Design an accelerator for Computing MRI-Q Matrix

COMS E6868 - Embedded Scalable Platforms - Spring 2020

Pei Liu
pl2748@columbia.edu

ABSTRACT

I plan to design an accelerator to calculate the Magnetic Resonance Imaging Q matrix through SystemC and HLS tool Stratus and implement this design on FPGA. Then I plan to explore the design space to get a thorough understanding of the methodology of designing accelerators though ESP.

1. INTRODUCTION

Magnetic Resonance Imaging is commonly used by medical community to safely and non-invasively probe the structure and function of human bodies. Images generated using MRI have a profound impact both in clinical and research fields. MRI has scan phase (data acquisition) and image reconstruction phase. Short scan time can increase scanner throughput and reduce patient discomfort, which tends to mitigate motion-related artifacts. High resolution of the image is preferable. Short scan time and high resolution conflict with each other if sampling with the Cartesian trajectory in k-space on a uniform grid, which allows image reconstruction to be performed quickly and efficiently. However, the reconstruction of non-Cartesian trajectory sampling data is faster and less sensitive to imaging artifacts caused by non-Cartesian trajectories, but it increases computation significantly [1]. The computation for MRI-Q matrix is an important algorithm used for image reconstruction with non-Cartesian trajectory sampling [2].

1.1 Motivation

Heterogeneous systems architecture is the future trend. Hardware specialization can bring order-of-magnitude more energy efficiency.Designing an accelerator for computing MRI-Q matrix through ESP is a good way to learn ESP design methodology.

2. SPECIFICATION

The algorithm for computing MRI-Q matrix is shown in Fig. 1. We want to accelerate the whole computation, mainly by loop unrolling and pipelining the inner for-loop. The programmer view algorithm C code and input dataset come from the Parboil benchmark suit [3]. In this project, we will design accelerators which can accomodate the three datasets provided by Parboil benchmark, and make it capable of dealing with input images with arbitray size.

2.1 Assessment

We aim to design accelerators which can implement Q-matrix computation for any arbitrary input image sizes. The first goal is correctness. The Q-matrix computed by our accelerator should match the Q-matrix computed by software program. We calculate the difference between every output and its golden output, and deem it as a match when the difference is less than a certain threshold, otherwise it is an error. And we also set an error_rate. If it is within a specified small value, we deem that our accelerator meet the correctness goal. The second goal is performance. We will measure the acceleration of our accelerator compared to its software execution running on FPGA board. We want our accelerators can be integrated with both Ariane core and Leon3 core. The generated RTL through Stratus HLS can be integrated and prototyped on FPGA. We will design both baremetal application and Linux application, the accelerator should pass both tests. At last we want to collect speedup data which indicates the acceleration effect. In achieving the second goal, we will do some amount of design space exploration, which includes designing accelerators with different area and latency trade-offs.

```
for (m = 0; m < M; m++) {
  phiMag[m] = rPhi[m]*rPhi[m] +
              iPhi[m]*iPhi[m];
}

for (n = 0; n < 8*N; n++) {
  for (m = 0; m < M; m++) {
    exp = 2*PI*(kx[m] * x[n] +
                ky[m] * y[n] +
                kz[m] * z[n]);
    rQ[n] += phiMag[m]*cos(exp);
    iQ[n] += phiMag[m]*sin(exp);
  }
}
```

Figure 1: Algorithm for computing MRI Q matrix [1]

2.2 Milestones

1. Analysis of the algorithm and the programmer's implementation in C (by Feb. 19)
2. Learning two tutorials: "How to design an accelerator in SystemC (Cadence Stratus HLS)" and "How to design a single-core SoC"[4]. (by Feb. 28)
3. High-Level-Synthesis implementation in SystemC (by Mar. 11)
   - Transform programmer view algorithm to HLS-ready SystemC
   - Behavioral simulation
4. Baremetal application and linux application design. Evaluation on an FPGA platform (by Mar. 25)
5. Mid-term presentation and report (by Mar. 25)
6. Initial Design Space Exploration (by Apr. 15)
7. Enhanced Design Space Exploration (by Apr. 22)
8. Final refinement and analysis (by May 5)

9. Design the same accelerator with Vivado HLS flow.
10. Final presentation ($\sim$ May 11) and report ($\sim$ May 15)

2.3 Critical Aspects

1. Implement sine and cosine functions in for-loop.
2. Try different methods of optimization to reduce latency or decrease area.

3. REFERENCES

[1] Sam S Stone, Justin P Haldar, Stephanie C Tsao, BP Sutton, Z-P Liang, et al. Accelerating advanced MRI reconstructions on GPUs. *Journal of parallel and distributed computing*, 68(10):1307–1318, 2008.

[2] John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and Wen-mei W Hwu. Parboil: A revised benchmark suite for scientific and commercial throughput computing. *Center for Reliable and High-Performance Computing*, 127, 2012.

[3] The impact research group. Parboil benchmarks. Available at `http://impact.crhc.illinois.edu/parboil/parboil.aspx`.

[4] sld-esp Columbia. How to design a signal core SoC. Available at `https://www.esp.cs.columbia.edu/docs/singlecore/`.