REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
09-09-2016	Final Report	31-May-2013 - 30-May-2016

4. TITLE AND SUBTITLE
Final Report: High Performance Reconfigurable Computing: Research and Education

5a. CONTRACT NUMBER
W911NF-13-1-0135

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
206022

6. AUTHORS
Clay Gloster

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
North Carolina A&T State University
1601 East Market Street

Greensboro, NC 27411 -0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
62801-CS-REP.11

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
Radar processing, remote sensing, object detection, autonomous vehicle path planning, and predictive simulation constitute a class of army applications. Those future army applications use high definition images, videos and context information to provide assistance to soldiers by providing lethality protections. Those floating point algorithms are difficult to accelerate on general purpose processors or graphic processors with limited power consumption. When hardware designers accelerate the algorithm using customized floating-point hardware units on FPGAs, it may be time consuming to explore the design space due to various parameters such as resource

15. SUBJECT TERMS
FPGA, Reconfigurable Computing, High Performance Computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Clay Gloster
UU	UU	UU	UU		19b. TELEPHONE NUMBER 336-285-3134

Report Title

Final Report: High Performance Reconfigurable Computing: Research and Education

ABSTRACT

Radar processing, remote sensing, object detection, autonomous vehicle path planning, and predictive simulation constitute a class of army applications. Those future army applications use high definition images, videos and context information to provide assistance to soldiers by providing lethality protections. Those floating point algorithms are difficult to accelerate on general purpose processors or graphic processors with limited power consumption. When hardware designers accelerate the algorithm using customized floating-point hardware units on FPGAs, it may be time consuming to explore the design space due to various parameters such as resource utilization, accuracy, power consumption, and performance values. It is critical to explore this design space early to ensure that the algorithm can be efficiently mapped onto an FPGA. Our research presents a reconfigurable acceleration environment while addressing the problem of porting High Performance Computing (HPC) applications directly to Field Programmable Gate Array (FPGA)-based architectures. Our methodology presents the development of a comprehensive floating point library of essential functions for scientific applications; demonstrating the order of magnitude speedup of reconfigurable computing applications and the effectiveness of a design framework for both development and test of scientific algorithms. Finally, we demonstrate accessing these FPGA-based solutions from a remote site.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 3.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper
09/07/2016	6.00	. IoTGuard: Scalable and Agile Safeguards for Internet of Things, IEEE Military Communications Conference, 2016 - MILCOM 2016. 01-NOV-16, Baltimore, MD. : ,
09/07/2016	8.00	. An Automated Design Framework for Floating Point Scientific Algorithms using Field Programmable Gate Arrays (FPGAs), IEEE International Symposium on Filed-Programmable Gate Arrays. 22-FEB-15, Monterey California. : ,
09/07/2016	9.00	. Acceleration of Synthetic Aperture Radar (SAR) Algorithms using Field Programmable Gate Arrays (FPGAs), 23rd ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 22-FEB-15, Monterery, California. : ,
TOTAL:		**3**

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper
TOTAL:	

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received		Paper
08/31/2014	1.00	Youngsoo Kim, William Harding, Clay Gloster, Jr., Winser Alexander. Acceleration of Radar Processing Algorithms Using Field Programmable Gate Arrays (FPGAs), IEEE Global Conference on Signal and Information Processing (06 2014)
08/31/2015	2.00	Ramsey Hourani, Clay Gloster, Winser Alexander, Youngsoo Kim. “Dataflow based Performance Estimation for Two Dimensional Image Processing Hardware Design" , Journal of Real Time Image Processing (08 2015)
TOTAL:	**2**	

Number of Manuscripts:

Books

Received	Book
TOTAL:	

Received	Book Chapter
TOTAL:	

Patents Submitted

Patents Awarded

C. Gloster, W. Gay, M. Amoo, "Multiple-Memory Application-Specific Digital Signal Processor", US Patent #911068, August 18, 2015.

Awards

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Shrikhant Jadhav	1.00	
Jannutan Naher	1.00	
Vance Alford	1.00	
FTE Equivalent:	**3.00**	
Total Number:	**3**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Clay Gloster	0.00	
Christopher Doss	0.00	
Naser El-Bathy	0.00	
FTE Equivalent:	**0.00**	
Total Number:	**3**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED	Discipline
Vance Alford	1.00	
Vernon Kornegay (URP)	0.33	
FTE Equivalent:	**1.33**	
Total Number:	**2**	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	1.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:	1.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:	1.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:	1.00

Names of Personnel receiving masters degrees

NAME
Total Number:

Names of personnel receiving PHDs

NAME
Total Number:

Names of other research staff

NAME	PERCENT_SUPPORTED
Dylan Jordan (HSRP)	0.33
Jacob Anderson	0.00
Andrew Harvey	0.00
FTE Equivalent:	**0.33**
Total Number:	**3**

Sub Contractors (DD882)

Inventions (DD882)

5 **Multiple Memory Application-Specific Digital Signal Processor**

Patent Filed in US? (5d-1) Y

Patent Filed in Foreign Countries? (5d-2) N

Was the assignment forwarded to the contracting officer? (5e) N

Foreign Countries of application (5g-2):

5a: Clay Gloster

5f-1a: North Carolina A&T State University

5f-c: 1601 E Market Street

Greensboro NC 27411

5a: Wanda Gay

5f-1a: NAWCAD 4.9.4.4

5f-c: 47561 Ranch Road, Building 4023

Patuxent River MD 20670

5a: Michaela Amoo

5f-1a: Howard University

5f-c: 2300 Sixth Street NW

Washington DC 20059

Scientific Progress

Technology Transfer

Table of Contents

Statement of the problem studied

Given an application that requires excessive computation time,

- **Develop tools and techniques to automate portion of the RC application development process:** Develop tools that will automate significant portions of the application development process for the RC systems.
- **Reduce the application development time:** RC systems require more application development time than general-purpose processors. Our goal is to reduce application development time significantly.
- **Increase the performance of the application:** It is well proven that applications can be speeded up when they are mapped to RC systems. Our goal is an order of magnitude speed up when the applications are mapped to RC system.
- **Open RC systems to the people who are not knowledgeable in RC system design:** Developing an RC application requires designers who are knowledgeable in the areas of hardware and software system design. Our goal is to develop a system where users with little or no hardware knowledge can be able to develop RC applications using tools and techniques developed as a part of this research.
- **Develop a system where users gain access to an RC system from a remote site:** Our goal is to develop tools that allow a user to send data from a remote site to an RC system, process the data on the RC system, and return results to the user. In some applications, even with the delay of the Internet, some applications can run faster using a remote RC system than a local system.
- **Engage college and high school students in research and expose them to potential career opportunities in the Department of Defense.** Many students are unaware of the potential career opportunities at DOD research labs and other military installations. Our goal is to have seminars to promote these opportunities through engaging students in research and conducting workshops.

Summary of the most important results

The Remote and Reconfigurable Environment (RARE) has been developed and configured for various applications. The primary tool for executing applications from a remote site is what we call the loader. It is a system that allows the user to execute applications on the FPGA board from a remote site.

The loader reads one or more session files which contain all information needed to execute the application on an RC system. The loader first configures the FPGA devices and the FPGA device(s) process the given data. After the processing data has been completed, the loader will grab the results from FPGA memory and return them to the application. Figure 1 shows the implementation of the loader on an FPGA device.

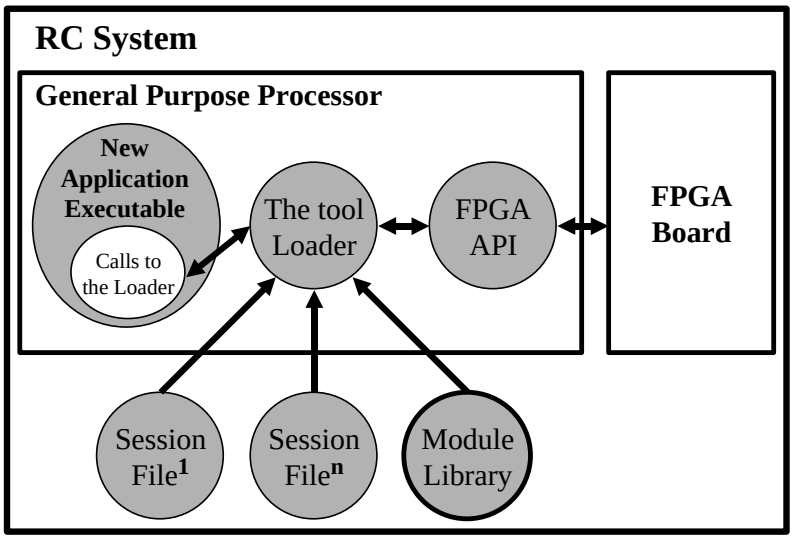

Figure 1: The Loader

All major components of the loader have been implemented. The module library, the loader, assembly language instruction set for the current modules and the session file format have been developed. In this section, these components are briefly introduced.

Floating-Point Module Library

In this study, several floating-point modules were developed including: addition, subtraction, multiplication, an accumulation module, square root, division, exponential function, the power function, and the logarithm function. All modules have been designed using 32 bit single precision floating point arithmetic. To create different types of modules that are useful for various applications we developed several standard components. These component types are floating-point core units, standard controllers and standard datapaths. These components are standardized in terms of the number of inputs, the number of outputs and module latency in order to facilitate module interconnection for complex operations. By combining unique core units with standard controllers and datapaths, several different types of applications have been created. Using this approach, the time required to design a new module is reduced significantly. When a new core unit is designed, one simply combines the new core with an off-the-shelf controller and datapath. This can be performed automatically using the tool we developed called hl2ll. It stitches together standard VHDL modules to create the complete Application Specific Digital Signal Processor shown in Figure 2.

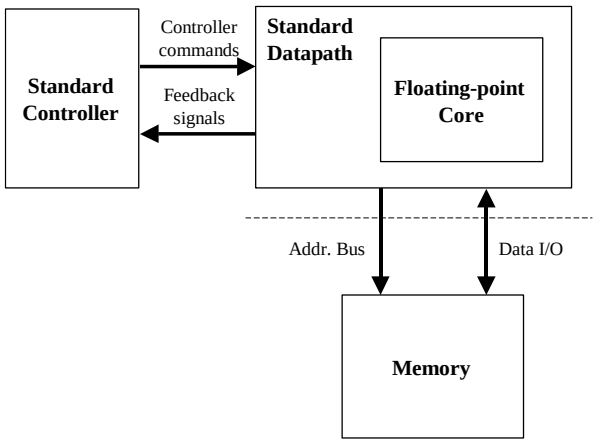

Figure 2: An Application Specific Digital Signal Processor (ASDSP).

