INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-02-19

EXAMENSARBETE Integration of a Cycle-approximate Model Into a Cycle-accurate Environment
STUDENT Andreas Hansson
HANDLEDARE Jörn Janneck (LTH), Reimar Döffinger (Arm)
EXAMINATOR Flavius Gruian (LTH)

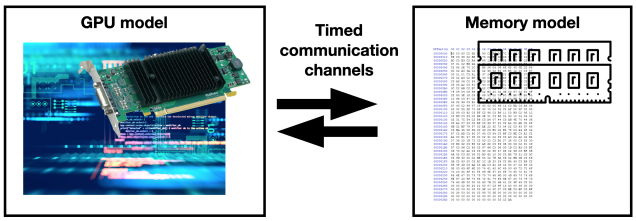
Getting Simulated Hardware Components to Work Together

POPULAR SCIENCE SUMMARY **Andreas Hansson**

Software models are used to aid the development of hardware components. To simulate systems with multiple components, multiple models can be connected together. In our project we show how mistimed communication can complicate such setups, and how this can be handled.

Over a billion phones are sold each year. The phone market is very competitive, with customers expecting constant improvements. As companies compete to satisfy these demands there is a need for optimised development processes. To speed up the development of chips, software models that mimic hardware behaviour are used. By having virtual versions of the chips being designed, designers and developers can try a lot of different changes without having to rebuild the actual parts. Different testing needs have led to the existence of a lot of different types of models, and the degree to which they stay true to the hardware behaviour varies.

In this investigation, we have looked at a model simulating an Arm Mali GPU – a popular chip used to handle graphics in many phone models. When a graphics processing unit (GPU) runs, it frequently needs to use data that is located in a memory chip that is external to the GPU itself. Transferring between these takes a bit of time, so the GPU has to plan accordingly when it wants to fetch data. However, this simulation model has been implemented under the assumption that it can instantly access the data when needed. This makes a lot of testing easier, but makes the model incompatible with memory models that only work with timed communication channels and do not allow for such instant access.



In order to enable compatibility our project has looked at the different ways the GPU model tries to access data before it has been fully transferred. To address these we have created a mechanism that tracks all transfers taking place and successfully identifies the incorrect data access attempts. When we discover an incorrect attempt we send out for the needed transfers and wait for these to be handled before we continue with the computation. This ensures that the GPU model never makes use of any different data than intended, which is necessary for the result of the computation to be correct. As we successfully identify and handle all the incorrect attempts taking place, we can now connect the GPU model to simulated systems that require timed transfers - enabling new test cases.