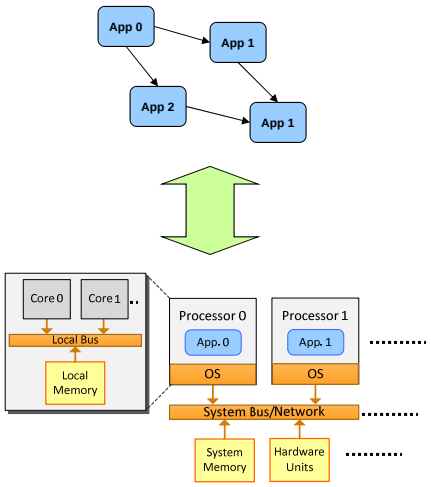
Models and Architectures for Heterogeneous System Design

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
`http://www.ece.utexas.edu/~gerstl`

Heterogeneous Computing

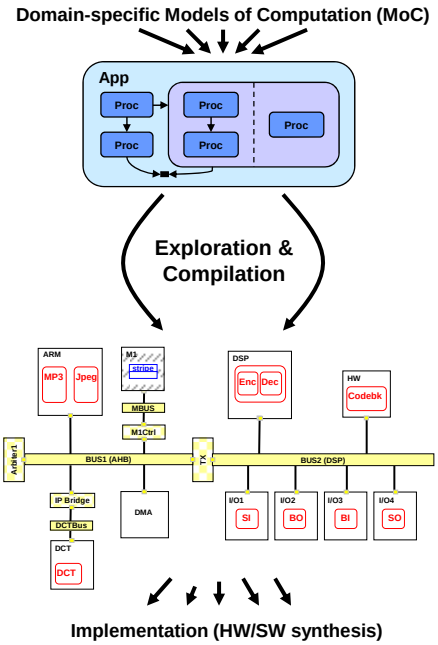
- **Embedded systems**
  - Costs of custom design
  - ➢ Increasingly programmable
- **General-purpose computing**
  - Power limits of scaling
  - ➢ Increasingly heterogeneous
- ➢ **Multi-Processor/Multi-Core Systems-on-Chip (MPCSoC)**
  - Mix of processors/cores?
  - Programming and mapping?
  - ➢ Application/architecture interactions



System-Level Design

- **From specification**
  - Application functionality
    - Parallel programming models
- **To implementation**
  - Heterogeneous MPCSoC
    - Hardware & software

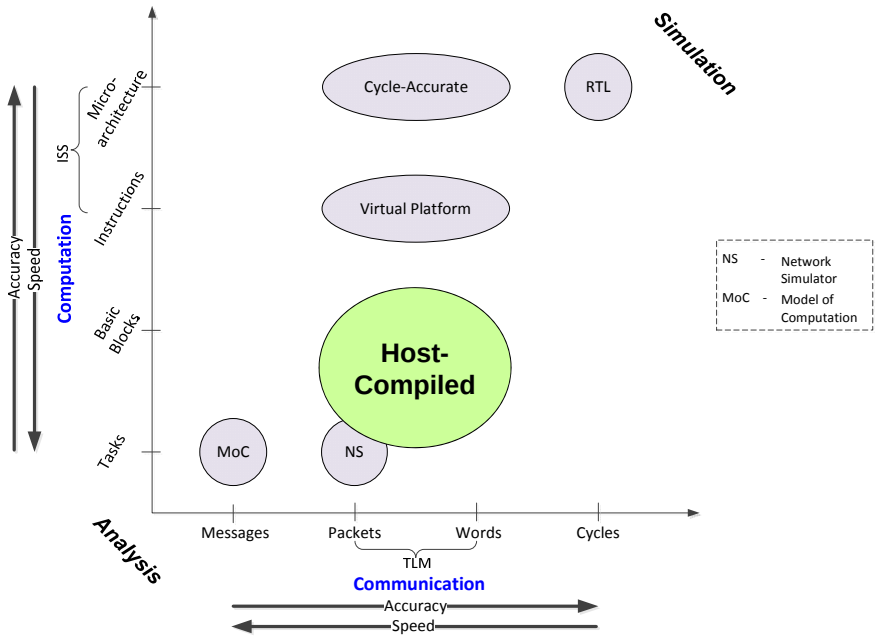
➢ **Design Automation**
  - Modeling
  - Synthesis
  - Architectures



