IOP Conf. Series: Materials Science and Engineering **719** (2020) 012031 doi:10.1088/1757-899X/719/1/012031

Implementation of SAR Echo Simulation Algorithm on Heterogeneous Embedded Computing Platform

Xiaoyu Hou[1], Daying Quan[1], Wei Fan[1], Xiaoping Jin[1] and Hengliang Liu[2]

[1]Key Laboratory of Electromagnetic Wave Information Technology and Metrology of Zhejiang Province, College of Information Engineering, China Jiliang University, Hangzhou, China
[2]Hangzhou Jianpu Information Co. Ltd., Hangzhou, China
Email: 862376302@qq.com; qdy@cjlu.edu.cn; l312361206@163.com

Abstract. The Synthetic Aperture Radar (SAR) echo simulation is to obtain the SAR original echo signal by performing reverse operation on the pre-set SAR image. The traditional platforms used to implement the simulation algorithms like central processing unit (CPU) + graphic processing unit (GPU) and digital signal processor (DSP) + field programmable gate array (FPGA) have disadvantages such as high power consumption or complicated programming. In order to make up for these shortcomings, the implementation structure of the SAR echo simulation algorithm was improved to be applicable on the heterogeneous embedded platform with the basic SAR simulation algorithms digitized and decomposed. Base on the improved structure, a mobile GPU based heterogeneous computing platform with one multiprocessor system-on-chip (MPSoC) and multiple GPUs was designed to implement SAR echo simulation algorithm. The platform can simultaneously utilize the real-time nature of register transfer level (RTL) design and the ease of programming on GPU. It can achieve relatively faster computing power at lower power consumption and has the characteristics of miniaturization and mobility.

1. Introduction

SAR is a kind of imaging radar used for remote sensing to create two-dimensional or three-dimensional images of scene objects. Its high resolution, strong transmissivity and long detection range make it highly military and civil value. However, the acquisition of the real SAR echo data requires the support of airborne or spaceborne radar, which requires high cost so that it is necessary to develop a SAR echo simulator.

When the traditional platform performs SAR echo simulation algorithm based on general-purpose computer, it takes too much time to simulate the scene with large amount of data, so it is difficult to realize real-time echo simulation. Most computing platforms that can be used for real-time echo simulation are based on FPGAs and DSPs. As a programmable device, FPGA has a wealth of logical computing resources and has the advantage of flexible configuration. A large number of independent multiply operations in echo simulation can be implemented in a highly parallel manner by means of hardware resources such as dedicated multipliers in the FPGA. Wang et al [1] designed a platform of four FPGAs to realize the generation of SAR echo signals and optimized the implementation of the slant range calculation, which is the most computationally intensive part of the entire algorithm. Although implementing the algorithm with FPGA can achieve its real-time parallel computing ability and flexibility, it has the disadvantages of large programming workload and high logic design difficulty. Chen [2] used DSP and FPGA to realize the fast simulation of radar echo. The DSP chip is responsible for the process of range direction information integration. The FPGA chip mainly

completes the process of slant range calculation, FFT, complex multiplication and IFFT. The cooperation between different computing units is realized by the heterogeneous form, but the multi-core parallel computing capability of the DSP chip is insufficient, so there is still room for improvement in the process of range direction information integration.

The rapid development of GPU provides a promising and efficient computing platform for SAR echo simulation. Li [3] realized SAR echo simulation algorithm on the GPU, but only stayed on a single GPU for optimization. Jing et al [4] implemented SAR echo simulation algorithm on dual GPUs. The SAR echo simulation using GPU platforms described above is still based on a desktop or server platform and has the problems of large volume and high power consumption, which is not suitable for scenarios with high mobility requirements. Therefore, it is necessary to design a SAR echo simulation platform that combines high performance, low power consumption and small size for these scenarios. On the other hand, with the demand of machine learning and deep learning, embedded GPU processors have been greatly developed, such as NVIDIA's Tegra series SoC.

