

Preliminary Study of Image Reconstruction Algorithm on a Digital Signal Processor

by Daniel Rankin, Jeanine Cook, Song Park, and Dale Shires

ARL-MR-865

March 2014

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5067

ARL-MR-865

March 2014

Preliminary Study of Image Reconstruction Algorithm on a Digital Signal Processor

Daniel Rankin and Jeanine Cook

**Klipsch School of Electrical and Computer Engineering,
New Mexico State University**

Song Park and Dale Shires

Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188		
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) March 2014		2. REPORT TYPE Final		3. DATES COVERED (From - To) 1 April 2012–1 April 2013	
4. TITLE AND SUBTITLE Preliminary Study of Image Reconstruction Algorithm on a Digital Signal Processor			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Daniel Rankin,* Jeanine Cook,* Song Park, and Dale Shires			5d. PROJECT NUMBER R.0006155.21		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-CIH-S Aberdeen Proving Ground, MD 21005-5067			8. PERFORMING ORGANIZATION REPORT NUMBER ARL-MR-865		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES *Klipsch School of Electrical and Computer Engineering, P.O. Box 30001, MSC 3-0, New Mexico State University Las Cruces, NM, 88003-8001					
14. ABSTRACT With the advent of the parallel computing era, alternative processing solutions are examined for performance. Accelerators promise the potential to augment a system with only Intel or AMD central processing units. Specifically, digital signal processor (DSP) architecture is evaluated for computing image reconstruction algorithm. A Freescale DSP, drawing power in the range of 10 W, is considered for analysis. Software tools and programming techniques used during development are documented for the DSP platform. Preliminary results of the execution times are compared with graphics processing units and field programmable gate array technologies.					
15. SUBJECT TERMS Digital signal processor, signal image processing, coprocessor, back-projection, DSP optimization					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 22	19a. NAME OF RESPONSIBLE PERSON Daniel Rankin
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) (410) 278-5444

Contents

List of Figures	iv
List of Tables	iv
1. Introduction	1
2. DSP Hardware Platform	2
3. Image Reconstruction Algorithm	5
3.1 Input Data Format and Storage	6
4. Performance Tools	6
4.1 Code Optimization	6
4.2 DSP Parallelization.....	7
4.3 Timing Measurement	7
4.4 Power Measurement	8
5. Results	8
5.1 DSP Code Optimization	8
5.2 Comparison of CPU-GPU, CPU-FPGA, and CPU-DSP Designs	9
5.2.1 Back-Projection Execution Time	9
5.2.2 Output Image Quality	10
5.2.3 Power Consumption	11
6. Conclusions and Future Work	13
7. References	14
Distribution List	15

List of Figures

Figure 1. MSC8156 block diagram (9).....	2
Figure 2. SC3850 block diagram (9).....	3
Figure 3. MSC8156 memory hierarchy (10).....	4
Figure 4. Radar processing flowchart.	5
Figure 5. Back-projected images: one aperture.	10
Figure 6. Back-projected images: 640 apertures.....	11
Figure 7. DSP current at idle state.	12
Figure 8. DSP current at runtime.	12

List of Tables

Table 1. Optimization level comparison.....	9
Table 2. Back-projection execution time: one aperture.	10
Table 3. Back-projection execution time: 640 apertures.	10
Table 4. 1 MSE comparisons: one aperture.	11

1. Introduction

The end of frequency scaling (1, 2) in Intel and AMD central processing units (CPUs) has hampered x86 dominance in performance, especially for inherently parallel applications. In this new era of parallel computing, the computational capabilities of non-x86 technologies such as graphics processing units (GPUs), field programmable gate arrays (FPGAs), and digital signal processors (DSPs) have the potential to dramatically augment the traditional CPU-centric systems via the exploitation of parallelism. Heterogeneous systems that utilize a mixture of processor types can enable an order of magnitude improvement in performance to achieve faster time to solution in a smaller footprint.

Currently, GPUs are as pervasive as CPUs, and graphics cards support general-purpose computing in addition to driving displays and rendering graphics on a screen (3). GPUs are traditionally designed to handle the inherently parallel operation of managing each pixel of graphics concurrently. The original purpose of GPU architecture was to optimize parallel processing. Accordingly, peak performance of high-end graphics cards are rated in excess of a trillion floating point operations per second (4). In terms of programming, the learning curve for getting started with GPUs is relatively low, assuming a programmer has a background in C development. However, despite clear benefits in terms of raw computing capabilities, high-end discrete GPUs consume more power where thermal design power ranges in the order of 200–300 W (5), exceeding CPUs. This power requirement can be a severe drawback in battery-constrained systems.

