Keep Talking and Nobody’s FPGA Explodes!!!

6.111 Fall 2016 Project Proposal Amelia Becker & Mitchell Hwang

1 Overview

Keep Talking and Nobody Explodes is a bomb defusal game created by Steel Crate Games. In this game, cooperation is demanded between players, whom are divided into two teams, one that sees the bomb and one that sees the manual to defuse the bomb. The defusal team is tasked with defusing the ticking time bomb, which has several individual modules, and upon failing three of these modules, or unsuccessfully finishing the modules before time runs out, the bomb explodes. The consulting team, based off of a highly descriptive and step-by-step manual, will instruct the defusal team on how to deactivate that bomb. This fun and innovative game is limited to virtual interactions, thus lacking in a real life tangible user interface.

In Keep Talking and Nobody’s FPGA Explodes, we will implement this game using an FPGA’s I/O pins, buttons, and switches for real time manipulation of the time bomb. This version will overcome the lack of tactile feedback, and will bring the modules into the physical world. In regards to gameplay, we plan on building modules that simulate the modules in game as well as our own unique modules. We will interface the FPGA with more peripherals such as a keypad, a big red button, and jumper wires to help create the illusion of a bomb defusing environment. The bomb itself will also be projected on a computer monitor. The events that occur in real time in accordance to the user’s input will then be processed by the FPGA and will be updated accordingly on the monitor.

Our main goal for this project is to implement a functional game with variable bomb setups on the FPGA which would be interfaced with a mouse, computer monitor, numerous peripherals to simulate the modules and the bomb itself. Our game will have a mixture of modules found in the original game as well as our own modules. The basic version of Keep Talking and Nobody’s FPGA Explodes will be a functional game with a fixed bomb setup, displayed and updating on the computer screen. Stretch goals include using memory to implement levels, the ability to randomize bomb modules, and a fully functional setup with the FPGA.

2 Design Decisions

Keep Talking and Nobody Explodes is inherently modular. We decided that one FPGA will suffice for our project. For our game, we will have an overall skeleton to hold all the modules, which includes the Game FSM, Bomb Logic, Bomb Modules, etc. Besides interfacing our labkit with numerous peripherals, we will also interface with the computer monitor to display a virtual bomb, that will not only contain important clues to defusing the bomb but also be used to correctly setup the bomb modules on the labkit itself.

