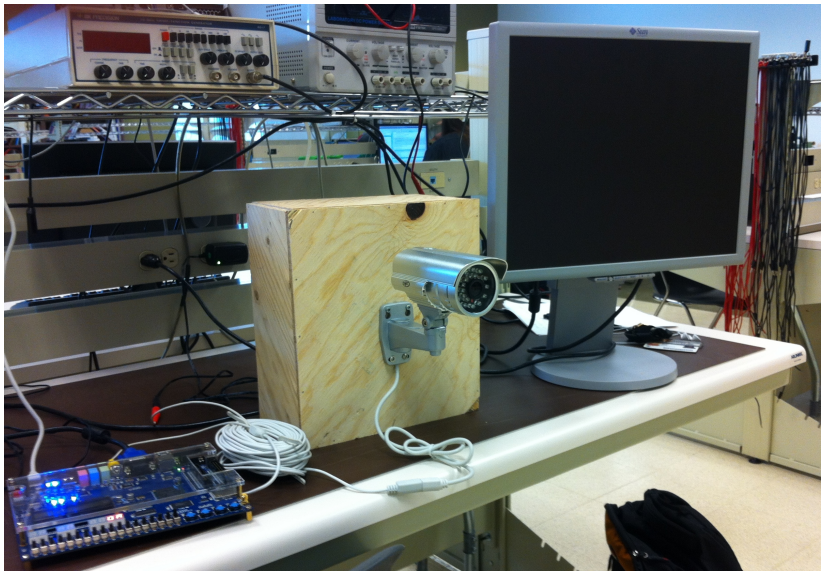
CMPE 450/490 DESIGN PROJECT

Intruder Alert System

Final Report



By:
Jordan Tymburski – tymbursk@ualberta.ca
Rachita Bhatia – rachita@ualberta.ca

Date: 4/13/2012

A security system that utilizes motion-tracking algorithms to detect intruders.

Abstract

Security is a vital issue that expands across many areas of life that require protecting vital information, goods, or areas from those who would want to trespass or steal. The primary objective of our project is to design a basic security system that would be able to track objects moving across the view of the camera. This would enable a security system that wouldn't require manpower until it detected an intrusion. It would also save space on local hard drives and backup devices since it would only need to record sequences of time when movement is detected. This concept will be implemented on the Altera DE2 board, which will be programmed to interface with an external VGA port and accept input through the onboard RCA female jack. The frames from a video camera will be received through the RCA female jack, processed using a micro C to detect motion in subsequent frames, and then the results will be sent out of the VGA port to a viewing screen. The design also implements FPGA design elements using VHDL to handle frame buffering in and out of the SDRAM. The micro C code can then access the frames from within the memory as needed. Our design uses as much FPGA programming as possible to attain a higher frame rate throughput, which results in a better, more accurate security system.

Contents

Functional Requirements

The project entails the design of an Intruder Alert System, which monitors an area using a camera, analyzes the captured video frames for detecting motion, and sends the resulting video feed to a screen. The system tracks motion by outlining any moving subjects with a red box. The practical applications of this project would be as a part of a larger security system, where monitoring may be done using multiple cameras to capture a 360 degree view.

The core functionality of the project involves interfacing with a camera for input, and with a VGA screen for output. The TV Decoder chip on the Altera DE2 board is then used for carrying out analysis based on the input to detect motion in real time, and for sending the output to the VGA port. Additional features that may be added to the project in future implementations are also discussed at the end of this section.

The three major components of our project and their functions are further explained below:

Monitoring via the Camera

Monitoring is done via a camera that takes pictures of its frame periodically and sends them to the TV Decoder chip ADV7181B on the Altera DE2 board for analysis. We used a camera with a Video Out port so that it can easily be interfaced with the Video In port of the DE2 board. The image frame received by the TV Decoder chip is then sent to the software side of our program, which is micro C code that implements a motion detection algorithm. It compares every subsequent frame with a static background frame and uses this comparison to detect motion. In our design, the movement from one frame to another is tracked by outlining the subjects with a red box. A more detailed explanation of our motion detection algorithm is given in the Software Design section of this report. The following pictures are an example of what our implementation looks like:

Motion Detection

The TV Decoder chip reads in each incoming frame from the camera and analyzes it by comparing against a background frame. The background frame can be set by pressing the frame-reset button on the DE2 board. To detect motion, we will then calculate the sum of absolute differences in pixel gray scale values between the background and the current frame, and compare that to a previously chosen threshold value. Once the sum is higher than that threshold, a red box outlining that corresponding area will appear on the image that is sent over to the VGA screen [1].

Screen Implementation

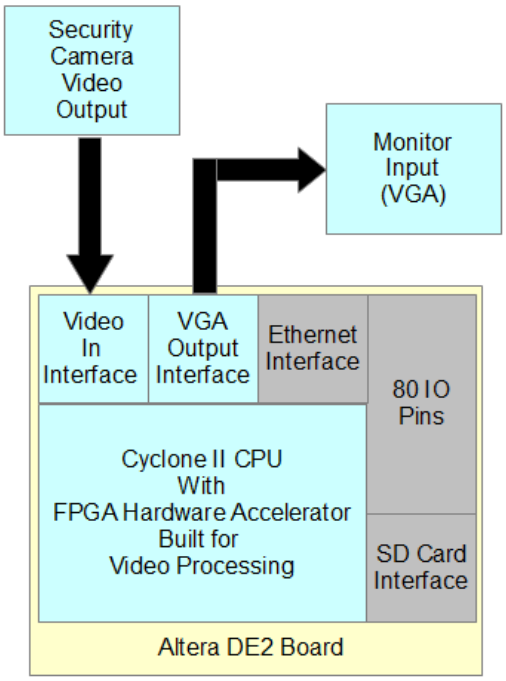
The resulting video after analysis is simultaneously sent over to the VGA screen for display. We are interfacing to a screen connected via a VGA port using the ADV7123 chip, because we can then potentially connect to any monitor or projector through VGA.

All the functional requirements of our project were met to a satisfactory degree. The Intruder Alert System does perform according to its specifications, and is able to detect any motion higher than its set threshold. It does, however, have a margin for further improvement, particularly in terms of accounting for light changes in the motion detection algorithm. Due to limited time, we could not implement any additional features as we had initially planned, however, these features are discussed in the Future Work section of this report.

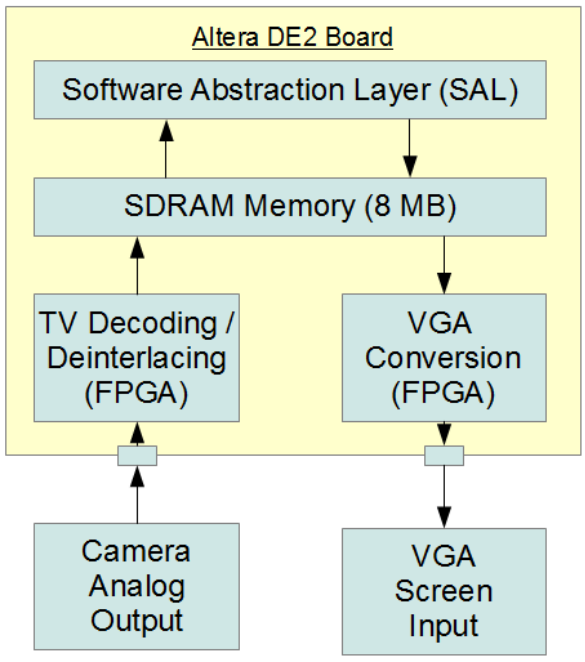
Design and Description of Operation

Core Implementation

The core implementation can be split into two main components: the security camera video input and the video output through the VGA interface. This is shown in the following block diagram:



The block diagram for the security system can be split into two primary components, the hardware and the software layers:

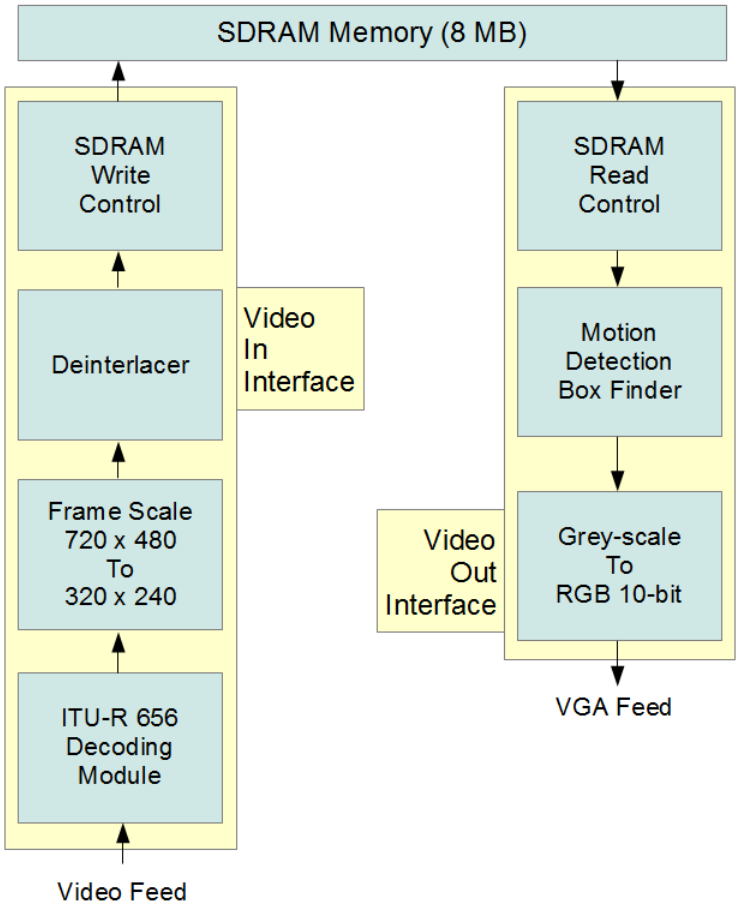


The system starts by receiving an NTSC video feed from the security camera through the video in interface. This is interfaced on the DE2 using the onboard video-in port with the onboard ADV7181B chip. The NTSC standard supports a maximum of 30 frames per second which is the targeted rate that the Cyclone II CPU will be able to process the data. The video feed will then be processed, which will be

discussed in the next section, before sending it to the software layer. At this point, the micro C software layer will then begin to algorithmically analyze the data frame by frame that was received from the video input. The goal is to use an algorithm that will be able to detect motion that travels across the video feed. After the software layer is done, the resulting frame data will be transferred into a separate location in the SDRAM where it then can be processed and outputted to the VGA DAC chip on the Altera board.

Hardware Layer

The hardware layer is split into two key components, the video in module and the video out module:



The video in interface takes the video in feed, which is decoded according to the ITU-R 656 standard, and converts it into frame data that the software layer can process. The first block extracts the Y component of the video from the serial feed. The Cb and Cr components (color luma values) are discarded since we only need the video in grayscale. From there, it enters a frame scaling routine that cuts the frame in half by removing every other x and y pixel. The y resolution is cut in half by only using the even frame from the ITU-R 656 standard. The x resolution is cut in half by just removing every other pixel from a line of data. The resulting frame resolution becomes 320 by 240 from dividing the height

and length of the frame by two. The new frame then enters the deinterlacer which takes on the task of determining where each pixel belongs by assigning it an x and y coordinate in the range of [0,0] to [320,240]. The pixels come with info bits that indicate end of line or end of even/odd frame to allow the deinterlacer to maintain the proper count. The resulting data is given to the circular buffer which has control over reads and writes to the SDRAM.

The circular buffer is a spinning buffer that points to three consecutive frame slots in memory. The first slot is the VGA read slot frame where the VGA reads the current frame data to print to the screen. The second consecutive slot is the frame which the software is currently executing motion detection on to find parts of the frame that don't match the reference frame, which is also stored in memory. The third slot is where the deinterlaced and decoded data from the TV analog input is stored into a new frame. This system shifts one slot when the software is finished with the motion detection algorithm. It continuously shifts in the allocated 8 megabytes of SDRAM with the software controlling when it needs to shift once. This design is required since there are three masters that are writing and reading from the SDRAM and this avoids conflicts on the reading or writing to the same location in memory.

For the SDRAM, multiple masters had to be added in the SOPC builder in order to allow the video in interface and the software layer both to read and write from memory. This is required since the SDRAM only has a single bus that needs to be arbitrated between calls for reading and writing. If the bus wasn't arbitrated, the reads and writes between the software layer and the video in interface would conflict which would result in incorrect data being read and written from memory. This system allows for the Avalon bus to deal with the arbitration between the memory masters instead of having to write separate VHDL code with an FSM machine to handle reads and writes to and from the SDRAM.

The VGA output is read from a location in the SDRAM that is specified by the circular buffer. This location contains the frame data that has already been processed by the software motion detection algorithm already. The pixel data is regular except for two special pixel values, 0xff and 0xfe. The software places these two pixel values when it wants to outline motion that has been detected. The pixel value 0xff corresponds to a red pixel (R=255, G=0, B=0) and the 0xfe corresponds to a green pixel (R=0, G=255,B=0). The remaining pixels are sent to the VGA DAC normally by setting the pixel value to the red register, the green register, and the blue register. The resulting image comes out as grayscale on the screen since the three registers are set the same. The shade of darkness is then indicated by the pixel value.

Finally, the system will interface with the external VGA port using the onboard ADV7123 chip. The data that needs to be displayed on the screen is stored in the SDRAM as an array that is 320x240 bytes which equals 76,800 bytes. This array is constantly updated by frames received from the software layer after the motion detection program has been implemented. This system allows for the VGA driver to just display the pixel values that are stored on the SDRAM. The VGA driver refreshes the displayed values as they are updated in the SDRAM by the software layer. There is no conversion required to RGB since we are printing to grayscale. The special cases where motion is detected and color needs to be printed to the screen is discussed in the previous paragraph.

Software Layer

The software layer will start by reading the frame values from the video in interface using a circular frame buffer. The circular frame buffer uses the SDRAM to store frames before the algorithm accesses them to detect motion.

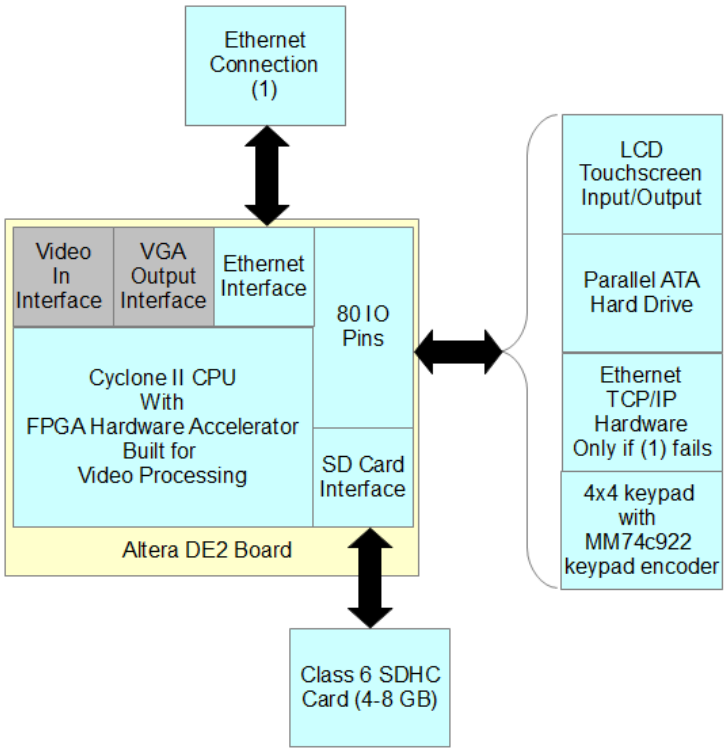
The algorithm that we plan to implement to detect motion requires 2 frames to process where movement occurred. [1] It requires the current frame and a background frame which is retrieved from a special location in the SDRAM pool of memory. The background frame is set by pressing a button on the DE2 which gives the software a reference frame to use. Then we compare the current frame with the background frame to determine where motion occurred. The algorithm checks blocks of pixels in the background frame against blocks in the current frame to determine absolute differences between pixels. If the absolute difference between blocks exceeds the threshold value, that block will be flagged as detected motion. Sections of frames where motion is detected are outlined in red and green before the frame is sent to the VGA port to be displayed on the screen. This would then be noticeable to anyone watching the video feed to where the algorithm detected motion and what caused the motion.

