COMPSYS 305 Mini-Project Final Report

Prasham Jhaveri and Kunal Bhatia
The University Of Auckland, pjha433@aucklanduni.ac.nz, kbha192@aucklanduni.ac.nz

***Abstract*— The goal of this Mini Project is to design a simple Pong like game on a FPGA device which only uses digital logic's and digital design.**

I. INTRODUCTION

The Pong game revolves around the user trying to collect 90 balls within the given time of two minutes. The game consists of 3 levels of increasing difficulty. It is the users job to manoeuvre the paddle which is located at the bottom of the screen and try to collect the required 90 balls within two minute time limit.

In this Mini Project we were provided with many components such as the ball, mouse and char ROM. Along with these components we also added additional components such as bar, timer, LFSR, FSM and game. The addition of these components ensure a fun and challenging experience for the user.

II. DESIGN SPECIFICATIONS

A. Equipment

In this project the game had to developed using the following equipment:

- VGA Cable
- DE0-Board
- PS/2 Mouse

B. Design Requirements

The Pong game is a simple 2D game which revolves around the user using the paddle/platform to catch the moving balls around the screen. Every time the user catches a ball the score increments and the ball relocates and appears at another position. This is done for a certain amount of time.

In this project the game will be programmed into the DE0 board which will be behaving like the console. The console (DE0) will interact with the monitor through a VGA cable which will display the game on the monitor. The game mode, i.e. Training Mode or Single Player will be selected by using the DIP switches which are located on the DE0 board. The DIP switches will also be used to pause the game (as discussed with Nadeem). The PS/2 mouse is connected to the DE0 board and is the user input. This can be said as the PS/2 mouse enables the player to control the paddle/platform from left to right. A push button will be used to initialise the game and will be also used to exit to the main menu. The user will play the game until they finish the three levels where each level is harder than the previous level.

III. BLOCK DIAGRAM

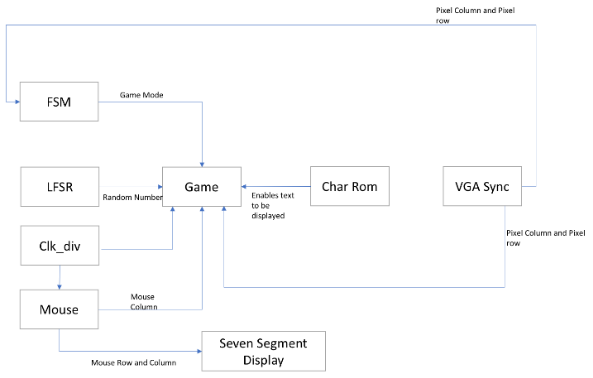


Fig. 1. Block Diagram

FSM: The FSM handles the different states of the game i.e. single player and multiplier and determines the next state depending on the inputs made.
LFSR: Generates a random number which is used to relocate the ball to a random position upon collision with the bar. It also generates a random angle every time the ball collides with the paddle.
Mouse: The mouse entity manages all the signals which are sent to and from the mouse. The mouse currently sends it row and column to the seven-segment display which displays the mouse position on the screen.
Seven Segment Display: This entity currently displays the mouse position on the screen. Once the game is completed this will display the players score.
char ROM: This takes the pixel column, pixel row, 25MHz clock and character address to read from the char ROM and output the bits needed to display on the screen. This is accessed by both the menu and game entity.
Game: This entity is responsible for generating the selected game mode. It generates the ball, paddle, score, timer and text. Depending on the mode selected the respective and appropriate actions are made.
VGA Sync: Takes in RGB values to be displayed at the current pixel column and pixel row of the display and the 25 MHz clock as the DE0 board can only support standard VGA resolution (640x480 pixels, at 25MHz). It syncs the VGA signal to be sent out of the VGA port to the monitor.
Clock Divider: This takes an input clock running at 50MHz and divides it by two to get an output clock running at 25MHz. It does this by inverting its output at every rising

edge of the input clock.

IV. Game Strategy

The user will play the game using the PS/2 mouse which will be connected to the DE0 board. The user will be controlling the paddle/platform with the PS/2 mouse.The game mode can be selected by moving Switch 0 and then pressing Button 2 when on the desired mode. Once in the game Switch 2 can be used to pause and Button 0 to exit. The aim of the game is to pass **ALL** the levels before the two-minute timer ends. As the levels progress the game speed increases causing the game to be more challenging. The challenging aspect of the game ensures a better user experience as they are less likely to lose interest in the game.

