Synthetic Aperture Radar on Low Power Multi-Core Digital Signal Processor

Dan Wang
Texas Instruments
Dallas, TX, USA

Murtaza Ali
Texas Instruments
Dallas, TX, USA

***Abstract*—Commercial off-the-self (COTS) components have recently gained popularity in Synthetic Aperture Radar (SAR) applications. The compute capabilities of these devices have advanced to a level where real time processing of complex SAR algorithms have become feasible. In this paper, we focus on a low power multi-core Digital Signal Processor (DSP) from Texas Instruments Inc. and evaluate its capability for SAR signal processing. The specific DSP studied here is an eight-core device, codenamed TMS320C6678, that provides a peak performance of 128 GFLOPS (single precision) for only 10 watts. We describe how the basic SAR operations can be implemented efficiently in such a device. Our results indicate that a baseline SAR range-Doppler algorithm takes around 0.25 second for a 16 M ($4K \times 4K$) image, achieving real-time performance.**

I. INTRODUCTION

Synthetic Aperture Radar (SAR) achieves high-resolution remote sensing imagery by moving the radar platform to create the effect of a large antenna. It involves advanced signal processing techniques to analyze the phase shift information and obtain fine resolution in the direction perpendicular to the beam direction. SAR technique has been widely applied in different areas including disaster observation and management, geological mapping, weather forecast, and strategic surveillance of military sites. There are different operating modes for the SARs, such as stripmap SAR, spotlight SAR, interferometric SAR and bistatic SAR. Each mode is tailored to the specific application requirements from the perspectives of resolution, coverage area and geometry information types.

The formation of the SAR image from the defocused received signal was originally obtained based on the principles of Fourier optics [9] using laser beams and lenses. Although producing well-focused images, the optical processor requires precise alignment of high-quality lenses and leads to limited dynamic range. Digital processor was first introduced by Cumming to produce images from the Seasat-A SAR data [6], which demonstrated the advantages of signal processing technology for efficient image formation. With decades of research, the processing techniques have reached a mature stage, while the remaining challenge is how to form a high resolution image in real-time considering the involved intense computational effort.

The signal processing procedures have been implemented in different commercial off-the-shelf architectures (COTS). Rudin [13] proposed to implement the polar formal algorithm for SAR imaging on the IBM dual cell-based platform which features high aggregate memory bandwidth and peak floating point performance. The reported processing time for a 100-Megapixel SAR image is 73.86 seconds. For comparison, the work in [11] evaluated similar SAR implementations on three generations of 64-bit Intel Quad Core CPUs. The Spiral framework [12] was applied to automatically generate efficient program achieving vectorization, parallelization and memory hierarchy tuning. The test results show that the runtimes for 16-Megapixel and 100-Megapixel SAR images are 0.56 and 3.76 seconds respectively. More recently, the General-Purpose Graphics Processing Unites (GPGPU) show great promises to improve the SAR image formation speed due to their increasing number of cores and larger vector widths. Bisceglie, et al. [4] discussed the processing power of GPGPU for a typical range-Doppler algorithm that consists of independent and separable steps for massive parallel computation. The execution time for a 132-Megapixel image is more than 8 seconds with near real-time performance. Due to the variations of the algorithms implemented, it is difficult to have a fair comparisons among architectures for SAR applications. These results, however, provide a good view of the current status of SAR implementations in various COTS architectures.

Besides the above mentioned hardware architectures, the new generations of multi-core Digital Signal Processor (DSP) is also a competitive platform for computational intensive applications [10]. The focus of this paper is to study the implementation of the SAR algorithm on the eight-core C6678-Shannon from Texas Instruments Inc.(TI). Specifically, we show the modularization of the SAR algorithm and mapping the modules to C6678 architecture to achieve parallel computation. Further, the profiling results for key modules are demonstrated to evaluate the implementation quantitatively.

