Fried Egg - Athlon XP1500+


International Roadmap for Semiconductors: 2003–2018

[Graphs showing cost performance predictions for hand-held, mid-performance, and high-performance over years 2003 to 2018]

Cost of High Power

- For high-end: Constantly increasing number of computers used, and the power that each consumes
  - Electricity is expensive: financially and environmentally
- For mobile: cannot keep up with

Electricity Usage

- Total power consumption of CPUs in PCs:
  1992: 160Mwatts (87M CPUs)
  2001: 9000Mwatts (500M CPUs)
- Data center:
  25000 sq ft., ~8000 servers: 2Mwatts

Source: Dataquest
**High power → Increasing Temperature**

- With increased temperature:
- Current (& thus speed) decreases exponentially
  - Diodes: \( I = I_s(e^{V/Vt} - 1) \), \( Vt = kT/q \), \( T=temperature \)
  - Resistors: \( R \propto T \), \( I = V/R \)
  - Rule of thumb: Speed ↓ 0.15% per 1°C
- But leakage current increases with temperature
  - Thermal runaway
  - ↑ temp ⇒ ↑ leakage ⇒ ↑ self-heating ⇒ ↑ temp
- Reliability decreases exponentially
  - \( T \uparrow 10-15°C \Rightarrow \) Chip lifetime ↓ 50%

**CPU Power Breakdown**

- Relative consumption of processor units varies by application
- Approaches to reducing power must also vary

Alpha 21464  PowerPC  StrongARM
Power Reduction: Current Approaches

- Voltage Scaling
- Clock Throttling/Gating
- Fetch Throttling

Energy vs. Power

- Decreasing power does not always mean decreasing energy
  - If power decrease can make process take more time, it may take more energy
- Power constrained vs. Energy constrained
  - e.g., Solar vs. Batteries

\[ P = \frac{dE}{dt} \]

CMOS Power

- Power consumption can be categorized:
  - Consumed during switching - dynamic
  - Consumed constantly - static

Dynamic + Short Circuit

Static

**Dynamic Power**

- \[ P_{DYNAMIC} = C_L \cdot N_{SW} \cdot V_{DD}^2 \cdot f \]
- \( C_L \) = load capacitance
- \( N_{SW} \) = avg switches/cycle

- Currently largest portion of consumption
- Reducing \( N_{SW} \):
  - Gating
  - Throttling

- Short circuit power <10% of dynamic power
- Consumed briefly when PMOS and NMOS are on simultaneously

---

**Static (leakage) power**

- Constant Leakage through NMOS when logic result is high
- Importance increasing as technology size decreases
- One solution: disable portions of \( \mu P \) when not needed

---

Local Clock Gating

- Disable clock to inactive units
- Can disable *global* clock if stalled
- Can disable local clocks when particular functional units are not in use
  - e.g.: may not have any multiply operations for unit to process

Example of Throttling

- Fetch Unit can watch branch decisions in pipeline and stop putting instructions into pipeline
- If fetch unit is throttled no power is consumed
- Speculative branches cause 16% to 105% instruction overhead
  - If fewer incorrect branches are fetched, can reduce power significantly with minor performance cost
    - Calculate a confidence level
    - Don't fetch uncertain branches
  - Alternatively: slow fetch relative to the number of instructions committed

Dynamic Power Reduction through Voltage Scaling

\[ P_{\text{DYNAMIC}} = C_L \cdot N_{SW} \cdot V_{DD}^2 \cdot f \]

- \( \delta = \text{delay}; \ \alpha, K \text{ are process-dependent} \)
- \( 1 < \alpha < 2 \)
- \( V_{DD} \downarrow \Rightarrow \delta \uparrow \quad \delta \uparrow \Rightarrow f \downarrow \)
- Decreasing \( V_{DD} \) results in reducing average dynamic power cubically
- Since time to complete increases, energy consumed is reduced only by square

Dynamic Voltage Scaling (DVS)

- Run-time algorithms
- Design processor for maximum performance
  - When maximum performance is not required, scale frequency and supply voltage appropriately
- DVS in Modern \( \mu P \):
  - Speedstep (Pentium IIIM, CeleronM, IV)
    - Controlled by operating system
  - Intelligent Energy Management (ARM \( \mu Ps \))
    - Processes submit performance requests to this software
    - The software decides what requests to honor, directs hardware


Power Issues in Multi-Many Core

Why Multi-Core?

