ECE4110/5110 -- Digital Systems Design

Fall 2021 Project Specification

Any HDL developed should follow the *best practices* for writing your VHDL given in the ECE4110/5110 VHDL Guidelines. All HDL should be documented thoroughly about its use and limitations. All HDL should be accompanied by a testbench that exercises the design fully (or in a logically complete manner for large designs). Testbenches should have judicious use of **assert** and **report** statements to clearly indicate proper function of the UUT.

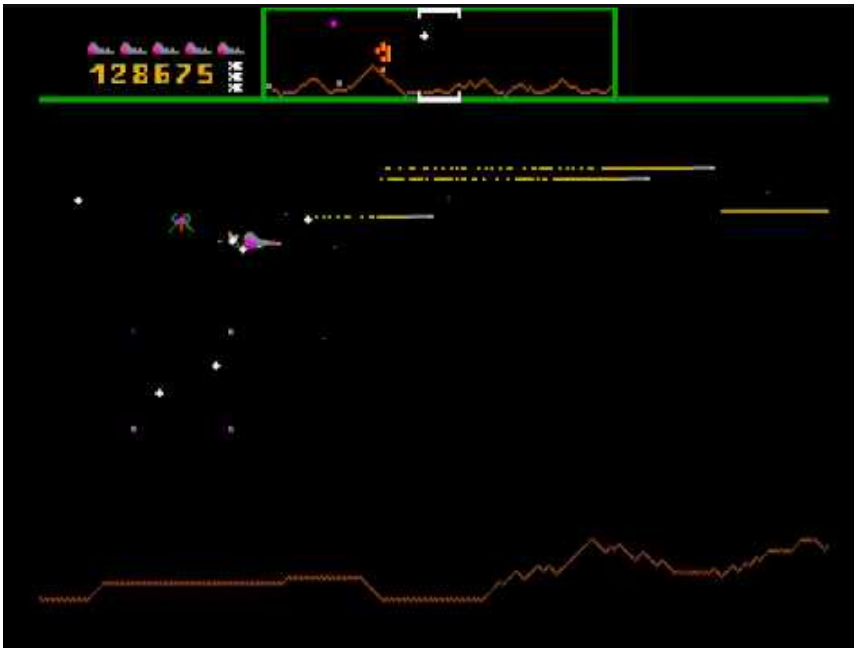
Development of any "logic" primitives, e.g. gates, flip-flops, larger basic blocks, etc. should be incorporated into your local copy of the ttu.vhdl library. *You might find these useful for your project.*

This homework must be submitted to iLearn in accordance with the Quartus Submission instructions. Make only one submission per team.

Tutorials and assistance for the DE10-Lite have been provided.

Defender

Defender was one of the most popular arcade games of all time, and is considered one of the best arcade games from the "Golden Era of Gaming" aka 8-bit arcade machines. Defender exhibited all the key characteristics of a great game: simple, snappy, fast-paced, and surpringly difficult to master. Your goal is to create a simpified and stylized version of this classic arcade game from the "golden age of gaming". A screenshot of the original Williams Defender arcade game is shown below:



FPGA Defender

Using your IP developed in previous homeworks, develop a VGA (640x480 @ 60Hz) application that allows for single player Defender gameplay. The real defender game incorporated a gameplay map (top-center) and the player's ship could scroll the game view in both X directions. These features are not required in your version of the game.

> **Note**
>
> Deviations from the original Defender arcade game
>
> - No gameplay map is required
> - Your game will always scroll to the right.
> - Steering the ship to the left will simply move the right-facing ship toward left edge of the screen. The game will not scroll left.
> - The player's ship is restricted to be located only in the left 1/2 of the game screen.

In this game, the player will use the DE10-Lite board itself as the controller. The player will command X-direction and Y-direction movement by tilting the board from horizontal. Pushbutton **KEY0** will serve as the trigger to fire the forward cannon.

