Real-Time Implementation of UWB-OFDM SAR Imaging System Using Shared Memory Based FPGA

Md Anowar Hossain, Ibrahim Elshafiey, and Majeed A. S. Alkanhal

Department of Electrical Engineering
King Saud University, Riyadh, Kingdom of Saudi Arabia
ahossain@ksu.edu.sa, ishafiey@ksu.edu.sa, majeed@ksu.edu.sa

***Abstract*—** This paper presents a novel technique for FPGA based implementation of high-resolution SAR system using UWB-OFDM architecture. Greater computation time and larger memory requirement are the main difficulties in processing large amount of raw data in real-time SAR imaging. Significant part of the computation time is related to range and azmuth compression of raw SAR data which in turns heavily depends on computing FFT, IFFT and complex multiplication. A shared memory based FPGA model is developed using Xilinx block-set that provides real-time SAR imaging with higher accuracy and less computation time. The design procedures are straightforward and can be applied to practical SAR system for real-time imaging. The model is designed as hardware co-simulation using Xilinx system generator and implemented on Xilinx Virtex-6 ML605 FPGA.

***Index Terms* —** Field Programmable Gate Araay (FPGA), Orthogonal Frequency Division Multiplexing (OFDM), Synthetic Aperture Radar (SAR), Ultra-Wideband (UWB).

I. INTRODUCTION

Synthetic Aperture Radar (SAR) is used to achieve high-resolution images of a terrain. SAR transmits signals at spaced intervals called Pulse Repetition Intervals (PRI). The reflections at each PRI are processed together to reconstruct a radar image [1]. In general, high-resolution SAR imaging is obtained using UWB waveforms as radar transmitted pulse. UWB pulses (500-MHz bandwidth and above) can enhance the range resolution considerably [2].

OFDM is a digital modulation scheme that shows a great potential to be used as radar waveforms. OFDM signal consists of several orthogonal sub-carriers, emitted over a single transmission path simultaneously. Each sub-carrier contains a small portion of the entire signal bandwidth [3]. Technology advances facilitates higher sampling speed, allowing accurate generation of UWB-OFDM waveforms. This results in a diverse signal that is capable of high resolution imaging.

Although OFDM has been studied and implemented in the digital communication field, it has not yet been widely considered by the radar scientific community rather than few efforts [4,5].

FPGA implementation of SAR imaging is presented in [6]. However, the processing steps were too complicated using both C++ and HDL platform. Real-time SAR imaging by FPGA is introduced in [7] and the processing result of a frame of airborne SAR data is demonstrated. The computation time of 12.8 ns for 1024-point FFT has been achieved using Altera Stratix II FPGA. FPGA implementation of synthetic aperture radar application is shown in [8] and reasonable speedup has been achieved when compared to a software solution. SAR processing with General-Purpose Graphics Processing Units (GPGPUs) is described in [9]. GPU-based *omega*-k tomographic processing by 1D non-uniform FFTs is introduced in [10]. However, incorporating GPU hardware into systems adds expense in terms of power consumption, heat production, and cost.

Greater computation time is a characteristic of SAR imaging algorithms that poses a particularly serious problem for real-time

Submitted On: April 8, 2013
Accepted On: October 4, 2014

application. Advances in the operating speed of the FPGA allow signal processing applications to be solved in commercially available hardware. Because of their highly parallel nature, a 10 to 20 times increase in SAR processing speed is expected using modern FPGAs compared to multi-node RISC processors such as the PowerPC. Historically, FPGAs were much harder to program than a processor. Recently, system generator for DSP from Xilinx® has made FPGA technology accessible to DSP engineers. Xilinx System Generator pioneered the idea of compiling an FPGA program from a high-level Simulink model based on Xilinx block-sets [11].

II. UWB OFDM SIGNAL GENERATION

UWB-OFDM signal is generated by randomly populating the digital frequency domain vector as:

$$\Psi_{\omega} = \left[\Pi_{ns} \;\; \Pi_0 \;\; \Pi_{ps} \right], \qquad (1)$$

where Π_{ps} and Π_{ns} represent the positive and negative sub-carriers respectively, whereas Π_0 represents the baseband DC value. IFFT is then applied to Ψ_{ω} to obtain the discrete time-domain OFDM signal as:

$$\Psi_{tx}(t) = F^{-1}\left[\Psi_{\omega} \right]. \qquad (2)$$

For example, UWB-OFDM waveform can be generated using the following parameters: number of OFDM sub-carriers=256, sampling time, Δt_s=1 ns results in baseband bandwidth, $B_0=1/2\Delta t_s$=500 MHz, dividing by a factor of two to satisfy Nyquist criterion. When Π_{ps} is randomly populated and modulation scheme is chosen as BPSK, the waveform is noise-like as shown in Fig. 1.

