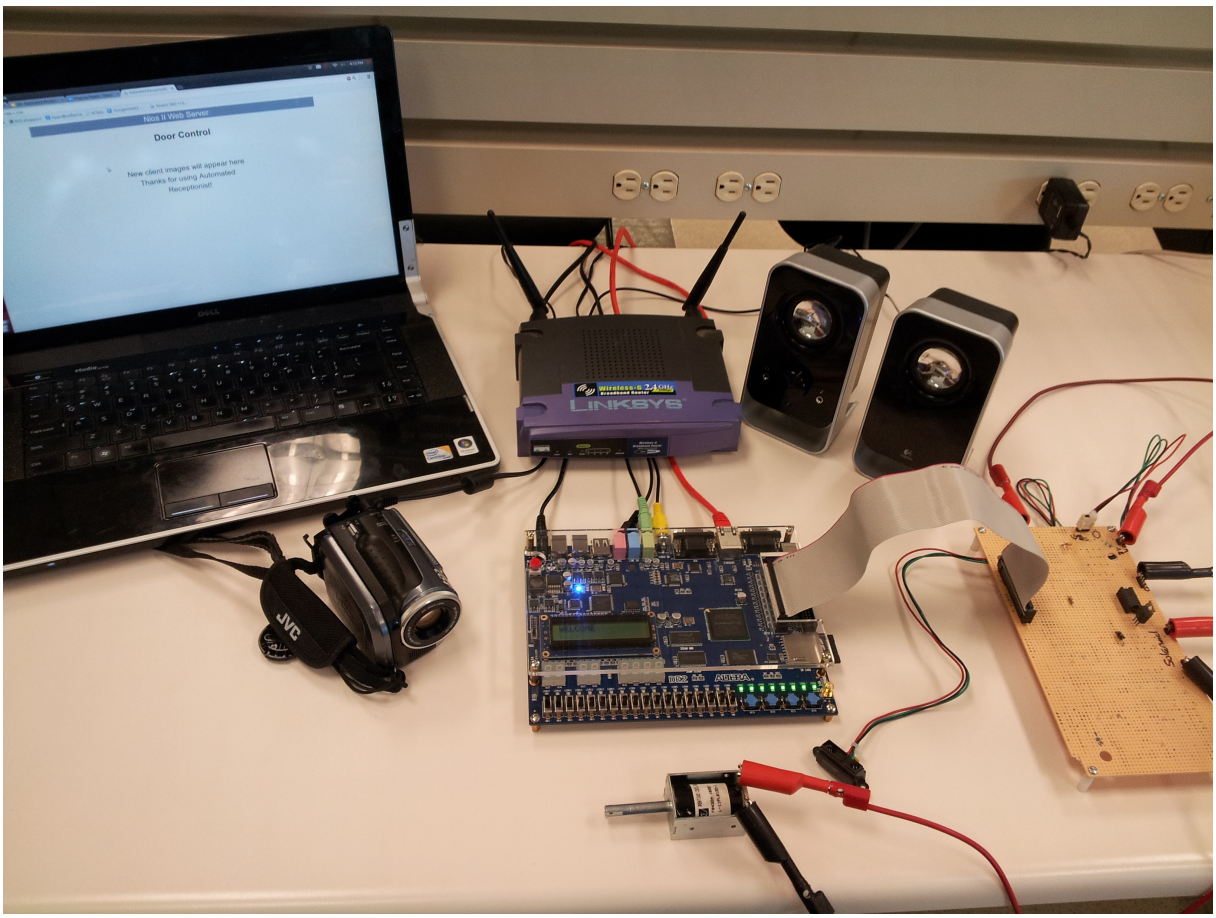
ECE 492 Final Report
Automated Receptionist



Project Summary
A system for monitoring the arrival of guests and grants or deny accesses to the premise as appropriate.

Group Number 9

Group Member
Andrew Maier amaier@ualberta.ca
Sila Luckanachai luckanac@ualberta.ca

Abstract
The project automates the role of a receptionist of a building and implements the security that is required to grant access to guests. The basic functionalities to be implemented are meant to provide an interface between the guest at the entrance and an authorized personnel. Once the presence of a guest is detected, an automated message will be played to welcome and inform the individual that the appropriate personnel has been notified of their arrival. The system will capture a picture of the guest, stores it on the SD card and update the photo to the web-server along with a timestamp. At this point the authorized personnel will be notified via an update to the web page from the web-server that a guest has arrived. Access can then be granted or denied as per the personnel request. This request will be carried out through the use of an attached solenoid latch. In order to implement these functionalities, specific hardware must be present. It will require an infrared sensor to detect the guests, a mounted camera to take the picture, a speaker to play the message, and an electronic locking device to complete the physical security feature. All of the above mentioned functionalities were successfully integrated, however a different camera solution had to be constructed due to the lack of quality of our original camera.

Tables of Contents

Functional Requirements[1]

1. Detects and plays an automatic greeting upon the arrival guests in the pre-designated area.
2. Takes the photo of the guests and serves it to the client via the onboard web server.[2]
3. Interactively presents options for door control to the authorizing personnel via the web based interface.[3]
4. Allows the guest to be granted access through the actuation of the solenoid latch.

Additional Function Implemented

1. Client-side door control web page may be accessed remotely via the added wireless network feature.

Evaluation of Functional Requirements Post-Project Completion

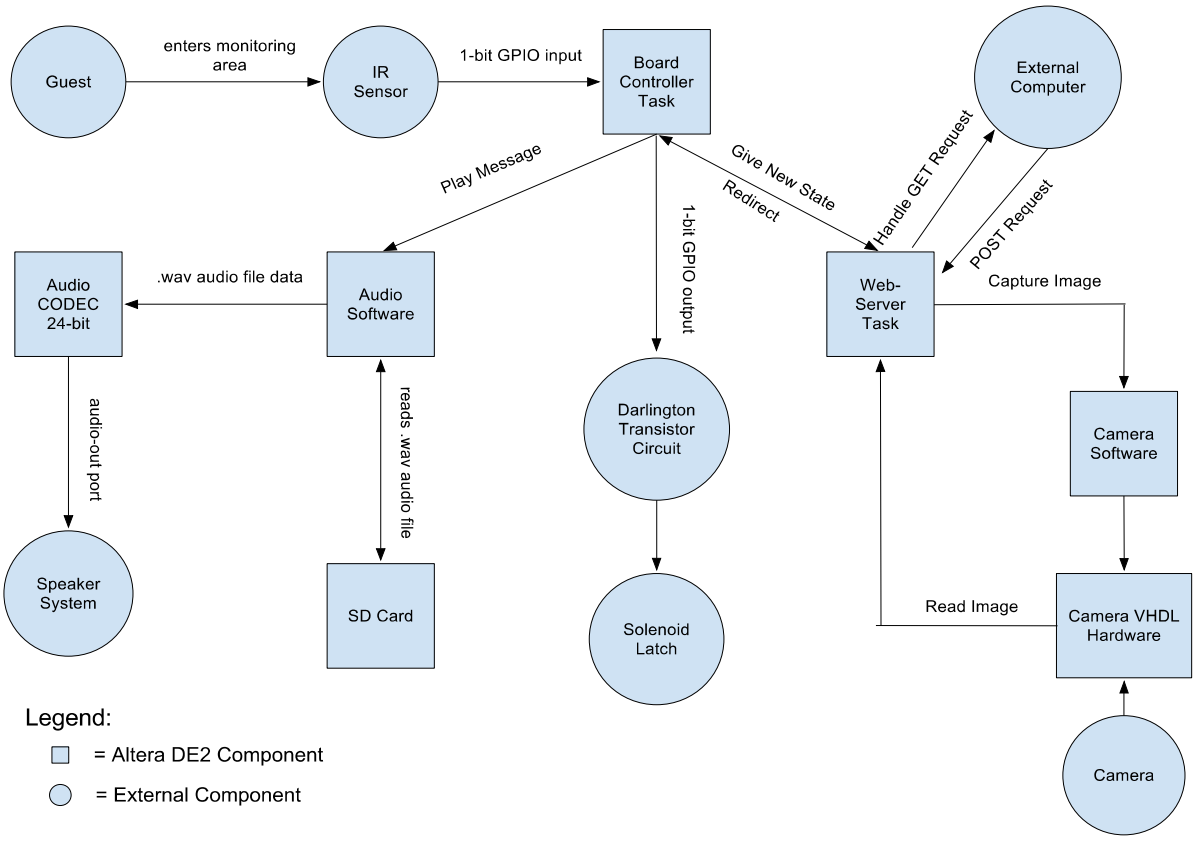
1. All functional requirements for the project were met fully.