In order to further improve the integration and mobility of the SAR echo simulators, this paper proposed an embedded heterogeneous computing platform based on one MPSoC and multiple embedded GPU processors. By arranging different computing tasks on different processors, the real-time SAR echo simulation can be realized on the embedded platform with high integration, small size and low power consumption.

2. SAR Echo Simulation Algorithm and it's Embedded Design

2.1. SAR Echo Simulation Algorithm

The most commonly used detection pulse in SAR is the linear frequency modulation (LFM) pulse, which can be written as

$$s_r(\tau) = \omega_r(\tau)\exp(j2\pi f_0\tau + \pi K_r\tau^2) \tag{1}$$

where τ is the time variable in seconds, K_r is the LFM rate in hertz per second, f_0 is the carrier frequency in hertz and $\omega_r(\tau)$ is the window in time domain or the pulse envelope. Suppose a single point target is located in the beam footprint, the baseband echo of the single point target can be written as [5]

$$s_0(\tau,\eta) = A_0\ \omega_r(\tau - 2R(\eta)/c)\ \omega_a(\eta-\eta_c)\exp\{-j4\pi f_0 R(\eta)/c\}\exp\{j\pi K_r(\tau-2R(\eta)/c)^2\} \tag{2}$$

where $A_0 = A_0^{'}\exp(j\psi)$ is the complex backscatter coefficient, τ and η are the fast time variable and slow time variable, respectively, in seconds, $\omega_a(\eta)$ is the antenna pattern, c is the speed of light in meter per second, and $R(\eta)$ is the slant range of the point target in meter.

In a more general scenario, there should be continuous targets or scatter points which make up the whole surface target reference scene in the beam footprint. Thus the backscatter coefficient becomes to be a three dimensional function with two dimension according to the scatter surface and the other in the slow time η, and we call it as range direction scatter function (RDSF). With this function denoted as $g(\tau,\eta) = g'(\tau,\eta)\exp\{j\psi(\tau,\eta)\}$, the baseband echo of the surface targets can be written as

$$s_{bb} = g(\tau,\eta)\otimes h(\tau,\eta) \tag{3}$$

where $h(\tau,\eta)$ is the impulse response of the unified single target and it is formulated as [5]

$$h(\tau,\eta) = \omega_r(\tau - 2R(\eta)/c)\ \omega_a(\eta-\eta_c)\exp\{-j4\pi f_0 R(\eta)/c\}\exp\{j\pi K_r(\tau-2R(\eta)/c)^2\} \tag{4}$$

It is difficult to directly implement equation (3) and equation (4) since they are the continuous form. Another problem in implementing equation (3) and equation (4) is that the parameters in them are coupled. For example, both $g(\tau,\eta)$ and $R(\eta)$ is not only space variant but also slow time variant.

In order to simplify the problem, for η can be seen as a discrete variable, which indicates the sampling time in the azimuth direction, equation (3) and equation (4) can be rewritten as [5]

$$\begin{aligned} h(\tau,\eta_i) &= \omega_r(\tau - 2R(\eta_i)/c)\,\omega_a(\eta_i-\eta_c) \\ &\quad \times \exp\{-j4\pi f_0 R(\eta_i)/c\}\exp\{j\pi K_r(\tau-2R(\eta_i)/c)^2\},\ i=0,1,...,I-1 \end{aligned} \tag{5}$$

$$\begin{aligned} s_{bb} &= g(\tau,\eta_i)\otimes h(\tau,\eta_i), \\ &= (g(\tau,\eta_i)\,\omega_a(\eta_i-\eta_c)\exp\{-j4\pi f_0 R(\eta_i)/c\}) \\ &\quad \otimes \omega_r(\tau-2R(\eta_i)/c)\exp\{j\pi K_r(\tau-2R(\eta_i)/c)^2\} \\ &= g_t(\tau,\eta_i)\otimes\omega_r(\tau-2R(\eta_i)/c)\exp\{j\pi K_r(\tau-2R(\eta_i)/c)^2,\ i=0,1,...,I-1 \end{aligned} \tag{6}$$

Where

$$g_t(\tau,\eta_i) = g(\tau,\eta_i)\,\omega_a(\eta_i-\eta_c)\exp\{-j4\pi f_0 R(\eta_i)/c\} \tag{7}$$

With a dedicated i, equation (6) can be regarded to execute a two dimensional convolution for there is a two dimensional scatter surface. Moreover equation (5) represents the i-th sampling along the slow time and meanwhile it is the i-th range direction echo.