The algorithm basically works on the principle that the if there is motion in a particular frame, its grayscale pixel value would change in comparison to the background frame. We make use of this to find the absolute difference between the 2 values and then take an average sum of all the absolute differences over the frame. If this sum is greater than a set threshold, the moving subject is flagged by creating a red outline around the pixels whose values were found to be changed in comparison to the background frame. The algorithm is split into 5 different modules, each which are explained in detail in the software design section.

The algorithm is designed to use the onboard SDRAM to store any necessary frames for the algorithm computation. A small array of values that indicate if a block in a frame detected motion is also stored in the SRAM to help reduce the number of reads and writes to the SDRAM. The background frame, the current frame, and any other additional frames will be stored on the SDRAM to speed up the execution of the algorithm. The newly analyzed frame will also be put onto the SDRAM overwriting the original frame data from the TV input. After this is complete, the software sends a signal through the Avalon fabric to the circular buffer that controls reads and writes to the SDRAM to indicate that the buffer needs to shift to the next frame. A small timer is used in the software to slow down this process. Without this timer, the SDRAM bus gets flooded with too many requests which can cause the system to stop working.

Possible Additional Features

The possible additional features can be split into 4 sections: video compression and storage, menu system for the VGA screen to access stored data, Ethernet implementation to access the security system anywhere, and a touch screen interface to view and access the security data. This is shown in the following block diagram:



The first additional feature is the ability to store past data that was processed by the CPU. This would require first a storage medium and also a method to compress the frame data into video files that can be stored. The storage medium that we want to use is the SD card interface with the onboard SD interface. For compression, we will try to utilize the MP4 compression algorithm to store the video.[8]

If the first additional feature is complete, our second feature is to create a menu system for the VGA screen that will be controlled by the 4x4 keypad using the MM74c922 keypad decoder. The menu will allow for past video data to be viewed and displayed on the screen as well as access to enabling and disabling the motion detection. It will also give a menu system that will allow for past alarms to be displayed and the ability to acknowledge current alarms.

For a third additional feature, we want to implement the Ethernet interface using the DM9000A onboard chip. If we cannot figure out how to interface with the onboard Ethernet, we plan to use an off board TCP/IP Ethernet hardware interface instead and connect it through the GPIO pins. Once we get an Ethernet port working, we want to implement a web server that will allow access to both video feed and to past data remotely. This would offer the ability to access and control the security system from anywhere in the world that is connected to the Internet.

For a final additional feature, we thought of interfacing with a touch screen. This touch screen would allow full access and control to the system. The purpose for the touch screen would be an interface that could be put up near the entrance or exit of the secured area to allow disabling/enabling the security system or alarms. It would also allow the current video feed to be viewed without going to the main

viewing screen or on the internet. This screen would be connected and interfaced through the onboard GPIO pins.

Parts List

We only had one part, aside from the DE2, that was used in our design:

1. VU500-C - SVAT Indoor/Outdoor Night Vision CCD Camera [5]

This camera provides colour images during the day and black and white when it gets dark. The built-in photo sensor activates the IR LEDs automatically to provide night vision in pitch black. Package includes camera, mounting bracket, power supply and 60 foot power/video cable with RCA video connection.

- Colour during the day
- Black & White (gray scale) at night
- IR LEDs turn on automatically at night to provide up to 50 feet night vision in complete darkness
- 1/4" Sony CCD
- 6mm fixed focus lens, approximately 60 degree viewing angle
- IP65 outdoor rated, can be used indoors
- Includes power supply, mounting bracket, decals, 60 foot video/power cable
- Video connection DIN at camera end, RCA male and barrel power at other end. RCA to BNC adapter included

Cost: $89.99
Order Status: sent the order to Nancy on Jan. 31, 2012
Link / Reference: http://www.aartech.ca/vu500-c-svat-long-range-night-vision-ccd-camera.html

Reusable Design Units

The primary three sets of reusable design units are the big three components that we could implement during our project.

The first is the VGA controller to help with printing out of the VGA port. The best that we found was one at opencores.org. [9]

Link: http://opencores.org/project,vga_lcd

It offers an expandable approach and lots of flexibility for controlling and accessing the LCD screen. This could really add as a feature for when we are programming the output port. This could provide better access times and a more thorough development of the VGA access. However, our current design currently works for our project so this could be counted as an extra addition.

Another block is the video compression algorithm that we may need if we implement video storage. [8]

Link: http://opencores.org/project,video_systems

This offers a great system since its working on compression in a pipelined interface. This would allow a much faster implementation of the video compression and save us a lot of time. Testing would be required though since the status of the online link is vague about how far along the project is.

For the ADV7181B chip, which controls the video input, we will be using some of the design ideas from an implementation on the Altera DE2 board from the University of Toronto [13]

Link: http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/vin/vin.html

This offers a working connection to the TV chip that we can modify to grayscale and change the buffering system to work with our design. Parts of this design were included in our final design, with the most important one being the ITU-R 656 Decoder module (in Verilog).

Datasheet

The following tables represent the top level signals that represent the user input and output. The input power for the camera may change depending on what time of camera used to plug in the analog input. The TV input and the VGA output are expanded below in the video in and the video out interface that indicate how the chips onboard interpret the signals using the ADV7123 and the ADV7181B respectively.

Top Level User I/O Signals		
Name	**Description**	**Type**
KEY(0)	System Reset	Input
KEY(3)	Set Background Frame	Input
SW(0)	Switches the software motion detection on and off	Input
VGA	Video Output to VGA Monitor	Output
Analog In (TV)	The TV input from an analog in connection from a camera	Input
DE2 Power	Requires 9V, 1.3A	N/A
Camera Power	Requires 12V, 500mA	N/A

The Video In interface is an internal FPGA module that takes the TV data from the ADV7123 chip and converts it to a grayscale pixel value with a corresponding x and y coordinate. This module executes the necessary deinterlacing, decoding and frame resizing before outputting the processed pixel data.

Video In Interface		
Signal	**Description**	**Type**
TD_D [0…7]	TV Data	Input
CLOCK_27	27 MHz clock used to control the TV decoding	Input
TD_RESET	The TV chip reset control	Output
TD_HS	The Horizontal Size indication	Input
TD_VS	The Vertical Size indication	Input
I2C_SCLK	The Clock for the I2C interface	Input
I2C_SDAT	The Serial data to the I2C interface that controls the TV chip	Inout
Reset	Module based reset – receives signal from KEY(0)	Input
Pixel [7:0]	Greyscale pixel value	Output
X [8:0]	The x value of the pixel – maximum of 320	Output
Y [7:0]	The y value of the pixel – maximum of 240	Output
Pixel_EN	If the pixel is good (rising edge)	Output

This module is also an internal FPGA module that controls the data access to the SDRAM to read out the data for the VGA as well as the VGA controller to properly print data to the screen. The outputs plug directly into the VGA ADV7181B chip that manages the final stage of printing to the output display. The SDRAM signals below are sent to the Avalon fabric which handles arbitration for read and write access.

Video Out Interface		
Signal	**Description**	**Type**
Clk25	25 MHz clock to control the VGA	Input
Clk50	50 MHz clock for testing	Input
Clk100	100 MHz clock to control read access to the SDRAM	Input
i_reset_n	Module Reset	Input
Sdram_x	The X coordinate to read – max 320	Output
Sdram_y	The Y coordinate to read – max 240	Output
Sdram_Read	Read enable for SDRAM	Output
Sdram_Data	The read data from the SDRAM	Input
Sdram_Ready	Input to signal if the sdram is ready to read	Input
VGA_R	Red pixel value to VGA	Output
VGA_G	Green Pixel value to VGA	Output
VGA_B	Blue Pixel value to VGA	Output
VGA_BLANK	Blanking signal to VGA	Output
VGA_CLK	VGA Clock – will be set to 25 MHz	Output
VGA_HS	Horizontal Position Control for VGA	Output
VGA_SYNC	VGA Sync for VGA	Output
VGA_VS	Vertical Position Control for VGA	Output

This module, the circular buffer, is the third remaining primary module (aside from the SOPC connections) that was used in the design. This was also entirely concealed within FPGA and therefore doesn't have any power specifications. It handled the read and write address control for the SDRAM for the three sections: TV, software, and VGA.

Circular Buffer		
Signal	**Description**	**Type**
Clk	Circular buffer clock control	Input
Reset_N	Module Reset	Input
Control_device	0 for software disabled, 1 for enabled	Input
Tv_base_addr	The base tv address in memory	Output
Tv_enable	The enable to control when tv address is valid.	Output

Software_finished	This bit shifts the buffer by 1 on a rising edge. – software controlled	Input
Software_base_addr	The software base address in memory	Output
Software_ready	The control to set the software finished back to 0 when this goes high	Output
Vga_base_addr	The Base address for the VGA output	Output
Set_background_n	Set the background frame	Input
Background_frame_addr	The Background frame base address in memory	Output

This is the set of signals that make up the three primary modules of the project. There are no other external signals that use power or the GPIO pins which can be measured. The only requirement to run this project is to turn on the board which will automatically load the code into SRAM and begin the hardware/software process. The signals for the internal FPGA access cannot be measured for peak, idle, or standby power.

Background Reading

A Color Video Camera Using FPGA Video Processor [14]

By: Yee-Lu Zhaog, Dar-chang Juang, Chun-Hsein Horng

Notes:

This provided the initial ideas on how to design connecting the video from the TV port to receive streaming data. Although the reference is dated, the actual design ideas are usable. It discusses the NTSC encoder as well as video processing in the pixel form (Y and chromiance). It gave an initial starting point to begin working on the FPGA design and how to make it faster.

State Machine Design Techniques for Verilog and VHDL [15]

By: Steve Golson

Notes:

This was very important since memory is the foundation of our project. After a discussion with Professor Elliott and Nancy, we realized that this project needed to arbitrate memory in order to have multiple synchronous reads and writes happening on only one memory bus at any given time. This paper helped with the design of the VHDL state machine that is going to be used to arbitrate the bus on the SRAM. This also helped with ideas for working with managing the SDRAM as well. It also introduced design times and how much time may be lost in between synchronous reads and writes.

A motion adaptive deinterlacing method with hierarchical motion detection algorithm

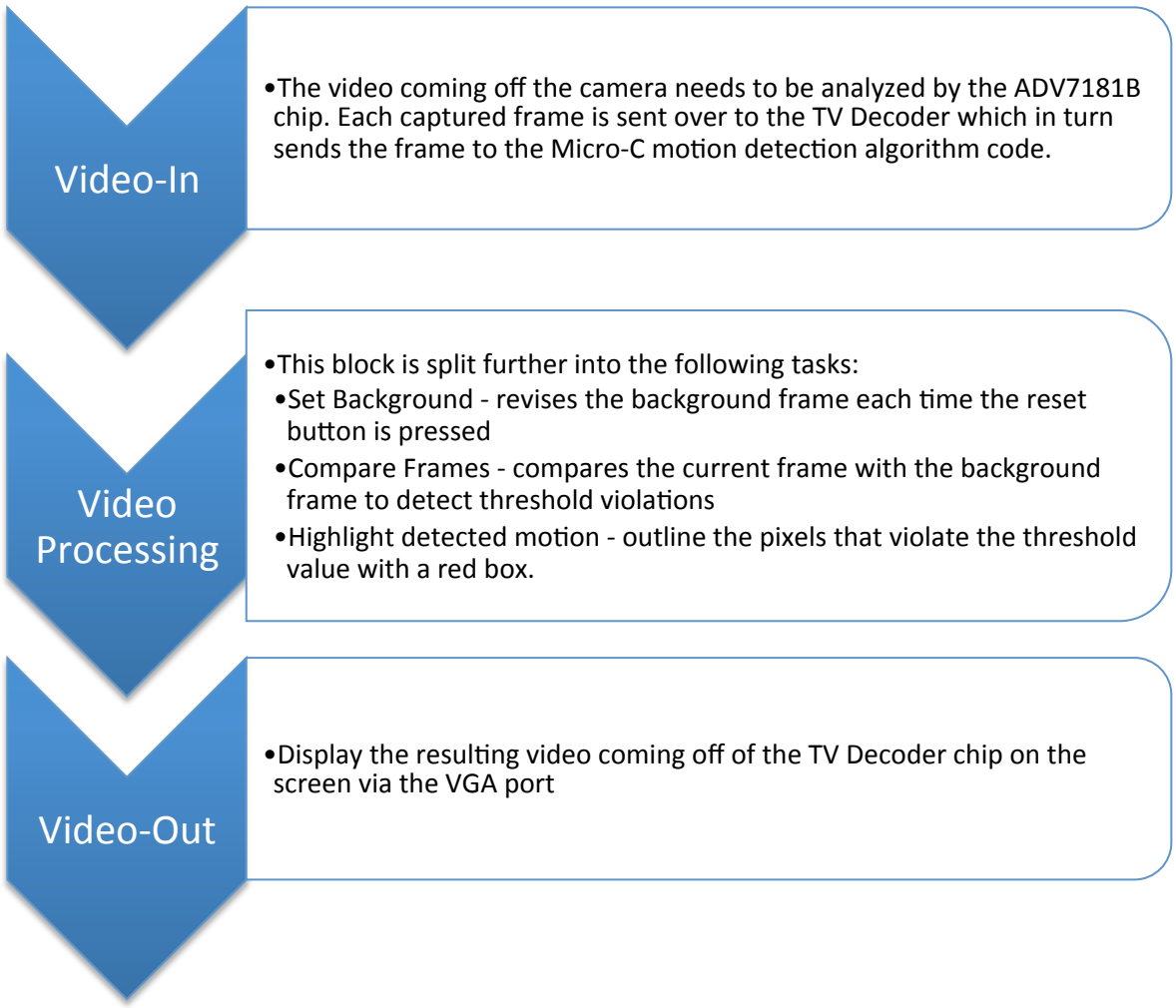
By: E. Shahinfard, M.A. Sid- Ahmed, M. Ahmadi

Notes:

This paper appears in the 15th IEEE International Conference in October 2008 [17]. It provided us with a better understanding of how we can divide our image into slots and detect motion in those slots before moving up a hierarchical level and highlighting the detected motion for the entire image. Even though the article uses a different algorithm for motion detection and implements a recursive approach to the different hierarchical levels of an image, it was helpful in the context of sub-dividing an image into blocks and analyzing them first.

