Power Management

Goal

• Basics of power dissipation and management

• Understand sources and sensitivities in GPU power dissipation
  - For on-chip GPUs vs. PCIe-based accelerators

• A look at two classes of power management techniques
Reading


---

*Introduction to Basics*
Background Reading

- [http://www.xbitlabs.com/articles/cpu/display/core-i5-2500t-2390t-i3-2100t-pentium-g620t.html](http://www.xbitlabs.com/articles/cpu/display/core-i5-2500t-2390t-i3-2100t-pentium-g620t.html)

- **Goal:** Understand
  - The sources of power dissipation in combinational and sequential circuits
  - Power vs. energy
  - Options for controlling power/energy dissipation

---

**Moore’s Law**

![Moore’s Law](image)

**Goal:** Sustain Performance Scaling

- Performance scaled with number of transistors
- **Dennard scaling***: power scaled with feature size

---

Where Does the Power Go in CMOS?

- **Dynamic Power Consumption**
  - Caused by switching transitions \( \rightarrow \) cost of switching state

- **Static Power Consumption**
  - Caused by leakage currents in the absence of any switching activity

- Power consumption per transistor changes with each technology generation
  - No longer reducing at the same rate
  - What happens to power density?

---

n-channel MOSFET

- \( V_{gs} < V_t \) transistor off - \( V_t \) is the threshold voltage
- \( V_{gs} > V_t \) transistor on

- Impact of threshold voltage
  - Higher \( V_t \), slower switching speed, lower leakage
  - Lower \( V_t \), faster switching speed, higher leakage

- Actual physics is more complex but this will do for now!
Charge as a State Variable

For computation we should be able to identify if each of the variable \((a,b,c,x,y)\) is in a ‘1’ or a ‘0’ state.

We could have used any physical quantity to do that
- Voltage
- Current
- Electron spin
- Orientation of magnetic field
- ........

All nodes have some capacitance associated with them

We choose voltage distinguish between a ‘0’ and a ‘1’.

Logic 1: Cap is charged
Logic 0: Cap is discharged

Abstracting Energy Behavior

- How can we abstract energy consumption for a digital device?
- Consider the energy cost of charge transfer
Switch from one state to another

To perform computation, we need to switch from one state to another.

Connect the cap to GND thorough an ON NMOS

Connect the cap to VCC thorough an ON PMOS

The logic dictates whether a node capacitor will be charged or discharged.

Power Vs. Energy

- Energy is a rate of expenditure of energy
  - One joule/sec = one watt
- Both profiles use the same amount of energy at different rates or power
Dynamic Power vs. Dynamic Energy

- Dynamic power: consider the rate at which switching (energy dissipation) takes place

\[ P_{\text{dynamic}} = \alpha \left( \frac{C_L}{2} \right) \cdot V_{\text{dd}} \cdot V_{\text{dd}} \cdot F \]

\[ \text{Delay} = k \cdot C \cdot \frac{V_{\text{dd}}}{(V_{\text{dd}} - V_t)^2} \]

Energy-Delay Interaction

- Delay decreases with supply voltage but energy/power increases

\[ P_{\text{dynamic}} = \alpha \left( \frac{C_L}{2} \right) \cdot V_{\text{dd}} \cdot V_{\text{dd}} \cdot F \]

\[ \text{Delay} = k \cdot C \cdot \frac{V_{\text{dd}}}{(V_{\text{dd}} - V_t)^2} \]
Technology scaling has caused transistors to become smaller and smaller. As a result, static power has become a substantial portion of the total power.

\[ P_{\text{static}} = V_{dd} \cdot I_{\text{static}} \]

Static Energy-Delay Interaction

- Static energy increases exponentially with decrease in threshold voltage
- Delay increases with threshold voltage
Higher Level Blocks

Temperature Dependence

- As temperature increases static power increases $^1$

$$P_{\text{static}} = V_{dd} \cdot N \cdot K_{\text{design}} \cdot I_{\text{leakage}}$$

Supply voltage $\rightarrow$ #Transistors $\rightarrow$ Technology Dependent $\rightarrow$ Normalized Leakage Current

$$I_{\text{leakage}} = F(\text{Temp})$$

$^1$ J. Butts and G. Sohi, "A Static Power Model for Architects, MICRO 2000
The World Today

