ECE532 - Pet Motion Tracing

Group Report

Jiaqi Tian
Wenyu Mao
Jiayi Xu

April 9 2018

Contents

1 Overview

1.1 Goals

The goal of our design is to provide information for pet’s motion and duration for staying at home. The project is motivated by the growing population of people having pets over the years.[1] It arises the curiosity that how is the life of our pets when we are not at home. Thus we decided to do something related to pet monitoring. From our research, there exists some pet camera options, which are just remote monitors or pet feeder[2]. To make up for the gap in current market, we then decided to implement a design trying to give users an analysis of pet’s behavior, provides information of recent paths of a pet’s motion and the place they prefer to stay in a statistical approach.

Then we created a design that can do object detection on our pet at home by background subtraction method to get center points with a video camera and draw the path of object movement on a monitor. Additionally, center points will be uploaded onto a computer client by using microblaze as TCP server and through WIFI connection, in order to generate a hot zone image which shows how much percent of the time a pet stays at each zone by dark and bright color as shown in Figure 1. Therefore users can see which area in the living room that the pet prefers to stay.

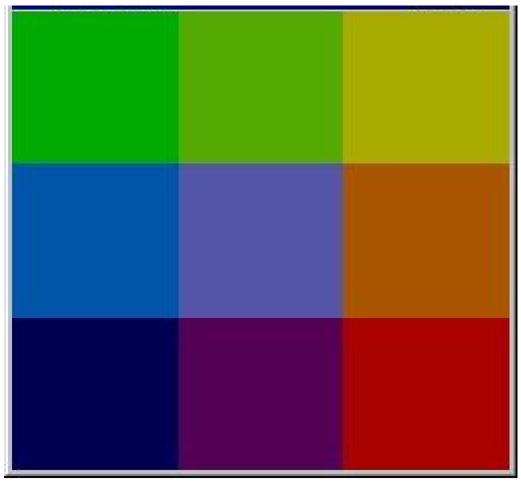


Figure 1: Example of hot zone image

1.2 Block Diagram

A block diagram of the design is shown in Figure 2. It shows the connections between all existing blocks and which level of design each block belongs to. Functionality of each block will be described later in section 4.

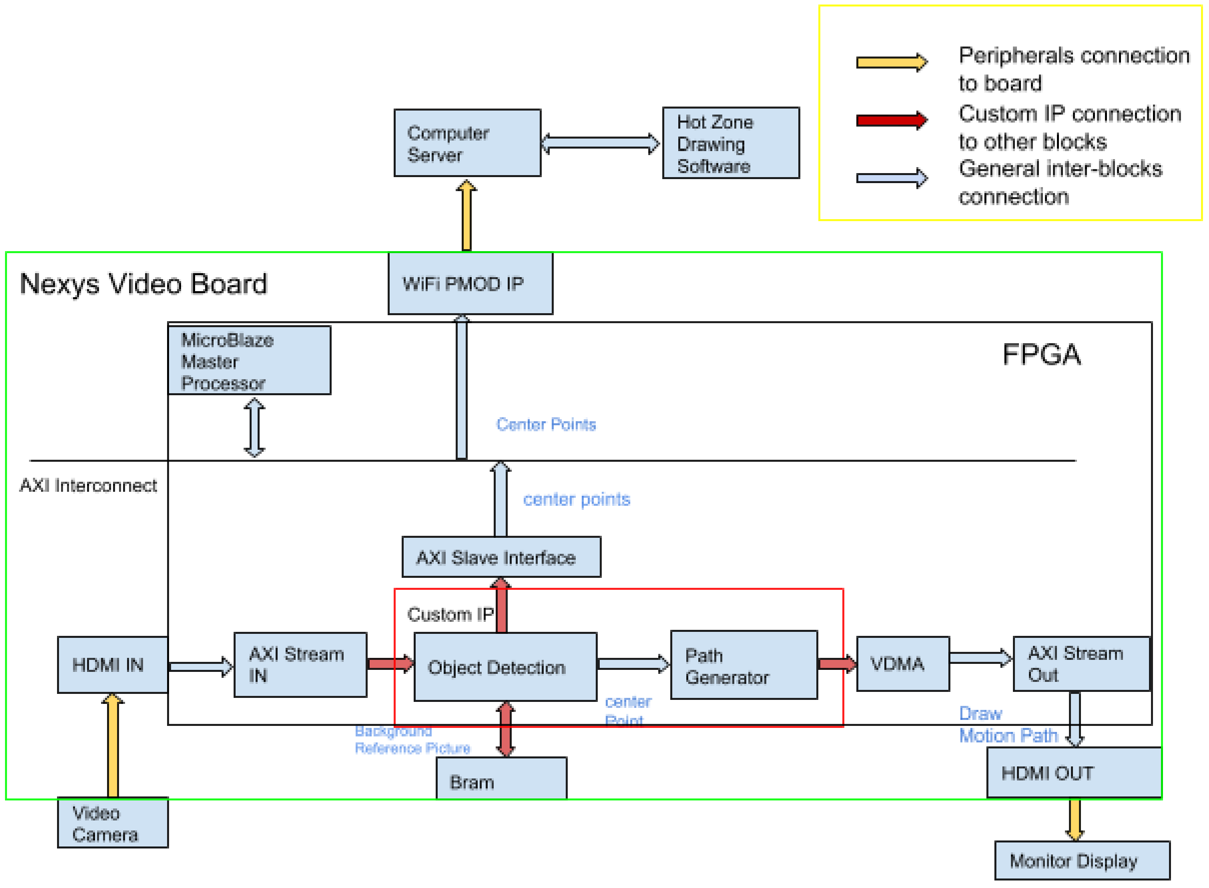
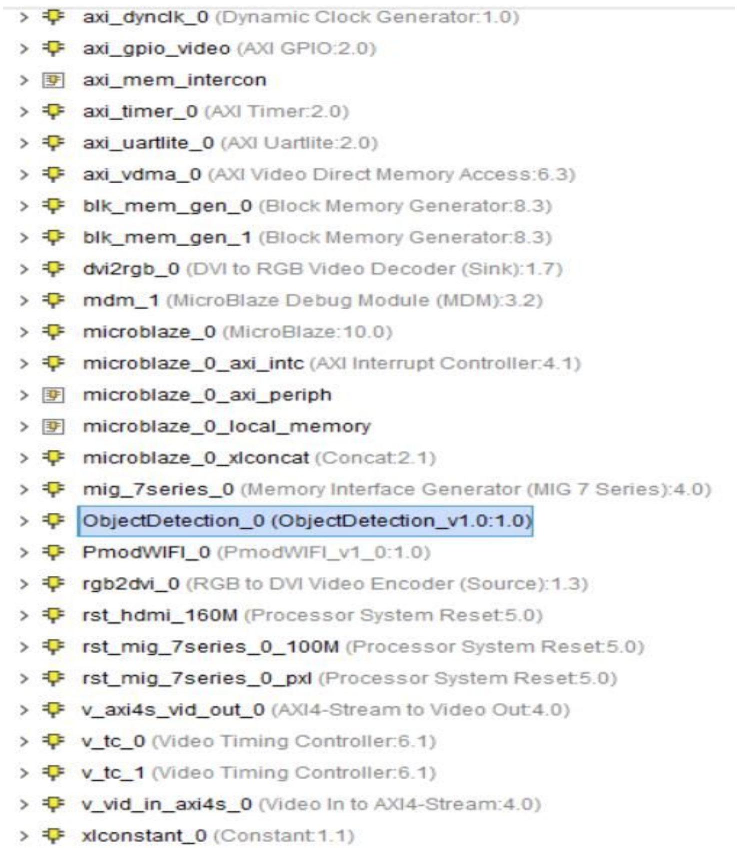


