An FPGA-based multi-core system for synthetic aperture radar data processing

Hou Ning¹, Zhao Hongmei¹ and Zhang Duoli²

¹College of Electrical and Information Engineering, Henan University of Urban Construction, Ping Ding Shan, China
²Institute of VLSI Design, Hefei University of Technology, He Fei, China

ABSTRACT

Synthetic aperture radar, SAR, is a high resolution imaging radar that provides high quality mapping independent of light and weather conditions. The Range-Doppler algorithm is the most widely used method for digital processing of SAR. Often graphic processing units, GPGPUs are used for data processing as the Range-Doppler algorithm is computationally expensive and highly parallel. However, GPGPUs may not be an appropriate solution for applications with strictly constrained space and power requirements. In this paper, we implemented a FPGA-based multi-core system for SAR data processing with Range-Doppler algorithm. Our system consists of 4 Xilinx Virtex-6-550T devices with speed grade -1 and several off-chip memory devices. The four FPGA communicates with each other through the high-speed GTX transceivers. We designed a multi-core processor consisting of 16 processing elements and a 2D-Mesh network-on-chip on the FPGAs. Our system calculates real-time SAR raw data in pipeline, and can acquire a 256 level gray SAR images containing 2048×4096 pixels in 12.03 second at the speed of 130MHz. We measured the power of the whole system is about 85 watt.

Key words: Network on Chip; SAR; FPGA

INTRODUCTION

Synthetic aperture radar, SAR, is a form of imaging radar that provides high quality mapping independent of light and weather conditions. SAR is used across a wide range of scientific and military applications including environmental monitoring, earth-resource mapping, surveillance, and reconnaissance. The principle of SAR operation is that a radar antenna is attached to an aircraft or spacecraft. The antenna transmits electromagnetic pulses and records their echoes. An output image is reconstructed from echoed data.

One of the difficult problems with SAR is the tremendous amount of signal processing to form a final image from raw echoed data. SAR data processing is a typical compute-intensive application, contains a lot of parallel and pipeline operations, which are suitable for high-performance multi-core system. GPGPU is a typical multi-core system and often used for SAR data processing [1-2]. However, for applications with strict space and power requirements, GPGPUs may not be an appropriate solution. For example, small unmanned aircraft systems may need to compensate for deviations in the flight track but do not provide space and power for a computing system with high performance GPGPUs.

FPGA accelerated SAR data processing has been proposed previously by Cordes et al [3]. Theirs system include a host machine and an FPGA accelerator. The host machine provides the FPGA with preprocessed data and application specific information at run-time, and the FPGA is just an acceleration component. Pascal Schleuniger et al implemented a multi-core system consisting of 64 Tinuso processor cores and a high throughput, low latency network-on-chip for SAR data processing [4]. Tinuso processor is dedicated design for FPGA, the frequency of whole system is 300MHz on a Xilinx Virtex-7 device.
In this paper, we implemented a FPGA-based multi-core system for SAR data processing with Range-Doppler algorithm, RDA. Our system consists of 4 Xilinx Virtex-6-550T devices with speed grade -1 and several off-chip memory devices. The four FPGA communicates with each other through the high-speed GTX transceivers. We design a scalable multi-core processor consisting of 16 processing elements and a 2D-Mesh network-on-chip on the FPGAs. Our system calculates real-time SAR raw data in pipeline, and can acquire a 256 level gray SAR images containing 2048×4096 pixels in 12.03 second at the speed of 130MHz.

SAR GEOMETRY AND RANGE-DOPPLER ALGORITHM

We briefly review SAR geometry shown in Figure 1. The sensor moves in the direction of flight and transmits a short chirped waveform with a pulse repetition time of 1/PRF. The echoes reflected by the earth surface are received. The SAR raw data are acquired across the range swath and the azimuth swath \([5-6]\). In range dimension, the length of data received is fixed, which only depends on devices. In azimuth dimension, the data length is relative to the aperture size and the whole raw data size is proportional to the observation time. Each dimension can be processed independently, thus the SAR raw data are suitable to be performed based on multi-core system.

