QED & Symbolic QED

Subhasish Mitra

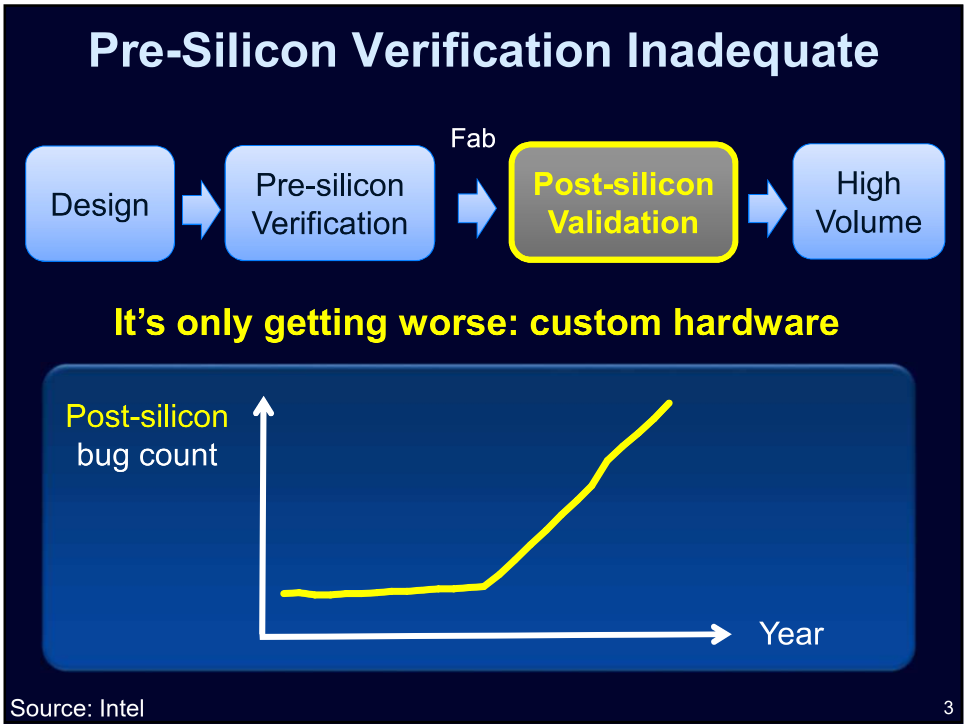
Department of EE & Department of CS
Stanford University

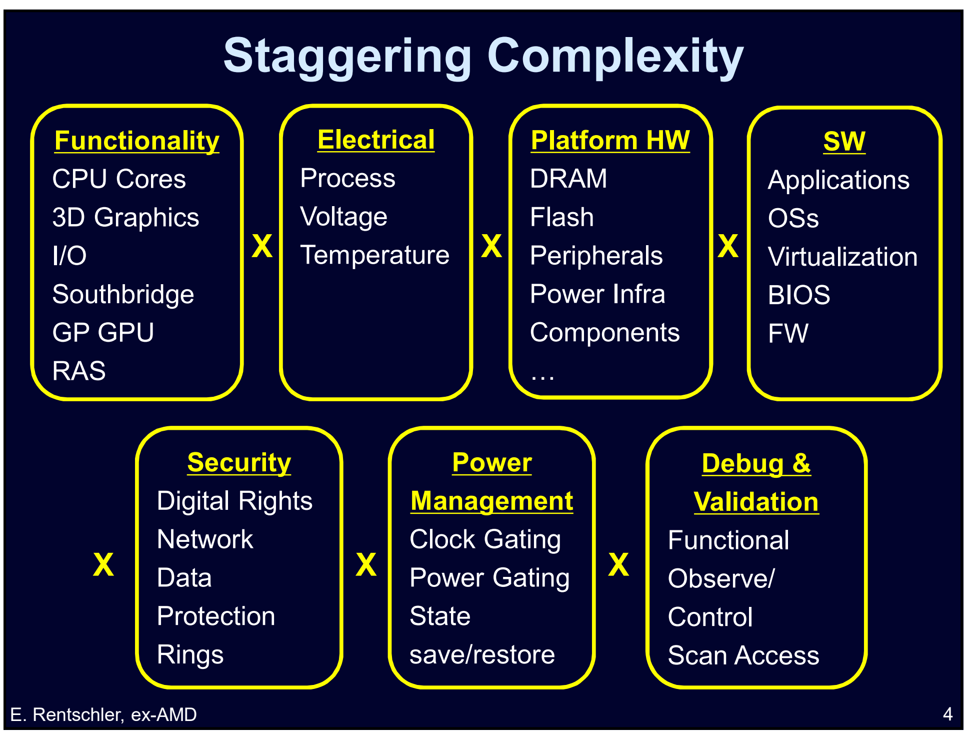
Acknowledgment: Students & Collaborators

Students, Sponsors, Collaborators

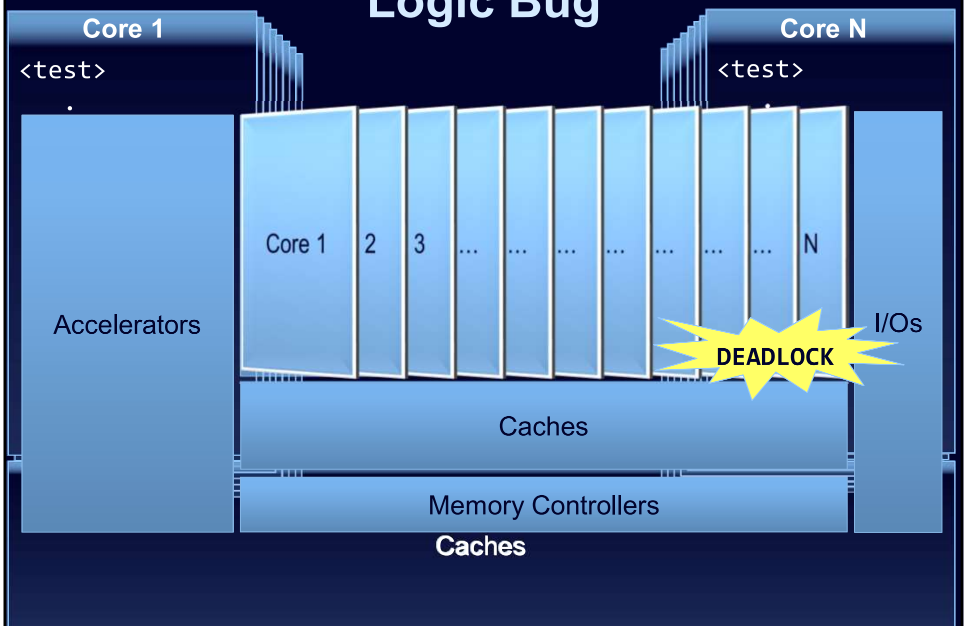
AMD BOEING DARPA freescale NXP infineon intel NSF SRC

2


Pre-Silicon Verification Inadequate
Design
Pre-silicon Verification
Fab
Post-silicon Validation
High Volume
It's only getting worse: custom hardware
Post-silicon bug count
Year
Source: Intel
3


Staggering Complexity
Functionality
CPU Cores
3D Graphics
I/O
Southbridge
GP GPU
RAS
X
Electrical
Process
Voltage
Temperature
X
Platform HW
DRAM
Flash
Peripherals
Power Infra
Components
…
X
SW
Applications
OSs
Virtualization
BIOS
FW
X
Security
Digital Rights
Network
Data
Protection
Rings
X
Power Management
Clock Gating
Power Gating
State
save/restore
X
Debug & Validation
Functional
Observe/
Control
Scan Access
E. Rentschler, ex-AMD
4

Logic Bug

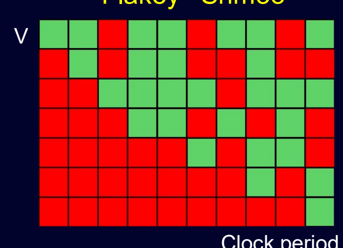


5

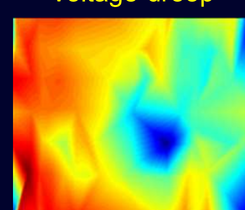
Electrical Bugs

- Speed-path, min-delay, noise, analog
  - Some P, V, T, F corners
  - “Heisenbug” characteristics

“Flakey” Shmoo



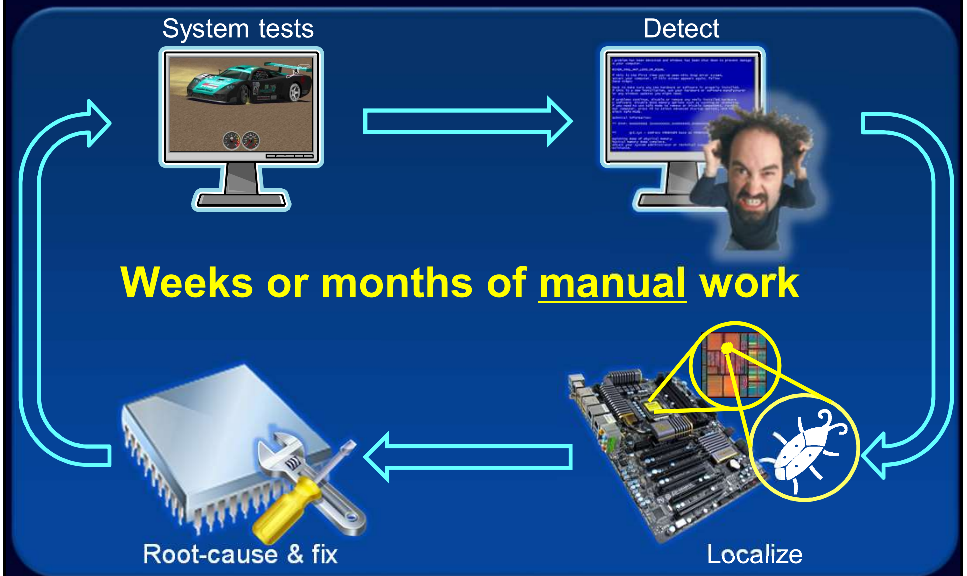
Voltage droop



N. Hakim, Intel J. Stinson, ex-Intel

6

Post-Silicon Validation Difficult



7

Scalability Barriers

- System failure reproduction
  - Asynchronous I/Os, clock domains
- Full-system simulation for golden response
  - 10^6 -10^9X slower than silicon

