UNSYS digital

Journal of Unmanned System Technology

AXI Bus Hardware Accelerator IP for SAR Image Processor Onboard Airborne

Good Fried Panggabean[†‡], Josaphat Tetuko Sri Sumantyo[†],
Chua Ming Yam[†], Hiroaki Kuze[†], Pakhrur Razi[†], Farohaji Kurniawan[†]

[†]Center for Environmental Remote Sensing, Chiba University, 1-33, Yayoi, Inage, Chiba 263-8522, Japan.

[‡]Faculty of Informatics and Electrical Engineering, Del Institute of Technology, Toba Samosir, Indonesia.

Abstract—This paper presents how a Synthetic Aperture Radar (SAR) image processor based on AXI Bus Hardware Accelerator (HA) IP was proposed and developed. The SAR image processor is targeted for use in an airborne Circularly Polarized SAR (CP-SAR) system. In the processor, The HA IP was built using software and hardware co-design approach to cater for the high computational requirement in the SAR image formation process. The IP core accelerators are based on Advanced eXtensible Interface (AXI) Bus for its high configurability. In addition, it is easy to get integrated with another IP core through an on-chip bus and low-cost in implementation. The IP core is co-assisted by MicroBlaze softcore processor in order to accelerate the Range Doppler Algorithm (RDA) processing, which is the most commonly used algorithm for SAR image formation. The proposed SAR image processor was implemented on a Xilinx Artix-7 FPGA AC701 Evaluation Kit using multi-core hardware accelerator and configured to work on raw data with segment size of 8192x8192 pixels. Several optimization techniques such as pipelining and parallelization were applied to the initial design to speed up the processing task and to reduce the resource utilization. The functionality of the implemented SAR image processor is validated by processing a raw dataset from recorded by ALOS PALSAR sensor. The results showed that the proposed SAR imaging processor could process SAR raw data in near real-time.

Keywords—SAR image processor, AXI bus hardware accelerator, Airborne SAR, RDA

DOI: 10.21535/just.v7i2.1028

I. Introduction

Synthetic Aperture Radar (SAR) is a broad remote sensing technology used for mapping the Earth's surface and monitoring a natural disaster area because it is suitable for use in day or night time and for all-weather condition [1]. SAR is an excellent tool that is able to create high-resolution images which can be used for various remote sensing applications such as research of geosciences and climate change, Earth mapping and environmental monitoring, land deformation and planetary exploration [2].

Josaphat Microwave Remote Sensing Laboratory (JMRL) is currently focusing on researching and developing a Circularly Polarized SAR (CP-SAR) sensor operating at L, C and X microwave frequency band [3,4]. The CP-SAR sensor transmits and receives left-handed circular polarization (LHCP) and right-handed circular polarization (RHCP) signal and then utilizes the circularly polarized scattering properties from land surface for SAR remote sensing applications. One of its key research roadmaps is to investigate the scattering behavior of circularly polarized waves on soil, sand, ice and vegetation and the generation of various images such as axial ratio (ARI), ellipticity, tilted image beside single look complex (SLC) and amplitude images [5].

As planned in the JMRSL research roadmap, the CP-SAR sensor will be deployed on several platforms, such as unmanned aerial vehicle (UAV), airborne and microsatellite. A UAV is also developed for ground calibration and CP-SAR sensors validation. For the airborne CP-SAR, the sensors (L, C, and X bands) will be deployed on Boeing 737-200 airborne for further CP characteristic research while the L band CP-SAR sensor will be launched on a microsatellite. Further, a C-band CP-SAR array antenna for airborne SAR will also be developed using Dolph-Chebyshev method for suppressing side-lobe and beam steering [6].

The airborne CP-SAR system will be installed in a Boeing 737-200 which has maximum cruising speed of 906.53 km/h and maximum cruising altitude of 10,668 m. The aircraft has maximum size of 0.4×5.1 m for antenna space located on the left and the right sides of its tail. The onboard CP-SAR sensor operates at the center frequency of 5.3 GHz, with off nadir angle of 20°-60°, altitude of 2130 m and cruising speed of 120-450 km/h. The specifications of the onboard CP-SAR sensor are as summarized in Table 1.

The general block diagram of CP-SAR system deployed on the aircraft is shown in Figure 1. The antennas marked on the right side of the diagram transmits either RHCP or LHCP polarization while the receiving antenna receives both the RHCP and LHCP scattering. The RF system consists of a transmitter, a receiver, a local oscillator and a low noise amplifier. The chirp pulse generator module generates a 150 MHz bandwidth chirp pulse modulated by the transmitter into the center frequency of 5.3 GHz. At the receiver module, the received waves will be demodulated into in-phase (I) and

Corresponding author: Good Fried Panggabean
(e-mail: goodfp25@gmail.com)

This paper was submitted on September 18, 2018;
and accepted on October 25, 2018.

quadrature (Q) component and then digitized for further processing. The SAR image processor will process the recorded echo signal data in the form of two-dimensional complex matrix data using a range-Doppler algorithm into an SLC image.

Table 1 Specifications of Boeing 737-200 for CP-SAR Mission

Parameters	Values
Altitude Maximum	10,668 m
Speed Maximum	906.53 km/h
Off nadir angel	30°
Antenna	
Polarization	Circular
Type	Microstrip patch array
Maximum size of antenna	
Elevation	0.4 m
Azimuth	5.1 m
Beamwidth	
Elevation	< 5°
Azimuth	< 2°
Antenna efficiency	80 %
Peak transmit power	< 400 watts
RF Specification	
Center Frequency	5.3 GHz
Baseband Bandwidth	300-400 MHz
Pulse Length	11-17 µs
Pulse Repetition Frequency (PRF)	1000-4000 Hz
Signal to Noise Ratio (SNR)	20 dB
Duty cycle	1.1-1.8 %
Swath width	< 600 m
Best azimuth resolution	0.67 m
Best range ground resolution	0.67 m