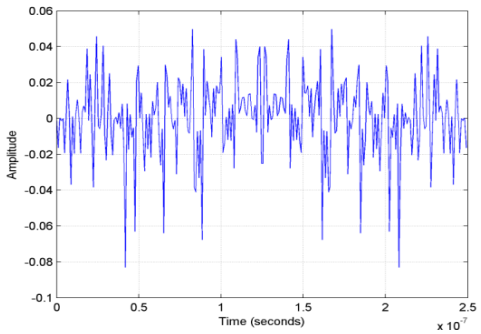

Fig. 1. UWB-OFDM signal using random sub-carriers.

III. RAW DATA GENERATION

Raw SAR data are generated based on simulation parameter shown in Table 1, using a gray-scale image of a 'battle tank' as a target profile. The input image is converted to a matrix. The position of each element is considered as the range and cross-range of the point target while the normalized value of each element of the matrix is used as reflectivity of the target.

At each synthetic aperture position, UWB-OFDM waveform is transmitted and the time-delayed signals reflected from the target are stored to form the raw SAR image space as shown in Fig. 2. The received radar signal is given as:

$$\Psi_{rx}(t,u) = \sum_{n=1}^{N} \sigma_n \Psi\left(t - t_{dn}\right) + \eta(t), \qquad (3)$$

where N, σ_n and t_{dn} are the number of targets, reflectivity and round-trip time delay associated with n^{th} target respectively. The term $\eta(t)$ denotes the AWGN. The round-trip delay t_{dn} is given by:

$$t_{dn} = \frac{2}{c}\sqrt{\left(X_c + x_n\right)^2 + \left(y_n - u\right)^2}, \qquad (4)$$

where X_c is the range distance to the center of the swath and (x_n, y_n) represents the location of the n^{th} target. The term u and c represents the synthetic aperture positions in azimuth direction and speed of light respectively.

Table 1: Simulation parameters

Parameter	Symbol	Value
Pulse Repetition Frequency	PRF	1024 Hz
Duration of flight	Dur	4 seconds
Velocity of the platform	V_P	200 m/s
Baseband bandwidth	B_0	500 MHz
Carrier frequency	f_c	7.5 GHz
Number of OFDM sub-carrier	N_{sub}	256
Range distance to center of swath	X_c	20 Km
Half target area width	X_0	200 m

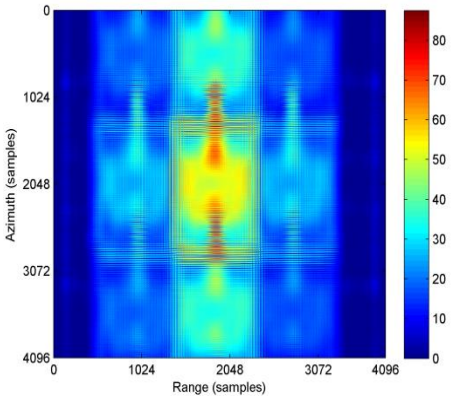

Fig. 2. Raw SAR image (battle tank).

IV. RANGE AND AZIMUTH COMPRESSION

Different imaging algorithms such as Range-Doppler algorithm and Omega-k algorithm can be used to process raw SAR data to reconstruct the final SAR image [12]. Almost all imaging algorithms perform matched filtering separately in range and cross-range domains. Range compression is performed by match filtering between each row of raw data and range reference signal as:

$$\Psi_{Mx}(t,u) = F^{-1}\left[\Psi_{rx}(\omega,u)\Psi^{*}_{ref}(\omega,u)\right]. \quad (5)$$

The range reference signal $\Psi_{ref}(\omega,u)$ is an ideal echo signal from a unit reflector at the center of the current range swath and is given in time-domain by:

$$\Psi_{ref}(t,u) = \Psi\left(t - t_{d0}\right), \text{ where } t_{d0} = \frac{2X_c}{c}. \quad (6)$$

Azimuth compression is obtained by match filtering in frequency domain between each column of the range compressed data and azimuth reference signal. IFFT is then applied to reconstruct the final SAR image. Azimuth reference signal is an ideal return from a unit reflector located at the center of the swath, i.e., $(x_n, y_n)=(X_c, 0)$ and is given as:

$$\Psi_{azref}(\omega,u) = e^{-j\frac{2\omega}{c}\sqrt{X_c^2+u^2}}. \quad (7)$$

V. FPGA IMPLEMENTATION

Shared memory based FPGA model is designed to reduce the computation time in range and azimuth compression and to achieve real-time SAR imaging. The prominent blocks used in the developed FPGA model from Xilinx block-set are introduced in the following subsections.

