Phoenix: Detecting and Recovering from Permanent Processor Design Bugs with Programmable Hardware

Smruti R. Sarangi

Abhishek Tiwari

Josep Torrellas

University of Illinois at Urbana-Champaign

http://iacoma.cs.uiuc.edu

ILLINOIS
1867
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

i-acoma group

Can a Processor have a Design Defect ?

No Way !!!

Yes, it is a major challenge.

A Major Challenge ???

50-70% effort spent on debugging

1-2 year verification times

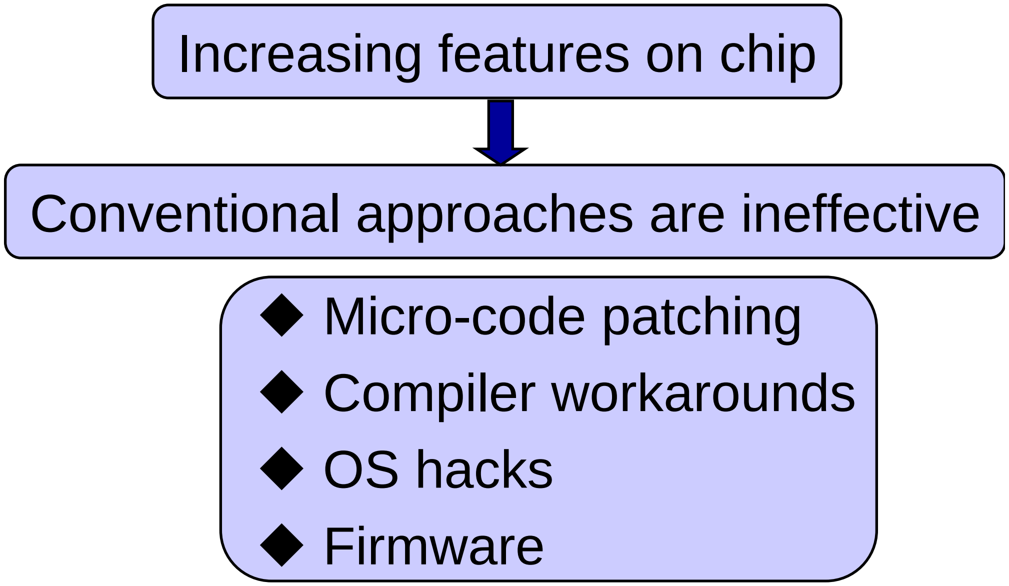
Massive computational resources

Some defects still slip through to production silicon

Defects slip through ???

1994	Pentium defect costs Intel $475 million
1999	Defect leads to stoppage in shipping Pentium III servers
2004	AMD Opteron defect leads to data loss
2005	A version of Itanium 2 recalled

Does not look like it will stop

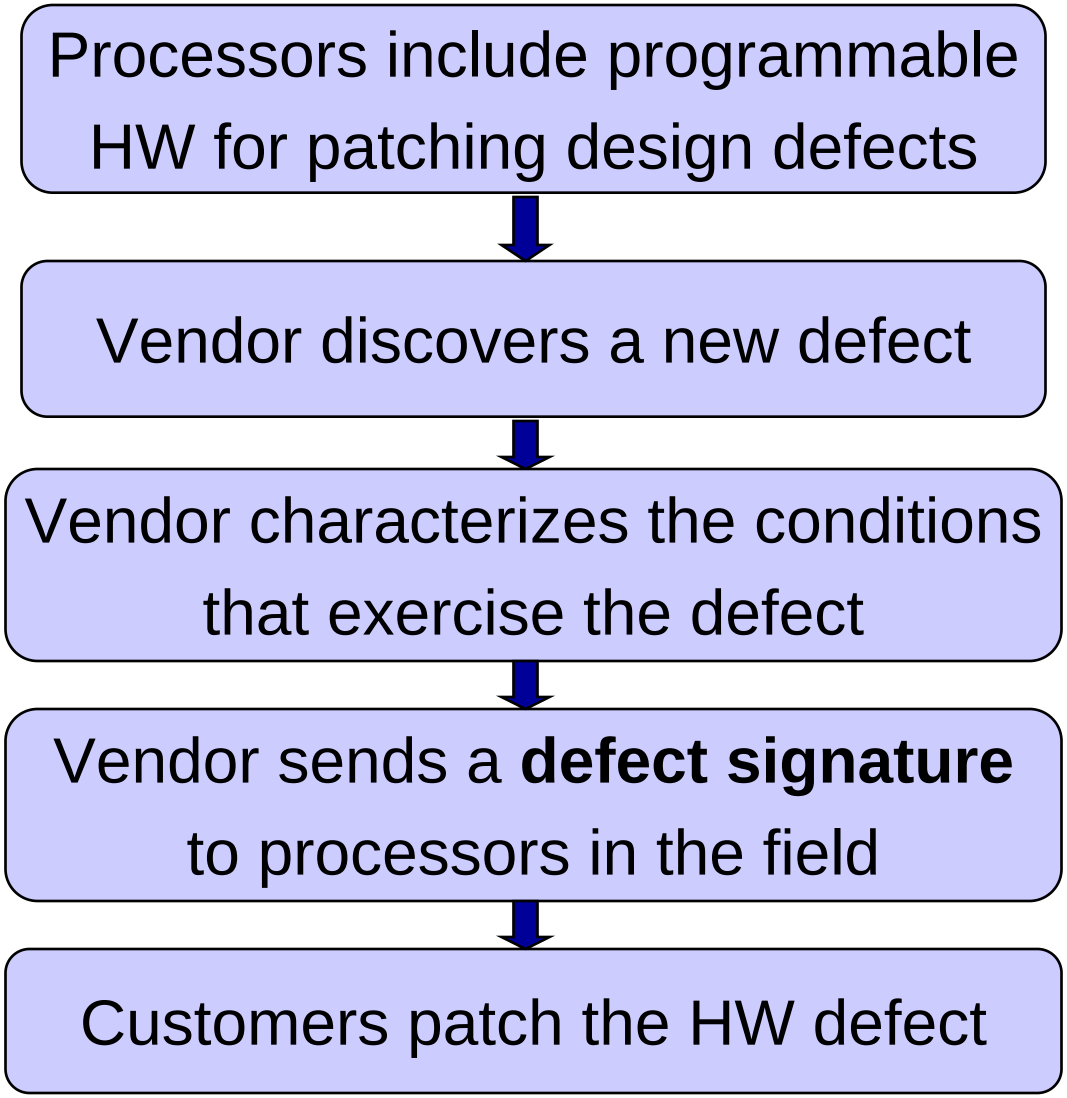


Increasing features on chip

Conventional approaches are ineffective

- Micro-code patching
- Compiler workarounds
- OS hacks
- Firmware

Vision



Processors include programmable HW for patching design defects

Vendor discovers a new defect

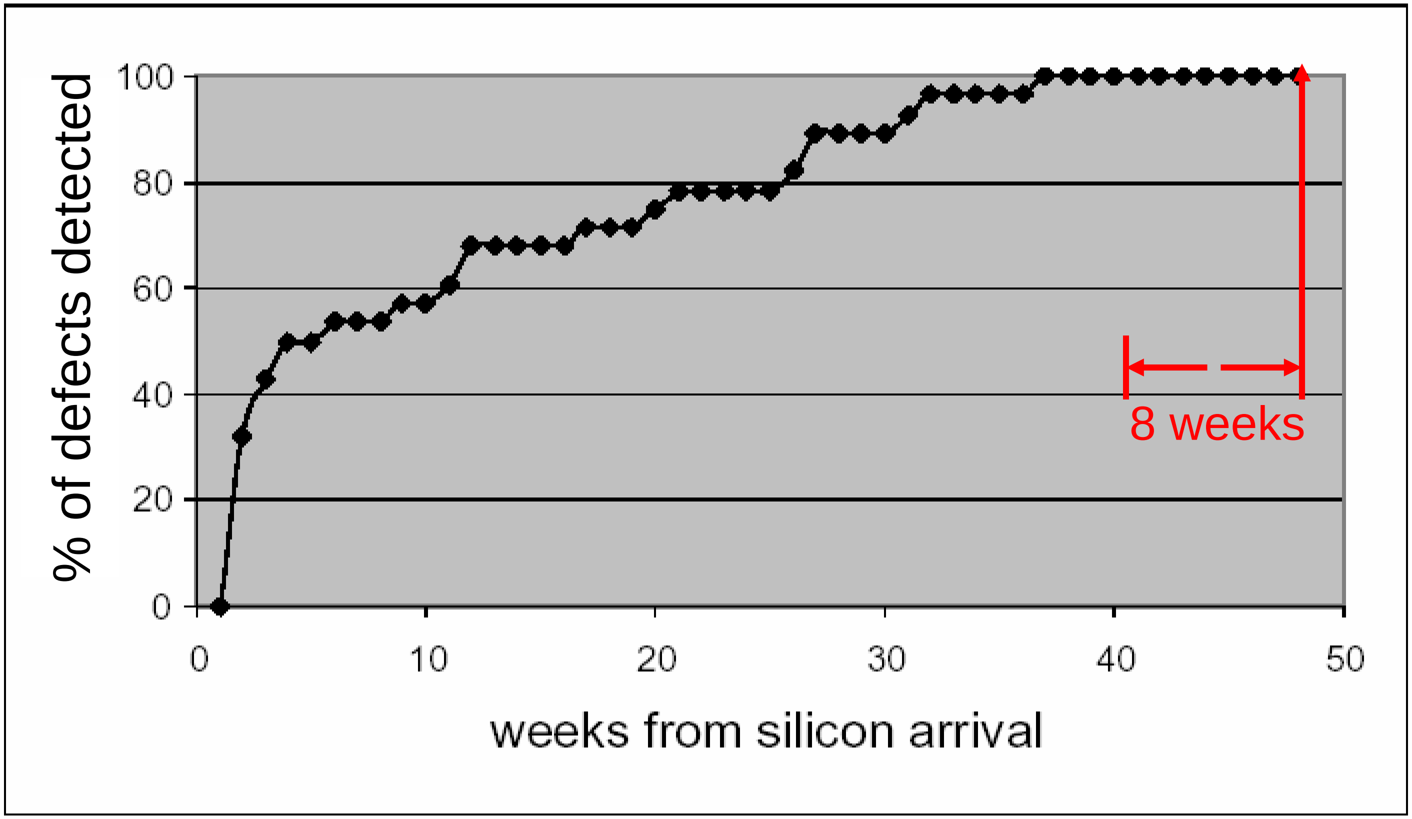
Vendor characterizes the conditions that exercise the defect

