Riding the Technology Curve
Dec. 1, 1998

Topics
- Moore’s Law
- Are exponential problems intractable?
- Impact on real-world problems
- The verification challenge
Impact of Technology

It’s the Technology, Stupid!
- Computer science has ridden the wave

Things Aren’t Over Yet
- Technology will continue to progress along current growth curves
- For at least 10 more years
- Difficult technical challenges in doing so

Even Technologists Can’t Beat Laws of Physics
Moore’s Law

Gordon Moore

- Co-founded Intel in early 70’s
- Observed in 1972 that number of transistors / chip doubled ~ every 1.5 years
- Really a “trend” rather than a “law”

Exponential Growth Trends

- DRAM technology
  - Capacity 4X every 3 years
  - Speed 3X in 10 years
- Magnetic disk technology
  - Capacity 4X every 3 years
- Microprocessor Performance
  - SPEC performance 2X every 1.5 years
- Software complexity
  - Typical program sizes growing 1.5--2X per year
## Semiconductor Industry Forecast

**Semiconductor Industry Association, 1992 Technology Workshop**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature size</td>
<td>0.5</td>
<td>0.35</td>
<td>0.25</td>
<td>0.18</td>
<td>0.12</td>
<td>0.10</td>
</tr>
<tr>
<td>DRAM cap</td>
<td>16M</td>
<td>64M</td>
<td>256M</td>
<td>1G</td>
<td>4G</td>
<td>16G</td>
</tr>
<tr>
<td>Gates/chip</td>
<td>300K</td>
<td>800K</td>
<td>2M</td>
<td>5M</td>
<td>10M</td>
<td>20M</td>
</tr>
<tr>
<td>Chip cm$^2$</td>
<td>2.5</td>
<td>4.0</td>
<td>6.0</td>
<td>8.0</td>
<td>10.0</td>
<td>12.5</td>
</tr>
<tr>
<td>I/Os</td>
<td>500</td>
<td>750</td>
<td>1500</td>
<td>2000</td>
<td>3500</td>
<td>5000</td>
</tr>
<tr>
<td>off chip MHz</td>
<td>60</td>
<td>100</td>
<td>175</td>
<td>250</td>
<td>350</td>
<td>500</td>
</tr>
<tr>
<td>on chip MHz</td>
<td>120</td>
<td>200</td>
<td>350</td>
<td>500</td>
<td>700</td>
<td>1000</td>
</tr>
</tbody>
</table>
Impact of Moore’s Law

Moore’s Law

- Performance factors of systems built with integrated circuit technology follow exponential curve
- E.g., computer speed / memory capacities double every 1.5 years

Implications

- Computers 10 years from now will run 100 X faster
- Problems that appear intractable today will be straightforward
- Must not limit future planning with today’s technology

Example Application Domains

- Speech recognition
  - Will be routinely done with handheld devices
- Breaking secret codes
  - Need to use large enough encryption keys
Solving Hard Problems

Conventional Wisdom

- Exponential problems are intractable

Operation

- Assume problem of size $n$ requires $2^n$ steps
- Each step takes $k$ years on a Y2K computer

Y2K Computer Performance

- Start computation Jan. 1, 2000
- Keep running same machine until problem solved
- Would take $k 2^n$ years
Solving with a Y2K Computer

Problem Size (n) vs. CPU Years

Time per Operation
- second
- minute
- hour
- day
- week
- year

Graph showing the relationship between problem size and CPU years, with time per operation indicated by different markers.
Moore’s Law Computer

Operation

- Start computing on Jan. 1, 2000
- Keep upgrading machine being used
- In year $y$, would have performance $1.587^y$ relative to Y2K machine

Performance

- After $y$ years of operation, would have performed as much computation as Y2K machine would do in time:

$$\int_0^y 1.587^x \, dx = 2.16(1.587^y - 1)$$

Examples

<table>
<thead>
<tr>
<th>$y$</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.27</td>
</tr>
<tr>
<td>2</td>
<td>3.29</td>
</tr>
<tr>
<td>5</td>
<td>20.</td>
</tr>
<tr>
<td>10</td>
<td>218.</td>
</tr>
<tr>
<td>100</td>
<td>$2.53 \times 10^{20}$</td>
</tr>
</tbody>
</table>
Solving Hard Problems

