FMCW Rail-mounted SAR: porting spotlight SAR imaging from MATLAB to FPGA

A. Melnikov, J. Le Kernec
EEE Dept., University of Nottingham Ningbo China
Ningbo, China
julien.lekernec@nottingham.edu.cn

D. Gray
EEE Dept., Xi'an Jiatong Liverpool University
Suzhou, China
derek.gray@xjtlu.edu.cn

***Abstract*—In this work, a low-cost laptop-based radar platform derived from the MIT open courseware has been implemented. It can perform ranging, Doppler measurement and SAR imaging using MATLAB as the processor. In this work, porting the signal processing algorithms onto a FPGA platform will be addressed as well as differences between results obtained using MATLAB and those obtained using the FPGA platform. The target FPGA platforms are a Virtex6 DSP kit and Spartan3A starter kit, the latter is also low-cost to further reduce the cost for students to access radar technology.**

***Index Terms*—SAR, Doppler, ranging, FPGA, radar**

I. Introduction

This project is based on the MIT open course project "Build a small radar system capable of sensing Range, Doppler and Synthetic Aperture Radar Imaging" [1].

This laptop-based radar is capable of measuring Range, Doppler and performing SAR Imaging. FPGA platforms like Virtex 6 or Spartan-3A starter kit [2] could be used to replace the laptop and provide real-time processing capability onboard, for example, a multi-rotor UAV in flight to reduce the data sent to the control station for applications such as urban mapping.

In section II, the modifications to the MIT radar frontend will be covered. In section III, the signal processing algorithms will be briefly introduced. Section IV will discuss the implementation of those algorithms on FPGA. Finally section V will compare results obtained using MATLAB against the real-time implementation on FPGA.

II. Hardware Modifications

The radar front end for the most part is similar to the MIT radar. However, the signal modulator used a XR-2206 [3] which is no longer manufactured and has been replaced here with the next generation XR2209 [3]. The modification to the circuit is shown in Fig. 1.

The triangular wave output is fed to the voltage-controlled oscillator yielding an up- and down-chirp of 92MHz bandwidth (2.405MHz to 2.497MHz – ISM band) with a period of 40ms.

The radar can be switched between Frequency-Modulated Continuous Wave (FMCW) operation for ranging and SAR to CW for Doppler measurement by simply reconnecting the voltage-controlled oscillator input to a constant voltage of 2.455V to produce a constant 2.413GHz.

The antenna has been modified slightly to have a smoother surface using a 1-liter beer can as opposed to a coffee can and losing the lid to reduce losses. The DIY dipole in cylindrical waveguide "can-tenna" (shown in Fig. 2) has an inner radius of 42.5 mm, an outer radius of 43 mm, a length of 28.1mm, the dipole-to-short circuit distance was 19mm and the dipole arms were 23.5 mm long and 4 mm wide. The measured directivity was about 8dBi determined experimentally at 2.45 GHz, and the return loss of the Transmitter (Tx) and Receiver (Rx) antennas were measured using a VNA, Fig. 3 respectively dipole 01 and 02. The S_{11} d -10dB bandwidth was about 200MHz from 2.35 GHz to 2.55 GHz which fitted the requirements for the radar. The Tx and Rx antennas were used in VV polarization to reduce mutual coupling. The radar Tx power was about 12dBm. The experimental frontend is depicted in Fig.4.

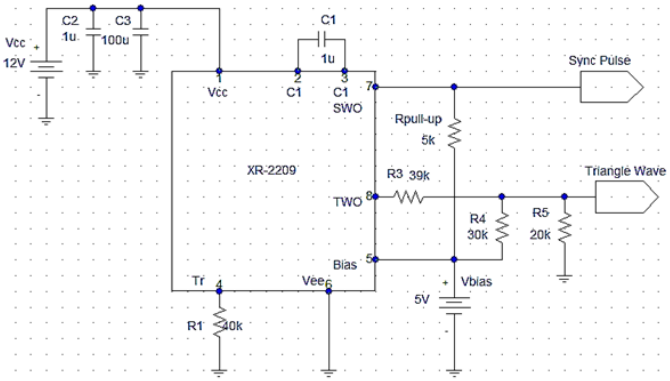


Fig1. Modified function generator using XR2209.