- Yesterday → scaling to minimize time (max $F$)

\[ P_{\text{dynamic}} = \alpha \left( \frac{C_L}{2} \right) \cdot V_{dd} \cdot V_{dd} \cdot F \quad \text{Delay} = k \cdot C \cdot \frac{V_{dd}}{\left(V_{dd} - V_t\right)^2} \]

- Maximum performance (minimum time) is too expensive in terms of power
- Today: trade/balance performance for power efficiency

Technology Factors Affecting Power

- Transistor size
  - Affects capacitance ($C_L$)
- Rise times and fall times (delay)
  - Affects short circuit power (not in this course)
- Threshold voltage
  - Affects leakage power
- Temperature
  - Affects leakage power

- Switching activity
  - Frequency ($F$) and number of switching transistors ($\alpha$)

\[ P_{\text{dynamic}} = \alpha \left( \frac{C_L}{2} \right) \cdot V_{dd} \cdot V_{dd} \cdot F \quad \text{Delay} = k \cdot C \cdot \frac{V_{dd}}{\left(V_{dd} - V_t\right)^2} \]
Low Power Design: Options?

\[ P_{\text{dynamic}} = \alpha \left( \frac{C_L}{2} \right) \cdot V_{dd} \cdot V_{dd} \cdot F \]

\[ \text{Delay} = k \cdot C \cdot \frac{V_{dd}}{(V_{dd} - V_t)^2} \]

- **Reduce \( V_{dd} \)**
  - Increases gate delay
  - Note that this means it reduces the frequency of operation of the processor!

- **Compensate by reducing threshold voltage?**
  - Increase in leakage power

- **Reduce frequency**
  - Computation takes longer to complete
  - Consumes more energy (but less power) if voltage is not scaled

---

**Example**

<table>
<thead>
<tr>
<th>CPU P-state</th>
<th>Voltage (V)</th>
<th>Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW Only (Boost)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pb0</td>
<td>1</td>
<td>2400</td>
</tr>
<tr>
<td>Pb1</td>
<td>0.875</td>
<td>1800</td>
</tr>
<tr>
<td>SW-Visible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P0</td>
<td>0.825</td>
<td>1600</td>
</tr>
<tr>
<td>P1</td>
<td>0.812</td>
<td>1400</td>
</tr>
<tr>
<td>P2</td>
<td>0.787</td>
<td>1300</td>
</tr>
<tr>
<td>P3</td>
<td>0.762</td>
<td>1100</td>
</tr>
<tr>
<td>P4</td>
<td>0.75</td>
<td>900</td>
</tr>
</tbody>
</table>

AMD Trinity A10-5800 APU: 100W TDP
Optimizing Power vs. Energy

Thermal envelopes → minimize peak power
Maximize battery life → minimize energy

Example:

What About Wires?

Lumped RC Model

\[ \frac{1}{2} C_{\text{line}} \]

\[ \frac{1}{2} C_{\text{line}} \]

\[ R_{\text{line}} = r \cdot l \]

\[ \tau = \frac{1}{2} r c \cdot l^2 \]

\[ C_{\text{line}} = c \cdot l \]

• We will not directly address delay or energy expended in the interconnect in this class
  - Simple architecture model: lump the energy/power with the source component
Power Management Basics

Parallelism and Power

- How much of the chip area is devoted to compute?
- Run many cores slower. Why does this reduce power?

Source: IBM
Source: forwardthinking.pcmag.com
The Power Wall

\[ P = \alpha C V_{dd}^2 f + V_{dd} I_{\text{leak}} \]

- Power per transistor scales with frequency but also scales with \( V_{dd} \)
  - Lower \( V_{dd} \) can be compensated for with increased pipelining to keep throughput constant
  - Power per transistor is not same as power per area → power density is the problem!
  - Multiple units can be run at lower frequencies to keep throughput constant, while saving power

What is the Problem?

- Based on scaling using Pentium-class cores
- While Moore’s Law continues, scaling phenomena have changed
- Power densities are increasing with each generation

Mukhopadhyay and Yalamanchili (2009)
What are my Options?

