ECE532 Group 10 Project Report:
The Page Turner
9 April 2012

Neil Isaac
William (Ziran) Jiang
James Robinson
Project Overview

Summary
We built a gesture detection system which can detect abrupt changes in the number of edges detected in a video signal. This will be used to detect rapid user movement, which will be used to infer a gesture. We used the detected gesture as a signal to turn one page forwards or backwards in a book being displayed on a monitor.

Motivation
There are situations where it is useful to be able to interact with a computer system without physically touching an input device. Examples include outdoor advertising and interactive gaming. This project explores whether the combination of edge detection and a simple software algorithm can be used to accurately interpret user movements, and provides a simple output scenario (reading a book) to provide visual verification of its results.

Goals
- Detect edges from video input
- Use detected edges to infer user motion
- Allow pages of a book to be displayed on a computer screen on command
- Run in real-time
Data Flow

Data Flow Steps:
1. Capture video input using a laptop webcam.
2. The video input is displayed on the laptop screen, which is replicated to its VGA output port.
3. The VGA output of the laptop is connected to the VGA input on the XUPV5 development board and is decoded by the Analog Devices AD9980 decoder chip.
4. Our custom edge detection hardware detects and filters edges in the video and accumulates them by frame.
5. The accumulated values are read by the MicroBlaze processor over the main PLB bus.
6. The processor detects variation in the accumulated values to distinguish gestures.
7. When a gesture is detected, it changes the current book page, and re-renders it in the framebuffer.
8. The framebuffer is read by the TFT controller, which sets the output signal driving the Chrontel CH7301C DVI output encoder.
9. The DVI encoder is connected to an external monitor using a DVI to VGA adapter.
**Brief description of IP Blocks**

**MicroBlaze (7.10.d from Xilinx EDK):** Microprocessor used for gesture detection algorithm, editing the framebuffer, initializing the video chips, and handling debug output.

**MPMC (4.03.a from Xilinx EDK):** Memory controller used to connect a 256MB DRAM to the main PLB bus. DRAM memory is used for the output framebuffer accessed by the display controller. Only one port is used.

**XPS_TFT (1.00.a from Xilinx EDK):** DVI Display output controller. Reads from framebuffer in DRAM memory to display the current page of the book.
**XPS IIC (2.00.a from Xilinx EDK):** Used to initialize the parameters for the video decoder and decoder chips.

**Instruction BRAM (2.10.a from Xilinx EDK):** Contains the software.

**Data BRAM (2.10.a from Xilinx EDK):** Used for:
- Software data memory
- Static font data array
- Static book text array

**UARTLite (1.00.a from Xilinx EDK):** Uart used for debugging output. We output a stream left/right accumulated weighting factors which can be dumped to a CSV file to plot in Excel for analytical algorithm design.

**CH7301C (Chrontel):** DVI encoder used for video output to display the book pages. Connected to a VGA monitor using a DVI to VGA adapter using the DVI analog signal.

**AD9980 (Analog Devices):** VGA decoder used to capture video. Video input is from the VGA output of a laptop displaying its webcam video feed.

**Edge Detection Hardware (Custom):** Reads the video signal and outputs edge amplitudes for each sequential pair of pixels.

**Edge Weighting Counters (Custom):** Reads edges from edge detection hardware and sorts the edges by location. Presents accumulated values per frame via PLB bus registers.

## Outcome

Our project was successful in meeting its goals, and is feature-complete: edges can be detected from video input, those edges can be used to infer user motion, and that user input can be used to flip pages of a book, all in real-time. Below is a table comparing the original features with their implementation in the final design:

<table>
<thead>
<tr>
<th>Original Feature</th>
<th>Final Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect edges in real-time from a video feed</td>
<td>Achieved using a hardware module.</td>
</tr>
<tr>
<td>Detect changes in locations of edges to infer a gesture</td>
<td>Edges were grouped according to their position on the screen and summed in hardware. A software algorithm used that</td>
</tr>
</tbody>
</table>
data to infer a gesture, with high accuracy in controlled conditions, albeit diminished accuracy in uncontrolled ones.