Fig2. Endfire dipole cantenna

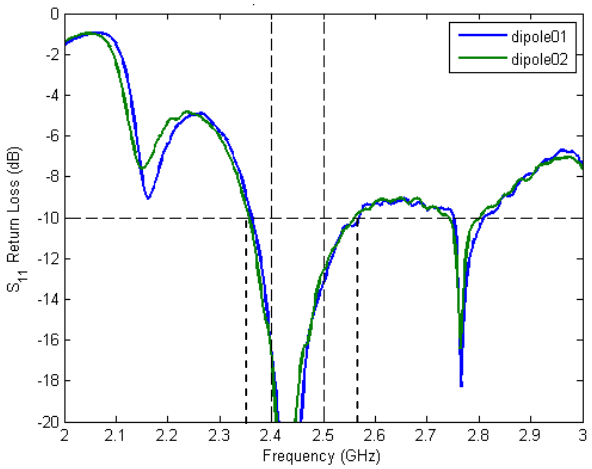


Fig3. Measured return loss of the antennas: Tx "dipole 01" & Rx "dipole 02."

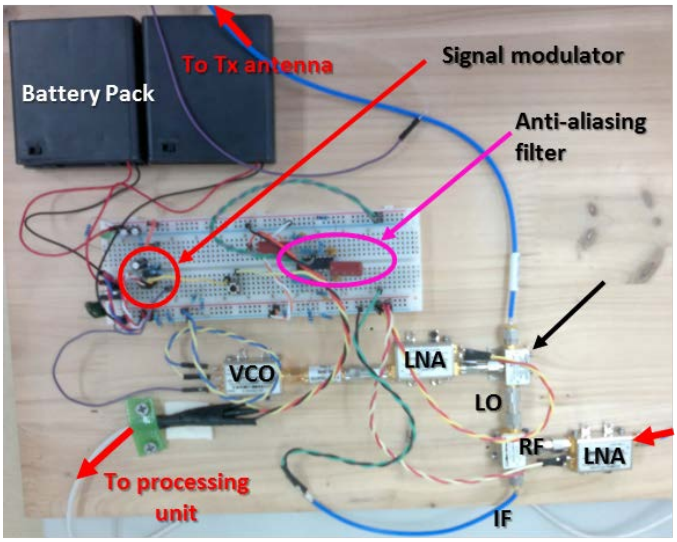


Fig4. Radar frontend implementation

III. Signal Processing: MATLAB Code Analysis

This radar can measure range, Doppler for moving targets and perform SAR imaging. The following subsections will introduce the details of the processing algorithms.

This type of frontend uses "stretch processing" [4] and is used to process Linear Frequency Modulated (LFM) waveforms to relax constraints on the analog-to-digital converter (ADC). Decomposing it in a few steps first, it down-converts the received signal with a reference LFM waveform then goes through an anti-aliasing low-pass filter and is fed to an ADC and the subsequent signal processing algorithms are based on filtering using banks of Narrow Band Filters (NBF) as shown in Fig. 5.

A. Ranging and Doppler

In this case, the processing algorithm will extract the frequency tones, proportional to the target range and the stretch processing effectively converts the time delay in a received signal into a beat frequency.

Considering the case when the radar receives returns from a few close (in range) targets, the transmitted up-chirp is given by:

$$s_t(t) = cos\left(2\pi\left(f_0 t + \frac{Bt^2}{2\tau}\right)\right) \quad (1)$$

where Äis the half priod of the up- and down- chirp 20ms, f_0 is the start frequency, B is the signal bandwidth and t ∈ [0, Ä].

