Facial Recognition General-Purpose Access Control Device

Alex Newcomb
newcomb@ualberta.ca

Tom Stefanyk
tom.stefanyk@ualberta.ca

Preferred Lab Day: Wednesday
Other lab days: Tuesdays and Thursdays after 3pm

Summary: Detecting and recognizing a face from input video stream and displaying on a web server.

Table of Contents

Abstract

The facial recognition device is designed with automated access control in mind. The device uses a camera to take in a video stream, of which still images will be periodically stored. Once algorithms detect the presence of a face within such a still image, key measurements will be taken of the facial features and compared to a database of known faces. The image will be converted to a simple image format and made available via an on-board web server hosting a simple website, which will display the image along with the name of the person (if recognized). Additionally, the website will contain a list of actions the person can take .

Functional Requirements

Original Requirements:

The device must take video input from a video camera or similar device interfaced via the composite video input. The image is to be saved both in a compressed JPEG color format and as a simple greyscale image. Said device must periodically check the saved image (the greyscale version) for the presence of a face. The device must also provide an web-based interface via wired Ethernet, which will only display images once the device has detected a face. Said interface must display identifying the face as either a specific known person or as an unknown person; the interface must also provide several actions for the user to choose in response.

Performance

The finished device met most requirements, but some trade-offs were needed. Due to the size and requirements of the facial recognition algorithm, this functionality was moved from the device to a computer on the network, which we call our "facial recognition server". The device serves images from the camera to the facial recognition server, which sends the facial measurements of any recognized faces.

Also, the JPEG compression functionality was cut due to time constraints, which slows the web-site loading considerably.

Design and Description

The device is on a network with a facial recognition server and a client computer. It connects to this network via the on-board Ethernet device. In our example, the facial recognition and client computers are the same. The device serves a dynamic (i.e. created on the fly) web-page to the client computer, and keeps track of whether the web-page is being viewed. The web-page has an auto-refresh function implemented with a simple javascript on-load function.

On request from the facial recognition server (via HTTP GET request), the device takes video input from a camcorder or other such video device via the composite in input. The onboard composite video reads in the video in YCbCr format. The image data is converted to 16-bit RGB format. The image data is placed in SRAM as a 320x240 resolution image. The device translates the image data to Bitmap image format and sends this image to the facial recognition server. The server sweeps the image, using patterns of pixels that correspond to specific facial features (eyes, nose and mouth). Should the server find all of the aforementioned features, the facial recognition server sends the device a signal that a face is present in the current image, along with the measurements of the distance between the facial features. These measurements are all normalized by the distance between the eyes, so a face's measurements should be close to the same regardless of how far away the face is from the camera (provided the server can still recognize the face). If the server is unable to find all the points, it will request an additional image from the device and repeat the sweep.

The device compares the facial measurements against a list of known faces to try and identify the person in the image. If a face is contained in the current image, the next time the client computer requests the web-page, the page will display the image, along with the name of the person (if identified), and an HTML form containing a list of actions the user can take. Examples of actions are “Unlock the door”, “Send a message” or “Release the hounds”. In our case, these choices result in different board actions, such as a sweep of the on-board LEDs, display of the message on an LED display or changing of the on-board 7-segment display.

If the person in the image is not identified, the web-page also prompts the user to enter the name of the person, who’s name and facial measurements are then stored in the list of known faces on the device.

To ensure that the image on the screen is the one in which the face is recognized, the facial recognitions stops requesting images once a face has been identified, and polls the device to see if it is ready for a new image. If the web-page is refreshed, the device will respond that it is ready for a new image.

Parts List

Terasic DE2 development board
Website: http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html
User Manual: ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

JVC Digital Video Camera GR-DVL510U ($0, already owned)
Website: http://support.jvc.com/consumer/product.jsp?archive=true&pathId=26&modelId=MODL026470
User Manual: http://resources.jvc.com/Resources/00/00/98/LYT0722-001A.pdf
Outputs: Composite video

Acer Aspire 5738 Laptop ($0, already owned)
Specification: http://support.acer.com/acerpanam/notebook/2009/acer/aspire/Aspire5738/Aspire5738sp2.shtml
Input/Output: Ethernet crossover cable

