Task Parallel Programming Model + Hardware Acceleration = Performance Advantage

Tamer Dallou[1], Divino Cesar Soares Lucas[2], Guido Araujo[2], Lucas Morais[2], Eduardo Ferreira Barbosa[2], Michael Frank[3], Richard Bagley[3], Raj Sayana[3]

[1] LG Electronics Technology Center Europe, [2]University of Campinas, Brazil (Unicamp), [3]LG Electronics Mobile Research, San Jose Lab

- **SoC are multi-core heterogeneous computing platforms**
  - No good tools for programming them efficiently (debugging, instrumentation) exist
  - writing parallel programs the old way (by using pthreads) does not scale
  - Programming for these platforms is hard and prohibits efficient use of all available compute resources
- **Support many-cores and heterogeneous configurations for applications**
  - We propose to adopt TaskSuperscalar (TSs) programming models
    - OmpSs, OpenMP 4.0, StarPU …
  - Break down large programs into a connected mesh of small tasks; running on heterogeneous group of processors, use dataflow dependencies for synchronization (instead of barriers)
  - Out-of-order execution of tasks depends on inter-task dependencies
- **Task definition, creation and submission**
  - Sequential code with pragmas to convey additional information to construct the data-flow-graph
  - Record dataflow dependencies as tasks are submitted

```
#pragma task input(a) output(b)
void fun1(float *a, float *b);

#pragma task input(a) output(b)
void fun2(float *a, float *b);

for (int i=1, j=1; i<N; i++) {
    fun1(&v[i-1], &v[i]);
    for (int k=0; k<i; k++, j++) {
        fun2(&v[i], &u[j]);
    }
    fun3( 3 * i );
}
```

fun1 and **fun2** are Kernel Functions.

Parameter **semantics** are informed: IN, OUT or INOUT

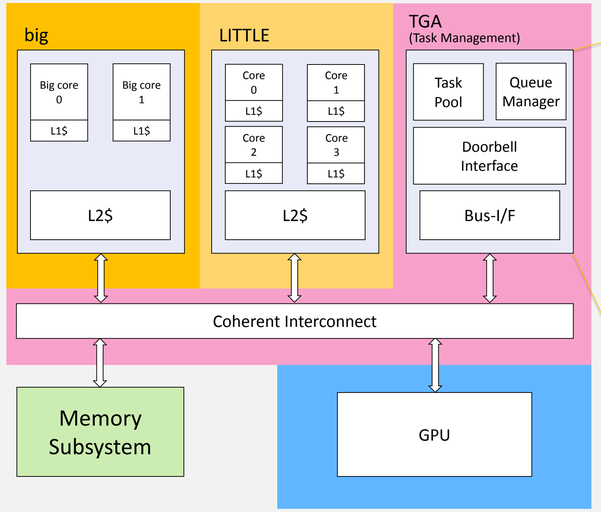
Will create a new Task!

Will create a new Task!

Will **NOT** create a new Task!

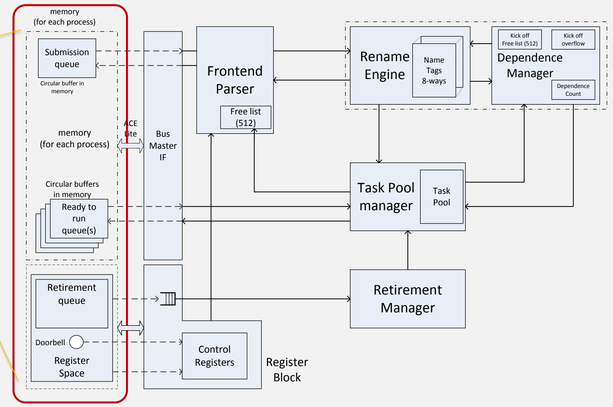
- **Running small tasks**
  - Run-time system introduces a fixed, non-negligible overhead for task creation, task submission, task issue and dependence management
  - HW support, acceleration, for critical functions, for example task scheduling and dependence resolution
  - Manage helper tasks (preload data for accelerators)

Heterogeneous Compute Platform Architecture



Important: *Coherent memory access minimizes communication overhead between CPU cores and TGA*

Tioga Task-Graph-Accelerator



- **Task Graph Accelerator key parameters**
  - Estimated die area: < 0.1 mm² (TSMC 16FF)
  - Clock speed: > 800 MHz
  - Capacity:
    - 512 active tasks
    - 4096 graph edges (data flow dependencies) on up to 512 variables
  - Design has been implemented and demonstrated on a ZEDBOARD/ZYBO (XILINX Zynq) platform

