

# **Division, Pentium Style**

**An Analysis of Intel's Mistake(s)**

**Randal E. Bryant**

***Carnegie Mellon University***

<http://www.cs.cmu.edu:8001/afs/cs.cmu.edu/user/bryant/www/home.html>

# Overview

## Summary

- ◆ **Pentium is Intel's most advanced microprocessor**
  - 3.3 Million transistors
  - In 42% of all personal computers sold Dec. '94
- ◆ **Until recently, all had error in floating point division hardware**
  - 5 missing transistors, fixed with change to single mask
- ◆ **Disclosed largely via Internet**
- ◆ **Intel ultimately offered replacements to everyone**
  - \$475 Million charge from 4Q94 revenue

## Outline

- ◆ **Brief chronology**
- ◆ **Technical details**
- ◆ **Overall impact**

# Discovery and Disclosure

## 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 cases with less than single precision accuracy

# Intel Plays Tough

- ◆ **Claimed had discovered in Summer '94**
  - Logged as minor bug, to be fixed on next revision
- ◆ **Andy Grove “posted” to Internet, Nov. 24**
  - Stating that other posters are overreacting
    - » All chips have flaws
  - Different Intel employee actually did the post
- ◆ **Agrees to replace, but only to those who can justify need**
- ◆ **Nov. 30: Report made available via WWW**
  - Describes algorithm and error
  - Estimates average user will encounter once in 27,000 years

# The Uproar

## IBM Breaks Ranks

- ◆ Produces report stating error may be encountered as much as once every 24 days, Dec. 12
  - Nonuniform number distribution
  - 4000X more divides per day than Intel estimates
- ◆ Stops shipment of all Pentium-based PCs
- ◆ Some question IBM's motives

## Internet

- ◆ Hundreds of posts daily to `comp.sys.intel`
- ◆ Arguments on both sides
- ◆ Angered about Intel's attitude

## Popular Press

- ◆ Starts following controversy in November
- ◆ Generally accepts Intel's description as "obscure error"

# Resolution

## Free Replacement Policy, Dec. 20

- ◆ No need to argue need
- ◆ Complex logistics
  - Many different versions
  - Actual replacement easy

## Intel Not Humbled

- ◆ Still state that error is “technically an extremely minor problem”

# Numerical Basics

## Floating Point Numbers

- ◆ **Numerical Form:**  $-1^s m 2^e$ 
  - Mantissa  $m$  either 24, 53, or 64 bits

## Given

- ◆ **Dividend mantissa**  $p$
- ◆ **Divisor mantissa**  $d$
- ◆ **Normalized forms:**  $1.xxxx_2$ 
  - Range Constraint:  $1 \leq p, d < 2$

## Compute

- ◆ **Quotient mantissa**  $q$ 
  - To sufficient precision
- ◆ **Exponent**
  - Based on exponents of dividend & divisor

# Radix 4 Division

## Conventional “Restoring” Algorithm

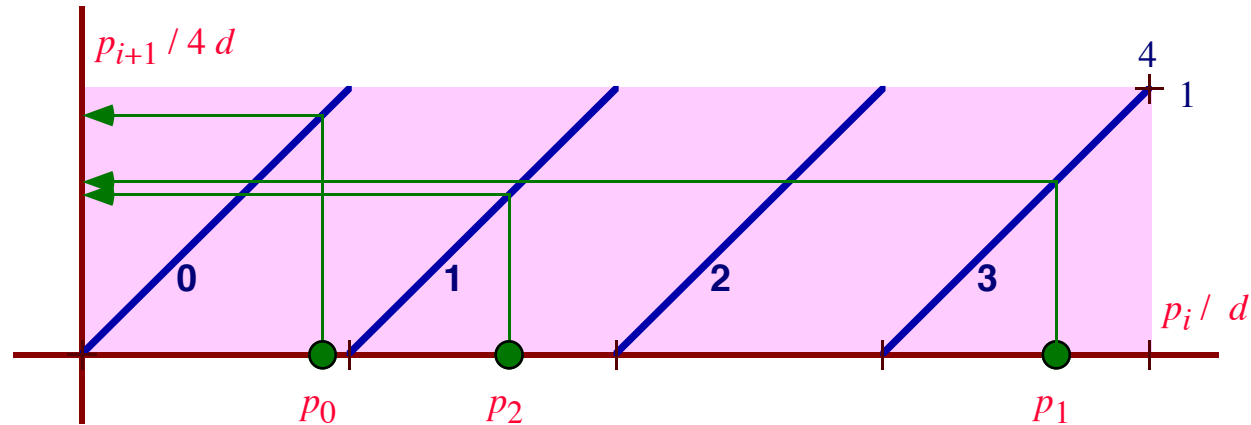
- ◆ Select quotient digit  $q_i$   
– 0, 1, 2, or 3
- ◆ Subtract  $q_i d$  from  $p$
- ◆ Shift  $p$  left