J. Stinson (ex-Intel)

Post-silicon costs rising faster than design cost

8

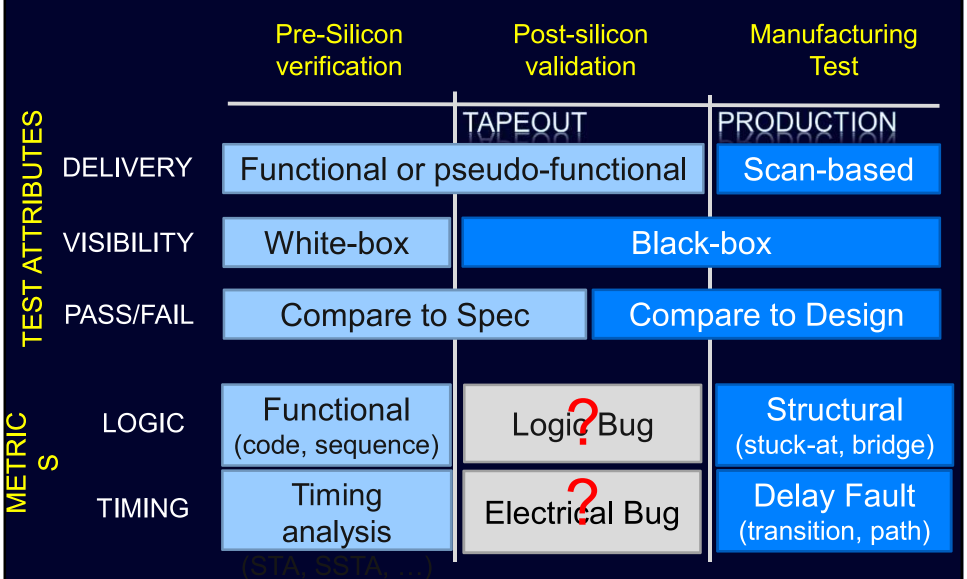
Pre- vs. Post-Silicon

Design → Pre-silicon Verification → Fab → **Post-silicon Validation** → High Volume

	Pre-Silicon	Post-Silicon
Debug & fix	Easier	Expensive
Internal signal access	Great	Limited
Electrical bugs	Model difficult	Already present
Platform interactions	Little (not real time)	Present
Speed	Slow simulation	Fast silicon

9

Pre- vs. Post-Silicon vs. Test



Prof. K.T. Cheng

10

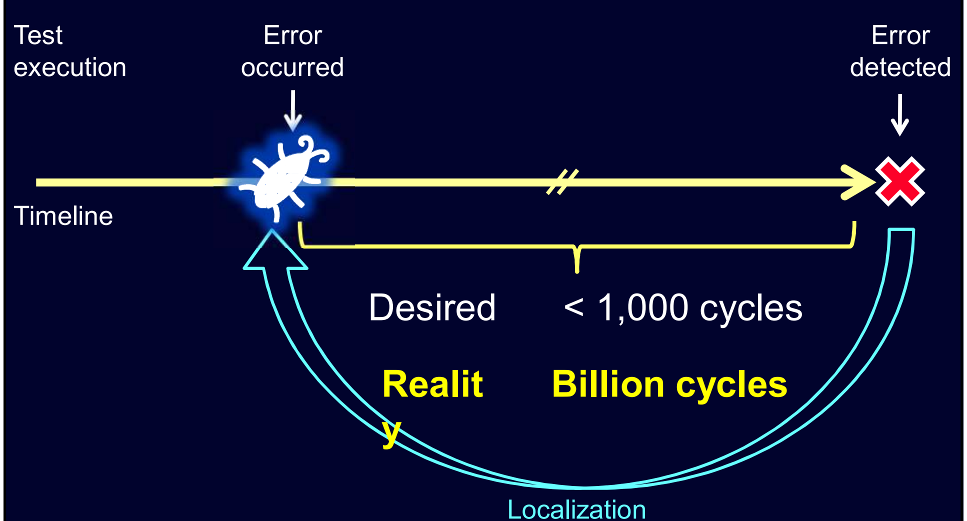
QED

- Post-silicon
  - Electrical bugs, logic bugs, system-level test

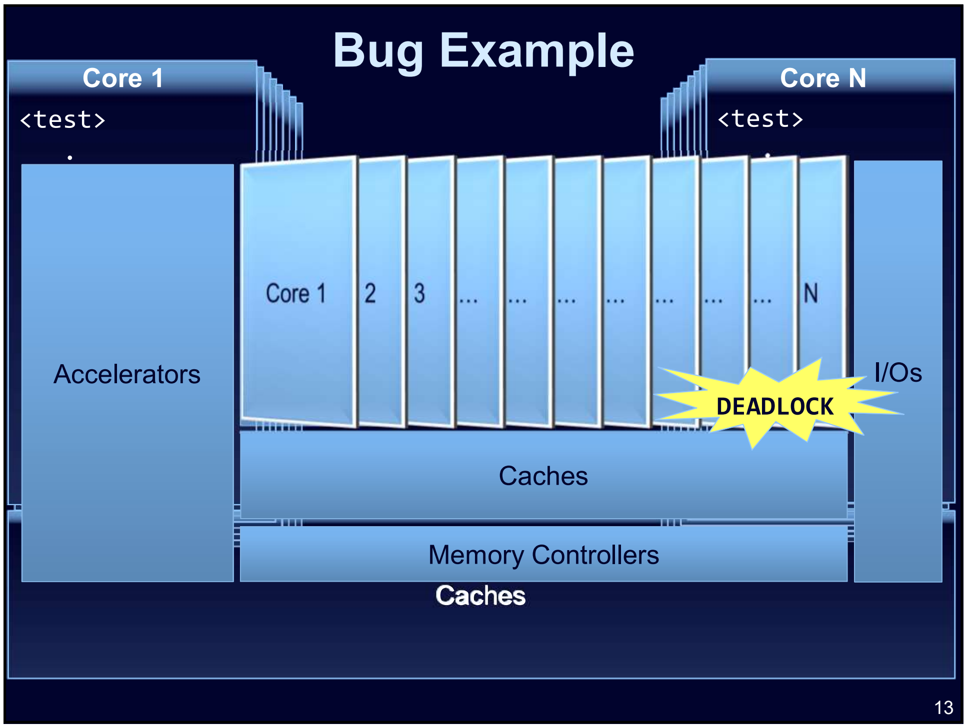
[Lin IEEE TCAD 14, Singh CAV 17]

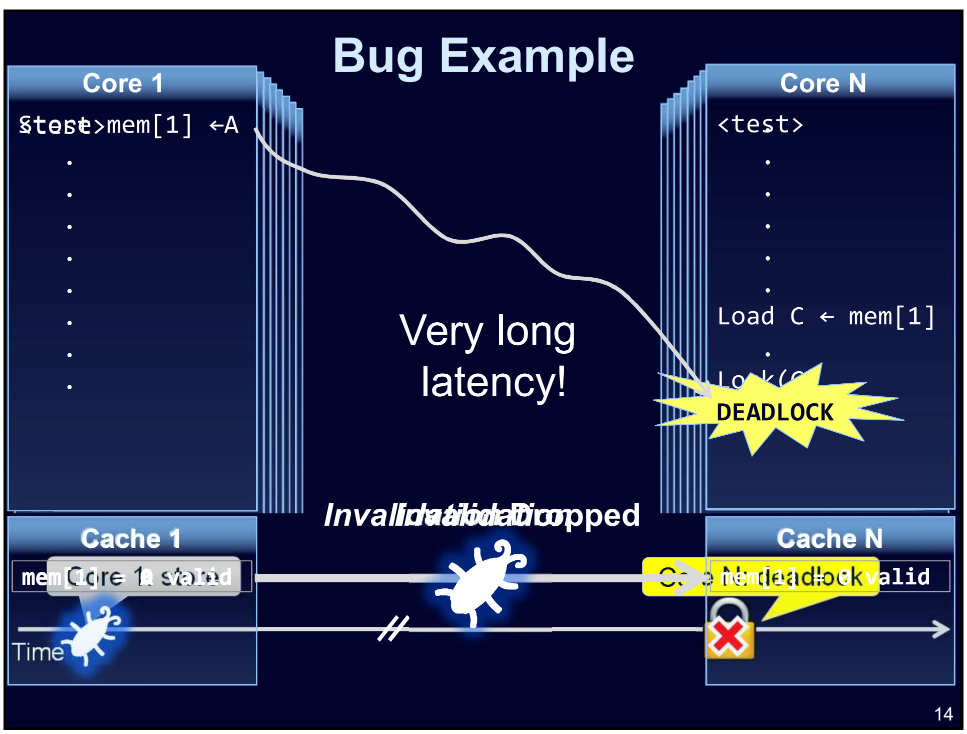
11

Error Detection Latency



12


Bug Example
Core 1
<test>
Core N
<test>
Accelerators
Core 1
2
3
N
DEADLOCK
I/Os
Caches
Memory Controllers
Caches
13


Bug Example
Core 1
Store mem[1] ←A
Core N
<test>
Load C ← mem[1]
DEADLOCK
Very long latency!
Cache 1
Cache N
Time
14