3 Implementation

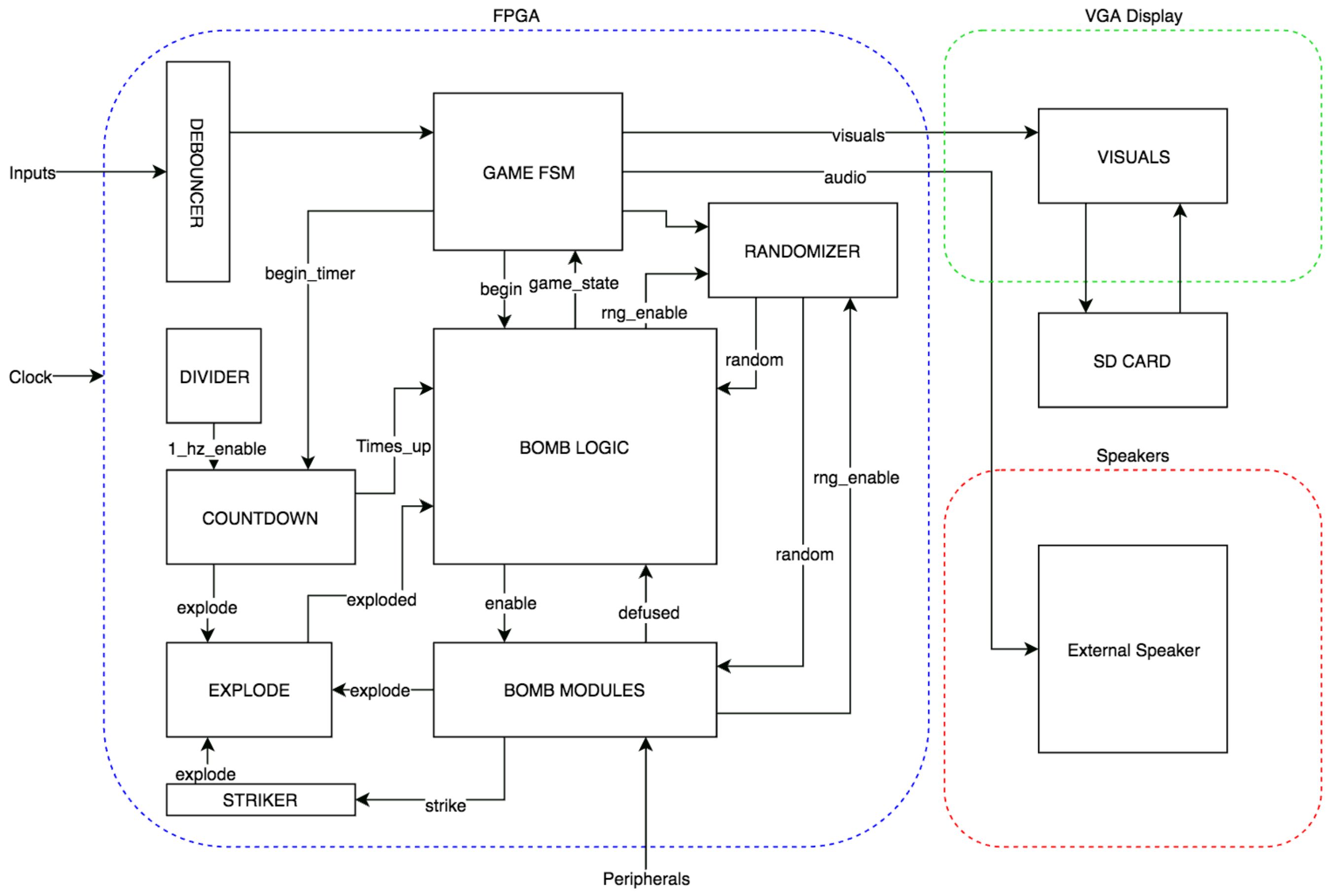


Figure 1: Block Diagram for Keep Talking and Nobody's FPGA Explodes.

3.1 Bomb Logic Block

The game logic is a state machine. The interconnections with the bomb module is shown in figure 3. The module will be receiving inputs from the bomb modules (whether they are defused or not). The main states are waiting for a new game, the game set-up, playing the game, and the states where the game is lost or won. The state transitions are described in figure 4. This module will link to the "strikes" module to track the times the module is messed up, and the explode module when the game is over.

