CG-solver on the embedded *ARM Mali-T604* GPU, in OpenCL

Olav Aanes Fagerlund
Department of Systems Innovation
School of Engineering
The University of Tokyo
7-3-1 Hongo Bunkyo-ku,
Tokyo 113-8656, Japan

Serban Georgescu
Fujitsu Laboratories of Europe Ltd.
Hayes Park Central, Hayes End Road
Hayes, Middlesex
United Kingdom
UB4 8FE

Hiroshi Okuda
Graduate School of Frontier Sciences
The University of Tokyo
5-1-5 Kashiwanoha, Kashiwa,
Chiba 277-8563, Japan

CUKr** – **CU**DA **Kr**ylov on the **ARM architecture

- GPGPU based Krylov solver library
- Originally developed in 2008/2009
- Supports single, double and quasi-double precision (single-double)
- Initially written in CUDA, kernels optimized for Nvidia GPUs
- In 2009/2010 extended with OpenCL as a build-alternative
- Recently; ported to the Samsung Chromebook XE303C12, which contains the Samsung Exynos 5250 SoC and the ARM Mali T604 embedded GPU
- Kernels are currently optimized for the Mali T604 architercture, a work in progress
- A power efficiency study is the goal of this port, for Exascale computing the power efficiency is increasingly more important, i.e. FLOP/s / watt (FLOP per second per watt)

Samsung Exynos 5250 SoC / ARM Mali T604 GPU

- The chip chosen for the current research system by the European Exascale initiative, headed by Barcelona Supercomputing Center
- Dual core ARM A15 CPU, 1.7 GHz
- Quad core ARM Mali T604 GPU, 533 MHz
- CPU and GPU shares a 2GB integrated SoC memory, over a bus. Peak memory bandwidth (they compete) is 12.8 GB/s
- The SoC has a total power consumption of ~10 watt
- The Mali T604 part has an estimated ~4 watt power consumption. This number is not certain, as it is not released in any form known to the authors.
- ***Mali T604 supports***
    - OpenCL v.1.1 full and embedded profile
    - Double precision computation
    - Efficient vector operations, including memory transfers (*vloadn* & *vstoren*), on 128-bit wide vectors (i.e. vectors of 4 floats, 2 doubles, 4 ints etc.)
    - No physical Local memory, it is mapped to Global memory

Upper roof of performance for our routines

We have 5 BLAS routines of importance

(1) SpMV (CSR, CSR4, HYB) (2) AXPY (3) AYPX (4) DOT (5) SCAL

Of these most time is spent doing SpMV.

Peak performance theoretically possible for kernels (2)-(5) can be estimated;

FLOP / Bytes x Bandwidth = Performance (1)

where 'FLOP' is # of floating point operations, 'Bytes' is # of bytes needed for the operations, 'Bandwidth' is device memory bandwidth in GB/s, and 'Performance' is the performance theoretically achievable in GFLOP/s.

BLAS level 1	AXPY	AYPX	DOT	SCAL
Single Precision	2FLOP/12Byte x 12.8 GB/s = **2.13 GFLOP/s**	2FLOP/12Byte x 12.8 GB/s = **2.13 GFLOP/s**	2FLOP/8Byte x 12.8 GB/s = **3.2 GFLOP/s**	1FLOP/8Byte x 12.8 GB/s = **1.6 GFLOP/s**
Double Precision	2FLOP/24Byte x 12.8 GB/s = **1.07 GFLOP/s**	2FLOP/24Byte x 12.8 GB/s = **1.07 GFLOP/s**	2FLOP/16Byte x 12.8 GB/s = **1.6 GFLOP/s**	1FLOP/16Byte x 12.8 GB/s = **0.8 GFLOP/s**