Existing Techniques Inadequate

- Very long error detection latencies

Deadlock detection

Design-specific assertions

Self-checking tests

Store readback tests

Checkpointing

etc…

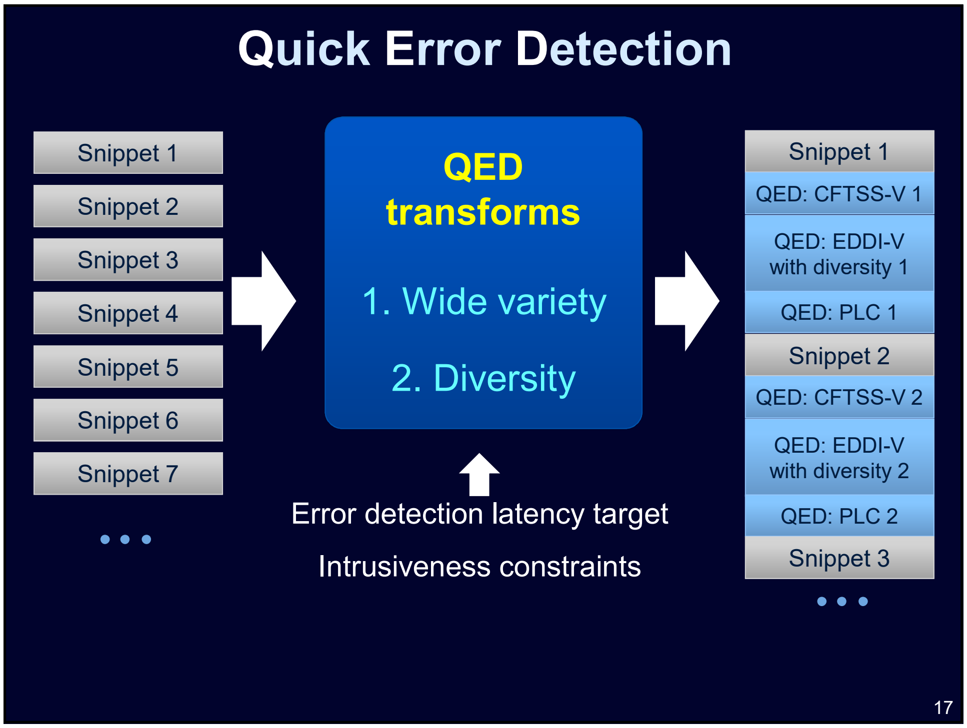
15

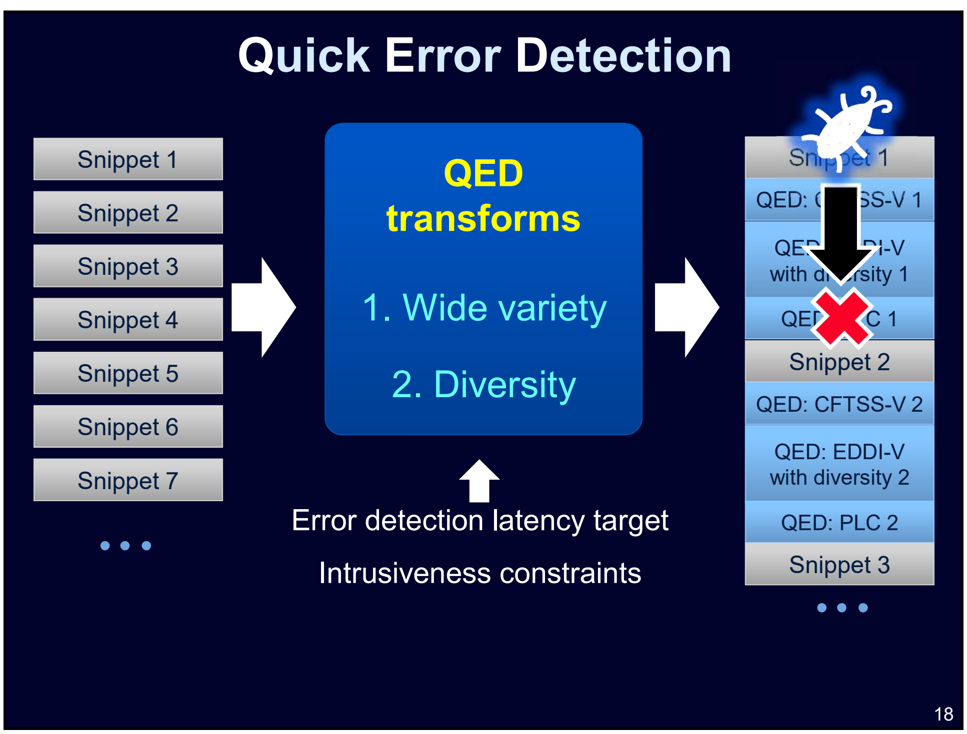
Quick Error Detection

- Error detection latency: guaranteed short
- Coverage: improved
- Software-only: readily applicable

[Lin IEEE TCAD 14]

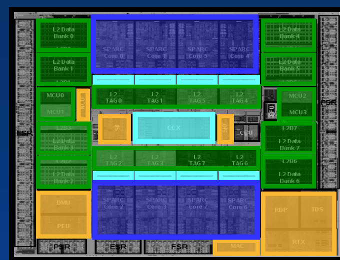
16


Quick Error Detection
Snippet 1
Snippet 2
Snippet 3
Snippet 4
Snippet 5
Snippet 6
Snippet 7
QED
transforms
1. Wide variety
2. Diversity
Error detection latency target
Intrusiveness constraints
Snippet 1
QED: CFTSS-V 1
QED: EDDI-V
with diversity 1
QED: PLC 1
Snippet 2
QED: CFTSS-V 2
QED: EDDI-V
with diversity 2
QED: PLC 2
Snippet 3
17


Quick Error Detection
Snippet 1
Snippet 2
Snippet 3
Snippet 4
Snippet 5
Snippet 6
Snippet 7
QED
transforms
1. Wide variety
2. Diversity
Error detection latency target
Intrusiveness constraints
Snippet 2
QED: CFTSS-V 2
QED: EDDI-V
with diversity 2
QED: PLC 2
Snippet 3
18

QED Transforms

System on Chip

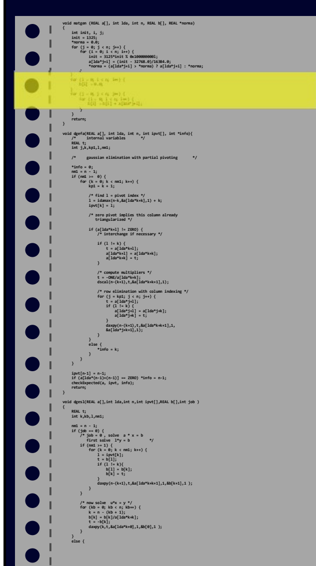


Processor cores	Uncore, accelerators
EDDI-V	PLC
CFTSS-V	Fast QED
CFCSS-V	Hybrid QED

19

QED Example: Duplicate & Check

Validation program



Trace

• • •

```
R1  ← R1  + 5
R2  ← R2  – R1
R3  ← R1  * R2
```

Very Long!

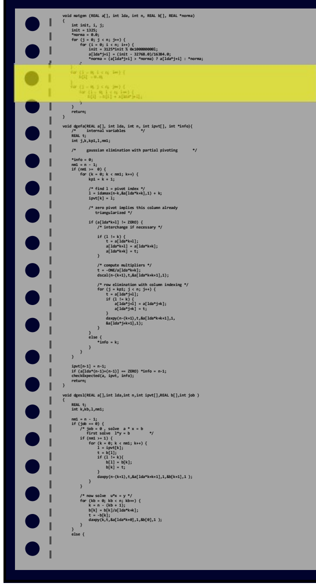
• • •

Check end_result

20

QED Example: Duplicate & Check

Validation program



QED Trace

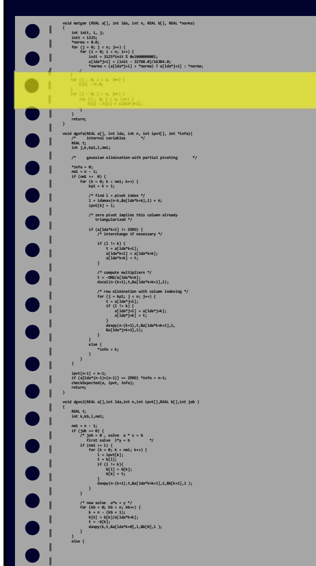
```
R16 ← R1; R18 ← R2; R19 ← R3
     • • •
R1  ← R1  + 5
R16 ← R16 + 5
R1  == R16
R2  ← R2  – R1
R18 ← R18 – R16
R2  == R18
R3  ← R1  * R2
R19 ← R16 * R18
R3  == R19
     • • •
Check end_result
```

Quick!

21

QED Improves Coverage

Validation program



Trace

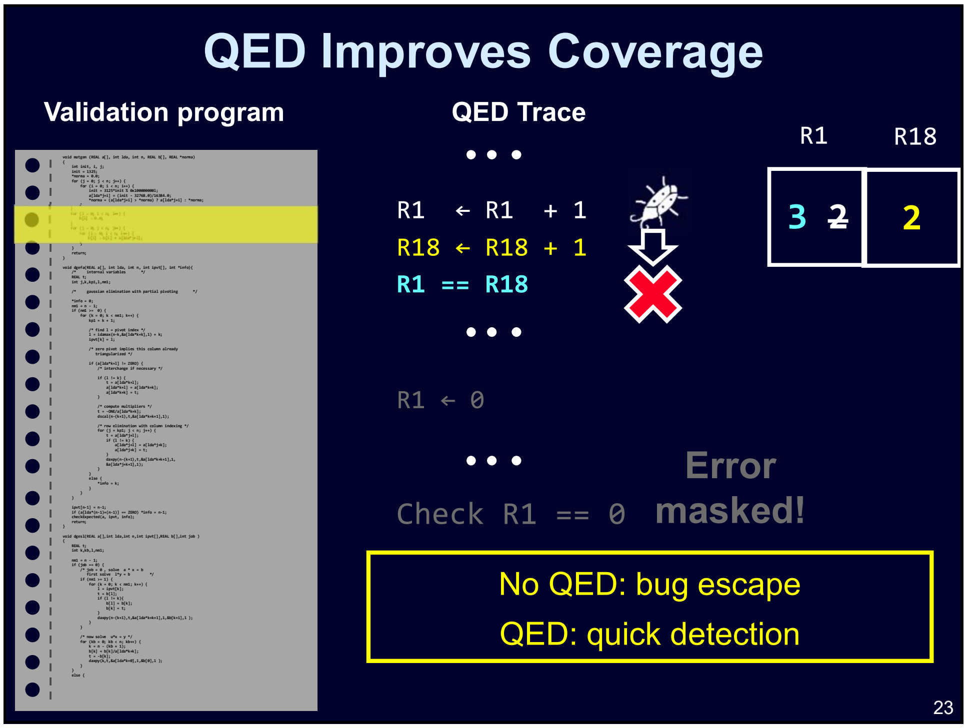
```
     • • •
R1  ← R1  + 1
     • • •
R1 ← 0
     • • •
Check R1 == 0
```

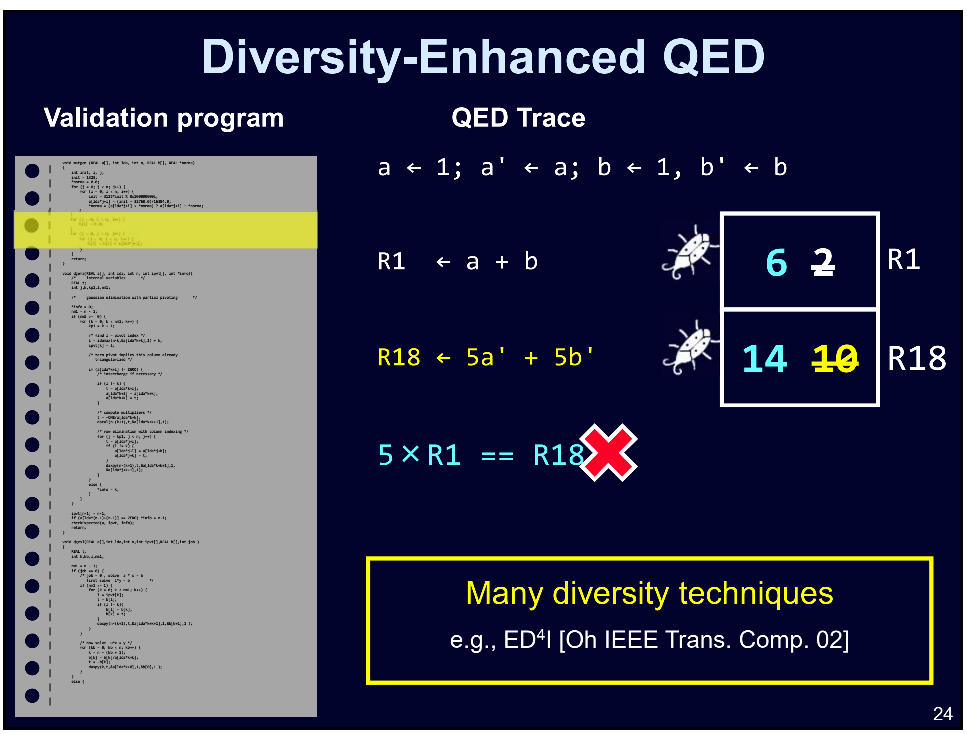
R1

3 ~~2~~

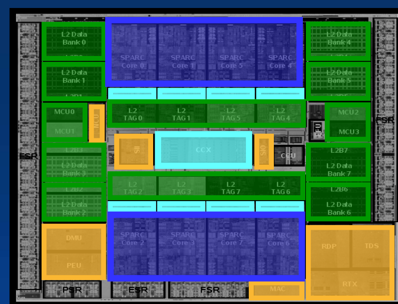
Error masked!

