Fault Tolerant Computing

Prof. David August/Prof. David Walker

2

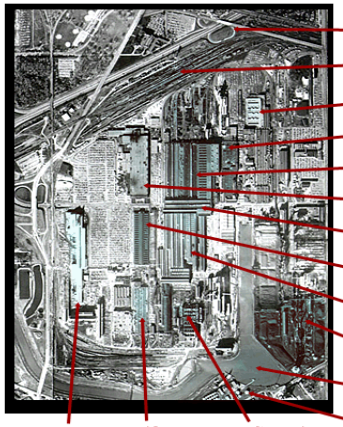
fault-tol·er·ant \ˈfȯlt-ˈtäl(-ə)-rənt\
adj : able to function in the absence of a major component

3

Transistors

4

Without the Transistor…



http://www.ominous-valve.com/vtsc.html

Basic MOSFET Transistor

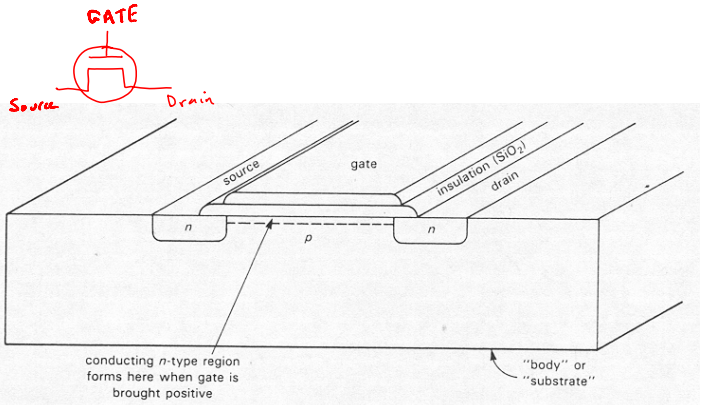


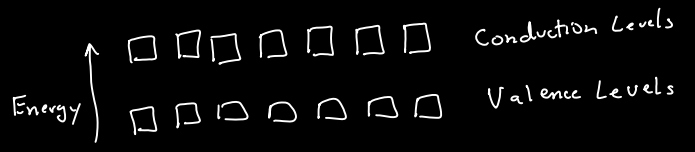
Figure 3.5. An *n*-channel MOSFET.

6

Semiconductors

Pure semiconductors are insulating (at low temps)

- Valence levels are filled and can't conduct
- Conduction levels are empty and can't conduct

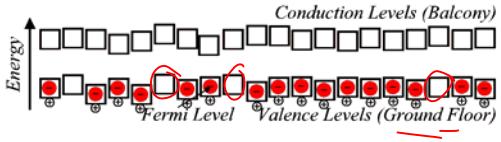


- Impure semiconductors can be conducting

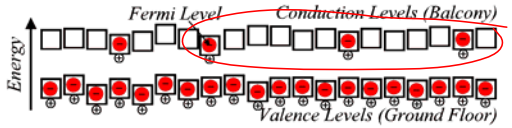
7

Impure/Doped Semiconductors

P-type: Substitute atoms with more empty orbitals



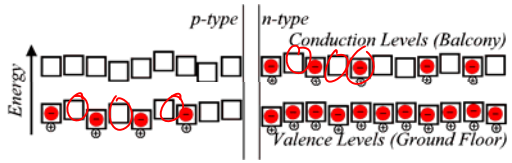
N-type: Substitute atoms with more filled orbitals



8

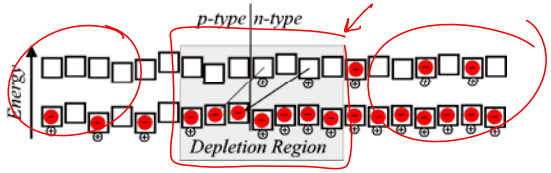
The PN-Junction

Before P-type meets N-type:



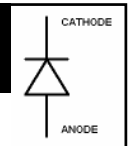
After P-type meets N-type:

- Depletion region is electrically polarized

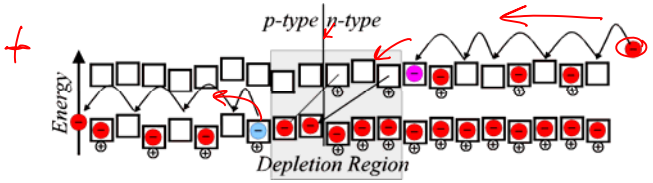


9

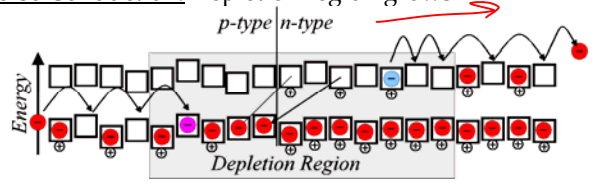
A Diode!



Forward Conduction: Depletion region shrinks



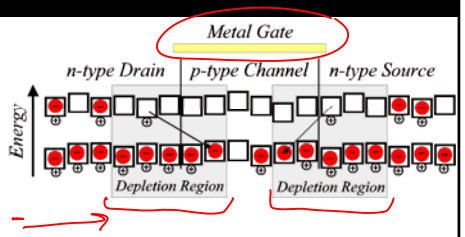
Reverse Conduction: Depletion region grows



10

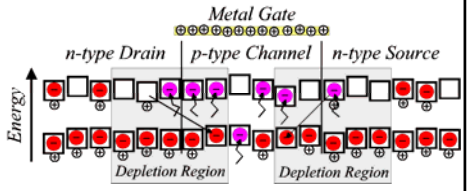
The MOSFET Transistor

Off



On

- Gate charge: changes the channel type
- Entire device becomes one

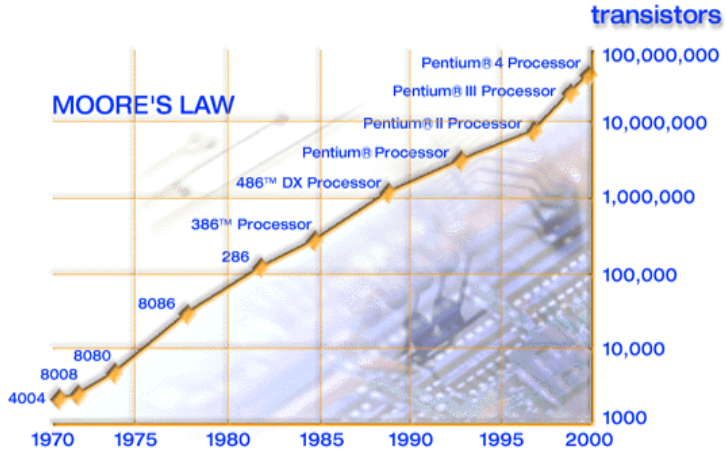


We have our switch!!!

11

Moore's Law

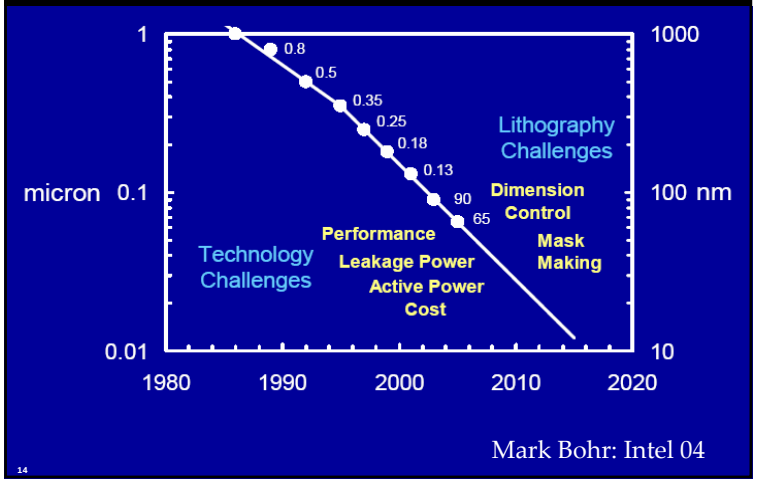
12



Processor chip and packaging is roughly the same size, so…

13

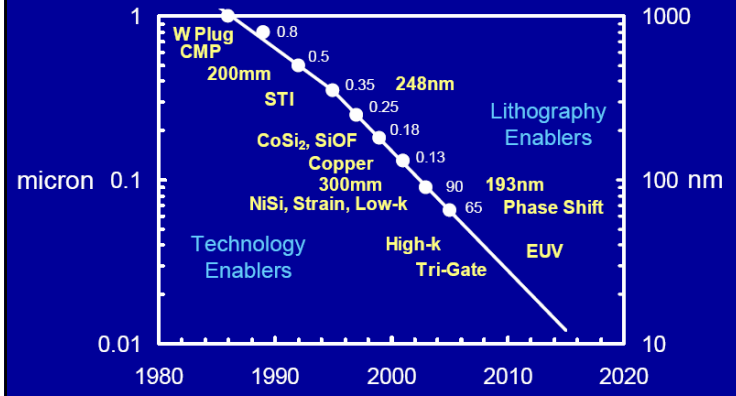
Scaling Gets Tougher…



Mark Bohr: Intel 04

14

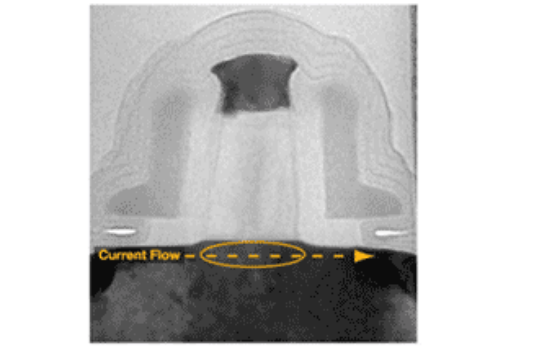
A Sequence of One-off Solutions



Mark Bohr: Intel 04

15

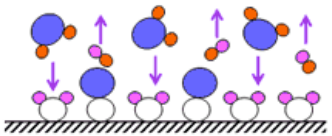
Just In Time…Strained Silicon



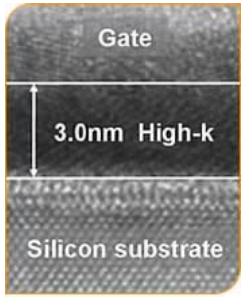
http://www.intel.com/technology/silicon/si12031.htm

16

Just In Time…High-k/Metal Gate



High-k materials are deposited one molecular layer at a time.