Divisor	$d$
X 1	1.301
X 2	3.202
X 3	11.103

- ◆ Must consider all digits to select digit  $q_i$
- ◆ Must form “awkward multiple”  $3d$

Example			
$p_0$	1.2130	$q_1$	0
$-q_1 d$	- 0.		
	<hr/>		
	1.2130		
	↙		
$p_1$	12.1300	$q_2$	3
$-q_2 d$	-11.1030		
	<hr/>		
	1.0210		
	↙		
$p_2$	10.2100	$q_3$	2
$-q_3 d$	- 3.2020		
	<hr/>		
	1.0020		
	↙		
	10.0200		

# Visualization of Digit Selection



## Interpretation

- ◆ **Horizontal axis indicates scaled partial remainder**
  - In range  $[0, 4)$
- ◆ **Vertical axis indicates result of subtracting quotient digit**
  - Unique choice of digit
  - In range  $[0, 1)$

# SRT Division

## History

- ◆ Radix 4 version due to Atkins (1968)
- ◆ Commonly used by others, first time for Intel

## Goals

- ◆ Predict quotient digit based on incomplete information about partial remainder and divisor
- ◆ No awkward multiples

## Key Ideas of Method

- ◆ Quotient Digits  $-2, -1, 0, +1, +2$ 
  - Redundancy allows imperfect digit prediction
  - Power of two multiples
    - » Implement with shift and/or complement

# SRT Iteration Step

## Given

- ◆ **Partial Remainder**

$p_i$

- Two's complement representation

- Can bound range

$$-16/3 < p_i < 16/3$$

- ◆ **Divisor**

$d$

- Range Constraint:

$$-8/3 d \leq p_i \leq 8/3 d$$

$P$   
SXXX.XXXXXX ...  $x_2$

$D$   
1.XXXXXX ...  $x_2$

## Select

- ◆ **Quotient Digit**

$q_{i+1}$

- By table lookup

- Using truncated values  $P$  and  $D$

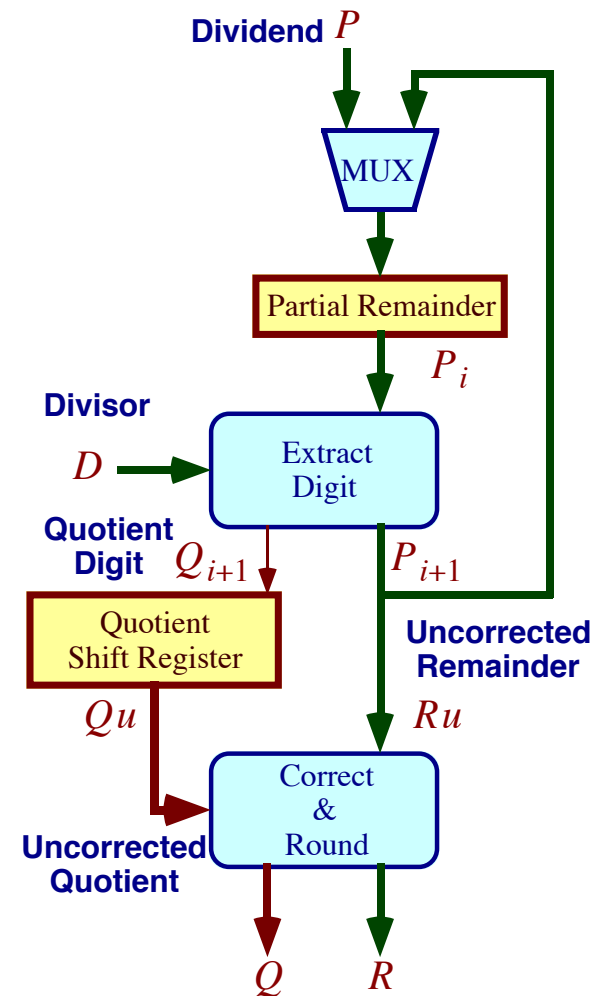
- ◆ **Compute**

$$p_{i+1} = 4 [p_i - q_{i+1} d]$$

# SRT Divider Circuit

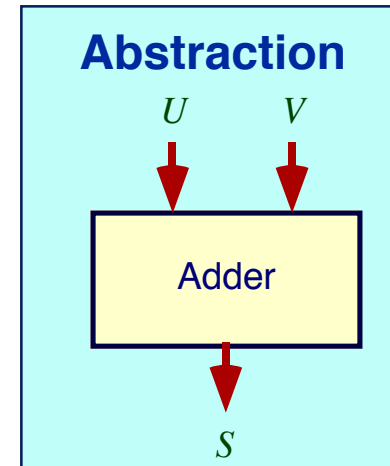
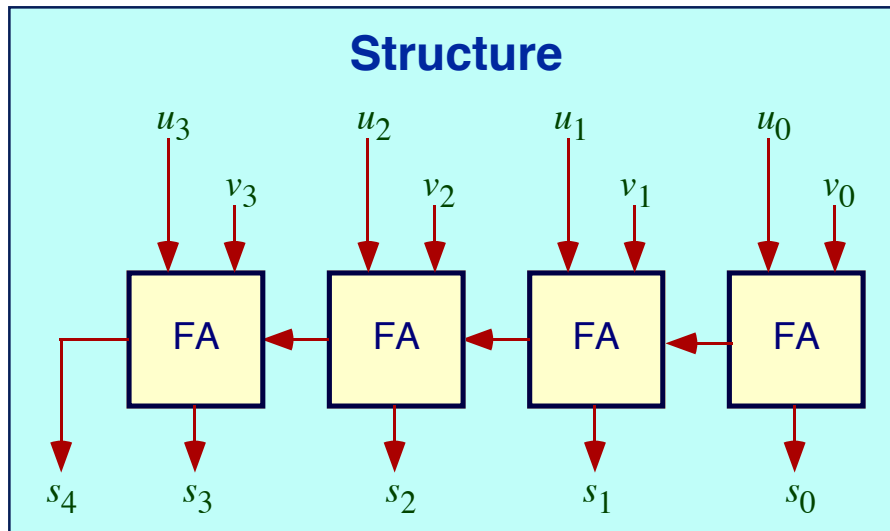
## Registers

- ◆ **Hold partial remainder**
  - Initially loaded with dividend
- ◆ **Divisor (not shown)**
  - Unchanged for entire computation
- ◆ **Quotient shift register**
  - Accumulates digits from iterations



