# Locality-Based Information Redundancy for Processor Reliability

Martin Dimitrov and Huiyang Zhou
University of Central Florida

School of Electrical Engineering and Computer Science
University of Central Florida

# Motivation

- Soft-Error rate is predicted to increase exponentially in microprocessors [P. Shivakumar et al, DSN 2002]

- Traditional Solutions
  - Space redundancy
  - Time redundancy
  - Information redundancy

- Provide balance between error protection and power/performance overhead

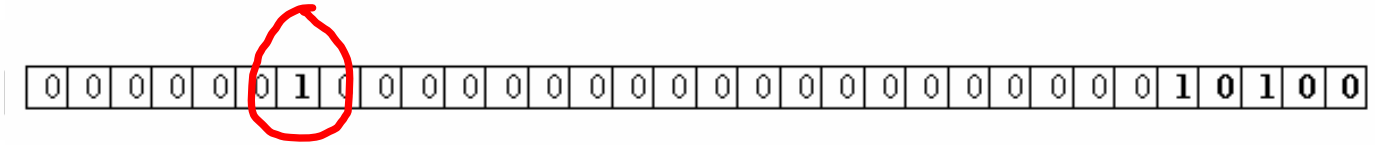
# Our Contribution

- Insight - program localities provide information redundancy
- Explore a novel locality – Limited Variance in Data Values (LVDV)
- Develop simple and effective mechanism for opportunistic soft-error protection

# Exploiting Locality for Reliability

Execution results of instruction A:

20 20 20 20 20 ··· 20 **33554452**




Violation of the constant locality hints the possibility of an error

# Exploiting Locality for Reliability

Execution results of instruction A:

1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

1 2 3 4 5 … 1 2 **1048578**

Violation of the stride locality
hints the possibility of an error

# Exploiting Locality for Reliability

- Utilize program behavior and localities to encode correctness
  - Value locality
  - Instruction Reuse [IRTR, Gomma et al, ISCA 2005 ]
  - Memory Region Locality
  - Branch Predictability [ReStore, Wang et al, DSN 2005]
- Choice of locality:
  - General – protect most instructions
  - Low false positive rate

# Limited Variance in Data Values (LVDV)

- Extends Traditional Value Locality
- Variance between two values is defined as a simple XOR
- Variance can be encoded by dividing 32 bits into N chunks
  - If any of the bits in a chunk are set, we use 1 to encode the entire chunk.
  - Otherwise, we use 0 to encode the chunk

# Application of LVDV Locality

Execution results of instruction A:

1 60 12 40 2 7 **33554452**

No apparent pattern! However, the values are usually within a certain small range.

# Applications of LVDV Locality

Heap memory addresses produced by instruction A exhibit no stride locality :

0x11112654 0x11117838 …

0x11111200 0x11119088 0x01117854

Text Segment Address!

# Applications of LVDV Locality

Heap memory addresses produced by instruction A exhibit no stride locality :

