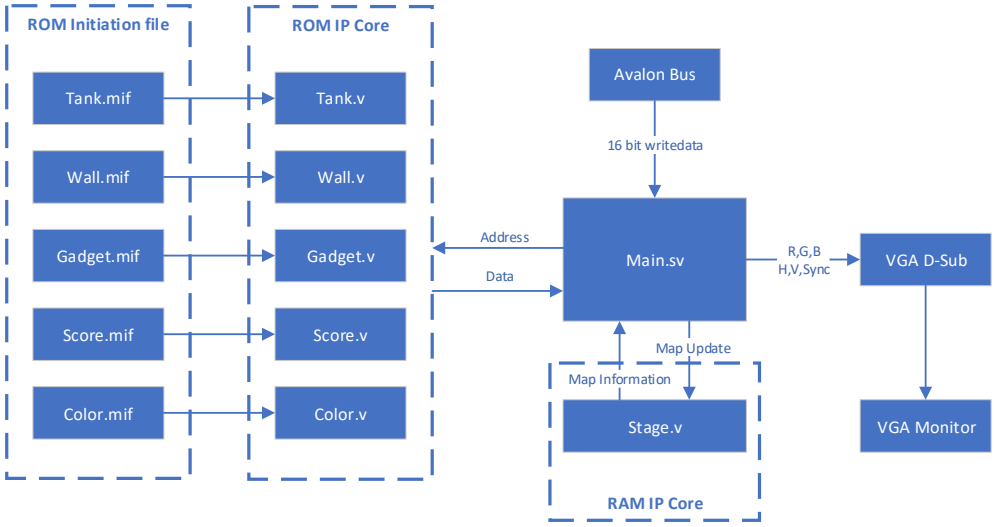


Fig.6 VGA Module Block Diagram

Fig.6 is the block diagram for VGA part. We use ROM IP Core to store tile pixel and color map. And we use RAM to store current Map wall information. At each cycle, software will only send updated wall information instead of send whole map information. This can be seen as an 'Differential Encoder'.

The different map information is store in software. When the stage starts, the software will send the current entire stage map to the hardware. After the stage is running, it will only send data when some wall are change. By doing this we can greatly reduce the data send at each time through Avalon interface. The data of Avalon Bus will be shown in section 5.

When display each pixel, first check the register to find out the name of tile or sprite on this pixel. Then check the ROM to find the pixel data, then check the ROM colormap to find the color of this pixel. Then send the color to VGA D-sub to display on the monitor.

We will use the same technic in Lab3 to process the data. From the analysis of the game logic, we need four layers to display the graphics.

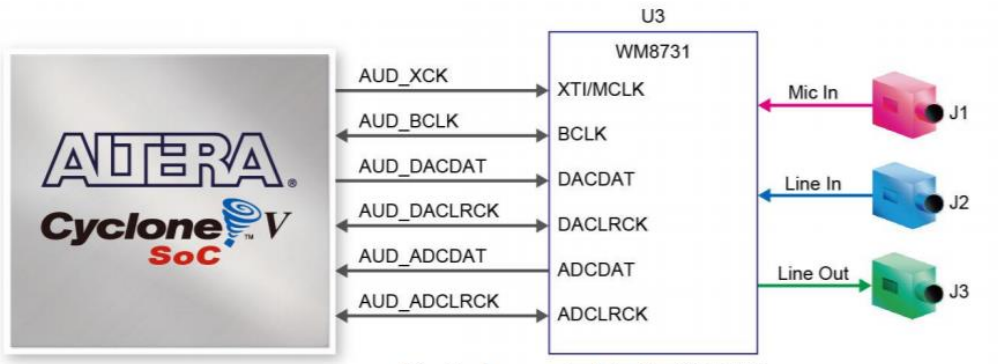
Table 6 Layer of the game

Layer	Name	Description
1	Background	background, wall, gadget, score
2	Tank	player's tank and enemy tank

(b) Audio Interface

On the DE1-SoC, there is a device called Wolfson WM8731 which has 2 24bits ADC, 2 24bits DAC

and a headphone amplifier and support 8-96 kHz sampling rates. The WM8731 is controlled via serial I^2C bus, which is connected to HPS or Cyclone V SoC FPGA through an I2C multiplexer. The connection of the circuit is shown as following.



