Raspberry PI GPU Auto-Overclocker

Development Project Proposal

*Sajeeb Roy Chowdhury, Caleb Maranian

Colorado State University

Fall 2019

Project Objective:

In this project we propose to develop an automatic GPU (Graphics Processing Unit) overclocker using a raspberry pi. The purpose of overclocking is to get more out of a system, or a piece of hardware, than what was initially paid for. Often GPU manufacturers such as NVIDIA will release a chip with a certain clock speed, and then release a slightly modified version of that chip with slightly higher clock speeds with a higher price tag. Since people do not want to purchase the more expensive chip, they buy the lower end chip and overclock it to as close as possible to the next tier chip. This process of overclocking is tedious for humans as it is an iterative process. The process of overclocking starts by first maxing out the power limit of the chip, and then iteratively incrementing the clock speed and stress testing the chip after every increment to ensure stability. To ensure the max overclock is reached, the increment is usually very small: 10 – 40 MHz on a GHz scale. If an increment causes instability or crashes the system, the clock is reset to what it was before the last increment and the increment amount is reduced. This reduced increment is then applied incrementally till the system is unstable or crashed again. At which point the reduced increment is again reduced, and this new reduced increment is applied incrementally till the next crash point is found. This process continues till the increment is less than 1MHz or the overclocker loses patience and is happy with the results. For GPUs this process is repeated twice: once for the core clock, and once for the memory clock.

It is due to this tedious nature of overclocking, companies such as MSI or EVGA have their own auto-overclockers for GPUs. These are Afterburner and Precision X1 respectively. Auto-overclockers are programs that automatically overclock the GPU using a lookup table of voltage-frequency values to find the right pair of voltage-frequency values that give a stable overclock. These programs start by first applying a well know overclock profile for the specific card that is guaranteed to give a stable overclock. This overclock profile is then stress tested for a certain amount of time. If the stress test completes with no errors, the next higher overclock profile is applied. If this profile succeeds, the next one is applied. The program continues doing this till it finds an overclock profile that fails, at which point the last known good overclock profile is applied and stress tested for a longer period of time to be guaranteed stability. If this profile fails, the last last known good profile is applied and stress tested. This process continues till it finds an overclocked profile.

The problem with these auto-overclockers, mentioned in the above paragraph, is that they can only go from one profile to the next in the table. If the next overclock profile fails, it applies the overclock profile before it in the table. However, there might be an overclock profile between the current failed profile and the last known good profile that is stable and gives a higher overclock than the last known

good profile. Oftentimes, these auto-overclockers cannot apply this optimal profile since it is not in their lookup table.

In this development project we develop a raspberry pi system with a sophisticated iterative algorithm that is guaranteed to find the best optimal overclock profile. Our algorithm not only overclocks the core clock, but also overclocks the memory and undervolts the GPU core to reduce heat and noise while also maintaining the overclock. For overclocking, our algorithm does the exact same thing that a human overclocker would do, but with more patience.

Required hardware:

We are developing the algorithm on a single board computer to prevent potential interference between stress tests and data collection performed by the algorithm. Therefore, the required hardware for this project will be a raspberry pi 4 and an Ethernet cable to connect the pi to the host machine containing the GPU to send instructions to adjust speed and power limits. The power limits will be partially adjusted based on temperature data obtained from temperature sensors on the GPU.

We chose the raspberry pi 4 since it is similar in cost to a raspberry pi 3, and the pi 4 has a more powerful CPU than pi 3. We could not have used a pi zero since it does not have a RJ45 jack.

Required software:

Our algorithm needs to be able to set the GPU core clock speed, the memory speed and the power offset. We have three options for software to be able to set these: (1) NVIDIA Settings, (2) Nvidiux, (3) GreenWithEnvy (GWE).

We are not using NVIDIA Settings since it is proprietary and is owned by NVIDIA. This software does offer terminal options to set the core clock and memory offset speed, but does not have any options to set the power offset.

Nvidiux is open source and seems promising. Unfortunately we are also not going to be using this since most of the documentation we found is in French. This software also does not have very many options for data collection, which is crucial for performance comparison of our algorithm with proprietary auto-overclockers.

We have decided to use GWE (GreenWithEnvy) overclocker since it is open source and also does some data collection for plotting. On top of that the documentation and code comments are written in English. Documentation for this software can be found by contacting the developers whose contact information is on gitlab.

Our algorithm also needs to be able to stress test the GPU after every clock increment. In order to stress the GPU, we are using a power virus called FurMark.

Obtaining the hardware:

The raspberry pi can be ordered from amazon with prime shipping with guaranteed delivery in 2 days. The Ethernet cable, GPU and the host machine do not need to be purchased as we already have them.

Obtaining software and documentation:

The software we are using for this project is open source and is freely available on gitlabs with documentation. For further queries regarding the software, the authors can be contacted. The power virus software can be obtained from geeks3d.com. Its documentation can also be obtained from the same website.

Expected cost:

The raspberry pi 4 with 1GB RAM costs about $50 and is the only equipment that needs to be purchased. This cost can be split between the teammates.