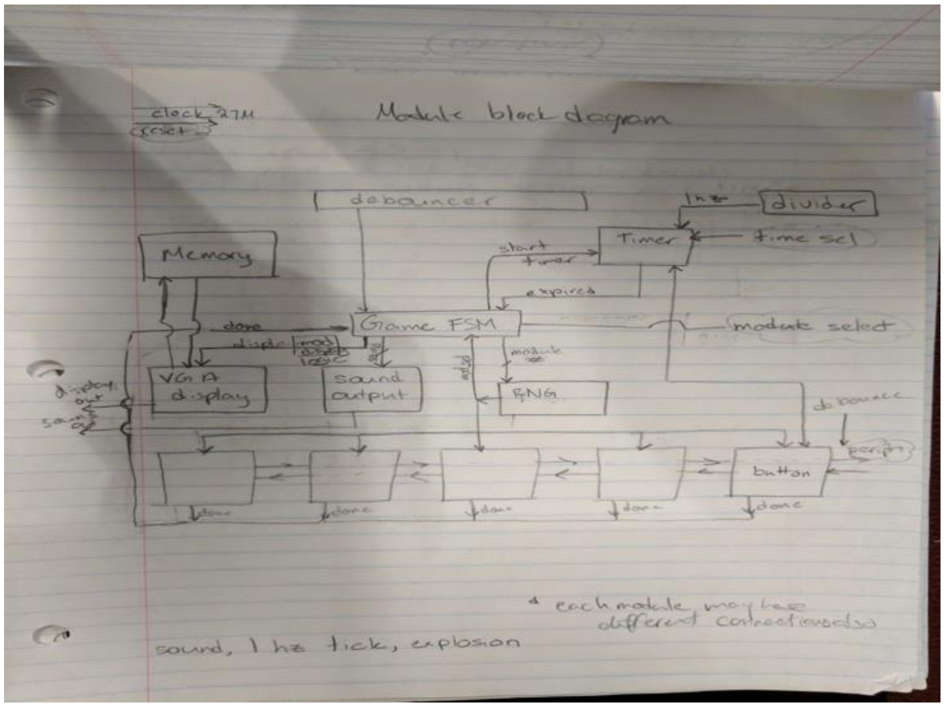
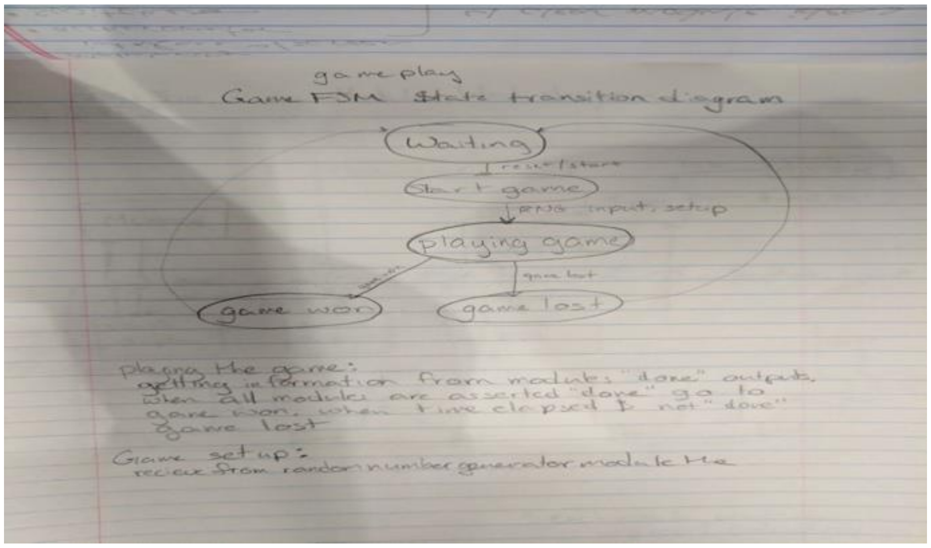


Figure 2: Game FSM interconnections



playing the game:
getting information from modules "done" outputs.
when all modules are asserted "done" go to
game won. when time elapsed & not "done"
game lost

Game set up:
recieve from random number generator module the

Figure 3: Bomb Logic FSM State machine diagram

3.1.1 Game set-up

The user of this game gets to select the number of modules on the bomb and the amount of time (we will have a default of 4 modules and 6 minutes). Given the user inputs and the “start” command (from a button) the game set-up will get a random number from the RNG module and configure the bomb. Figure 4 shows what the bomb will look like on the display.

Slot #1	Slot #2	Slot #3
Slot #4	Slot #5	Slot #6

Figure 4: Bomb configuration on display

3.2 Computer Monitor Block

Each module for the bomb has its own data output for displaying its features on the screen. These will be placed on the screen according to the locations dictated by the “game module” by scaling the “origin” point for each will be the top left corner and then we adjust based on which slot the module to be displayed is on. These sprites for each of these modules will be combined to display on the VGA output of the labkit. When a module is completed, instead of being displayed as its components, it will be displayed as a “completed” graphic.