Vendor sends a **defect signature** to processors in the field

Customers patch the HW defect

Additional Advantage: Reduced Time to Market

Pentium-M, Silas et al., 2003

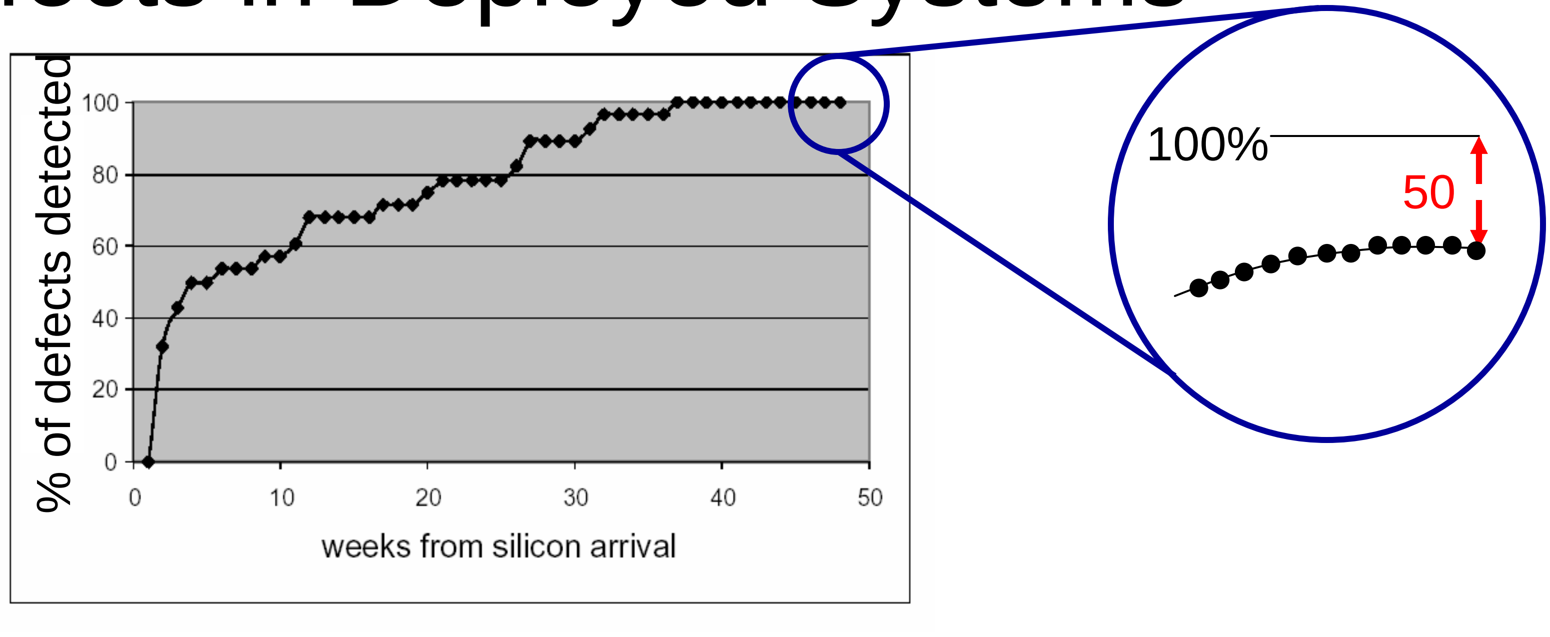


- Reduced time to market → Vital ingredient of profitability

Outline

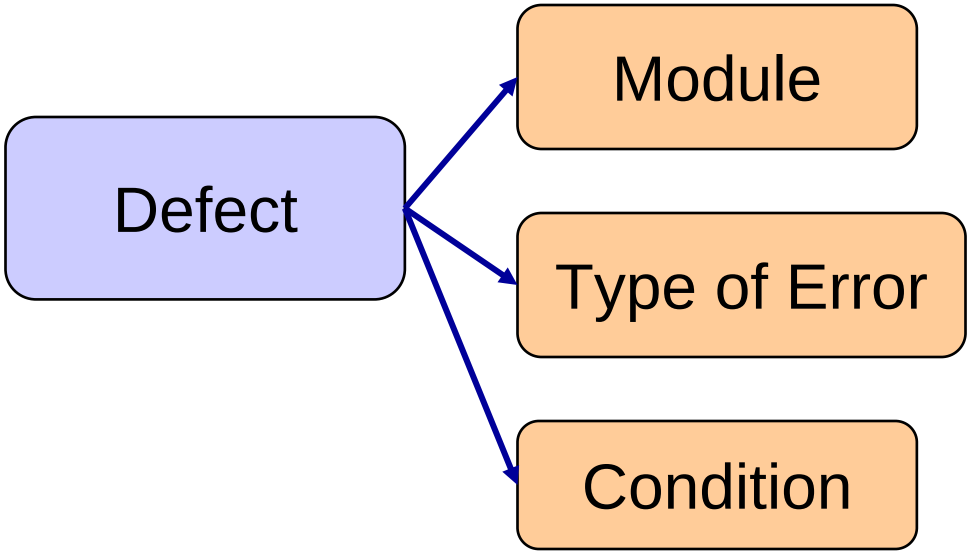
- Analysis and Characterization
- Architecture for Hardware Patching
- Evaluation

Defects in Deployed Systems

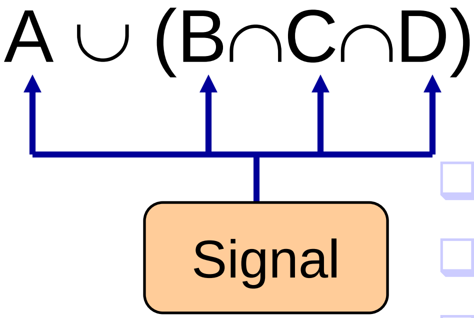


- We studied public domain errata documents for 10 current processors
  - Intel Pentium III, IV, M, and Itanium I and II
  - AMD K6, Athlon, Athlon 64
  - IBM G3 (PPC 750 FX), MOT G4 (MPC 7457)

Dissecting a Defect – from Errata doc.



