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

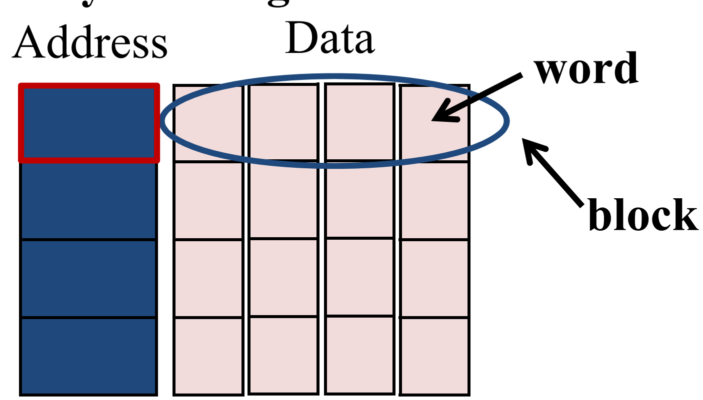
MICRO 46, Davis, California, December 9th 2013

Logical and Physical Memory Organization

- Logical Organization (programming model): a table with addresses and data
- Physical Organization (manufacturing, cost, performance):
  - multi level hierarchy of arrays (DRAM, cache etc)
  - an array consist of multiple blocks each with a unique address
  - each block with many words

Logical Organization

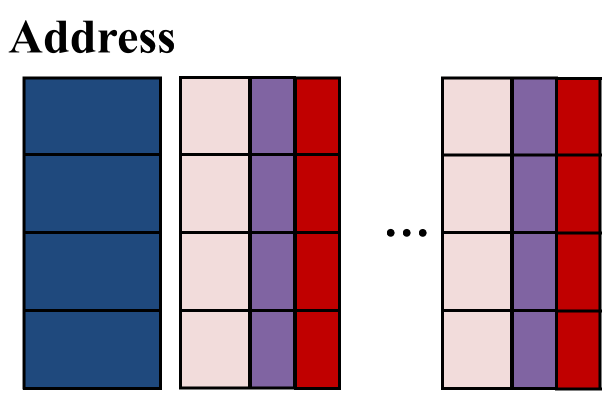
Physical Organization



Reliability Implications on the Memory Organization

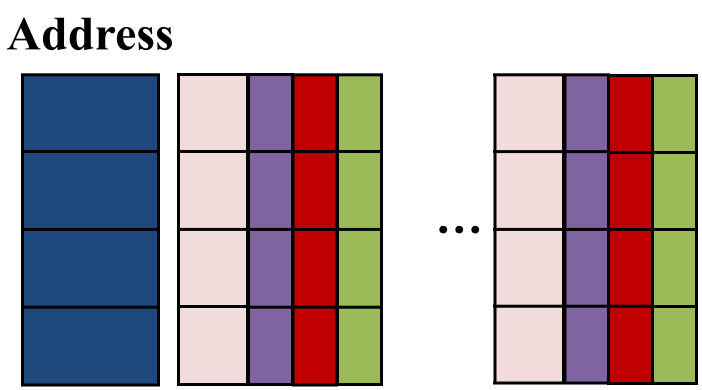
- **Protect data from faults**
  - add ECC code to detect and correct errors **[Hamming 1950]**

- **Increase availability**
  - add Poison bit to minimize failures from uncorrectable errors **[Weaver 2004]**
  - propagate dependence to data with uncorrectable errors



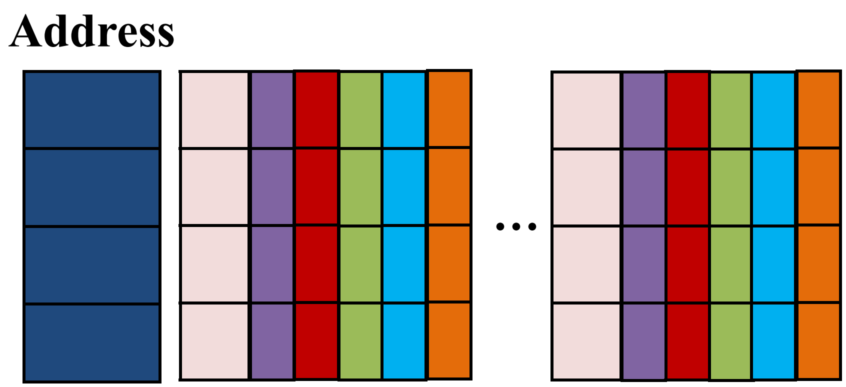
Security Implications on the Memory Organization

- **Prevent malicious attacks**
  - Track dynamically dependence to input data with taint bits **[Suh 2004]**



Performance and Energy Implications on the Memory Organization

- **Performance and energy benefits**
  - Track the dirty status of sub-block with extra bits **[Wang 2009]**
  - Full-Empty bits **[Smith 1981]**
  - Tagged Memory **[Gumpertz 1983]**
  - …

