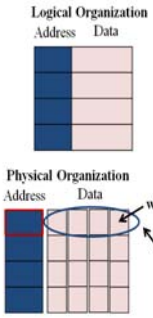
Implicit-Storing and Redundant-Encoding-of-Attribute Information in Error-Correction-Codes

Yiannakis Sazeides[1], Emre Ozer[2], Danny Kershaw[3], Panagiota Nikolaou[1],
Marios Kleanthous[1], Jaume Abella[4]
[1]University of Cyprus, [2]ARM, [3]NXP, [4]Barcelona Supercomputing Center
cs05np1@cs.ucy.ac.cy
www.cs.ucy.ac.cy/carch/xi

MICRO-46
MICRO 2013
MICRO 46, Davis, California, December 9th 2013

Memory Organization



Logical Memory Model:
table of addresses and data

Physical Memory Organization:
- design constraints/locality
- multi-level hierarchy
- arrays with multiple blocks
- each block with many words

Extra Information in Memory

Extra information in memory locations useful for:

Reliability,	ECC bits
Availability,	Poison Bits
Security,	Taint Bits
Performance,	Full/Empty Bits
Energy,	Sublock Dirty bits

Extra overheads: area, energy, delay ☹

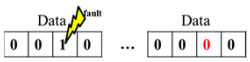
Our Proposition

Implicit Storing:
- encode in ECC extra information
- ☺ extend the logical capacity of a memory array without increasing its physical capacity
- ☺ minimal area and energy overheads
- ☺ low performance overhead
- ☹ reduction in the code strength

Redundant Encoding of Attributes:
- redundantly encode the same attributes in multiple codewords
- ☺ minimize the reduction of the code strength
- ☺ minimal impact on performance

Background

- **Error:**



- **Erasure:** a specific bit position of the data with unknown value

- **Shortened codes:** number of protected bits is smaller than max number that can be protected
  - k check bits can protect for p bits.
  - SECDED, for k=8

$p = 2^{k-1} - k$ extra bits ➔ p=120 bits
Data 64 bits ➔ Can protect 56 bits

Implicit-Storing (IS)

Store Extra Information without overheads

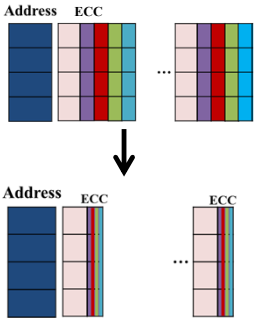
Basic Idea
Extend the logical capacity of a memory array without increasing its physical capacity

HOW
Encode the extra information in the ECC

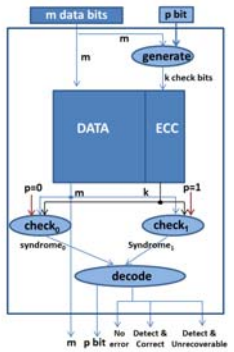
Using shortened codes

Extra information is erased

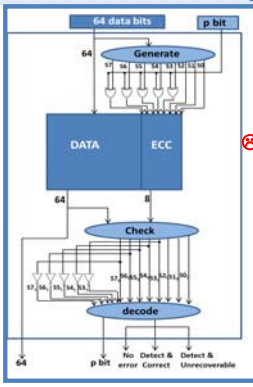
Infers the extra information on reads:
Multiple decodings use codeword with fewer errors



Implementation of 1-bit Implicit-Storing



Overhead of Implicit Storing



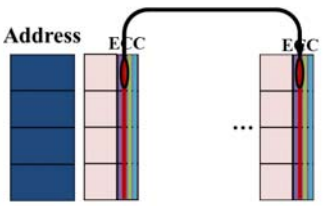
☺ Extra hardware
5 XOR gates
and
5 Inverters

☹ Reduce code strength
Miss-correct some uncorrectable data errors

Savings in Area and Energy
Cache:3.75%
DRAM:11.11%

Redundant Encoding of Attributes

Encode the same attribute in multiple codewords of a block
Read multiple correlated locations and produce their codewords
The decoder uses many codewords to determine data and implict bit
When to use REA: when an error is detected



IS & REA (IREA)

Implicit Storing

Missing bit value is X	Number of Actual Errors			
	0	1	2	
Syndrome from checker that assumes missing Bit is X	No error	Detect odd & Correct	Detect even & Unrecoverable	Detect even & Unrecoverable
Syndrome from checker that assumes missing bit is X'	Detect odd & Correct	Detect even & Unrecoverable	Detect odd & Unrecoverable	Detect odd & Correct
Decoder Decision	No error	Detect odd & Correct	Detect even & Unrecoverable	Detect odd & Correct
Correct Decision	YES	YES	YES	NO (miss-correction)

IS & REA (IREA)

	Data Errors Word 0/Word1																	
Syndrome		0/0	0/1	0/2		1/0	1/1	1/2		2/0		2/1		2/2				
C_0	X	No Error	No Error	No Error	No Error	Odd	Odd	Odd	Odd	Even	Even	Even	Even	Even	Even	Even	Even	
	X'	Odd	Odd	Odd	Odd	Even	Even	Even	Even	Odd3	Odd	Odd3	Odd	Odd3	Odd3	Odd	Odd	
C_1	X	No Error	Odd	Even	Even	No Error	Odd	Even	Even	No Error	No Error	Odd	Odd	Even	Even	Even	Even	
	X'	Odd	Even	Odd3	Odd	Odd	Even	Odd3	Odd	Odd	Odd	Even	Even	Odd3	Odd	Odd3	Odd	
Infer p value		X	X	X	X	X	X	X	?	X	X	X	?	X	X	X	X'	
Global action		NN	NC	NU	NU	CN	CC	CU	U?	UN	UN	UC	U?	UU	UU	UU	CC	