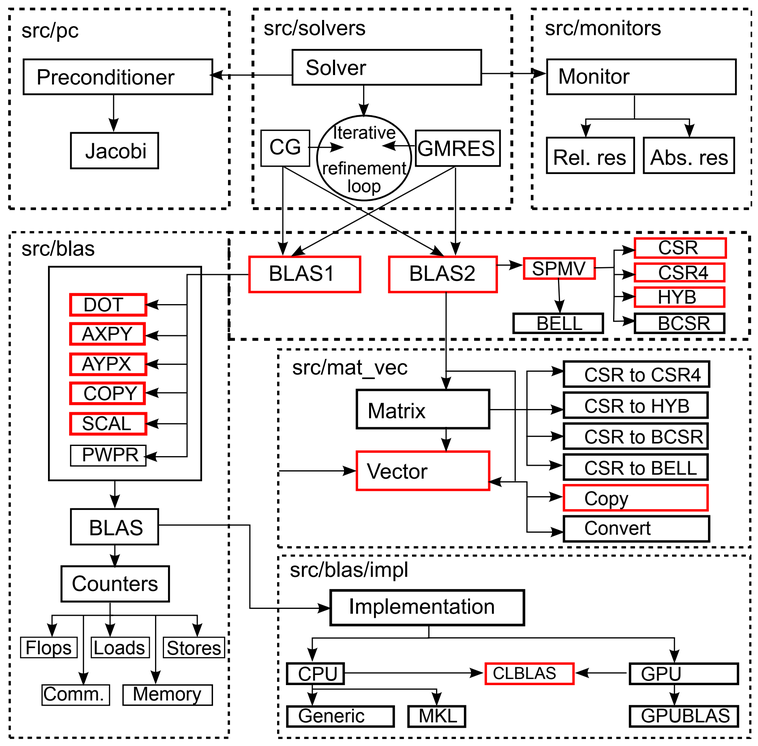


Figure 1: CUKr block diagram, red indicates the areas where OpenCL was integrated.

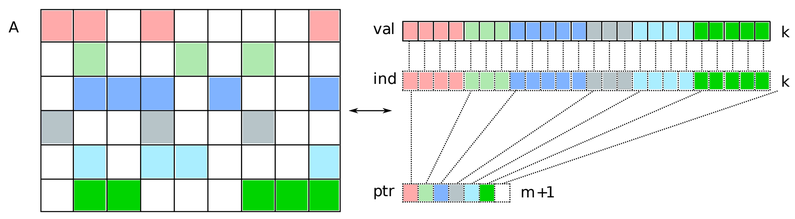
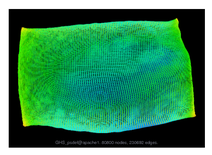
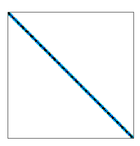
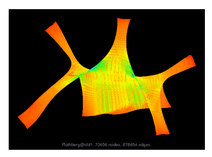
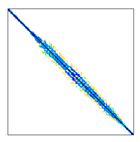
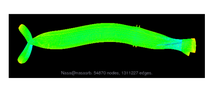
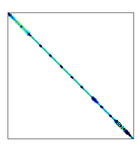
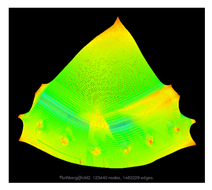
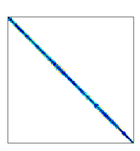
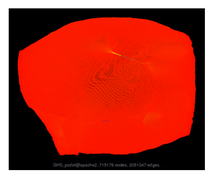
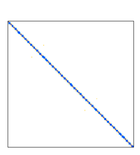
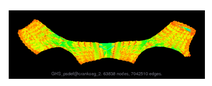
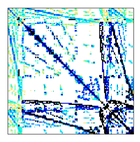


Figure 2: The plain CSR matrix storage scheme. Ideal for CPUs. For GPUs there are even more efficient formats. Like the HYB format which is a hybrid of CSR4 and the ELLPACK format (more ideal for vector processors, which the GPU arguably is). CUKr supports CSR, CSR4, and HYB, in OpenCL. Optimization and adoptation of the latter two formats remains.

