Porting Spotlight Range Migration Algorithm processor from Matlab to Virtex 6

A. Melnikov[1] J. Le Kernec[2] D. Gray[3]

***Abstract* – This paper describes the implementation and optimization of a Synthetic Aperture Radar process Spotlight Range Migration Algorithm processor on FPGA Virtex 6 DSP kit that fits on the chip. The mean/max error compared to a software implementation is -54/-28.74dB for 55 elements and 882 samples.**

1 INTRODUCTION

In [1, 2], the authors proposed a HDL design of the Spotlight Range Migration Algorithm (SRMA) implementation using System Generator (SysGen), a design tool by Xilinx ISE [3] to be ported on a Virtex-6 DSP kit [4] (US$4000).

The SRMA algorithm for Matlab is based on the MIT open course project "Build a small radar system capable of sensing Range, Doppler and Synthetic Aperture Radar Imaging" [5].

In order to validate the hardware (HW) design, the resulting SAR images were benchmarked against the output from Matlab and both HW and software (SW) implementations using the datasets from [5].

In [6], a ω-K Synthetic Aperture Radar (SAR) processor for a 77GHz front end is designed with a combiniation of FPGA (Cyclone II) and DSP (TI TMS320-C6713). In [7], a real-time back projection SAR processor is implemented also for a 77GHz front end on FPGA. The trend is to port signal processing on HW for speed on either FPGA, DSP or a combination of those. In this paper, the focus is on the implementation of an FPGA-only solution.

2 IMAGE FORMATION PROCESSOR DESIGN

The SAR processor consisted of a pre-processor (PP) and an image-processor (IFP).

The raw data was fed to the pre-processor where the up-chirps were extracted. Eight 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] which 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.

2.1 SAR processor implementation

The following two sections discuss the IFP design in detail and suggest the possible ways to reduce the resource consumption in order to fit both algorithms (PP and IFP) onto Virtex-6.

Firstly, it is important to note that the data, being processed by the SAR algorithm, is a vector, but it is very convenient to think of it as a matrix where the rows contain the samples of the echoes received by different array elements giving a matrix D of 55 elements by 882 samples.

When the data is said to be read in the row-wise (range) direction that means that all the data samples, corresponding to the 1st array element (row 1) are being read one-by-one ($D_{1,1}$, $D_{1,2}$, $D_{1,3}$,…, $D_{1,882}$), then all the elements from the 2nd row are being read and so on until the last 55th row is read.

Similarly, the cross-range (column-wise) direction, refers to the following pattern of reading the data, when the 1st column is read first ($D_{1,1}$, $D_{2,1}$,…,$D_{55,1}$), followed by column 2 and so on until the last column is read.

"Hanning Window" block essentially comprises ROM-register storing the window's coefficients (882) and two multipliers for real and imaginary parts of data.

"Matrix transposition" (MTr) entity contains 2 RAM-registers (real and imaginary parts are processed in parallel in IFP), where the windowed data is being written in the row-wise manner, but read in the column-wise order.

To write the data into the memory a simple counter is used, that increments the address every time when the new data sample is available.

To read out the data from the memory in the columns-wise order another RAM-block containing the correct data address sequence, pre-generated in Matlab, is used. It is worth emphasizing here again that the data array is a vector, not a matrix, so the problem of reading out the data in cross-range manner cannot be solved with the single counter (referring to vector notation the address sequence will be – 0, 882, 1763, 2645,…, 48509).

Cross-range zero-padding (ZP) contains 2 FIFO (First-In-First-Out) registers, where the data is being written column-wise and read out one-by-one. The process of reading data out is interrupted every time

[1] EIE Dpt., The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
[2] EEE Dpt., University of Nottingham Ningbo China, 199 Taikang East Road 315100 Ningbo, China, e-mail: julien.lekernec@nottingham.edu.cn, tel.: +86 (0) 574 8818 2453.
[3] Xi'an Jiatong Liverpool University, Suzhou, Jiangsu, China; e-mail: derek.gray@xjtlu.edu.cn.

55 samples are extracted from registers (55 elements in every column) and zero-padded (9 zeros to obtain the new 64-element long sequence).

FFT is performed using designated SysGen block and shift operation is done simply by swapping the last 32 samples and the first 32 samples (in every row) using the "Delay" and "Mux" blocks (the data is split into 2 identical branches, one of them is delayed by 32 clock periods, "Mux" block is used to "activate" the correct channel).