A. System generator

System Generator block provides control of system and simulation parameters, and is used to invoke the code generator. This block is also referred as System Generator "token" because of its unique role in the design. Every Simulink model containing any element from the Xilinx Block-set must contain at least one System Generator block. Once a System Generator block is added to a model, it is possible to specify how code generation and simulation should be handled. System Generator automatically compiles designs into low-level representations. Designs are compiled and simulated using the System Generator block. When the type is a variety of hardware co-simulation, then System Generator produces an FPGA configuration bit-stream that is ready to run in an actual hardware based on FPGA platform. System Generator also produces a hardware co-simulation block to which the bit-stream is associated. This block is able to participate in Simulink simulations. In a simulation, the block delivers the same results as those produced by the portion, but the results are computed in actual hardware, which speed up the computation time dramatically.

B. Shared memory

Xilinx shared memory block implements a Random Access Memory (RAM) that can be shared among multiple designs or sections of a design. A shared memory block is uniquely identified by its name. Instances of shared memories of same name within the same model automatically share the same memory space. These interfaces make it possible for hardware-based shared memory resources to map transparently to common address spaces on a host PC. System generator's hardware co-simulation interfaces allow shared memory to be compiled and co-simulated in FPGA hardware. Shared memories facilitate high-speed data transfers between the host computer and FPGA, and bolster the tool's real-time hardware co-simulation capabilities. The access to the shared memory can either be Lockable or Unprotected. An unprotected memory has no restrictions concerning when a read or write can occur. In a

locked shared memory, the block can only be written to when granted access to the memory. When the grant port outputs a 1, access is granted to the memory and the write or read request can proceed. The depth specifies the number of words in the memory. The word size is inferred from the bit width of the data input port.

C. Shared memory write/shared memory read

The Xilinx shared memory write block facilitates a high-speed interface for writing data into a Xilinx shared memory object. The shared memory write block is driven by the vector or matrix signal containing the data to be written into the shared memory object.

VI. FPGA MODEL FOR RANGE/AZIMUTH COMPRESSION

The schematic block diagram of the proposed FPGA model shown in Fig. 3 includes: input buffer subsystem, FFT and IFFT block, complex multiplier and output buffer subsystem. The developed FPGA model based on Xilinx block-set is presented in Fig. 4. The design consists of two subsystems that implements input and output buffer storage, named input and output buffer subsystem as shown in Figs. 5 and 6 respectively.

Each buffering subsystem consists of two shared memory blocks to provide the buffer storage for real and imaginary part of the raw data and output data respectively. Each shared memory is wrapped by logic circuitry (Counter, SR Flip-flops, delay and relational blocks) that controls the flow of data from the host computer, through the FPGA interface, and back to the host computer. This circuitry enables high-speed transfers of the memory data when the FPGA acquires or releases lock of the shared memory. It takes advantage of the lockable shared memory mutual exclusion to implement a high speed I/O buffering interface for hardware co-simulation. The Fast Fourier Transform block computes the FFT of the SAR raw data and the FFT of reference data is computed in MATLAB® for resource optimization as it is a row (column) matrix with fewer samples. The output of FFT block and the output of the shared memories of reference data are fed to the complex multiplier. Inverse Fast Fourier Transform block then computes IFFT of the data found after complex multiplication. Finally, the real and imaginary part of the output data is separately stored in two shared memories named shared memory (Real_out) and shared memory (Imag_out) inside the output buffer subsystem.

