GUITAR HERO

Kiryl Beliauski (kb3338), Patrick Cronin (pjc2192), & Dan Ivanovich (dmi2115)

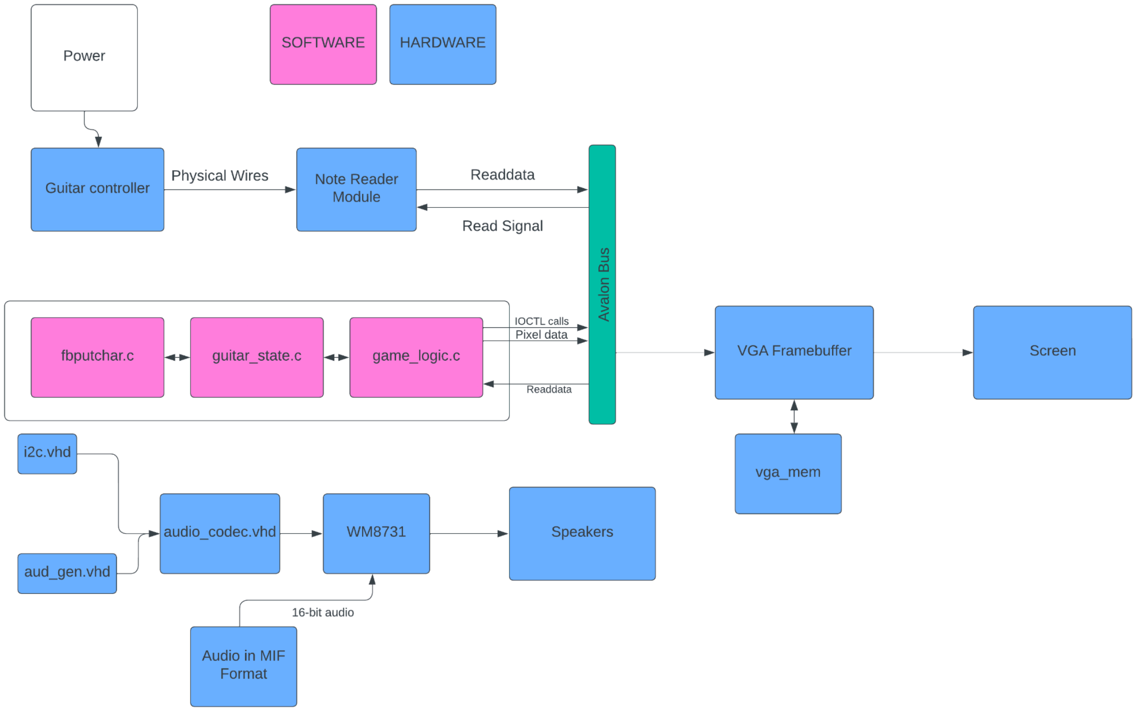
CSEE 4840 at Columbia University, Spring 2024

Table of Contents

Abstract

For our final project, we chose to recreate a level from Activision's Guitar Hero, using their original controller hardware with our FPGA. Our project had four main components: 1) the guitar controller 2) VGA graphics 3) game logic/software 4) audio. While nowhere near as polished as the original game, we were able to reproduce the core gameplay for a single level. Unfortunately, difficulties with campus access meant we were unable to get the audio component fully working in time, but the final results of the other components are robust and complete. We made hardware modifications to the controller, designed and implemented custom hardware modules in Verilog for processing the controller input and outputting to VGA, custom kernel modules to communicate with these modules, game logic, a custom sprite-rendering system, and a small set of development tools to automate tedious parts of the development process. We even implemented a complete emulation system that allowed us to develop graphics and gameplay without access to the lab — the full game is completely playable in emulation.

Design Overview

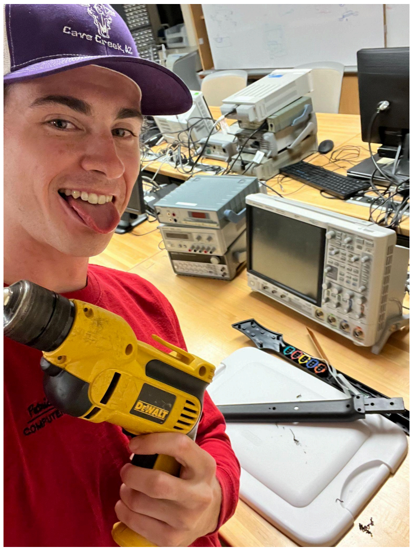


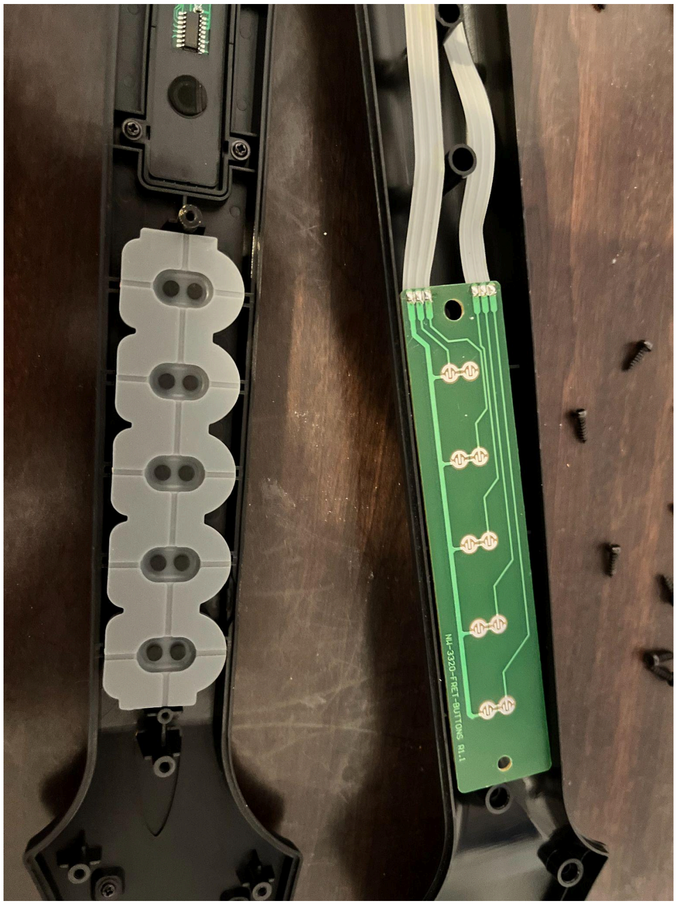
Hardware

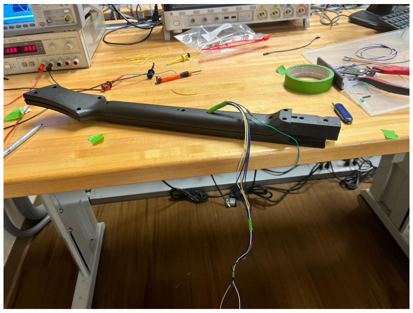
INPUT (Patrick)

1. **Guitar Adapter Modding**

In order to create the controller we salvaged a vintage wii guitar hero controller. I then opened the controller and saw that the circuit was a very simple set of switches. When one of the colored buttons on the controller is pressed, the compression pushes down a metal piece that closes the circuit. After testing the functionality in the EE lab, I soldered wires to each of the note pins, and soldered an additional wire for the input power.



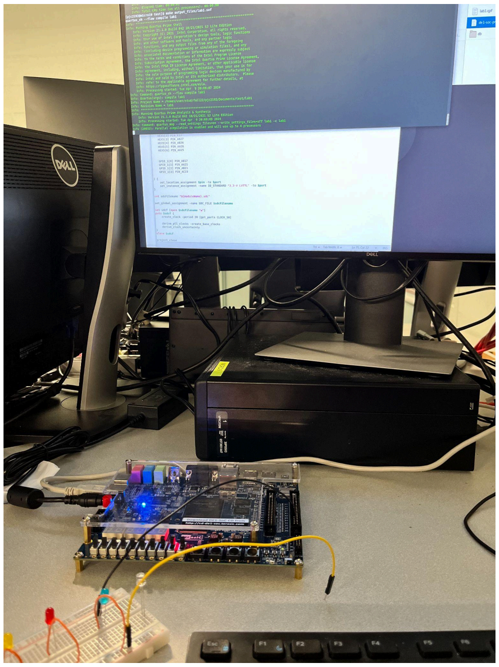




Then, I drilled a hole in the guitar and threaded long wires out to be used as a connecting cable to the FPGA.

2. **Note Reader Hardware Module**

In order to send the signals from our guitar to the linux operating system we needed a hardware driver to interface with the avalon bus. The Wires carrying the input from the guitar controller are connected to GPIO pins on the FPGA. So the note reading module must read these inputs and send them to the avalon bus to be stored in memory. To design the hardware file, I started by creating a simple Verilog file that connects GPIO pins to the LEDR outputs, pictured below. I was able to successfully test my guitar inputs by seeing the lights change when I pressed the buttons, I was able to guarantee that the controller was working as expected. I then referenced the Avalon® Interface Specifications, and learned how to handle read requests coming from the avalon bus. When a read and chip select signal are read as inputs, the hardware module assigns the value of each of the 6 inputs from the guitar to the first 6 bits of the `readata[7:0]` signal. This readdata is read in by the avalon bus and then stored in a register file to be accessed by the software.



GRAPHICS (Dan)

1. **VGA Framebuffer**

I wanted as much of the graphics as possible to be handled by C, not Verilog, to avoid the long compilation times of lab 3. Thus, I decided to re-create the functionality of the DE1-SOC's built-in `/dev/fb0` device. In order to significantly reduce our usage of embedded memory, I made two major optimizations over `/dev/fb0`. First, only the 150-pixel-wide strip down the middle of the screen can be controlled by the framebuffer. All of the gameplay of the level itself occurs in 5 columns of notes, each of which is a 24 px by 24 px sprite. This meant 150 pixels of width was enough to draw 5 24-pixel-wide columns of notes, with 3 pixels of horizontal margin per column. Second, the R, G, and B values of each pixel are not directly stored in the framebuffer. Instead, each pixel is represented by a 6-bit number and explicitly mapped to an RGB value hard-coded in a switch statement used as a LUT. This supports a color palette of up to 64 colors, which ended up being more than enough for our project.

The VGA framebuffer takes in 32-bit `writedata` from the Avalon bus. As our memory module (see the VGA Memory section below) required a 17-bit address and the Avalon bus wouldn't support an address of that width or a 6-bit `writedata`, I decided to combine the two into writedata. Thus, the VGA framebuffer ignores the address field and takes in 32-bit `writedata` of this format: `{9 unused bits, 17-bit pixel number, 6-bit pixel data}`. In every clock cycle, if the framebuffer is written to, the 6-bit pixel data value at `writedata[5:0]` is written into the memory module at address `writedata[22:6]`. In an always_comb block, if the current pixel falls within the 150-px wide strip, the 6-bit value stored at that pixel's address in memory, `pixel_data`, is read from memory and an RGB output value is assigned based on the switch statement LUT.

2. **VGA Memory**

The VGA memory module, `vga_mem`, is simply adapted from the Synchronous Memory module discussed in lecture. It takes 17-bit read/write addresses and takes/outputs 6-bit data. The memory itself is represented by `logic [5:0] data[122775:0]`. The rationale for 6-bit data was previously described — it gives us a 64-color palette. The addresses are 17 bits, as I just went with the simplest way of addressing a specific pixel: address = `{pixel_y, pixel_x}`, where `pixel_y` is a 9-bit value (our largest Y value, 479, is representable in 9 bits) and `pixel_x` is an

8-bit value (our largest X value, 149, is representable in 8 bits). This gives us a largest-possible address of {479 = 111011111, 149 = 10010101} = 11101111110010101 = 122773, which will fit in our `data[122775:0]`.

Our total memory usage:

$$122776\, chunks \ \times\ 6\, \frac{bits}{chunks} = 92082\, bits \ \approx\ 92\, kB$$

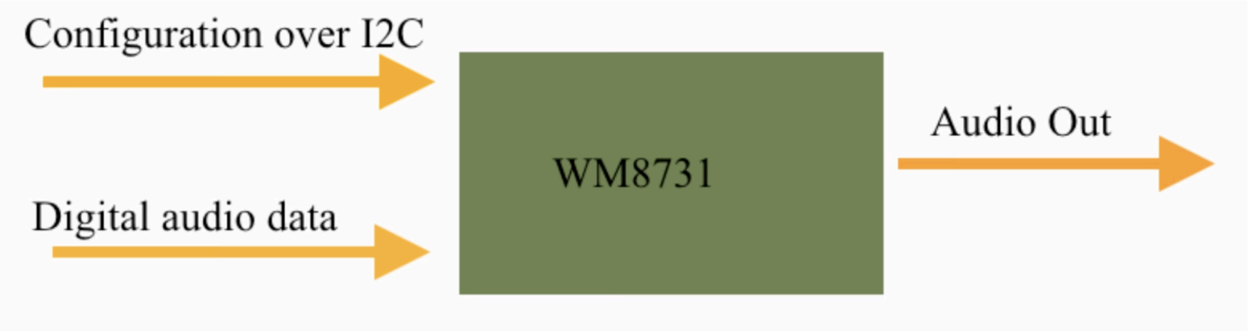
The original framebuffer's memory usage:

$$(480\ \times\ 640)\, pixels\ \times 3 \frac{Bytes}{pixel} = 921600\, Bytes \ \approx\ 921\, kB$$

This is a huge (10x) optimization over `/dev/fb0`

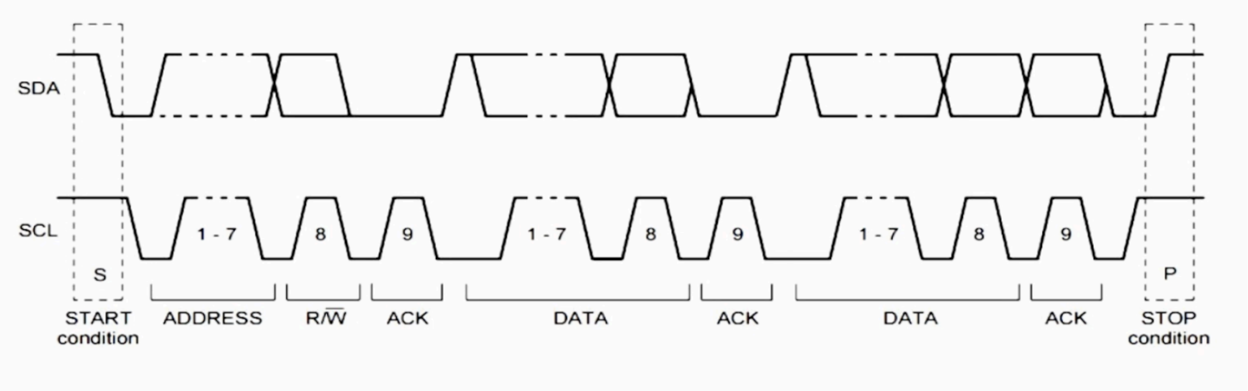
AUDIO (Kiryl)

1. **CODEC**



CODEC Data Flow

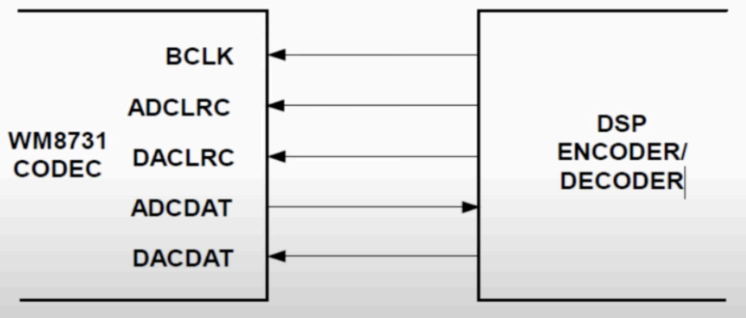
The CODEC of the board (WM8731) is configured using the I2C protocol. We have several VHDL files, specifically i2c.vhd, aud_gen.vhd, and audio_codec.vhd, which control several Altera IP blocks like PLL and the On-Chip Memory blocks. All of these blocks are then wrapped using a wrapper Audio Codec block that was made by us to ease the use of it with other modules of the whole project (like Patrick's Note Reader Module).



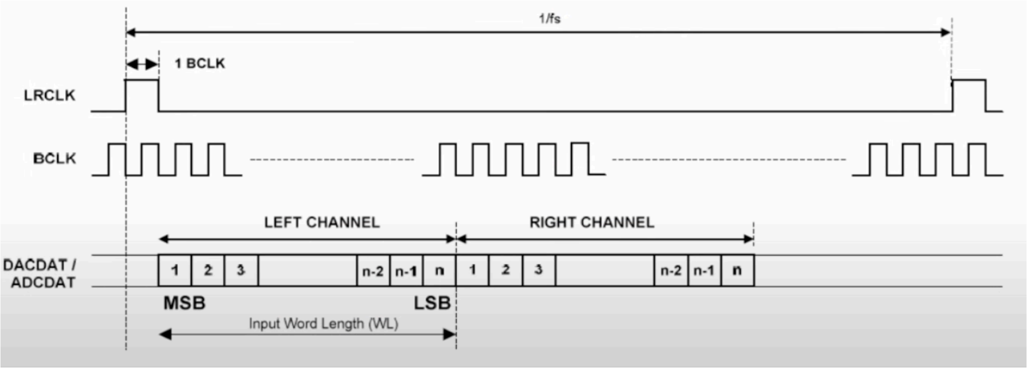
I2C Protocol

The CODEC is configured in the following 3 ways:

1. Slave mode. This allows our CODEC to receive all the necessary signals like main clock, interface clock, DAC synchronization, and actual audio data from the FPGA.
2. USB mode. Here we set it up to use the frequency of 12MHz to generate a sample rate of 48 ks/s.
3. PCM-mode audio interface. It is configured in the aud_gen.vhd file.



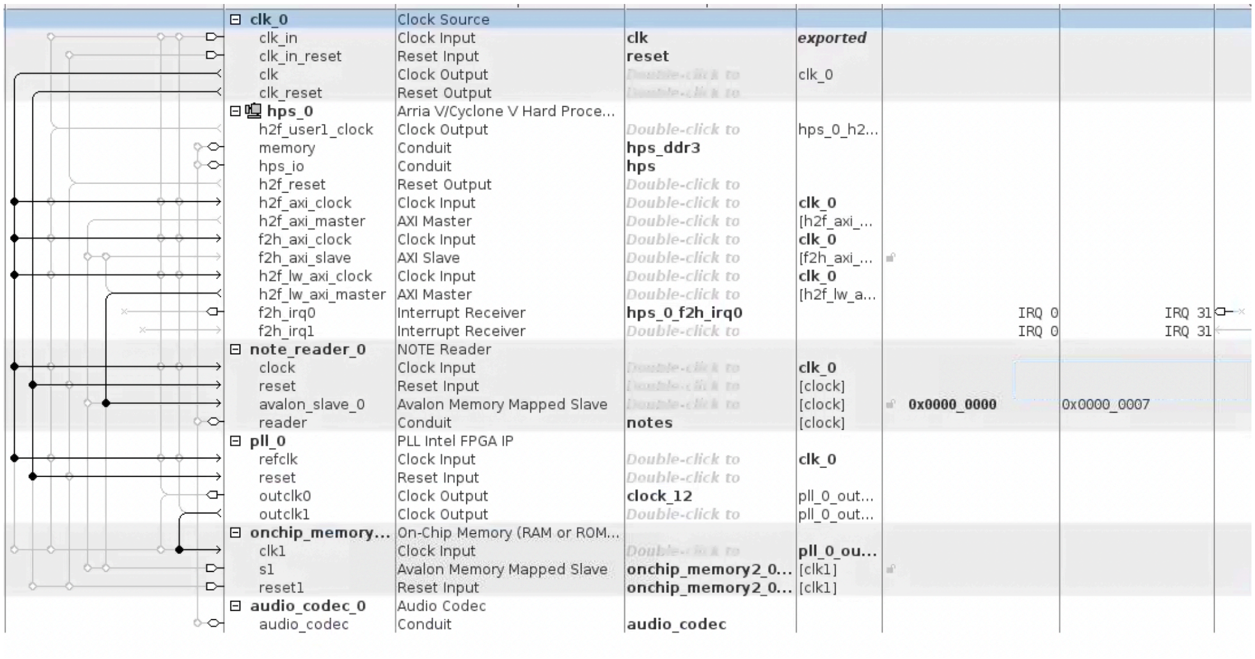
BLCK, DACLRC, DACDAT are provided by FPGA in Slave Mode



PCM

2. **Audio File** (Kiryl)

The audio file is stored on the on-chip memory, so it has to be preprocessed to be stored there efficiently. We use the Altera IP block for the on-chip memory and the required file format is MIF. To get the file from MP3 to MIF we first converted the file into a 16-bit mono PCM WAV file. After that, we have to strip everything from the file except the actual data, since the PCM data file has all the additional information at the beginning that we don’t need (bitrate, number of channels, file size, etc.). After that we convert the resulting data file into MIF format and its path is added to the Altera IP block.