ASDSP Instruction Format and Module Execution

All ASDSPS are designed to execute a specific machine language instruction. Each module instruction corresponds to a single floating-point vector operation. A standard instruction includes three or four operands depending on the type of module used. Figure 3 shows the instruction format for each module type. For each three-operand module instruction, the first operand is the starting address of the input vector, the second is the starting address of the output vector, and the third is the size of the input vectors.

For each four-operand module instruction, the first two operands are the starting addresses of the two input vectors, the third operand is the starting address of the output vector and the last operand is the size of the vectors.

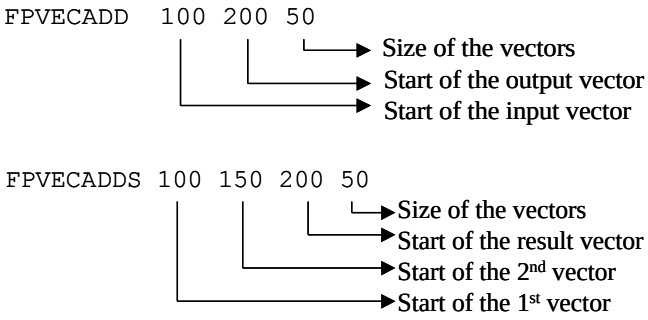

Figure 3: Modules instruction formats. (a) Module instruction for 2 operand vector modules. (b) Module instruction format for 3 operand vector module and the multiply-accumulate module.

All modules were designed for a commercial FPGA board that is readily available in our laboratory. This board includes one FPGA devices or Processing Elements (PEs). The PE can access 5 SRAMS and 1 SDRAM memory banks. The host computer and the PE can both read/write from/to the local memories. The memory space of each module is partitioned into two sections, instruction and data. The instructions are stored in the block RAM located on the FPGA device. The other memories on the FPGA board are used to store the data to be processed.

Once a module configuration has been loaded into a PE and the local memory has been initialized by the host computer, the module waits for the StartPE signal to be asserted. When this occurs, the module reads the first instruction from the memory location $00000. It then begins executing the instruction. When the current instruction has completed, the module reads the next instruction from the instruction memory. This process continues until the module reads a HALT instruction ($FFFFFFFF) from the instruction memory. When this value is read, the module stops and sends an interrupt signal to the host computer.

Core Units

The most important component of a module is the floating-point arithmetic core. For each floating-point operation, we developed a standard core unit. Each core unit is highly pipelined, has the same control inputs, multiple single precision floating point inputs and outputs, and a latency that is a multiple of 8. This was selected so standard delay units of size 8 can be inserted to balance complex cores as required in a pipelined unit. By instantiating each unique core unit into a standard module structure, we created a new complex core for each operation.

Figure 4 shows the block diagram of the standardized core unit. Each core has two 32-bit inputs and one, 32-bit output to accommodate single precision FP numbers. For addition, subtraction and multiplication, different floating-point core units were developed. There is a standard interface definition for the core units to reduce design time. Once a new core unit is designed, it is easy to create a new module by just instantiating the new core unit into the standard module structure.

All core units are divided into a standard number of pipeline stages (8) to improve the maximum clock speed that can be applied to the units. We used a standard number of pipeline stages to alleviate the need to develop a unique controller within each core. However, the main controller can handle cores with arbitrary latencies. While using pipeline units requires additional registers, resulting in an increase in FPGA CLB resources, it provides significant benefit in terms of increased clock speed.

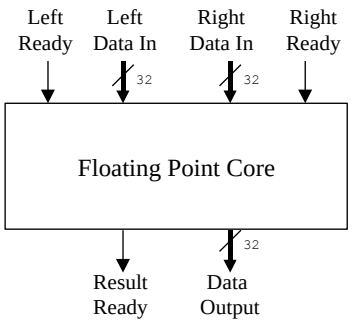

Figure 4: Block diagram of the standard core units.

Core units are designed as self-controlled units to reduce the hardware requirements and to make the module controller simpler. Once data is available at both inputs, the core unit starts processing. Results are available at the output of the unit 8 clock cycles later.

This is accomplished with a standard floating–point core I/O interface. Each core has two input signals and one output signal for control and core interconnection. Each time that the module controller reads a floating-point number from the memory, it asserts either the LEFT_READY or RIGHT_READY signal corresponding to the core input that has valid data. When both inputs to the core have valid data and both ready signals are asserted, the core begins the floating-point operation. When the core finishes

processing the data, it asserts the RESULT_READY signal. The main controller then stores the result in memory.

Use of the standard interface control signals serves two purposes. The main purpose is to reduce controller complexity. Hence, a single controller can handle future cores with arbitrary latencies. The controller does not send command signals to each stage of the core. Instead, it uses the interface signals to signal the core that the input data is ready. It also uses the RESULT_READY signal produced by the core to determine when the result is ready. This simplification in the controller saves control states, logic gates, and future application development time. The other purpose is to facilitate the addition of complex cores into the library. The use of the standard interface control signals makes it is easy to form larger cores by simply linking existing cores together.

RARE: An End-to-End High Performance Computing Environment for Army Applications

Radar processing, remote sensing, object detection, autonomous vehicle path planning, and predictive simulation constitute a class of army applications. Those future army applications use high definition images, videos and context information to provide assistance to soldiers by providing lethality protections. Those floating point algorithms are difficult to accelerate on general purpose processors or graphic processors with limited power consumption. When hardware designers accelerate the algorithm using customized floating-point hardware units on FPGAs, it may be time consuming to explore design space due to various parameters such as resource utilization, accuracy, power consumption, and performance values. It is critical to explore this design space early to ensure that the algorithm can be efficiently mapped onto an FPGA. Our research presents a reconfigurable acceleration environment while addressing the problem of porting High Performance Computing (HPC) applications directly to Field Programmable Gate Array (FPGA)-based architectures. Our methodology presents the development of a comprehensive floating point library of essential functions for scientific applications; demonstrating the order of magnitude speedup of reconfigurable computing applications and the effectiveness of automated design framework for both development and test of scientific algorithms.

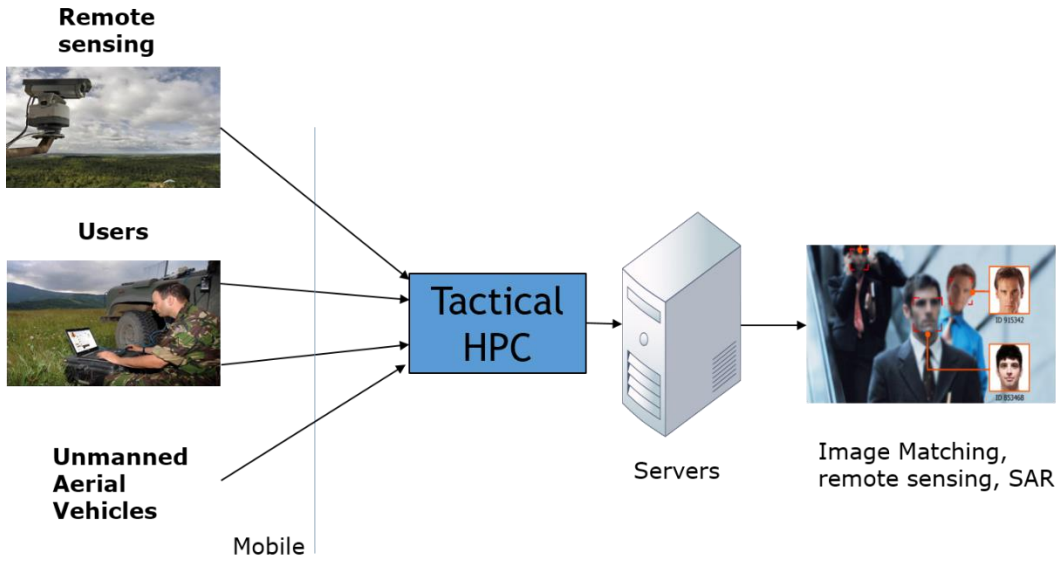

Fig. 5. End-to-End diagram of RARE Acceleration

With this perspective in our framework and benchmark applications, we perform an investigation of various acceleration methodologies based on our RARE platform. As shown in Figure 5, we provide insights into future tactical HPC design and custom FPGA based accelerator design for emerging applications. We make the following contributions:

- **Remote sensing including Synthetic Aperture Radar (SAR)** – We constructed an end-to-end SAR case study with real high definition (HD) input images. We verified our FPGA based RARE platform by accelerating a kernel code including logarithmic homomorphic transformation.

- **Streaming architecture for FPGAs** – Based on cycle breakdown analysis with multiple ARL applications, we developed prototype streaming memory architecture and showed its performance improvement on the Zynq embedded vision system prototype.

- **Real-time Object detection** – We improved the state-of-the-art object detection algorithm by adopting our floating point hyperbolic functions. We show that there is potential speed-up for the Army's workload including night vision object detection and unmanned aerial vehicles' application algorithms.

- **Floating point math computation acceleration** – We developed an automatic floating point hardware block generator and used it for our own acceleration.

- **Future Server Execution Environment Design** – Based on our acceleration results, we investigated the implication for future remote method invocation environment. The prototype interface was developed by using Java Remote Method Invocation (RMI).

Remote sensing, Synthetic Aperture Radar Processing Acceleration

SAR is a radar signal processing technique used to produce accurate resolution images beyond the several kilometer range. The basic operation for SAR is to transmit radio signals to the ground and measure the reflected signal. Due to this active property, SAR can produce high resolution 2-D and 3-D images of the earth's surface. Recently, SAR has been used more frequently in military applications such as Unmanned Autonomous Vehicles (UAVs). The SAR hardware system is required to be implemented in a number of small stacks such as 3 x 3 x 3 inch cubes with high performance and low power requirement. Hence, using an FPGA to implement SAR is desirable.