The paper is organized as follows. Sec. II briefly reviews the basic algorithms for SAR image formation. Sec. III introduces the architecture of the eight-core C6678 platform. Sec. IV-D discusses the modularization and mapping strategy of the SAR algorithm, followed by profiling results shown in Sec. V. Finally, Sec. VI concludes the paper with conclusions and discussions.

II. SAR GEOMETRY AND SIGNAL PROCESSING

SAR utilizes a single physical antenna to gather signals reflected from the targets at different positions at different times. The radar is carried by a spaceborne or an airborne

platform moving with a certain speed along a desired trajectory. The relative motion between the radar and the targets encodes the targets' information, which is processed to form a focused image of the surface area. At each radar position, the antenna system transmits a short chirped waveform. Then the reflected echoes from the earth surface are collected, digitized and stored by the antenna for later processing.

Fig. 1 shows the SAR geometry model of the radar location and the target surface. The pulse repetition time is the inverse of the Pulse Repetition Frequency (PRF). The acquisition geometry makes the SAR image processing a two dimension operation. The first dimension is called range (or cross track) that measures the "line-of-sight" distance from the radar to the target, shown as slant range along the radar sight or ground range along the ground in Fig. 1. Range resolution is determined by the transmitted pulse width, so narrow pulses yield finer range resolution. The second dimension is azimuth aligned with the relative platform velocity vector. Azimuth resolution depends on the actual radar antenna length. The angle between the slant range and the closest approach is called squint angle, as shown in Fig. 1.

For the most commonly used pulse with linear FM characteristic, the received signal from a single point target after baseband demodulation is (we follow the nomenclature and descriptions as given in [7] in this paper)

$$s_0(\tau,\eta) = w_r(\tau - 2R(\eta)/c) \times w_a(\eta - \eta_c) \times exp\{-j4\pi f_0 R(\eta)/c\} \times exp\{j\pi K_r(\tau - 2R(\eta)/c)^2\} \quad (1)$$

where the symbols are defined according to Table I. The SAR image generation corresponds to the process of focusing on each target point by weighting, phase shifting and summation the phase histories of the responses. The phase change in $exp\{-j4\pi f_0 R(\eta)/c\}$ in Eq. 1 due to radar-to-target distance variations introduces Doppler frequency which determines the target's azimuth position, while the second exponential in Eq. 1 consists of the information of the range location.

Different methods have been proposed to form a well-focused image from the raw data. This generation operation is often referred to as focusing or compression. The major two classes of algorithms are Range-Doppler (RD) algorithm and $\omega - k$ 2D algorithm.

The $\omega - k$ algorithm [5][8] uses a special operation in the 2D frequency domain to correct the range dependence and azimuth frequency dependence, relying on the assumption that the platform velocity is constant. This method is preferred to deal with data acquired over wide azimuth apertures or high squint angles but can only handle limited range swathes. The comparison of RD and $\omega - k$ algorithms can be found in [1][2]. This paper implements the RD algorithm for its high processing simplicity and efficiency.

τ	fast time along range direction
η	slow time along azimuth direction
w_r	range envelope (rectangular function)
w_a	azimuth envelope (sinc-squared function)
$R(\eta)$	instant range distance
c	speed of light
η_c	beam center crossing time
K_r	pulse chirp rate
R_0	range of the closest approach
V_r	effective radar velocity

TABLE I
DEFINITION OF SYMBOLS.

A. RD Algorithm

The RD algorithm [6][3][14] is the first developed digital SAR processor. This algorithm achieves block processing efficiency by taking the advantage of the approximate separability of processing in range and azimuth dimensions. Such separability is enabled by the large difference in time scales of the two directions and the use of Range Cell Migration Correction (RCMC). The main steps involved in the RD method are outlined in Fig. 2.

Range Compression

Range compression is to compress the received pulse along the range direction to concentrate the main energy into a narrower duration. It is performed with a fast convolution between the raw data and a reference signal in the frequency (range) - time (azimuth) domain. Therefore, FFT along the range direction is first performed, followed by matched filter multiplication and range IFFT.