FPGAs are inherently concurrent devices by definition. In FPGA development, a specific and custom hardware design is generated to solve a problem. FPGAs address computing an algorithm at the hardware level rather than at the software level. Depending on an application's characteristics, FPGA implementations can provide a significant performance advantage because the circuit in an FPGA maps directly to an application's specific task. For digital image processing, a Nallatech FPGA solution can potentially facilitate real-time computation with power consumption on the order of 25 W (6). However, prevalent disadvantages of FPGAs compared with CPU or GPU programming methods are the longer development time and the steep learning curve of Verilog or very-high-speed integrated circuit (VHSIC) hardware description language (VHDL) languages. The programming process is completely different for FPGAs since the development essentially involves designing hardware that requires the knowledge of digital circuits.

This report presents work on evaluating DSPs for image formation execution. Consuming power in the 10-W range (7), feasibility of a Freescale DSP architecture for field-deployable application is investigated to achieve real or near real-time solutions. The benefits of utilizing DSPs include

the ease in programming and design optimization. Yet, it remains unclear if DSP architectures can offer performance advantages competitive to GPU and FPGA implementations.

This project focuses on a radar imaging algorithm previously implemented on a heterogeneous CPU-GPU platform (8). The CPU-FPGA platform implementation is currently in simulation stage and near acquiring empirical results. The contribution of this research include an implementation of image formation algorithm on a CPU-DSP hybrid system that compares power consumption and execution speed with that of the previous CPU-GPU and CPU-FPGA designs.

2. DSP Hardware Platform

The hardware platform configuration for development comprises of a Dell Optiplex 390 workstation in conjunction with a Freescale MSC8156EVM evaluation module that supports six Star Core SC3850 DSP cores with an onboard 1024-MB double data rate (DDR) synchronous dynamic random-access memory (referred to as DDR1). The DDR1 module is an MT8JSF12864H(I)Y-1G1 by Micron inserted in a SODIMM204 socket on the evaluation module. The module has 1 GB access via a 64-bit data bus from the MSC8156's DDR1 controller. A block diagram of the MSC8156EVM is depicted in figure 1. Although the Freescale evaluation module has high-speed peripheral component interconnect (PCI) express capability, the module is primarily configured through a standard universal serial bus (USB) 2.0 interface connecting the board to the Dell host system.

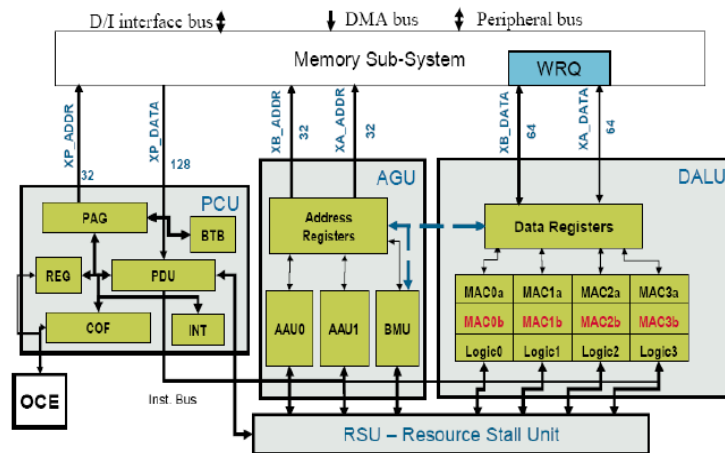


Figure 1. MSC8156 block diagram (9).

The Freescale SC3850 core uses instruction packing to fit the maximum amount of code into the minimum amount of space. The SC3850's Variable Length Execution Set (VLES) instruction architecture allows the program to pack multiple instructions of varying sizes into a single execution set. The reading of the instructions and identifying the start and end of each instruction

set is handled by the prefetch, fetch, and instruction dispatch hardware of the SC3850 (10). All instruction words are 16 bits wide and each SC3850 instruction encodes an atomic (lowest-level) operation. Atomic operations need fewer bits to execute, thus allowing for a fully orthogonal (all instruction types can use all addressing modes) 16-bit instruction set. This VLES architecture allows the grouping of up to four Data Arithmetic and Logic Unit (DALU) and two address generation unit (AGU) instructions in a single clock cycle (9).