Figure 2: Block diagram of system design showing internal and external connections

1.3 Description of IP



```
> axi_dynclk_0 (Dynamic Clock Generator:1.0)
> axi_gpio_video (AXI GPIO:2.0)
> axi_mem_intercon
> axi_timer_0 (AXI Timer:2.0)
> axi_uartlite_0 (AXI Uartlite:2.0)
> axi_vdma_0 (AXI Video Direct Memory Access:6.3)
> blk_mem_gen_0 (Block Memory Generator:8.3)
> blk_mem_gen_1 (Block Memory Generator:8.3)
> dvi2rgb_0 (DVI to RGB Video Decoder (Sink):1.7)
> mdm_1 (MicroBlaze Debug Module (MDM):3.2)
> microblaze_0 (MicroBlaze:10.0)
> microblaze_0_axi_intc (AXI Interrupt Controller:4.1)
> microblaze_0_axi_periph
> microblaze_0_local_memory
> microblaze_0_xlconcat (Concat:2.1)
> mig_7series_0 (Memory Interface Generator (MIG 7 Series):4.0)
> ObjectDetection_0 (ObjectDetection_v1.0:1.0)
> PmodWIFI_0 (PmodWIFI_v1_0:1.0)
> rgb2dvi_0 (RGB to DVI Video Encoder (Source):1.3)
> rst_hdmi_160M (Processor System Reset:5.0)
> rst_mig_7series_0_100M (Processor System Reset:5.0)
> rst_mig_7series_0_pxl (Processor System Reset:5.0)
> v_axi4s_vid_out_0 (AXI4-Stream to Video Out:4.0)
> v_tc_0 (Video Timing Controller:6.1)
> v_tc_1 (Video Timing Controller:6.1)
> v_vid_in_axi4s_0 (Video In to AXI4-Stream:4.0)
> xlconstant_0 (Constant:1.1)
```

Figure 3: All IP blocks instantiated in the design

IP	Function	Origin
On Board IPs		
Microblaze 10.0	32-bit soft processor Only one processor is used: One processor is used to server as TCP server and make connection to TCP client via WIFI, also controls HDMI input and output, also controls line drawing algorithm by writing and reading registers in custom IP	Xilinx Group design
Dynamic Clock Generator 1.0	Generate serial clock for HDMI pixels	Xilinx
AXI GPIO 2.0	Takes pixel clock lock and generate HDMI hpd signal	Xilinx