SAR images have high dynamic range (HDR) due to the characteristics of its capturing method. The dynamic range of SAR images is often represented by the lightest spot in a speckle and the darkest point in the background. The original SAR images from SANDIA National Laboratories

have a word size of 16 bits. The image size is typically larger than 4,096 pixels both horizontally and vertically. SAR images have shown characteristics that are different from moving picture formats such as MPEG and H.264 format. General filtering approaches are not often preferred for SAR images since they are developed with optical signals in mind. Often, general purpose fixed point hardware does not efficiently suppress speckles present in SAR images due to its limited accuracy. Hence, we developed a floating point hardware implementation of the natural logarithm function.

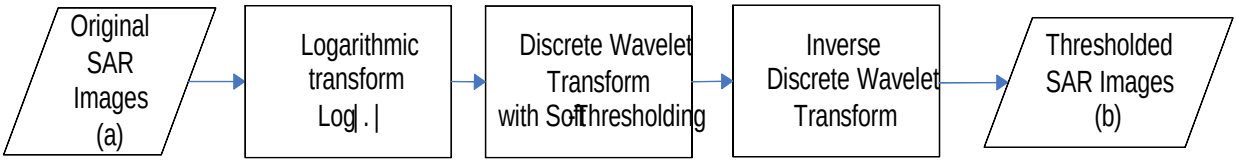

Fig. 6. SAR Image Enhancement

Fig. 6 shows a high level block diagram of our SAR case study. We directly apply a logarithmic filter with a 2D Discrete Wavelet Transform (DWT) to reduce the presence of speckles by applying the soft-thresholding technique. The objects' edges, which reside in the low frequency component of the signal, are preserved from blurring while removing the noise. We validated the high level model of the SAR processing algorithm using MATLAB. We used the real SAR image data provided by Sandia National Laboratories MiniSAR image archives.

We apply the logarithmic transformation to boost the signal background which contains a great deal of information. In SAR images, the signal can include cars and buildings in the region of interest. Soft thresholding is a method to remove and suppress the noise signal in the wavelet domain and is well defined in Mallat and Donoho's work. We apply the nonlinearity to the wavelet coefficients with a threshold value chosen by MATLAB and our empirical image SAR output visual inspections. The objective of a logarithmic filtering is to change the statistics of a SAR image. The original speckle statistics are changed from a multiplicative noise to an additive noise. This noise in the logarithmic transformed SAR image can be more easily removed by an appropriate filtering since speckles reside in the high frequency component of the signal.

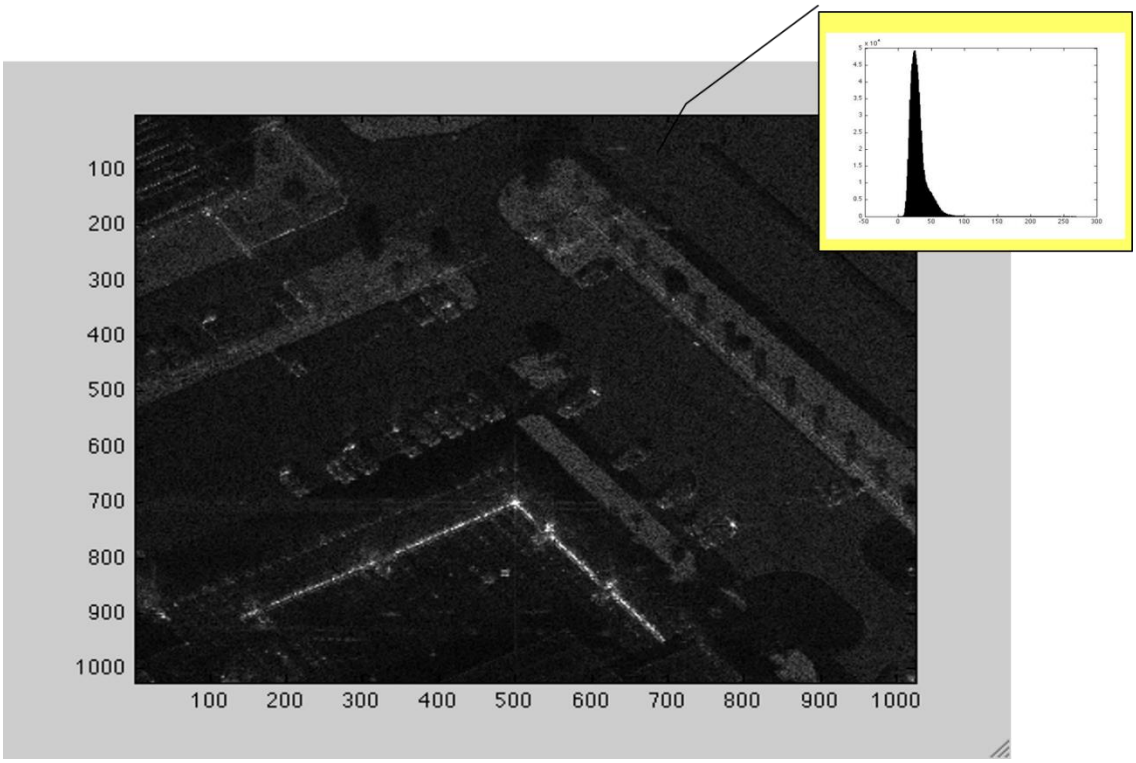

Fig. 7. Original SAR image (a)

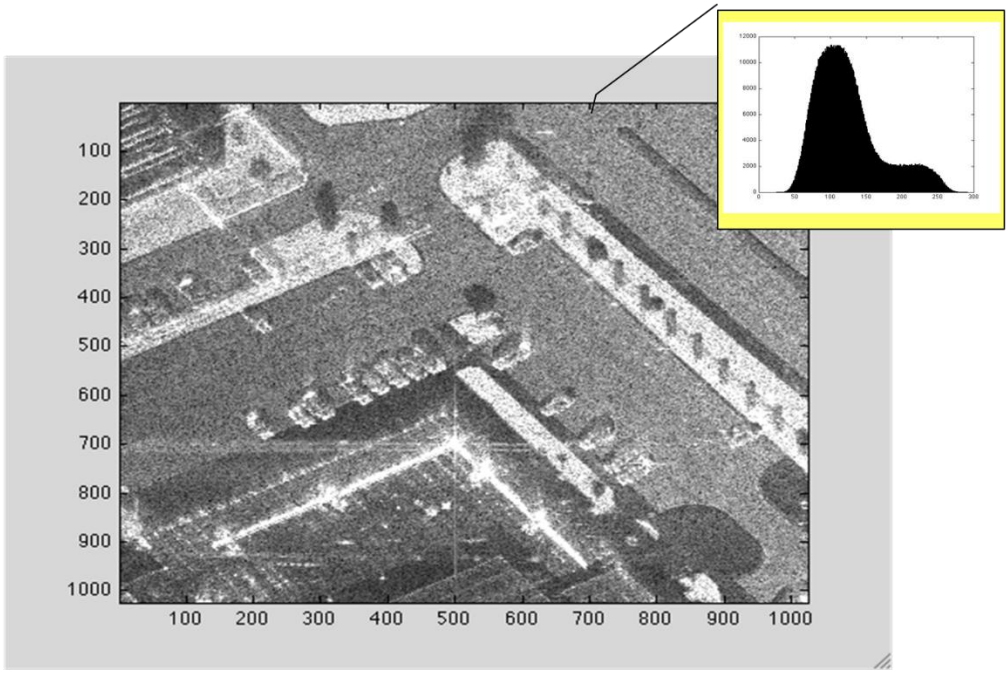

Fig. 8. Log-transformed SAR image (b)

The impact of the logarithmic filter is presented in Fig. 7 and Fig. 8. Restored SAR images were compared with and without a logarithmic filter. These image results verify that the filtered image boosts low frequency signals and exhibits an effective reduction of dynamic range. It is clearer to the human observer because of this dynamic range reduction. We have verified that logarithmic filtering produces a better SAR image when it is restored using the inverse DWT.

$$\begin{aligned} Q = log_e(x) &= c_1\ x + c_2\ x^2 + c_3\ x^3 + c_4\ x^4 \\ &+ c_5\ x^5 + c_6\ x^6 + c_7\ x^7 \end{aligned}$$

$$\begin{aligned} log_e(x) &= (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} \\ &- \frac{(x-1)^4}{4} + \frac{(x-1)^5}{5} - \frac{(x-1)^6}{6} \\ &+ \frac{(x-1)^7}{7} \end{aligned}$$

We used the Taylor Series to compute the natural logarithm on the FPGA. We selected the logarithmic transform since it was integral to image enhancement. The Taylor series is a representation of a function using an infinite sum of terms that are calculated from the values of the function's derivative at a single point. We truncated the infinite summation to eight terms as adding more terms did not increase overall accuracy. The output Q is calculated as shown in the equations above.

Our case study for the logarithmic hardware was synthesized and mapped onto an FPGA in order to validate our performance and resource usage in terms of maximum clock frequency and slice utilization. Synthesizing the model on the FPGA provided the opportunity to extract performance estimates including, throughput, power, and resource usage.

Our design was synthesized using the ISE 9.2 design tools. We executed the function on an H101-PCIXM board containing a Xilinx Virtex-4 LX100 FPGA. We implemented a design with two logarithmic cores, executed it on the FPGA board, and measured execution time. Since the maximum clock frequency was 100MHz, two logarithmic cores generate two outputs every 10 nanoseconds (ns). An implementation with four logarithmic cores generates four outputs every 10 ns.