Each Freescale SC3850 core contains a DALU with four ALUs, a program control unit (PCU), a resource stall unit (RSU), and an AGU that includes two address arithmetic units (AAUs). Figure 2 illustrates a block diagram of the SC3850 DSP core. The SC3850 uses dual-multiply ALUs supporting two 16×16 -bit multipliers that can accumulate results into 40-bit-wide destination data registers. Each ALU performs two multiply-accumulate operations per clock cycle, so that a single core running at up to 1 GHz can perform 8 billion multiply-accumulates per second. This rate is for 16-bit operands, 8-bit operands, or mixed 8- and 16-bit operands. Each AAU in the AGU can perform one address calculation and drive one data memory access per cycle. Data access widths are flexible and can be between 8 and 64 bits wide. The AGU can support a throughput of up to 128 gigabits per second between the core and the memory. The PCU performs instruction fetch, instruction dispatch, hardware loop control, and exception processing. It includes a 48-entry branch target buffer that is used to accelerate change of flow execution. The RSU controls the hardware interlocks. It collects information from the instruction bus, holds the status for all resources in the core, and inserts enough stalls to resolve the pipeline hazards.

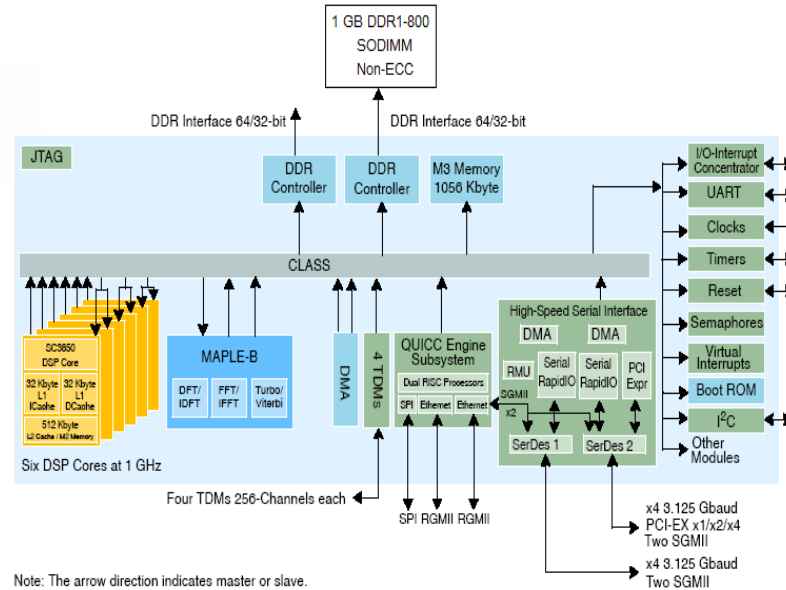


Figure 2. SC3850 block diagram (9).

The MSC8156 has a four-level memory subsystem. It consists of two external DDR memory controllers, two levels of internal static random access memory (SRAM) (1056-KB M3 and 512-KB M2), and two separate 32-KB L1 cache memories. The image formation algorithm is

characterized by a very large number of indirect (nonsequential) array accesses, favoring SRAM greatly over a traditional DRAM memory. An SRAM offers short access times and is optimal for real-time systems with timing constraints. However, due to the large amount of terrain signal data that must be processed (over 40 MB for certain data sets), only level 4 memory (DDR) on the MSC8156EVM can store the complete set of data at one time. Figure 3 conveys the memory hierarchy and latencies of each memory level for the MSC8156 device.

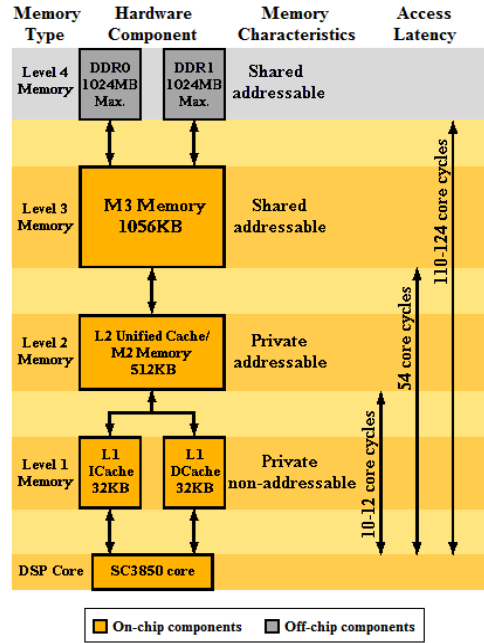


Figure 3. MSC8156 memory hierarchy (10).