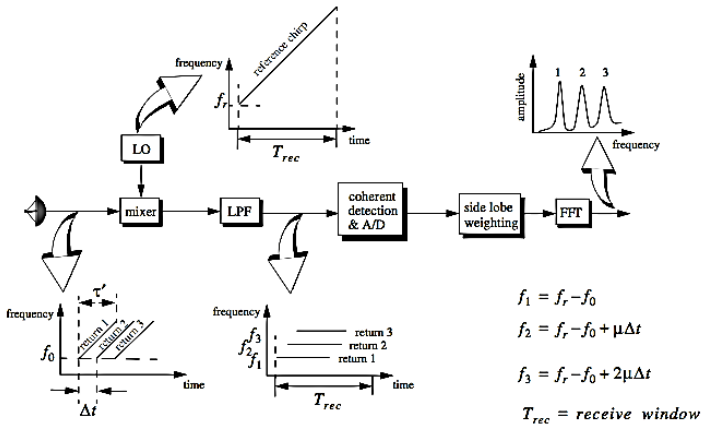


Fig5. Stretch processing block diagram

The beat frequency deviation for Range is:

$$f_b = \frac{2BR}{c\tau} \leftrightarrow R = \frac{f_b c\tau}{2B} \quad (2)$$

where R is the target range. The beat frequency deviation for Doppler is:

$$f_d = \pm\frac{2v}{c-v} f_0 \approx \pm\frac{2v}{\lambda} \leftrightarrow v \approx \pm\frac{f_d\lambda}{2}. \quad (3)$$

where v is the target velocity, c the speed of light and » is the wavelength of the carrier frequency f_0.

1) Ranging

For ranging, the radar operates in FMCW mode. Chirp suffers from range-Doppler coupling thus with up and down chirp the distance and Doppler can be determined unambiguously. However, here, it is assumed that the distortion caused by the observed targets is negligible and only the up chirp was used to simplify the processing.

The first step of the signal processor must extract the up-chirps from the raw echo signal, making use of the synchronization pulses.

The data is split into vectors of 20ms and arranged in rows to form a Range-time matrix. Then a 2-pulse moving target indicator (MTI) filter was used to suppress clutter coherently. Finally, an "Along-range" FFT implements the filter bank. Resolution along the time axis equal to the size of the window, i.e. 0.02 s and range resolution is defined as the ratio of sampling frequency to the FFT length times scaling factor, relating the frequency and range (2) and yields about 0.09 m (0.3 ft) with an ADC sampling frequency of 44.1 kHz.

2) Doppler

For Doppler signal processing the radar works in CW mode. The recorded data is split into vectors, corresponding to a set time (in this case 0.1s) and arrange those vectors as rows to form a Doppler-time matrix.

Then, the DC component was subtracted from every vector to remove stationary targets' (clutter) echoes.

Finally, a bank of narrowband bandpass filters were implemented using fast Fourier Transform (FFT) along the rows of the Doppler matrix, which processed the whole spectrum of every vector, with the length of FFT having determined the bandwidth of the individual filters. The output was the Doppler information from the targets present in the field of view of the radar against time.

B. SAR imaging

The radar transmits and receives a series of pulses as it moves along its path. Every time when the pulse is transmitted the radar occupies a new along-track position. The numbers of elements that constitute the "synthetic array" are dependent on the antenna beam pattern and the distance between measurements.

The coherent processing of the radar returns will produce a narrow azimuth beamwidth, for cross-range (along-track) and allow the generation of 2D images (range and cross-range).

Spotlight Range Migration Algorithm (SRMA) is used for SAR imaging and its main features are explored next.

1) Spotlight Range Migration Algorithm

The Spotlight Range Migration Algorithm (SRMA) compared to some other SAR algorithms (Polar Formation Algorithm, for instance) has an important advantage assuming that the illuminating wave fronts are not planar, that allows to avoid space-variant defocusing and geometric distortion of the final image.

The image processing can be split in two parts: pre-processing and image formation.

a) Pre-processing

Before the image formation matrix is created, where the rows will be populated with the echo signals received by different synthetic array elements and the columns will contain those scatterings in range direction.