A. Level 1

In the first level the user is responsible to move the paddle and collect 30 balls. Every time the player collects a ball the score will increase. If the user is able to collect the 30 balls they move onto level 2.

B. Level 2

In level 2 the paddle/platform the user controls become relatively smaller in terms of length. Furthermore, a new ball is introduced which is relatively more faster when compared to the first ball. Therefore, it becomes relatively more difficult for the user as they have to track and collect two balls. The game is programmed in such a way that if the user decides to collect one ball in level 2 he/she won't have enough time in level 3 to collect the required balls and win the game. If the user collects 30 balls they move onto level 3.

C. Level 3

In level 3 the paddle gets even smaller, and an additional ball is introduced. Therefore, there are a total of 3 balls moving at 3 different speeds. The final level is designed to be challenging as the user will only have a small amount of time to collect the remaining 30 balls. If the user is able to collect an additional 30 balls (Total score of 90) within the allocated time then the user wins the game. You win will display if the user wins and game over if they lose.

V. Final System Design and Implementation

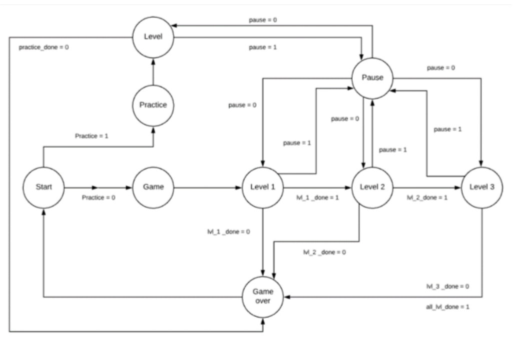


Fig. 2. Finite State Machine

The Finite State Machine (FSM) is made up of 8 main states; Start (main menu), Game, Practice, Level1, Level2, Level3, Pause, Game over. The FSM is used in the game to keep a track of game modes. Which is then used to determine which graphics are to be used and what text to display on screen. The FSM also has an input form the FPGA board, which is used to check is the game is paused or not.

Game component is the main component in this project as most of the logic is done in there. Game has a total of 7 processes, timer, ballDisplay, gameOver, youWin, barDisplay, collisionDetection and textDisplay.

The timer process creates a 2 minute countdown for the game and depends on the signals pause, gameOver, youWin and gameOn. The signals are set to 1 or 0 in their corresponding processes.

The ball display process is what creates the ball and displays it on screen. Ball display is dependent on gameOn, gameMode, gameOver, youWin and score. Each game mode has different conditions for ball display, as practice mode just has one ball being displayed whereas in single player mode, the number of balls increases as the level increases, which also depends on the score.

The gameOver process sets he signal gameOver to 1 or 0 depending on the game mode, the timer and the score. In practice mode, if the score reaches 90, gameOver is set to 1. In single player mode, when the timer hits 0 and score is less than 90, gameOver is set to 1.

The youWin process is similar to gameOver, except you dont win in practice mode. In single player, however, when the score reaches 90 and the timer is not 0, youWin is set to 1.

barDisplay is similar to ballDisplay as they are both same in both the game modes and the process depends on the game mode. The major difference is that this process takes an input from the mouse component. The mouse component outputs the column of the mouses position which is used to move the bar around in the game. As levels increases, the size of the bar gets smaller to make the game a little more challenging and interesting.

collisionDetection is the main part of the game as this is what the score depends on. This process checks if the ball is hitting the paddle or not. This depends on the size of the paddle and the radius of the ball and using the pixel row and column the process determines if it is colliding. The score increments on collision. The process also sets all values to initial if the gameOn is ever 0, which means if users exits to the main menu.