3.3 RNG Block

This block will be used to help configure the modules. It will take the last 3 bits of a floating analog sensor for a pseudo random generation. This will constantly produce a number between 0 and 7 for the other modules to use.

3.4 Individual bomb modules:

These are the fundamental pieces of our game. These connect to the bomb logic module and then the peripherals that they need as inputs. Each one will send an output to the the VGA output block for processing the visuals, and an output to the game logic module for its state (incomplete, complete, or erroneously manipulated--which will ultimately count as a strike against the user and the end of the attempts to solve that particular module). A few examples of modules include switches, accelerometer, LED simon, cut wires, and big red button. Module ideas and details can be found below.

3.4.1 Switches

The connections of the switches module will be inputs from the switches on the labkit, the RNG module, and from the game bomb logic fsm. Its outputs will be the “progress” line to the game FSM (triggered, messed up, or defused) and the output to the display. This will only track whether the switches are all in the correct position for the desired output (based on the random number received).

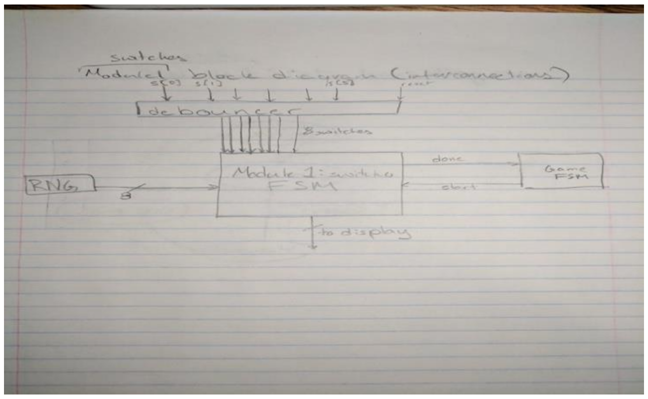


Figure 5: The block diagram for the switches module

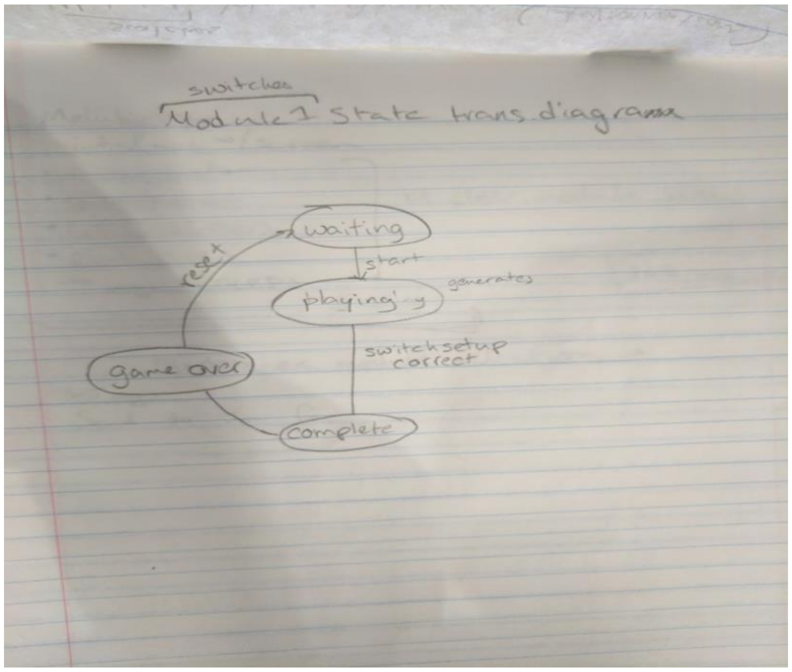


Figure 6: The states within the switches module

3.4.2 Accelerometer (I2C interface)

The accelerometer/gyroscope module will have a dice-like interface. The objective will be to put the correct side facing up for a specific amount of time. The movement of the die will be tracked using an accelerometer as input.