The integrated development environment (IDE) CodeWarrior Development Studio Version 1.0.0 assisted the programming development. This development studio contains the C and C++ compilers necessary to access and configure each component of the MSC8156EVM. This software tool coordinated the reading of GPS coordinates and terrain signal echo files stored on the hard drive, writing to the DSP's on-chip DDR (level 4) memory, processing the data, and back-projecting onto an image matrix that is stored in a portable pixmap file (.ppm) on the CPU hard drive. Once generated, the image output is displayed using GIMP 2.8.2, a freely downloadable GNU image manipulation program.

3. Image Reconstruction Algorithm

The image algorithm processes GPS and terrain signal data for the purpose of reconstructing an accurate radar image. Figure 4 portrays the details of the image reconstruction algorithm. The DSP design follows the original algorithm presented by Park et al. (8) that describes GPU implementation.

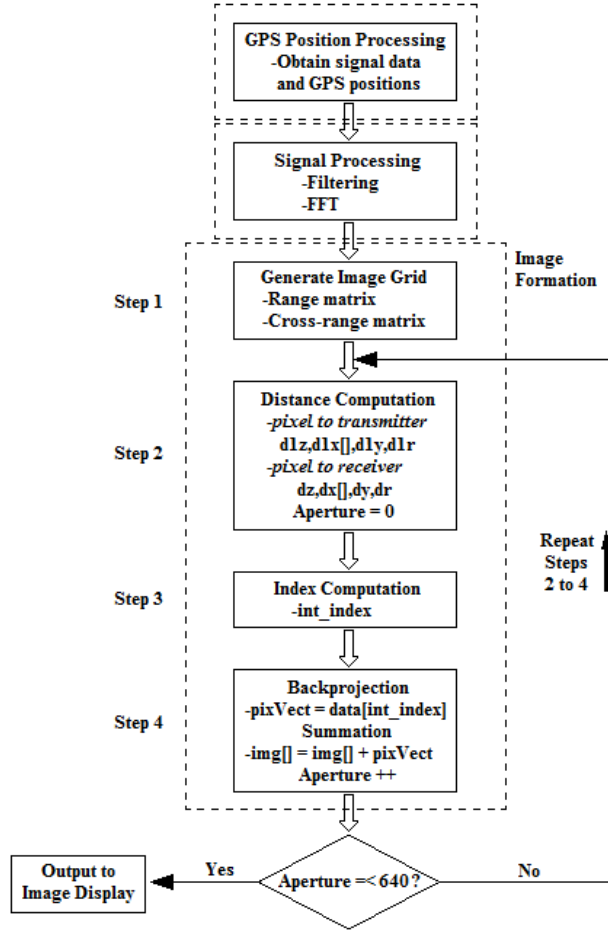


Figure 4. Radar processing flowchart.

The image formation begins by receiving the signal data returned from the transmitted radar pulse and tagging these data with a GPS position to subsequently determine the location in the target area of each returned signal. The energy of the return signal (reflectivity) indicates objects in the area to be imaged (target area). Because the radar platform is moving, a distance calculation is performed that determines the position of each pixel in the target area relative to the position of the transmitting and receiving antennas. The distance-time correlation takes into consideration the round-trip time of signal propagation using the computed distances between

transmitter/receiver and pixels. From this correlation, signal strength and location are determined. Finally, the target image area is mapped onto the smaller image display area (image matrix), the current return data are summed with the prior data (e.g., “summation” or coherent summation), and back projected onto the image display area. This implementation study considered an output image with a resolution of 100×250 pixels.

3.1 Input Data Format and Storage

The input data for testing the DSP implementation contained a real numerical format in seven files located on the CPU hard drive. The first six files contain 6–9 KB of positional coordinate data (x, y, and z) of the transmitting and receiving antennas obtained by GPS for each of 640 apertures. The title of each file designates the corresponding antenna and coordinates. The seventh file contains about 42 MB of return signal data for each of 640 apertures. This file is significantly larger because the reflected radar data is sampled at a particular frequency depending on the frequency of the radar pulse (11). In this application, 5397 samples are generated for each aperture and formatted into a one-dimensional (1-D) matrix that contains 3,454,080 data values. The total size of the input data files needed for this implementation (42.43 MB) exceeds the amount of SRAM available on the MSC8156EVM (1.6 MB), thus it is stored in the form of 1-D arrays in onboard DDR1 memory. Because the access latency to onboard DDR is high, the optimal DSP solution must include more available on-chip SRAM.

To utilize these seven files, a code written in C++ converts all input data from real format to IEEE 754 single-precision floating-point. This same code is also used to convert the floating-point data back into real format after the data have been processed but before the data are written to the CPU hard disk to be displayed. Focusing on the more computationally intensive case, this work investigates floating-point data only.