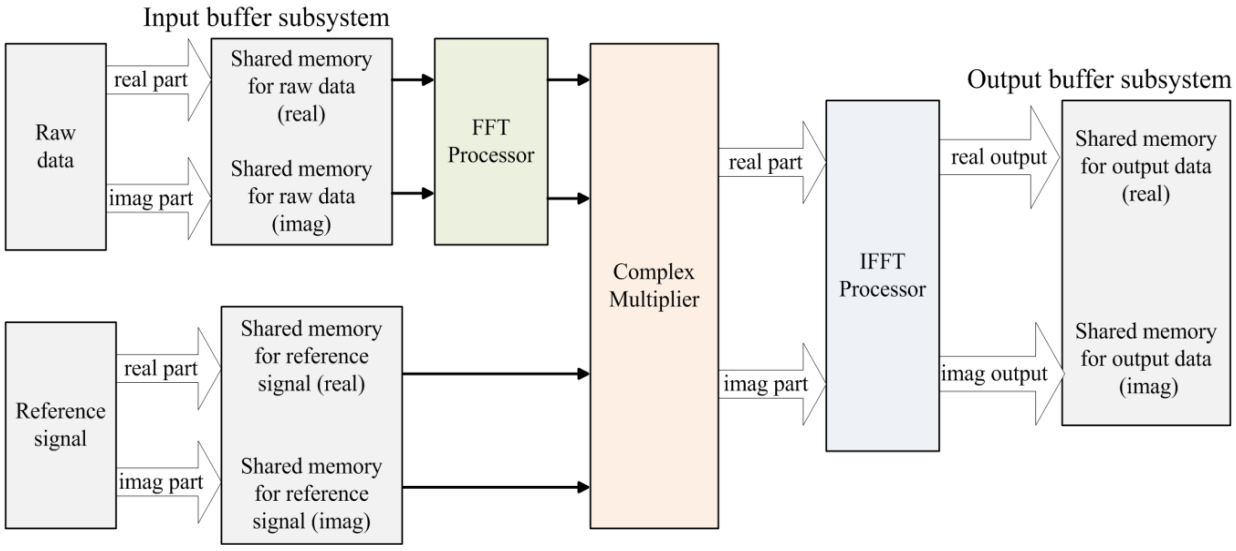

Fig. 3. Schematic block diagram of the proposed FPGA model.

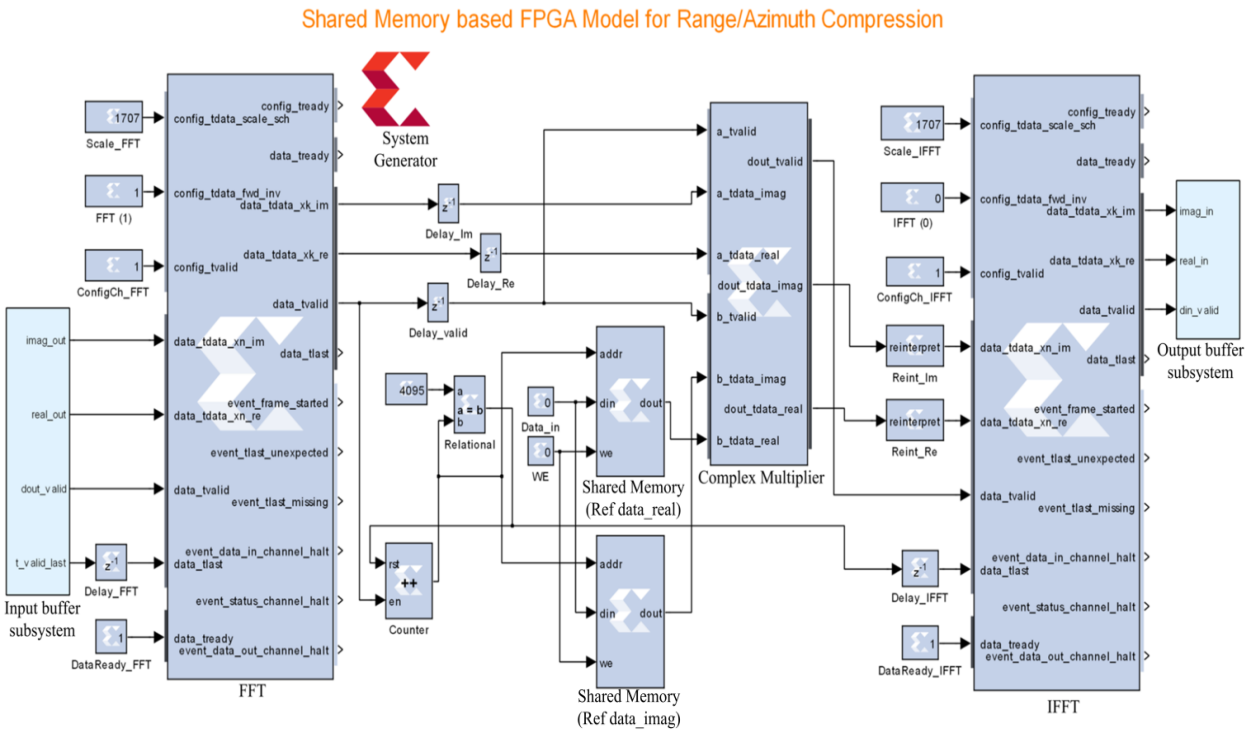

Fig. 4. FPGA model for real-time SAR imaging.