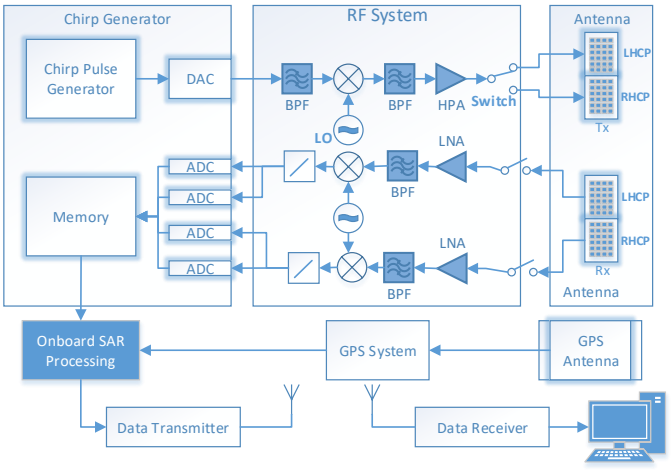


Figure 1 General block diagram of SAR system

The SAR image processor plays an important role for SAR system intended for monitoring natural disaster areas. One of the important requirements imposed on disaster monitoring system is the requirement to distribute observation data rapidly from disaster area to the disaster management center. The data will be used for damage estimation and for supporting the saved and rescued operations. Thus, onboard SAR image processor is one of the solutions to shorten the SAR image availability time whereas the image of the observation area can be directly transferred to the ground station as short time as possible.

Onboard SAR image processor is a critical part in a SAR system to provide the real or near real time SAR image processing. This can be implemented on a variety of computing platform. For example, the high-performance computing platforms with high parallelism are recently implemented using Graphic Processor Units (GPUs) [7,8], collaborative computing to accelerate the processing using multiple Central Processing Units (CPUs) and GPUs is explored extensively at [9,10,11,12] and near real time SAR processing using general purpose CPU [13] and high performance computing cluster in grid environments [14]. Recently, near real time onboard SAR image processing can be implemented using Digital Signal Processing (DSP) or a Field Programmable Gate Array (FPGA) platform [15,16] as well.

This paper describes the design and development of onboard SAR image processor using AXI bus hardware accelerator IP implemented on FPGA board. The main contribution of this paper is to present hardware and software solutions for SAR image processor onboard based on the AXI Bus Hardware Accelerator IP that provides reconfigurable, lightweight, and low power consumption. This paper also presented the proposed architecture and parallel approach to enhance the speed of data processing. The paper is organized as follows: Section II describes Hardware Accelerator (HA) design flow, including the proposed system architecture, the design description of HA IP cores for Range/Azimuth compression, corner turn and finally some optimization approaches. The implementation results and validation experiments are presented in section III. Finally, the conclusion is drawn in section IV.

II. Hardware Accelerator Design

This section presents a hardware accelerator desing, system architecture, range and azimuth reference, HP IP core for range compression, corner turn, azimuth compression and proposed optimization approaches.

A. System Architecture

There are popular algorithms used for processing of SAR raw data to a Single Look Complex (SLC) image namely, the Range-Doppler algorithm (RDA), Chirp Scaling Algorithm (CSA), OmegaK Algorithm (ωKA) and Spectral Analysis SAR Algorithm (SPECAN) [1,17]. The RDA algorithm is the algorithm that is most commonly used for SAR image formation. The algorithm was developed to process SEASAT SAR data in 1976-1978 which is still widely used for processing spaceborne and airborne SAR data today. The algorithm operates in frequency domain for both range and azimuth domain. All mathematical operations are performed with one-dimensional data array which results in providing simple and efficient processing. In RDA, Range Cell Migration Correction (RCMC) is performed in range Doppler domain that makes the correction of target's trajectory in this domain

correct. However, it still requires high computation power for interpolation operations [1].

The block diagram of the basic RDA algorithm is illustrated in Figure 2(a). In RDA, for the case airborne SAR, at altitude of 4000 m and off nadir angle of 30°, the slant range difference between closest range target ($R_1 - R_0 = 0.056$ m) is less than $1/4$ of the range resolution ($c/2B)/4 = 0.999$ m. Therefore, RCMC process is not needed, where $R_1 = R_0/(\cos 0.5\theta_{az})$, R_0 is the closest range target to the radar, θ_{az} is 3-dB azimuth beamwidth. Thus, the simplified RDA algorithm for the proposed SAR image processor is shown in Figure 2(b) where the RDA is implemented without RCMC.

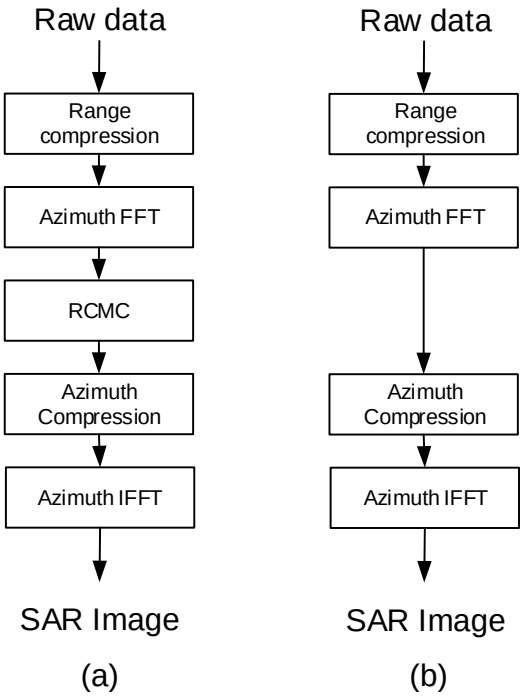


Figure 2 Block diagram of the basic RDA algorithm

It is proposed that the proposed SAR image processor is implemented using the softcore processor and programmable logic in the FPGA hardware. Some parts of the RDA algorithm that requires flexibility and reconfigurability are implemented in the softcore processor while the parts that have high computation complexity are implemented as Hardware Accelerator (HA) using FPGA resources. It was decided that the part of the RDA algorithm to be implemented in softcore processor is the generation of reference function (range and azimuth). The functions that require high computational power such as FFT/IFFT transform, multiplication of a range/azimuth line with the reference function are to be implemented as an HA.

