Hardware support for online resources management

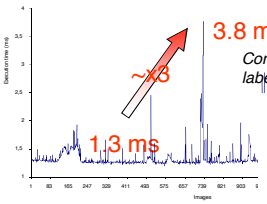
Raphaël David

CEA LIST, Embedded Computing Lab

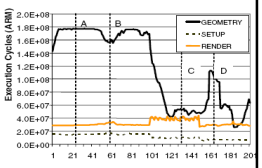
MPSOC 2011

Embedded Applications trends

- Applications are more and more dynamic
  - Dynamic control flow
  - Data-dependent processing
- Systems become less and less predictible
  - Variability
  - Defects
  - Aging

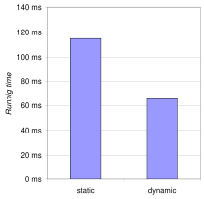


Connected-component labeling algorithm



Graphic application [Bren C. Mochock et al. DAC 2006]

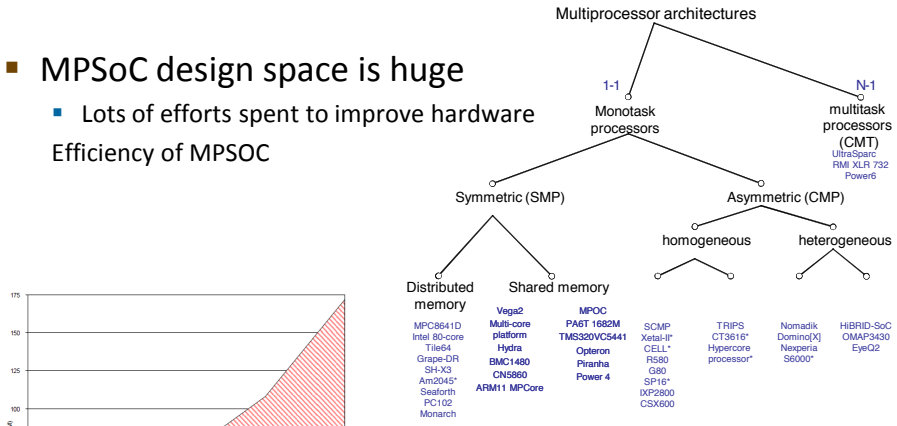
- Dynamic Load balancing is needed to improve processor utilization rate

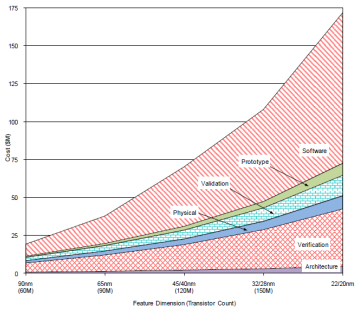


Connected component labeling execution time for a parallelization on 8 processors (128 tasks)

MPSOC Design Space

- MPSoC design space is huge
  - Lots of efforts spent to improve hardware Efficiency of MPSOC