Matched filtering (MF) is preceded by another "MTr" block, absolutely identical to that one, described above, but this time it rearranges data from cross-range to range-wise order.

"MF" itself performs multiplication of the cross-range Fourier transformed data by the coefficients of the matched filter, given by (1) from [6] and stored in another RAM-block:

$$s_{mf}(K_X, K_R) = \exp\left(j\left(-K_R R_s + R_s\sqrt{K_R^2 - K_X^2}\right)\right) \qquad (1)$$

where K_R and K_X are spatial frequencies in range and cross-range directions, respectively, and R_S is the distance to the center of the scene being mapped.

Stolt interpolation is the crucial element in IFP design, it simultaneously remaps data from (K_X, K_R) into (K_X, K_Y) space, removing the range curvature from the radar returns and interpolates the result.

The main difficulties in HW implementation of this procedure are related to the change of the sample rate of the data. Thus, if the input is 882-element long in the range direction, it must be interpolated against 1024-element long K_Y vector (which is the result of linearization of original quadratic K_Y vector, given by (2).

$$K_Y = \sqrt{K_R^2 - K_X^2} \qquad (2)$$

The interpolation process itself is a linear interpolation, given by (3) [7], when the original data $D_{i,j}(K_X,K_R)$ is interpolated in query points K_Y, to find $D_{i,j}(K_X,K_Y)$:

$$D(K_X, K_{Y(t)}) = D(K_X, K_{Y(n-1)}) + \left(D(K_X, K_{Y(n)}) - D(K_X, K_{Y(n-1)})\right) \cdot \frac{K_{Y(t)} - K_{Y(n-1)}}{K_{Y(n)} - K_{Y(n-1)}} \qquad (3)$$

for $K_{R(n-1)}$, $K_{R(n)}$ and $K_{Y(t)}$ if

$$K_{Y(n-1)} < K_{Y(t)} < K_{Y(n)} \qquad (4)$$

where $K_{Y(n)}$ is found using (2) and $K_{Y(t)}$ is the result of its linearization.

From (3) and (4), it can be observed that the Stolt interpolation is a complex process. First the input data samples, satisfying criterion (4), must be found, and then, the new complex values are determined with (3). Also, given that the original data $D(K_X, K_{Y(n)})$ and $K_{Y(n)}$ (before linearization) are 882-sample long and resultant $D(K_X, K_{Y(t)})$ and $K_{Y(t)}$ (after linearization) are 1024-samples long.

Implementation of (3) and (4) is done as follows:

- Condition (4) is used to create an auxiliary data-array (64x882) in Matlab, containing just zeros and ones, to indicate the presence of the valid input data for interpolation. This array will be stored into RAM-block #1 in SysGen design;
- Again, using Matlab, two extra data arrays are generated. The first (64x882), will be stored in RAM-block #2, and it will contain the addresses of the results of interpolation in the output array (64x1024) and at the same time it will point into the memory elements in another array, stored in RAM-block #3 (64x1024), holding results for the operation over $K_{Y(n)}$ and $K_{Y(t)}$ in expression (3).

Thus, the HW implementation of Stolt interpolation can be summarised as follows.

The input data is split into 2 branches, one of them is being delayed by 1 clock cycle, compared to another, since the operation is performed row-wise over successive elements (see expression (3) – successive data samples from the same row are used). Every time when the new set of data (pair of the row's elements) appears at the Stolt-block input, RAM #1 provides a validity indicator for this data set. If this indicator is "false", nothing happens, if it is "true", RAM #2 supplies the address, indicating where the interpolation kernel for this set of data is stored in RAM #3, and provides the memory address where the result of interpolation must be stored, in the output buffer once expression (3) is calculated. When the whole block of the input data is processed, the output buffer is read range-wise.

The last stages are essentially the combination of Hanning Window; Range IFFT (without shift), MTr and ZP in cross-range direction, and again Cross-Range IFFT together form the "2D IFFT" block.

2.2 Synthesis and results

The synthesis of the separate designs for the PP and the IFP is shown in Table 1.

Signal processor	SAR PP (original)	SAR IFP (original)
Max Freq (MHz)	86	n/a
Slice registers	1%	10%
Slice LUTs	9%	17%
Memory	9%	245%
IO	57%	98%

Table 1: Resource utilization for PP and IFP algorithms on Virtex-6 DSP kit. [1-2]

The PP on FPGA yielded a mean error compared to its SW equivalent of -70.7dB and -36dB at most. The IFP had a mean error compared to its SW equivalent of -28.2dB and -12.5dB at most.