⊟ **clk_0**	Clock Source				
clk_in	Clock Input	**clk**	***exported***		
clk_in_reset	Reset Input	**reset**			
clk	Clock Output	*Double-click to*	clk_0		
clk_reset	Reset Output	*Double-click to*			
⊟ **hps_0**	Arria V/Cyclone V Hard Proce...				
h2f_user1_clock	Clock Output	*Double-click to*	hps_0_h2...		
memory	Conduit	**hps_ddr3**			
hps_io	Conduit	**hps**			
h2f_reset	Reset Output	*Double-click to*			
h2f_axi_clock	Clock Input	*Double-click to*	**clk_0**		
h2f_axi_master	AXI Master	*Double-click to*	[h2f_axi_...		
f2h_axi_clock	Clock Input	*Double-click to*	**clk_0**		
f2h_axi_slave	AXI Slave	*Double-click to*	[f2h_axi_...		
h2f_lw_axi_clock	Clock Input	*Double-click to*	**clk_0**		
h2f_lw_axi_master	AXI Master	*Double-click to*	[h2f_lw_a...		
f2h_irq0	Interrupt Receiver	**hps_0_f2h_irq0**		IRQ 0	IRQ 31
f2h_irq1	Interrupt Receiver	*Double-click to*		IRQ 0	IRQ 31
⊟ **note_reader_0**	NOTE Reader				
clock	Clock Input	*Double-click to*	**clk_0**		
reset	Reset Input	*Double-click to*	[clock]		
avalon_slave_0	Avalon Memory Mapped Slave	*Double-click to*	[clock]	**0x0000_0000**	0x0000_0007
reader	Conduit	**notes**	[clock]		
⊟ **pll_0**	PLL Intel FPGA IP				
refclk	Clock Input	*Double-click to*	**clk_0**		
reset	Reset Input	*Double-click to*			
outclk0	Clock Output	**clock_12**	pll_0_out...		
outclk1	Clock Output	*Double-click to*	pll_0_out...		
⊟ **onchip_memory...**	On-Chip Memory (RAM or ROM...				
clk1	Clock Input	*Double-click to*	**pll_0_ou...**		
s1	Avalon Memory Mapped Slave	**onchip_memory2_0...**	[clk1]		
reset1	Reset Input	**onchip_memory2_0...**	[clk1]		
⊟ **audio_codec_0**	Audio Codec				
audio_codec	Conduit	**audio_codec**			

The Audio Modules

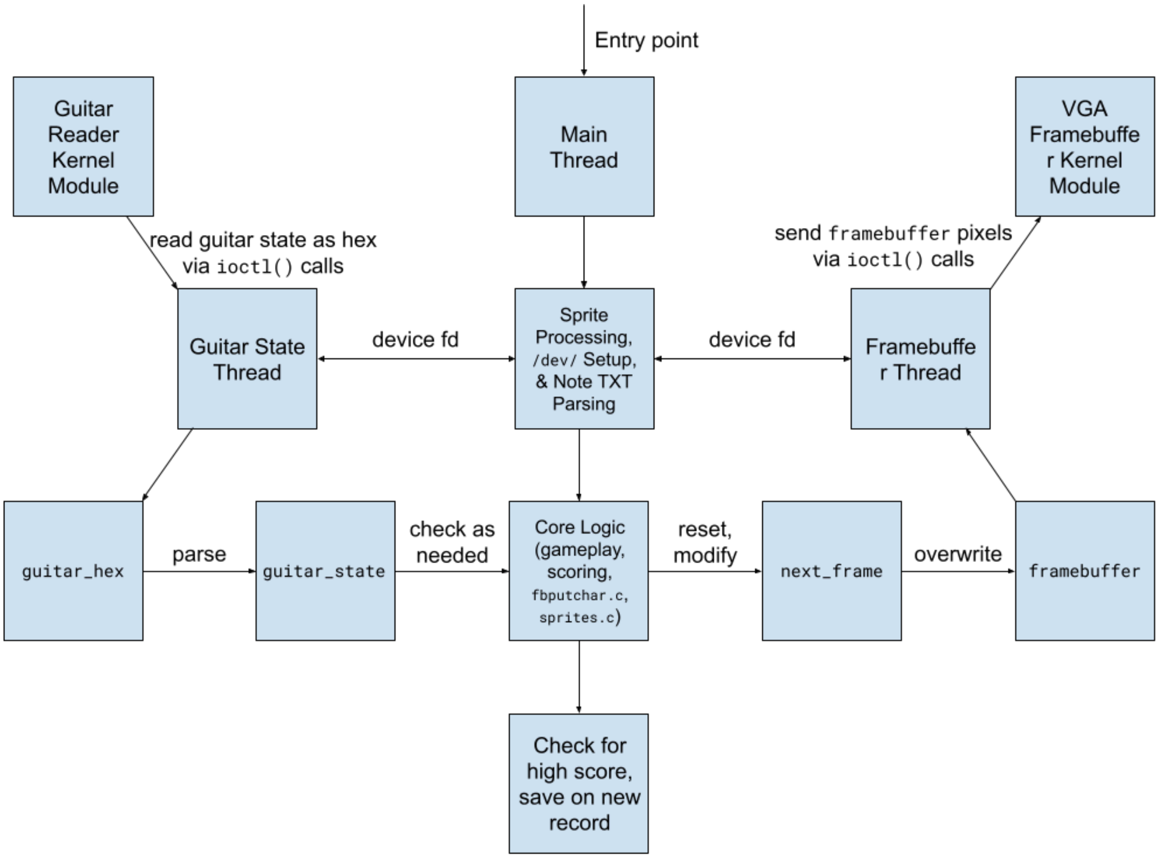
Software Overview

File listing

Essential software files: colors.c/h, `fbputchar.c/h`, `game_logic.c`, `global_consts.h`, `guitar_reader.c/h`, `guitar_state.c/h`, `helpers.c/h`, `song_data.h`, `sprites.c/h`, `vga_framebuffer.c/h`

Other files: `sprites/*.png`, `Makefile`, `high_score.txt`, `single_note_commaless.txt` `vga_emulator.c/h`, `generate_verilog_colors.py`

Overview of Software Structure:


Entry point
Guitar Reader Kernel Module
Main Thread
VGA Framebuffer Kernel Module
read guitar state as hex via ioctl() calls
send framebuffer pixels via ioctl() calls
Guitar State Thread
device fd
Sprite Processing, /dev/ Setup, & Note TXT Parsing
device fd
Framebuffer Thread
guitar_hex
parse
guitar_state
check as needed
Core Logic (gameplay, scoring, fbputchar.c, sprites.c)
reset, modify
next_frame
overwrite
framebuffer
Check for high score, save on new record

game_logic.c

The most important file. This is the orchestrator that coordinates the input and output. First, the main thread processes the sprites, connects to our HW kernel modules, and parses the hard-coded notes for the level from a .txt file. The game then steps into a start screen menu, then the main gameplay loop, and finally an ending screen that checks for a high score.

guitar_state.c/h

Defines a struct that represents the state of the guitar, plus helper methods for processing the data from the Verilog module into this struct.

guitar_reader.c/h

An adapted version of the lab 3 VGA ball kernel module that communicates with our Verilog module for the guitar.

vga_framebuffer.c/h

An adapted version of the lab 3 VGA ball kernel module that communicates with our Verilog module for the VGA framebuffer.

sprites.c/h

Structs and helper methods to parse, store, and render a sprite from a PNG.

fbputchar.c/h

An adapted version of the lab 2 `fbputchar` code that works with our custom framebuffer and color palette.

colors.c/h

Defines structs and an enum for representing colors in our color palette.

generate_verilog_colors.py

A helper script that parses our color palette from colors.c and outputs the body of the Verilog switch statement used as a `pixel_data` LUT in `vga_framebuffer.sv`.

vga_emulator.c/h

Uses `libsdl2-dev` to emulate the VGA. Also uses the keyboard to emulate the guitar.

global_consts.h

Defines global constants about the VGA screen output.

song_data.h

Defines constants about the song, such as BPM and notes per measure. Also defines a struct to represent a note of rows on screen.

helpers.c/h

Defines miscellaneous helper methods.

high_score.txt

Stores the highest score (accuracy) a player has achieved on this level.

single_note_commaless.txt

A cleaned export of the CSV for our song, containing information about the notes in every row. `game_logic.c` automatically parses this file and plays those notes in the level.

Makefile

Defines two important targets: `make all`, which compiles all the files necessary to run `game_logic`, and `make modules`, which compiles all the kernel modules.

Custom Development Tools

Throughout this project, we also developed some custom tools to expedite or automate parts of the development process. For example, the `generate_verilog_colors.py` tool made it very easy to keep our color palettes synchronized across C and Verilog. When campus shut down, we also took the time to develop an emulation system in C using `libsdl2-dev`. Dan programmed a system that used the `game_logic`'s framebuffer in a SDL2 window to render the graphics, which allowed us to continue to develop sprite rendering without our FPGA or monitor. The system also uses the keyboard in place of the guitar, using the keys 1-5 in place of each button on the guitar and the spacebar in place of the strumbar. We were therefore able to develop the graphics and general game logic without physical lab access, and the system was designed robustly enough that, even after all the in-person development we did once campus re-opened, the emulator still worked and the final game can be played entirely in emulation. Just flip `int EMULATING_VGA` to `1` in the start of `game_logic.c` to enable emulation.

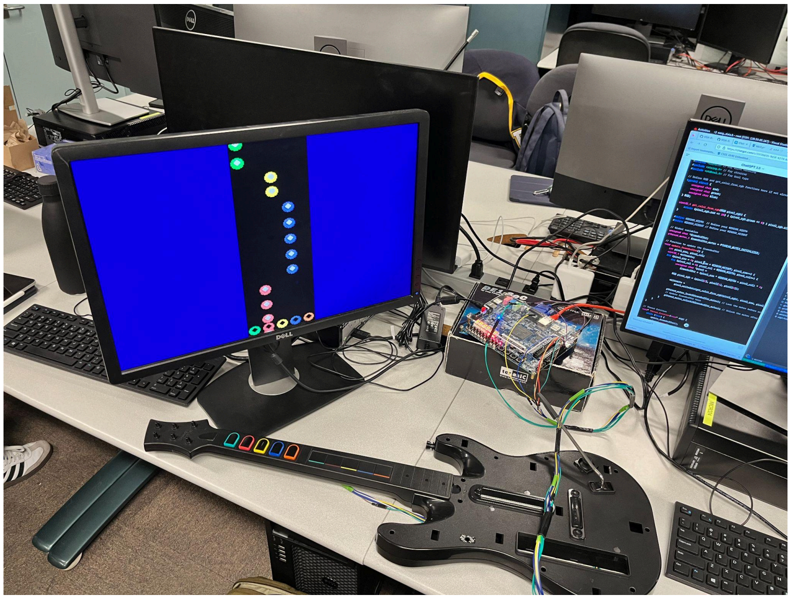
Results and Performance

The final results of the game logic, VGA framebuffer, and guitar reader are robust and complete. We even implemented a complete emulation system that allowed us to develop graphics and gameplay without access to the lab — the full game is completely playable in emulation. The scoring algorithm is fairly accurate, but will sometimes fall victim to imperfect strum bar debouncing.

Ultimately, the VGA graphics could use some more optimizations to be smoother. Notably, we could redesign the framebuffer to take in the pixel data for chunks of 5 pixels at a time, and use the address to define the address of the first of these 5 pixels. This way, we could update the contents of the `vga_mem` module 5 times faster, which might help with smoothness.

The guitar itself could also use some fixing. This guitar is almost 15 years old and underwent several years of use before making its way to this project, and the strumbar was already partially broken (strumming down hardly works), which does notably reduce the rate at which you can comfortably play a rapid series of notes.

The audio part of the project was not successful. The issue was mainly in the fact that we could not set up the wrapper module to work with the other parts of the project (it worked in isolation).



Lessons Learned

Quartus

One of the most important skills we developed over the course of this project is our ability to use Quartus to develop a complete hardware project. Beginning the project, we had a slight introduction to Quartus in lab 3 but had no ability to handle it with agency to create a novel application. What was a learning experience was the need to read documentation slowly and thoroughly to understand how to use the software. After reading Intel user guides that TAs showed us, Quartus became much more manageable and an extremely useful tool.

Memory Efficiency

When writing software on our laptops, we don't have to concern ourselves with memory efficiency anywhere near the extent to which we needed to in our Verilog modules, where every single bit counted. If we were to do this project again, we'd definitely focus even more on memory efficiency. For example, the VGA framebuffer's current addressing leaves about ⅓ of its memory unused — a more complex addressing system could probably be designed that would reduce this inefficiency. As mentioned in the previous section, we could also have redesigned how `writedata` and `address` are used to have pushed data to the framebuffer 5x faster.

Planning Ahead

Another lesson learned was surrounding planning. Our original block diagram that we submitted for our design proposal lacked a lot of knowledge and research for future steps, especially in communicating with the guitar (which we had not yet obtained). We assumed that each hardware (visuals, controller, and audio) would be relatively simple, but struggled significantly with each. If We were to do a project like this again, we would have spent much more time thinking about what we need to learn before embarking out to build. We chose to build on the fly, and this was fine because the project was of small size, but with a bigger project, the lack of planning definitely would have shot us in the foot.

Resources:

http://www.cs.columbia.edu/~sedwards/classes/2008/4840/Wolfson-WM8731-audio-CODEC.pdf
http://www-ug.eecg.toronto.edu/msl/manuals/tutorial_DE1-SoC.v5.1.pdf
https://www.youtube.com/watch?v=zzIi7ErWhAA&t=362s
https://www.cs.columbia.edu/~sedwards/classes/2024/4840-spring/memory.pdf

File Listing

Folder hardware

11 printable files

hardware/Makefile
hardware/aud_gen.vhd
hardware/audio_codec.vhd
hardware/audio_codec_hw.tcl
hardware/i2c.vhd
hardware/note_reader.sv
hardware/note_reader_hw.tcl
hardware/soc_system.tcl
hardware/soc_system_top.sv
hardware/vga_framebuffer.sv
hardware/vga_framebuffer_hw.tcl

hardware/Makefile

```
SYSTEM = soc_system

TCL = $(SYSTEM).tcl
QSYS = $(SYSTEM).qsys
SOPCINFO = $(SYSTEM).sopcinfo
QIP = $(SYSTEM)/synthesis/$(SYSTEM).qip
HPS_PIN_TCL = $(SYSTEM)/synthesis/submodules/hps_sdram_p0_pin_assignments.tcl
HPS_PIN_MAP = hps_sdram_p0_all_pins.txt
QPF = $(SYSTEM).qpf
QSF = $(SYSTEM).qsf
SDC = $(SYSTEM).sdc

BOARD_INFO = $(SYSTEM)_board_info.xml
DTS = $(SYSTEM).dts
DTB = $(SYSTEM).dtb

SOF = output_files/$(SYSTEM).sof
RBF = output_files/$(SYSTEM).rbf

SOFTWARE_DIR = software

BSP_DIR = $(SOFTWARE_DIR)/spl_bsp
BSP_SETTINGS = $(BSP_DIR)/settings.bsp
PRELOADER_SETTINGS_DIR = hps_isw_handoff/soc_system_hps_0
PRELOADER_MAKEFILE = $(BSP_DIR)/Makefile
PRELOADER_MKPIMAGE = $(BSP_DIR)/preloader-mkpimage.bin

UBOOT_IMAGE = $(BSP_DIR)/uboot-socfpga/u-boot.img

KERNEL_REPO = https://github.com/altera-opensource/linux-socfpga.git
KERNEL_BRANCH = socfpga-4.19
DEFAULT_CONFIG = socfpga_defconfig
CROSS = env CROSS_COMPILE=arm-altera-eabi- ARCH=arm

KERNEL_DIR = $(SOFTWARE_DIR)/linux-socfpga
KERNEL_CONFIG = $(KERNEL_DIR)/.config
ZIMAGE = $(KERNEL_DIR)/arch/arm/boot/zImage
```

```

TARFILES = Makefile \
    $(TCL) \
    $(QSYS) \
    $(SYSTEM)_top.sv \
    $(BOARD_INFO) \
    ip/intr_capturer/intr_capturer.v \
    ip/intr_capturer/intr_capturer_hw.tcl \
    vga_ball.sv

TARFILE = lab3-hw.tar.gz

# project
#
# Run the topmost tcl script to generate the initial project files

.PHONY : project
project : $(QPF) $(QSF) $(SDC)

$(QPF) $(QSF) $(SDC) : $(TCL)
    quartus_sh -t $(TCL)

# qsys
#
# From the .qsys file, generate the .sopcinfo, .qip, and directory
# (named according to the system) with all the Verilog files, etc.

.PHONY : qsys
qsys : $(SOPCINFO)

$(SOPCINFO) $(QIP) $(HPS_PIN_TCL) $(SYSTEM)/ $(PRELOADER_SETTINGS_DIR) : $(QSYS)
    rm -rf $(SOPCINFO) $(SYSTEM)/
    qsys-generate $(QSYS) --synthesis=VERILOG

# quartus
#
# Run Quartus on the Qsys-generated files
#
#    Build netlist
# quartus_map soc_system
#
#    Use netlist information to determine HPS stuff
# quartus_sta -t hps_sdram_p0_pin_assignments.tcl soc_system
#
#    Do the rest
#   FIXME: this is wasteful.  Really want not to repeat the "map" step
# quartus_sh --flow compile
#
# quartus_fit
# quartus_asm
# quartus_sta

.PHONY : quartus
quartus : $(SOF)

$(SOF) $(HPS_PIN_MAP) : $(QIP) $(QPF) $(QSF) $(HPS_PIN_TCL)
```

```
    quartus_map $(SYSTEM)
    quartus_sta -t $(HPS_PIN_TCL) $(SYSTEM)
    quartus_fit $(SYSTEM)
    quartus_asm $(SYSTEM)
#   quartus_sh --flow compile $(QPF)

# rbf
#
# Convert the .sof file (for programming through the USB blaster)
# to an .rbf file to be placed on an SD card and written by u-boot
.PHONY : rbf
rbf : $(RBF)

$(RBF) : $(SOF)
    quartus_cpf -c $(SOF) $(RBF)

# dtb
#
# Use the .sopcinfo file to generate a device tree blob file
# with information about the memory map of the peripherals
.PHONY : dtb
dtb : $(DTB)

$(DTB) : $(DTS)
    @which dtc || (echo "dtc not found.  Did you run embedded_command_shell.sh?"; exit 1)
    dtc -I dts -O dtb -o $(DTB) $(DTS)

$(DTS) : $(SOPCINFO) $(BOARD_INFO)
    @which sopc2dts || (echo "sopc2dts not found.  Did you run embedded_command_shell.sh?"; exit 1)
    sopc2dts --input $(SOPCINFO) \
        --output $(DTS) \
        --type dts \
        --board $(BOARD_INFO) \
        --clocks

# preloader
#
# Builds the SPL's preloader-mkpimage.bin image file, which should
# be written to the "magic" 3rd parition on the SD card
# in software/spl_bsp
#
# Requires the embedded_command_shell.sh script to have run so the compiler,
# etc is available
#
.PHONY : preloader
preloader : $(PRELOADER_MKPIMAGE)

$(PRELOADER_MKPIMAGE) : $(PRELOADER_MAKEFILE) $(BSP_SETTINGS)
    $(MAKE) -C $(BSP_DIR)

$(BSP_SETTINGS) $(PRELOADER_MAKEFILE) : $(PRELOADER_SETTINGS_DIR)
    mkdir -p $(BSP_DIR)
    bsp-create-settings \
      --type spl \
      --bsp-dir $(BSP_DIR) \
      --settings $(BSP_SETTINGS) \
      --preloader-settings-dir $(PRELOADER_SETTINGS_DIR) \
      --set spl.boot.FAT_SUPPORT 1

# uboot
#
# Build the bootloader

.PHONY : uboot
uboot : $(UBOOT_IMAGE)

$(UBOOT_IMAGE) : $(PRELOADER_MAKEFILE) $(BSP_SETTINGS)
    $(MAKE) -C $(BSP_DIR) uboot

# kernel-download
#
# Clone the Linux kernel repository
#
# kernel-config
#
#   Set up the default kernel configuration
#
# kernel-menuconfig
#
#   (Optional) Access the kernel configuration menu to make further
#   adjustments about which modules are included
#
# zimage
#
#   Compile the kernel

.PHONY : download-kernel config-kernel zimage
kernel-download : $(KERNEL_DIR)
kernel-config : $(KERNEL_CONFIG)
kernel-menuconfig :
    $(CROSS) $(MAKE) -C $(KERNEL_DIR) menuconfig
zimage : $(ZIMAGE)

$(KERNEL_DIR) :
    mkdir -p $(KERNEL_DIR)
    git clone --branch $(KERNEL_BRANCH) $(KERNEL_REPO) $(KERNEL_DIR)

# Configure the kernel.  Start from a provided default,
#
# Turn off version checking (makes it easier to compile kernel
# modules and not have them complain about version)
#
# Turn on large file (+2TB) support, which the ext4 filesystem
# requires by default (it will not be able to mount the root
# filesystem read/write otherwise)
$(KERNEL_CONFIG) : $(KERNEL_DIR)
    $(CROSS) $(MAKE) -C $(KERNEL_DIR) $(DEFAULT_CONFIG)
    $(KERNEL_DIR)/scripts/config --file $(KERNEL_CONFIG) \
      --disable CONFIG_LOCALVERSION_AUTO \
      --enable CONFIG_LBDAF \
      --disable CONFIG_XFS_FS \
      --disable CONFIG_GFS2_FS \
      --disable CONFIG_TEST_KMOD

# Compile the kernel

$(ZIMAGE) : $(KERNEL_CONFIG)
    $(CROSS) $(MAKE) -C $(KERNEL_DIR) LOCALVERSION= zImage

# tar
#
# Build soc_system.tar.gz

.phony : tar
tar : $(TARFILE)

$(TARFILE) : $(TARFILES)
    tar zcfC $(TARFILE) .. $(TARFILES:%=lab3-hw/%)

# clean
#
# Remove all generated files

.PHONY : clean quartus-clean qsys-clean project-clean
clean : quartus-clean qsys-clean project-clean dtb-clean preloader-clean \
    uboot-clean

project-clean :
    rm -rf $(QPF) $(QSF) $(SDC)

qsys-clean :
    rm -rf $(SOPCINFO) $(QIP) $(SYSTEM)/ .qsys_edit \
    hps_isw_handoff/ hps_sdram_p0_summary.csv

quartus-clean :
    rm -rf  $(SOF) output_files db incremental_db $(SYSTEM).qdf \
    c5_pin_model_dump.txt $(HPS_PIN_MAP)

dtb-clean :
    rm -rf $(DTS) $(DTB)

preloader-clean :
    rm -rf $(BSP_DIR)

uboot-clean :
    rm -rf $(BSP_DIR)/uboot-socfpga

kernel-clean :
    rm -rf $(KERNEL_DIR)

config-clean :
    rm -rf $(KERNEL_CONFIG)

```

hardware/aud_gen.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity  aud_gen is
port (
    aud_clock_12: in std_logic;
    aud_bk: out std_logic;
    aud_dalr: out std_logic;
    aud_dadat: out std_logic;
    aud_data_in: in std_logic_vector(31 downto 0)