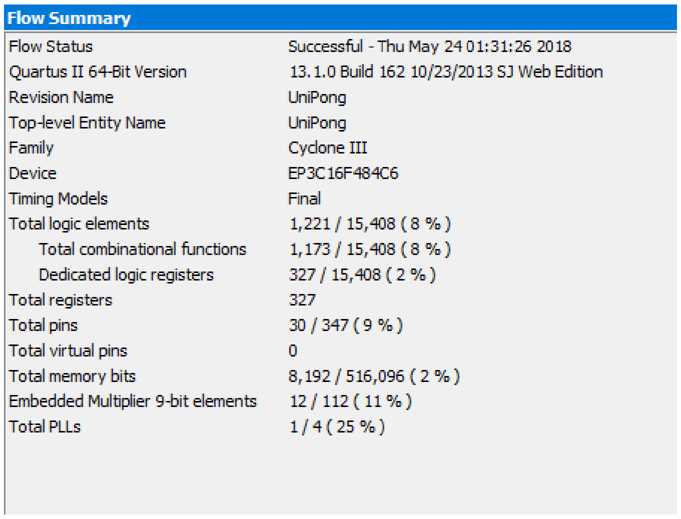
textDisplay is also a main process as all text is displayed from within this process. The displaying of text depends on the gameMode, gameOn, gameOver, youWin, pause and titleOn. gameMode is the selection between Practice of Single Player, which is used to determine which text to highlight. gameOn is to check whether the user is on the main menu or not, which is used to display necessary texts such as score, time and levels. gameOver is used to display Game Over screen and similarly youWin is used to display the win screen, and titleOn decides whether to display the game title or not. Using all these conditions, the process decides which text(s) to display and sets the signal charOn to 0 or 1 which is responsible for actually displaying the text.

VI. Design Decisions

We designed the game with the intention of writing the least amount of code possible and using the least amount of LEs. So our games main attributes are handled in the game component which has several different process which handle different tasks, as opposed to having a separate component for each task. This allowed us to minimise the amount of code and maximise the usage of code while also minimising the wiring required between components. The game component calls in all other components such as mouse, FSM and LFSR for the functionality of the game. We tried to keep the design of our game as less complex as we could so it is easier to understand and work with.

VII. Results

A. Statement of Resource Consumption



Flow Summary	
Flow Status	Successful - Thu May 24 01:31:26 2018
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	UniPong
Top-level Entity Name	UniPong
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	1,221 / 15,408 (8 %)
Total combinational functions	1,173 / 15,408 (8 %)
Dedicated logic registers	327 / 15,408 (2 %)
Total registers	327
Total pins	30 / 347 (9 %)
Total virtual pins	0
Total memory bits	8,192 / 516,096 (2 %)
Embedded Multiplier 9-bit elements	12 / 112 (11 %)
Total PLLs	1 / 4 (25 %)

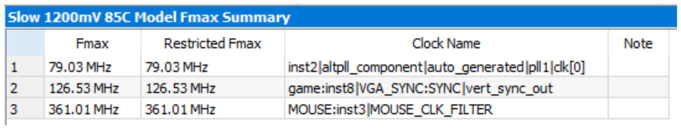
Fig. 3. Flow Summary

The implementation of this project will require 1,221 Logic Elements (LE) out of the 15,408 LE available in the Cyclone III FPGA. From the 1,221 LE's used, 1,173 are combinations functions whereas the remaining 327 are dedicated logic registers.

In addition from the 516,096 memory bits available only 8,192 were used in this project. These bits are used to store the character ROM which are displayed on the screen. This was the only data stored on the memory part of the FPGA.In order to read/load this data a mif file was used which had been provided to us.

As shown in the figure above only 30 pins were used. These pins where mostly used by the VGA and 7 segment display.

B. Timing Analysis



Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	79.03 MHz	79.03 MHz	inst2|altpll_component|auto_generated|pll1|clk[0]	
2	126.53 MHz	126.53 MHz	game:inst8|VGA_SYNC:SYNC|vert_sync_out	
3	361.01 MHz	361.01 MHz	MOUSE:inst3|MOUSE_CLK_FILTER	