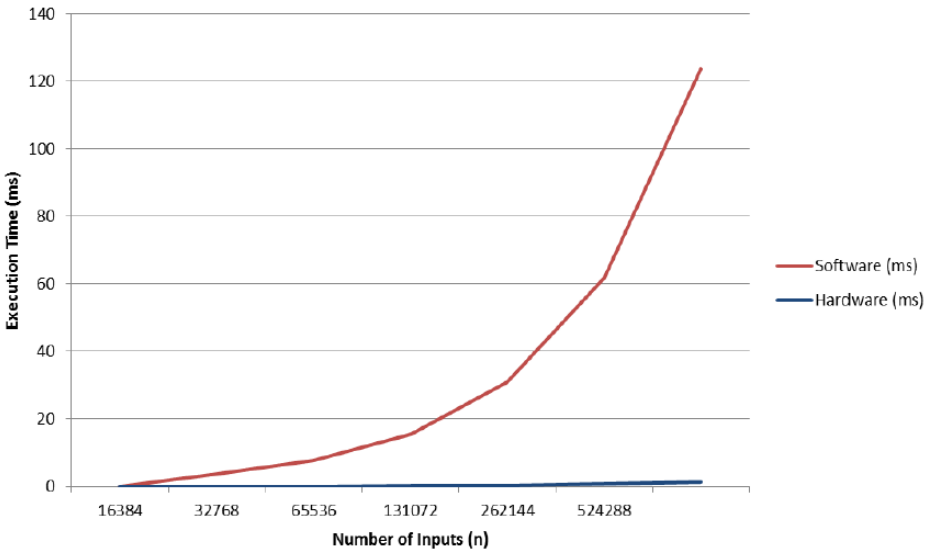

Fig. 9. FPGA performance comparison with software execution

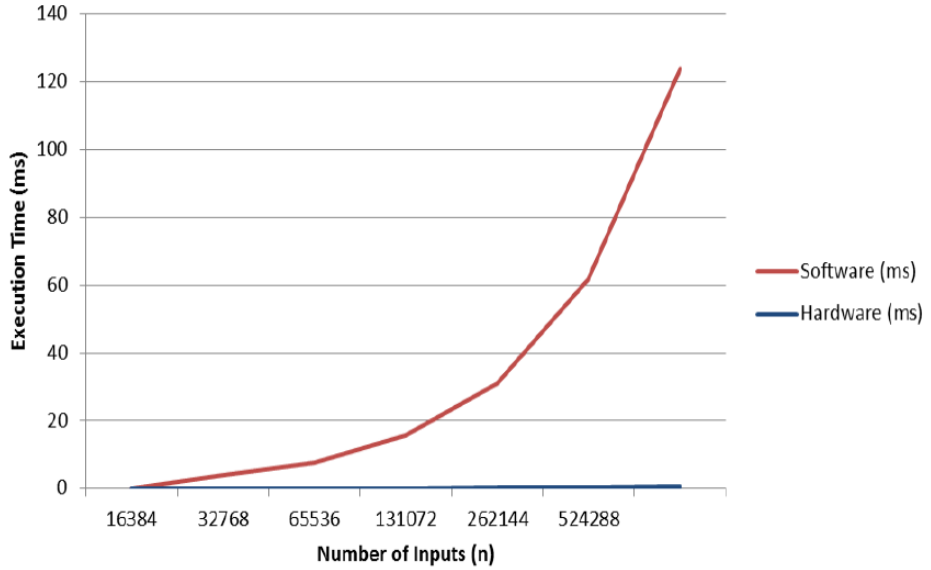

Fig. 10. Estimated four logarithmic unit on an FPGA results

Figure 9 shows our results with two logarithmic units on the H101-PCIXM board. The x-axis shows the number of input samples streaming through the FPGA. The y-axis shows the execution time in milliseconds. The use of our logarithmic unit resulted in an average speed-up of 75 for the hardware implementation over the software. We estimated performance for four logarithmic units on an FPGA to achieve the throughput shown in Figure 10. While we did not execute a four logarithmic unit core on the board, synthesis results verified that it would fit on the FPGA. The results using four logarithmic units showed an average speed-up of 188.

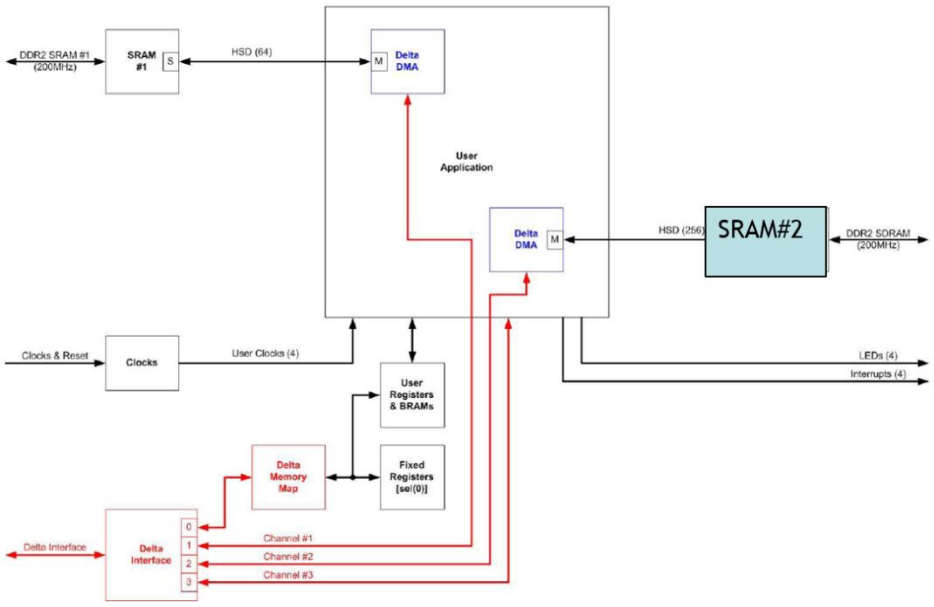

Fig. 11. Updated Memory Controller Design

We are extending our previous RARE implementations based on H101-PCIXM board by creating and porting memory controllers for three PCIe-180 boards we purchased in the second year of this project. Fig. 11 shows the memory controller entity that communicates with two SRAMs using the memory map in order to store input values for the accelerator (e.g. logarithmic, exponential) circuit. The controller must also be able to read outputs from SRAMs when requested by software. To be able to read/write to a register in the FPGA from the host processor, you need some way of assigning the targeted register/signal an address. For each transfer, a memory map node specifies the address that is being accessed, and whether or not the access is a read or a write.

To use a memory map node, you must first assign an address to each register/signal that you want to access from the host processor. Now that you have assigned an address to each signal in the FPGA you need to access from software, you need to design the memory controller logic to decode the signals from the memory map node and read/write the appropriate signal/register. Likewise, when the memory map specifies that a read is taking place, the glue logic should read data from the appropriate signal and send it to the memory map.

Streaming Architecture for FPGAs

In today's world there are number of applications which require video streaming. For example traffic monitoring, face detection, motion estimation, remote sensing, synthetic aperture radar, etc. All these applications need continuous streaming of video frames. Efficient streaming

architecture/framework is based on certain factors such as, the camera used, streaming architecture/algorithm, resolution, bandwidth, hardware used to implement the architecture. There is one more important factor responsible for efficient video streaming and that is memory I/O copying.

FPGA is a semiconductor device that can be used to program desired hardware functions and applications. FPGA gives hardware designers the benefits of achieving high performance as well as re-configurability. HW designers can reconfigure hardware blocks in FPGA for specific applications or kernels in their application algorithms. While designing hardware unit using FPGA the memory optimizations plays important role. In order to provide accelerated framework memory hierarchy need to be taken into consideration. For example, for any streaming application the memory latency will be increased if video frames are copied from off chip memory to kernel and back to memory. This will affect the overall framework. The copying from on-chip and off-chip memory has major effect on the framework acceleration.

We performed one experiment to determine the time required for memory I/O operations. The processing element (PE) used in our experiment is Sobel Filter hardware provided by Xilinx. Sobel Filter is simply an edge detection algorithm. So we used this sobel filter with streaming architecture to determine the amount of time required for memory I/O operation. The result we got are shown in Table 1. From the result, it is clear that the time required for memory I/O is much greater than the time required by a processing element to process the data. Our experimental setup is shown in Fig. 12.

Multi-Processor System-on-Chip (MPSoC) is widely used to implement streaming architecture. MPSoC is well suited to satisfy the computing requirements of streaming based applications. Even today FPGA lacks optimized memory architecture. The latency varies according to memory hierarchy. Most of the time the designer creates application specific streaming architecture

Table. 1 Execution time required for HW Sobel routine

Routine (HW Sobel)	Exe. time (s)	% of time
Frame capture	0.095	2.23
Copy from memory	1.885	44.33
HW Sobel filter	0.007	0.16
Copy to memory	2.265	53.26
Total	4.252	100

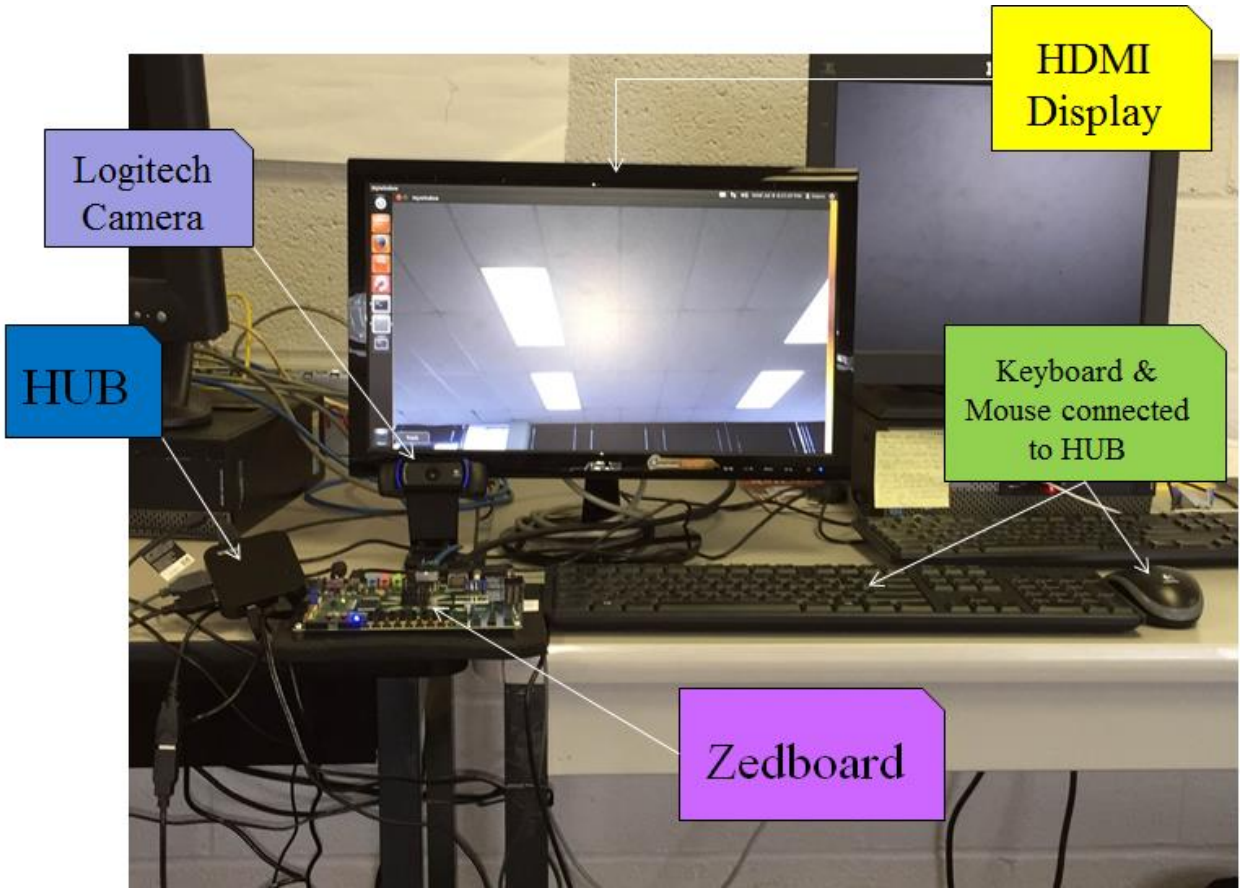

Fig. 12 Experimental Setup

Streaming applications such as remote sensing, hyper-spectral remote sensing, stereo vision and synthetic aperture radar operate on high resolution video frames. To process these high