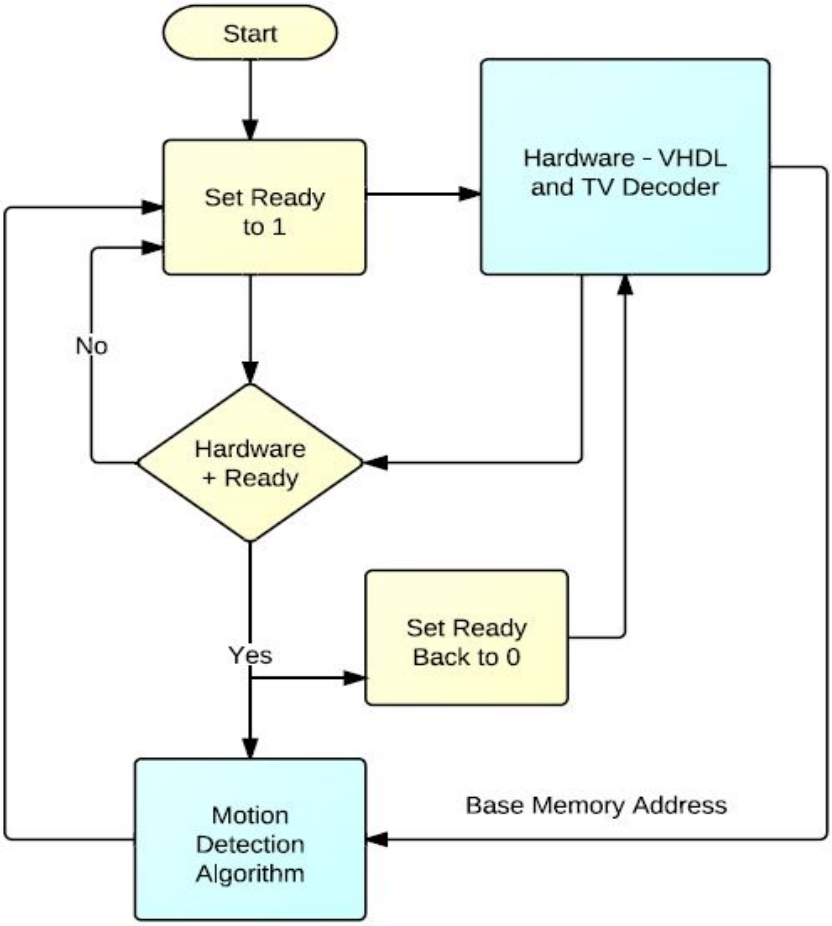
Software Design

The software design of our project essentially involves the set up of three major tasks:



Our design code is basically divided into the following modules:

- **Building the Array:** Once the camera read in the current frame and pushed it on to the hardware, the software side was responsible for building an array to compare the frame with the background. This was done by reading the base memory address of the array from the SDRAM and sending it over to the MicroC code. In place modification was done to the array in SDRAM by the algorithm for comparison and outlining. This is explained in the following sub points. The figure below is an overview of how an array is built by the software:



- **Comparison**: The next major task is to carry out the comparison between the 2 frames. As explained in the design section, this is done by calculating the absolute difference of the gray scale pixel values between the two 2-dimensional frame arrays:

$$[abs(\text{backgroundFrame[pixel\_i\_grayscale\_value]} - \text{currentFrame[pixel\_(i\_(grayscale\_value)]}$$

Our design does not take into account the day/night changes and minor light changes in the background. Therefore, we only have a fixed threshold value in regards to our comparison of gray scale values. This means that each time the absolute difference between a background pixel and a current frame pixel is more than 16 shades of gray, then a variable keeping count of the number of violated pixels is increased by 1.

- **Creation of Image Slots**: We divide the main frame into several smaller slots (squares), each of dimensions 10x10 pixels. This is done to account for another threshold - that of the total number of violated pixels. If the total number of violated pixels in a slot is less than 10% of the area of that slot, then that slot is not flagged and the error in the pixel values is ignored as it is too small. If the total is greater than 10% of the slot area, however, then that slot is flagged. The flagging was done by building another array, which was smaller in size as compared to the main array: 32x24. Each element of this

array represents a 10x10 slot in the current frame. The value of that element is set to a 0 or a 1 depending on whether the slot is flagged or not. This array is then helpful in checking adjacency and outlining the slots, which is explained below.

- **Outlining the Slots**: Once all the slots with detected motion have been flagged, the next step is to check which slots are adjacent to each other. This is done because we do not want to outline all moving parts of a single subject with multiple small red boxes; we aim to have one bigger red box for the entire moving body instead. Adjacency is checked by traversing through the smaller array and checking which elements have a neighbor set to 1. The ones that do, the shared boundaries for them are not outlined. Only the non-shared boundaries of adjacent slots are outlined with red. In this way the entire moving subject is outlined with a single box.

The languages that we have used to design the Intruder Alert System include MicroC and HDL. All our code has been archived and submitted along with this report.

Test Plan

Hardware

The hardware testing can be split into sections for each part that we will be interfacing on the Altera DE2 board. Since most of our hardware is onboard, we won't need to test if the hardware works and instead test that the interface for reading/writing to the onboard chips is working. The devices that we will be testing is the ADV7181B (TV Decoder Chip), the ADV7123 (VGA output chip), and the interface with the SDRAM.

The functional requirements testing will be setting up test benches that compare the connected signals against what the data sheets specify. This includes checking each signal and writing across them. These tests will be built in software as their own MicroC implementation and do full tests. For the VGA, the test will be visual since we can't actually confirm that data is being written to the screen.

The process of writing to the SDRAM is controlled by the Avalon memory fabric for arbitration but the address control is done by a separate circular buffer that we wrote. This section was included in the testing since it's a very essential feature in the design. To do this, we implemented read and write tests to the SDRAM to create simulated frames of a specific pattern that would then be read at the other end and displayed to the VGA. Testing of this section occurred after the VGA section was completed which allowed us to view the success or failure of the SDRAM components. Timing was also tested to determine how much of the SDRAM bus time was being used by the TV in, VGA out, and software

processing. This section required the longest amount of time due to the fact that if something goes wrong here, it will result in very unpredictable errors.

The testing of the FPGA code can also be split into sections, one for the incoming data processing and one for the exit data processing. The incoming will be tested by taking the incoming feed, putting it into the frame buffer, and outputting single frames through the JTAG interface in MicroC. Using this system, we can then see the pixel values of actual frames to confirm that the proper data is actually coming through. For the outgoing FPGA module, a test module will be used to simulate printing a frame to the screen. It will produce a specified pattern that can be changed by selecting different buttons on the Altera DE2. This will confirm two things: that the video out module is working and that the SDRAM reading into an alternating line by line FIFO system is working properly.

Software

One of the major tests on the software side includes testing the hardware-independent components in order to ensure that the signals are coming in properly. This means testing all the read and write signals available in the 'system.h' and confirming that they are working according to the specification.

For the individual software tasks, we will be performing tests to check for both memory leaks by watching the memory and making sure there are no deadlocks that cause the synchronization of the tasks to fall apart. To test this, we will be running the system for prolonged periods of time and watching the results over time.

Testing of the software modules was done by the creation of simultaneous test suites alongside the motion detection algorithm code that was initially written in C. Therefore, testing was done in C by creating two-dimensional arrays filled with random char values from 0 to 255 representing the grey scale pixel values of the current and the background images. The elements at each row and column were then compared to each other and absolute differences between their values were calculated, which were then compared to the thresholds. Slots were accordingly flagged and outlined by changing the boundaries to a particular char value if the thresholds were exceeded,

We also tested the algorithm by sending in pixel values generated using MATLAB from still images taken from an external camera. Timing was tested with respect to different images and arrays to find out the optimum speed with which our algorithm could handle the current resolution of 320x240.

Overall Plan

We will be following a similar structure to what was discussed in the lecture: [11]

1. Unit Testing – Subsystem
    - The first sub system will be the TV input
    - The second sub system will by the VGA output interface

- The third will be the SDRAM FSM interface
- The fourth will be the software on a computer using GCC

2. Integration Testing – Groups of subsystems
   - Determine if our algorithms work
   - This will include connecting the TV and VGA together
   - Testing the SDRAM for valid reads and writes on the 100 MHz clock.
   - And testing the software on the DE2 board.

3. System Testing
   - Check that our whole system works together as a whole.
   - Check that it meets requirements
   - We will test motion detection limitations and requirements by moving through the camera feed.
   - This will be the last test, ran near the end of the term
   - Here, we connect everything together and test.

4. Acceptance Testing
   - Ran by Nancy or Elliott to determine if our design conforms to the expectations.

Results of Experiments and Characterization

Frame Sizes and Color Scheme

640 x 480 = 307, 200 pixels

320 x 240 = 76,800 pixels

Greyscale: 8 bits (1 byte) – 256 bit grayscale value.

Color: 24 bits (3 bytes) – 256 R, B, and G values.

Frame Analysis

Frame Resolution	Greyscale or Color	Frame Size (bytes)	# of Frames in 4 MB SDRAM *	# of Frames in 512 KB SRAM
640 x 480	Color	921,600 bytes	4	0
640 x 480	Greyscale	307,200 bytes	13	1

320 x 240	Color	230,400 bytes	18	2
320 x 240	Greyscale	76,800 bytes	54	6

* Assuming 4 MB of SDRAM used out of the total 8 MB for frame buffers. The remaining 4 MB would be used for the program stack and heap.

FPGA Requirements Analysis

Onboard Total Logic Elements: 33,216

Onboard FPGA Memory: 483,840 bits

TV Compilation

Logic Elements Usage: 1% (400/33216)

Memory Bits Usage: 2% (8956/483,840)

VGA Compilation

Logic Elements Usage: 1% (200/33216)

Memory Bits Usage: 2% (8192/483,840)

CPU Compilation (For Software/MicroC implementation)

Logic Elements Usage: 12% (4089/33216)

Memory Bits Usage: 41% (195,968 / 483,840)

Embedded Multipliers 6% (4/70)

PLL: 1/4 (25%)

* This is currently using the fastest processor available on the DE2

Integrated Circuit Design

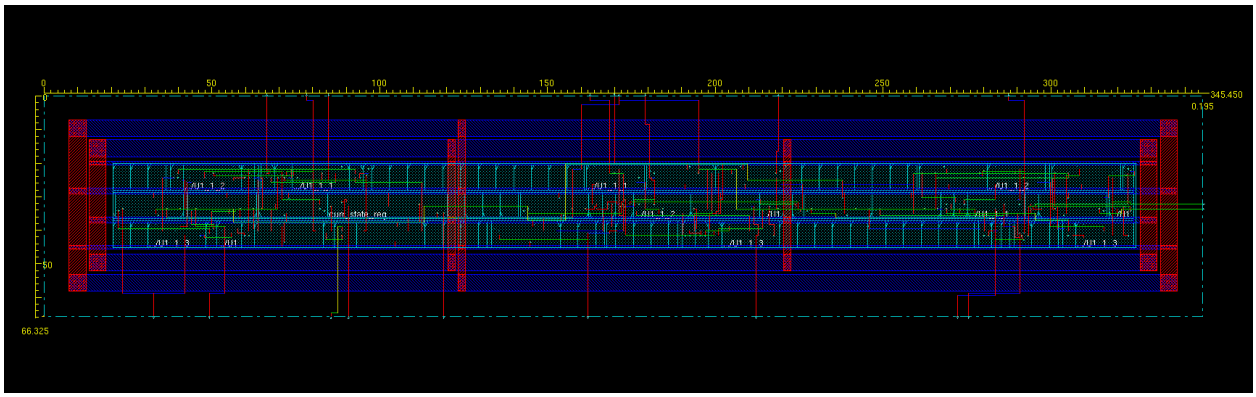
We decided to implement a counter and a circular buffer that we used in our project for creating an Integrated Circuit Design. The files “counter.vhd” and “circular_buffer.vhd” were used, and are attached in Appendix F. The code was initially compiled and synthesized in Synopsys, and the synthesis area, power and speed reports were generated. The timing constraint used for the Synthesis was 10ns. The performance of this IC was a little bit worse in comparison to the FPGA as there were a number of slack and hold time violations that needed to be fixed for a timing constraint of 10 ns, whereas the FPGA ran flawlessly at 100 MHz.

The reports generated during synthesis, along with a script showing the commands used to generate the reports, are attached in Appendix E.2 and E.1 respectively. Finally, a .v and a .sdc file was generated as a result of the synthesis process, and these files are attached in Appendix E.3.

Following the Synthesis, Place & Routing of the circuit design was carried out in Encounter. A command log file “encounter.cmd” shows the details of the P&R process and is attached in Appendix E.4. During place & routing, timing reports were generated at the following intervals to detect and fix hold time violations:

- Pre CTS Timing Report (Appendix E.5) – circular_buffer_preCTS.summary, circular_buffer_preCTS_all.tarpt, circular_buffer_preCTS.slk
- Hold Time Report Post CTS (AppendixE.6) – circular_buffer_postCTSHold.summary, circular_buffer_postCTSHold_all.tarpt, circular_buffer_postCTSHold.slk
- In Place Optimization Post CTS to fix hold time violations (Appendix E.7) – circular_buffer_postCTSHold_ipo.summary, circular_buffer_postCTSHold_ipo_all.tarpt, circular_buffer_postCTSHold_ipo.slk
- Post Routing (Appendix E.8) – circular_buffer_postRoute.summary, circular_buffer_postRoute_all.tarpt, circular_buffer_postRoute.slk
- At the end, for the final setup (Appendix E.9) – circular_buffer_finalsetup.summary, circular_buffer_finalsetup_all.tarpt, circular_buffer_finalsetup.slk

A snapshot of the final Integrated Circuit Design is shown in below:



References

[1] A. Kirillov, “Motion Detection Algorithms”, *The Code Project, http://www.codeproject.com/Articles/10248/Motion-Detection-Algorithms,* Date Accssed: February 5, 2012.

[2] “*DM9000A Ethernet Controller with General Purpose Interface - Datasheet*”, DAVICOM Semiconductor Inc., Version: DM900A-DS-P03, April 21, 2005.

[3]”*Multiformat SDTV Video Decoder - ADV7181B*”, Analog Devices Inc, 2005.

[4] ”*CMOS, 240 MHz Triple 10-Bit High Speed Video DAC- ADV7123*”, Analog Devices Inc, 1998.

[5] “VU500-C - SVAT Indoor/Outdoor Night Vision CCD Camera”, AARTech Canada, *http://www.aartech.ca/vu500-c-svat-long-range-night-vision-ccd-camera.html,* Date Accssed: February 5, 2012.

[6] “Multi-touch LCD Module (MTL)” Terasic Technologies Inc*., http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=653*, Date Accessed: February 5, 2012.

[7] “MM74C922 - Datasheet”, Fairchild Semiconductor, October 1987, Revised April 2001.

[8] R. Herveille and A. Henson, “Video Compression Systems” OpenCores.org, *http://opencores.org/project,video_systems*, Date Accessed: February 5, 2012

[9]R. Herveille and Hoffer, “VGA/LCD COntroller” OpenCores.org, *http://opencores.org/project,vga_lcd*, Date Accessed: February 5, 2012

[10] S. Nawaz et al, “VOIP Project: Final Report.” Internet: *http://www.cs.columbia.edu/~sedwards/classes/2009/4840/reports/VOIP.pdf*, May 16, 2009 [Feb 5, 2012].

[11] D. Eliott. CMPE 490. Class Lecture, Topic: “Project Testing.” GSB 211, Faculty of Engineering, University of Alberta, Edmonton, Alberta, Feb 2, 2012.

[12] E. Hamilton. “JPEG File Interchange Format.” Internet: *http://www.jpeg.org/public/jfif.pdf*, September 1, 1992 [March 6, 2012].

[13] University of Toronto. “Video-in Controller.” Internet: *http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/vin/vin.html*, August 29, 2008 [February 29, 2012].

[14] Y.L. Zhaog, D.C. Juang, and C.H. Horng. (1991, Sept.). “A color video camera using FPGA video processor.” *ASIC Conference and Exhibit, 1991 Proceedings., Fourth Annual IEEE International.* [On-line].

pp. P16-7/1-4. Available: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=242879* [March 7, 2012].