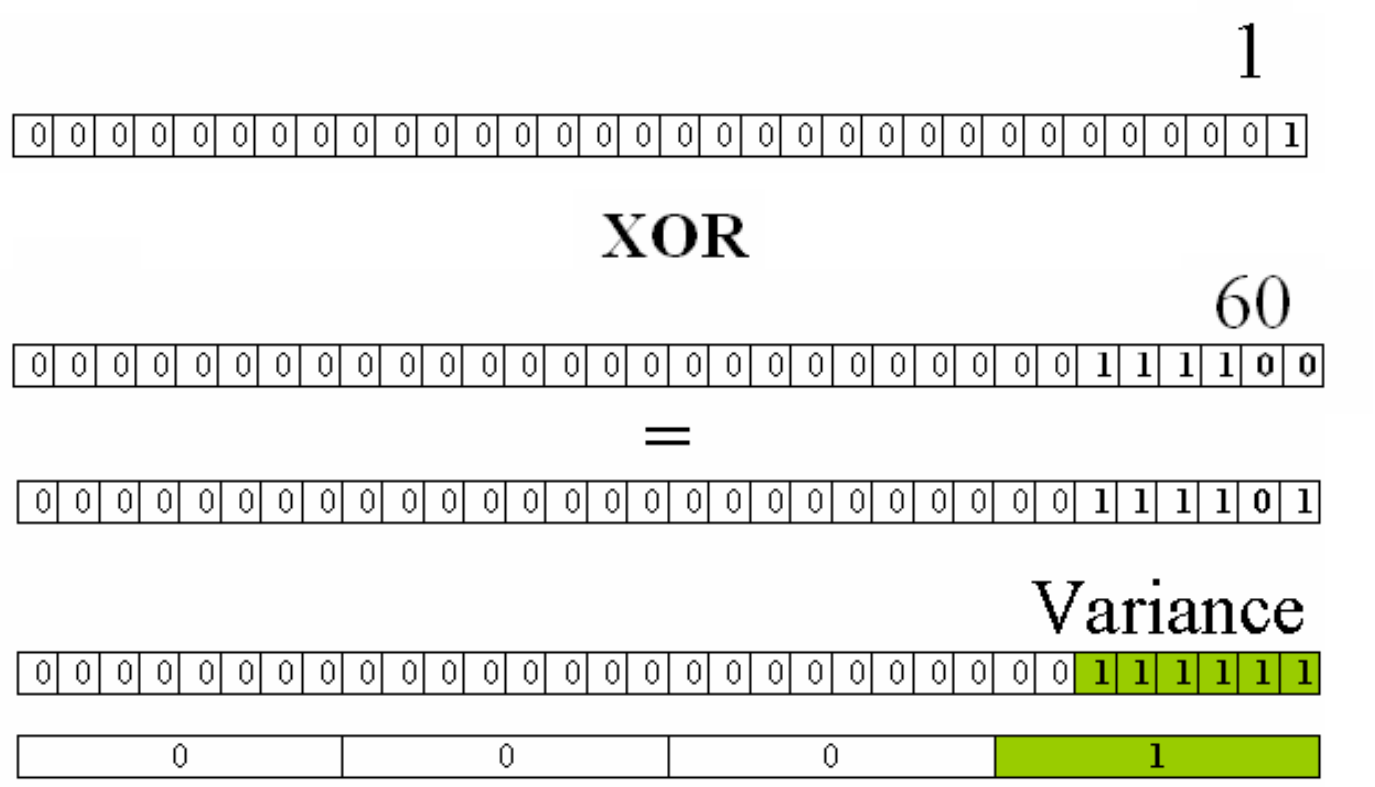
0x11112654 0x11117838 …

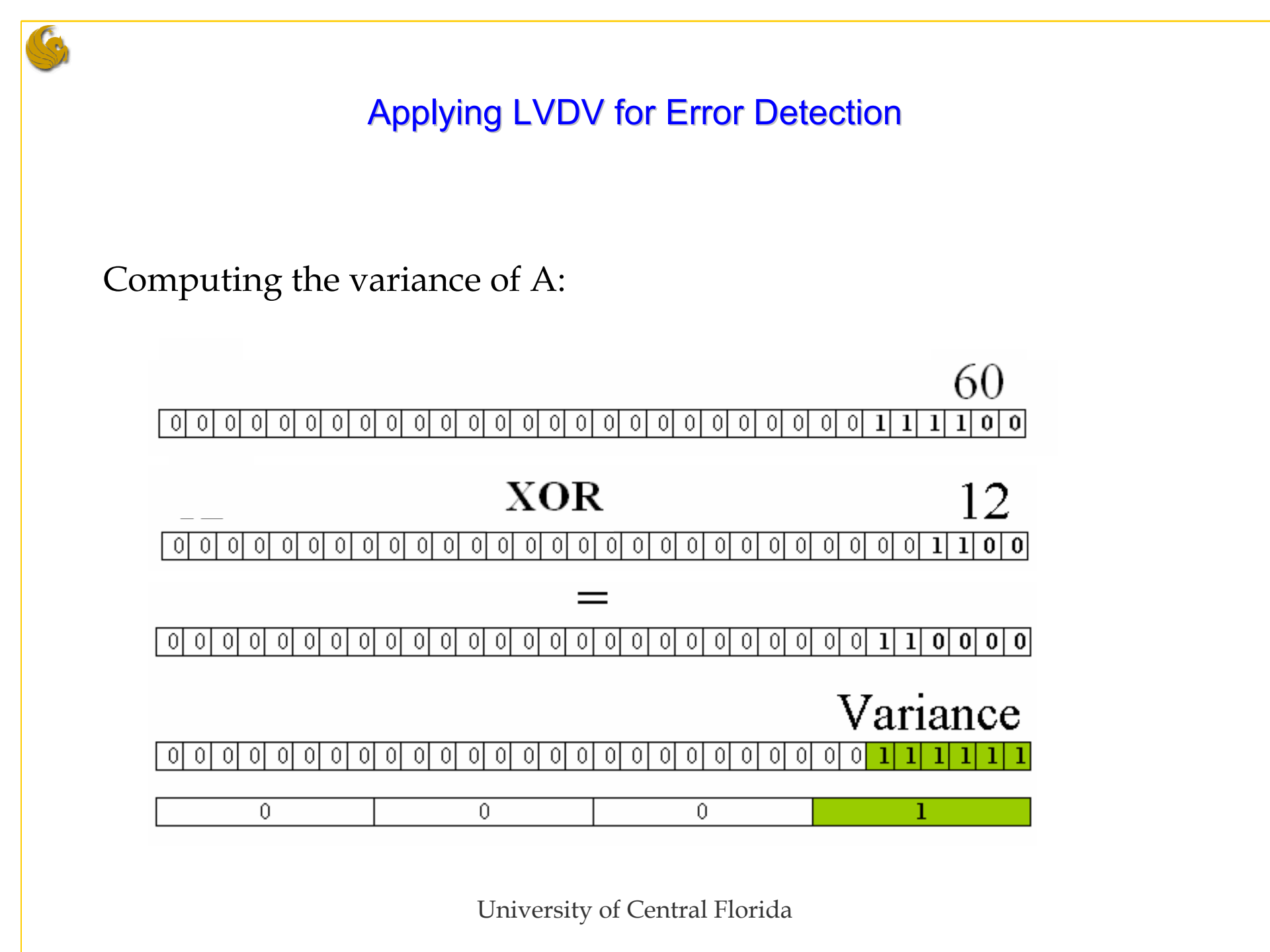
0x11111200 0x11119088 0x71117800

Stack Segment Address!