In general, the hardware architecture of SAR image processor consists of one MicroBlaze softcore processor and some HA connected through the AMBA AXI bus system. The RDA algorithm that is described using Vivado High-Level Synthesize (HLS) is mapped into the hardware structure by performing hardware and software partitioning. As shown in Figure 3, the range FFT functions, multiplies and ranges IFFT from the algorithm which are grouped into an RGC HA module. The corner turn function is implement into a COR HA module. Finally, the azimuth FFT/IFFT and multiplication with azimuth reference are grouped into AZC HA module. The reference function generator is implemented in the MicroBlaze processor. The communication among components is handled by the bus interface that is available to every HA.

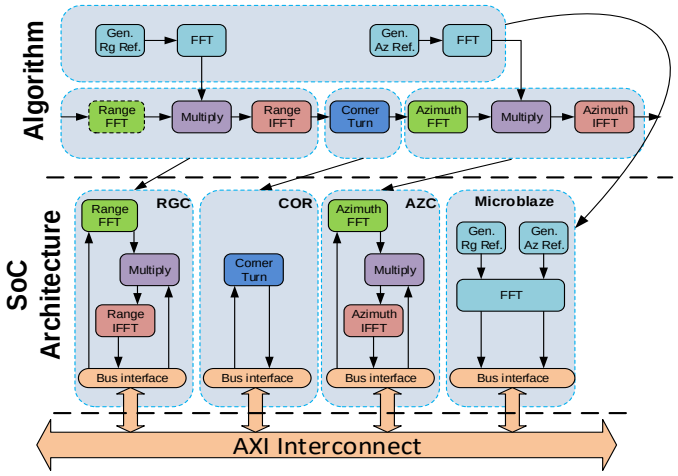


Figure 3 Software and hardware partitioning of the RDA algorithm

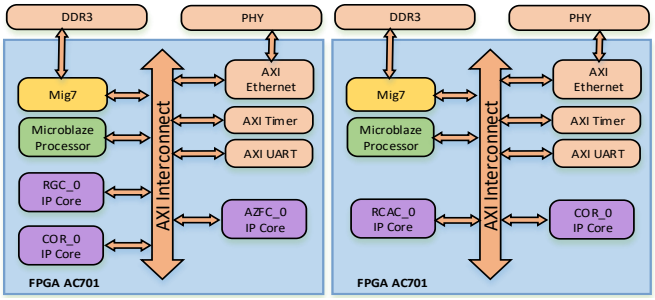


Figure 4 The proposed hardware architectures of SAR image processor for implementation: (a) RDA-A (initial design); (b) RDA-B (optimized design).

Figure 4(a) illustrates the initial architectural design of the proposed SAR image processor (RDA-A). The proposed architecture is composed of a MicroBlaze softcore processor and hardware accelerators. The system consists of range compression module (RGC), corner turn module (COR) and azimuth compression module (AZFC). If we look into more details about what the range and azimuth compression algorithm do, we will find significant similarities in the work of performing multiplying FFT and the IFFT tasks. These functions are distinguishable by the reference function used in the matched filtering process. Accordingly, these two functions can be combined into a more general HA by providing different input parameters into the HA. Thus, an optimized hardware architecture, namely, RDA-B is proposed. In RDA-B, RGC module and AZCF module is grouped into an RCAC HA module with the addition of input parameters. The proposed RDA-B model reduces the estimation use of BRAM 18K, DSP48E, FF and LUT resource respectively 34.94%, 6.21%, 11.19% and 20.61% with similar performance, shown in Table 2. In addition, instead of being used to generate range azimuth

reference functions, MicroBlaze softcore processors also control the data flow and control the operating modes of RCAC HA IP core.

Table 2 Resource utilization comparison (Vivado HLS Result)

Name	Available	RDA-A		RDA-B		
		Used	Utilized (%)	Used	Utilized (%)	(%)
BRAM_18K	730	612	83.84	357	48.90	34.94
DSP48E	740	102	13.78	56	7.57	6.21
FF	269200	61610	22.89	31495	11.70	11.19
LUT	129000	57328	44.44	30739	23.83	20.61

The interface of the IP cores is designed using an AXI master interface with pointer arguments that implement burst mode data transfers. The burst mode operation provides a high-throughput data transfer. The operation in burst mode in Vivado HLS is implemented by using the C `memcpy` function. The SAR raw data is transferred by the client application on PC via Ethernet to the proposed SAR image processor and stored the data to on board RAM. The RCAC HA IP core which was first configured as a range compression function reads each line of raw data from RAM and performs a range compression and writes the result back to the RAM. The RCAC IP core reads and writes the input output data at the same address memory. After performing range compression to every range line, the COR HA IP core reads a block of data and performs corner turn. The result of COR HA IP core will be written into different memory location. Finally, the RCAC IP core is configured as azimuth compression, reads the data line by line in the azimuth direction and performs azimuth compression. The final result of data is written back to the same memory location. The entire HA operations is controlled by the MicroBlaze processor.

B. *Range and Azimuth Reference*

This section will describe the detail of the range and azimuth reference function generated by MicroBlaze processor. Range compression is a correlation processing within the range of frequency based on range reference functions. The range reference function can be defined as

$$Rg(m) = \exp\left(-2\pi j\left(\frac{K_r}{2}\left(\frac{m}{f_s}\right)^2\right)\right) \quad (1)$$

where $m = \left\{-\frac{N_s}{2}..\frac{N_s}{2}\right\}$, N_s is a number of samples, $N_s = \tau_p f_s$, f_s is a sampling frequency, τ_p is pulse width.