resolution frames high processing power is required. Today's processing elements have made tremendous improvements in processing speed over the last few years of processing elements. But the memory access speed has not improved in comparison to the processing element. This has led to a performance gap between the processing element and the memory subsystem.

The streaming architecture is one of the more efficient ways to optimize the memory transfer operation. CoRAM proposes FPGA memory architecture to serve as a portable bridge between the distributed computation kernel and the external memory interface [7]. They have provided shared, scalable memory architecture for FPGA computing devices. But incorporating CoRAM in the computing devices would would require more area and greater power overhead.

The memory hierarchy based on scratchpad memories therefore load/store latency can be determined at compile time is discussed in [9]. The L0 scratchpad is the lowest in memory hierarchy and closest to processing elements (PE) allowing for fast delivery of pixels to PE. The L1 scratchpad deals with the low bandwidth towards off-chip memory.

OpenCV library functions are used to build the framework for video processing application [11]. This paper compares the execution time required using different instances of memory copy operation (e.g. Linux RAM to DMA RAM, etc.). This paper also introduces NEON processor where NEON copy operations prove to be efficient for memory copy. There are certain factors that affect hardware systems used for image/video processing application.

CbIA trades image quality for reduced cost of image acquisition process [8]. CbIA alters the conventional memory access pattern, in order to progressively and adaptively access pixels from a memory subsystem. CbIA addresses the cost of image acquisition using the execution time and energy consumption required for the hardware system.

Mixture-of-Gaussian (MoG) architecture has been proposed in [2]. MoG architecture implementation is used for realizing background subtraction. Authors propose parameter compression to achieve a trade-off between image quality and memory bandwidth. With the acceptable loss in image quality they were able to achieve reduction in memory bandwidth. But due to compression there is increase in computation.

Smart camera is used to accelerate two very different image processing applications in [13]. The author proposes minimizing movement of large datasets and minimizing the latency by starting to process data before a complete frame has been acquired. But they do not process streaming data directly. They block up and store data from camera in on-chip memory before processing it on the FPGA.

Our work is related but differs from above discussed work. We are building a streaming architecture which can directly handle streaming data. Also we are developing line-based FIFO to process the pixels from the streaming frames in order to reduce memory latency and increase throughput.

We propose a novel streaming architecture using a line based First In First Out (FIFO) approach. Our framework captures the video frames using a webcam which are processed using line based FIFO. The pixels from the video frames are copied line-by-line into the FIFO. Using this streaming architecture memory input/output (I/O) latency is reduced. This architecture can be used with other kernels which require streaming for their application.

We are building a framework featuring a streaming architecture, shown in Fig. 13, using Zynq 7000 processor. For experimental purposes we are using Logitech camera for video capturing. We are designing our streaming architecture framework using the FIFO approach. Using the FIFO approach, memory bound operations can be optimized and thus enhance the video processing application. The pixels from video frames captured using a camera will be processed using FIFO based streaming approach.

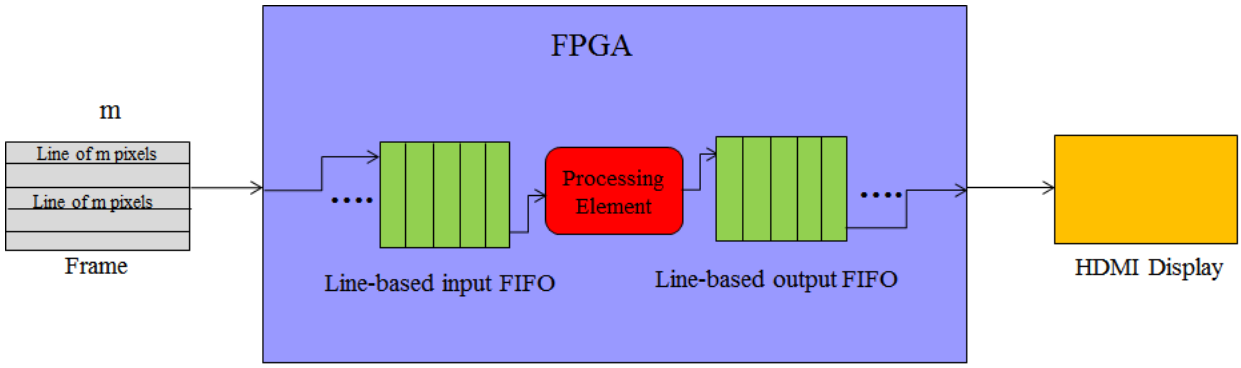

Fig. 13. Line-based FIFO Architecture

We are developing novel line based FIFO architecture for frame pixel processing. Depending on the frame resolution the kernel must be modified to yield high performance. In FIFO architecture the pixels from the frame will be copied to input FIFO line-by-line. This FIFO architecture can have variations in pixel copying i.e. copying 1, 2, 4, etc. lines of pixels at a time. These pixels will be further handled by the processing elements and will be copied to output FIFO line-by-line. These processed pixels will then be displayed on HDMI output.

Therefore using FIFO architecture video streaming the memory bound operations will be accelerated and latency for memory copying will be reduced. This will result in increase in throughput and the performance of video processing application will be increased.

Before proceeding with the framework we need to consider different aspects that will impact the framework. In short we should do a mathematical study of the framework. There are certain factors which will have an effect on the framework. Factors like data rate, clock rate, frame resolution, FIFO filling rate, etc. The speed at which camera is streaming data to frame grabber is called data rate. The following equations are used to calculate data rate.

Data Rate = Clock Rate x Number of bits per pixel

Once data rate is calculated, we can then determine whether our processing element can handle the frame acquisition. Also we need to consider throughput from the frame grabber over the bus. If the throughput is less than the data rate, then we have to consider the rate at which the FIFO will fill. If the FIFO fills up before a complete frame is acquired, there will be buffer overflow, even if we are only acquiring one frame. The Following equation is used to calculate FIFO filling rate.

FIFO Filling Rate = Data Rate - Bus Throughput

In analyzing the framework there is one more factor we must consider. We need to determine the time it takes to capture one frame from the camera.

One Frame Time = Resolution / Clock Rate.

FIFO filling rate and the one frame time can be used to calculate the FIFO prerequisite for one frame.

FIFO Prerequisite = FIFO Filling Rate x One Frame Time

To implement streaming architecture we need to install Open Source Computer Vision (OpenCV). OpenCV is a library consisting of 2500 computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, etc.

We are using the sobel filter provided by Xilinx to implement streaming architecture. We need to partition the SD card that is used with the Zynq 7000 board. An SD card is partitioned into two segments. In one part, boot files are copied and in other the Linaro Ubuntu file system is ported. OpenCV is installed from the repository of the Linaro Ubuntu. For our experimental purposes we are using Logitech HD pro C920 webcam. This webcam can be replaced by high resolution image sensors.