1. Better technology
   - Manufacturing
   - Better devices (FinFet)
   - New Devices → non-CMOS? → this is the future

2. Be more efficient – activity management
   - Clock gating – dynamic energy/power
   - Power gating – static energy/power
   - Power state management - both

3. Improved architecture
   - Simpler pipelines

4. Parallelism
Activity Management

Clock Gating

- Turn off clock to a block of logic
- Eliminate unnecessary transitions/activity
- Clock distribution power

Power Gating

- Turn off power to a block of logic, e.g., core
- No leakage

---

Multiple Voltage Frequency Domains

- Cores and ring in one DVFS domain
- Graphics unit in another DVFS domain
- Cores and portion of cache can be gated off

*From E. Rotem et. Al. HotChips 2011*
Processor Power States

• Performance States – P-states
  - Operate at different voltage/frequencies
    o Recall delay-voltage relationship
  - Lower voltage → lower leakage
  - Lower frequency → lower power (not the same as energy!)
  - Lower frequency → longer execution time

• Idle States - C-states
  - Sleep states
  - Differ is how much state is saved

• SW or HW managed transitions between states!

Example of P-states

AMD Trinity A10-5800 APU: 100W TDP

<table>
<thead>
<tr>
<th></th>
<th>CPU P-state</th>
<th>Voltage (V)</th>
<th>Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW Only (Boost)</td>
<td>Pb0</td>
<td>1</td>
<td>2400</td>
</tr>
<tr>
<td></td>
<td>Pb1</td>
<td>0.875</td>
<td>1800</td>
</tr>
<tr>
<td>SW-Visible</td>
<td>P0</td>
<td>0.825</td>
<td>1600</td>
</tr>
<tr>
<td></td>
<td>P1</td>
<td>0.812</td>
<td>1400</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>0.787</td>
<td>1300</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>0.762</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>0.75</td>
<td>900</td>
</tr>
</tbody>
</table>

• Changing Power States is not free

(33)
Example of P-states

<table>
<thead>
<tr>
<th>Segment</th>
<th>State</th>
<th>CPU Core Frequency</th>
<th>Processor Graphics Core Frequency</th>
<th>Thermal Design Power</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme Edition (X)</td>
<td>HFM</td>
<td>3.5 GHz up to 3.5 GHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>55</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td></td>
<td>LFM</td>
<td>800 MHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>36</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td>Quad Core SV</td>
<td>HFM</td>
<td>2.2 GHz up to 3.4 GHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>45</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td></td>
<td>LFM</td>
<td>800 MHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>33</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td>Dual Core SD</td>
<td>HFM</td>
<td>3.5 GHz up to 3.4 GHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>35</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td></td>
<td>LFM</td>
<td>800 MHz</td>
<td>650 MHz up to 1300 MHz</td>
<td>26</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td>Low Voltage</td>
<td>HFM</td>
<td>2.1 GHz up to 3.2 GHz</td>
<td>500 MHz up to 1100 MHz</td>
<td>25</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td></td>
<td>LFM</td>
<td>800 MHz</td>
<td>500 MHz up to 1100 MHz</td>
<td>12</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td>Ultra Low Voltage</td>
<td>HFM</td>
<td>1.4 GHz up to 2.7 GHz</td>
<td>350 MHz up to 1000 MHz</td>
<td>17</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
<tr>
<td></td>
<td>LFM</td>
<td>800 MHz</td>
<td>350 MHz up to 1000 MHz</td>
<td>10</td>
<td>W</td>
<td>1, 2, 7</td>
</tr>
</tbody>
</table>


Management Knobs

- Each core can be in any one of a multiple of states
- How do I decide what state to set each core?
  - Who decides? HW? SW?
- How do I decide when I can turn off a core?
- What am I saving? Static energy or dynamic energy?
Power Management

• Software controlled power management
  ❖ Optimize power and/or energy
  ❖ Orchestrated by the operating system or application libraries
  ❖ Industry standard interfaces for power management
    o Advanced Configuration and Power Interface (ACPI)
      ▪ https://www.acpica.org/
      ▪ http://www.acpi.info/