The MATLAB program provided in [1] for SAR imaging presents some technical issues that were identified as follows:

- to separate the data coming from the different array elements and parse the data within a single array element a triggering threshold was used, but this approach did not take into account that the mechanical switch used to control the reference signal ON/OFF is not debounced, producing oscillations during "start-up", which made the approach to search for the start of up-chirp was unreliable;
- to average the spectrum of the signal, obtained within a single array element a fractional number of impulses were used where it should be an integer number of impulses to avoid distortions in the results;
- to perform the Hilbert transform, an inverse FFT (IFFT) and then a FFT on half of IFFT result was used; in spite of the fact that this approach is correct [5], it does not produce exactly same result as its mathematical analog and limits the amount of data available for further processing by half and increases computational load on the processor.

The modified program used in this work corrected the identified problems and now averages 8 pulses per array element, the synthetic aperture has 55 elements and uses the Hilbert function provided by MATLAB or a Finite Impulse Response (FIR) filter implementation of the Hilbert transform [5] to compare with hardware (HW) implementation).

b) Image formation

Image formation processor (IFP) for the SRMA is given in Fig. 6 and it comprises 4 stages: along-track FFT, Matched filtering, Stolt interpolation and 2D iFFT that are detailed in [6].

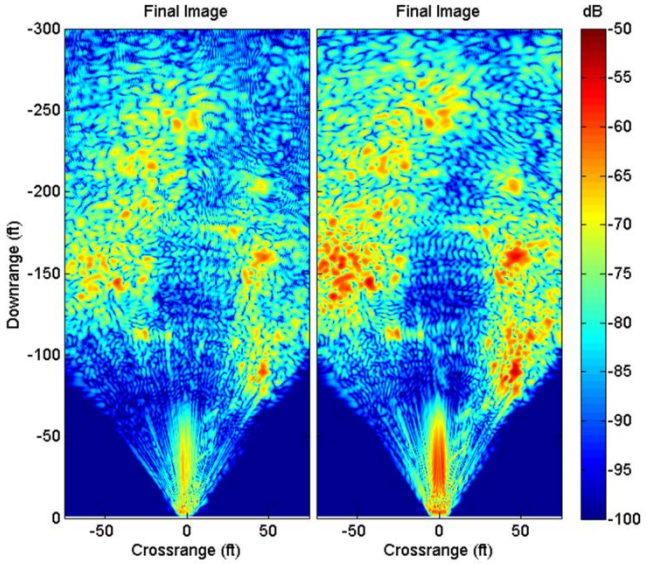


Fig6. 2D-SAR images of data from [1]; (left) original algorithm, (right) modified algorithm.

Based on the dataset from [1], the 2D image obtained with the modified algorithm (Fig.7) is comparable to the image provided in [1]. All the features are present although some differences can be seen, which result from changes implemented in the modified algorithm.

IV. FPGA IMPLEMENTATION

For speed, the HDL code was designed using System Generator (SysGen), a design tool by Xilinx ISE, which is highly suitable for algorithm exploration, design prototyping and model analysis. This tool is based on MATLAB Simulink, but uses Xilinx HDL block functions to design the system which can then be synthesized and ported on the intended platform [7] in this case a Virtex-6 DSP kit (US$4000) and a Spartan 3A Starter kit (US$242).

In order to validate the HW design, the results were benchmarked against the output from the programs in [1] and the programs and the HW designs use the datasets provided in [1] to produce images.

A. Range processor

The range processor, employs a 3-pulse MTI filter [4] instead of 2-pulse MTI filter in the original program.

It was decided to replace the 2-pulse by the 3-pulse MTI filter in a HW design, because of its better performance in terms of noise cancellation. The 3-pulse MTI filter has a broader null at the clutter frequency than the 2-pulse, which gives better rejection, but on the other hand it distorts the spectrum of the signal to the less extent than proposed in [1] magnitude canceller.

The HDL flow is as follows. First, the up-chirps are extracted from the raw data in chunks of 882 samples (0.02 s) at 44.1 kHz . Then, the data goes through a 3-pulse MTI filter. The data is then zero-padded before being processed by a Radix-2 Fast-Fourier Transform (FFT) and then the magnitude of the response is calculated to display the range intensity against time.