# Applying LVDV for Error Detection

Computing the variance of A:



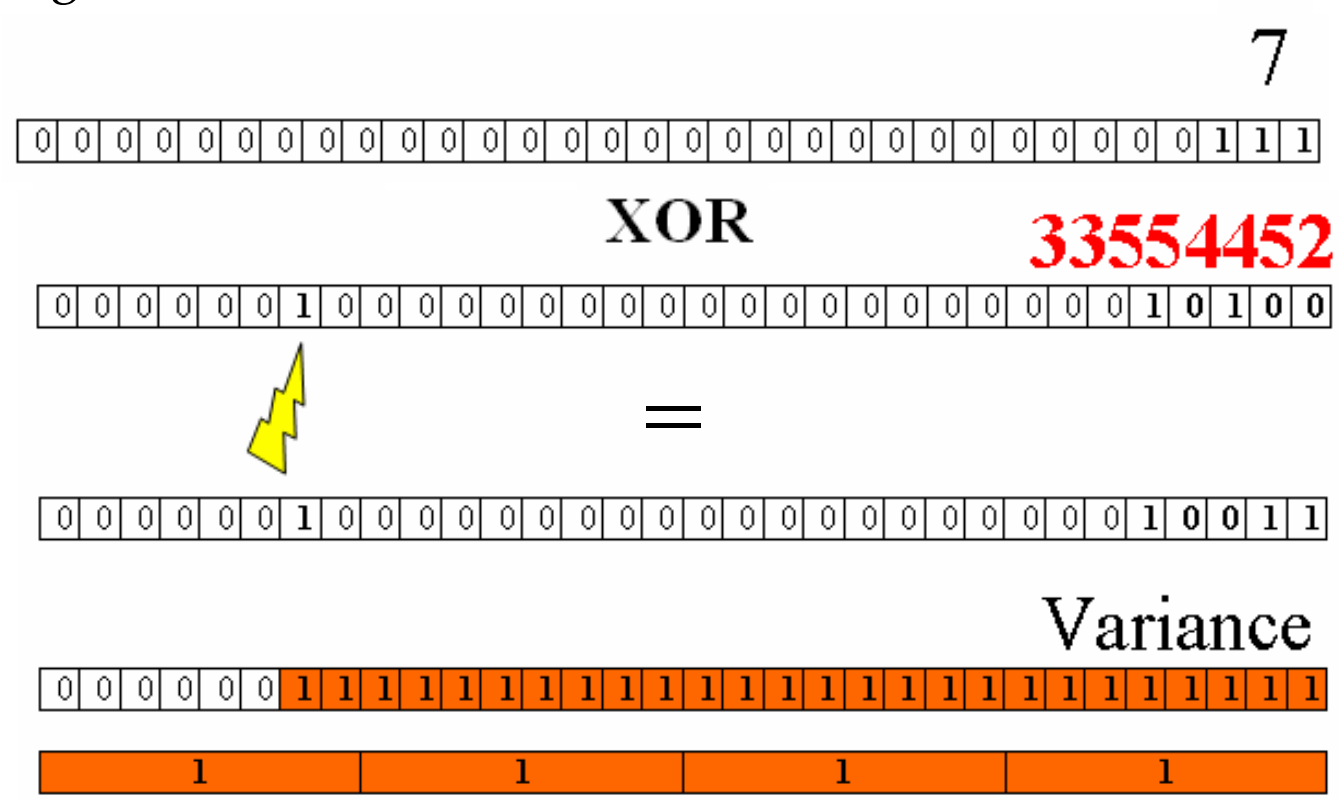


# Applying LVDV for Error Detection

Computing the variance of A:

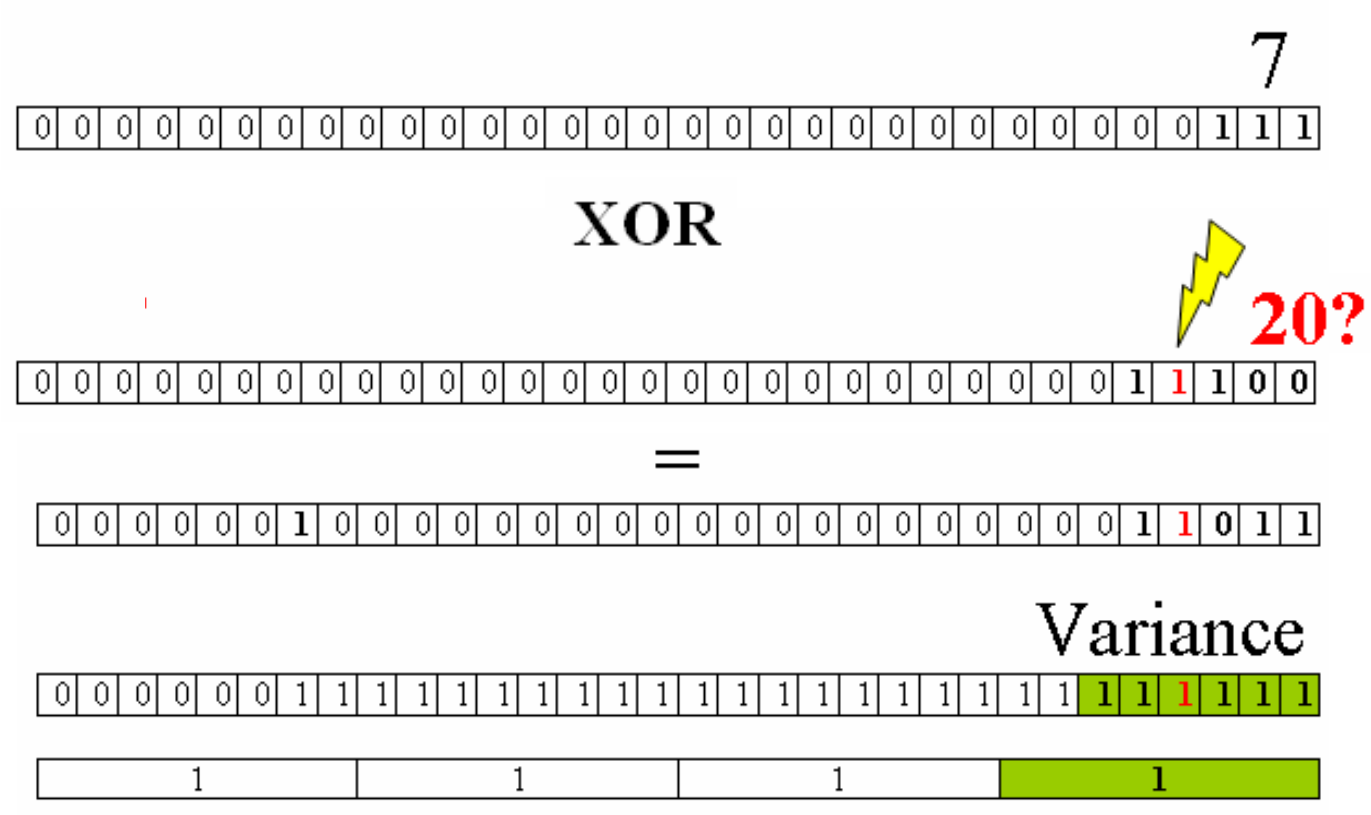
# Applying LVDV for Error Detection

Detecting an error in A:



# Applying LVDV for Error Detection

Loss of Error Coverage:



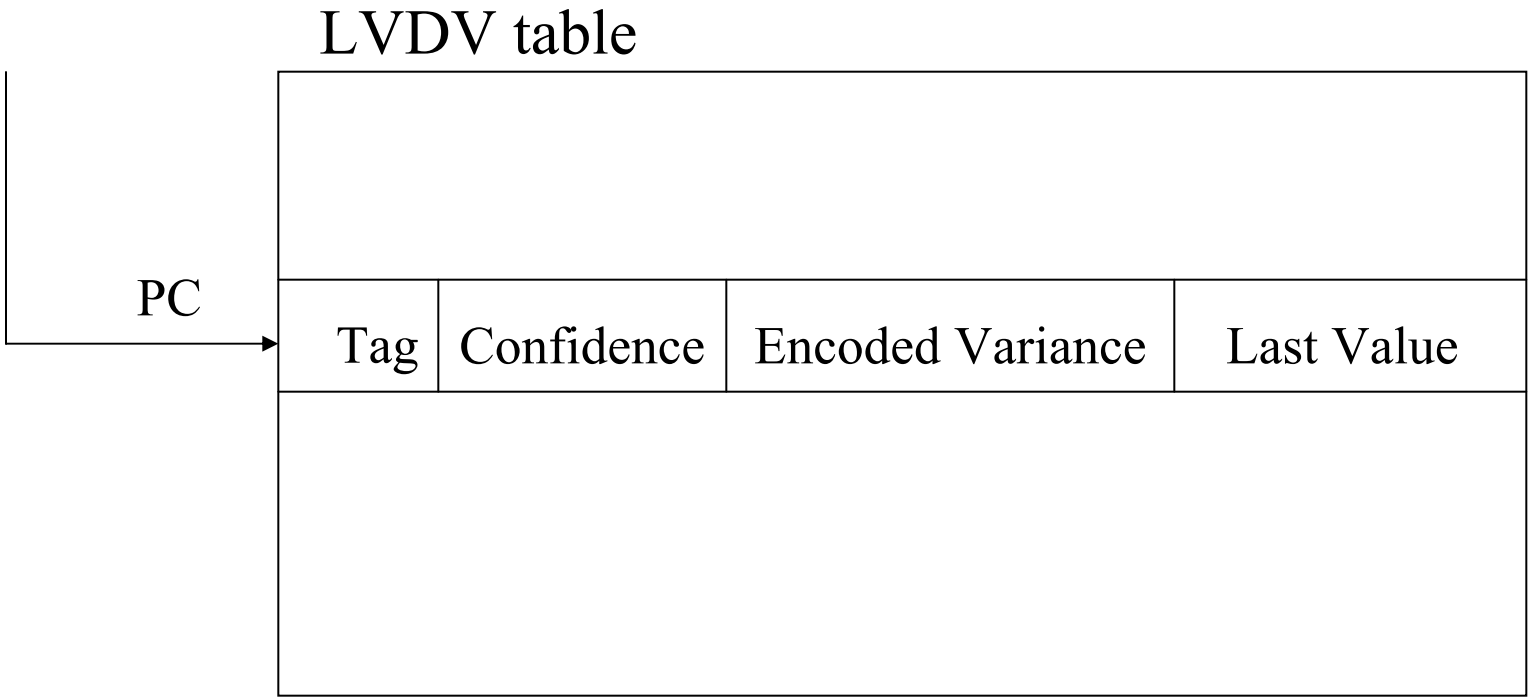
# Advantages of LVDV

- Extents traditional value locality to protect a large fraction of result bits
- Can also capture the region locality in memory references
- Provides information redundancy – no redundant execution required

