Star Wars
Overview of the project

- Inspired by the classic game *Geometry Wars*
  - Various enemy flying round or chasing after the spaceship.
  - Player’s goal is to survive as long as possible and get a score as high as possible with 3 lives.
  - Bomb available to destroy all the enemies at once
- Overall 60 entities, first entity saved for spaceship, 2nd to 30th for bullets, and last 30 for enemies.
  - ID number indicating entity type
  - X, Y coordinates and direction information also contained in each unit data
Module: VGA_BALL

submodule: VGA_BALL_Emulator
VGA_BALL

- Receives 10-bit writedata in a total of 256 (2-reg structure)
- Combines every four of them to form the information for every object (in a total of 64):
  \[31:0\] logic data_to_emulator: [id, x, y, direction]
- Connects to the submodule VGA_BALL_Emulator to draw the graph
Flow Chart
(Processing state)
Receives data from the software:

2-reg structure
One for transmission
One for updating

reg [9:0] data1 [0:255];
reg [9:0] data2 [0:255];
VGA_BALL_Emulator

- Receives 32-bit object information and stores into 2 RAMs: One for updating, one for transmission.

- Stores the RGB value of every object into the line buffer (3 RAMs: one for updating, one for drawing, one for cleaning) according to the object information.

- Read the rom and draw the objects according to the RGB value.
VGA_BALL_Emulator

Flow chart of Line Buffers
Audio Implementation

- I2C protocol: data is sent a bit at a time over the SDAT wire, with the separation between bits determined by clock cycles on the SCLK wire.
- I2C is a master-slave protocol. In our project, the FPGA is the master and the audio codec is the slave.

Audio components:
- I2C controller: control the transmission timing, configuration interface.
- Configuration controller: determines what data to send--16-bit words. Use 19 9-bit regs to record configurations, the first 7 bits are the reg address and the last 9 bits are the register contents.

Reference: Exploring the Arrow SoCKit Part - The Audio Codec
Audio Implementation contd.

- Audio components (contd.):
  - Clocks: use Cyclone V’s Phase-Locked Loops to generate master clock for audio codec. Other bit clock and LRC are generated using frequency divider.
  - Audio codec driver: the data is pushed out or read in through shift registers.

- Audio output:
  - Receive flag information from software. Control production of sound.
  - The .wav file is converted into .mif and the data is stored in ROMs.

Reference: Exploring the Arrow SoCKit Part - The Audio Codec
Software and algorithms

- Overall game logic control
  - bomb detection
  - bullet generation
  - enemy generation
  - collision detection
  - units movement control
  - score, life, bomb data collection

- Sending array messages of 256 elements to hardware containing information of 60 entities and player data information (scores, lifes, bombs, etc.)
Experiences and Issues

- Game logic moved from hardware to software.
- Improved logic usage (34% to 17%) on the board.
- Better VGA display using sprite scheme.
- Treat the reg/ram as memory and ensure only to read/write one value from/into the memory at one clock cycle.
- After writing into the memory, the data could only be read out two cycles later. Thus the state for stabilize the data is needed.
Experiences and Issues

- Overlap:
  
  Solution 1: Change the C code to avoid overlap (not good)
  
  Solution 2: Use the line buffers to store the 32-bit information about the objects for each pixel (cannot solve this problem)
  
  Solution 3: Use the line buffers to store the RGB value (currently use)