B. Doppler processor

The Doppler processor takes in 4410 raw samples, which were then zero-padded to a length of 16384 before the FFT. The magnitude was calculated after clutter cancellation.

C. SAR processor

The SAR processor comprises a pre-processor and an image-processor.

The raw data was fed to the pre-processor where the up-chirps were extracted. 8 chirps were averaged per array element. Then the data went through a clutter canceller for the FIR implementation of the Hilbert transform [5].

The output of the Hilbert transform went to the image processor. The signal was first windowed (in this case Hanning). The matrix was then transposed and then zero-padded for the along-track FFT. The result was matched filtered against coefficients stored in memory. The next step is the Stolt interpolation [6] was used to correct the range curvatures for scatterers at different ranges. After interpolation, windowing is once again applied (in this case Hanning) before the 2D-inverse FFT to finally generate the image.

Here it must be mentioned that the second part of the SAR processor was extremely memory consuming since it performed operations in both range and cross range dimensions and the data flow rearrangement requires usage of the memory. This imposed limitations on the maximum amount of data being processed, since the depth of the RAM blocks in the Virtex-6 board is limited to 64 Kb only [8]. This limits the maximum possible size of FFT, and causes degradation in image resolution in cross-range.

Since every array element contains 882 samples and there are 55 array elements overall, the maximum possible FFT size in the cross range direction is 64, which will hold in the available memory to rearrange the signal order from range to cross-range wise flow (882×64= 56Kb < 64 Kb), but a small FFT length in the cross-range does not allow to build a final image with good resolution as in Fig.7.

Another problem was that the SAR processor of such a moderate size requires more memory than the board (Virtex-6 DSP kit) can provide and even though the HDL code for the corresponding SysGen design was generated it cannot be mapped onto the FPGA platform (the number of RAM blocks required to map the design was 577, which is 138% of the available blocks – 416).

Table I summarizes the characteristics of the implemented processors mapped on the target platforms. From the results in Table I, it can be clearly seen that the FPGA platform Virtex-6 and is not feasible on Spartan-3A without modification is capable of processing the data in real-time for the range and Doppler processors as well as the SAR-pre-processor. The image processor clearly needs to be redesigned to fit on any of the platforms. To fit the Doppler processor on Spartan 3A, an algorithm based on zoom-FFT[1] [5] was used.

Table I: HW implementation synthesis2

Signal processor	Range		Doppler			SAR PP	SAR IP[3]
Platform	V6	S3	V6	S3	S3 zoom	V6	V6
Max Freq (MHz)	20.02	n/a	17.92	n/a	33.33	86	n/a
Slice flip-flops	n/a	132%	n/a	105%	14%	n/a	n/a
Slice registers	3%	789%	2%	609%	38%	1%	5%
Slice LUTs	6%	479%	4%	388%	51%	9%	10%
Memory	4%	825%	1%	500%	45%	9%	138%
IO	29%	45%	28%	44%	23%	57%	114%

V. COMPARING THE RESULTS FROM MATLAB AND FPGA PROCESSOR

Both SW and HW implementations use the original datasets provided in [1] for the validation of the HW implementation. The results for the three processors: Range, Doppler and SAR are presented in this section.

A. Range processor results

Using the dataset “Two People Walking in the Woods” from [1], the results for the HW implementation of the Range processor are shown in Fig. 8 and the relative error is shown in Fig.9. Further analysis shows that 99.97% of errors are within ±5% of the mean error.

B. Doppler processor results

Using the dataset “Tremont Street off Newton Corner” from [1], the results for the HW implementation of the Doppler processor are shown in Fig. 9 and the relative error is shown in Fig.10. Further analysis shows that 98.2% of errors are within ±5% of the mean error. The results for the zoom-FFT processor on Spartan3A are shown in Fig.11.

---

[1] zoom FFT is a DSP technique useful in case when only a small portion of a signal’s bandwidth, over the full frequency range, contains the valuable information for any particular application.

