A RISC-V ISA Extension to Accelerate the Rendering of 3D Graphics – Cyrus Goodarzi FYP Proposal

Aim of the project

The aim of the project would be to create an extension to the RISC-V ISA which would allow for the hardware acceleration of graphics rendering.
Before multicore general purpose GPUs become common, GPUs were designed to accelerate rendering of computer graphics, taking a 3D scene made of vertices and lines as an input and providing rendered frames as an output.

The graphics pipeline can be split into 5 broad stages:

1. Vertex Data (input)
2. Vertex processing - Transformations, Lighting, Interpolation
3. Rasterization – mapping of 3D scene to 2D screen
4. Fragment processing - Lighting, Texture operations, Filtering
5. Frame Buffer (output)

The minimum viable product for this project would be to add a single instruction that would accelerate rasterization in hardware. This is because rasterization forms a fundamental part of the 3D graphics pipeline and one that can benefit greatly from a fixed function accelerator.
This is highly extensible since other parts of the graphics pipeline can be added as additional instructions such as adding an instruction to do texture mapping or vertex transformations.
The project can be considered a success if chosen benchmarks are able to run on the design and produce correct results.

Why do this project

Currently, an open-source graphical instruction extension available for RISC-V doesn't exist. FGPU [1] [2] is a soft GPU designed as a standalone system, this means that if it were to interface with a RISC-V soft CPU it may suffer from data transfer bottlenecks that are common to this kind of system, the GPU would also have to be accessed using drivers of some sort making the creation and use of such a system more complex.

The Pixilica group [3] aims to implement an ISA extension for RISC-V for general purpose compute. However, other than their initial document explaining the ISA and their system architecture they provide no other information or technical specifications on the project.
While an ISA extension for general purpose compute is very flexible, it comes with increased area, and power costs.

The fixed function and more specialised nature of my project will allow the user of my system to render graphics and output frames with lower area utilisation and power consumption and potentially greater performance for those selected instructions. The addition of a common GPU feature as an ISA extension also means drivers for my accelerator do not need to be developed.

Skills Used

What courses does this bring together:

- Computer Architecture and Advanced computer architecture
- Digital System Design
- Computer Graphics
- Language processors

It will also use most if not all of skills I learnt during my placement.

Challenges with this project that I will not have encountered before (new skills to learn):

- Hardware implementation of graphical operations
- Learning about new graphics programming APIs
- Most complex hardware system I will have ever built on my own

Apparatus

- FPGA
- FPGA Design and Verification CAD tools
- Open-source compiler, RISC-V core, benchmarks
- Access to MATLAB or NumPy for modelling

Method

1. Find benchmarks that can be used for evaluation of the system
    a. Should use the whole graphics pipeline so I don't have to find new benchmarks for each new feature I want to add
2. Choose systems I wish to compare my design to:
    a. Configurable RISC-V core (one which I am planning to extend with my instr.)
    b. RISC-V core with vector extension (same configurable core as above)
    c. Commercial CPU, other non-RISC soft CPUs
    d. Commercial GPU, soft GPU
3. Implement those benchmarks on the above systems
    a. Find an open-source RISC-V ISA that can be extended with new instructions
        i. Use this to compile the benchmarks for a fair test between the RISC-V cores
4. Measure performance (cycles or real time) and power of benchmarks – 1
5. Based on the above measurements, create performance and power specifications for system
6. Specification of instruction format and behaviour
    a. Use benchmarks to decide on the above
7. Modelling and Validation
    a. Model the behaviour of the instruction in a high-level language like MATLAB so architectural features like data types can be designed more easily
    b. Allows for precise definition of functional requirements of the block
    c. Allows for model checking during top level verification of the block
8. System Design and verification
    a. Modify RISC-V soft core to implement this new instruction in RTL
        i. What parts of the architecture will need changing
        ii. Microarchitectural design
        iii. Implementation in RTL
        iv. RTL simulation
        v. Unit level testing and subsystem testing
        vi. FPGA implementation
        vii. Top level testing
            1. Implement benchmark with new instruction and run on system
    b. Verification
        i. Likely do directed testing, maybe do constrained random if time?
        ii. Each level of testing should have a well thought out test plan
9. Evaluation
    a. Take power and performance measurements on design while running benchmarks as in step 3
    b. Compare my design to currently available software implementations from step 2
    c. The project can be considered a success if the benchmarks are able to run on the design and produce correct results.
    d. If power and performance specifications are met this is a bonus.

e. If not, a discussion about optimisation can be had

Rough timetable and Intermediate milestones

Task	Time needed (weeks)	End date
Basic Background research	2	11 Nov
Inception Report		11 Nov (6 weeks between this report and interim report)
More Background research	continuous until the next report	2 Jan
Find benchmarks that can be used for evaluation of the system	1	18 Nov
Choose systems I wish to compare my design to	2	2 Dec
Implement those benchmarks on the above systems	2	16 Dec
Measure performance (cycles or real time) and power of benchmarks	1	23 Dec
Interim Report		2 Jan (20 weeks between this report and first draft final report)
Create performance and power specifications for system	2	16 Jan
Specification of instruction format and behaviour	2	30 Jan
Modelling and Validation	4	27 Feb
System Design and verification	10	15 May
Evaluation	1	22 May
Abstract and Draft Report		31 May
Final Report		16 June
Presentations		21 June

Green rows represent project deliverables, others represent milestones based on the method section.

NB: The steps between the interim report and the first draft of the final report can be repeated for any additional feature I wish to add. I would not need to repeat any of the steps before January as long as the new instruction accelerates part of the benchmarks. This makes it critical to find benchmarks which use the entire graphics pipeline.

References

[1] M. Al Kadi, B. Janssen and M. Huebner, "FGPU: An SIMT-Architecture for FPGAs," *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays,* pp. 254-263, 2016.

[2] M. Al Kadi, B. Janssen and M. Huebner, "FGPU," 1 July 2018. [Online]. Available: https://github.com/malkadi/FGPU.

[3] A. Zafar, W. Radochonski and N. Siret, "RISC-V Graphics," 19 October 2021. [Online]. Available: https://www.pixilica.com/graphics.