[1] Due to the gap between the progress report and the final report, the functional requirements were altered to better reflect the changes in implementation.

[2] The timestamp is no longer necessary due to the added javascript which redirects the focus of the client to the door control page when a guest arrives.

[3] The 320x240 image captured from the JVC Video Camera via the RCA input is sufficiently small to be stored on the SRAM. As a result, the image no longer needs to be stored on the SD Card which is solely reserved for the .WAV files.

Design and Description of Operation



The project is run using the Altera DE2 hardware FPGA board. The processor for the software is the NiosII fast-type processor with 8KB Instruction and Data Caches, a 32-bit cache line size, and burst mode enabled for both. The uC/OS-II operating system runs atop the Nios processor and is beneficial for the program because it creates a real-time system using preemptive multitasking.

Operation of the entire program is controlled with a general Control Task on the board which holds the state of the program. By reacting to an interrupt generated from the infrared sensor GPIO input, control task invokes an audio greeting message. This task is set at a lower priority than the web server task. This setup allows the multi-tasking to continue to respond to ethernet http requests.

The web server task is then instructed to redirect generic client requests to the interactive interface. Upon a request for this page (via javascript refresh), the camera hardware is invoked to capture a frame through the video input. The raw image is then sent over the ethernet in the form of a bitmap image. This page is where the client is displayed the guest's photograph and presented with a list of options.

The http posts directed to the web server are passed to the control task on the board. The board then reacts accordingly by playing the appropriate audio message to the guest and actuating the latch if necessary.

Bill of Materials[1]

Parts	Description, Supplier, Parts Number and Datasheet	Qty	Cost
Development Board	Altera DE2 Development Board Altera Corporation http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html Parts Number: P0301 Data Sheet: ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2/DE2_Datasheets.zip	1	269
IR Sensor	Sharp GP2Y0A02YK0F Distance Measuring Sensor Unit Digi-Key Corporation http://www.digikey.ca/product-search/en?x=22&y=15&lang=en&site=ca&KeyWords=GP2Y0A02YK0F Parts Number: 425-2062-ND Data Sheet: http://www.sharpsma.com/webfm_send/1487	1	24.22
Camera	JVC GZ-E505B Full HD Camcorder JVC Americas Corp http://camcorder.jvc.com/product.jsp?modelId=MODL029206&pathId=171&page=10 Parts Number: GZ-E505B Data Sheet: http://www.jvc.ca/service-manuals/man2602ien.pdf	1	349.95
SD Card	Kingston SDHC Card, Class 2, 2 GB Amazon.com, Inc http://www.amazon.ca/Kingston-SD-Flash-Memory-	1	5.99

[1] Due to the layout of the report, the relevant specifications of the parts listed are summarized in the "Datasheet" section.

	2GB/dp/B000EOMXM0 Parts Number: B000EOMXM0 Data Sheet: http://www.kingston.com/datasheets/sd2_us.pdf		
Speakers	Logitech LS11 Stereo Speaker Systems Amazon.com, Inc http://www.amazon.ca/Logitech-LS11-Stereo-Speaker-System/dp/B0015BYQJE Parts Number: 980-000048 Data Sheet http://logitech-en-amr.custhelp.com/app/answers/detail/a_id/8151/section/troubleshoot/crid/409/lt_product_id/4252/tabs/1,3,2,4,5/cl/us,en	1	19.99
Solenoid Latch	Amico DC 12V Open Frame Type Solenoid for Electrical Lock Amazon.com, Inc http://www.amazon.com/Amico-Open-Frame-Solenoid-Electric/dp/B005FOTJF8/ref=sr_1_1?s=electronics&ie=UTF8&qid=1365979914&sr=1-1&keywords=solenoid+latch Parts Number: B005FOTJF8 Data Sheet: Under the header 'Product Features' http://www.amazon.com/Amico-Open-Frame-Solenoid-Electric/dp/B005FOTJF8	1	23.22
Wireless Router	Linksys WRT54GL Wireless Router TigerDirect.ca, Inc http://www.tigerdirect.ca/applications/SearchTools/item-details.asp?EdpNo=1831582&csid=_61 Parts Number: YYI2-H54784 Data Sheet:	1	74.99

	http://downloads.linksys.com/downloads/userguide/1224653767384/WRT54GL_V11_UG_C-Web.pdf		
Darlington Transistor	TIP120 Power Darlington Transistor Digi-Key Corporation http://www.digikey.ca/product-detail/en/TIP120/TIP120-ND/1052441 Parts Number: TIP120-ND Data Sheet: http://www.adafruit.com/datasheets/TIP120.pdf	1	0.73
Capacitor	0.1 uFarad Capacitor	1	0.25
Resistor	150 Ohms Resistor	1	0.25
Wires	Red and Black Lab Wires	10	1
Insulated Alligator Clips	Fully Insulated 2” Alligator Clips	4	12
Ribbon Cables	30 cm 40-Pin Ribbon Cables	1	9
Power Supply	9V 1.5A Power Supply	1	0
Peripheral Circuit Board	Plain, wire wrap and general purpose board.	1	5
Wire Wrap and Solder Materials	22 and 24 Gauge Wiring plus Soldering Materials	10	0.25

Total Cost: $798.09

Available Resources of Reusable Designs

Terasic Demonstration Project -- SD Card Music Player
Available: ECE 492 Lab Section E-Class Website or Terasic Demonstration CD
Source Size: 16.27 MB
Evaluation: This is a viable source that provides a suitable starting point for an audio-based project which requires an SD card to store the .WAV file data. The SD Card is read using the SPI interface and is sufficiently fast for audio playback. The project itself is straightforward but is written entirely in Verilog subsequently, any students looking to integrate this project as a part of a larger VHDL project must correctly instantiate all of the Verilog components in their respective VHDL top level.

University of Toronto Video-in Controller
Available: http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/vin/vin.html
Source Size: N/A
Evaluation: This is a viable source that provides valuable background information about video input and processing. The site has several additional links clarify technical terms or topics to help the student familiarize themselves with the DE2 video controller. Additionally, the website provides a basic project which reads the raw video data from the RCA input, through the TV decoder and displays the result on an external monitor.

Cornell University ECE 5760 -- Final Project Music Player
Available:
http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2010/vs327_rw363/WAV_player/ECE%205760.htm
Source Size: N/A
Evaluation: This is a viable source that provides a comprehensive overview of the different protocols and signals required to implement a SD card music player. Similar to the Terasic SD Card Demonstration Project, the project uses the same audio core and top level file but comes with a source code section that provides a sample SOPC setup thus allowing the student to mimic the project on their own machine.The project contains a source code which may be reference for use in order to scan through the SD card and store known location of .WAV files as oppose to manually location the sector with which song cluster is located.

University of Alberta ECE492 Winter 2012 -- Group 8 Facial Recognition Project
Available: http://www.ece.ualberta.ca/~elliott/ece492/projects/2012w/g8_Facial_Recognition/
Source Size: N/A
Evaluation: This is a viable source that provides a starting point for any projects involving a video input and onboard web server. Group 8's report and diagrams provides an overview of the different components required to accomplish the previously stated tasks as well as how they interact with the DE2 board. The project contains the source code based on the University of Toronto's Video-in Controller which may be referenced for use in order to serve a bitmap image from a video camera to the client via the web server. The group had also uploaded an application note outlining how to setup and run the web server on the board. As of present, a newer and more detailed note has been updated by the ECE 492 Winter 2013 Group 9.

Datasheet

Proximity Sensor: *Sharp - GP2Y0A02YK0F*
A 15-centimetre proximity sensor detects the presence of guest in the designated area. Upon detection, the sensor output terminal voltage rises and triggers the ISR.