4. Performance Tools

The primary purpose behind designing and optimizing the back-projection algorithm for a CPU-DSP hybrid system is to compare its power consumption and execution speed with that of the previous CPU-GPU and CPU-FPGA designs. Therefore, we tried to keep code optimization and parallelization constant. The following section describes tools pertinent to timing and power measurement.

4.1 Code Optimization

The CodeWarrior IDE offers a variety of performance analysis and enhancement tools, one of which is the provision of four levels of code optimization. The empirical results are documented according to the optimization level implemented on the DSP. The FPGA implementation being

fairly well optimized, results are expected to be comparable to the higher levels of optimization (3 or 4). The following describes the differences between each level of optimization:

1. Optimization Level 0: This optimization level disables all optimizations. The compiler generates unoptimized, linear assembly language code.
2. Optimization Level 1: The compiler performs all target-independent (that is, nonparallelized) optimizations, such as function inlining. The compiler omits all target-specific optimizations and generates linear assembly language code.
3. Optimization Level 2: The compiler performs all optimizations (both target-independent and target-specific). The compiler outputs optimized, nonlinear, and parallelized assembly language code.
4. Optimization Level 3: The compiler performs all the level 2 optimizations and then the low-level optimizer performs global-algorithm register allocation.
5. Optimization Level 4: The compiler performs advanced scalar and loop optimizations.
6. Faster Execution Speed: The compiler optimizes object code at the specified optimization level such that the resulting binary file has a faster execution speed, as opposed to a smaller executable code size.

4.2 DSP Parallelization

To fully utilize the capabilities of the MSC8156EVM, a multicore application code was developed that distributes the data among all six cores for parallel processing. However, no documentation could be found explaining how to link multicore application code within the CodeWarrior development environment. Through submitted service request, a documentation regarding to this topic was received. The response included an attached file that explained how to link a multicore application for a device similar to the MSC8156EVM. After following the instructions, the code no longer compiled. The development of multicore application codes that CodeWarrior compilers are able to link and run simultaneously on separate cores is an area of future work that will greatly benefit this project.

4.3 Timing Measurement

The MSC8156EVM contains a powerful component known as the Enhanced On-Chip Emulation module. This module serves as a debugging and testing tool for monitoring StarCore SC3850 core activity. This module allows determining the number of DSP core cycles required to initialize all variables, read each input data file, perform all back-projection computations, construct the output image file, and store output data into a text file.

The emulation module monitors core cycles by synchronizing a 64-bit counter to the 1-GHz core frequency and both resetting and logging the counter value when issued the appropriate C++ command. A certain amount of overhead is associated with both issuing commands to the core

cycle counter and retrieving counter data. To calculate the amount of core counter overhead, we initialized the counter, immediately call for the value of the counter, and save the difference into the variable ‘overhead.’ This overhead is then subtracted from every core counter call made in the code.

4.4 Power Measurement

DSP power consumption was measured for comparison with the FPGA and CPU implementations. Tools used for the comparison consist of LabVIEW Signal Express 2011 edition software tool, NI USB-6216 (BNC) data acquisition system (12), and two FLUKE i30s current clamps (13). The current is measured from both the power supply input cord connecting the module to wall socket power and the USB communication channel between the DSP and CPU.

The first power measurement was recorded while the DSP was powered up and idling. After calibrating each current clamp, acquiring data for 30 s begins through LabVIEW Signal Express while there was no input to the current clamp. Then, the current clamps are placed on the DSP power and USB communication cords for another 30 s. Finally, the current clamps are removed and acquire 30 s more of data. Data were collected in this way to provide a clear visual representation of the DSP’s real-time power consumption characteristics during an idle state.

In addition, the power consumption of the DSP executing the back-projection code was collected. The current clamps were placed on the power and USB cords to measure power consumption in real time at an idle state and during the code execution.

5. Results

The following results are reported in this section. First, the DSP back-projection execution time for various levels of code optimization is presented. Followed by a comparison of output image quality, execution speed, and power consumption to GPU (8) and FPGA (14) implementations.

5.1 DSP Code Optimization

Table 1 refers to the number of core cycles required to execute the back-projection portion of the radar algorithm for one aperture on the DSP processor. Optimization level 3 outperformed the other levels and is thus used as the base case by which DSP performance is compared to that of an FPGA and GPU.

Table 1. Optimization level comparison.