[15] S. Golson. "State Machine Design Techniques for Verilog and VHDL." Internet: *http://mail.littlepondfarm.com/pdf/golson_snug94.pdf* , 1994 [Feb. 28, 2012].

[16] "MPEG Video Copression Technique - a brief discussion". Internet*: http://vsr.informatik.tu-chemnitz.de/~jan/MPEG/HTML/mpeg_tech.html*, Accessed on March 7, 2012.

[17] E. Shahinfard; M.A. Sid- Ahmed, M. Ahmadi, "A motion adaptive deinterlacing method with hierarchical motion detection algorithm" *Image Processing 2008.* [December 12, 2008]

Appendix A: Quick Start Manual

There can be two cases when starting up the project: either the project has been flashed to the board and is available when the power button is pressed or the board is empty and needs to be programmed with the code available online.

Case 1: Board has been flashed with the design

Since the board already has the design programmed in, the setup is fairly easy.

1. Plug the VGA cord from an LCD monitor into the Altera DE2 board.
2. Turn on the LCD monitor
3. Plug the composite (yellow cable) connection from an NTSC camera source into the Altera DE2 board.
4. Turn on the camera (before turning on the board).
5. Press the red power button on the DE2 board to start it up.

That's the extent of the initial setup.

Case 2: The DE2 board has not been flashed with the design (the default DE2 case – especially if this is the first time you are running this design).

In this case, the design needs to be programmed to the board that you are running.

1. Make sure to start that you start by following the steps in case 1 which include plugging the devices to the DE2 board and then turning on the board.
2. Now, untar the project that was available in the same location as this report
3. Inside the 'capstone_main' directory is a directory called scripts which has the three necessary files for running the project. In each file, you need to change the following variables to the appropriate paths on your machine: QUARTUS,NIOS2_IDE,SOPC_KIT_NIOS2,and JTAGCONFIG.
4. Once these are changed, 'cd' into 'capstone_main' and run './scripts/launch_quartus.sh'
5. This will open the familiar quartus interface where you can now program the design to your DE2 using the Programmer.
6. After this is complete, go back to the terminal where you launched quartus and run './scripts/launch_nios2_ide.sh'
7. Clean, build and run as nios II hardware the project 'motion_detection'.

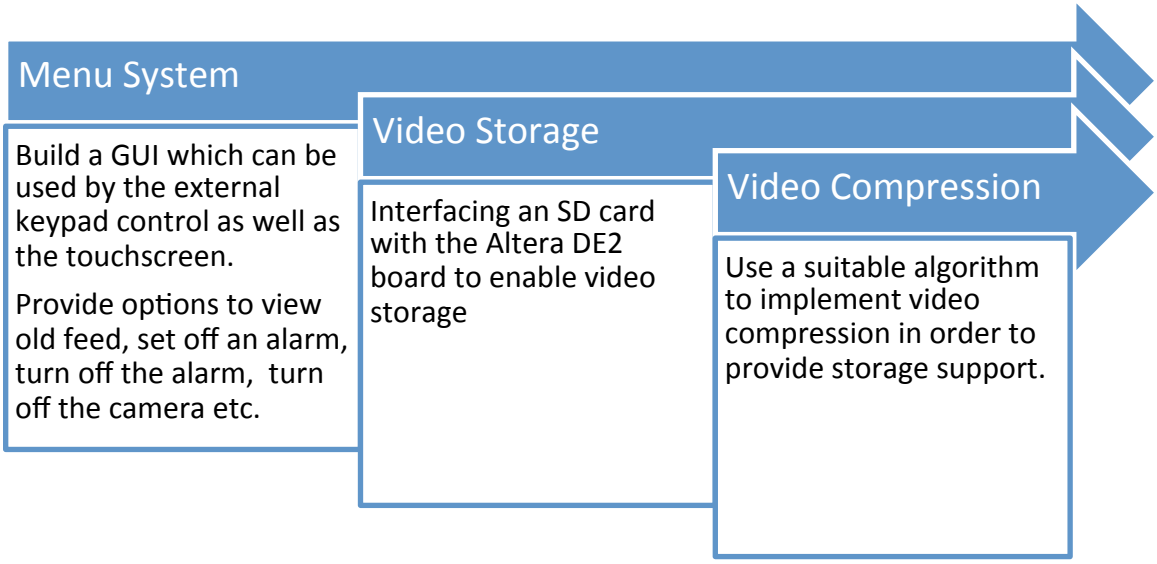
Now, both the software and hardware portion of the design is now programmed to your DE2.

After programming is complete and the project is running, the following list is the available features/buttons that can be pressed during operation.

1. KEY(0) is the reset button. Press and hold it to restart the project
2. SW(0) is to enable motion detection. It needs to be flipped on (up) for detection to be enabled. At this point, the frame rate will be approximately 8 frames per second.
3. KEY(3) sets the background reference frame that the algorithm uses to compare against the current frame. However, this button won't set the background frame if SW(0) is in the off position (which will mean motion detection is disabled).

Appendix B: Future Work

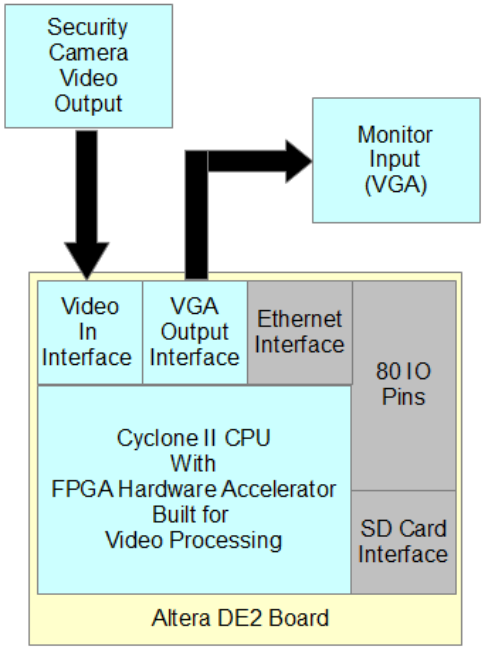
Due to a limited amount of time, our final design could not include all the features that we had initially planned to add. Therefore, future implementations of our project could improve upon our current design and modify it to include the following features:



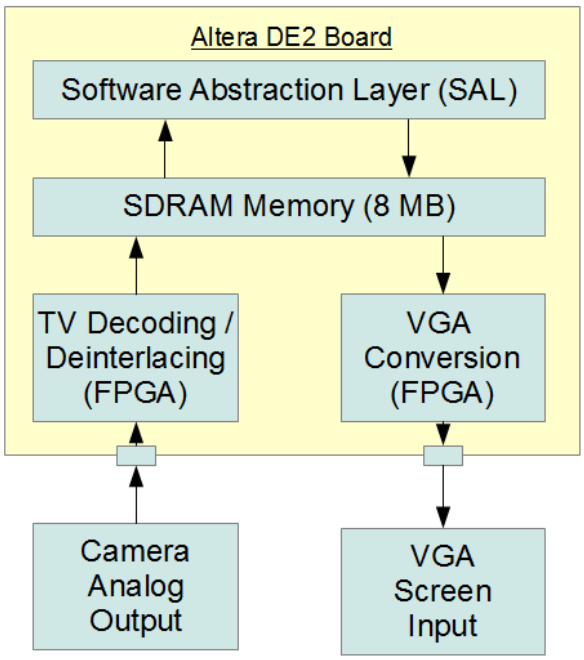
- Extend the functionality from grey scale to color display.
- Improve upon our existing motion detection algorithm to account for day/night changes.
- Increase the video quality from 320x240 to 640x480.
- Achieve higher frames per second video feed.
- Implement an external keypad control for the VGA monitor.
- Extend the VGA screen and external keypad control functionality to a touch screen interface that would provide the user with options to control the camera and the video feed. The touch screen would display a menu with options that would allow the user to set off a manual alarm, view the feed, turn off the camera etc.
- Provide the user with options to view old videos that have been stored in memory. This would need video compression and storage as well. Compression algorithms may be used, such as the Discrete Cosine Transform, where values above a certain frequency are rejected to give result to a compressed form of the initial video [16].
- Finally, create a web server for the Intruder Alert System so that it can be accessed remotely.