# Disadvantages of LVDV

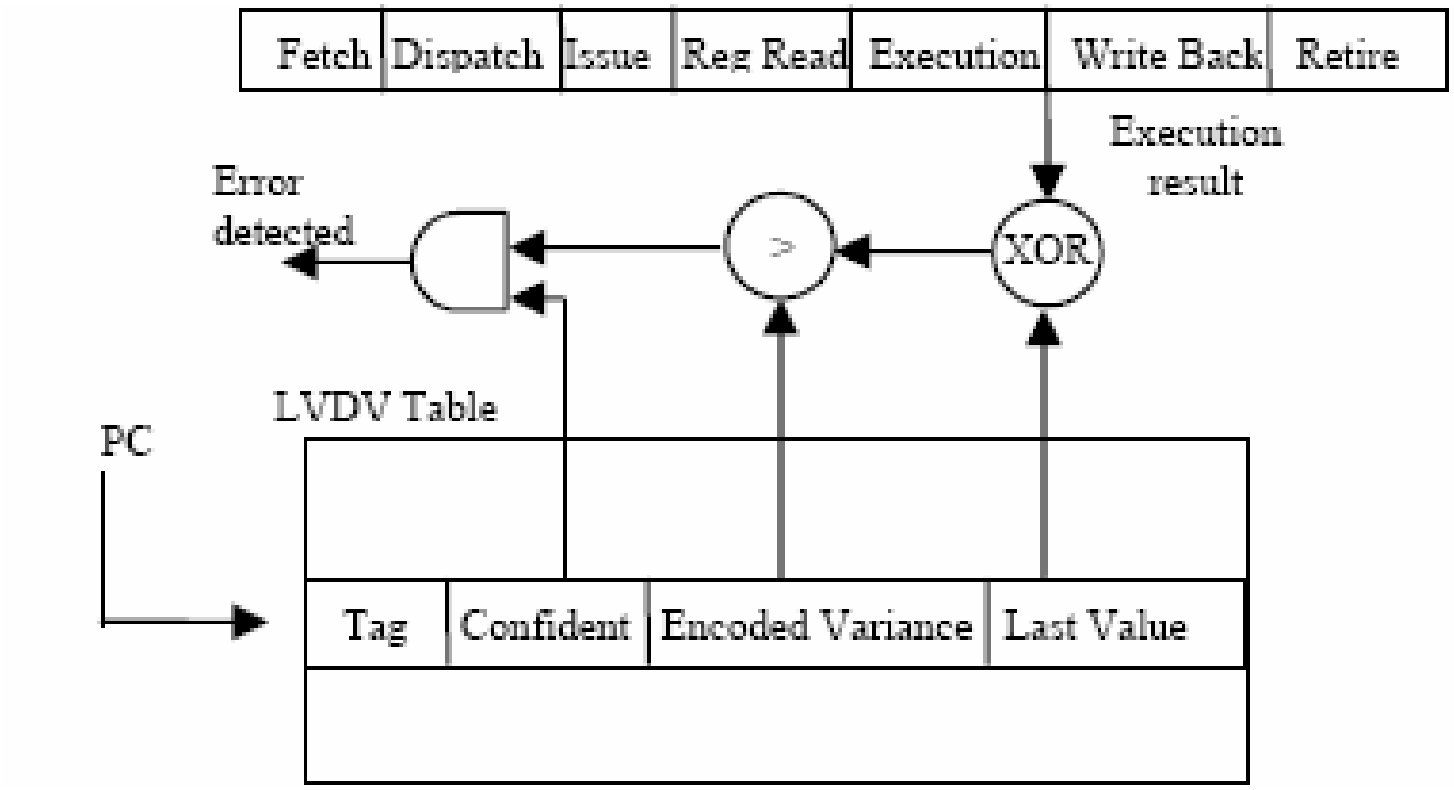
- Some result bits remain unprotected
- Does not work well with floating point values

# LVDV Architecture
## The LVDV table




The main structure in our architecture is an LVDV table.

# LVDV Architecture




Interaction of the LVDV table with the processor pipeline.

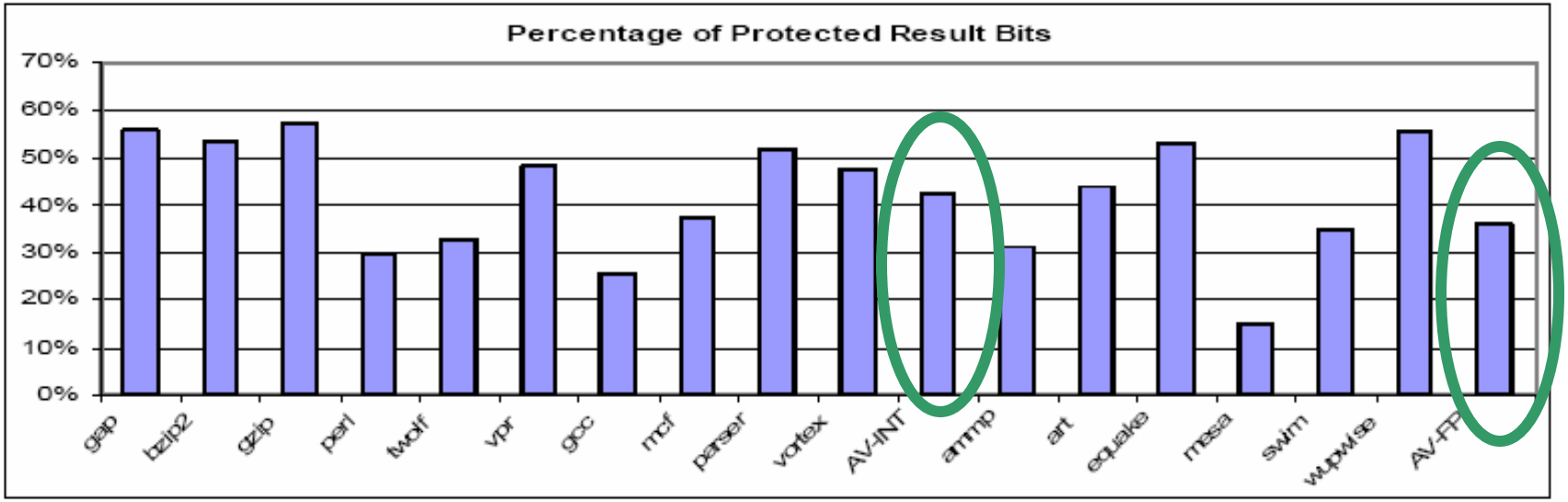
# Reliability and Complexity impact of the LVDV Table