RCMC

Range migration is caused by the range variations due to the platform movement. Fig. 3(a) shows the trajectories for three targets with a same closest range distance to the radar in the original time domain. Fig. 3(b) shows the corresponding trajectories in the range-Doppler domain, where the three lines collapse into one trajectory. The correction is to rearrange the data in the memory to straighten the trajectory as shown in

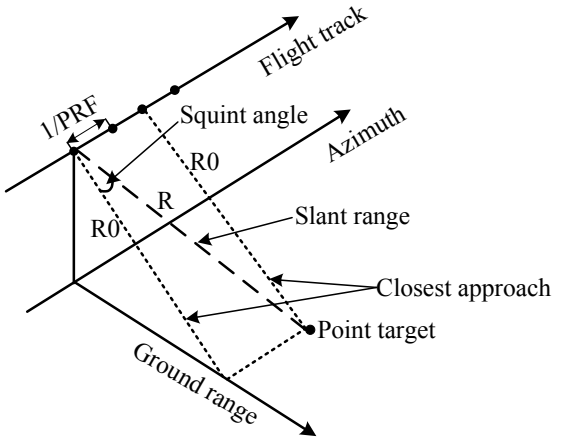

Fig. 1. SAR data acquisition geometry.

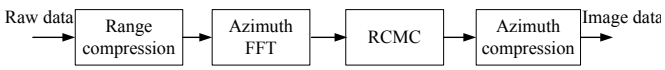

Fig. 2. Main steps in RD algorithm.

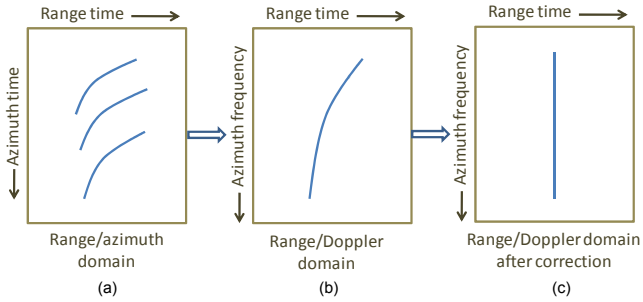

Fig. 3. RCMC for a set of target points with the same range distance.

Fig. 3(c), such that azimuth compression can be conducted along each parallel azimuth line. Note that the migration for the whole family of targets with a same range distance is corrected simultaneously in this range-Doppler domain method.

Mathematically, the amount of cell migration is given by

$$\delta R_\eta = \frac{V_r^2 \eta^2}{2R_0} \tag{2}$$

This equation calculates the displacement of each target as a function of azimuth time η and range R_0. RCMC can be achieved by a range interpolation operation based on an interpolation kernel, such as sinc function or spline [7].

Azimuth compression

Azimuth compression is to compress the spread energy in the trajectory to a single cell in the azimuth direction. This procedure is similar to range compression except that the azimuth reference function is range dependent. In other words, the azimuth reference function at each range line is different, which leads to a more complicate procedure compared to the range compression. Similar to RCMC, azimuth compression is also performed in range Doppler domain. The final image is obtained by transforming the azimuth compressed signal back to the time domain, followed by some post-processing steps.

The basic RD algorithm achieves high accuracy and can be easily implemented in a pipeline architecture. However, it needs to incorporate a secondary range compression (SRC) to handle the range-azimuth coupling problem when it comes to data with a moderate amount of squint. For simplicity, this paper focuses on the basic RD algorithm that is implemented in a multi-core DSP, as discussed in Sec. IV.

III. TMS320C6678 Architecture

The TMS320C6678 DSP is an eight core high performance DSP with both fixed and floating point capabilities [15]. The individual cores are called C66x [16]. This device can run at a core speed of up to 1.25 GHz. For our analysis here, we have analyzed a 1 GHz device which dissipates 10 W of power. Fig. 4 shows the functional block diagram of the device. It has a rich set of industry standard peripherals. The PCIe interface could be used to communicate with a CPU host. The serial rapid I/O (SRIO) running at 20 Gbps is useful to communicate among multiple DSP devices.