Solution Time

- Problem of size $n$
- Running $y$ years on Moore’s Law computer

\[ y = 2.16 \ln(1 + 0.462 k 2^n) \]

- For large values of $n$:

\[ y \sim 1.5 n + 2.16 \ln k - 1.67 = O(n) \]

Complexity

- Linear in problem size
Solving with a Moore’s Law Computer

Moore's Law Computer

Problem size (n) vs. CPU Years

Time per Operation:
- second
- minute
- hour
- day
- week
- year

The graph shows the relationship between problem size (n) and CPU years required for different time units, illustrating the exponential growth due to Moore's Law.
**Effect of Step Complexity**

**Observe**
- Step complexity $k$ adds only additive factor of $2.16 \ln k$ to running time

**Example**
- For $n = 100$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 second</td>
<td>111</td>
</tr>
<tr>
<td>1 minute</td>
<td>120</td>
</tr>
<tr>
<td>1 hour</td>
<td>129</td>
</tr>
<tr>
<td>1 day</td>
<td>136</td>
</tr>
<tr>
<td>1 week</td>
<td>140</td>
</tr>
<tr>
<td>1 year</td>
<td>148</td>
</tr>
</tbody>
</table>

**Explanation**
- Final years of computation will be on exponentially faster machines
Implications of Moore’s Law

**P=NP (Effectively)**
- Problems of exponential complexity can be solved in linear time

**Caveat**
- Cannot hold forever

**Fundamental Limit**
- Argument due to Ed Fredkin
- Claim that ultimate limit to growth in memory capacity is cubic
- Cannot build storage device with less than one electron
- Assume consume all available material to build memories
  - Would soon exhaust planetary resources
  - Cannot travel into outer space faster than speed of light
- Total amount of material available at time $t$ is $\Omega(t^3)$
- This limit will be hit in ~400 years
How to Be a Visionary

Pick a Really Hard Problem

- Sequencing of human genome
- Accurate weather prediction
- Flying helicopter autonomously

Make Proclamations

- “In 20 years, problem X will be solved”

Wait

- But make sure everyone credits you with the vision
- Maybe make a few contributions to technology

Amass Glory

- Turing Award Citation:
  - “He/She had the foresight to see that this problem could be solved.”
Truly Hard Problems

Those That Get Harder over Time
- Track Moore’s law growth
- How do I make sure my chip will operate correctly?
- How do I make sure my programs are correct?
- How do I manufacture state-of-the-art chips?

Highlight
- Research at CMU on formal verification of hardware
Motivation for Formal Verification

Intel’s Challenge, (ca. 1992)
- Design a high performance, state of the art microprocessor to succeed the 486
- Maintain compatibility to 20-year old x86 product line
- Provide new levels of performance on floating point

Floating Point Divider
- Use radix-4, SRT algorithm developed in 1960’s
- First time ever used by Intel

Validation
- Run lots of simulation tests
- Make sure it runs set of Windows applications

Manufacturing Environment
- Will produce millions of chips
- Cannot make any changes after manufacture
The Pentium Fiasco

Events

- **Prof. Thomas Nicely, Lynchburg College, VA**
  - Looking at properties of “twin primes”
  - Incorrect reciprocals for 824633702441 and 824633702443
    » ~ Single precision accuracy \((4 \times 10^{-9})\)
  - Contacted others on Oct. 30, ‘94

- **Spreading of Information on Internet news group**
  - `comp.sys.intel`
  - Terje Mathisen of Norway posts Nicely’s findings on Nov. 3
  - Andreas Kaiser of Germany finds 23 bad reciprocals, Nov. 10

- **Tim Coe, Vitesse Semiconductor, Nov. 16**
  - Created (good enough) software model of flawed divide algorithm
  - Discovered (nonreciprocal) cases with error up to \(6 \times 10^{-5}\)
  - Later showed 1738 mantissa pairs with less than single precision accuracy
    » out of \(7.4 \times 10^{13}\) single precision mantissa pairs
Resolution

Free Replacement Policy, Dec. 20
- No need to argue need
- Complex logistics
  - Many different versions
  - Actual replacement easy

Financial Impact
- Intel charged $475 million to it’s 4Q94 earnings
- Still was 2nd most profitable year ever
- Few companies could survive such an expensive mistake
- In the end, generated lots of valuable PR for Intel
Is There a Better Way?