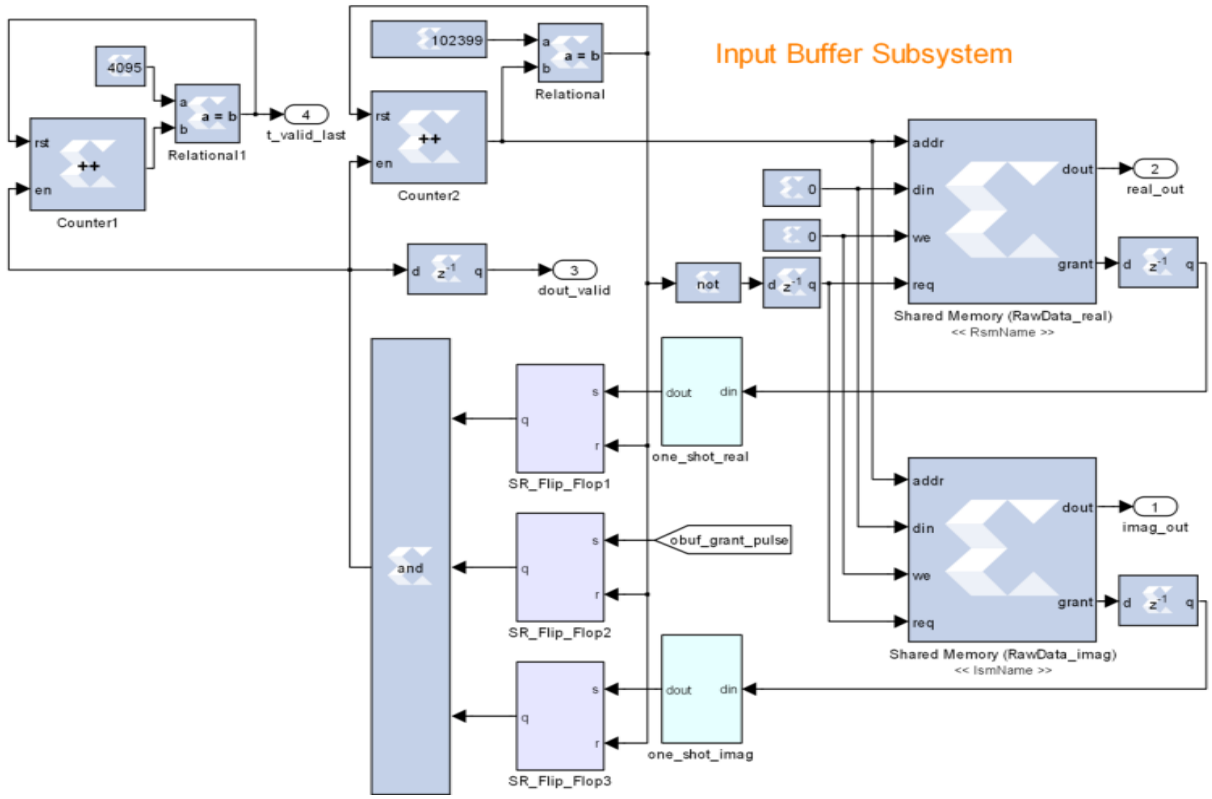

Fig. 5. Input buffer subsystem.

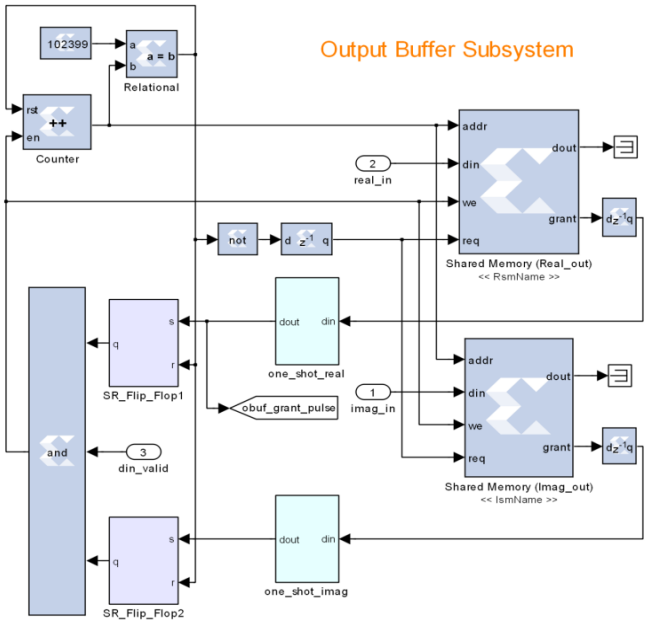

Fig. 6. Output buffer subsystem.