The middle layer, called the gate dielectric, is a thick, high-k material which controls leakage yet gives excellent transistor performance characteristics.

Air has k=1. "High-k" materials, such as hafnium dioxide (HfO2), zirconium dioxide (ZrO2) and titanium dioxide (TiO2) inherently have a dielectric constant or "k" above 3.9, the "k" of silicon dioxide.

http://www.intel.com/technology/silicon/si11031.htm

17

Just In Time…Tri-Gate Transistor

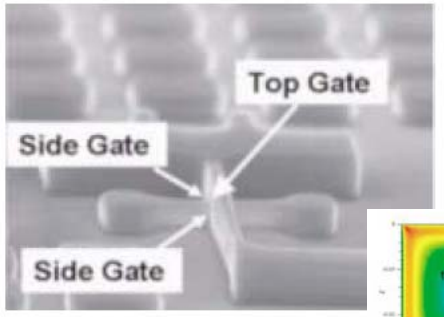


Figure 1. Photo of a 30-nm tri-gate tra

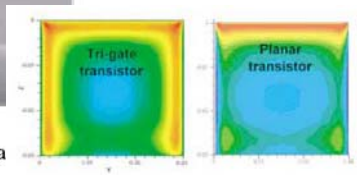
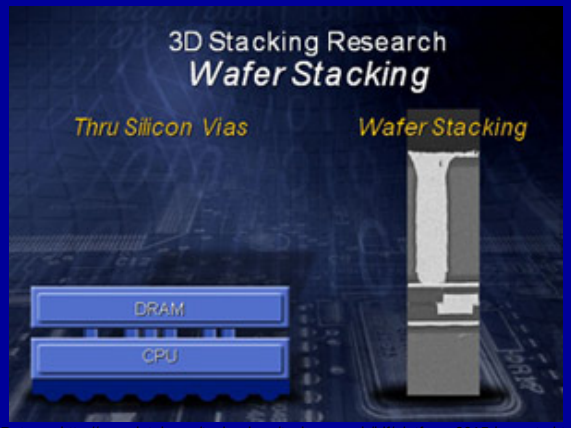


Figure 2. This simulation of a cross-section of silicon channel shows much more current flow (indicated by red) in a tri-gate transistor than in a planar transistor. Current flows into/out of the paper.

http://www.intel.com/technology/magazine/silicon/si07031.pdf

18

Just In Time…Wafer Stacking



Justin Ratner - http://www.intel.com/technology/techresearch/idf/platform-2015-keynote.htm

19

Chaos

20

Los Alamos National Lab
ASC Q 2048-node supercomputer
crashes regularly [Michalak 2005]

21

In 2000, Sun server systems deployed to America Online, eBay, and others crash [Baumann 2002]

22

Forbe's Magazine, November 13, 2000: "It's ridiculous. I've got a $300,000 server that doesn't work. The thing should be bulletproof," says [Bell South's] president.

23


All News

home / ZDNet news

Technology News

SEARC

Sun tries to cope with server flaw

By David Hamilton, The Wall Street Journal Online
Published on ZDNet News: November 7, 2000, 4:00 PM PT

TALKBACK

Worthwhile? + 0

0 VOTES

ZDNet Tags: Sun Microsystems Inc

For the past year, Sun Microsystems Inc. has struggled to solve a mysterious fault that can cause its high-end servers to crash unexpectedly, an embarrassing problem for a computer maker that routinely refers to its servers as "rock solid" reliable.

The Palo Alto, Calif., company said the problem, which was eventually traced to a memory flaw, is rare and probably affects fewer than 1% of all computers it has sold. Since the problem was first identified in its servers, the large computers often used to manage databases and handle e-commerce tasks, Sun (sunw) engineers have put together a variety of hardware and software fixes that appear to reduce the risk of spontaneous crashes. Sun

24

"it was found that a single soft fail … was causing an entire interleaved system farm (hundreds of computers) to crash." [Cypress Semiconductor reports in 2004, SER: History, Trends, and Challenges 2004]

25

"a single soft error brought a billion-dollar automotive factory to halt every month." [Cypress Semiconductor reports in 2004, Mukherjee 2007]

26

History Provides a Clue

1954-1957: Discovery of soft fails in digital electronics – Nuclear Bomb Testing

*1975: Soft errors in satellites from **solar particles** – 4 errors in 17 satellite-years – Hughes Aircraft*

27

History Provides More Clues

*1978: Discovery of soft errors from **alpha-particles** – "May and Woods Incident"*
- Intel 2107 series 16Kb DRAMS.
- Trace radioactivity in ceramic packaging
- Downstream from uranium mine in CO!

1981: IBM experiences soft error problems
- 16 Kbit DRAM memory chips
- Radioactive Kr85 contaminating the packaging
- Chemical used to clean the containers holding acid
- Approximately 2% of the modules contaminated

1992: Radioactive thorium in bat guano!
- Bat cave near Louisiana mine producing raw material for phosphoric acid used to etch aluminum wires in chip
- Shut down semiconductor factory for 8 weeks

28

Today's Problem Foretold by Profits

1978: IBM predicts soft errors from cosmic rays

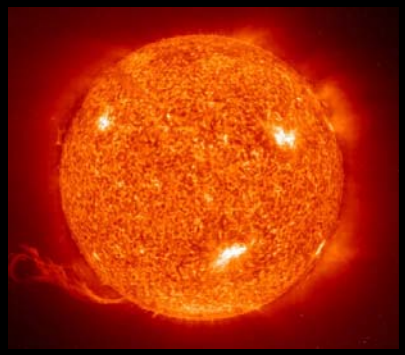
"An alpha-particle can cause a sudden burst of 1M electrons in a semiconductor over a path length of a few microns. This was the dimension of the new 16Kb FET memory cells." [Zeigler et al. 1998]