);
end aud_gen;

architecture main of aud_gen is

signal sample_flag: std_logic:='0';
signal data_index: integer range 0 to 31:=0;
signal da_data :std_logic_vector(15 downto 0):=(others=>'0');
signal da_data_out: std_logic_vector(31 downto 0):=(others=>'0');
signal aud_prscl: integer range 0 to 300:=0;
signal clk_en: std_logic:='0';
begin

aud_bk<=aud_clock_12;

process(aud_clock_12)
begin

if falling_edge(aud_clock_12) then

    aud_dalr<=clk_en;

    if(aud_prscl<250)then------48k sample rate
        aud_prscl<=aud_prscl+1;
        clk_en<='0';
        else
        aud_prscl<=0;
        da_data_out<=aud_data_in;--get sample
        clk_en<='1';
    end if;



    if(clk_en='1')then-------send new sample
    sample_flag<='1';
    data_index<=31;
    end if;

    if(sample_flag='1')then

        if(data_index>0)then
            aud_dadat<=da_data_out(data_index);
            data_index<=data_index-1;
```

```vhdl
            else
            aud_dadat<=da_data_out(data_index);
            sample_flag<='0';
        end if;

    end if;
end if;

end process;
end main;
```

hardware/audio_codec.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity  audio_codec is
port (


----------WM8731 pins-----
AUD_BCLK: out std_logic;
AUD_XCK: out std_logic;
AUD_ADCLRCK:  out std_logic;
AUD_ADCDAT: in std_logic;
AUD_DACLRCK: out std_logic;
AUD_DACDAT: out std_logic;

---------FPGA pins-----

clock_50: in std_logic;
key: in std_logic_vector(3 downto 0);
ledr: out std_logic_vector(9 downto 0);
sw: in std_logic_vector(9 downto 0);
FPGA_I2C_SCLK: out std_logic;
FPGA_I2C_SDAT: inout std_logic

);

end audio_codec;


architecture main of audio_codec is

signal bitprsc: integer range 0 to 4:=0;
signal aud_mono: std_logic_vector(31 downto 0):=(others=>'0');
signal read_addr: integer range 0 to 240254:=0;
signal ROM_ADDR: std_logic_vector(17 downto 0);
signal ROM_OUT: std_logic_vector(15 downto 0);
signal clock_12pll: std_logic;
signal WM_i2c_busy: std_logic;
signal WM_i2c_done: std_logic;
signal WM_i2c_send_flag: std_logic;
signal WM_i2c_data: std_logic_vector(15 downto 0);
signal DA_CLR: std_logic:='0';

    component pll is
        port (
            clk_clk                         : in  std_logic                     := '
X';             -- clk
            clock_12_clk                    : out std_logic;
-- clk
            reset_reset_n                   : in  std_logic                     := '
X';             -- reset_n
            onchip_memory2_0_s1_address     : in  std_logic_vector(17 downto 0) :=
(others => 'X'); -- address
            onchip_memory2_0_s1_debugaccess : in  std_logic                     := '
X';             -- debugaccess
            onchip_memory2_0_s1_clken       : in  std_logic                     := '
X';             -- clken
            onchip_memory2_0_s1_chipselect  : in  std_logic                     := '
X';             -- chipselect
            onchip_memory2_0_s1_write       : in  std_logic                     := '
X';             -- write
            onchip_memory2_0_s1_readdata    : out std_logic_vector(15 downto 0);
-- readdata
            onchip_memory2_0_s1_writedata   : in  std_logic_vector(15 downto 0) :=
(others => 'X'); -- writedata
            onchip_memory2_0_s1_byteenable  : in  std_logic_vector(1 downto 0)  :=
(others => 'X'); -- byteenable
              onchip_memory2_0_reset1_reset   : in  std_logic                     :=
'X'              -- reset
        );
          end component pll;

 component aud_gen is
 port (
    aud_clock_12: in std_logic;
    aud_bk: out std_logic;
    aud_dalr: out std_logic;
    aud_dadat: out std_logic;
    aud_data_in: in std_logic_vector(31 downto 0)
 );
 end component aud_gen;




 component i2c is
 port(
    i2c_busy: out std_logic;
    i2c_scl: out std_logic;
    i2c_send_flag: in std_logic;
    i2c_sda: inout std_logic;
    i2c_addr: in std_logic_vector(7 downto 0);
    i2c_done: out std_logic;
    i2c_data: in std_logic_vector(15 downto 0);
    i2c_clock_50: in std_logic
 );

 end component i2c;
```

```


begin


 u0 : component pll
        port map (
            clk_clk       => clock_50,                                                     -- clk.clk
            reset_reset_n => '1',                                                          -- reset.reset_n
            clock_12_clk  => clock_12pll,                                                  -- clock_12.clk
                onchip_memory2_0_s1_address     => ROM_ADDR,
            onchip_memory2_0_s1_debugaccess =>'0',                                         -- debugaccess
            onchip_memory2_0_s1_clken       =>'1',                                         -- clken
            onchip_memory2_0_s1_chipselect  =>'1',                                         -- chipselect
            onchip_memory2_0_s1_write      =>'0',                                          -- write
            onchip_memory2_0_s1_readdata   =>ROM_OUT,                                      -- readdata
            onchip_memory2_0_s1_writedata  =>(others=>'0'),
            onchip_memory2_0_s1_byteenable  =>"11",
                onchip_memory2_0_reset1_reset=>'0'
                );




sound: component aud_gen
        port map(
        aud_clock_12=>clock_12pll,
        aud_bk=>AUD_BCLK,
        aud_dalr=>DA_CLR,
        aud_dadat=>AUD_DACDAT,
        aud_data_in=>aud_mono

        );

WM8731: component i2c
        port map(
            i2c_busy=>WM_i2c_busy,
            i2c_scl=>FPGA_I2C_SCLK,
            i2c_send_flag=>WM_i2c_send_flag,
            i2c_sda=>FPGA_I2C_SDAT,
            i2c_addr=>"00110100",
            i2c_done=>WM_i2c_done,
            i2c_data=>WM_i2c_data,
            i2c_clock_50=>clock_50
        );


AUD_XCK<=clock_12pll;
AUD_DACLRCK<=DA_CLR;

```

```
ROM_ADDR<=std_logic_vector(to_unsigned(read_addr,18));

process (clock_12pll)
begin

if rising_edge(clock_12pll)then

    if(SW(8)='1')then--------reset
    read_addr<=0;
    bitprsc<=0;
    aud_mono<=(others=>'0');
    else
    LEDR(1)<=SW(7);
    aud_mono(15 downto 0)<=ROM_OUT;----mono sound
    aud_mono(31 downto 16)<=ROM_OUT;
      if(DA_CLR='1')then
            if(bitprsc<5)then----8ksps
            bitprsc<=bitprsc+1;
            else
            bitprsc<=0;
                if(read_addr<240254)then
                read_addr<=read_addr+1;
                else
                read_addr<=0;
                end if;
            end if;
        end if;
    end if;



end if;

end process;

process (clock_50)
begin

    if rising_edge (clock_50)then
        if(KEY="1111")then
        WM_i2c_send_flag<='0';
        end if;
    end if;
 if rising_edge(clock_50) and WM_i2c_busy='0' then


        if (KEY(0)='0') then ----Digital Interface: DSP, 16 bit, slave mode
        WM_i2c_data(15 downto 9)<="0000111";
        WM_i2c_data(8 downto 0)<="000010011";
        WM_i2c_send_flag<='1';

        elsif (KEY(0)='0'AND SW(0)='1' ) then---HEADPHONE VOLUME
        WM_i2c_data(15 downto 9)<="0000010";
        WM_i2c_data(8 downto 0)<="101111001";
        WM_i2c_send_flag<='1';

        elsif (KEY(1)='0'AND SW(0)='0' ) then---ADC of, DAC on, Linout ON, Power ON
        WM_i2c_data(15 downto 9)<="0000110";
        WM_i2c_data(8 downto 0)<="000000111";

        WM_i2c_send_flag<='1';
        elsif (KEY(1)='0'AND SW(0)='1' ) then---USB mode
        WM_i2c_data(15 downto 9)<="0001000";
        WM_i2c_data(8 downto 0)<="000000001";

        WM_i2c_send_flag<='1';
        elsif (KEY(2)='0'AND SW(0)='0') then---activ interface
        WM_i2c_data(15 downto 9)<="0001001";
        WM_i2c_data(8 downto 0)<="111111111";

        WM_i2c_send_flag<='1';
        elsif (KEY(2)='0'AND SW(0)='1') then---Enable DAC to LINOUT
        WM_i2c_data(15 downto 9)<="0000100";
        WM_i2c_data(8 downto 0)<="000010010";

        WM_i2c_send_flag<='1';
        elsif (KEY(3)='0' AND SW(0)='0') then---remove mute DAC
        WM_i2c_data(15 downto 9)<="0000101";
        WM_i2c_data(8 downto 0)<="000000000";

        WM_i2c_send_flag<='1';
        elsif (KEY(3)='0' AND SW(0)='1') then---reset
        WM_i2c_data(15 downto 9)<="0001111";
        WM_i2c_data(8 downto 0)<="000000000";

        WM_i2c_send_flag<='1';
        end if;




 end if;
end process;
end main;
```

hardware/audio_codec_hw.tcl

```
# TCL File Generated by Component Editor 21.1
# Fri May 10 13:43:51 EDT 2024
# DO NOT MODIFY


#
# audio_codec "Audio Codec" v1.0
#  2024.05.10.13:43:51
#
#

#
# request TCL package from ACDS 16.1
#
```

```
package require -exact qsys 16.1


#
# module audio_codec
#
set_module_property DESCRIPTION ""
set_module_property NAME audio_codec
set_module_property VERSION 1.0
set_module_property INTERNAL false
set_module_property OPAQUE_ADDRESS_MAP true
set_module_property AUTHOR ""
set_module_property DISPLAY_NAME "Audio Codec"
set_module_property INSTANTIATE_IN_SYSTEM_MODULE true
set_module_property EDITABLE true
set_module_property REPORT_TO_TALKBACK false
set_module_property ALLOW_GREYBOX_GENERATION false
set_module_property REPORT_HIERARCHY false

set_module_assignment embeddedsw.dts.vendor "csee4840"
set_module_assignment embeddedsw.dts.name "audio_codec"
set_module_assignment embeddedsw.dts.group "audio"


#
# file sets
#
add_fileset QUARTUS_SYNTH QUARTUS_SYNTH "" ""
set_fileset_property QUARTUS_SYNTH TOP_LEVEL audio_codec
set_fileset_property QUARTUS_SYNTH ENABLE_RELATIVE_INCLUDE_PATHS false
set_fileset_property QUARTUS_SYNTH ENABLE_FILE_OVERWRITE_MODE false
add_fileset_file aud_gen.vhd VHDL PATH aud_gen.vhd
add_fileset_file audio_codec.vhd VHDL PATH audio_codec.vhd TOP_LEVEL_FILE
add_fileset_file i2c.vhd VHDL PATH i2c.vhd


#
# parameters
#


#
# display items
#


#
# connection point audio_codec
#
add_interface audio_codec conduit end
set_interface_property audio_codec associatedClock ""
set_interface_property audio_codec associatedReset ""
set_interface_property audio_codec ENABLED true
set_interface_property audio_codec EXPORT_OF ""
set_interface_property audio_codec PORT_NAME_MAP ""
set_interface_property audio_codec CMSIS_SVD_VARIABLES ""
set_interface_property audio_codec SVD_ADDRESS_GROUP ""

add_interface_port audio_codec AUD_ADCDAT adcdat Input 1
add_interface_port audio_codec AUD_ADCLRCK adcclrck Output 1
add_interface_port audio_codec AUD_BCLK bclk Output 1
add_interface_port audio_codec AUD_DACDAT dacdat Output 1
add_interface_port audio_codec AUD_DACLRCK daclrck Output 1
add_interface_port audio_codec AUD_XCK xck Output 1


```

hardware/i2c.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity i2c is
port(
i2c_busy: out std_logic;
i2c_scl: out std_logic;
i2c_send_flag: in std_logic;
i2c_sda: inout std_logic;
i2c_addr: in std_logic_vector(7 downto 0);
i2c_done: out std_logic;
i2c_data: in std_logic_vector(15 downto 0);
i2c_clock_50: in std_logic
);
end i2c;


architecture main of i2c is

signal i2c_clk_en: std_logic:='0';
signal clk_prs: integer range 0 to 300:=0;
signal clk_en: std_logic:='0';
signal ack_en: std_logic:='0';
signal clk_i2c: std_logic:='0';
signal get_ack: std_logic:='0';
signal data_index: integer range 0 to 15:=0;
type fsm is (st0,st1,st2,st3,st4,st5,st6,st7,st8);
signal i2c_fsm:fsm:=st0;
begin


------generate two clocks for i2c and data transitions
process(i2c_clock_50)
begin

if rising_edge(i2c_clock_50) then

    if(clk_prs<250)then
    clk_prs<=clk_prs+1;
    else
    clk_prs<=0;
    end if;

    if(clk_prs<125)then ---50 % duty cylce clock for i2c
    clk_i2c<='1';
    else
    clk_i2c<='0';
    end if;

    ---- clock for ack  on SCL=HIGH
    if(clk_prs=62)then
    ack_en<='1';
    else
    ack_en<='0';
    end if;


    ---- clock for data on SCL=LOW
    if(clk_prs=187)then
    clk_en<='1';
    else
    clk_en<='0';
    end if;

end if;

if rising_edge(i2c_clock_50) then


    if(i2c_clk_en='1')then
        i2c_scl<=clk_i2c;
    else
        i2c_scl<='1';
    end if;


    ----ack on SCL=HIGH
    if(ack_en='1')then
        case i2c_fsm is
        when st3=> ---- get ack

                if(i2c_sda='0')then
                    i2c_fsm<=st4;---ack
                    data_index<=15;
                    else
                    i2c_clk_en<='0';
                    i2c_fsm<=st0;---nack
                end if;

        when st5=> --- get ack

                if(i2c_sda='0')then
                    i2c_fsm<=st6;---ack
                    data_index<=7;

                    else
                    i2c_fsm<=st0;---nack
                    i2c_clk_en<='0';
                end if;

        when st7 => ----get ack

                if(i2c_sda='0')then
                    i2c_fsm<=st8;---ack
                    else
                    i2c_fsm<=st0;---nack
                    i2c_clk_en<='0';
                end if;

       when others=>NULL;
        end case;
    end if;



    -----data tranfer on SCL=LOW
    if(clk_en='1')then
        case i2c_fsm is
            when st0=> ----------stand by
                i2c_sda<='1';
                i2c_busy<='0';
                i2c_done<='0';
                if(i2c_send_flag='1')then
                i2c_fsm<=st1;
                i2c_busy<='1';
                end if;

            when st1=> -------start condition
                i2c_sda<='0';
                i2c_fsm<=st2;
                data_index<=7;
            when st2=> -------send addr
               i2c_clk_en<='1';---start clocking i2c_scl

                if(data_index>0) then
                    data_index<=data_index-1;
                    i2c_sda<=i2c_addr(data_index);
                    else
                    i2c_sda<=i2c_addr(data_index);
                    get_ack<='1';
                end if;

                if(get_ack='1')then
                    get_ack<='0';
                    i2c_fsm<=st3;
                    i2c_sda<='Z';
                end if;

            when st4=> ---- send 1st 8 bit

                if(data_index>8) then
                    data_index<=data_index-1;
                    i2c_sda<=i2c_data(data_index);
                    else
                    i2c_sda<=i2c_data(data_index);
                    get_ack<='1';
                end if;

                if(get_ack='1')then
                    get_ack<='0';
                    i2c_fsm<=st5;
                    i2c_sda<='Z';
                end if;

            when st6 => ---send 2nd 8 bit

                if(data_index>0) then
                    data_index<=data_index-1;
                    i2c_sda<=i2c_data(data_index);
                    else
                    i2c_sda<=i2c_data(data_index);
                    get_ack<='1';
                end if;

                if(get_ack='1')then
                    get_ack<='0';
                    i2c_fsm<=st7;
                    i2c_sda<='Z';
                end if;

            when st8 => --stop condition
                i2c_clk_en<='0';
                i2c_sda<='0';
                i2c_fsm<=st0;
                i2c_done<='1';
            when others=>NULL;
            end case;
    end if;

end if;
end process;
end main;
```

hardware/note_reader.sv

```
module note_reader (
    input logic clk,
    input logic reset,
    input chipselect,
    input logic [2:0] address,
    input logic [3:0] KEY,
    input logic [5:0] GPIO_1,

    output [7:0] LEDR,
    output logic [7:0] readdata,
    input logic read,
    output logic waitrequest
);

  assign LEDR = GPIO_1;  // Assign GPIO_1 directly to LEDR output
  //assign LEDR[6] = KEY[0]

  // Combinational logic to assign readdata based on KEY inputs
  always_comb begin
    if (read && chipselect) begin
      readdata = 8'b00000000;  // Initialize readdata to all zeros
      // Loop through each key
      for (int i = 0; i < 6; i = i + 1) begin
        // If the corresponding key is pressed, set the corresponding bit in readdata
        if (GPIO_1[i]) begin
          readdata = readdata | (1 << i);
        end
      end
      if (KEY[0]) begin
        readdata = readdata | (1 << 6);
      end
    end else begin
      readdata = 8'b00000000;  // Clear readdata if read or chipselect is low
    end
  end

endmodule
```

hardware/note_reader_hw.tcl

```
# TCL File Generated by Component Editor 21.1
# Mon May 06 22:08:31 EDT 2024
# DO NOT MODIFY


#
# note_reader "NOTE Reader" v1.0
#  2024.05.06.22:08:31
#
#

#
# request TCL package from ACDS 16.1
#
package require -exact qsys 16.1


#
# module note_reader
#
set_module_property DESCRIPTION ""
set_module_property NAME note_reader
set_module_property VERSION 1.0
set_module_property INTERNAL false
set_module_property OPAQUE_ADDRESS_MAP true
set_module_property AUTHOR ""
set_module_property DISPLAY_NAME "NOTE Reader"
set_module_property INSTANTIATE_IN_SYSTEM_MODULE true
set_module_property EDITABLE true
set_module_property REPORT_TO_TALKBACK false
set_module_property ALLOW_GREYBOX_GENERATION false
set_module_property REPORT_HIERARCHY false

set_module_assignment embeddedsw.dts.vendor "csee4840"
set_module_assignment embeddedsw.dts.name "note_reader"
set_module_assignment embeddedsw.dts.group "guitar"

#
# file sets
#
add_fileset QUARTUS_SYNTH QUARTUS_SYNTH "" ""
set_fileset_property QUARTUS_SYNTH TOP_LEVEL note_reader
set_fileset_property QUARTUS_SYNTH ENABLE_RELATIVE_INCLUDE_PATHS false
set_fileset_property QUARTUS_SYNTH ENABLE_FILE_OVERWRITE_MODE false
add_fileset_file note_reader.sv SYSTEM_VERILOG PATH note_reader.sv TOP_LEVEL_FILE


#
# parameters
#


#
# module assignments
#
set_module_assignment embeddedsw.dts.group notes
set_module_assignment embeddedsw.dts.name note_reader
set_module_assignment embeddedsw.dts.vendor csee4840


#
# display items
#


#
# connection point clock
#
add_interface clock clock end
set_interface_property clock clockRate 0
set_interface_property clock ENABLED true
set_interface_property clock EXPORT_OF ""
set_interface_property clock PORT_NAME_MAP ""
set_interface_property clock CMSIS_SVD_VARIABLES ""
set_interface_property clock SVD_ADDRESS_GROUP ""

add_interface_port clock clk clk Input 1


#
# connection point reset
```

```
#
add_interface reset reset end
set_interface_property reset associatedClock clock
set_interface_property reset synchronousEdges DEASSERT
set_interface_property reset ENABLED true
set_interface_property reset EXPORT_OF ""
set_interface_property reset PORT_NAME_MAP ""
set_interface_property reset CMSIS_SVD_VARIABLES ""
set_interface_property reset SVD_ADDRESS_GROUP ""

add_interface_port reset reset reset Input 1


#
# connection point avalon_slave_0
#
add_interface avalon_slave_0 avalon end
set_interface_property avalon_slave_0 addressUnits WORDS
set_interface_property avalon_slave_0 associatedClock clock
set_interface_property avalon_slave_0 associatedReset reset
set_interface_property avalon_slave_0 bitsPerSymbol 8
set_interface_property avalon_slave_0 burstOnBurstBoundariesOnly false
set_interface_property avalon_slave_0 burstcountUnits WORDS
set_interface_property avalon_slave_0 explicitAddressSpan 0
set_interface_property avalon_slave_0 holdTime 0
set_interface_property avalon_slave_0 linewrapBursts false
set_interface_property avalon_slave_0 maximumPendingReadTransactions 0
set_interface_property avalon_slave_0 maximumPendingWriteTransactions 0
set_interface_property avalon_slave_0 readLatency 0
set_interface_property avalon_slave_0 readWaitTime 1
set_interface_property avalon_slave_0 setupTime 0
set_interface_property avalon_slave_0 timingUnits Cycles
set_interface_property avalon_slave_0 writeWaitTime 0
set_interface_property avalon_slave_0 ENABLED true
set_interface_property avalon_slave_0 EXPORT_OF ""
set_interface_property avalon_slave_0 PORT_NAME_MAP ""
set_interface_property avalon_slave_0 CMSIS_SVD_VARIABLES ""
set_interface_property avalon_slave_0 SVD_ADDRESS_GROUP ""

add_interface_port avalon_slave_0 address address Input 3
add_interface_port avalon_slave_0 chipselect chipselect Input 1
add_interface_port avalon_slave_0 read read Input 1
add_interface_port avalon_slave_0 readdata readdata Output 8
add_interface_port avalon_slave_0 waitrequest waitrequest Output 1
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isFlash 0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isMemoryDevice 0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isNonVolatileStorage 0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isPrintableDevice 0


#
# connection point reader
#
add_interface reader conduit end
set_interface_property reader associatedClock clock
set_interface_property reader associatedReset ""
set_interface_property reader ENABLED true
set_interface_property reader EXPORT_OF ""
set_interface_property reader PORT_NAME_MAP ""
set_interface_property reader CMSIS_SVD_VARIABLES ""
set_interface_property reader SVD_ADDRESS_GROUP ""

add_interface_port reader KEY key_n Input 4
add_interface_port reader LEDR ledr_n Output 8
add_interface_port reader GPIO_1 pins_n Input 6


```

hardware/soc_system.tcl

```
# Generate Quartus project files for the DE1-SoC board
#
# Stephen A. Edwards, Columbia University

# Invoke as
#
# quartus_sh -t soc_system.tcl

set project "soc_system"