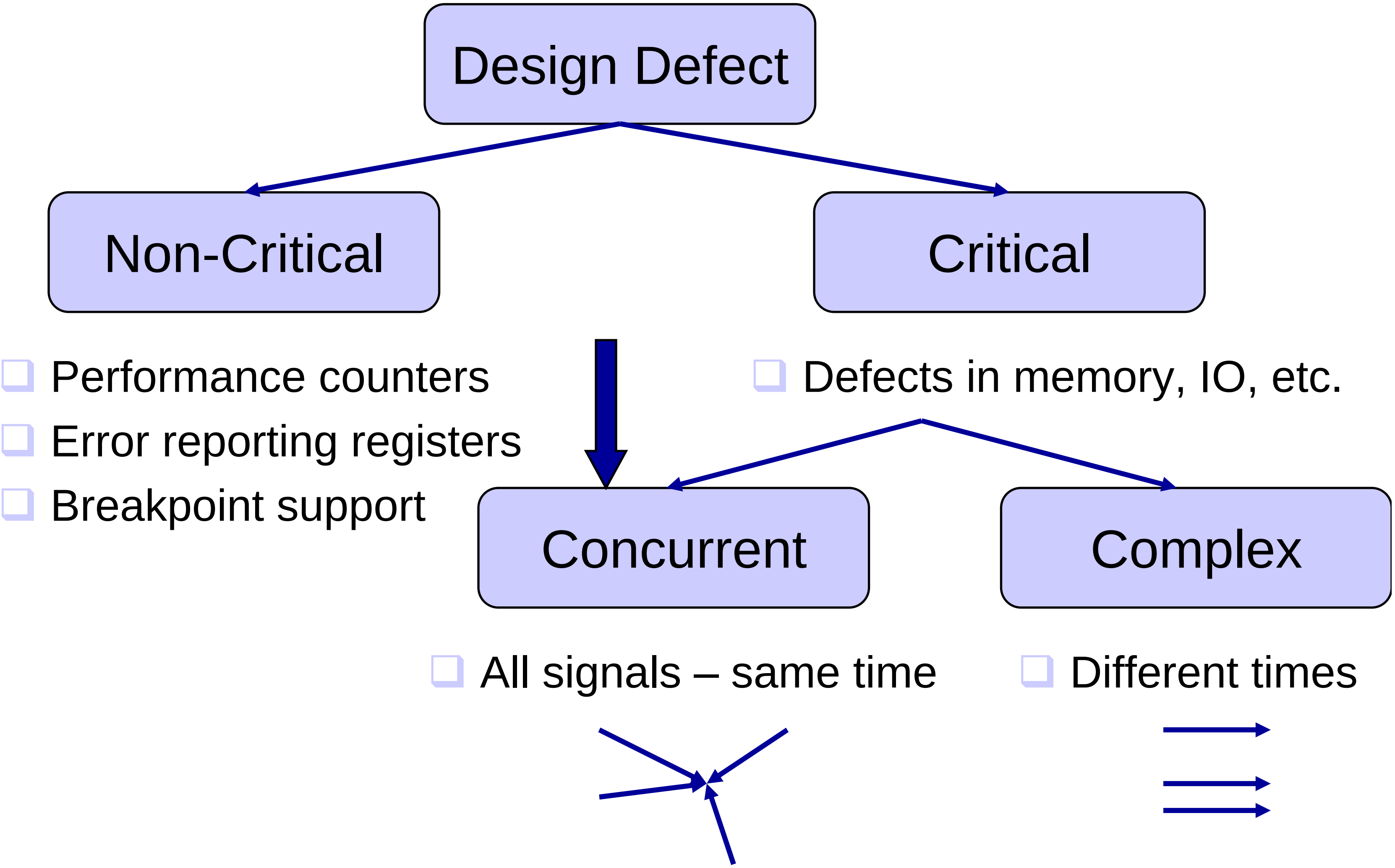
- L1, ALU, Memory, etc.
- Hang, data corruption
  IO failure, wrong data



$$A \cup (B \cap C \cap D)$$

- Snoop
- L1 hit
- IO request
- Low power mode

Types of Defects



Design Defect

Non-Critical

- Performance counters
- Error reporting registers
- Breakpoint support

Critical

- Defects in memory, IO, etc.