I/O Signals

format: signal name: signal description: signal size

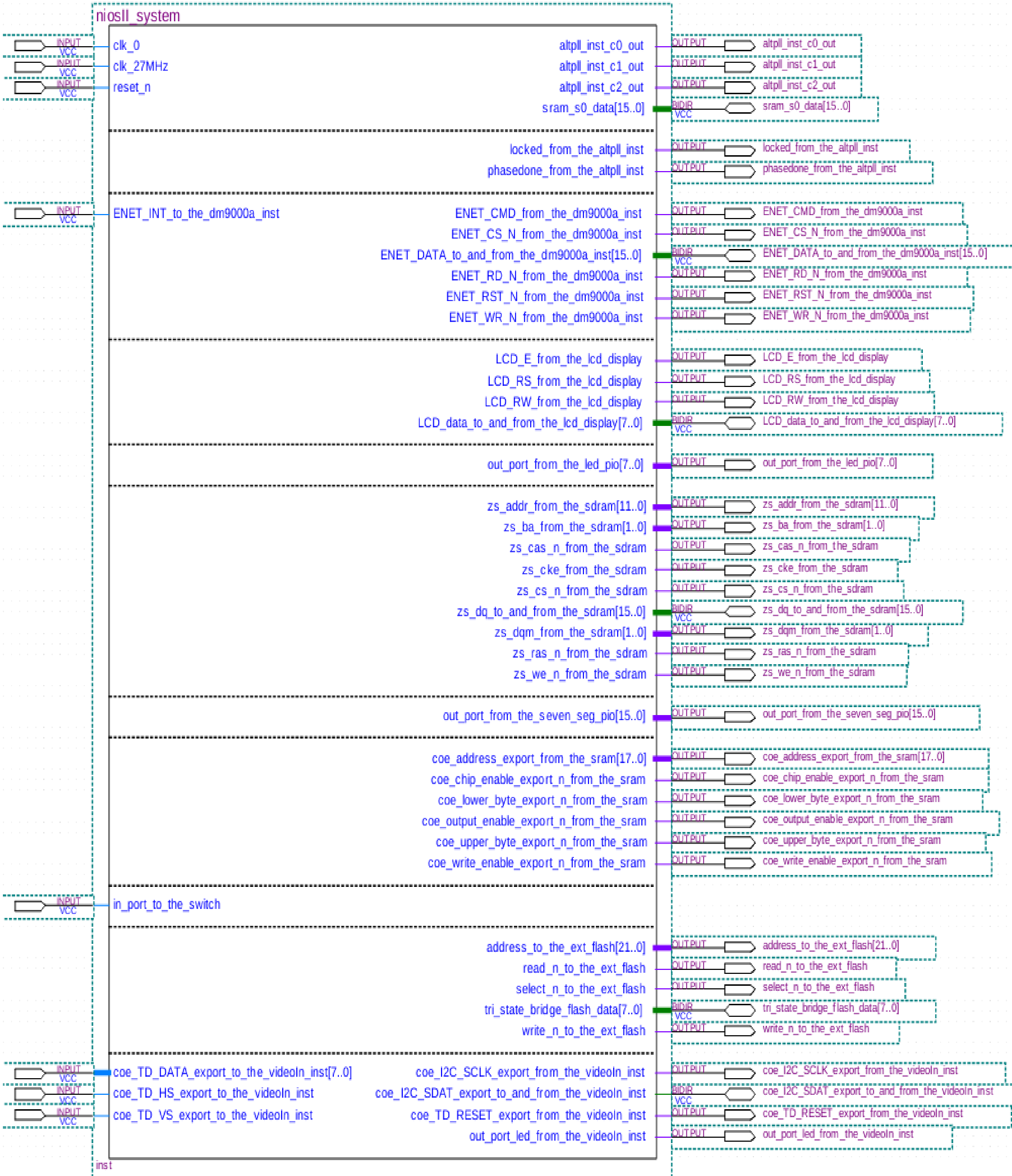
1. Microprocessor to custom HDL
    - Control signals to video in logic
        - avm_vidIn_address: Avalon Memory Master address: 26 wires
        - avm_vidIn_write: Avalon Memory Master write enable:1 wire
        - avm_vidIn writedata: Avalon Memory Master data line: 16 wires
        - avm_vidIn_waitrequest: Avalon Memory Master wait request: 1 wire
        - avm_vidIn_byteenable:Avalon Memory Master byte enable: 2 wires
        - avm_vidIn_arbiterlock: Avalon Memory Master arbiter lock: 1 wire
        - avs_control_address: Avalon Memory Slave address: 4 wires
        - avs_control_write: Avalon Memory Slave write enable: 1 wire
        - avs_control_writedata: Avalon Memory Slave data line: 8 wires
        - avs_control_byteenable: Avalon Memory Slave byte enable: 1 wire

2. FPGA to Board

- Data and control lines to/from TV encoder
    - coe_TV_DATA : data line: 8 wires
    - coe_TD_HS: Horizontal sync: 1 wire
    - coe_TD_VS: Vertical sync: 1 wire
    - coe_TD_RESET: reset: 1 wire
- Control lines to/from I2C unit
    - coe_I2C_SCLK: I2C clock, 27Mhz: 1 wire
    - coe_I2C_SDAT: data line: 1 wire
- Data and control lines to CFI Flash
    - address_to the ext_flash: address lines: 22 wires
    - tri_state_bridge_flash_data: data lines: 8 wires
    - read_n_to_the_ext_flash: read enable, active low: 1 wire
    - select_n_to_the_ext_flash: chip enable, active low: 1 wire
    - write_n_to_the_ext_flash:write enable, active low: 1 wire
- Reset switch
    - in_port_to_the_switch: reset: 1 wire
- Data and control lines to SRAM
    - coe_address_export_from_the_sram: address lines: 18 wires
    - sram_s0_data: data lines: 16 wires
    - coe_chip_enable_export_n_from_the_sram: chip enable,active low: 1 wire
    - coe_lower_byte_export_n_from_the_sram: lower byte enable, active low: 1 wire
    - coe_upper_byte_export_n_from_the_sram: upper byte enable, active low: 1 wire
    - coe_output_enable_eport_n_from_the_sram: read enable, active low: 1 wire
    - coe_write_enable_export_n_from_the_sram: write enable, active low: 1 wire
- Seven segment control lines
    - out_port_from_the_seven_seg_pio: control lines: 16 wires
- Data and control lines to/from SDRAM
    - zs_addr_from_the_sdram: address lines: 12 wires
    - zs_ba_from_The_sdram: byte enable lines: 2 wires
    - zs_cas_n_from_the_sdram: Column Address Strobe, active low: 1 wire
    - zs_cke_from_The_sdram: clock enable: 1 wire
    - zs_cs_n_from_the_sdram: chip select, active low: 1 wire
    - zs_dq_t_and_from_the_sdram: data lines: 16 wires
    - zs_ras_n_From_the_sdram: row address strobe, active low: 1 wire
    - zs_we_n_from_the_sdram: write enable, active low: 1 wire
- LED control
    - out_port_from_the_led_pio: led control line: 1 wire
- LCD Display control lines
    - LCD_E_from_the_lcd_display: enable line: 1 wire
    - LCD_RS_from the lcd_display: row select: 1 wire
    - LCD_RW_from the lcd_dislay: read/write select: 1 wire
    - LCD_data_to_and_from_the_lcd_display: data lines: 8 wires
- Data and control lines to/from Ethernet controller
    - ENET_CMD_from_the_dm9000a_inst: command line: 1 wire

