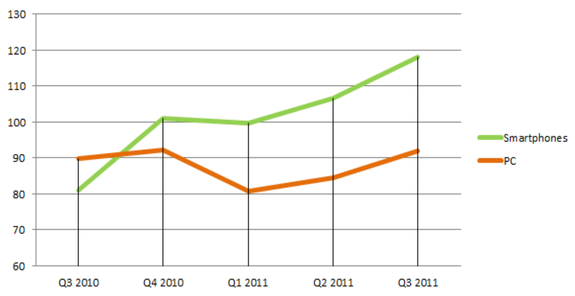
A Decoupled Access/Execute Architecture for Mobile GPUs

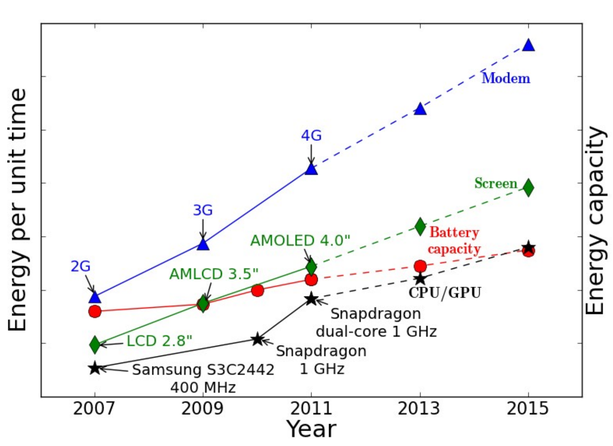
PhD Student: Jose-Maria Arnau[1] **Advisors:** Joan-Manuel Parcerisa[1], Polychronis Xekalakis[2]

[1] Computer Architecture Department, Universitat Politècnica de Catalunya
[2] Intel Barcelona Research Center, Intel Labs Barcelona

1. Motivation

- Smartphones represent a huge growing **market**.
- Real **computing experience** on a smartphone: web browsing, email, video and picture editing, 3D games...





- **Battery operated devices!** Supporting all the capabilities reduces the operating time per battery charge.
- Most of the power is consumed by the **screen** and the **communications subsystem**.

- Battery capacity evolves at the modest pace of 5-10% per year, whereas the energy demand increases significantly on each generation of smartphones.
- The GPU is only left with a small fraction of the power budget.
- Better screens will require better graphics rendering, that is, higher performing GPUs.

More energy-efficient mobile GPUs are required to maintain the operating time per battery charge.

2. Background

GPU pipeline

- Non-programmable stage (ASIC)
- Programmable stage

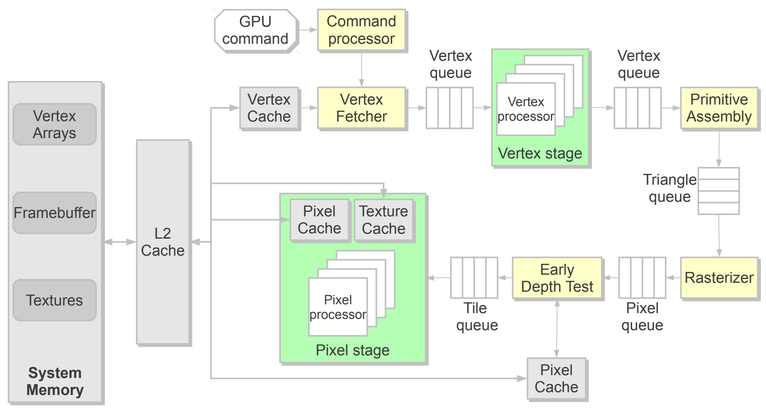
Memory hierarchy

- On-chip caches
  - Specialized L1 caches (vertex, texture, pixel caches).
  - L2 cache
- Off-chip system memory