Appendix C: Hardware Documentation

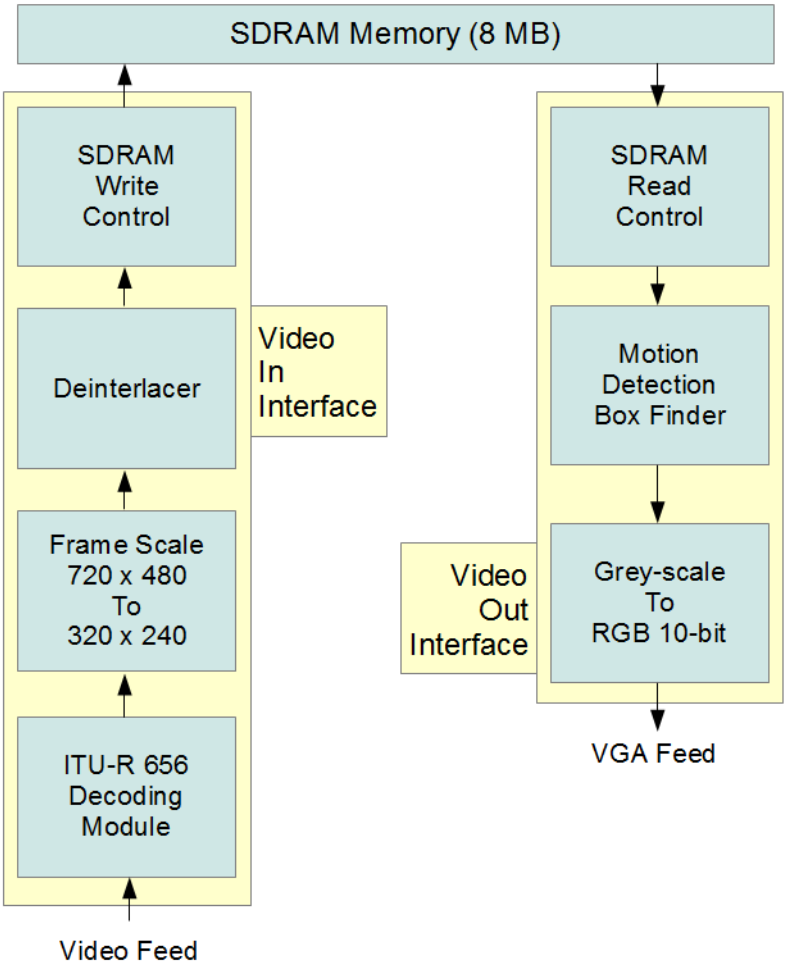
Primary Features



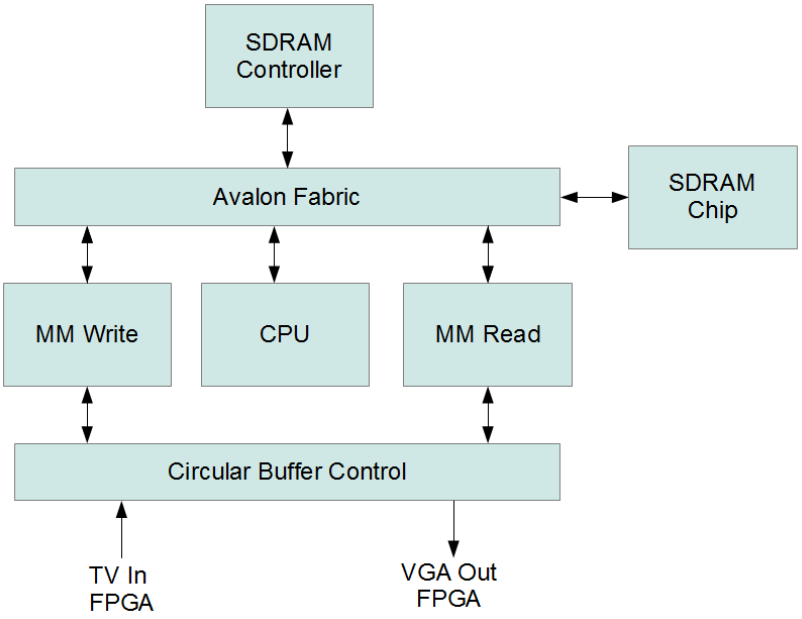
Main Design



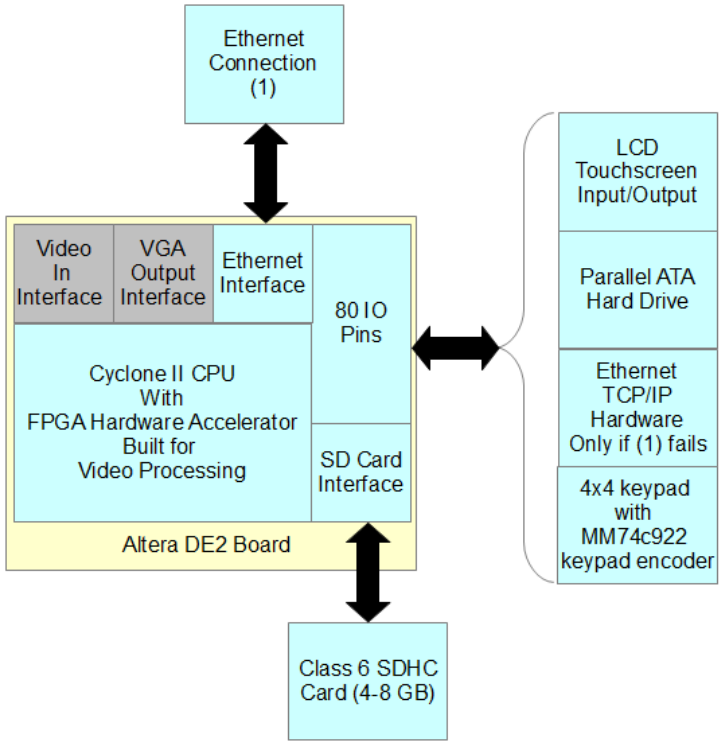
Hardware Design



SDRAM Design

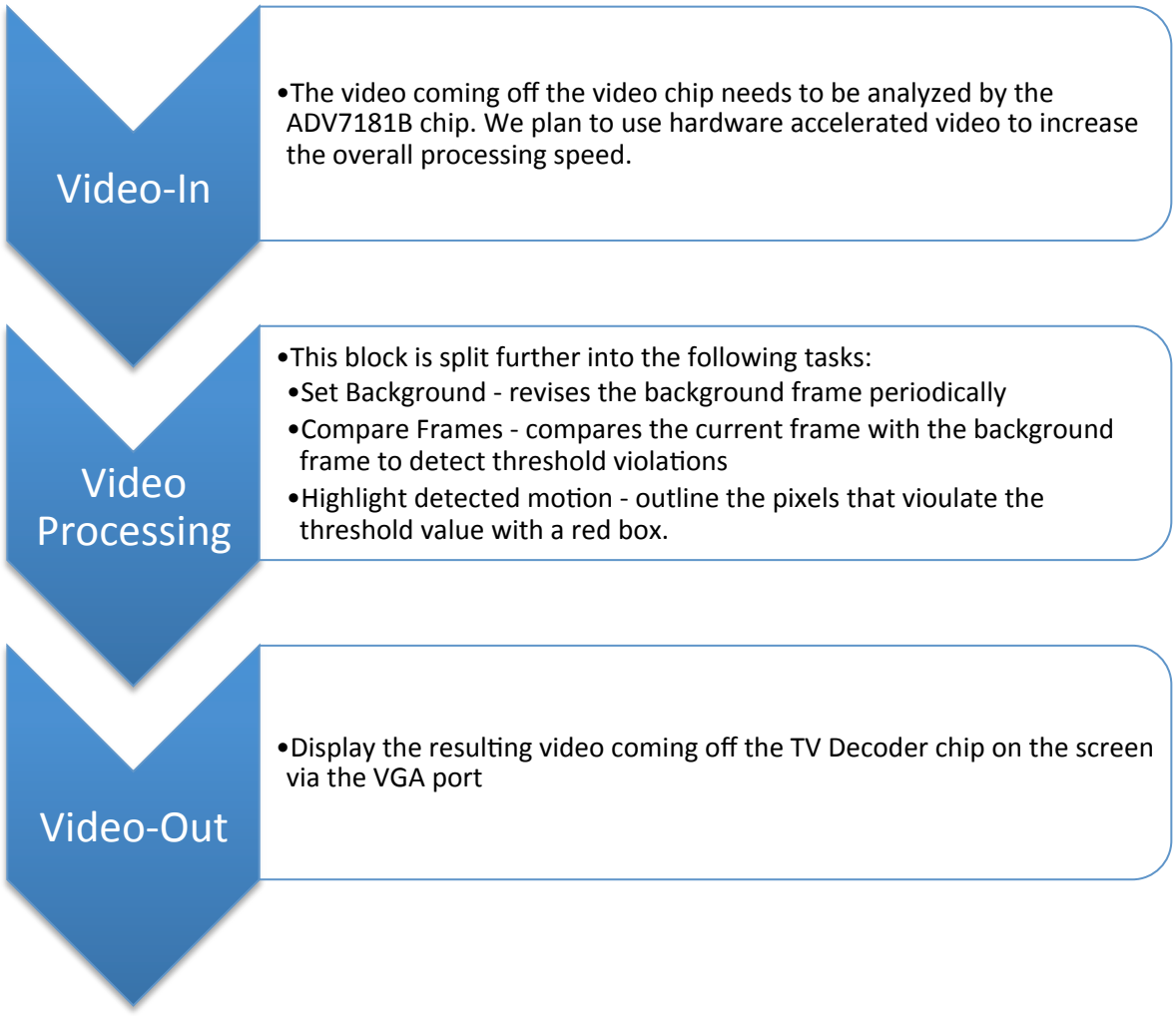


Additional Features

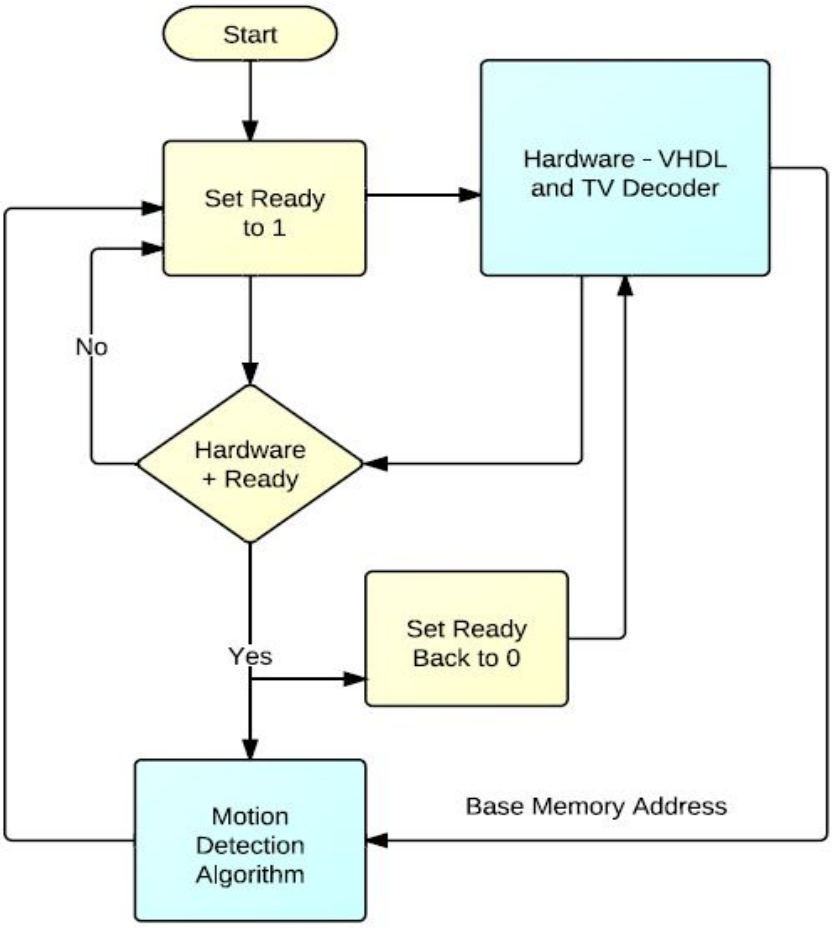


Appendix D: Software Documentation

Primary Software Tasks:



An overview of how the software interacts with the hardware:



Appendix E: Integrated Circuit Design Results

E.1 Script for synthesizing the IC in Synopsys: compile_circular_buffer.tcl

```
gui_start
analyze -format vhdl ./counter.vhd
analyze -format vhdl ./circular_buffer.vhd
elaborate circular_buffer
check_design -multiple_designs
uniquify
create_clock "clk" -period 10
current_design circular_buffer
compile -map_effort high -boundary_optimization
report_area
report_power
report_timing -path full -delay max -max_paths 1 -nworst 1
write_sdc ./Synthesized/circular_buffer.sdc
write -format verilog -hierarchy -output ./Synthesized/circular_buffer.v
```

E.2 Area (size), Power and Timing (speed) Reports generated in Synopsys after Synthesis

```
****************************************
Report : area
Design : circular_buffer
Version: Y-2006.06-SP4
Date   : Fri Apr 13 11:55:00 2012
****************************************
Library(s) Used:  GSCLib_2.0 (File: /EDA/kits/gpdk18/GSCLib_3.0/timing/GSCLib_3.0.db)

Number of ports:            74
Number of nets:             24
Number of cells:            4
Number of references:       4

Combinational area:         0.000000
Noncombinational area:      0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:        0.000000
Total area:             undefined
Loading db file '/EDA/kits/gpdk18/GSCLib_3.0/timing/GSCLib_3.0.db'
Loading db file '/EDA/kits/gpdk18/GSCLib_IO_1.4/timing/GSCLib_IO.db'
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: Design has unannotated primary inputs. (PWR-414)
```

```
Warning: Design has unannotated sequential cell outputs. (PWR-415)


****************************************
Report : power
      -analysis_effort low
Design : circular_buffer
Version: Y-2006.06-SP4
Date   : Fri Apr 13 11:55:05 2012
****************************************
Library(s) Used: GSCLib_2.0 (File: /EDA/kits/gpdk18/GSCLib_3.0/timing/GSCLib_3.0.db)

Operating Conditions: typical   Library: GSCLib_2.0
Wire Load Model Mode: top

Global Operating Voltage = 3
Power-specific unit information :
   Voltage Units = 1V
   Capacitance Units = 1.000000pf
   Time Units = 1ns
   Dynamic Power Units = 1mW    (derived from V,C,T units)
   Leakage Power Units = 1nW


 Cell Internal Power  = 201.5325 uW   (86%)
 Net Switching Power  =  31.5499 uW   (14%)
                         ---------
Total Dynamic Power    = 233.0825 uW  (100%)
Cell Leakage Power     =  11.9846 nW



****************************************
Report : timing
      -path full
      -delay max
      -max_paths 1
Design : circular_buffer
Version: Y-2006.06-SP4
Date   : Fri Apr 13 11:55:12 2012
****************************************
Operating Conditions: typical   Library: GSCLib_2.0
Wire Load Model Mode: top

  Startpoint: software_finished
              (input port)
  Endpoint: curr_state_reg
            (rising edge-triggered flip-flop clocked by clk)
```

```
 Path Group: clk
 Path Type: max

 Point                                    Incr      Path
 ---------------------------------------------------------------------------
 clock (input port clock) (rise edge)     0.00      0.00
 input external delay                     0.00      0.00 f
 software_finished (in)                   0.00      0.00 f
 curr_state_reg/D (SDFFSRX1)              0.00      0.00 f
 data arrival time                        0.00

 clock clk (rise edge)                    10.00     10.00
 clock network delay (ideal)              0.00      10.00
 curr_state_reg/CK (SDFFSRX1)             0.00      10.00 r
 library setup time                       -0.14      9.86
 data required time                       9.86
 ---------------------------------------------------------------------------
 data required time                       9.86
 data arrival time                        0.00
 ---------------------------------------------------------------------------
 slack (MET)                              9.86
```

E.3 Resulting Files after Synthesis: circular_buffer.sdc and circular_buffer.v

-- circular_buffer.sdc

```
################################################################################

# Created by write_sdc on Fri Apr 13 10:36:16 2012

################################################################################
set sdc_version 1.6

create_clock [get_ports clk]  -period 10  -waveform {0 5}
```