# Top-level SystemVerilog file should be <project>_top.sv, with Verilog module
# <project>_top in it

set systemVerilogSource "${project}_top.sv"
set qip "${project}/synthesis/${project}.qip"

project_new $project -overwrite

foreach {name value} {
    FAMILY "Cyclone V"
    DEVICE 5CSEMA5F31C6

    PROJECT_OUTPUT_DIRECTORY output_files

    CYCLONEII_RESERVE_NCEO_AFTER_CONFIGURATION "USE AS REGULAR IO"

    NUM_PARALLEL_PROCESSORS 4

} { set_global_assignment -name $name $value }

set_global_assignment -name TOP_LEVEL_ENTITY "${project}_top"

foreach filename $systemVerilogSource {
    set_global_assignment -name SYSTEMVERILOG_FILE $filename
}

foreach filename $qip {
    set_global_assignment -name QIP_FILE $filename
}

```

```
# FPGA pin assignments

foreach {pin port} {
    PIN_AJ4 ADC_CS_N
    PIN_AK4 ADC_DIN
    PIN_AK3 ADC_DOUT
    PIN_AK2 ADC_SCLK

    PIN_K7 AUD_ADCDAT
    PIN_K8 AUD_ADCLRCK
    PIN_H7 AUD_BCLK
    PIN_J7 AUD_DACDAT
    PIN_H8 AUD_DACLRCK
    PIN_G7 AUD_XCK

    PIN_AA16 CLOCK2_50
    PIN_Y26 CLOCK3_50
    PIN_K14 CLOCK4_50
    PIN_AF14 CLOCK_50

    PIN_AK14 DRAM_ADDR[0]
    PIN_AH14 DRAM_ADDR[1]
    PIN_AG15 DRAM_ADDR[2]
    PIN_AE14 DRAM_ADDR[3]
    PIN_AB15 DRAM_ADDR[4]
    PIN_AC14 DRAM_ADDR[5]
    PIN_AD14 DRAM_ADDR[6]
    PIN_AF15 DRAM_ADDR[7]
    PIN_AH15 DRAM_ADDR[8]
    PIN_AG13 DRAM_ADDR[9]
    PIN_AG12 DRAM_ADDR[10]
    PIN_AH13 DRAM_ADDR[11]
    PIN_AJ14 DRAM_ADDR[12]
    PIN_AF13 DRAM_BA[0]
    PIN_AJ12 DRAM_BA[1]
    PIN_AF11 DRAM_CAS_N
    PIN_AK13 DRAM_CKE
    PIN_AH12 DRAM_CLK
    PIN_AG11 DRAM_CS_N
    PIN_AK6 DRAM_DQ[0]
    PIN_AJ7 DRAM_DQ[1]
    PIN_AK7 DRAM_DQ[2]
    PIN_AK8 DRAM_DQ[3]
    PIN_AK9 DRAM_DQ[4]
    PIN_AG10 DRAM_DQ[5]
    PIN_AK11 DRAM_DQ[6]
    PIN_AJ11 DRAM_DQ[7]
    PIN_AH10 DRAM_DQ[8]
    PIN_AJ10 DRAM_DQ[9]
    PIN_AJ9 DRAM_DQ[10]
    PIN_AH9 DRAM_DQ[11]
    PIN_AH8 DRAM_DQ[12]
    PIN_AH7 DRAM_DQ[13]
    PIN_AJ6 DRAM_DQ[14]
    PIN_AJ5 DRAM_DQ[15]
    PIN_AB13 DRAM_LDQM
    PIN_AE13 DRAM_RAS_N
    PIN_AK12 DRAM_UDQM
    PIN_AA13 DRAM_WE_N

    PIN_AA12 FAN_CTRL

    PIN_J12 FPGA_I2C_SCLK
    PIN_K12 FPGA_I2C_SDAT

    PIN_AC18 GPIO_0[0]
    PIN_Y17  GPIO_0[1]
    PIN_AD17 GPIO_0[2]
    PIN_Y18  GPIO_0[3]
    PIN_AK16 GPIO_0[4]
    PIN_AK18 GPIO_0[5]
    PIN_AK19 GPIO_0[6]
    PIN_AJ19 GPIO_0[7]
    PIN_AJ17 GPIO_0[8]
    PIN_AJ16 GPIO_0[9]
    PIN_AH18 GPIO_0[10]
    PIN_AH17 GPIO_0[11]
    PIN_AG16 GPIO_0[12]
    PIN_AE16 GPIO_0[13]
    PIN_AF16 GPIO_0[14]
    PIN_AG17 GPIO_0[15]
    PIN_AA18 GPIO_0[16]
    PIN_AA19 GPIO_0[17]
    PIN_AE17 GPIO_0[18]
    PIN_AC20 GPIO_0[19]
    PIN_AH19 GPIO_0[20]
    PIN_AJ20 GPIO_0[21]
    PIN_AH20 GPIO_0[22]
    PIN_AK21 GPIO_0[23]
    PIN_AD19 GPIO_0[24]
    PIN_AD20 GPIO_0[25]
    PIN_AE18 GPIO_0[26]
    PIN_AE19 GPIO_0[27]
    PIN_AF20 GPIO_0[28]
    PIN_AF21 GPIO_0[29]
    PIN_AF19 GPIO_0[30]
    PIN_AG21 GPIO_0[31]
    PIN_AF18 GPIO_0[32]
    PIN_AG20 GPIO_0[33]
    PIN_AG18 GPIO_0[34]
    PIN_AJ21 GPIO_0[35]

    PIN_AB17 GPIO_1[0]
    PIN_AA21 GPIO_1[1]
    PIN_AB21 GPIO_1[2]
    PIN_AC23 GPIO_1[3]
    PIN_AD24 GPIO_1[4]
    PIN_AE23 GPIO_1[5]
    PIN_AE24 GPIO_1[6]
    PIN_AF25 GPIO_1[7]
    PIN_AF26 GPIO_1[8]
    PIN_AG25 GPIO_1[9]
    PIN_AG26 GPIO_1[10]
    PIN_AH24 GPIO_1[11]
    PIN_AH27 GPIO_1[12]
    PIN_AJ27 GPIO_1[13]
    PIN_AK29 GPIO_1[14]
    PIN_AK28 GPIO_1[15]
    PIN_AK27 GPIO_1[16]
    PIN_AJ26 GPIO_1[17]
    PIN_AK26 GPIO_1[18]
    PIN_AH25 GPIO_1[19]
    PIN_AJ25 GPIO_1[20]
    PIN_AJ24 GPIO_1[21]
    PIN_AK24 GPIO_1[22]
    PIN_AG23 GPIO_1[23]
    PIN_AK23 GPIO_1[24]
    PIN_AH23 GPIO_1[25]
    PIN_AK22 GPIO_1[26]
    PIN_AJ22 GPIO_1[27]
    PIN_AH22 GPIO_1[28]
    PIN_AG22 GPIO_1[29]
    PIN_AF24 GPIO_1[30]
    PIN_AF23 GPIO_1[31]
    PIN_AE22 GPIO_1[32]
    PIN_AD21 GPIO_1[33]
    PIN_AA20 GPIO_1[34]
    PIN_AC22 GPIO_1[35]

    PIN_AE26 HEX0[0]
    PIN_AE27 HEX0[1]
    PIN_AE28 HEX0[2]
    PIN_AG27 HEX0[3]
    PIN_AF28 HEX0[4]
    PIN_AG28 HEX0[5]
    PIN_AH28 HEX0[6]

    PIN_AJ29 HEX1[0]
    PIN_AH29 HEX1[1]
    PIN_AH30 HEX1[2]
    PIN_AG30 HEX1[3]
    PIN_AF29 HEX1[4]
    PIN_AF30 HEX1[5]
    PIN_AD27 HEX1[6]

    PIN_AB23 HEX2[0]
    PIN_AE29 HEX2[1]
    PIN_AD29 HEX2[2]
    PIN_AC28 HEX2[3]
    PIN_AD30 HEX2[4]
    PIN_AC29 HEX2[5]
    PIN_AC30 HEX2[6]

    PIN_AD26 HEX3[0]
    PIN_AC27 HEX3[1]
    PIN_AD25 HEX3[2]
    PIN_AC25 HEX3[3]
    PIN_AB28 HEX3[4]
    PIN_AB25 HEX3[5]
    PIN_AB22 HEX3[6]

    PIN_AA24 HEX4[0]
    PIN_Y23  HEX4[1]
    PIN_Y24  HEX4[2]
    PIN_W22  HEX4[3]
    PIN_W24  HEX4[4]
    PIN_V23  HEX4[5]
    PIN_W25  HEX4[6]

    PIN_V25  HEX5[0]
    PIN_AA28 HEX5[1]
    PIN_Y27  HEX5[2]
    PIN_AB27 HEX5[3]
    PIN_AB26 HEX5[4]
    PIN_AA26 HEX5[5]
    PIN_AA25 HEX5[6]

    PIN_AA30 IRDA_RXD
    PIN_AB30 IRDA_TXD

    PIN_AA14 KEY[0]
    PIN_AA15 KEY[1]
    PIN_W15  KEY[2]
    PIN_Y16  KEY[3]

    PIN_V16 LEDR[0]
    PIN_W16 LEDR[1]
    PIN_V17 LEDR[2]
    PIN_V18 LEDR[3]
    PIN_W17 LEDR[4]
    PIN_W19 LEDR[5]
    PIN_Y19 LEDR[6]
    PIN_W20 LEDR[7]
    PIN_W21 LEDR[8]
    PIN_Y21 LEDR[9]

    PIN_AD7 PS2_CLK
    PIN_AD9 PS2_CLK2
    PIN_AE7 PS2_DAT
    PIN_AE9 PS2_DAT2

    PIN_AB12 SW[0]
    PIN_AC12 SW[1]
    PIN_AF9  SW[2]
    PIN_AF10 SW[3]
    PIN_AD11 SW[4]
    PIN_AD12 SW[5]
    PIN_AE11 SW[6]
    PIN_AC9  SW[7]
    PIN_AD10 SW[8]
    PIN_AE12 SW[9]

    PIN_H15 TD_CLK27
    PIN_D2  TD_DATA[0]
    PIN_B1  TD_DATA[1]
    PIN_E2  TD_DATA[2]
    PIN_B2  TD_DATA[3]
    PIN_D1  TD_DATA[4]
    PIN_E1  TD_DATA[5]
    PIN_C2  TD_DATA[6]
    PIN_B3  TD_DATA[7]
    PIN_A5  TD_HS
    PIN_F6  TD_RESET_N
    PIN_A3  TD_VS

    PIN_A13 VGA_R[0]
    PIN_C13 VGA_R[1]
    PIN_E13 VGA_R[2]
    PIN_B12 VGA_R[3]
    PIN_C12 VGA_R[4]
    PIN_D12 VGA_R[5]
    PIN_E12 VGA_R[6]
    PIN_F13 VGA_R[7]

    PIN_J9  VGA_G[0]
    PIN_J10 VGA_G[1]
    PIN_H12 VGA_G[2]
    PIN_G10 VGA_G[3]
    PIN_G11 VGA_G[4]
    PIN_G12 VGA_G[5]
    PIN_F11 VGA_G[6]
    PIN_E11 VGA_G[7]

    PIN_B13 VGA_B[0]
    PIN_G13 VGA_B[1]
    PIN_H13 VGA_B[2]
    PIN_F14 VGA_B[3]
    PIN_H14 VGA_B[4]
    PIN_F15 VGA_B[5]
    PIN_G15 VGA_B[6]
    PIN_J14 VGA_B[7]

    PIN_A11 VGA_CLK
    PIN_B11 VGA_HS
    PIN_D11 VGA_VS
    PIN_F10 VGA_BLANK_N
    PIN_C10 VGA_SYNC_N
} {
    set_location_assignment $pin -to $port
    set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to $port
}

# HPS assignments

# 3.3-V LVTTL pins
foreach port {
    HPS_CONV_USB_N
    HPS_ENET_GTX_CLK
    HPS_ENET_INT_N
    HPS_ENET_MDC
    HPS_ENET_MDIO
    HPS_ENET_RX_CLK
    HPS_ENET_RX_DATA[0]
    HPS_ENET_RX_DATA[1]
    HPS_ENET_RX_DATA[2]
    HPS_ENET_RX_DATA[3]
    HPS_ENET_RX_DV
    HPS_ENET_TX_DATA[0]
    HPS_ENET_TX_DATA[1]
    HPS_ENET_TX_DATA[2]
    HPS_ENET_TX_DATA[3]
    HPS_ENET_TX_EN
    HPS_GSENSOR_INT
    HPS_I2C1_SCLK
    HPS_I2C1_SDAT
    HPS_I2C2_SCLK
    HPS_I2C2_SDAT
    HPS_I2C_CONTROL
    HPS_KEY
    HPS_LED
    HPS_LTC_GPIO
    HPS_SD_CLK
    HPS_SD_CMD
    HPS_SD_DATA[0]
    HPS_SD_DATA[1]
    HPS_SD_DATA[2]
    HPS_SD_DATA[3]
    HPS_SPIM_CLK
    HPS_SPIM_MISO
    HPS_SPIM_MOSI
    HPS_SPIM_SS
    HPS_UART_RX
    HPS_UART_TX
    HPS_USB_CLKOUT
    HPS_USB_DATA[0]
    HPS_USB_DATA[1]
    HPS_USB_DATA[2]
    HPS_USB_DATA[3]
    HPS_USB_DATA[4]
    HPS_USB_DATA[5]
    HPS_USB_DATA[6]
    HPS_USB_DATA[7]
    HPS_USB_DIR
    HPS_USB_NXT
    HPS_USB_STP
} {
    set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to $port
}

# There are a lot of settings for the HPS_DDR3 interface not listed here.
# Instead, the
#
# soc_system/synthesis/submodules/hps_sdram_p0_pin_assignments.tcl
#
# script generated by qsys adds that information.  However, quartus_map
# must be run before this .tcl script may run because the script
# relies on being able to look at the (HPS) netlist to determine which
# pins to constrain

set sdcFilename "${project}.sdc"

set_global_assignment -name SDC_FILE $sdcFilename

set sdcf [open $sdcFilename "w"]
puts $sdcf {
    foreach {clock port} {
    clock_50_1 CLOCK_50
    clock_50_2 CLOCK2_50
    clock_50_3 CLOCK3_50
    clock_50_4 CLOCK4_50
    } {
    create_clock -name $clock -period 20ns [get_ports $port]
    }

    create_clock -name clock_27_1 -period 37 [get_ports TD_CLK27]

    derive_pll_clocks -create_base_clocks
    derive_clock_uncertainty
}
close $sdcf

project_close

```

hardware/soc_system_top.sv

```
// ==================================================================
// Copyright (c) 2013 by Terasic Technologies Inc.
// ==================================================================
//
// Modified 2019 by Stephen A. Edwards
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera
//   Development Kits made by Terasic.  Other use of this code,
//   including the selling ,duplication, or modification of any
//   portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design
//   reference which illustrates how these types of functions can be
//   implemented.  It is the user's responsibility to verify their
//   design for consistency and functionality through the use of
//   formal verification methods.  Terasic provides no warranty
//   regarding the use or functionality of this code.
//
// ===============================================================
//
```

```
//  Terasic Technologies Inc

//  9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan
//
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
module soc_system_top(

 ///////// ADC /////////
 inout         ADC_CS_N,
 output        ADC_DIN,
 input         ADC_DOUT,
 output        ADC_SCLK,

 ///////// AUD /////////
 input         AUD_ADCDAT,
 inout         AUD_ADCLRCK,
 inout         AUD_BCLK,
 output        AUD_DACDAT,
 inout         AUD_DACLRCK,
 output        AUD_XCK,

 ///////// CLOCK2 /////////
 input         CLOCK2_50,

 ///////// CLOCK3 /////////
 input         CLOCK3_50,

 ///////// CLOCK4 /////////
 input         CLOCK4_50,

 ///////// CLOCK /////////
 input         CLOCK_50,

 ///////// DRAM /////////
 output [12:0] DRAM_ADDR,
 output [1:0]  DRAM_BA,
 output        DRAM_CAS_N,
 output        DRAM_CKE,
 output        DRAM_CLK,
 output        DRAM_CS_N,
 inout [15:0]  DRAM_DQ,
 output        DRAM_LDQM,
 output        DRAM_RAS_N,
 output        DRAM_UDQM,
 output        DRAM_WE_N,

 ///////// FAN /////////
 output        FAN_CTRL,

 ///////// FPGA /////////
 output        FPGA_I2C_SCLK,
 inout         FPGA_I2C_SDAT,

 ///////// GPIO /////////
```

```
inout [35:0]  GPIO_0,
inout [35:0]  GPIO_1,

///////// HEX0 /////////
output [6:0]  HEX0,

///////// HEX1 /////////
output [6:0]  HEX1,

///////// HEX2 /////////
output [6:0]  HEX2,

///////// HEX3 /////////
output [6:0]  HEX3,

///////// HEX4 /////////
output [6:0]  HEX4,

///////// HEX5 /////////
output [6:0]  HEX5,

///////// HPS /////////
inout         HPS_CONV_USB_N,
output [14:0] HPS_DDR3_ADDR,
output [2:0]  HPS_DDR3_BA,
output        HPS_DDR3_CAS_N,
output        HPS_DDR3_CKE,
output        HPS_DDR3_CK_N,
output        HPS_DDR3_CK_P,
output        HPS_DDR3_CS_N,
output [3:0]  HPS_DDR3_DM,
inout [31:0]  HPS_DDR3_DQ,
inout [3:0]   HPS_DDR3_DQS_N,
inout [3:0]   HPS_DDR3_DQS_P,
output        HPS_DDR3_ODT,
output        HPS_DDR3_RAS_N,
output        HPS_DDR3_RESET_N,
input         HPS_DDR3_RZQ,
output        HPS_DDR3_WE_N,
output        HPS_ENET_GTX_CLK,
inout         HPS_ENET_INT_N,
output        HPS_ENET_MDC,
inout         HPS_ENET_MDIO,
input         HPS_ENET_RX_CLK,
input [3:0]   HPS_ENET_RX_DATA,
input         HPS_ENET_RX_DV,
output [3:0]  HPS_ENET_TX_DATA,
output        HPS_ENET_TX_EN,
inout         HPS_GSENSOR_INT,
inout         HPS_I2C1_SCLK,
inout         HPS_I2C1_SDAT,
inout         HPS_I2C2_SCLK,
inout         HPS_I2C2_SDAT,
inout         HPS_I2C_CONTROL,
inout         HPS_KEY,
inout         HPS_LED,
 inout         HPS_LTC_GPIO,
 output        HPS_SD_CLK,
 inout         HPS_SD_CMD,
 inout [3:0]   HPS_SD_DATA,
 output        HPS_SPIM_CLK,
 input         HPS_SPIM_MISO,
 output        HPS_SPIM_MOSI,
 inout         HPS_SPIM_SS,
 input         HPS_UART_RX,
 output        HPS_UART_TX,
 input         HPS_USB_CLKOUT,
 inout [7:0]   HPS_USB_DATA,
 input         HPS_USB_DIR,
 input         HPS_USB_NXT,
 output        HPS_USB_STP,

 ///////// IRDA /////////
 input         IRDA_RXD,
 output        IRDA_TXD,

 ///////// KEY /////////
 input [3:0]   KEY,

 ///////// LEDR /////////
 output [9:0]  LEDR,

 ///////// PS2 /////////
 inout         PS2_CLK,
 inout         PS2_CLK2,
 inout         PS2_DAT,
 inout         PS2_DAT2,

 ///////// SW /////////
 input [9:0]   SW,

 ///////// TD /////////
 input         TD_CLK27,
 input [7:0]   TD_DATA,
 input         TD_HS,
 output        TD_RESET_N,
 input         TD_VS,


 ///////// VGA /////////
 output [7:0]  VGA_B,
 output        VGA_BLANK_N,
 output        VGA_CLK,
 output [7:0]  VGA_G,
 output        VGA_HS,
 output [7:0]  VGA_R,
 output        VGA_SYNC_N,
 output        VGA_VS
);

   soc_system soc_system0(
     .clk_clk                      ( CLOCK_50 ),
    .reset_reset_n                ( 1'b1 ),

    .hps_ddr3_mem_a               ( HPS_DDR3_ADDR ),
    .hps_ddr3_mem_ba              ( HPS_DDR3_BA ),
    .hps_ddr3_mem_ck              ( HPS_DDR3_CK_P ),
    .hps_ddr3_mem_ck_n            ( HPS_DDR3_CK_N ),
    .hps_ddr3_mem_cke             ( HPS_DDR3_CKE ),
    .hps_ddr3_mem_cs_n            ( HPS_DDR3_CS_N ),
    .hps_ddr3_mem_ras_n           ( HPS_DDR3_RAS_N ),
    .hps_ddr3_mem_cas_n           ( HPS_DDR3_CAS_N ),
    .hps_ddr3_mem_we_n            ( HPS_DDR3_WE_N ),
    .hps_ddr3_mem_reset_n         ( HPS_DDR3_RESET_N ),
    .hps_ddr3_mem_dq              ( HPS_DDR3_DQ ),
    .hps_ddr3_mem_dqs             ( HPS_DDR3_DQS_P ),
    .hps_ddr3_mem_dqs_n           ( HPS_DDR3_DQS_N ),
    .hps_ddr3_mem_odt             ( HPS_DDR3_ODT ),
    .hps_ddr3_mem_dm              ( HPS_DDR3_DM ),
    .hps_ddr3_oct_rzqin           ( HPS_DDR3_RZQ ),

    .hps_hps_io_emac1_inst_TX_CLK ( HPS_ENET_GTX_CLK ),
    .hps_hps_io_emac1_inst_TXD0   ( HPS_ENET_TX_DATA[0] ),
    .hps_hps_io_emac1_inst_TXD1   ( HPS_ENET_TX_DATA[1] ),
    .hps_hps_io_emac1_inst_TXD2   ( HPS_ENET_TX_DATA[2] ),
    .hps_hps_io_emac1_inst_TXD3   ( HPS_ENET_TX_DATA[3] ),
    .hps_hps_io_emac1_inst_RXD0   ( HPS_ENET_RX_DATA[0] ),
    .hps_hps_io_emac1_inst_MDIO   ( HPS_ENET_MDIO  ),
    .hps_hps_io_emac1_inst_MDC    ( HPS_ENET_MDC   ),
    .hps_hps_io_emac1_inst_RX_CTL ( HPS_ENET_RX_DV ),
    .hps_hps_io_emac1_inst_TX_CTL ( HPS_ENET_TX_EN ),
    .hps_hps_io_emac1_inst_RX_CLK ( HPS_ENET_RX_CLK ),
    .hps_hps_io_emac1_inst_RXD1   ( HPS_ENET_RX_DATA[1]  ),
    .hps_hps_io_emac1_inst_RXD2   ( HPS_ENET_RX_DATA[2]  ),
    .hps_hps_io_emac1_inst_RXD3   ( HPS_ENET_RX_DATA[3]  ),