Azimuth reference function can be defined as

$$Az(m) = exp\left(2\pi j\left(f_{dc}\frac{m}{PRF}\right) + \frac{K_a}{2}\left(\frac{m}{PRF}\right)^2\right) \quad (2)$$

where $m = \left\{-\frac{N_s}{2}..\frac{N_s}{2}\right\}$, $N_s = BW_d PRF/K_a$, f_{dc} is the Doppler centroid frequency, BW_d is Doppler bandwidth, PRF is effective pulse repetition frequency, K_a is the rate of change of azimuth or Doppler frequency.

C. *HA IP core for Range and Azimuth Compression*

This section describes the details of range and azimuth compression grouped into RCAC HA. Range compression performs the matched filtering on every row of range samples. The matched filter is generated by MicroBlaze processor and serves as a reference function operated for each row of data during the range compression process. Before matched filtering, a row of data is transformed into the frequency domain using FFT. Then, the data is multiplied by its conjugate reference function, and finally applies IFFT to obtain time domain range compressed data. The FFT algorithm uses the Xilinx FFT IP core implemented with the Cooley-Tukey FFT algorithm [19].

D. *HA IP core for Corner Turn*

This section describes the details of corner turn design and its implementation. Corner turn is the transposition of the 2D arrays of data performed between range compression and azimuth compression. It is the operation of copying a block of data array at range direction in continuous address into a temporary memory space, transposing the data array and rewriting the data array into azimuth direction of the memory in the continuous block address. This operation is needed for the execution of FFT and IFFT which has high computation load that can perform efficiently. The classic problem in the corner turn is the problem of how to access the memory when reading and writing the data array. If the corner turn is performed based on the reader side where the array is read line by line in the range direction, the data will be read from the continuous memory address. Yet, the writing will access the discontinuous memory and vice versa (Figure 5(a)).

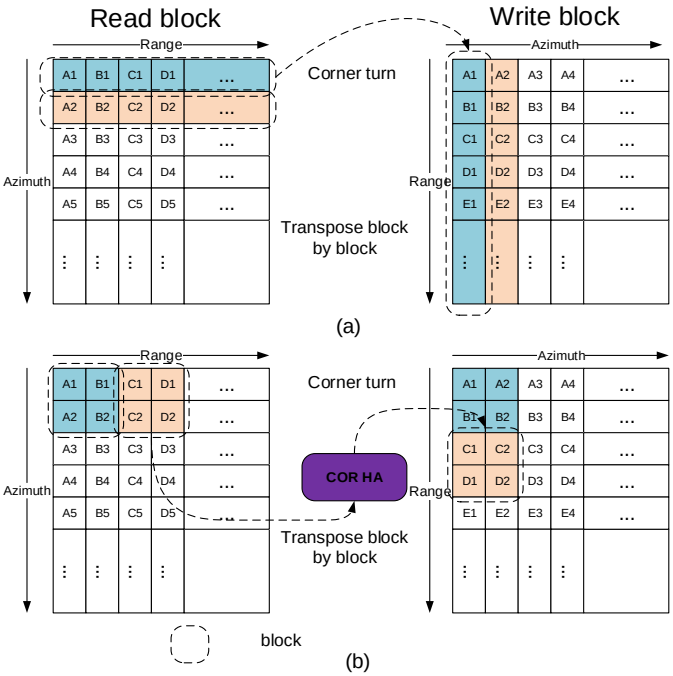


Figure 5 Corner turn: (a) Corner turn basic problem to read/write data; (b) Corner turn using square block technique

The corner turn HA IP core is built by adopting block corner turn technique [20,21]. The 2D array of data is divided into several blocks and the corner turn is performed to each block (Figure 5(b)). The size of the block array data in the source memory is chosen and the array of data is fetched into a Block RAM (BRAM). Transpose is performed at BRAM (a faster

memory than DDR) and it writes the result into the DDR. The data segment size is 8192×8192 elements of I and Q data, which is 4 Bytes each. This gives a total size of 512 Mbytes for corner turn. The implementation of the correct architectural option selection of HA using BRAM rather than DRAM as a block array cache offers excellent performance improvements [22]. Table 3 shows resource utilization estimation, latency, and speedup for corner turn module implemented with different block size. There are two methods of performance optimization used, the first one is loop-pipelining used to allow operations to happen concurrently. The second optimization using DATAFLOW directive to ensure the pipelined functions will be executed in parallel. The following table shows that implementation of optimization can improve the speedup of 3.01. However, increasing the block size for corner turn does not significantly affect the performance.

Table 3 Resource Utilization Estimation, Latency, and Speedup for Various Block Array Size

Size of block Array(NxN)	BRAM_18K	FF	LUT	Latency (s)	Speedup improvement
64x64	8	1424	1799	17.530	1.000
64x64,pl	8	1457	1713	5.825	3.010
64x64,pl, df	8	1451	1762	5.133	3.415
128x128, pl, df	32	1481	1814	4.563	3.842
256x256, pl, df	128	1511	1876	4.304	4.073
512x512, pl, df	512	1541	1926	4.161	4.213

pl =pipeline, df= DATAFLOW

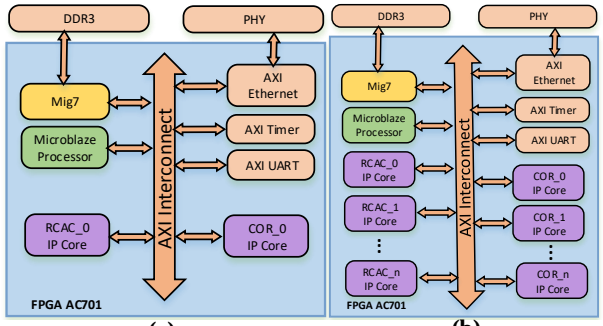


Figure 6 The proposed hardware architectures of parallelized SAR image processor: (a) RDA-B architecture as reference design; (b) Proposed parallelized architecture for N HA IP cores.

