6.111 Final Project Proposal

Venkat Chandar and Ben Gelb

November 4, 2005

Overview

Our project is a real-time video editing system. This system will be able to modify an input video stream in several ways. First, we will implement a zoom feature. This will enlarge a portion of image by a variable scale factor selected by the user. The second feature we will implement is overlay. The user will be able to type in text to overlay on the video. Finally, we will implement a trace function. This allows the user to draw an overlay object freehand using a mouse.

System Description

Figure 1 shows a block diagram for the design of our video processor system.

Our system has keyboard and mouse inputs inputs, and a video input. The output is a modified video signal. From the block diagram, we see that that several intermediate modules are used to generate final output. The video stream is first processed by the video input module. This module interfaces with the video input hardware on the 6.111 labkit and produces horizontal and vertical sync signals, a field signal, and a 24-bit pixel value encoded in YCrCb format. These outputs are fed to the zoom module. The 6.111 labkit includes sample code for the video input module. We plan to use this code with minimal, if any, modification.

Framegrab Module

The framegrab module takes in the horizontal and vertical sync signals from the input module, along with the frame and pixel signals. The motivation for the frame grab module is that the user could switch to an alternate video stream, load in a frame, and switch back to the original video stream. The stored frame can then be superimposed onto the video stream in different ways by the blue screen or overlay modules.

The framegrab module will write one frame to the ZBT memory. The control module sends a trigger signal, which is asserted when the frame grab module should store the current frame. By storing a frame in the memory, the overlay module can read this frame, scale it to a smaller size, and superimpose it onto the video stream for a picture-in-picture effect. Also, the blue screen module can superimpose the stored frame into the background. The framegrab module will be responsible for making sure that the frame buffer is never accessed simultaneously by the blue screen and overlay modules. To accomplish, this the overlay and blue screen modules send inputs to the framegrab module, and receive the data in the memory from the framegrab module. The address_select input from the control module is used to determine whether to allow the blue screen