Output terminal voltage Vo = -0.3 to VCC+7.0
Supply voltage Vcc = -0.3 to +7.0

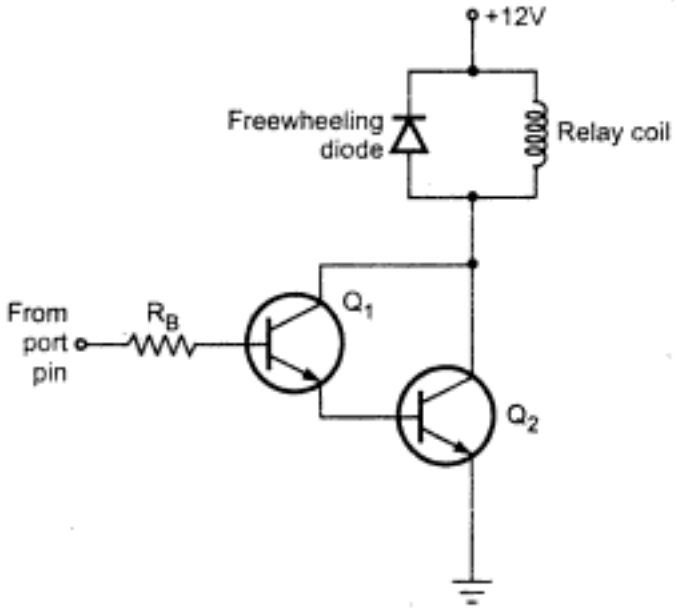
Solenoid Latch: *Amico DC 12V Open Frame Type Solenoid for Electrical Lock*
A 12V DC solenoid latch use for unlocking the door upon receiving the authentication approval from the internal personnel.

Supply voltage Vcc = 12 V
Supply Current I = 1.5 A

Darlington Transistor: *TIP120 Power Darlington Transistor*
The TIP120 is used to to increase the current gain supplied by the DE2 board by approximately 1000-times. As the DE2 provides 15 mA as an input to the TIP120, an output of 1.5 A is produced and used to actuate the Amico DC12V Open Frame Type Solenoid for Electrical Lock.

V_{BE} = 1.5V (with 2.5V max from datasheet)
I_B = 12mA peak
I_C = 1.2 A peak
R_B = 150 ohms

See the following circuit diagram implementation:



http://books.google.ca/books?id=VPaXf8ZCrmoC&pg=SA9-PA61&lpg=SA9-PA61&ots=I1AT7BQE94&dq=darlington+pair+to+drive+solenoid&output=html_text

SD Card: *Kingston SDHC Card, Class 2, 2 GB*
The SD Card stores the pre-recorded .WAV used to interact with the guest depending on the value of the HTML post from the personnel.

The SD card port to be connected to the FPGA pins are listed in the table below [1].

Signal Name	**Pin Number**
b_SD_cmd	PIN_Y21
b_SD_dat	PIN_AD24
b_SD_dat3	PIN_AC23
o_SD_clock	PIN_AD25

Speakers: *Logitech LS 11 Speakers*
The speakers play a pre-recorded greeting message through the Altera DE2 line-out upon receiving input from the GPIO due to the detection of the output terminal voltage of the *Sharp - GP2Y0A02YK0F* sensor going high to 7.0 V.

The pin assignments for in order to utilize the Altera DE2 24-bit audio CODEC line-out port is listed in the table below [2].

Signal Name	**Pin Number**	**Description**
AUD_ADCLRCK	PIN_C5	Audio CODEC ADC LR Clock
AUD_ADCDAT	PIN_B5	Audio CODEC ADC Data
AUD_DACLRCK	PIN_C6	Audio CODEC DAC LR Clock
AUD_DACDAT	PIN_A4	Audio CODEC DAC Data
AUD_XCK	PIN_A5	Audio CODEC Chip Clock
AUD_BCLK	PIN_B4	Audio CODEC Bit-Stream Clock
I2C_SCLK	PIN_A6	I2C Data
I2C_SDAT	PIN_B6	I2C Clock

Camera: JVC GZ-E505B Full HD Camcorder

A portable 2.5MP camera capable of outputting a composite RCA video output which is fed to the DE2 board via the RCA input. The raw video data is converted from ITU-R 651 format to the RGB format and stored as a 320x240 bitmap image on the SRAM.

Operating Power: 2.4 W
Focal Length: 2.1 - 79.8
Maximum Aperture: 1.8 - 3.9

Wireless Router: Linksys WRT54GL Wireless Router
A 2.4 GHz band wireless router used as a gateway for to serve and connect the client-side door control web page to the onboard web-server.

Supply Voltage: 12 V DC
Supply Current: 0.5 A
Wireless Signal: IEEE 802.3, IEEE 802.3u, IEEE 802.3g, IEEE802.11b
Channels: 11

Memory Element Usage
The memory element usage of the FPGA was calculated during the fitting process of Quartus:

Flow Status	Successful - Fri Apr 5 12:54:51 2013
Quartus II Version	10.1 Build 197 01/19/2011 SP 1 SJ Web Edition
Revision Name	full_project
Top-level Entity Name	project_top
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	8,590 / 33,216 (26 %)
Total combinational functions	7,592 / 33,216 (23 %)
Dedicated logic registers	5,013 / 33,216 (15 %)
Total registers	5185
Total pins	254 / 475 (53 %)
Total virtual pins	0
Total memory bits	360,704 / 483,840 (75 %)
Embedded Multiplier 9-bit elements	14 / 70 (20 %)
Total PLLs	2 / 4 (50 %)

See Appendix E for more details regarding the system setup.

Calculated Power Consumption
Calculated Static Power:9 V and 1.3 A or 11.7 W
Calculated by: As all external components connected to the system has their own power supply, the calculation of static power is based solely on the information provided by Altera in the DE2 Hardware Specification Manual which states that the board requires 9V and 1.3A.

Calculated Peak Power: 9V and 1.3 A or 11.7 W
Calculated by: Ideally the peak power should be the same as the static power given that there are all power supply provide sufficient power and no power is consumed/lost due to resistance, leakage, etc.

Measured Static Power: 9.09 V and 0.519 A or 4.72 W
Measured by: With the entire system hooked up, the DE2 board power was redirected through a serial ammeter and a parallel voltage reader. With these two DMM's in place, the voltage and current drawn by the board could be measured through the different operational states of the program. To measure the static power dissipation, we started the operation and allowed the system to rest and only reacting to ethernet client requests. This gave us a power of 4.72 W.

Measured Peak Power: 9.09 V and 0.560 A or 5.09 W
Measured by: With the entire system hooked up as described in the static power state. We ran the entire operation of the project and noted the changes to the current and voltage. The maximum power dissipation occurred when the audio codec was invoked to play wav files from the SD card. This resulted in a power usage of 5.09 W.

Top-Level I/O Signals

Signal	Type	Description
CLOCK_50	IN	The 50 MHz clock signal used by Ethernet PLL to divide down to the required 25 MHz clock. It is also used by the altpll_inst PLL to generate the 100 MHz clock for the system.
CLOCK_27	IN	The 27 MHz clock signal used by the Audio PLL to divide down to the required 18.4 MHz clock.
SW	IN	Signal for the DE2 switches.
KEY	IN	Signal for the DE2 keys, in this case it is only used for the system reset which is tied to KEY(0).
TD_DATA	IN	System signal used for passing the video in data from the DE2 serial video input to the video in buffer.
TD_HS	IN	The TV decoder's horizontal sync signal used to keep track of the horizontal position.
TD_VS	IN	The TV decoder's vertical sync signal used to keep track of the vertical position.
TD_RESET	OUT	The reset output signal of the TV decoder.
I2C_SCLK	OUT	Clock signal used by the I2C_AV_Config.v file to serially initialize the components registers.
I2C_SDAT	INOUT	The data containing the address and register values of the components.
GPIO_O	INOUT	The signal for each of the 40-GPIO pins for the GPIO bank 0.
SD_DAT	INOUT	Data signal for the SD card interface with the DE2 board.
SD_DAT3	INOUT	Data signal used to initialize the SD card mode, in this project it initializes the SPI protocol.
SD_CMD	INOUT	Data signal used by the SPI protocol to send various commands such as read, write, etc (this is done in hex value)
SD_CLK	OUT	The clock signal used by the SPI protocol to read and write to the SD card.
AUD_ADCLRCK	INOUT	The audio codec ADC left and right channel sampling clock running at 18.4 MHz.
AUD_ADCDAT	IN	The data signal mapped to AUD_DACDAT as it latches the audio data based on the AUD_ADCLRCK on each rising clock edge.