22


QED Improves Coverage
Validation program
QED Trace
R1
R18
• • •
R1 ← R1 + 1
R18 ← R18 + 1
R1 == R18
3 ~~2~~
2
• • •
R1 ← 0
• • •
Check R1 == 0
Error masked!
No QED: bug escape
QED: quick detection
23


Diversity-Enhanced QED
Validation program
QED Trace
a ← 1; a' ← a; b ← 1, b' ← b
R1 ← a + b
6 ~~2~~
R1
R18 ← 5a' + 5b'
14 ~~10~~
R18
5 × R1 == R18
Many diversity techniques
e.g., ED4I [Oh IEEE Trans. Comp. 02]
24

QED Transforms



Processor cores

EDDI-V

CFTSS-V

CFCSS-V

Uncore, accelerators

PLC

Fast QED

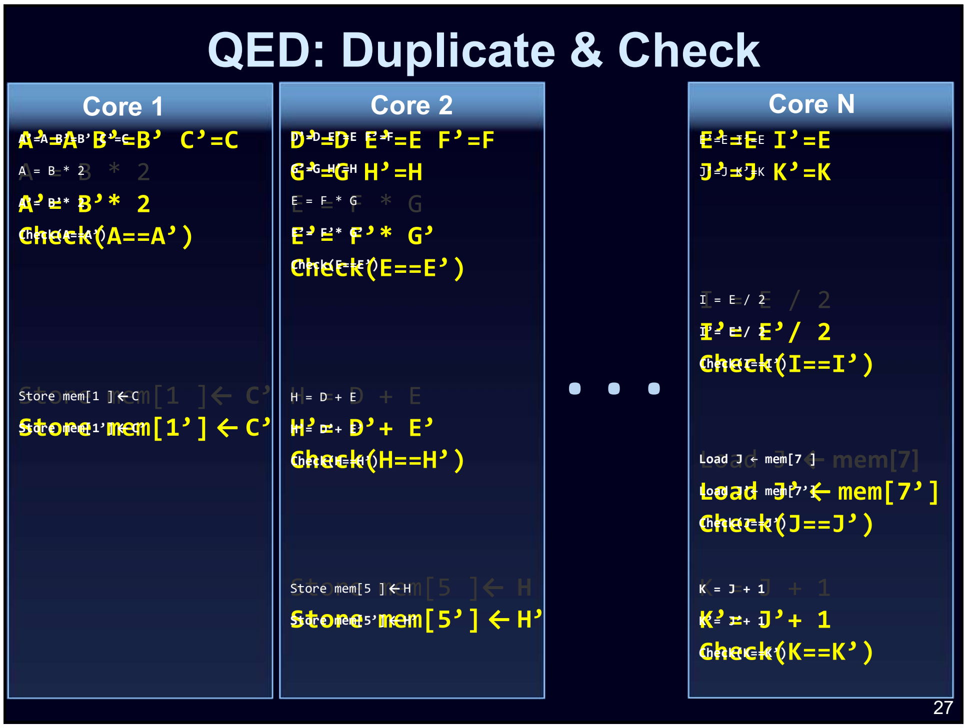
Hybrid QED

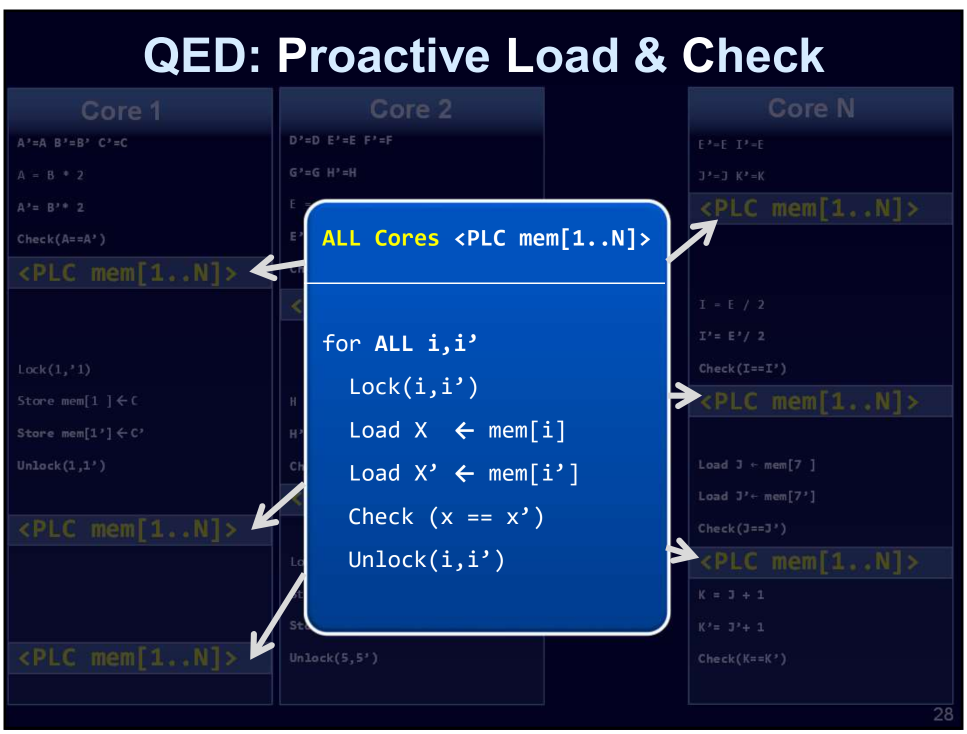
25

Original Test

Core 1	Core 2		Core N
A = B * 2			
	E = F * G		
			I = E / 2
Store mem[1] ← C	H = D + E	• • •	
			Load J ← mem[7]
	Store mem[5] ← H		K = J + 1

26


QED: Duplicate & Check
Core 1
A'=A B'=B' C'=C
A = B * 2
A'= B'* 2
Check(A==A')
Store mem[1]←C
Store mem[1']←C'
Core 2
D'=D E'=E F'=F
G'=G H'=H
E = F * G
E'= F'* G'
Check(E==E')
H = D + E
H'= D'+ E'
Check(H==H')
Store mem[5]←H
Store mem[5']←H'
Core N
E'=E I'=E
J'=J K'=K
I = E / 2
I'= E'/ 2
Check(I==I')
Load J ← mem[7]
Load J'← mem[7']
Check(J==J')
K = J + 1
K'= J'+ 1
Check(K==K')
27

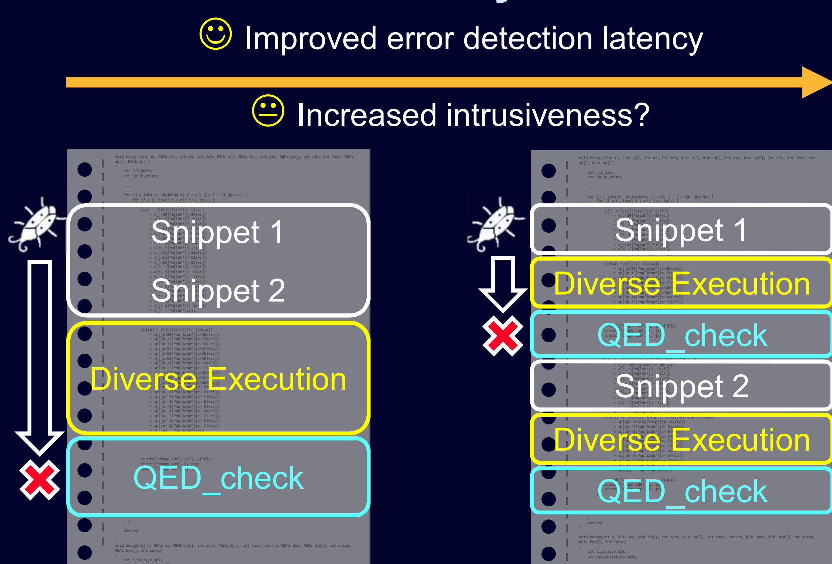

QED: Proactive Load & Check
Core 1
Core 2
Core N
A'=A B'=B' C'=C
A = B * 2
A'= B'* 2
Check(A==A')
<PLC mem[1..N]>
Lock(1,'1)
Store mem[1]←C
Store mem[1']←C'
Unlock(1,1')
<PLC mem[1..N]>
<PLC mem[1..N]>
D'=D E'=E F'=F
G'=G H'=H
Unlock(5,5')
E'=E I'=E
J'=J K'=K
<PLC mem[1..N]>
I = E / 2
I'= E'/ 2
Check(I==I')
<PLC mem[1..N]>
Load J ← mem[7]
Load J'← mem[7']
Check(J==J')
<PLC mem[1..N]>
K = J + 1
K'= J'+ 1
Check(K==K')
ALL Cores <PLC mem[1..N]>
for ALL i,i'
Lock(i,i')
Load X ← mem[i]
Load X' ← mem[i']
Check (x == x')
Unlock(i,i')
28

QED Coverage Considerations

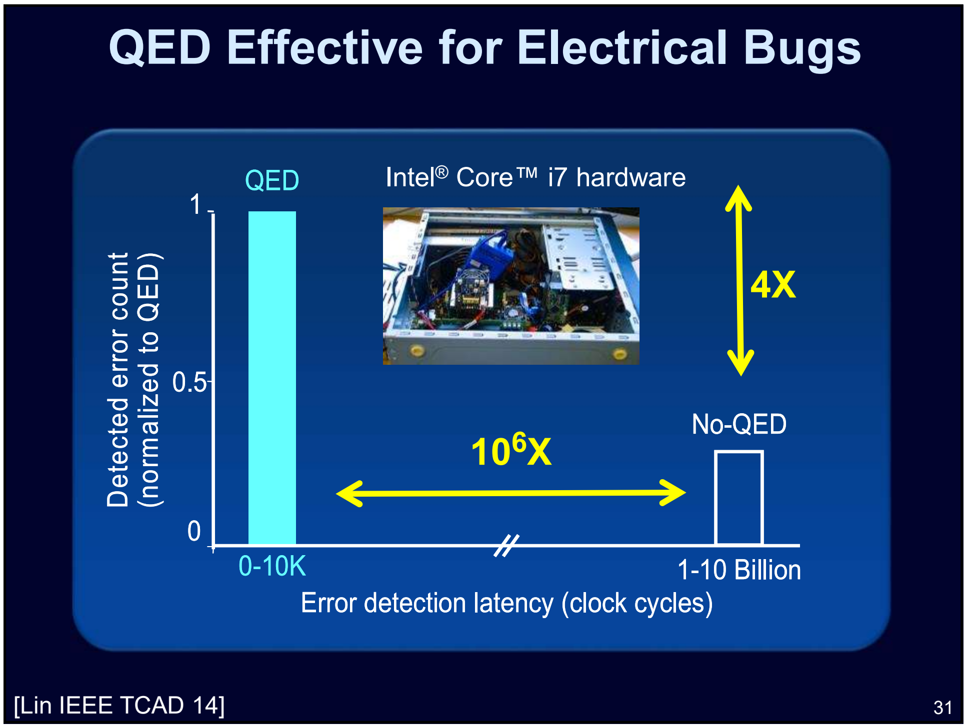
- Challenge
  - Intrusiveness: coverage impact ?
- Systematic solutions
  - QED family tests
  - Hardware-enhanced QED