The goal is to destroy all of the enemy aliens and obstacles that come toward the player's ship. Alien and ostacles will always originate from the right screen-edge. If the player's ship touches any alien or obstacle, the player's ship is destroyed. When the player runs out of their initial allotment of three ships, the game is over. As the player successfully progresses through the game, aliens and obstacles will move faster and become incresingly more numerousl. Score will be recorded for each alien or obstacle destroyed, and displayed in the upper-right corner of the screen. Simply avoiding an alien or obstacle will not result in any points.

You may **optionally** choose to award the player with an extra ship after the player earns some (appropriate) number of points.

Williams Defender game action

Game play starts when the player presses **KEY0**.

Unlike the real game, your version will allow the player to "pause" the game by pressing **KEY1**. Game play will stop with the graphics frozen. Pressing **KEY1** will resume gameplay.

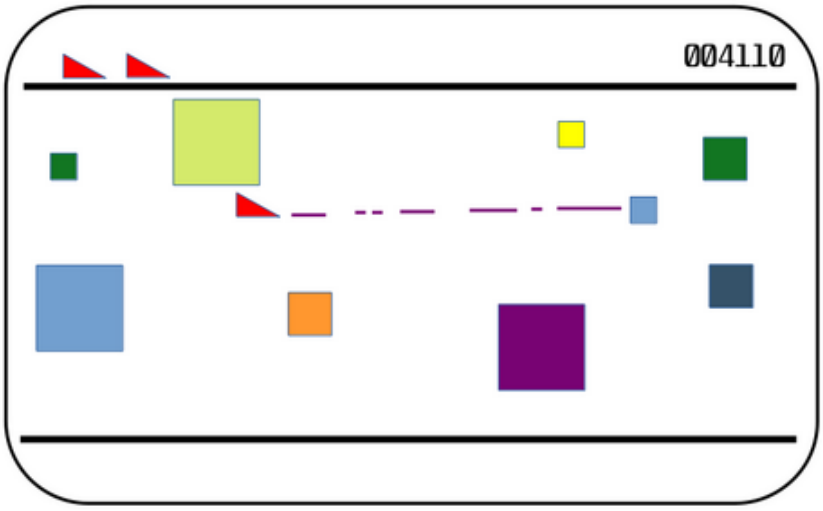
Appropriate sound effects shall be created on the piezoelectric buzzer connected between **ARDUINO_IO12** and **GND**.

Basic Defender game mode (PROJ0 -- REQUIRED)

The player's ship is a right triangle. Alien and obstacles must include squares. Alien and obstacles can also be other uniform polygons: equilateral triangles, hexagons, etc. Alien and obstacles must be created in different sizes. Smaller targets are worth more points. You must implement at least three different target sizes. Alien/obstacles may be different colors, but may not be the same color

as the player's ship. The player's ship must be clearly distinguishable at all times. You may choose to make targets of certain colors worth more than other colors. Cannon fire tracers should be clearly identifiable from the player's ship and all other objects.

A notional diagram of your FPGA Defender screen is



The image above would represent a *minimum* visual standard for your game. You may render your game graphics in a *just-in-time* pixel-by-pixel manner similar to the DE10-Lite VGA demonstration project provided on the class website.

Deploy your *proj0* deliverable into the first bitstream location in the configuration memory of your DE10-Lite.

BONUS DELIVERABLES

B0: Framebuffer framework

Implement your game using ping-pong framebuffers -- two memory arrays that contain the display information. Your design will draw the screen using the contents of one framebuffer (FB0). Meanwhile, your design is concurrently "drawing" the next frame data into the other framebuffer (FB1). Upon the completion of a drawing FB0 to the screen, the roles of the two framebuffers are swapped. The HW will draw FB1 to the screen, while FB0 is erased and being filled with pixels for the next frame. Rinse and repeat. Using the the DE10-Lite on-board 64Mb SDRAM, implement a ping-pong framebuffer scheme for your game.

Your approach should include framebuffer framework IP where you incorporate the provided SDRAM controller IP with the VGA controller IP. The VGA display is updated from a portion (framebuffer) of the SDRAM while another portion of SDRAM (another framebuffer) is being written by the game grfx render hardware. Your framework should include a clear and localized location(s) wherein "game rendering" hardware is described by the designer.

B1: Terrain

Implement an interesting but no-collision terrain along the game play lower screen like the Williams arcade Defender game.