The C66x core is based on Very Long Instruction Word (VLIW) architecture. The instruction set also includes Single Input Multiple Data (SIMD) operating on up to 128-bit vectors. The core can support 4 single precision multiplications and additions in a single cycle. With 8 cores running at 1 GHz, the TMS320C6678 thus has the peak performance of 128 single precision GFLOPS (12.8 GFLOPS/watt).

From the memory perspective, in addition to 32KB of L1 program and data cache, c6678 integrates 512KB of dedicated memory per core that can be configured as mapped RAM or cache. The device also has 4096KB of multi-core shared memory that can be used as a shared L2 SRAM and/or shared L3 SRAM. All L2 memories incorporate error detection and error correction. The external memory is accessed via a 64 bit DDR3 interface running at 1330 MHz. The total addressable space for external memory is 8 GB with paging.

These enhancements make c6678 a suitable architecture for computationally demanding applications, such as SAR.

A. DSP Programming

TI's DSPs run a lightweight real time operating system called SYS/BIOS. Since SYS/BIOS can be used in a wide variety of processing and memory constraints, it was designed to be highly configurable. TI also provides an eclipse based Integrated Development Environment (IDE) for code development, including a C/C++ compiler. The compiler is C89 compliant and virtually every C89 compliant code can be ported with no additional effort. The compiler also allows the use of *pragma*s and intrinsic operators to fully exploit the core architecture and extract all the potential performance without resorting to assembly programming. TI provides standard libraries which contain highly tuned often used signal processing and imaging functions. The implementation in this paper has heavily used the FFT/IFFT functions supplied with TI's DSPLib.

The compiler supports openMP 3.0 [18] that allows rapid porting of existing multi-threaded codes to multi-core DSP. TI's C66x compiler translates the openMP into multi-threaded code with calls to a custom runtime library. In the evaluation section, we have used the openMP framework to instantiate individual threads across multiple cores.

IV. Modularization of RD algorithm

This section addresses the modularization of the RD algorithm, mapping it to the c6678 architecture and detailed implementation of each module. The unique features of c6678 are utilized to achieve parallel computing within each core and among multiple cores.

Fig. 5 shows the flowchart of the involved modules suited to process data with relatively small squint angles and short aperture lengths. Each block in Fig. 2 is now further broken down into smaller functional units. The following subsections describe the implementation of each module.

A. Batch Compression with DMA

The range/azimuth compression step consists of FFT on both the data and the range reference function, complex multiplication, and IFFT on the compressed data. The large

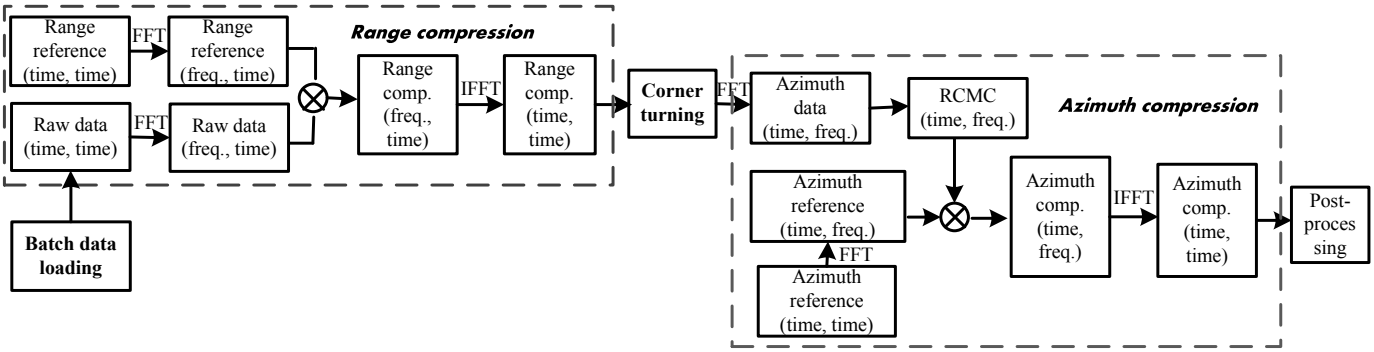

Fig. 5. Implementation flowchart of the proposed design.