VII. TESTBENCH MODEL

The FPGA model shown in Fig. 4 is then compiled using system generator to generate hardware co-simulation block. A testbench model shown in Fig. 7 is designed which includes the hardware co-simulation block. The test-bench model includes a 'From Workspace' block from Simulink to read the SAR raw data from a *.MAT file. The real and imaginary part of the data is separated and written to two lockable shared memories named shared memory write (Real_data) and shared memory write (Imag_data). Similar name is used for the shared memories inside the input buffer subsystem so that the data are shared to the FPGA hardware.

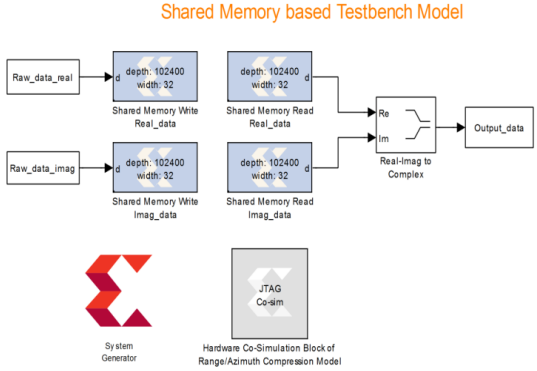

Fig. 7. Testbench Model.

Finally, the real and imaginary part of the FPGA processed output data are stored into shared memory write blocks named shared memory write (Real_out) and shared memory write (Imag_out). To control the data processing flow shown in Fig. 8, the blocks provided in the testbench model is pre-configured with appropriate priorities for proper wake-up sequence such as: shared memory write (1), hardware co-simulation block (2) and shared memory read (3). The operation of the model is described as follows: (a) Shared memory write blocks wake up and request a lock of the input buffer lockable shared memories. Once lock is granted, the blocks write the data into lockable shared memories and release the lock. (b) The hardware co-simulation block then wakes up and the host computer shared memory data are transferred to the FPGA. The FPGA processes the input buffer data and writes the output into the output buffer shared memories. Finally, the FPGA releases lock, causing the FPGA shared memory data to be transferred back to the host computer. (c) Shared memory read blocks wake up and request a lock of the output buffer lockable shared memories. The blocks read data from output buffer and drive its output port with the processed output data. The experimental test is then performed using the raw data generated by

using gray-scale image of a 'battle tank' as a target profile. Range compression is performed by feeding raw data to testbench model and running the model on Xilinx Virtex-6 ML605 FPGA [11]. Figure 9 shows the SAR image after range compression. Finally, azimuth compression is accomplished by using the same model to reconstruct the final SAR image shown in Fig. 10.

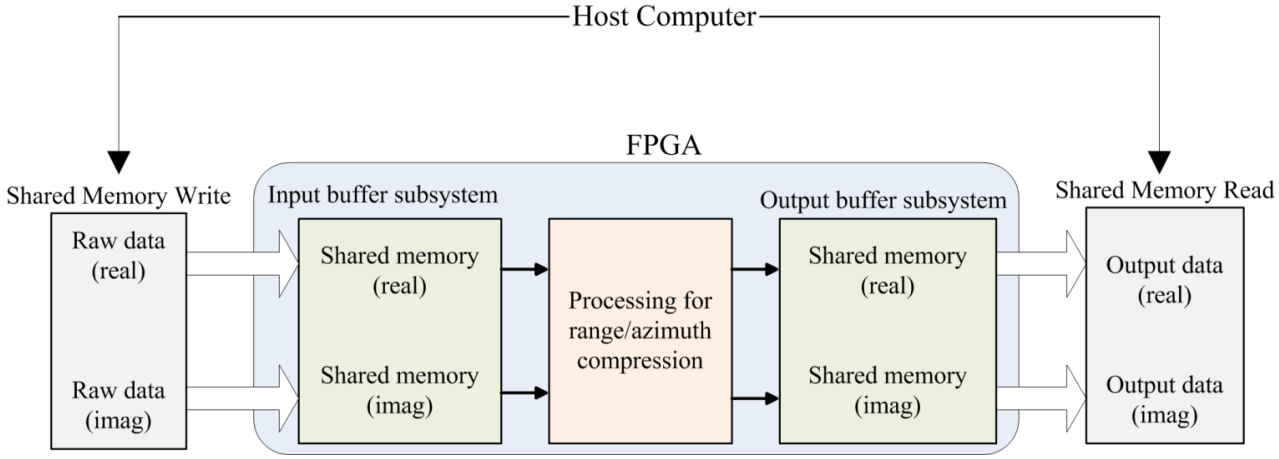

Fig. 8. Data Processing flow.