        - ENET_CS_N_from_the_dm9000a_inst: chip select, active low: 1 wire
        - ENET_DATA_from_the_dm9000a_inst: data lines: 16 wires
        - ENET_INT_from_the_dm9000a_inst: interrupt line: 1 wire
        - ENET_RD_N_from_the_dm9000a_inst: read enable, active low: 1 wire
        - ENET_RST_N_from_the_dm9000a_inst:reset, active low: 1 wire
        - ENET_WR_N_from_the_dm9000a_inst: write enable, active low: 1 wire
    - Clock source
        - clk_0: 50 Mhz clock: 1 wire
        - clk_27MHz: 27 Mhz clock: 1 wire
3. Board to Off-board Peripherals
    - Ethernet cable between Ethernet controller and other computer
    - Composite video input from camera
4. Off-board Peripherals to World
    - Power adapter to Altera DE2 board: 9V @ 1.3A
    - Power adapter for camera: 11V @ 1A

Datasheet


niosII_system
clk_0
clk_27MHz
reset_n
altpll_inst_c0_out
altpll_inst_c1_out
altpll_inst_c2_out
sram_s0_data[15..0]
locked_from_the_altpll_inst
phasedone_from_the_altpll_inst
ENET_INT_to_the_dm9000a_inst
ENET_CMD_from_the_dm9000a_inst
ENET_CS_N_from_the_dm9000a_inst
ENET_DATA_to_and_from_the_dm9000a_inst[15..0]
ENET_RD_N_from_the_dm9000a_inst
ENET_RST_N_from_the_dm9000a_inst
ENET_WR_N_from_the_dm9000a_inst
LCD_E_from_the_lcd_display
LCD_RS_from_the_lcd_display
LCD_RW_from_the_lcd_display
LCD_data_to_and_from_the_lcd_display[7..0]
out_port_from_the_led_pio[7..0]
zs_addr_from_the_sdram[11..0]
zs_ba_from_the_sdram[1..0]
zs_cas_n_from_the_sdram
zs_cke_from_the_sdram
zs_cs_n_from_the_sdram
zs_dq_to_and_from_the_sdram[15..0]
zs_dqm_from_the_sdram[1..0]
zs_ras_n_from_the_sdram
zs_we_n_from_the_sdram
out_port_from_the_seven_seg_pio[15..0]
coe_address_export_from_the_sram[17..0]
coe_chip_enable_export_n_from_the_sram
coe_lower_byte_export_n_from_the_sram
coe_output_enable_export_n_from_the_sram
coe_upper_byte_export_n_from_the_sram
coe_write_enable_export_n_from_the_sram
in_port_to_the_switch
address_to_the_ext_flash[21..0]
read_n_to_the_ext_flash
select_n_to_the_ext_flash
tri_state_bridge_flash_data[7..0]
write_n_to_the_ext_flash
coe_TD_DATA_export_to_the_videoIn_inst[7..0]
coe_TD_HS_export_to_the_videoIn_inst
coe_TD_VS_export_to_the_videoIn_inst
coe_I2C_SCLK_export_from_the_videoIn_inst
coe_I2C_SDAT_export_to_and_from_the_videoIn_inst
coe_TD_RESET_export_from_the_videoIn_inst
out_port_led_from_the_videoIn_inst
inst
INPUT
VCC
OUTPUT
BIDIR

Software Design

The interface of the system is a web-page on any computer on the system that regularly polls the web-server on the system for updates using client-side javascript. An update will consist of an image of an identified face and the name of an identified person. The web-page will also have a form that can be submitted to activate various security features in response to an identified face (which will likely be represented by LED responses on the DE2 board), or to request a new image. If an update is received, the web-page will stop polling the web-server until the user submits an action.

On the server side, the system will convert the signal from the on-board composite video into 16-bit RGB image data. This image data is served up by the web-server as a bitmap image. Once an image is created it will be analyzed for signatures that identify locations of facial features by a facial recognition program running on a computer on the same network as the web-server. If such signatures are found, and a face is recognized, the relative facial measurements (relative to the distance between the eyes, E) will be sent to the web-server where they will be compared against a list of known faces. If the relative distances of the facial features of the captured face are close enough to that of an entry in the list, the system will identify the face as belonging to a known person. In this case, close enough means each measurement is within 0.2E of the known face. If the facial features do not match those in the database, the system will note that the face belongs to an unknown person.
Regardless of whether the person is identified, if a face is recognized an update will be prepared for the web-server consisting of a picture of the identified face and the name of the identified person (if applicable). If a face is not recognized in the image taken from the composite input, the image will be discarded and the system will fetch a new image.

Processes:
Web-Server: This will be based off of the example provided by Altera and included with the Nios II IDE. It monitors for GET and POST requests from clients. If it receives a GET request for an image “find_face.bmp”, it will trigger the system to retrieve an image from the video input stream, save it on SRAM and serve the found image as a Bitmap file. If it receives a GET request for “found_face.bmp”, it will serve up the image data that is already on the SRAM as a Bitmap image. If it receives a GET request for “facefinder.html”, it will serve up a dynamically generated html page. If no face has been found, this page will be an automatically refreshing simple page with no information. If a face has been found and the person has been identified, the page will display the image, the name of the identified person, and an html form with a list of actions to take in response (unlock door, release hounds, etc). If a face has been found but not identified, the page will include a form where a name can be entered by the user, which will be saved with the measurements in the list of known faces.

The web-server can also respond to a number of POST requests.

If it receives a STATUS_REQUEST request, it will return NOT_READY if the dynamic page is either not being viewed, or is currently showing a face and waiting for the user to refresh the page. If the page is being viewed but is in automatic refresh mode (i.e. no face has been found) it will return READY_FOR_FACE.

If it receives a FACE_FOUND.. request, where … hold the facial measurements, it will extract those measurements and signal that a face has been found. It will then attempt to identify the person in the image by the measurements, and if the measurements are in the list of known faces, the system will signal that the person has been identified.