29

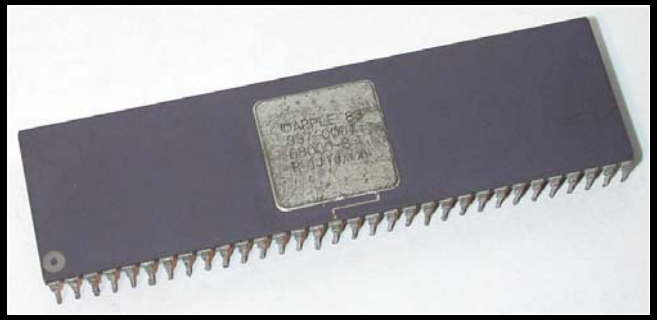
Causes

30

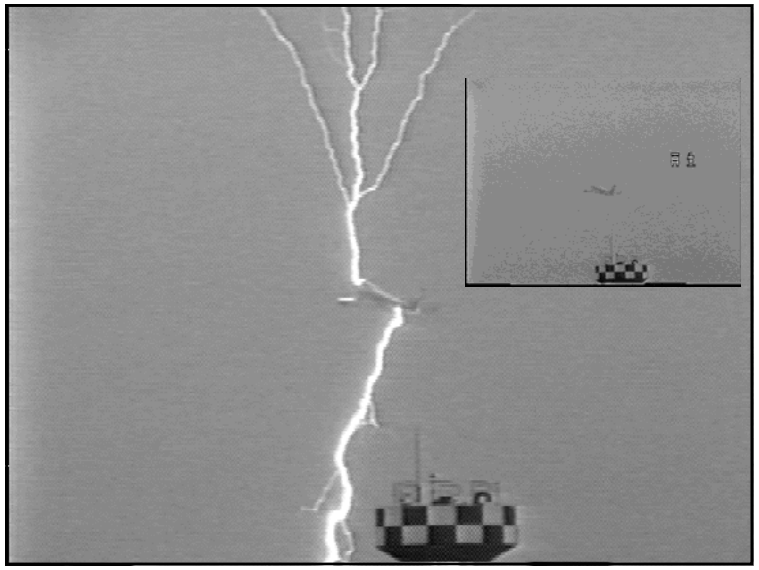
31

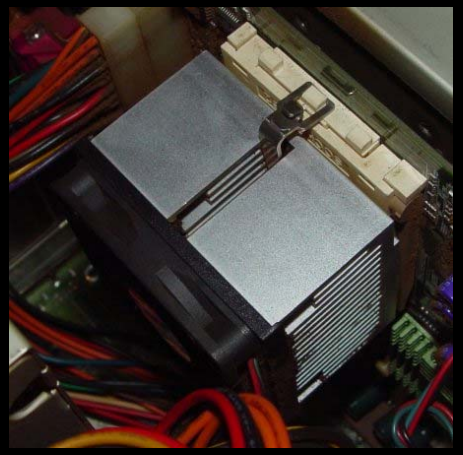


32



33





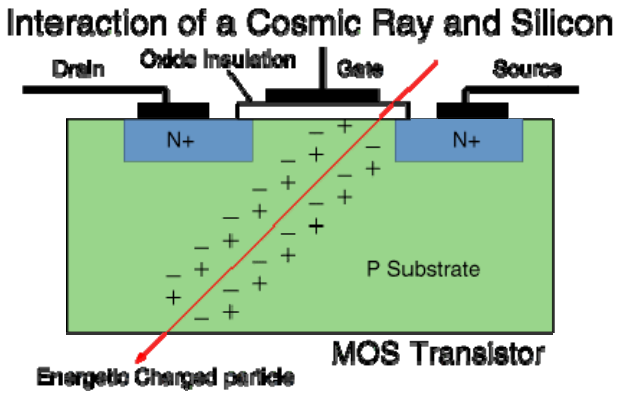
35

36

…and
Moore's Law

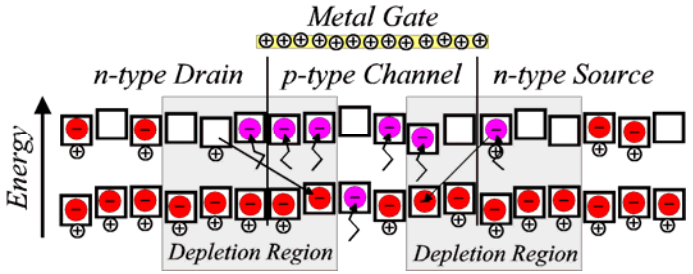
37

Smaller Transistors



38

Recall How They Work…



39

Soft Errors

- For a soft error to occur at a node in a circuit, the collected charge Q at that node must be more than $Q_{critical}$.
- $Q_{critical}$ is proportional to the node capacitance and the supply voltage.
- As CMOS device sizes decrease, the charge stored at each node decreases (due to lower nodal capacitance and lower supply voltages).

40

Soft Errors

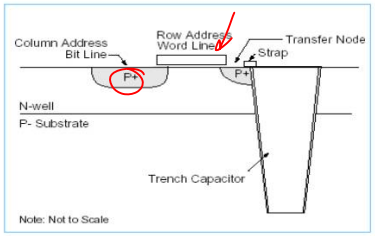
- As silicon process shrinks, the node area decreases, lowering the probability of a particle hitting. This lowers per bit error rate.
- There are more bits per chip, so the system is now less reliable.

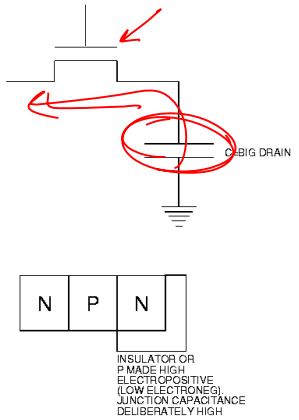
41

Moore's Law as it relates to reliability:
Error rate doubles every generation!

42

Memory Forgets

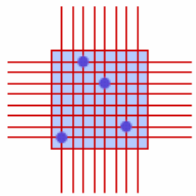




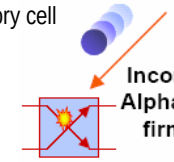
43

FPGA: Firm Errors

- FPGA is logic based on memory cell



GRM: General Routing Matrix



Incoming neutron or Alpha particle causes firm error in GRM

Firm error leads to . . .

misrouted signal

or missing signal

44

Impact of Elevation

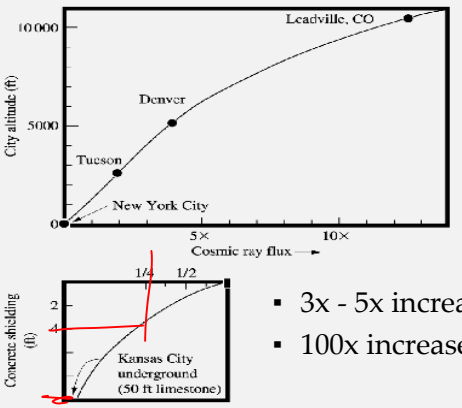


Figure 8, Ziegler, et al., "IBM experiments in soft fails in computer electronics (1978 - 1994)," IBM J. of R. & D., Vol. 40, No. 1, Jan. 1996.

- 3x - 5x increase in Denver
- 100x increase in airplanes

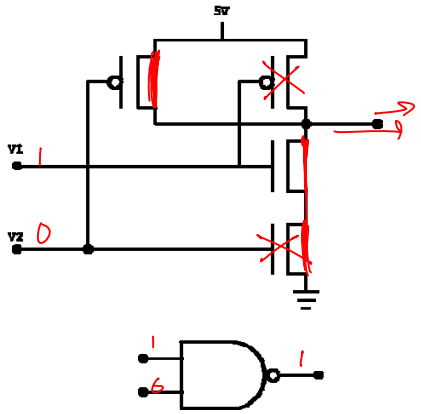
45

Aircraft Control

- Altitude of 30,000 feet - 100x error rate
- Use four 1M 130nm SRAM-based FPGA (0.074 upsets/day)
- One system per aircraft
- 4,000 flights over the north pole/day (increased neutron flux)
- 37 aircraft will experience an FPGA configuration error everyday!

46

Logic Failures



47

Logic Failures

Soft errors become hard truth for logic

By Ron Wilson David Lammers , EE Times
May 03, 2004 (9:39 AM EDT)
URL: http://www.eetimes.com/article/showArticle.jhtml?articleId=19400052

Phoenix — Those nasty neutrons that have plagued memory chip designers for the past two decades are now giving logic designers a headache, too. But while error correction coding has reduced soft-error rates (SERs) in DRAMs and SRAMs, no such quick fix exists for logic, and all current solutions involve extra cost and a drag on performance.

"Logic SER may become as significant as SRAM error rates," predicted Hans Stork, the chief technology officer at Texas Instruments Inc. (Dallas), in a keynote speech here last week at the International Reliability Physics Symposium.

Soft errors in logic devices are a growing concern for mission-critical systems such as servers, automotive ICs and networking equipment. Logic chip vendors already are working with system customers on ways to guard against the effects of cosmic rays and alpha particles emitted from packaging.

However, reliability engineers at last week's symposium said no easy solutions exist.

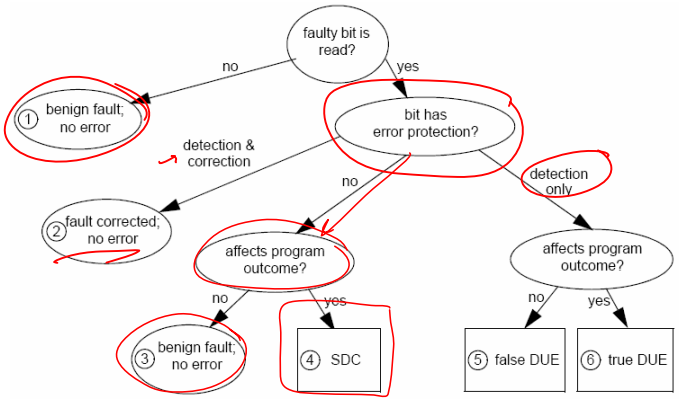
In the case of ASIC designs, the implications are different and the countermeasures reach deeper into the system design.

48

Measuring

49

Classification



50

Important Definitions

- SDC: Silent Data Corruption
- DUE: Detected & Unrecoverable Error
  - TRUE DUE: Affects Output
  - FALSE DUE: Doesn't Affect Output
- SER: Soft Error Rate (SDC + DUE)

51

Measuring Faults: Interval Based

- MTTF = Mean Time to Failure
- MTTR = Mean Time to Repair
- MTBF = Mean Time Between Failures = MTTF + MTTR
- Availability = MTTF / MTBF
- MITF = Mean Instructions to Failure [Mukherjee]
- MWTF = Mean Work to Failure [Reis]
- Performability = MWTF

52

Measuring Faults: Rate-Based

- FIT = Failure in Time
- 1 FIT = 1 failure in a billion hours
- 1 year MTTF = 10^9 / (24 * 365) FIT = 114,155 FIT
- SER FIT = SDC FIT + DUE FIT

53

Measuring a Chip's FIT

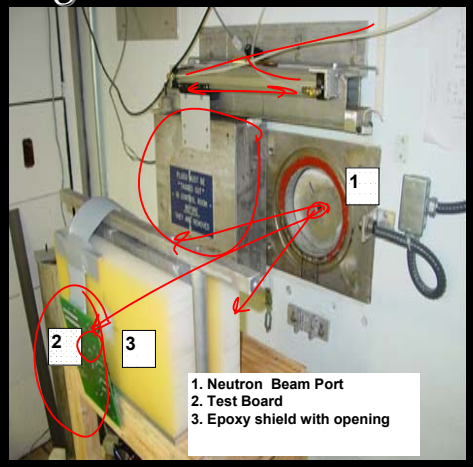
Chip	Physically bombard with neutrons from nuclear source (Accelerated Testing)
Circuit Models + RTL	Obtain raw error rate Statistical fault injection

54

Accelerated Testing

55

Testing at Outer Core



56

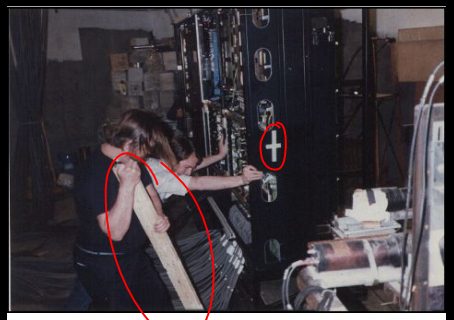


Figure 7

Fine adjustment of a cache module in a neutron beam. This photograph shows adjustment of the tester to cause a beam of neutrons to hit a bipolar memory module. The cross-mark indicates the desired position. The equipment in the right foreground is the instrumentation to measure the intensity of the neutron beam when it is turned on.