![Fig. 1: SAR Geometry](image)

The range-Doppler algorithm, RDA, was developed in 1976-1978 for processing SEASAT SAR data. Later it was used to digitally process space-borne SAR image in 1978 and it is still the most widely used algorithm today. Figure 2 shows the procedure of classical RDA. In general, the algorithm operates in range and azimuth frequency domain, which are mutually independent. Thus, the whole procedure includes two steps, range focusing and azimuth focusing. There are three levels of parallelism in the procedure. First, the raw SAR data are comprise of multiple apertures observed sequentially, which can be processed independently. Second, single aperture data in different range and azimuth direction can be operated in parallel for FFTs and vector calculation. Third, each pixel of the image can be achieved in parallel for interpolations. Actually, in addition to the basic algorithm procedure, the RDA also needs complete various compensation operations.

![Fig. 2: Procedure of classical range-Doppler algorithm](image)
SYSTEM ARCHITECTURE
The system architecture contains SAR signal and data processing, as shown in Figure 3. Signal processing is done partially in an analog front-end where the received echo is mixed down to base-band, IQ de-modulated, and A/D converted. The digital front-end is used for filtering and data pre-processing. Pre-processed data is then stored in DDR3 memory chip connected with FPGA 1. The Range-Doppler algorithm is mapped to a number of parallel processing elements that communicate with each other through an interconnection network. The number of the processing elements is 16, and they are distributed in four FPGAs. The SAR data processing involves a high number of integer and floating-point operations. Therefore, the processing elements are dedicated designed to speed up calculation. To support efficient communication between processing elements, we use a message passing communication scheme and design a 2D mesh interconnect. The sub-mesh communicates with each other through the GTX transceivers, which are the resources of the Xilinx FPGA and can provide a maximum link bandwidth of 10Gbit/s, nearly correspond to the link bandwidth of network-on-chip inside the FPGA. Besides, a mailbox is integrated into each processing element to accomplish data exchange and task synchronization between neighbour four processing elements in one FPGA. Four DDR3 memory controllers communicate with processing elements to meet the needs of high bandwidth and volume memory space of the RDA. Buffers in memory controllers are used to make up the speed difference between computing logic and DDR3 memory. At last, the dedicated designed transpose modules connecting with SRAM chip are integrated into our system to speed up matrix transpose operations of RDA.

Fig. 3: Block diagram of SAR signal and data processing system

PROCESSING ELEMENT
The range-Doppler algorithm is a frequency-domain algorithm, which base on the fast Fourier transform, FFT, technique. Therefore, the processing element must be able to speed up FFT operation. In addition, through the algorithm simulation with hardware features, to meet real-time SAR data processing, the processing element also need accelerate the vector operation, such as vector multiplication.

We aim for a processing element as shown in Figure 4 to achieve the real-time SAR data processing. At the center of processing element, there are three memories, which a FFT processor, a floating point unit, and a network interface can direct access. We design a parallel radix-2x2 FFT processor, which can achieve conflict-free parallel memory access [7]. According to the conflict-free memory access algorithm, two 2-radix butterflies can realize parallel accessing 4 operands, and the processing speed of FFT is improved two times. Memory 1 and 2 connecting with FFT processor are divided to into four smaller memories to realize parallel accessing 4 operands. Software achieves FFT reverse operation and allocates the location of operands in smaller memories through the switch.
We decided to use an embedded processor that features low gate count. In processing element, the processor mainly uses to achieve task scheduling and manages calculation unit. The processor mainly used for study is ARM instruction compatible, three-stage pipeline and Harvard processor architecture. To support floating point calculation, we use the Xilinx floating-point core to implement a single precision unit, FPU, with addition, subtraction, multiplication, division, square root. Implementing mathematical functions on an FPGA is a trade-off between accuracy, clock speed and utilized resources. We use a lookup-table to implement sine and Hamming window functions, which are also integrated into FPU to facilitate invoking. In addition, the FPU supports vector operation. It can pipeline loading operands from memories and storing results into another one. Operand-memory and result-memory can be selected by FPU. To save processor time, when vector operation completes, FPU informs processor by interrupt, not by halting processor instruction pipeline.

The processing element communicates with the outer world through a network interface and a mailbox. By configuring the network interface, the volume data from 2D-mesh can be directly stored to memory 0 and vice versa. The network interface also contains a small buffer for processor accessing. The four neighbor processing elements in a FPGA need cooperate to complete one sub-task. There are many synchronization of task and broadcasting of parameter between them, so the most efficiency communication scheme is mailbox [8].

