UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Checkpoint 4
Paint Engine

1.0 Motivation

This is the checkpoint where we allow some creativity. I.E. this is the "fun" checkpoint. While we will provide basic requirements for the paint engine, you are free to implement the extra "bells and whistles" to make the project that much more exciting. There may even be extra credit opportunities.

In addition to just coming up with the design for the paint engine, this checkpoint also involves wiring up the entire project. You will use your modules from checkpoint one (N64 controller) and use the controller output to move the paint cursor. You will use checkpoint three (SDRAM) as a "canvas" for your paint area. Then finally, you will use your checkpoint two (Video) to display your paint area onto a standard LCD monitor.

2.0 Introduction

The module you will building for this last checkpoint will be the basis of the paint engine.

The primary responsibilities of your module are:

1. Display a paint cursor on the LCD monitor
    a. You will need to keep track of its location and size
    b. The position of this cursor will be controlled by a N64 controller
2. "Paint" the area under the cursor when the user presses the paint button.

The requirements for this checkpoint are a lot less stringent compared to the previous three checkpoints. While this means you are allowed much more freedom in designing this checkpoint, this also means **we will not be telling you exactly how to implement your paint engine.**

3.0 Prelab

Please make sure to complete the prelab before you attend your lab section. **You will not be able to finish this checkpoint in 3hrs!**

1. **Read this handout thoroughly**.
    a. Pay particular attention to section **4.0 Lab Procedure** as it describes what you will be doing in detail.
2. **Start your design ahead of time**.
    a. Begin with **schematics** and **bubble-and-arc** diagrams
    b. Come prepared for your design review
    c. Start building your testbench early

i. Perhaps have one person design the module and the other design the testbench

3. You will need **at least the entire 3hr lab**!
   a. You will need to test and debug your verilog thoroughly.
   b. **You must build a reliable interface with a real hardware component!**

4.0 Lab Procedure

Remember to **manage your Verilog, projects and folders well**. Doing a poor job of managing your files can cost you **hours of rewriting code**, if you accidentally delete your files.

4.1 PaintEngine.v

This is the **main module you will need to build** for this checkpoint.

Shown in Table 1 below is a **sample** port specification list for PaintEngine.v. You are free to use whatever i/o ports necessary in order to wire up with your other checkpoints and/or to implement other features.

Signal	Width	Dir	Description
`N64Status`	30	I	N64 Controller button data
`NewFrame`	1	I	Indicates a new frame is being drawn
`Color`	32	O	Selected color of the paint brush
`CursorPosition`	x	O	Position of the paint brush
`CursorSize`	x	O	Size of the paint brush
`IsPainting`	1	O	The signal to write to the SDRAM
`EraseAll`	1	O	The command to erase the SDRAM buffer
`Clock`	1	I	The `Clock` signal
`Reset`	1	I	The `Reset` signal

Table 1: Sample Port Specification for PaintEngine.v

4.2 PaintRenderer.v

This module takes the outputs from your paint engine (PaintEngine.v) and converts them to the format accepted by your SDRAM controller. Table 2 below shows a **sample** port list for PaintRenderer.v. Keep in mind that you **do not** have to follow these port specifications; if you do not understand these ports descriptions, you can implement a set of ports that make more sense to you.

Signal	Width	Dir	Description
`Color`	32	I	Selected color of the paint brush
`CursorPosition`	x	I	Position of the paint brush
`CursorSize`	x	I	Size of the paint brush
`IsPainting`	1	I	The signal to write to the SDRAM
`EraseAll`	1	I	The command to erase the SDRAM buffer

`RendDin`	32	O	Data to SDRAM
`RendAin`	24	O	SDRAM Address
`RendInRequest`	1	I	This signal indicates that the arbiter is requesting a new operation.
`RendInDataRequest`	1	I	This signal is used by the arbiter to request new data to be written to the SDRAM.
`RendInAdressEnable`	1	I	When this signal is on, the paint renderer should drive the address for its current request onto the `AIn` bus.
`RendInValid`	1	O	Indicates that renderer has a valid operation it wishes the SDRAM to perform.
`RendInMask`	1	O	Mask to control whether each individual data gets written or not in a burst.
`EraseAll`	1	O	The command to erase the SDRAM buffer
`Clock`	1	I	The `Clock` signal
`Reset`	1	I	The `Reset` signal

Table 2: Sample Port Specification for PaintRenderer.v

4.3 Requirements

While there is nothing stopping you from implementing a full-scale Photoshop clone, there are a couple of minimum functions we ask you to implement.

- A cursor indicating the "brush" location. This visible cursor must be controllable via the analog stick on the N64 controller.
- The ability to modify the paint area by "painting"; i.e. areas change color after the paint cursor moves over.
- The ability to paint with more than two colors.
- A "erase all" function. This command will clear the entire screen to a background color.

5.0 Checkpoint4 Checkoff

Name: _____ Name: _____
Section: _____

I Final Project Demo _____ (100%)

II Total: _____
III TA: _____
IV Date: _____
V Hours Spent: _____

RevA – 10/4/2005	Po-kai Chen	Original design