Software Architecture

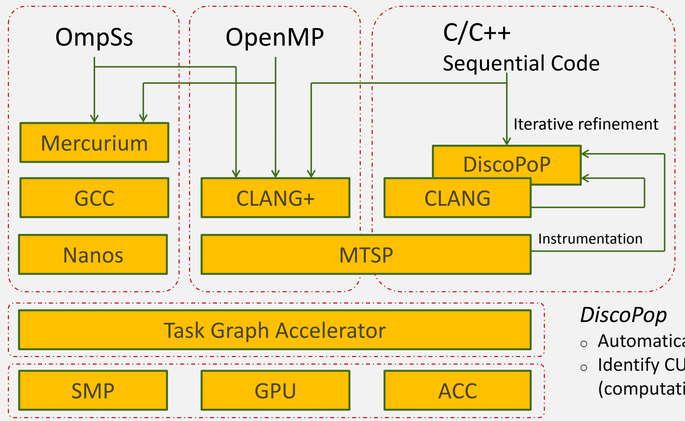
Application Level

Compiler Support
- GCC = GNU compiler
- CLANG = LLVM compiler
- CLANG+ = LLVM compiler with OmpSs support added

TSs Runtime
- Nanos = BSC runtime
- MTSP = UNICAMP runtime

HW Runtime Accelerator

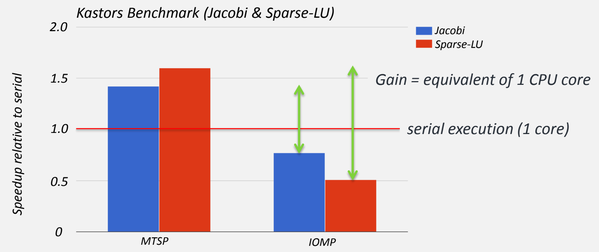
SoC Compute Platform



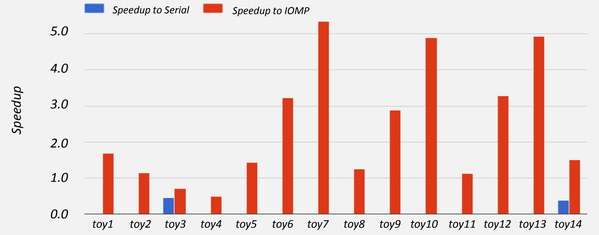
DiscoPop
- Automatically parallelize
- Identify CU tasks (computational Units)

Performance (run on Xilinx Zynq platform)

2 cores = CA9 @660 MHz; Tioga @50 MHz



MTSP = UNICAMP runtime (with HWS)
IOMP = Intel OpenMP runtime
Both benchmarks use 2 cores



Kastors Benchmark (https://gforge.inria.fr/projects/kastors/)
- The KASTORS benchmarks suite was designed to evaluate OpenMP 4.0 task dependencies
- Modified state-of-the-art OpenMP 3.0 benchmarks and data-flow parallel linear algebra kernels to make use of tasks with dependencies
- KASTORS can also be used to evaluate performance of OpenMP implementations of task dependencies compared to global taskwait-based approaches

Toy Benchmarks
- Synthetic with different number and type of dataflow dependencies

Literature

[1] Y. Etsion, A. Ramirez, R. M. Badia, E. Ayguade, J. Labarta, and M. Valero. Task Superscalar: Using processors as functional units. In Hot Topics in Parallelism (HOTPAR), 2010.

[2] Y. Etsion, F. Cabarcas, A. Rico, A. Ramirez, R. M. Badia, E. Ayguade, J. Labarta, and M. Valero. Task Superscalar: An Out-of-Order task pipeline. In Intl. Symposium on Microarchitecture (MICRO)

[3] T. Dallou, B. Juurlink, FPGA-based prototype of Nexus++ task manager, in: 6th Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS13), 2013.

[4] F. Yazdanpanah, D. Jiménez-González, C. Álvarez-Martínez, Y. Etsion, R.M. Badia, Analysis of the task superscalar architecture hardware design, in: ICCS'13, 2013

[5] Fahimeh Yazdanpanah; Hardware Design of Task Superscalar Architecture, PhD Thesis, Universitat Politecnica de Catalunya, 2014

[6] J. Planas, R. M. Badia, E. Ayguade, and J. Labarta. Hierarchical task-based programming with StarSs. J. High Performance Computing Application, 23(3). 27

[7] A. Duran, E. Ayguade, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas. OmpSs: A proposal for programming heterogeneous multi-core architectures. Parallel Processing Letters, 2011