AXI Timer 2.0	Use timer/counter to issue a single interrupt	Xilinx
AXI Uartlite 2.0	Provides UART connection to PC for debugging purpose	Xilinx
AXI Video Direct Memory(VDMA) 6.3	Write HDMI video RGB pixel value into memory and send it to video output streaming interface	Xilinx
Block Memory Generator 8.3	Generate block RAM based on parameters set by users.2 block RAM are instantiated in this design	Xilinx
MicroBlaze Debug Mode 3.2	MicroBlaze Debug Module allows JTAG-based debugging	Xilinx
DVI to RGB Video Decoder 1.7	Convert DVI format pixel value to RGB format video value for pixel data processing	Xilinx
RGB to DVI Video Encoder 1.3	Convert RGB format pixel value to DVI format pixel value for video display through HDMI out pins	Xilinx
AXI Interrupt Controller 4.1	Handling interrupt generated by all blocks	Xilinx
Memory Interface Generator 4.0	Generate DDR3 memory interface to store WIFI instructions and pixel values for HDMI	Xilinx
ObjectDetection 1.0	Takes RGB HDMI pixels for object detection, generate coordinates of each video frame based on start of frame signal and continuously draw line paths based on the coordinates of center points of object	Group Design
Video In to AXI4-Stream 4.0	Takes RGB HDMI pixel signal and convert into data and signals in AXI4-Streaming interface	Xilinx
AXI4-Stream to Video Out 4.0	Takes AXI4-Streaming interface data with signals, and convert them into video output format	Xilinx
Concat 2.1	Concatenates multiple interrupt bits into a multi-bits wire	Xilinx
Processor System Reset 5.0	Generate reset signals based on input clock. 3 are instantiated, one for 160 MHz clock, one for 100 MHz system clock, and one for pixel input clock	Xilinx
Video Timing Controller 6.1	Serve as video timing detector and generator, detect timing and format of input video signal and generate timing for video out streaming	Xilinx
Constant 1.1	Set constant value of 1 for HDMI rx_txen signal	Xilinx
Off Board IPs		

PmodWIFI 1.0	Serves for TCP connection for devices within same network	Microchip

2 Outcome

2.1 Results

In the end the project is partially working. There was great difficulty in integrating the all of the separate working components into a final working project. The object detection and path generator are integrated with the HDMI project, with the working functionality to detect the object from the video input and display the moving trace. However, there is a compromise that it can only detect the bright object with a dark background, because of the brightness of the camera. The WiFi part, which uploads the center point from the microblaze server to the computer client, and generates the hot zone image on the computer, can work individually but failed to integrate with the final HDMI project. The original features versus the final features are compared in the following table.

Original Feature	**Final Feature**	**Status**
Object Recognition by using background subtraction. Edge detection will be used as an improvement. Only work with pets in bright color.	Object recognition is realized by background subtraction. The background needs to be dark with pets in bright color.	Implemented
Center point of object will be calculated by hardware and send to software.	Center point is calculated in Object Detection block and read in SDK.	Implemented
FPGA will visualize the center point and show the path of movements in real-time on monitor	The real-time moving path is displayed on the monitor based on the center point	Implemented
Center points will be uploaded onto computer server.	Computer is used as a client rather than a server. Points can be uploaded from	Partially Implemented

	microblaze to computer based on a Wifi connection, but the Wifi part hasn't been integrated with the full HDMI project.	
Camera range will be separated into 9 zones.Computer server will make use of software to visualize a hot zone image, showing the percentage of time the pet stays at each zone. A zone gets brighter color if the pet stays for longer time	Modified. The center point will be displayed on the image with various kind of colors. Therefore the zone where the pet stays for longer time will have a mixed color.	Implemented

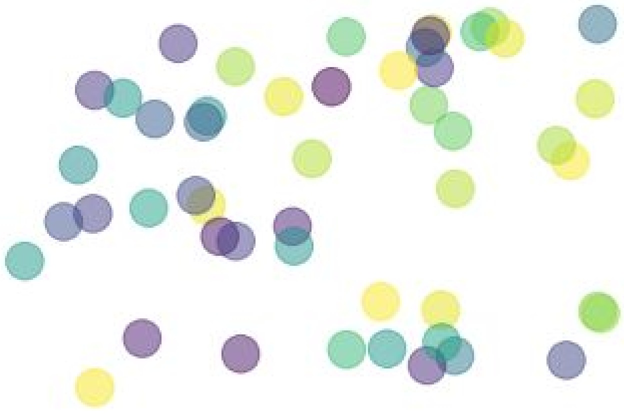


Figure 4: The final image generated at computer side

2.2 Improvement and Future Plan

The final project doesn't fully meet the original criteria but the major features are all implemented. The reason for this is due to the underestimation of the difficulty of integration. Although we don't have many separate components need to be integrated, each integration meets difficulty and new bugs to figure out, and those bugs are normally hard to discover before integration. If the project can be attempted again, we will leave more time for integration and make it in the milestone plan, therefore we may have time to finish the final integration because it is just steps from success now.

