Lecture #3 Wed Jan 22

****
Review -- 1 min
****

Unifying idea: quantitative basis for architecture

Engineering Methodology
tech trends, quantitative approach
Bottom line: performance
throughput or latency
execution time is the real measure of computer performance
benchmarking
o games people play, mistakes people make

o good benchmarks --> good products

o bad benchmarks create dilemma -- improve
real performance or improve benchmark number.
Marketing almost always wins.
Amdahl's law
CPI = instr/program * cycles/instr * seconds/cycle = seconds/program
EXECUTION TIME is real measure of performance

****
Outline - 1 min
****

cost

<project discussion>

basic pipelining

****
Lecture - 20 min
****

Costs
----

cost/performance is driving force in CS today

others in department might look down on considering costs
"transient engineering factors" not "scientific principles"

wrong for 3 reasons
1. cost will always be part of design

(correct trade-offs may change from year-to-year, but your approach to making those trade-offs will last)

2. cost/performance factors change at exponential rates
--> exert qualitative,lasting impact on design
just as algorithm design would't ignore an exponential factor, neither should computer design

3. some cost/performance rules are close to laws
e.g. "smaller is faster," "smaller is cheaper"

IC Cost

Details in book
I'm assuming that most people have seen math in UG class

Not going to cover in detail, but you may want to spend some time working through an example problem (this is stuff you should know how to do)

$$\text{IC cost} = \frac{\text{die cost} + \text{testing cost} + \text{packaging cost}}{\text{final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{dies per wafer} * \text{die yield}}$$

pi * (wafer_diam/2)^2 pi * wafer_diam
Dies per wafer = ----------------------------- - ----------------------
- Test dies
die area sqrt(2*die area)

/* area/area - correction factor - test dies */

Die yield = Wafer yield * { 1 + (defects per unit area * die area) / alpha} ^(-alpha)

--> empirical formula
--> alpha depends on complexity of process (more layers --> larger alpha)
also learning curve --> alpha falls over time for a chip
--> larger chip --> double whammy: fewer dies per wafer, more defects per die

Die cost goes roughly with die area ^4

Real World Examples (Aug 1993)

Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	.9	900	1.0	43	360	71%	$4
486DX2	3	.8	1200	1	81	181	54%	$12
PowerPC 601	4	.8	1700	1.3	121	115	28%	$53
HP PA 7100	3	.8	1300	1	196	66	27%	$73
DEC Alpha	3	.7	1500	1.2	234	53	19%	$149
SuperSPARC	3	.7	1700	1.6	256	48	13%	$272

Pentium	3	.8	1500	1.5	296	40	9%	$417

source: "Estimating IC Manufacturing Costs" Linley Gwennap _Microprocessor Report_ Aug 2 1993 p15

SKIP: Cost v. Price
o Component cost: 15% - 33%
o Direct costs (6-8%) -- recurring costs: labor, purchasing, scrap, warrenty
o Gross Margin (34-39%) -- nonrecurring costs: R&D, marketing, sales, equipment maintenence, rental, financing, cost, pretax profits, taxes
--> Avg selling price
o Average Discount (25%-40%) -- volume discounts/retailor markup
--> List price

Chip Prices (Aug 1993) (10K units price)

Chip	Area mm^2	Mfg cost	Price	Multiplier	Comment
386DX 43	$9	$31	3.4		Intense COmpetition
486DX2	81	$35	$245	7.0	No Competition
PowerPC 601 121		$77	$280	3.6	
DEC Alpha	234	$202	$1231	6.1	Recoup R&D?
Pentium	296	$473	$965	2.0	Early in shipments

Volume v. Cost

Learning Curve: Rule of thumb "When volume doubles, cost reduces 10%"
-- _A DEC View of Computer Engineering_ C. G. Bell, J. C. Mudge, and J. E. McNamara. Digital Press, Bedford, MA 1978.

example: NOW v. custom MPP
40MPPs @ 200 nodes = 8000 nodes/year
100,000 Workstations/year
12.5X + 2^3.6 ==> 0.9^3.6 ==> 0.68
--> cost should be 1/3 less for same components

examples: PCs v. Workstations

	1990	1992	1994	1997
PC	24M	33M	44M	65M
WS	0.41M	0.58M	0.68M	0.98M
Ratio	59	57	65	67

65x ~ 2^6 ==> 0.9^6.0 = 0.53

High Margins on High-end Machines

- o R&D considered return on investment (ROI) = 10%
  - --> every $1 in R&D must generate $7-$13 in sales
- o High end machines need more R&D
- o Sell fewer high end machines
  - --fewer to amortize R&D
  - -- much higher margins

- o example: price of $1MB Memory (Jan 1994)
  - PC $40 (Mac Quadra)
  - WS $42 (SS-10)

QUESTION: Guess how much $1MB of mainframe memory costs?