<table>
<thead>
<tr>
<th>Display a page of a book on a monitor</th>
<th>Achieved in software using freely available literature and fonts stored uncompressed as static code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch to next or previous page in response to an input gesture</td>
<td>Achieved in software, albeit without fancy transition effects, which proved to be too slow to implement in real-time.</td>
</tr>
<tr>
<td>If necessary, filter using a static background frame</td>
<td>Used in an initial implementation, but eventually scrapped, since the background never remained static during actual tests.</td>
</tr>
</tbody>
</table>

It is also useful to compare our final design against our original acceptance criteria.

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
<th>Final Implementation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognize a user’s gesture reliably (90% of the time or more)</td>
<td>This reliability is only achieved under controlled conditions. If variables such as lighting change, or there is additional user motion (such as moving their body while doing a hand gesture) reliability drops dramatically.</td>
</tr>
<tr>
<td>Respond quickly (within ½ a second)</td>
<td>Achieved - if the gesture is recognized, the time between gesture action and response is below the limits of human perception.</td>
</tr>
</tbody>
</table>

As can be seen from the above tables, the project was a success, albeit requiring controlled conditions for reliable gesture recognition. The project would be a greater success if it worked in a larger range of conditions, but in its current state, it does meet our original design goals.

If we could do it all over again, we would make the following changes to our design and methodology:

- Establish what our input data will look like at the very beginning, and not vary it.
- Write the motion detection algorithm based on the actual input data, not a guess as to what the input data will look like.
- Use an already-written video codec input module as a starting point, rather than reinventing the wheel.

**Suggested Improvements and Next Steps**
- Improve gesture recognition by fine-tuning the algorithm (ex. using additional detection regions, filtering “background” edges that don’t move.)
- Allow multiple books to be read.
- Allow books to be stored on CompactFlash rather than compiled into the code.
- Allow on-the-fly decompression of compressed book files.
- Implement edge clustering, so that similar edges can be grouped together and “lines” of edges (i.e. from the end of a hand or a head) can be detected.

**Project Schedule**

The project was initially divided into the following modules, each worked on independently by a separate group member:

<table>
<thead>
<tr>
<th>Module</th>
<th>Group Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware edge detection and input stream</td>
<td>Neil Isaac</td>
</tr>
<tr>
<td>Software motion detection</td>
<td>James Robinson</td>
</tr>
<tr>
<td>Output stream</td>
<td>William Jiang</td>
</tr>
</tbody>
</table>

After each module had attained a certain level of development, they were combined to form the complete project. Thus, the milestones were to first be divided along module lines, and then united when we started integrating the different parts.

The table below shows both our original milestones and the actual milestones achieved each week. Our progress generally followed the plan, although as expected, sometimes group members completed their tasks faster or slower than expected. As a result, some weeks’ milestones were adjusted so that the group member with faster progress could take some of the workload off of the member with slower progress.

We anticipated there will be a lot of debugging during the integration phase, so we allocated four weeks for integration. This proved to be helpful, since during the integration phase, a significant amount of tweaking was needed to optimize the motion detection. Overall, the original milestones were well planned, and the actual project progress ended up following them closely.
<table>
<thead>
<tr>
<th>Date</th>
<th>Group Member</th>
<th>Original Milestone</th>
<th>Modified Milestone</th>
<th>Actual Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 8</td>
<td>James</td>
<td>Parse edge location signals in software from a pre-recorded video</td>
<td>Same as original</td>
<td>Milestone complete</td>
</tr>
<tr>
<td></td>
<td>Neil</td>
<td>Receive video input in hardware (on FPGA)</td>
<td>Same as original</td>
<td>Milestone complete</td>
</tr>
<tr>
<td></td>
<td>William</td>
<td>Test DAC using the built in tests on board, and add TFT controller IP in XPS</td>
<td>Same as original</td>
<td>Milestone complete</td>
</tr>
<tr>
<td>Feb 15</td>
<td>James</td>
<td>Have a software implementation of Milestone 1 to define the expected interface</td>
<td>Same as original</td>
<td>Milestone complete</td>
</tr>
<tr>
<td></td>
<td>Neil</td>
<td>Detect horizontal edges and display on VGA monitor</td>
<td>Interface with video decoder chip, Detect horizontal edges</td>
<td>Milestone incomplete, but progress made</td>
</tr>
<tr>
<td></td>
<td>William</td>
<td>Use VGA driver to display a bitmap on VGA monitor (using CPU)</td>
<td>Same as original</td>
<td>Milestone complete, as well as future milestones</td>
</tr>
</tbody>
</table>

