**Mini-Project Final Report**

**COMPSYS 305**

Digital Systems Design

*Faculty of Engineering, University of Auckland*

**Jonathan Soulsby & Chenggang Lin**

# Introduction

The objective of this mini-project is to design a game-console using a DE0 board with a game programmed onto it using digital logic and design. The game to be designed is based off a single-player tank shooter that involves a mouse-controlled tank at the bottom of the screen and a moving enemy tank at the top of the screen. The aim of the game is to shoot the tank at the top of the screen to score points. The scheduled deadline date is 8:00pm Tuesday 30th May.  
The mini-project will be designed and developed using the following provided equipment:

* An Altera DE0 board (used as the game console)
* USB cable
* power supply
* PS/2 Mouse (A USB Mouse + PS2 Adapter)
* VGA cable
* 640x480 pixel VGA screen

With respect to the digital logic and design processes, the game is being coded in VHDL and programmed onto the DE0 board using Quartus II.

# Game Strategy

# Overview As the aim of the game is to score points by shooting the enemy tank, the general game strategy is to account for the enemy tank’s current trajectory and bullet projectile speed, positioning the player’s tank accordingly and shooting to hit the enemy tank. As the enemy tank follows a simple x-plane travel trajectory with a constant speed, this method can be reliably used to hit the enemy tank. Even if the player misses the first few shots, the player tank’s bullet projectile speed and required positioning to land a successful hit on the enemy tank can be learnt fairly quickly; making this a relatively simple and easy game to master. Features

Our game follows a simple, intuitive user interface, with a minimalistic design to reduce on-screen clutter and reflect the ‘retro’ style graphics the game incorporates. It consists of two game modes; ‘play’ and ‘training’. The ‘play’ game mode consists of two levels that each possess a sixty second time limit of increasing difficulty (in the form of a higher score required to clear each level, as well as faster moving enemy tanks), the player’s score is incremented each time the player successfully shoots the enemy tank, with ‘level 1’ requiring ten points to clear, and ‘level 2’ requiring fifteen points to clear (and in doing so win the game). In saying so, should the player be unable to reach ten and subsequently, fifteen points respectively, the game is lost and the game over screen is displayed. A summary of the game’s main displays are explained in finer detail below:

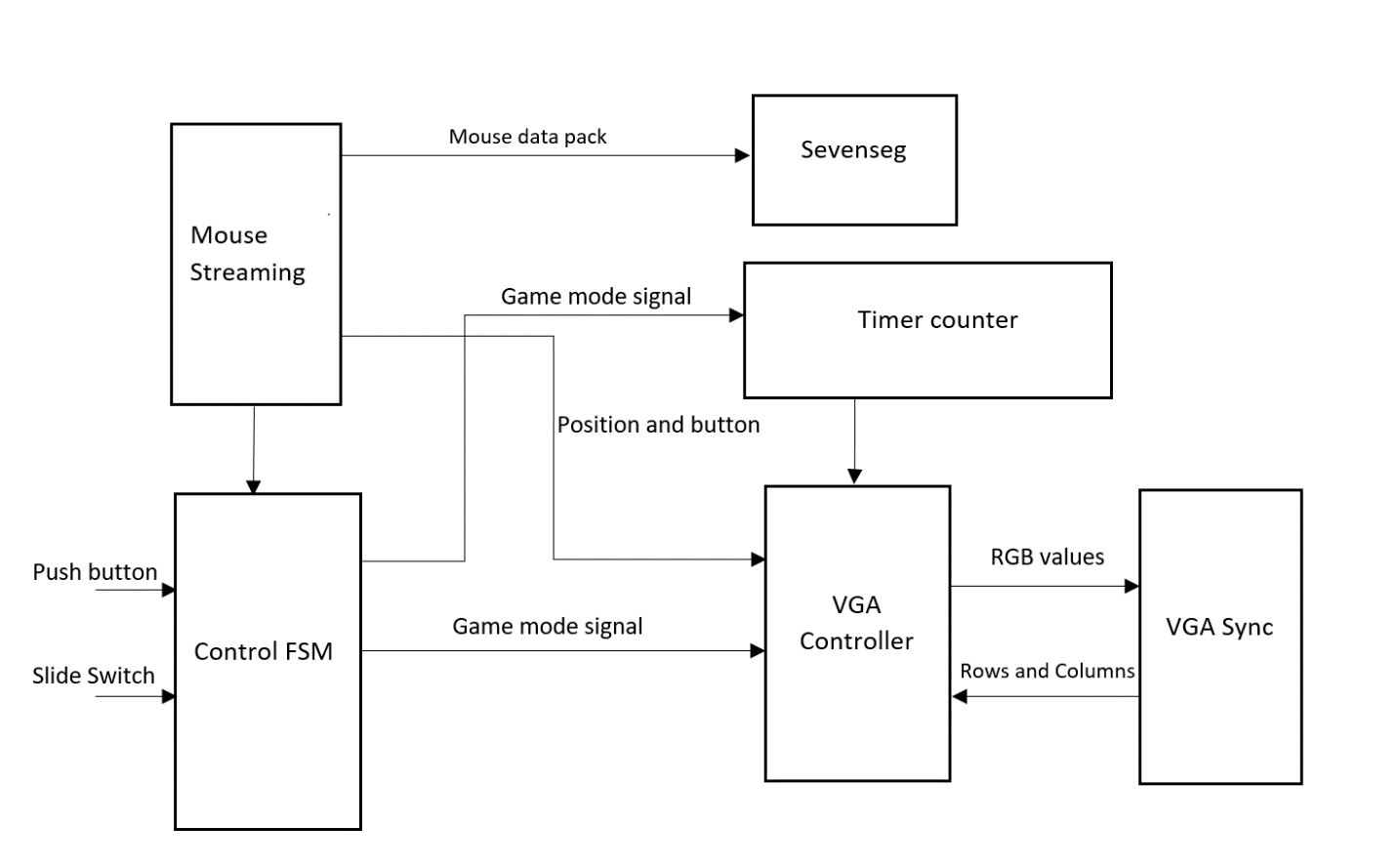
* Idle/Menu
  + The default menu screen that is displayed when the game starts, or when the user clicks the left mouse button on the game over or game won screen. It can also be accessed by pressing PB1 at any time (the first push button on the de0 board) as a method of resetting the game. From this screen, the user can select which game mode they wish to play by using the right mouse button to enter the main ‘play’ game mode, and flipping switch zero (SW0) to enter the ‘training’ game mode. This is clearly detailed in the form of on screen text underneath the ‘TANK HUNTING’ header.
* Training
  + This game mode is accessed flipping switch zero (SW0) at any point during the game’s runtime. In this mode the timer display in the top right is paused, removing the time pressure from the normal ‘play’ mode. This mode can be exited by pressing PB1 at any time to bring the game back to the idle/menu screen.
* Level 1
  + This is the first level of the game and features two enemy tanks moving horizontally across the top of the screen, the player must shoot the enemy tanks ten times in sixty seconds to advance to the next level, should either tank reach the bottom of the screen, the game is lost.
* Level 2
  + This is the second level of the game and is similar to level 1. However, the goal has been increased to fifteen points and the enemy tanks move faster, evidently increasing the difficulty of the game (with respect to level 1). Once fifteen points have been reached or the time limit reaches zero, the game displays the ‘Game Won’ or ‘Game Over’ screen respectively.
* Game Won
  + This game screen is displayed after ‘level 2’ is cleared with a simple, concise congratulatory message for beating the game. It can also be directly accessed by pressing push button 2 (PB2) at any time. Pressing left click will return to idle.
* Game Over
  + This game screen is displayed after either ‘level 1’ or ‘level 2’ is lost (i.e. the sixty second time limit is reached) with “You Lose” displayed on screen. From here, the left mouse button will return the game to the idle/menu screen.