Optimization Level	Cycles (1-GHz clock)	Time (ms)
0	82,967,850	83
1	60,681,152	60.7
2	55,019,518	55
3	52,292,781	52.3
4	52,444,969	52.4

5.2 Comparison of CPU-GPU, CPU-FPGA, and CPU-DSP Designs

The work for implementing VHDL description of the back-projection algorithm on a physical FPGA was not complete. Hence, the DSP implementation results are compared with the simulated results for the VHDL design. Simulating VHDL provides an accurate measure. In the Xilinx IDE, the user can choose a particular FPGA model on which to simulate a design. For this project, the Virtex5-LX155-FF1153 model was selected for the simulation device. When an FPGA is simulated, all of the timing constraints with respect to technology and gate delays are maintained such that the simulated execution time of the design models an actual gate.

To mitigate memory latency, the project selected an FPGA board that integrates the Virtex5 device with an SRAM memory. Unfortunately, the integrated SRAM memory is outside the scope of the device simulator, requiring an alternate method of obtaining accurate estimated results. To simulate the faster speed of SRAM (compared with DRAM), we implement the SRAM memory using standard logic blocks on the FPGA. On-chip logic is used to store the input data instead of block RAM to optimize for speed rather than area. However, this will still be a pessimistic time estimate since SRAM will be faster than using FPGA logic blocks and an SRAM controller will further optimize memory access for speed. In using FPGA's on-chip logic accordingly, limitation exists for the amount of memory that can be mapped to an area constrained configurable logic. Therefore, simulated results for small data sets and extrapolated the execution time for larger data sets based on this smaller data simulation.

5.2.1 Back-Projection Execution Time

These results represent the execution time for the back projection only, excluding read/write time to memory for the original signal input data. Results are reported first for a single aperture (5397 samples) because of a memory constraint associated with the FPGA simulation results. For the execution time for 640 apertures, the FPGA results are extrapolated while the CPU and DSP results are measured. GPU-CPU results were obtained for the NVIDIA Tesla C1060 GPU. FPGA-CPU results were obtained by simulating a Xilinx Virtex5-LX155-FF1153 on an HP XW-9400 workstation. The results for single aperture computation are shown in table 2. Next, table 3 summarizes the performance comparison between the measured GPU-CPU, simulated FPGA-CPU, and measured DSP-CPU implementations, with an estimated FPGA execution time for processing 640 apertures.

Table 2. Back-projection execution time: one aperture.

Hardware Technology	Execution Time (ms)
GPU-CPU	1.72
FPGA-CPU (simulated)	0.316
DSP-CPU	52.3

Table 3. Back-projection execution time: 640 apertures.

Hardware Technology	Execution Time (s)
GPU-CPU	1.1
FPGA-CPU (simulated)	0.202
DSP-CPU	33.5

The resulting data indicate that the simulated FPGA implementation of back-projection is about 5.4 times faster than on the original GPU-CPU system and about 165 times faster than on the DSP-CPU implementation. The simulated FPGA-CPU implementation is faster than the GPU-CPU system essentially because the FPGA implementation is a one-to-one mapping of the back-projection task to hardware, which provides a significant performance advantage over the GPU-CPU implementation. The DSP-CPU implementation is much slower due to its memory hierarchy. The number of clock cycles required to access data stored in DDR is much slower than that of SRAM, especially when SRAM is located on-chip and DDR is off-chip. Future work includes this DSP-CPU implementation executing the same back-projection algorithm using on-chip SRAM instead of off-chip DDR memory and a parallel DSP-CPU implementation.

5.2.2 Output Image Quality

The output images for the GPU-CPU, FPGA-CPU, and DSP-CPU back-projection implementations are shown in figure 5. These output images are for a single aperture data. There are a number of what seem like concentric arches with their apexes slightly left of center on the display. Each arch contains a slightly different shade of grey than its adjacent arches (looking like a black-and-white rainbow).

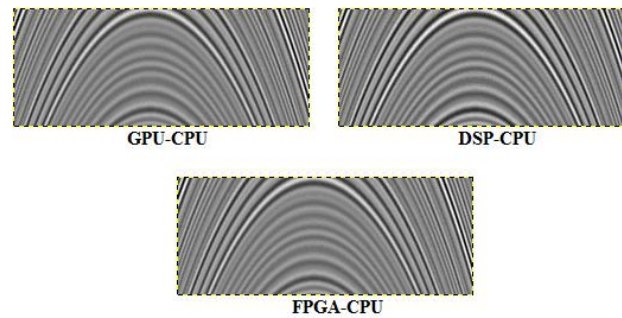


Figure 5. Back-projected images: one aperture.

The darker portions of the FPGA-CPU stand out slightly more than that of the other two images, which could prove helpful in generating a sharper output on the field. To perform a more accurate comparison of these three images, we developed a short Matlab script that performs a mean squared error (MSE) test. The pixel values from the image matrices used for these three implementations to compute MSE in table 4.