E. Parallelization Approach

As discussed in the previous section, RDA-B architecture has equal performance and uses less FPGA resource when compared to RDA-A. Thus, RDA-B architecture is used as the reference design for implementing parallelization approaches for computation speedup. Figure 6 shows the architectures of proposed parallelized SAR image processor using AXI Bus HA IP. The parallelization is implemented using the multi IP cores.

A data segment is divided into sub-segment data and processed by each parallel module. Taking an example of 2 parallel IP cores, a data segment with 8192×8192 elements is divided into 2 sub-segments of data, in which the first sub-segment is from the 1st until 4096th line and the second sub-segment is from 4097th line until 8192nd line. As shown in Figure 7, the RCAC_0 module will perform matched filtering to each line in the first sub-segment, and at the same time, the RCAC_1 module will do the same processing to each line of data from the second sub-segment. In this way, the parallelization approach does not introduce gaps between the sub-segments. Therefore, data overlap between the two sub-segments is not necessary.

III. Results and Discussion

The SAR image processor was implemented on a Xilinx Artix-7 FPGA AC701 Evaluation Kit. It is found that SAR image processor with two RCAC HA cores and one COR HA is the best configuration for the implementation as the design can fit well into the FPGA in the evaluation kit. Due to the unavailability of airborne SAR data from CP-SAR sensor, the SAR image processor is tested using PALSAR data with very high-resolution observation mode (FBS) at Mt. Fuji took from [18]. The data set is chosen because the main functional algorithms such as range compression, azimuth compression, and corner turning function are relatively similar. In order to determine the accuracy of the processor, the same dataset was processed by a PC-based SAR processor while the proposed processor and the formed SAR images are compared. Figure 8 shows the SAR image focused by the PC-based processor (Figure 8a) and the SAR image focused by the proposed SAR image processor (Figure 8b). Comparing the two, we can clearly see that the image produced by the PC-based SAR processor is more focused as compared to the image produced by the proposed SAR processor. This is mainly because the SAR image produced by the PC-based SAR processor is inclusive of RCMC process and its azimuth reference functions were generated for every line in azimuth direction. Figure 9 shows the slices target distribution profile at both range and azimuth direction. The figure clearly shows that both the range