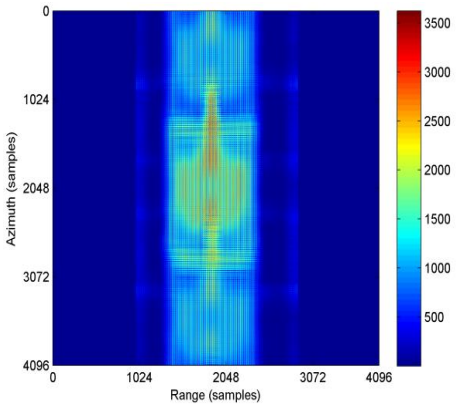

Fig. 9. SAR image after range compression.

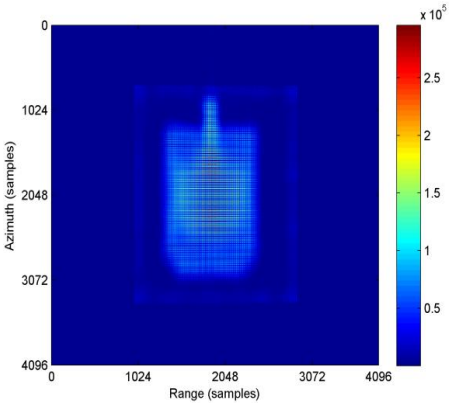

Fig. 10. Final SAR image after azimuth compression.

VIII. RESULTS AND DISCUSSIONS

The error level of FPGA output in comparison with MATLAB® results is shown in Fig. 11. As the difference between MATLAB results and FPGA output are extremely low, these results are computed by subtracting the FPGA output from MATLAB® output. It is observed that very low error fluctuated between 0 and .0001, i.e., at 4th decimal place which is negligible. Significant parts of the errors arise from the truncation at the complex multiplier output. The reason behind the FPGA output with lower error is that the raw data is converted into integer format before feeding to FPGA.

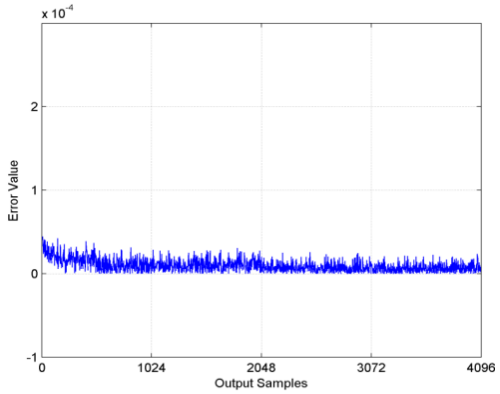

Fig. 11. Error level comparison of FPGA output and MATLAB® simulated results.

Data can be written to or read from unprotected shared memory blocks at any time. This type of memory has no mutual exclusion. Data transfers to and from an unprotected shared memory occurs a single-word at a time, unlike the high-speed data transfer mode used by lockable shared memories. The computation time and maximum frequency for both lockable and unprotected shared memory are compared and tabulated in Table 2. It is observed that the lockable shared memory requires less computation time as compared to unprotected shared memory.

Table 2: Computation time and frequency

Shared Memory Type	FPGA Clock	Maximum Frequency
Lockable	2.97 ns	335.68 MHz
Unprotected	3.99 ns	250.43 MHz

The significant reduction in computation time can be achieved by taking the advantage of parallel processing of FPGA, i.e., by running several hardware co-simulation blocks simultaneously. However, the major drawback is the memory I/O time. Table 3 shows the speedups with FPGA against a typical CPU using MATLAB for different image sizes. It is observed that FPGA processing achieves better speeds against CPU processing as the amount of data (image size) increases.

Table 3: Comparison of processing time for different image size

Image Size	CPU Processing (MATLAB)	FPGA Processing
4096×4096	4.24 sec	2.98 ns
8192×8192	8.96 sec	5.18 ns
16384×16384	18.27 sec	9.68 ns

Table 4 summarizes the resource estimation. It is observed that the lockable shared memory requires more resources than unprotected shared memory because it uses some logic circuitry to handle mutual exclusions. On the other hand, lockable shared memory requires less computation time as shown in Table 2. Thus, lockable shared memory is the best choice for the proposed FPGA model.