• Hardware power management
  ❖ Optimized power/energy
  ❖ Failsafe operation, e.g., protect against thermal emergencies

---

Boosting

• Exploit package physics
  ❖ Temperature changes on the order of milliseconds

• Use the thermal headroom

![Turbo boost region]

Max Power

TDP Power

10s of seconds

Low power – build up thermal credits
Power Gating

- Turn off components that are not being used
  - Lose all state information
- Costs of powering down
- Costs of powering up
- Smart shutdown
  - Models to guide decisions

Parallelism

- Concurrency + lower frequency $\rightarrow$ greater energy efficiency

Example

\[ P = \alpha CV^2df + V_{dd}I_{leak} \]
Simplify Core Design

AMD Bulldozer Core

- Support for branch prediction, schedulers, etc. consumes more energy per instruction
- Can fit many more simpler cores on a die

ARM A7 Core (arm.com)

Metrics

- Power efficiency
  - MIPS/watt
  - Ops/watt
- Energy efficiency
  - Joules/instruction
  - Joules/op
- Composite
  - Energy-delay product
  - Energy-delay$^2$

Why are these useful?
Thermal Issues

- Heat can cause damage to the chip
  - Need failsafe operation

- Thermal fields change the physical characteristics
  - Leakage current and therefore power increases
  - Delay increases
  - Device degradation becomes worse

- Cooling solution determines the permitted power dissipation
Thermal Design Power (TDP)

- This is the **maximum** power at which the part is designed to operate
  - Dictates the design of the cooling system
    - Max temperature $T_{\text{max}}$
  - Typically fixed by worst case workload
- Parts are typically operating below the TDP
- Opportunities for turbo mode?

Heat Sink Limits on Performance

- Thermal design power (TDP)
  - Determines the cooling solution & package limits
- Performance depends on effective utilization of this thermal headroom

![Diagram showing thermal headroom and workload](image-url)

**Boost power**

- Convert thermal headroom to higher performance through boosting
### State-of-the-art Heterogeneous processor

Multi-threaded CPU cores

Shared Northbridge → access to overlapping CPU/GPU physical address spaces

Accelerated processing unit (APU)

Graphics processing unit (GPU):
- 384 AMD Radeon™ cores

Many resources are shared between the CPU and GPU
- For example, memory hierarchy, power, and thermal capacity

---

• Significant rise in temperature of the idle component due to thermal coupling and pollution from the active components within a die

• CPU consumes thermal headroom more rapidly (4X faster) → GPU can sustain higher power boosts longer

Goal

- Goal:
  - Optimize energy efficiency under power and performance constraints in a heterogeneous processor

- Outline:
  - Problem
  - State-of-the-Art Power Management
  - HPC Application Characteristics and Frequency Sensitivity
  - Run-time Coordinated Energy Management
  - Results

---

State-of-the-art Heterogeneous processor

Shared Northbridge ➔ access to overlapping CPU/GPU physical address spaces

Accelerated processing unit (APU)

Many resources are shared between the CPU and GPU
- For example, memory hierarchy, power, and thermal capacity
- Coupled programming model → Offload compute intensive tasks to the GPU

CPU-GPU Phase behavior in an Exascale Proxy Application (Lulesh)

- CPU-GPU coupled execution → time-varying redistribution of compute intensity
- Energy efficient operation → coordinated distribution of power to CPU vs. GPU
- Coordinated power states → sensitivity of performance to CPU and GPU power state (frequency)
  - Need to characterize ROI: Return (performance) on investment (power)
Challenge: CPU-GPU Coupling effects

- Direct Performance Coupling
- Indirect Performance Coupling: Shared Resources

HPC applications have **uncompromising** performance requirements!
- Need more efficient energy management

State of the Art Power Management
State-of-the-art: Bi-directional application power management (BAPM)

- Power management algorithm
  1. Calculate digital estimate of power consumption
  2. Convert power to temperature
     - RC network model for heat transfer
  3. Assign new power budgets to TEs based on temperature headroom
  4. TEs locally control (boost) their own DVFS states to maximize performance

Chip is divided into BAPM-controlled thermal entities (TEs)