3 HARWARE PROCESSOR OPTIMIZATION

The PP is simple, since it mainly involves data selection using a triggering threshold and conceptually nothing can be done to optimize it. The output buffer is required when PP is a standalone processor to interface with the PC. So by merging PP and IFP, it can be removed thus saving about 97 KB of memory (48,510 bytes both for real and imaginary parts).

The IFP, on the other hand, leaves room for much improvement. The algorithm optimizations are explained next.

3.1 Cross-range zero-padding

ZP is complimentary to the process of reading out the data from the RAM-block, used to store the data after the Hanning windowing. Interrupting the data flow to insert zero-valued data samples can be performed when the data is read column-wise from RAM-blocks. Eliminating the "ZP" entity will save another 97 KB of memory.

Secondly, the RAM-block, containing the addresses for column-wise data reading is replaced by a custom counter, generating the sought sequence of addresses saving another 47,510 bytes. This can be done as follows:

- The 1st counter counts from 0 (1st memory slot in the data RAM-block) to 47,628 (address of the 1st element in the 55th row) with increments of 882;
- The 2nd counter counts from 0 to 881, but this counter must be incremented only once every 55 increments of the 1st counter.

3.2 Matched filtering.

Since stretch processing is used, the distance to the center of the target scene is equal to zero [6] and expression (1) will be equal to unity for any (K_X, K_R). The "MF" entity can be omitted saving 56.4 KB of memory.

3.3 Stolt Interpolation and Matrix transformation from cross-range to range.

The data validity indicator, since it is used to trigger the address RAM and interpolation kernel RAM, may also be used as a foundation to save some memory on those two RAM-blocks.

Previously the whole address array (64x882) and kernel array (64x1024) were stored into corresponding RAM addresses, but most of the elements are zeros and storing them will not improve the entity's performance in any way. It is possible to store only non-zero values into RAM #2 and #3 and read out the data using a simple counter, enabled only when the data validity indicator is "true". This will save 29 KB for RAM #2 (addresses) and 38.1 KB for RAM #3 (interpolation kernel).

Also, the Stolt interpolation should be performed in the range direction, meaning that two samples from the same row, but different columns must be present in the input to the block and that was the reason to transform the data flow after along-track FFT from cross-range-wise to range-wise. Delaying the data in the second branch by 64 clock periods (subsequent row elements are separated by 64 samples when data flows column-wise), will give the same result as MTr, since in this way the data from the same row but different columns will always be present at the input of the "Stolt Interpolation" entity, which saves another 169.3 KB of RAM (2x56.4 KB for MTr RAM and 56.4 KB for address RAM).

3.4 Cross-range zero-padding in 2D IFFT.

This procedure of merging functionality of the "ZP" entity with "MTr" in the 2D IFFT entity and replacement of the address RAM with a custom counter as previously described in Section 4.3 will save an extra 192 KB (2x65,536 bytes for real and imaginary parts' RAMs and 65,536 bytes for address RAM), giving a total count of about -726 KB to the overall efficiency improvement.

Table 2 provides the resource consumption of the combined PP+IFP designs for Virtex-6 and shows the difference compared to the sum of the previous resources for PP+IFP in [1-2].

From Table 2 the following aspects of the optimization should be mentioned:

- The speed of the Pre-processor is reduced by 82 MHz (-72%), compared to the original design, which is the result of switching to floating-point operation, but it increases accuracy and the speed is sufficient to carry SAR operations given the ADC sample rate 44.1kHz.
- Removal of the output buffer and associated circuitry from Pre-processor block positively affected resource consumption all around.
- Memory optimization procedures for the IFP made its size compatible with the resources available on a single Virtex-6 board.

Signal processor	Previous [1-2]	Optimized
Max Freq (MHz)	N/A	11.7(N/A)
Slice registers	6%	12%(+3491)
Slice LUTs	19%	32%(+15061)
Memory	254%	94%(-683)
IO	155%	29%(-433)
Error Mean/Max	-28.2/-12.5dB	-54/-28.74dB

Table 2. Resource consumption for original and modified PP and IFP designs.

Figure 1 gives the relative error in amplitude and phase between the optimized HW signal processor and the SW signal processor outputs shown in Figure 2. The mean error is about -54dB and the maximum error is -28.74dB. It is worth noting here that the row data used for signal processing in SW was acquired with a 16 bit resolution ADC, whereas for HW processing it was resampled to have 14 bit resolution, corresponding to the resolution of the on-board ADC (FMC for Virtex-6), which is another source of the error between the HW and SW implementation.