- Written in pure C (so could be put on MicroBlaze)  
- Uses mmap() to implement simulated in-memory register interface  
- Can compute edges using absolute difference between adjacent pixels.  
- Communicating with video decoder via iic microblaze core.  
- Having difficulty understanding decoder chip setup parameters.  
- Can generate bitmap in C header file format, using XBM files.  
- In MicroBlaze, can convert a character to bitmap using the
<table>
<thead>
<tr>
<th>Date</th>
<th>Group Member</th>
<th>Original Milestone</th>
<th>Modified Milestone</th>
<th>Actual Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 22</td>
<td>Reading Week</td>
<td>C header file, and display the character on the monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 29</td>
<td>James</td>
<td>Use the software implementation of Milestone 1 to drive motion detection</td>
<td>Same as original</td>
<td><strong>Milestone complete</strong>&lt;br&gt;- Successfully detects directional motion on test input videos</td>
</tr>
<tr>
<td></td>
<td>Neil</td>
<td>Implement edge clustering</td>
<td>Properly interface with video decoder</td>
<td><strong>Milestone incomplete, but progress made</strong>&lt;br&gt;- Can set decoder parameters via microblaze i2c core.&lt;br&gt;- Can detect resolution by counting clocks per hsync and hsyncs per vsync. I detect 1023x624 for 800x600 input; the spec sheet suggests 20-30% padding around active pixels.&lt;br&gt;- Currently trying to make sense of the RGB output from the decoder by relaying it over a serial port.</td>
</tr>
<tr>
<td>Mar-07</td>
<td>Group</td>
<td>n/a</td>
<td>Agree on mutually acceptable and implementable interfaces between components</td>
<td><strong>Milestone complete</strong>&lt;br&gt;- New hardware-&gt;software interface defined, with separate stages depending on the maturity of the hardware implementation&lt;br&gt;- Interface between motion recognition component and book reading component defined as a couple of function calls (load_page(N)).</td>
</tr>
</tbody>
</table>