Outline

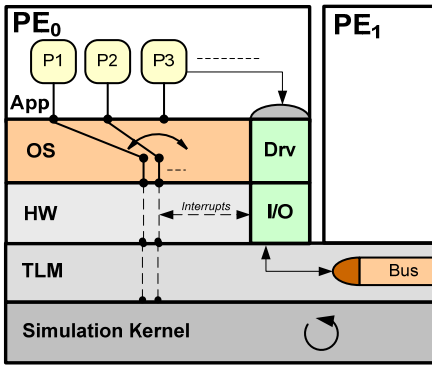
- **Modeling and simulation**
  - Host-compiled modeling
- **Synthesis**
  - Design-space exploration
  - System compilation
- **Architectures**
  - Domain-specific processors
  - Approximate Computing

System-Level Modeling Space



Host-Compiled Modeling

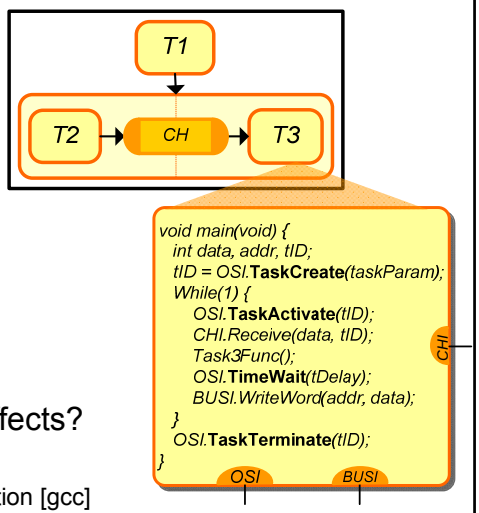
- **Coarse-grain system model [ASPDAC'12,RSP'10]**
  - Source-level application model
  - Abstract OS and processor models
  - Transaction-level model (TLM) backplane
  - C-based discrete-event simulation kernel [SpecC,SystemC]



➢ **Fast and accurate full-system simulation**
  ➢ Speed vs. accuracy tradeoffs

Application Model

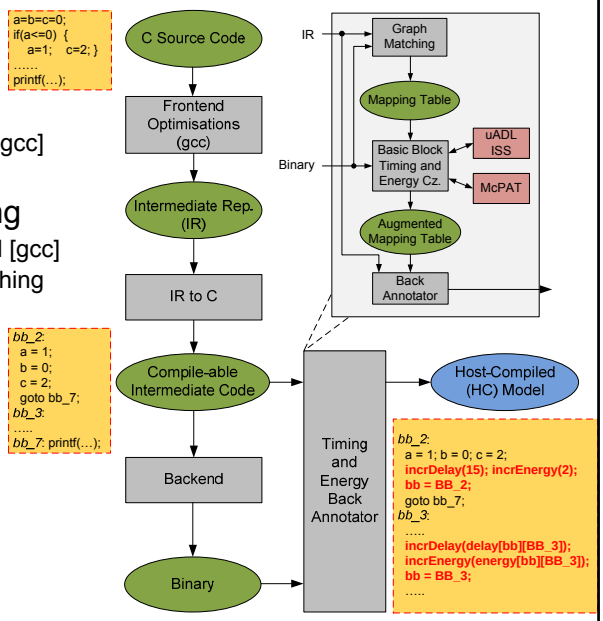
- **Application source code**
  - C-based task/process model
    - Parallel programming model
    - Canonical API
  - Communication primitives
    - IPC channel library



- **Timing and energy model**
  - Granularity?
  - Compiler optimizations?
  - Dynamic micro-architecture effects?
  - ➢ Basic-block level annotation
    - ➢ Compile to intermediate representation [gcc]
    - ➢ Block-level timing and energy estimates [ISS + McPAT]
    - ➢ Hybrid simulation (static annotation + cache, branch predictor models)

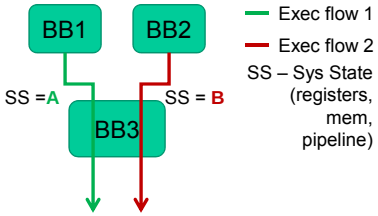
Retargetable Back-Annotation

- **Back-annotation flow**
  - Intermediate representation (IR)
    - Frontend optimizations [gcc]
    - IR to C conversion
  - Target binary matching
    - Cross-compiler backend [gcc]
    - Control-flow graph matching
  - Timing and power estimation
    - Micro-architecture description language (uADL) or RTL
    - Cycle-accurate timing
    - Reference power model [McPAT]
  - Back-annotation
    - IR basic block level