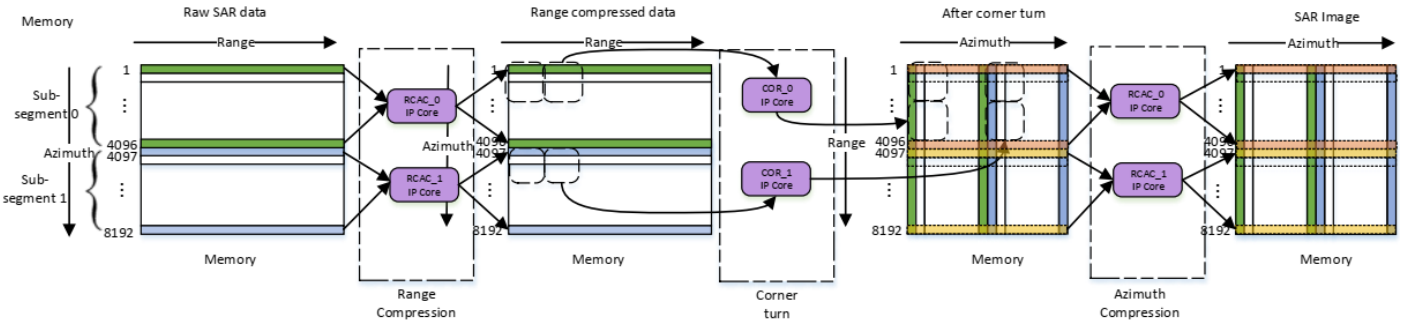
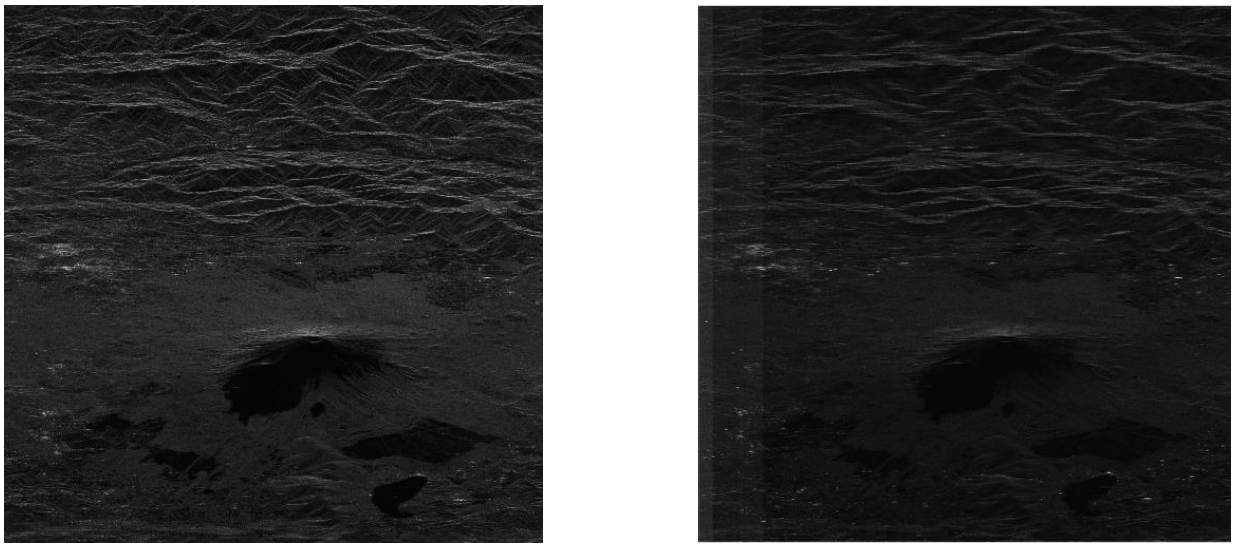
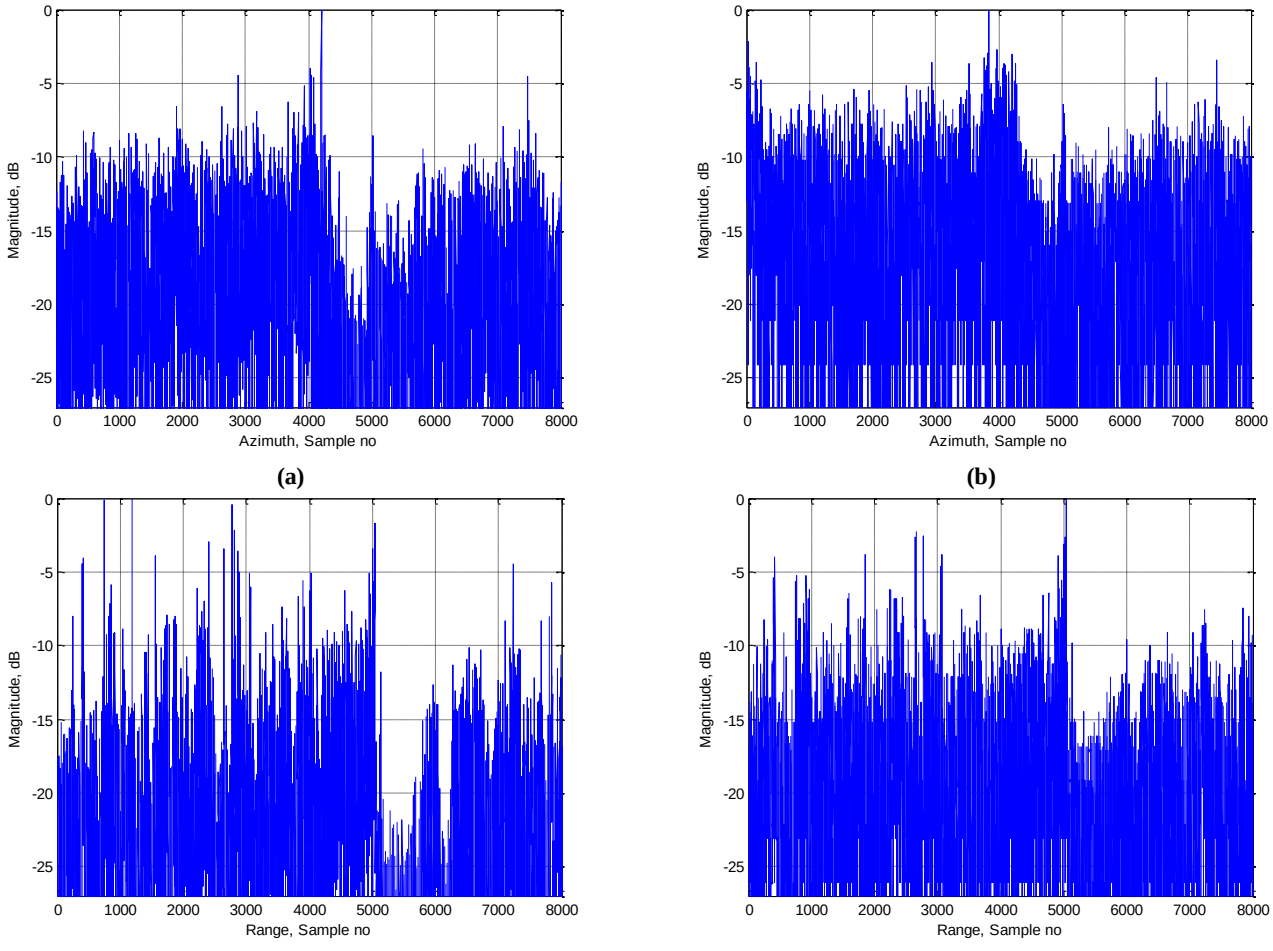


Figure 7 The flow of sub-segment data for proposed parallelized architecture



(a) (b)

Figure 8 SAR image from: (a) PC-based processor; (b) the proposed SAR image processor.



(a) (b)

(c) (d)

Figure 9 Range and azimuth target distribution profiles: (a) Cross-range profile at line 5100 from PC-based processor; (b) Cross-range profile at line 5100 from the proposed SAR image processor; (c) Range profile at line 4800 PC-based processor; (d) Range profile at line 4800 the proposed SAR image processor.

and the azimuth profiles produced by the PC-based SAR image processor and the proposed processor are almost similar. Based on this analysis, the accuracy of the proposed SAR image processor is acceptable especially for onboard quicklook processing of SAR raw data. For a more focused SAR image, the data will be processed off-line on the ground with a better image formation and more focused SAR image formation algorithm.

The SAR processor is designed to work with data with segment size of 8192×8192 pixels. The post-implementation of the 2 cores SAR image processor utilizes 88526 LUTs (66.16% of the device utilization), 76447 flip-flops (28.57%) and 361 blocks of BRAM (98.90%). The entire system consumes 4.78W of power as estimated by Xilinx Power Estimator. The most power consuming component is the mig 7series DDR3 controller (1.582W), followed by the RCAC_0 block (0.617W) and the RCAC_1 block (0.617W). The MicroBlaze processor and Ethernet controller only consume 0.192 W and 0.19 W, respectively, while the COR_0 block consumes only 0.14 W. 4.78W of power as estimated by Xilinx Power Estimator. The most power consuming component is the mig 7series DDR3 controller (1.582W), followed by the RCAC_0 block (0.617W) and the RCAC_1 block (0.617W). The MicroBlaze processor and Ethernet controller only consume 0.192 W and 0.19 W, respectively, while the COR_0 block consumes only 0.14 W.

