Accelerating DICe on FPGA

Status Update on Honeywell Research

Keaten Stokke & Atiyeh Panahi

Advisor: Dr. David Andrews

April 4th, 2019

Last Report

- DICe Custom Correlation IPs
  - Correlation on different image sizes
  - Up to 320x240 pixels (5.4 times smaller than needed)
  - **Correlation was 3 times faster than DICe GUI**
- Improved Arithmetic Functions
  - Improved from 100 MHz to 150 MHz
- Ethernet
  - Not Implemented
- Python scripts
  - Not Implemented

UNIVERSITY OF ARKANSAS

Accomplishments to Date

- Prototype end-to-end flow
  - Reduced image size due to hardware constraints
- Improved Arithmetic Functions
  - Reduced clock cycles per operation
- Ethernet Data Transfer
  - VC707: Unable
  - KC705: ~100 Mbps
  - Zybo: ~1000 Mbps
- Python Scripts
  - Handles transferring/receiving of data to/from FPGA via TCP
  - Image processing: Converts images to and from IEEE-754 single precision floating point format

UNIVERSITY OF ARKANSAS

Python Scripts

- Scripts handle data transfer from PC to FPGA through TCP client
  - Python scripts currently support any number of frames
- Python code can easily be converted to another high-level language
  - Image pre/post-processing adds overhead
  - Could reduce latency
  - C/C++/Java
- This step is completed, however small tweaks will be added throughout development
  - Python 2.7.15 is used

Python Scripts

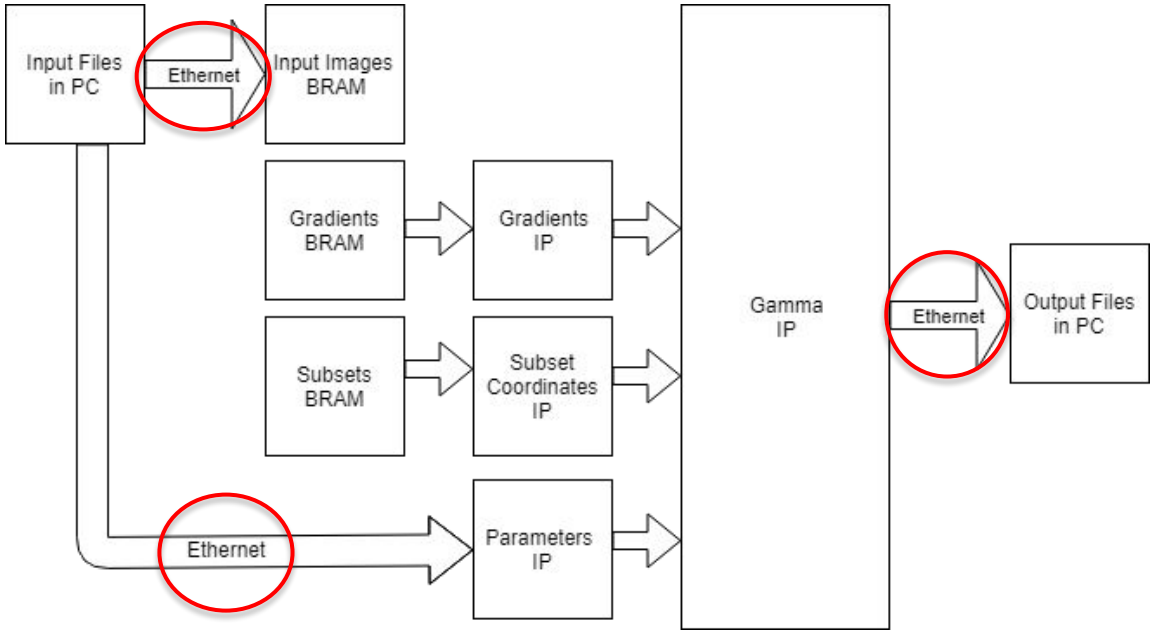
Pre-Processing

1. Read in all of the “.tif” images
   a. Value is between 0 and 255
2. Normalize each pixel in the image
   a. Divide pixel value by 255 for normalized value
3. Convert the pixels decimal value to IEEE-754 single precision floating point format
   a. Each 32-bit binary number is casted into an integer that represents the binary form in a 32-bit address space, but is less than 2,147,483,647 (max int value)
   b. I.e. Binary = 01001110011100000000001000000010 □ Integer = 1,006,665,857
4. Setup Client and establish TCP connection with the server
   a. Sticking with TCP for now to verify all data is received
   b. UDP is faster, but packet loss exists at 1000 Mbps speeds; ALL packets are needed
5. Send frame data from PC to FPGA as packets
   a. Begin image correlation on FPGA (not Python) using custom DICe IPs

