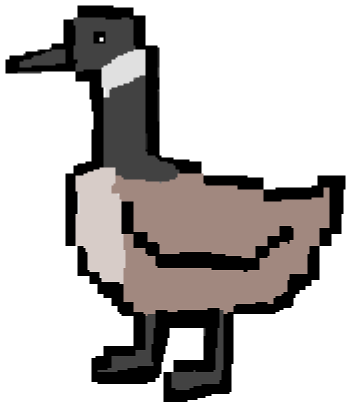
OIIA Goose

ECE 298A Project Proposal
Joyce Mai, Victoria Mak



Description

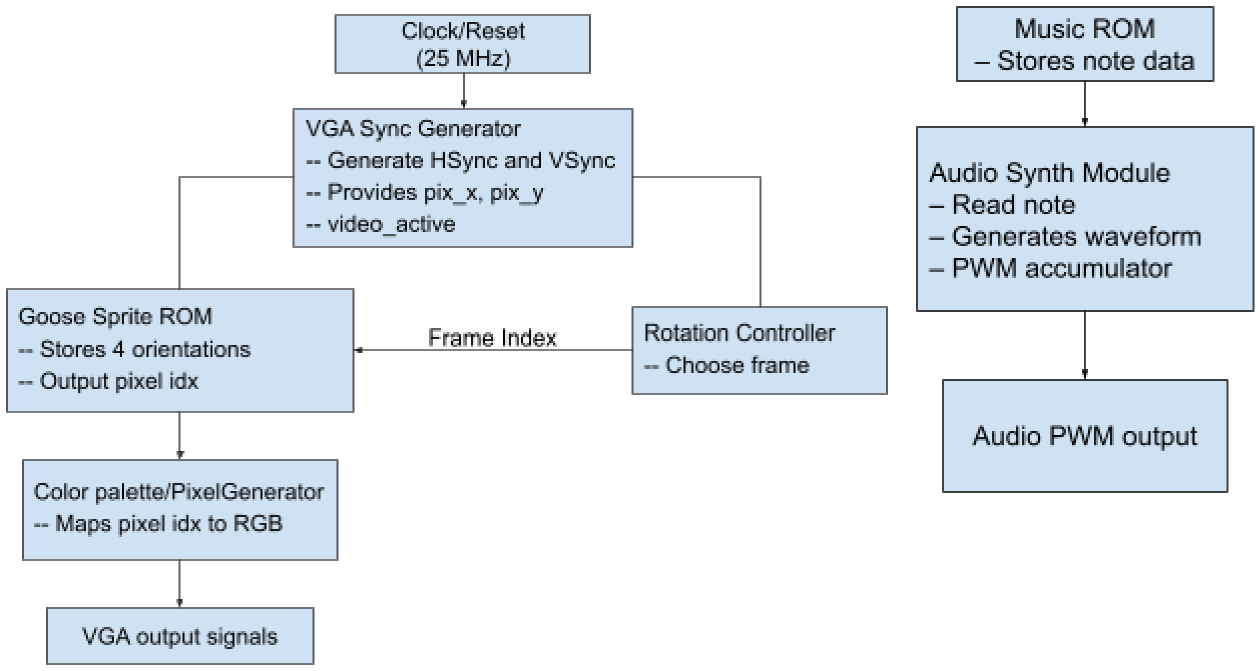
OIIA Goose, inspired by the popular "OIIA Cat", outputs a rotating goose on VGA with music. It shows a pixel Canada goose that rotates by cycling through four directions.
The display uses 2 bits per color channel (64 colors total) along with horizontal and vertical sync for VGA. We'll also add simple OIIA-style chiptune music through an audio PWM pin, so the goose spins with sound (hopefully :p).
The background can be toggled to one of four different backgrounds based on the user input pins. To put in a University of Waterloo easter egg, we will display the University of Waterloo logo in one of the backgrounds of the rotating goose. This user input allows for a better customization of the imagery.
Additionally, if the project timeline permits, user input pins can also be used to determine the direction of the goose spin. However, this is an optional feature of the demoscene.

With reference to the nyan cat demoscene project https://github.com/a1k0n/tt08-nyan, OIIA Goose will require the following components:

1. Scripts to prepare frames for the display hardware (the equivalent of extractcat.py or extractrainbow.py).
    a. It should generate *_r.hex, *_g.hex, and *_b.hex files to create indexed pixel data
2. Since two bits are used for the R, B and G channels each, a script is needed to perform the following (the equivalent of gamma.py):
    a. Linearize colors (srgb_to_linear) so brightness looks natural.
    b. Apply gamma correction (gamma_correct) to adjust perception.
    c. Pre-scale for dithering (dither_correct) so limited bit-depth still looks good.

3. Hvsync generator verilog code
4. Top level verilog code for the project

Block Diagram



TT I/O Assignments

Table of TT I/O Assignments (8 inputs, 8 outputs, 8 bidirectional I/O)

#	Input	Output	Bidirectional
0	BG0	R1	
1	BG1	G1	
2		B1	
3		VSync	
4		R0	
5		G0	
6		B0	
7		HSync	AudioPWM

Work Schedule

Week	Progress
Sept 24	**Task 1: Design Proposal and Counter Project Complete**
Oct 1	- Preliminary sketches, graphics, and any other data sources complete. Ensure the ROM data sources are completed Coding the scripts: Rotation controller, VGA sync generator, Audio synth module
Oct 8	Coding the scripts: Rotation controller, VGA sync generator, Audio synth module
Oct 13-17	Reading Week
Oct 22	**Task 2: Sub-block (verilog) evaluation - ensure that all verilog modules are completed by Oct. 22**
Oct 29	Synthesis and verification with Open Lane and CocoTB
Nov 5	Synthesis and verification with Open Lane and CocoTB
Nov 12	**Task 3: System integration - ensure that display and audio are working as expected**
Nov 19	Final verification complete - ensure that display and audio are working as expected, ensure that there are no errors on the chip
Nov 26	Documentation on github complete
Dec 3	Evaluation of final submission and documentation

Glossary

- **VGA** (Video Graphics Array): A standard for displaying images on a monitor. It's an analog video signal where you control:
  - RGB signals
  - **HSYNC** (horizontal sync pulse: tells the monitor when a line ends)
  - **VSYNC** (vertical sync pulse: tells the monitor when a frame ends)
- **Sync Generator**
  - It counts (keeps track of) pixel positions horizontally (x) and vertically (y)
  - Produces sync pulses so the monitor knows when to start new lines/frames
  - It tells the monitor when we're in the visible area (draw pixels) and when we're in the blanking area (no pixels, just syncing)
- **Sprite**: A two-dimensional image or animation that is integrated into a larger scene, often derived from bitmap images
  - It is usually stored as a grid of pixels. Each pixel isn't full RGB, but instead a color index (a small number like 0,1,2...) that points to the color palette
- **Color Palette**: A lookup table that maps a color index → actual RGB value
- **Pixel Generator**: The logic that decides what color should appear at the current pixel
- **PWM (Pulse Width Modulation)**: The PWM is to create an analog sound from a digital timing signal. The ratio ON/(ON+OFF) in time determine the analog value of the signal
  - A note is a frequency, we can generate these notes by using a **counter.** Count clock cycles, then toggle the PWM duty cycle at the right period