Fig. 4. Maximum Operating Frequency Summary

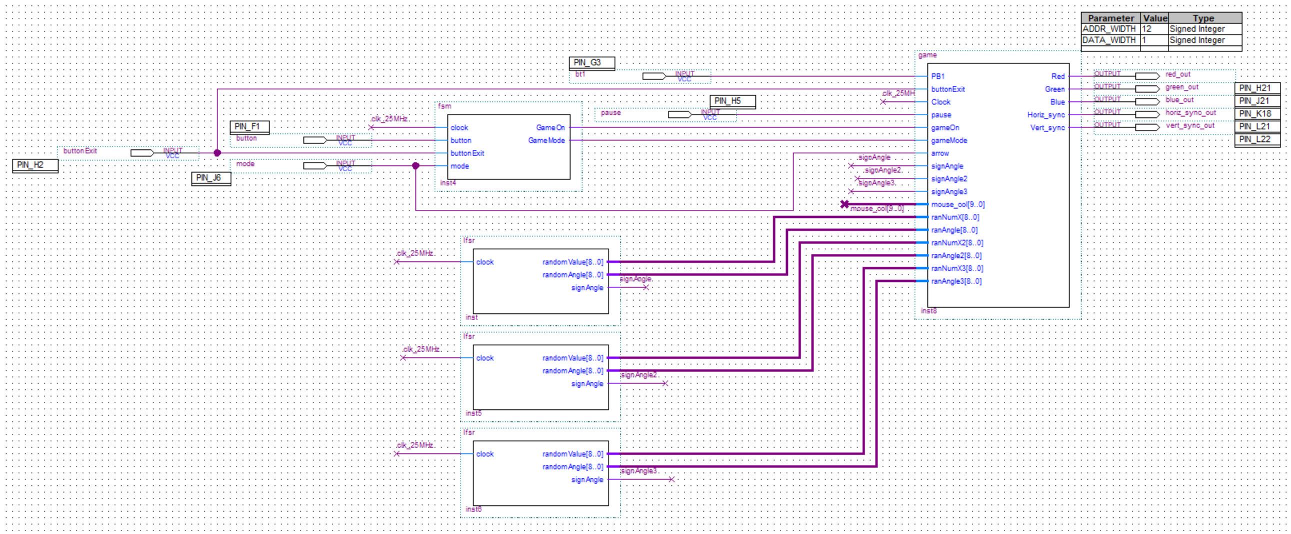
The maximum frequency of this project is 79.03MHz. The maximum operating frequency of our design is greater than the FPGA's clock frequency of 25MHz. If the frequency of the FPGA was to be greater than our operating frequency then there would be serious problems and hence the game will not be up to standard. As our clock frequency is much greater than FPGA frequency it means that this is an ideal situation.