Power Management

Performance and energy efficiency depend on effective utilization of power and thermal headroom

- HW Boost states
- SW visible states

Convert thermal headroom to higher performance through boost

<table>
<thead>
<tr>
<th>CPU</th>
<th>SW-Visible</th>
<th>HW Only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P0</td>
<td>Pb0</td>
</tr>
<tr>
<td></td>
<td>P1</td>
<td>Pb1</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Pmin</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPU</th>
<th>SW-Visible</th>
<th>HW Only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Mediu</td>
<td></td>
</tr>
</tbody>
</table>

Instructions/cycle Time

Performance and energy efficiency depend on effective utilization of power and thermal headroom
Key observations

- Overall application performance is a function of both the CPU and the GPU
- State of the practice: Manage to thermal limits by locally boosting when power and thermal headroom are available \(\rightarrow\) **utilize all of the available headroom**
- Pitfall: boosting may not lead to proportional performance improvement \(\rightarrow\)**energy inefficient**
- Need a concept of performance sensitivity to power states

Application Characteristics
Some kernels are more sensitive to GPU frequency than others → more power efficient

- Some kernels are more tightly coupled to CPU’s performance
- Smaller kernels such as Comm have high overheads in launching and feeding the GPU
Sensitivity to Shared resource interference

Performance actually limited by GPU memory demand

Power management locally boosts CPU to highest DVFS states

miniMD – Neighbor kernel

Wasted energy → power inefficient
Need online estimates of sensitivity to interference

Computation and control divergence

Graph Algorithm – BFS
Control divergence
→ increased thread serialization
→ increased frequency sensitivity
Key Observations

- HPC applications exhibit varying degrees of CPU and GPU frequency sensitivities due to
  - Control divergence
  - Interference at shared resources
  - Performance coupling between CPU and GPU

- Efficient energy management requires metrics that can predict frequency sensitivity (power) in heterogeneous processors

- Sensitivity metrics drive the coordinated setting of CPU and GPU power states

---

Energy Management
Performance metrics for APU frequency sensitivity

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Correlation Coefficient to GPU FS (meas)</th>
<th>Correlation Coefficient to CPU FS (meas)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeightedALUBusy</td>
<td>ALUBusy weighted by GPU ClockBusy</td>
<td>0.85</td>
<td>-0.62</td>
</tr>
<tr>
<td>ALUInsts PTI</td>
<td>Compute instructions per thousand instructions.</td>
<td>0.78</td>
<td>-0.54</td>
</tr>
<tr>
<td>ALUBusy</td>
<td>The percentage of GPU time that ALU instructions are processed.</td>
<td>0.76</td>
<td>-0.54</td>
</tr>
<tr>
<td>ALU/FetchRatio</td>
<td>The ratio of ALU to fetch instructions. If the number of fetch instructions is 0, then 1 will be used instead.</td>
<td>0.57</td>
<td>-0.31</td>
</tr>
<tr>
<td>L2 cache miss/cycle</td>
<td>Level 2 cache miss rate to main memory for CPU.</td>
<td>0.13</td>
<td>-0.41</td>
</tr>
<tr>
<td>ALUPacking</td>
<td>The ALU vector packing efficiency (in percentage).</td>
<td>0.11</td>
<td>-0.22</td>
</tr>
<tr>
<td>GPU/ClockBusy</td>
<td>GPU utilization: Ratio of time when at least one of the SIMD units in the GPU is active compared to total execution time.</td>
<td>0.06</td>
<td>-0.13</td>
</tr>
<tr>
<td>FetchUnitBusy</td>
<td>The percentage of GPU time the fetch unit is active.</td>
<td>-0.28</td>
<td>-0.01</td>
</tr>
<tr>
<td>FetchUnitStalled</td>
<td>The % of GPU time main memory fetch/load unit is stalled.</td>
<td>-0.49</td>
<td>-0.15</td>
</tr>
<tr>
<td>WriteUnitStalled</td>
<td>The % of GPU time main memory write/store unit is stalled.</td>
<td>-0.51</td>
<td>0.12</td>
</tr>
<tr>
<td>Writes to memory PTI</td>
<td>Main memory writes per thousand instructions.</td>
<td>-0.60</td>
<td>-0.28</td>
</tr>
<tr>
<td>Fetch from memory PTI</td>
<td>Main memory reads per thousand instructions.</td>
<td>-0.67</td>
<td>-0.73</td>
</tr>
<tr>
<td>Global_MemUtil</td>
<td>Aggregated CPU-GPU memory bandwidth consumed during theoretical peak bandwidth.</td>
<td>-0.63</td>
<td>-0.56</td>
</tr>
<tr>
<td>ClockWeightedUPC</td>
<td>Retired macro-operations (includes all processor activity) per cycle weighted by each core's active clocks.</td>
<td>-0.83</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Linear regression model using the above metrics to compute measures of performance coupling.