If the system receives a STORE... Post request, with … containing the facial measurements of the face currently being displayed, it will read the name contained in the post request and save it with the facial measurements in the list of known faces.

There are also Post requests for the actions the user can take, which trigger board functions like sweeping the LEDs (SWEEP), and writing text to the LCD display (LCD).

Board Control process: This process will monitor for messages sent from the web-server to perform basic board functions, such as turning LEDs on or off, displaying text on the LCD, or sending pixel data to the Ethernet controller.

Communications server: This is a java program running on a connected computer. We had intended for this to be contained on the board, but it required more resources than the board could provide. This program will periodically poll the Web-server process (with STATUS_REQUEST POST requests) to determine whether or not the system is ready to analyze image data. Once it is ready, the process will send a GET request to the webserver for find_face.bmp, storing the image data, and saving it as image.bmp. The process will then wait for values.txt to be created by the landmark detector; once it has been detected, the float values will be read in and stored, and values.txt will be deleted. If all 6 required values are there, they will be used to construct a doPost request sending those values back to the Web-server process, signifying a face has been successfully recognized. If not, the program will request another image via the doGet request, and repeat the cycle until a face has been successfully recognized.

Facial Landmark Detector: This process will wait for the Communications process to create image.bmp, at which point it will read it into a data structure and delete the file. A greyscale copy will be made of the color image, and both images will be resized from 320x240 to 640x480. Next, the resized greyscale image will be equalized to increase contrast, and the XML file containing the eye patterns to search for will be loaded into memory. A function will then sweep through the image to locate matches for the eye pattern. The coordinates of the matching points will be saved, and circles will be placed over the points on the resized color image. This process will be repeated with XML files containing nose and mouth patterns.
For each of those sweeps, if there are too many points found, only the point most likely to correspond to the desired facial feature will be kept. Once the three sweeps have been made, the distance between each point will be calculated, divided by the distance between the first two points (which will be the left and right eye, upon a successful sweep), and stored as single-precision float values. A file called “values.txt” will be created, and all the values written to it.

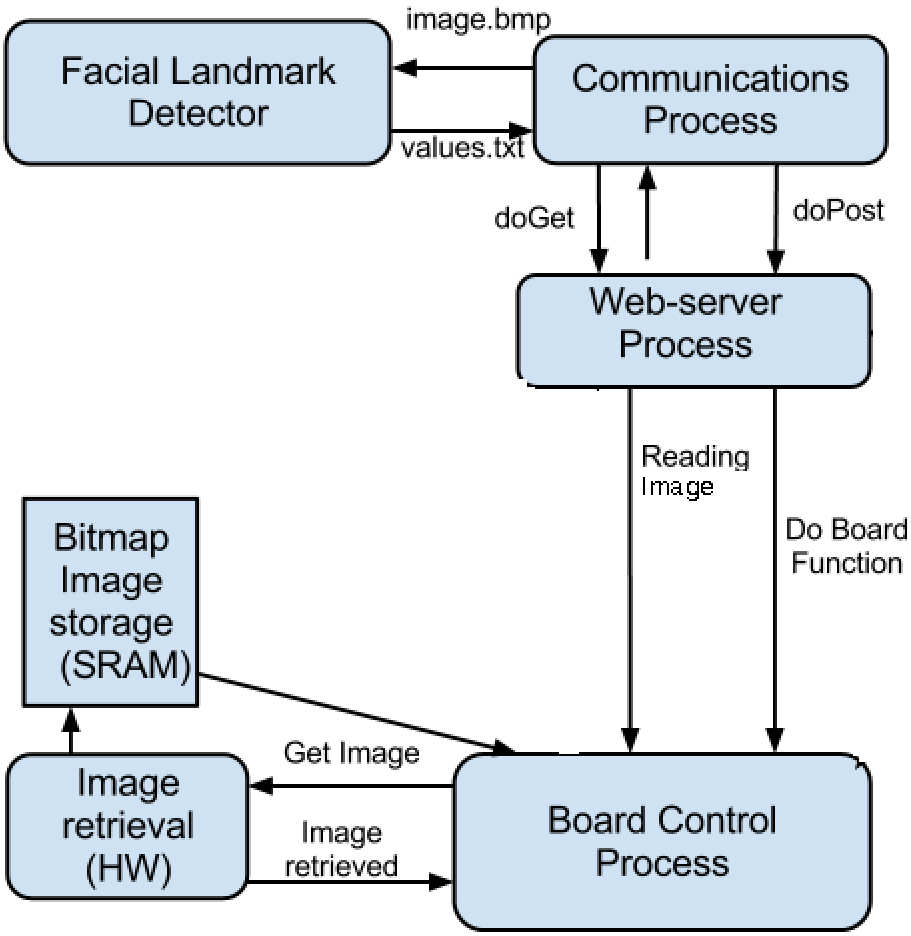


Fig 1: Process interaction

Test Plan
Software:
Image analysis:
Test of facial recognition in example images on laptop: Success
Test of facial recognition software on Altera DE2 board: Failure, not enough RAM to fit requirements.
Test of facial recognition of images pulled from web-server: Success.

Compression:
Compression abandoned due to time

Web-Server:
1) Serve sample web-page: Success
2) Serve sample image in GET response: Success, avg 55kB/s speed
3) Respond to POST requests with sample reply: Success
4) Convert sample raw RGB image data to Bitmap image: Success
5) Trigger image read from camera and display as bitmap: Success, avg 55 kB/s speed
6) Serve auto-refreshing web-page: Success
7) Serve Identified Person web-page: Success

8) Serve unidentified person web-page: Success
9) Search database with sample facial measurements and identify person: Success
10) Store name and sample facial measurements from unidentified person STORE POST request: Success
11) Test STATUS_REQUEST POST response with facial recognition server: Success
12) Respond to FACE_FOUND request from facial recognition server: Success
13) System will auto-refresh until facial recognition server finds a face in image served by web-server, find returned measurements in database, show web-page, repeat on page refresh: Success
14) Writing to SRAM: Failure due to SRAM clock speed being higher

Board Control:
Test that each function results in the desired response from the board.

