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]

- 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

- Potentially large preemption errors
  - Not bounded by simulation granularity

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)

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

- General Purpose Processors
- GP-GPUs
- Application Specific Instruction Set Processors
- Domain Specific Processors
- Flexible Programmable Devices
- Application Specific ICs
- Physically Optimized ICs


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

© 2013 A. Gerstlauer
Parallelism and locality

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)

<table>
<thead>
<tr>
<th></th>
<th>W (mm²)</th>
<th>GFLOPS (mm²)</th>
<th>GFLOPS (W)</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell SPE (SP)</td>
<td>0.4</td>
<td>6.4</td>
<td>16</td>
<td>83%</td>
</tr>
<tr>
<td>NVidia GTX480 SM (SP)</td>
<td>0.5</td>
<td>3.8</td>
<td>7.0</td>
<td>58%</td>
</tr>
<tr>
<td>NVidia GTX480 SM (DP)</td>
<td>0.5</td>
<td>1.7</td>
<td>3.4</td>
<td>58%</td>
</tr>
<tr>
<td>Intel Core (DP)</td>
<td>0.5</td>
<td>0.4</td>
<td>0.9</td>
<td>95%</td>
</tr>
<tr>
<td>ClearSpeed (DP)</td>
<td>0.02</td>
<td>0.3</td>
<td>13</td>
<td>80%</td>
</tr>
<tr>
<td>LAC (SP)</td>
<td>0.2</td>
<td>20</td>
<td>104</td>
<td>95+%</td>
</tr>
<tr>
<td>LAC (DP)</td>
<td>0.3</td>
<td>16</td>
<td>47</td>
<td>95+%</td>
</tr>
</tbody>
</table>
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 $n \times n C_{ij}$
  - 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

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

- **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 \(1 / \sqrt{x}\) unit

- **Signal processing [ASAP’13]**
  - FFT, Neural nets
    - Addition of memory interfaces

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

<table>
<thead>
<tr>
<th></th>
<th>FFT</th>
<th>GFLOPS</th>
<th>GFLOPS/mm²</th>
<th>GFLOPS/W</th>
<th>Util.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon E3</td>
<td>16</td>
<td>0.65</td>
<td>0.64</td>
<td>66%</td>
<td></td>
</tr>
<tr>
<td>ARM A9</td>
<td>0.6</td>
<td>0.09</td>
<td>2.13</td>
<td>60%</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td>12</td>
<td>0.12</td>
<td>0.19</td>
<td>12%</td>
<td></td>
</tr>
<tr>
<td>Tesla</td>
<td>110</td>
<td>0.21</td>
<td>0.73</td>
<td>21%</td>
<td></td>
</tr>
<tr>
<td>LAP-FFT</td>
<td>26.7</td>
<td>2.01</td>
<td>27.7</td>
<td>83%</td>
<td></td>
</tr>
</tbody>
</table>

© 2013 A. Gerstlauer
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

- **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