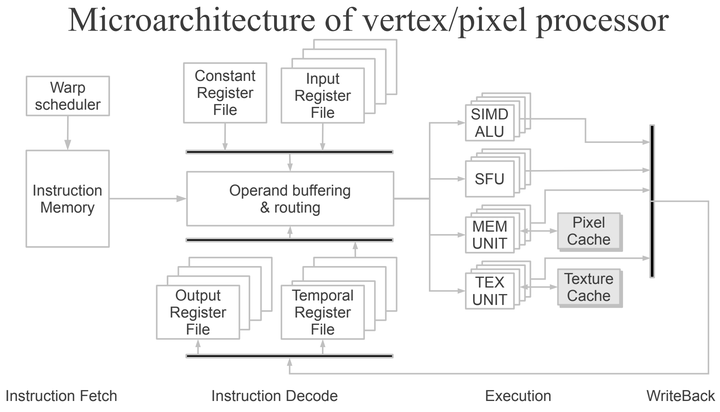
Vertex/Pixel processors

- Simple 4-stage pipeline
- In-order
- SIMD instructions
- Threads grouped in warps (SMT).

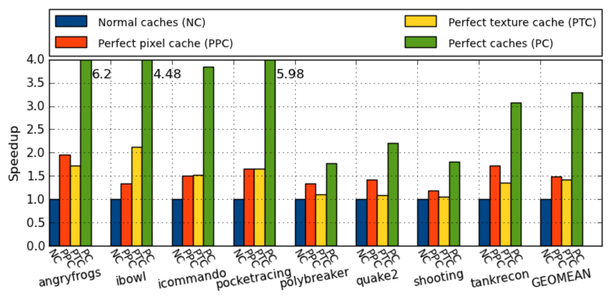
Microarchitecture of Tegra-like mobile GPU



Microarchitecture of vertex/pixel processor

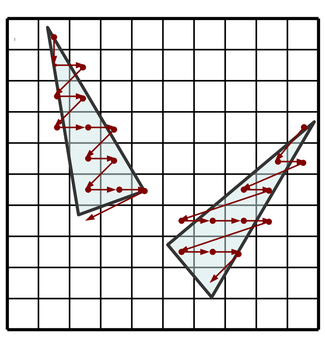


3. Memory hierarchy



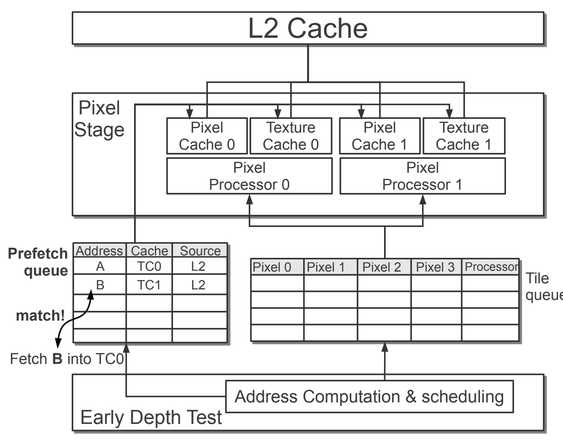
- Due to the **highly parallel nature** of graphical applications, we believe that the main complication that will have to be dealt is how to bridge the **memory gap** in an energy efficient manner.
- Traditional ways to hide the memory latency applied to a mobile GPU:
  - **Caches**: save bandwidth, but not so effective like in a CPU.
  - **Prefetching**: Useful but there is ample room for improvement.
  - **Multithreading**: Effective but power-hungry.

- Triangles can be at any position and orientation.
- Graphical workloads exhibit **irregular** and **unpredictable** memory access patterns.



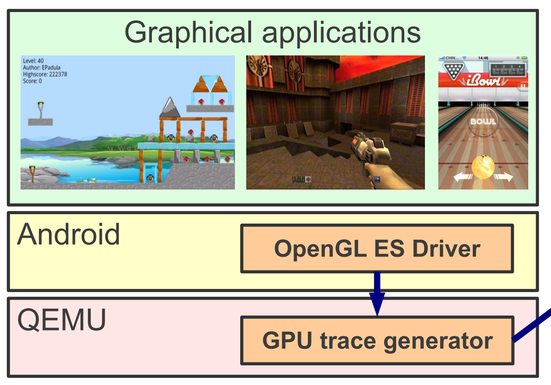
4. Decoupled Access/Execute

- Traditional DAE architectures:
  - Divide the program into **two independent instruction streams**, one doing memory accesses and the other performing computations.
  - PROBLEM: loss of decoupling events (LODs). But GPU **pixel programs are typically free of LODs!**.
- DAE on a mobile GPU:



- Use the per-pixel information to **prefetch** all the necessary data while the pixels are waiting in the tile queue:
  - **Computed** addresses instead of **predicted** addresses (better accuracy than hardware prefetchers).
- Use the prefetch queue to detect and exploit **inter-core data sharing:**
  - L2 cache **bandwidth** savings.
  - **Energy** savings: accessing a small first level cache requires less energy than accessing the L2 cache.