There are three important implications in equation (3)-(6):

(1) With different slow time (azimuth) sampling point, equation (5) and equation (6) are independent. That means although slow time sampling η_i is sequencing in the real world, it doesn't matter that the k-th sample is calculated earlier than the l-th one, even if $k > l$. It is also doesn't matter that they are calculated at the same time, thus range direction echo can be calculated in parallel.

(2) Although $g_t(\tau,\eta_i), i=0,1,...,I-1$ in equation (7) is two dimensional in the scatter surface for a dedicated η_i, dimension reduction could be employed via a digitization of equation (6) and equation (7) in order to ease the implementation.

(3) In equation (6), the convolution calculation requires massive calculation resources.

As mentioned above in implication (2), the discrete sampling in range (resulting in range cells) and discrete scatter surface are introduced so that RDSF in equation (7) can be approximately expressed as

$$g_t(\tau_{n\_rc},\eta_i) = \sum_{n\_rc} g(\tau_{n\_rc},\eta_i)\,\omega_a(\eta_i-\eta_c)\exp\{-j4\pi f_0 R_{rc}(\eta_i)/c\} \tag{8}$$

where n_rc denotes the index of the scatter points located in the same range cell. According to the calculation of the RDSF in equation (8), all of the scatter points in the whole two dimensional scatter surface should be traversed and there are 4 steps of calculation for each dedicated azimuth sample:

(1) for every scatter point, the slant range is calculated and then it is approximate to a proper range cell,

(2) for every scatter point, the phase modulation term $\exp\{-j4\pi f_0 R_{rc}(\eta_i)/c$ is calculated,

(3) for every scatter point, the backscatter coefficient $g(\tau_{n\_rc},\eta_i)$ and the antenna pattern modulation $\omega_a(\eta_i-\eta_c)$ are multiplied, and finally,

(4) for every range cell, all scatter points located in are integrated.

For the 4 steps above to calculate the RDSF in equation (8), the range cells and scatter points are located in the range-azimuth two dimensional surface, and it doesn't matter which range cell or which scatter point is calculated first. That means they can be calculated simultaneously, and the only requirement is that equation (8) is calculated before equation (6).

As summarized, to calculate equation (3)-(8) an embedded architecture can be proposed.

2.2. Embedded Design of SAR Echo Simulation Algorithm

The embedded architecture of SAR echo simulation algorithm is presented in Figure 1. For the parallel nature of calculation along the azimuth samples, the whole task is firstly partitioned into sub routine for several azimuth sample subsets, which could be parallel assigned to separate processing units, thus all the range direction echoes of these subsets are computed in parallel. Then inside the procedure in every separate processing unit, there is a repetition to traverse all azimuth points in the corresponding subsets. Meanwhile there are two kinds of different requirements on calculation in these procedures: one is RDSF calculation which requires parallel calculation along all scatter points; the other is the convolution of scatter function and the detection pulse which requires massive and real-time computing.