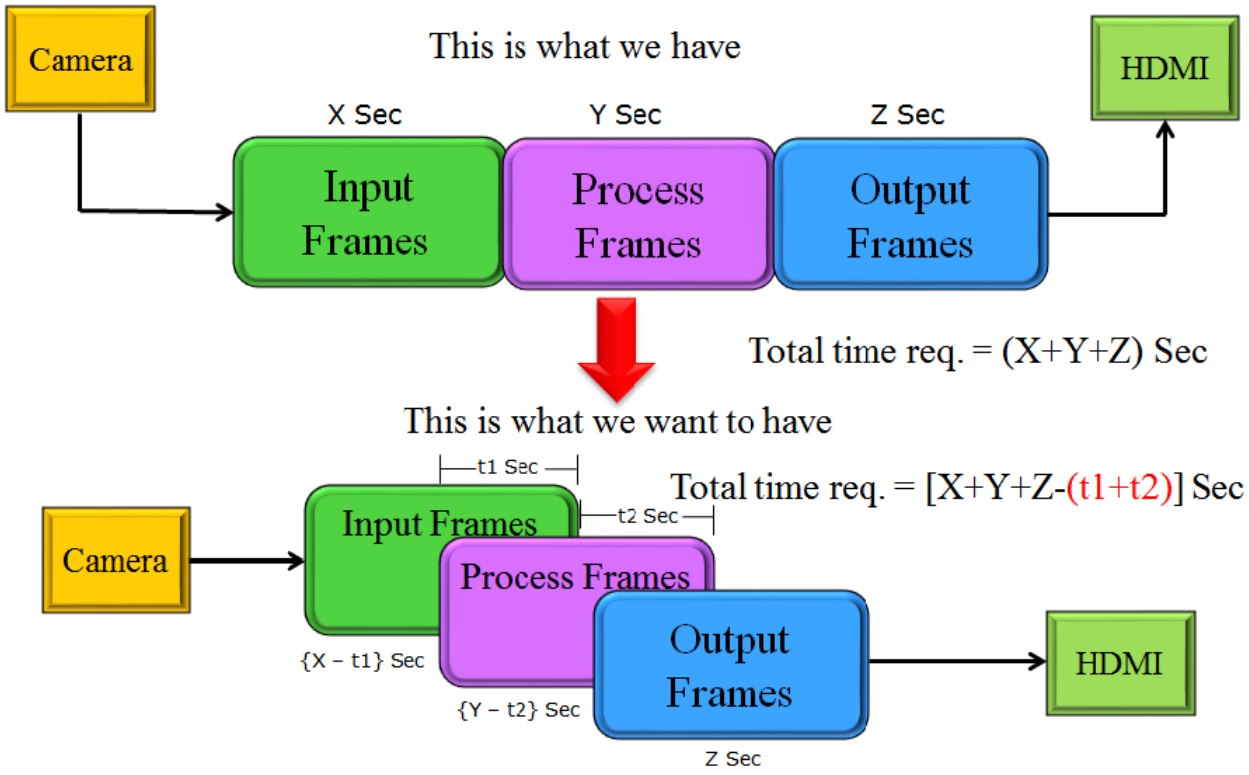

Fig. 14. Our approach for streaming architecture

Fig. 14 above illustration shows the approach that we are use to implement streaming architecture. As shown in the diagram, if we are waiting to process the pixels in frame in each block more time will be required to process the frame. This will lead to an increase in latency and a decrease in throughput. But if we start processing as soon as we receive the pixels and the subsequent blocks are not waiting for the entire frame we are going to save some time. By using this approach, latency will decease and throughput will increase. As seen in the diagram as soon as we get the frames from the camera all of the blocks involved will start processing the pixels and will send the output to the HDMI monitor. So by using our approach (t1-t2) sec time required is less than conventional method. Thus this will help to reduce the latency and increase the throughput.

Real-time Object Detection

Object detection is quite challenging considering both software and hardware implementation. There are several algorithms [15] [16] [17] [18] that are developed earlier considering the input as an unknown image and it can have different Poses, clothes, carry different objects, sizes, heights, background, illumination, pedestrian and camera motion, at different viewing angles. Viola Jones [19] is a very popular face detection algorithm but it can't accurately detect the objects in images if the image has different illuminated conditions. The histogram oriented gradient algorithm [20] provides reliable results with many of the state of art of object detection

tools. But the major objective is the achievement of minimal cost with superior accuracy. The implementation of HoG is possible in software and on hardware platforms.

The previous work shows the implementation both in OpenCV and in MATLAB. For hardware implementation the FPGA various platform is chosen. The mathematical functions of HoG implementation are quite challenging. Using IP cores takes more resource utilization and implementing its own mathematical function makes the device slower. As usual, there is a tradeoff among speed, accuracy and computational cost.

$$m = \sqrt{{g_x}^2 + {g_y}^2} \qquad (1)$$

$$a = arctg\left(\frac{g_y}{x}\right) \qquad (2)$$

$$v = \frac{v}{\sqrt{{||v||_2}^2 + €^2}} \qquad (3)$$

A histogram of oriented gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI). This algorithm was first described in Wolfram [21]. There are several steps for implanting the HoG both in both software and hardware. For software, available C++ implementation of OpenCV library was used. The implementation procedure of HoG has several steps [23]. The first step is to divide the image into small connected regions called cells, and then to compute for each cell a histogram of gradient directions or edge orientations for the pixels within the cell. The gradient magnitude and directions are showed in equations [15] and [16]. The second step is to discretize each cell into angular bins according to the gradient orientation. Each cell's pixel contributes a weighted gradient to its corresponding angular bin. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms. A normalized group of histograms represents the block histogram showed in equation [17]. The set of these block histograms represents the descriptor.

Block Design

The basic block design of Histogram Oriented Object detection [17] is shown in Figure 15. The module has several parts. This paper will focus on the gradient portion. The gradient computation [17] has two parts, magnitude calculation and direction calculation. This paper presents the design of gradient calculation direction and the result where it shows how much image detection accuracy is gained.

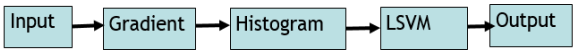

Fig. 15. HoG Object Detection module.

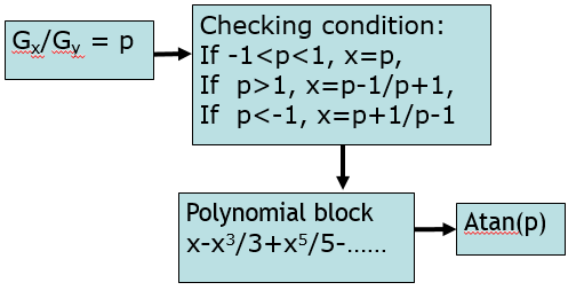

Fig. 16. Proposed Gradient Direction Computation module.

Previously for implementing the gradient direction LUT [16] [17] [18] and taking the ratios are used [15]. The advantage of LUT is its exceptional speed. The disadvantage is that it requires significant memory, especially if you need high resolution for the function input. The y/x ratio method doesn't yield an accurate result even though it is very simple to implement, faster and requires less memory. The error result of this method is shown in Table 2. Here, the proposed arctangent mathematical block is used instead of the IP core. The MaClaurin series are used for calculating the polynomial block shown in Figure 16.

$$\arctan(x) = \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{2i+1}$$

, where -1 ≤x≤ 1 and n=0, 1, 2, 3……..

To calculate the arctangent the the Maclaurin series can be used with the following conditions.

$$x = p$$

, when p <= 1 && p >= -1

$$x = \frac{p-1}{p+1}$$

, when p >= 1

$$x = \frac{p+1}{p-1}$$

,*w*hen p <= -1

The result from Arctangent Mathematical Block implementation that is discussed above is shown in Table 1. This result is found by using the MATLAB tool. The arctangent function is calculated by both MATLAB 'atan' function and Maclaurin series (where n=4) and the error is estimated between these two results. This calculation is done for 0-360 degrees with 10 degree interval. Maximum, minimum and average error is listed in Table
2. The maximum error found in whole range is .0495 (in radian) and minimum is error. Another noticeable thing about the result is this error has one trend, for lower angle the error is less and for higher angle the error is higher.

Result and Error Accuracy

The described Arctangent block was tested before implementing to HoG block. The result obtained from the implementation of the proposed Arctangent block shows that using higher values of n gives greater accuracy. But the problem is that a higher value of n requires more resources and yields higher output latency. In Table 2 the estimated error is shown where n=4. The error is estimated comparing the results between arctangent original value and the proposed block. This estimation is done for 0-360 degree ranges with 10 degree interval and it is found that with each 180 degree difference all the angles has repeated value. The maximum error found in the entire range is .0495 (in radian) and minimum is 0. Another noticeable issue is that this error is less for a lower angle the error is greater for a higher angle. The average error from the all angles found from Arctangent block is .002338.

In order to justify the error compared to the previous method [16], the previous Gy/Gx method was implemented. The result is shown in Table 3 and the result says the absolute error is larger than 1.

Table 2: Estimated Error from Arctangent Block

Angle Range	Max	Average	Min
1—10 & 181--190	4.54E-10	6.42E-11	0
11—20 & 191--200	1.21E-06	2.59E-07	1.32E-09
21—30 & 201--210	1.69E-04	4.62E-05	2.16E-06
31—40 & 211--220	8.30E-03	2.60E-03	2.57E-04
41—50 & 221--230	3.48E-02	8.83E-03	0
51—60 & 231--240	4.38E-08	7.79E-09	1.56E-12
61—70 & 241--250	1.74E-05	4.28E-06	9.16E-08

71—80 & 251--260	1.28E-03	3.79E-04	2.81E-05
81—90 & 261--270	3.48E-02	NaN	1.88E-03
91--100& 271--280	3.50E-02	1.10E-02	1.2E-03
101—110 & 281--290	8.65E-04	2.53E-04	1.74E-05
111—120 & 291--300	1.06E-05	2.54E-06	4.38E-08
121—130 & 301--310	1.99E-08	3.41E-09	2.07E-13
131—140 & 311--320	4.95E-02	1.46E-02	0
141—150 & 321--330	5.76E-03	1.79E-03	1.69E-04
151—160 & 331--340	1.10E-04	2.95E-05	1.21E-06
161—170 & 341--350	6.67E-07	1.38E-07	4.54E-10
171—180 & 351--360	1.40E-10	1.88E-11	0

Before proceeding to hardware implementation using the proposed block, our goal is to justify this HoG block using C++ and compare it with the face detection in OpenCV. As a part of this work, loading image in Fig. 17, video streaming in Fig. 18 and face detection in Fig. 19 are accomplished using source code. In the case of face detection the program is tested using different resolutions of image e.g., 8 bit, 16 bit, 24 bit and 32 bit. The detection time is less for higher resolution image but for 32 bit the detection time is greater. The result is shown in Table 4. So, there is a tradeoff between image resolution and detection time for faster face detection. As the final goal of this project is to use floating point operation for faster face detection with higher accuracy, the number of bits can be decided after implementing the full HoG using the proposed block.

Table 3. Estimated Error implemented from previous Gy/Gx method

Ratio Method(atan)	Exact Value(atan)	Error
0	0	0
5.710593137	4.5	1.210593
11.30993247	9	2.309932
16.69924423	13.5	3.199244
21.80140949	18	3.801409
26.56505118	22.5	4.065051

30.96375653	27	3.963757
34.9920202	31.5	3.49202
38.65980825	36	2.659808
41.9872125	40.5	1.487212
45	45	0

Table 4. Face Detection time from different size of Images.

Image Type	Detection time (ms)
8 bit	218.513
16 bit	215.283
24 bit	213.659
32 bit	217.83

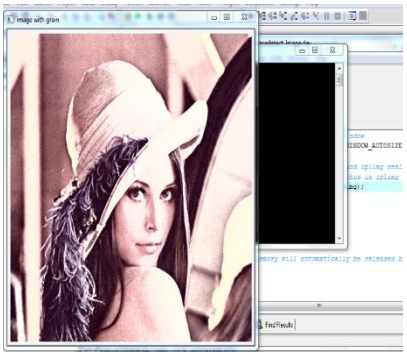

Fig.17. Loading Images from stored image

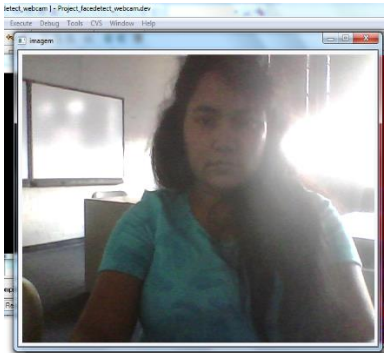

Fig.18.Video streaming image from

Fig.19. Face Detection from lena Images.