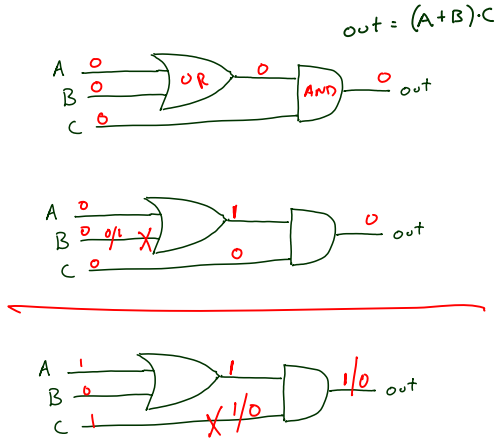
Figure 7, Ziegler, et al., "IBM experiments in soft fails in computer electronics (1978 - 1994)," IBM J. of R. & D., Vol. 40, No. 1, Jan. 1996.

57

Solutions

58

Masking (Benign Faults)



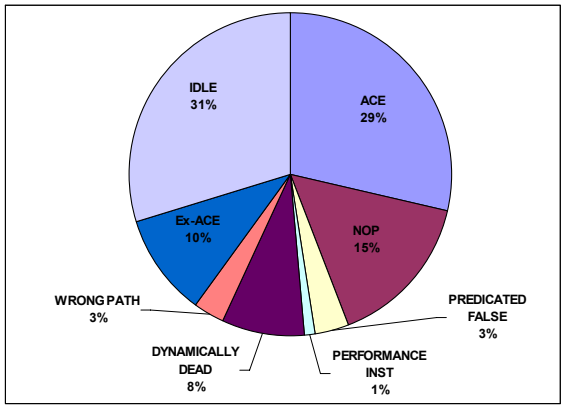
59

Not All Errors Result in a Problem

- Architectural Correct Execution (ACE)
- Some bits are ACE
- Some are unACE
  - Logical masking
  - Performance enhancing instructions
  - NOPs
  - Wrong-path instructions
  - Idle units
  - Dynamically dead instructions and registers

60

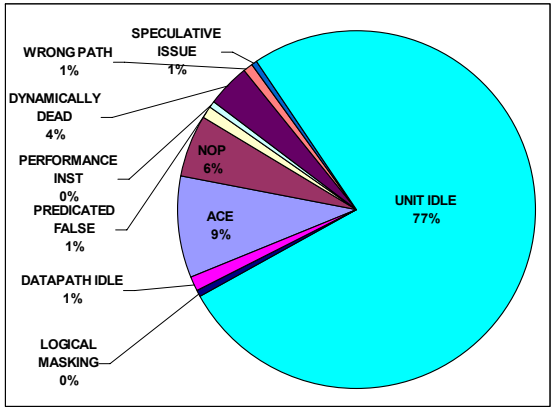
Instruction Queue/AVF Concept



ACE percentage = AVF = 29%

61

Functional Units



ACE percentage = AVF = 9%

62

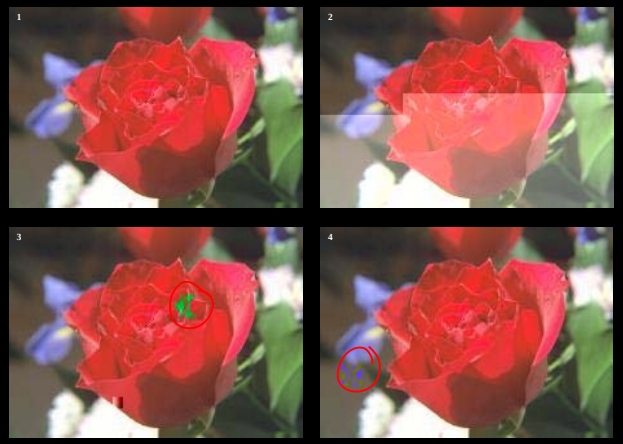
Computing FIT Rate of a Chip

Total FIT = $\sum$ (FIT per bit$_i$ X # of bits$_i$ X AVF$_i$)

Structure	FIT per bit	# of bits	AVF	Total FIT
Branch Predictor	.001	1K	0	0
Program Counter	.001	64	1	0.064
Instruction Queue	.001	6400	**0.29**	**?**
Funtional Units	.001	4000	**0.09**	**?**
…				…
Total FIT of whole chip				= Σ column

63

Not All Errors are Worth Fixing



64

Technology & Circuit

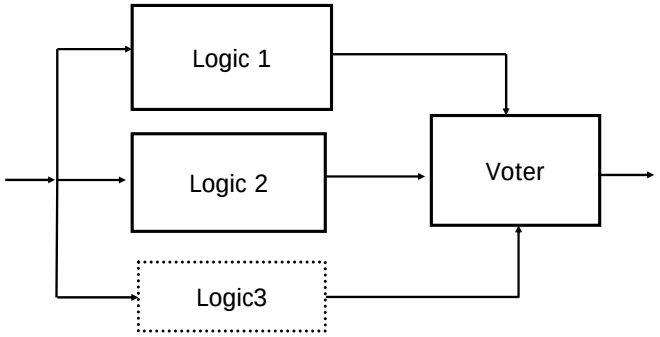
65

Physical Solutions are Hard

- Shielding?
  - No practical absorbent (e.g., approximately > 10 ft of concrete)
- Radiation-hardened cells?
  - Improvement possible with significant penalty in performance, area, cost

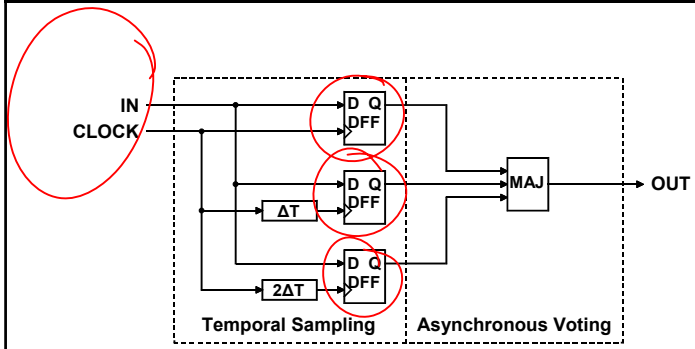
66

Redundant Logic



67

Time Redundancy



68

Design Tradeoffs

- Chip layout area penalties
  - Latch areas increase from ~3x to >5x
- Operating frequency penalties
  - Setup time increases by twice the sampling ΔT
  - Maximum frequency dependency:

$$1/F_1 = 1/F_0 + 2\Delta T$$

F_1 and F_0 = maximum frequencies

ΔT = Sampling delay time

69

Fault Protection is Expensive

- IBM historically adds 20-30% additional logic for mainframe processors for fault tolerance [Slegel 1999]
- In 2003, Fujitsu released SPARC64 with 80% of 200,000 latches covered by transient fault protection [Ando 2003]

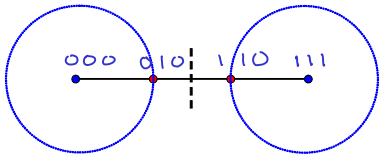
70

Memory Protection

71

Error Correcting Codes

The hamming distance between code words is 3.



Allows:

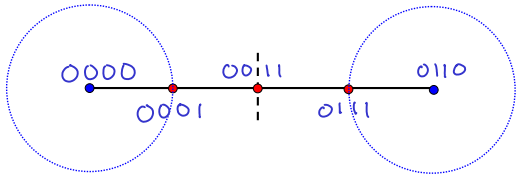
Error DETECTION for Hamming Distance = 1.

Error CORRECTION for Hamming Distance =1

For errors of Hamming distances greater than 1 an error gives a false correction.

72

More Dots…



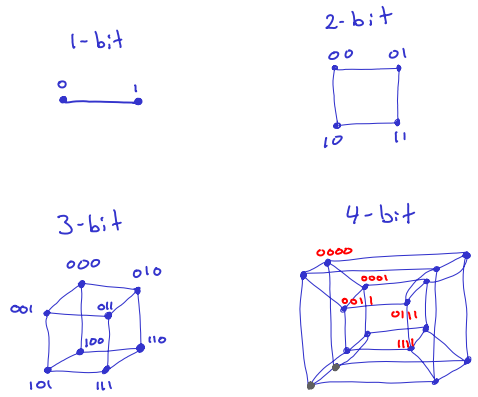
Allows:
Error DETECTION for Hamming Distance = 2.