-- circular_buffer.v

```
module counter_0_DW01_inc_0 ( A_4_, A_3_, A_2_, A_1_, A_0_, SUM_4_, SUM_3_,
      SUM_2_, SUM_1_, SUM_0_ );
 input A_4_, A_3_, A_2_, A_1_, A_0_;
 output SUM_4_, SUM_3_, SUM_2_, SUM_1_, SUM_0_;
 wire   \A[4] , \A[3] , \A[2] , \A[1] , \A[0] , \SUM[4] , \SUM[3] , \SUM[2] ,
       \SUM[1] , \SUM[0] , \carry[4] , \carry[3] , \carry[2] ;
 assign \A[4]  = A_4_;
```

```
  assign \A[3]  = A_3_;
  assign \A[2]  = A_2_;
  assign \A[1]  = A_1_;
  assign \A[0]  = A_0_;
  assign SUM_4_ = \SUM[4] ;
  assign SUM_3_ = \SUM[3] ;
  assign SUM_2_ = \SUM[2] ;
  assign SUM_1_ = \SUM[1] ;
  assign SUM_0_ = \SUM[0] ;

  ADDHX1 U1_1_3 ( .A(\A[3] ), .B(\carry[3] ), .CO(\carry[4] ), .S(\SUM[3] ) );
  ADDHX1 U1_1_2 ( .A(\A[2] ), .B(\carry[2] ), .CO(\carry[3] ), .S(\SUM[2] ) );
  ADDHX1 U1_1_1 ( .A(\A[1] ), .B(\A[0] ), .CO(\carry[2] ), .S(\SUM[1] ) );
  XOR2X1 U1 ( .A(\carry[4] ), .B(\A[4] ), .Y(\SUM[4] ) );
  INVX8 U2 ( .A(\A[0] ), .Y(\SUM[0] ) );
endmodule


module counter_1_DW01_inc_0 ( A_4_, A_3_, A_2_, A_1_, A_0_, SUM_4_, SUM_3_,
      SUM_2_, SUM_1_, SUM_0_ );
  input A_4_, A_3_, A_2_, A_1_, A_0_;
  output SUM_4_, SUM_3_, SUM_2_, SUM_1_, SUM_0_;
  wire   \A[4] , \A[3] , \A[2] , \A[1] , \A[0] , \SUM[4] , \SUM[3] , \SUM[2] ,
       \SUM[1] , \SUM[0] , \carry[4] , \carry[3] , \carry[2] ;
  assign \A[4]  = A_4_;
  assign \A[3]  = A_3_;
  assign \A[2]  = A_2_;
  assign \A[1]  = A_1_;
  assign \A[0]  = A_0_;
  assign SUM_4_ = \SUM[4] ;
  assign SUM_3_ = \SUM[3] ;
  assign SUM_2_ = \SUM[2] ;
  assign SUM_1_ = \SUM[1] ;
  assign SUM_0_ = \SUM[0] ;

  ADDHX1 U1_1_3 ( .A(\A[3] ), .B(\carry[3] ), .CO(\carry[4] ), .S(\SUM[3] ) );
  ADDHX1 U1_1_2 ( .A(\A[2] ), .B(\carry[2] ), .CO(\carry[3] ), .S(\SUM[2] ) );
  ADDHX1 U1_1_1 ( .A(\A[1] ), .B(\A[0] ), .CO(\carry[2] ), .S(\SUM[1] ) );
  XOR2X1 U1 ( .A(\carry[4] ), .B(\A[4] ), .Y(\SUM[4] ) );
  INVX8 U2 ( .A(\A[0] ), .Y(\SUM[0] ) );
endmodule


module counter_2_DW01_inc_0 ( A_4_, A_3_, A_2_, A_1_, A_0_, SUM_4_, SUM_3_,
      SUM_2_, SUM_1_, SUM_0_ );
  input A_4_, A_3_, A_2_, A_1_, A_0_;
  output SUM_4_, SUM_3_, SUM_2_, SUM_1_, SUM_0_;
  wire   \A[4] , \A[3] , \A[2] , \A[1] , \A[0] , \SUM[4] , \SUM[3] , \SUM[2] ,
```

```
      \SUM[1] , \SUM[0] , \carry[4] , \carry[3] , \carry[2] ;
  assign \A[4]  = A_4_;
  assign \A[3]  = A_3_;
  assign \A[2]  = A_2_;
  assign \A[1]  = A_1_;
  assign \A[0]  = A_0_;
  assign SUM_4_ = \SUM[4] ;
  assign SUM_3_ = \SUM[3] ;
  assign SUM_2_ = \SUM[2] ;
  assign SUM_1_ = \SUM[1] ;
  assign SUM_0_ = \SUM[0] ;

  ADDHX1 U1_1_3 ( .A(\A[3] ), .B(\carry[3] ), .CO(\carry[4] ), .S(\SUM[3] ) );
  ADDHX1 U1_1_2 ( .A(\A[2] ), .B(\carry[2] ), .CO(\carry[3] ), .S(\SUM[2] ) );
  ADDHX1 U1_1_1 ( .A(\A[1] ), .B(\A[0] ), .CO(\carry[2] ), .S(\SUM[1] ) );
  XOR2X1 U1 ( .A(\carry[4] ), .B(\A[4] ), .Y(\SUM[4] ) );
  INVX8 U2 ( .A(\A[0] ), .Y(\SUM[0] ) );
endmodule


module counter_0 ( count_up, reset_n, init_count_4_, init_count_3_,
      init_count_2_, init_count_1_, init_count_0_, count_4_, count_3_,
      count_2_, count_1_, count_0_ );
  input count_up, reset_n, init_count_4_, init_count_3_, init_count_2_,
       init_count_1_, init_count_0_;
  output count_4_, count_3_, count_2_, count_1_, count_0_;
  wire   N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, n9, n10, n3, n4;

  DFFSRX1 pre_count_reg_0_ ( .D(N12), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_0_) );
  DFFSRX1 pre_count_reg_1_ ( .D(N13), .CK(count_up), .RN(1'b1), .SN(reset_n),
      .Q(count_1_) );
  DFFSRX1 pre_count_reg_2_ ( .D(N14), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_2_) );
  DFFSRX1 pre_count_reg_3_ ( .D(N15), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_3_), .QN(n9) );
  DFFSRX1 pre_count_reg_4_ ( .D(N16), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_4_), .QN(n10) );
  AND2X1 U8 ( .A(N11), .B(n3), .Y(N16) );
  AND2X1 U9 ( .A(N10), .B(n3), .Y(N15) );
  AND2X1 U10 ( .A(N9), .B(n3), .Y(N14) );
  AND2X1 U11 ( .A(N8), .B(n3), .Y(N13) );
  AND2X1 U12 ( .A(N7), .B(n3), .Y(N12) );
  NAND4X1 U13 ( .A(count_1_), .B(count_0_), .C(count_2_), .D(n4), .Y(n3) );
  NOR2X1 U14 ( .A(n9), .B(n10), .Y(n4) );
  counter_0_DW01_inc_0 add_37 ( .A_4_(count_4_), .A_3_(count_3_),
      .A_2_(count_2_), .A_1_(count_1_), .A_0_(count_0_), .SUM_4_(N11),
      .SUM_3_(N10), .SUM_2_(N9), .SUM_1_(N8), .SUM_0_(N7) );
```

```
endmodule


module counter_1 ( count_up, reset_n, init_count_4_, init_count_3_,
      init_count_2_, init_count_1_, init_count_0_, count_4_, count_3_,
      count_2_, count_1_, count_0_ );
 input count_up, reset_n, init_count_4_, init_count_3_, init_count_2_,
       init_count_1_, init_count_0_;
 output count_4_, count_3_, count_2_, count_1_, count_0_;
 wire   N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, n9, n10, n3, n4;

 DFFSRX1 pre_count_reg_0_ ( .D(N12), .CK(count_up), .RN(1'b1), .SN(reset_n),
      .Q(count_0_) );
 DFFSRX1 pre_count_reg_1_ ( .D(N13), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_1_) );
 DFFSRX1 pre_count_reg_2_ ( .D(N14), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_2_) );
 DFFSRX1 pre_count_reg_3_ ( .D(N15), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_3_), .QN(n9) );
 DFFSRX1 pre_count_reg_4_ ( .D(N16), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_4_), .QN(n10) );
 AND2X1 U8 ( .A(N11), .B(n3), .Y(N16) );
 AND2X1 U9 ( .A(N10), .B(n3), .Y(N15) );
 AND2X1 U10 ( .A(N9), .B(n3), .Y(N14) );
 AND2X1 U11 ( .A(N8), .B(n3), .Y(N13) );
 AND2X1 U12 ( .A(N7), .B(n3), .Y(N12) );
 NAND4X1 U13 ( .A(count_1_), .B(count_0_), .C(count_2_), .D(n4), .Y(n3) );
 NOR2X1 U14 ( .A(n9), .B(n10), .Y(n4) );
 counter_1_DW01_inc_0 add_37 ( .A_4_(count_4_), .A_3_(count_3_),
      .A_2_(count_2_), .A_1_(count_1_), .A_0_(count_0_), .SUM_4_(N11),
      .SUM_3_(N10), .SUM_2_(N9), .SUM_1_(N8), .SUM_0_(N7) );
endmodule


module counter_2 ( count_up, reset_n, init_count_4_, init_count_3_,
      init_count_2_, init_count_1_, init_count_0_, count_4_, count_3_,
      count_2_, count_1_, count_0_ );
 input count_up, reset_n, init_count_4_, init_count_3_, init_count_2_,
       init_count_1_, init_count_0_;
 output count_4_, count_3_, count_2_, count_1_, count_0_;
 wire   N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, n7, n10, n3, n4;

 DFFSRX1 pre_count_reg_0_ ( .D(N12), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_0_) );
 DFFSRX1 pre_count_reg_1_ ( .D(N13), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_1_) );
 DFFSRX1 pre_count_reg_2_ ( .D(N14), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_2_) );
```

```
  DFFSRX1 pre_count_reg_3_ ( .D(N15), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_3_), .QN(n10) );
  DFFSRX1 pre_count_reg_4_ ( .D(N16), .CK(count_up), .RN(reset_n), .SN(1'b1),
      .Q(count_4_), .QN(n7) );
  AND2X1 U8 ( .A(N11), .B(n3), .Y(N16) );
  AND2X1 U9 ( .A(N10), .B(n3), .Y(N15) );
  AND2X1 U10 ( .A(N9), .B(n3), .Y(N14) );
  AND2X1 U11 ( .A(N8), .B(n3), .Y(N13) );
  AND2X1 U12 ( .A(N7), .B(n3), .Y(N12) );
  NAND4X1 U13 ( .A(count_1_), .B(count_0_), .C(count_2_), .D(n4), .Y(n3) );
  NOR2X1 U14 ( .A(n10), .B(n7), .Y(n4) );
  counter_2_DW01_inc_0 add_37 ( .A_4_(count_4_), .A_3_(count_3_),
      .A_2_(count_2_), .A_1_(count_1_), .A_0_(count_0_), .SUM_4_(N11),
      .SUM_3_(N10), .SUM_2_(N9), .SUM_1_(N8), .SUM_0_(N7) );
endmodule


module circular_buffer ( clk, reset_n, tv_base_addr_22_, tv_base_addr_21_,
      tv_base_addr_20_, tv_base_addr_19_, tv_base_addr_18_, tv_base_addr_17_,
      tv_base_addr_16_, tv_base_addr_15_, tv_base_addr_14_, tv_base_addr_13_,
      tv_base_addr_12_, tv_base_addr_11_, tv_base_addr_10_, tv_base_addr_9_,
      tv_base_addr_8_, tv_base_addr_7_, tv_base_addr_6_, tv_base_addr_5_,
      tv_base_addr_4_, tv_base_addr_3_, tv_base_addr_2_, tv_base_addr_1_,
      tv_base_addr_0_, tv_enable, software_finished, software_base_addr_22_,
      software_base_addr_21_, software_base_addr_20_, software_base_addr_19_,
      software_base_addr_18_, software_base_addr_17_, software_base_addr_16_,
      software_base_addr_15_, software_base_addr_14_, software_base_addr_13_,
      software_base_addr_12_, software_base_addr_11_, software_base_addr_10_,
      software_base_addr_9_, software_base_addr_8_, software_base_addr_7_,
      software_base_addr_6_, software_base_addr_5_, software_base_addr_4_,
      software_base_addr_3_, software_base_addr_2_, software_base_addr_1_,
      software_base_addr_0_, software_ready, vga_base_addr_22_,
      vga_base_addr_21_, vga_base_addr_20_, vga_base_addr_19_,
      vga_base_addr_18_, vga_base_addr_17_, vga_base_addr_16_,
      vga_base_addr_15_, vga_base_addr_14_, vga_base_addr_13_,
      vga_base_addr_12_, vga_base_addr_11_, vga_base_addr_10_,
      vga_base_addr_9_, vga_base_addr_8_, vga_base_addr_7_, vga_base_addr_6_,
      vga_base_addr_5_, vga_base_addr_4_, vga_base_addr_3_, vga_base_addr_2_,
      vga_base_addr_1_, vga_base_addr_0_ );
  input clk, reset_n, software_finished;
  output tv_base_addr_22_, tv_base_addr_21_, tv_base_addr_20_,
       tv_base_addr_19_, tv_base_addr_18_, tv_base_addr_17_,
       tv_base_addr_16_, tv_base_addr_15_, tv_base_addr_14_,
       tv_base_addr_13_, tv_base_addr_12_, tv_base_addr_11_,
       tv_base_addr_10_, tv_base_addr_9_, tv_base_addr_8_, tv_base_addr_7_,
       tv_base_addr_6_, tv_base_addr_5_, tv_base_addr_4_, tv_base_addr_3_,
       tv_base_addr_2_, tv_base_addr_1_, tv_base_addr_0_, tv_enable,
       software_base_addr_22_, software_base_addr_21_,
```

```
    software_base_addr_20_, software_base_addr_19_,
    software_base_addr_18_, software_base_addr_17_,
    software_base_addr_16_, software_base_addr_15_,
    software_base_addr_14_, software_base_addr_13_,
    software_base_addr_12_, software_base_addr_11_,
    software_base_addr_10_, software_base_addr_9_, software_base_addr_8_,
    software_base_addr_7_, software_base_addr_6_, software_base_addr_5_,
    software_base_addr_4_, software_base_addr_3_, software_base_addr_2_,
    software_base_addr_1_, software_base_addr_0_, software_ready,
    vga_base_addr_22_, vga_base_addr_21_, vga_base_addr_20_,
    vga_base_addr_19_, vga_base_addr_18_, vga_base_addr_17_,
    vga_base_addr_16_, vga_base_addr_15_, vga_base_addr_14_,
    vga_base_addr_13_, vga_base_addr_12_, vga_base_addr_11_,
    vga_base_addr_10_, vga_base_addr_9_, vga_base_addr_8_,
    vga_base_addr_7_, vga_base_addr_6_, vga_base_addr_5_,
    vga_base_addr_4_, vga_base_addr_3_, vga_base_addr_2_,
    vga_base_addr_1_, vga_base_addr_0_;
wire   next_state;
assign vga_base_addr_0_ = 1'b0;
assign vga_base_addr_1_ = 1'b0;
assign vga_base_addr_2_ = 1'b0;
assign vga_base_addr_3_ = 1'b0;
assign vga_base_addr_4_ = 1'b0;
assign vga_base_addr_5_ = 1'b0;
assign vga_base_addr_6_ = 1'b0;
assign vga_base_addr_7_ = 1'b0;
assign vga_base_addr_8_ = 1'b0;
assign vga_base_addr_9_ = 1'b0;
assign vga_base_addr_10_ = 1'b0;
assign vga_base_addr_11_ = 1'b0;
assign vga_base_addr_12_ = 1'b0;
assign vga_base_addr_13_ = 1'b0;
assign vga_base_addr_14_ = 1'b0;
assign vga_base_addr_15_ = 1'b0;
assign vga_base_addr_16_ = 1'b0;
assign vga_base_addr_22_ = 1'b0;
assign software_base_addr_0_ = 1'b0;
assign software_base_addr_1_ = 1'b0;
assign software_base_addr_2_ = 1'b0;
assign software_base_addr_3_ = 1'b0;
assign software_base_addr_4_ = 1'b0;
assign software_base_addr_5_ = 1'b0;
assign software_base_addr_6_ = 1'b0;
assign software_base_addr_7_ = 1'b0;
assign software_base_addr_8_ = 1'b0;
assign software_base_addr_9_ = 1'b0;
assign software_base_addr_10_ = 1'b0;
assign software_base_addr_11_ = 1'b0;
```

```
  assign software_base_addr_12_ = 1'b0;
  assign software_base_addr_13_ = 1'b0;
  assign software_base_addr_14_ = 1'b0;
  assign software_base_addr_15_ = 1'b0;
  assign software_base_addr_16_ = 1'b0;
  assign software_base_addr_22_ = 1'b0;
  assign tv_base_addr_0_ = 1'b0;
  assign tv_base_addr_1_ = 1'b0;
  assign tv_base_addr_2_ = 1'b0;
  assign tv_base_addr_3_ = 1'b0;
  assign tv_base_addr_4_ = 1'b0;
  assign tv_base_addr_5_ = 1'b0;
  assign tv_base_addr_6_ = 1'b0;
  assign tv_base_addr_7_ = 1'b0;
  assign tv_base_addr_8_ = 1'b0;
  assign tv_base_addr_9_ = 1'b0;
  assign tv_base_addr_10_ = 1'b0;
  assign tv_base_addr_11_ = 1'b0;
  assign tv_base_addr_12_ = 1'b0;
  assign tv_base_addr_13_ = 1'b0;
  assign tv_base_addr_14_ = 1'b0;
  assign tv_base_addr_15_ = 1'b0;
  assign tv_base_addr_16_ = 1'b0;
  assign tv_base_addr_22_ = 1'b0;
  assign next_state = software_finished;

  SDFFSRX1 curr_state_reg ( .D(next_state), .SI(1'b0), .SE(1'b0), .CK(clk),
      .SN(1'b1), .RN(reset_n), .Q(software_ready), .QN(tv_enable) );
  counter_2 f0 ( .count_up(software_ready), .reset_n(reset_n),
      .init_count_4_(1'b0), .init_count_3_(1'b0), .init_count_2_(1'b0),
      .init_count_1_(1'b0), .init_count_0_(1'b0),
      .count_4_(vga_base_addr_21_), .count_3_(vga_base_addr_20_),
      .count_2_(vga_base_addr_19_), .count_1_(vga_base_addr_18_),
      .count_0_(vga_base_addr_17_) );
  counter_1 f1 ( .count_up(software_ready), .reset_n(reset_n),
      .init_count_4_(1'b0), .init_count_3_(1'b0), .init_count_2_(1'b0),
      .init_count_1_(1'b0), .init_count_0_(1'b1),
      .count_4_(software_base_addr_21_), .count_3_(software_base_addr_20_),
      .count_2_(software_base_addr_19_), .count_1_(software_base_addr_18_),
      .count_0_(software_base_addr_17_) );
  counter_0 f2 ( .count_up(software_ready), .reset_n(reset_n),
      .init_count_4_(1'b0), .init_count_3_(1'b0), .init_count_2_(1'b0),
      .init_count_1_(1'b1), .init_count_0_(1'b0),
      .count_4_(tv_base_addr_21_), .count_3_(tv_base_addr_20_),
      .count_2_(tv_base_addr_19_), .count_1_(tv_base_addr_18_),
      .count_0_(tv_base_addr_17_) );
endmodule
```