In this project report, a working system is presented for multiple object detection using existing HoG block and OpenCV source code. Though the software and hardware implementation is not yet complete. the result from the comparison of Tables 2 and 3 shows that the accuracy of the proposed arctangent mathematical block is higher than with previously considered methods [15].

Floating point math computation acceleration

Enhancing the execution of certain high performance applications by increasing speed and lowering power consumption is a primary objective of the research conducted here in the RARE Group. This can be achieved by coupling hardware designs with a High Performance Computing (HPC) hardware platform such as Graphic Processor units (GPUs), Field-Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs) and many more. Here in the RARE group we focus on using FPGAs due to major benefits such as low cost and reconfiguration at any point, as compared other HPC platforms.

The RARE toolset consists of floating-point arithmetic cores from simple designs such as adders, multipliers and subtractors, to more complex designs such as exponential units, logarithmic units, square root units, Power units and many more. In addition, these Floating-point designs are used in high performance applications such as: Synthetic Aperture Radar (SAR), Polynomial Approximation, Polynomial Approximation, High Dynamic Range Imaging, weather prediction etc.

The first experiment was to make initial improvements to a few components within the RARE toolset, which would then be used in a test to compare the performance (speed and power) of the RARE tools to the performance of identical floating point components designed by an outside source. My second experiment of the summer serves as a continuation of the initial comparison conducted in the first experiment. The second experiment consisted of porting the designs of both the outside source and RARE onto an actual FPGA and again documenting their performance. The third and final experiment of the summer was to design an error module that is implemented inside of the Power unit used in the RARE toolset.

Our initial benchmarking experiment was to run synthesis and record the speed and power of the RARE Floating-point exponential, logarithm, and power unit. Next a comparison of the performance of identical components designed by FloPoCo was conducted. FloPoCo, which was founded and is maintained by a group of professors and scholars at INSA Lyon in France, has presented a way to generate arithmetic cores, ranging from simple to complex units, which are fully parameterized and pipelined for FPGAs and other hardware platforms. FloPoCo generator is based on a C++ platform which is used to create the arithmetic logic cores. Along with using C++, FloPoCo provides a command line interface that is used to convert user specific inputs into synthesizable VHDL. What makes FloPoCo unique is that their generators are tuned to design very precise arithmetic cores in relation to accuracy, and cores that are fast and power efficient. Based on the results, this experiment serves as potential step in the RARE group being recognized at a higher level within the FPGA community.

The preliminary setup of this experiment involved making baseline improvements to the exponential unit, Logarithm unit, and Power Unit. This task required making enhancements that catered to increasing the speed and reducing the power consumption of each component. This was achieved by strategically removing certain areas of the design that were redundant or not needed.

The FloPoCo comparison experiment was conducted using the Xilinx ISE design software which was installed on a regular Laptop computer. This third party software is dedicated to VHDL design and FPGA design. This software allows users to design and maintain hardware components. Features are included that allow users to view a visual representation of a designed hardware component in the form of a schematic. The user is then able to simulate a design, giving an accurate account for resource allocation and performance without actually implementing the design on a physical FPGA board. For this experiment, the simulation feature was used to produce findings.

To initiate the experiment, a simulation of each component design was run for the Virtex 7 XC7VX485T -1 FFG1157 device. This is the latest device that the Xilinx ISE software offers. Also the Virtex 7 board is one of the newest boards on the market and offers more than enough memory and resources to handle the given designs being tested. Below, are the tables that represent the findings for both speed and power performance for each design simulated on the Virtex 7 board.

Table 5. FloPoCo Vs RARE – Power

	FloPoCo		RARE	
Logarithm Unit	Slices	LUTs	Slices	LUTs
	836	***1034***	***7879***	***13363***
Exponential Unit	***501***	***662***	***1818***	***3542***
Power Unit	***2180***	***2751***	***11084***	***18533***

Table 6. FloPoCo Vs RARE - Speed

	FloPoCo		RARE	
Logarithm Unit	Min Period	Max Freq.	Min Period	Min Period
	4.734ns	***210.853MHz***	***4.161ns***	**240.327MHz**
Exponential Unit	***4.431ns***	***225.695MHz***	***3.660ns***	***273.224MHz***

Power Unit	*4.840ns*	*206.606MHz*	*4.168ns*	*239.923MHz*

Table 5 displays the number of "Slices" and "Look up Tables (LUTs)". The Virtex 7 is capable of holding 607,200 slices and 303,600 LUTs before it reaches capacity. When measuring the power of a FPGA or hardware design, these two categories represent the resource allocation. A power efficient design uses as few resources as possible. As shown above, the components designed by the FloPoCo generator use a significantly lower amount of resources compared to the RARE components which means that the RARE tools use a great deal more power than the FloPoCo tools.

Table 6 displays the "Maximum Frequency" and "Minimum Period" of each design. These two categories are primarily gauged when monitoring speed. In the case of speed, a component that is faster would have a lower minimum period and a higher maximum frequency. Unlike the power results, the RARE tools are slightly faster than the FloPoCo designed tools. Considering both power and speed, The FloPoCo designed tools would be the best choice for the present. Sacrificing a little speed is a good trade off when considering you're using significantly less power than the competition. These results serve as a baseline for further improvement of the RARE tools used in this experiment.

Power Unit and Error Module

We complete a Power Unit and an error module use in the project. The Power unit is a vital component within the realm of executing higher performance applications. Aside from computing basic power computations such as 2^3 or X^n, implementing this component in a larger scale design can present the potential for significant speedup. Figure 20 represents a simple block diagram of how the Power Unit works; as I move further along I will explain the underlying components that are included in the power units design.

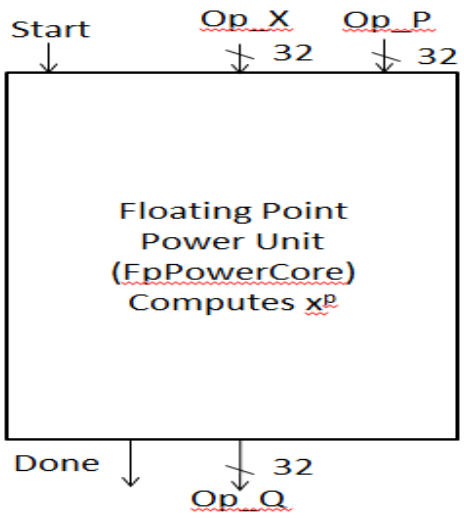

Figure 20. Power Unit First Level Block Diagram

Building the power unit is more complicated than it may seem. The equation to compute the power of a number is: $X^p = (e^{ln(x)})^P = e^{pln(x)}$,this equation must be designed in hardware in order to efficiently complete this computation. The power Unit is comprised of four components that are linked together to compute the power of a number. These components are as follows: Logarithm Unit, Exponential Unit, Multiplier, and a delay. These components are all included in the RARE library as well. Figure 21 below displays the second level block diagram of the power unit.

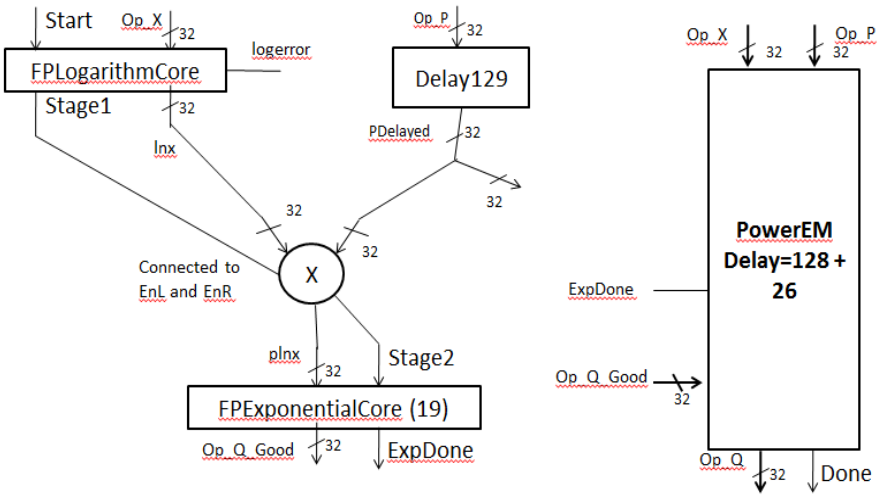

Figure 21. Power Unit Second Level Block Diagram

As shown in figure 21, the components are design to follow the direction of the equation. Also in the figure, one should notice a component that is named “PowerEM”. This is the Error Module. The purpose of this error module is to verify the inputs that are entering the Power Unit and give either a response of the actual finished value or an error. When computing simple power computations, there are cases where two numbers may produce an answer of infinity or an undefined number, which is referred to as Not a Number (NaN). The error module has to be able to account for all of these cases and accurately depict which given two numbers will produce a real output and which two numbers will produce an error. Some error cases include: raising a negative integer to the power of a negative non-integer ($-2^{-2/3}$), raising zero to the power of a negative integer(0^{-1}), raising zero to the power of a negative non-integer($0^{-2/3}$), etc.

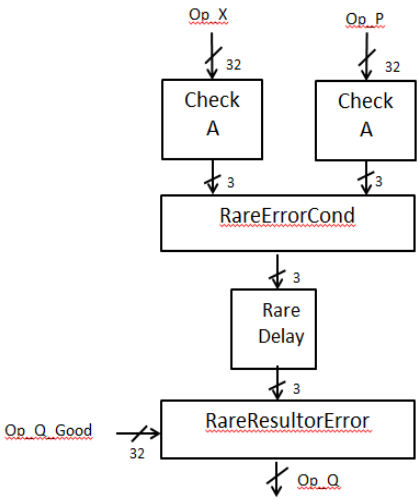

Figure 22. Error Module Second Level Block Diagram

Figure 22 shows the second level block diagram of the Error module and the process required for verification. Once two values enter the power unit those same values are entered into the error module as well. Next, they are sent through a series of components that conduct further evaluation based on conditions statements and cases. If everything complies, the actuall value is passed through as the result. If the two numbers trigger an error, the error value is passed through as the result.