**DynaCO: Run-time system for coordinated energy management**

- **DynaCo-1levelTh**: Lowest CPU DVFS-state limited to P2
- **DynaCo-multilevelTh**: Lowest CPU DVFS-state allowed to use up to Pmin based on degree of performance coupling.
Key observations

- Coordinated CPU-GPU execution
- Linear combination of three key high level performance metrics proposed to model APU frequency sensitivity behavior
- Run-time coordinated energy management scheme DynaCo to manage CPU and GPU DVFS states dynamically based on measured frequency sensitivities

---

Experimental Set-Up

- Trinity A10-5800 APU: 100W TDP
- CPU: Managed by HW or SW
- GPU: Managed by sending software messages through driver layer

<table>
<thead>
<tr>
<th>CPU P-state</th>
<th>Voltage (V)</th>
<th>Freq (MHz)</th>
<th>GPU P-state</th>
<th>Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW Only (Boost)</td>
<td>Pb0</td>
<td>1</td>
<td>2400</td>
<td>GPU-high</td>
</tr>
<tr>
<td></td>
<td>Pb1</td>
<td>0.875</td>
<td>1800</td>
<td>GPU-med</td>
</tr>
<tr>
<td>SW-Visible</td>
<td>P0</td>
<td>0.825</td>
<td>1600</td>
<td>GPU-low</td>
</tr>
<tr>
<td></td>
<td>P1</td>
<td>0.812</td>
<td>1400</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>0.787</td>
<td>1300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>0.762</td>
<td>1100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>0.75</td>
<td>900</td>
<td></td>
</tr>
</tbody>
</table>

DynaCo implemented as a run-time software policy overlaid on top of BAPM in real hardware
Benchmarks

<table>
<thead>
<tr>
<th>BM (Description)</th>
<th>Problem Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>miniMD</td>
<td>32 x 32 x 32 elements</td>
</tr>
<tr>
<td>miniFE</td>
<td>100 x 100 x 100 elements</td>
</tr>
<tr>
<td>Lulesh</td>
<td>100 x 100 x 100 elements</td>
</tr>
<tr>
<td>Sort</td>
<td>2,097,152 elements</td>
</tr>
<tr>
<td>Stencil2D</td>
<td>4,096 x 4,096 elements</td>
</tr>
<tr>
<td>S3D</td>
<td>SHOC default for integrated GPU</td>
</tr>
<tr>
<td>BFS</td>
<td>1,000,000 nodes</td>
</tr>
</tbody>
</table>

Average energy efficiency improvement of 24% and 30% with DynaCo-1levelTh and DynaCo-multilevelTh respectively
Execution time Impact

Average performance slow down of 0.78% and 1.61% with DynaCo-1levelTh and DynaCo-multilevelTh respectively.

Power Savings

Average power savings of 24% and 31% with DynaCo-1levelTh and DynaCo-multilevelTh respectively.
Conclusions

• Note effects of shared resource interference, control divergence and performance coupling on energy management for HPC applications

• Importance and scope of frequency sensitivity in characterizing energy behaviors in tightly coupled heterogeneous architecture

• Dynamic power shifting power to the entity that can best utilize it

Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes:

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

© 2013 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. SPEC is a registered trademark of the Standard Performance Evaluation Corporation (SPEC). Other names are for informational purposes only and may be trademarks of their respective owners.
Harmonia: Balancing Compute and Memory Power in High Performance GPUs