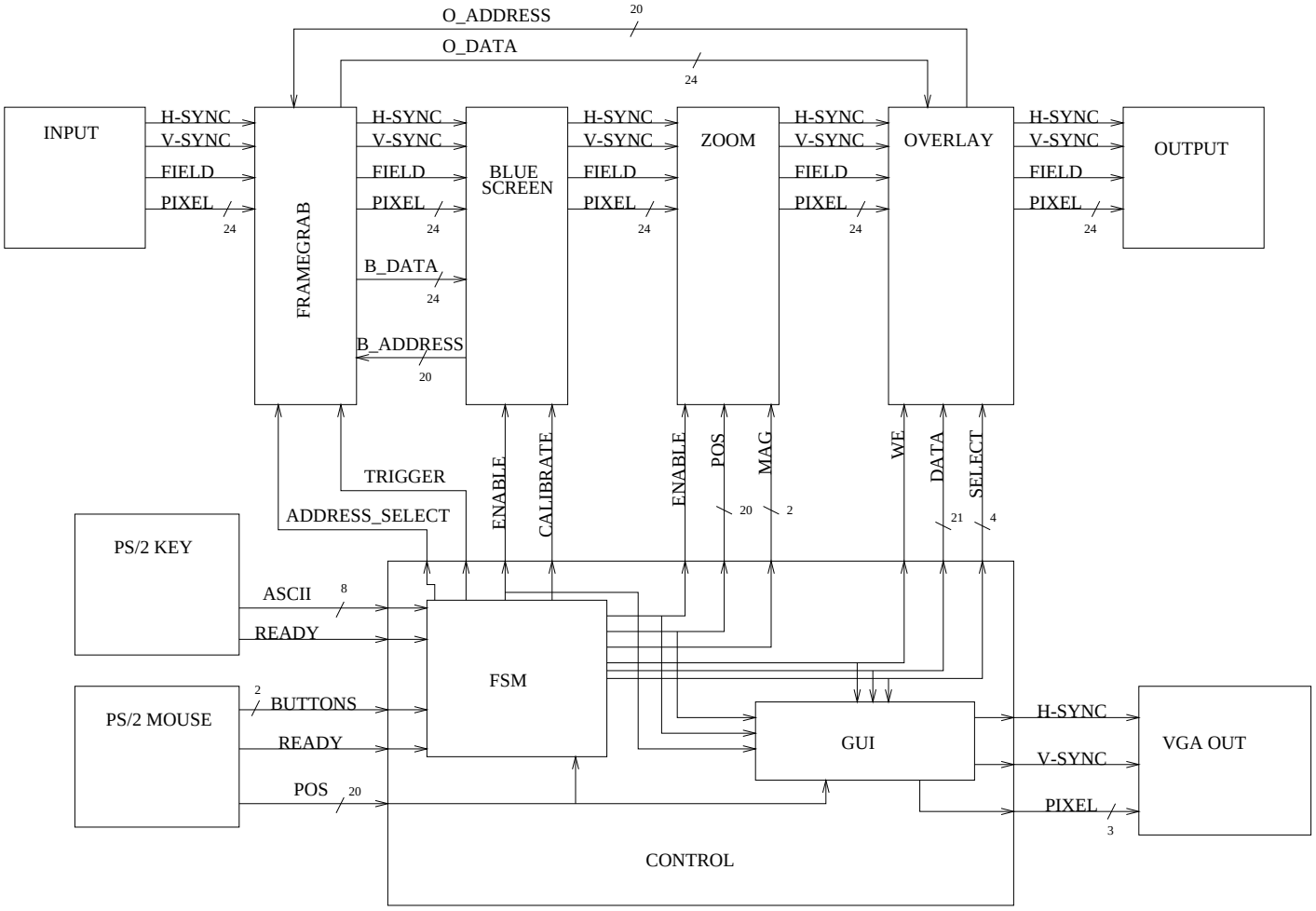

Figure 1: Video Processor Block Diagram

or the overlay module to read the buffer. This is not a huge limitation, since it does not really make sense to simultaneously superimpose the frame into the background and display a smaller version of the same frame in a single image.

Blue Screen Module

The blue screen module takes in the horizontal and vertical sync signals, the field, and the pixel value from the framegrab module. When enable is asserted, this module will replace any blue background pixels with the contents in the frame buffer.To calibrate the module so that we can determine what pixel values correspond to blue, the blue screen module takes in a calibrate input from the control module. When calibrate is asserted, the incoming video stream should just be the background. By examining this stream, the module should be able to determine appropriate thresholds to use in order to classify pixels as background or not.

Zoom Module

The zoom module takes in inputs from the control module and the blue screen module. The enable input indicates whether zoom should be used. The 20-bit position input gives the position on screen where we want to zoom in. The 2-bit magnification input tells us how much to enlarge the image around the point given by the position input. The zoom module produces horizontal and vertical sync output signals, a field signal, and a 24-bit pixel output. The zoom module interacts with a ZBT memory. This is because we need a one frame buffer to perform the necessary computation, and the on-chip BRAMs do not have enough memory to store a frame.

Overlay Module

After the zoom module comes the overlay module. This module is responsible for adding any overlay objects the user may have specified. In our design, any overlay objects will be rendered by the control module. The overlay module just needs to read the rendered overlay objects from BRAM modules instantiated within the overlay module.

The overlay module takes in the outputs of the zoom module, as well as some signals from the control module. Table 1 describes the interactions with the control module that program the overlay module's BRAMs and registers.

First, there is a write-enable input. If write-enable is asserted, the control module needs to write to the BRAM or some associated registers. In this case, a 21-bit data bus is used to tell the overlay module what needs to be written. The 4-bit select signal controls whether the BRAM or a register get written to. The high-order two bits of select are used to select between one of three possbile overlay objects: 2 text objects and the trace object. The low order 2 bits control which information is being written. If the low order bits are 00, we write to the BRAM associated with the right object. If the low order bits are 01, we write to the length register for the object. The length register for text objects stores how many characters of text are in this object. The trace does not need a length register. If the low order bits are 10, the overlay module writes to the position register, which stores the position where text overlay objects should appear on screen. Finally, if the low order bits are 11, the overlay module writes to the register, which indicates whether the object should actually be overlayed or not.

Table 1: Overlay Programming Specification

Programming Function	Select Bus Value	Data Bus Format
Text1 Video Buffer	0000	Bits 19:0 for address, bit 20 for value
Text1 Horizontal Length	0001	Bits 9:0 for value
Text1 Position	0010	Bits 19:0 for value
Text1 Enable	0011	Bit 20 for value
Text2 Video Buffer	0100	Bits 19:0 for address, bit 20 for value
Text2 Horizontal Length	0101	Bits 9:0 for value
Text2 Position	0110	Bits 19:0 for value
Text2 Enable	0111	Bit 20 for value
Frame Buffer Position	1000	Bits 19:0 for value
Frame Buffer Enable	1011	Bit 20 for value
Trace Video Buffer	1100	Bits 19:0 for address, bit 20 for value
Trace Enable	1111	Bit 20 for value

The overlay module also interacts with the framegrab module. By reading the ZBT frame buffer, the overlay module can superimpose a downsampled version of the stored frame onto the video stream. The frame can be superimposed at a variable position supplied by the user. This position is sent to the overlay module by the control module. The control module also uses the enable signal to tell the overlay module if the stored frame should be superimposed ot not. Table 1 details how this works.