![Fig. 4: Block diagram of processing element](image)

**NETWORK-ON-CHIP**

The interconnection network plays a vital role in the performance of applications with a high communication to computation ratio. Therefore, our system needs a high throughput, low latency network-on-chip to match the requirements of the SAR data processing application. Normally, the multi-core system aiming for the general application containing a lot of cache consistency and synchronization operations prefers to adopt the wormhole-switched router, which need supports virtual channel structure to try hard to avoid congestion[9-10]. However, allocating the virtual channels results in high gate count and energy consumption. When mapping SAR data processing application to multi-core system, the most of task synchronization operation is completed though Mailbox scheme and network-on-chip is mainly used for the transfers of the SAR data, which are regular and bursting. Therefore, we can accurately arrange the paths of the traffic to reduce congestion on network-on-chip.

We implement a simple wormhole-switched router with five bidirectional ports for a 2D-Mesh interconnection. The router adopts distributed architecture to avoid the speed penalty of big crossbar. In every bidirectional port, there are an input channel and an output channel, as shown in Figure 5. When flits arrive at an input channel, the input flow control, IFC, decides to buffer it or not by the write-ok signal of the input buffer, IB. The buffered header flit with four bits destination information will be decoded by the input control, IC. We decided for XY routing scheme as it is deterministic, deadlock-free and very simple to implement. Following the routing scheme, IC selects an output channel and generates the corresponding request signal. The output control, OC of the output channel checks whether the desired port is available or not according to the output flow control, OFC. If more than one request signals arrive at the same time and want to use the same output channel, the arbiter decides which is to succeed. We use a fast, fixed priority arbitration scheme where priorities are given in a clockwise manner. If the desired output channel is available and the request signal got the permission from the arbiter, the buffered flits will be sent out by the output data selection, ODS. The implemented router operates in the same clock domain as the processing elements.
APPLICATION MAPPING
The SAR data processing application mapping is an iterative process following the steps shown as in Figure 6. The iterative mapping result also influences the hardware design. We static analyze the algorithm and draw up a parallel scheme. According to Section 2, there are three levels of parallelism in the procedure of the range-Doppler algorithm. First, the raw SAR data are comprise of multiple apertures observed sequentially, which can be processed independently. Second, single aperture data in different range and azimuth direction can be operated in parallel for FFTs and vector calculation. Third, each pixel of the image can be achieved in parallel for interpolations.

Base on the parallel scheme, we rewrite the range-Doppler algorithm. The rewritten parallel algorithm defines the tasks of processing element, storage scheme of the SAR data. By simulating, we can find the critical path of the parallel algorithm and determine which path needs to be accelerated. In our system, we accelerate the FFT, vector operation and SAR data matrix transpose. Then, we can simulate the parallel algorithm again with some hardware parameters. When the simulation result meets the requirement of real-time processing, we can go to the next step to accurately simulate parallel algorithm, or else back to redraft the parallel scheme. Besides, because of lack of communication latency simulation, simulation result must contain margin.

Because the SAR data processing is extremely sophisticated, it is to waste time to accurately simulate the whole system running parallel algorithm. Therefore, we implement two accurate cycle simulator, computing simulator and traffic simulator, to accelerate simulation. The computing simulator accomplishes the accurate simulation of
processing element with the computing task allocated by the rewritten parallel algorithm. The simulator records the 
simulation time, which can be used to generate traffic task sent to traffic simulator. The Traffic simulator 
accomplishes the accurate simulation of network-on-chip and main-memory. Traffic generator, TG, contain traffic 
task is used to generate stimulus of traffic simulator. The computing and traffic simulators work together to complete 
the whole system simulation running parallel algorithm. If the simulation result meets the requirement of real-time 
processing, application mapping complete, or else back to redraft the parallel scheme.

We design computing simulator base on SWARM simulator, which is a modular simulation of an ARM 7 processor 
written in C++. SWARM is cycle accurate, and models the datapath at the bus level [11]. We extend the SWARM 
with FPU, FFT, mailbox, and interrupt controller. We test and verify the software base on the computing simulator. 
The traffic simulator is also written in C++. Traffic simulation shows that the highest traffic is in the corners of 
network-on-chip where the DDR3 memory controllers are located. Therefore, the two individual ports of DDR3 
memory controller connect to network-on-chip to avoid congestion, and every port has buffer to make up the speed 
difference between network-on-chip and DDR3 memory controller.

