EE382
Processor Design

Winter 1999
Silicon Area and Cost Models

Topics

- Methodology
- Area-Cost Model
- Processor Area Allocation Model
- Baseline Processor Cost
- Summary
Baseline Processor Model

- **Cost (Area)**
  - Study 2.3, pages 94-99
- **Pipeline Performance**
  - Study 4.11, pages 253-259
- **Cache Performance**
  - Study 5.5, Pages 335-336

Pentium Processor Floorplan
Methodology

- Pick target cost based on market requirements
- Determine total area available within cost budget
  - defect and yield model
- Compute net available area for processor logic
  - account for I/O, buses, test hooks, etc.
- Determine integer core area
  - feasibility studies and empirical design targets
- Allocate remaining area to optimize performance
  - cache, FPU, multimedia, branch buffer, etc.

Cost

- Fixed Cost (Non-recurring)
  - Engineering plus overhead
    - salaries, management, computers, CAD tools
  - Pre-production wafer processing
  - Marketing, General and Administrative
- Incremental Costs
  - Wafer processing
  - Assembly
  - Test
- And the system
  - memory, peripherals, other components, boards, cabinets, power supplies, ...

Focus for this Course is Processor Design
Tradeoffs Related to Silicon Area
Wafers and Chips

suppose the wafer has diameter $d$ and each die is square with area $A$

then if $N$ is the number of dice on the wafer,

$$N = \pi \left( d - \sqrt{A} \right)^2 / (4 \ A) \text{ [Gross Yield]}$$

Let $N_G$ be number of good dice
and $N_D$ be the number of defects on a wafer.
Given $N$ dice of which $N_G$ are good.....suppose we randomly add
1 new defect to the wafer. What’s the probability that it strikes a
good die....and changes $N_G$?
Probability of the defect hitting a good die = $N_G / N$

The change in $N_G$ is 

$$d N_G / d N_D = - N_G / N$$

Rewriting this we get

$$d N_G / N_G = - (1/N) d N_D$$

Integrating and solving

$$\ln(N_G) = -N_D/N + C$$

Since $N_G = N \Rightarrow N_D = 0$, $C$ must be $\ln(N)$

$$N_G / N = \text{Yield} = e^{-N_D/N}$$

let defect density (defects/cm$^2$) = $\rho_D$

$$N_d = \rho_D \times \text{wafer area}$$

$$N_d = \rho_D \times (\pi d^2/4)$$

$$N = \pi d^2/4 \ A$$

$$\text{Yield} = N_G / N = e^{-\rho_D A}$$

typically $\rho_D = 1$ defect/cm$^2$
Using Yield to size a die

to find the cost per die:

1. find $N$, the number of die on a wafer
2. find Yield
3. find $N_g = \text{Yield} \times N$
4. cost/die = wafer cost/ $N_g$

<table>
<thead>
<tr>
<th>Wafer Diameter (cm)</th>
<th>Defect Density (per cm$^2$)</th>
<th>Wafer Cost ($)</th>
<th>Die Size (cm)</th>
<th>Gross Yield</th>
<th>Yield</th>
<th>Good dice</th>
<th>Cost per good die ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>1</td>
<td>5000</td>
<td>1</td>
<td>314</td>
<td>0.37</td>
<td>116</td>
<td>$43</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>5000</td>
<td>1.5</td>
<td>133</td>
<td>0.11</td>
<td>14</td>
<td>$357</td>
</tr>
</tbody>
</table>

Other Effects

- Aspect ratio
- Number of metal layers
- I/O
  - Number of I/O pads (wire bond)
  - Array bond
- Reticule size for step and repeat lithography
target yields

Most processor designs start with a target yield = 0.1

Why so low?
This allows an increases design lifetime. Since
as f (feature size) decreases we can lower A, hence the cost.

Over the design life we can lower \( \rho_d \) and increase diameter of wafer.

Semiconductor Technology Roadmap

<table>
<thead>
<tr>
<th>Year</th>
<th>Technology Generation (nm)</th>
<th>Wafer Size (mm)</th>
<th>Defect Density (per m(^2))</th>
<th>( \mu )processor Die Size (mm(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>250</td>
<td>200</td>
<td>1940</td>
<td>300</td>
</tr>
<tr>
<td>2001</td>
<td>150</td>
<td>300</td>
<td>1510</td>
<td>385</td>
</tr>
<tr>
<td>2006</td>
<td>100</td>
<td>300</td>
<td>1120</td>
<td>520</td>
</tr>
<tr>
<td>2012</td>
<td>50</td>
<td>450</td>
<td>775</td>
<td>750</td>
</tr>
</tbody>
</table>

http://notes.sematech.org/index4.htm
What can be put on the die?