Extension to this project would be first to get the WiFi feature integrated with the full HDMI project and make it work properly. If we can continue on this project, our next step is to improve the algorithm of object detection. As the camera is very sensitive to light, sometimes the pixel value can be significantly different even if it doesn't move and the lighting condition hasn't change. Furthermore, it's hard to detect the dark object in the bright background because the dark object will also become bright if the lighting condition is good. Therefore, we need to find out another algorithm which take the lighting difference into account so that the object detection can have better accuracy.

3 Project Schedule

Generally the project was on the right track, however it underwent some difficulties in the second half of the project and in the final integration stage. The blow table includes highlight for every week. Detailed weekly milestone report and explanation of progress and difficulties encountered are articulated in Appendix A.

Week	**Highlight**
1	● Improved Ethernet interaction based on warmup project . ● Created block diagram as shown in figure 2.
2	● Video Buffer: Implemented HDMI demo based on tutorial provided by Digilent. Extracted one frame image from HDMI input and read the data in ● Learned how background subtraction. Determined the detailed algorithm used in the project.
3	● Custom IP: Implemented background subtraction algorithm which can calculate the centre point of a moving object ● Custom IP: Implemented testbench for background subtraction algorithm ● Able to connect a monitor and display no matter what on it. ● Displayed the path on monitor based on the vectors given by our own IP.
4	● Custom IP: Able to generate the motion vector for each cycle based on the setting of counter, and generate the path/trace based on a sequential of vectors ● Line drawing algorithm: Displayed the path based on the vectors given by our own IP. ● Debugged timing issue in both background subtraction algorithm and line drawing algorithm

5	• Displayed a real-time trace according to vectors provided by custom ip • Implemented wifi module • Set up the computer server which can receive the vector for each cycle.
6	• Fixed BRAM timing issue • Improved background subtraction algorithm to get more accurate coordinate of the center point • Alternated server implementation from computer to FPGA board. Computer was working as client. Pmod wifi module successfully communicated with computer server.
7	• Fixed bug in line drawing algorithm • Integrated wifi component and custom ip • Tested overall accuracy

4 Description of Blocks

4.1 HDMI

HDMI is a well written demo project that can be found on Diligent website. The instruction demonstrates the process download the source demo project on Github link: https://github.com/Digilent/Nexys-Video-HDMI . Then instructs on how to use HDMI demo project through UART connection to PC. Thus we can change test patterns, resolutions and colors, etc[3].

In order to make use the demo project for our design. We decided to build our design based on the HDMI project. Our design requires to process input RGB pixel value generated from *DVI to RGB decoder* and output a customized RGB pixel value to VDMA block in a custom IP then pass through *RGB to DVI encoder* for display. So we need to disconnect the connection between *Video In to AXI4-Stream* and *AXI VDMA*. Then create a new custom IP *ObjectDetection* and connect it in between of these two blocks. Since we first chose to use block RAM to store reference frame in background subtraction of object detection method, we have to switch to 640x480 resolution for output display. In order to draw the motion path, we also added coordinate control to record x, y value for each pixels coming in as well as line drawing algorithm in custom IP with start and end x,y coordinates set by microblaze software side. In order to store previous paths drawn, we have a 640x480x1 size block RAM used to store all paths drawn. In addition, clock used in custom IP AXI slave interface that is connected to microblaze is changed from 100 MHz to 160 MHz in order to solve cross clock domain problem.

4.2 Object Detection

Object detection is the core algorithm part of our design. We implemented the object detection method in the custom IP *ObjectDection* to connect with line drawing algorithm. In order to implement object detection, we chose a method called background subtraction[4], which has the structure shown in Figure 4.

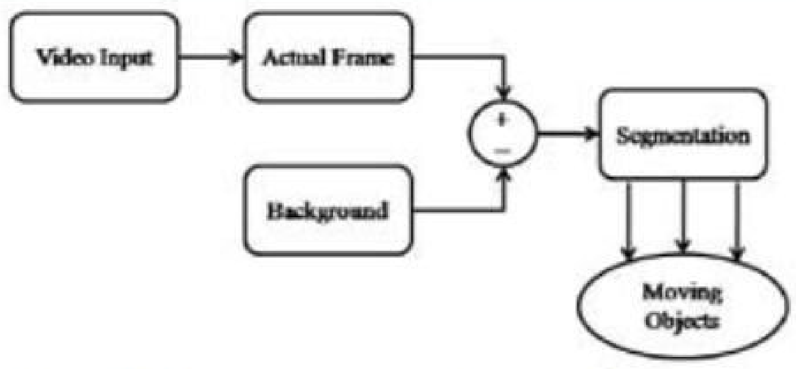


Figure 5: Background subtraction algorithm diagram[4]

We then chose to store a reference HDMI frame in a 640x480x24 block RAM and use the current frame to subtract the reference frame, then use the following equation to set a flag for each pixel to obtain the moving object:

Is_object=(Current_Red-Reference_Red)>=30 || *(Current_Red-Reference_Red)<=-30 && (Current_Blue-Reference_Blue)>=30* || *(Current_Blue-Reference_Blue)<=-30 && (Current_Green-Reference_Green)>=30* || *(Current_Green-Reference_Green)<=-30*

After this we calculate the center point of the object in current frame and send the point coordinate to microblaze software code. Then on the microblaze side we use the previous centerpoint and current centerpoint to draw a motion path.