Table 4 Comparison of latency between RDA-A and RDA-B (seconds)

Module	RDA-A	Module	RDA-B
RGC	7.428	RCAC(rc)	7.428
COR	4.400	COR	4.400
AZFC	7.428	RCAC(ac)	7.428
Total	19.256		19.256

Table 5 Percentage of resource utilization for different size of data segment

Segment Size	1024 × 1024	2048 × 2048	4096 × 4096	8192 × 8192	16384 × 16384
BRAM_18K	41.51	45.34	52.19	66.44	94.11
DSP48E	5.95	6.76	6.76	7.57	7.57
FF	9.06	9.92	10.79	11.7	12.68
LUT	17.69	19.47	21.45	23.83	27

Table 6 Latency comparison for various size of segment process (seconds)

Modul	1024 × 1024	2048 × 2048	4096 × 4096	8192 × 8192	16384 × 16384
RCAC(rc)	0.12	0.47	1.87	7.43	29.64
COR	0.07	0.27	1.10	4.38	17.31
RCAC(ac)	0.12	0.47	1.87	7.43	29.64
Total	0.31	1.21	4.83	19.25	76.59

The second test has been carried out to measure the latency of the SAR image processor. Table 4 compares the latency between RDA-A (initial design) and RDA-B (optimized design). The latencies produced by both architectures are the same. However, the resource utilization in each design is different from RDA-B which uses lesser resources (as shown in Table 2). The percentage of utilization estimation and its latency for various size of processing segment using single HA IP core are shown in Table 4 and Table 5. The number of required BRAM_18K memory block increases with proportion to the increase in data segment size while the number of required DSP48E block, FF, and LUT increased slightly.

Finally, parallelization performance is benchmarked by measuring the computation speed-up and the parallelization efficiency of the proposed SAR image processor. The speedup is defined as the ratio of the processor latency single core HA IP as compared to the processor latency on N cores HA IP. The speedup can be defined as

$$S_N = \frac{\tau_1}{\tau_N} \tag{3}$$

where τ_1 is the processor latency of single core HA IP and τ_N is the processor latency of N cores HA IP. The efficiency is used for measures the percentage of cores utilization, which is defined as

$$E_N = \frac{\tau_1}{N\tau_N} = \frac{S_N}{N} \tag{4}$$

Table 7 Multicore implementation latency (seconds)

Modules	Number of cores		
	1	2	4
RCAC(rc)	7.428	3.714	1.867
COR	4.400	3.089	2.549
RCAC(ac)	7.428	3.715	1.867
Total	19.256	10.517	6.282

Table 8 Multicore implementation speedup

Modules	Number of cores		
	1	2	4
RCAC(rc)	1.00	2.00	3.98
COR	1.00	1.42	1.73
RCAC(ac)	1.00	2.00	3.98
Total	1.00	1.83	3.07

Table 9 Multicore implementation efficiency

Modules	Number of cores		
	1	2	4
RCAC(rc)	1.00	1.00	0.99
COR	1.00	0.71	0.43
RCAC(ac)	1.00	1.00	0.99
Total	1.00	0.92	0.77

Table 7 and Figure 10 show the processor latency of 1,2 and 4 cores HA implementation. As shown in Table 8 and Table 9, the computation speedup for the RCAC module is almost doubled by doubling the number of cores and efficiency is almost 100%. The COR module only speedup 1.42 times for 2 cores and 1.73 times for 4 cores. This is due to the COR module which spends most of the time in accessing the shared memory pool.

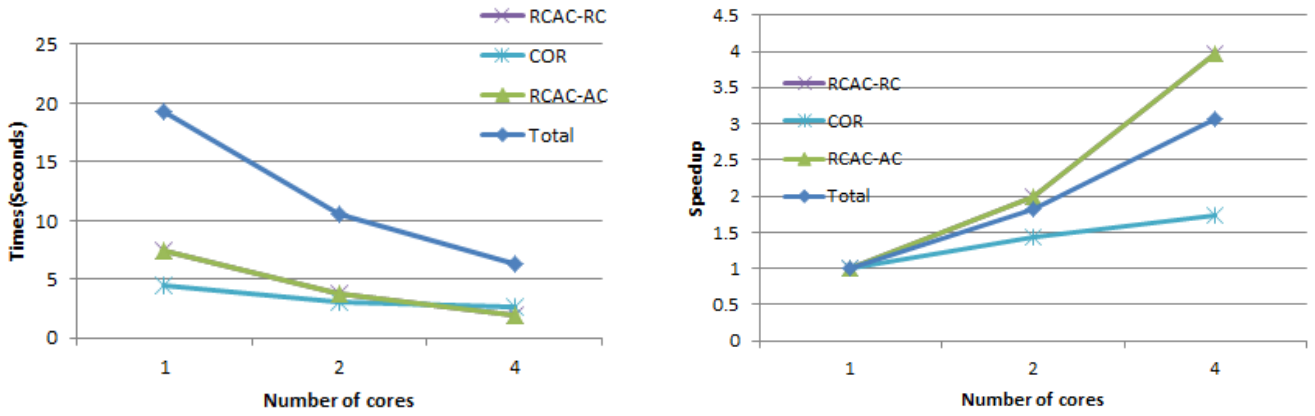


Figure 10 Multicore implementation:
(a) Process latency (seconds); (b) Speedup.

IV. CONCLUSIONS

On this paper, we have proposed an onboard SAR image processor architecture based on AXI Bus Hardware Accelerator IP for airborne SAR. The SAR image processor is implemented on a Xilinx Artix-7 FPGA AC701 Evaluation Kit with multicore hardware accelerator. The proposed processor uses simplified RDA algorithm and is capable of generating high-resolution SAR image. The hardware architecture is implemented using software-hardware embedded design paradigm that provides flexibility and easiness to be reused for other systems. The functionality of system is tested by processing the raw image data from ALOS PALSAR into an SAR image. The results of the implementation showed that the proposed HA IP cores are modularized, highly configurable and consume very low power. It is highly suitable and feasible for use in airborne and spaceborne CP-SAR onboard data processing.