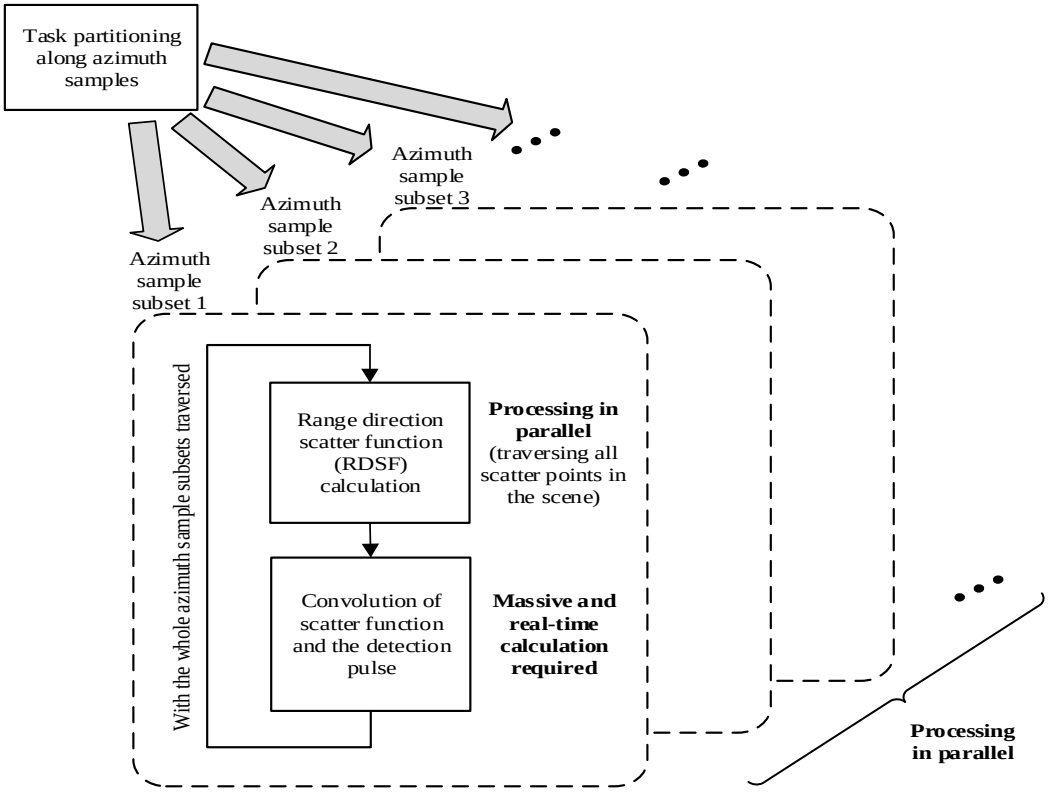

Figure 1. Embedded architecture of SAR echo simulation algorithm

With the different requirements of the calculation tasks depicted in Figure 1, different kinds of processing units could be properly adapted. For task partitioning, small amount of computing but relatively complex controlling is required, so an ARM processor may be suitable to be employed. For RDSF calculation, because the two-dimensional massive scatter points need to be processed in parallel, GPUs are suitable to be used. And finally, due to massive and real-time calculation of convolution, FPGAs are most suitable for it.

3. Prototype Design of Heterogeneous SAR Echo Simulator

3.1. Hardware Design

Base on the architecture depicted in Figure 1 and the analysis presented above, a heterogeneous computing platform featuring in one MPSoC and four GPUs is proposed in Figure 2. The heterogeneous platform is mainly composed of an MPSoC as the central control node, GPU computing nodes, switching nodes and a storage system. GPU computing nodes and switching nodes are connected through PCIe interface. Switching nodes and the central control node are connected through SRIO interface.

The central control node uses the XILINX's Zynq-7000 series MPSoC. GPU computing nodes use the NVIDIA's JETSON Tegra X2 modules. Switching nodes use the XC7K70T from XILINX's Kintex-7 series FPGA, which is used to implement the communication interconnection between the central control node and all GPU computing nodes. The fabricated platform prototype is shown in Figure **3**.