Error CORRECTION for Hamming Distance = **1.**

SECDED

73

Hybercubes



74

Error Correction Codes

- Hamming Created Correction Concept in 1950's
- Provides correction, not just detection

$$\begin{bmatrix}1010101\\0110011\\0001111\end{bmatrix}\begin{bmatrix}c0\\c1\\d0\\c2\\d1\\d2\\d3\end{bmatrix}=\begin{bmatrix}S_0 S_1 S_2\end{bmatrix}$$

a) Hamming Matrix, Data, and Syndrome

$$c_0 \oplus d_0 \oplus d_1 \oplus d_3 = S_0$$
$$c_1 \oplus d_0 \oplus d_2 \oplus d_3 = S_1$$
$$c_2 \oplus d_1 \oplus d_2 \oplus d_3 = S_2$$

b) Syndrome Formation Equations

75

Error-Correcting Codes

Example: Hamming Codes

$$P_1 P_2 B_3 P_4 B_5 B_6 B_7$$

$$P_1 \oplus B_3 \oplus B_5 \oplus B_7 = 0$$
$$P_2 \oplus B_3 \oplus B_6 \oplus B_7 = 0$$
$$P_4 \oplus B_5 \oplus B_6 \oplus B_7 = 0$$

e.g. If B3 flips

1
1 = 3
0

with

$2^K >= m+k+1$. m # data bit, k # check bit
For 64 data bits, needs 7 check bits

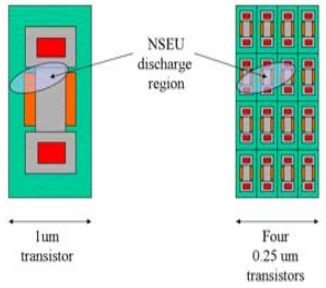
76

Overhead for ECC

Word Length	Number ECC Bits	Area Increase for ECC bits	Delay (EXOR-Gate Tree Depth)
16	5	31%	4
32	6	19%	5
64	7	11%	6
128	8	6%	7
256	9	3.5%	8

77

Multiple Bit Upsets

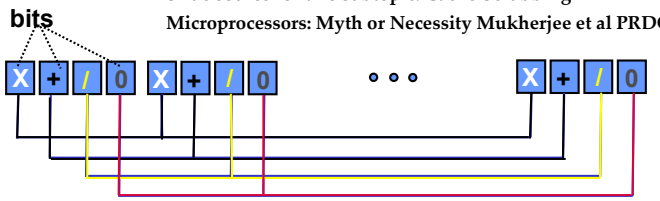


- SECDEC codes ineffective for MBUs
- Serious Problem

78

Interleaving bits

Slide source for this subtopic: Cache Scrubbing in Microprocessors: Myth or Necessity Mukherjee et al PRDC'04

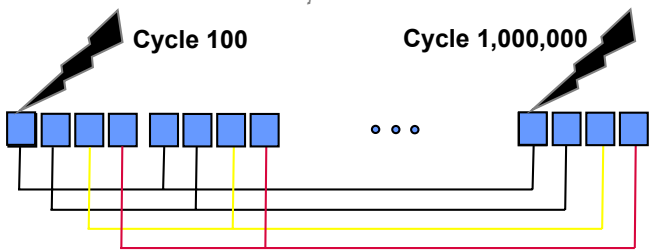


X = covered with single ECC code
+ = covered with different ECC code

Interleaving converts spatial multi-bit error → multiple single bit errors

79

Two Separate Strikes on Different Bits
Temporal Double Bit Errors



- SECDED ECC (single error correction, double error detection)
  - could detect error, but cannot correct the error
  - if errors accumulate single bit correctable error becomes a double bit detectable error

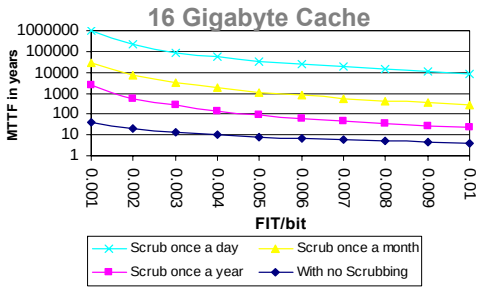
80

Solutions for Temporal Multiple Errors

- Natural Effects
  - Whenever a processor reads a cache block, we can correct the single bit error
  - Check for errors when cache blocks are replaced from the cache
- More Powerful ECC
- Scrubbing
  - Periodically read memory and correct all single bit errors
  - Disallows accumulation of temporal double bit errors
  - Standard technique in main memories (DRAMs)

81

Impact of Scrubbing on Temporal Double Bit MTTF



- For 16 gigabytes of cache, scrubbing can help
- IBM DUE MTTF goal of 10 years

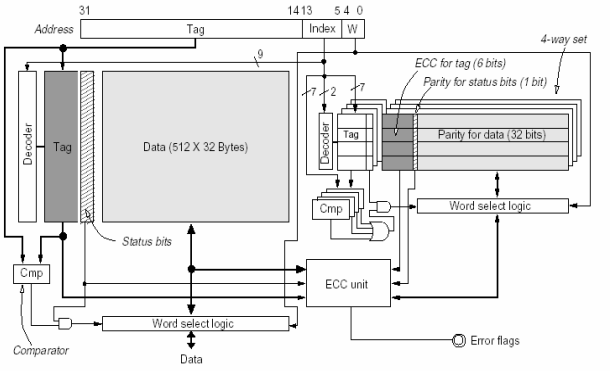
82

Efficient Cache Integrity Approaches

- Parity Caches [Kim et. al. ISCA 1999]
- Shadow Checking [Kim et. al. ISCA 1999]
- In Cache Replication [Zhang et.al. DSN 2003]

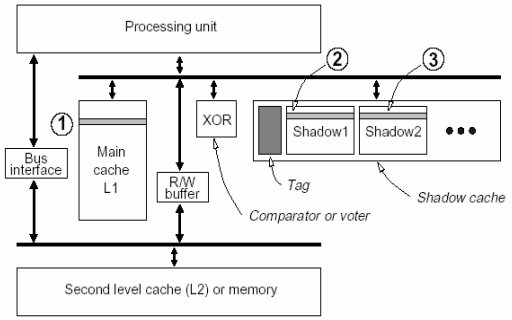
83

Parity Caches



84

Shadow Checking



85

ICR: In-Cache Replication

Simple Idea:

Replicate actively used data within the cache by evicting data that may not be needed.

86

ICR: Exploring the Design Space

- How aggressively to predict dead blocks?
- When to replicate?
- Where to replicate?
- How aggressively to replicate?
- How many replicas do we need?
- How to protect cache blocks?
- How to place a replica in a set?

87

Recall Sun Microsystem's Problems

- In 2000, Sun server systems deployed to America Online, eBay, and others crash.
- Cache memory is a large target for cosmic rays
- UltraSPARC II ECC mechanism was defective
- Proved damaging to Sun's image

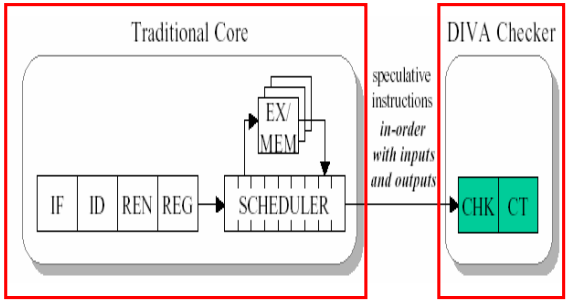
88

Microarchitecture

89

DIVA[Austin Micro 99]

Dynamic Implementation Verification Architecture

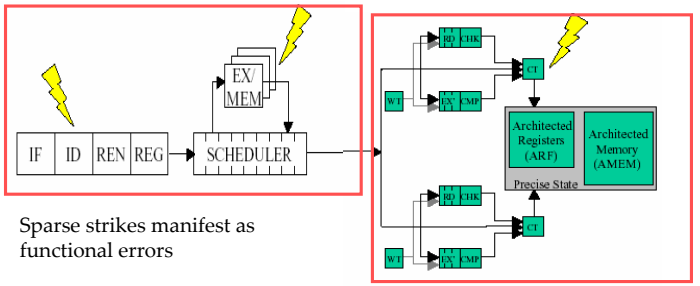


Lifts the burden of correctness from core processor

- All core computation, communication, control is speculative
- Tolerates design errors, electrical faults, silicon defects, and failures

90

DIVA for Soft Error Protection



Sparse strikes manifest as functional errors

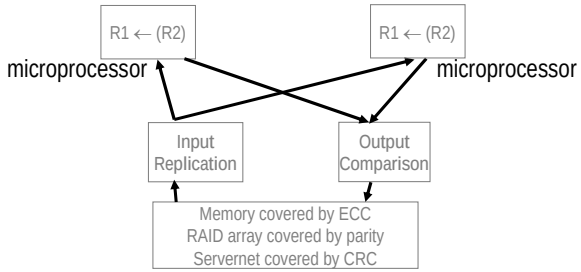
Rad-hard checker detects and corrects faults

Small checker will provide natural resistance to SER (small target!)

Or, replicate the checker logic, restart pipes on disagreement

91

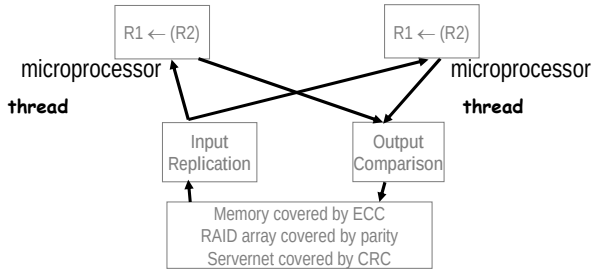
Lockstepping [HP Himalaya]



Replicated Microprocessors + Cycle-by-Cycle Lockstepping

92

Lockstepping with Threads?



