Does the Raspberry Pi 4B Have a GPGPU?

Jaidon Lybbert, University of Washington

January 24, 2024

Abstract

The Raspberry Pi 4B uses the Broadcomm BCM2711 SoC, which features a quad-core ARM Cortex-A72 CPU, and a Broadcomm VideoCore VI GPU. The VideoCore VI GPU is purpose-built for graphics applications, and there is very limited support for directly programming the device from the manufacturer. The focus of this article is to investigate the capabilities of the VideoCore GPU for General Purpose GPU (GPGPU) tasks, that is, compute tasks not involving graphics. Several open-source methods of programming are evaluated, and performance of the GPU is benchmarked against the CPU for the General Matrix Multiply (GEMM) algorithm extensively used in machine learning and machine vision applications.

1 Introduction

This document is organized as follows. In Section 2 I describe my experience with embedded systems and software, motivations and intent. I describe the setup of my environment in Section 3 and perform some benchmarking tests. I show the benchmark results in Section 4, and discuss the results along with the failures and hurdles I faced in Section 5. I conclude in 6 by discussing the improvements I would like to make in the future.

2 Background

2.1 GPUs and GPGPUs

As clock rates stopped increasing, and Moore's law began to taper off, parallel processers emerged to increase compute throughput by allowing some algorithms to be broken down into pieces that could be solved simeoultaneously. Fueled by the `$300B` gaming industry, parallel processors have rapidly taken over for video-encoding and decoding in computers and mobile phones. Nearly anything displaying to a screen now has a graphics processor. The emergence of AI training and inference has created new demand for General Purpose GPUs (GPGPUs) which are not restricted to graphics processing, but can be easily programmed for other compute tasks. NVIDIAs CUDA platform is built for this market, making programming their family of GPUs easy by providing a language, compiler, and debugging tools that integrate with existing languages and tools like C++, and Visual Studio.

Machine vision algorithms like Simultaneous Localization And Mapping (SLAM), and other applications, demand portable GPGPUs which can be programmed with arbitrary code and be integrated into a small portable device to perform parallel processing. The NVIDIA Jetson family of devices is tailored to this market, offering scaled down chips, and development boards for embedded programmers as low as `$130`.

In this article, we will look at the GPU on the Raspberry Pi. A device which, as we'll see, was clearly not intended to be used as a GPGPU. However, recent developments in software drivers and graphics APIs just might enable this device to massively speed up some compute workloads.

2.2 VideoCore IV and VI

The Broadcomm chips in the Raspberry Pis have VideoCore GPUs. The Raspberry Pi 1, 2, and 3 have the VideoCore IV GPU, and the Raspberry Pi 4 has the VideoCore VI GPU. The VideoCore IV is also found in devices such as the Roku, Amazon Fire TV Stick, and phones from Samsung, Nokia, and Apple's iPod.

We will focus on the BCM2711 SoC featured in the Raspberry Pi 4B, which has a VideoCore VI GPU.

2.3 Setting the Baseline

To evaluate the performance of the VideoCore VI GPU, there are two useful comparisons:

- How does it perform compared to the CPU on-chip?
- How does it perform compared to an NVIDIA Jetson?

The first question is arguably more relevant to someone who already has a Raspberry Pi. After all, why go through the trouble if the CPU is faster anyway?

The second question is really asking, is it ever a good idea to buy a Raspberry Pi for GPGPU-related tasks?

2.4 Motivation

Two big questions motivate my research: "what is the performance of the VideoCore VI GPU?" and "how do I program it?"

This small list of large questions breaks down into a large list of smaller questions.

- What is the compute throughput of the VideoCore VI GPU?
- How do instructions map to hardware?
- How many parallel threads does the GPU support?
- What other APIs exist for GPU programming?
- Previous work done and results
- What is the memory model for the GPU + CPU?
- CPU and GPU caching?
- Supported language bindings?
- Memory latency?

2.5 Prior Works

Broadcomm has not released much documentation for the VideoCore VI GPU. Much of the information about the GPU is derived from the previous VideoCore IV reference manual [1], which Broadcomm did release, and open-source drivers written with the help of Broadcomm engineers in the Linux kernel repository [10], and the MESA graphics library [3].

Various projects over the years have attempted to make programming the GPUs easier on the Raspberry Pi.

The GitHub user "hermanhermitage" investigated the VideoCore IV GPU, and created a wiki their findings [2]. They used US patent applications filed by Broadcomm along with experimentation to characterize the architecture, memory layout, instruction set, and registers for the device. The project includes a guide on intercepting GPU execution by replacing the `bootcode.bin` in the boot partition of the SD card. Broadcomm eventually published a full specification for the VideoCore IV, superceding this effort with official documentation.