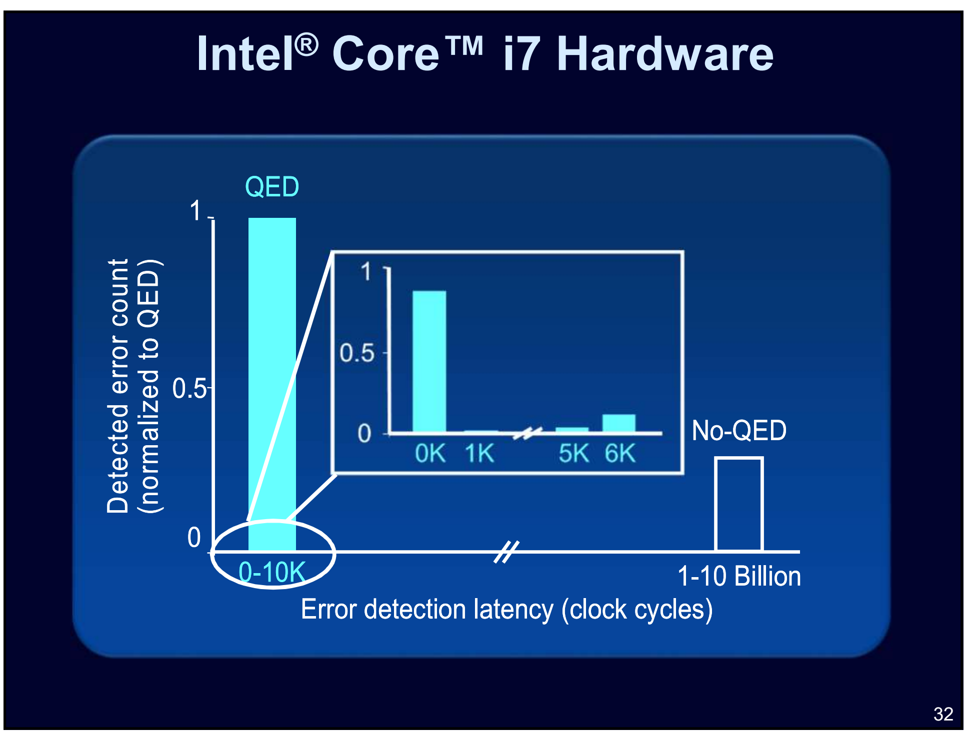
29

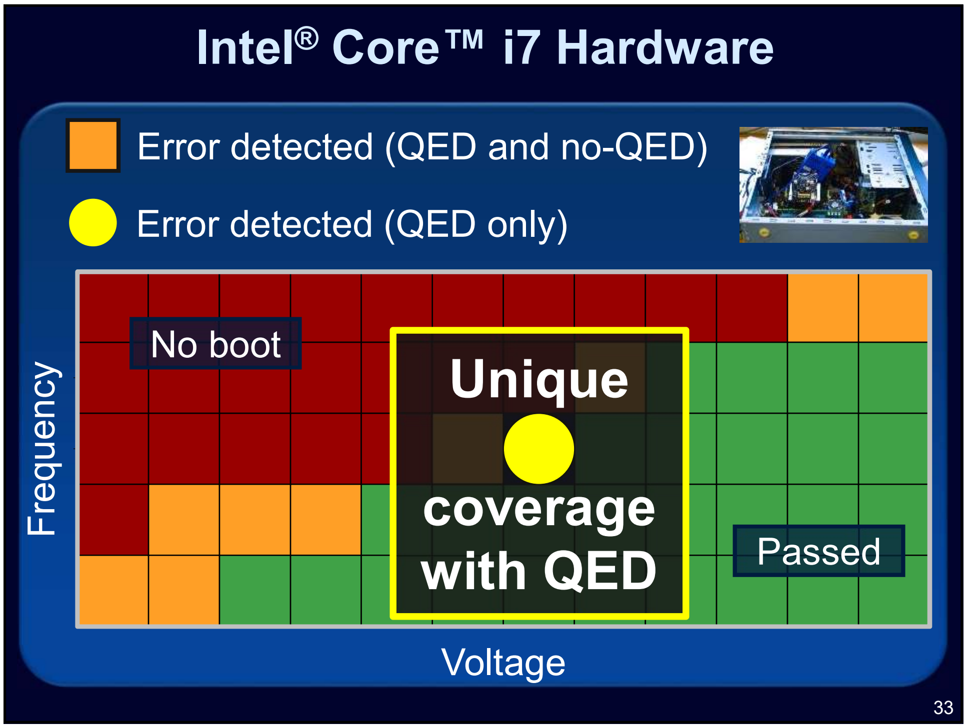
Error Detection Latency vs. Intrusiveness

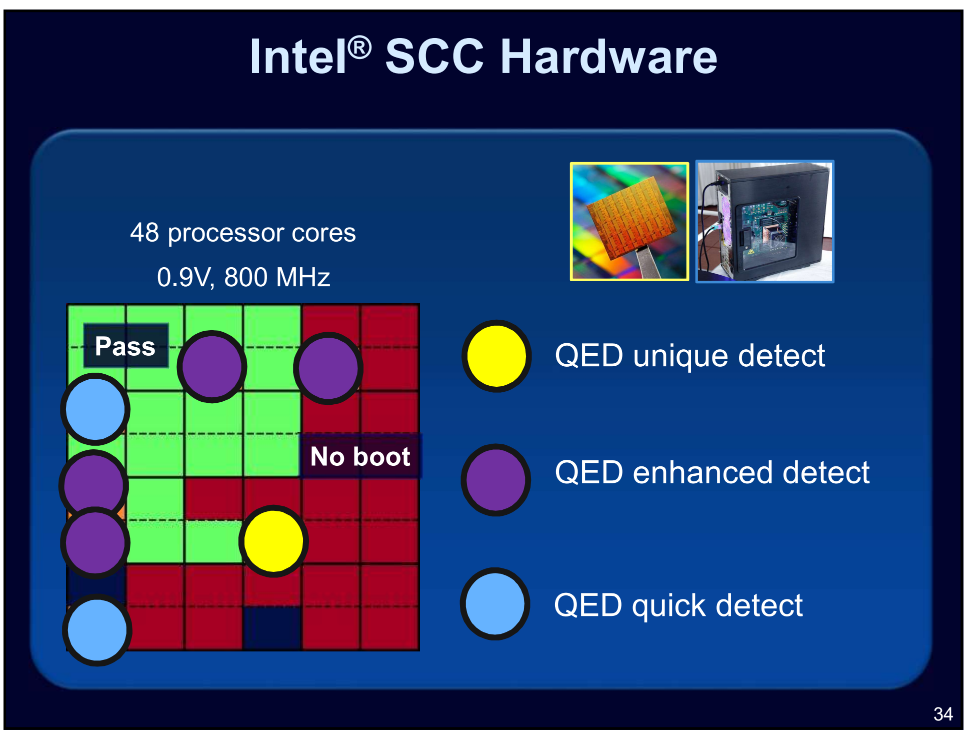


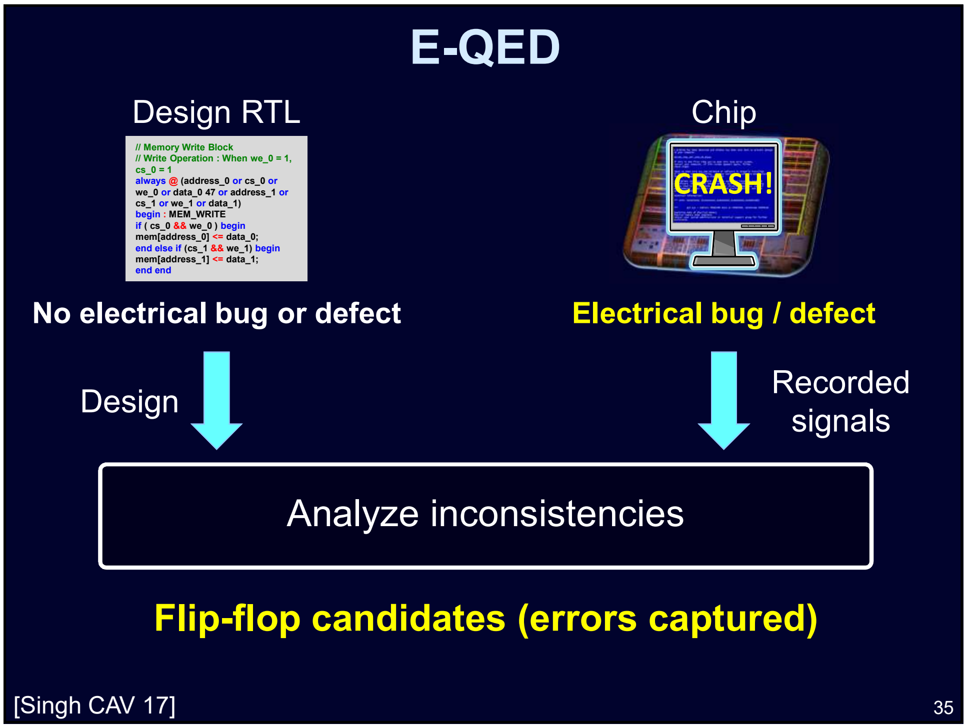
30


QED Effective for Electrical Bugs
QED
Intel® Core™ i7 hardware
4X
Detected error count
(normalized to QED)
1
0.5
0
10^6X
No-QED
0-10K
1-10 Billion
Error detection latency (clock cycles)
[Lin IEEE TCAD 14]
31