ACKNOWLEDGEMENTS

This research was supported by Chiba University Strategic Priority Research Promotion Program FY2016-FY2018. We would like to address our gratitude to Del Foundation, Tanoto Foundation and Citramas Foundation for funding of the study.

REFERENCES

[1] Cumming, Ian G, “Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation”, Artech House, 2005.

[2] A. Moreira, P. Prats-iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, A tutorial on synthetic aperture radar, IEEE Geosci. Remote Sens. Mag., vol. 1, no. 1, pp. 643, 2013.

[3] Sumantyo, J.T.S., Progress On Development Of Synthetic Aperture Radar Onboard Uav And Microsatellite, Geoscience and Remote Sensing Symposium (IGARSS), IEEE International, 2014.

[4] Sri Sumantyo, J. T. Circularly Polarized Synthetic Aperture Radar Onboard Unmanned Aerial Vehicle (CP-SAR UAV). In Autonomous Control Systems and Vehicles; Nonami, K.; Kartidjo, M.; Yoon, K.-J.; Budiyono, A., Eds.; Springer Japan, 2013; pp. 175192.

[5] Josaphat Tetuko Sri Sumantyo; Nobuyoshi Imura, Development of circularly polarized synthetic aperture radar for airborne and microsatellite, IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2016.

[6] Mohd Zafri Baharuddin, Josaphat Tetuko Sri Sumantyo and Hiroaki Kuze, Suppressed Side-lobe, Beam Steered, C-band Circularly Polarized Array Antenna for Airborne Synthetic Aperture Radar, Journal of Unmanned System Technologies, 31 May 2016 In press.

[7] Maddikonda S.S, Shanmugha S. G. A., "SAR image processing using GPU", International Conference Communications and Signal Processing (ICCSP), pp. 448-452, 2014.

[8] T. Hartley, A. Fasih, C. Berdanier, F. Ozguner, and U. Catalyurek, Investigating the use of GPU-accelerated nodes for SAR image formation, in luster Computing and Workshops, 2009. CLUSTER 09. IEEE International Conference on, pp. 1 8, 31 2009-Sept. 4 2009.

[9] Di Bisceglie, M., M. Di Santo, C. Galdi, R. Lanari, and N. Ranaldo, "Synthetic aperture radar processing with GPGPU", IEEE Signal Proc. Mag., Vol. 27, No. 2, 6978, Sept. 2010.

[10] C. Clemente, M. di Bisceglie, M. Di Santo, N. Ranaldo, and M. Spinelli, Processing of synthetic aperture radar data with GPGPU, in Proc. IEEE Workshop Signal Process. Syst., Tampere, Finland, Oct. 2009, pp. 309314.

[11] Bambang Setiadi, Good Fried Panggabean, Josaphat Tetuko Sri Sumantyo, and Koo Voon Chet, “Development of raw data processing system for JX-2 UAV using mobile heterogenenous computing,” Journal of Unmanned System Technology, 23 July 2016 – In press.

[12] Wu, Y.; Chen, J.; Zhang, H. A real-time SAR imaging system based on CPUGPU heterogeneous platform. In: International Conference on Signal Processing (ICSP), 11, 2012, Beijing. Proceedings Piscataway: IEEE, 2012. v.1, p. 461-464.

[13] M. Cafaro, I. Epicoco and S. Fiore et al. Near real-time parallel processing and advanced data management of SAR images in grid environments, Journal of Real-Time Image Processing, 4(2009), 219-227.

[14] Bambang Setiadi, Zafri Baharuddin, Good Fried Panggabean, Hiroaki Kuze, and Josaphat Tetuko Sri Sumantyo, Development of Quicklook Processor for Circularly Polarized Synthetic Aperture Radar onboard GAIA-II Microsatellite, Journal on Progress and Communication and Science (PCS), Vol 2, No 2, pp. 32-38, January 2016.

[15] C.-M. Huang, C.-M. Wu, C.-C. Yang, S.-L. Chen, C.-S. Chen, J.-J. Wang. K.-J. Lee, C.-L. Wey, Programmable System-on-Chip for Silicon Prototyping, Industrial Electronics, IEEE Transactions, Vol 58, Issue 3, pp. 830-838, March 2011.

[16] Xin Xiao, Rui Zhang, Xiaobo Yang and Gang Zhang, Realization of SAR real-time processor by FPGA, IEEE International Geoscience and Remote Sensing Symposium, vol.6, 2004, pp. 3942-3944.

[17] A. M. SMITH, A new approach to range-Doppler SAR processing, International Journal of Remote Sensing, Volume 12, Issue 2, 1991.

[18] Remote Sensing Technology Center: PALSAR Sample Picture, http://www.alosrestec.jp/en/staticpages/index.php/service-sampledata-03, 2009.

[19] Xilinx, Fast Fourier Transform v9.0, LogicCORE IP Product Guide, Vivado Design Suite, PG109, November 18, 2015.

[20] A. W. Lim, S.-W. Liao, and M. S. Lam., Blocking and array contraction across arbitrarily nested loops using affine partitioning, ACM SIGPLAN NOTICE, 36(7):103112, July 2001.

[21] H. Izumi; K. Sasaki; K. Nakajima; H. Sato, An efficient technique for corner-turn in SAR image reconstruction by improving cache access, Proceedings 16th International Parallel and Distributed Processing Symposium, 2002.

[22] Sotiriou-Xanthopoulos E., Diamantopoulos D., Economakos G. (2015) Evaluation of High-Level Synthesis Techniques for Memory and Datapath Tradeoffs in FPGA Based SoC Architectures. In: Sano K., Soudris D., Hbner M., Diniz P. (eds) Applied Reconfigurable Computing. ARC 2015. Lecture Notes in Computer Science, vol 9040. Springer, Cham.