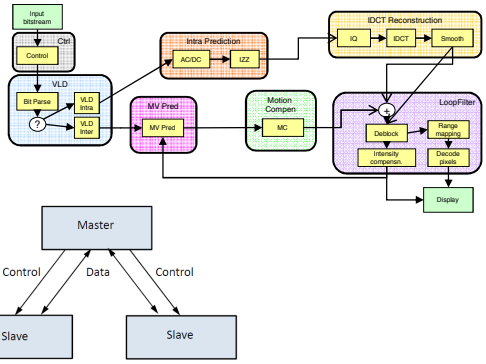


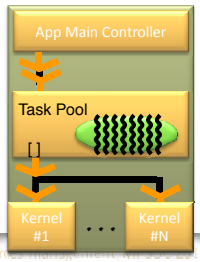


- But SW becomes predominant

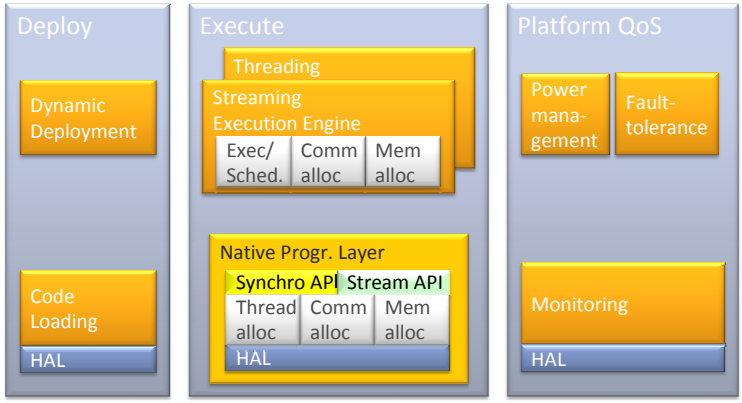
Programming Multi-threaded Applications

- Software Pipelining
- Master / Slaves
- Work Pool / Workers





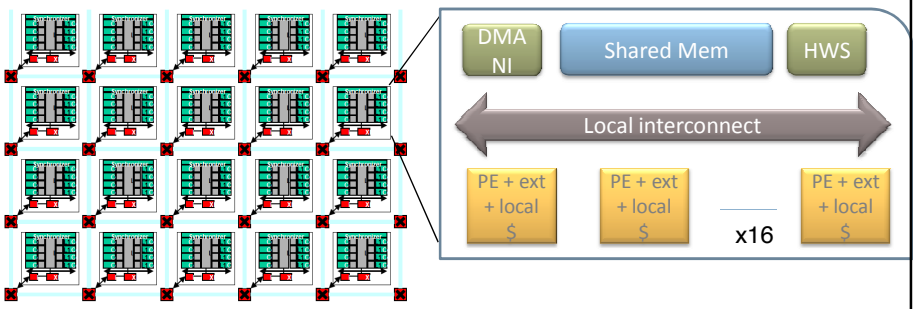
Software Runtime Architecture for massively parallel architecture



- Programming model free runtime SW
  - Full virtualization of HW resources
  - Multiple execution model (implicit-/explicit-MT, RTM for fine-grain task management, Master/slave, …)
  - Low-level services for power and error management

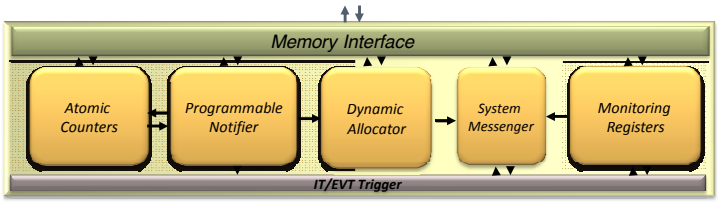
Hardware support for runtime software

- Studied in the field of the P2012 program with STM and CEA Leti to design a regular computing fabric capable to improve manufacturing yield
  - The goal is twofold
    - Provide flexibility through massive parallelism and scalable computing power
    - Deal with increasing manufacturability issues and energy constraints
  - Build on a set of SMP clusters connected through high performance NoC
    - Support the integration of dedicated accelerators
  - Does not assume a unique programming model
    - Several are models are made available to better suit programmers' needs
    - Rely on flexible runtime SW that provides an extensible framework to deploy and execute applications onto the fabric.