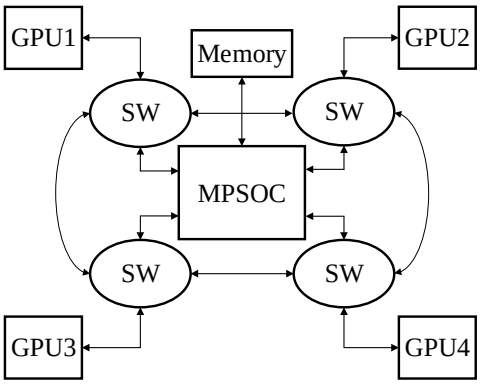

Figure 2. MPSoC+4 GPUs platform structure

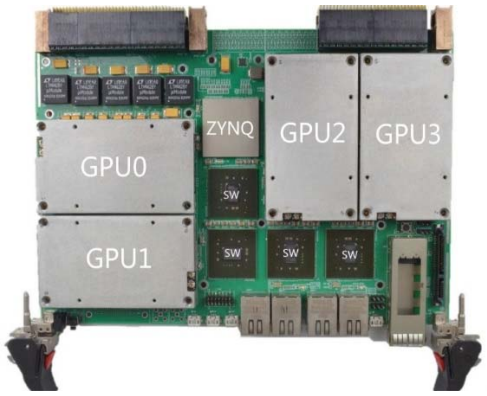

Figure 3. The fabricated simulator prototype

3.2. Software Design

3.2.1. Software architecture. Porting the architecture in Figure 1 into hardware prototype shown in Figure 2 and Figure ***3***, the proposed software architecture for implementation is presented in Figure 4. PS of Zynq SoC is used to partition calculation tasks for different azimuth sample subsets and assign them to GPUs. GPUs are employed to calculate the RDSF. Finally PL of Zynq SoC is mainly responsible for convolution operations.

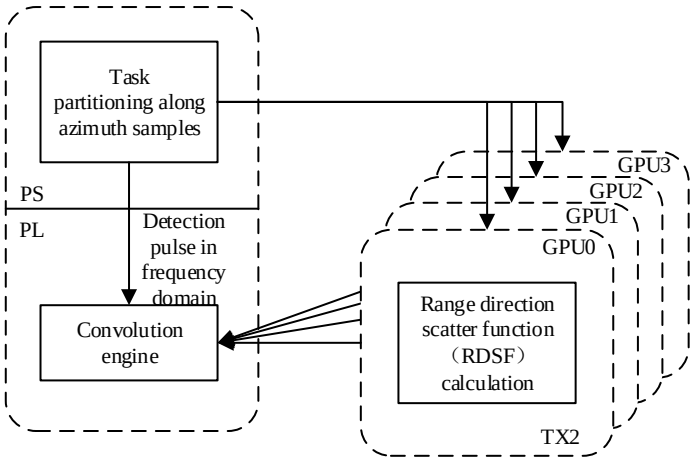

Figure 4. Software architecture of SAR echo simulation algorithm

3.2.2. Zynq SoC software. When PS of Zynq SoC initializes the tasks and scene data, it sends the data to GPU0, GPU1, GPU2 and GPU3, so that they can calculate the point target echoes at the sampling time of 4N, 4N+1, 4N+2 and 4N+3 respectively. Thus the azimuth samples are equally partitioned into four subsets and an average payload along the four GPUs are achieved.

When GPUs send the RDSF according to an azimuth sampling time to PL of Zynq SoC, the PL adopts a convolution engine to perform the convolution in frequency domain, with the detection pulse in frequency domain preset by PS. The convolution in frequency domain performs the procedure of FFT on RDSF, multiplication of detection pulse in frequency domain and finally IFFT operation, which decrease the computation complexity. PL mainly implements the convolution engine and

multiplexes along all azimuth samples, so the workload of RTL programming is highly reduced. When PL of Zynq SoC completes convolution, the final result is summarized on PS of Zynq SoC.

3.2.3. GPU software. RDSF in equation (8) is calculated by a four-step procedure as discussed in chapter 2.1, which can be divided into two stages with high parallelism respectively. Two kernel functions are opened for these two stages in each GPU. In the first kernel function, each point target corresponds to one thread. In each thread, the calculation of slant range, range cell, phase modulation term, backscatter coefficient and antenna pattern modulation are implemented. In the second kernel function, different threads in the same block correspond to the product term in Equation (8) of different point targets in the same range cell. In each block, the accumulation of RDSF is implemented by using the reduction algorithm. For a dedicated azimuth sample, the procedure to calculate RDSF is presented in Figure 5.

