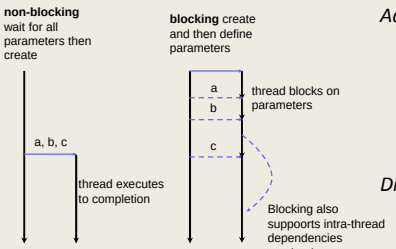
Architecture Paradigms and Programming Languages for Efficient programming of multiple CORES

Objectives

- **Apple-CORE** is developing **compilers**, **operating systems** and **execution platforms** for an architecture that can exploit many-core computer systems to the end of silicon
- It adopts a systematic model of concurrency, **SVP**, implemented as instructions in a base processors' ISA – SVP was developed in the EU FP6 **AETHER** project
- This approach has enormous potential but is disruptive and requires a new infrastructure of tools as it effectively implements an OS kernel in silicon
- There are large benefits from this approach as compilers need only capture abstract concurrency and not map and schedule it… this is done by the SVP implementation
- SVP separates the concerns of programming and concurrency engineering and opens the door for successful parallelising compilers.
- Apple-CORE will exploit concurrency in applications using **sequential, data-parallel** and **functional** languages
- Another major advantage of the SVP approach is that it provides both **backward** and **future binary compatibility** to the extended processor's ISA
- Code compiled with the new tools is executable on an arbitrary numbers of processors enabling dynamic resource mapping to binary programs from a pool of processors
- SVP provides a data-driven or conservative scheduling of instructions in contrasts to TM and other "safe" approaches to concurrency, which are speculative
- Information from the HW scheduler can provide powerful mechanisms for the management of power by load balancing processors based on clock/frequency scaling

The objective of Apple-CORE therefore is the development aninfrastructure to evaluate the SVP model and provide opportunities to exploit the results of this research in a variety of markets, including embedded and commodity processors, and also high-performance applications. In particular, the binary compatibility provides a unique opportunity to make an impact on commodity processors in Europe.

SVP ISA extensions manage hierarchical families of blocking threads



Advantages of blocking

- gives more concurrency
- allows fine-grain scheduling
- tolerates latency in operations such as loading data from asynchronous memory

Disadvantages

- like dataflow it requires synchronising memory
- but this is cheap if reasonably bounded

SVP captures data dependencies - Plain-old C to µTC to Assembly example

- The simplest of loops… it contains absolutely no concurrency but is expressed as a family of threads
- SVP implements this by creating 64 very simple threads
- In the assembly code each thread adds its index ($L0) to the previous thread's *shared* variable ($D0) and sets its own *shared* variable ($S0) - *3 registers only in each thread's context*
- As threads block no thread can execute its one instruction until the prior thread has executed its instruction

PoC

```
int sum = 0;
for(i = 0; i < 64; i++)
sum += i;
```

µTC

```
void thread sidx(shared int s)
  {index i; s+=i};

family f; int sum = 0;
create(f;;0;64;1;)sidx(sum);
sync(f);
```

Assembly

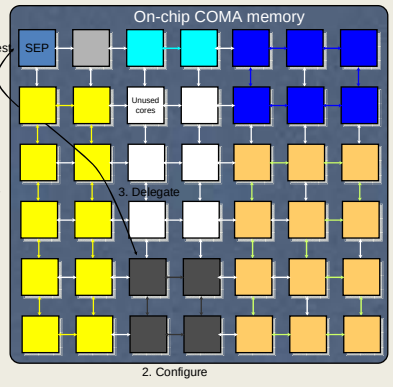
```
sidx: add $S0, $D0, $L0


main: allocate $L0
   mv $L1, 0
   setregs $L0, 1
   setlimit $L0,64
   cred $L2, sidx
   mv $L2, $31
```

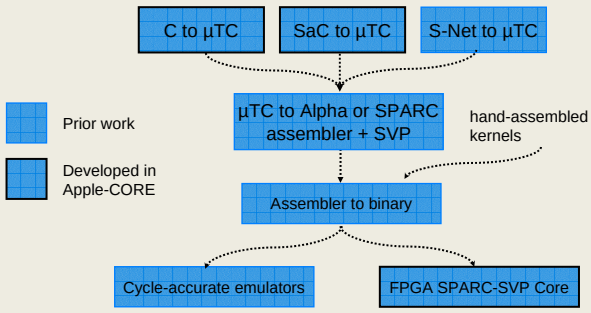
Dynamic management of processor resources in SVP Microgrids

SVP Microgrid

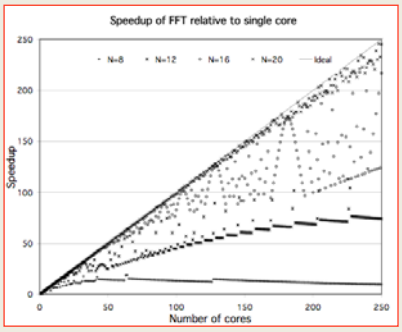
1. Single-address space with coherent cache-only memory (COMA)
2. A reconfigurable grid of SVP processors where clusters are configured into rings for delegating units of work (a family and all subordinate families)
3. An on-chip packet network for delegation, resource management & configuration



Apple-CORE toolchain



Emulation results



Partners

University of Amsterdam – coordinator (NL)
University of Hertfordshire (GB)
Institute of Information Theory and Automation (CZ)
University of Ioannina (GR)
Associated Compiler Experts (NL)
Gaisler Research (S)

Workpackages

1. Project management and dissemination
2. Application selection and evaluation
3. Parallelising C compiler
4. SAC data-parallel compiler
5. Emulation environment and operating system
6. Soft-core prototype

Apple-CORE is an EC-funded collaborative R&D project (contract no. FP7-ICT-215216). It was launched on November 1, 2007 and will run for 36 months. Apple-CORE is a spin-off project of the EC-funded FET-ACA project AETHER. Contact: Chris JESSHOPE, the project coordinator Email: info@apple-core.info.