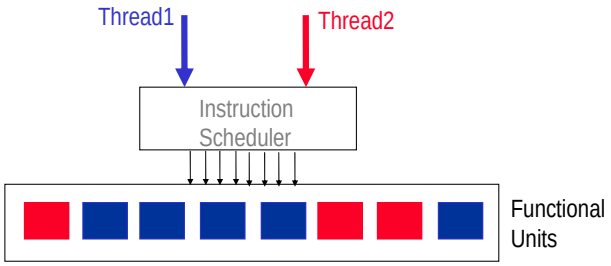
Replicated ~~Microprocessors~~ + ~~Cycle-by-Cycle Lockstepping~~

thread

?

93

Simultaneous Multithreading (SMT)



Examples: Alpha 21464, Intel Northwood, Sun Niagara

94

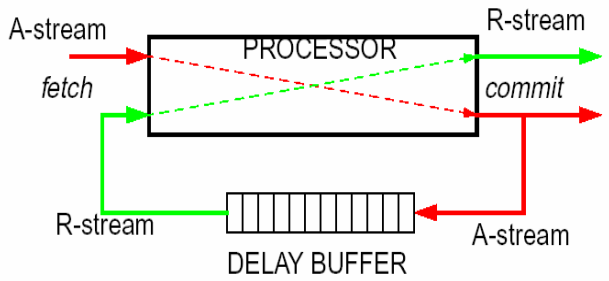
Redundant Multithreading (RMT)

RMT = Multithreading + Fault Detection (& Recovery)

	Multithreading (MT)	Redundant Multithreading (RMT)
Multithreaded Uniprocessor	Simultaneous Multithreading (SMT)	Simultaneous & Redundant Threading (SRT)
Chip Multiprocessor (CMP)	Multiple Threads running on CMP	Chip-Level Redundant Threading (CRT)

95

AR-SMT [Rotenberg FTCS 99]



"A" => "**A**ctive stream"
"R" => "**R**edundant stream"
"SMT" => "**S**imultaneous **M**ulti **T**hreading"

96

AR-SMT Delay Buffer

- Simple, fast, hardware-only state passing for comparing thread state
- Ensures time redundancy: the A- and R-stream copies of an instruction execute at different times
- Buffer length adjusted to cover transient fault lifetimes
- Delay Buffer contains perfect "predictions" for R-stream!

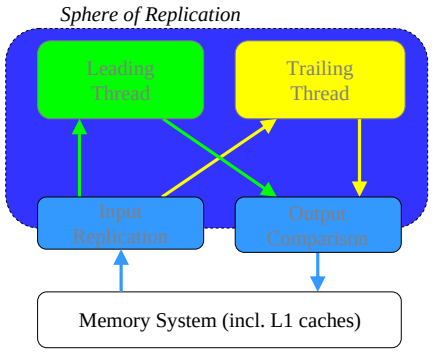
97

AR-SMT Fault Detection/Recovery

- Fault detected when thread state does not match
- Committed R-stream state is checkpoint for recovery
- Introducing a second, redundant thread increases execution time by only 10% to 30%

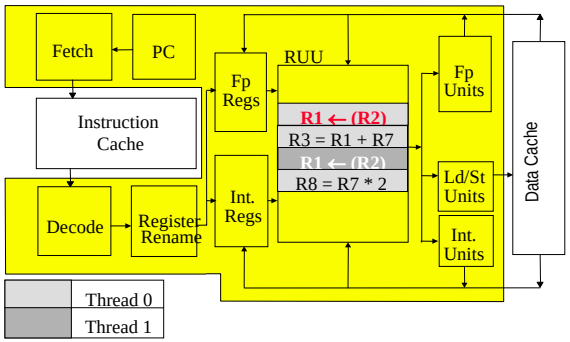
98

SRT: Sphere of Replication



99

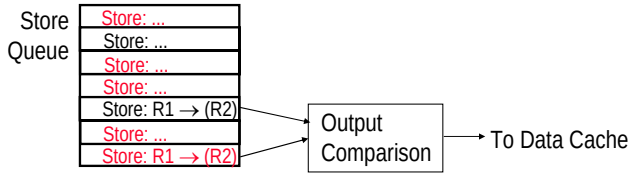
SRT: Sphere of Replication



100

SRT: Output Comparison

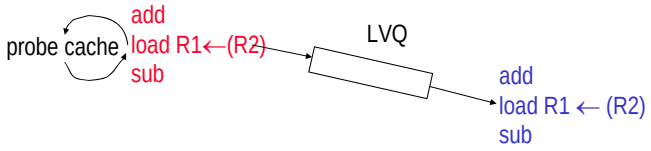
- <address, data> for stores from redundant threads
- Compare & validate at commit time



101

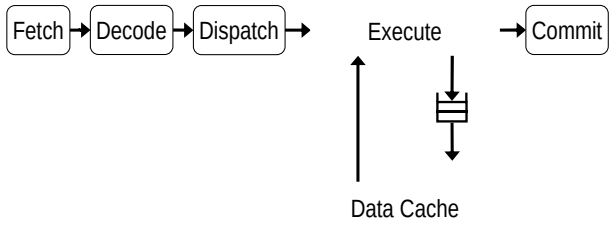
SRT: Input Replication

- Allow both loads to probe cache: false faults with I/O or shared mem
- Load Value Queue (LVQ)
  - pre-designated leading & trailing threads



102

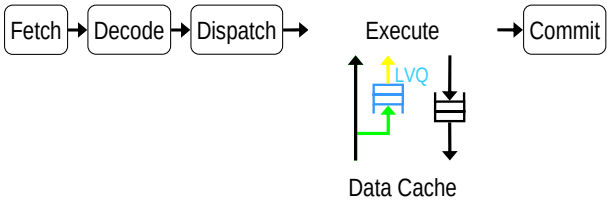
SRT: Basic Pipeline



Both leading & trailing threads would go through this pipeline

103

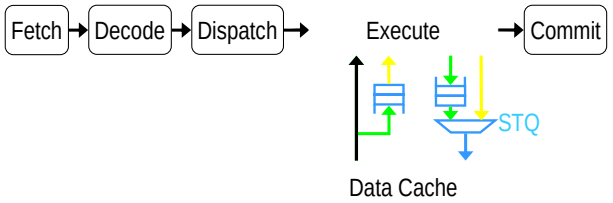
SRT: Load Value Queue (LVQ)



- Keep threads on same path despite I/O or shared memory
- Out-of-order load issue possible

104

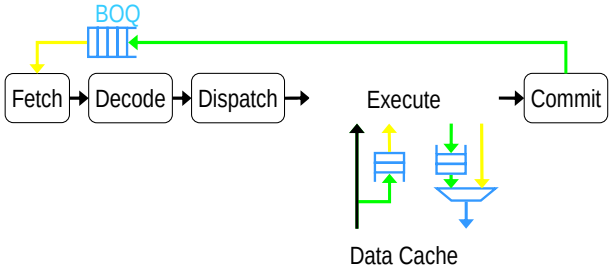
SRT: Store Queue Comparator (STQ)



- Compares outputs to data cache
- Catch faults before propagating to rest of system

105

SRT: Branch Outcome Queue (BOQ)



- Forward leading-thread branch targets to trailing fetch
- 100% prediction accuracy in absence of faults

106

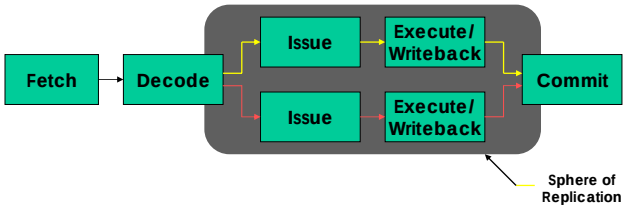
SRT: Temporal Redundancy

- Provides protection for random logic from transient faults
- Execute multiple copies of same instruction over time
  - Minimal hardware overheads
- Performance Overheads
  - 30% IPC drop on superscalar (Ray et al. '01)
  - 21% IPC drop on SMT (Vijaykumar et al. '02)
- Reason for IPC drop?
  - Resource contention

107

Dual Instruction Execution (DIE)

- DIE [Ray et al., MICRO'01]
- Instructions Replicated at Decode



- Computation Checked at Commit

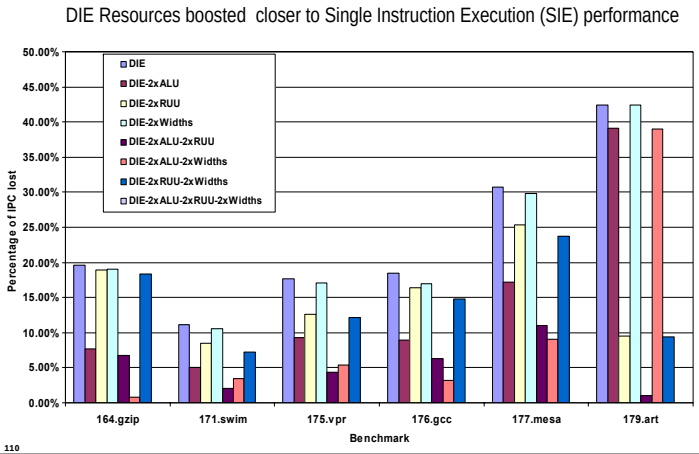
108

DIE: Processor Resources

- Contended Resources
  - ALUs
  - Issue Window/ROB/PRF (RUU) entries
  - Decode/Issue/Commit Bandwidth
- Un-contended Resources
  - Fetch Unit
  - Memory Ports

109

DIE: Resource-Boosting Results



110

Software/Compiler

111

Software Techniques

- Can be applied to existing hardware
- Can be applied to existing applications
- Can be used today to increase reliability

112

Basic Philosophy

If a tree falls in the forest,
but nobody is around to hear it,
does it make a sound?