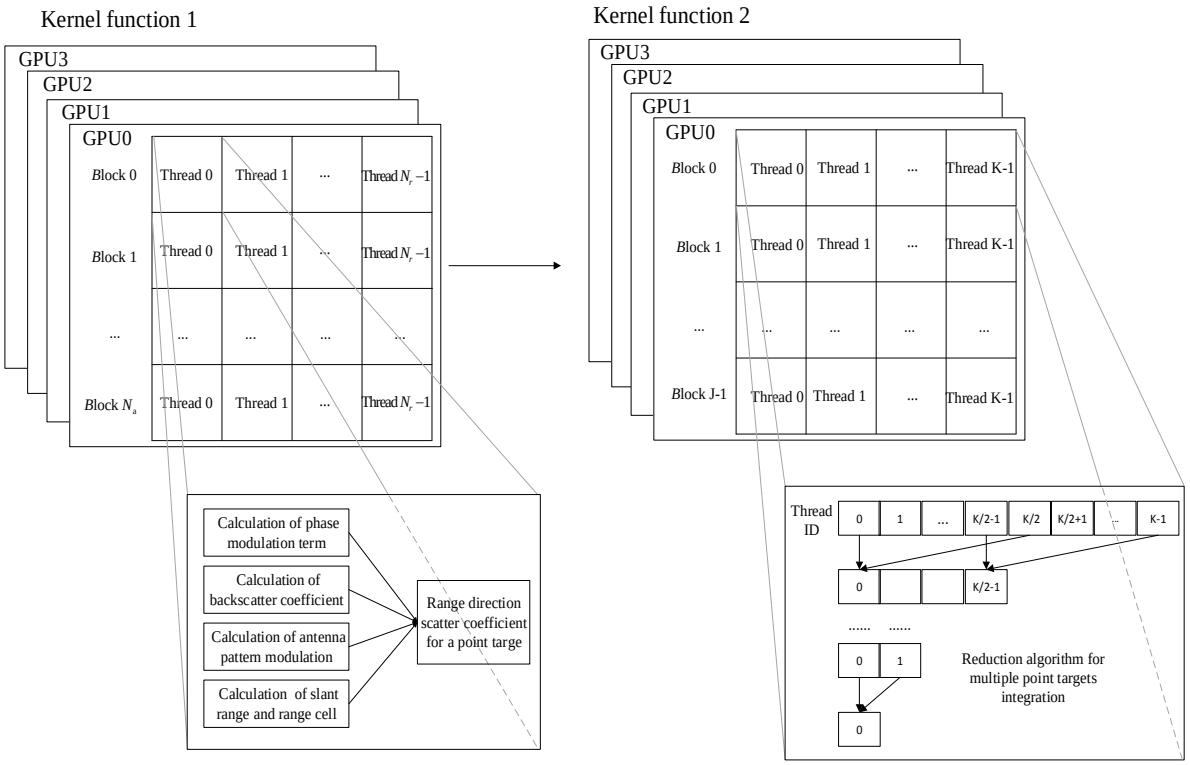

Figure 5. GPU kernel function operation diagram

In the first kernel function, it is assumed that the size of surface target reference scene is $N_a \times N_r$ pixels, where N_a is the sample number of azimuth direction and N_r is the sample number of range direction. At each sampling instant, $N_a \times N_r$ threads are opened, where N_a is the number of blocks and N_r is the number of threads in each block. In each thread, slant range, phase and range cell of the corresponding point target are calculated.

In the second kernel function, it is assumed that there are J range cells in the result of the former and there are $K_1, K_2, ..., K_J$ points in different range cells. Suppose that $K >= \max\{K_1, K_2, \ldots, K_J\}$ and K is integer powers of 2. $J \times K$ threads are opened. When the number of points in a range cell is less than K, the value in the corresponding thread is assigned to 0. Then, the accumulation of product term in Equation (8) in the same range cell is realized by reduction algorithm. In the reduction algorithm, at the first step, the algorithm simultaneously implements the addition of the 0th and the

$(K/2)$ th, the 1st and the $(K/2+1)$ th, ..., thus obtaining $K/2$ addition results. After $\log_2 K$ steps, we get the RDSF of the corresponding rang cell.