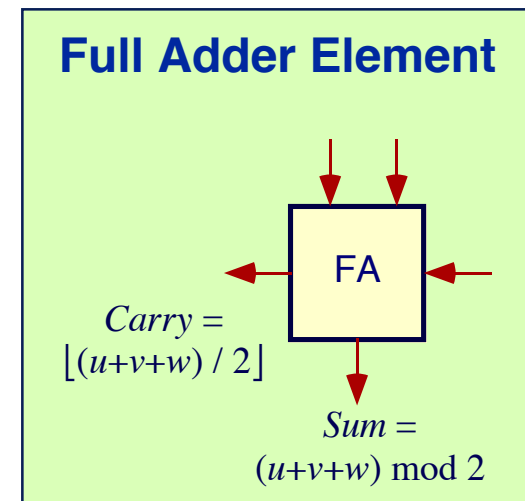


# Conventional Adder Design

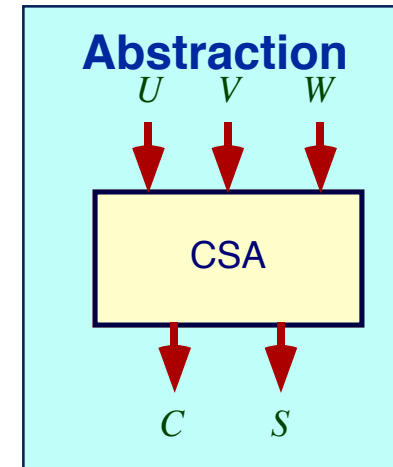
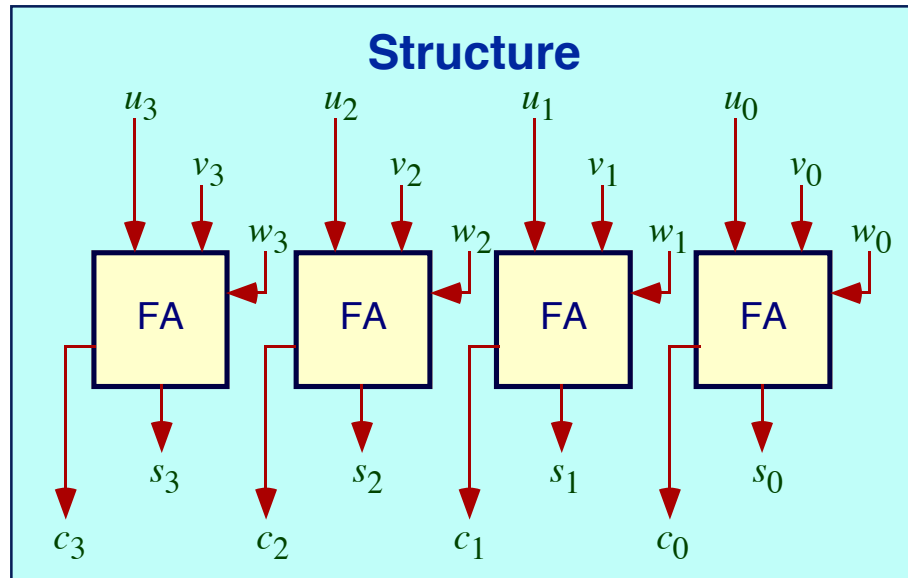


## Shortcoming

- ◆ Must propagate carries across all cells
- ◆ Can improve with more costly hardware



# Carry Save Adders



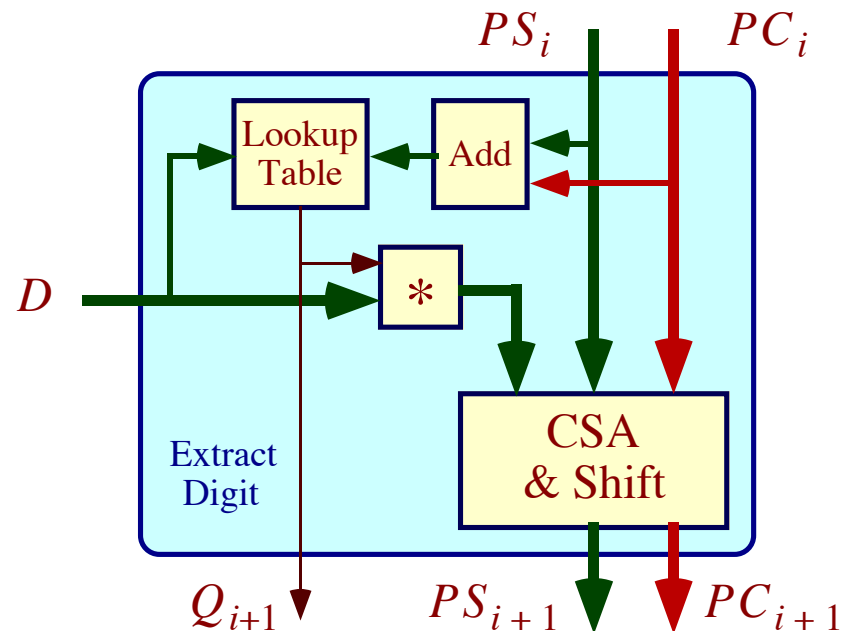
$$S + 2C = U + V + W$$

- ◆ Rearrange elements to eliminate carry chain
- ◆ Three input words
- ◆ Two output words
- ◆ 3 to 2 reducer