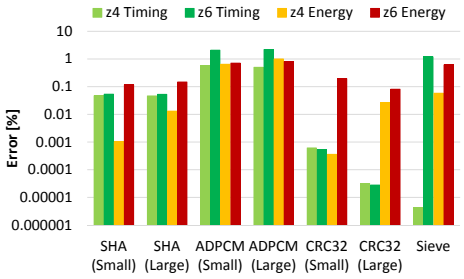


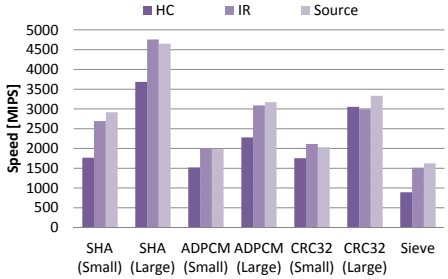
Power and Performance Estimation

- **Pairwise block characterization**
  - Path-dependent metrics
    - Timing & energy
  - Over all immediate predecessors



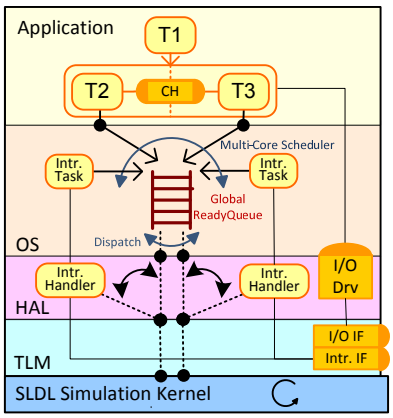
➢ **Close to cycle-accurate at source-level speeds**
  - Single- (z4) and dual-issue (z6) PowerPC [MiBench]
  - ➢ >98% timing and energy accuracy @ 2000 MIPS

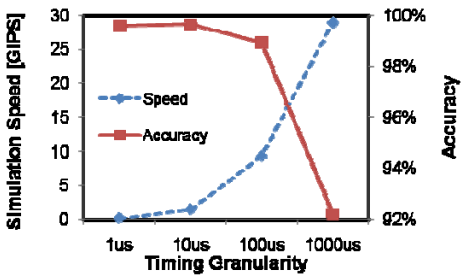




Multi-Core Processor Model

- **Layered processor model [DATE'11,TODAES'10]**
  - Abstract RTOS model
  - Hardware abstraction layer
  - TLM backplane

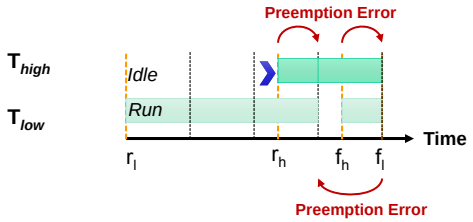




➢ **Full-system simulation**
  - Heterogeneous MPCSoC
  - 600 MIPS (≈ real time)
  - >97% accuracy

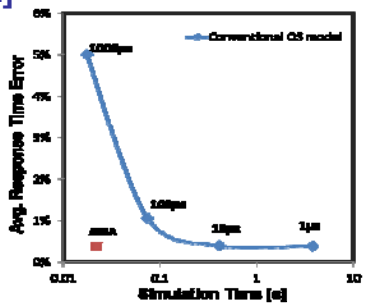
Automatic Timing Granularity Adjustment

- **Errors in discrete preemption models [ESL'12]**



- Potentially large preemption errors
  - Not bounded by simulation granularity

➢ **ATGA-based OS model [ASPDAC'12]**
- Observe system state to predict preemption points
- Dynamically and optimally adjust timing model
- ➢ Full-system simulation at 900 MIPS w/ 96% accuracy



Ongoing Modeling Work

- **Dynamic architecture features**
  - Multi-core memory hierarchy, caches [ESLSyn'13]
  - Microarchitecture effects (branch predictors, …)
- **Parallelizing the simulation**
  - Parallel SLDL simulation kernels [ASPDAC'11]
  - Parallelizing the model itself
- **Modeling of implementation effects**
  - Performance, energy, reliability, power, thermal (PERPT)
  - Application to other processor models (GPUs)
  - Integration with network simulation