Table 4. 1 MSE comparisons: one aperture.

Hardware Technology	MSE (compared to original GPU-CPU image)
FPGA-CPU	2.17%
DSP-CPU	0%

The output images for the GPU-CPU and DSP-CPU back-projection implementations for all 640 apertures are shown in figure 6. FPGA-CPU hardware implementation is a work in progress. Notice the increase in the amount of detail compared with the single aperture images. The general arch shape still exists, but individual arches containing their own specific shade of grey have disappeared. The MSE between these two images was 0%.

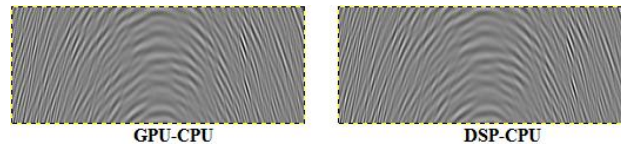


Figure 6. Back-projected images: 640 apertures.

A 2.17% error between the FPGA-CPU and original GPU-CPU back projected image for one aperture indicates that the FPGA-CPU implementation is indeed capable of accurately processing this back-projection signal data. A 0% error (identical) between the DSP-CPU and original GPU-CPU back-projected images for both 1 and 640 apertures indicates that it is also capable of accurately implementing this back-projection algorithm.

5.2.3 Power Consumption

At idle state, the current clamps measure approximately 0 mA of current when disconnected from any input signal as expected. When connected (at idle state) to the power and USB cords of the DSP, a 4- to 5-mA increase in current was experienced on both channels. Since the DSP was not currently executing code or communicating with the CPU, little to no change on the USB channel is to be expected. However, since it is receiving power and is advertised to operate on the order of 10 W, there should be a larger amount of current measured on the power channel. Figure 7 illustrates the Signal Express current measurements read from the DSP power cord. As can be seen from the x-axis, measurements were taken for 30 s before, during, and after the current clamps were connected to the DSP power cord (the break in signal indicates where the current clamp was connected and removed). The amount of current measured when the clamp was removed from the DSP power cord was slightly higher than before it was connected.

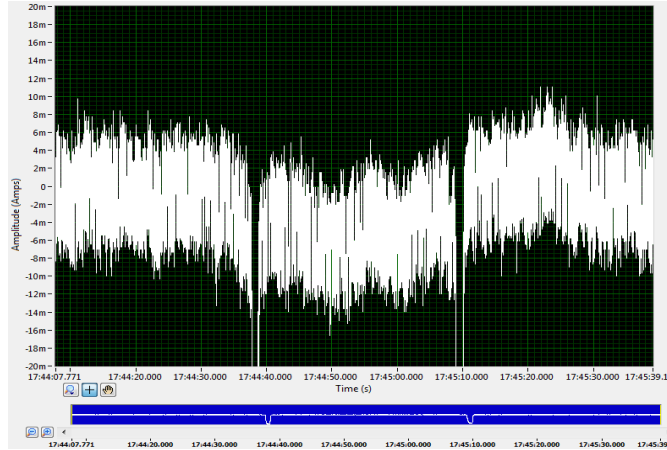


Figure 7. DSP current at idle state.

Visual analysis of the measurements collected during the program execution, depicted in figure 8, yields inconclusive results. The white plot represents the power cable to the DSP board and the red represents current over the USB communication line to the CPU. There is about an 8-mA increase as the clamp is connected to the power line, but as the DSP runs the code for over 8 min, there is little to no change observed on either line.

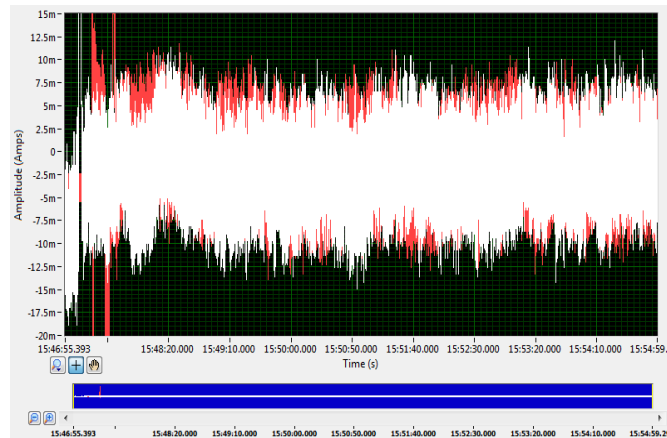


Figure 8. DSP current at runtime.