3.4.3 LED simon

LED simon will be a brain game. This will have four buttons associated with either LED's or colors and positions on the screen. The point is to correctly replicate the pattern that is given by the module. The push-button inputs will be tracked for sequential correctness.

3.4.4 Cut wires

No bomb defusal game is complete without real wires. This physical interface will present the user with a variety of wires (of various colors) on a breadboard. The user will need to physically cut (or disconnect) these wires in a specific order to defuse the module.

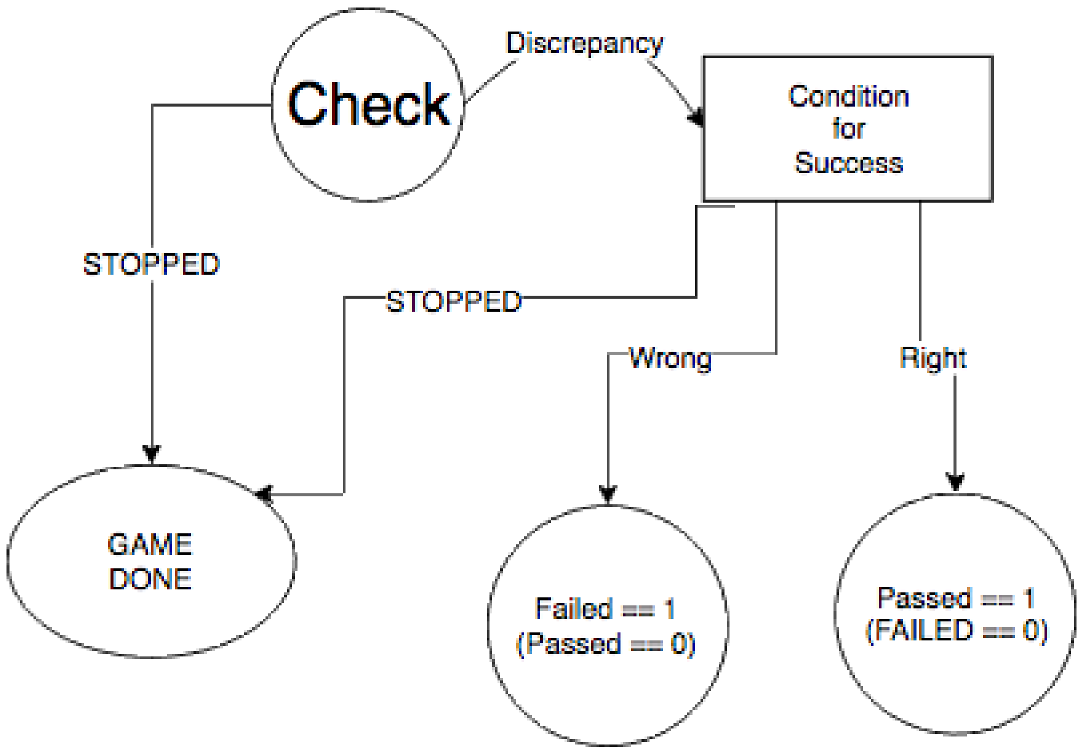


Figure 7: Wires Module

3.4.5 Big Button

This is a simple module. Once the user presses the button, it should not be released until a certain number is displayed on the countdown. The instructions will have details about the conditions when the button can be released.