[2] V6: Virtex 6 DSP kit, S3: Spartn 3A starter kit, PP: pre-processor, IP: image processor

[3] Estimated with the down-sized image processor because the current algorithm can’t be mapped on the target platform because it exceeds the available resources and the full-sized image can’t be simulated

C. SAR processing results

Using the dataset "Back of Warehouse" from [1], the results for the HW implementation of the Doppler processor are shown in Fig. 12 and the relative error is shown in Fig.13.

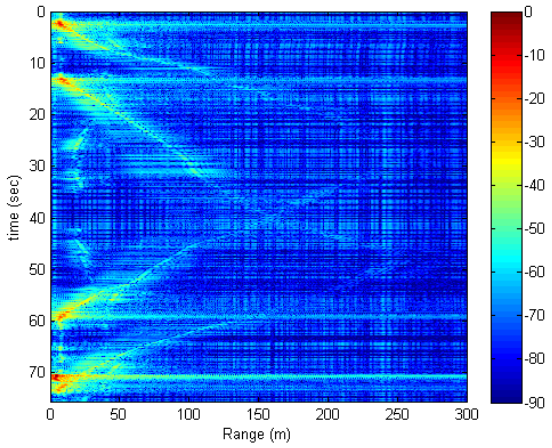


Fig7.HW range processor result for "Two People Walking in the Woods" data from [1] on Virtex 6.

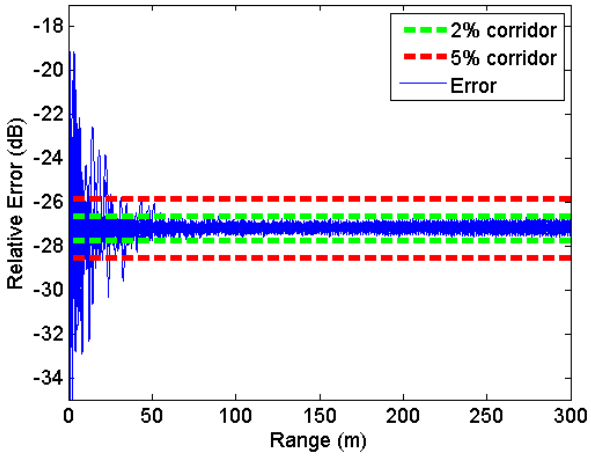


Fig8.Relative error distribution of HW vs SW implementations (mean error -27.2dB) for range on Virtex 6.

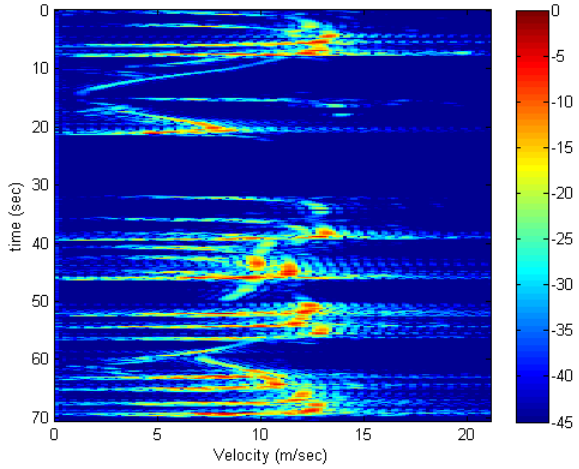


Fig.9.HW Doppler processor result for "Tremont Street off Newton Corner" data from [1] on Virtex6.

It can be seen that the results produced using HW presents all the features present in Matlab for all three processors. The error between the two are mainly due to the difference between fixed and floating point notations, different FFT sizes, zero-padding and the extra noise effect, introduced by FFT FPGA block, when it performs a real-valued transform [9].