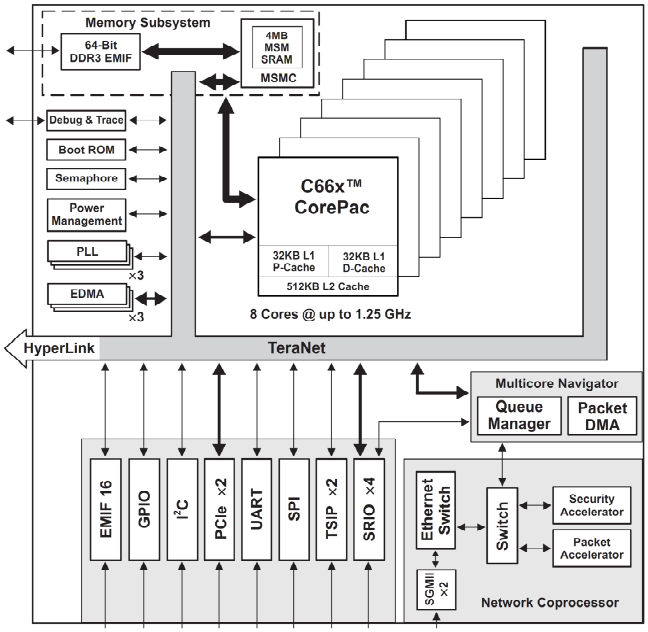

Fig. 4. Architecture of c6678.

chunk of raw data is first stored in the external DDR3 memory of c6678. To efficiently load/write the data between the external memory and internal memory (L2), the enhanced direct memory access (EDMA3) [17] is applied to service data transfer. EDMA3 is a unique design of TI's architecture. It features fully orthogonal transfer on three dimensions with synchronization on two dimensions, flexible transfer

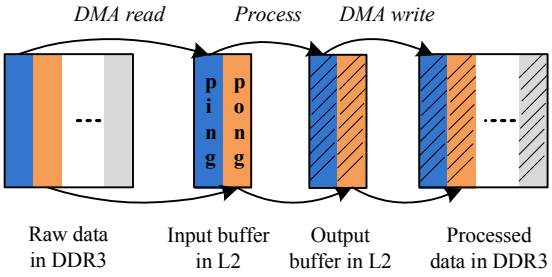

Fig. 6. Pingpong DMA reading and writing.

```
initialize DMA;
first DMA load;
for each patch data
        wait for DMA load completion;
        prepare for next DMA load;
        DMA load;
        data compression;
        wait for DMA write completion;
        prepare for next DMA write;
        DMA write;
end
wait for last DMA write completion;
close DMA;
```

Fig. 7. Pseudocode of ping-pong DMA operation and processing.

definitions, multiple DMA channels, and memory protection support. More detailed can be found in TI user manual [17].

Fig. 6 shows the ping-pong strategy for achieving parallelism among DMA loading, data processing and DMA writing steps. Two input buffers are allocated in the local L2 memory to store the latest two batches of raw data, while the two output buffers in L2 are used to hold the processed data before being written to the external memory. Data processing can take place while the DMA operation is on-going to avoid unnecessary stalling. Fig. 7 provides the pseudocode for the batch compression program with DMA operations. The reference function and twiddle factors are placed in the shared memory of c6678, so they are visible to all cores.