4. Test Results and Analysis

In the echo simulation experiment, the simulation platform parameters are shown in Table 1. The echo simulation of surface target reference scene used in this paper is shown in Figure 6 and its size is 1024 × 1024 pixels. The echo simulation is performed by the heterogeneous computing platform and then the generated echo data is imaged by the Rang-Doppler algorithm, which is shown in Figure **7**.

Table 1. Simulation platform parameters.

Parameter	Value
CUDA version	9.0
CUDA core number	256*4
Memory size	8GB
Memory bandwidth	58.3GB/S

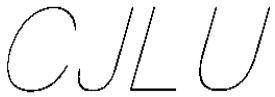

Figure 6. Original image of the simulated surface target **Figure 7.** The simulated SAR image

In order to evaluate the performance of the architecture designed in this paper, the paper compares the time consumption and typical power consumption with literature [1], [3] and [6]. The test results are shown in Table 2. Compared with the traditional CPU+GPU architecture, the simulator proposed in this paper not only takes less time but also consumes less power. The time consumption of literature [3] is 114 times that of this paper, while the power consumption is 1.5 times that of this paper. The time consumption of literature [6] is 44 times that of this paper, while the power consumption is 6.8 times that of this paper. Compared with FPGA architecture, there is still a certain gap in computing performance, but the GPU modules used in this paper are low-power products and totally the proposed simulator prototype typically consumes less power.

Table 2. Comparison to results of different platforms.

Platform architecture	Main device	Main device	Time consuming (ms)	Typical power consumption(W)
Zynq SoC+GPU (This paper)	XC7Z100 + 4*Tegra X2	1024*1024	3.87	45
CPU+GPU[3]	GeForce GT620M	1024*1338	441.6	70
CPU+GPU[6]	Tesla K20C	1024*1024	172	310
FPGA[1]	4* StratixII	1000*1000	2.0	80

The detailed time-consuming decomposition of the simulator prototype proposed in this paper is shown in Table 3. It can be seen that it takes most of the time for data transmission, which means wider bandwidth of data interaction among multiple devices is required.

Table 3. Time-consuming composition.

Component	Time consuming(ms)
Calculation	0.73
Data transmission	3.14
Total	3.87

5. Conclusion

This paper digitized and decomposed the basic SAR simulation algorithm, proposed an embedded implementation structure, and designed a heterogeneous computing platform based on one MPSoC and multiple GPUs to implement the proposed algorithm. Compared with traditional platforms, the proposed platform features in higher integration, smaller size, better mobility, and higher energy efficiency.

6. Acknowledgements

This research was supported in part by Natural Science Foundation of Zhejiang Province, China, under Grants No. LY17F010012.

7. References

[1] Hongxian Wang, Yinghui Quan, Mengdao Xing and et al, 2010 Fast Implementation Method of SAR Echo Simulation Based on FPGA *Systems Engineering and Electronics* **32** 2284-89
[2] Lulu Chen, 2014 SAR Echo Signal Simulation Fast Parallel Algorithm and Engineering Implementation Method *Xidian University*
[3] Bo Li 2014 Research and Implementation of GPU-based SAR Echo Signal Simulation *Xidian University*
[4] Guobin Jing, Yunji Zhang and Zhenyu Li, 2016 Efficient Implementation Method of GPU-based SAR Echo Simulation *Systems Engineering and Electronics* **38** 2493-98
[5] Cumming, Ian G. and Frank H. Wong 2005 *Digital processing of synthetic aperture radar data: Algorithms and implementation* (Boston: Artech House) pp 141-147
[6] Yunji Zhang, 2015 GPU-based SAR Echo Simulation and Imaging Method *Xidian University*