AUD_DACLRCK	INOUT	The audio codec DAC left and right channel. It is port mapped to the AUD_ADCLRCK clock and runs at 18.4 MHz.
AUD_DACDAT	OUT	The data signal mapped to the AUD_ADC_DAT as well as its latched value.
AUD_BCLK	INOUT	The audio codec bit stream clock which controls the playback speed of the line-out port.
AUD_XCK	OUT	The clock signal for the audio codec. Its value is chosen in accordance to the codec sampling rate initialized by the I2C_AV_Config.v

Background Reading

IEEE -- A SoPC-Based Mini VGA Video Capture and Storage System
K. Gao, “A SoPC-Based Mini VGA Video Capture and Storage System”, Biomedical Engineering and Informatics (BMEI), vol. 7, p. 2770 - 2774, October 16-18, 2010.
Available:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5639844&contentType=Conference+Publications&queryText%3Dimage+capture+sopc.

Background Information:
The IEEE article discusses the the detail of implementing a portable video capture system using FPGA, SDRAM, ADC, DAC, etc. The output of the video device is connected to a display monitor in stationary mode, but the video could be reconfigured to display on the system’s mini LCD when on-to-go and running on the lithium-ion battery. By pressing and holding the designated ‘capture’ key on the FPGA board, the system, begins writing the image from the video device frame-by-frame into the SDRAM or the SD Card.

Utility with project:
The article clearly shows the system’s hardware and software design whose example we could emulate for our upcoming report revision as we had scored poorly in the diagrams section previously. As we are also using a video capturing device, TRDB-D5m, the author’s discussion on the SD Card transfer mode directly impacts us. The SD Card core from Terasic implements both a 512-byte multiple block read/write as well as a single 512-byte block read/write function. Based on the article, the multiple block read/write should be selected over the single read/write as the multiple block read/write requires less overhead for each for all of the operations after the first block due to a use of simple tokens to signal the continuation of the operation. The author also provided statistics for image size for three common video image resolution as well as the amount of time required to store them in the SD Card after the ‘capture’ command is initiated. This figure will be important to us as we had projected a rough estimate in our I/O section of the report but have verified if the storing/retrieving process would be too cumbersome. Again, the statistic provided in the article shows the trade-off between resolution vs speed.

I2C Tutorial
Robot Electronics, “Using the I2C Bus”, 2001, Available:http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html.

Best-Micro-Controller-Projects, “A single master I2C tutorial”, 2005, Available:http://www.best-microcontroller-projects.com/i2c-tutorial.html.

Background Information:
The article outlines what the physical constitution of the I2C bus, description of the wires, timing diagram of the master-slave interaction/addressing and sample I2C controller code.

Utility with project:
The readings helped our group to better come to grasp with the purpose and operation of the I2C bus. At first we were uncertain of when and where I2C would be required/employed and the

extent to which it would affect the different components we are looking to interface, but after the background reading were able to understand the signal name, their purposes and the sequences required for master-slave interactions. This was handy as the I2C bus was used to control the audio codec on the DE2 and is present in virtually every example code involving audio usage. When we had originally tested out the audio codec application notes for previous terms, we did not understand why we were required to copy over the I2C_Controller.v and I2C_Config.v file as we had anticipated the audio codec to behave like any other hardware component i.e. SDRAM, GPIO, etc. As we chose to use the audio codec core from Terasic, we did not have to directly implement the I2C interface, nevertheless, the background knowledge of how it played a part in our component was crucial to our understanding of the audio CODEC (Wolfson 8731) component as a whole.

Miscellaneous Source

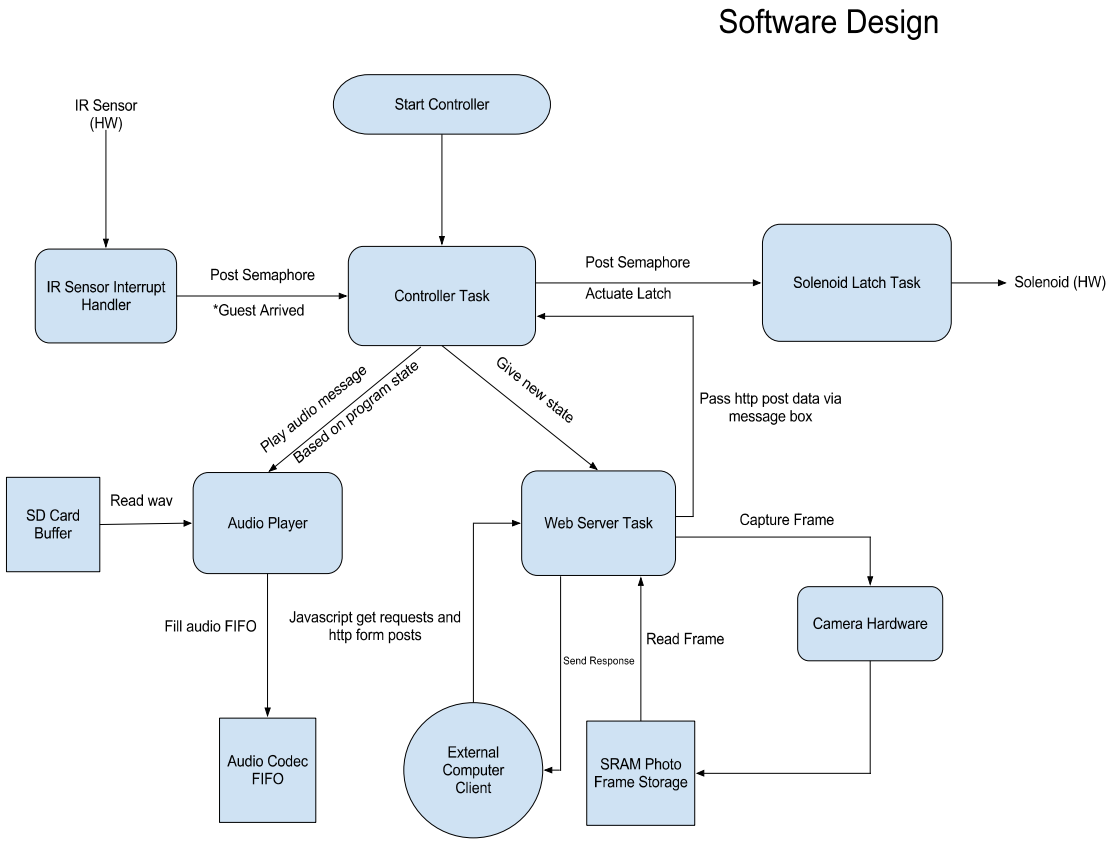
Cornell's University ECE4760 Lecture Series
ece4760 & ece5760 Cornell University, *Cornell's University ECE4760 Lecture Series.* [Videorecording]. Ithaca, New York : Youtube, LLC, 2012.
Available:http://www.youtube.com/course?list=ECD7F7ED1F3505D8D5

Background:
Cornell University and VideoNote had published a lecture series for the ECE 4760 class whose course material heavily involved embedded system design on the Altera DE2. The instructor, Dr. Bruce Land, lectures about different topics and components involved in various system design on the DE2 spanning from Quartus, timers, ADC/DAC, motor control, etc.

Utility with project:
The lecturer series was helpful as we could specifically skip to the topic of ADC/DAC topic for our SD card audio implementation. The video was helpful in solidify concepts borne from the high-level block diagram discussed in our class as well as clarifying the operation of the 24 kHz clock for the left-right channel data sampling (combined to give 48 kHz of data.)

Software Design

The project software ran on the NiosII fast type processor with 8KB instruction and data cache sizes, 32-bit cache line sizes, and burst mode enabled for both. Using the preemptive multitasking system uC/OS-II we needed approximately twelve tasks each with a stack size of 2048 bytes to accomplish the project.