A Krylov solver

- A iterative solver of importance for Finite Element Method (FEM) problems
- Iterative solvers are needed for larger problems due to memory consumption properties
- FEM problems in mechanical engineering are of a highly sparse nature.
- A matrix can typically contain less than 0.01% non-zero elements.
- ***Highly memory bound problems*** – the processor is constantly waiting for the data needed: high bytes-per-FLOP ratio
- Efficiency of the memory subsystem and its bandwidth, and how the algorithm suits the memory architecture, heavily affects the performance

Name and description	*Undirected graph drawing*	*Matrix non-zero pattern*
Matrix: GHS_psdef/apache1 Description: SPD matrix (finite difference 3D) from APACHE small		
Matrix: Rothberg/cfd1 Description: CFD, symmetric pressure matrix, from Ed Rothberg, Silicon Graphics, Inc.		
Matrix: Nasa/nasasrb Description: STRUCTURE FROM NASA LANGLEY, SHUTTLE ROCKET BOOSTER		
Matrix: Rothberg/cfd2 Description: CFD, symmetric pressure matrix, from Ed Rothberg, Silicon Graphics, Inc.		
Matrix: GHS_psdef/apache2 Description: SPD matrix (finite difference 3D) from APACHE small		
Matrix: GHS_psdef/crankseg_2 Description: OUTPUT4-Matrix		

Matrix	Rows	Cols	Nz	Rank	Full Rank	Structure	SPD	Type	Kind
apache1	80800	80800	542184	80800	yes	symmetric	yes	real	structural problem
cfd1	70656	70656	1825580	70656	yes	symmetric	yes	real	CFD problem
nasasrb	54870	54870	2677324	54870	yes	symmetric	yes	real	structural problem
cfd2	123440	123440	3085406	123440	yes	symmetric	yes	real	CFD problem
apache2	715176	715176	4817870	715176	yes	symmetric	yes	real	structural problem
crankseg_2	63838	63838	14148858	63838	yes	symmetric	yes	real	structural problem

Figure 3: The 6 problems and their properties, solved on the Mali T604.

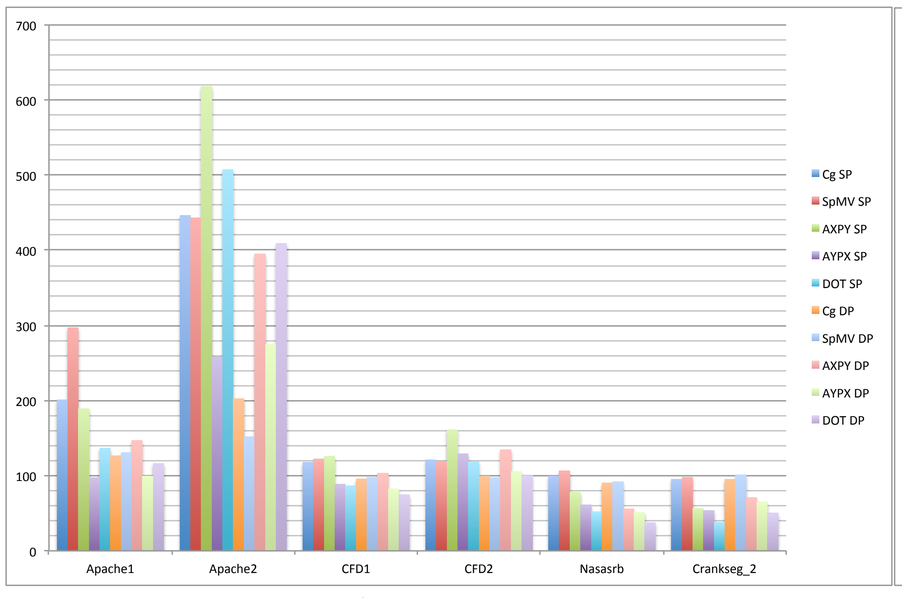


Figure 4: Performance in MFLOP/s for the different kernels, when running the solver. AXPY is the most optimized kernel. SpMV, where most time is spent, has not been optimized yet.