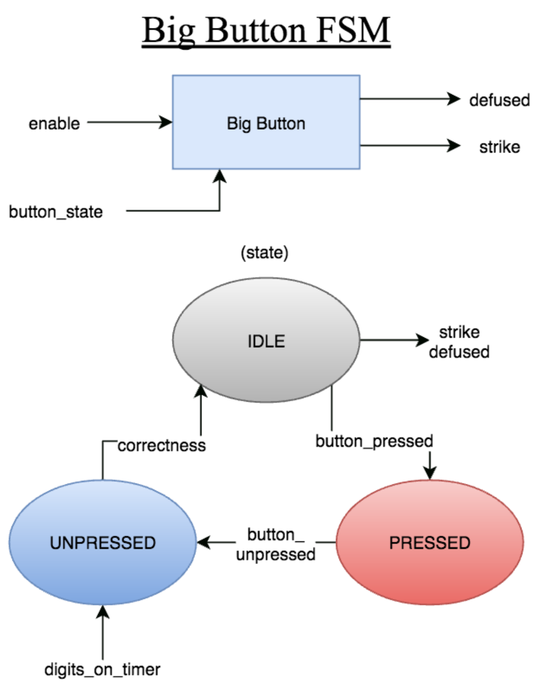


Figure 8: Big Button Module

3.5 Strikes Module

The strikes module is a finite state machine that takes as input a signal called ‘strike’ sent by the bomb logic block, and outputs the signals might be ‘warning_lights’ and ‘explode’. The strike state machine has a default state of ZERO, and upon each assertion of ‘strike’, will change states and update ‘warning_lights’/’explode’ accordingly. The ‘warning_lights’ signal will be sent to the labkit to track the number of strikes they have accumulated. It will also be used to update the bomb image on the computer monitor. The ‘explode’ output signal will be sent to the explode module.

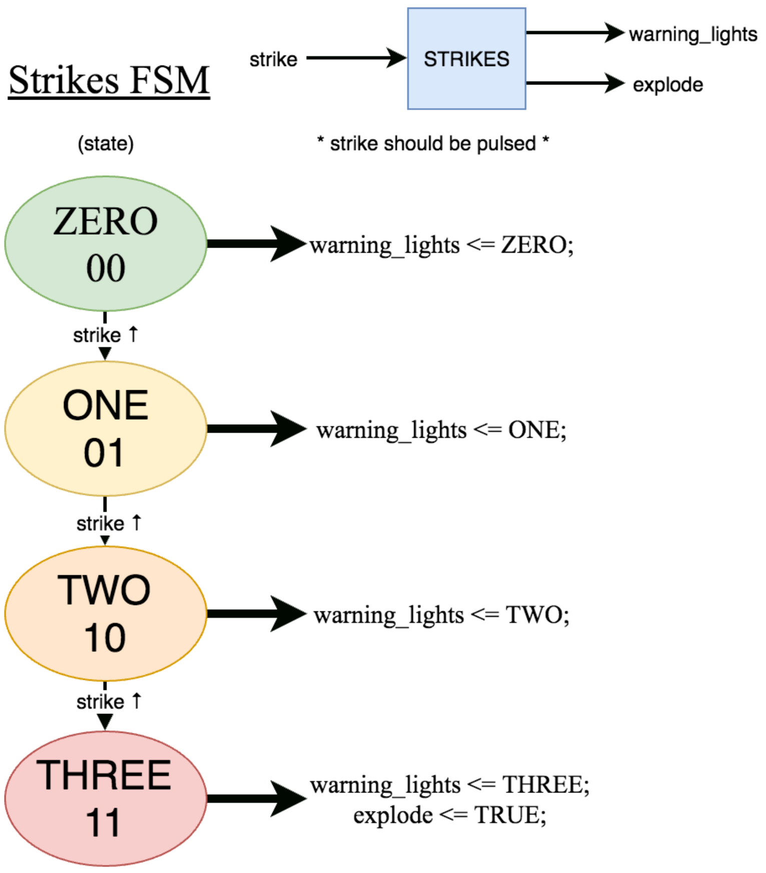


Figure 9: Strikes Module

3.6 Explode Module

The explode module operates on the clock(65mhz?) and takes as input a signal called ‘explode’, which will be asserted by either the ‘strikes module' or ‘countdown timer module’ (Also asserted if a needy module goes off). The explode module will output a ‘game_over’ signal that will be sent to the ‘bomb logic block’. Furthermore, the explode module will trigger the speakers to play an explosion sound.