- **Single Core**
  - **Performance** is unable to keep pace with the increasing transistor counts. Law of diminishing returns prevailing.
  - **Power usage and heat generation** increasing

- **Multi-Core**
  - **Performance** is increasing as the multi-cores can utilize Thread level parallelism
  - More power efficiency
  - Lower die temperature
  - Improves reliability and leakage

Based on Intel tests using the SPECint2000 and SPECfp2000 benchmarks
Dual Core
Cache Coherence

- Since one or more levels of private cache
- How to keep consistency across all the cores?

Cache Coherence problem

- Core 1
  - One or more levels of cache
  - Core 1 reads x = 15213
  - Core 1 changes x = 21660

- Core 2
  - One or more levels of cache
  - Core 2 reads x

- Core 3
  - One or more levels of cache

- Core 4
  - One or more levels of cache

Main memory

multi-core chip
Solutions to Cache Coherence Problem

- **Invalidation Protocol with Snooping**
  - Invalidation: If a core writes a data item, it sends invalidation requests
  - Snooping: All other cores snoop the data
- **Validation Protocol**
  - Broadcasts updated value

What happens when the number of cores increase?

Multi-Threading is the key in multi-core

Threads are a way for a program to fork (or split) itself into two or more simultaneously running tasks.
Power

- 1K Cores on a Die
- At the least 1W per core
- 1K Watts of power

Power efficiency is very critical in many-core architecture

From Multi—to Many-Cores

By 2015 100Billion Transistors on a 300mm²

The question is

Less number of cores with more area

OR

More number of cores lesser area

What Should be the tradeoff?

Anant Agarwal et al
**KILL Rule**

1% Increase in core area should result in at least 1% increase in core performance

- **Example**: MPEG2 Decode
- **Assumption**: 512 byte occupies 1% of the core area
- **IPC** is used as the performance metric
- One third instruction are load and store instructions
  \[
  \text{Core IPC} = \frac{1}{1+0.33m \times 300} = \frac{1}{1+mx100}
  \]
  where \(m\) is the data cache miss rate “m”
- For mpeg2 decode with 512 byte cache, \(m=0.25\)
- Core IPC = 0.04  For 100 cores the IPC = 4
- If the data cache increases to 2K bytes, \(m=0.05\)
- So new IPC for the entire chip = 17

**So Area 3% increase, performance increased by 325%**
**Kill Rule**

![Diagram](image)

In simple what does Kill Rule say?

- Can't follow existing single processor design approach
  - Increase cache size
  - Increase pipeline stages
  - Many Port Register files
- Simpler Core and Smaller caches
- Can't keep on increasing the number of cores without striking the right balance
- Power Efficiency and performance
## Kill Rule for Power

![Graph showing Whetstone with area increase and IPC/Watt increase metrics for different number of cores.]

<table>
<thead>
<tr>
<th>No. of cores</th>
<th>IPC/Watt Increase</th>
<th>IPC/Watt Increase</th>
<th>IPC/Watt Increase</th>
<th>IPC/Watt Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.02</td>
<td>0.015</td>
<td>0.015</td>
<td>0.03</td>
</tr>
<tr>
<td>96</td>
<td>0.02</td>
<td>0.015</td>
<td>0.015</td>
<td>0.03</td>
</tr>
<tr>
<td>89</td>
<td>0.02</td>
<td>0.015</td>
<td>0.015</td>
<td>0.03</td>
</tr>
<tr>
<td>78</td>
<td>0.02</td>
<td>0.015</td>
<td>0.015</td>
<td>0.03</td>
</tr>
<tr>
<td>62</td>
<td>0.02</td>
<td>0.015</td>
<td>0.015</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Area Increase: 4%, 8%, 14%, 25%
IPC/Watt Increase: 12%, 13%, 6%, 6%