Hardware:
Ethernet:
Test that connection detection is working (i.e. lights turn on with cable is plugged in): Success
Test that TCP/IP communication is working (ping responses are correct):Success

Image Retrieval:
Test image data conversion from YCbCr to RGB by outputting directly to VGA: Success, occasional discoloured lines appearing in image.
Test writing image to SDRAM: Failure
Test writing image to SRAM: Success, some discoloured lines, but less noticeable.

Results of Experiments
Originally we had intended to use 640x480 JPEG images, but due to lack of space on SRAM and a not having enough time to implement JPEG compression, we ended up using 320x240 RGB image data which is converted to Bitmap format on the fly.

Despite our best efforts, we could not fit the facial recognition program onto the DE2 board, so we wrote a program to do the facial recognition on the computer connected to the board, leaving the board to handle image generation and web-serving. This required setting up a 2 way communication system between the web-server and the facial recognition program.

References

[1] Sample Webserver & DM9000A driver from Terasic samples at http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=30&PartNo=4

[2] Video input example from University of Toronto ECE 241 website at
http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/vin/vin.html

[3] Facial recognition via Eigenfaces method from Open source examples at http://www.face-rec.org/source-codes/

[4] SimpleSocketServer by user hasil86 referenced to help setting up web-server on DE2 board
http://alteraforums.com/forum/showthread.php?t=24020&page=7

[5] OpenCV library and wiki at
http://opencv.willowgarage.com/wiki/

[6] JPEG compression code found at
http://www.ijg.org/

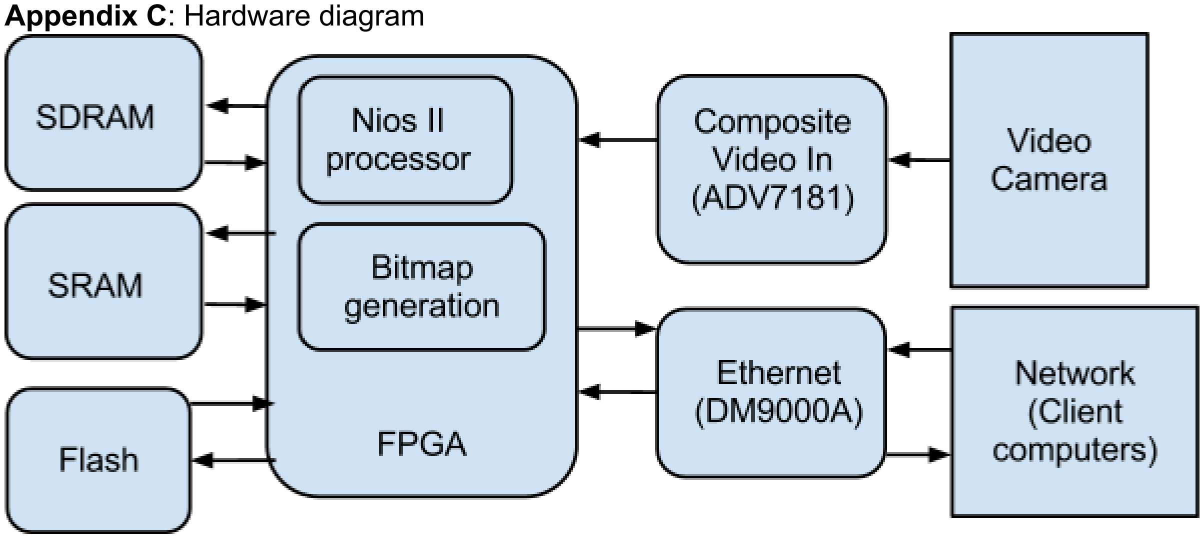
[7] Javascript auto-refresh
http://www.quackit.com/javascript/javascript_refresh_page.cfm

Appendix A: Quick Start Guide
0. If the laptop does not have the OpenCV library, compile and install it at C:/OpenCV2.3.
1. Compile the webserver.<extension> Quartus project.
2. Connect DE2 board and refresh the JTAG daemon.
3. Program webserver.sof or webserver.pof to the DE2 board.
4. Open Nios II Software Build Tools for Eclipse, via the SOPC builder.
5. Generate the BSP for the webserver_bsp project.
6. Build the webserver project.
7. Run the webserver project as "Nios II Hardware"
8. On the laptop, set the Local Area Connection IP address to 196.254.70.xxx (where xxx does not equal 234), the subnet mask to 255.255.255.0, and the default gateway to 196.254.70.1.
9. Connect the composite video output of the camera to the DE2 board, and connect the laptop to the DE2 board via Ethernet cable.
10. Build and run the face_detector0 project on Eclipse, on the laptop.
11. Build and run the Face_Recog_Server project on Eclipse, on the laptop.
12. Open your browser of choice and enter "http://196.254.70.234/facefinder.html" to start the face recognition polling.