E.4 Command log file for Encounter – encounter.cmd

```
#############################################################
#                                   #
#  Encounter Command Logging File           #
#  Created on Fri Apr 13 10:40:40 2012          #
#                                   #
#############################################################

#@(#)CDS: Encounter v09.11-s084_1 (32bit) 04/26/2010 12:41 (Linux 2.6)
#@(#)CDS: NanoRoute v09.11-s008 NR100226-1806/USR63-UB (database version 2.30, 93.1.1)
{superthreading v1.14}
#@(#)CDS: CeltIC v09.11-s011_1 (32bit) 03/04/2010 09:23:40 (Linux 2.6.9-78.0.25.ELsmp)
#@(#)CDS: CTE 09.11-s016_1 (32bit) Apr  8 2010 03:34:50 (Linux 2.6.9-78.0.25.ELsmp)
#@(#)CDS: CPE v09.11-s023

getenv ENCOUNTER_CONFIG_RELATIVE_CWD
setDoAssign
getIoFlowFlag
setUIVar rda_Input rel_c_thresh 0.01
setUIVar rda_Input ui_gndnet GRND
setUIVar rda_Input ui_cts_cell_list {buf inv}
setUIVar rda_Input ui_timingcon_file Synthesized/circular_buffer.sdc
setUIVar rda_Input ui_leffile {/EDA/kits/gpdk_MIET_2.0/GSCLib_3.0/lef/GSCLib_3.0.lef
/EDA/kits/gpdk_MIET_2.0/GSCLib_IO_1.4/lef/GSCLib_IO.lef}
setUIVar rda_Input ui_timelib {/EDA/kits/gpdk_MIET_2.0/GSCLib_IO_1.4/timing/GSCLib_IO.lib
/EDA/kits/gpdk_MIET_2.0/GSCLib_3.0/timing/GSCLib_3.0.lib}
setUIVar rda_Input ui_netlist {Synthesized/circular_buffer.v
/EDA/kits/gpdk_MIET_2.0/GSCLib_3.0/verilog/GSCLib_3.0_stub.v}
setUIVar rda_Input ui_topcell circular_buffer
setUIVar rda_Input ui_rel_c_thresh 0.01
setUIVar rda_Input ui_pwrnet POWR
commitConfig
fit
setDrawView fplan
getIoFlowFlag
setFPlanRowSpacingAndType 0.66 1
setBottomIoPadOrient R180
setIoFlowFlag 0
floorPlan -site CORE -r 0.1 0.7 20 20 20 20
uiSetTool select
getIoFlowFlag
fit
addRing -spacing_bottom 1 -width_left 5 -width_bottom 5 -width_top 5 -spacing_top 1 -layer_bottom
Metal1 -stacked_via_top_layer Metal6 -width_right 5 -around core -jog_distance 0.33 -offset_bottom 2
-layer_top Metal1 -threshold 0.33 -offset_left 2 -spacing_right 1 -spacing_left 1 -offset_right 2 -
offset_top 2 -layer_right Metal2 -nets {GRND POWR } -stacked_via_bottom_layer Metal1 -layer_left
Metal2
```

```
addStripe -block_ring_top_layer_limit Metal3 -max_same_layer_jog_length 0.6 -
padcore_ring_bottom_layer_limit Metal1 -set_to_set_distance 100 -stacked_via_top_layer Metal6 -
padcore_ring_top_layer_limit Metal3 -spacing 1 -xleft_offset 100 -xright_offset 100 -
merge_stripes_value 0.33 -layer Metal2 -block_ring_bottom_layer_limit Metal1 -width 2 -nets {GRND
POWR } -stacked_via_bottom_layer Metal1
getMultiCpuUsage -localCpu
setPlaceMode -fp false
placeDesign -prePlaceOpt
setDrawView place
getIoFlowFlag
clearGlobalNets
globalNetConnect GRND -type pgpin -pin GRND -inst *
globalNetConnect POWR -type pgpin -pin POWR -inst *
clearGlobalNets
globalNetConnect GRND -type pgpin -pin GRND -inst *
globalNetConnect POWR -type pgpin -pin POWR -inst *
clearGlobalNets
globalNetConnect GRND -type pgpin -pin GRND -inst *
globalNetConnect POWR -type pgpin -pin POWR -inst *
sroute -connect { blockPin padPin padRing corePin floatingStripe } -layerChangeRange { Metal1 Metal6 }
-blockPinTarget { nearestRingStripe nearestTarget } -padPinPortConnect { allPort oneGeom } -
checkAlignedSecondaryPin 1 -blockPin useLef -allowJogging 1 -crossoverViaBottomLayer Metal1 -
allowLayerChange 1 -targetViaTopLayer Metal6 -crossoverViaTopLayer Metal6 -targetViaBottomLayer
Metal1 -nets { GRND POWR }
trialRoute -maxRouteLayer 6
setDrawView place
clearClockDomains
setClockDomains -all
timeDesign -preCTS -idealClock -pathReports -drvReports -slackReports -numPaths 50 -prefix
circular_buffer_preCTS -outDir report
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -preCTS
clearClockDomains
setClockDomains -all
timeDesign -preCTS -idealClock -pathReports -drvReports -slackReports -numPaths 50 -prefix
circular_buffer_ipo -outDir report
addCTSCellList {CLKBUFX1 CLKBUFX2 CLKBUFX3 INVX1 INVX2 INVX4 INVX8}
clockDesign -genSpecOnly Clock.ctstch
clockDesign -specFile Clock.ctstch -outDir output -fixedInstBeforeCTS
trialRoute -maxRouteLayer 6 -highEffort
setDrawView place
clearClockDomains
setClockDomains -all
timeDesign -postCTS -pathReports -drvReports -slackReports -numPaths 50 -prefix circular_buffer_cts -
outDir report
clearClockDomains
setClockDomains -all
```

```
timeDesign -postCTS -hold -pathReports -slackReports -numPaths 50 -prefix circular_buffer_postCTShold
-outDir report
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS -drv
clearClockDomains
setClockDomains -all
timeDesign -postCTS -hold -pathReports -slackReports -numPaths 50 -prefix
circular_buffer_postCTShold_ipo -outDir report
getFillerMode -quiet
findCoreFillerCells
findCoreFillerCells
addFiller -cell FILL8 FILL4 FILL2 FILL1 -prefix FILL -markFixed
wroute -multiCpu 2
rcOut -setload circular_buffer.setload
rcOut -setres circular_buffer.setres
rcOut -spf circular_buffer.spf
clearClockDomains
setClockDomains -all
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths 50 -prefix
circular_buffer.finalsetup -outDir report
clearClockDomains
setClockDomains -all
timeDesign -postRoute -hold -pathReports -slackReports -numPaths 50 -prefix
circular_buffer_postRoute_hold -outDir report
saveNetlist circular_buffer.v
global dbgLefDefOutVersion
set dbgLefDefOutVersion 5.4
defOut -floorplan -netlist -routing output/circular_buffer.def
set dbgLefDefOutVersion 5.4
saveDesign circular_buffer.enc
extractRC
write_sdf -version 2.1 -precision 4 design.sdf
zoomBox -9.733 77.350 350.593 -16.600
uiSetTool ruler
panPage 1 0
panPage -1 0
panPage 1 0
panPage -1 0
fit
uiSetTool ruler
```

E.5 Pre CTS Setup Timing Reports (.summary, all.tarpt, .slk files)

-- circular_buffer_preCTS.summary

```
#####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:00:25 2012
#  Command:           timeDesign -preCTS -idealClock -pathReports -drvReport...
#####################################################################

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|    Setup mode|   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|     WNS (ns):|  9.859  |   N/A   |  9.859  |   N/A   |   N/A   |   N/A   |
|      TNS (ns):|  0.000  |   N/A   |  0.000  |   N/A   |   N/A   |   N/A   |
| Violating Paths:|    0    |   N/A   |    0    |   N/A   |   N/A   |   N/A   |
|          All Paths:|    2    |   N/A   |    2    |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |            Real               |       Total      |
|   DRVs         +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |     0 (0)        |   0.000    |     0 (0)        |
|   max_tran     |     0 (0)        |   0.000    |     0 (0)        |
|   max_fanout   |     0 (0)        |     0      |     0 (0)        |
+----------------+------------------+------------+------------------+

Density: 70.156%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

-- circular_buffer_preCTS_all.tarpt

```
#####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:00:25 2012
```

```
# Command:          timeDesign -preCTS -idealClock -pathReports -drvReports -slackReports -numPaths
50 -prefix circular_buffer_preCTS -outDir report
#####################################################################
Path 1: MET Setup Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/D  (v) checked with  leading edge of 'clk'
Beginpoint: software_finished (v) triggered by  leading edge of '@'
Path Groups:  {in2reg}
Other End Arrival Time         0.000
- Setup                    0.141
+ Phase Shift               10.000
= Required Time              9.859
- Arrival Time               0.000
= Slack Time                 9.859
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time          0.000
    Timing Path:
    +-------------------------------------------------------------------------------------+
    |   Instance    |       Arc        |   Cell   | Slew | Delay | Arrival | Required |
    |               |                  |          |      |       |  Time   |   Time   |
    |---------------+------------------+----------+------+-------+---------+----------|
    |               | software_finished v |          | 0.000 |       |   0.000 |    9.859 |
    | curr_state_reg | D v              | SDFFSRX1 | 0.000 | 0.000 |   0.000 |    9.859 |
    +-------------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time          0.000
    Other End Path:
    +-----------------------------------------------------------------------+
    |   Instance    | Arc |   Cell   | Slew | Delay | Arrival | Required |
    |               |     |          |      |       |  Time   |   Time   |
    |---------------+-----+----------+------+-------+---------+----------|
    |               | clk ^ |          | 0.000 |       |   0.000 |   -9.859 |
    | curr_state_reg | CK ^  | SDFFSRX1 | 0.000 | 0.000 |   0.000 |   -9.859 |
    +-----------------------------------------------------------------------+
Path 2: MET Recovery Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/RN (^) checked with  leading edge of 'clk'
Beginpoint: reset_n          (^) triggered by  leading edge of '@'
Path Groups:  {in2reg}
Other End Arrival Time         0.000
- Recovery                 0.068
+ Phase Shift               10.000
= Required Time              9.932
- Arrival Time               0.005
= Slack Time                 9.927
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time          0.000
    Timing Path:
```

```
+-------------------------------------------------------------------------+
|   Instance    |    Arc    |   Cell   |  Slew | Delay | Arrival | Required |
|               |           |          |       |       |  Time   |   Time   |
|---------------+-----------+----------+-------+-------+---------+----------|
|               | reset_n ^ |          | 0.000 |       |   0.000 |    9.927 |
| curr_state_reg | RN ^     | SDFFSRX1 | 0.012 | 0.005 |   0.005 |    9.932 |
+-------------------------------------------------------------------------+
Clock Rise Edge                   0.000
= Beginpoint Arrival Time         0.000
Other End Path:
+-------------------------------------------------------------------------+
|   Instance    | Arc |   Cell   |  Slew | Delay | Arrival | Required |
|               |     |          |       |       |  Time   |   Time   |
|---------------+-----+----------+-------+-------+---------+----------|
|               | clk ^ |        | 0.000 |       |   0.000 |   -9.927 |
| curr_state_reg | CK ^ | SDFFSRX1 | 0.000 | 0.000 |   0.000 |   -9.927 |
+-------------------------------------------------------------------------+
```

-- circular_buffer_preCTS.slk

```
# Format: clock  timeReq  slackR/slackF  setupR/setupF  instName/pinName   # cycle(s)
@(R)->clk(R)     9.859    */9.859        */0.141        curr_state_reg/D    1
@(R)->clk(R)     9.932    9.927/*        0.068/*        curr_state_reg/RN    1
```

E.6 Hold Time Reports Post CTS (.summary, all.tarpt, .slk files)

-- circular_buffer_postCTSHold.summary

```
#####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:05:43 2012
#  Command:           timeDesign -postCTS -hold -pathReports -slackReports -...
#####################################################################

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|    Hold mode       |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|         WNS (ns):| -0.035  |   N/A   | -0.035  |   N/A   |   N/A   |   N/A   |
|         TNS (ns):| -0.038  |   N/A   | -0.038  |   N/A   |   N/A   |   N/A   |
```

```
|   Violating Paths:|   2   |  N/A  |   2   |  N/A  |  N/A  |  N/A  |
|        All Paths:|   2   |  N/A  |   2   |  N/A  |  N/A  |  N/A  |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 69.933%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

-- circular_buffer_postCTSHold_all.tarpt

```
####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:05:44 2012
#  Command:           timeDesign -postCTS -hold -pathReports -slackReports -numPaths 50 -prefix
circular_buffer_postCTShold -outDir report
####################################################################
Path 1: VIOLATED Hold Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/D  (^) checked with  leading edge of 'clk'
Beginpoint: software_finished (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time          0.030
+ Hold                         0.005
+ Phase Shift                  0.000
= Required Time                0.035
  Arrival Time                 0.000
  Slack Time                  -0.035
    Clock Rise Edge                    0.000
    + Input Delay                      0.000
    = Beginpoint Arrival Time          0.000
    Timing Path:
    +-------------------------------------------------------------------------------------+
    |    Instance    |        Arc        |   Cell   |  Slew | Delay | Arrival | Required |
    |                |                   |          |       |       |  Time   |   Time   |
    |----------------+-------------------+----------+-------+-------+---------+----------|
    |                | software_finished ^ |          | 0.000 |       |   0.000 |    0.035 |
    | curr_state_reg | D ^               | SDFFSRX1 | 0.000 | 0.000 |   0.000 |    0.035 |
    +-------------------------------------------------------------------------------------+
    Clock Rise Edge                    0.000
    = Beginpoint Arrival Time          0.000
    Other End Path:
    +-----------------------------------------------------------------------+
    |    Instance    |    Arc     |   Cell   |  Slew | Delay | Arrival | Required |
    |                |            |          |       |       |  Time   |   Time   |
    |----------------+------------+----------+-------+-------+---------+----------|
    |                | clk ^      |          | 0.000 |       |   0.000 |   -0.035 |
    | clk__L1_I0     | A ^ -> Y v | INVX8    | 0.007 | 0.017 |   0.017 |   -0.018 |
```

```
    | clk__L2_I0    | A v -> Y ^ | INVX2    | 0.011 | 0.013 |   0.030 |   -0.005 |
    | curr_state_reg | CK ^      | SDFFSRX1 | 0.011 | 0.000 |   0.030 |   -0.005 |
    +-------------------------------------------------------------------------------+
Path 2: VIOLATED Removal Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/RN (^) checked with  leading edge of 'clk'
Beginpoint: reset_n          (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.030
+ Removal                    -0.021
+ Phase Shift                 0.000
= Required Time               0.009
  Arrival Time                0.006
  Slack Time                 -0.003
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +-------------------------------------------------------------------------------+
    |   Instance    |   Arc    |   Cell   |  Slew | Delay | Arrival | Required |
    |               |          |          |       |       |  Time   |   Time   |
    |---------------+----------+----------+-------+-------+---------+----------|
    |               | reset_n ^ |          | 0.000 |       |   0.000 |    0.003 |
    | curr_state_reg | RN ^     | SDFFSRX1 | 0.012 | 0.006 |   0.006 |    0.009 |
    +-------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time        0.000
    Other End Path:
    +-------------------------------------------------------------------------------+
    |   Instance    |   Arc    |   Cell   |  Slew | Delay | Arrival | Required |
    |               |          |          |       |       |  Time   |   Time   |
    |---------------+----------+----------+-------+-------+---------+----------|
    |               | clk ^    |          | 0.000 |       |   0.000 |   -0.003 |
    | clk__L1_I0    | A ^ -> Y v | INVX8    | 0.007 | 0.017 |   0.017 |    0.014 |
    | clk__L2_I0    | A v -> Y ^ | INVX2    | 0.011 | 0.013 |   0.030 |    0.026 |
    | curr_state_reg | CK ^      | SDFFSRX1 | 0.011 | 0.000 |   0.030 |    0.026 |
    +-------------------------------------------------------------------------------+
```

-- circular_buffer_postCTSHold.slk

```
# Format: clock  timeReq  slackR/slackF  holdR/holdF  instName/pinName   # cycle(s)
@(R)->clk(R)      0.035    -0.035/*       -0.005/*       curr_state_reg/D    1
@(R)->clk(R)      0.009    -0.003/*       0.021/*        curr_state_reg/RN   1
```

E.7 In Place Optimization Post CTS to fix hold time violations (.summary, all.tarpt, .slk files)

-- circular_buffer_postCTSHold_ipo.summary

```
#############################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:13:16 2012
#  Command:           timeDesign -postCTS -hold -pathReports -slackReports -...
#############################################################

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|         WNS (ns):| -0.035  |   N/A   | -0.035  |   N/A   |   N/A   |   N/A   |
|         TNS (ns):| -0.038  |   N/A   | -0.038  |   N/A   |   N/A   |   N/A   |
|   Violating Paths:|    2    |   N/A   |    2    |   N/A   |   N/A   |   N/A   |
|        All Paths:|    2    |   N/A   |    2    |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 69.933%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

-- circular_buffer_postCTSHold_ipo_all.tarpt

```
#############################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:13:16 2012
#  Command:           timeDesign -postCTS -hold -pathReports -slackReports -numPaths 50 -prefix
circular_buffer_postCTShold_ipo -outDir report
#############################################################
Path 1: VIOLATED Hold Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/D  (^) checked with  leading edge of 'clk'
Beginpoint: software_finished (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time          0.030
+ Hold                         0.005
+ Phase Shift                  0.000
```