    .hps_hps_io_sdio_inst_CMD     ( HPS_SD_CMD          ),
    .hps_hps_io_sdio_inst_D0      ( HPS_SD_DATA[0]      ),
    .hps_hps_io_sdio_inst_D1      ( HPS_SD_DATA[1]      ),
    .hps_hps_io_sdio_inst_CLK     ( HPS_SD_CLK          ),
    .hps_hps_io_sdio_inst_D2      ( HPS_SD_DATA[2]      ),
    .hps_hps_io_sdio_inst_D3      ( HPS_SD_DATA[3]      ),

    .hps_hps_io_usb1_inst_D0      ( HPS_USB_DATA[0]     ),
    .hps_hps_io_usb1_inst_D1      ( HPS_USB_DATA[1]     ),
    .hps_hps_io_usb1_inst_D2      ( HPS_USB_DATA[2]     ),
    .hps_hps_io_usb1_inst_D3      ( HPS_USB_DATA[3]     ),
    .hps_hps_io_usb1_inst_D4      ( HPS_USB_DATA[4]     ),
    .hps_hps_io_usb1_inst_D5      ( HPS_USB_DATA[5]     ),
    .hps_hps_io_usb1_inst_D6      ( HPS_USB_DATA[6]     ),
    .hps_hps_io_usb1_inst_D7      ( HPS_USB_DATA[7]     ),
    .hps_hps_io_usb1_inst_CLK     ( HPS_USB_CLKOUT      ),
    .hps_hps_io_usb1_inst_STP     ( HPS_USB_STP         ),
    .hps_hps_io_usb1_inst_DIR     ( HPS_USB_DIR         ),
    .hps_hps_io_usb1_inst_NXT     ( HPS_USB_NXT         ),

    .hps_hps_io_spim1_inst_CLK    ( HPS_SPIM_CLK  ),
    .hps_hps_io_spim1_inst_MOSI   ( HPS_SPIM_MOSI ),
      .hps_hps_io_spim1_inst_MISO   ( HPS_SPIM_MISO ),
      .hps_hps_io_spim1_inst_SS0    ( HPS_SPIM_SS   ),

      .hps_hps_io_uart0_inst_RX     ( HPS_UART_RX     ),
      .hps_hps_io_uart0_inst_TX     ( HPS_UART_TX     ),

      .hps_hps_io_i2c0_inst_SDA     ( HPS_I2C1_SDAT     ),
      .hps_hps_io_i2c0_inst_SCL     ( HPS_I2C1_SCLK     ),

      .hps_hps_io_i2c1_inst_SDA     ( HPS_I2C2_SDAT     ),
      .hps_hps_io_i2c1_inst_SCL     ( HPS_I2C2_SCLK     ),

      .hps_hps_io_gpio_inst_GPIO09  ( HPS_CONV_USB_N ),
      .hps_hps_io_gpio_inst_GPIO35  ( HPS_ENET_INT_N ),
      .hps_hps_io_gpio_inst_GPIO40  ( HPS_LTC_GPIO ),

      .hps_hps_io_gpio_inst_GPIO48  ( HPS_I2C_CONTROL ),
      .hps_hps_io_gpio_inst_GPIO53  ( HPS_LED ),
      .hps_hps_io_gpio_inst_GPIO54  ( HPS_KEY ),
      .hps_hps_io_gpio_inst_GPIO61  ( HPS_GSENSOR_INT ),

      .notes_ledr_n ( LEDR[7:0] ),
      .notes_key_n ( KEY ),
      .notes_pins_n ( GPIO_1[5:0] ),
      .vga_r (VGA_R),
      .vga_g (VGA_G),
      .vga_b (VGA_B),
      .vga_clk (VGA_CLK),
      .vga_hs (VGA_HS),
      .vga_vs (VGA_VS),
      .vga_blank_n (VGA_BLANK_N),
      .vga_sync_n (VGA_SYNC_N)
   );

    // The following quiet the "no driver" warnings for output
    // pins and should be removed if you use any of these peripherals

    assign ADC_CS_N = SW[1] ? SW[0] : 1'bZ;
    assign ADC_DIN = SW[0];
    assign ADC_SCLK = SW[0];

    assign AUD_ADCLRCK = SW[1] ? SW[0] : 1'bZ;
    assign AUD_BCLK = SW[1] ? SW[0] : 1'bZ;
    assign AUD_DACDAT = SW[0];
    assign AUD_DACLRCK = SW[1] ? SW[0] : 1'bZ;
    assign AUD_XCK = SW[0];

    assign DRAM_ADDR = { 13{ SW[0] } };
    assign DRAM_BA = { 2{ SW[0] } };
    assign DRAM_DQ = SW[1] ? { 16{ SW[0] } } : 16'bZ;
    assign {DRAM_CAS_N, DRAM_CKE, DRAM_CLK, DRAM_CS_N,
        DRAM_LDQM, DRAM_RAS_N, DRAM_UDQM, DRAM_WE_N} = { 8{SW[0]} };

    assign FAN_CTRL = SW[0];

    assign FPGA_I2C_SCLK = SW[0];
    assign FPGA_I2C_SDAT = SW[1] ? SW[0] : 1'bZ;

    assign GPIO_0 = SW[1] ? { 36{ SW[0] } } : 36'bZ;
    //assign GPIO_1 = SW[1] ? { 36{ SW[0] } } : 36'bZ;
    assign GPIO_1 [35:6] = SW[1] ? { 36{ SW[0] } } : 36'bZ;

    assign HEX0 = { 7{ SW[1] } };
    assign HEX1 = { 7{ SW[2] } };
    assign HEX2 = { 7{ SW[3] } };
    assign HEX3 = { 7{ SW[4] } };
    assign HEX4 = { 7{ SW[5] } };
    assign HEX5 = { 7{ SW[6] } };

    assign IRDA_TXD = SW[0];

    assign LEDR[9:8] = { 2{SW[7]} };

    assign PS2_CLK = SW[1] ? SW[0] : 1'bZ;
    assign PS2_CLK2 = SW[1] ? SW[0] : 1'bZ;
    assign PS2_DAT = SW[1] ? SW[0] : 1'bZ;
    assign PS2_DAT2 = SW[1] ? SW[0] : 1'bZ;

    assign TD_RESET_N = SW[0];



 endmodule

```

hardware/vga_framebuffer.sv

```
/*
 * Avalon memory-mapped peripheral that generates VGA
 *
 * Stephen A. Edwards
 * Columbia University
 */

module vga_framebuffer (
    input logic clk,
    input logic reset,
    input logic [31:0] writedata,  // Format: {9 unused bits, 17-bit pixel number, 6-bit pixel data}
    input logic write,
    input chipselect,
    input logic [1:0] address,  // Unused, because a 17-bit address is weird

    output logic [7:0] VGA_R,
    VGA_G,
    VGA_B,
    output logic       VGA_CLK,
    VGA_HS,
    VGA_VS,
    VGA_BLANK_n,
    output logic       VGA_SYNC_n
);

  logic [10:0] hcount;
  logic [ 9:0] vcount;
  logic [ 8:0] pixel_y;
  logic [ 9:0] pixel_x;
  logic [16:0] read_addr, write_addr;

  logic [7:0] background_r, background_g, background_b;
  logic [5:0] write_data, pixel_data;
  logic write_mem;

  assign pixel_y   = vcount[8:0];
  assign pixel_x   = hcount[10:1] - 10'd244;  // Offset by 1 b/c we need a clock cycle to read
  assign read_addr = {pixel_y, pixel_x[7:0]};

  vga_counters counters (
      .clk50(clk),
      .*
  );


  vga_mem mem (
      .clk(clk),
      .ra(read_addr),
      .wa(write_addr),
      .write(write_mem),
      .wd(write_data),
      .rd(pixel_data)
  );

  always_ff @(posedge clk)
    if (reset) begin
      background_r <= 8'h0;
      background_g <= 8'h0;
      background_b <= 8'h80;
      write_addr <= 17'h0;
      write_data <= 8'h0;
      write_mem <= 1'd0;
    end else begin
      write_mem <= 1'd0;
      if (chipselect && write) begin
        //write_addr <= 17'd5;
        write_addr <= writedata[22:6];  // Extracting 17-bit pixel number from writedata
        write_data <= writedata[5:0];  // Extracting 8-bit pixel data from writedata
        write_mem  <= 1'd1;
      end
    end


  always_comb begin
    {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
    if (VGA_BLANK_n) begin
      if (hcount[10:1] >= 10'd245 && hcount[10:1] < 10'd395 && vcount[9:0] < 10'd480) begin
```

```
        case (pixel_data)
          6'd0: {VGA_R, VGA_G, VGA_B} = 24'h000000;  // Black
          6'd1: {VGA_R, VGA_G, VGA_B} = 24'hffffff;  // White
          6'd2: {VGA_R, VGA_G, VGA_B} = 24'hff0000;  // Red
          6'd3: {VGA_R, VGA_G, VGA_B} = 24'h00ff00;  // Green
          6'd4: {VGA_R, VGA_G, VGA_B} = 24'h0000ff;  // Blue
          6'd5: {VGA_R, VGA_G, VGA_B} = 24'h14d345;  // Light_green
          6'd6: {VGA_R, VGA_G, VGA_B} = 24'h11a132;  // Middle_green
          6'd7: {VGA_R, VGA_G, VGA_B} = 24'h10a237;  // Dark_green
          6'd8: {VGA_R, VGA_G, VGA_B} = 24'hd3362f;  // Light_red
          6'd9: {VGA_R, VGA_G, VGA_B} = 24'h9a2a26;  // Middle_red
          6'd10: {VGA_R, VGA_G, VGA_B} = 24'h9b2929;  // Dark_red
          6'd11: {VGA_R, VGA_G, VGA_B} = 24'hfef335;  // Light_yellow
          6'd12: {VGA_R, VGA_G, VGA_B} = 24'hc5bd1a;  // Middle_yellow
          6'd13: {VGA_R, VGA_G, VGA_B} = 24'hcfbd3d;  // Dark_yellow
          6'd14: {VGA_R, VGA_G, VGA_B} = 24'h5375e0;  // Light_blue
          6'd15: {VGA_R, VGA_G, VGA_B} = 24'h3b59af;  // Middle_blue
          6'd16: {VGA_R, VGA_G, VGA_B} = 24'h4059ab;  // Dark_blue
          6'd17: {VGA_R, VGA_G, VGA_B} = 24'hda562b;  // Light_orange
          6'd18: {VGA_R, VGA_G, VGA_B} = 24'h8b3518;  // Middle_orange
          6'd19: {VGA_R, VGA_G, VGA_B} = 24'h8f3719;  // Dark_orange
          default: {VGA_R, VGA_G, VGA_B} = 24'hffffff;  // Default to white
        endcase
      end else begin
        {VGA_R, VGA_G, VGA_B} = {background_r, background_g, background_b};
      end
    end
  end

endmodule

module vga_mem (
    input logic clk,
    input logic [16:0] ra,
    wa,
    input logic write,
    input logic [5:0] wd,
    output logic [5:0] rd
);

  logic [5:0] data[122775:0];  // 480 rows * 256 offset / row, plus up to 150 col offset
  always_ff @(posedge clk) begin
    if (write) data[wa] <= wd;
    rd <= data[ra];
  end
endmodule

module vga_counters (
    input  logic        clk50,
    reset,
    output logic [10:0] hcount,  // hcount[10:1] is pixel column
    output logic [ 9:0] vcount,  // vcount[9:0] is pixel row
    output logic        VGA_CLK,
    VGA_HS,
    VGA_VS,
    VGA_BLANK_n,
    VGA_SYNC_n
);

  /*
 * 640 X 480 VGA timing for a 50 MHz clock: one pixel every other cycle
 *
 * HCOUNT 1599 0             1279       1599 0
 *             _______________             ________
 * ___________|    Video      |____________|  Video
 *
 *
 * |SYNC| BP |<-- HACTIVE -->|FP|SYNC| BP |<-- HACTIVE
 *       _______________________      _____________
 * |____|       VGA_HS          |____|
 */
  // Parameters for hcount
  parameter HACTIVE      = 11'd 1280,
             HFRONT_PORCH = 11'd 32,
             HSYNC        = 11'd 192,
             HBACK_PORCH  = 11'd 96,
             HTOTAL       = HACTIVE + HFRONT_PORCH + HSYNC +
                            HBACK_PORCH; // 1600

  // Parameters for vcount
  parameter VACTIVE      = 10'd 480,
             VFRONT_PORCH = 10'd 10,
             VSYNC        = 10'd 2,
             VBACK_PORCH  = 10'd 33,
             VTOTAL       = VACTIVE + VFRONT_PORCH + VSYNC +
                            VBACK_PORCH; // 525

  logic endOfLine;

  always_ff @(posedge clk50 or posedge reset)
    if (reset) hcount <= 0;
    else if (endOfLine) hcount <= 0;
    else hcount <= hcount + 11'd1;

  assign endOfLine = hcount == HTOTAL - 1;

  logic endOfField;

  always_ff @(posedge clk50 or posedge reset)
    if (reset) vcount <= 0;
    else if (endOfLine)
      if (endOfField) vcount <= 0;
      else vcount <= vcount + 10'd1;

  assign endOfField = vcount == VTOTAL - 1;

  // Horizontal sync: from 0x520 to 0x5DF (0x57F)
  // 101 0010 0000 to 101 1101 1111
  assign VGA_HS = !((hcount[10:8] == 3'b101) & !(hcount[7:5] == 3'b111));
  assign VGA_VS = !(vcount[9:1] == (VACTIVE + VFRONT_PORCH) / 2);

  assign VGA_SYNC_n = 1'b0;  // For putting sync on the green signal; unused

  // Horizontal active: 0 to 1279     Vertical active: 0 to 479
  // 101 0000 0000  1280           01 1110 0000  480
  // 110 0011 1111  1599           10 0000 1100  524
  assign VGA_BLANK_n = !( hcount[10] & (hcount[9] | hcount[8]) ) &
            !( vcount[9] | (vcount[8:5] == 4'b1111) );

  /* VGA_CLK is 25 MHz
    *             __    __    __
    * clk50    __|  |__|  |__|
    *
    *             _____       __
    * hcount[0]__|     |_____|
    */
  assign VGA_CLK = hcount[0];  // 25 MHz clock: rising edge sensitive

endmodule

```

hardware/vga_framebuffer_hw.tcl

```
# TCL File Generated by Component Editor 21.1
# Fri May 10 12:04:38 EDT 2024
# DO NOT MODIFY


#
# vga_framebuffer "VGA Framebuffer" v1.0
#  2024.05.10.12:04:38
#
#

#
# request TCL package from ACDS 16.1
#
package require -exact qsys 16.1


#
# module vga_framebuffer
#
set_module_property DESCRIPTION ""
set_module_property NAME vga_framebuffer
set_module_property VERSION 1.0
set_module_property INTERNAL false
set_module_property OPAQUE_ADDRESS_MAP true
set_module_property AUTHOR ""
set_module_property DISPLAY_NAME "VGA Framebuffer"
set_module_property INSTANTIATE_IN_SYSTEM_MODULE true
set_module_property EDITABLE true
set_module_property REPORT_TO_TALKBACK false
set_module_property ALLOW_GREYBOX_GENERATION false
set_module_property REPORT_HIERARCHY false

```

```
set_module_assignment embeddedsw.dts.vendor "csee4840"
set_module_assignment embeddedsw.dts.name "vga_framebuffer"
set_module_assignment embeddedsw.dts.group "vga"

#
# file sets
#
add_fileset QUARTUS_SYNTH QUARTUS_SYNTH "" ""
set_fileset_property QUARTUS_SYNTH TOP_LEVEL vga_framebuffer
set_fileset_property QUARTUS_SYNTH ENABLE_RELATIVE_INCLUDE_PATHS false
set_fileset_property QUARTUS_SYNTH ENABLE_FILE_OVERWRITE_MODE false
add_fileset_file vga_framebuffer.sv SYSTEM_VERILOG PATH vga_framebuffer.sv TOP_LEVEL_FILE


#
# parameters
#


#
# display items
#


#
# connection point clock
#
add_interface clock clock end
set_interface_property clock clockRate 0
set_interface_property clock ENABLED true
set_interface_property clock EXPORT_OF ""
set_interface_property clock PORT_NAME_MAP ""
set_interface_property clock CMSIS_SVD_VARIABLES ""
set_interface_property clock SVD_ADDRESS_GROUP ""

add_interface_port clock clk clk Input 1


#
# connection point reset
#
add_interface reset reset end
set_interface_property reset associatedClock clock
set_interface_property reset synchronousEdges DEASSERT
set_interface_property reset ENABLED true
set_interface_property reset EXPORT_OF ""
set_interface_property reset PORT_NAME_MAP ""
set_interface_property reset CMSIS_SVD_VARIABLES ""
set_interface_property reset SVD_ADDRESS_GROUP ""

add_interface_port reset reset reset Input 1


#
# connection point avalon_slave_0
```

```
#
add_interface avalon_slave_0 avalon end
set_interface_property avalon_slave_0 addressUnits WORDS
set_interface_property avalon_slave_0 associatedClock clock
set_interface_property avalon_slave_0 associatedReset reset
set_interface_property avalon_slave_0 bitsPerSymbol 8
set_interface_property avalon_slave_0 burstOnBurstBoundariesOnly false
set_interface_property avalon_slave_0 burstcountUnits WORDS
set_interface_property avalon_slave_0 explicitAddressSpan 0
set_interface_property avalon_slave_0 holdTime 0
set_interface_property avalon_slave_0 linewrapBursts false
set_interface_property avalon_slave_0 maximumPendingReadTransactions 0
set_interface_property avalon_slave_0 maximumPendingWriteTransactions 0
set_interface_property avalon_slave_0 readLatency 0
set_interface_property avalon_slave_0 readWaitTime 1
set_interface_property avalon_slave_0 setupTime 0
set_interface_property avalon_slave_0 timingUnits Cycles
set_interface_property avalon_slave_0 writeWaitTime 0
set_interface_property avalon_slave_0 ENABLED true
set_interface_property avalon_slave_0 EXPORT_OF ""
set_interface_property avalon_slave_0 PORT_NAME_MAP ""
set_interface_property avalon_slave_0 CMSIS_SVD_VARIABLES ""
set_interface_property avalon_slave_0 SVD_ADDRESS_GROUP ""

add_interface_port avalon_slave_0 writedata writedata Input 32
add_interface_port avalon_slave_0 write write Input 1
add_interface_port avalon_slave_0 chipselect chipselect Input 1
add_interface_port avalon_slave_0 address address Input 2
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isFlash 0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isMemoryDevice 0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isNonVolatileStorage
0
set_interface_assignment avalon_slave_0 embeddedsw.configuration.isPrintableDevice 0


#
# connection point vga
#
add_interface vga conduit end
set_interface_property vga associatedClock clock
set_interface_property vga associatedReset ""
set_interface_property vga ENABLED true
set_interface_property vga EXPORT_OF ""
set_interface_property vga PORT_NAME_MAP ""
set_interface_property vga CMSIS_SVD_VARIABLES ""
set_interface_property vga SVD_ADDRESS_GROUP ""

add_interface_port vga VGA_B b Output 8
add_interface_port vga VGA_BLANK_n blank_n Output 1
add_interface_port vga VGA_CLK clk Output 1
add_interface_port vga VGA_G g Output 8
add_interface_port vga VGA_HS hs Output 1
add_interface_port vga VGA_R r Output 8
add_interface_port vga VGA_SYNC_n sync_n Output 1
add_interface_port vga VGA_VS vs Output 1

```

Folder software

23 printable files

software/Makefile
software/colors.c
software/colors.h
software/fbputchar.c
software/fbputchar.h
software/game_logic.c
software/generate_verilog_colors.py
software/global_consts.h
software/guitar_reader.c
software/guitar_reader.h
software/guitar_state.c
software/guitar_state.h
software/helpers.c
software/helpers.h
software/high_score.txt
software/single_note_commaless.txt
software/song_data.h
software/sprites.c
software/sprites.h
software/vga_emulator.c
software/vga_emulator.h
software/vga_framebuffer.c
software/vga_framebuffer.h

software/Makefile

```makefile
ifneq (${KERNELRELEASE},)
    # KERNELRELEASE defined: we are being compiled as part of the Kernel
    obj-m := vga_framebuffer.o guitar_reader.o
endif

CC=gcc
CFLAGS=-Wall -Wextra -pedantic -std=c99 -D_XOPEN_SOURCE=600
LDFLAGS=-lSDL2 -lpthread -lpng -lm

SRCS=game_logic.c sprites.c vga_emulator.c guitar_state.c colors.c helpers.c fbputchar.c
OBJS=$(SRCS:.c=.o)
TARGET=game_logic

KERNEL_SOURCE := /usr/src/linux-headers-$(shell uname -r)
PWD := $(shell pwd)

all: $(TARGET)

$(TARGET): $(OBJS)
    $(CC) $(CFLAGS) $(OBJS) -o $(TARGET) $(LDFLAGS)

modules:
    ${MAKE} -C ${KERNEL_SOURCE} SUBDIRS=${PWD} modules CFLAGS="$(CFLAGS)"

clean:
    rm -f $(OBJS) $(TARGET)
    ${MAKE} -C ${KERNEL_SOURCE} SUBDIRS=${PWD} clean
    ${RM} vga_framebuffer.ko