B. Corner Turning

Corner turning is essentially to rearranges the compressed data such that it can be read in azimuth line order for processing along the azimuth direction. Fig. 8 illustrates the process of the adopted corner turning approach. The range compressed data stored in memory cells are grouped into blocks. Squared size is preferred because DMA access achieves higher efficiency when the row size of the loading/writing data is no less than the number of rows. Block size is bounded by the

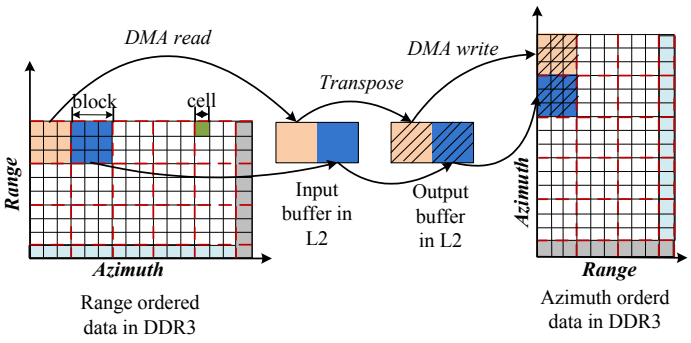

Fig. 8. Corner turning with ping-pong DMA operations.

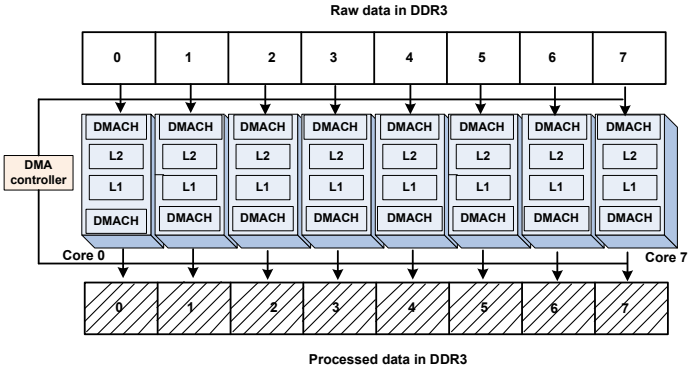

Fig. 9. Mapping on eight cores.

available L2 memory space. Similar to batch compression, the ping-pong strategy is applied on the DMA operations. The boundary cells (colored area on the right and on the bottom in Fig. 8) are processed in separate loops for transposing. An alternative strategy is to pre-process the raw data, such that the image size is of exact multiple times of block size. Then, processing on the boundary cells can be avoided to save online computation time.

C. RCMC

RCMC is carried out in the range-Doppler domain, so FFT along the azimuth direction is required before the migration correction. Data is fetched with ping-pong DMA from the DDR3 to the L2 memory. Each block corresponds to a certain number of range bins and Doppler frequency bins. Ideally, the migration amount is both range and azimuth dependent. For implementation efficiency, we assume that the migration is only azimuth dependent within each block. Across different blocks, the migration amount is both range and azimuth dependent.

The integer part of the migration amount is used for range sample shift, while the fractional part is used for interpolation. In this implementation, the 16-set 8-tap sinc filter is adopted as the interpolation filter. The coefficients of the 16 sets of filters are stored as constant number. The index of which filter should be selected is determined by the fractional part. The same filter is used for each range cell. The c6678 unique double-float load, write and arithmetic instructions are utilized to improve the interpolation computation efficiency.

D. Multi-core Mapping

The parallel implementation using the eight cores on the c6678 platform can be achieved by letting each core process a different portion of the data. Fig. 9 demonstrates how the task is assigned to each core. The data in the external memory is divided into eight portions. Each core retrieves the data according to the start point of the allocated location through DMA. Within each core, the above mentioned compression and corner turning steps can be implemented accordingly using local memory devices (L2 and L1). The access to the DDR3 memory among multiple cores is scheduled by the DMA controller.

Since the DDR3 memory space is shared fully across all cores, the above division of computation across cores can be easily achieved through openMP constructs. The openMP runtime on the multi-core DSP will run multiple threads (one thread per core) with their own allocated data portion.

V. Evaluation

In order to evaluate the performance of the SAR implementation on c6678, five key modules are profiled to estimate the execution time. Specifically, we evaluated the execution time for range compression, corner turning, azimuth FFT, RCMC and azimuth compression. Based on the available literature of similar implementations, two typical examples of image sizes of 2048 by 2048 and 4096 by 4096 are utilized for verification. This results in 2048 (4096) FFTs and IFFTs with 2048 (4096) points in each transformation. The actual FFT size varies widely for different SAR applications. Very small FFT may be necessary for auto-focusing algorithms and very large FFT size may be needed specially in azimuth direction for full resolution SAR image formation. It is also feasible to divide the large image into 2-D patches and use smaller FFT sizes with some sort of overlap-add processing.