Pin Assignment of Audio CODEC

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
AUD_ADCLRCK	PIN_K8	Audio CODEC ADC LR Clock	3.3V
AUD_ADCDAT	PIN_K7	Audio CODEC ADC Data	3.3V
AUD_DACLRCK	PIN_H8	Audio CODEC DAC LR Clock	3.3V
AUD_DACDAT	PIN_J7	Audio CODEC DAC Data	3.3V
AUD_XCK	PIN_G7	Audio CODEC Chip Clock	3.3V
AUD_BCLK	PIN_H7	Audio CODEC Bit-stream Clock	3.3V
I2C_SCLK	PIN_J12 or PIN_E23	I2C Clock	3.3V
I2C_SDAT	PIN_K12 or PIN_C24	I2C Data	3.3V

Fig.7 Block Diagram Connection of circuit

Basically, we need two signal lines to control the WM8731. One is for I^2C bus clock and another is for data communication. To start the process, pulling down the data line and pulling high the clock line. The first step is to send a device address through the data line. If R/W signal is sent, the master releases the data line. If an I2C device receives its address, it pulls the data line LOW at the next clock cycle. Then two bytes of data are sent. Finally, in order to put it into an end, pulling the clock line High, while the data line is kept LOW.

As for our case, we assume that a sample rate of 40kHz is adequate, which means for one second there are 40000 sample points. And one sample contains two bytes. Therefore, in the end for one second, we need 80K Bytes. In our design, we have several sound in the game: 0.3s bullet sound,0.3s hit sound, 0.5s destroy sound, 0.5s upgrade sound, 0.3s tank moving sound, 1s game over sound. So, in total, we need 2.9s sound requiring at most 232,000 Bytes space.

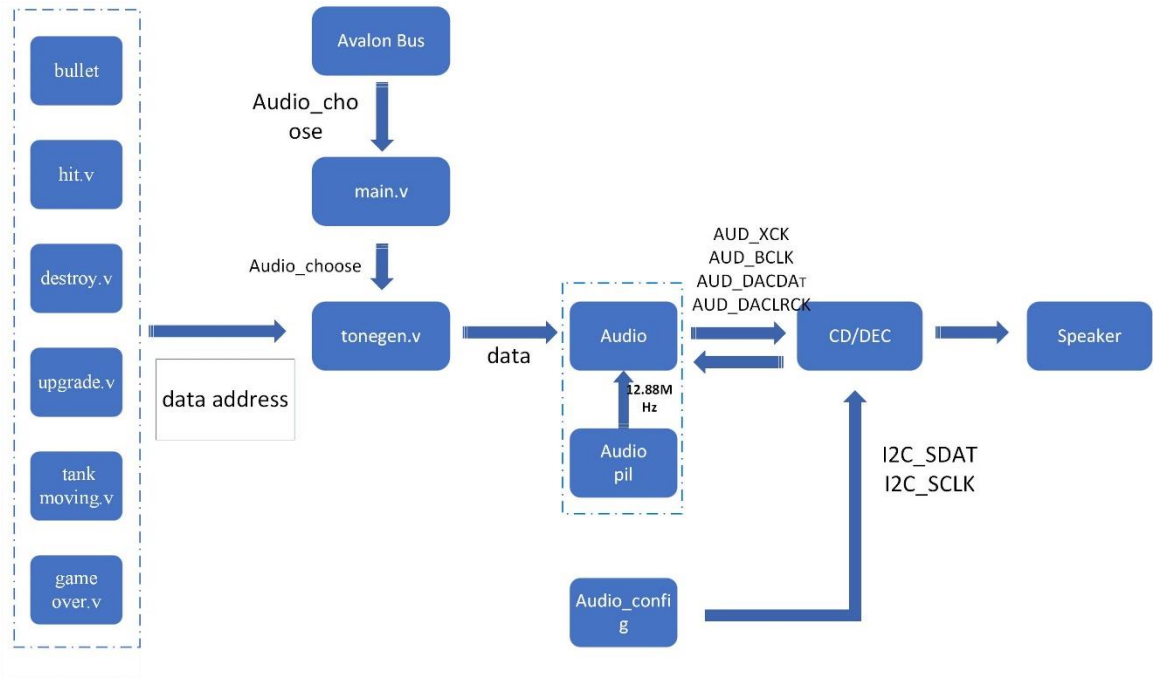


Fig.8 Audio Module Block Diagram

The whole structure and details are shown in Fig.8. The first columns are audio stored in on-chip Rom IP Core， after converting original file to .mif file and finally to .v file. The main function tells the tongen.v which audio to choose. Afterwards, what tonegen does is to pick the data from the Rom using FIFO at a certain frequency and feed the data to WM8731. The WM8731 configuration can be accomplished by Qsys built-in IP core to automatically configure the audio chip. The audio pill will offer 12.88MHz driving clock.