## Controls

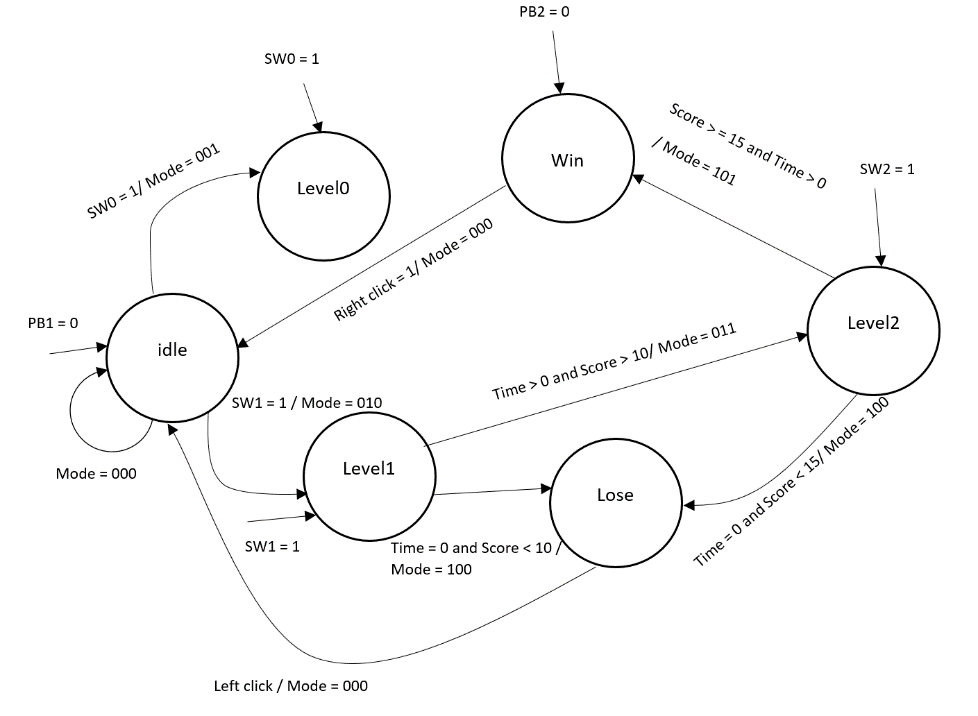
The game is controlled primarily using a PS/2 mouse, with a few buttons and switches on the DE0 board used to navigate the game states directly. The mouse dictates the majority of player input; moving the player’s tank horizontally across the bottom of the screen, shooting with the left click mouse button and navigating the idle/menu screen are all done using the mouse. This type of usage was selected to fit project criteria, as well as retain a simplistic and accessible interface for playing the game. Push buttons one and two (PB1 and PB2) skip to the ‘Game Won’ screen and ‘Menu/Idle’ screen respectively, and serve for testing and debugging purposes.

## Implementation

## High Level Block Diagram

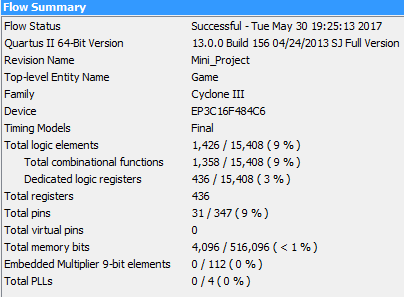
Our game consists of a finite state machine control, mouse streaming finite state machine block (sourced from the provided project resources), sevenseg decoder, VGA controller and VGA sync block. The mouse streaming block streams data to the game’s FSM and VGA controller as user input, this input is used to enter or exit game modes, move the player’s tank and shoot bullets. The sevenseg decoder displays the relative mouse movement (which can be used to check if the mouse if functioning as intended), and each of two decimal points that blink if the left or right mouse button is pressed respectively. The timer counter consists of two almost identical BCD components (with a slight difference in initialisation and reset values), and provides the time and score values for the VGA controller. Moving onto the VGA controller itself, we have the main component of our game. It controls the RGB values sent to the VGA syncing block, used to display text, score, game time, bullets and both player and enemy tank. It also contains a pseudo random number generator based on clock counter to generate a random enemy tank respawn position and movement direction at the top of the screen. The VGA syncing block scans through the 640x480 resolution screen from top to bottom, and provides the current pixel’s row and column value for the VGA controller to use as a grid to display text.