QPULib is a C++ library by Matthew Naylor for compiling and offloading programs for the VideoCore IV GPU (Raspberry Pi versions 1, 2, 3) during runtime [6]. Efforts by Wim Rijnders extended this idea to the VideoCore VI in the Raspberry Pi 4, with the C++ library V3DLib [9]. V3DLib depends on the Mesa graphics library for disassembly of VideoCore instructions. It's worth noting that neither of these projects have had active development in the last 3 years, and support for new OS versions is unknown.

A totally separate project, `py-videocore6` [7], is a Python library enabling the programming of the VideoCore VI through the Direct Rendering Manager (DRM) graphics driver in the Linux kernel.

The Raspberry Pi Foundation recently announced that the Raspberry Pi 4 is now Vulkan 1.2 compliant through the Mesa graphics driver. This was a joint effort by Raspberry Pi, Mesa, and Broadcomm developers. Vulkan is a graphics library, which in contrast to older graphics libraries like OpenGL ES, supports an efficient compute-only pipeline, without any involvement of the framebuffer. Vulkan has been getting used more recently in machine learning applications for its cross-platform support. However, the Vulkan 1.2 specification does allow many features to be optional, so it isn't clear to what extent compute can be done through Vulkan until it is installed and queried for the supported features.

3 Methods

In order to make any meaningful observations about the capability of the Broadcomm VideoCore VI GPU on the Raspberry Pi 4B, I will compare the programming experience and capabilities of the device to the widely used NVIDIA CUDA platform. Obviously, with the NVIDIA chips being orders of magnitude more expensive, they have many more cores and features, but on a core-to-core basis they are similar SIMD processors with similar function. For comparisons sake, I will show how the `$35` 2GB Raspberry Pi 4B **theoretically** stacks against the `$600` 11GB NVIDIA GTX 2080 TI graphics card at scale in terms of power consumption, compute throughput, and memory bandwidth.

3.1 CUDA Overview

Through my graduate research, I studied parallel computing using the NVIDIA Cuda toolkit on NVIDIA GPU architectures. The Turing Streaming Multiprocessor (SM) architecture featured in the RTX 20xx family of GPUs features Tensor Core hardware units for performing tensor calculations using reduced precision data types including IEEE-754 FP16, and unsigned 8-bit integers. Each SM can perform 1024 FP16 fused multiply accumulate (FMA) operations per clock cycle, or 2048 INT8 FMA operations. With 68 SMs on a consumer-grade chip (RTX 2080 Ti), that means 69k FMA operations per clock cycle, giving it an FP16 compute throughput of

3.2 CPU GMEM Benchmarks

As a benchmark, I chose to use a naive matrix multiply algorithm, as it has the potential to be memory-bound or compute-bound, and is easy to implement and analyze. The algorithmic complexity is O(nˆ3). For the basic CPU version, I use Bing AI to generate the algorithm in Rust which iterates over many matrix sizes, and records their execution times to a `.csv` file. I then use Bing AI to generate a Python function to plot the results from the `.csv` file.

4 Results

4.1 Cortex-A72 Characterization

The ARM Cortex-A72 CPU is thoroughly documented by ARM, with a public reference manual [4]. Details specific to the Raspberry Pi 4B are published in the Raspberry Pi documentation [8]. The relevant high-level details are summarized here.

Quad-core Cortex-A72 64-bit @ 1.5 GHz

4.1.1 Memory

- 4GB LPDDR4-2400 SDRAM
- 32kB data + 48kB instruction L1 cache per core
- 1MB L2 cache

4.1.2 CPU Architecture

The Cortex-A72 CPU in the Raspberry Pi 4B has 4 cores, each with 32kB data cache and 48kB instruction cache. The cores share a 1MB L2 cache.

4.1.3 Throughput

4.2 VideoCore VI Characterization

VideoCore VI 32-bit @ 500 MHz. MMU.

4.3 Programming

4.3.1 V3DLib

V3DLib is a language and compiler for the VideoCore VI GPU created by Wim Rjinder's in C++. The project has a large number of dependencies, and ultimately uses the Mesa graphics library to translate code to run on the device. Because of the large number of dependencies and stale development (hasn't been updated in 3 years), it may be prohibitively difficult to use this method of programming.

I cloned the repository and ran the build script.