## Performance of Many-Core Systems

- **Amdahl's law**
  - Parallel Speedup is limited by serial code in a program

- **Multi-Thread is vital.**
Fine Grain Power Management

Leakage power contributes to 40% of the total power in the microprocessors

- Voltage Scaling
- Frequency Scaling
- Multiple Vt
- Body Biasing
- Design style changes
  - Avoid Domino logic

Power Consumption for 300mm² after power management will be approximately 300W

---

Nothing is free

- Latency
  - 1 Microseconds sleep latency, 2 clock cycles wake up latency for a single microprocessor.
  - sleep transistor technique can reduce leakage by 37% at a 3%-4% performance reduction

- Asynchronous Interfaces
  - Different cores running at different frequency and voltages

---
On-Chip Network Power

- Networks are power hungry
  - Eg: 500mW of power at each node
- More number or cores, More number of nodes, So More power.

Memory

- 100’s of GB of memory bandwidth is required
- Memory bus consumes around 25W for 100GB/s memory bandwidth
- Need to reduce the bus lengths
- 3D integration of memory with processor is a potential solution
SPARC Processor -- ROCK

- 16 Core, 14mm²
- 32 Thread
- 32 Scout Thread
- 2.3 GHz, 396mm², 65nm
- Each core ~10W power
- No re-order buffer
- Out of order commit
- No crossbar network
- Cluster of 4 cores, each shares an I$ 32 Kb, 2 32 KB data caches (D$),
- 2 floating point unit for 4 cores
- So resource sharing, low power and area without affecting performance...But need to find it

ROCK – SPARC Processor

- 16 Core
- 4 clusters, 4 cores each
- 2.3 GHz, 65nm
- 396 mm²
- Scout Thread
- 2 integer execution unit/core
- 2 FPU/cluster
- 32KB I$ and two 32KB D$ /cluster
- Power 250 W
Main Highlight: Scout Threading

- Scout Thread
  - Prefetches code and data
  - Predicts and resolves branches
  - Retires instructions which are not dependent on the outcome of the stuck instruction
  - In simple out of order execution and out of order committing resulting in higher performance and throughput
  - Avoids power hungry CAM based Re-Order Buffers

- Deferred queue – 128 entry
  - Low power SRAM
- High Single thread performance

How does ROCK compare to Niagara

<table>
<thead>
<tr>
<th>ROCK</th>
<th>Niagara</th>
</tr>
</thead>
<tbody>
<tr>
<td>240 W</td>
<td>123 W</td>
</tr>
<tr>
<td>16 Cores</td>
<td>8 Cores</td>
</tr>
<tr>
<td>2 Integer execution units/core</td>
<td>2 Integer execution units/core</td>
</tr>
<tr>
<td>2 floating point unit /4 core</td>
<td>1 floating point unit/core</td>
</tr>
<tr>
<td>396 mm²</td>
<td>342 mm²</td>
</tr>
<tr>
<td>2 threads/core</td>
<td>8 threads/core</td>
</tr>
<tr>
<td>Scout threads</td>
<td>No scout threads</td>
</tr>
<tr>
<td>No Reorder buffer</td>
<td>Has Reorder Buffer</td>
</tr>
<tr>
<td>2 levels of cache sharing</td>
<td>1 level of cache sharing</td>
</tr>
<tr>
<td>Transactional Memory</td>
<td>Lock based synchronization</td>
</tr>
<tr>
<td>2.4GHz</td>
<td>1.4GHz</td>
</tr>
</tbody>
</table>