4.3 PmodWIFI

Pmod WIFI is an external microchip that connect to JA pins on Nexys Video board.

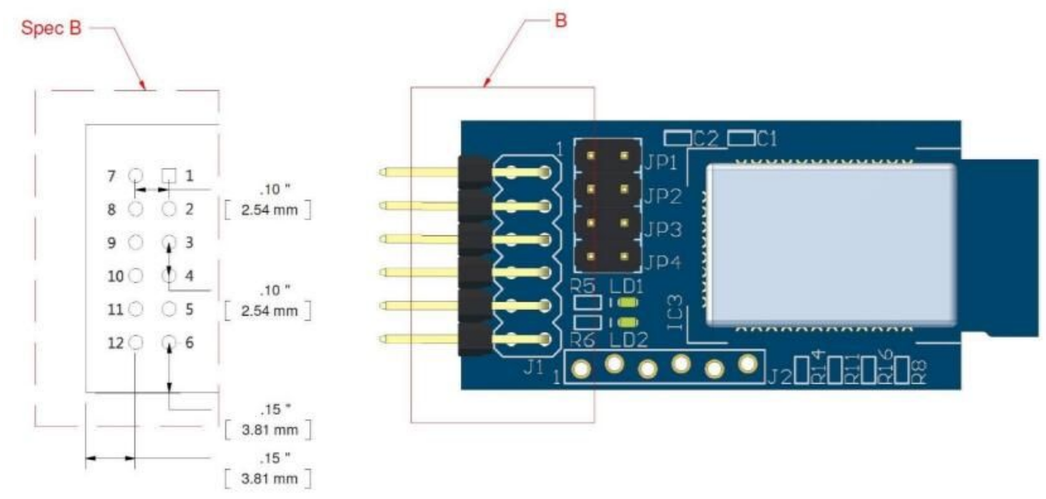


Figure 6: Pmod WIFI chip pin connection to Nexys board[6]

It is not a well documented module, I read through a instruction about TCP server design flow as shown in Figure 6 found online written by Microchip.

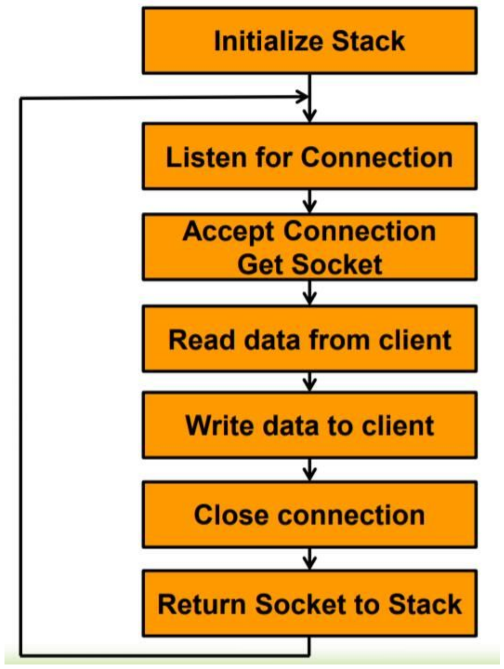


Figure 7: TCP server application flow through WIFI[5]

Then following this instruction, our Pmod WIFI chip can serve as a TCP server can send data into a Python TCP client successfully by modify the original C++ source code. Since we were planning to share the same microblaze with HDMI project, we have to merge the C project from HDMI and C++ project from WIFI using extern C command. But unfortunately we can not reach the final result due to interrupt setting issues when running the program containing both HDMI and WIFI initialization.

4.4 AXI Interrupt Controller

This IP takes multiple interrupt signal from peripheral devices and arrange priority for each interrupt request, then merge into an output interrupt signal through AXI lite interface to have processor handle it[7]. In this design, interrupt controller can handle interrupt from VDMA, video timing controller, UART, AXI Timer and Pmod WIFI. All these signals are concatenated inside a Concat block and feed into AXI interrupt controller.

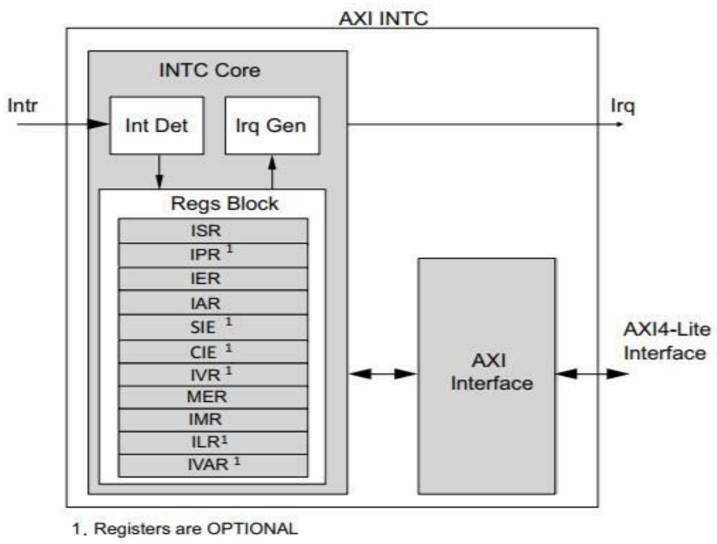


Figure 8: Block Diagram for AXI Interrupt Controller[7]

4.5 Block Memory Generator

Block memory generator is used to generate a block RAM. Currently our design has two block RAMs, one for storage of reference frame for background subtraction and one for storing the past drawn motion paths. In this case we used dual ports RAM with write first setting in order to set read and write address based on x, y coordinate of frame at the same time.