William Render characters using a bitmap font Display one page of of a book on the monitor. **Milestone complete**
<table>
<thead>
<tr>
<th>Date</th>
<th>Group Member</th>
<th>Original Milestone</th>
<th>Modified Milestone</th>
<th>Actual Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>James</td>
<td>Fine-tune motion detection accuracy</td>
<td>Implement the new hardware to software interface, as well as start writing a Verilog implementation</td>
<td><strong>Milestone complete</strong> - Software interface complete (reading from memory-mapped registers) - Verilog implementation has not been debugged</td>
</tr>
<tr>
<td></td>
<td>Neil</td>
<td>Output list of clustered edges with bus component</td>
<td>Find most important edges in each row.</td>
<td><strong>Milestone complete</strong></td>
</tr>
<tr>
<td></td>
<td>William</td>
<td>Display one page of a book on VGA monitor by loading a text file</td>
<td>Implement functions to allow external control of book page flipping</td>
<td><strong>Milestone incomplete</strong> - Agreed on the interface of book page flipping, function not complete due to getting sick and not having time to complete it.</td>
</tr>
<tr>
<td>Mar 14</td>
<td>James</td>
<td>Make motion detection run on a processor on the XUP board</td>
<td>Debug the hardware interface for the motion detector, attaching it as a PLB slave to the microblaze processor, and attach James’ and Williams’ software parts together and make sure they work in hardware.</td>
<td><strong>Milestone complete</strong> - Simulation of Neil’s hardware (not XUP board) completed in Modelsim using EDK. - James’ and Williams’ software portions merged on XUP board.</td>
</tr>
<tr>
<td></td>
<td>Neil</td>
<td>Tune/improve clustering algorithm</td>
<td>Build hardware interface for the motion detector, and make sure it works on Neil’s hardware.</td>
<td><strong>Milestone complete</strong> - Video input attached to PLB bus - Software test of video input functionality and interface run on board</td>
</tr>
<tr>
<td></td>
<td>William</td>
<td>Make the display change (flip page forward and backward) using buttons</td>
<td>Finish implementation of book page flipping interface</td>
<td><strong>Milestone complete</strong></td>
</tr>
<tr>
<td>Date</td>
<td>Group/Member</td>
<td>Original Milestone</td>
<td>Modified Milestone</td>
<td>Actual Progress</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Mar 21 | Group        | Finalize inter-module interface specifications                                     | Separate milestones for each group member                                          | Milestone complete  
- James and William's software portions work on the XUP board, and merge James, William, and Neil's software portions for implementation on Neil's board.  
- Framework created for integration of Neil's software, compiles successfully, not tested yet. |
|        | James        | n/a                                                                                 | Ensure James and William's software portions work on the XUP board, and merge James, William, and Neil's software portions for implementation on Neil's board. | Milestone complete  
- Can use custom pcore to detect edges from VGA input video signal  
- Can access “interesting” edge data over PLB interface from MicroBlaze  
- Can display edges on DVI video output (colour mapped by magnitude)  
- Integrated basic book viewer code with manual page flipping buttons (not yet connected to edge detection; just an alternative output display mode) |
|        | Neil         | n/a                                                                                 | Get video input working in hardware on Neil's board.                               | Milestone complete  
- Can use custom pcore to detect edges from VGA input video signal  
- Can access “interesting” edge data over PLB interface from MicroBlaze  
- Can display edges on DVI video output (colour mapped by magnitude)  
- Integrated basic book viewer code with manual page flipping buttons (not yet connected to edge detection; just an alternative output display mode) |
|        | William      | n/a                                                                                 | Simulate Neil's hardware video_in module.                                          | Milestone incomplete, but progress made  
Simulation in progress with various test benches |
| Mar 28 | Group        | Debugging inter-module interfaces                                                   | Separate milestones for each group member                                          | Milestone complete  
- All code is now running on Neil’s board, including video |
|        | James and Neil | n/a                                                                             | - Ensure the full integrated software system                                      | Milestone complete  
- All code is now running on Neil’s board, including video |
<table>
<thead>
<tr>
<th>Date</th>
<th>Group Member</th>
<th>Original Milestone</th>
<th>Modified Milestone</th>
<th>Actual Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>works properly on Neil's board - Tune the motion detection algorithm to be more accurate</td>
<td>input, edge detection, motion detection, and page display</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Work In Progress</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Motion detection algorithm has been tuned to resync and more tuning variables have been added - Motion detection still needs more work to be completely reliable</td>
</tr>
<tr>
<td>Apr 4</td>
<td>Group</td>
<td>Debugging and tweaking parameters to optimize responsiveness</td>
<td>Separate milestones for each group member</td>
<td>Milestone complete - Test benches created and run for video_in module</td>
</tr>
<tr>
<td>Apr 11</td>
<td>Final</td>
<td></td>
<td></td>
<td>Final Demo</td>
</tr>
</tbody>
</table>
Description of the IP Blocks

Edge Detection Hardware (hardware)

Inputs
- YCbCr video data from the VGA video decoder chip (Analog Devices AD9980) on the Virtex 5 development board.

Outputs
- Filtered edge amplitudes, computed as the absolute difference between adjacent pixels within the active video. Output is in the pcore’s clock domain, which is the same as the PLB bus clock domain.