Output Module

The overlay module outputs horizontal and vertical sync signals, a field signal, and a pixel output. This data is sent to the output module. The output module uses these signals to interface with the NTSC decoder on the 6.111 labkit. This produces the final video image with all modifications, which can be sent to a video monitor. We plan to use the code supplied with the 6.111 labkit for the output module, perhaps with minor modifications.

Control Module

The control module allows the video processor to interface with a user. It receives keyboard and mouse input from the PS/2 keyboard and mouse submodules and renders a graphical user interface which is output to a VGA display submodule via pixel data, horizontal and vertical sync lines. A concept drawing of the graphical user interface is seen in Figure 2.

The control module receives a ready signal from each of the PS/2 modules which indicates that a new piece of user input is ready to be processed. The control module will then read a new key press, mouse location, or mouse click. The combination of the keyboard/mouse input and VGA output allow the user to enter text to be stored for overlay, and then visually place the overlay text on top of the video using the mouse to select a position. This interface also allows the user to freehand draw on top of the video image using the trace functionality. Finally, the user interface will allow the user to control the zoom functionality by selecting a center point and magnification

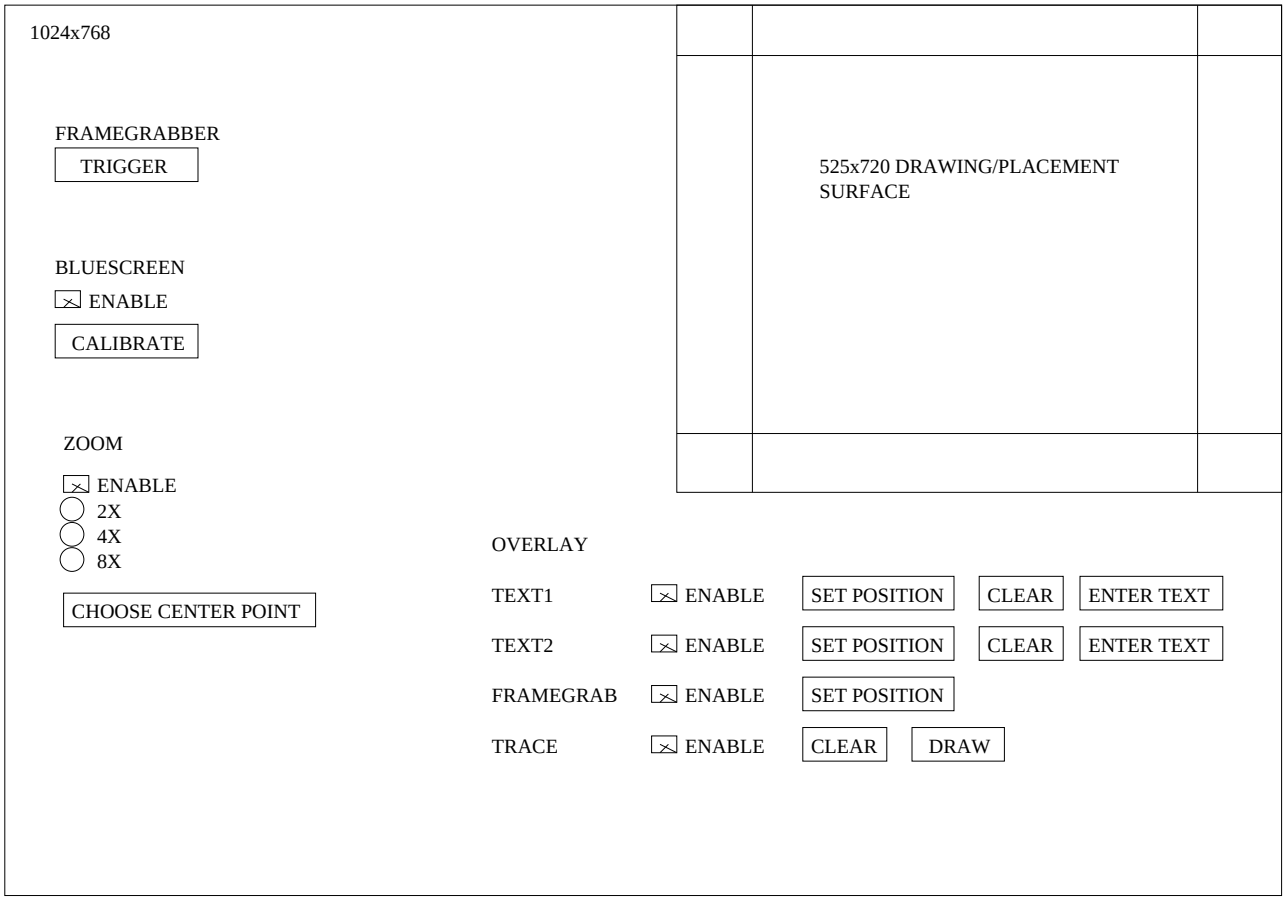

Figure 2: GUI Concept Drawing

level.

The user's interactions with the control module are translated to output lines that control the framegrab, blue screen, zoom and overlay submodules. This is accomplished through the use of a finite state machine sub-part within the control module. Three of these lines go to the zoom module an enable line which switches the zoom module in and out of the video path, a position output, which defines the center point of the zoom, and a magnification output which specifies the magnification level of the zoom. The other three outputs go to the overlay submodule. These outputs are used to write to various video memories that are in the overlay submodule. The write enable line signifies that a write should be performed on the next rising clock edge. The select output chooses what memory in the overlay module is being written, and the data bus carries the appropriate data and addressing necessary to carry out the type of write specified by the value of the select output. The use of these outputs is presented in detail in Table 1.

PS/2 Input Modules

The PS/2 keyboard module interfaces with the PS/2 hardware on the 6.111 labkit and has two outputs, an ASCII data output, which carries an 8-bit ASCII representation of a key press. The ready line goes high for one clock cycle after a keypress to indicate that there is a new characater. This module has been released as sample code for the labkit. We intend to use this sample module with minimal, if any, modification.