The application mapping result is shown in Figure 7. There are three-stage pipeline. FPGA 1 completes single 
aperture data processing as first stage. After 16 times aperture data processing, FPGA 2 and FPGA 3 work together 
to complete the major synthetic aperture data processing as second stage. At last, FPGA 4 completes the rest 
operations and output 256 level gray SAR image. The main memories buffer processing result of the last stage. The 
every stage is parallel processed by multiple processing elements.

**RESULTS**

Figure 8 is the photo of the implemented SAR data processing board. The board mainly contains four Xilinx 
Virtex6-550T FPGAs with speed grade -1, totally 48Gb DDR3 memory chips and 288Mb SARM chips, for redundancy design. The communication between FPGAs realizes by high-speed GTX transceiver. The fabric utilizes about 67% of the hardware resources of the FPGA, and work at the speed of 130MHz. We measure the power of the whole system is about 85 watt.

We evaluate the proposed FPGA-based multi-core system by running the SAR data processing application. The quality of the results and the speed of imaging are mainly considered. To get a credible conclusion, we continuously process 8 SAR images with 256 level gray and 2048x4096 pixels, respectively using CPU platform and processing board, and obtain the average time as final results. The CPU platform is Intel Core i7-930 processor at 2.8 GHz, 8
GB RAM on 64-bit Windows 7 OS. We calculate the Peak Signal to Noise Ratio, PSNR, and mean square error, MSE, to compare SAR imaging results, which is illustrated in Table 1. The quality of images obtained under two different platforms is almost identical.

<table>
<thead>
<tr>
<th>Image number</th>
<th>MSE</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.410</td>
<td>42.8</td>
</tr>
<tr>
<td>2</td>
<td>4.676</td>
<td>41.5</td>
</tr>
<tr>
<td>3</td>
<td>5.763</td>
<td>40.6</td>
</tr>
<tr>
<td>4</td>
<td>8.496</td>
<td>38.9</td>
</tr>
<tr>
<td>5</td>
<td>9.395</td>
<td>38.4</td>
</tr>
<tr>
<td>6</td>
<td>8.544</td>
<td>38.8</td>
</tr>
<tr>
<td>7</td>
<td>10.395</td>
<td>38.0</td>
</tr>
<tr>
<td>8</td>
<td>9.544</td>
<td>38.4</td>
</tr>
</tbody>
</table>

Our system calculates SAR raw data in pipeline. Table 2 shows processing time of three stages. Our system can acquire a 256 level gray SAR images containing 2048×4096 pixels in 12.03 second at the speed of 130MHz. Compared with CPU platform, which complete a SAR image in 189.34 second, we obtain a speedup of 15.74.

<table>
<thead>
<tr>
<th>Pipeline Stage</th>
<th>Devices</th>
<th>Processing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FPGA 1</td>
<td>12.03 second</td>
</tr>
<tr>
<td>2</td>
<td>FPGA 2 and FPGA 3</td>
<td>10.35 second</td>
</tr>
<tr>
<td>3</td>
<td>FPGA 4</td>
<td>7.480 second</td>
</tr>
</tbody>
</table>

CONCLUSION

We implemented a FPGA-based multi-core system for SAR data processing with Range-Doppler algorithm. Our system consists of 4 Xilinx Virtex-6-550T devices with speed grade -1 and several off-chip memory devices. We design a scalable multi-core processor consisting of 16 processing elements and a high throughput, low latency network-on-chip. To speed up computing, the processing element contains a parallel radix-2×2 FFT processor and a FPU supporting vector operation. The hardware resource of each FPGA is 67%. The whole system works at the speed of 130MHz and consumes about 85 watt. We described how to map a complex application like the SAR data processing application on a multi-core system. We static analyze the algorithm, draw up a parallel scheme, rewrite the algorithm and define features of the hardware. The rewritten parallel algorithm provides the stimulus of the simulator to acquire the more accurate system performance. Besides, aiming for the sophisticated multi-core system, the simulator is divided into computing simulator and traffic simulator to save simulation time. Finally, our system calculates real-time SAR raw data in pipeline, and can acquire a 256 level gray SAR images containing 2048×4096 pixels in 12.03 second. We measure the power of the whole system is about 85 watt. Therefore, we conclude that multi-core systems on FPGA are an attractive choice for application with strictly constrained space and power budgets.

Acknowledgments
The Project Supported by Institute 14 of China Electronics Technology Group Corporation and National Natural Science Foundation of Chinas for contract61179036.

REFERENCES

2010, IEEE CONFERENCE PUBLICATIONS, 310-313.