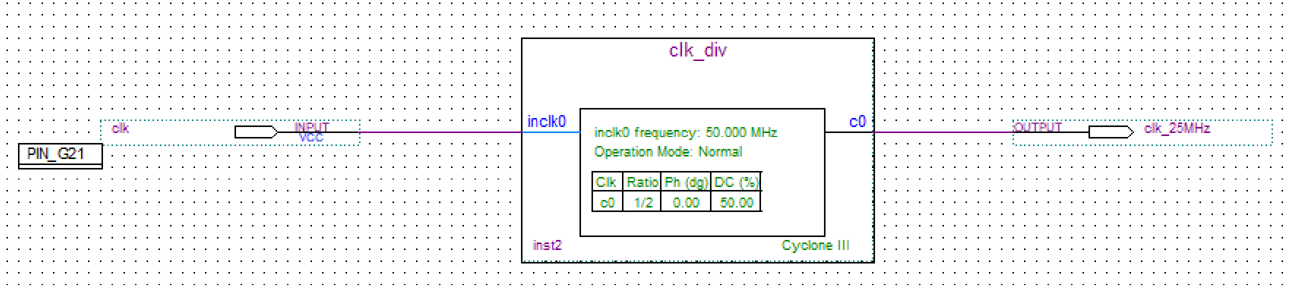
VIII. Summary and Future Improvements

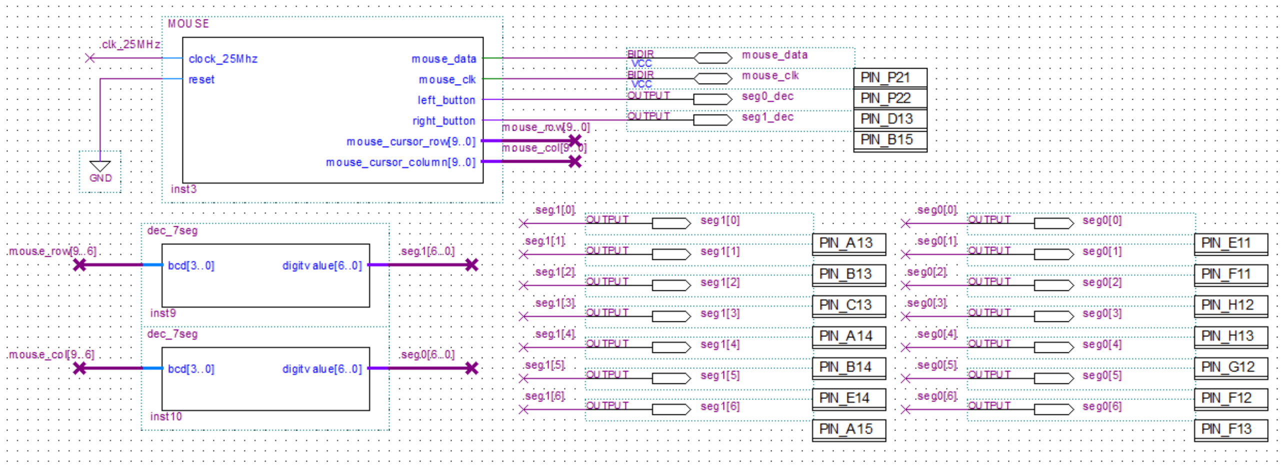
The project was to create simple Pong game implemented with a FPGA device. The minimum requirements set by the clients were all met. However, the game can be made more interesting and challenging in the future. Firstly, the game was displayed using a VGA connection, this could be improved for future by using a DVI or a HDMI connection for better resolution. The game contained three levels in the single player mode, which can be upgraded to 5 or even 10 to make the game challenging. The use of dip switches can be used to change color schemes during the game is being played so users can choose a color of their own choice. Something more challenging for an improvement can be a multiplayer mode, where users can connect with friends and play either via LAN or locally using a keyboard and a mouse. Another additional feature is a possible addition of power ups/downs, where they either increase speed or make paddle longer. The aim of the game is to entertain and these additional features would ultimately make the game more fun.