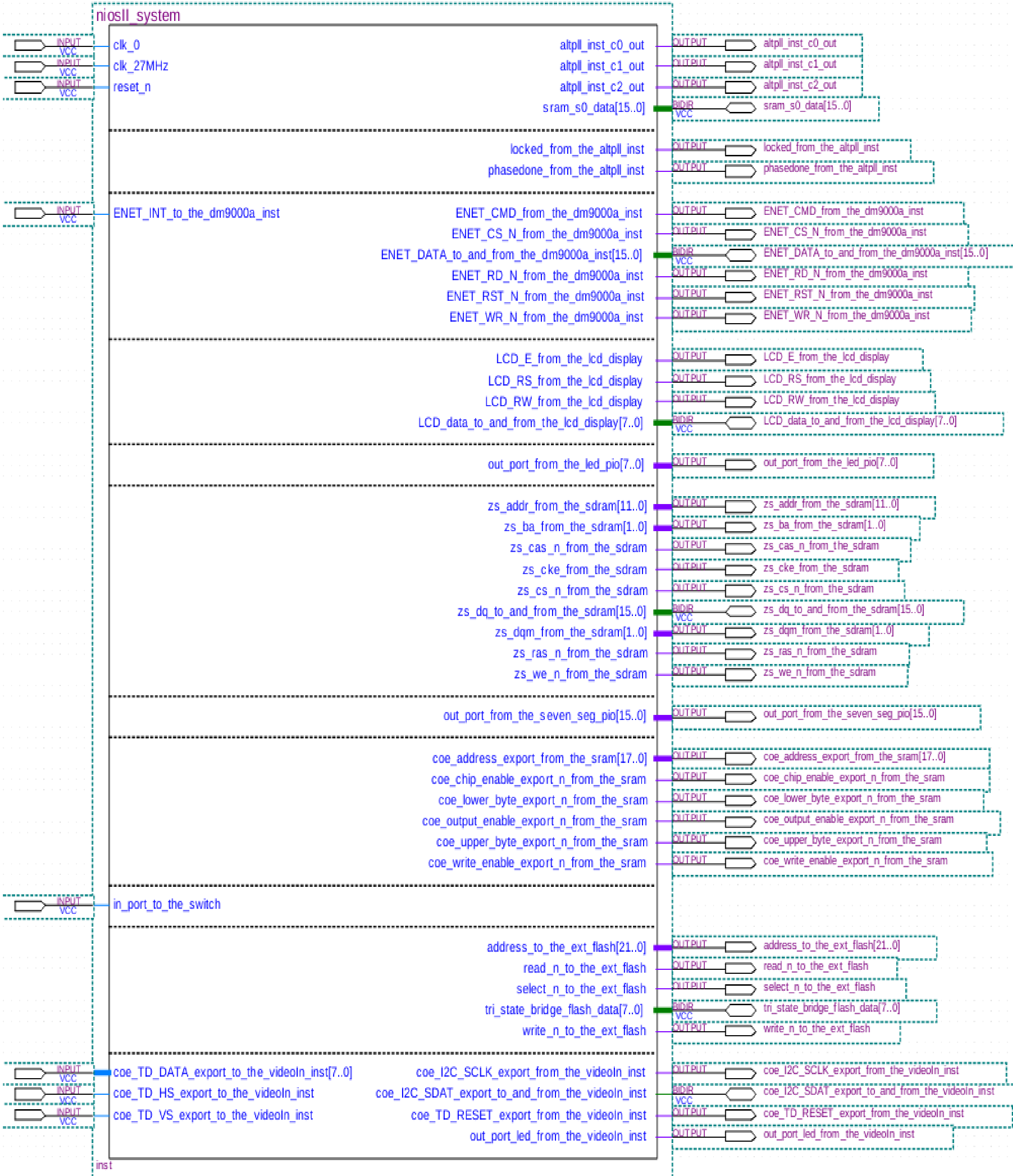
Appendix B: Future Work

- Implement rudimentary emotion recognition (eg. smiling == happy, frowning == sad, furrowed brow == angry)
- Humorous automated image modifications (eg. change color of irises, add silly mustache, et cetera)
- Implement more advanced facial recognition algorithm
- Provide audio notification from the board once a face has been recognized
- Add in JPEG compression of the image file to allow for speedier operation
- Include audio/video streaming capability

Appendix C: Hardware diagram

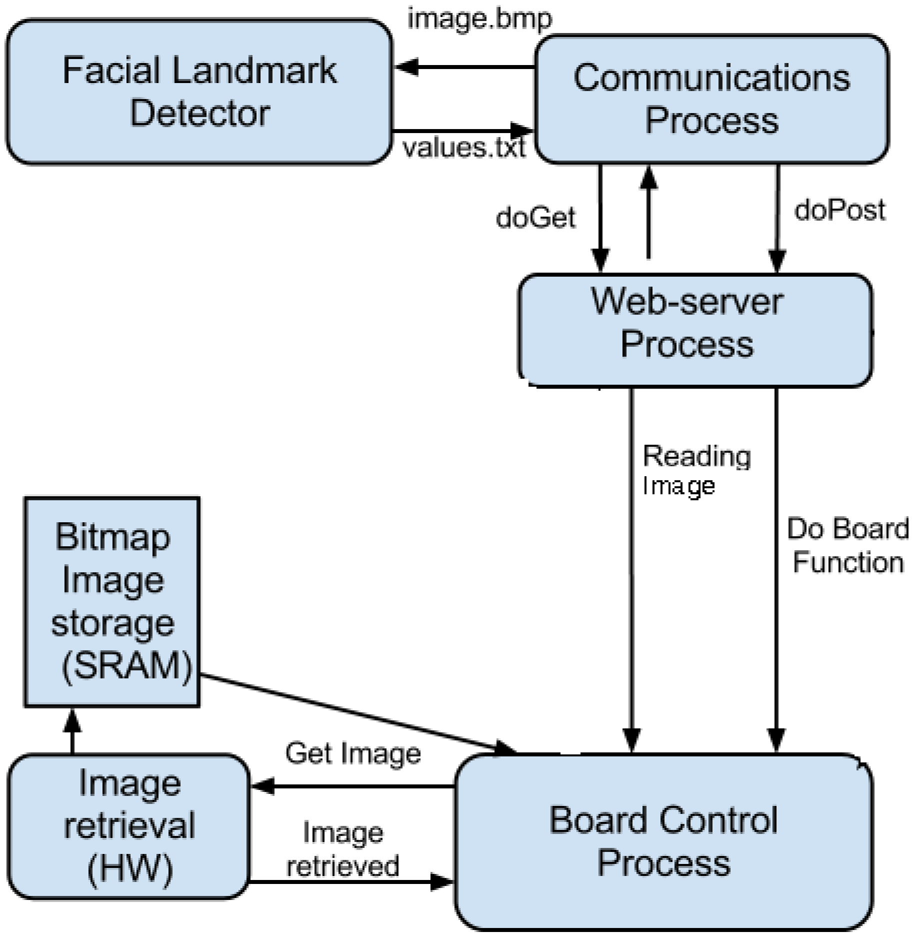


Schematics:



