**CS210**

**2201CS65**

**SAUMYA SINHA**

**MINI PROJECT PHASE II**

**Under the guidance of : PROF. JIMSON MATHEW**

**IIT PATNA**

# 

# **TANK WARS GAME REPORT**

**INTRODUCTION**

Tank Wars is indeed a classic in the artillery game genre, offering a delightful mix of strategic depth and straightforward gameplay. The incorporation of physics mechanics, such as projectile motion and gravity, adds a layer of realism to the gameplay while also introducing a challenge for players to master. The ability to adjust shot power and aim adds strategic depth, requiring players to carefully plan their moves to outsmart their opponents.

The bouncing projectile adds an extra dimension to the game, making each shot not only about hitting the target directly but also about utilizing the environment to your advantage. This aspect can lead to some truly creative and unexpected plays, further enhancing the game's replay value.

Overall, Pocket Tanks provides a balance of accessibility and depth, making it a timeless favorite among gamers of all ages. Whether you're a casual player looking for some quick fun or a seasoned strategist aiming for precise shots, Tank Wars offers something for everyone.

# **CONTRIBUTIONS**

Saumya Sinha 2201CS65

· Power changing push buttons

· Moons and stars

· VGA connection

· Led for health

· Projectile of bomb

· Push buttons to move

· Color changing after hitting

· Seven segment display for power

**WORKING**

It looks like you're developing a game using an FPGA board, incorporating features like displaying tank power and health on seven-segment displays and LEDs, as well as handling user input through push buttons. Additionally, you're implementing graphics rendering for displaying backgrounds using a VGA interface.

Your code snippets demonstrate an understanding of interfacing with hardware components and implementing basic game mechanics. Here's a breakdown of what each part does:

1. \*\*Displaying Tank Power (Seven-Segment Display)\*\*:

- The `display\_num` function takes an integer `x` representing the tank power and displays it on a seven-segment display using hexadecimal values for each digit.

2. \*\*Displaying Health of Players (LEDR Section)\*\*:

- The `updateHealthToLEDR` function updates the health of both players simultaneously on the LEDR section using bitwise XOR operations to manipulate the LEDs based on the health values.

3. \*\*Handling User Input (Push Buttons)\*\*:

- The code checks for button presses using a while loop and updates the game state accordingly. For example, pressing the buttons moves the tanks or adjusts the power of shots.

4. \*\*Graphics Rendering (VGA Interface)\*\*:

- The `write\_pixel` function writes a pixel of a specified color at a given coordinate on the VGA display. This function likely gets called multiple times to draw various shapes and backgrounds required for the game.

We have used seven segment display to display the power of the tanks :

void display\_num(int x)

{

volatile int \*hex3\_hex0\_ptr = (int \*)HEX3\_HEX0\_BASE; // Pointer to HEX3\_HEX0

int digit\_values[] = { 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F };

int ones = x % 10; // Extract the ones digit

int tens = (x / 10) % 10; // Extract the tens digit

// Set the values of the individual digits on the 7-segment displays

\*hex3\_hex0\_ptr = (digit\_values[tens] << 8) | digit\_values[ones];

}

For displaying the health of both of the players simultaneously in the LEDR section we have used Bitwise-XOR operation:

void updateHealthToLEDR(int health1,int health2)

{

volatile int \*LEDR = (int \*)LEDR\_BASE;

\*LEDR = ((((int)(1<<health2)) - 1) + ((((int)(1<<10) - 1))^(((int)(1<<(10-health1)) - 1))))& 0b1111111111;

}

For using the push buttons on the FPGA board we have used while loop for shoot ,moving position of the tanks , adjusting power :

for (int i = 0; i < 2000000; i++);

PBval = \*PUSHBUTTONS;

display\_num(pwr);

if (PBval & 0x01) // Check if button KEY0 is pressed

{

if(turn){

erase\_power\_bar(player\_A.posx,player\_A.posy-40);

player\_A.posx++;

draw\_power\_bar(player\_A.posx,player\_A.posy-40,pwr);

}

else{

erase\_power\_bar(player\_B.posx,player\_B.posy-40);

player\_B.posx++;

draw\_power\_bar(player\_B.posx,player\_B.posy-40,pwr);

}

prnt\_rect(player\_A.posx-1, player\_A.posx-1 + 20, 240 - 20, 240, 0); //tank 1

prnt\_rect(player\_B.posx-1, player\_B.posx-1 + 20, 240 - 20, 240, 0);

}

else if (PBval & 0x02) // Check if button KEY1 is pressed

{

if(turn){

erase\_power\_bar(player\_A.posx,player\_A.posy-40);

player\_A.posx--;

draw\_power\_bar(player\_A.posx,player\_A.posy-40,pwr);

}

else{

erase\_power\_bar(player\_B.posx,player\_B.posy-40);

player\_B.posx--;  
 }

For displaying the various backgrounds in the game we have used write pixel function and drawn various shapes as needed :

void write\_pixel(int x, int y, int colour)

{

volatile short \*vga\_addr = (volatile short \*)(0x08000000 + (y << 10) + (x << 1));

\*vga\_addr = colour;

}