Test Plan

Software

Hardware-Independent Components

Target Module	Description
Controller	Test the controller's semaphore behaviours by exhaustively posting to the semaphore for each task and verifying a correct print-out statement.
	Test that the interrupts supersedes any currently running task, a safe context switch is performed, the ISR executes correctly and a correct return to the original running task.
Web-server	Ensure correct operation by simulating a GET request to the web-server from the controller and checking for correct updates to the page displayed.
	Ensure proper interaction of the web-server's form posting by writing writing a message to the board's LCD showing the correct origin of the form post signal (i.e. "Allow Access" button, "Deny Access" button, etc)
Solenoid latch controller	Test the solenoid latch controller's semaphore behaviour by arbitrarily posting to them and verifying a correct print-out statement.

Hardware-Dependant Components

Target Module	Description
IR-sensor interrupt handler	Test the correctness of the interrupt handler by setting a designated threshold output terminal voltage which with the interrupt will be raised. Connect a voltmeter to the output terminal of the sensor and slowly shift an object from outside the proximity sensor field of vision closer until an interrupt is produced. Compare the voltmeter reading to the specified threshold for correctness.
Solenoid latch controller	Test the latch controller's functionality by writing an 'off' value and checking that the latch is not actuated. Conversely, write an 'on' value to the latch controller and check to see of the latch actuates.

Memory Leaks

Target Module	Description
Memory leaks	Test the system for memory leaks by powering the system on and leaving it to run for 1-2 hours, afterwards, proceed to test the system to ensure that all software components still function as expected without any occurrences of misbehaviours, slow-downs or deadlocks.

Hardware

Functional Tests

Target Module	Description
SRAM	Test the entire memory range of the RAM by write a ‘1’ to the starting address and rotating the bit through to the ending address and checking for mirroring also. The test module developed in Lab 2 will be reused for this functionality test.
SDRAM	Test the entire memory range of the RAM by write a ‘1’ to the starting address and rotating the bit through to the ending address and checking for mirroring also. The test module developed in Lab 2 will be reused for this functionality test.
Flash	Load a known-functional program onto the flash, run to ensure correct operation, power cycle the DE2 and test again for correct operation.
LCD module	Test the LCD display by running the code from Lab 1 with a message that spans the length of the LCD and toggling between the upper and lower display.
Line-out	Runs the Terasic I2Sound Demonstration, connect an audio source to the board’s line-in, listens for audio output through a speaker connected to the line-out jack.
DM9000A Ethernet	Ping the on-board web-server from an off-board machine. The arrival of the ping signifies a working ethernet.
Camera	Runs the Terasic TV Demonstration, connect the camera’s video out to the board’s Video RCA input and observe the video output on the external monitor connected to the board’s VGA output.
IR sensor	Connect the three terminals of the IR sensor to a supply voltage

	source, ground, and voltmeter respectively. By moving an object into the sensor's view, the voltmeter should display an increasing reading starting from 0.3V up to 7V.
Solenoid latch	Connect the two terminals of the solenoid latch to a 12V DC source and a ground respectively. By turning on the source, the solenoid latch should actuate thereby attracting the metal latch inward.
SD card	Format the SD card to the DE2 compliant FAT16 format and read/write to the card through a card reader to ensure functionality. After the successful SD card interface, use the API to write a '1' to the starting address and rotating it through til the ending address to verify that all of the memory are working.

Results of Experiments and Characterization

All of the hardware components used for this project are black-box thereby we do not and cannot calibrate the hardware themselves. If calibrations are required, they will be implemented in the hardware's respective software controller.

As of current we have verified the SRAM and SDRAM to be fully functional by applying the RAM testbench implemented in ECE 492 Lab 2. The IR Sensor and Solenoid latch appears to behave correctly upon the application of their respective hardware functional tests outlined above.

SD Card

Through our experimentation and research, we were able to deduce that In order to read from the SD Card using the SPI interface, the card may not exceed a 2GB size limit. and must be formatted as a FAT16 system. It is recommended that student use a SD card formatter which supports a function that write '0' to all memory to clear out any remnant data without a pointer (i.e. old 'deleted' audio files, etc.)

Video-in Image

The code for the TV decoder provided with the Terasic Demonstration Project natively handles a video input, via the RCA jack, of the resolution 640x480 and converts it to a 320x240 resolution. This is especially useful for any group that looks to serve an image from the camera to a client as the 320x240 is sufficiently small to be stored and read back through the SRAM.

I2C Configuration

There are two-approaches to implementing the I2C configuration: (1) custom VHDL core or (2) the bundled Terasic Demonstration I2C Verilog file. We have selected the latter of the two approaches and have found it to be sufficient to initialize both our audio codec and video input. The I2C provided requires a 50 MHz input clock and is unable to function properly with other clock frequency. The data is written serially to each register of the specific component based on the I2C_Config.v file (the name of the file may vary depending on Terasic Corporation's naming convention) and the is meant to be used with the I2C_Controller.v file which specifies the format which with the data is interpreted by the config file. Additionally components may be added so long as the student correctly reference the component's address and data registers.

Audio Files

The I2C initialization of the audio codec defaults a playback sample rate of 48KHz and thus the audio files used with the project must be also be recorded at 48KHz. Additionally, as the audio codec samples the data for both the left and right channel, the audio file must be in a 2-channel 16-bit .WAV format unless a decoder for other format was implemented.

100 MHz Clock Implementation

Upon integration of all the design components for the project, it was clear that the system could greatly benefit from a faster clock (ie 100 MHz). At the original 50 MHz it was clear our audio FIFO was not passing data into the Audio Codec at a speed that was optimal for our sampling frequency of 48 KHz. This resulted in the audio sounding distorted and slow. However, after multiplying the clock frequency up past 80 MHz the system seemed to inexplicably halt without

warning. Upon further system review, it was found that this error was caused by the internal feedback loop of our PLL. By default, the PLL compensates for the first (c0) clock output. However, this clock was purposefully phase shifted -3.00 ns to correct the delay in communication with the sdram. After changing the default compensation to the non-phase shifted output clock this error was corrected. An application note was created here to assist future students who experience this issue.

Terasic Camera

Upon experimenting with the product originally purchased, it was obvious that the quality of the image was sub-par. This ultimately led to the following of our backup plan to introduce a picture via the video in port of the DE2.

References

[1] J. Rose, "Video-in Controller," (Video-in Controller), [online] 2008, (Accessed: 14 Apr. 2013).
Available: http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/vin/vin.html

[2] A. Newcomb, T. Stefanyk, "Index of /~elliott/ece492/projects/2012w/g8_Facial_Recognition," (Index of /~elliott/ece492/projects/2012w/g8_Facial_Recognition), [online] 2012, (Accessed: 14 Apr. 2013).
Available: http://www.ece.ualberta.ca/~elliott/ece492/projects/2012w/g8_Facial_Recognition/

[3] Altera Corporation, “Altera DE2 Development and Education Board User Manual”, 2006.
Available: ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

[4] Adafruit Industries, “Adafruit Industries TIP120 Datasheet”, March 2005.
Available: www.adafruit.com/datasheets/TIP120.pdf

[5] R. Wong, V. Santhanagopalan, “ECE 5760 - Final Project Music Player,” [Webpage] 2011, (Accessed: 14 Apr. 2013).
Available:
http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2010/vs327_rw363/WAV_player/ECE%205760.htm

Appendix A: Quick Start Guide

Non-Volatile:

1. Uncompress the source files archive into a known location.
2. Connect the power to an Altera DE2 development board and plug in the USB connection to your computer via the USB Blaster port.
3. Open Quartus by navigating into the project folder in the terminal and executing "./scripts/launch_quartus.sh".
4. Turn off the board
5. Switch the left switch of the board from "Run" to "Prog".
6. Turn on the board.
7. Run the script "./scripts/reconnect_jtag.sh"
8. Program the "fullproject.pof" file onto the board using the Quartus Programmer with the Active Serial Programming mode.
9. Upon completion, turn the board off.
10. Switch the left switch of the board back from "Prog" to "Run"
11. Turn on the board and run the "./scripts/reconnect_jtag.sh" script.
12. Open the SOPC builder from Quartus (Tools->SOPC Builder).
13. Click on the System Generation tab at the top.
14. Click on the "NiosII Software Build Tools for Eclipse" button and when prompted make the workspace <project folder> -> software.
15. Open the flash programmer (Tools -> Flash Programmer...)
16. Start a new flash (File -> New..).
17. Use the SOPC info file and browse for the project2_system.sopcinfo file in the project folder.
18. Click "Add.." on the right.
19. Select the webserver.elf file (<project directory->software->webserver->webserver.elf).
20. Click "Add.." on the right again.
21. Select the ro_zipfs.zip file (<project_directory->software->webserver->system->ro_zipfs.zip).
22. Change the Flash Offset for the ro_zipfs.zip file to "0x200000" and hit enter.
23. Select the Erase flash before programming option (Options->Erase Flash before Program).
24. Click Start.
25. Upon finish, restart the board.
26. The project should now be running and an IP address displayed on the LCD.
27. Connect the external hardware peripheral board as per Figure 1 below.
28. Connect the camera AV output to the video in of the board.
29. Connect speakers via the 3.5mm line out of the board.
30. Connect the ethernet to a client laptop with the following configurations
    a. Gateway: 192.168.1.1
    b. Assign any ip address to external computers other than 192.168.1.234
31. Go to 192.168.1.234 on your web browser.

Volatile:

1. Uncompress the source files archive into a known location.
2. Connect the power to an Altera DE2 development board and plug in the USB connection to your computer via the USB Blaster port.
3. Open Quartus by navigating into the project folder in the terminal and executing "./scripts/launch_quartus.sh".
4. Run the script "./script/reconnect_jtag.sh".
5. Program the "fullproject.sof" file using the Quartus Programmer.
6. Open the SOPC builder from Quartus (Tools->SOPC Builder).
7. Click on the System Generation tab at the top.
8. Click on the "NiosII Software Build Tools for Eclipse" button and when prompted make the workspace <project folder> -> software.
9. Open the flash programmer (Tools -> Flash Programmer...)
10. Start a new flash (File -> New..).
11. Use the SOPC info file and browse for the project2_system.sopcinfo file in the project folder.
12. Click "Add.." on the right.
13. Select the ro_zipfs.zip file (<project_directory->software->webserver->system->ro_zipfs.zip).
14. Change the Flash Offset for the ro_zipfs.zip file to "0x200000" and hit enter.
15. Select the Erase flash before programming option (Options->Erase Flash before Program).
16. Click Start and exit upon completion.
17. Run the "webserver" project as "NiosII Hardware".
    a. Note if this fails, refresh your connection in the run configurations.
18. The project should now be running and an IP address displayed on the LCD.
19. Connect the external hardware peripheral board as per Figure 1 below.
20. Connect the camera AV output to the video in of the board.
21. Connect speakers via the 3.5mm line out of the board.
22. Connect the ethernet to a client laptop with the following configurations
    a. Gateway 192.168.1.1
    b. Assign any ip address to external computers other than 192.168.1.234
23. Go to 192.168.1.234 on your web browser.

*Note it is best to run the project in Non-volatile memory because the Jtag Uart is somewhat unreliable at 100 MHz.

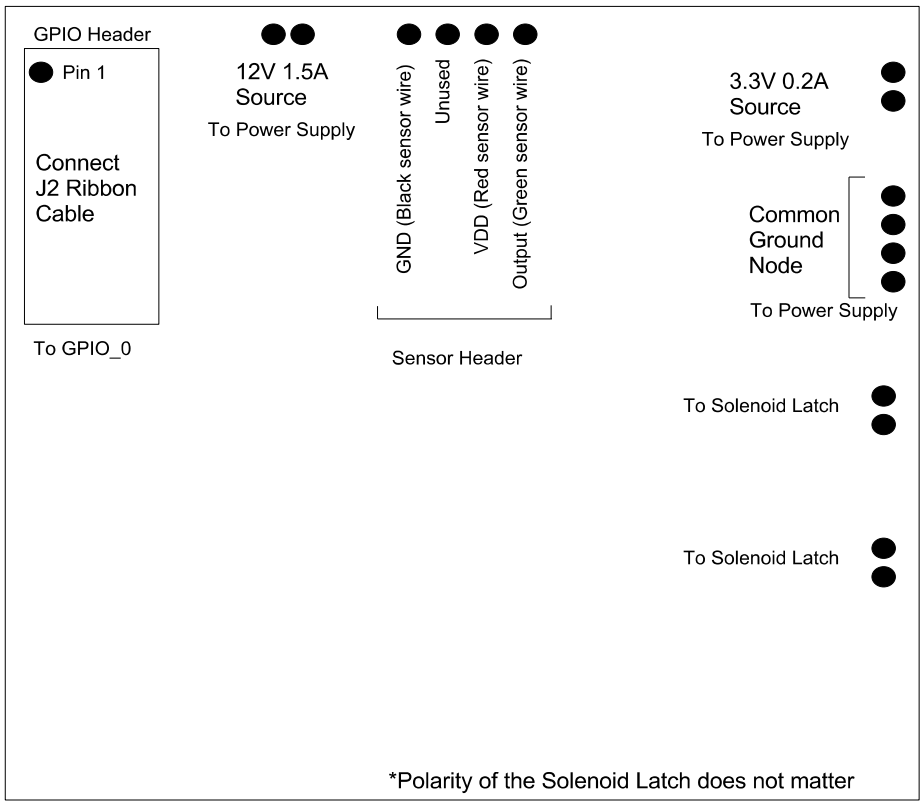


Figure 1: Hardware Peripheral Wiring Diagram

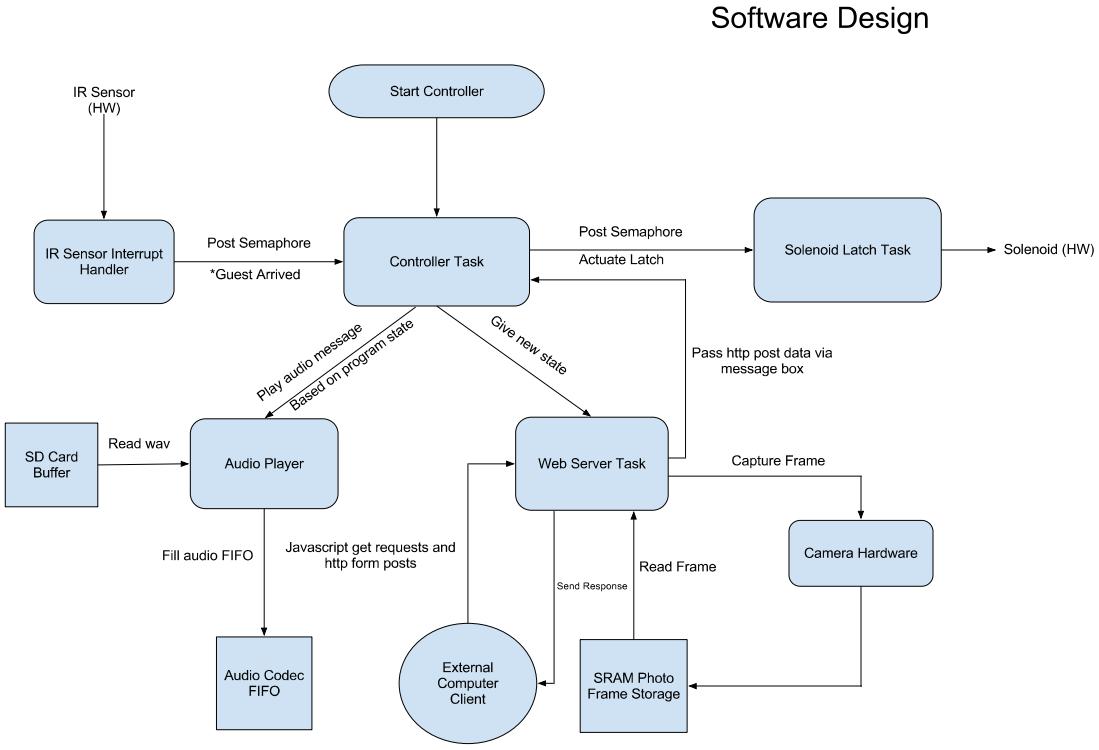
Please note that special consideration must be taken for the ordering of the files on the SD Card so that they are indexed and played properly. I recommend using one of our two pre-installed versions to keep the functionality.

