Falco Girgis
2/13/17
CPE526

Project Proposal - FPGA-based Dreamcast Visual Memory Unit

**1.0 Introduction**

The “Visual Memory Unit” or VMU is a small video gaming device created by Sega that launched alongside the “Dreamcast” video game console in 1999, serving three purposes:

1) Offering a file storage mechanism for save files and other data when plugged into the Dreamcast controller.
2) Serving as an additional display screen when plugged into the Dreamcast controller.
3) Functioning as a minimalistic gaming device when used standalone.


Close-up look at the VMU while run in standalone mode. It’s powered by 2 CR2032 batteries.


VMU (left) sitting next to a Dreamcast controller (right). The VMU slides into the top of the Dreamcast controller (upside-down), where its connectors make contact with connectors within the controller, enabling the actual Dreamcast console to communicate with the device through a custom communication protocol transmitted over the controller's cord.


The VMU being used as a secondary display screen and for saving userdata files when plugged into the Dreamcast controller. The Dreamcast is supplying power and the clock when used in this configuration.

Two standalone VMUs may be connected together to transfer files or to enable multiplayer support by using the two 8-bit synchronous serial modules within each device.

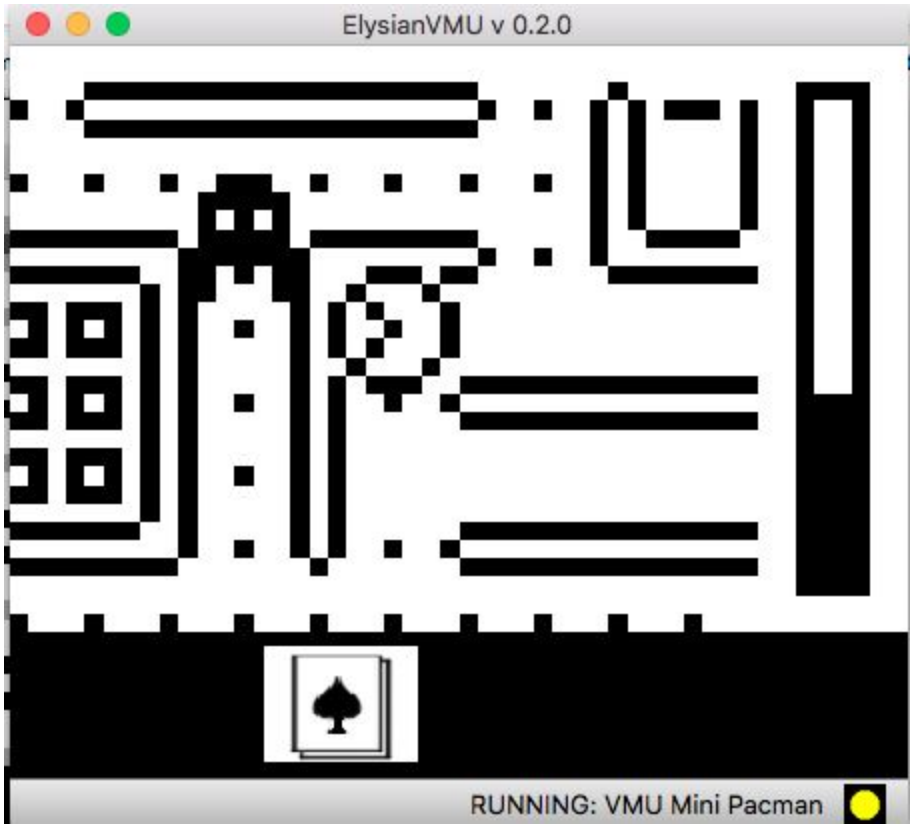
The VMU's hardware specifications are as follows:

- CPU: Sanyo LC86K87 (8-bit CPU, energy saving)[4]
- Memory: 128 KB (200 blocks)
- Power source: Two CR2032 batteries with auto-off function[4]
- Display: 48 dot Width x 32 dot Height resolution,[4] Monochrome
- Display size: 37 mm (1.46 inches) Width x 26 mm (1.02 inches) Height
- Case dimensions: 47 mm (1.85 inches) Width x 80 mm (3.15 inches) Height x 16 *mm (0.63 inches) Depth
- Sound: PWM sound chip,[4] 1-channel PWM sound source
- Weight: 45g (0.099 pounds)

**2.0 Purpose**

My Kickstarted video game project, Elysian Shadows, targets the Dreamcast as one of its supported platforms, and even though it's an older platform, it still has a massive indie following and is still very profitable to develop software for. One of the hallmarks of a "great" Dreamcast game is VMU-exclusive content. This content is usually using the VMU screen to display additional content to the player when it is connected to the controller and offering a separate mini game for when the device is used standalone.

Since ES is targeting many other non-Dreamcast platformers, we cannot justify spending development time on Dreamcast-specific features to people who purchased the game on other platforms… But if we don't have these Dreamcast-specific features, people who bought the game for the Dreamcast will feel cheated. Because of this dilemma, I have opted to create "ElysianVMU," a software-based VMU emulator for PC, smartphones, and any device the Elysian Shadows game targets, so that non-Dreamcast players can experience Dreamcast-specific features on non-Dreamcast platforms.



Screenshot of our emulator, ElysianVMU, running a Pac-Man game written for the VMU.

The concept has been met fairly widely with enthusiasm, so we're now wondering if we can take it a step further and eventually offer our own "VMU 2.0" device that can communicate with the Dreamcast or work independently of it. There are some flaws with the original VMU device that are widely criticized such as limited storage space, low resolution, and poor battery life that could be improved upon with a new design and could theoretically reach a much larger market than just people who want to play our game, Elysian Shadows.

**3.0 Project and Scope**

I plan to work alone as a single-person team to recreate the Visual Memory Unit using an FPGA. While my overall goal is to support everything, for the purpose of the project, my scope will be specifically focusing on implementing the CPU, RAM, External/Internal registers, interrupt controller, Flash memory, ROM, and 8-bit FAT filesystem. The goal is to have enough functionality present to run actual assembly programs written for the hardware with the exception of I/O devices and external interrupts. In the event that this winds up being easier than planned, or I finish early, I will continue adding peripherals like the buzzer, timers, buttons, LCD display, serial ports, and battery monitor, as the goal is to ultimately have a fully playable, Dreamcast-compatible device.

**4.0 Testing**

Since only a subset of the final device will be implemented, loading complete games and software on it to test will not be feasible. Instead, I plan to use a few trivial assembly programs that I will write to exercise all of the address space and CPU opcodes before writing a single result to a specific, known memory address which can be checked for correctness. If an opcode, register, peripheral, or section of memory is not implemented properly, it should be reflected in the output of these test programs, which can also be tested for correctness on several of the available VMU software emulators online (including my own, ElysianVMU).