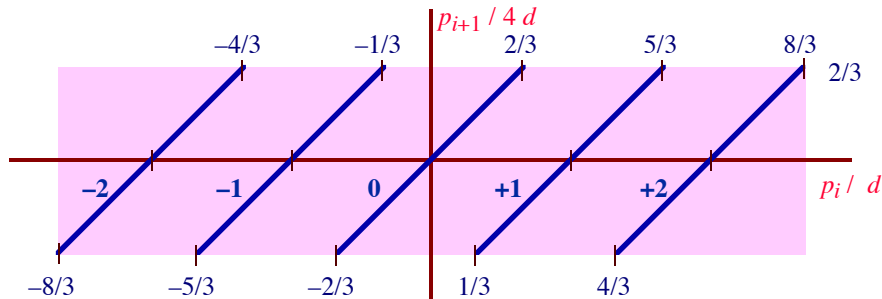
# SRT Division Step

## Elements

- ◆ Carry Save Adder to subtract weighted divisor
- ◆ 7-bit adder to get estimate of partial remainder
  - High order bits
- ◆ Lookup table to determine quotient digit
  - 4 “interesting” bits of divisor
    - » Leading bit always 1
  - Implemented as PLA
  - Where Intel made mistake
- ◆ Shift/complementer to weight divisor
  - Multiply by quotient digit

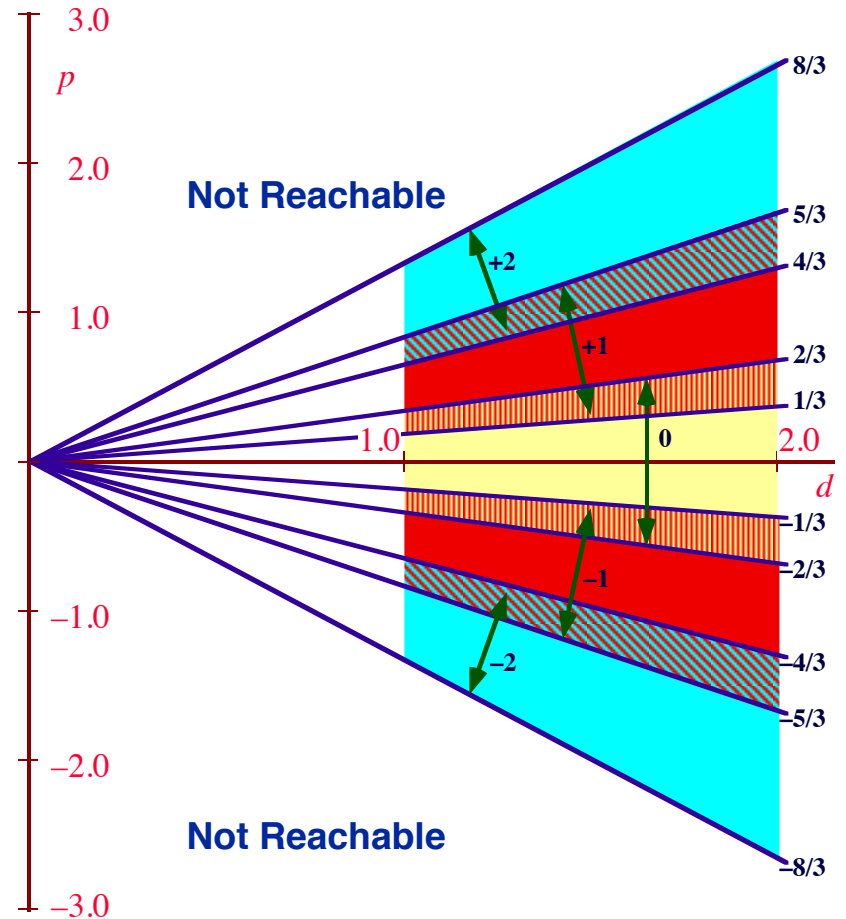


# Lookup Table Design



## Digit Selection

- ◆ Depends on ratio of  $d$  and  $p$
- ◆ Redundancy leads to overlapping regions
  - May choose either digit



