GPGPU on ARM

Tom Gall, Gil Pitney, 30th Oct 2013

Session Description

- This session will discuss the current state of the art of GPGPU technologies on ARM SoC systems.
- What standards are there? Where are the drivers? What is the best hardware? What are the problem areas? What open source software has been accelerated? What opportunities for taking advantage of GPGPU computing on ARM exist?
- There will also be some time for interactive discussion about current and potential plans.

GWG: New GPGPU Sub-group

- At previous Connect (Dublin) discussed GPGPU, and what Linaro should do in that area.
- Recognized that:
  - GPUs are much more energy efficient for highly parallel computing problems than using just the main processor, even with NEON SIMD.
  - Big performance gains (> 20X) using the GPU for certain open source workloads.
- Decided:
  - to get hardware and start gaining some experience with GPGPU drivers;
  - it would be good to accelerate some OSS projects with GPGPU.
- So, we formed a GPGPU sub-team as part of the GWG.

GPU to GPGPU: Hardware Evolution

- **Graphics pipeline operations migrate from the CPU to GPU:**
  - **’80’s: CPU + framebuffers with blitters.**
  - **’90’s: GPUs output one pixel per clock cycle, add pipeline operations:**
    - texture mapping, z-buffering, rasterization;
    - transform and lighting (still fixed hardware) operations.
- **Fixed functions become programmable kernels, adding multiple pipelines:**
  - **2000’s: GPUs become programmable, multicore:**
    - Programmable kernels (shaders) added to the per-pixel and vertex processing stages.
    - More floating point precision, more GPU memory
    - Unified shaders (eg: GeForce 8 “Streaming Multiprocessor”) handle vertex, pixel **and** geometry computations.
- **GPU**: initially a single core, fixed function hardware pipeline implementation designed solely to accelerate graphics becomes…
- **GPGPU**: set of parallel and highly programmable cores (shaders) which can be leveraged for more general purpose computation.

GPU to GPGPU: Software Evolution

- Software co-evolves with GPU hardware.
  - ’90’s: Graphics Languages: OpenGL (led GPU hardware features), DirectX.
  - 2000: Shading Languages: OpenGL/GLSL, Direct3D/HLSL, Cg, others...
- Shading Languages: Need to understand graphics pipeline ops.
- But, shaders basically perform matrix and vector operations, ideal for many non-graphics, scientific applications.
- Enter higher level parallel computing “stream processing” languages:
  - 2007: CUDA “*Compute Unified Device Architecture*” (NVIDIA)
  - 2008: OpenCL “*Open Compute Language*” (Apple/Khronos)

GPGPU Use Cases

- Ideal GPGPU applications have large data sets, high parallelism, and minimal dependency between data elements:
  - Advanced imaging: (Face recognition, cloth simulation, object detection).
  - Cryptography searches: (Bitcoin, distributed.net).
  - Video decode and post-processing: (iDCT, motion compensation, VLD, IQ, edge enhancement)
  - Bioinformatics: (BLAST (protein and genome sequence comparisons), Folding@Home).
  - Genetic Algorithms, Weather prediction, Particle Physics simulations
  - MapReduce algorithm.
- The “13 Dwarfs of GP/GPU computing” [2] and benchmarks [1] evaluate parallel programming models and hardware architectures for algorithms sharing similar computation and inter-processor communication patterns:
  - Computationally limited: Dense linear algebra, N-Body methods, cryptography.
  - Memory bandwidth limited: Sparse linear algebra, structured grid.
  - Memory latency limited: Graph traversal, dynamic programming, unstructured grid, FFT.

Standards / Programming Platforms

- Hardware:
  - **HSA (Heterogeneous System Architecture)** foundation creates hardware and software standards. Implementations coming…. (AMD/Kaveri, PS4: 4Q13?). Hardware must be certified HSA compliant.
  - **HSA** requires one fully coherent shared memory heap, with unified addressing, between CPUs and Compute Devices (i.e. a pointer can be passed between CPU and GPU directly). An HSA-specified MMU (hMMU) allows the compute devices to share the page table mappings with host CPU(s).
- Software:
  - **OpenCL (Open Compute Language)** is an open standard by the Khronos Group. One is free to implement the standard and pay for certification. A large and growing number of problem domain libraries and apps have been built on OpenCL.
  - **CUDA (Compute Unified Device Architecture)** by NVidia is limited to NVidia hardware. There is no standardization body and no certification. Historically CUDA was not able to be licensed but on June 18th, 2013 nVidia announced it was going to start licensing its GPU IP.
  - **Renderscript,** part of Android, by Google. Renderscript is not a standard. There is no certification. There is no implementation outside of Android.
  - **HSA tools**: Compiler to HSAIL, HSAIL Finalizer to accelerator code, and HSA host runtime software.