niosII_system
clk_0
clk_27MHz
reset_n
altpll_inst_c0_out
altpll_inst_c1_out
altpll_inst_c2_out
sram_s0_data[15..0]
locked_from_the_altpll_inst
phasedone_from_the_altpll_inst
ENET_INT_to_the_dm9000a_inst
ENET_CMD_from_the_dm9000a_inst
ENET_CS_N_from_the_dm9000a_inst
ENET_DATA_to_and_from_the_dm9000a_inst[15..0]
ENET_RD_N_from_the_dm9000a_inst
ENET_RST_N_from_the_dm9000a_inst
ENET_WR_N_from_the_dm9000a_inst
LCD_E_from_the_lcd_display
LCD_RS_from_the_lcd_display
LCD_RW_from_the_lcd_display
LCD_data_to_and_from_the_lcd_display[7..0]
out_port_from_the_led_pio[7..0]
zs_addr_from_the_sdram[11..0]
zs_ba_from_the_sdram[1..0]
zs_cas_n_from_the_sdram
zs_cke_from_the_sdram
zs_cs_n_from_the_sdram
zs_dq_to_and_from_the_sdram[15..0]
zs_dqm_from_the_sdram[1..0]
zs_ras_n_from_the_sdram
zs_we_n_from_the_sdram
out_port_from_the_seven_seg_pio[15..0]
coe_address_export_from_the_sram[17..0]
coe_chip_enable_export_n_from_the_sram
coe_lower_byte_export_n_from_the_sram
coe_output_enable_export_n_from_the_sram
coe_upper_byte_export_n_from_the_sram
coe_write_enable_export_n_from_the_sram
in_port_to_the_switch
address_to_the_ext_flash[21..0]
read_n_to_the_ext_flash
select_n_to_the_ext_flash
tri_state_bridge_flash_data[7..0]
write_n_to_the_ext_flash
coe_TD_DATA_export_to_the_videoIn_inst[7..0]
coe_TD_HS_export_to_the_videoIn_inst
coe_TD_VS_export_to_the_videoIn_inst
coe_I2C_SCLK_export_from_the_videoIn_inst
coe_I2C_SDAT_export_to_and_from_the_videoIn_inst
coe_TD_RESET_export_from_the_videoIn_inst
out_port_led_from_the_videoIn_inst
inst
INPUT
VCC
OUTPUT
BIDIR

Appendix D: Software Diagram



Source code

Verilog/VHDL:
(In directory webServer)
altpll_inst.vhd
altpll_inst.vho
bigfatFIFO.cmp
bigfatFIFO.qip
bigfatFIFO.vhd
clock_source_27.vhd
cpu.ocp
cpu.sdc
cpu.vhd
cpu_bht_ram.mif
cpu_dc_tag_ram.mif
cpu_ic_tag_ram.mif
cpu_inst.ocp
cpu_inst.sdc
cpu_inst.vhd
cpu_inst_bht_ram.mif

cpu_inst_dc_tag_ram.mif
cpu_inst_ic_tag_ram.mif
cpu_inst_jtag_debug_module_sysclk.vhd
cpu_inst_jtag_debug_module_tck.vhd
cpu_inst_jtag_debug_module_wrapper.vhd
cpu_inst_mult_cell.vhd
cpu_inst_ociram_default_contents.mif
cpu_inst_oci_test_bench.vhd
cpu_inst_rf_ram_a.mif
cpu_inst_rf_ram_b.mif
cpu_inst_test_bench.vhd
cpu_jtag_debug_module_sysclk.vhd
cpu_jtag_debug_module_tck.vhd
cpu_jtag_debug_module_wrapper.vhd
cpu_mult_cell.vhd
cpu_ociram_default_contents.mif
cpu_oci_test_bench.vhd
cpu_rf_ram_a.mif
cpu_rf_ram_b.mif
cpu_test_bench.vhd
dm9000a
dm9000a.vhd
dm9000a_hw.tcl
dm9000a_hw.tcl~
dm9000a_inst.vhd
high_res_timer.vhd
jtag_uart.vhd
lcd_display.vhd
led_pio.vhd
niosII_system.bsf
niosII_system.html
niosII_system.ptf
niosII_system.ptf.8.0
niosII_system.ptf.bak
niosII_system.ptf.pre_generation_ptf
niosII_system.qip
niosII_system.sopc
niosII_system.sopcinfo
niosII_system.vhd
niosII_system_burst_0.vhd
niosII_system_burst_1.vhd
niosII_system_burst_10.vhd
niosII_system_burst_11.vhd
niosII_system_burst_12.vhd
niosII_system_burst_13.vhd
niosII_system_burst_14.vhd
niosII_system_burst_15.vhd
niosII_system_burst_16.vhd
niosII_system_burst_17.vhd
niosII_system_burst_18.vhd
niosII_system_burst_19.vhd

