A RISC-V-based, Multi-threaded General Purpose GPU Core

Tamer Eren[1*], Mehmet Eyyüp Ergin[1], Ömer Güzel[1], and Hasan Erdem Yantır[1]

[1]The Scientific And Technological Research Council Of Türkiye (TÜBİTAK), BİLGEM

Abstract

Graphics processing units (GPUs) are specialized processors commonly used in computing devices for high quality visual displays and parallel computations. GPUs offer several advantages over CPUs, including the ability to handle multiple tasks simultaneously and efficiently process complex data. The demand for GPUs has grown in recent years due to advancements in areas such as artificial intelligence, crypto mining and gaming. Addressing the sufficient GPU problem in market, more flexible and cost-effective solutions designed with open-source methods may provide an alternative in both academic and industrial applications. In Tübitak BİLGEM a multi-core multi-thread GPU architecture is being designed with IMF extensions of RISC-V. First prototype was constructed with a single core (RISC-V IMF) and 5 threads which can be adjusted parametrically by the user. The initial design was synthesized with TSMC 28nm technology at 2GHz with Synopsys Design Compiler. Finally, functionality demonstration was performed with Digilent Nexys A7 FPGA at 100 MHz to create the Tübitak Bilgem logo by calculating triangle points and resterazing with specific colors.

Introduction

A graphics processing unit (GPU) is a specialized type of processor or video card that is commonly installed in computing devices to provide high-quality visual display. Basically, there are two main goals of using GPU in a system. The first one is completing "general purpose" calculations which belong to machine learning, block-chain, artificial intelligence, mining, applications of biophysics, and other disciplines that require rapid and challenging computations [1]. The other goal is processing dots, triangles, matrices, and pixels to form a picture and maintaining this process until obtain a meaningful visual output. Actually all of these tasks could be performed with CPUs, but there are certain advantages of GPUs among other type of processors. CPUs and GPUs differ in their processing capabilities; while CPUs execute tasks one at a time, GPUs can handle multiple tasks simultaneously by dividing them up into smaller pieces and processing them in parallel using numerous cores which allowing them to efficiently process complex data and perform intensive computations that would be impractical for a CPU to handle alone [2]. As an alternative to the graphics cards that are frequently used in the industry today, more flexible and cost-effective solutions, which are designed entirely with open-source methods can provide a solution in both academic and industrial applications. In order make graphics processing unit architectures more accessible and easier to understand, some studies have been initiated within Integrated Circuits Design and Training Laboratory (TUTEL) in Tübitak BİLGEM. The General Purpose Graphics Processing Unit (GP-GPU) Core prototype, has been developed using the I, M, and F extensions of the RISC-V ISA, and adopts a multi-thread structure.

[*] ***Corresponding author*** : tamer.eren@tubitak.gov.tr

Methodologies

The most significant purpose of the GPU cores is increasing the throughput by processing a large amount of data that arrives simultaneously. To obtain multiple data outputs in a single clock cycle, data handling must be paralleled at multiple levels. Almost every single GPU cores utilize a parallel computing language like CUDA or OpenCL as an initial step of this parallelization, which divides the instructions sent by the CPU into kernels as a workload. Then, each work package is distributed to multiple threads of corresponding core. In literature, there are two fundamental approaches for the thread mechanism. The first one is, fetching the same instructions for each thread but processing them on multiple data [3]. However, the other one is distributing different instructions to different threads [4]. The first method increases the throughput of the cores, but the second one can complete different programs simultaneously. Since data parallelization was aimed for the first prototype of the proposed GPU core, it was deemed appropriate to divide different data into the threads which runs same instruction. For the final step of the parallelization, a complete vector processor unit (VPU), which executes the instructions of the V extension of RISC-V ISA will be utilized the future works. In summary, parallel processing of incoming data will be achieved at three fundamental levels, which are core, thread, and VPU. The compiler's abilities are also essential for converting written code into RISC-V instructions that can be executed in parallel by applying the required optimizations. If the required transformations are not achieved, OpenCL instructions may not be able to work simultaneously with sufficient efficiency. The C function given below, increments the variables a, b, and c by one at each step, and repeats this process 250 times. Since the processing of these variables is not dependent on each

```
 void loop_250( ){
 int a = 0, b = 0, c = 0;
 for (int i = 0; i < 250; i++){
   a++;  b++; c++;  }
}
```

other, the compiler can divide this process into 3 different loops, and different cores or threads can work on these values independently. In other words, even in the compiler step, total run time reduces significantly without any further parallelization technique.

Discussion

Architecture

The core structure is built to meet the requirements of in-order execution and consists of a 5-stage pipeline that includes Fetch, Decode, Issue, Execute, and Writeback operations. In the execute stage, there is an ALU for integer operations, multiplication and division. Overall core design was prepared in System Verilog and the initial prototype was combined with a 2-level Cache hierarchy.