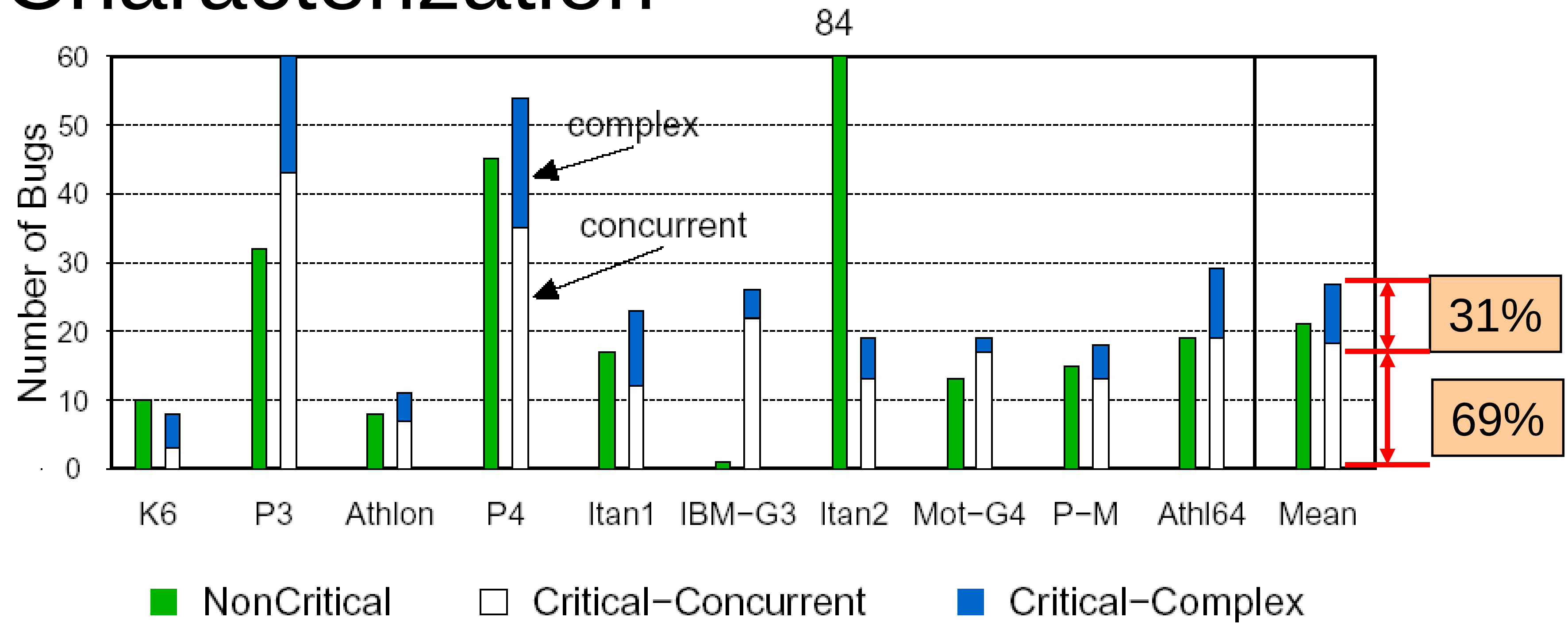
Concurrent

- All signals – same time

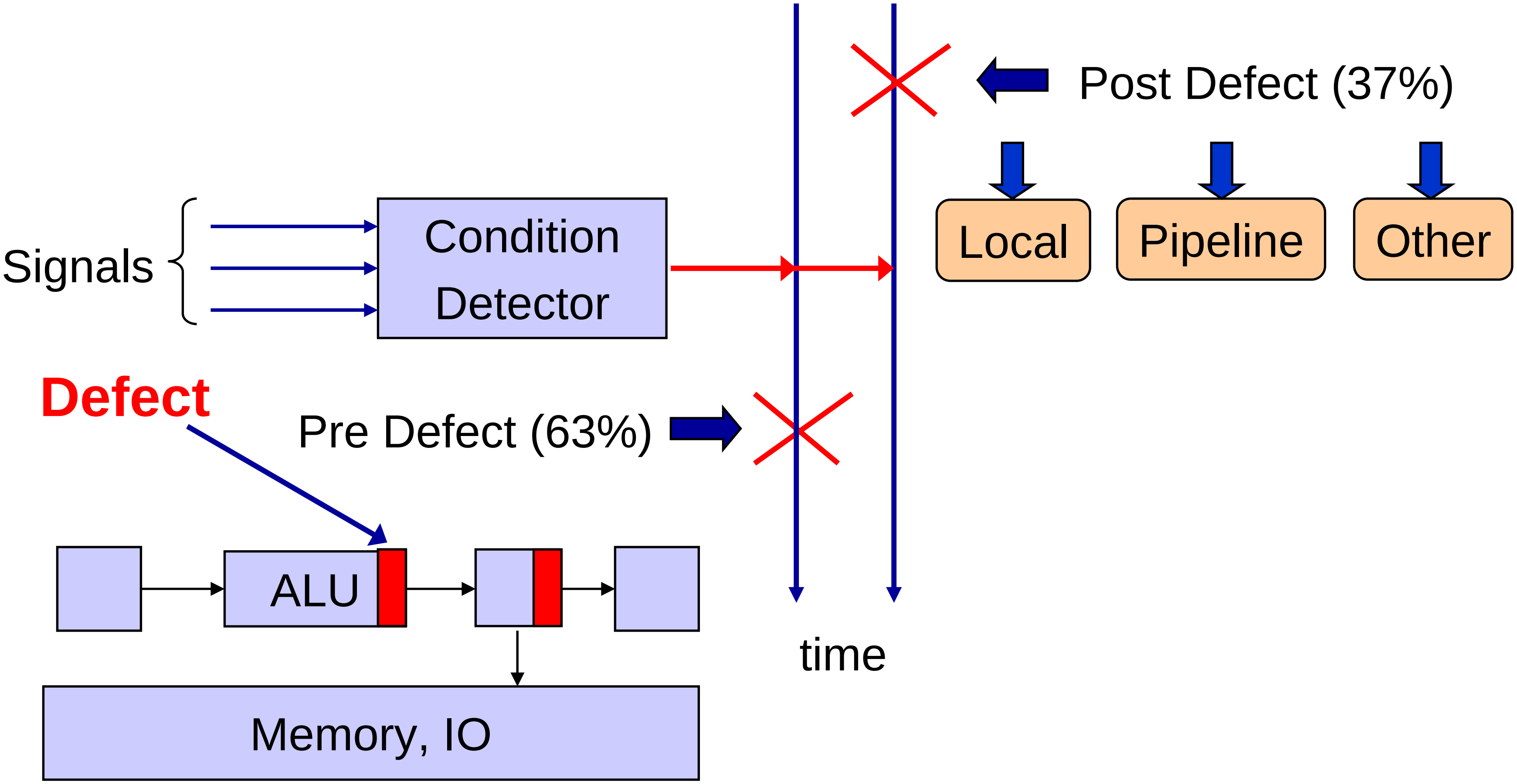
Complex

- Different times

Characterization



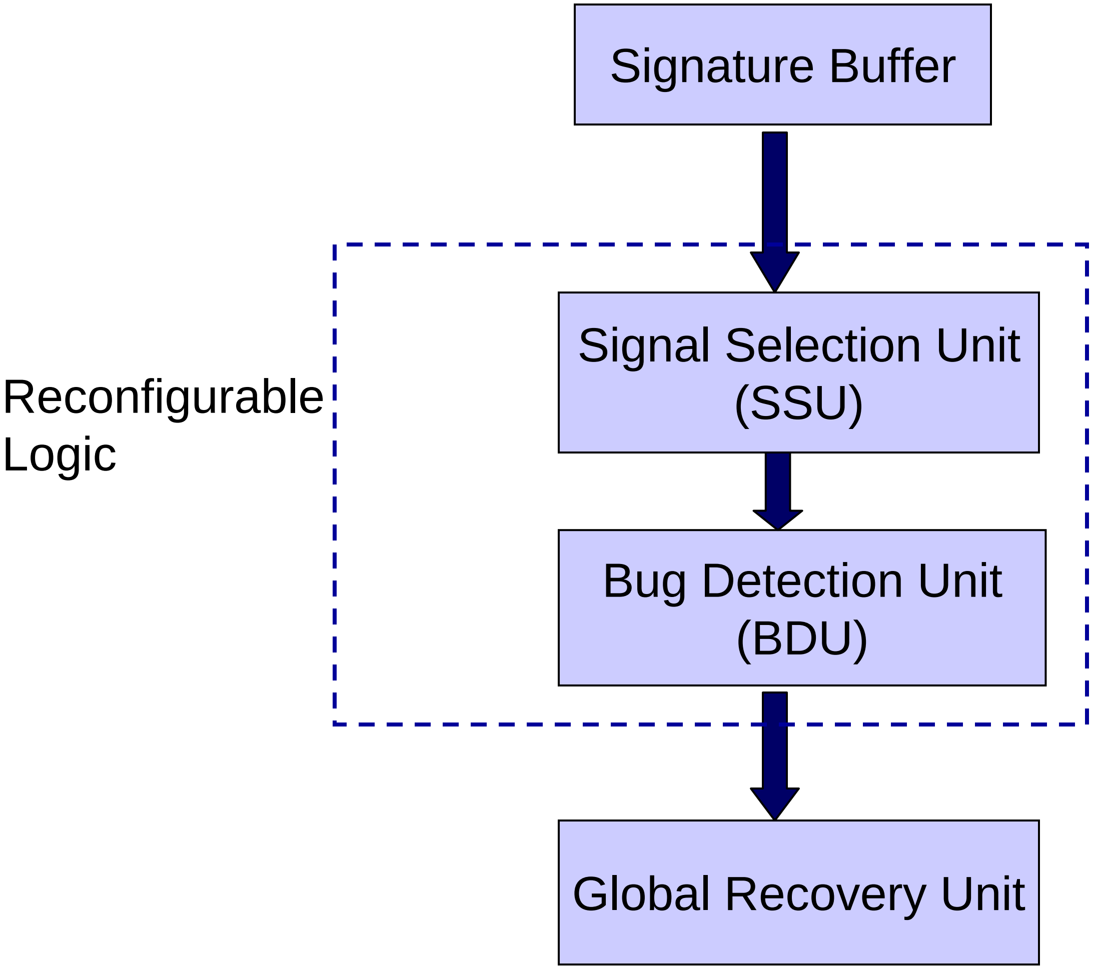
When can the defects be detected ?



Outline

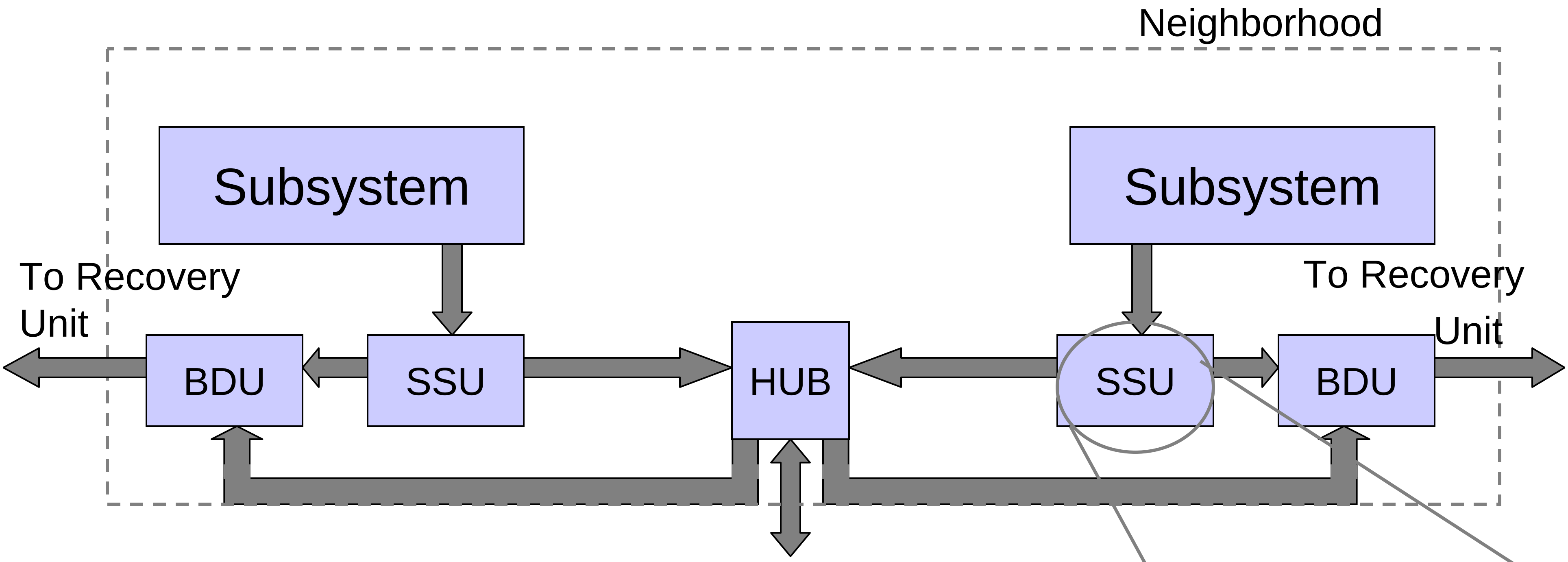
- Analysis and Characterization
- Architecture for Hardware Patching
- Evaluation

Phoenix Conceptual Design



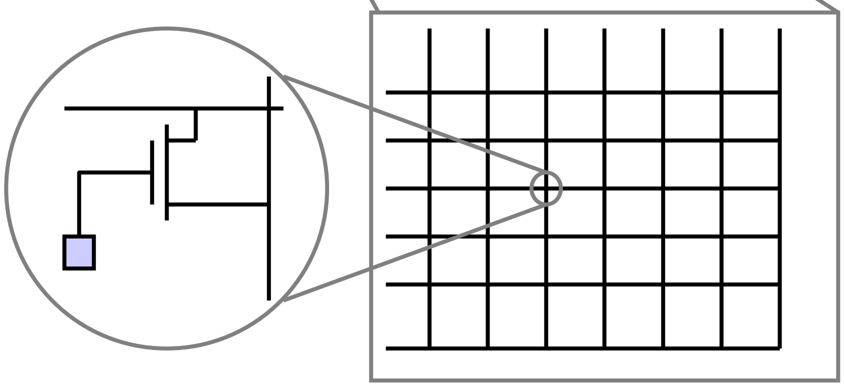
- Store **defect signatures** obtained from vendor
- Program the on-chip reconfigurable logic
- Tap signals from units
- Select a subset
- Collect signals from SSUs
- Compute defect conditions
- Initiate recovery if a defect condition is true