Indrani Paul$^{1,2}$, Wei Huang$^1$, Manish Arora$^{1,3}$, Sudhakar Yalamanchili

June 2015

1 AMD Research
2 Georgia Institute of Technology
3 University of California, San Diego

This is Indrani Paul’s presentation at ISCA15

Goal & Outline

• Goal:
  ◆ Optimize energy efficiency under power and performance constraints in a high performance GPU-memory system

• Outline:
  ◆ State-of-the-Art Power Management
  ◆ Hardware Balance
  ◆ Compute and Memory Bandwidth Sensitivity Analysis
  ◆ Harmonia: Two-level Coordinated Power Management
Post-Dennard Performance Scaling:

\[ \text{Perf} = \text{Power} \times \text{Efficiency} \]

W. J. Dally, Keynote IITC 2012

---

**Single-Core Era**

- **Enabled by:**
  - Moore’s Law
  - Voltage Scaling
- **Constrained by:**
  - Power
  - Complexity

**Multi-Core Era**

- **Enabled by:**
  - Moore’s Law
  - SMP architecture
- **Constrained by:**
  - Power
  - Parallel SW
  - Scalability

**Heterogeneous Systems Era**

- **Enabled by:**
  - Abundant data parallelism
  - Power efficient GPUs
- **Temporarily Constrained by:**
  - Programming models
  - Comm. overhead

---

Phil Rogers, “Heterogeneous System Architecture Overview,” Hotchips Tutorial – August 2013
© Copyright 2012 HSA Foundation. All Rights Reserved.

---

High performance GPU-Memory systems

- High performance GPU connected to high bandwidth memory
- Shared board level power and thermal capacity
  - Future heterogeneous systems will share die-level power budget →
    - Integrated on-die CPU-GPU-Memory (e.g. stacked 2.5D, 3D packages)
- Kernel off-load is the dominant programming model: CUDA and OpenCL

---
The State of Art Memory Management

Board level power breakdown in AMD Tahiti dGPU

DRAM can operate at multiple bus frequencies → Reduce power
- Lower frequency reduces switching power, background, PLL power, memory controller and phy power
- Lower frequency allows for lower voltage

Frequency scaling increases latency → Reduce performance
- Array access times do not change but longer intervals between array accesses (bus transfer rate)
- Beyond a certain bus bandwidth, memory latency no longer hidden through GPU thread-level parallelism

GPU-memory performance sensitivity

Compute and memory behavior are fundamentally performance coupled
Overall board power scales by 70% relative to minimum compute by scaling compute frequency and voltage and active CUs.

Overall board power scales by 10% relative to minimum memory bandwidth, by scaling memory bus frequency.

Memory voltage scaling would provide greater power savings.

---

Device Memory’s GPU card power across compute configurations at constant 264GB/s memory bandwidth.

MaxFlops’s GPU card power across memory bandwidth configurations at 32 CUs and 1GHz compute frequency.

---

Coordinate Power States to Achieve Balance.

How efficiently is energy used in the core or in memory system?

Relative energy costs of compute and memory access.

Relative ops/byte demand of application.
Key Insights

- HPC applications exhibit varying degrees of GPU and memory sensitivities
  - Active CUs
  - Compute DVFS
  - Memory bandwidth

- Compute and memory behavior are performance coupled

- Compute and memory behavior have significant implications on overall system level power

- Balance time and energy costs of compute and memory to improve energy efficiency with minimal performance loss ➔ Hardware balance ➔ Need for sensitivity metrics

---

**Compute and BW Sensitivity Analysis**

\[ \text{Kernel Occupancy} = f(WG_{\text{size}}, \text{VGPR}_{\text{threads}}, \text{SGPR}_{\text{threads}}, \text{LDS}_{\text{WG}}) \]

- **Kernel occupancy and latency hiding capability:**
  - Measure of concurrent execution and utilization of hardware resources: LDS, SGPRs and VGPRs
  - Higher the concurrency, more the number of waves in flight, more sensitivity to memory bandwidth

---

% VGPRs per wave
<table>
<thead>
<tr>
<th>%</th>
<th>Kernel Occupancy</th>
<th>Mem BW Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