The MSC8156EVM onboard power system comprises of multiple regulator steps. The primary power system contains a 1.0-V point-of-load regulator for MSC8156 loads, cores, and M3. It also regulates 2.5 V for input/output and 3.3 V for onboard peripherals. Therefore, using this voltage information and the current clamp measurements taken at runtime, power consumption ($V \times I$) on the order of milliwatts for this implementation is entirely too low to be considered credible.

Fortunately, Freescale provides typical power consumption measurements associated with their DSP processors. They assert that with cores running at 1 V at 75% utilization, a single 64-bit DDR1 running at 800 MHz at 50% utilization and M3 memory 50% utilized, the MSC8156 will

consume approximately 6 W of power. Single-digit power consumption is lower than the CPU's 65.16 W and is comparable to the FPGA's simulated power consumption of 6.97 W.

6. Conclusions and Future Work

Presented here are the initial results for executing a radar-imaging algorithm on a CPU-DSP hybrid system. The preliminary results are compared to that of the GPU-CPU and FPGA-CPU hybrid systems. After implementing a portion of the back-projection algorithm on the DSP's SRAM on-chip memory, the execution speed failed to reach the simulated FPGA implementation. DSP power consumption does rival that of the FPGA in that both systems operate on the order of 10 W. Further work is required to optimize memory latency and to parallelize for insightful analysis.

Logical continuation work consists of updating the comparison analysis with the results from a PCI-connected Nallatech FPGA card. Moreover, performance efficiency achieved on the DSP calls for further examination. Investigation of multiple core operation, architecture specific parallel computing optimizations, and instruction vectorization requires closer analysis. Accelerators tend to be sensitive to optimization guidelines and techniques, resulting in a narrow window of architecture specific configuration for achieving maximum performance. Subpar performance of DSP observed for the image reconstruction algorithm can be related to an un-optimized configuration and is subject of future work.

7. References

1. Sutter, H. The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software. *Dr. Dobbs's Journal* 30.3 **2005**, 30, 202–210.
2. Olukotun, K.; Hammond, L. The Future of Microprocessors. *Queue* 3.7 **2005**, 3, 26–29.
3. Nickolls, J.; Dally, W.J. The GPU Computing Era. *Micro, IEEE* 30.2 **2010**, 30, 56–69.
4. AMD Radeon HD 6970 Graphics. <http://www.amd.com/US/PRODUCTS/DESKTOP/GRAPHICS/AMD-RADEON-HD-6000/HD-6970/Pages/amd-radeon-hd-6970-overview.aspx#2> (accessed February 2013).
5. Radeon HD 6000 Series. http://en.wikipedia.org/wiki/Radeon_HD_6000_Series (accessed February 2013).
6. Showerman, M.; Enos, J.; Pant, A.; Kindratenko, V.; Steffen, C.; Pennington, R.; Hwu, W. QP: A Heterogeneous Multi-Accelerator Cluster. *Proceedings of the 10th LCI International Conference on High-Performance Clustered Computing*, Boulder, CA, 10-12 March 2009.
7. Freescale Reboots Base Station DSPs, Leapfrogs TI. http://www.eetimes.com/document.asp?doc_id=1310843 (Accessed January 2013).
8. Park, S. J.; Ross, J.; Shires, D.; Richie, D.; Henz, B.; Nguyen, L. Hybrid Core Acceleration of UWB Sire Radar Signal Processing. *IEEE Trans. Parallel Distrib. Syst.* **January 2011**, 22, 46–57.
9. Freescale Semiconductor, MSC8156 Reference Manual – Six Core Digital Signal Processor. Freescale Semiconductor Inc., Revision 2, Tempe, AZ, 2009–2011.
10. Oshana, R. *DSP for Embedded and Real-time Systems*; Elsevier Inc.: Waltham, MA, 2012.
11. Cordes, B.; Leeser, M. Parallel Backprojection: A Case Study in High-Performance Reconfigurable Computing. *EURASIP J. Emb. Sys.* **2009**, 2009, 727965.
12. NI-labview and data acquisition systems. <http://www.ni.com/> (accessed January 2013).
13. Fluke-electrical measurement equipments. <http://www.fluke.com/> (accessed January 2013).
14. Banerjee, S.; Cook, J. Heterogeneous Architecture for Image Reconstruction in UWB SIRE Radar Systems. Klipsch School of Electrical and Computer Engineering, New Mexico State University; in revision.

NO. OF
COPIES ORGANIZATION

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
2 (PDF)	RDRL CIH S S PARK D SHIRES

INTENTIONALLY LEFT BLANK.