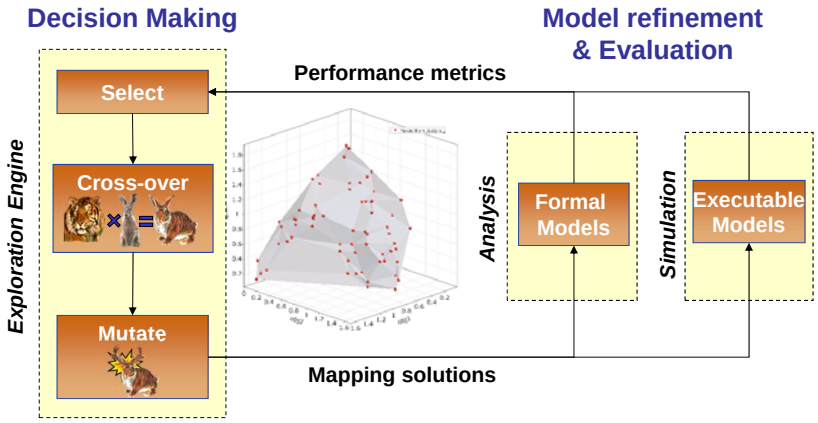
Outline

✓ Modeling and simulation
  ✓ Host-compiled modeling

- **Synthesis**
  - Design-space exploration
  - System compilation

- Architectures
  - Domain-specific processors
  - Approximate computing

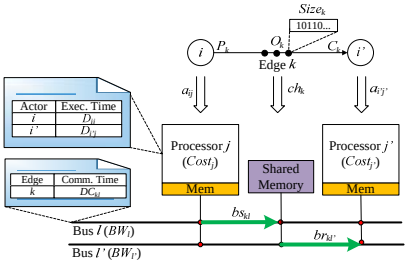
Design Space Exploration



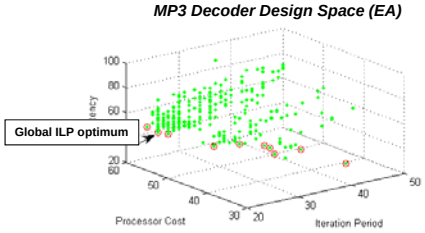
- **Electronic system-level synthesis [TCAD'10]**
  - From formal dataflow models to MPCSoC architectures
  - Genetic algorithms + host-compiled models [T-HiPEAC'11]
    - Hierarchy of layered models for successive design space pruning

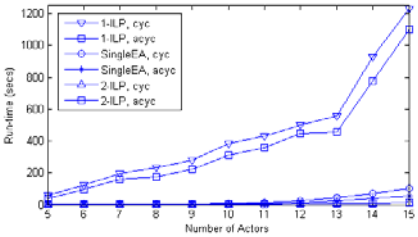
Dataflow Mapping

- **Streaming real-time applications (Synchronous Dataflow, SDF)**
  - Heterogeneous multi-processor partitioning & scheduling
  - Computation & communication
  - Pipelined latency & throughput
- **Mapping heuristics [JSPS'12]**
  - Globally optimal ILP formulation (1-ILP)
  - Partitioning ILP + scheduling ILP (2-ILP)
  - Evolutionary algorithm + scheduling ILP / list scheduler (EA)



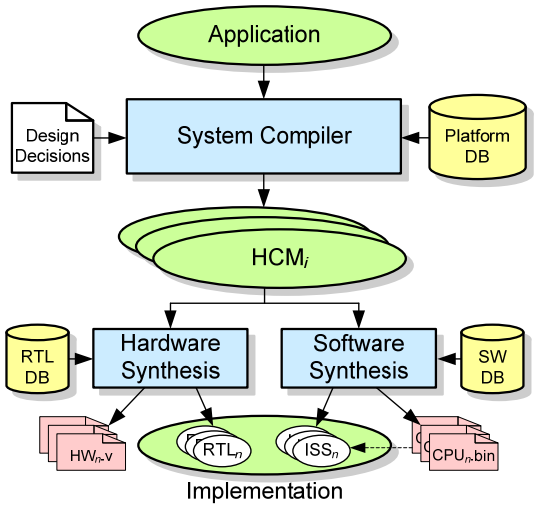
MP3 Decoder Design Space (EA)