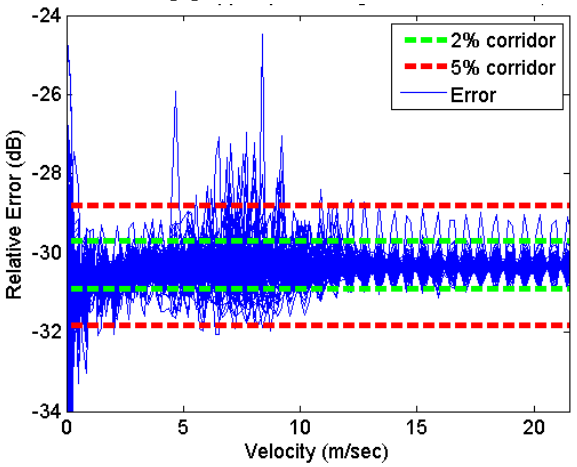


Fig.10.Relative error distribution of HW vs SW implementations (mean error -30.3dB) for Doppler on Virtex6

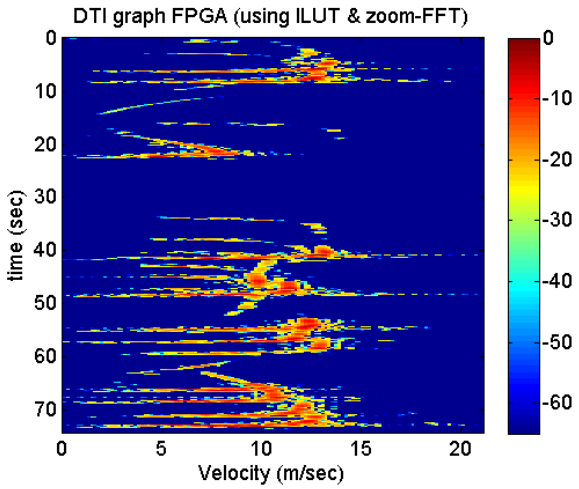


Fig.11.HW Doppler processor result for "Tremont Street off Newton Corner" data from [1] on Spartan 3.

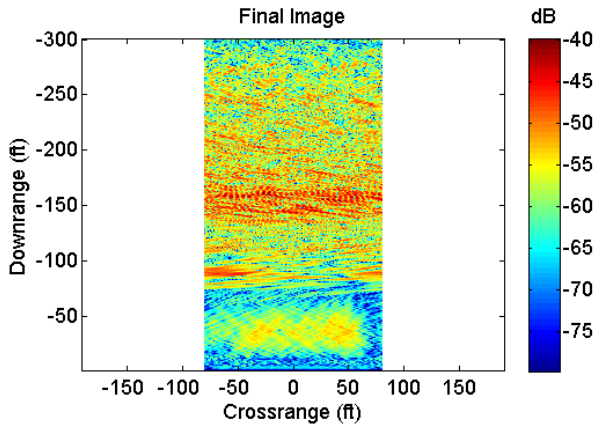
Table II: Error range in dB between MAtlab and HW Virtex6 implementation in fixed point

Error dB	Doppler	Range	SAR PP	SAR IFP
Max	-24.4	-17.8	-36	-12.5
Mean	-30.3	-27.2	-70.7	-28.2

VI. Conclusion

During this project, the theoretical basics of the FMCW radar were considered and an experimental radar setup was implemented. The software (SW) signal processors, based on the high-level code, proposed in [1], were used to verify the radar's functionality and the results obtained using modified MATLAB programs from [1] were used to validate the results of the signal processed with FPGA.

A comparison of the results of the range signal processor in both HW and SW, proved the validity of the FPGA-based design with a mean error of -27.2dB.



Fug12. HW SAR processor result for "Back of Warehouse" data from [1].

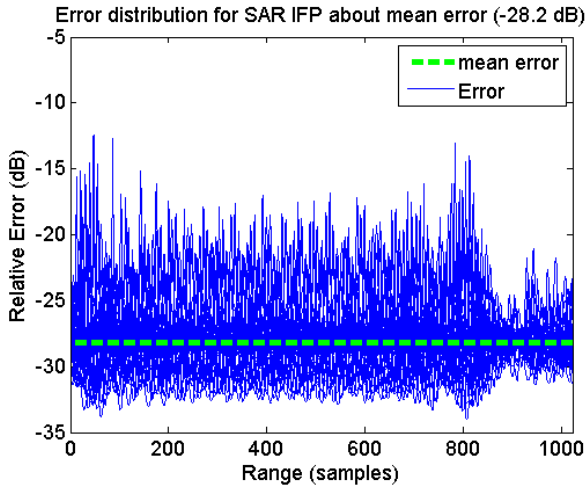


Fig.13.Relative error distribution of HW vs SW implementations (mean error -28.2dB) for IFP on Virtex 6.