Post-Processing

6. Receive results from FPGA via Ethernet
7. Convert results from IEEE-754 single precision format to scientific notation
8. Format the results and write them to “.txt” files

Ethernet


Input Files in PC
Ethernet
Input Images BRAM
Gradients BRAM
Gradients IP
Subsets BRAM
Subset Coordinates IP
Ethernet
Parameters IP
Gamma IP
Ethernet
Output Files in PC

UNIVERSITY OF ARKANSAS

Ethernet

- Virtex7 (MicroBlaze)
  - Capable of up to 1000 Mbps transfer speeds
  - IP: AXI Ethernet Subsystem (SGMII)
  - **Failed to get any connection working with design**
  - Not a practical option without a working IP reference design
- Kintex7 (MicroBlaze)
  - Capable of up to 1000 Mbps transfer speeds
  - IP: AXI EthernetLite (MII)
  - Only able to achieve 100 Mbps Ethernet connection with IP
  - LWIP overhead reduces max speed to 56.6 Mbps, we have 8 Mbps...
- Zybo (Zynq)
  - Capable of up to 1000 Mbps transfer speeds
  - IP: None (PHY IP)
  - Achieved 210 Mbps Ethernet connection **with ease,** 500 Mbps with buffer errors
  - Not practical for full design due to low BRAM (0.2 MB)
  - Used to demonstrate easy and fast Ethernet
- UltraScale+ MPSoC/RFSoC (Zynq)
  - Capable of up to 1000 Mbps transfer speeds
  - IP: None (PHY IP)
  - UltraScale+ maintains ease of use and BRAM sizes < 8.825 MB

Ethernet

- Lightweight IP (LWIP)
  - Currently the most prominent interface for FPGA-based Ethernet
  - Will explore using Petalinux as the Ethernet interface

Hardware Design Name	RAW Mode		Socket Mode	
	RX (Mb/s)	TX (Mb/s)	RX (Mb/s)	TX (Mb/s)
KC705_AxiEth_32kb_Cache	290	190	52.9	56.2
KC705_AxiEth_64kb_Cache	380	250	58.4	69.5
KC705_AxiEthernetlite_64kb_Cache	46	67	29	44
ZC702_GigE	943	949	521	542

- Zynq is the only processor capable of ~1 Gbps speeds!
  - Xilinx 10G IPs are unusable because PC only supports 1 Gbps (expensive hardware)
  - 10,000 FPS = 16.63 GB of data!

* Table from Xilinx Reference Guide

FPGA Comparisons

- So which FPGA is preferrable?
  - Zynq processor
  - The more BRAM, the better (6.5 MB is optimal)

FPGA	BRAM (MB)	Ethernet IP	Protocol	Speed (Mbps)
VC707	4.5	Ethernet Subsystem	SGMII	X
KC705	2	AXI Ethernet Lite	MII/GMII/SGMII	~100
Zybo	0.2	PHY IP	MII	~1000
UltraScale+	< 8.825	PHY IP	MII	~1000

FPGA Comparisons

Echo Server

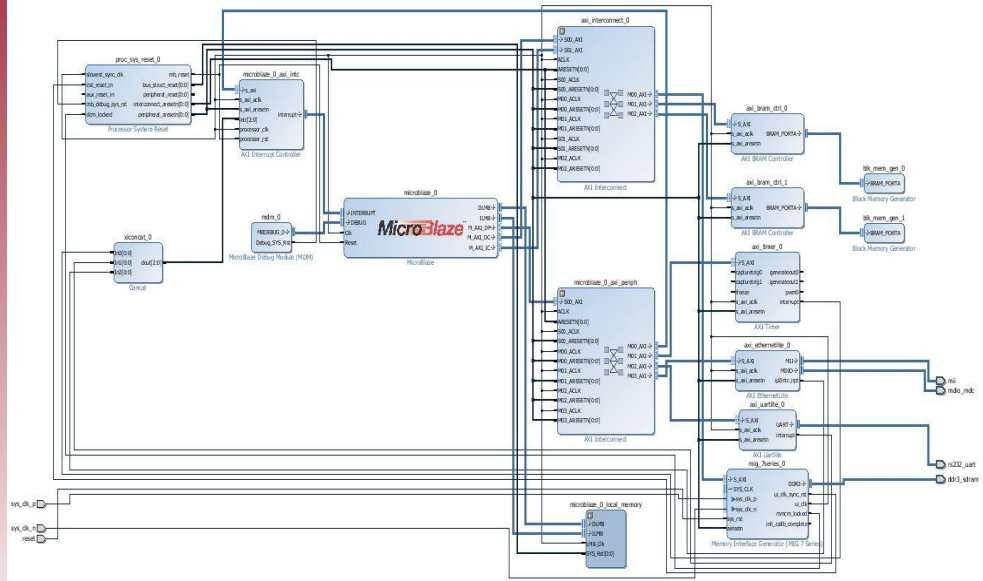
- MicroBlaze vs Zynq
  - Reading .tif files (x2 64*48 images)
  - Convert .tif to floating point
  - Sending the frames
  - Writing the images to BRAM
  - Echoing them back