If a fault affects some data,
but does not change the output,
does it make a error?

Only store operations effect output,
so validate data before stores.

113

SWIFT [Reis et al.], EDDI [Oh et al.]

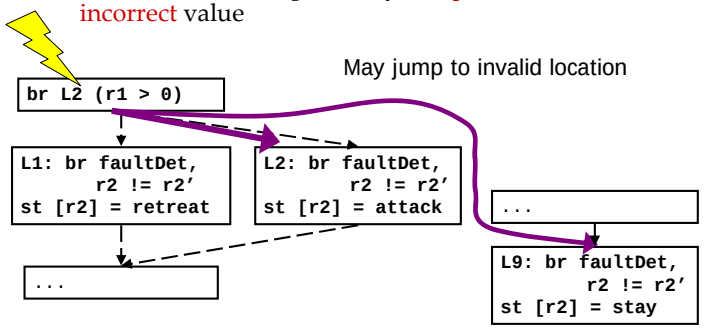
```
ld  r1  = [r2 ]

add r1  = r1 +1



st [r2 ] = r1
```

```
ld  r1  = [r2 ]
ld  r1' = [r2']
add r1  = r1 +1
add r1' = r1'+1
br faultDet, r1 != r1'
br faultDet, r2 != r2'
st [r2 ] = r1
```

114

Control Flow: Next PC

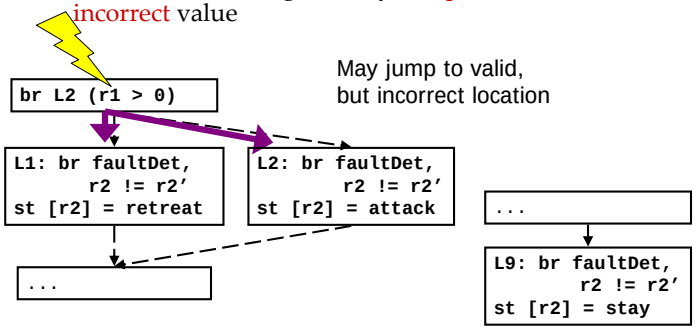
- Only have one control path during execution
  - Incorrect control flow will divert both versions
  - Redundant and original may compute the same incorrect value



115

Control Flow: Condition

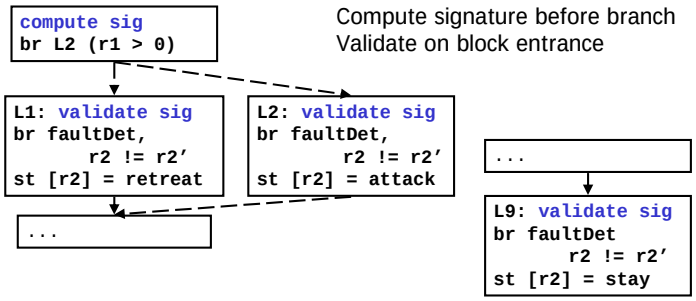
- Only have one control path during execution
  - Incorrect control flow will divert both versions
  - Redundant and original may compute the same incorrect value



116

Control Flow: Compute Signature

- Only have one control path during execution
  - Incorrect control flow will divert both versions
  - Redundant and original may compute the same incorrect value



117

Control Flow Protection: Example

- Compute signature before branch
- Validate on block entrance

```
br L_T (r1' > 0)
r_sig = 1
jmp L_F
L_T: r_sig = 2
L_F: br L2 (r1 > 0)
```

Block 1

```
L1: br faultDet
      r_sig != 1
br faultDet,
      r2 != r2'
st [r2] = retreat
```

Block 2

```
L2: br faultDet
      r_sig != 2
br faultDet,
      r2 != r2'
st [r2] = attack
```

```
...
```

```
...
```

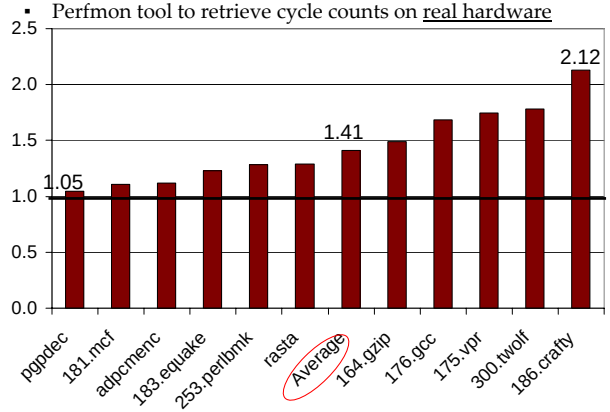
Block 9

```
L9: br faultDet
      r_sig != 9
br faultDet
      r2 != r2'
st [r2] = stay
```

118

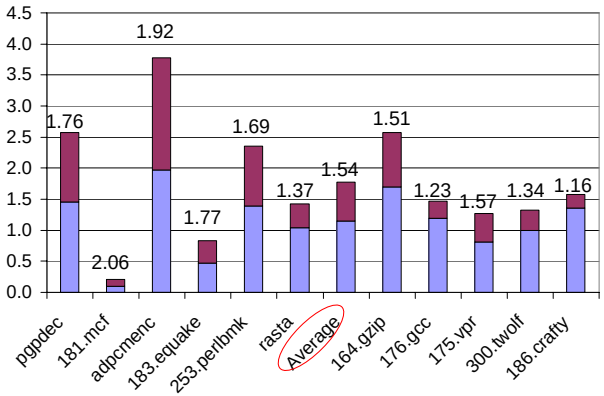
SWIFT Performance

- OpenIMPACT compiler targeted for Intel Itanium 2
- Perfmon tool to retrieve cycle counts on real hardware



119

Performance Evaluation: IPC



120

Reliability Evaluation

	Correct	Fault Detected	Abnormal Execution	Incorrect
		Fault Detected		Incorrect
Normal Program	81.35	00.00	10.76	7.89
SWIFT	74.86	23.33	1.69	0.12

Fault injection outcome distribution
(percent of all injections)

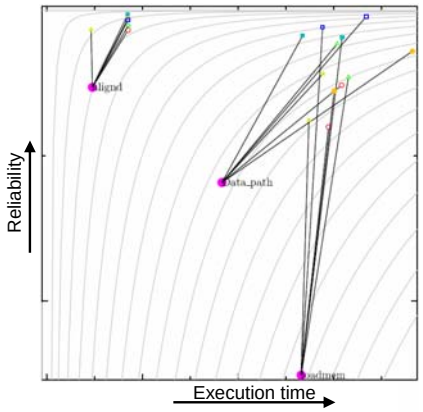
GOOD → BAD

95% confidence interval of ±0.31%, 5000 injections per benchmark

121

Software Modulated Fault Tolerance

- Software flexibility allows tradeoff between performance and reliability
- Tune redundancy based on function reliability
- Compared to best hybrid
  - Same reliability
  - Better performance
  - 6.4% speedup



122

Hybrid

123

Review of Instruction Level Techniques

Hardware solutions
- Lockstepping [Stratus DMR]
- RMT [Reinhardt & Mukherjee, ISCA '00]

Hardware cost
No application software changes
Fixed solution applied to all

Visibility into all state
More resources reduce performance degradation

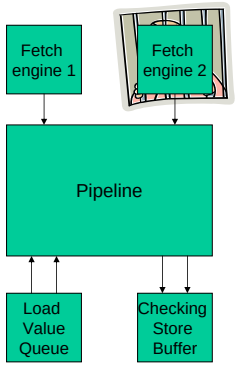
Software solutions
- EDDI, CFCSS [Oh et al.]
- Source-to-source [Rebaudengo et al.]

No hardware cost
Require software changes
Flexibility to continually trade off reliability and costs in the field
Visibility limited to architectural state
Fixed resources

Hybrid solutions take benefits from both
Tradeoff hardware, performance, and reliability

124

Redundant Multithreading



- Hardware-only approach
- Redundant code executes in separate hardware context
- Hardware requirements:
  - Multi-threaded machine
  - Load Value Queue
    - to ensure data loaded from memory is identical to both hardware contexts
  - Checking Store Buffer
    - compare both versions of data before committing data to memory
- No software changes
- Fixed redundancy for application
- Only half of the hardware contexts available to Operating System

125

Schedule

Thread 1	Thread 2
`ld r1 = [r2 ]`	
`add r1 = r1 +1`	
`st [r2 ] = r1`	

126

Schedule

Thread 1	Thread 2
`ld r1 = [r2 ]`	`ld r1 = [r2 ]`
`add r1 = r1 +1`	`add r1 = r1 +1`
`st [r2 ] = r1`	`st [r2 ] = r1`

127

Schedule

Thread 1	Thread 2
`ld r1 = [r2 ]`	`ld r1' = [r2']`
`add r1 = r1 +1`	`add r1' = r1' +1`
`st [r2 ] = r1`	`st [r2'] = r1'`

128

Schedule

Thread 1	Thread 2
`ld r1 = [r2 ]`	
`ld r1' = [r2']`	`ld r1' = [r2']`
`add r1 = r1 +1`	
`add r1' = r1' +1`	`add r1' = r1' +1`
`st [r2 ] = r1`	
`st [r2'] = r1'`	`st [r2'] = r1'`

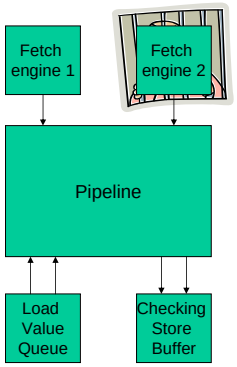
Instruction duplication, register allocation, and scheduling can be moved into software

Hybrid scheme:

CRAFT – CompileR Assisted Fault Tolerance [Reis et al.]