Open Source Software accelerated with GPGPU

- Many OSS libraries support CUDA and/or OpenCL.
  - Linear Algebra Libraries: ViennaCL, MAGMA, clMath(clBLAS), VexCL.
  - Vision: OpenCV (Computer Vision) some modules written to CUDA/OpenCL.
  - Database: sqLite on CUDA (20x-70x speedup for SELECT queries).
  - Finance: QuantLib (faster option pricing for HFT, Monte Carlo simulations, and company credit risk).
  - Scientific Domains: OpenMM (Molecular Modeling), OpenBR (Biometric Recognition).
- Language bindings ease programmability, proliferate GPU usage:
  - WebCL: JavaScript binding to OpenCL, and Khronos standard. Demos have shown (50x-100x) performance improvements over pure javascript web apps.
  - Python: PyOpenCL, Java: Aparapi, Jogamp/JOCL.

OpenCL Drivers on Various Hardware

- ZiiLabs:
  - H/W: ZMS-40: Quad ARM Cortex A9 with 96 StemCell cores.
  - S/W: OpenCL 1.1 (desktop profile) drivers from Creative for custom devices.
- Vivante:
  - H/W: GC600+ series licensed to Marvell ARMADA, Freescale i.MX6, etc.
  - S/W: OpenCL 1.1 EP drivers that work on FreeScale processors (SabreBoard).
- Qualcomm:
  - H/W: Adreno 300+ series, in Snapdragon SoC's.
  - S/W: OpenCL 1.2 EP Android drivers in Adreno SDK (not shipped on device).
- ARM Mali:
  - H/W: Samsung Exynos 4/5 series, many others. (eg: Arndale, Chromebook).
  - S/W: Drivers for Exynos only seem to work on the Arndale-board when the LCD is also ordered; Chromebook Mali drivers just made available (Oct 2013).
- Imagination Technologies:
  - H/W: TI OMAP, Exynos 4/5 series with PowerVR SGX GPUs
  - S/W: OpenCL 1.1 EP: TI requires NDA, Samsung ODROID-XU Android drivers avail (Linux drivers require NDA).
    - » Source: http://streamcomputing.eu/blog/2013-09-05/mobile-processor-opencl-drivers-q3-2013/

Challenges and Opportunities

- Challenges
  - GPGPU drivers are mostly closed, and only slowly becoming available.
  - Google invested in Renderscript vs OpenCL on Android.
  - HSA compliant hardware can extend the GPGPU into new problem domains by increasing performance for memory latency constrained algorithms, but new hardware designs take a long time to implement.
  - Programming models like Renderscript/OpenCL still not so simple.
- Opportunities
  - Performance: Select a set of key algorithm libraries to benchmark and accelerate.
  - OpenCL: Invest in a CPU OSS version of OpenCL, and update to latest OpenCL spec.
  - Simplify Programmability: Explore directive based models: eg: OpenACC/OpenMP 4.0 ?

GWG Roadmap: GPGPU Next Steps

- **Address availability of OpenCL driver needs** through the use of **Clover** (an open source CPU only driver) and advance the use of non GPU devices such as TI DSPs which will also be supported by Clover. *Update Clover to OCL v1.2*.
- **Accelerate libjpeg-turbo** : utilizing an existing OpenCL port, validate it works on an ARM OpenCL implementation, and compare to a NEON optimized version.
- **Accelerate sqlite** : Sqlite is a database found in mobile and embedded solutions for both Android and Linux. A past project accelerated it using CUDA. We propose to accelerate the database using OpenCL.
- **Accelerate OpenCV**: OpenCV is an open source computer vision and machine learning software library built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Accelerate using OpenCL.

connect.linaro.org

More about Linaro: http://www.linaro.org/about/
More about Linaro engineering: http://www.linaro.org/engineering/
How to join: http://www.linaro.org/about/how-to-join
Linaro members: www.linaro.org/members