Finite State Machine

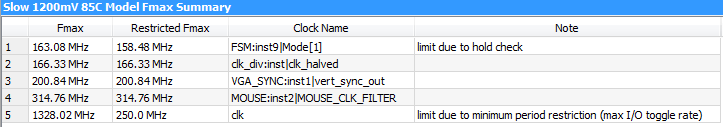
Our game implements a Mealy state machine as a central control unit, this was decided over a Moore state machine due to the increased responsiveness and dependency on player inputs. The finite state machine has 6 states; idle/menu, level0/training, level 1, level 2, win and lose. Push buttons, slide switches, left/right mouse buttons, the in-game timer and game score are used to determine the current state of the finite state machine. In the idle state, the right mouse button will move the finite state machine to Level1, and switch zero will move it to level0/training. In the state Level1, if the timer reaches zero, then the finite state machines moves to a ‘lose’ state (indicating game over). Otherwise the user must reach ten points to advance to the level2 state. In the state Level2, if the player’s score reaches fifteen before the timer reaches zero, then the next state in the finite state machine is ‘win’ (indicating the game has been won). In both the ‘win’ and ‘lose’ states, the mouse buttons left click will transition the game into the idle/training mode.   
Independently from the game’s intended design, switches zero (SW0), one (SW1) and two (SW2) will move the finite state machine to level0/training, level1 and level2 automatically and respectively. Also, push button one (PB1) resets the game to the idle state, and push button two (PB2) moves the game to the ‘win’ state and displays the corresponding ‘You Win’ text on screen.

## Results

## Memory Allocation

As seen in the attached resource consumption analysis (as generated by Quartus II), our game uses only a fraction of the Altera’s Cyclone III performance capabilities. Using the provided “tcgrom.mif” file to display text on screen, we have edited the last index in the file from the letter ‘F’ to a circle which is used as our bullet, in doing so, our game saves the usage of a separate .mif file and another 9Kbits of on-chip memory (in the form of one fewer M9K memory block in ROM mode assigned to another .mif file), as well as this we have minimised our usage of large memory-heavy data types such as integers and opted for std\_logic\_vectors instead to keep memory usage to a minimum.

### Maximum Operating Frequency

Our maximum operating frequency was identified using Quartus II’s Timing Analyzer, as seen above. This was identified as 158.48MHz, note that we have determined this using the Slow 1.2V 85°C model from Quartus II, not the faster and lower temperature alternative models from Quartus II (which resultantly obtains higher maximum operating frequencies due to the improved circuit performance under those conditions).

Discussion and Future Work  
The game itself is relatively simplistic, as such it has the flexibility to be laid out as a foundation for future development. Additional features that could be implemented in the future are a strong possibility and the nature of the game itself serves to accommodate the added functionality. Player lives, some sort of upgrade or power-up system, angled shooting, blocking objects and bouncing bullets are just a few potential ideas that could be implemented in future iterations of the game. However these are not just limited to visual/game developments, a backing soundtrack, or even sound effects could be implemented in future too.  
  
Interface wise, due to the PS/2 mouse user input, the current iteration of the game is solely single-player, the option of changing user input to a keyboard would grant flexibility for future iterations of the game to support multi-player functionality as well as more buttons that can be mapped for extra functionality (for example letters A/D to move left/right and W/S to change shooting angle for a player, and Left/Right and Up/Down arrow keys correspondingly for another player).

Acknowledgements  
This project would not have been made reality without the time and assistance from our lecturers and teaching assistants. In saying so we’d like to especially thank Dr Muhammad Nadeem and Dr Morteza Biglari-Abhari for their feedback and guidance throughout this project.