Appendix B: Future Works

1. Verifies the identity of the guest by comparing the photo of the guest to a stored known-guest database.
2. Develop an Android application using the push-notification feature to alert the authorizing personnel of the presence of a guest.
3. Establishes a bi-directional audio link between the guest and a personnel upon pressing the “Contact Personnel” button.
4. Implement auto-framing and alignment of the guest’s photograph taken by the camera to alleviate constraint on deployment location.
5. Increases the photo diversity through the use of additional cameras.
6. Authorizes guest entry upon correct password entry via a keypad.
7. Authorizes guest entry upon correct password spoken via voice recognition.

Appendix C: Software and Source Code

Block Diagram of Interaction of Processes



Source Code Index[1]

AUDIO_DAC_FIFO
- hdl
  - AUDIO_DAC_FIFO.v
  - FIFO_16_256.v

avconf
- I2C_Controller.v

vga_adapter
- vga_adapter.v
- vga_address_translator.v
- vga_controller.v

[1] Any files and folders generated which were auto-generated are omitted from the index

- vga_pll.v

VideoIn_Core
- Altera_UP_ITU_R_656_Decoder.v
- Altera_UP_Video_In_Buffer.v
- Altera_UP_Video_In_Deinterlacer.v
- Altera_UP_Video_In_Resize.v
- Altera_UP_YCrCb_422_to_444_Converter.v
- Altera_UP_YCrCb_to_RGB_Converter.v
- Dual_Clock_FIFO.v
- Video_In.v

Audio_PLL.v

full_project.pin

I2C_AV_Config.v

project_top.vhd

Reset_Delay.v

Source Code Description

AUDIO_DAC_FIFO.v
- The audio core to be added as a component in SOPC.

FIFO_16_256.v
- Instantiate a 256 B FIFO to hold the data being read, in this case, from the SD card.

I2C_Controller.v
- Used in conjunction with the I2C_AV_Config.v to initialize the audio and video codec.

vga_adapter.v
- Implementation of VGA adapter in 320x240 resolution as oppose to the default 640x480.

vga_address_translator.v
- Converts the user defined resolution into an address memory to store the image data.

vga_controller.v
- Reads the pixel data line-by-line from the memory address defined by the vga_address_translator.

vga_pll.v
- A PLL for the vga clock, its input is the 50 MHz clock and outputs a 25 MHz clock.

Altera_UP_ITU_R_656_Decoder.v
- Decodes the raw video data in ITU-R 656 format into the YCrCb-422 format.

Altera_UP_Video_In_Buffer.v
- A FIFO buffer used to store the data output from the Altera_UP_ITU_R_656_Decoder.v

Altera_UP_Video_In_Deinterlacer.v
- Deinterlace the NTSC video format to produce a progressive video frame.

Altera_UP_Video_In_Resize.v
- Resizes the video input from 640x480 to 320x240 resolution.

Altera_UP_YCrCb_422_to_444_Converter.v

- Converts the video data from YCrCb-422 to YCrCb-444 format.

Altera_UP_YCrCb_to_RGB_Converter.v

- Converts the video data from YCrCb-444 to RGB format which is bitmap compatible.

Dual_Clock_FIFO.v

- A FIFO buffer that supports different clock speed for reading and writing.

Video_In.v

- A module which calls all of the Altera_UP video components to take a raw video data and outputs them a 320x240 RGB format.

Audio_PLL.v

- A PLL for the audio clock which takes a 27 MHz clock input and divides it down to produce a 18.4 MHz clock for audio codec.

full_project.pin

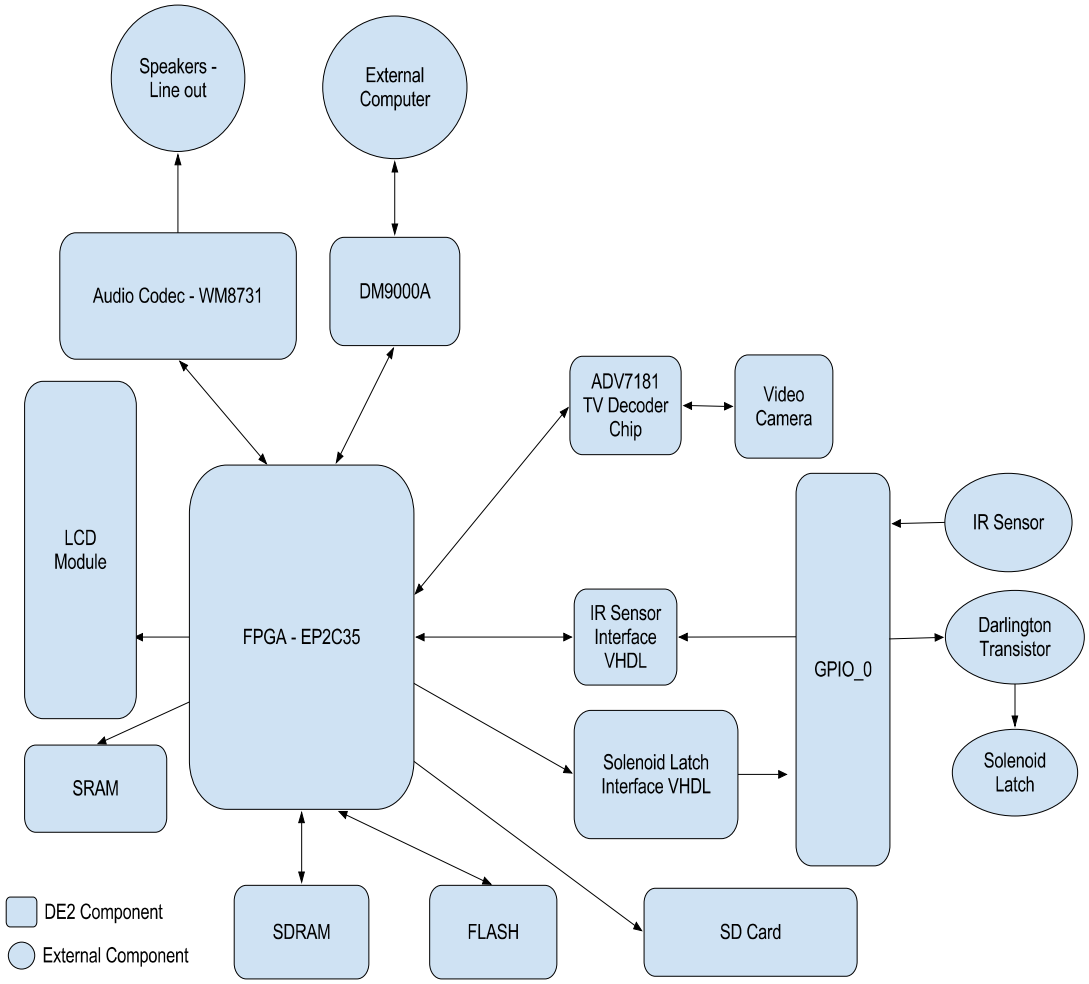
- A file containing an assignment for pins used in the DE2, not all pins are specified which will produce warnings in Quartus during the compilation.