85% (85)

86% (86)
Load imbalance: Branch divergence and kernel complexity

- Speeding up serial thread execution will shorten overall execution time

- Low divergence in large kernels can have significant impact

**Architectural clock domains:**

- Chip-scale global interactions between multiple clock domains can create non-obvious sensitivities.
- L2 cache (compute clock) and on-chip MC (memory clock) in different domains.
- Reducing compute frequency reduces L2-MC BW making memory intensive benchmark sensitive to compute frequency
GPU-Memory Coordination

System-scale optimization of hardware tunables: Maximize efficiency under performance constraints

How much memory bandwidth is needed?

What frequency and voltage should they be set to?

How many active CUs are needed?

Sensitivity Predictors

- Weighted linear regression model captures sensitivities
- Simple, effective and practical sensitivity predictors to determine hardware balance point
Key Insights

- Efficient power management requires metrics that can predict compute and bandwidth sensitivities in a high-performance GPU-memory system
- Many factors → Kernel occupancy, branch divergence, architectural clock domain crossings, data movement, compute and memory intensities
- Sensitivity metrics drive the coordinated setting of GPU and memory power states
- Two-level coordinated power management algorithm to achieve optimal hardware balance → Harmonia
Experimental Set-up

- AND HD7970 dGPU
- 450 possible hardware configurations

<table>
<thead>
<tr>
<th>CU</th>
<th>CU Freq (MHz)</th>
<th>Voltage (V)</th>
<th>MemBW (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>300</td>
<td>0.85</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>400</td>
<td>0.90</td>
<td>120</td>
</tr>
<tr>
<td>12</td>
<td>500</td>
<td>0.95</td>
<td>150</td>
</tr>
<tr>
<td>16</td>
<td>600</td>
<td>1.00</td>
<td>178</td>
</tr>
<tr>
<td>20</td>
<td>700</td>
<td>1.05</td>
<td>206</td>
</tr>
<tr>
<td>24</td>
<td>800</td>
<td>1.10</td>
<td>235</td>
</tr>
<tr>
<td>28</td>
<td>900</td>
<td>1.15</td>
<td>264</td>
</tr>
<tr>
<td>32</td>
<td>1000</td>
<td>1.17</td>
<td></td>
</tr>
</tbody>
</table>

Power measurement
- DAQ: NI PCIe-6353 (1000Sa/s) and on-chip heuristics
- Memory power includes DDR phy, channels and DRAM

Benchmark
- Exascale proxy apps (CoMD, XSBench, miniFE), Graph500, BPT
- Rodinia and SHOC

Metrics: Energy-Delay^2
- Harmonia implemented as a run-time software policy overlaid on top of default power management in real hardware

Energy Efficiency Results

- Average 12% energy efficiency savings across the entire GPU card (up to 36%)
  - 6% coming from CG predictors
  - Remaining 6% from FG tuning based on real time feedback
  - Time varying redistribution of compute and memory intensities
**Performance Impact**

- Average loss in performance by 0.36% (i.e. increase in run-time) (up to 3.6% maximum slow-down)
- Captures “Performance peaks” scaling behavior and tunes hardware to the “peak” configurations
  - BPT, XSBench, CFD

---

**Coordination of Power states**

- 12% power savings → 64% came from GPU and 36% came from memory tuning
  - Voltage was not scaled for memory, so opportunity is higher
  - Clock domain crossings and interconnect bandwidth limit compute DVFS opportunity
- Harmonia tracks intra-kernel and inter-kernel phase behavior
Conclusions

• Demonstrated compute and memory behavior are performance coupled → **Hardware balance point**

• Predicting **sensitivities to hardware tunables** for determining optimal hardware balance point
  - Role of concurrency, branch divergence and architectural clock domain crossings

• Coordinate GPU and memory power states across three hardware tunables (CU, core frequency, memory bandwidth)

• Concepts of hardware balance and coordinate power management are the key to managing **future die-stacked systems**

Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

© 2013 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. SPEC is a registered trademark of the Standard Performance Evaluation Corporation (SPEC). Other names are for informational purposes only and may be trademarks of their respective owners.