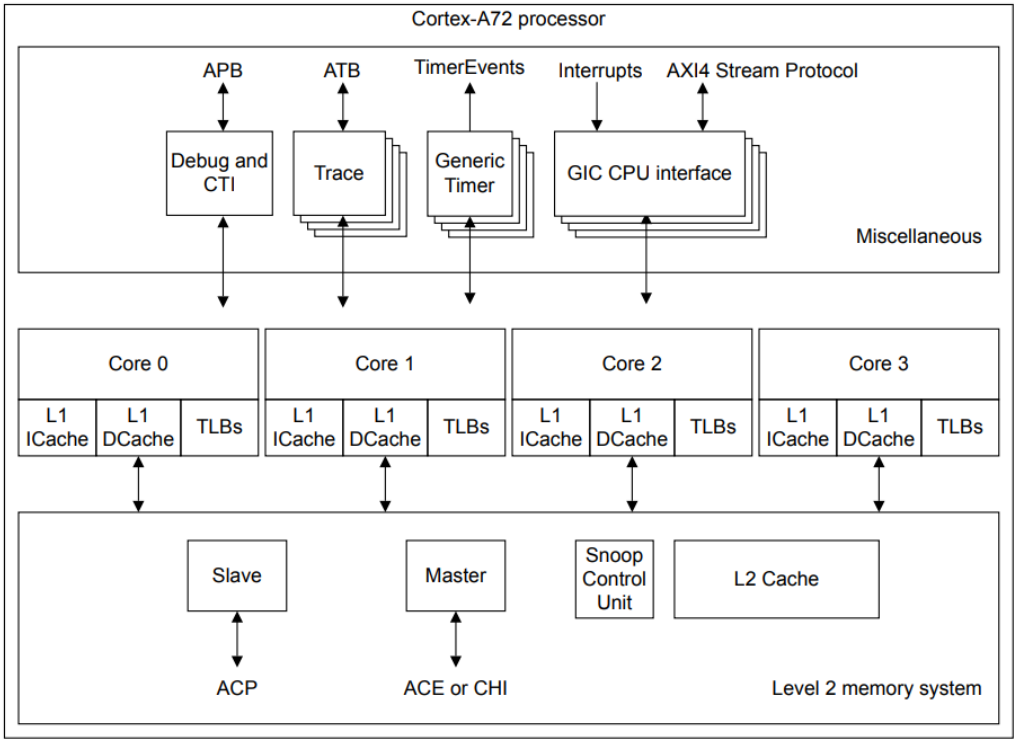

Figure 1: Block diagram of the quad-core ARM Cortex-A72 CPU.

```
> sudo apt-get install git
> sudo apt install libexpat1-dev
> git clone --depth 1 https://github.com/wimrijnders
   /V3DLib.git
> cd V3DLib
> make QPU=1 DEBUG=1 all
```

The build failed due to missing headers, which I found were included in the raspberry pi development headers

4.3.2 pyvideocore6

4.3.3 Vulkan 1.2

5 Discussion

6 Conclusion

Note *All my code and project files for this and future reports, can be found on my GitHub repository [5].*

References

[1] Broadcom Corporation. Videocore® iv 3d architecture reference guide. Technical Report VideoCoreIV-AG100-R, Broadcom Corporation, 2013.

[2] hermanhermitage. videocoreiv: Tools and information for the broadcom videocore iv (raspberrypi). `https://github.com/hermanhermitage/videocoreiv`, 2024. Accessed: 2024-02-05.

[3] intel. external-mesa/src/gallium/drivers/v3d at master · intel/external-mesa, 2024. Accessed: 2024-01-30.

[4] ARM Limited. Arm cortex-a72 mpcore processor technical reference manual. Technical report, ARM Limited, 2021.

[5] Jaidon Lybbert. Classwork for embedded and real-time systems, university of washington, winter 2024. `https://github.com/jaidonlybbert/EEP522A-W24`, 2024. Accessed: 2024-01-23.

[6] Matthew Naylor. Qpulib: A programming language and compiler for the raspberry pi's quad processing units, 2021. Accessed: 2024-01-30.

[7] nineties. py-videocore6: A python library for gpgpu programming on raspberry pi 4. `https://github.com/nineties/py-videocore6`, 2021. Accessed: 2024-01-30.

[8] Raspberry Pi. Processors, 2024. Accessed: 2024-02-07.

[9] Wim Rijnders. V3dlib: A c++ library for creating programs to run on the videocore gpu's of all versions of the raspberry pi. `https://github.com/wimrijnders/V3DLib`, 2021. Accessed: 2024-01-30.

[10] Linus Torvalds. linux/drivers/gpu/drm/v3d, 2024. Accessed: 2024-01-30.