```
= Required Time              0.035
  Arrival Time               0.000
  Slack Time                -0.035
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +-------------------------------------------------------------------------------------+
    |   Instance    |       Arc       |   Cell   | Slew | Delay | Arrival | Required |
    |               |                 |          |      |       |  Time   |   Time   |
    |---------------+-----------------+----------+------+-------+---------+----------|
    |               | software_finished ^ |          | 0.000 |       |   0.000 |    0.035 |
    | curr_state_reg | D ^             | SDFFSRX1 | 0.000 | 0.000 |   0.000 |    0.035 |
    +-------------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time        0.000
    Other End Path:
    +-----------------------------------------------------------------------------+
    |   Instance    |   Arc    |   Cell   | Slew | Delay | Arrival | Required |
    |               |          |          |      |       |  Time   |   Time   |
    |---------------+----------+----------+------+-------+---------+----------|
    |               | clk ^    |          | 0.000 |       |   0.000 |   -0.035 |
    | clk__L1_I0    | A ^ -> Y v | INVX8    | 0.007 | 0.017 |   0.017 |   -0.018 |
    | clk__L2_I0    | A v -> Y ^ | INVX2    | 0.011 | 0.013 |   0.030 |   -0.005 |
    | curr_state_reg | CK ^     | SDFFSRX1 | 0.011 | 0.000 |   0.030 |   -0.005 |
    +-----------------------------------------------------------------------------+
Path 2: VIOLATED Removal Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/RN (^) checked with  leading edge of 'clk'
Beginpoint: reset_n          (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.030
+ Removal                    -0.021
+ Phase Shift                 0.000
= Required Time              0.009
  Arrival Time               0.006
  Slack Time                -0.003
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +-----------------------------------------------------------------------------+
    |   Instance    |   Arc    |   Cell   | Slew | Delay | Arrival | Required |
    |               |          |          |      |       |  Time   |   Time   |
    |---------------+----------+----------+------+-------+---------+----------|
    |               | reset_n ^ |          | 0.000 |       |   0.000 |    0.003 |
    | curr_state_reg | RN ^     | SDFFSRX1 | 0.012 | 0.006 |   0.006 |    0.009 |
    +-----------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
```

```
= Beginpoint Arrival Time          0.000
Other End Path:
+-------------------------------------------------------------------------------+
|   Instance   |   Arc    |   Cell   | Slew | Delay | Arrival | Required |
|              |          |          |      |       |  Time   |   Time   |
|--------------+----------+----------+------+-------+---------+----------|
|              | clk ^    |          | 0.000 |       |   0.000 |   -0.003 |
| clk__L1_I0   | A ^ -> Y v | INVX8    | 0.007 | 0.017 |   0.017 |    0.014 |
| clk__L2_I0   | A v -> Y ^ | INVX2    | 0.011 | 0.013 |   0.030 |    0.026 |
| curr_state_reg | CK ^     | SDFFSRX1 | 0.011 | 0.000 |   0.030 |    0.026 |
+-------------------------------------------------------------------------------+
```

-- circular_buffer_postCTSHold_ipo.slk

```
# Format: clock  timeReq  slackR/slackF  holdR/holdF  instName/pinName   # cycle(s)

@(R)->clk(R)     0.035    -0.035/*       -0.005/*       curr_state_reg/D    1

@(R)->clk(R)     0.009    -0.003/*       0.021/*        curr_state_reg/RN    1
```

E.8 Timing Report Post Routing (.summary, all.tarpt, .slk files)

-- circular_buffer_postRoute.summary

```
####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:19:39 2012
#  Command:           timeDesign -postRoute -hold -pathReports -slackReports...
####################################################################

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|    Hold mode       |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|         WNS (ns):| -0.031  |   N/A   | -0.031  |   N/A   |   N/A   |   N/A   |
|         TNS (ns):| -0.032  |   N/A   | -0.032  |   N/A   |   N/A   |   N/A   |
```

```
|   Violating Paths:|   2   |  N/A  |   2   |  N/A  |  N/A  |  N/A  |
|        All Paths:|   2   |  N/A  |   2   |  N/A  |  N/A  |  N/A  |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 100.000%
------------------------------------------------------------
```

-- circular_buffer_postRoute_all.tarpt

```
####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:19:39 2012
#  Command:           timeDesign -postRoute -hold -pathReports -slackReports -numPaths 50 -prefix
circular_buffer_postRoute_hold -outDir report
####################################################################
Path 1: VIOLATED Hold Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/D  (^) checked with  leading edge of 'clk'
Beginpoint: software_finished (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.026
+ Hold                        0.005
+ Phase Shift                 0.000
= Required Time               0.031
  Arrival Time                0.000
  Slack Time                 -0.031
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +-------------------------------------------------------------------------------------+
    |   Instance    |       Arc       |  Cell   | Slew | Delay | Arrival | Required |
    |               |                 |         |      |       |  Time   |   Time   |
    |---------------+-----------------+---------+------+-------+---------+----------|
    |               | software_finished ^ |       | 0.000 |       |  0.000 |    0.031 |
    | curr_state_reg | D ^             | SDFFSRX1 | 0.000 | 0.000 |   0.000 |    0.031 |
    +-------------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time        0.000
    Other End Path:
    +-----------------------------------------------------------------------+
    |   Instance    |   Arc    |  Cell   | Slew | Delay | Arrival | Required |
    |               |          |         |      |       |  Time   |   Time   |
    |---------------+----------+---------+------+-------+---------+----------|
    |               | clk ^    |         | 0.000 |       |  0.000 |   -0.031 |
    | clk__L1_I0    | A ^ -> Y v | INVX8   | 0.007 | 0.015 |   0.015 |   -0.016 |
```

```
    | clk__L2_I0    | A v -> Y ^ | INVX2    | 0.010 | 0.012 |   0.026 |   -0.005 |
    | curr_state_reg | CK ^      | SDFFSRX1 | 0.010 | 0.000 |   0.026 |   -0.005 |
    +--------------------------------------------------------------------------------+
Path 2: VIOLATED Removal Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/RN (^) checked with  leading edge of 'clk'
Beginpoint: reset_n          (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.026
+ Removal                    -0.018
+ Phase Shift                 0.000
= Required Time               0.008
  Arrival Time                0.007
  Slack Time                 -0.001
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +--------------------------------------------------------------------------------+
    |    Instance    |    Arc    |   Cell   |  Slew | Delay | Arrival | Required |
    |                |           |          |       |       |  Time   |   Time   |
    |----------------+-----------+----------+-------+-------+---------+----------|
    |                | reset_n ^ |          | 0.000 |       |   0.000 |    0.001 |
    | curr_state_reg | RN ^      | SDFFSRX1 | 0.014 | 0.007 |   0.007 |    0.008 |
    +--------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time        0.000
    Other End Path:
    +--------------------------------------------------------------------------------+
    |    Instance    |    Arc     |   Cell   |  Slew | Delay | Arrival | Required |
    |                |            |          |       |       |  Time   |   Time   |
    |----------------+------------+----------+-------+-------+---------+----------|
    |                | clk ^      |          | 0.000 |       |   0.000 |   -0.001 |
    | clk__L1_I0     | A ^ -> Y v | INVX8    | 0.007 | 0.015 |   0.015 |    0.014 |
    | clk__L2_I0     | A v -> Y ^ | INVX2    | 0.010 | 0.012 |   0.026 |    0.025 |
    | curr_state_reg | CK ^       | SDFFSRX1 | 0.010 | 0.000 |   0.026 |    0.025 |
    +--------------------------------------------------------------------------------+
```

-- circular_buffer_postRoute.slk

```
# Format: clock  timeReq  slackR/slackF  holdR/holdF  instName/pinName   # cycle(s)
@(R)->clk(R)     0.031    -0.031/*       -0.005/*       curr_state_reg/D    1
@(R)->clk(R)     0.008    -0.001/*       0.018/*        curr_state_reg/RN   1
```

E.9 Timing Report Post Final Setup (.summary, all.tarpt, .slk files)

-- circular_buffer_finalsetup.summary

```
#####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:19:22 2012
#  Command:           timeDesign -postRoute -pathReports -drvReports -slackR...
#####################################################################

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|    Setup mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|         WNS (ns):|  9.870  |   N/A   |  9.870  |   N/A   |   N/A   |   N/A   |
|         TNS (ns):|  0.000  |   N/A   |  0.000  |   N/A   |   N/A   |   N/A   |
|   Violating Paths:|    0    |   N/A   |    0    |   N/A   |   N/A   |   N/A   |
|        All Paths:|    2    |   N/A   |    2    |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|  max_cap       |      0 (0)       |   0.000    |      0 (0)       |
|  max_tran      |      0 (0)       |   0.000    |      0 (0)       |
|  max_fanout    |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 100.000%
------------------------------------------------------------
```

-- circular_buffer_finalsetup_all.tarpt

```
#####################################################################
#  Generated by:      Cadence Encounter 09.11-s084_1
#  OS:                Linux x86_64(Host ID 5013-w47)
#  Generated on:      Fri Apr 13 11:19:23 2012
```

```
# Command:          timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths 50 -prefix
circular_buffer.finalsetup -outDir report
#####################################################################
Path 1: MET Setup Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/D  (v) checked with  leading edge of 'clk'
Beginpoint: software_finished (v) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.026
- Setup                     0.156
+ Phase Shift               10.000
= Required Time              9.870
- Arrival Time               0.000
= Slack Time                 9.870
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
    = Beginpoint Arrival Time        0.000
    Timing Path:
    +-------------------------------------------------------------------------------------+
    |   Instance    |       Arc        |   Cell   |  Slew | Delay | Arrival | Required |
    |               |                  |          |       |       |  Time   |   Time   |
    |---------------+------------------+----------+-------+-------+---------+----------|
    |               | software_finished v |          | 0.000 |       |   0.000 |    9.870 |
    | curr_state_reg | D v              | SDFFSRX1 | 0.000 | 0.000 |   0.000 |    9.870 |
    +-------------------------------------------------------------------------------------+
    Clock Rise Edge                  0.000
    = Beginpoint Arrival Time        0.000
    Other End Path:
    +-----------------------------------------------------------------------------+
    |   Instance    |   Arc    |   Cell   |  Slew | Delay | Arrival | Required |
    |               |          |          |       |       |  Time   |   Time   |
    |---------------+----------+----------+-------+-------+---------+----------|
    |               | clk ^    |          | 0.000 |       |   0.000 |   -9.870 |
    | clk__L1_I0    | A ^ -> Y v | INVX8    | 0.007 | 0.015 |   0.015 |   -9.856 |
    | clk__L2_I0    | A v -> Y ^ | INVX2    | 0.010 | 0.012 |   0.026 |   -9.844 |
    | curr_state_reg | CK ^     | SDFFSRX1 | 0.010 | 0.000 |   0.026 |   -9.844 |
    +-----------------------------------------------------------------------------+
Path 2: MET Recovery Check with Pin curr_state_reg/CK
Endpoint:   curr_state_reg/RN (^) checked with  leading edge of 'clk'
Beginpoint: reset_n          (^) triggered by  leading edge of '@'
Path Groups:  {inclkSrc2reg}
Other End Arrival Time         0.026
- Recovery                  0.064
+ Phase Shift               10.000
= Required Time              9.962
- Arrival Time               0.007
= Slack Time                 9.955
    Clock Rise Edge                  0.000
    + Input Delay                    0.000
```

```
= Beginpoint Arrival Time          0.000
Timing Path:
+-------------------------------------------------------------------------------+
|   Instance    |    Arc    |   Cell   | Slew | Delay | Arrival | Required |
|               |           |          |      |       |  Time   |   Time   |
|---------------+-----------+----------+------+-------+---------+----------|
|               | reset_n ^ |          | 0.000 |       |  0.000  |   9.955 |
| curr_state_reg | RN ^     | SDFFSRX1 | 0.014 | 0.007 |  0.007  |   9.962 |
+-------------------------------------------------------------------------------+
Clock Rise Edge                   0.000
= Beginpoint Arrival Time          0.000
Other End Path:
+-------------------------------------------------------------------------------+
|   Instance    |    Arc     |   Cell   | Slew | Delay | Arrival | Required |
|               |            |          |      |       |  Time   |   Time   |
|---------------+------------+----------+------+-------+---------+----------|
|               | clk ^      |          | 0.000 |       |  0.000  |  -9.955 |
| clk__L1_I0    | A ^ -> Y v | INVX8    | 0.007 | 0.015 |  0.015  |  -9.940 |
| clk__L2_I0    | A v -> Y ^ | INVX2    | 0.010 | 0.012 |  0.026  |  -9.928 |
| curr_state_reg | CK ^      | SDFFSRX1 | 0.010 | 0.000 |  0.026  |  -9.928 |
+-------------------------------------------------------------------------------+
```

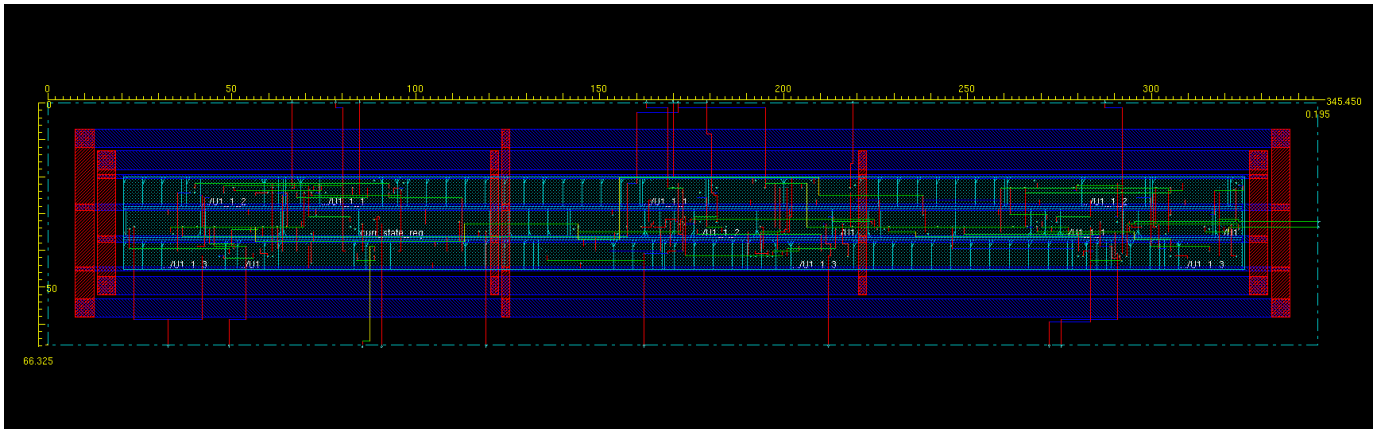
-- circular_buffer_finalsetup.slk

```
 # Format: clock  timeReq  slackR/slackF  setupR/setupF  instName/pinName   # cycle(s)

@(R)->clk(R)     9.870   */9.870       */0.156       curr_state_reg/D    1

@(R)->clk(R)     9.962   9.955/*       0.064/*       curr_state_reg/RN    1
```

E.10 Resulting Integrated Circuit snapshot (from Encounter)



Appendix F: Source Code Section

The code can be found tarballed on the server webpage. Here is an index to the files we modified/created:

Hardware

capstone_main.vhd

hex_display_interface.vhd

sram_interface.vhd

write_to_memory.vhd

read_from_memory.vhd

frame_addressing.vhd

i2c_av_config/i2c_av_config.v

i2c_av_config/i2c_controller.v

video_in/itu_r_656_decoder.v

video_in/video_in.v

video_in/video_in_deinterlacer.v

video_in/video_in_fifo.v

video_in/video_in_resize.v

vga_interface/LineBuffer.v

vga_interface/LineBufferRam.v

vga_interface/process_data_out.vhd

vga_interface/vga_controller.vhd

vga_interface/vga_interface.vhd

process_data_in/data_in_buffer.vhd

process_data_in/process_data_in.vhd

frame_buffer/circular_buffer.vhd

frame_buffer/counter.vhd

frame_buffer/frame_buffer.vhd

* There is also a whole bunch of files generated by SOPC which can be viewed in the SOPC editor

Software

motion_detection/main.c

motion_detection/main.h

These files are all included in the tarballed file with comments.