5. Methodology



- **Workloads**: 8 Android games.
  - 2D games: *angryfrogs, icommando, pocketracing.*
  - Simple 3D games: *polybreaker, shooting.*
  - Complex 3D games: *ibowl, quake2, tankrecon.*

GPU simulator parameters	
Frequency	600 Mhz
Voltage	1 V
Scree resolution	800x480 (WVGA)
Technology	45 nm
Main memory	latency = 100 cycles bandwidth = 4 bytes/cycle
Pixel/Texture caches	2 KB, 2-way, 2 cycles
L2 cache	32 KB, 8-way, 12 cycles
Number of cores	4 pixel and 4 vertex processors
Number of threads per warp	4

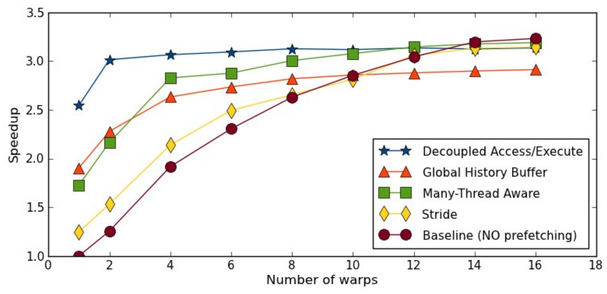
Comparison with hardware prefetchers

Stride Prefetcher: John W. C. Fu, Janak H. Patel, and Bob L. Janssens. "Stride directed prefetching in scalar processors". SIGMICRO Newsl., pp. 102-110, Dec. 1992.

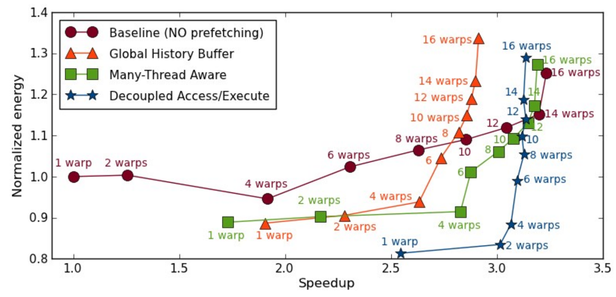
Global History Buffer: K. J. Nesbit and J. E. Smith. "Data Cache Prefetching Using a Global History Buffer". In Proc. of HPCA, pp. 96-105, February 2004.

Many-Thread Aware: J. Lee, N. B. Lakshminarayana, H. Kim and R, Vuduc. "Many-Thread Aware Prefetching Mechanisms for GPGPU Applications". In Proc. of MICRO, pp. 213-224, December 2010.

6. Results



- DAE provides 33% speedup and 9% energy saving over state-of-the-art hardware prefetchers.
- Multithreading provides significant benefits (3.23x speedup for 16 warps), but it increases energy consumption (25% more energy for 16 warps).
- DAE achieves its maximum performance with just 4 warps, whereas the baseline needs 14-16 warps to achieve the same performance (the rest of prefetchers lay in between).
- DAE with 2 warps achieves 93% of the performance of a bigger GPU with 16 warps, but consuming 34% less energy.
- DAE can take benefit of a small degree of multithreading (e.g. 4 warps) to hide the latency of the functional units and keep them busy, which is an issue that the access/execute mechanism does not address.



7. Conclusions

- Energy-efficient mobile GPUs can be architected based on the decoupled access/execute paradigm.
- A small degree of multithreading is still useful, a combination of DAE (to hide the memory latency) and non-aggressive multithreading (to hide the latency of the functional units) provides the most energy efficient solution.
- A DAE architecture with 2 warps/core achieves 93% of the performance of a bigger GPU with 16 warps/core, but it consumes 34% less energy.
- A DAE architecture outperforms hardware prefetchers by providing 33% speedup and 9% energy savings.