4.6 Memory Interface Generator

We used one memory interface generator to generate a DDR3 memory interface. We used DDR3 to store the HDMI pixels as well as the instructions for TCP WIFI. We also set the memory interface generator to act as a PLL and generate a 160 MHz clock for HDMI out streaming. Here in Figure 8 shows the clock generator structure.

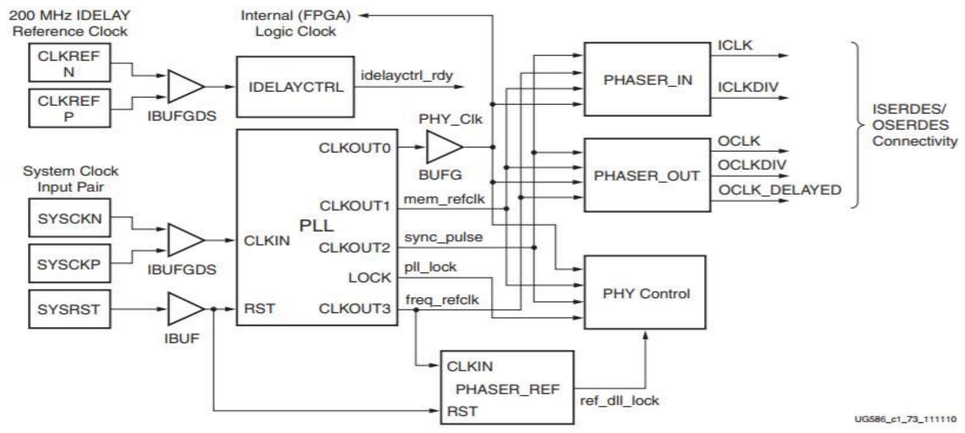


Figure 9: Clock Architecture in Memory Interface Generator[8]

4.7 Dynamic Clock Generator

A dynamic clock generator is used to create a clock for Video output pixel clock and serial clock.

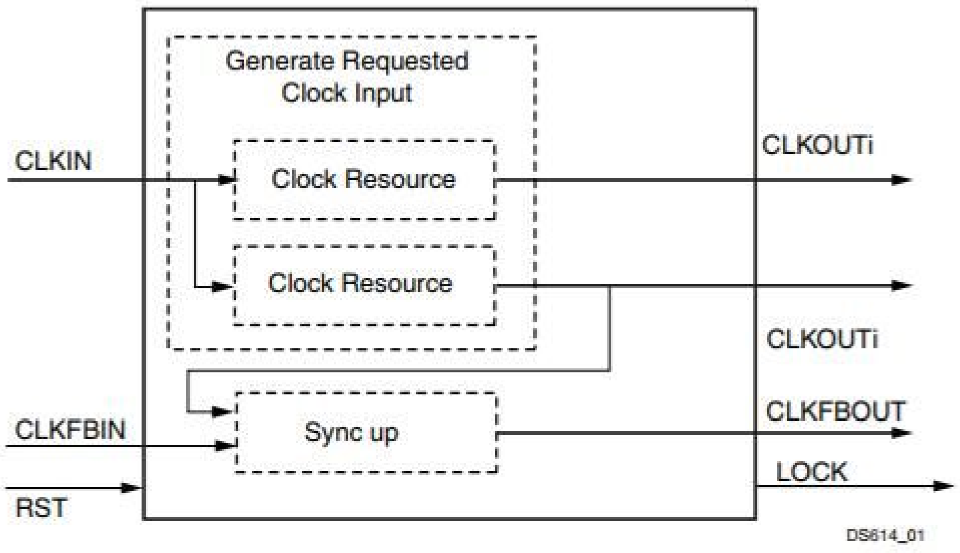


Figure 10: Clock generator structure[9]

4.8 AXI Timer

This IP provides a counter for microprocessor and generate an interrupt if timing is met. Thus prevent the processor from stuck with one process for too long. Figure 10 shows the detailed structure on input timing registers through AXI interface and use 32-bit counter to generate interrupt signal.