Distributed Design of Phoenix

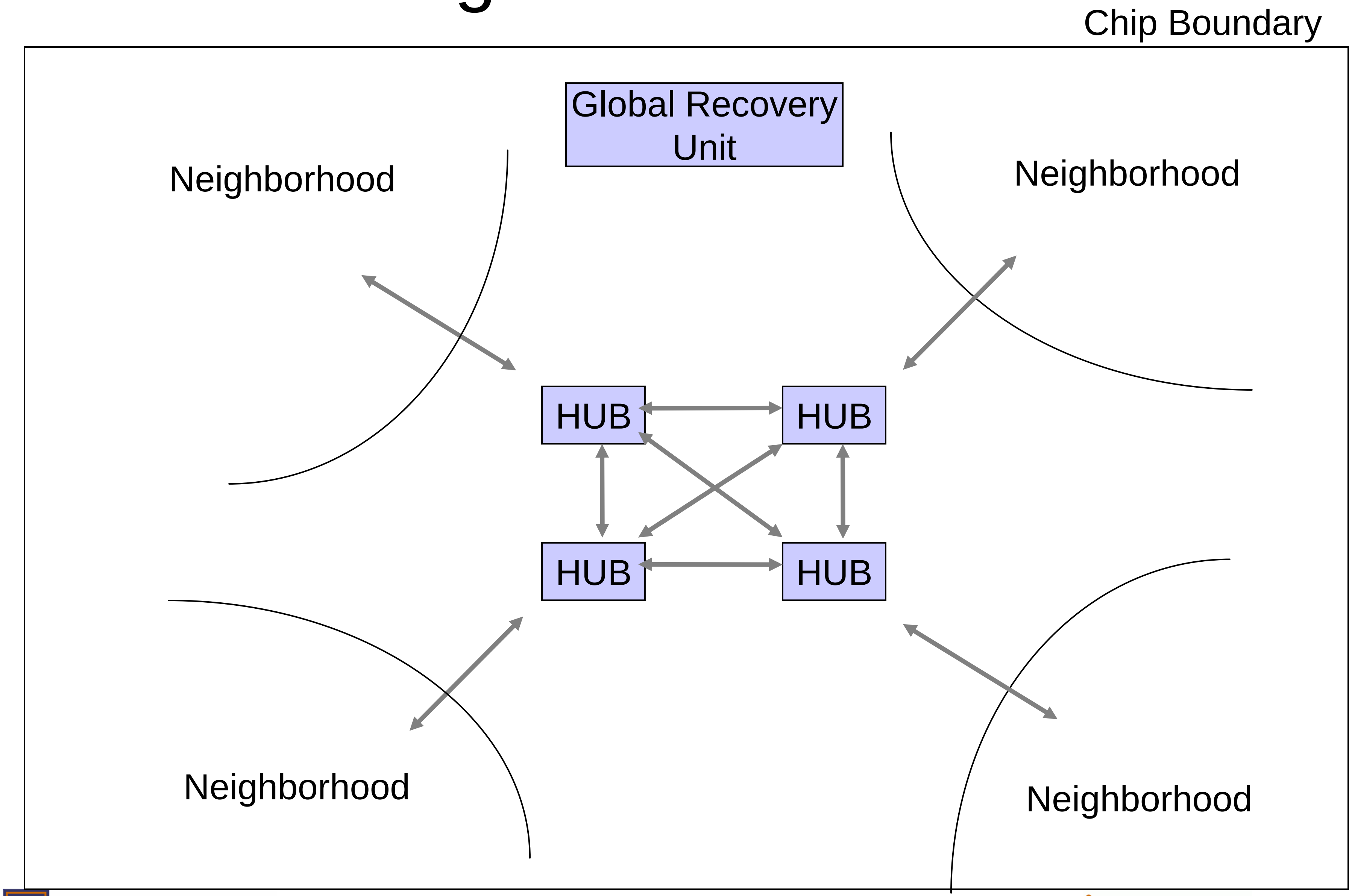


Examples of Subsystems

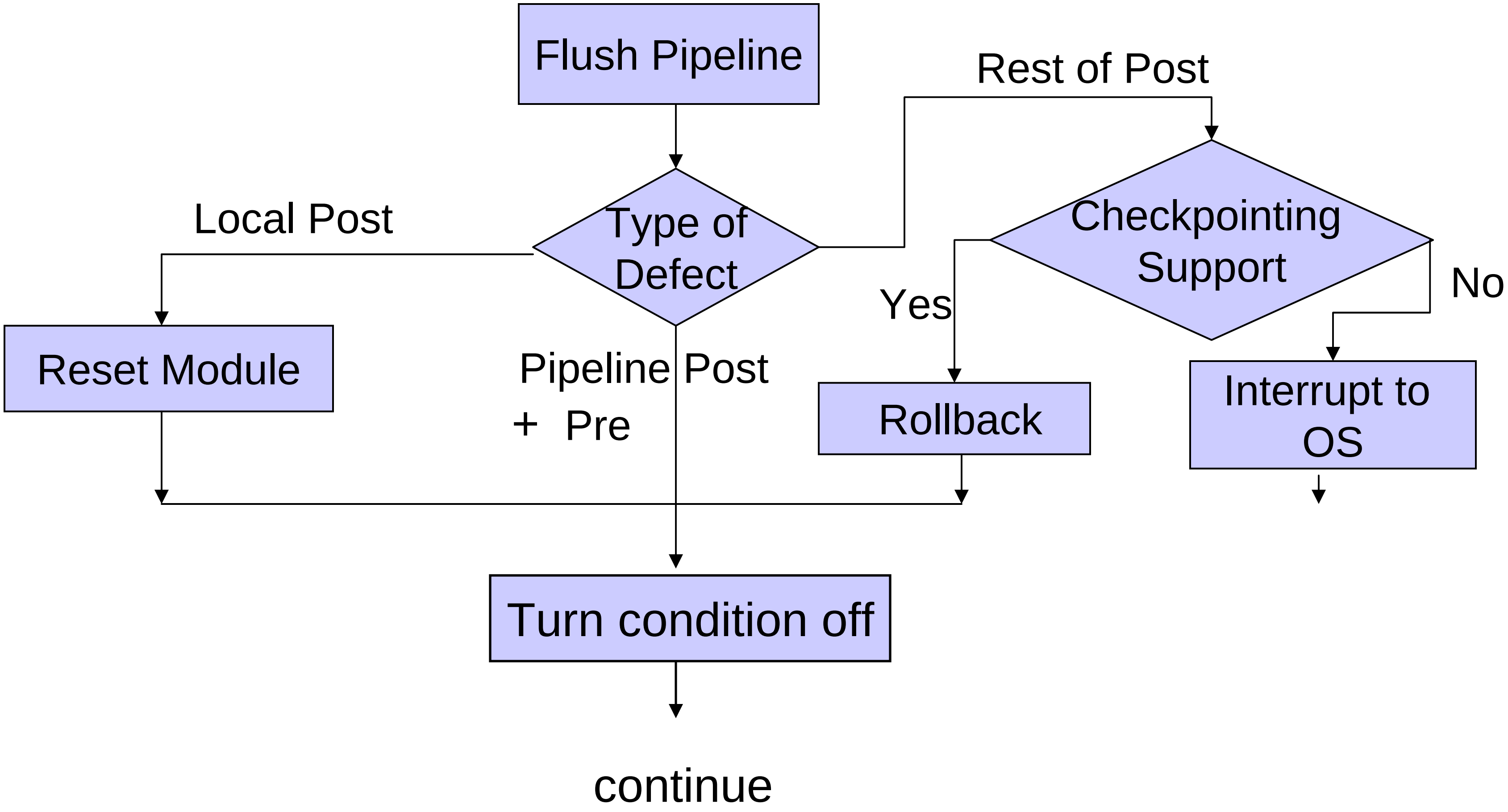
Inst. Cache	FP ALU	Virtual Mem.
Fetch Unit	L1 Cache	IO Cntrl.