1MB of the shared memory is set as non cached to avoid the cache incoherence problem among the eight cores. For each core, part of the L2 memory is set as L2 SRAM (384 KB) and the rest is used as cache (128 KB). Intermediate ping-pong buffers are allocated in the L2 SRAM. The batch size, i.e., the number of rows loaded from DDR3 each time (as shown in Fig. 6) is determined by the available L2 SRAM. For the 2048 and 4096 point FFT, it is set to be two and one, respectively. The squared block size during corner turning is set as 64×64.

Table II and Table III show the results of profiling the five modules with different image sizes on a TMS320C6678 based evaluation module (EVM). Results with varying number of cores being active within the device are presented in the table. The unit for measurement is millisecond. First of all, comparing the results under the two different cases, the computation time scales well with the image size. As expected, the timing required for range compression and azimuth compression scales very well with the increase of the number of operational cores. On the other hand, the corner turning

# of active cores	range comp. (ms)	corner turn (ms)	azimuth FFT (ms)	RCMC (ms)	azimuth comp. (ms)	total (ms)
1	142	17	34	46	44	283
4	36	6	9	13	11	74
8	18	6	7	12	6	50

TABLE II
EXECUTION TIME FOR KEY STEPS OF THE SAR PROCESSOR IN SINGLE CORE AND MULTI-CORE CASES, WITH IMAGE SIZE OF 2K BY 2K.

# of active cores	range comp. (ms)	corner turn (ms)	azimuth FFT (ms)	RCMC (ms)	azimuth comp. (ms)	total (ms)
1	573	100	199	269	263	1404
4	144	33	50	71	66	363
8	72	34	45	66	33	251

TABLE III
EXECUTION TIME FOR KEY STEPS OF THE SAR PROCESSOR IN SINGLE CORE AND MULTI-CORE CASES, WITH IMAGE SIZE OF 4K BY 4K.

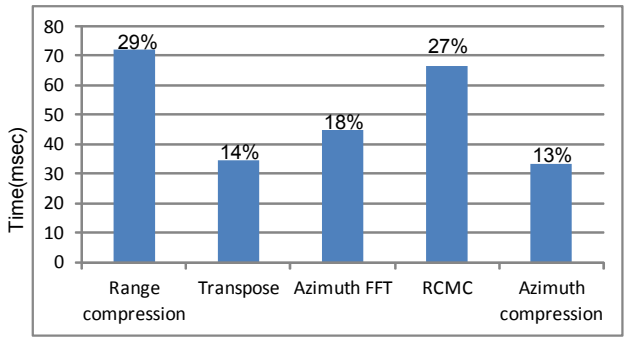

Fig. 10. Time consumption percentages for key steps with a 4k by 4k image.

timing, RCMC and azimuth FFT saturates at around 4 cores. This is due to the fact that these steps are memory I/O bound. Once it saturates the DDR3 bandwidth, increasing the number of operational cores does not help. For the total execution time, the acceleration factor with 8 cores relative to a single core is around 6.

Fig. 10 shows the percentage of required processing time for each step in the case of 4096 by 4096 data set. We can see that range compression and RCMC are the most computationally intensive steps. The sum of portions for azimuth FFT and azimuth compression is similar to that of the range compression step. Overall, it takes around 0.25 second to process the whole 4096 by 4096 image using 8 cores in parallel. Given that this processing is done with a 10 W device, the above demonstrated result makes c6678 very competitive among other alternatives, such as GPGPU, CPU and Cell, as discussed in the introduction part. The processing time for range compression, RCMC and azimuth compression Further, since the SAR processing is embarrassingly parallel, multiple devices can be employed to further improve the throughput.