- What if a soft error occurs in the LVDV table itself?
  - Loss of coverage
  - False positive error alert
- Impact on cycle time
  - Only PC needed to start the access
  - The access has to be complete by the end of execution stage
  - LVDV is not on the critical path of the processor

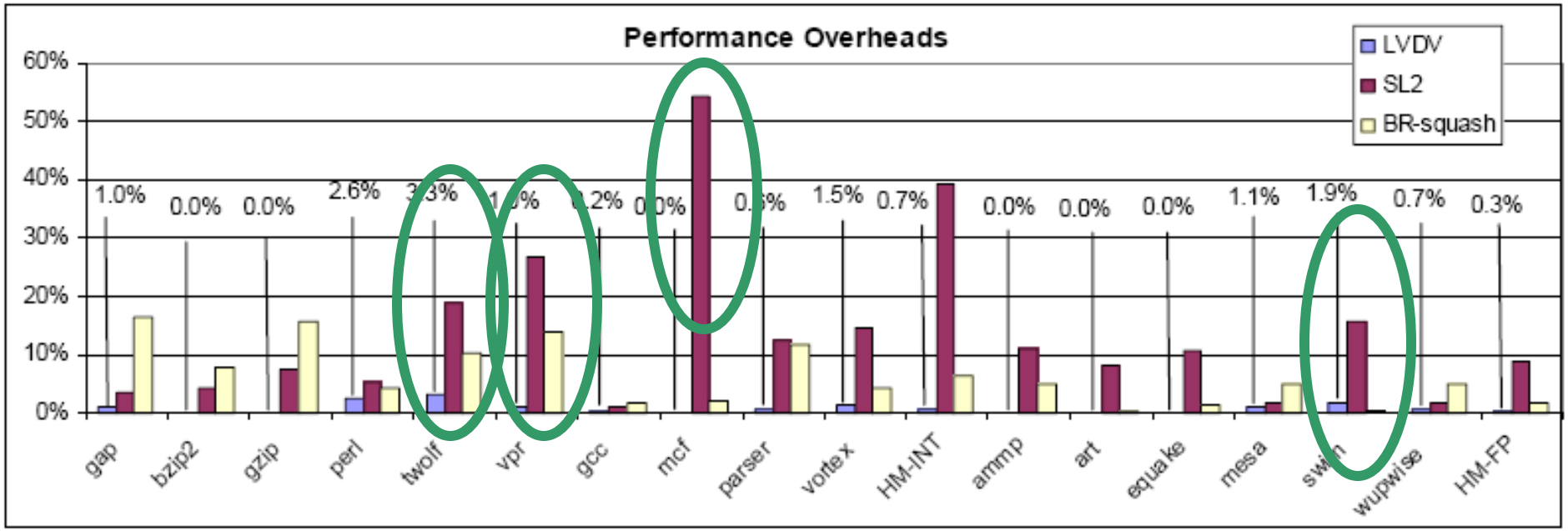
# Experimental Methodology

- Error injection into the Issue Queue and Functional Units
- Compare our approach to:
  - Implicit Redundancy Through Reuse (IRTR)
  - Squash on L2-cache miss (SL2) [Weaver, ISCA 2004]
  - Squash on Branch misprediction (BR-squash)