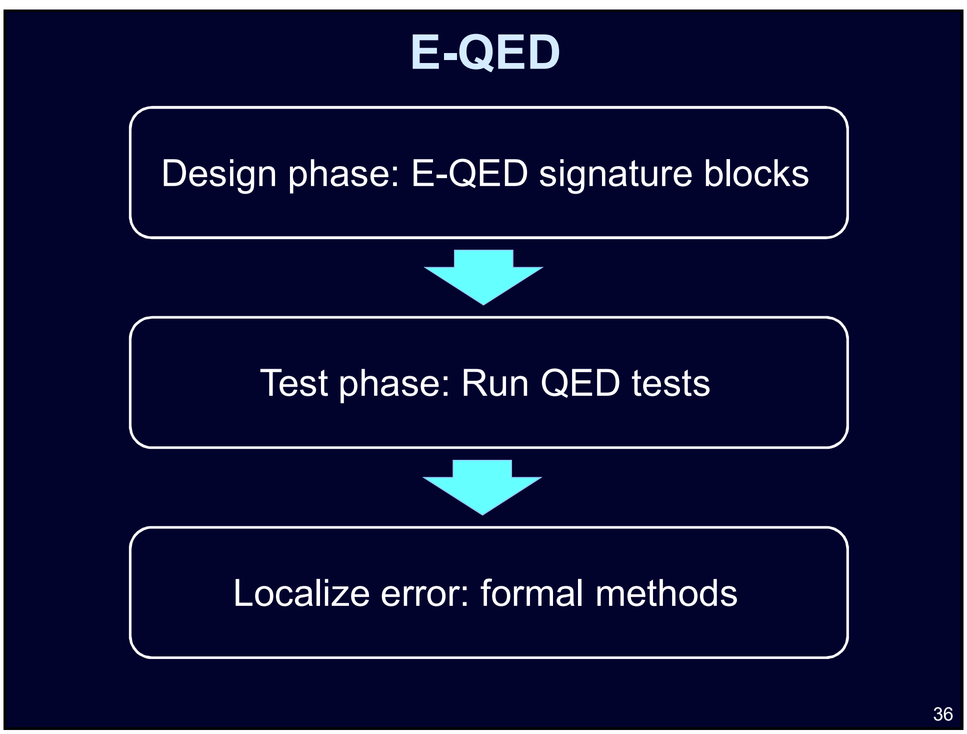

Intel® Core™ i7 Hardware
QED
Detected error count
(normalized to QED)
1
0.5
0
0K 1K 5K 6K
No-QED
0-10K
1-10 Billion
Error detection latency (clock cycles)
32


Intel® Core™ i7 Hardware
Error detected (QED and no-QED)
Error detected (QED only)
No boot
Unique
coverage
with QED
Passed
Frequency
Voltage
33


Intel® SCC Hardware
48 processor cores
0.9V, 800 MHz
Pass
No boot
QED unique detect
QED enhanced detect
QED quick detect
34



E-QED
Design RTL
// Memory Write Block
// Write Operation : When we_0 = 1,
cs_0 = 1
always @ (address_0 or cs_0 or
we_0 or data_0 47 or address_1 or
cs_1 or we_1 or data_1)
begin : MEM_WRITE
if (cs_0 && we_0) begin
mem[address_0] <= data_0;
end else if (cs_1 && we_1) begin
mem[address_1] <= data_1;
end end
Chip
CRASH!
No electrical bug or defect
Electrical bug / defect
Design
Recorded signals
Analyze inconsistencies
Flip-flop candidates (errors captured)
[Singh CAV 17]
35



E-QED
Design phase: E-QED signature blocks
Test phase: Run QED tests
Localize error: formal methods
36

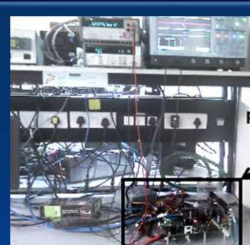
E-QED Results

Flip-flop candidates	18 (out of 1 million) **50,000× localization**
Area impact	2.5% (actually 0%)
Effort	Automatic
Runtime	~ 9 hours
Failure reproduction	**<u>None</u>**
Full system simulation	**<u>None</u>**

OpenSPARC T2 SoC (500M transistors)

QED Effective for Logic Bugs

From Months to Seconds



Freescale multi-core SoC hardware

Error detection latency (cycles)

No QED	QED
15 Billion	**9**

[Lin IEEE TCAD 14]

QED Effective for Logic Bugs

- "Difficult" logic bugs
  - Industrial bug databases
- Processor cores, accelerators, uncore, power management

[Lin IEEE TCAD 14]

39

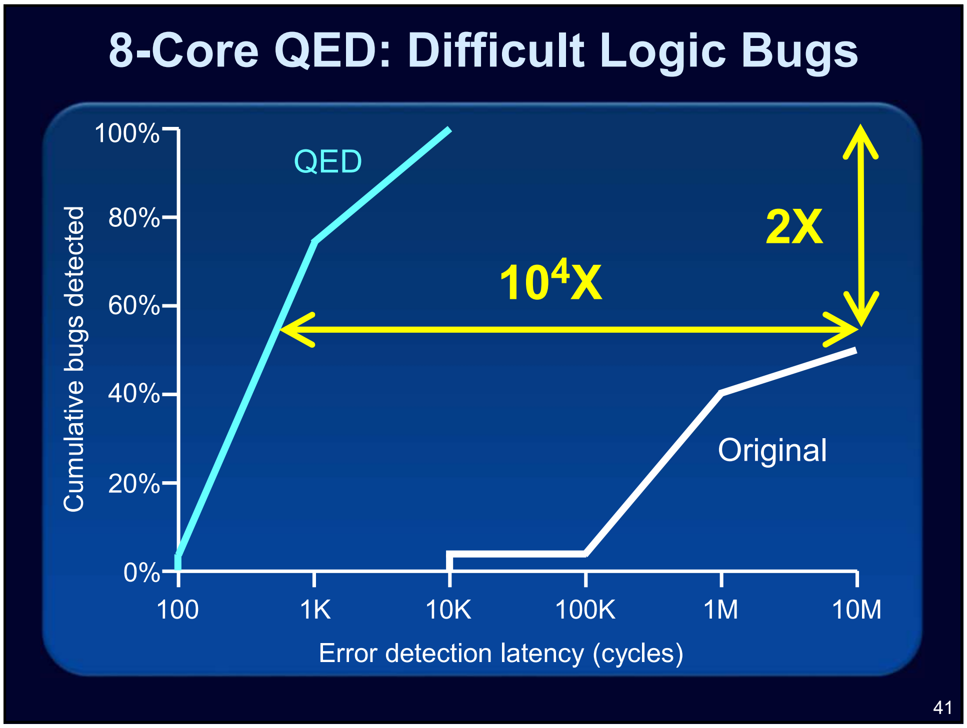
Activation Criteria × Effects

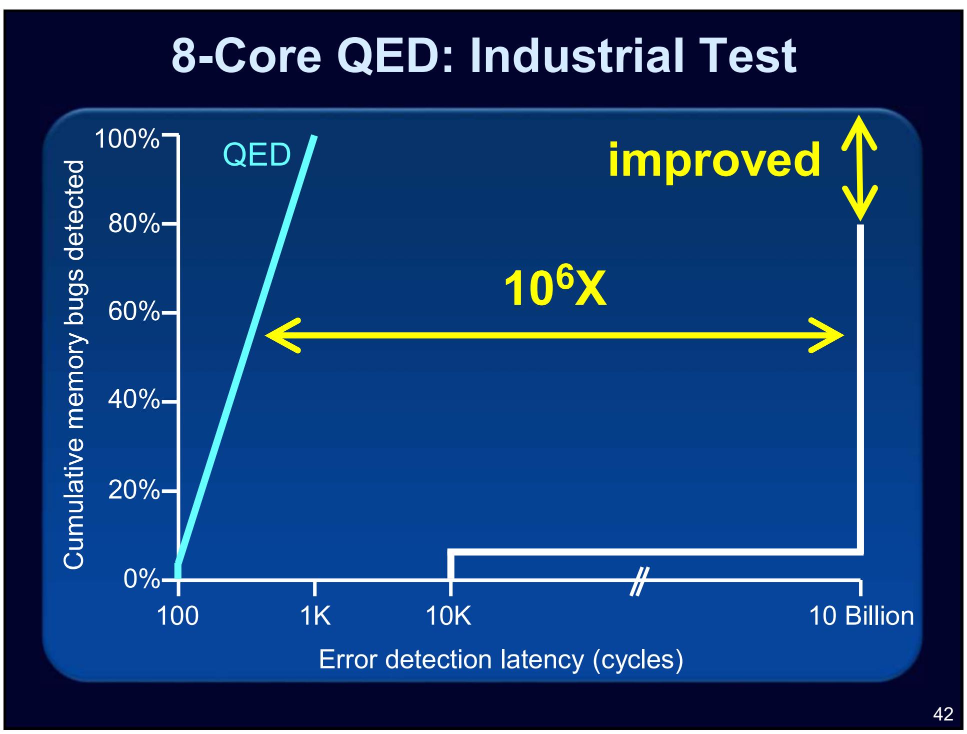
Activation Criteria
1. Two stores in X cycles
2. Two stores in X cycles to same cache line
3. Two stores in X cycles to adjacent cache line
4. Specific sequence of loads and stores
5. Two branches in X cycles
6. Forwarding data X to inst Y
7. Two icache misses in X cycles
8. Random

X

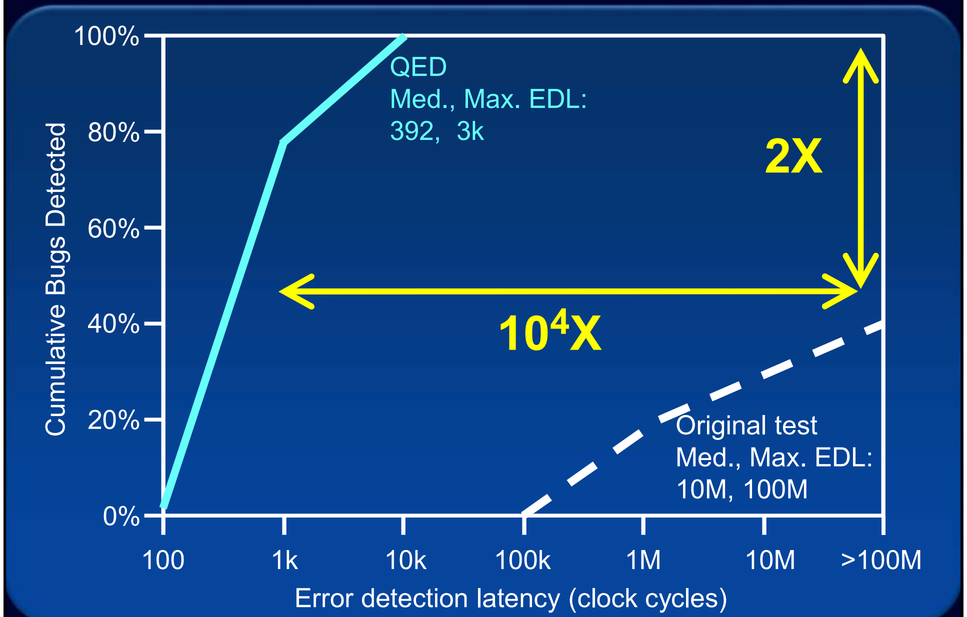
Bug effects
A. Cache coherence msg. dropped
B. Cache coherence msg. delayed
C. Store not allocated a cache line
D. Update delayed by Z cycles
E. Loaded value corrupted
F. Data in main memory corrupted
G. Data in L1$ corrupted
H. Jump to wrong address
I. Error in instruction operand
J. Wrong instruction decoded

40


8-Core QED: Difficult Logic Bugs
100%
80%
60%
40%
20%
0%
Cumulative bugs detected
QED
10^4X
2X
Original
100
1K
10K
100K
1M
10M
Error detection latency (cycles)
41


8-Core QED: Industrial Test
100%
80%
60%
40%
20%
0%
Cumulative memory bugs detected
QED
improved
10^6X
100
1K
10K
10 Billion
Error detection latency (cycles)
42