FPGA	Transfer Time (s)	Total Exe Time (s)	Speed (Mbps)
KC705	1.12	1.34	100
Zybo	0.09	0.30	1000

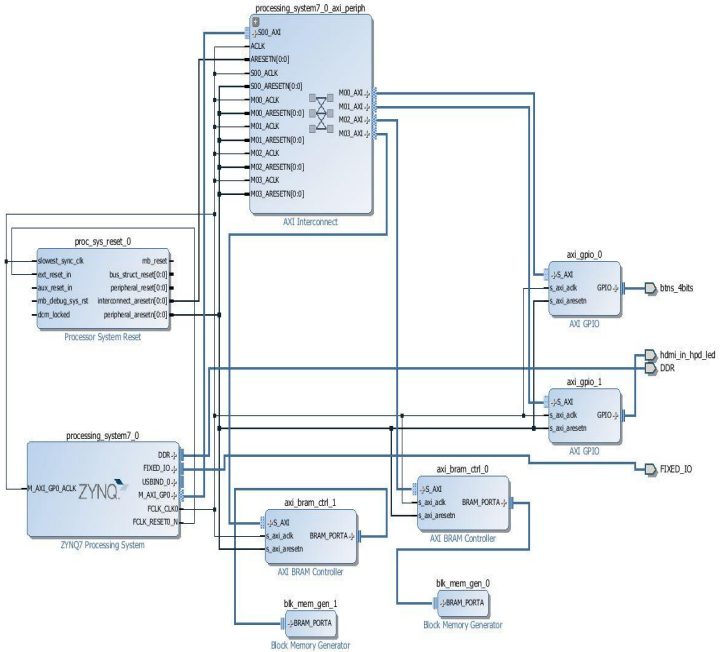
FPGA Comparisons

Block Design

KC705 (MicroBlaze) vs Zybo (Zynq 7010)



MicroBlaze



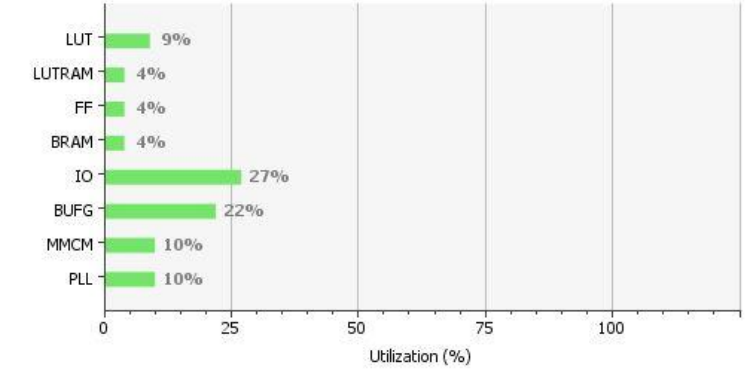
Zynq

FPGA Comparisons

Echo Server

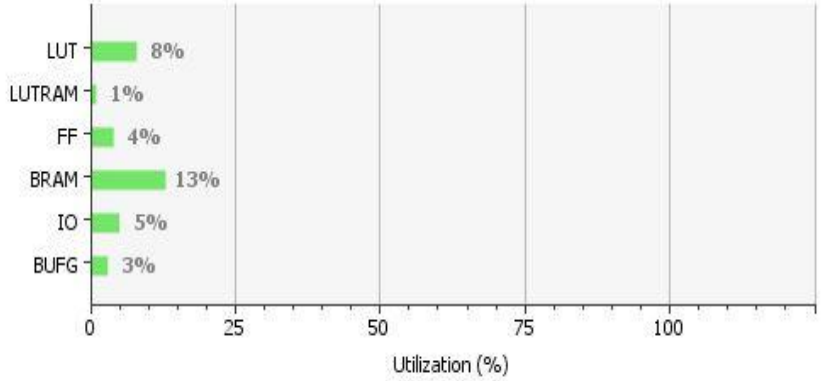
KC705 (MicroBlaze) vs Zybo (Zynq 7010)