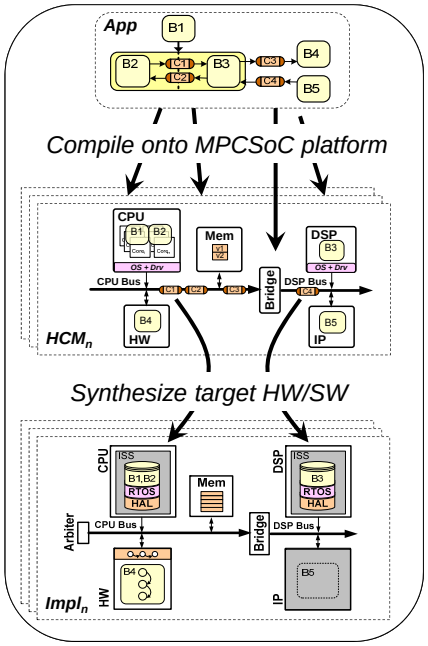




System Compilation

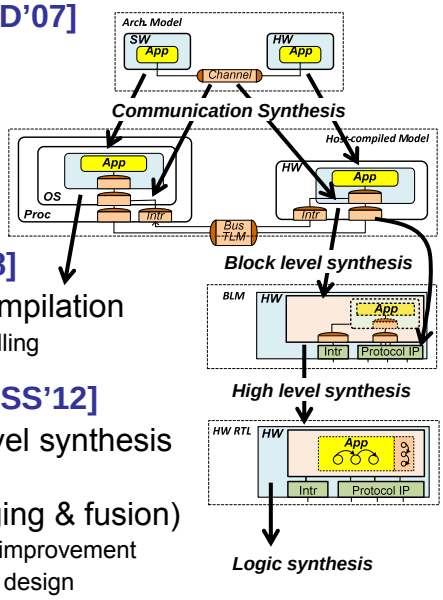
- **System-on-Chip Environment (SCE) [JES'08]**





System Compiler Backend

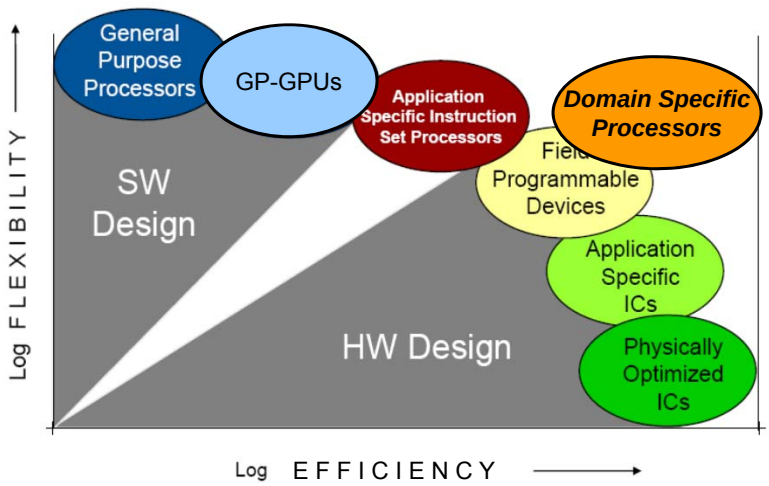
- **Communication synthesis [TCAD'07]**
  - From dataflow primitives
    - Queues, message-passing
  - To bus transactions
    - Protocol stacks over heterogeneous architectures
- **Software synthesis [ASPDAC'08]**
  - Code gen. + OS targeting + compilation
    - Drivers, interrupt handlers, task handling
- **Hardware synthesis [CODES+ISSS'12]**
  - Transactor synthesis + high-level synthesis
    - Custom communication interfaces
  - Transactor optimizations (merging & fusion)
    - Average 20% latency and 70% area improvement
    - 16% faster in similar area as manual design



Outline

- ✓ **Modeling and simulation**
  - ✓ Host-compiled modeling
- ✓ **Synthesis**
  - ✓ Design-space exploration
  - ✓ System compilation
- **Architectures**
  - Domain-specific processors
  - Approximate computing

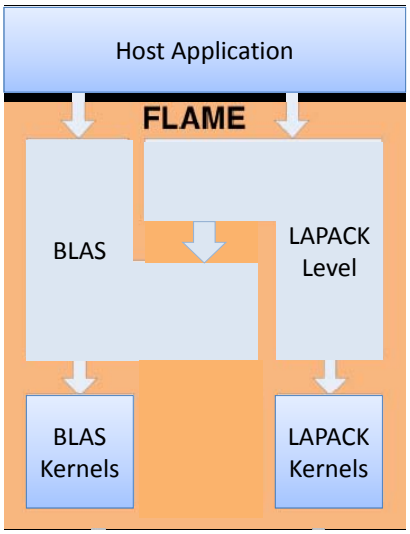
Algorithm/Architecture Co-Design



Adapted from: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

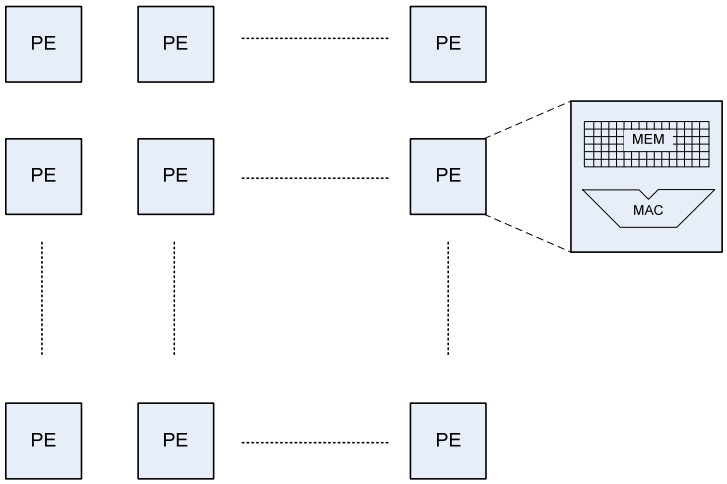
Linear Algebra (w/ R. van de Geijn, CS)