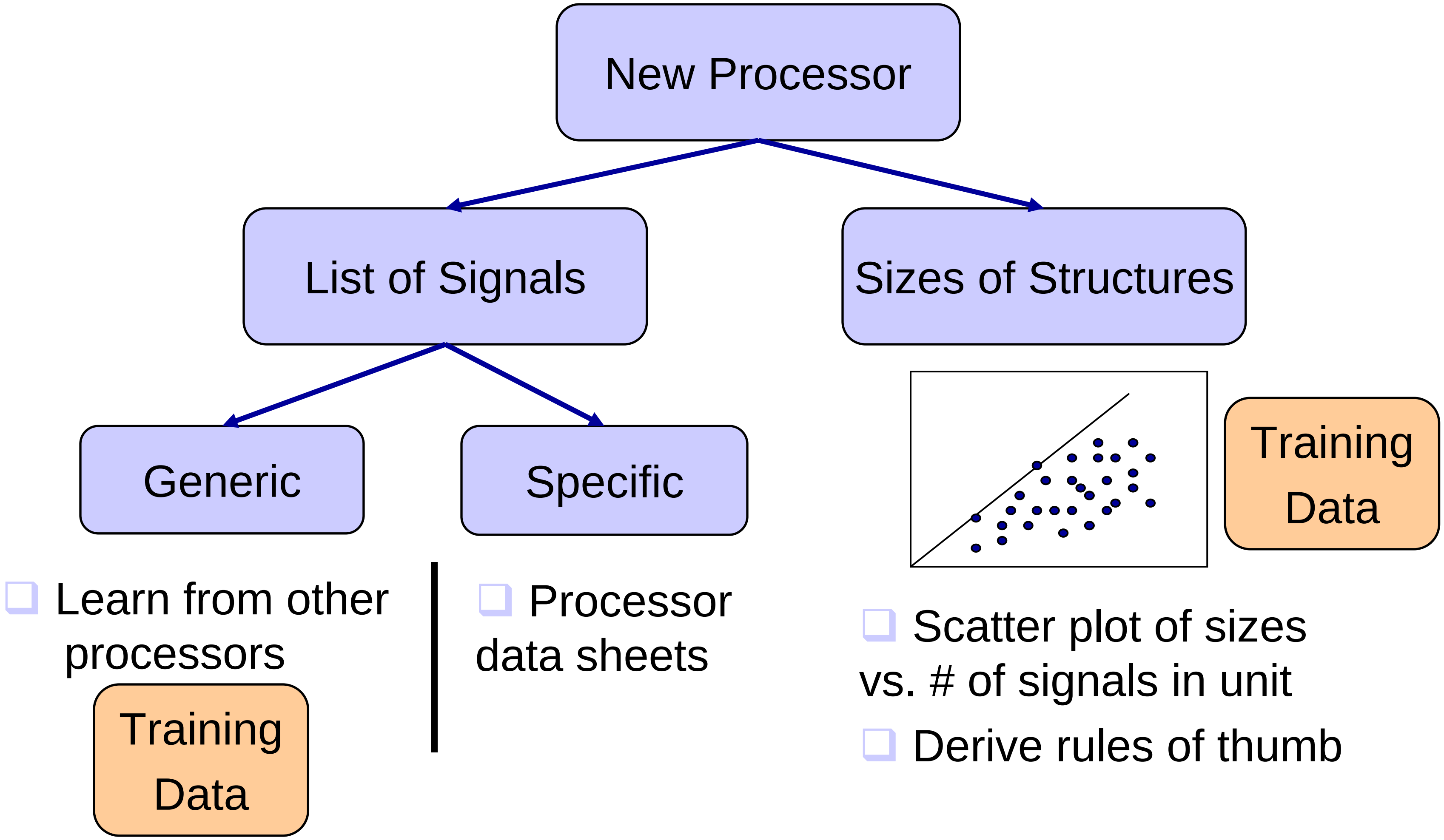
Overall Design



Software Recovery Handler



Designing Phoenix for a New Processor



New Processor

List of Signals

Sizes of Structures

Generic

Specific

Training Data

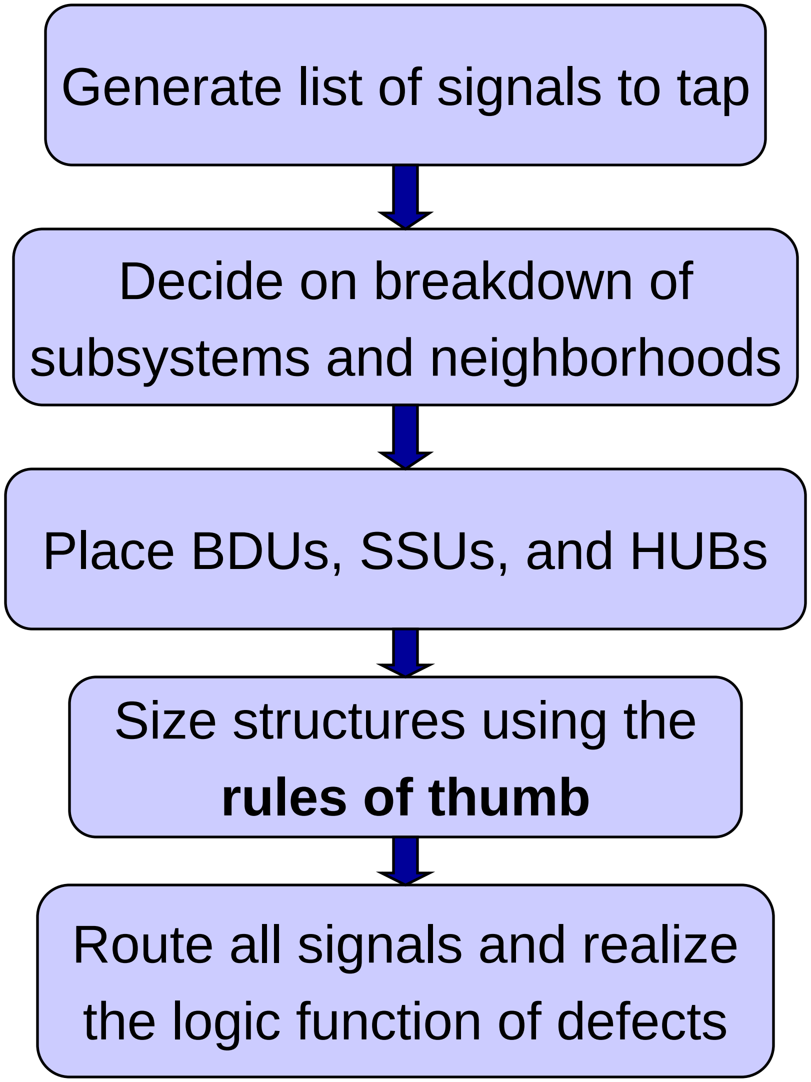
- ❑ Learn from other processors

Training Data

- ❑ Processor data sheets

- ❑ Scatter plot of sizes vs. # of signals in unit
- ❑ Derive rules of thumb

Designing Phoenix for a New Proc. – II



Generate list of signals to tap

Decide on breakdown of subsystems and neighborhoods

Place BDUs, SSUs, and HUBs

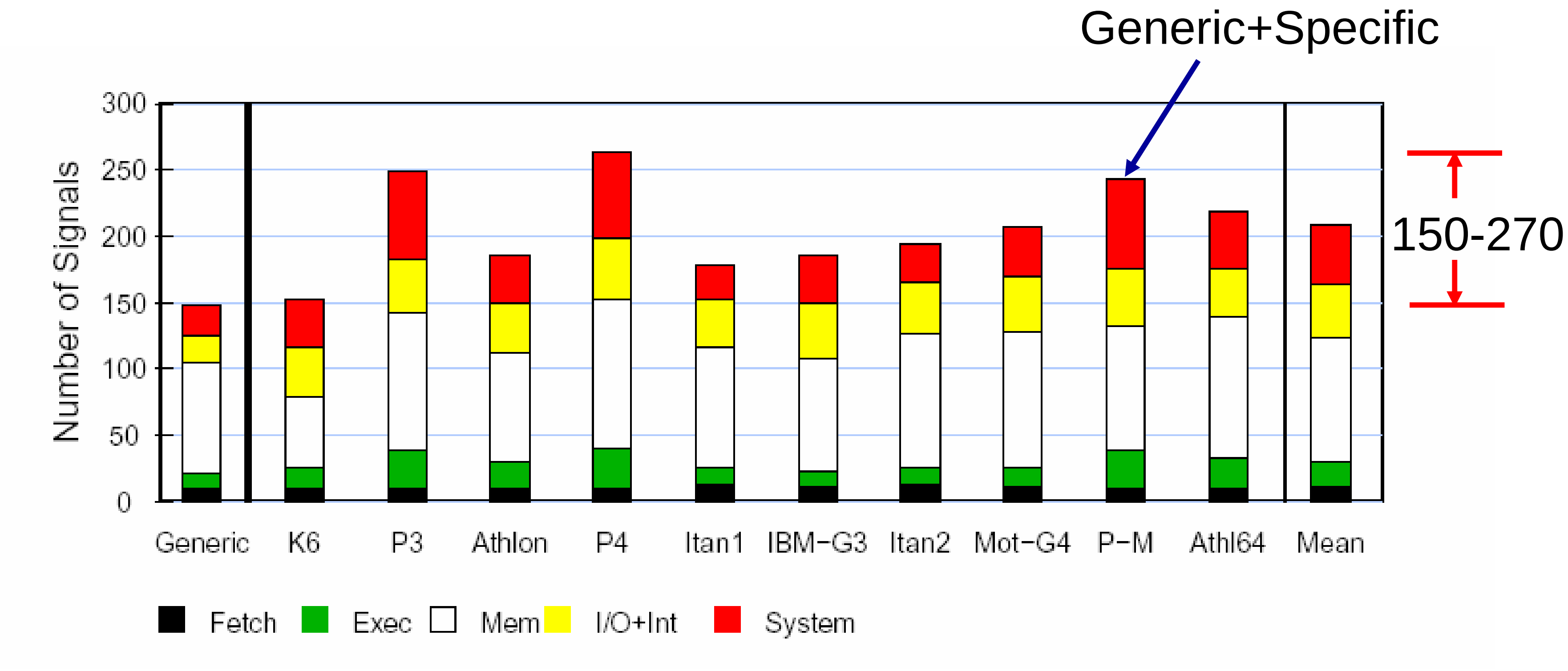
Size structures using the **rules of thumb**

Route all signals and realize the logic function of defects

Outline

- Analysis and Characterization
- Architecture for Hardware Patching
- Evaluation