I2C_AV_Config.v

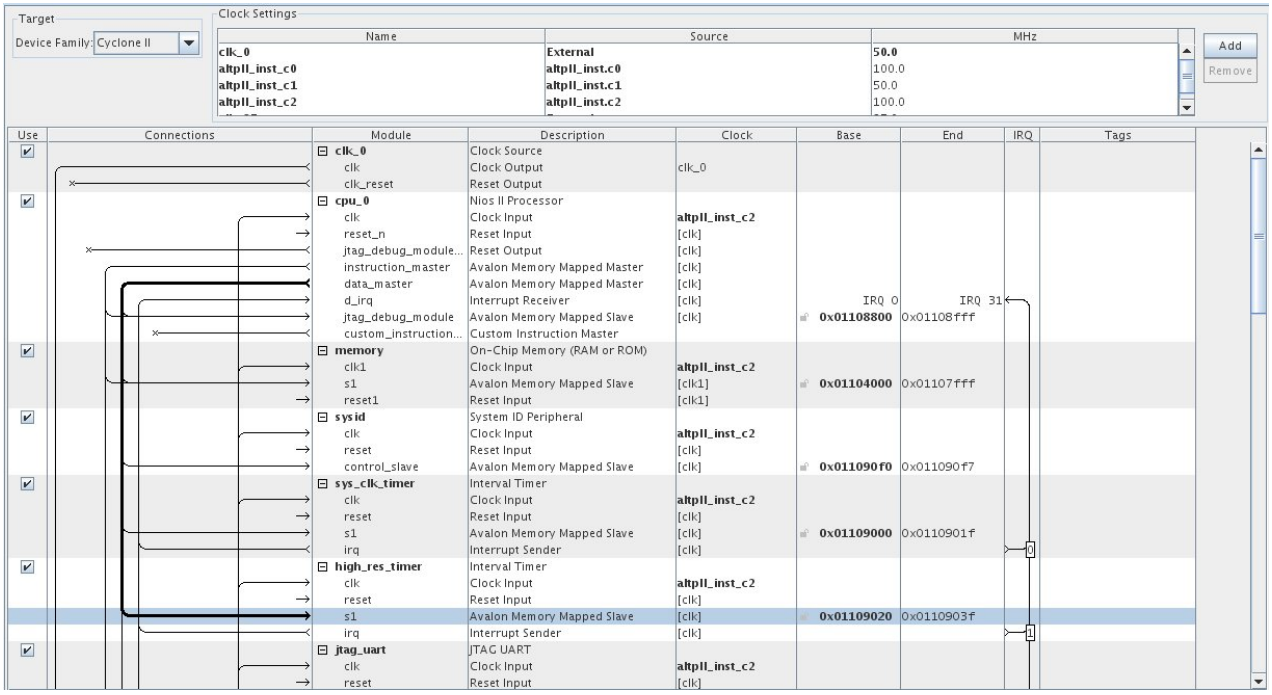
- Contains the base address of the components as well as their respective register values to be initialized. It is meant to be used in conjunction with the I2C_Controller.v file.

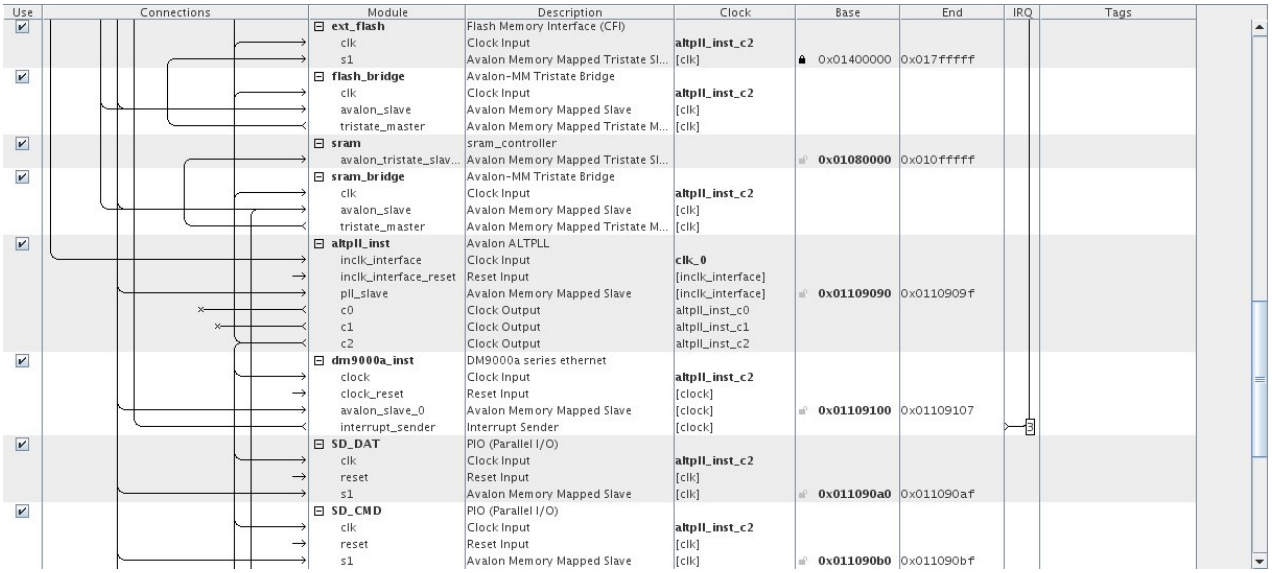
Appendix D: Hardware Documentation and System Schematic

See the below block diagram of the hardware showing all components.



Now see the system schematic via SOPC below:





Use	Connections	Module	Description	Clock	Base	End	IRQ	Tags
☑		**ext_flash**	Flash Memory Interface (CFI)					
		clk	Clock Input	**altpll_inst_c2**				
		s1	Avalon Memory Mapped Tristate Sl...	[clk]	0x01400000	0x017fffff		
☑		**flash_bridge**	Avalon-MM Tristate Bridge					
		clk	Clock Input	**altpll_inst_c2**				
		avalon_slave	Avalon Memory Mapped Slave	[clk]				
		tristate_master	Avalon Memory Mapped Tristate M...	[clk]				
☑		**sram**	sram_controller					
		avalon_tristate_slav...	Avalon Memory Mapped Tristate Sl...		**0x01080000**	0x010fffff		
☑		**sram_bridge**	Avalon-MM Tristate Bridge					
		clk	Clock Input	**altpll_inst_c2**				
		avalon_slave	Avalon Memory Mapped Slave	[clk]				
		tristate_master	Avalon Memory Mapped Tristate M...	[clk]				
☑		**altpll_inst**	Avalon ALTPLL					
		inclk_interface	Clock Input	**clk_0**				
		inclk_interface_reset	Reset Input	[inclk_interface]				
		pll_slave	Avalon Memory Mapped Slave	[inclk_interface]	**0x01109090**	0x0110909f		
		c0	Clock Output	altpll_inst_c0				
		c1	Clock Output	altpll_inst_c1				
		c2	Clock Output	altpll_inst_c2				
☑		**dm9000a_inst**	DM9000a series ethernet					
		clock	Clock Input	**altpll_inst_c2**				
		clock_reset	Reset Input	[clock]				
		avalon_slave_0	Avalon Memory Mapped Slave	[clock]	**0x01109100**	0x01109107		
		interrupt_sender	Interrupt Sender	[clock]			31	
☑		**SD_DAT**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090a0**	0x011090af		
☑		**SD_CMD**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090b0**	0x011090bf		

Use	Connections	Module	Description	Clock	Base	End	IRQ	Tags
☑		**dm9000a_inst**	DM9000a series ethernet					
		clock	Clock Input	**altpll_inst_c2**				
		clock_reset	Reset Input	[clock]				
		avalon_slave_0	Avalon Memory Mapped Slave	[clock]	**0x01109100**	0x01109107		
		interrupt_sender	Interrupt Sender	[clock]			31	
☑		**SD_DAT**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090a0**	0x011090af		
☑		**SD_CMD**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090b0**	0x011090bf		
☑		**SD_CLK**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090c0**	0x011090cf		
☑		**Audio_0**	AUDIO_DAC_FIFO					
		clk	Clock Input	**altpll_inst_c2**				
		avalon_slave_0	Avalon Memory Mapped Slave	[clk]	**0x01109108**	0x0110910b		
☑		**videoIn_inst**	videoIn					
		vidIn	Avalon Memory Mapped Master	[FastClock]				
		control	Avalon Memory Mapped Slave	[FastClock]	**0x011090d0**	0x011090df		
		27MHzClock	Clock Input	**clk_27**				
		FastClock	Clock Input	**altpll_inst_c2**				
		FastClock_reset	Reset Input	[FastClock]				
☑		**red_led_pio**	PIO (Parallel I/O)					
		clk	Clock Input	**altpll_inst_c2**				
		reset	Reset Input	[clk]				
		s1	Avalon Memory Mapped Slave	[clk]	**0x011090e0**	0x011090ef		
☑		**clk_27**	Clock Source					
		clk	Clock Output	clk_27				
		clk_reset	Reset Output					