- **Layered libraries [FLAME]**
  - Linear Algebra Package (LAPACK) level
    - Cholesky and QR factorization
  - Basic Linear Algebra Subroutines (BLAS)
    - Matrix-matrix and matrix-vector operations
    - General matrix-matrix multiplication (GEMM)
  - Inner kernels
    - Hand-optimized assembly
- ➢ **Key to many applications**
  - ➢ High-performance computing
  - ➢ Embedded computing



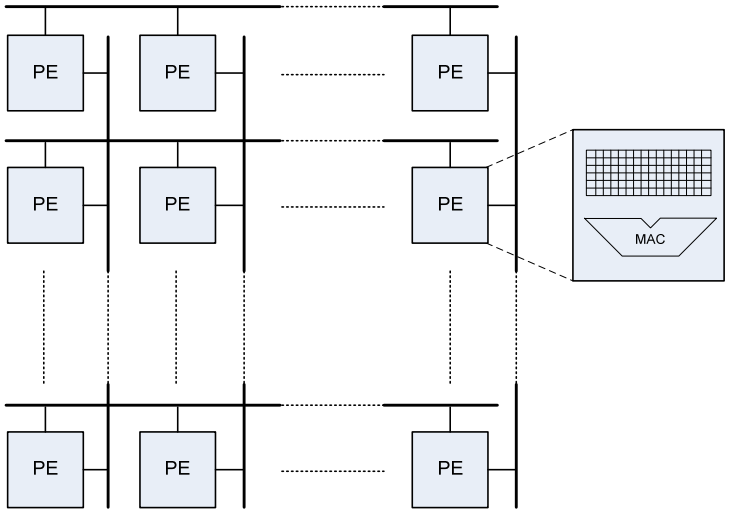
Linear Algebra Fabric

- **Parallelism and locality**

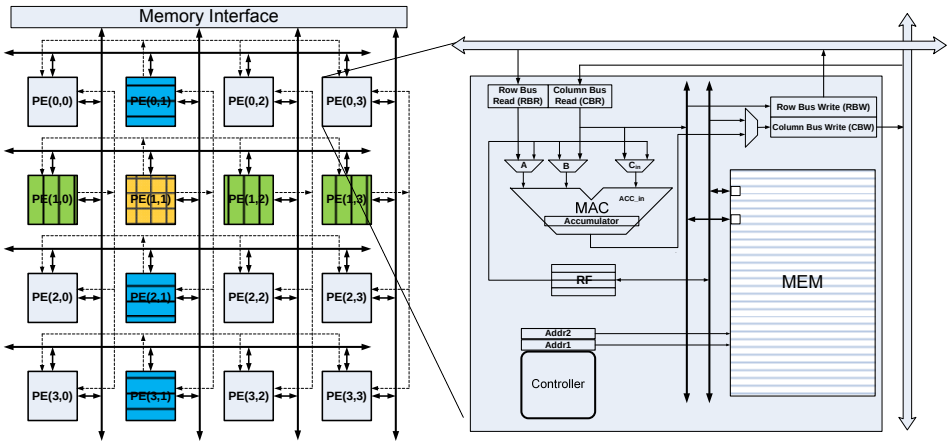


Linear Algebra Fabric

- **Broadcast nature of collective communications**



A Linear Algebra Core (LAC)



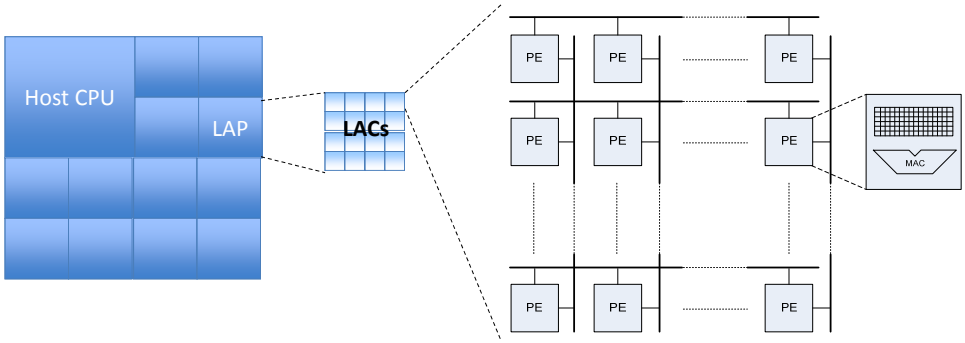
- **Scalable 2-D array of $n \times n$ processing elements (PEs) [ASAP'11]**
  - Specialized floating-point units w/ 1-cycle MAC throughput
  - Broadcast busses (possibly pipelined)
  - Distributed memory architecture
  - Distributed, PE-local micro-coded control

