High Performance Implementation of Ultrasound Color Doppler Imaging on GPU platform

Thi Yen Phuong, Jeong-Gun Lee
Department of Computer Engineering, Hallym University, Korea
{phuongyen, jeonggun.lee}@hallym.ac.kr

Abstract. The ability to detect and assess information of blood flow in color Doppler imaging (CDI) has played an important role in a modern ultrasound imaging system. However, it has been mainly implemented on custom-designed hardware due to large amount of data and computations. Recent trend of programmable approach offers the advantages of flexibility and quick implementation. For best exploiting outstanding computational power, high memory bandwidth and SIMD architecture of a GPU, this paper presents a high performance implementation of CDI on the GPU platform using CUDA API. The performance analysis shows our GPU-based CDI can achieve a frame rate of 152 for 800 range samples and 200 scan lines with an ensemble size of 12. Speedup of 19.8x can be obtained when compared with that on a CPU platform.

Keywords: color Doppler imaging; autocorrelation, GPU; parallel processing;

1 Introduction

High computing power and computational demands of ultrasound imaging makes its implementation on hardware such as ASICs or FPGA complicated, time-consuming and costly. Reported as excellent accelerators, GPUs can become a promising solution for moving from hardware-based to software-based ultrasound imaging system. The main purpose of this study is to demonstrate the use of GPU on implementation of CDI under CUDA environment.

This paper is organized as follows: section 2 describes the implementation of CDI on GPU, section 3 discusses performance results and analysis and section 4 concludes the paper.

2 CDI Implementation on GPU

Fig. 1 shows the block diagram of a simplified ultrasound system. This paper focuses on implementation of real-time color Doppler mode based on autocorrelation technique in which the phase shift from a fixed-depth is used for velocity estimation.
Baseband I/Q data is stored as an array of $I/Q[N_S][N_L][N_E]$, where $N_S$ is the number of range samples in a scan line, $N_L$ is the number of scan lines and $N_E$ is ensemble size. Although data transfer from CPU to GPU takes a certain amount of time, the overall processing speed achieved by GPU still higher than that of CPU. The main processing steps of CDI are wall filtering and velocity estimation.

**Wall filtering.** Since real and imaginary data processing are independent, clutter filtering is divided into two kernels: `filter_real()` and `filter_img()` by taking advantage of concurrent stream. Each thread in the kernel produces filtered real or imaginary data after performing convolution operation with a 4-tap IIR filter. Read-only filter coefficients are kept in constant memory. The baseband I/Q array index in global memory is arranged in such a way that it exposes coalesced memory access pattern.

**Velocity Estimation.** Each thread in `veloc_est()` kernel calculates Doppler frequency shift, velocity and variance for each sample and each scan line. More computation is involved in this kernel, which results in better latency hiding. To maximize instruction throughput, the kernel uses arithmetic instructions with high throughput, including trading precision for speed when it does not affect the end result (`-use_fast_math` functions), using single precision instead of double-precision.

### 3 Performance Results

In this paper, CDI algorithm is implemented under Windows 8 64-bit operating system and CUDA driver 5.0 running on the hardware platform of Intel Core i7-3770 CPU at 3.40GHz. Two GPUs are used: NVIDIA GeForce 660 OEM Kepler architecture with 1152 CUDA cores; NVIDIA GeForce 560 Ti Fermi architecture with 384 CUDA cores.

Fig. 2 shows the performance comparison between GPU-based and CPU-based implementation with various data sizes. Though Kepler 660 architecture has triple more number of cores than Fermi 560 Ti, the application actually runs faster on Fermi. One of the reasons is that Fermi platform used in this implementation has GPU clock speed of 1.8GHz while Kepler 660 clock speed is only 0.89GHz (double slower).
Moreover, with important factors that directly affect memory-bound kernels such as larger memory bus width, bigger L2 cache size and higher memory bandwidth, Fermi 560 Ti outperforms Kepler 660.

![CDI performance on different platforms](image)

Fig. 2. CDI performance on different platforms (in milisecond).

For 800x200x12 volume size, speedup achieved by running the application on GPU is up to 19.8x compared to CPU in GTX 560 Ti case and 7.5x in GTX 660 case. For the same volume size, a frame rate of 152 can be achieved on Fermi 560 Ti architecture.

4 Conclusion

The paper presents an implementation of Color Doppler Imaging on GPU platform using CUDA parallel programming. Our experiment results show that by employing various optimization strategies, the best speedup of a GPU-based CDI system achieved is up to 19.8x over CPU for a volume data of 800x200x12. GPU framework provides advantages of flexibility, programmability and easy-to-use design languages while meeting the real-time requirement of current as well as future expectation of CDI application.

References