# Discretization of P & D

## Values

- ◆ **Least Resolvable Unit (LRU)**

- Weight of least significant bit

- ◆ **Partial Remainder**  $p_i$

- Approximate by  $P$

- Potentially underestimates by two LRUs

- One each from  $pc$  and  $ps$

- ◆ **Divisor**  $d$

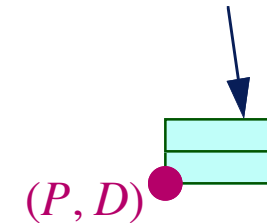
- Approximate by  $D$

- Potentially underestimates by one LRU

$P$   
 $\overbrace{\text{Sxxx.xxxxxx} \dots}$

$D$   
 $\overbrace{1.xxxxxx} \dots$

Range of Values for  $(p_i, d)$



## Compensating for Inaccuracy

- ◆ **Redundancy enables valid digit selection for all cases**

- As long as have enough bits of  $p_i$  and  $d$

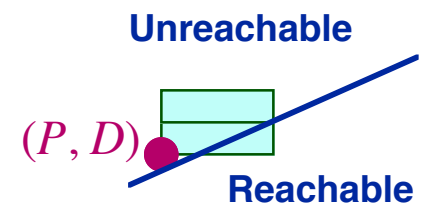
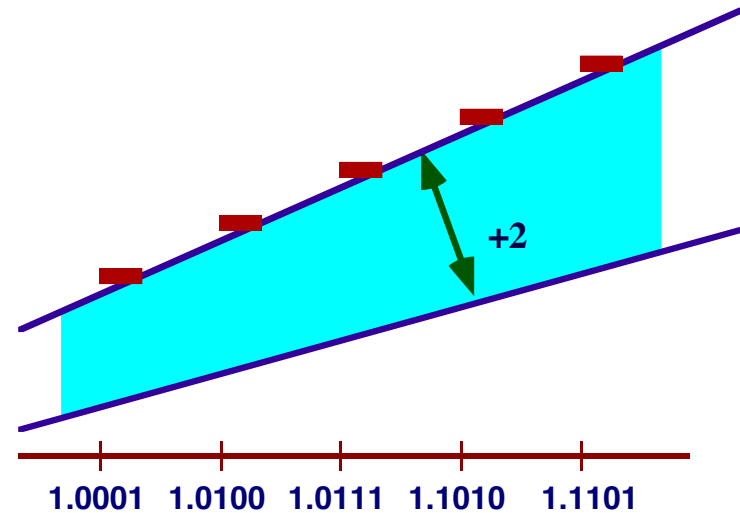
# Intel's Table Errors

## Actual Bug

- ◆ Erroneous entries for 5 cases
- ◆ Table generates 0 instead of +2

## Occurs Only under Marginal Conditions

- ◆  $(P, D)$  appear to be out of range
- ◆  $D$  must under-estimate  $d$
- ◆  $P$  must closely estimate  $p_i$ 
  - Unlikely with carry-save format



# Example of Error

## Correct Behavior

$$\begin{array}{r}
 p_0 \quad 0100.111000 \quad q_1 \quad +2 \\
 + - q_1 d \quad \underline{1100.010010} \\
 \quad \quad \quad 0001.001010 \\
 p_1 \quad 0100.101000
 \end{array}$$

## Incorrect Behavior

$$\begin{array}{r}
 p_0 \quad 0100.111000 \quad q_1 \quad 0 \\
 + - q_1 d \quad \underline{0000.000000} \\
 \quad \quad \quad 0100.111000 \\
 p_1 \quad 0011.100000
 \end{array}$$

Leading digits drop off end

## Divisor $d$

X	1	0001.110111
X	2	0011.101110
X	-1	1110.001001
X	-2	1100.010010

- ◆ Once hit bad table entry, cannot recover
- ◆ Contrary to some statements in press
  - Not “Self Correcting” !