- Depends on the lithography and die area
- Lithography determined by $f$, minimum feature size
- Feature size is related to $\lambda$, the mask registration variation
  \[ f = 2\lambda \]

Smallest device is $5\lambda \times 5\lambda$
register bit as an area unit

register cell contains 6 transistors, but...
because of transistor sizing, wires, buses, etc cell is much larger than 150 $\lambda^2$

one register bit occupies one rbe register bit equivalent of area

$$\text{one rbe} = 2700 \lambda^2 = 675 \text{f}^2$$

rbe is the basic unit used in this course for modeling cell-based area cost

Register File

- a register file consists of register cells (bits) and port logic.
- a simple register file has one read port plus a shared read/write port.
- multiple issue machines have many more ports.
  — Size of register bit generally grows quadratically with the number of ports
- the area occupied by a register file in rbe is
  $$\text{rbe} = (\text{no. of regs.} + 3P)(\text{bits/word} + 3P)$$
  where P = number of ports, shared or unshared.
- a simple integer file (1 read + 1 read/write) with 32 words of 32 bits/word has area of
  $$\text{rbe} = (32 + 3\times2)(32 + 3\times2) = 1444.$$
Area Units: rbe and A

- rbe is still a small area unit when sizing the functional units of the processor.
- Suppose we define another larger unit, A, as \( 1A = f^2 \times 10^6 \)
- Then \( 1A = 10^6 / 675 = 1481 \text{ rbe} \)
- Since 1481 is close to 1444 we can also refer to the simple register file as occupying 1 A units.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Relative Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda ) mask registration</td>
<td>2 ( \lambda )</td>
</tr>
<tr>
<td>( F ) minimum feature size</td>
<td>( 2700 \lambda^2 = 675 f^2 )</td>
</tr>
<tr>
<td>rbe register bit equivalent</td>
<td>( 10^6 f^2 = 1481 \text{ rbe} )</td>
</tr>
<tr>
<td>A functional unit area</td>
<td></td>
</tr>
</tbody>
</table>

area of other cells

- 1 register bit = 1 rbe
- 1 CAM bit = 2 rbe
- 1 cache bit (6 tx cell) = 0.6 rbe
- 1 DRAM bit = 0.1 rbe = 67.5 \( f^2 \)

These are the parameters for basic cells in most design tradeoffs
Floorplan and Area Allocation

- The challenge is to choose what functions to integrate and allocate area to maximize performance within an overall area budget.
- Reasons to Integrate a Function
  - Performance
  - System Requirement
  - Platform standard
  - Test/Quality Cost
  - Power
  - Other reasons lead to less flexible, more expensive system solutions.

Cache Area

- the cache occupies (approximately)
- cache area = 195 + 0.6(cache array bits) + 0.6(directory bits) in rbe.
- cache access time and cycle time depend on cache size and organization.
- so as not to slow down processor cycle cache access can be multi cycle OR we can use two levels of cache.
Cache Access time

- The access time to a cache can be approximated by:

  \[ \text{Access time (ns)} = (0.35 + 3.8f + (0.006 + 0.025f)C) \times (1 + 0.3(1 - 1/A)) \]

  - \( f \) is the feature size in microns
  - \( C \) is the cache capacity in K bytes
  - \( A \) is the associativity, e.g. direct map \( A = 1 \)

- For example, at \( f = 0.1 \mu \text{m}, \) \( A = 1 \) and \( C = 32 \) (KB) the access time is 1.00 ns.

- The problem with small feature size is cache access time, not cache size.

---

Cache access time

- At \( f = 0.1 \mu \text{m} \) we expect \( \Delta t = 1 \text{ns}, \) so we're limited to 32 KB cache size for first level cache for one cycle access.

- Some alternatives:
  - 2 cycle pipelined access
  - interleaved 2 cycle caches
  - live with 32 KB; back it up with fast Level 2 cache

- The maximum cache size with access 2ns (\( A = 1 \)) is 128KB; 3ns is 256 KB; 5ns is 512 KB (almost).
CacheOpt

- we can use CacheOpt tool (available at web site) to optimize use of cache area.
- we can determine largest level 1 cache to match a processor cycle time then use remaining area for a level 2 cache.
- NOTE: below f = 0.5u and especially below f = 0.3u the simple area scaling described in the text is inaccurate as the wire size and spacing do not scale as devices scale
- CacheOpt corrects for this, so for small f use the tool for the best area estimates.