Function
Processes the video to detect and output edge amplitudes. Makes output data accessible in the main 100Mhz clock domain used by the bus and the rest of the pcore. Filtering is done by outputting only the largest edge amplitude out of every set of 8 sequential pixels, which reduces the effect of a strong edge oscillating left or right slightly. The edge amplitudes represent the edges of features of interest in the video.

Edge Weighting Counters (hardware)

Inputs
- Continuous edge stream for each pixel
- End of frame signal

Outputs
- New data available signal (exposed in “status” register)
- Amplitude-weighted sum of edges in the left and right halves of the previous frame (exposed in the upper and lower half-words of the “data” register)

Function
For each frame, it accumulates the amplitudes of the edges on the left and right sides. When a frame is done, the “status” register indicates this until a PLB read occurs. The accumulated values are buffered and made accessible from our “data” PLB register. The microblaze has 1/60 of a second to read the “data” register before the next frame replaces its value. Since the edges represent features of interest, the accumulated amplitudes for the left and right sides represent the amount of features on each side of the frame. Fluctuation in these values from frame to frame represents features being added, removed, or moving from one side to the other.

Motion Detection (software)

Inputs
Horizontal edges of the video image, post-processed with a clustering algorithm

**Outputs**
- Turn page left signal
- Turn page right signal

**Function**
Determine, based on the edge data, whether or not a user has made a gesture indicating that the page should be turned, and if so, send the appropriate signal. This is done by smoothing the data using an averaging function, using peak detection to find points of interest, and then looking for specific patterns within a sliding window.

**Page Display (software)**

**Inputs**
- Turn page left function call
- Turn page right function call
- Bitmapped fonts (from data memory)
- Book text (from data memory)

**Outputs**
- Writes to framebuffer in DRAM

**Function**
Print book page text on the screen (using a framebuffer in DRAM) by looking up and displaying the appropriate glyphs from a font. Respond to page turn signals by changing what text is displayed.

**Description of the Design Tree**

Gzip tar file location: [http://neilisaac.com/ece532project.tgz](http://neilisaac.com/ece532project.tgz)

This gzipped tar file will be updated with the final versions of all files after the demo on Wednesday.

- **README**: Documentation of files within design directory
- **presentation.pdf**: Presentation for demo
- **docs/:** Chipset and board documentation
- **motiondetector/:** Motion detection algorithm test code
  - **detect_motion.c**: Motion detection algorithms
  - **hwint.c**: Hardware-independent interface for input data
  - **disk_interface.c**: Bootstrapping code and mmio testing interface
  - **left_to_right.h, right_to_left.h, noise.h**: Sample input data from the lab
  - **triplets_handwavefast.txt**: Sample input data from a webcam
system: SOC system and code
  ○ code: Folder containing code for microblaze
  ○ data/system.ucf: Pins and constraints for development board
  ○ etc: Configuration files generated by EDK
  ○ pcores/video_in_v1.00.al: Custom hardware core
    ■ data: Files for core and core port assignments
    ■ hdl: HDL code for the core
  ● verilog: Main functional Verilog code for the core
    ○ testbench.v: Tests for the video input and edge detection HDL
    ○ uart.v: Hardware UART that can be connected to a UARTLite to debug hardware directly
    ○ user_logic.v: Connects “video.v” module to “video_in.vhd” module
    ○ vga.v: Calculates edges in VGA signal from decoder
    ○ video.v: Reads edges in each frame, sets and handles reads to the PLB registers
  ● vhdl/video_in.vhd: Top-level VHDL code that handles the PLB interface

Tips and Tricks

- Work with real data as soon as possible. It’s hard to make the right net if you don’t know what you’re trying to catch.
- Anything that can be done in software, do in software. Don’t assume the MicroBlaze will be too slow.
- Just because your code works properly on your desktop doesn’t mean it will work properly on the MicroBlaze.
- The KVM switches in the lab can mess up your board. If something’s not working as expected, try disconnecting the output to the switch.
- If you can possibly help it, don’t write your own module to interact with the video codec - use someone else’s that is already working.