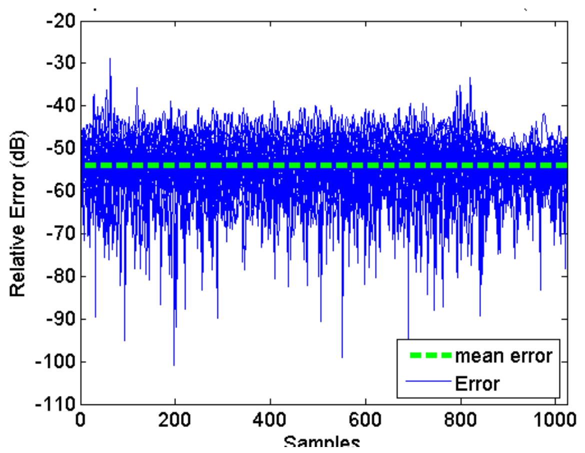

Figure 1. Complex error distribution.

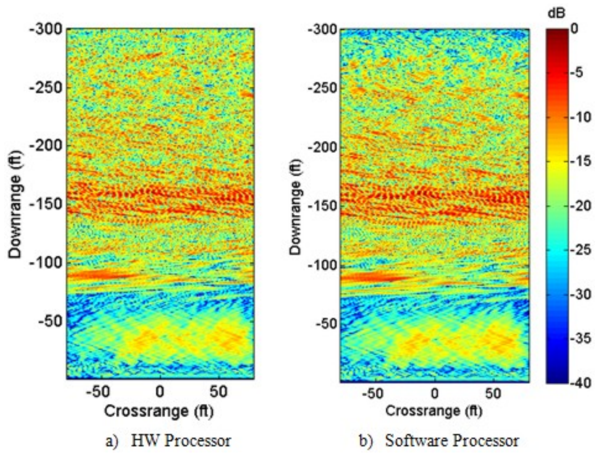

Figure 2. HW (right) vs SW (left) SAR Images.

4 CONCLUSION AND FUTURE WORK

A SAR processor of limited size has been successfully designed for HW implementation using SysGen. Even though its performance is comparable with the SW processor, the cross-range resolution of the SAR image is low due to the limited size of the cross-range FFT: 64 instead of 2048.

Further developments will focus on the resolution improvement and the crucial step is to reduce memory consumption in the "Stolt interpolation" stage resampling the input data, which will save another 240 KB of RAM.

The maximum resolution in the previous designs was limited by the maximum size of the on-chip RAM block (64 KB). The next phase will be to use external RAM (512MB available on-board) to handle matrices larger than 64 KB cross-range FFT and IFFT operations. Also since the SAR processor speed is large compared to the requirement, processing the real and imaginary part sequentially as opposed to simultaneously will further save resources in terms of IO and memory blocks. The only time when the real and imaginary part need to be together is for the FFT and IFFT operations, for all other processes Windowing, ZP, even Stolt can be done sequentially.

References

[1] A. Melnikov, J. Le Kernec and D. Gray, "FMCW Rail-mounted SAR: porting spotlight SAR imaging from MATLAB to FPGA", Int. Conf. on Signal Process., Commun. Computing (ICSPCC 2014), Guilin, Aug. 2014.

[2] A. Melnikov, J. Le Kernec and D. Gray, "A Case Implementation of a Spotlight Range Migration Algorithm on FPGA Platform", Int. Symp. on Ant. Prop. (ISAP 2014), Kaohsiung, Taiwan, 2-5 Dec. 2014.

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

[4] www.xilinx.com, Virtex-6 DSP kit

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

[6] A. Haderer, C. Wagner, R. Feger and A. Stelzer, "A 77-GHz FMCW Front-end with FPGA and DSP Support", Int. Radar Symp., Wroclaw, Poland, 21-23 May, 2008.

[7] A. Haderer, P. Scherz, J. Schrattenecker and A. Stelzer, "Real-Time Implementation of an FMCW Backprojection Algorithm for 1D and 2D Apertures", European Radar Conf. (EuRAD 2011), Manchester, UK, 12-14 Oct 2011.

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

[7] G. James, *Modern Engineering Mathematics*, Harlow: MyMathLab., Pearson Ed., 4th ed, 2010.