129

Hybrid Reliability



- Leverage reliability that software can provide
  - Compiler duplicates and schedules instructions, allocates registers
- Free up hardware thread resources
  - Work can be done on other thread
  - Applicable to single-threaded machines
- Maintain Input / Output hardware
  - Load Value Queue
  - Checking Store Buffer

130

Fault Detection Requirements

Mechanism to ensure original and redundant reads from memory receive same values

Mechanism to create redundant computation

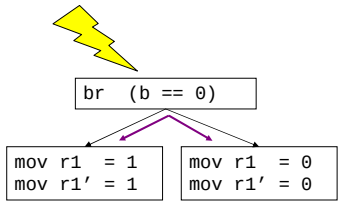
Mechanism to compare original and redundant results before writes to memory

Mechanism to guarantee correct control flow

131

Control Flow Protection

- Original and redundant computation in one thread
  - Incorrect control flow will divert both versions
  - Redundant and original may compute the same, but incorrect value
- Compiler adds instructions to compute redundant PC
  - Set before branch
  - Validate at destination
  - Not perfect, but effective



132

Fault Detection Requirements

Mechanism to ensure original and redundant reads from memory receive same values

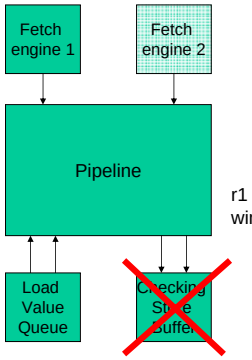
Mechanism to create redundant computation

Mechanism to compare original and redundant results before writes to memory

Mechanism to guarantee correct control flow

133

Removing the Checking Store Buffer



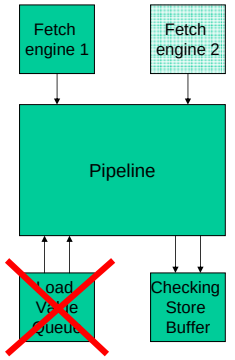

```
ld  r1  = [r2 ]
ld  r1' = [r2']

add r1  = r1 +1
add r1' = r1'+1
br faultdet, r1!=r1'
br faultdet, r2!=r2'
st [r2]  = r1
st [r2'] = r1'
```

r1 window

r2 window

CRAFT:LVQ
(still has Load Value Queue)

Removing the Load Value Queue



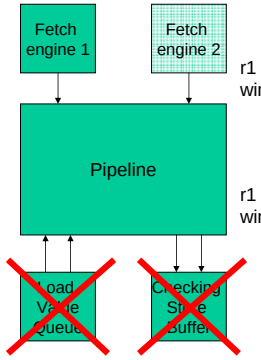

```
br faultdet, r2!=r2'
ld  r1  = [r2 ]
ld  r1' = [r2']
mov r1' = r1
add r1  = r1 +1
add r1' = r1'+1

st [r2]  = r1
st [r2'] = r1'
```

r1 window

r2 window

CRAFT:CSB
(still has Checking Store Buffer)

Removing Both Structures




```
br faultdet, r2!=r2'
ld  r1  = [r2 ]
ld  r1' = [r2']
mov r1' = r1
add r1  = r1 +1
add r1' = r1'+1
br faultdet, r1!=r1'
br faultdet, r2!=r2'
st [r2]  = r1
st [r2'] = r1'
```

r1 window

r2 window

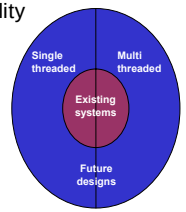
r1 window

r2 window

CRAFT:NONE is SWIFT (Software-only)

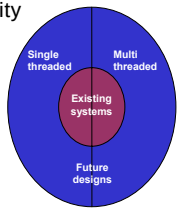
Spectrum of Solutions

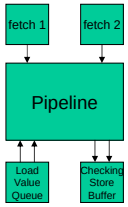
- Spectrum of solutions
  - Redundant Multithreading
  - CRAFT w/ Load Value Queue and Checking Store Buffer
  - CRAFT w/ Load Value Queue
  - CRAFT w/ Checking Store Buffer
  - SWIFT
- Applicability



Spectrum of Solutions

- Spectrum of solutions
  - Redundant Multithreading
  - CRAFT w/ Load Value Queue and Checking Store Buffer
  - CRAFT w/ Load Value Queue
  - CRAFT w/ Checking Store Buffer
  - SWIFT
- Applicability

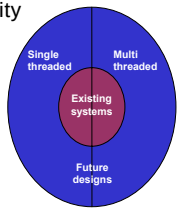


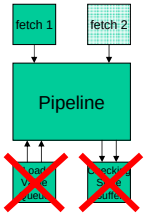


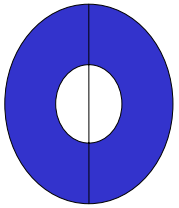
138

Spectrum of Solutions

- Spectrum of solutions
  - Redundant Multithreading
  - CRAFT w/ Load Value Queue and Checking Store Buffer
  - CRAFT w/ Load Value Queue
  - CRAFT w/ Checking Store Buffer
  - SWIFT
- Applicability



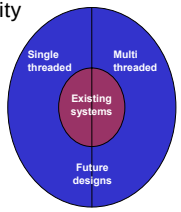


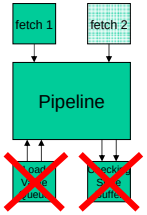


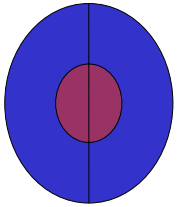
139

Spectrum of Solutions

- Spectrum of solutions
  - Redundant Multithreading
  - CRAFT w/ Load Value Queue and Checking Store Buffer
  - CRAFT w/ Load Value Queue
  - CRAFT w/ Checking Store Buffer
  - SWIFT
- Applicability

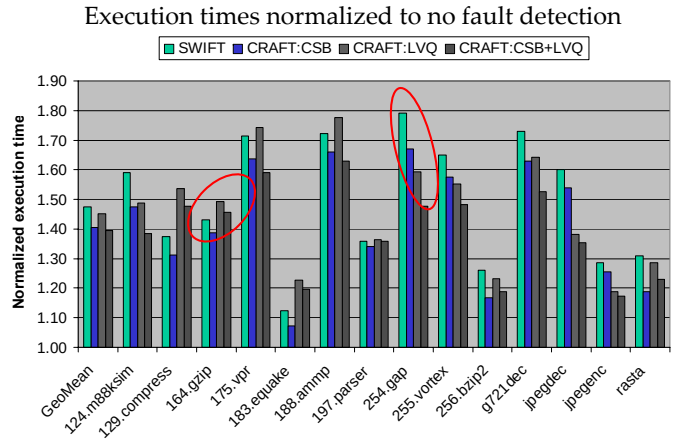






140

Hybrid Performance



141

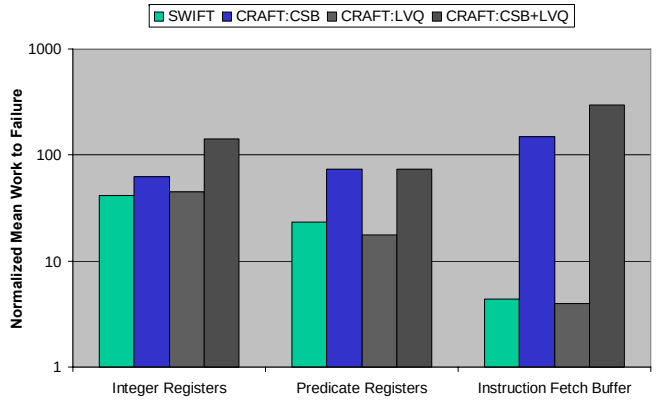
Hybrid Reliability

- Reliability evaluated using fault injection (3 structures)
  - Single bit flip per execution
  - 5000 injection executions per structure per benchmark per system
  - Use combination of microarchitectural and architectural simulation
- Mean Work To Failure
  - Encompass longer execution time and increased reliability
  - Generalization of Mean Instructions To Failure [Weaver et al. ISCA '04]
    - Instruction not constant unit of work in hybrid systems
  - Proportional to:
    1 / (Architectural Vulnerability * Execution time$_{\text{unit of work}}$)

142

Hybrid Reliability

Mean Work to Failure normalized to no fault detection



143

Hybrid Fault Tolerance Summary

"Design and Evaluation of Hybrid Fault-Detection Systems"
[Reis ISCA-32 05]

```
ld  r1  = [r2 ]          ld  r1  = [r2 ]
ld  r1' = [r2']          ld  r1' = [r2']
add r1  = r1 +1          add r1  = r1 +1
add r1' = r1'+1          add r1' = r1'+1
br faultDet, r1 != r1'   br faultDet, r1 != r1'
br faultDet, r2 != r2'   br faultDet, r2 != r2'
st [r2 ] = r1            st [r2 ] = r1
                         st [r2'] = r1'
```

Better performance than SWIFT:
3% speedup

Better reliability than SWIFT:
75% reduction in abnormal execution
25% reduction in incorrect output

144

Acknowledgements

- Vijay Narayanan
- Shubu Mukherjee
- George Reis
- Mark Bohr

145