The Doppler HW processor when compared to the SW implementation yields a mean error of -30.3dB results obtained are similar.

Also based on the design summary reports for Range and Doppler, the designs can't be fitted on Spartan3A [2] without modifications. The Doppler processor was ported on Spartan with a zoom-FFT based algorithm.

The SAR Image processing algorithm in HW, following the flow of the modified SW processor, was subdivided into 2 parts – signal pre-processing and image formation routine. The latter was implemented in HW but the limitations in the memory architecture of the FPGA board imposed a downsized copy of the SW processor, i.e. the signal processing blocks are identical but used a reduced dataset. Even though the resolution of the final image obtained in the HW is much poorer than the one produced in MATLAB, the digital signal processing algorithms yielded a mean error compared to MATLAB of -28.2dB.

This project proved the concept of the HW implementation of the Doppler, Range and SAR Image processing algorithm simulating the designs in the SW environment. Future research activities will focus on further validation of the signal processing techniques using HW co-simulation, followed by optimization it is expected that Sysgen automatic HDL coding yields similar results to MATLAB did when generating C code automatically. A complete HW implementation of the FMCW radar on the target Virtex-6 or Spartan-3 for Range DSPs using the on-board interfaces: ADCs and displaying the results in real-time on a VGA monitor.

Also, the separate research efforts may be undertaken on further DSP optimization, in particular in SAR Image processing, by making better use of memory resources or enhancement of the FFT algorithm for Range and Doppler processors.

The main disadvantage of the standard FFT is that it requires zero-padding to achieve a finer resolution and has to process the whole spectrum of the signal, whereas quite often we are only interested in a small fraction of the spectrum. A promising solution to improve efficiency could be a zoom-FFT, which allows the implementation of a standard FFT of smaller size to only "zoom" on the desired part of the spectrum, also indirectly improving the DSP efficiency by reducing amount of zero-padding required [5]. It was shown that using a zoom-FFT based algorithm the Doppler processor could fit on Spartan 3A with LUTs. The LUTs to implement the log10 calculations is the main cause of the degradation in image quality.

The cost of the frontend plus the Spartan 3A starter kit platform would bring the Doppler (and maybe Range) processor(s) together to about US$500. However it would seem that implementing the SAR processor on Spartan 3A starter kit might prove to be more challenging.

ACKNOWLEDGMENT

The authors thank the Faculty of Science and Engineering and the EEE Department of University of Nottingham Ningbo for funding the equipment that allowed the authors to undertake the reported experimental work.

REFERENCES

[1] G. Charvat et al., "RES.LL-003 Build a Small Radar System Capable of Sensing Range, Doppler, and Synthetic Aperture Radar Imaging", January IAP 2011. (MIT OpenCourseWare: Massachusetts Institute of Technology),

[2] www.xilinx.com, Spartan-3A starter kit & Virtex-6 DSP kit

[3] www.exar.com, XR2206(obsolete), XR2209

[4] "Radar systems analysis and design using MATLAB", B.R. Mahafsa, 2000, editor Chapman and Hall/CRC

[5] L. Lyons, "Understanding Digital Signal Processing", 2001, Prentice Hall, USA

[6] "Spotlight Synthetic Aperture Radar Signal Processing Algorithms", W. Carrara, R. Goodman, R. Majewski, 1995, Artech House, USA.

[7] Xilinx, "UG UG626. Synthesis and Simulation Design Guide", 2012, www.xilinx.com

[8] Xilinx, "UG363. Virtex-6 FPGA Memory Recources", 2014, , www.xilinx.com

[9] Xilinx, "Datasheet. LogiCore IP FFT v7.1", 2011, www.xilinx.com