niosII_system_burst_2.vhd
niosII_system_burst_20.vhd
niosII_system_burst_21.vhd
niosII_system_burst_22.vhd
niosII_system_burst_3.vhd
niosII_system_burst_4.vhd
niosII_system_burst_5.vhd
niosII_system_burst_6.vhd
niosII_system_burst_7.vhd
niosII_system_burst_7.vhd.bak
niosII_system_burst_8.vhd
niosII_system_burst_9.vhd
niosII_system_clock_0.vhd
niosII_system_clock_1.vhd
niosII_system_clock_10.vhd
niosII_system_clock_11.vhd
niosII_system_clock_12.vhd
niosII_system_clock_13.vhd
niosII_system_clock_14.vhd
niosII_system_clock_15.vhd
niosII_system_clock_2.vhd
niosII_system_clock_3.vhd
niosII_system_clock_4.vhd
niosII_system_clock_5.vhd
niosII_system_clock_6.vhd
niosII_system_clock_7.vhd
niosII_system_clock_8.vhd
niosII_system_clock_9.vhd
niosII_system_generation_script
niosII_system_inst.vhd
niosII_system_log.txt
niosII_system_sim
nvram_controller.vhd
nvram_controller.vhd.bak
nvram_controller_0.vhd
nvram_controller_hw.tcl
nvram_controller_hw.tcl~
onchip_memory2_0.hex
onchip_memory2_0.vhd
onchip_memory2_inst.hex
onchip_memory2_inst.vhd
sdram.vhd
seven_seg_pio.vhd
sopc_add_qip_file.tcl
sopc_builder_log.txt
sram.vhd
sram_controller.vhd
sram_controller_hw.tcl
sram_controller_hw.tcl~
stp1.stp
switch.vhd

sysid.vhd
sys_clk_timer.vhd
transfer_FIFO.cmp
transfer_FIFO.qip
transfer_FIFO.vhd
VGA_Adapter.vhd
VGA_Adapter.vhd.bak
VGA_Adapter_hw.tcl
VGA_Adapter_inst.vhd
VGA_PLL.qip
VGA_PLL.v
videoIn 48.vhd
videoIn.vhd
videoIn.vhd.bak
videoIn_copy.vhd
VideoIn_Core
VideoIn_Core_Mod
VideoIn_Demo_Mod.vhd
VideoIn_Demo_Mod.vhd.bak
videoIn_hw.tcl
videoIn_hw.tcl.bak
videoIn_hw.tcl~
videoIn_inst.vhd
videoIn_no_FIFO.vhd
Video_input_fifo.cmp
Video_input_fifo.qip
Video_input_fifo.vhd
Video_In_FIFO.qip
Video_In_FIFO_mod.cmp
Video_In_FIFO_mod.qip
Video_In_FIFO_mod.vhd
webServer.asm.rpt
webServer.cdf
webServer.done
webServer.dpf
webServer.eda.rpt
webServer.fit.rpt
webServer.fit.smsg
webServer.fit.summary
webServer.flow.rpt
webServer.jdi
webServer.map.rpt
webServer.map.summary
webServer.merge.rpt
webServer.pin
webServer.pof
webServer.pow.rpt
webServer.pow.summary
webServer.qpf
webServer.qsf
webServer.qsf.bak

webServer.sof
webServer.sta.rpt
webServer.sta.summary
webServer.vhd
webServer.vhd.bak
avconf/avconf.v
avconf/I2C_Controller.v
greybox_tmp/greybox_tmp/mg3ms.v
greybox_tmp/greybox_tmp/mglks.v
greybox_tmp/greybox_tmp/mgons.v
VideoIn_Core/Altera_UP_ITU_R_656_Decoder.v
VideoIn_Core/Altera_UP_Video_In_Buffer.v
VideoIn_Core/Altera_UP_Video_In_Deinterlacer.v
VideoIn_Core/Altera_UP_Video_In_Deinterlacer.v.bak
VideoIn_Core/Altera_UP_Video_In_Resize.v
VideoIn_Core/Altera_UP_Video_In_Resize.v.bak
VideoIn_Core/Altera_UP_YCrCb_422_to_444_Converter.v
VideoIn_Core/Altera_UP_YCrCb_to_RGB_Converter.v
VideoIn_Core/Altera_UP_YCrCb_to_RGB_Converter.v.bak
VideoIn_Core/Dual_Clock_FIFO.v
VideoIn_Core/Video_In.v
VideoIn_Core/Video_In.v.bak
VideoIn_Core/Video_In_FIFO.qip
VideoIn_Core/Video_In_FIFO.v
VideoIn_Core/Video_In_FIFO.v.bak
VideoIn_Core/Video_In_FIFO_inst.v
VideoIn_Core/Video_In_FIFO_old.v

Program Code (C, Java, Assembly, etc.)
(In directory Facial_Recognition_Server)
face_detector0/detector.c
face_detector0/facedetector.c
face_detector0/structs.h
face_detector0/haarcascade_eye.xml
face_detector0/haarcascade_mcs_mouth.xml
face_detector0/haarcascade_mcs_nose.xml
Face_Recog_Server/src/server.java

(In directory webServer)
dm9000a/HAL/inc/...
dm9000a/UCOSII/src/...
software/webServer/alt_error_handler.c
software/webServer/alt_error_handler.h
software/webServer/create-this-app
software/webServer/http.c :This handles the HTTP GET and POST responses, it keeps track of facial data, serves up the dynamic web-page, retrieves image data from SRAM and converts to Bitmap, and holds and searches the list of known faces.
software/webServer/http.h
software/webServer/image.h
software/webServer/Makefile
software/webServer/network_utilities.c

software/webServer/srec_flash.c
software/webServer/web_server.c
software/webServer/web_server.h
software/webserver_bsp/alt_sys_init.c
software/webserver_bsp/libucosii_bsp.a
software/webserver_bsp/linker.h
software/webserver_bsp/linker.x
software/webserver_bsp/Makefile
software/webserver_bsp/system.h
software/webserver_bsp/drivers/inc/...
software/webserver_bsp/drivers/src/...
software/webserver_bsp/HAL/inc/...
software/webserver_bsp/HAL/src/...
software/webserver_bsp/iniche/inc/...
software/webserver_bsp/iniche/src/...
software/webserver_bsp/UCOSII/inc/...
software/webserver_bsp/UCOSII/src/...