# When Can Error Occur?

## Requires at Least 7 Iterations

- ◆ Hard to get to bad positions
- ◆ Worst case error  $6 \times 10^{-5}$

## Only for “At Risk” Divisors

- ◆ Stay in single column for entire division
- ◆ Remaining part of divisor should have lots of 1's
  - To ensure that  $D$  underestimates  $d$

### ◆ Coe’s Example

–  $d = 3145727_{10}$

10111111111111111111111111111111

– Correct result: 1.33382044913624100

– Pentium’s result: 1.33373906890203759

At Risk!

### ◆ Nicely’s Example

–  $d = 824633702441_{10}$  10111111111111111111111111111111011100000101001

# Table Realization

## Programmed Logic Array

- ◆ Encodes sum-of-products logic expression
- ◆ Coding generated automatically
  - Given complete table
  - With ESPRESSO program
- ◆ Mask patterns generated automatically

## Likely Source of Error

- ◆ Specified entries as “Don’t Cares”
  - Since appear to be out of range
- ◆ PLA optimizer chose to assign 0’s

Table			
<i>u</i>	<i>v</i>	<i>c</i>	<i>Carry</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

PLA			
<i>u</i>	<i>v</i>	<i>c</i>	<i>Carry</i>
–	1	1	1
1	–	1	1
1	1	–	1

# How Serious is the Error?

## Intel's Claims

- ◆ **Most applications don't do much division**
  - 1000 per day
- ◆ **Many applications can tolerate occasional errors**
  - E.g., image generation
- ◆ **Maximum error fairly small**
  - Once every  $1.5 \times 10^9$  divides
- ◆ **Statistical likelihood remote**
  - Once every 27,000 years

## IBM's Counterclaims

- ◆ **Even spreadsheet users do lots of divides**
  - Up to 4.2 Million per day
- ◆ **Error more likely for numbers just below integer**
  - Up to once every  $10^8$  divides

# Other Concerns

## Error is not Random

- ◆ Get same result every time on every Pentium

## Numbers Used in Practice NOT Uniformly Distributed

- ◆ E.g.,  $4.999999 / 14.999999$ 
  - gives  $0.33332922$  instead of  $0.33333329$
  - Vaughan Pratt's "bruised integers"

## Error is Hard to Work Around

- ◆ Can shut off floating point
  - Machine then runs slower than 486
- ◆ Can recompile with workaround divide subroutine
  - But most users only have binaries

## Bad Numerical Properties

- ◆ E.g.,  $4.999999 / 15.0 > 4.999999 / 14.999999$ 
  - Problem for iterative methods

# Who is Affected?

## Average PC User

- ◆ Not likely

## Heavy Users of Numerical Software

- ◆ Significant subset of Pentium customers
- ◆ Concern for numerical accuracy very high
  - Single inaccurate result can be magnified by later steps
- ◆ Ethical/legal responsibilities
  - Negligent to knowingly use flawed system
  - Would you buy a bridge from a civil engineer who had used a Pentium?

# Why Didn't Intel Discover it?

## Standard Steps in Verifying Design

- ◆ **Simulate many cases on high-level software model**
  - Probably had correct P-D table entries
- ◆ **Simulate/emulate final logic design**
  - Hardware emulators costing \$Millions
  - Run complete chip model at  $\sim 100\text{Hz}$ 
    - »  $< 10^{-6} X$  real time
  - Not clear which version of table was used
- ◆ **Run tests on initial production chips**
  - Feasible to run billions of tests
  - Should have caught error here

## Observations

- ◆ **Hard to test all aspects of such a complex system**
  - But this is a weak excuse