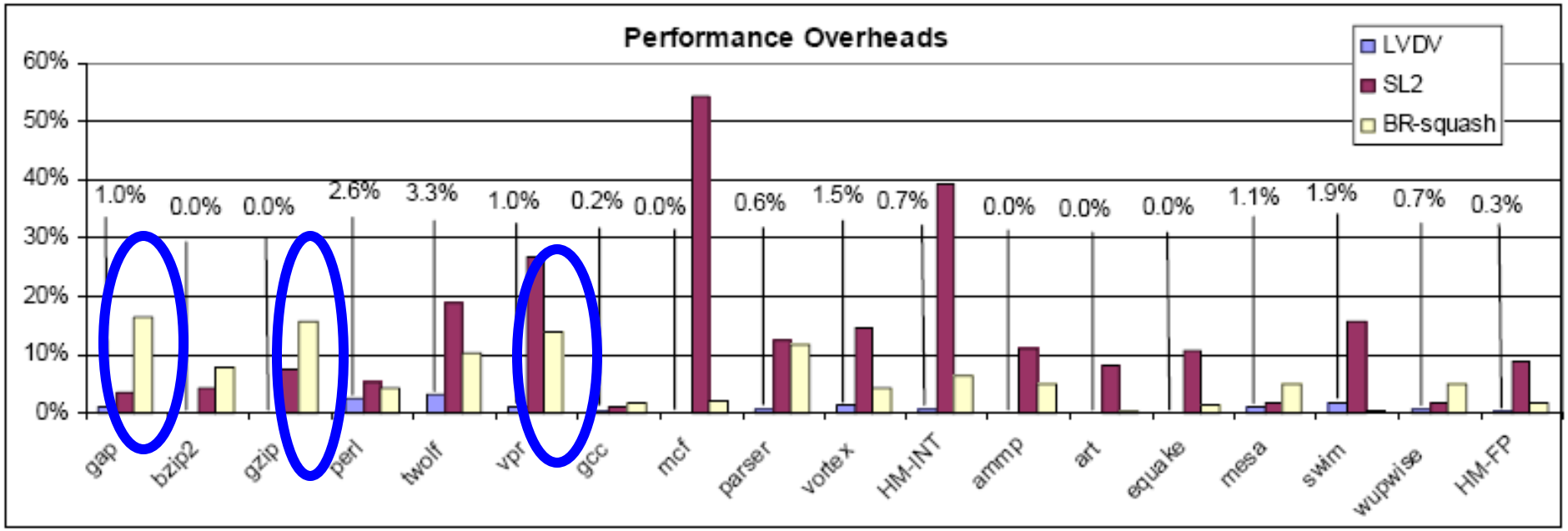
# Experimental Results



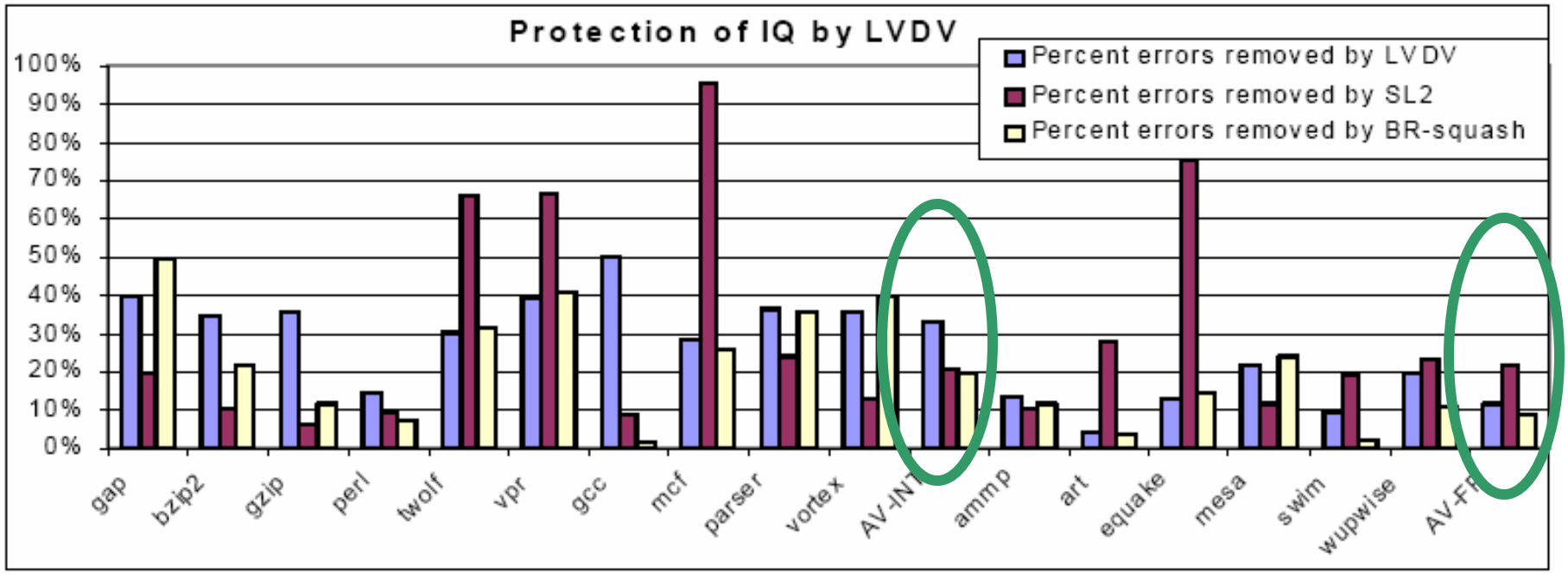
# Experimental Results
# Percent Performance Slowdown