For this part, capture your submissions in a folder/project called *proj1*. Prepare this folder and its files in accordance with Quartus Submission.

B2: Shooting aliens

Implement aliens that "return fire". These aliens randomly deploy torpedoes in random directions generally toward the player's ship. Not all aliens should return fire. (Maybe only ones of a particular color and/or shape?) The alien torpedoes may have varying velocities. Obviously, collision detection must be implemented and player ship-torpedo collisions result in player ship destruction.

For this part, capture your submissions in a folder/project called *proj2*. Prepare this folder and its files in accordance with Quartus Submission.

B3: Player ship exhaust and left-facing ship

Implement player ship exhaust proportional to player ship ship velocity. No movement means no exhaust. Fast movement results in large exhaust plume. Furthermore, allow player ship to *reverse* direction, *i.e.* face left if moving left and face right if moving right. Stationary ships will face the direction of last movement. The base requirements on ship movement and scrolling remain: *(i)* The ship must still remain in the left one-half of the play area, and *(ii)* game scrolls only right.

For this part, capture your submissions in a folder/project called *proj3*. Prepare this folder and its files in accordance with Quartus Submission.

B4: Explosions

Implement explosion graphics and sounds when an alien or obstacle is destroyed.

For this part, capture your submissions in a folder/project called *proj4*. Prepare this folder and its files in accordance with Quartus Submission.

B5: Significant audio FX and background music

Implement modular FPGA IP to easily allow for creation of many and varied sound effects and "music" playback on your piezoelectric buzzer. Clearly document your IP and use in your design to highlight its capabilities.

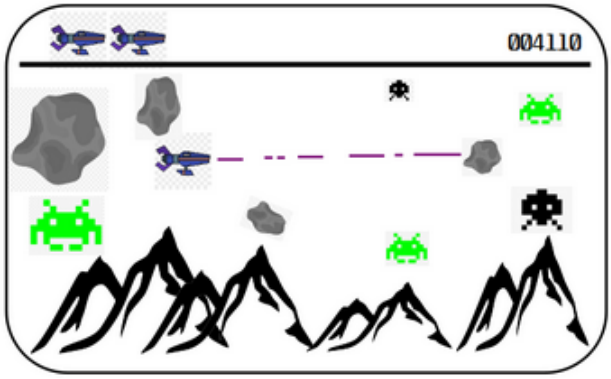
For this part, capture your submissions in a folder/project called *proj5*. Prepare this folder and its files in accordance with Quartus Submission.

B9: New game theme and visuals

Develop a new theme for your game. Game action will include base specification plus all bonuses submitted. Instead of simply geometric shapes, your game should feature graphical icons that

convey a story or theme for your game. Make your sound effects match accordingly.

Additional points will be awarded for creativity, pleasing and/or interesting aesthetics.



For this part, capture your submissions in a folder/project called *new*. Prepare this folder and its files in accordance with Quartus Submission.

GRADING

Project are graded on a 100-point scale. A base project should be deployed in the first (default) bitstream location of your DE10-Lite. If you pursue any bonus specification, place your *best* bonus into the second bitstream location.

No more than one-half of the teams can pursue the "Base project with B0 option". You must notify Tyler and Dr. Bruce if this base project is your team's intent. They can provide some additional guidance as to the the desired approach for the framebuffer framework.

The maximum *potential* project points will be determined by the correctly-functioning project options:

Option (bonuses require succesful base project)	**Points (max)**
Base: *proj0*	90
Base: *proj0* with B0	100
B1 with a successful Base project	+5 pts
B2 with a successful Base project	+5 pts
B3 with a successful Base project	+5 pts
B4 with a successful Base project	+5 pts
B5 with a successful Base project	+5 pts
B9 *new* theme (must incorporate all created bonuses)	+5 pts

Submissions are due 23 November @ 2359. Submit via iLearn.

Good luck.

REVISION History:

- 22 October -- final design specification

[simulated]	The ModelSim tool we have does not calculate timing delays for the Cyclone V chips on the DE10-Standard or the MAX10 device on the DE10-Lite boards. See the hints on doing gate-level simulations at *Quartus timing*.