IX. Appendices

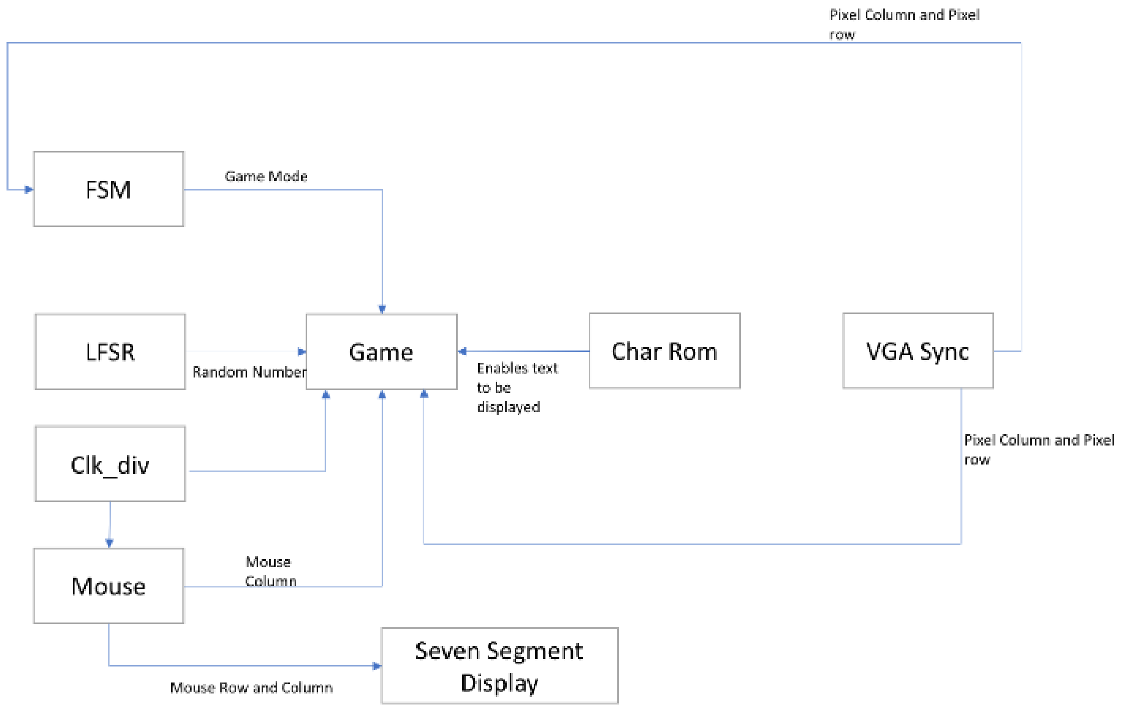
A. Extended Block Diagram



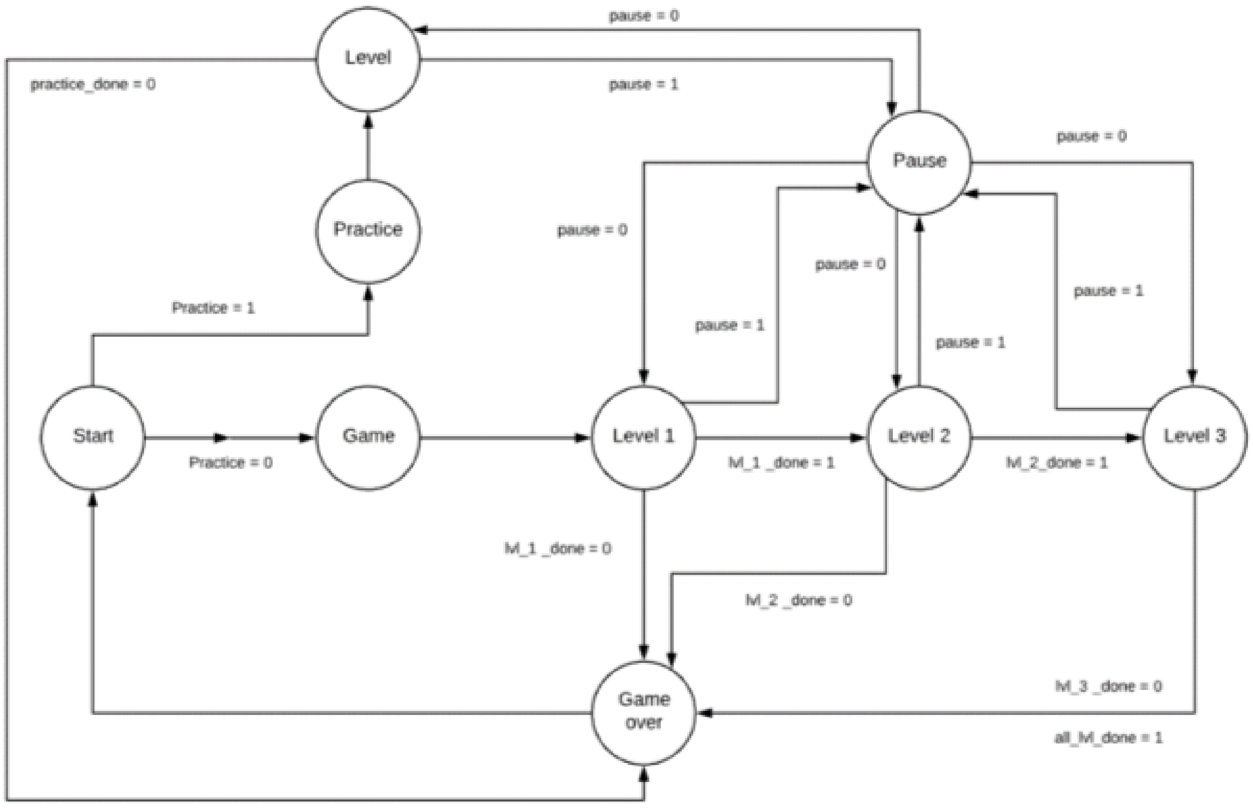




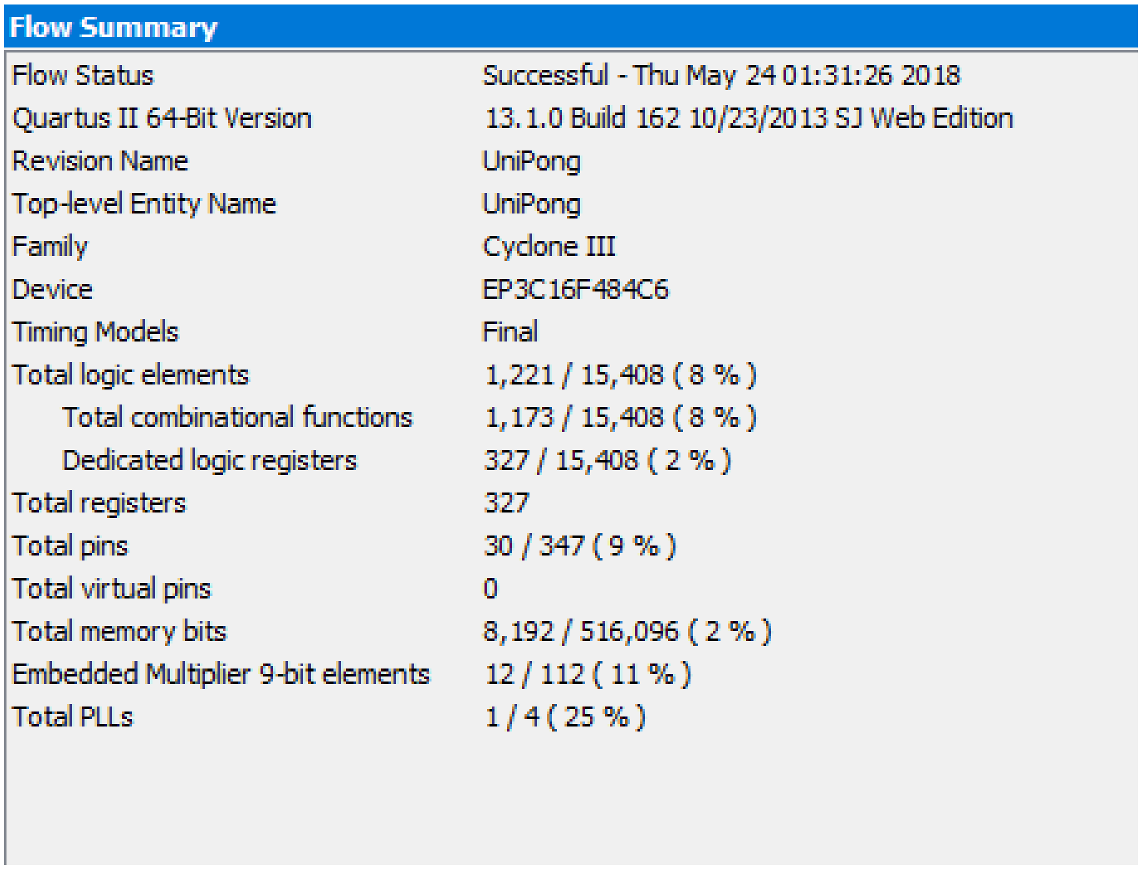
B. Figure 1 Enlarged:



C. Figure 2 Enlarged:



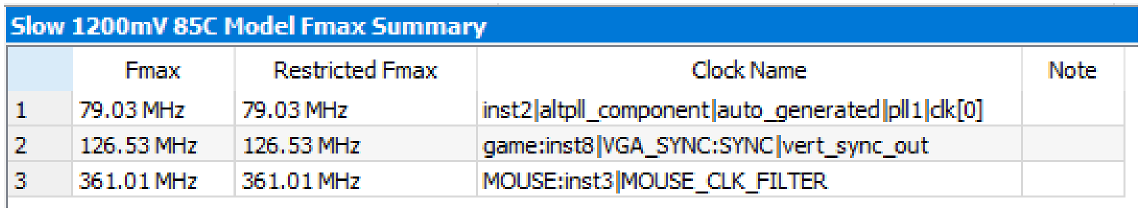
D. Figure 3 Enlarged:



Flow Summary

Flow Status	Successful - Thu May 24 01:31:26 2018
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	UniPong
Top-level Entity Name	UniPong
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	1,221 / 15,408 (8 %)
Total combinational functions	1,173 / 15,408 (8 %)
Dedicated logic registers	327 / 15,408 (2 %)
Total registers	327
Total pins	30 / 347 (9 %)
Total virtual pins	0
Total memory bits	8,192 / 516,096 (2 %)
Embedded Multiplier 9-bit elements	12 / 112 (11 %)
Total PLLs	1 / 4 (25 %)

E. Figure 4 Enlarged:



Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	79.03 MHz	79.03 MHz	inst2|altpll_component|auto_generated|pll1|clk[0]	
2	126.53 MHz	126.53 MHz	game:inst8|VGA_SYNC:SYNC|vert_sync_out	
3	361.01 MHz	361.01 MHz	MOUSE:inst3|MOUSE_CLK_FILTER	