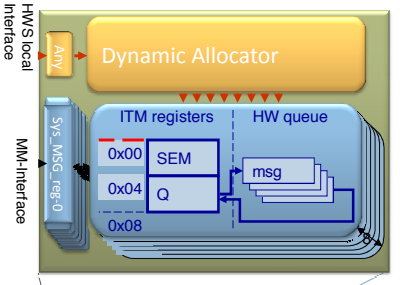
Hardware Synchronizer (HWS) Overview

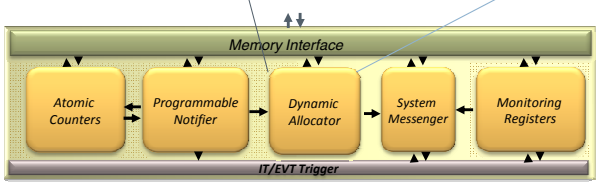
- Shared operator to optimize low-level system primitives
  - Synchronizations (Mutex, Semaphore, Barriers)
  - Signaling between cores
  - Centralized basic monitoring data (defects, power, ...)
- Simple low-cost hardware
  - Based on a set of registers with specific interface guaranteeing Atomic operations even for those having side-effects (e.g. post-inc/-dec)
  - Including HW support to ease system messaging and for automatic notification of events to prevent processor polling
  - Dynamic sorting of processors
    - for improved processor allocation
    - for automatic balancing of system activities when coupled to system messaging module
  - Dynamic task management support thanks to HW/SW virtualization of processors



HWS dynamic systems support

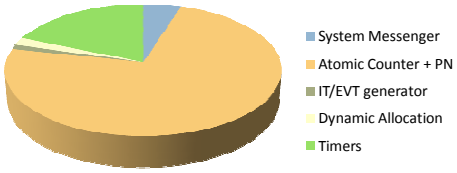
- Minimize runtime SW involvement in handling low level signaling even in the case of dynamic resources management approaches
  - Does not assume a static assignment of task
  - Synchronization events are link with tasks and task with processor, both association table being dynamically configured
    - e.g. Barriers operation does not rely on direct signaling between processor cores
- Dynamic allocation can be used to balanced the execution of system activities between all the processor cores.
- Dedicated HW allows to do some efficient message passing between cores with lows SW cost for shared Q management



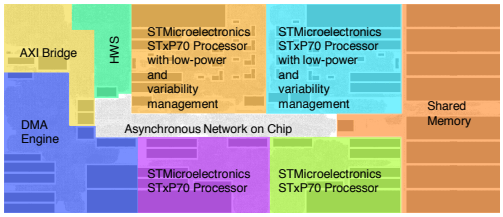


Implementation results

- Scalable solution with the amount of processor and Atomic Counters
  - HW Complexity mostly coming from Atomic Counters

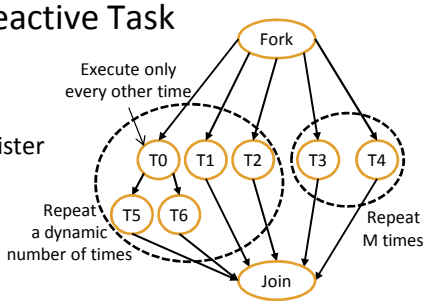


- Integrated in Locomotiv Chip designed in 32nm
  - Less than 1% cluster area while supporting complete system software stack
  - 0,06 mm² for a 500MHz operating frequency



HWS usage for scheduling dynamic applications

- Execution model based on Reactive Task Management
  - Optimized for fork-join operations
  - Based on a structure of tables to register forked tasks and activation conditions
  - HWS used for fast atomic task table accesses, iteration management and automatic signaling



Example of a typical task graph

- Used to schedule tasks of a VC1 decoder
  - Very fine grain task parallelization (average task duration < 600cycles)
  - Typical system loop no longer than 23 cycles
  - Impact lower than 3.5% on the application critical path

Conclusions

- Design of a low cost accelerator for the implementation of runtime software in parallel processors
  - Integration in the P2012 FPGA prototypes and the Locomotiv Chip
- Very low overhead demonstrated through the implementation of VC1 application
  - Content understanding application analysis ongoing
- Support for massively parallel devices ongoing
  - Initial solution based on the cooperation between multiple HWS
  - Extension to support dynamic load balancing between clusters
- Support of heterogeneous resources in progress

leti
LABORATOIRE D'ÉLECTRONIQUE ET DE TECHNOLOGIES DE L'INFORMATION
list
LABORATOIRE D'INTÉGRATION DES SYSTÈMES ET DES TECHNOLOGIES

Tank you for your attention !