Signals Tapped



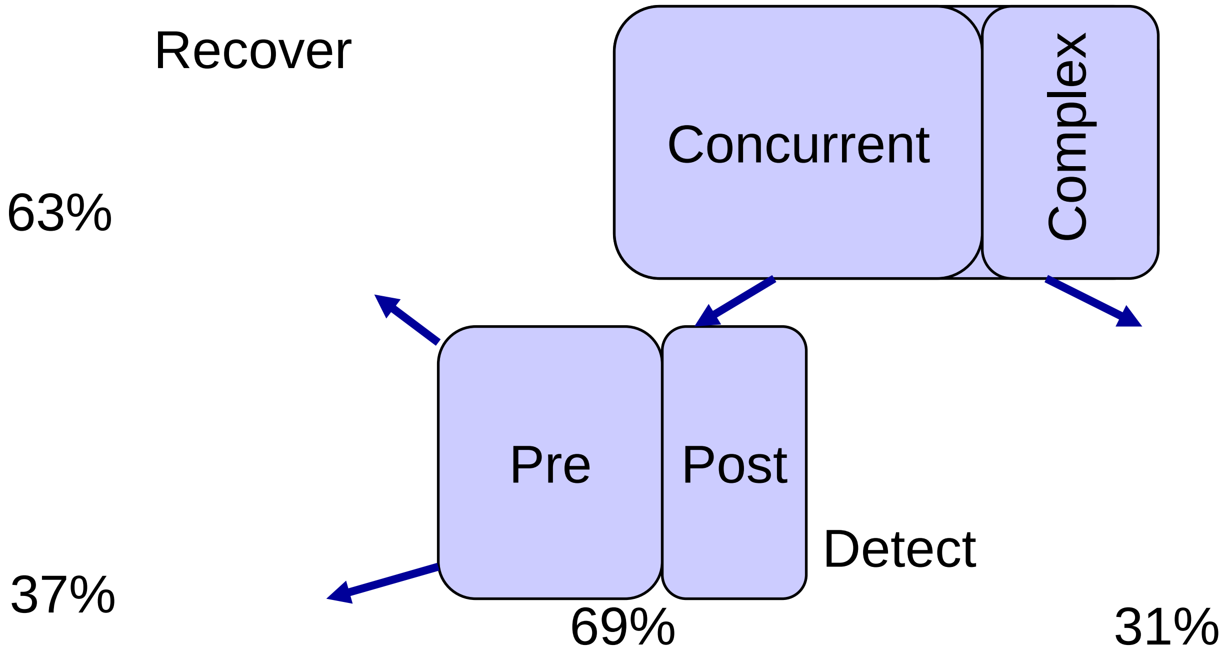
Generic Signals

- L2 hit, low power mode
- ALU access, etc.

Specific Signals

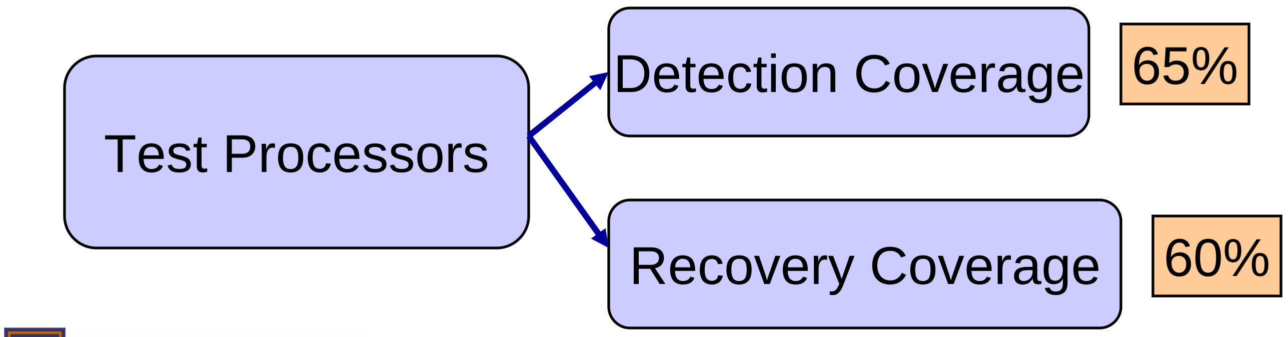
- A20 pin set in Pentium 4
- BAT mode in IBM 750FX

Defect Coverage Results



Training Set:

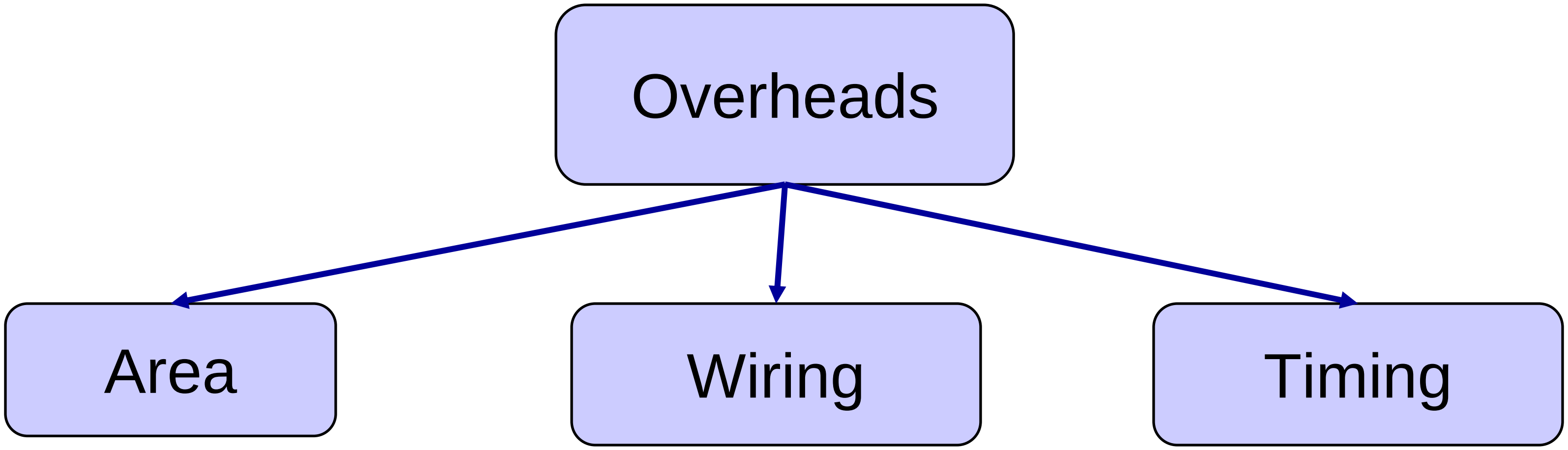
Intel P3, P4, P-M
Itanium I & II
AMD K6, K7
AMD Opteron
IBM G3
Motorola G4



Test Set:

UltraSparc II
Intel IXP 1200
Intel PXA 270
PPC 970
Pentium D

Overheads



Overheads

Area

Wiring

Timing

- Programmable logic (PLA & interconnect)
- Estimated using PLA layouts (Khatri et al.)

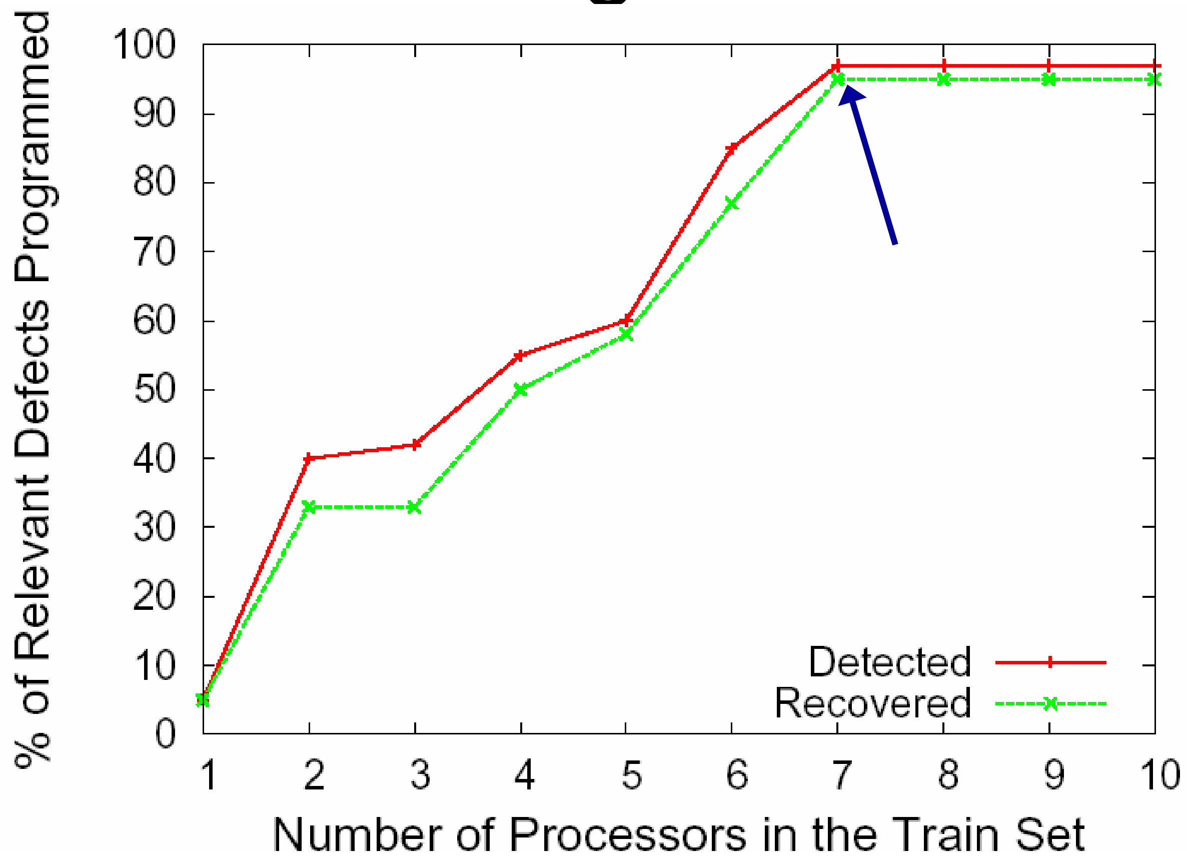
0.05%

- Wires to route signals
- Estimated using Rent’s rule

0.48%

None

Impact of Training Set Size



- Train set only needs to have 7 processors
- Coverage in new processors is very high

Conclusion

- We analyzed the defects in 10 processors
- Phoenix novel on-chip programmable HW
- Evaluated impact:
  - 150 – 270 signals tapped
  - Negligible area, wiring, and performance overhead
  - Defect coverage: 69% detected, 63% recovered
  - Algorithm to automatically size Phoenix for new procs

- We can now live with defects !!!