(c)ROM Storage

We will store all used tile and sound information in FPGA Rom IP Core module. The IP Core can be directly access in Main.sv without claim in top.v. And the length and size can be set arbitrarily.

Table 7 Rom Size Caculation

	Number	Color	Length each(bit)	Total length (Byte)
Color	21	/	24	63 Byte
Tank	144	4	Pixel: 16*16*2 Color Map 4*8	9792 Byte
Block	8	4	Pixel: 16*16*2	544 Byte

			Color Map 4*8	
Explosion	3	4	Pixel: 16*16*2 Color Map 4*8	204 Byte
Number & Character &Tank_count_icon	30	2	Each pixel 8*8	240 Byte
Base Icon	2	4	Pixel: 16*16*2 Color Map 4*8	136 Byte
Enemy show up tile	4	2	Pixel: 16*16	128 Byte
Destroy Score	5	2	Pixel: 16*16	160 Byte
Gadget	6	4	Pixel: 16*16*2 Color Map 4*8	408 Byte
Bullet	4	2	Pixel: 8*16	64 Byte
Total tile				**11,739** Byte
Audio				232,000 Byte
Total				**243,739 Byte**

We need about 250kB to store the data we need, which is Ok for the FPGA. If there is no enough space, we will tired to limit the music size.

5. Software Implementation

(a) Joystick Control

We will use Joystick to control the game. The Joystick we used is shown in Fig.9

Fig.9 INNEXT gamepad

The function of each button is list in Table 8.

Table 8 Button Function of Joystick

Button	Function	Button	Function
Up	Tank move up	Start	/
Down	Tank move down	Enter	Pause and Confirm
Left	Tank move left	A	Shoot & Select Stage
Right	Tank move right	B	Select Stage

At each interrupt, the gamepad will send 8 bytes data to the master device through USB interface. The data is shown in the following table.

(b) Avalon bus Data Table

This table specify the data than transfer from software to hardware at each software cycle. We use ‘iowrite16()’ to transfer.

Table 9 Avalon Bus Data Table

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Remark
00	X Position					Y Position					Brick					En.	Wall update, each will update 8*8
01	X Position					Y Position					Brick					En.	Score, Explosive, Gadget Tile
02	X Position								Y Position								Player tank Sprite
03	Tank Class		Color number						Gadagt		Direction					En.	
04	X Position								Y Position								Emeny tank 1 Sprite
05	Tank Class		Color number						Gadagt		Direction					En.	
06	X Position								Y Position								Emeny tank 2 Sprite
07	Tank Class		Color number						Gadagt		Direction					En.	
08	X Position								Y Position								Emeny tank 3 Sprite
09	Tank Class		Color number						Gadagt		Direction					En.	
10	X Position								Y Position								Emeny tank 4 Sprite
11	Tank Class		Color number						Gadagt		Direction					En.	
12	X Position								Y Position								Bullet 1 Position
13	X Position								Y Position								Bullet 2 Position
14	X Position								Y Position								Bullet 3 Position
15	X Position								Y Position								Bullet 4 Position
16	Direction 1		Direction 2		Direction 3			Direction 4	En. 1		En. 2		En. 3		En. 4		Bullet direction and visiable
17	Score								Tank life				Ramain Tank				Game Information

(b) Calculate Game logic

Software will compute the game logic and send the result to hardware. The game logic is described previously. Software will calculate the position of tank and wall on the screen. It will also control the show up of enemy tank and their movement. The software will also send correspond audio control signal to hardware.

(c) Count Score

There will be a score board display after each stage. To show how much score the player scores this stage and add up with the previous score.

6. Milestone Plan

Milestone 1:

Accomplishment using a joystick gamepad control a single tank move on the VGA Monitor. Add simple song when triggered.

Milestone 2:

Add map display. Create a simple map. And add one enemy tank and one gadget. Achieve fire bullet

and destroy logic both for the tank and wall. Add more sound to the game.

Milestone 3:

Add more enemy tank and gadget. Also, create more maps. Add a score phase when players finish each stage. And edit other details to make it more like the 1985 ‘Battle City’ tank game.

7. Reference:

1. Battle city online game

https://www.retrogames.cz/play_014-NES.php

2. Battle city General Sprite

http://mirrors.pdp-11.ru/_sprites/www.spriters-resource.com/nes/batcity/sheet/60016/index.html

3. DE1-SoC User Manal

4. ROM IP Core usage

https://blog.csdn.net/weixin_41445387/article/details/86835663

5. Audio CODEC

https://www.cl.cam.ac.uk/teaching/1617/ECAD+Arch/optional-tonegen.html