8-Core QED: Power Management Bugs



EDL = Error Detection Latency

43

QED Automation

- Application test, random test, …
  - Source code, assembly, binary

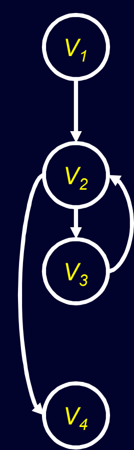
44

Program Representation: Control Flow Graph

```
i=0; x=1; y=5;
While (i < 10) {
        z = x + y * i;
        i = i + 1;
}
```

Basic Block (BB)

```
      mov  r1, 0        V1
      mov  r2, 1
      mov  r3, 5
L1:
      inc  r1           V2
      bge  r1, 10, L2
      mul  r4, r3, r1   V3
      add  r5, r2, r4
      br   L1
L2:
…                       V4
```



45

QED Automation Example

Original test

↓

Compile to LLVM IR

↓

QED transformations on LLVM IR ← QED parameters: Instr_min, Instr_max

↓

LLVM backend

↓

QED family tests

IR = Intermediate Representation

46

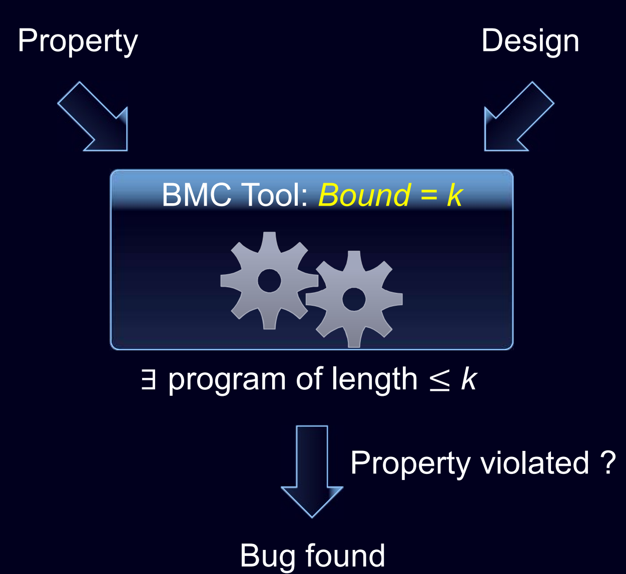
Symbolic QED

- <u>Pre-silicon</u>
  - Logic bugs: processors, accelerators, SoCs

[Lin ITC 15, Singh IEEE TCAD 18, Lonsing ICCAD 19]

47

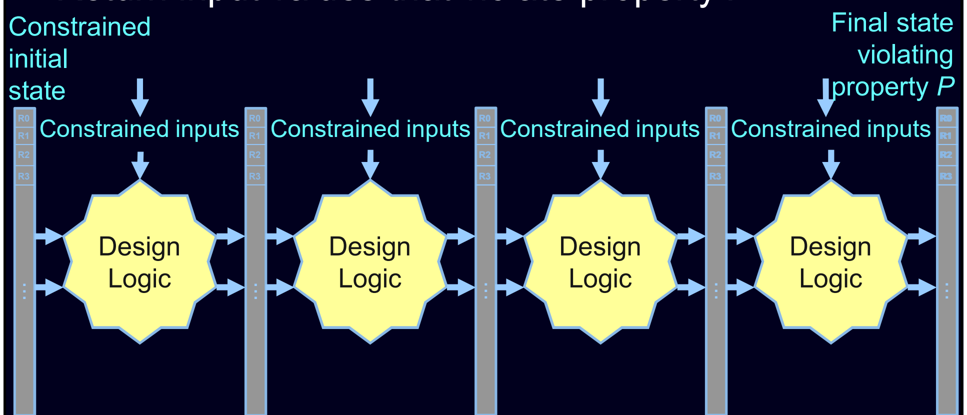
Traditional Bounded Model Checking



48

Bounded Model Checking (BMC)

- Input constraints, initial state constraints, property *P*
- Exhaustively explore design for *k* clock cycles
- Return input values that violate property *P*



Unrolled Circuit, *k* = 4

49

Traditional BMC vs. Symbolic QED

	Traditional BMC	Symbolic QED
Properties	Manual	**Automatic QED checks (Universal property)**
Design size	Small blocks	**Large SoCs**

50

Traditional BMC Challenges

1. What property ?

2. Design size ?

51

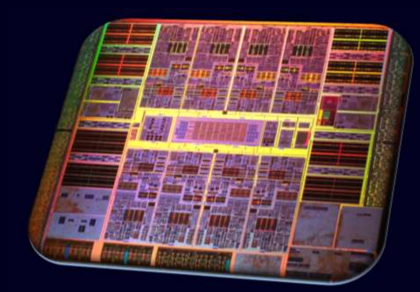
What Property ?

- Bugs not known *a priori*
- Manually written: time-consuming
- Automatic: challenging for difficult bugs
- Too many cycles to detect bugs

52

Design Size ?

- Big designs, cannot load



OpenSPARC T2 SoC
500 Million Transistors

53

BMC Symbolic QED

Design

↓

BMC Tool: *Bound = k*

∃ **QED program** of length ≤ k **that fails ?**

↓

Bug found

54

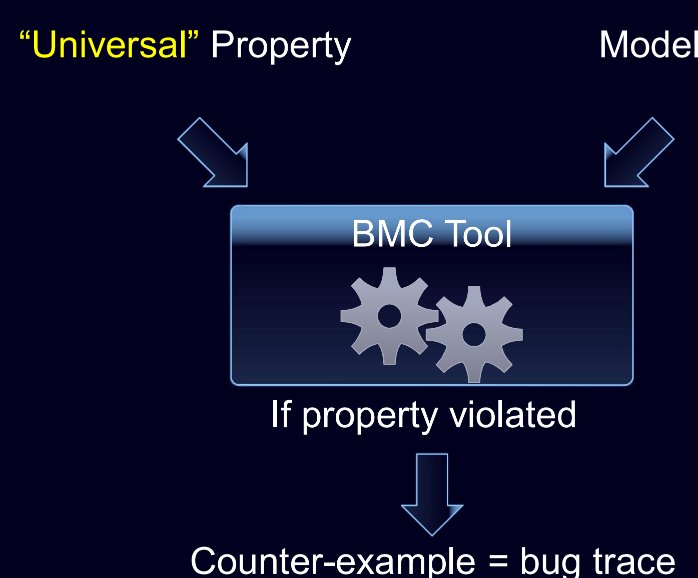
"Universal" Property: QED Check

CMP Ra == Ra'

- Ra – original register
- Ra' – corresponding duplicated register
- Ra ≠ Ra' – error detected

55

BMC Symbolic QED



56

Need Input Constraints

- If unconstrained, BMC may choose any inputs

Ra ← 1

Ra' ← 2

CMP Ra == Ra'

- False fail – not a bug

57

Input Constraints

- Original sequence then duplicated sequence
  - CMP Ra == Ra' after both executed
- Enforced by QED module automatically
  - Only during BMC
  - Not in fabricated design

58

Solution: QED Module

Only during BMC, not in fabricated chip

Input
(Chosen by BMC)

```
ST [0x10000], Ra
ST [0x10040], Rb
LD Rc, [0x10000]
```

QED Module

Output

```
ST [0x10000], Ra
ST [0x10040], Rb
LD Rc, [0x10000]
ST [0x20000], Ra'
ST [0x20040], Rb'
LD Rc', [0x20000]
<COMPARES>
```

59

Solution: QED Module

Only during BMC, not in fabricated chip

QED Module

Processor core fetch unit

- Automatically duplicates instructions
- Determines when QED checks happen

60

BMC Symbolic QED

"Universal" Property

Model + QED Module
(no hardware overhead)

BMC Tool

If property violated

Counter-example = bug trace

61

Starting State Important

- If unconstrained, BMC may choose any initial state

Ra = 1 Ra' = 2 // initial state

Ra = Ra + 1; Ra' = Ra' + 1; CMP Ra == Ra'

- QED-consistent starting state
  - Run "simple" QED test
- More sophisticated
  - [Fadiheh DATE 18, Devarajegowda DATE 20]

62

BMC Symbolic QED

"Universal" Property + QED-consistent starting state

Model + QED Module (no hardware overhead)

BMC Tool

If property violated

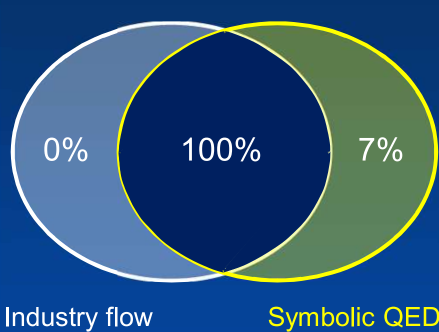
Counter-example = bug trace

63

Symbolic QED: Infineon Study

- Several automotive microcontroller cores

All (known) bugs + more



60× productivity

[Singh DATE 19]

64

Symbolic QED Study: A Major Company

- RISC V-based SoC (late-stage design)

1. New corner-case bugs found

Previously unknown

(despite countless simulations, emulation)

2. All "tricky" synthetic bugs correctly analyzed

No spurious counterexamples

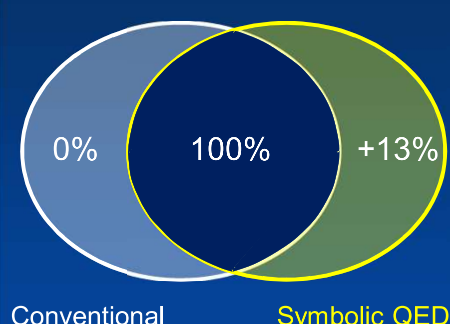
3. Symbolic QED setup: 3 person days only