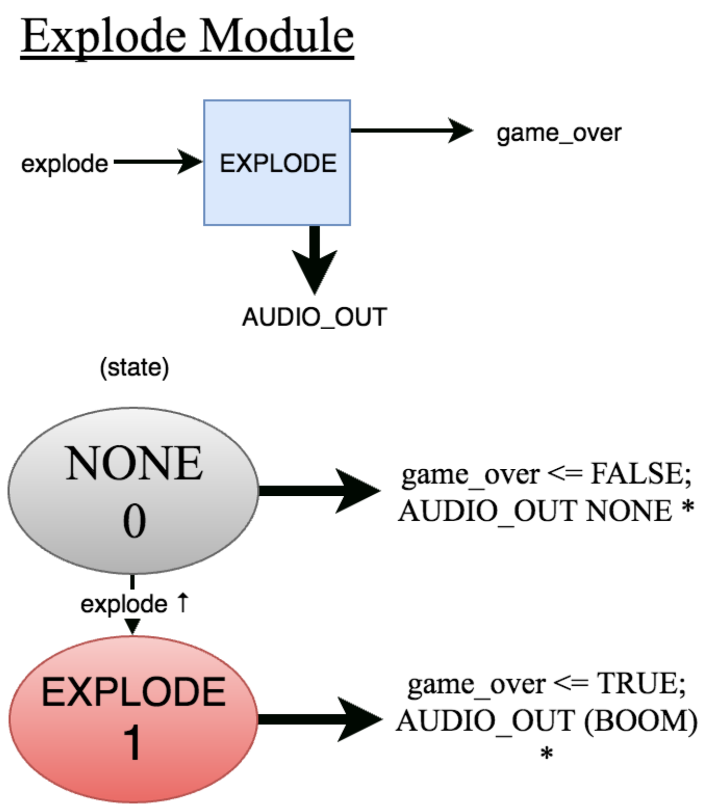


Figure 10: Explode module

4. Testing

Since this project is inherently modular, it will be easy to test each bomb module before adding it to the game. The other modules (the overall game FSM, RNG module, the display output, and sound output) will each be tested as well based on the specifications we write.

5. Timeline

a. Commitment: Functional game with fixed bomb setup, displayed and updating the computer screen.
b. Goal: Functional game with various bomb setups on the FPGA, interfaced with a mouse, computer monitor, speakers, and other peripherals for bomb manipulation. A variety of modules including self-designed modules.
c. Stretch goals: Using memory to simulate levels, adding a way to randomize the bomb setup, and a plethora of bomb modules.

Task	11/11	11/18	11/25	12/2	12/9
Game FSM/Bomb Logic	🟥	🟥			
Expert Manual	🟪	🟪	🟪		
Game Elements (strike, explode, RNG)		🟥	🟥		
Displaying on Monitor, SD interface		🟦	🟦	🟦	
Bomb Modules		🟪	🟪	🟥	
Bomb Module Peripherals			🟪	🟥	
Testing/Debugging			🟪	🟪	🟪
Stretch goals: speakers, randomness				🟪	🟪
Presentation, Checkoff, Final Report					🟪

Figure 11: Blue-Mitchell, Red-Amelia, Purple-Both

6. Resources

For our game we will be using the labkit and the monitor. Each module will include input from peripherals, based on what sorts of interfaces we can get. Examples include an LED Button Pad, a large button, a keypad, some standard LEDs, and a segmented LED bar graph. We are also thinking of using a keyboard, an accelerometer, and a wired breadboard interface.

7. Conclusion

This bomb defusal game allows the user to view a bomb puzzle on the screen and use inputs on the labkit to manipulate the bomb. This is a two player game as the user manipulating the labkit inputs will not have the instructions to defuse it, so it is also about teamwork and effective communication. The game will have the visual of the bomb on the screen and a countdown timer for the players on the LEDs. Each module in the bomb will be controlled with its own inputs and have wires to the main game FSM for its completion. This project will connect interaction and processing of peripherals to the labkit as well as a game state machine element and a vga display output.