VI. CONCLUSION

The performance per power efficiency of the multi-core DSP, TMX320C6678, makes it a good choice for various computationally intensive applciations. This device is widely used in various embedded applications including cellular base-stations, radio controllers and industrial/medical imaging devices. In this paper, we have presented benchmarking results of the key SAR processing modules and showed that real time SAR processing is feasible at high power efficiency using this multi-core DSP device. The scalability of SAR operations across multiple devices also suits well for DSPs to provide embedded platforms for the wide variations in SAR applications.

Our future activity includes demonstrating the full SAR imaging using one of the PCIe based boards with multiple TMS320C6678 DSPs. We plan to include parameter estimation and auto-focus algorithm to make the design higher precision and more robust. We also intend to profile compression scheme with various FFT sizes to account for different application needs of SAR processing.

REFERENCES

[1] R. Bamler, "A Systematic Comparison Of Sar Focusing Algorithms", International Geoscience and Remote Sensing Symposium, IGARSS, vol. 2, pp. 1005-1009, 1991.
[2] R. Bamler, "A Comparison of Range-Doppler and Wavenumber Domain SAR Focusing Algorithms", IEEE Transactions on Geoscience and Remote Sensing, vol. 30(4), pp. 706-713, 1992.
[3] J. R. Bennett and I. G. Cumming, "A Digital Processor for the Production of SEASAT Synthetic Aperture Radar Imagery", SURGE Work-shop, 1979.
[4] M. D. Bisceglie, M. D. Santo, C. Galdi, R. Lanari, N. Ranaldo, "Synthetic Aperture Radar Processing with GPGPU", IEEE Signal Processing Magazine, vol. 27(2), pp. 69-78, 2010.
[5] C. Caffario, C. Prati, and F. Rocca, "SAR data focusing using seismic migration techniques", IEEE Transactions on Aerospace and Electronic Systems, vol. 27(2), pp. 194-207, Mar. 1991.
[6] I. G. Cumming and J. R. Bennett, "Digital Processing of SEASAT SAR Data", IEEE International Conference on Acoustics, Speech and Signal Processing, Washington, D.C., 1979.
[7] I. G. Cumming and F. H. Wong, "Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation", Norwood, MA: Artech House, 2005.
[8] G. Franceschetti and G. Schirinzi, "A SAR processor based on two-dimensional FFT codes", IEEE Transactions on Aerospace and Electronic Systems, vol. 26(2), pp. 356-366, 1990.
[9] J. W. Goodman, "Introduction to Fourier Optics", McGraw-Hill. New York, 1968.
[10] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz. D. V. Anderson, B. L. Evans, "Trends in Multi-core DSP Platforms", IEEE Signal Processing Magazine, vol. 26(6), pp. 38-49, 2009.
[11] D. S. McFarlin, F. Franchetti, M. Pschel, and J. M. F. Moura, "High-performance synthetic aperture radar image formation on commodity multicore architectures", in Proc. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series, vol. 7337, May 2009.
[12] M. Puschel, J. M. F. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, "Spiral: Code generation for DSP transforms", Proc. IEEE, vol. 93(2), pp. 232-275, Feb. 2005.
[13] J. Rudin, "Implementation of polar format SAR image formation on the IBM cell broadband engine", in Proc. High Performance Embedded Computing(HPEC), 2007.
[14] A. M. Smith, "A New Approach to Range Doppler SAR Processing", International Journal of Remote Sensing, vol. 12(2), pp. 235-251, 1991.
[15] TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor, Data Manual. Available at : http://www.ti.com/lit/ds/sprs691c/sprs691c.pdf
[16] TMS320C66x DSP CPU and Instructions Set Reference Guide. Available at : http://www.ti.com/lit/ug/sprugh7/sprugh7.pdf
[17] Enhanced Direct Memory Access (EDMA3) Controller User Guide. Available at: http://www.ti.com/lit/ug/sprugs5a/sprugs5a.pdf
[18] OpenMP Application program Interface: version 3.0, May 2008. Available at http://www.openmp.org/mp-documents/OpenMP3.1.pdf