Table 4: Resource estimation

Component	Shared Memory Type	
	Lockable	Unprotected
Flipflop	6%	5%
LUTs	8.5%	7%
BRAM	3.6%	3.6%
IOB	73%	72.5%
DSP48s	12.5%	10%

IX. CONCLUSION

Real-time imaging of UWB-OFDM SAR system has been investigated using the high speed Xilinx Virtex-6 ML605 FPGA. FPGA output of the FFT/IFFT and complex multiplications as part of match filtering are compared with simulated results obtained under standard CPU (MATLAB®) processing. Experimental results show that FPGA implementation using shared memory approach is effective to implement SAR imaging algorithm and provides a real-time tool for SAR imaging.

ACKNOWLEDGEMENTS

This work is funded in part by the National Plan for Science and Technology, Saudi Arabia, under project number: 08-ELE262-2.The research is also supported by the Deanship of the Scientific Research and the Research Center at the College of Engineering, King Saud University, Saudi Arabia.

REFERENCES

[1] M. Soumekh, "Synthetic aperture radar signal processing," *A Wiley-Interscience Publication*, United States of America, 1999.

[2] J. D. Taylor, "Ultra-wideband radar technology," *CRC Press*, 2000.

[3] L. Hanzo, M. Münster, B. Choi, and T. Keller, "OFDM and MC-CDMA," *Wiley*, March 2004.

[4] M. A. Hossain, I. Elshafiey, and M. A. Alkanhal, "High resolution UWB SAR based on OFDM architecture," *2011 3rd International Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)*, pp. 1-4, 2011.

[5] M. A. Hossain, I. Elshafiey, M. A. Alkanhal, and A. Mabrouk, "Anti-jamming capabilities of UWB-OFDM SAR," *2011 European Radar*

Conference (EuRAD), pp. 313-316, 2011.

[6] C. Le, S. Chan, F. Cheng, W. Fang, M. Fischman, S. Hensley, R. Johnson, M. Jourdan, M. Marina, and B. Parham, "Onboard FPGA-based SAR processing for future spaceborne systems," 2004.

[7] X. Xiao, R. Zhang, X. Yang, and G. Zhang, "Realization of SAR real-time processor by FPGA," *In IGARSS 2004 IEEE International Geoscience and Remote Sensing Symposium*, pp. 3942-3944, 2004.

[8] J. Greco, G. Cieslewski, A. Jacobs, I. A. Troxel, and A. D. George, "Hardware/software interface for high-performance space computing with FPGA coprocessors," *In Aerospace Conference, 2006 IEEE*, p. 10, 2006.

[9] M. Di Bisceglie, M. Di Santo, C. Galdi, R. Lanari, and N. Ranaldo, "Synthetic aperture radar processing with GPGPU," *Signal Processing Magazine, IEEE*, vol. 27, pp. 69-78, 2010.

[10] A. Capozzoli, C. Curcio, and A. Liseno, "GPU-based ω-k tomographic processing by 1D non-uniform FFTs," *Progress In Electromagnetics Research M*, vol. 23, pp. 279-298, 2012.

[11] *Xilinx Inc.* (January, Xilinx Website. [Online], "Xilinx ISE design suite 13.4," http://www.xilinx.com/.

[12] I. G. Cumming and F. H. C. Wong, "Digital processing of synthetic aperture radar data: algorithms and implementation," *Artech House*, 2005.

Md Anowar Hossain received his B.S. degree in Computer and Communication Engineer-ing from International Islamic University Chittagong (IIUC), Bangladesh in 2005 and his M.S. degree in Electrical Engineering from King Saud University, Saudi Arabia in 2012. He has been working as a Researcher in the Department of Electrical Engineering, King Saud University, from October 2008 till now. Currently, he is pursuing his Ph.D. degree in Electrical Engineering at King Saud University. His research interests include UWB radar, Synthetic Aperture Radar (SAR) imaging, radar jamming and anti-jamming, radar signal processing and FPGA based real-time signal processing.

Ibrahim Elshafiey received his B.S. degree in Communications and Electronics Engineering from Cairo University in 1985. He obtained his M.S. and Ph.D. degrees from Iowa State University in 1992 and 1994, respectively. He is currently a Professor in the Electrical Engineering Department, King Saud University, on loan from the Electrical Engineering Department, Fayoum University, Egypt. His research interests include computational electromagnetics, biomedical imaging and nondestructive evaluation.

Majeed A. S. Alkanhal received his B.S. and M.S. degrees from King Saud University, Riyadh, Saudi Arabia, in 1984 and 1986 respectively, and the Ph.D. degree from Syracuse University, Syracuse, NY, in 1994, all in Electrical Engineering. He is presently a Professor in the Electrical Engineering Department at King Saud University, Riyadh, Saudi Arabia. His research interests are in antennas and propagation, computational electromagnetics, and microwave engineering.