# What about Formal Verification?

## Currently Used Tools

- ◆ Most based on Ordered Binary Decision Diagrams (OBDDs)
- ◆ Check transistor circuit design against low-level software model
- ◆ Compare two logic networks for equivalence
  - E.g., pre- and post- optimization

## Other Alternatives

- ◆ Sequential verification by symbolic model checking
  - Used on protocols and small sequential circuits
- ◆ Tools based on automatic theorem provers
  - Unpopular with industry

## What About Dividers?

- ◆ Some have been done with theorem prover, but messy
- ◆ BDD-based tools cannot handle complete algorithm
- ◆ Our own research shows promise

# Verifying Single Step

## Specification

- ◆ Define

$$P_i = PS_i + PC_i$$

$$P_{i+1} = PS_{i+1} + PC_{i+1}$$

- ◆ Input Constraint

$$|P_i| \leq 8/3 D$$

- ◆ I/O Relation

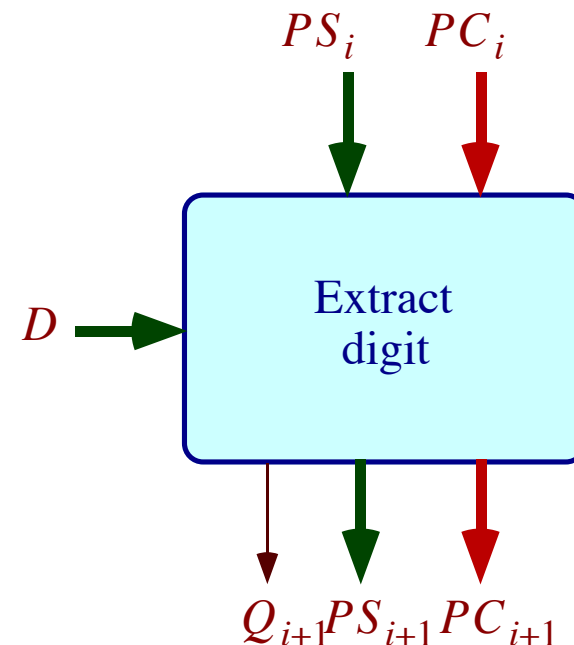
$$P_{i+1} = 4 [P_i - Q_{i+1}D]$$

- ◆ Output Constraint

$$|P_{i+1}| \leq 8/3 D$$

## Verification

- ◆ Can do with OBDDs
- ◆ Create “checker circuits”
- ◆ Would uncover Intel’s problem



# System Considerations

## Microprocessor Complexity Increasing at Rapid Pace

- ◆ Pentium has features historically found in high-end mainframes
  - Hardware support for multiplication, division, square root, and transcendental functions
  - Sophisticated instruction pipeline
- ◆ Exercising all possible system behaviors especially difficult
- ◆ Future generations even more complex
  - Speculative execution

## Must Get it Right First Time

- ◆ Cannot send field service to replace boards
- ◆ Typically have more people working on verification than on design

## Ubiquitous Systems of Unprecedented Complexity

- ◆ Industry's headaches caused by own successes

# Impact on Intel

## Short Term

- ◆ **Embarassing to make mistake**
- ◆ **Bad PR due to poor crisis management**
- ◆ **\$475 Million is a lot of money**
  - Still had very profitable year, including \$372 Million for 4Q94

## Long Term

- ◆ **Intel still way ahead of competition**
  - No one has cloned Pentium yet
  - Intel has announced next generation processor
  - 75% of \$11 Billion microprocessor market
- ◆ **Everyone will be more careful in the future**
  - Good opportunities for research on formal verification
- ◆ **Intel benefits even from this publicity**
  - as playing key role in computer industry

# Impact on Society

## Reminded of Importance of Computing

- ◆ Have come to expect ever faster & cheaper machines
- ◆ Rely heavily on correctness of results

## The Power of the Internet

- ◆ Created *ad-hoc*, international group of collaborators
  - U.S., Germany, Norway, Canada
  - University, industry
  - No external control or authority
- ◆ Rapid dissemination of information and results
- ◆ Caught Intel by surprise
  - Had never dealt with force of this kind
- ◆ System does not scale
  - A few gems among the drivel