65

Symbolic QED for Accelerators

- Stand-alone hardware accelerators for AI

All (known) bugs + more



30× productivity

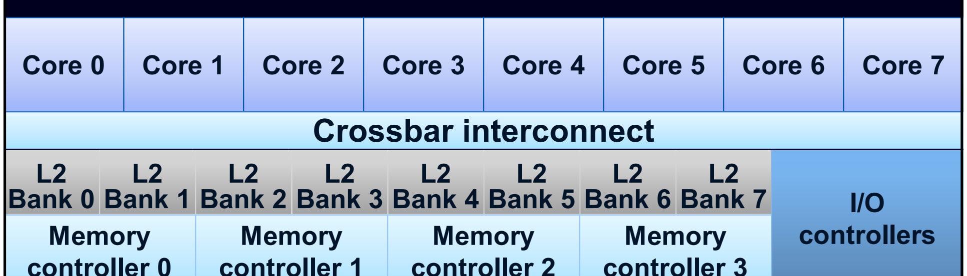
66

BUT…

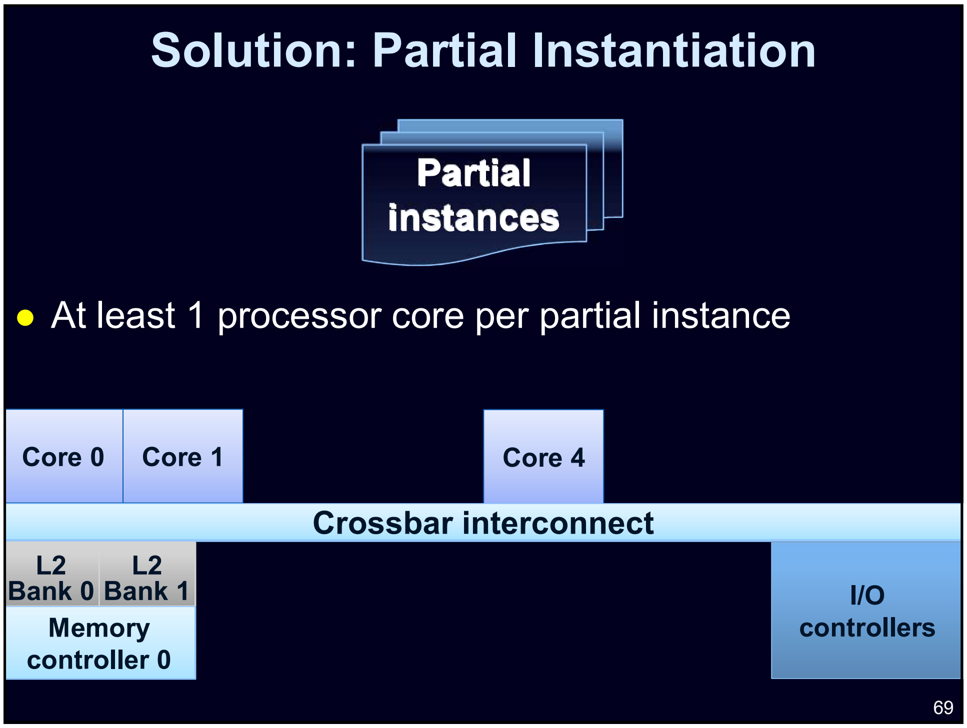
- Big designs ?
- **Solution: Partial instantiation**

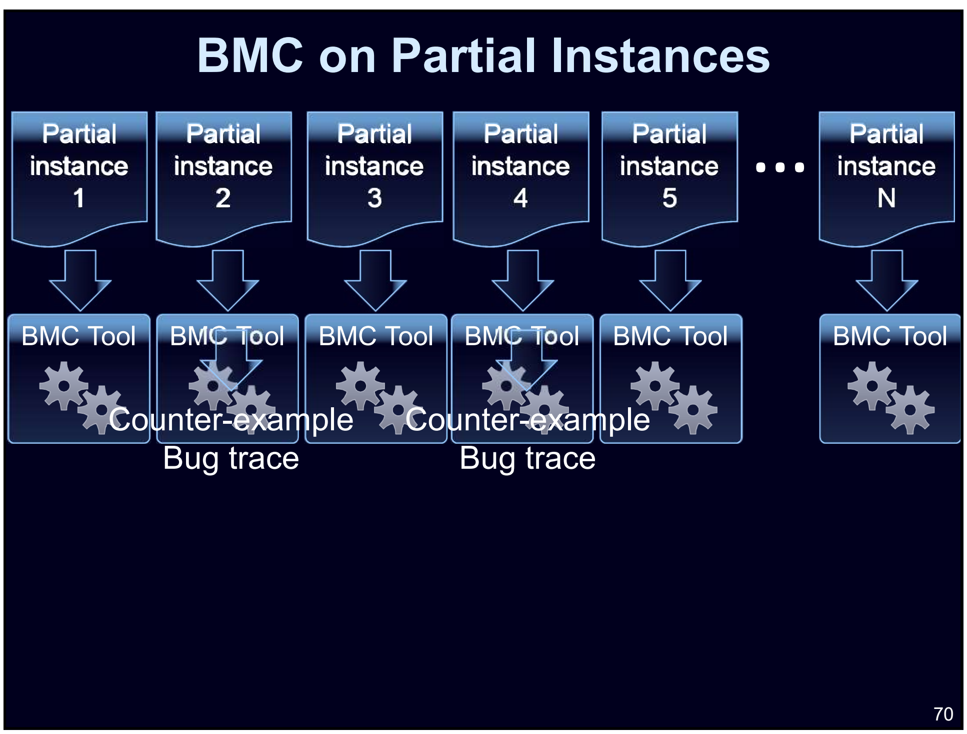
67

Solution: Partial Instantiation

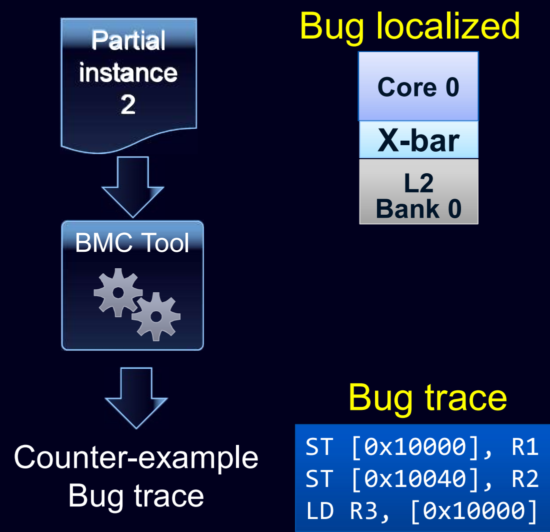


68


Solution: Partial Instantiation
Partial instances
At least 1 processor core per partial instance
Core 0
Core 1
Core 4
Crossbar interconnect
L2 Bank 0
L2 Bank 1
Memory controller 0
I/O controllers
69


BMC on Partial Instances
Partial instance 1
Partial instance 2
Partial instance 3
Partial instance 4
Partial instance 5
• • •
Partial instance N
BMC Tool
BMC Tool
BMC Tool
BMC Tool
BMC Tool
BMC Tool
Counter-example
Bug trace
Counter-example
Bug trace
70

BMC on Partial Instances



- Smallest partial instance → best localization

71

Traditional Properties <u>Not</u> Compositional

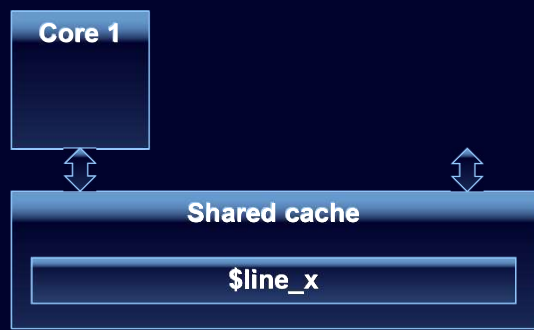
- Not preserved across all partial instances

Core 1 writes $line_x ##1 Core 2 reads $line_x →
$line_x in shared state

72

Traditional Properties <u>Not</u> Compositional

- Not preserved across all partial instances



Core 1 writes $line_x ##1 ~~Core 2 reads $line_x~~ →
$line_x in shared state

- Does not work if Core 2 missing

73

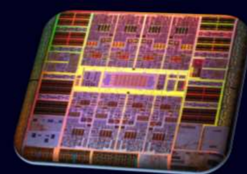
QED Checks Compositional

- Preserved across partial instances
- Not design- / implementation-specific

74

Case Study: OpenSPARC T2 SoC

- 92 difficult logic bugs
  - Commercial multi-core SoCs
  - Core, uncore, power management bugs



OpenSPARC T2 SoC
500 Million Transistors

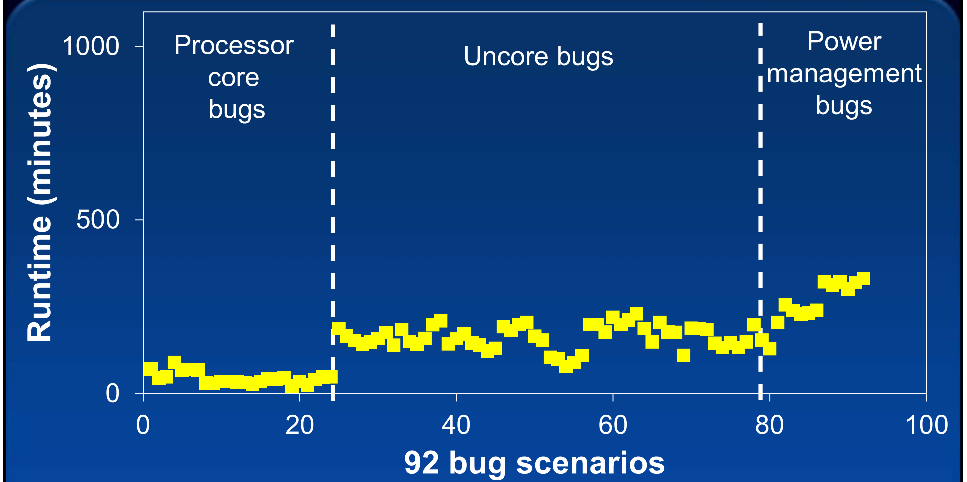
75

100X Reduction in Time

Traditional	**Automatic** Symbolic QED
Weeks to months (manual)	20 mins. to 7 hours

76

Symbolic QED Runtime



Symbolic QED with QED-consistent starting state obtained from running FFT QED test

77

10^6X Reduction In Bug Trace Length

Traditional	Automatic Symbolic QED
Millions of cycles	Less than 30 cycles

- Short bug trace
  - Easier to understand
  - Easier to fix bugs

78

More Opportunities

- Hardware security
  - Derive new attacks: beyond Spectre, Meltdown
  - Trojans
- Firmware
- Large-scale systems

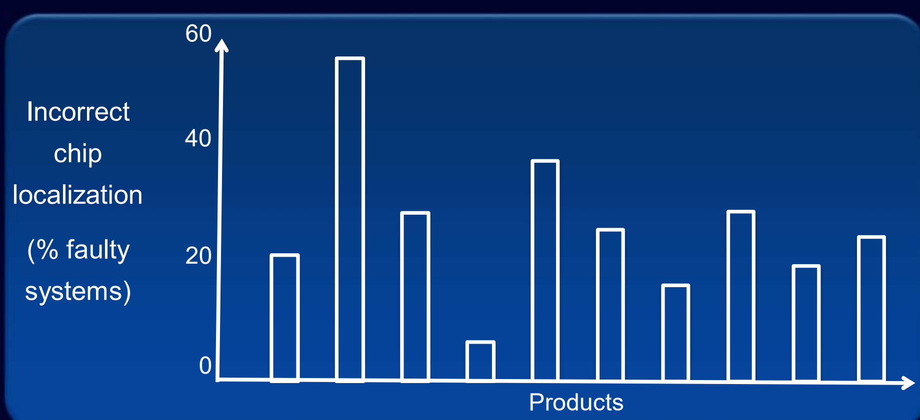
[Fadiheh DATE 19]

79

Big Obstacles at Full System Level

1. Faulty chip localization

2. System-Level Testing (No Trouble Found)



Z. Conroy, Cisco

80

QED & Symbolic QED

- Pre-silicon, post-silicon
  - Automatic, overnight, billion-transistor SoCs
- Widely applicable
  - Core, uncore, accelerator, bugs, defects
- More opportunities
  - Security, full systems, System-level Testing

81