.PHONY: all clean

```

software/colors.c

```
/* colors.c
 *
 * Defines the color palette and related helpers
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */
#include "colors.h"

int get_color_index(Color color) {
  if (color >= 0 && color < COLOR_COUNT)
    // The label itself corresponds to the index in the palette
    return color;
  else
    return -1;
}

int get_color_from_rgb(RGB color) {
  for (int i = 0; i < COLOR_COUNT; i++) {
    if (palette[i].R == color.R && palette[i].G == color.G &&
        palette[i].B == color.B) {
      return i;
    }
  }

  return -1;
}

// Gives the RGB values of each color
RGB palette[COLOR_COUNT] = {[BLACK] = {0, 0, 0},
                            [WHITE] = {255, 255, 255},
                            [RED] = {255, 0, 0},
                            [GREEN] = {0, 255, 0},
                            [BLUE] = {0, 0, 255},
                            [LIGHT_GREEN] = {20, 211, 69},
                            [MIDDLE_GREEN] = {17, 161, 50},
                            [DARK_GREEN] = {16, 162, 55},
                            [LIGHT_RED] = {211, 54, 47},
                            [MIDDLE_RED] = {154, 42, 38},
                            [DARK_RED] = {155, 41, 41},
                            [LIGHT_YELLOW] = {254, 243, 53},
                            [MIDDLE_YELLOW] = {197, 189, 26},
                            [DARK_YELLOW] = {207, 189, 61},
                            [LIGHT_BLUE] = {83, 117, 224},
                            [MIDDLE_BLUE] = {59, 89, 175},
                            [DARK_BLUE] = {64, 89, 171},
                            [LIGHT_ORANGE] = {218, 86, 43},
                            [MIDDLE_ORANGE] = {139, 53, 24},
                            [DARK_ORANGE] = {143, 55, 25}};
```

software/colors.h

```
/* colors.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef COLORS_H
#define COLORS_H

typedef struct {
  unsigned char R;
  unsigned char G;
  unsigned char B;
} RGB;

typedef enum {
  BLACK,
  WHITE,
  RED,
  GREEN,
  BLUE,
  // Note sprite colors
  LIGHT_GREEN,
  MIDDLE_GREEN,
  DARK_GREEN,
  LIGHT_RED,
  MIDDLE_RED,
  DARK_RED,
  LIGHT_YELLOW,
  MIDDLE_YELLOW,
  DARK_YELLOW,
  LIGHT_BLUE,
  MIDDLE_BLUE,
  DARK_BLUE,
  LIGHT_ORANGE,
  MIDDLE_ORANGE,
  DARK_ORANGE,
  COLOR_COUNT // Gives number of predefined colors
} Color;

int get_color_index(
    Color color); // Maps a Color to an int, or returns -1 if not found
int get_color_from_rgb(
    RGB color); // Maps an RGB to an int, or returns -1 if not found

extern RGB palette[COLOR_COUNT]; // Color palette

#define BACKGROUND_COLOR palette[BLACK];

#endif /* COLORS_H */
```

software/fbputchar.c

```
/* fbputchar.c: Framebuffer character generator
 *
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
 *
 * Assumes 32bpp
 *
 */
#include "fbputchar.h"
#include "global_consts.h"

#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>

#include <linux/fb.h>

#include <stdio.h>
#include <unistd.h>

#define FONT_WIDTH 8
#define FONT_HEIGHT 16
#define BITS_PER_PIXEL 32
#define LINE_LENGTH WINDOW_WIDTH * 4

/* 8 X 16 console font from /lib/kbd/consolefonts/lat0-16.psfu.gz

od --address-radix=n --width=16 -v -t x1 -j 4 -N 2048 lat0-16.psfu

*/
static unsigned char font[2048] = {
    0x00, 0x00, 0x7e, 0xc3, 0x99, 0x99, 0xf3, 0xe7, 0xe7, 0xff, 0xe7, 0xe7,
    0x7e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xdc, 0x00,
    0x76, 0xdc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0xf8,
    0xd8, 0xd8, 0xdc, 0xd8, 0xd8, 0xd8, 0xf8, 0x6e, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0xdb, 0xdb, 0xdf, 0xd8, 0xdb, 0x6e,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe,
    0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x88, 0x88, 0xf8,
    0x88, 0x88, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00,
    0x00, 0xf8, 0x80, 0xe0, 0x80, 0x80, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x88, 0x80, 0x88, 0x70, 0x00, 0x3c,
    0x22, 0x3c, 0x24, 0x22, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80,
    0x80, 0xf8, 0x00, 0x3e, 0x20, 0x38, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00,
    0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44, 0x11, 0x44,
    0x11, 0x44, 0x11, 0x44, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa,
    0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0xdd, 0x77, 0xdd, 0x77,
    0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77, 0xdd, 0x77,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
    0xf0, 0xf0, 0xf0, 0xf0, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f,
    0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x00, 0x88, 0xc8, 0xa8,
    0x98, 0x88, 0x00, 0x20, 0x20, 0x20, 0x20, 0x3e, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x88, 0x88, 0x50, 0x50, 0x20, 0x00, 0x3e, 0x08, 0x08, 0x08, 0x08,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x38, 0xe0, 0x38,
    0x0e, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xe0, 0x38, 0x0e, 0x38, 0xe0, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x06, 0x0c, 0xfe, 0x18, 0x30, 0xfe, 0x60, 0xc0, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x1e, 0x7e, 0xfe,
    0x7e, 0x1e, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xc0, 0xf0, 0xfc, 0xfe, 0xfc, 0xf0, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x18, 0x0c, 0xfe, 0x0c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x60, 0xfe, 0x60, 0x30, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18,
    0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x28, 0x6c, 0xfe, 0x6c, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x06, 0x36, 0x66, 0xfe, 0x60, 0x30, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xfe, 0x6e,
    0x6c, 0x6c, 0x6c, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x3c, 0x3c, 0x3c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x24, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x6c,
    0x6c, 0xfe, 0x6c, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x10, 0x10, 0x7c, 0xd6, 0xd0, 0xd0, 0x7c, 0x16, 0x16, 0xd6, 0x7c,
    0x10, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc2, 0xc6, 0x0c, 0x18,
    0x30, 0x60, 0xc6, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x6c,
    0x6c, 0x38, 0x76, 0xdc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30,
    0x30, 0x30, 0x18, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18,
    0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3c, 0xff, 0x3c, 0x66, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7e,
    0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0xce, 0xce, 0xd6, 0xd6, 0xe6, 0xe6, 0xc6, 0x7c,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6,
    0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0xc6, 0xfe, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x7c, 0xc6, 0x06, 0x06, 0x3c, 0x06, 0x06, 0x06, 0xc6, 0x7c,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x1c, 0x3c, 0x6c, 0xcc, 0xfe,
    0x0c, 0x0c, 0x0c, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xc0,
        0xc0, 0xc0, 0xfc, 0x06, 0x06, 0x06, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x38, 0x60, 0xc0, 0xc0, 0xfc, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xc6, 0x06, 0x06, 0x0c, 0x18,
        0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6,
        0xc6, 0xc6, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x06, 0x06, 0x0c, 0x78,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
        0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x18, 0x18, 0x00, 0x00, 0x00, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0c, 0x06,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00,
        0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60,
        0x30, 0x18, 0x0c, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x0c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xde, 0xde,
        0xde, 0xdc, 0xc0, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x38,
        0x6c, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x66, 0x66, 0x66, 0x66, 0xfc,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xc0,
        0xc0, 0xc2, 0x66, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x6c,
        0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x6c, 0xf8, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68, 0x60, 0x62, 0x66, 0xfe,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x78, 0x68,
        0x60, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x66,
        0xc2, 0xc0, 0xc0, 0xde, 0xc6, 0xc6, 0x66, 0x3a, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x18,
        0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x0c,
        0x0c, 0x0c, 0x0c, 0x0c, 0xcc, 0xcc, 0xcc, 0x78, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xe6, 0x66, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0x66, 0xe6,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x60, 0x60, 0x60, 0x60, 0x60,
        0x60, 0x62, 0x66, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xee,
        0xfe, 0xfe, 0xd6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xc6, 0xe6, 0xf6, 0xfe, 0xde, 0xce, 0xc6, 0xc6, 0xc6, 0xc6,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6,
        0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x66,
        0x66, 0x66, 0x7c, 0x60, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xde, 0x7c,
        0x0c, 0x0e, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x66, 0x66, 0x66, 0x7c, 0x6c,
        0x66, 0x66, 0x66, 0xe6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6,
        0xc6, 0x60, 0x38, 0x0c, 0x06, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7e, 0x7e, 0x5a, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6,
        0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6,
        0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x6c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0xee, 0x6c,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0x6c, 0x7c, 0x38, 0x38,
        0x7c, 0x6c, 0xc6, 0xc6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x66,
        0x66, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0xfe, 0xc6, 0x86, 0x0c, 0x18, 0x30, 0x60, 0xc2, 0xc6, 0xfe,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x30, 0x30, 0x30, 0x30, 0x30,
        0x30, 0x30, 0x30, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0xc0, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x3c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x3c,
        0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x00,
    0x00, 0x30, 0x30, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0c, 0x7c,
    0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x60,
    0x60, 0x78, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc0, 0xc0, 0xc0, 0xc6, 0x7c,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x0c, 0x0c, 0x3c, 0x6c, 0xcc,
    0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x38, 0x6c, 0x64, 0x60, 0xf0, 0x60, 0x60, 0x60, 0x60, 0xf0,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc,
    0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0xcc, 0x78, 0x00, 0x00, 0x00, 0xe0, 0x60,
    0x60, 0x6c, 0x76, 0x66, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x06, 0x00, 0x0e, 0x06, 0x06,
    0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x3c, 0x00, 0x00, 0x00, 0xe0, 0x60,
    0x60, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0xe6, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x70, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x34, 0x18,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xec, 0xfe, 0xd6,
    0xd6, 0xd6, 0xd6, 0xc6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66,
    0x66, 0x66, 0x66, 0x7c, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0x0c, 0x1e, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x76, 0x66, 0x60, 0x60, 0x60, 0xf0,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0x60,
    0x38, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x30,
    0x30, 0xfc, 0x30, 0x30, 0x30, 0x30, 0x36, 0x1c, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x76,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66,
    0x66, 0x66, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0xc6, 0xc6, 0xd6, 0xd6, 0xd6, 0xfe, 0x6c, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0x6c, 0x38, 0x38, 0x38, 0x6c, 0xc6,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xc6,
    0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x0c, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0xfe, 0xcc, 0x18, 0x30, 0x60, 0xc6, 0xfe, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x0e, 0x18, 0x18, 0x18, 0x70, 0x18, 0x18, 0x18, 0x18, 0x0e,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x18,
    0x18, 0x18, 0x0e, 0x18, 0x18, 0x18, 0x18, 0x70, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x76, 0xdc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3c,
    0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,
};

/*
 * Draw the given character at the given row/column.
 * fbopen() must be called first.
 */
void fbputchar(unsigned char *framebuffer, char c, int row, int col,
               RGB color) {
  int x, y;
  unsigned char pixels, *pixelp = font + FONT_HEIGHT * c;
  unsigned char mask;
  unsigned char *pixel, *left = framebuffer +
                                (row * FONT_HEIGHT * 2) * LINE_LENGTH +
                                (col * FONT_WIDTH * 2) * BITS_PER_PIXEL / 8;
  for (y = 0; y < FONT_HEIGHT * 2; y++, left += LINE_LENGTH) {
    pixels = *pixelp;
    pixel = left;
    mask = 0x80;
    for (x = 0; x < FONT_WIDTH; x++) {
      if (pixels & mask) {
        pixel[0] = color.R;
        pixel[1] = color.G;
        pixel[2] = color.B;
        pixel[3] = 0;
      } else {
        pixel[0] = 0;
        pixel[1] = 0;
        pixel[2] = 0;
        pixel[3] = 0;
      }
      pixel += 4;
      if (pixels & mask) {
        pixel[0] = color.R;
        pixel[1] = color.G;
        pixel[2] = color.B;
        pixel[3] = 0;
      } else {
        pixel[0] = 0;
        pixel[1] = 0;
        pixel[2] = 0;
        pixel[3] = 0;
      }
      pixel += 4;
      mask >>= 1;
    }
    if (y & 0x1)
      pixelp++;
  }
}

/*
 * Draw the given string at the given row/column.
 * String must fit on a single line: wrap-around is not handled.
 */
void fbputs(unsigned char *framebuffer, const char *s, int row, int col,
            RGB color) {
  char c;
  while ((c = *s++) != 0)
    fbputchar(framebuffer, c, row, col++, color);
}

```

software/fbputchar.h

```
/* fbputchar.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
 */

#ifndef _FBPUTCHAR_H
#define _FBPUTCHAR_H
#include "colors.h"

extern void fbputchar(unsigned char *, char, int, int, RGB);
extern void fbputs(unsigned char *, const char *, int, int, RGB);

#endif
```

software/game_logic.c

```
/* game_logic.c
 *
 * Runs the main game logic, on the FPGA or in emulation
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#include "colors.h"
#include "fbputchar.h"
#include "global_consts.h"
#include "guitar_reader.h"
#include "guitar_state.h"
#include "helpers.h"
#include "song_data.h"
#include "sprites.h"
#include "vga_emulator.h"
#include "vga_framebuffer.h"

#include <SDL2/SDL_blendmode.h>
#include <fcntl.h>
#include <linux/fb.h>
#include <math.h>
#include <ncurses.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

// Set this to 1 to emulate the game. Will not attempt to connect to VGA/drivers
int EMULATING_VGA = 1;

int SCREEN_LINE_LENGTH;
int vga_framebuffer_fd, guitar_fd;
```

```
unsigned char *framebuffer;
pthread_mutex_t framebuffer_mutex = PTHREAD_MUTEX_INITIALIZER,
                controller_mutex = PTHREAD_MUTEX_INITIALIZER;
guitar_state controller_state;

struct {
  int green;
  int red;
  int yellow;
  int blue;
  int orange;
} color_cols_x = {15, 45, 75, 105, 135};

void *update_guitar_state(void *arg) {
  (void)arg; // Suppress unused warning

  while (1) {
    char *guitar_hex = read_note(guitar_fd);
    char *binary_string = hex_string_to_binary(guitar_hex);

    guitar_state note;
    set_note_guitar(&note, binary_string);

    pthread_mutex_lock(&controller_mutex);
    controller_state = note;
    pthread_mutex_unlock(&controller_mutex);

    usleep(16667); // 60 Hz refresh rate
  }
  return NULL;
}

void *update_framebuffer(void *arg) {
  (void)arg; // Suppress warning

  while (1) {
    pthread_mutex_lock(&framebuffer_mutex);
    for (int pixel_row = 0; pixel_row < WINDOW_HEIGHT; pixel_row++) {
      for (int pixel_col = 0; pixel_col < WINDOW_WIDTH; pixel_col++) {
        unsigned char *pixel =
            framebuffer + (pixel_row * WINDOW_WIDTH + pixel_col) * 4;
        RGB pixel_rgb = {pixel[2], pixel[1], pixel[0]};

        vga_framebuffer_arg_t vfba;
        vfba.pixel_writedata = pixel_writedata(get_color_from_rgb(pixel_rgb),
                                               pixel_row, pixel_col);

        if (ioctl(vga_framebuffer_fd, VGA_FRAMEBUFFER_UPDATE, &vfba)) {
          perror("ioctl(VGA_FRAMEBUFFER_UPDATE) failed");
        }
      }
    }
    pthread_mutex_unlock(&framebuffer_mutex);
  }
  return NULL;
}
```

```c

void set_note(note_row *note_state, const char *binary_string) {
  if (note_state == NULL || binary_string == NULL) {
    return; // Error handling: Ensure note_state and binary_string are not NULL
  }

  // Convert the binary string to integer values
  int green = binary_string[7] - '0';
  int red = binary_string[6] - '0';
  int yellow = binary_string[5] - '0';
  int blue = binary_string[4] - '0';
  int orange = binary_string[3] - '0';

  // Assign the values to the struct fields
  note_state->green = green;
  note_state->red = red;
  note_state->yellow = yellow;
  note_state->blue = blue;
  note_state->orange = orange;
}

int hit_notes(guitar_state controller_state, note_row notes) {
  return controller_state.green == notes.green &&
         controller_state.red == notes.red &&
         controller_state.yellow == notes.yellow &&
         controller_state.blue == notes.blue &&
         controller_state.orange == notes.orange;
}

int main() {
  // Color definitions (hardcoded).
  // Inspired by https://oaksstudio.itch.io/guitarheroui, recreated from scratch
  circle_colors green_colors = {.white = palette[WHITE],
                                .light_gray = palette[LIGHT_GREEN],
                                .middle_gray = palette[MIDDLE_GREEN],
                                .dark_gray = palette[DARK_GREEN]};

  circle_colors red_colors = {.white = palette[WHITE],
                              .light_gray = palette[LIGHT_RED],
                              .middle_gray = palette[MIDDLE_RED],
                              .dark_gray = palette[DARK_RED]};

  circle_colors yellow_colors = {.white = palette[WHITE],
                                 .light_gray = palette[LIGHT_YELLOW],
                                 .middle_gray = palette[MIDDLE_YELLOW],
                                 .dark_gray = palette[DARK_YELLOW]};

  circle_colors blue_colors = {.white = palette[WHITE],
                               .light_gray = palette[LIGHT_BLUE],
                               .middle_gray = palette[MIDDLE_BLUE],
                               .dark_gray = palette[DARK_BLUE]};

  circle_colors orange_colors = {.white = palette[WHITE],
                                 .light_gray = palette[LIGHT_ORANGE],
                                 .middle_gray = palette[MIDDLE_ORANGE],
                                 .dark_gray = palette[DARK_ORANGE]};
  // 32 bits/pixel = 4 B/pixel
  unsigned char *next_frame;
  pthread_t fb_update_thread, guitar_thread;
  VGAEmulator emulator;

  init_guitar_state(&controller_state);

  if (EMULATING_VGA) {
    printf("Running in VGA EMULATION MODE\n");
  }

  if ((framebuffer = malloc(WINDOW_WIDTH * WINDOW_HEIGHT * 4)) == NULL) {
    perror("Error allocating framebuffer!\n");
    return 1;
  }

  SCREEN_LINE_LENGTH = WINDOW_WIDTH * 4;

  if ((next_frame = malloc(WINDOW_WIDTH * WINDOW_HEIGHT * 4)) == NULL) {
    perror("Error allocating next_frame!\n");
    return 1;
  }

  // Set black background by default
  memset(framebuffer, 0, WINDOW_WIDTH * WINDOW_HEIGHT * 4);
  // Load necessary sprites into memory
  sprite GH_circle_base = load_sprite("sprites/GH-Circle.png");
  sprite GH_logo = load_sprite("sprites/GH-Logo.png");
  // Generate the sprites for the notes
  generated_circles note_circles =
      generate_circles(GH_circle_base, green_colors, red_colors, yellow_colors,
                       blue_colors, orange_colors);
  // Generate the sprites for the indicators to play:
  green_colors.white = BACKGROUND_COLOR;
  red_colors.white = BACKGROUND_COLOR;
  yellow_colors.white = BACKGROUND_COLOR;
  blue_colors.white = BACKGROUND_COLOR;
  orange_colors.white = BACKGROUND_COLOR;
  generated_circles play_circles_released =
      generate_circles(GH_circle_base, green_colors, red_colors, yellow_colors,
                       blue_colors, orange_colors);
  green_colors.white = green_colors.dark_gray;
  red_colors.white = red_colors.dark_gray;
  yellow_colors.white = yellow_colors.dark_gray;
  blue_colors.white = blue_colors.dark_gray;
  orange_colors.white = orange_colors.dark_gray;
  generated_circles play_circles_held =
      generate_circles(GH_circle_base, green_colors, red_colors, yellow_colors,
                       blue_colors, orange_colors);

  // Set up VGA emulator. Requires libsdl2-dev
  if (EMULATING_VGA) {
    if (VGAEmulator_init(&emulator, framebuffer, &controller_state))
      return 1;
  } else {
    // Set up VGA framebuffer connection
```

```c
    if ((vga_framebuffer_fd = open("/dev/vga_framebuffer", O_WRONLY)) == -1) {
      perror("could not open /dev/vga_framebuffer\n");
      return -1;
    }

    if ((guitar_fd = open("/dev/note_reader", O_RDONLY)) == -1) {
      perror("could not open /dev/note_reader\n");
      return -1;
    }

    if (pthread_create(&fb_update_thread, NULL, &update_framebuffer, NULL) !=
        0) {
      perror("pthread_create(fb_update_thread) failed\n");
      return 1;
    }

    if (pthread_create(&guitar_thread, NULL, update_guitar_state, NULL) != 0) {
      perror("pthread_create(fb_update_thread) failed\n");
      return 1;
    }
  }

  note_row song_rows[NUM_NOTE_ROWS];

  char line[9]; // Buffer to store each line (8 characters + null terminator)
  FILE *file = fopen("single_note_commaless.txt", "r");

  int i = 0;
  while (fgets(line, sizeof(line), file) != NULL) {
    // Remove the newline character if present
    if (line[strlen(line) - 1] == '\n') {
      line[strlen(line) - 1] = '\0';
    }
    if (strlen(line) == 8) {
      // printf("note: %s\n", line);
      note_row note_row;
      set_note(&note_row, line);
      song_rows[i++] = note_row;
    }
  }

  fclose(file);

  int current_bottom_row_idx = 0, num_note_rows = 100;
  double current_bottom_row_Y = 0;
  int note_duration = round((60.0 / SONG_BPM) / NOTES_PER_MEASURE * 1000);

  // The Y coordinate of the middle of the guitar state line
  int guitar_state_line_Y = WINDOW_HEIGHT - 24;
  // How many pixels of "margin" (top and bottom) to apply to each note
  int note_row_veritcal_padding = 12;
  // THe total height including the 24x24 px sprite and the margin
  int note_height_px = 24 + 2 * note_row_veritcal_padding;
  // How many pixels each note row has to move down the screen in one ms
  double note_row_pixels_per_ms = (double)(note_height_px) / note_duration;

  printf("---SONG INFORMATION---\n");
  printf("BPM: %d\n", SONG_BPM);
  printf("Beat duration: %dms\n", note_duration);
  printf("Note row pixels/ms: %f\n", note_row_pixels_per_ms);

  // Start menu
  pthread_mutex_lock(&framebuffer_mutex);
  draw_sprite(GH_logo, framebuffer, WINDOW_WIDTH / 2, 192);

  fbputs(framebuffer, "Press", 8, 2, palette[WHITE]);
  fbputs(framebuffer, "Green", 9, 2, palette[GREEN]);
  fbputs(framebuffer, "To Play", 10, 1, palette[WHITE]);

  pthread_mutex_unlock(&framebuffer_mutex);

  while (1) {
    // Press green to continue
    pthread_mutex_unlock(&controller_mutex);
    if (controller_state.green) {
      pthread_mutex_unlock(&controller_mutex);
      break;
    }
    pthread_mutex_unlock(&controller_mutex);

    usleep(1000);
  }

  long long song_start_time = current_time_in_ms();
  long long last_draw_time = song_start_time;
  long long last_score_adjust_time = song_start_time;