# Experimental Results
# Percent Performance Slowdown

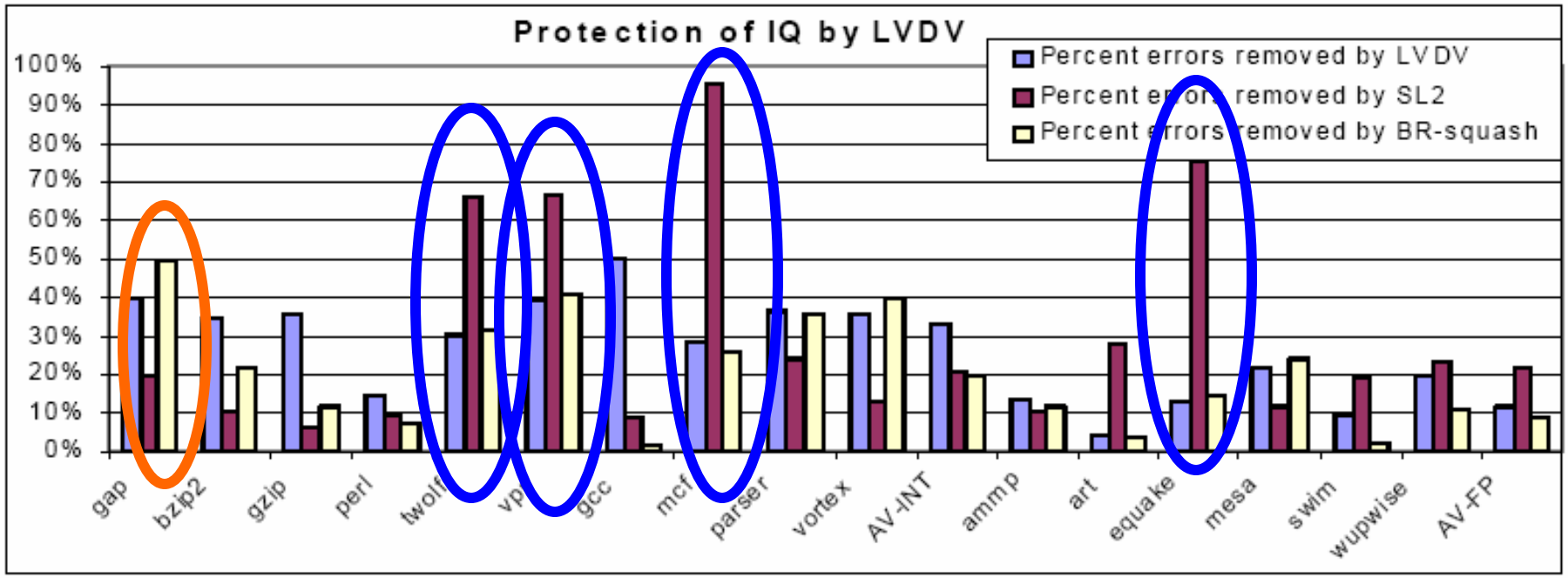


# Experimental Results



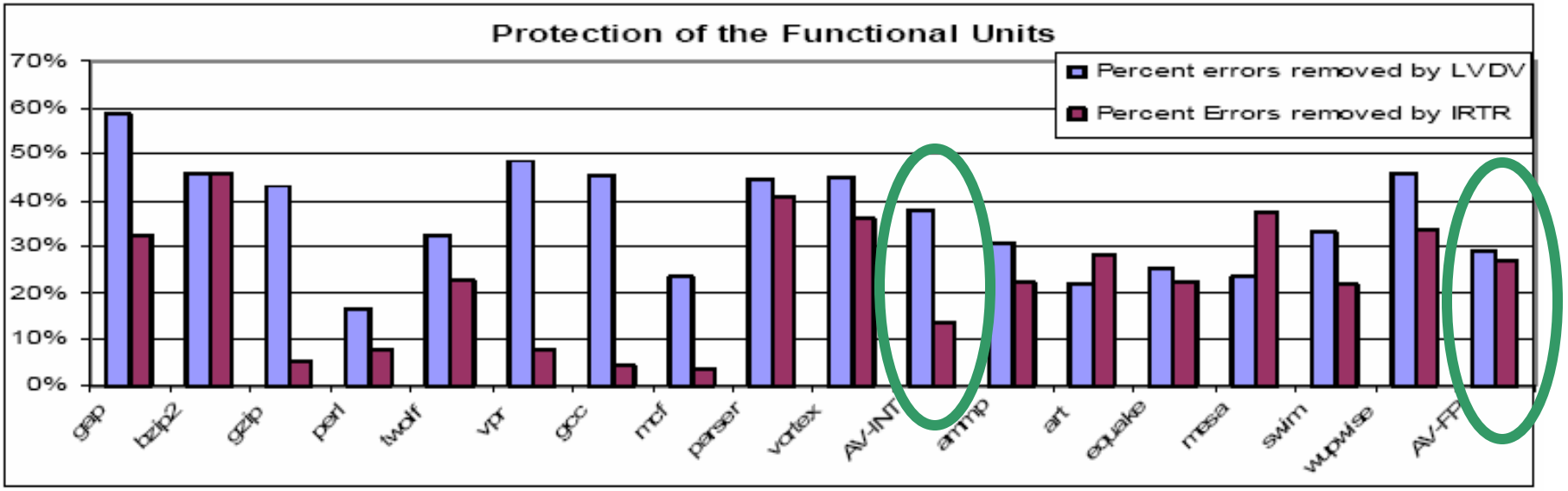

- LVDV increases the average mean time to failure MTTF of the IQ by 41% for integer benchmarks

# Experimental Results




- SL2 and BR-squash can be very effective for some benchmarks

# Experimental Results




- LVDV increases the average mean time to failure of the FUs by 61%

# Conclusions And Future Work

- A single hardware structure can provide protection to multiple logic units
- Very limited performance overhead: up to 3.3% and 0.7% on average
- Improves the MTTF of the IQ by 41% and the FUs by 61% on average
- Focus on exploiting other localities for information redundancy