Core-Level Implementation

- **4x4 LAC w/ 256kB of local SRAM @ 1GHz**
  - Running GEMM
    - FMAC alone are 120 GFLOPS/W (single) or 60 GFLOPS/W (double)

	$\frac{W}{mm^2}$	$\frac{GFLOPS}{mm^2}$	$\frac{GFLOPS}{W}$	Utilization
Cell SPE (SP)	0.4	6.4	16	83%
NVidia GTX480 SM (SP)	0.5	3.8	7.0	58%
NVidia GTX480 SM (DP)	0.5	1.7	3.4	58%
Intel Core (DP)	0.5	0.4	0.9	95%
ClearSpeed (DP)	0.02	0.3	13	80%
LAC (SP)	**0.2**	**20**	**104**	**95+%**
LAC (DP)	**0.3**	**16**	**47**	**95+%**

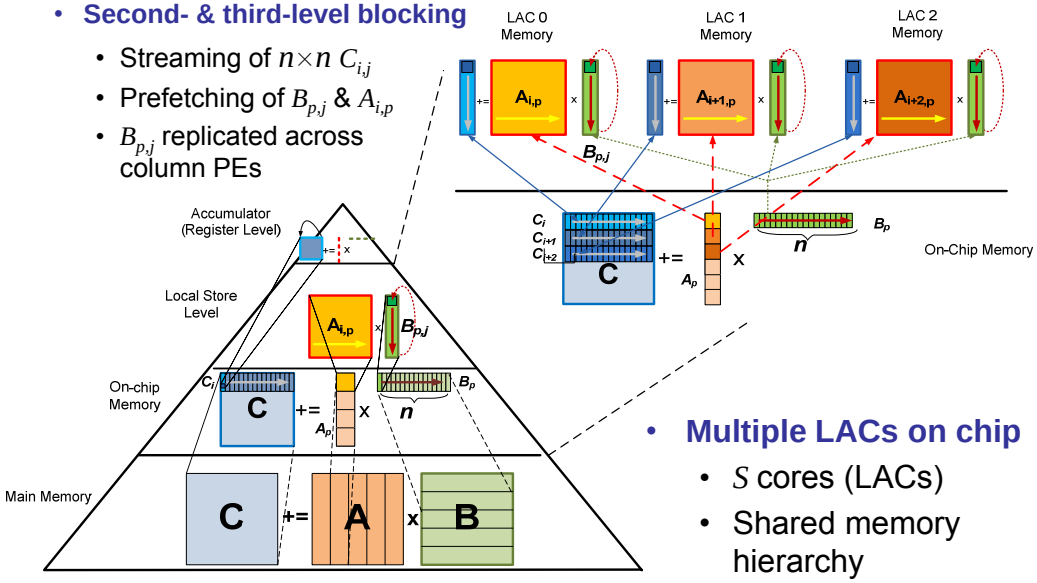
Linear Algebra Processor (LAP)



- **Multiple Linear Algebra Cores [TC'12]**
  - Fine- and coarse-grain parallelism
  - System integration
    - Memory hierarchy via shared memory interface
    - Host applications and libraries

Mapping GEMM to the LAP

- **Second- & third-level blocking**
  - Streaming of $n \times n$ $C_{i,j}$
  - Prefetching of $B_{p,j}$ & $A_{i,p}$
  - $B_{p,j}$ replicated across column PEs



- **Multiple LACs on chip**
  - S cores (LACs)
  - Shared memory hierarchy

LAP Summary and Comparison

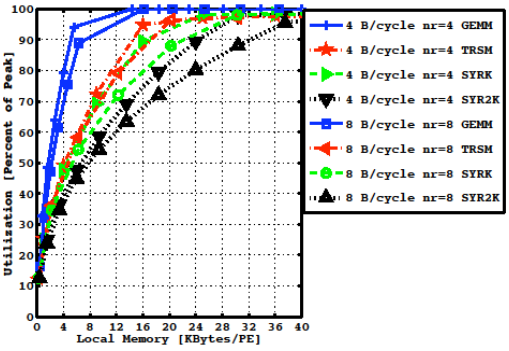
- **45nm scaled power/performance @ 1.4GHz**
  - Equivalent throughput (# of cores), running GEMM