Area for an integer core

- integer core processor must accommodate registers (say 32x32), integer ALU, I fetch decoder, LD/ST buffers and maybe two TLBs (I and D)
- area estimate for a simple core is 16.1 A
Area for Floating Point Unit

- for our baseline processor with performance of FADD = 3~s, FMPY = 3~, FDIV = 15~s we would have 13.5A for the FADD and 23.3A for a combined FMPY/FDIV.
- with a register file the total area for this FPU is 37.8 A
- for relatively simple core processors, it’s easy for the either the FPU or the cache to occupy significantly more area than the core processor itself. This had led to significantly larger cores....big superscalar processors with instruction reorder, BTBs, etc. in an attempt to balance area allocation and improve performance.

FUPA

- a broad tradeoff exists for FPUs trading performance for area.
- this can be expressed as a time x area product, called the FUPA.
- assume time is measured in (ave. EX ~) -1
- if area is in A, then time x A = FUPA, a constant.
- functional area - time products are of the form (generally) of A x T^n = K where n is between 1 and 2...depending on the interconnections required.
- this AT^n = K curve defines the state of the art (par) for a particular functional unit(s) implementation.
other considerations

- the net die area is only 80% of the gross due to I/O pads and guard ring.
- there’s an additional overhead of 10% of FPU + integer area for latches and 40% for buses and control.
- the remaining area (less 10% for aspect ratio loss) is available for cache.

We can fit a fully-functional 32-bit processor in a 1µm process with acceptable yield

Cache

- The cache has been allocated an area of 92.8 mm²
  - For 1 µm this allows 229K bits or 28.6KB for data, tags, and overhead
  - Icache is allocated 16KB with 16B lines
  - Dcache is allocated 8KB with 16B lines
Area Optimization

- the problem is to find the combination of core (integer) unit + FPU + cache that gives the best performance.
  - Performance has multiple dimensions
  - Weighting the dimensions is a marketing issue
- what provides the most improvement in performance per unit area?
  - Decisions to adjust total area budget
    - Increase or “Die-Diet”
  - Decisions to allocate among functions within budget
- Technical innovation and analysis intersect with business decisions to make a product!

Future Directions

In any processor $\Delta t$ scales with feature size, $f$.

Today’s processors have $f = 0.35$ u and have $\Delta t = 2$ to 3 ns.
We expect that the technology will support $f = 0.1$ u and $\Delta t = 1$ ns

Observation:
$\Delta t$ usually scales linearly with $f$, but at small $f$ the improvement is less than linear since the voltage is lower and the interconnection delay dominates.

At 0.35μm, we have ~9x the area of the baseline processor. That is why current processors have BTBs, larger caches, multiple integer and FP pipelines. At 0.1 μm we will have ~100X the area of the baseline processor. What will that architecture look like?
Summary

- Cost is an exponential function of area
- Successful business model targets initial production at relatively low yield (~0.1)
  - ride learning curve and leverage technology to reduce cost and improve performance
- Technical innovation and analysis intersect with business decisions to make a product!
  - Use design feasibility studies and empirical targets
  - Methodology for cost and performance evaluation
  - Marketing targets determine weighting of performance metrics

Microprocessor Generations

<table>
<thead>
<tr>
<th>1.5(\mu)m</th>
<th>1.0(\mu)m</th>
<th>0.8(\mu)m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1986</td>
<td>1989</td>
<td>1992</td>
</tr>
<tr>
<td>i386 CPU</td>
<td>i486 CPU</td>
<td>Pentium Processor</td>
</tr>
<tr>
<td>275K Tx</td>
<td>275K Tx</td>
<td>1.2M Tx</td>
</tr>
<tr>
<td>103 mm(^2)</td>
<td>55 mm(^2)</td>
<td>160 mm(^2)</td>
</tr>
<tr>
<td>20 MHz</td>
<td>33 MHz</td>
<td>33 MHz</td>
</tr>
<tr>
<td>208 mm</td>
<td>235 mm</td>
<td>238 mm</td>
</tr>
<tr>
<td>20 MHz</td>
<td>66 MHz</td>
<td>66 MHz</td>
</tr>
</tbody>
</table>