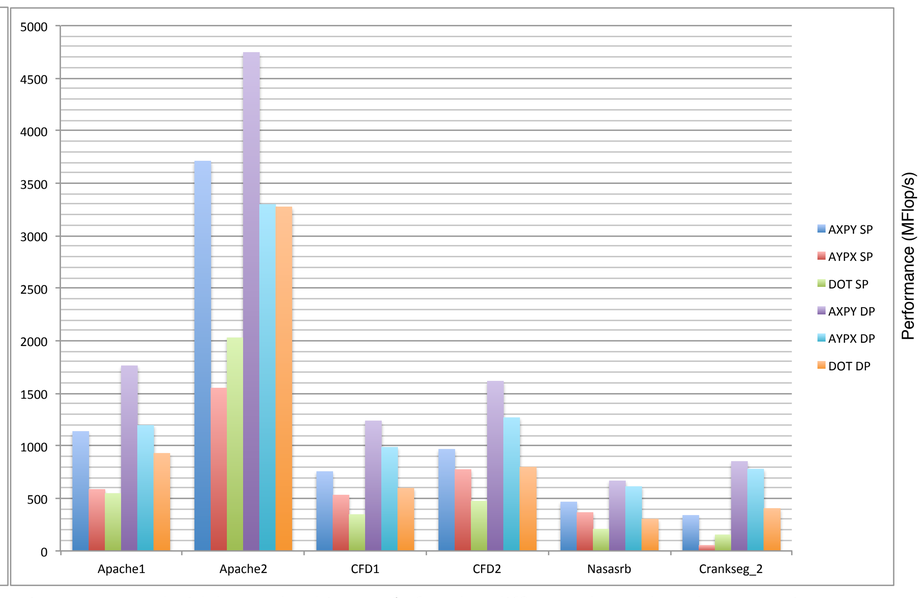


Figure 5: Bandwidth reached in MB/s for the different kernels, when running the solver. AXPY is the most optimized kernel.

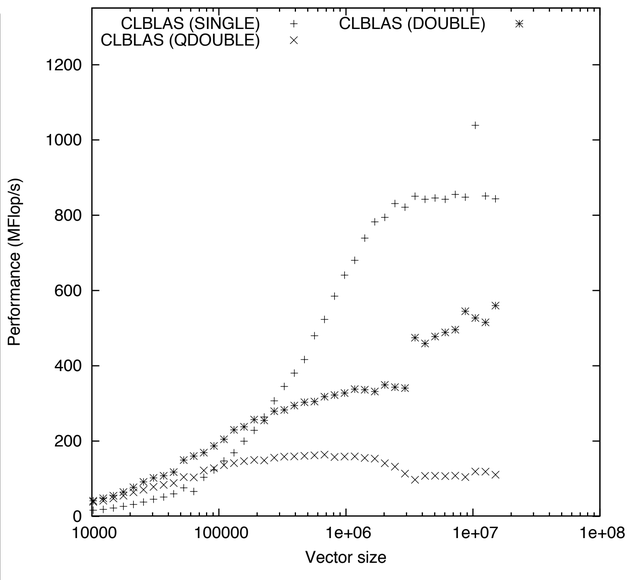


Figure 6: Performance in all precisions for AXPY, tuned for Mali T604. At peak a bandwidth of close to 5 GB/s and 7 GB/s is reached for single and double precision, respectively.

Conclusions:

Real world problems can have a varying performance as ideal access pattern and utilization is not reached constantly. Still work remains in optimizing the kernels better. The estimated ~4 watt power consumption of the Mali T604 makes this research of interest. Future work includes further optimizations towards peak, and measuring the efficiency in performance per watt compared to standard desktop discrete GPUs.

References:

Serban Georgescu. *Krylov Solvers Accelerated by Manycore Processors*. PhD thesis, The University of Tokyo, Japan, 2009.

Olav Aa. Fagerlund. *Multi-core programming with OpenCL: performance and portability*. Master's Thesis, NTNU, Norway, 2010.

ARM Mali OpenCL Developer Guide: *http://malideveloper.arm.com/develop-for-mali/tutorials-developer-guides/*

Richard W. Vuduc. *Automatic Performance tuning of sparse matrix kernels*. PhD thesis, University of California, Berkeley USA, 2003.

The University of Florida Sparse Matrix Collection: *http://www.cise.ufl.edu/research/sparse/matrices/*