  // For accuracy calculation
  int num_strummed = 0, num_hit = 0;
  double current_accuracy = 0;

  while (1) {
    // Fresh start
    memset(next_frame, 0, WINDOW_WIDTH * WINDOW_HEIGHT * 4);

    long long time_delta = current_time_in_ms() - last_draw_time;
    last_draw_time = current_time_in_ms();

    for (int row_on_screen = 0;
         row_on_screen < WINDOW_HEIGHT / note_height_px + 1; row_on_screen++) {
      if (current_bottom_row_idx + row_on_screen >= num_note_rows)
        break; // We've run out of notes

      note_row row = song_rows[current_bottom_row_idx + row_on_screen];

      int row_y = round(current_bottom_row_Y - note_height_px * row_on_screen);

      if (row.green)
        draw_sprite(note_circles.green, next_frame, color_cols_x.green, row_y);
      if (row.red)
        draw_sprite(note_circles.red, next_frame, color_cols_x.red, row_y);
      if (row.yellow)
        draw_sprite(note_circles.yellow, next_frame, color_cols_x.yellow,
                    row_y);
      if (row.blue)
        draw_sprite(note_circles.blue, next_frame, color_cols_x.blue, row_y);
      if (row.orange)
        draw_sprite(note_circles.orange, next_frame, color_cols_x.orange,
                    row_y);
    }

    current_bottom_row_Y += note_row_pixels_per_ms * time_delta;

    pthread_mutex_lock(&controller_mutex);
    if (controller_state.strum) {
      long long strum_time = current_time_in_ms();

      // Is the bottom note in a playable range, and did we try?
      if (current_bottom_row_Y <= guitar_state_line_Y + 12 &&
          current_bottom_row_Y >= guitar_state_line_Y - 12)
        if (hit_notes(controller_state, song_rows[current_bottom_row_idx])) {
          // We hit the note!
          if (strum_time - last_score_adjust_time <= 150) {
            // We barely missed the previous note. Undo that
            num_strummed--;
          }
          current_bottom_row_idx++;
          current_bottom_row_Y -= (24 + 2 * note_row_veritcal_padding);

          num_strummed++;
          num_hit++;
          last_score_adjust_time = current_time_in_ms();
          goto accuracy_adjust;
        }

      // We missed
      if (strum_time - last_score_adjust_time <= 150)
        // This is the same strum as last time, don't punish them again
        goto accuracy_adjust;

      num_strummed++;
      last_score_adjust_time = current_time_in_ms();

    accuracy_adjust:
      current_accuracy = (num_hit * 100.0) / num_strummed;
    }

    // Draw the Guitar state line
    draw_sprite(controller_state.green ? play_circles_held.green
                                       : play_circles_released.green,
                next_frame, color_cols_x.green, guitar_state_line_Y);
    draw_sprite(controller_state.red ? play_circles_held.red
                                     : play_circles_released.red,
                next_frame, color_cols_x.red, guitar_state_line_Y);
    draw_sprite(controller_state.yellow ? play_circles_held.yellow
                                        : play_circles_released.yellow,
                next_frame, color_cols_x.yellow, guitar_state_line_Y);
    draw_sprite(controller_state.blue ? play_circles_held.blue
                                      : play_circles_released.blue,
                next_frame, color_cols_x.blue, guitar_state_line_Y);
    draw_sprite(controller_state.orange ? play_circles_held.orange
                                        : play_circles_released.orange,
                next_frame, color_cols_x.orange, guitar_state_line_Y);
    pthread_mutex_unlock(&controller_mutex);

    // Score bar
    memset(next_frame, 0, WINDOW_WIDTH * 4 * 31);
    char score_line[11];
    if ((int)round(current_accuracy) < 10)
      sprintf(score_line, "Score: %d%%", (int)round(current_accuracy));
    else if ((int)round(current_accuracy) < 100)
      sprintf(score_line, "Score:%d%%", (int)round(current_accuracy));
    else
      // No colon
      sprintf(score_line, "Score%d%%", (int)round(current_accuracy));
    fbputs(next_frame, score_line, 0, 0, palette[WHITE]);
    memset(next_frame + WINDOW_WIDTH * 4 * 31, 255, WINDOW_WIDTH * 4);

    // Is it time to shift the buffer because a note has gone off-screen?
    if (round(current_bottom_row_Y) >= WINDOW_HEIGHT + 6) {
      // The bottom row is off screen
      current_bottom_row_idx++;
      current_bottom_row_Y -= (24 + 2 * note_row_veritcal_padding);
      // This counts as a miss
      num_strummed++;
      current_accuracy = (num_hit * 100.0) / num_strummed;

      if (current_bottom_row_idx >= num_note_rows + 2) {
        // We are done with the game
        break;
      }
    }

    // Push next frame to buffer
    pthread_mutex_lock(&framebuffer_mutex);
    memcpy(framebuffer, next_frame, WINDOW_WIDTH * WINDOW_HEIGHT * 4);
    pthread_mutex_unlock(&framebuffer_mutex);
  }

  // Game ends
  pthread_mutex_lock(&framebuffer_mutex);
  memset(framebuffer, 0, WINDOW_WIDTH * WINDOW_HEIGHT * 4);

  char *score_1 = (int)round(current_accuracy) < 100 ? "Score:" : "Score";
  char score_2[5];

  if ((int)round(current_accuracy) < 10)
    sprintf(score_2, " %d%%", (int)round(current_accuracy));
  else if ((int)round(current_accuracy) < 100)
    sprintf(score_2, "%d%%", (int)round(current_accuracy));
  else
    // No colon
    sprintf(score_2, "%d%%", (int)round(current_accuracy));

```

```c
  fbputs(framebuffer, "Final", 0, 2, palette[WHITE]);
  fbputs(framebuffer, "Accuracy", 1, 0, palette[WHITE]);
  fbputs(framebuffer, score_1, 2, 0, palette[WHITE]);
  fbputs(framebuffer, score_2, 2, 6, palette[GREEN]);
  fbputs(framebuffer, "Thanks", 12, 1, palette[WHITE]);
  fbputs(framebuffer, "For", 13, 2, palette[WHITE]);
  fbputs(framebuffer, "Playing!", 14, 0, palette[WHITE]);

  // Try reading a high score
  FILE *hs_file = fopen("high_score.txt", "r");

  int high_score;

  if (hs_file == NULL) {
    high_score = 0;
  } else {
    fscanf(hs_file, "%d", &high_score);
    fclose(hs_file);
  }

  char best_score[11];
  sprintf(best_score, "Best: %d%%", high_score);

  int your_score = (int)round(current_accuracy);

  if (your_score > high_score) {
    fbputs(framebuffer, best_score, 6, 0, palette[WHITE]);
    fbputs(framebuffer, "NewRecord", 7, 0, palette[GREEN]);
    high_score = your_score;
  } else
    fbputs(framebuffer, best_score, 7, 0, palette[WHITE]);

  // Open the file for writing
  hs_file = fopen("high_score.txt", "w");

  if (hs_file != NULL) {
    // Write the new high score to the file
    fprintf(hs_file, "%d", high_score);
    fclose(hs_file);
  } else {
    // Error handling if unable to open file for writing
    printf("Error: Unable to open high score file for writing.\n");
  }

  pthread_mutex_unlock(&framebuffer_mutex);

  usleep(150000);

  if (EMULATING_VGA)
    VGAEmulator_destroy(&emulator);
  // Clear sprites
  unload_sprite(GH_circle_base);
  unload_sprite(GH_logo);
  unload_sprites(note_circles);
  unload_sprites(play_circles_released);
  unload_sprites(play_circles_held);

  if (EMULATING_VGA)
    free(framebuffer);

  return 0;
}

```

software/generate_verilog_colors.py

```
"""
generate_verilog_colors.py

A development tool that generates Verilog for vga_framebuffer.sv from palette in colors.c

Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
Columbia University CSEE 4840 - Embedded Systems
"""

import os
import re

if __name__ == "__main__":
    game_logic_path = os.path.join(os.path.dirname(
        os.path.abspath(__file__)), 'colors.c')

    with open(game_logic_path, 'r') as f:
        lines = f.readlines()  # Read all lines from the file
        for i in range(len(lines)):
            if "palette[COLOR_COUNT]" in lines[i]:
                # Stop reading when you reach the line containing palette[COLOR_COUNT]
                break

        palette_start_line = i
        print("case (pixel_data)")

        # Extract {R, G, B} values from each line
        for i in range(palette_start_line, len(lines)):
            # Use regex to find {R, G, B} values
            match = re.search(
                r'\[(\w+)\]\s*=\s*{(\d+), (\d+), (\d+)}', lines[i])
            if match:
                color_name = match.group(1)
                rgb_values = (int(match.group(2)), int(
                    match.group(3)), int(match.group(4)))
                rgb_values = ['{:02x}'.format(x) for x in rgb_values]
                color_index = i - palette_start_line
                print(f"  6'd{color_index}: " + "{VGA_R, VGA_G, VGA_B} = " +
                      f"24'h{rgb_values[0]}{rgb_values[1]}{rgb_values[2]}; // {color_name.capitalize()}")

        print(
            "  default: {VGA_R, VGA_G, VGA_B} = 24'hffffff; // Default to white")
        print("endcase")
```

software/global_consts.h

```c
/* global_consts.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef GLOBAL_CONSTS_H
#define GLOBAL_CONSTS_H

#define WINDOW_WIDTH 150
#define WINDOW_HEIGHT 480

extern int SCREEN_LINE_LENGTH;

#endif /* GLOBAL_CONSTS_H */

```

software/guitar_reader.c

```c
/* Device driver for the Guitar Hero Guitar
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
 * A Platform device implemented using the misc subsystem
 *
 *
 * References:
 * Linux source: Documentation/driver-model/platform.txt
 *               drivers/misc/arm-charlcd.c
 * http://www.linuxforu.com/tag/linux-device-drivers/
 * http://free-electrons.com/docs/
 *
 * Check code style with
 * checkpatch.pl --file --no-tree guitar_reader.c
 */

#include "guitar_reader.h"
#include <linux/errno.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/io.h>
#include <linux/kernel.h>
#include <linux/miscdevice.h>
#include <linux/module.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/platform_device.h>
```

```c
#include <linux/slab.h>
#include <linux/uaccess.h>
#include <linux/version.h>

#define DRIVER_NAME "note_reader"

/* Device registers */
#define FIRST_CHUNK(x) (x)

/*
 * Information about our device
 */
struct guitar_reader_dev {
  struct resource res;    /* Resource: our registers */
  void __iomem *virtbase; /* Where registers can be accessed in memory */
} dev;

/*
 * Reads the guitar state
 */
int read_guitar_state(void) { return ioread8(dev.virtbase); }

/*
 * Handle ioctl() calls from userspace:
 * Read or write the segments on single digits.
 * Note extensive error checking of arguments
 */
static long guitar_reader_ioctl(struct file *f, unsigned int cmd,
                                unsigned long arg) {
  // int chunk = write_background();
  int chunk = read_guitar_state();

  switch (cmd) {

  case GUITAR_READER_READ:
    if (copy_to_user((void __user *)arg, &chunk, sizeof(int)))
      return -EACCES;
    break;

  default:
    return -EINVAL;
  }

  return 0;
}

/* The operations our device knows how to do */
static const struct file_operations guitar_reader_fops = {
    .owner = THIS_MODULE,
    .unlocked_ioctl = guitar_reader_ioctl,
};

/* Information about our device for the "misc" framework -- like a char dev */
static struct miscdevice guitar_reader_misc_device = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = DRIVER_NAME,
    .fops = &guitar_reader_fops,
};

/*
 * Initialization code: get resources (registers) and display
 * a welcome message
 */
static int __init guitar_reader_probe(struct platform_device *pdev) {
  int ret;

  /* Register ourselves as a misc device: creates /dev/guitar_reader */
  ret = misc_register(&guitar_reader_misc_device);

  /* Get the address of our registers from the device tree */
  ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
  if (ret) {
    ret = -ENOENT;
    goto out_deregister;
  }

  /* Make sure we can use these registers */
  if (request_mem_region(dev.res.start, resource_size(&dev.res), DRIVER_NAME) ==
      NULL) {
    ret = -EBUSY;
    goto out_deregister;
  }

  /* Arrange access to our registers */
  dev.virtbase = of_iomap(pdev->dev.of_node, 0);
  if (dev.virtbase == NULL) {
    ret = -ENOMEM;
    goto out_release_mem_region;
  }

  return 0;

out_release_mem_region:
  release_mem_region(dev.res.start, resource_size(&dev.res));
out_deregister:
  misc_deregister(&guitar_reader_misc_device);
  return ret;
}

/* Clean-up code: release resources */
static int guitar_reader_remove(struct platform_device *pdev) {
  iounmap(dev.virtbase);
  release_mem_region(dev.res.start, resource_size(&dev.res));
  misc_deregister(&guitar_reader_misc_device);
  return 0;
}

/* Which "compatible" string(s) to search for in the Device Tree */
#ifdef CONFIG_OF
static const struct of_device_id guitar_reader_of_match[] = {
    {.compatible = "csee4840,note_reader-1.0"},
    {},
};
MODULE_DEVICE_TABLE(of, guitar_reader_of_match);
#endif

/* Information for registering ourselves as a "platform" driver */
static struct platform_driver guitar_reader_driver = {
    .driver =
        {
            .name = DRIVER_NAME,
            .owner = THIS_MODULE,
            .of_match_table = of_match_ptr(guitar_reader_of_match),
        },
    .remove = __exit_p(guitar_reader_remove),
};

/* Called when the module is loaded: set things up */
static int __init guitar_reader_init(void) {
  pr_info(DRIVER_NAME ": init\n");
  return platform_driver_probe(&guitar_reader_driver, guitar_reader_probe);
}

/* Calball when the module is unloaded: release resources */
static void __exit guitar_reader_exit(void) {
  platform_driver_unregister(&guitar_reader_driver);
  pr_info(DRIVER_NAME ": exit\n");
}

module_init(guitar_reader_init);
module_exit(guitar_reader_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Dan Ivanovich & Patrick Cronin, Columbia University");
MODULE_DESCRIPTION("VGA Framebuffer Driver");

```

software/guitar_reader.h

```
/* guitar_reader.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef _GUITAR_READER_H
#define _GUITAR_READER_H

#include <linux/ioctl.h>

#define GUITAR_READER_MAGIC 'q'

/* ioctls and their arguments */
#define GUITAR_READER_READ _IOR(GUITAR_READER_MAGIC, 1, int *)

#endif
```

software/guitar_state.c

```c
/* guitar_state.c
 *
 * Helpers for parsing and managing the state of the guitar
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#include "guitar_state.h"
#include "guitar_reader.h"
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>

pthread_t keyboard_thread;
int RUNNING = 1;

void init_guitar_state(guitar_state *gs) {
  gs->green = 0;
  gs->red = 0;
  gs->yellow = 0;
  gs->blue = 0;
  gs->orange = 0;
  gs->strum = 0;
}

char *read_note(int guitar_fd) {
  int arg;

  if (ioctl(guitar_fd, GUITAR_READER_READ, &arg)) {
    perror("ioctl(GUITAR_READER_READ) failed");
  }

  // Static buffer to hold the string (two characters + null terminator)
  static char result_string[3];

  // Convert the integer value to a two-digit hexadecimal string
  snprintf(result_string, 3, "%02x", arg);

  return result_string;
}

char *hex_to_binary(char hex) {
  switch (hex) {
  case '0':
    return "0000";
  case '1':
    return "0001";
  case '2':
    return "0010";
  case '3':
    return "0011";
  case '4':
    return "0100";
  case '5':
    return "0101";
  case '6':
    return "0110";
  case '7':
    return "0111";
  case '8':
    return "1000";
  case '9':
    return "1001";
  case 'a':
    return "1010";
  case 'b':
    return "1011";
  case 'c':
    return "1100";
  case 'd':
    return "1101";
  case 'e':
    return "1110";
  case 'f':
    return "1111";
  default:
    return NULL;
  }
}

// Function to convert a hexadecimal string to its binary representation
char *hex_string_to_binary(const char *hex_string) {
  size_t length = strlen(hex_string);
  size_t binary_length =
      length * 4; // Each hexadecimal character represents 4 bits
  char *binary_string =
      (char *)malloc(binary_length + 1); // +1 for null terminator

  if (binary_string == NULL) {
    fprintf(stderr, "Memory allocation error\n");
    return NULL;
  }

  binary_string[binary_length] = '\0'; // Null terminate the binary string

  for (size_t i = 0; i < length; i++) {
    char *binary_digit = hex_to_binary(hex_string[i]);
    if (binary_digit == NULL) {
      free(binary_string);
      return NULL;
    }
    strcat(binary_string, binary_digit);
  }

```

```c
  return binary_string;
}

void set_note_guitar(guitar_state *guitar_state, const char *binary_string) {
  if (guitar_state == NULL || binary_string == NULL) {
    return; // Error handling: Ensure note_state and binary_string are not NULL
  }

  // Convert the binary string to integer values
  int green = binary_string[7] - '0';
  int red = binary_string[6] - '0';
  int yellow = binary_string[5] - '0';
  int blue = binary_string[4] - '0';
  int orange = binary_string[3] - '0';
  int strum = binary_string[2] - '0';

  // Assign the values to the struct fields
  guitar_state->green = !green;
  guitar_state->red = !red;
  guitar_state->yellow = !yellow;
  guitar_state->blue = !blue;
  guitar_state->orange = !orange;
  guitar_state->strum = strum;
}
```

software/guitar_state.h

```c
/* guitar_state.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
 */

#ifndef GUITAR_STATE_H
#define GUITAR_STATE_H

typedef struct {
  int green;
  int red;
  int yellow;
  int blue;
  int orange;
  int strum;
} guitar_state;

void init_guitar_state(guitar_state *gs);

char *read_note(int guitar_fd);
char *hex_to_binary(char hex);
char *hex_string_to_binary(const char *hex_string);
void set_note_guitar(guitar_state *guitar_state, const char *binary_string);
#endif
```

software/helpers.c

```c
/* helpers.c
 *
 * Miscellaneous/homeless helper functions
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#include "helpers.h"
#include <stddef.h>
#include <sys/time.h>

long long current_time_in_ms() {
  struct timeval tv;
  gettimeofday(&tv, NULL);
  long long ms =
      (tv.tv_sec * 1000LL) +
      (tv.tv_usec /
       1000); // Convert seconds to ms and add microseconds converted to ms
  return ms;
}

/* Constructs a properly-formatted writedata packet for the Avalon Bus */
uint32_t pixel_writedata(unsigned char pixel_color, int pixel_row,
                         int pixel_col) {
  uint32_t pixel_writedata;
  // Make pixel_color pixel_writedata fits into 6 bits
  pixel_color &= 0x3F;

  // Make sure pixel_col fits into 8 bits
  pixel_col &= 0xFF;

  // Make sure pixel_row fits into 9 bits
  pixel_row &= 0x1FF;

  // Combine the values
  pixel_writedata = 0;
  pixel_writedata |= (uint32_t)pixel_color;       // 6 least significant bits
  pixel_writedata |= ((uint32_t)pixel_col << 6);  // Next 8 bits
  pixel_writedata |= ((uint32_t)pixel_row << 14); // Next 9 bits

  return pixel_writedata;
}
```

software/helpers.h

```c
/* helpers.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef HELPERS_H
#define HELPERS_H

#ifdef __KERNEL__
#include <linux/io.h>
#else
#include <stdint.h>
#endif

long long current_time_in_ms();
uint32_t pixel_writedata(unsigned char pixel_color, int pixel_row,
                         int pixel_col);

#endif /* HELPERS_H */

```

software/high_score.txt

```
80
```

software/single_note_commaless.txt

```
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000010
00000010
00000010
00000010
00000010
00000010
00001000
00001000
00001000
00001000
00001000
00000100
00000100
00010000
00010000
00010000
00010000
00010000
00010000
```

```
00010000
00010000
00010000
00010000
00010000
00010000
00010000
00010000
00010000
00000010
00000010
00000010
00000010
00000010
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000010
00000100
00000100
00000100
00000100
00001010
00001010
00001010
00001010
00001010
00001010
00001010
00001010
00001010
00000010
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
```

```
00000001
00000001
00000001
00000001
00000010
00000100
00000100
00000100
00000100
00000000
00001000
00001000
00001000
00001000
00001000
00001000
00001000
00001000
00001000
00001000
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000001
00000010
00000010
00000010
00000010
00000010
00000010
00000010
00000010
00000010
00000000
00000010
00000010
00000010
00000000
00000100
00001000
00000100
00001000
00001000
00001000
00001000
00001000
00001000
00001000
00001000
```

```
00001000
00001000
00001000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00011000
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000100
00000010
00000010
00000000
00000100
00001000
00000100
00001000
00000001
00000001
00000011
00000011
00000001
00000001
00000011
00000011
00000001
00000001
00000011
00000011
00000001
00000001
00000011
00000011
00000001
00000001
00000011
00000011
00001000
00001000
00001000
00000100
00000100
```

```
00000100
00000010
00000010
00000010
00000001
00000000
00000001
00000001
00000001
00000001
```

software/song_data.h

```
/* song_data.h
 *
 * Defines key information about the song
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef SONG_DATA_H
#define SONG_DATA_H

#define NUM_NOTE_ROWS 300      // Number of note rows in song_data buffer
#define SONG_BPM 137           // Barracuda's BPM
#define NOTES_PER_MEASURE 1.75 // How many note rows per measure

// Each of these is a bool: 1 if there's one of these notes in this line
typedef struct {
  int green;
  int red;
  int yellow;
  int blue;
  int orange;
} note_row;

#endif /* SONG_DATA_H */

```

software/sprites.c

```
/* sprites.c
 *
 * Methods to handle the loading, drawing, manipulating, and destruction of
 * sprites
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#include "sprites.h"
#include "global_consts.h"
#include <stdlib.h>
```