Remote Execution Environment

The basis for the Remote and Reconfigurable (RARE) project is setting up a series of Reconfigurable Computers to be controlled remotely, or from a different location than where the computer is based. A Reconfigurable Computer is a Host PC connected to one or more Field Programmable Gate Arrays, or FPGAs. FPGAs are circuits that can have memory allocated to perform certain programmed tasks, and are more customizable that similar circuits.

RARE has been developed primarily through the Java programming language for several reasons. Its native method feature allows for some other languages to be called within Java applications. One reason this could be useful is because FPGAs might only be able to read a certain language, but Java offers the flexibility of using many similar languages. Other features include remote method invocation (RMI) and network security, which allows Java to be executed from a remote site (from one location to another), one of the main aspects of the RARE project.

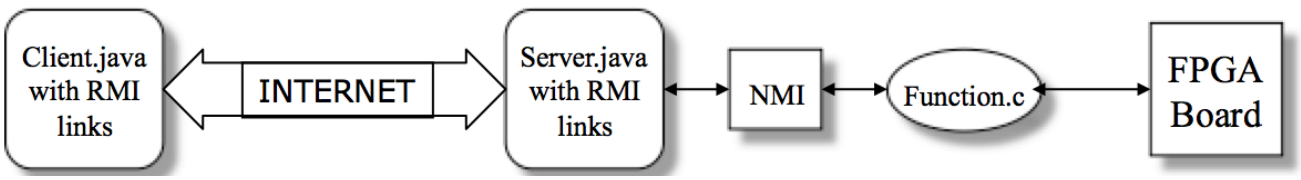

Fig. 23 RARE infrastructure

The infrastructure of the RARE project infrastructure is as shown in Figure 23. A client computer is connected with a server computer over the Internet as shown, and the server is intended to be connected to the FPGA board to run the program more efficiently. This sets up the remote connection, as well as running the program on the intended hardware. We now have the RARE environment running on our servers at NCAT. We hope to publish results for various applications running in this environment in the near future.

DOD Information Sessions

Two information sessions where held during this reporting period to stimulate student interest in DOD careers. The first information session took place on the September 24, 2015 at Smith Hall - Room 2014. The session started at 6:00pm and ended at 7:00pm with 44 students and 3 faculty members in attendance. Mr. Jonathan Stanton, who currently holds the position of the Contracts and Quality Team Lead for the Defense Contract Management Agency, was the guest speaker at the information session. This session focused on enlightening students and faculty members about career opportunities available with the Department of Defense.

The session started with participating students being welcomed and invited for light refreshments by Dr. Andrea Ofori-Boadu. The introduction of the guest speaker, Mr Jonathan Stanton, was by Dr Frank Yeboah. Mr Stanton proceeded to give students helpful information on job opportunities available at Department of Defense. He demonstrated through the use of a computer and overhead projector, the different steps and processes involved in applying for a federal position posted at the USA JOBS website. Mr. Stanton answered all questions asked by the participating students. The session ended at 7:00 p.m. with the vote of thanks given by Miss Mariam Adebakin. After the session, about ten students waited to further discuss specific issues with Mr. Stanton on a one-on-one basis.

PICTURES FROM THE FIRST DOD INFORMATION SESSION

LIST OF PARTICIPATING STUDENTS

Jonathan Bryant	Anthony Waters	Monique Priah
Shawnice Johnson	Aaron Coles	Justin Castro
Esther Majekodunmi	Travis Gould	Shavonne Duckett
Devon James	Kimberly Brunson	Isiah Magett
Elias Yanes	Rashaad Joyner	Mariam Adebakin
Steven D. Chestnut Jr.	Sage Fountain	Taylor Robinson
Pedro E. Ortiz	Anthony Marsh	Antwan Lock
Tanira Boone	Bien Vary	Derrius Todd
Ki'ona McKeever	Dwong Eran	Moyosore Abiodun
Raven Williams	Jalyn Caesar	Justin T. Keener
Marquivias Sutton	Brittany Hoover	Jacob James

Travis Wilson	Wasilat Usman	Frank Gronwald
Nzinga Hawkins	Chemeka Moore	Cara McAdoo
Lamont Patterson	Indya Strickland	Aaron Mayo
Shakira Austin	Jeremy Mowzzan	

LIST OF PARTICIPATING FACULTY MEMBERS

Dr. Andrea Ofori-Boadu
Dr. Frank Yeboah
Dr. Clay Gloster

The second information session took place on November 3, 2015 at Smith Hall - Room 2014. The session started at 6:00 p.m, and ended at 7:00 p.m. with 25 students and 2 faculty members in attendance. Mr. Jonathan Stanton, who currently holds the position of the Contracts and Quality Team Lead for the Defense Contract Management Agency, was the guest speaker at the information session focused on enlightening students and faculty members about career opportunities available with the Department of Defense.

The seminar started with the introduction of the guest speaker by Ms. Cara McAdoo. Mr Stanton proceeded to give students helpful information on job opportunities available at Department of Defense. He demonstrated through the use of a computer and overhead projector, the different steps and processes involved in applying for a federal position posted at the USA JOBS website. Mr. Stanton answered all questions asked by the participating students. The seminar ended at 7:00pm with the vote of thanks given by Mr. Devon James. After the session, about four students waited to further discuss specific issues with Mr. Stanton on a one-on-one basis.

PICTURES FROM THE SECOND DOD INFORMATION SESSION

LIST OF PARTICIPATING STUDENTS

Ahmed Lamarre	Anthony Waters	Daryl Bryant
Danielle Spearman	Aaron Coles	Arwa Qammash
Esther Majekodunmi	Travis Gould	Marklavious Patton
Devon James	Derrius Todd	Xavier Montgomery
Zachary Hinton	Avery Sandidge	Mariam Adebakin
Sasha Cheek	Armond Moore	Brittany Austin
Isiah Washington	Kevin Fisher	Nzinga Hawkins
Allen Nqean	Mitchell Gain	Cara McAdoo

LIST OF PARTICIPATING FACULTY MEMBERS

Dr. Andrea Ofori-Boadu
Dr. Jacqueline Chestnut
Dr. Clay Gloster

Bibliography

[1] Hamed Tabkhi, Majid Sabbagh and G. Schirner, “A Power efficient FPGA based Mixture-of-Gaussian Background Subtraction for Full HD Resolution”, 22nd International Symposium on FCCM, 2014.

[2] Hamed Tabkhi, Robert Bushey and G. Schirner, “Algorithm and Architecture Co-Design of Mixture-of-Gaussian Background Subtraction for Embedded Vision”, Proceedings of the Asilomar Conference on Signals, Systems, and Computers (AsilomarSSC), 2013.

[3] M. Sadri, Christian Weis, Norbert Weis and Luca Benini, “Energy and Performance Exploration of Accelerator Coherency Port using Xilinx ZYNQ”, FPGA World, 2013.

[4] Kerem Seyid, Vladan Popovic, et al, “A Real Time Multi Aperture Omnidirectional Visual Sensor Based on Interconnected Network of Smart Camera”, 22nd International, 2014.

[5] Bita Rouhani, Ebrahim Songhori, et al, “SSketch: An Automated Framework for Streaming Sketch-based Analysis of big Data on FPGA”, 23rd International Symposium on FCCM, 2015.

[6] Qi Tang, Twan Basten, Marc Geilen, et al, “Task-FIFO Co-Scheduling of Streaming Applications on MPSoCs with Predictable Memory Hierarchy”, ACSD 2015.

[7] Eric Chung, James Hoe and Ken Mai, “CoRAM: An In-Fabric Memory Architecture for FPGA-based Computing”, 19th International Symposium on FPGA, 2011.

[8] J. Liu, C. Bouganis and Peter Cheun, “Context-based Image Acquisition in Hardware Systems”, Journal of Real-Time Image Processing.

[9] A. Beric, Jef Meerbergen, G. Haan and R. Sethuraman, “Memory Centric Video Processing”, IEEE Transaction on Circuits and Systems for Video Technology, Vol. 18 No 4 April 2008.

[10] Harm Peters, R. Sethuraman, A. Beric, et al, “Application Specific Instruction-Set processor Template for Motion Estimation in Video Application”, IEEE Transaction on Circuits and Systems for Video Technology, Vol. 15 No 4 April 2005.

[11] Floris Driessen, “Throughput Exploration and Optimization of a Consumer Camera Interface for a Reconfigurable Platform”, University of Technology Eindhoven, 2013.

[12] Johann Hauswald, Michael A. Laurenzano, et.al, “Sirius: An Open End-to-End Voice and Vision Personal Assistant ans Its Implication for Future Warehouse scale Computers”, In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS),* ASPLOS ’15, New York, NY, USA, 2015. ACM

[13] Miriam Leeser, Shawn Miller and Haiqian Yu, “Smart Camera Based on Reconfigurable Hardware Enables Diverse Real-time Application”, 12thIEEE International Symposium on FCCM, 2004.

[14] Nikil Dutt, Puneet Gupta and Alex Nicolau, “Multi-Layer Memory Resiliency”, Proceedings of the 51st Annual DAC 2014.

[15] Mateusz Komorkiewicz, Maciej Kluczewski, Floating point HoG implementation for real-time multiple object detection, AGH University, Poland, IEEE, 2012

[16] Michael Hahnle and Matthias, FPGA-based Real-Time Pedestrain Detection on High-Resolution Images, Gueircke University, Germany, CVPR2013

[17] Jiye Duan, William MacDowell, Pedestrian Detection Image Processing with FPGA, Electrical and Computer Engineering, Worcester Polytechnic Institute, Project report submitted on April 30, 2014

[18] Xiaoyin Ma and Walid A. Najjar, “Evaluation and Acceleration of High-Throughput Fixed-Point Object Detection on FPGAs,” in IEEE Circuits and Systems for Video Technology, vol. 25. No.6, June 2015, pp. 1051-1062.

[19] Paul Viola, Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, Conference on Computer Vision and Pattern Recognition 2001

[20] Histograms of Oriented Gradients for Human Detection Navneet Dalal and Bill Triggs INRIA, France.

[21] http://mathworld.wolfram.com/MaclaurinSeries.html

[22] https://software.intel.com/en-us/node/529070