Mainframe $19200 (IBM 3090)
Supercomputer $600 (M90 DRAM)
$1375 (C90 15ns SRAM)

- o example: recouping development costs on low-volume microprocessors

MIPS R4000 cost $30M to develop (Hennessy)
SGI/MIPS sells 300K R4000s over product lifetime
--> $100/chip

80486 cost $100M (rumor)
Intel sells 50M?
--> $2/chip

SKIP: Price/performance -- gross margin v. market segment

BARGRAPH the following

	Mini	Workstation	PC
Avg Discount	1.2	1.3	.3
Gross Margin	2.2	1.3	0.4
Direct Costs	0.3	0.2	0.1
Component Costs	1.0	1.0	1.0

Impact of Margin Shrink on Society/Industry

Gross Margin ==> entire research and development budget

What does this mean to your jobs?

****
Admin - 3 min
****

TA: Youxin Gao gao@cs.utexas.edu
TA office hours: TBD

WED: Project interests (mail to TA)
FRI: HW1
does everyone have homework partner?

project discussion
----

goals

1) Learn to do systems research
Good science = skills + attitude

work independently -- from topic selection to execution

(diferent from undergrad projects)

quantitative methodology
(don't just measure a bunch of stuff, test hypothesis
--> better results with less effort

2) Learn about an area of architecture in depth
3) Advance state of the art

approach

1. select a topic
2. state hypothesis
3. design experiment to test hypothesis

ask yourself: what graph would I draw at the end of the experiment
to convince you that I'm right?
ask yourself: suppose I'm wrong, will experiment tell me?

4. Execute plan

Checkpoints for class

1) Wednesday Jan 27 – topic interests – mail to gao@cs

- your name, email
- project interests
- whether you have a partner or are looking for one

We'll post the list – you can use it to find people that might be good project parnters

2) Feb 5 – topic selection

1 paragraph description of the
hypothesis you plan to test and the general approach you plan to take

Selecting a topic:

- General approach – different ways to go
  - Reproduce the results of an important paper (other fields do this a lot)
    - Harder than you think – papers omit important details
    - You will learn a lot
    - This type of project will be graded on good scientific procedures
  - take a paper and extend its ideas just a bit
  - a new idea

- Topic ideas – see handout for long list; highlight a few here
- Factors to consider
  - Time limits – make progress in 3 months
    (important to have at least part of it completely working; hard to evaluate a simulator that didn't quite work.
    Ideally project has a short term (2 month) goal, and a longer-term goal
  - interest – work on something you care about; tie it into your research interests if possible (bring special insight and expertise that others may not have)
  - impact – work on something others will care about; goal is something that, if successful, would be at least the start of a good conference paper

3) Feb 26 – proposal
- literature review/critique
- detailed plan of attack

4) April 9 – checkpoint
- show me 1 graph! I will be impressed with anyone that has some initial results, no matter how minor!

5) May 3-7 in-class presentations
6) May 12 – written report

****
Lecture - 24 min
****
Basic Pipelining

It's natural!

Laundry example

I have 4 loads
Washer: 30 minutes
Dryer: 40 minutes
Folding: 20 minutes

Sequential -- unpipelined

```
Load1:      Wash Dry        Fold
       <---30-->   <----- 40 ----->    <---20--->
Load2:                              Wash Dry        Fold
                               <--30-->    <-----40----->    <--20-->
```

etc <draw rest on board>....

--> Total time for 4 loads -- 6 hours

Would never do it like that!!

Pipelined

```
Load1:      Wash Dry        Fold
       <--30--><----40---->    <--20->
Load2:           Wash       Dry          Fold
            <--30--><-10->   <----40----> <--20-->
Load3:                      Wash         Dry          Fold
                       <--30--><-10->   <----40----> <--20-->
Load4:                                  Wash         Dry          Fold
                                   <--30--><-10->   <----40----> <--
20-->
```

Total time: 30 + 40 + 40 + 40 +40 +20 = 210 = 3.5 hours

Laundry Lessons

1. Doesn't help latency of single task (helps throughput)
2. rate limited by slowest stage
3. multiple tasks operating in parallel

4. potential speedup = # pipe stages
5. unbalanced lenghts of stages reduce speedup
   (b/c rate limited by slowest)
6. time to "fill" and "drain" pipeline reduces speedup

<picture of DLX data path>

MIPS Instruction Set

32-bit fixed instruction format
32 registers (r0 contains 0)
3 address reg-reg ALU instructions
load/store
simple branch conditions

Register-Register
    Op   Rs1   Rs2   Rd   Opx--
Register-Immediate
    Op   Rs1   Rd   immediate------
Branch
    Op   Rs1   Rs2/Opx   immediate------
Jump/Call
    Op   Target----------------------------------

<step through how each type of instruction works
using picture of data path and instruction types>

Unpipelined -- <CLOCK>
        start an instruction

wait for logic to work itself out
<CLOCK> latch result in register
start next instruction

<picture of DLX datapath with pipeline registers>

<CLOCK> start I1
<CLOCK> start I2 ID I1
<CLOCK> ...

****
Summay - 1 min
****

cost – changing costs and relationships to design are key

also -- who will pay for the innovation in the new world? Who will pay engineers salary?

<project discussion>

basic pipelining – its natural!