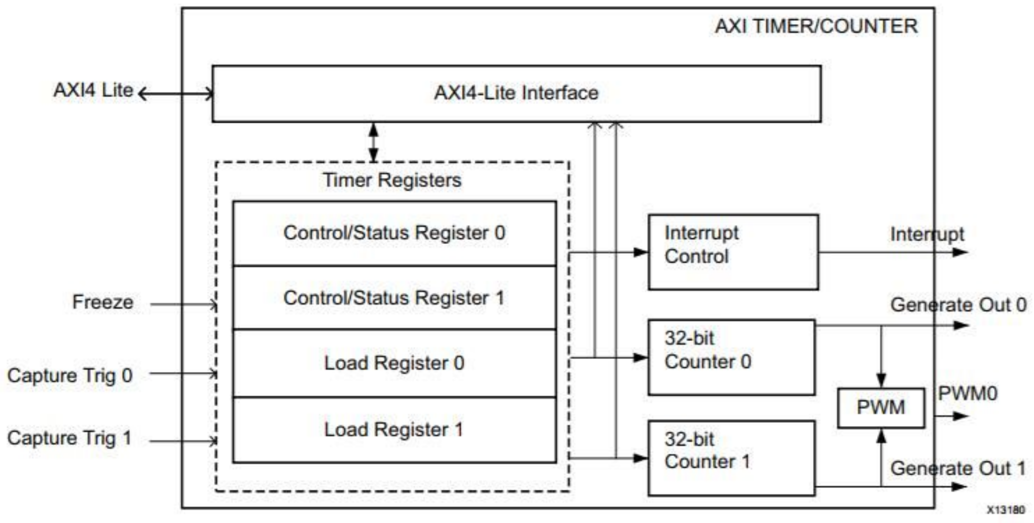


Figure 11: AXI Timer Structure[10]

4.9 AXI Video Direct Memory(VDMA)

VDMA block enables the system to write HDMI video pixel RGB values directly into DDR memory through AXI streaming interface in a faster rate. Therefore designers can get access to DDR for video processing in the future. Here on Figure 11 shows the detailed structure of AXI VDMA and how video data flows into memory.

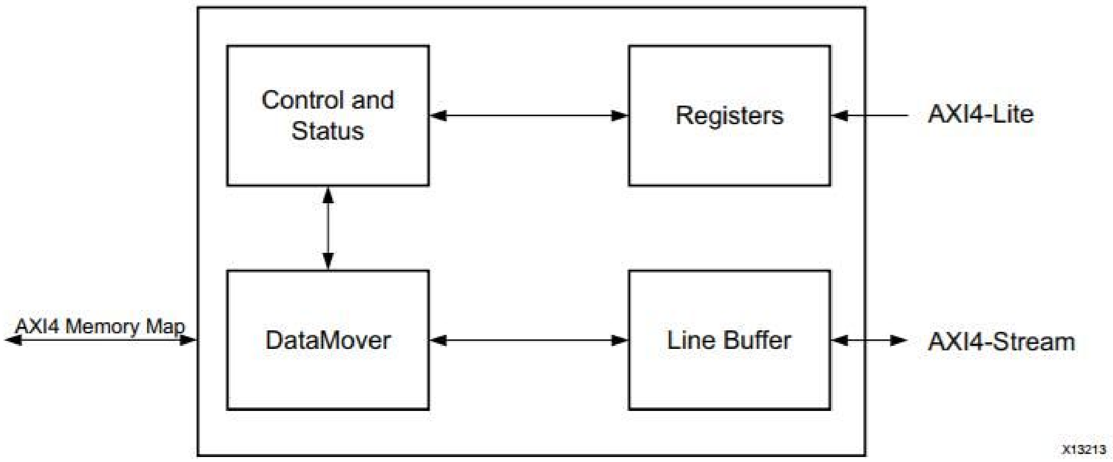


Figure 12: AXI VDMA Structure[11]

5 Description of Design Tree

The entire project and related files can be found here:
https://github.com/tianjiaq/G11-Pet-Motion-Tracing/tree/master/project2-LDAcomplete/Nexys-Video-HDMI

The structure of the project is shown in the table below:

Directory	**Description**
doc	Documentations of the project including final report and presentation slides
src/bd	All source files of the project
src/bd/hdmi/ip	All verilog files, including object detection ip wifi pmod module and all other blocks in the block diagram
src/bd/hdmi/ip/hdmi_ObjectDetection_0_0	Verilog files for custom IP
sdk	All C++/C codes written in SDK

6 Tips and Tricks

Due the magnitude of the project, one of the most important things to remember is to always leave enough time for debugging and integration. There will be lots of unexpected difficulties during the progress especially when integrating different components together. During the debugging, testbench is very important. A testbench with wrong setting will result in major difficulties in the integration stage. Moreover, simulation are not always reliable, especially behaviour simulation. Because timing is crucial in FPGA design, wrong timing will impact the functionalities seriously. When doing post synthesis simulation, always apply (dont_touch =”true”) to avoid the undesired optimization by vivado tool. Last but not least, when working separately, make sure to communicate with team members to keep the consistency of different components, otherwise the integration will be a huge problem.