Provide Mechanism to “Patch” Functionality

- Make chips more “malleable” so that can update as would software
- Intel has started to incorporate such mechanisms
- Future technologies such as field-programmable logic could help (Seth Goldstein)

Make Sure Hardware is Really Correct

- Formal hardware verification
- Apply automated, mathematical techniques to prove properties about system
- Focus of this presentation
CMU’s Research Contributions

Symbolic Model Checking

- Developed by Ken McMillan while CMU PhD student
  - Building on work by advisor Ed Clarke
- Verify properties of finite state systems with $10^{20}$ or more states

Binary Moment Diagrams

- Developed by Bryant & Chen in 1994.
- Symbolic representation of functions having bit-level inputs and numeric outputs
- Compact for common logical and arithmetic operations

Word-Level Model Checking

- Developed by Xudong Zhao while CMU PhD student
  - Advisor Ed Clarke
- Allow specification to contain arithmetic relations among words of data
Temporal Logic Model Checking

Verifying Reactive Systems

- Construct state machine representation of reactive system
  - Nondeterminism expresses range of possible behaviors
  - “Product” of component state machines
- Express desired behavior as formula in temporal logic
- Determine whether or not property holds

“It is never possible to have a green light for both N-S and E-W.”
Word-Level Abstractions

- View bundle of wires as encoding numeric value
- Represent as function
  - Over Boolean variables
  - Yielding numeric value

Example Encoding Function

- Unsigned binary

\[ X = x_0 + 2 x_1 + 4 x_2 + \ldots + 2^{n-1} x_{n-1} \]
Word-Level Verification

- Lai [USC], Vrudhula [Arizona]

**Given**
- Bit-level circuit representation
- Encodings of inputs and outputs
- Word-level specification

**Compare**
- Correspondence between two representations
- Under I/O encoding

**Observation**
- Crossing abstraction boundary

Multiplier Circuit

Spec: \( P = X \cdot Y \)
BMD Representations of Integers

**Unsigned**

\[ x_0 + 2x_1 + 4x_2 + 8x_3 \]

**2’s Complement**

\[ x_0 + 2x_1 + 4x_2 - 8x_3 \]

**1’s Complement**

\[ x_0 + 2x_1 + 4x_2 - 7x_3 \]

**Sign-Magnitude**

\[ (x_0 + 2x_1 + 4x_2)(1 - 2x_3) \]
Word-Level *BMDs

$X + Y$

$X \cdot Y$

$2^X$
Using Word-Level Verification

Word-Level Model Checking

- Xudong Zhao, CMU PhD ‘97

Idea

- Introduce word-level specifications into model checker’s specification language
- Implement with combination of BDDs and BMDs

Applying to Intel’s Circuits

- Verified that each iteration of SRT divider is correct
- Major breakthrough for Intel
- Still cannot do “end-to-end” verification of divider
Recent Result on Arithmetic Circuits

Yirng-An Chen, PhD ‘98

Verifying Floating Point Adders

- Able to automatic verify complete behavior, including rounding
- That it realizes IEEE FP standard
- Completely “hands-off”

- No guidance from user on how circuit really works
Formal Verification Tasks

Digital Circuits
- Arithmetic circuits
  - “Does this circuit compute the specified mathematical operation?”
- Pipelined processors
  - “Does this circuit implement the specified instruction set?”

Reactive Systems
- Cache protocols
  - “Is it possible for 2 processors to have write access to a single block?”
- Controllers
  - “If a car approaches the traffic light, will it eventually turn green?”

Software Systems
- Operating Systems
  - “Is it possible for the scheduler to exclude a process indefinitely?”
Long Term Verification Challenges

Processor Verification

- Verification becoming much more difficult
  - Out of order & speculative execution
  - Widening gap between abstract specification & actual system operation
- Are we catching up or falling behind?

Reactive Systems

- Still requires reducing system complexity by abstraction
  - E.g., reduce to single cache line, make buffers nondeterministic
  - Limited success with automated techniques

Software Systems

- Can only deal with highly simplified system models
- Possible to catch bugs, but not to guarantee correctness