Phoenix: Detecting and Recovering from Permanent Processor Design Bugs with Programmable Hardware

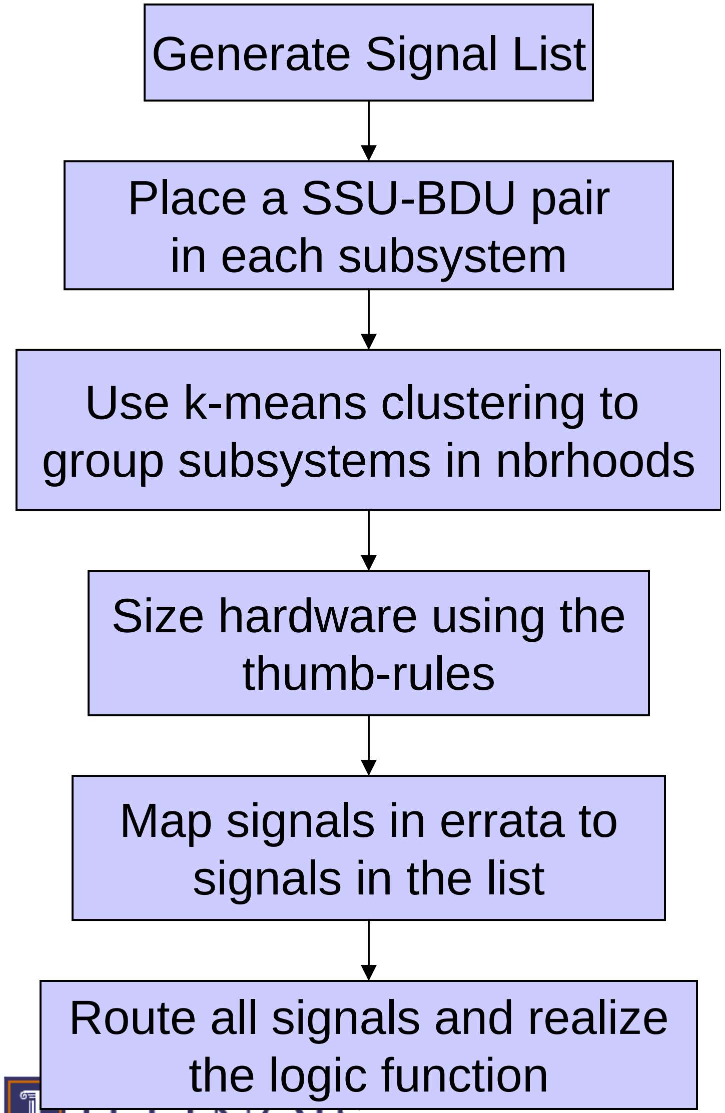
Smruti R. Sarangi
Abhishek Tiwari
Josep Torrellas

University of Illinois at Urbana-Champaign

http://iacoma.cs.uiuc.edu

Backup

Phoenix Algorithm for New Processors



Generate Signal List

Place a SSU-BDU pair in each subsystem

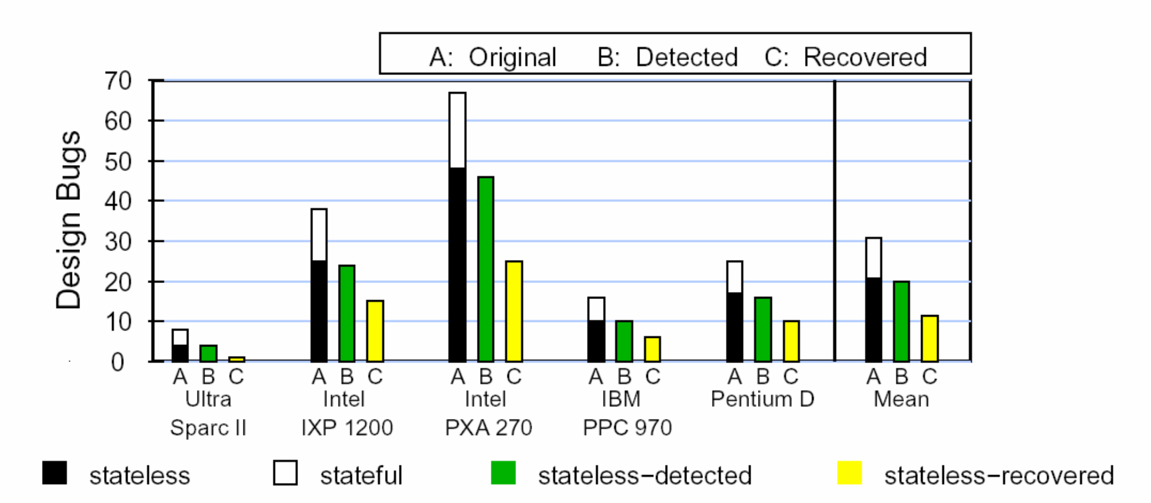
Use k-means clustering to group subsystems in nbrhoods

Size hardware using the thumb-rules

Map signals in errata to signals in the list

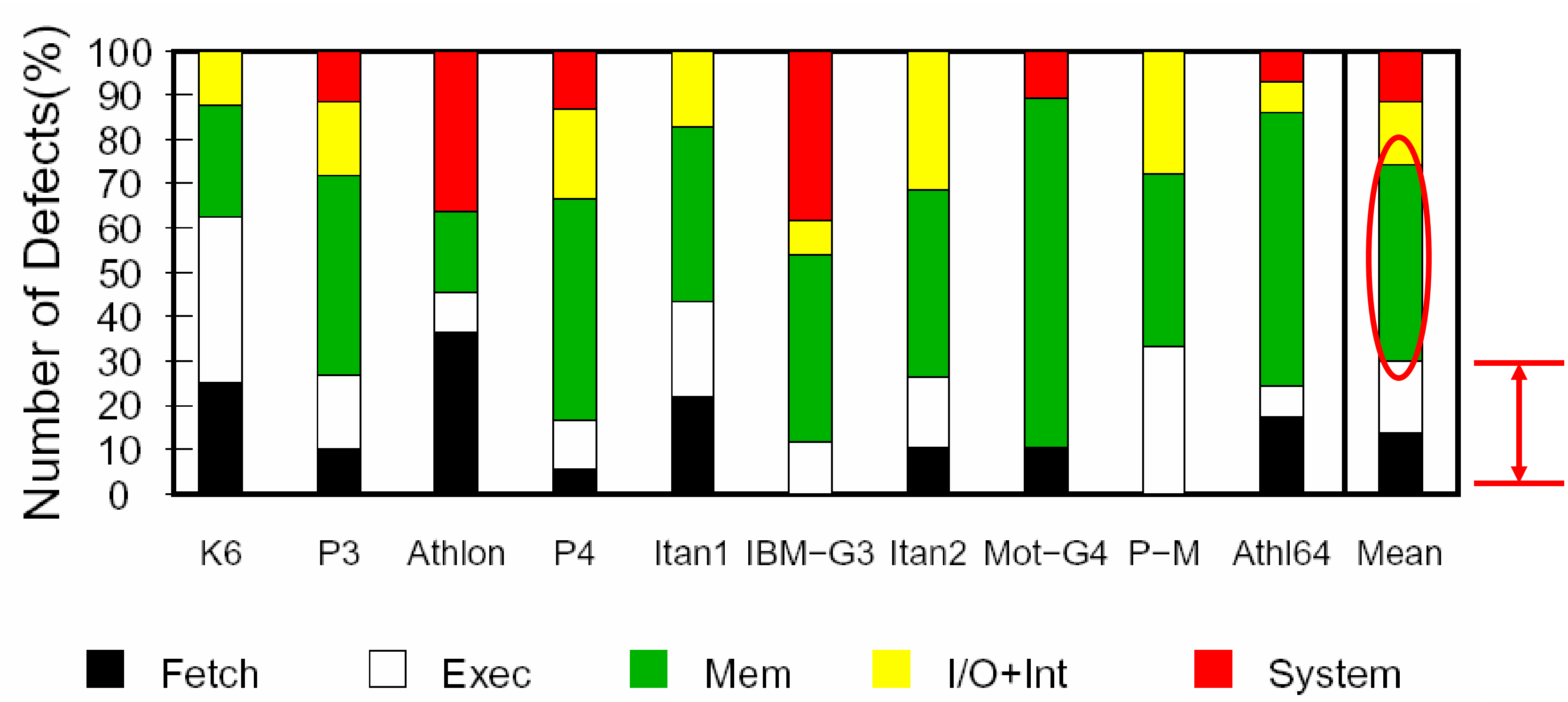
Route all signals and realize the logic function

Defect Coverage for New Processors



- Similar results obtained for 9 Sun processors – UltraSparc III, III+, III++, IIIi, IIIe, IV, IV+, Niagara I and II

Where are the Critical defects ?



- The core is well debugged
- Most of the defects are in the mem. system