```c
#include <string.h>

// Adapted from https://gist.github.com/niw/5963798
sprite load_sprite(char *filename) {
  sprite loaded_sprite;
  FILE *fp = fopen(filename, "rb");

  png_structp png =
      png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
  if (!png) {
    perror("png_create_read_struct() encountered a fatal error!");
    exit(EXIT_FAILURE);
  }

  png_infop info = png_create_info_struct(png);
  if (!info) {
    perror("png_create_info_struct() encountered a fatal error!");
    exit(EXIT_FAILURE);
  }

  if (setjmp(png_jmpbuf(png))) {
    perror("setjmp() encountered a fatal error!");
    exit(EXIT_FAILURE);
  }

  png_init_io(png, fp);

  png_read_info(png, info);

  loaded_sprite.width = png_get_image_width(png, info);
  loaded_sprite.height = png_get_image_height(png, info);
  png_byte color_type = png_get_color_type(png, info);
  png_byte bit_depth = png_get_bit_depth(png, info);

  if (bit_depth == 16)
    png_set_strip_16(png);

  if (color_type == PNG_COLOR_TYPE_PALETTE)
    png_set_palette_to_rgb(png);

  // PNG_COLOR_TYPE_GRAY_ALPHA is always 8 or 16bit depth.
  if (color_type == PNG_COLOR_TYPE_GRAY && bit_depth < 8)
    png_set_expand_gray_1_2_4_to_8(png);

  if (png_get_valid(png, info, PNG_INFO_tRNS))
    png_set_tRNS_to_alpha(png);

  // These color_type don't have an alpha channel then fill it with 0xff.
  if (color_type == PNG_COLOR_TYPE_RGB || color_type == PNG_COLOR_TYPE_GRAY ||
      color_type == PNG_COLOR_TYPE_PALETTE)
    png_set_filler(png, 0xFF, PNG_FILLER_AFTER);

  if (color_type == PNG_COLOR_TYPE_GRAY ||
      color_type == PNG_COLOR_TYPE_GRAY_ALPHA)
    png_set_gray_to_rgb(png);

  png_read_update_info(png, info);

  loaded_sprite.B_per_row = png_get_rowbytes(png, info);

  png_bytep *row_pointers =
      (png_bytep *)malloc(sizeof(png_bytep) * loaded_sprite.height);

  for (int y = 0; y < loaded_sprite.height; y++) {
    row_pointers[y] = (png_byte *)malloc(loaded_sprite.B_per_row);
  }

  png_read_image(png, row_pointers);

  loaded_sprite.pixel_buffer =
      malloc(loaded_sprite.height * loaded_sprite.B_per_row * 4);

  for (int y = 0; y < loaded_sprite.height; y++) {
    memcpy(loaded_sprite.pixel_buffer + y * loaded_sprite.B_per_row * 4,
           row_pointers[y], loaded_sprite.B_per_row);
    free(row_pointers[y]);
  }

  free(row_pointers);

  fclose(fp);
  png_destroy_read_struct(&png, &info, NULL);

  return loaded_sprite;
}

sprite deep_copy_sprite(sprite original) {
  sprite copy;

  copy.pixel_buffer = malloc(original.height * original.B_per_row * 4);
  if (copy.pixel_buffer == NULL) {
    // Handle memory allocation error
    perror("Error allocating memory for pixel buffer");
    exit(EXIT_FAILURE);
  }

  // Copy the pixel data
  memcpy(copy.pixel_buffer, original.pixel_buffer,
         original.height * original.B_per_row * 4);

  // Copy other fields
  copy.width = original.width;
  copy.height = original.height;
  copy.B_per_row = original.B_per_row;

  return copy;
}

void unload_sprite(sprite loaded_sprite) { free(loaded_sprite.pixel_buffer); }
void unload_sprites(generated_circles circles) {
  unload_sprite(circles.green);
  unload_sprite(circles.red);
```

```c
  unload_sprite(circles.yellow);
  unload_sprite(circles.blue);
  unload_sprite(circles.orange);
}

void sprite_for_each_pixel(sprite loaded_sprite,
                           void (*fn)(png_bytep px, int px_row, int px_col)) {
  for (int y = 0; y < loaded_sprite.height; y++) {
    for (int x = 0; x < loaded_sprite.width; x++) {
      png_bytep px = &(
          loaded_sprite.pixel_buffer[y * loaded_sprite.B_per_row * 4 + x * 4]);
      fn(px, y, x);
    }
  }
}

void draw_sprite(sprite loaded_sprite, unsigned char *framebuffer, int screenX,
                 int screenY) {
  // Determine the coordinates of the top left corner of the sprite on the
  // screen
  int tl[] = {screenX - loaded_sprite.width / 2,
              screenY - loaded_sprite.height / 2};

  for (int sprite_row = 0; sprite_row < loaded_sprite.height; sprite_row++) {
    for (int sprite_col = 0; sprite_col < loaded_sprite.width; sprite_col++) {
      png_bytep px = &(
          loaded_sprite.pixel_buffer[sprite_row * loaded_sprite.B_per_row * 4 +
                                     sprite_col * 4]);

      unsigned char R = (unsigned char)px[0], G = (unsigned char)px[1],
                    B = (unsigned char)px[2];

      if (!pixel_visible(px))
        continue;

      // Determine the offset of the framebuffer for this pixel
      int screen_x = tl[0] + sprite_col;
      int screen_y = tl[1] + sprite_row;

      if (screen_x < 0 || screen_y < 0)
        continue;

      long long framebuffer_offset =
          screen_y * SCREEN_LINE_LENGTH + screen_x * 4;

      if (framebuffer_offset < 0 ||
          framebuffer_offset >= WINDOW_WIDTH * WINDOW_HEIGHT * 4 ||
          screen_x < 0 || screen_x >= WINDOW_WIDTH || screen_y < 0 ||
          screen_y >= WINDOW_HEIGHT)
        continue;

      // printf("Drawing (R: %d, G: %d, B: %d) at screen coords (%d, %d), png "
      //  "coords (%d, %d)\n",
      //  R, G, B, screen_x, screen_y, sprite_col, sprite_row);

      // Set R, G, B
      (framebuffer + framebuffer_offset)[2] = R;
      (framebuffer + framebuffer_offset)[1] = G;
      (framebuffer + framebuffer_offset)[0] = B;
    }
  }
}

// Modifies in-place, so the return is only needed to condense generate_circles
sprite color_circle(sprite circle_base, circle_colors colors) {
  for (int y = 0; y < circle_base.height; y++) {
    for (int x = 0; x < circle_base.width; x++) {
      png_bytep px =
          &(circle_base.pixel_buffer[y * circle_base.B_per_row * 4 + x * 4]);

      int pixel_avg = average_pixel(px);
      RGB pixel_color;

      if (WHITE_THRESHOLD - COLOR_SELECTION_RANGE <= pixel_avg &&
          pixel_avg <= WHITE_THRESHOLD + COLOR_SELECTION_RANGE)
        pixel_color = colors.white;
      else if (DARK_GRAY_THRESHOLD - COLOR_SELECTION_RANGE <= pixel_avg &&
               pixel_avg <= DARK_GRAY_THRESHOLD + COLOR_SELECTION_RANGE)
        pixel_color = colors.dark_gray;
      else if (MIDDLE_GRAY_THRESHOLD - COLOR_SELECTION_RANGE <= pixel_avg &&
               pixel_avg <= MIDDLE_GRAY_THRESHOLD + COLOR_SELECTION_RANGE)
        pixel_color = colors.middle_gray;
      else if (LIGHT_GRAY_THRESHOLD - COLOR_SELECTION_RANGE <= pixel_avg &&
               pixel_avg <= LIGHT_GRAY_THRESHOLD + COLOR_SELECTION_RANGE)
        pixel_color = colors.light_gray;
      else
        continue; // We will not modify this pixel

      // Update pixel color from template
      px[0] = pixel_color.R;
      px[1] = pixel_color.G;
      px[2] = pixel_color.B;
      px[3] = 255;
    }
  }

  return circle_base;
}

generated_circles
generate_circles(sprite circle_base, circle_colors green_colors,
                 circle_colors red_colors, circle_colors yellow_colors,
                 circle_colors blue_colors, circle_colors orange_colors) {
  generated_circles circles;

  // Make deep copies of the original circle sprite for each color & converts
  circles.green = color_circle(deep_copy_sprite(circle_base), green_colors);
  circles.red = color_circle(deep_copy_sprite(circle_base), red_colors);
  circles.yellow = color_circle(deep_copy_sprite(circle_base), yellow_colors);
  circles.blue = color_circle(deep_copy_sprite(circle_base), blue_colors);
  circles.orange = color_circle(deep_copy_sprite(circle_base), orange_colors);

  return circles;
}

int average_pixel(png_bytep px) { return (px[0] + px[1] + px[2]) / 3; }

int pixel_visible(png_bytep px) {
  return px[3] >= 127; // Our VGA doesn't support transparency anyways
}

// For debugging purposes
void print_pixel_data(png_bytep px, int px_row, int px_col) {
  printf("Got pixel: row %d, col %d = RGBA(%3d, %3d, %3d, %3d)\n", px_row,
         px_col, px[0], px[1], px[2], px[3]);
}

```

software/sprites.h

```
/* sprites.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef SPRITES_H
#define SPRITES_H
#include "colors.h"
#include <png.h>

typedef struct {
  char *filename;
  unsigned char *pixel_buffer;

  int width;
  int height;
  int B_per_row;
} sprite;

typedef struct {
  sprite green;
  sprite red;
  sprite yellow;
  sprite blue;
  sprite orange;
} generated_circles;

#define DARK_GRAY_THRESHOLD 70
#define MIDDLE_GRAY_THRESHOLD 125
#define LIGHT_GRAY_THRESHOLD 180
#define WHITE_THRESHOLD 255
#define COLOR_SELECTION_RANGE 5

// The RGB values to replace the colors with. Key:
// white: replaces (WHITE_THRESHOLD, WHITE_THRESHOLD, WHITE_THRESHOLD) +/-
// COLOR_SELECTION_RANGE light_gray: replaces (LIGHT_GRAY_THRESHOLD,
// LIGHT_GRAY_THRESHOLD, LIGHT_GRAY_THRESHOLD) +/- COLOR_SELECTION_RANGE
// middle_gray: replaces (MIDDLE_GRAY_THRESHOLD, MIDDLE_GRAY_THRESHOLD,
// MIDDLE_GRAY_THRESHOLD) +/- COLOR_SELECTION_RANGE dark_gray: replaces
// (DARK_GRAY_THRESHOLD, DARK_GRAY_THRESHOLD, DARK_GRAY_THRESHOLD) +/-
// COLOR_SELECTION_RANGE This assumes you're providing a grayscale image and
// uses average_pixel(). Design your base correctly!
typedef struct {
  RGB white;
  RGB light_gray;
  RGB middle_gray;
  RGB dark_gray;
} circle_colors;

// Load a sprite from a filename
sprite load_sprite(char *filename);
// Free remaining memory
void unload_sprite(sprite loaded_sprite);
void unload_sprites(generated_circles circles);

void sprite_for_each_pixel(sprite loaded_sprite,
                           void (*fn)(png_bytep px, int px_row, int px_col));

// Draws the loaded_sprite centered around screenX and screenY
// Considers the top left corner of the screen (0, 0);
void draw_sprite(sprite loaded_sprite, unsigned char *framebuffer, int screenX,
                 int screenY);

// Performs a deep copy of the given sprite. Does NOT copy the filename
sprite deep_copy_sprite(sprite original);

// Returns the average of the RGB values
int average_pixel(png_bytep px);
// Returns 1 if the pixel Alpha is high enough to be visible on our VGA
int pixel_visible(png_bytep px);

void print_pixel_data(png_bytep px, int px_row, int px_col);

// Uses the template information in the base circle to generate the colored
// sprites
generated_circles
generate_circles(sprite circle_base, circle_colors green_colors,
                 circle_colors red_colors, circle_colors yellow_colors,
                 circle_colors blue_colors, circle_colors orange_colors);

#endif /* SPRITES_H */
```

software/vga_emulator.c

```
/* vga_emulator.c
 *
 * Defines and controls a custom VGA emulation system.
 * This also emulates the guitar via keyboard.
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
```

```c
 */

#include "vga_emulator.h"
#include "global_consts.h"
#include "guitar_state.h"
#include <SDL2/SDL_events.h>
#include <unistd.h>

extern int SCREEN_LINE_LENGTH;

void *render(void *args) {
  VGAEmulator *emulator = (VGAEmulator *)args;
  SDL_Surface *surface = emulator->surface;
  unsigned char *framebuffer = emulator->framebuffer;

  while (emulator->running) {
    for (int y = 0; y < WINDOW_HEIGHT; ++y) {
      for (int x = 0; x < WINDOW_WIDTH; ++x) {
        unsigned char *pixel = framebuffer + (y * WINDOW_WIDTH + x) * 4;

        Uint8 r = pixel[2];
        Uint8 g = pixel[1];
        Uint8 b = pixel[0];

        Uint32 color = SDL_MapRGB(surface->format, r, g, b);
        SDL_Rect pixel_rect = {x, y, 1, 1};

        SDL_FillRect(surface, &pixel_rect, color);
      }
    }
    SDL_UpdateWindowSurface(emulator->window);
    usleep(16667); // 60 Hz refresh rate
  }
  return NULL;
}

void *handle_events(void *args) {
  VGAEmulator *emulator = (VGAEmulator *)args;
  SDL_Event event;
  while (1) {
    while (emulator->running && SDL_PollEvent(&event)) {
      if (event.type == SDL_QUIT) {
        VGAEmulator_destroy(emulator);
        return NULL;
      } else if (event.type == SDL_KEYDOWN || event.type == SDL_KEYUP) {
        int new_value = event.type == SDL_KEYDOWN;
        switch (event.key.keysym.sym) {
        case SDLK_1:
          emulator->gs->green = new_value;
          break;
        case SDLK_2:
          emulator->gs->red = new_value;
          break;
        case SDLK_3:
          emulator->gs->yellow = new_value;
          break;
        case SDLK_4:
          emulator->gs->blue = new_value;
          break;
        case SDLK_5:
          emulator->gs->orange = new_value;
          break;
        case SDLK_SPACE:
          emulator->gs->strum = new_value;
          break;
        }
      }
    }
  }
}

int VGAEmulator_init(VGAEmulator *emulator, unsigned char *framebuffer,
                     guitar_state *gs) {
  // Initialize SDL
  if (SDL_Init(SDL_INIT_VIDEO) < 0) {
    printf("SDL could not initialize! SDL_Error: %s\n", SDL_GetError());
    return 1;
  }

  // Create window
  emulator->window = SDL_CreateWindow("VGA Emulator", SDL_WINDOWPOS_UNDEFINED,
                                      SDL_WINDOWPOS_UNDEFINED, WINDOW_WIDTH,
                                      WINDOW_HEIGHT, SDL_WINDOW_SHOWN);
  if (emulator->window == NULL) {
    printf("Window could not be created! SDL_Error: %s\n", SDL_GetError());
    exit(1);
  }

  // Get the window's surface
  emulator->surface = SDL_GetWindowSurface(emulator->window);

  emulator->framebuffer = framebuffer;
  emulator->running = 1;
  emulator->gs = gs;

  if (pthread_create(&emulator->render_thread, NULL, render, emulator) != 0) {
    printf("Error creating render thread\n");
    return 2;
  }

  if (pthread_create(&emulator->event_thread, NULL, handle_events, emulator) !=
      0) {
    printf("Error creating event handling thread\n");
    return 2;
  }

  return 0;
}

void VGAEmulator_destroy(VGAEmulator *emulator) {
  emulator->running = 0;

  // Wait for the threads to finish
  pthread_join(emulator->render_thread, NULL);
  pthread_join(emulator->event_thread, NULL);

  // Clean up SDL resources
  SDL_DestroyWindow(emulator->window);
  SDL_Quit();
}

```

software/vga_emulator.h

```
/* vga_emulator.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 */

#ifndef VGA_EMULATOR_H
#define VGA_EMULATOR_H

#include "guitar_state.h"
#include <SDL2/SDL.h>
#include <pthread.h>

typedef struct {
  SDL_Window *window;
  SDL_Surface *surface;
  guitar_state *gs;
  pthread_t render_thread;
  pthread_t event_thread;
  int running;
  unsigned char *framebuffer;
} VGAEmulator;

// Initialize the VGA emulator
int VGAEmulator_init(VGAEmulator *emulator, unsigned char *framebuffer,
                     guitar_state *gs);

// Destroy the VGA emulator
void VGAEmulator_destroy(VGAEmulator *emulator);

#endif /* VGA_EMULATOR_H */

```

software/vga_framebuffer.c

```
/* Device driver for the Guitar Hero Guitar
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
```

```c
 * A Platform device implemented using the misc subsystem
 *
 * References:
 * Linux source: Documentation/driver-model/platform.txt
 *               drivers/misc/arm-charlcd.c
 * http://www.linuxforu.com/tag/linux-device-drivers/
 * http://free-electrons.com/docs/
 *
 * "make" to build
 * insmod vga_framebuffer.ko
 *
 * Check code style with
 * checkpatch.pl --file --no-tree vga_framebuffer.c
 */

#include "vga_framebuffer.h"
#include "colors.h"
#include "global_consts.h"
#include <linux/errno.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/io.h>
#include <linux/kernel.h>
#include <linux/miscdevice.h>
#include <linux/module.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/platform_device.h>
#include <linux/slab.h>
#include <linux/uaccess.h>
#include <linux/version.h>

#define DRIVER_NAME "vga_framebuffer"

/* Device registers */
#define FIRST_CHUNK(x) (x)

/*
 * Information about our device
 */
struct framebuffer_dev {
  struct resource res;    /* Resource: our registers */
  void __iomem *virtbase; /* Where registers can be accessed in memory */
} dev;

/*
 * Write segments of a single digit
 * Assumes digit is in range and the device information has been set up
 */
static void write_background(uint32_t writedata) {
  iowrite32(writedata, FIRST_CHUNK(dev.virtbase));
}

/*
 * Handle ioctl() calls from userspace:
 * Read or write the segments on single digits.
 * Note extensive error checking of arguments
 */
static long vga_framebuffer_ioctl(struct file *f, unsigned int cmd,
                                  unsigned long arg) {
  vga_framebuffer_arg_t vfba;

  switch (cmd) {
  case VGA_FRAMEBUFFER_UPDATE:
    if (copy_from_user(&vfba, (vga_framebuffer_arg_t *)arg,
                       sizeof(vga_framebuffer_arg_t)))
      return -EACCES;
    write_background(vfba.pixel_writedata);
    break;

  default:
    return -EINVAL;
  }

  return 0;
}

/* The operations our device knows how to do */
static const struct file_operations vga_framebuffer_fops = {
    .owner = THIS_MODULE,
    .unlocked_ioctl = vga_framebuffer_ioctl,
};

/* Information about our device for the "misc" framework -- like a char dev */
static struct miscdevice vga_framebuffer_misc_device = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = DRIVER_NAME,
    .fops = &vga_framebuffer_fops,
};

/*
 * Initialization code: get resources (registers) and display
 * a welcome message
 */
static int __init vga_framebuffer_probe(struct platform_device *pdev) {
  int ret;

  /* Register ourselves as a misc device: creates /dev/vga_framebuffer */
  ret = misc_register(&vga_framebuffer_misc_device);

  /* Get the address of our registers from the device tree */
  ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
  if (ret) {
    ret = -ENOENT;
    goto out_deregister;
  }

  /* Make sure we can use these registers */
  if (request_mem_region(dev.res.start, resource_size(&dev.res), DRIVER_NAME) ==
      NULL) {
    ret = -EBUSY;
    goto out_deregister;
  }

  /* Arrange access to our registers */
  dev.virtbase = of_iomap(pdev->dev.of_node, 0);
  if (dev.virtbase == NULL) {
    ret = -ENOMEM;
    goto out_release_mem_region;
  }

  return 0;

out_release_mem_region:
  release_mem_region(dev.res.start, resource_size(&dev.res));
out_deregister:
  misc_deregister(&vga_framebuffer_misc_device);
  return ret;
}

/* Clean-up code: release resources */
static int vga_framebuffer_remove(struct platform_device *pdev) {
  iounmap(dev.virtbase);
  release_mem_region(dev.res.start, resource_size(&dev.res));
  misc_deregister(&vga_framebuffer_misc_device);
  return 0;
}

/* Which "compatible" string(s) to search for in the Device Tree */
#ifdef CONFIG_OF
static const struct of_device_id vga_framebuffer_of_match[] = {
    {.compatible = "csee4840,vga_framebuffer-1.0"},
    {},
};
MODULE_DEVICE_TABLE(of, vga_framebuffer_of_match);
#endif

/* Information for registering ourselves as a "platform" driver */
static struct platform_driver vga_framebuffer_driver = {
    .driver =
        {
            .name = DRIVER_NAME,
            .owner = THIS_MODULE,
            .of_match_table = of_match_ptr(vga_framebuffer_of_match),
        },
    .remove = __exit_p(vga_framebuffer_remove),
};

/* Called when the module is loaded: set things up */
static int __init vga_framebuffer_init(void) {
  pr_info(DRIVER_NAME ": init\n");
  return platform_driver_probe(&vga_framebuffer_driver, vga_framebuffer_probe);
}

/* Calball when the module is unloaded: release resources */
static void __exit vga_framebuffer_exit(void) {
  platform_driver_unregister(&vga_framebuffer_driver);
  pr_info(DRIVER_NAME ": exit\n");
```

```
}

module_init(vga_framebuffer_init);
module_exit(vga_framebuffer_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Dan Ivanovich & Patrick Cronin, Columbia University");
MODULE_DESCRIPTION("VGA Framebuffer Driver");

```

software/vga_framebuffer.h

```
/* vga_framebuffer.h
 *
 * Patrick Cronin, Dan Ivanovich, & Kiryl Beliauski
 * Columbia University CSEE 4840 - Embedded Systems
 *
 * Adapted from code by Stephen A. Edwards, Columbia University
 */

#ifndef _VGA_FRAMEBUFFER_H
#define _VGA_FRAMEBUFFER_H

#include <linux/ioctl.h>
#ifdef __KERNEL__
#include <linux/io.h>
#else
#include <stdint.h>
#endif

typedef struct {
  uint32_t pixel_writedata;
} vga_framebuffer_arg_t;

#define VGA_FRAMEBUFFER_MAGIC 'q'

/* ioctls and their arguments */
#define VGA_FRAMEBUFFER_UPDATE                                                  \
  _IOW(VGA_FRAMEBUFFER_MAGIC, 1, vga_framebuffer_arg_t *)

#endif

```