	GFLOPS	W/mm²	GFLOPS/mm²	GFLOPS/W	Utilization
Cell BE (SP)	200	0.3	1.5	5	88%
NVidia GTX480 SM (SP)	780	0.2	0.9	4.5	58%
NVidia GTX480 SM (DP)	390	0.2	0.4	2.2	58%
Intel Core-i7 960 (SP)	96	0.4	0.5	1.2	95%
Intel Core-i7 960 (DP)	48	0.4	0.25	0.6	95%
Altera Stratix IV (DP)	100	0.02	0.05	3.5	90+%
ClearSpeed CSX700 (DP)	75	0.02	0.2	12.5	78%
LAP (SP)	**1200**	**0.2**	**6-11**	**55**	**90+%**
LAP (DP)	**600**	**0.2**	**3-5**	**25**	**90+%**

- **15-core LAP chip**
  - SGEMM/DGEMM in 1200/600 GFLOPS
  - 90% utilization, 25W, 120mm²
  - 1-2 orders of magnitude improvement vs. CPUs/GPUs

LAP Generalization

- **Level-3 BLAS [ASAP'12]**
  - Triangular Solve with Mult. Right-hand side (TRSM)
  - Sym. Rank-K Up. (SYRK)
  - Sym. Rank-2K Up. (SYR2K)
  - Addition of (1 / *x*) unit
- **Linear solvers [ARITH'13]**
  - Cholesky, LU, QR factorization
  - Addition of (1 /√) unit
- **Signal processing [ASAP'13]**
  - FFT, Neural nets
  - Addition of memory interfaces

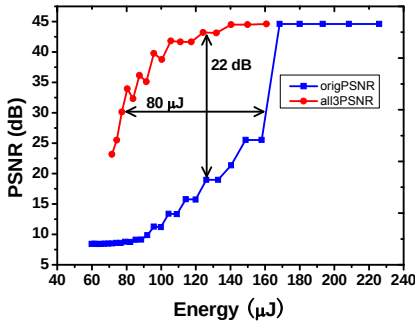


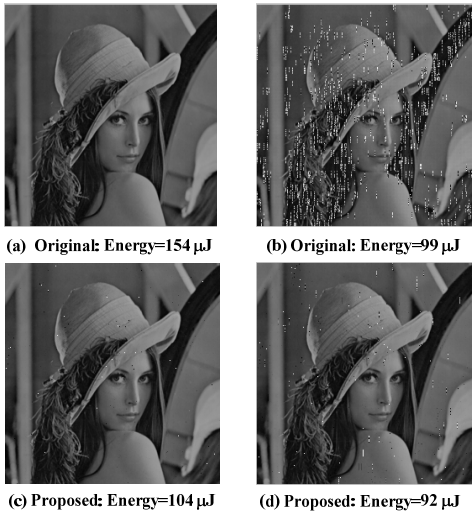
FFT	GFLOPS	GFLOPS/ mm²	GFLOPS /W	Utili.
Xeon E3	16	0.65	0.64	66%
ARM A9	0.6	0.09	2.13	60%
Cell	12	0.12	0.19	12%
Tesla	110	0.21	0.73	21%
LAP-FFT	**26.7**	**2.01**	**27.7**	**83%**

- **Minimal modifications for flexibility**
  - Coarse-grain programming model

Approximate Computing

- **Controlled timing error acceptance (w/ Prof. Orshansky)**
  - Statistical error treatment under scaled Vdd
  - Quality-energy profile shaping
- **2D-IDCT prototype [TCSVT'13]**
  - ~50% energy savings possible
  - < 3% area overhead





(a) Original: Energy=154 μJ (b) Original: Energy=99 μJ

(c) Proposed: Energy=104 μJ (d) Proposed: Energy=92 μJ

- **Adder synthesis [ICCAD'12]**
  - Family of quality-energy optimal arithmetic units

Summary & Conclusions

- **Heterogeneous system design**
  - Host-compiled modeling
    - Full-system simulation in close to real time with >90% accuracy
    - Automatic timing granularity adjustment
  - System compilation & synthesis
    - Design space exploration, mapping, scheduling
    - Hardware/software/interface synthesis
  - Domain-specific processor architectures
    - High efficiency, performance and flexibility
    - Linear algebra processor (LAP)
    - Approximate computing for digital signal processing