The PS/2 mouse module is similar to the keyboard module. It outputs the mouse's current position over a position output, and indicates a transition in the state of any of the buttons on the buttons output. An internal register keeps track of the mouse's position on a grid the size of the VGA display.

VGA Output Module

The VGA output module takes as input horizontal and vertical sync as well as a pixel value and interfaces with the Triple DAC hardware on the labkit. This output module is implemented in the lab4 code. We plan to use this implementation with minimal, if any, modification.

Testing

We plan to test the system in several stages. First, we will test out the input and output modules by themselves. When these modules work, we should be able to take in a video input and see it on the video monitor at the output. After verifying this basic performance, we can start testing the processing modules. The overlay module can be tested easily by hardcoding data into the BRAM and registers. The overlay module can be added between the input and output modules, and when the module is working properly we should see overlay objects on top of the video at the output.

While the overlay module is being tested, the PS/2 input modules can be tested separately. Using the logic analyzer, we should be able to check that these modules output the correct values. After the PS/2 input modules have been tested, part of the control module can be tested. Specifically, we will check that the control module can generate the correct VGA output for a computer monitor. Note that all this testing can be done in parallel with the overlay testing described above, because all the tests so far do not require any interaction between the control module and the processing modules.

The next step of testing will be to verify the operation of the zoom module. To start, we will check that the zoom module interface with the ZBT memory is working properly. Then, in the same way we tested the overlay module, we can test the zoom module with a hardcoded position and magnification factor. When the module is working properly, we should see a magnified version of the image on the output video monitor.

After the zoom module has been tested, the framebgrab and blue screen modules will be tested. Again, we can hardcode the control signals and check that these modules work. We can shift out the zoom and overlay modules for initial testing of the blue screen module, and then add the overlay module back in to test the picture-in-picture functionality.

The last tests we need to perform are to verify that the control module produces the correct programming signals for the zoom and overlay modules. We should be able to check this easily using the Xilinx ISE simulator or the logic analyzer. Then, we can connect all the modules together, and the entire system should work properly.

Division of Labor

The division of labor reflects the testing procedure described above. Ben will work on the PS/2 input modules, the control module, the framegrab module, and the VGA output module. Venkat will work on the blue screen, zoom, overlay, and output modules.