Appendix:

Appendix A: summary of weekly milestone reports

Milestone	**Status**
Milestone 1	Current milestone: ● Improve Ethernet interaction based on warmup project . ● Create block diagram as shown in figure 2. Milestone status: ● Successfully finished both goals Project modification: ● None
Milestone 2	Current milestone: ● Implement HDMI demo on 2017.2 version ● Receive video frame data from HDMI IN ● Learn how background subtraction algorithm works Milestone status: ● HDMI demo can work on 2017.2 version ● Find a way to implement background subtraction algorithm pixel by pixel ● Finished connection of custom IP, trying to display HDMI frames on monitor through custom IP Reason for discrepancies between goals and status: ● No discrepancy Project modification: ● None
Milestone 3	Current milestone: ● Able to connect a monitor and display no matter what on it. ● Receive video frame data from HDMI IN ● Learn how background subtraction algorithm works Milestone status: ● Custom IP can draw multiple paths given each start and end x,y coordinate locations. ● Use one Block RAM to store the input frames, and

	another Block RAM to store drawn paths. ● Build a custom IP to calculate the center point of the object for a black background Reason for discrepancies between goals and status: ● No discrepancy Project modification: ● As Bram is big enough to hold a reference picture, we change to save the reference picture in Bram instead of DDR. ● Modified resolution of input frame from 1080P to 480P
Milestone 4	Current milestone: ● Able to generate motion vectors of center points of object and draw the path ● Calculate the center point of each frame to generate the path/trace Milestone status: ● Continued on the custom IP to calculate the center point of the object for a reference background stored in Bram ● Debugged and fix the timing problem in background subtraction ● Trying to connect line drawing algorithm with object detection part ● Fixed some timing problems in line drawing algorithm Reason for discrepancies between goals and status: ● Unexpected timing issues in line drawing algorithm and background subtraction algorithm. ● Differences between behaviour simulation, post-synthesis simulation and post-implementation simulation ● Used wrong FSM encoding before and took some time to notice and fix. Project modification: ● None
Milestone 5	Current milestone: ● Implement wifi module

	● Combine line drawing and background subtraction algorithm together to display the real-time trace Milestone status: ● Finished setup hardware connections by following instructions ● Integrated line drawing and background subtraction algorithm but encountered new bram timing issue, which cause the failure of integration Reason for discrepancies between goals and status: ● SDK side has unknown issue during build ● Debugging with hardware debugger took a long time to resynthesize even after a small modification on the IP ● Unexpected timing issue when integrating two algorithm together Project modification: ● To improve the accuracy, add the constraint that the background need to be dark color.
Milestone 6	Current milestone: ● Implement Pmod wifi module and make stable communication with TCP server ● Fix Bram issue encountered last week ● Implement the generation of hot zone image Milestone status: ● Pmod WIFI is able to connect WIFI but fails to connect TCP server. ● Integrated line drawing and background subtraction algorithm but encountered inconsistency ● Improved the background subtraction algorithm to not include garbage data in. Reason for discrepancies between goals and status: ● Not familiar with the timing information. From the path, it's hard to get where the related code is and how to fix it. ● The pixel value reading from the camera is more

	random than expectation. Almost every pixel value has some change even if the camera doesn't move. Therefore, the threshold for the detection has to be bigger. ● There are some issues with the line drawing part. Need to go back to the algorithm to re-debug it. Project modification: ● Decided to use computer as client and video board as server.
Milestone 7	Current milestone: ● Manipulate threshold value to get best object detection performance ● Finish integration of hardware and software components Milestone status: ● Wifi connection and communication between server and client is working ● Object detection and line drawing algorithm are successfully integrated and working well. ● Hot zone image generating is working ● The integration of wifi component and detection algorithm has some issue because of the inconsistency on software side Reason for discrepancies between goals and status: ● The difficulty of integrating wifi and rest of the project in SDK. Since the original tutorial we followed to develop the HDMI blocks is written in C++ whereas the wifi tutorial is written in C. It took a long time to rewrite everything in same language. ● We didn’t expect integration is such a tough work, so didn’t leave enough time for integration, which results in the failure in the end. Project modification: ● None

References

[1]statista. *Number of cats in the United States from 2000 to 2017/2018.*[Online] Available: https://www.statista.com/statistics/198102/cats-in-the-united-states-since-2000/
[2]Alexia Chianis. *8 Pet Cameras Every Pet Owner Should Know*.[Online] Available: https://www.safewise.com/blog/8-pet-cameras-every-pet-owner-should-know-about/
[3]Diligent. *Nexys Video HDMI Demo*.[Online] Available: https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-video-hdmi-demo/start
[4]Anu Susan Philip. *Background Subtraction Algorithm for Moving Object Detection Using Denoising Architecture in FPGA*. [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=BC4B934E70B9BDCC11F4268B7D5AC8BD?doi=10.1.1.680.8732&rep=rep1&type=pdf
[5]Microchip. *18020 EWN*. [Online] Available: https://reference.digilentinc.com/_media/reference/pmod/pmodwifi/networkslides.pdf
[6]Diligent. *Diligent Pmod Interface Specification 2.0*. [Online] Available: https://reference.digilentinc.com/_media/reference/pmod/pmod-interface-specification-1_2_0.pdf
[7]Xilinx. *AXI Interrupt Controller(INTC) v4.1*. [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_intc/v4_1/pg099-axi-intc.pdf
[8]Xilinx. *Block Memory Generator v8.3*. [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/blk_mem_gen/v8_3/pg058-blk-mem-gen.pdf
[9]Xilinx. *Clock Generator*. [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/clock_generator/v4_03_a/clock_generator.pdf
[10]Xilinx. *AXI Timer*. [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_timer/v2_0/pg079-axi-timer.pdf
[11]Xilinx. *AXI Video Direct Memory Access*. [Online] Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_vdma/v6_2/pg020_axi_vdma.pdf