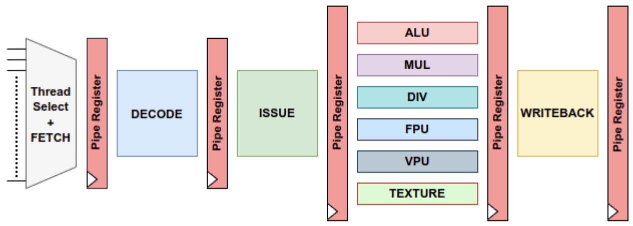

Figure 1: *Block Diagram of the proposed GP-GPU core*

Figure 1 demonstrates the block diagram of the proposed GP-GPU core's architectures. In order to fit the design to Digilent Nexys A7 FPGA, single core and 5-thread configuration was synthesized by using 22032 LUTs, and 80 RAMB36E1 type Block RAMs.

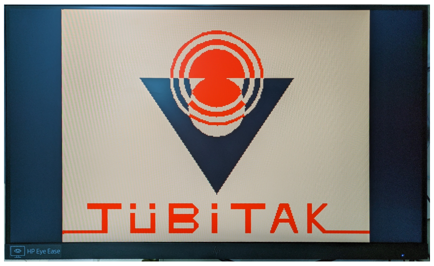

Figure 2: *Tübitak Bilgem Logo*

Total on chip power was obtained as 0.47 W at 100 MHz operating frequency. To observe the success of the architecture and texture unit, visualization of the Tübitak Bilgem logo was performed and the result is presented in Figure 2. When an image needs to be printed on the screen, the points of the triangles that make up this picture can be loaded directly to the system (LSU) or calculated by the GPU internally. After loading the point, triangles processed by the texture unit using an 8-bit RGB color code for triangle rasterization, and placement stages. Furthermore, VPU is partially integrated for the initial version of the design, and functional tests were completed for the applied parts. Moreover, the GPGPU core utilizes custom instructions designed for multi-core, multi-thread, and texture operations to expand the RISC-V ISA, and accelerate the operations along the pipeline, particularly in the execution stage. For example, when the texture mode is activated, a custom instruction with three source register addresses is designed to quickly read the triangle corner values from the source registers, allowing for more operations to be completed in fewer cycles.
After synthesizing the Core in FPGA, in order to have an idea about the performance in ASIC level, the proposed model (a single RISC-V based GPU core) was synthesized with TSMC 28nm technology at 2 GHz using Synopsys' Design Compiler and IC Compiler tools.

Future Works

In a multi-core configuration, different cores will become responsible for different parts of the screen, which will lead to both increased resolution and speed. Moreover, parallel data operations such as matrix multiplication becomes faster when multiple cores were utilized. For the later versions of the core, RTL prototyping, synthesis, and PnR with 16 threads and 256 cores are aimed to be completed.

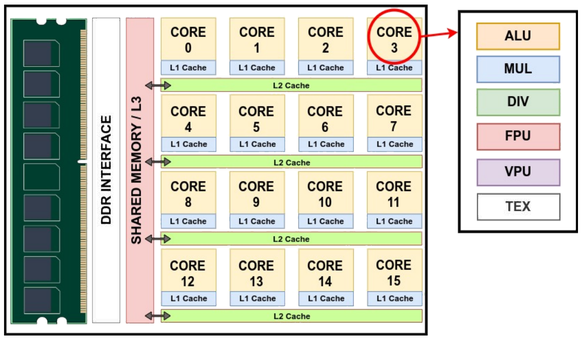

Figure 3: *Block level diagram of the multi-core version*

Furthermore, while the single-core version includes a DDR connection, the aim is to preserve and upgrade this interface for the multi-core configuration. Figure 3 shows the block diagram of the next version which will utilize 16 cores and DDR connection. Finally, the cores and threads inside the GPU will be activated and deactivated depending on the processing load by the central processing unit and OpenCL, which indirectly controls also the enabling power consumption of the overall design.

References

[1] J. Peddie, The history of the GPU - steps to invention. Springer International Publishing AG, 2023. pp. 333-345.

[2] S. Asano, T. Maruyama and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 2009, pp. 126-131, doi: 10.1109/FPL.2009.5272532.

[3] A. Alawneh, M. Khairy and T. G. Rogers, "A SIMT Analyzer for Multi-Threaded CPU Applications," 2022 IEEE, ISPASS Singapore, 2022, pp. 248-250, doi: 10.1109/ISPASS55109.2022.00037.

[4] Y. Yang and H. Zhou, "Cuda-NP," ACM SIGPLAN Notices, vol. 49, no. 8, pp. 93–106, 2014. doi.org/10.1145/2692916.2555254