Resource	Utilization	Available	Utilization %
LUT	18329	203800	8.99
LUTRAM	2595	64000	4.05
FF	15504	407600	3.80
BRAM	17	445	3.82
IO	137	500	27.40
BUFG	7	32	21.88
MMCM	1	10	10.00
PLL	1	10	10.00



KC705

Resource	Utilization	Available	Utilization %
LUT	1364	17600	7.75
LUTRAM	83	6000	1.38
FF	1413	35200	4.01
BRAM	8	60	13.33
IO	5	100	5.00
BUFG	1	32	3.12



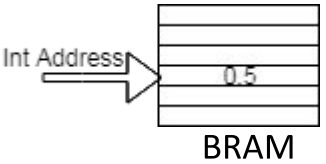
Zybo

Arithmetic Functions

- Why other methods do not work
  - Pipeline
    - Total saved at 150 MHz in 320x240 (x5.4 smaller) -> 475.2 ns which is 0.0007 % total execution time
  - BRAM-based

Method (10^{-5})	One Function (MB)	Total (MB)
[0, 2π]	2.39	9.58
[0, π/4]	0.29	1.16

  - How to do the address?
    - Have to convert degree (radian) into address



UNIVERSITY OF ARKANSAS

Arithmetic Functions

Function	# Clks	Freq (MHz)	Delay (ns)	DSP	# Slices	Power (W)	F/ DxSxM
Adder	**4**	**150**	**23.1**	**-**	**40**	**0.37**	**129.8**
Adder	5	150	29.1	-	52	0.37	72.1
Subtractor	**5**	**150**	**29.7**	**-**	**44**	**0.37**	**84.1**
Subtractor	6	150	36.3	-	44	0.37	68.3
Multiplier	**5**	**150**	**29.7**	**✓**	**56**	**0.37**	**72.1**
Multiplier	6	150	36.3	✓	78	0.37	41.3
Divider	**30**	**150**	**194.7**	**-**	**39**	**0.37**	**15.4**
Divider	33	150	214.5	-	47	0.37	11.6

UNIVERSITY OF ARKANSAS

Arithmetic Functions

Function	# Clks	Freq (MHz)	Delay (ns)	DSP	# Slices	Power (W)	F/ DxSxM
sin	**92**	**150**	**603.9**	✓	**351**	**0.39**	**0.53**
sin	106	150	696.3	✓	366	0.39	0.44
cos	**86**	**150**	**564.3**	✓	**328**	**0.39**	**0.61**
cos	98	150	643.5	✓	357	0.39	0.49
asin	**97**	**150**	**639.6**	✓	**365**	**0.39**	**0.48**
asin	113	150	742.5	✓	353	0.39	0.42
acos	**105**	**150**	**689.7**	✓	**339**	**0.39**	**0.48**
acos	113	150	742.5	✓	382	0.39	0.40

Arithmetic Functions

- Improved Functions on DICe

320x240 at 150 MHz	
Method	Exe Time (ms)
Previous	62.21
Improved	48.19
BRAM-based	> 46.19

UNIVERSITY OF ARKANSAS

Next Steps

- Going back to the DICe Correlation IPs
  - Validating the results
    - Verify that each IP operates correctly
    - Compare image correlation between DICe GUI and FPGA
  - Scale up
    - Support for more than two frames
    - Larger image sizes (< 896x464)
  - Adding more DICe features
    - Multiple subsets and unique subset shapes
    - Image obstructions
    - Etc.
- I/O Data Stream & Memory Management
  - BRAM: Load full/partial reference and deformed images for processing
  - DRAM: If BRAM is full, load the follow up deformed images that are to be loaded into BRAM
  - SD Card: If DRAM is full, load deformed images onto a large SD card that are to be loaded into DRAM

Conclusion

Questions?

Conclusion

Demo

Results

- Correlation time is 6x faster
- Total execution time is 25x slower!
  - With 1Gbps, it would be 5x slower
  - With using PetaLinux, it would be ….TBD

Testcase	FPGA Correlation time (s)	GUI Correlation time (s)	Transfer time (s)	Preprocessing time (s)	Total Execution time (s)	Total GUI time (s)
64x48	0.0025	0.016	1.25	0.21	1.48	0.058