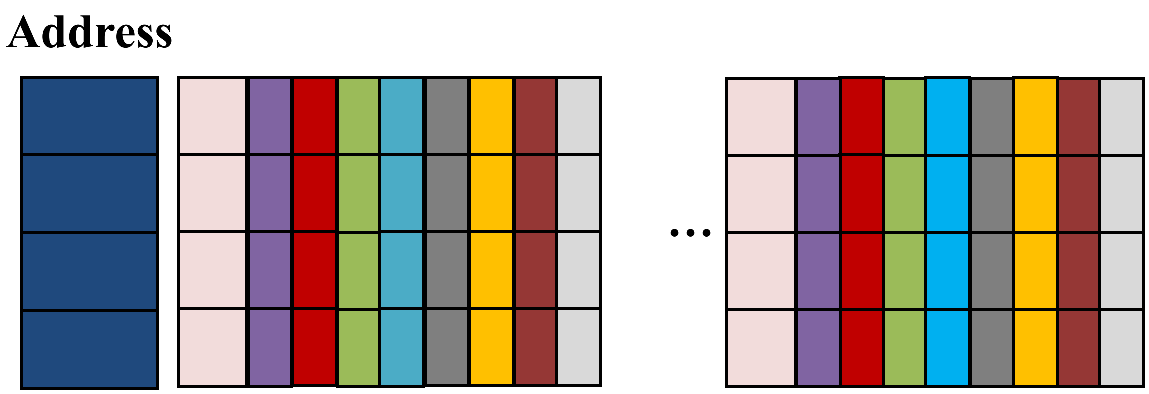


What we need!

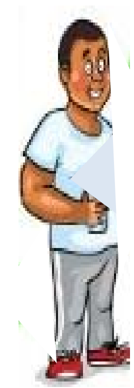
- Extra information in memory arrays for reliability, availability, security, performance, energy, …

But:

- more area overheads
- slower memory
- consumes more energy



What we propose!!



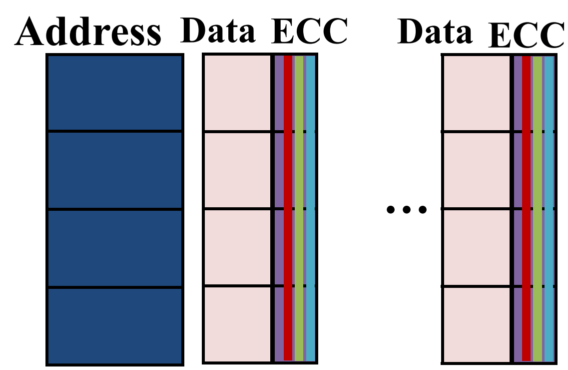
1. Implicit storing (IS)

- Do not store the extra information in the array
- Encode the extra information in the ECC codes

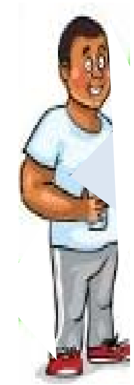
☺ Cost-effective, minimal impact on:

- Area
- Energy
- Performance

☹ Weakens strength of ECC for data



What we propose!!



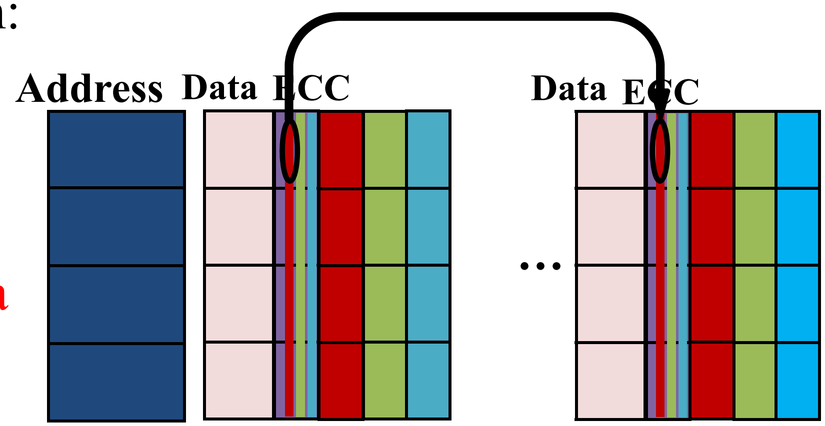
1. Implicit storing (IS)

- Do not store the extra information in the array
- Encode the extra information in the ECC codes

☺ Cost-effective, minimal impact on:

  - Area
  - Energy
  - Performance

☹ Weakens strength of ECC for data



2. Redundant Encoding of Attribute Information (REA)

- Encode the same information in multiple codewords of a block

☺ Recovers some ECC code strength lost due to IS

Outline

- Background
- Implicit Storing (IS)
- Redundant- Encoding-of-Attributes (REA)
- IS with REA
- Conclusions

Error correction codes

- Detection and Correction capability
- Shortened codes:
  - The number of protected bits is smaller than the maximum number that can be protected
  - e.g. SECDED code

    single error correction, double error detection

    k check bits can provide protection for p bits as long as:

    $$p \leq 2^{k-1} - k$$

    for k=8 bits ➔ maximum p=120 bits

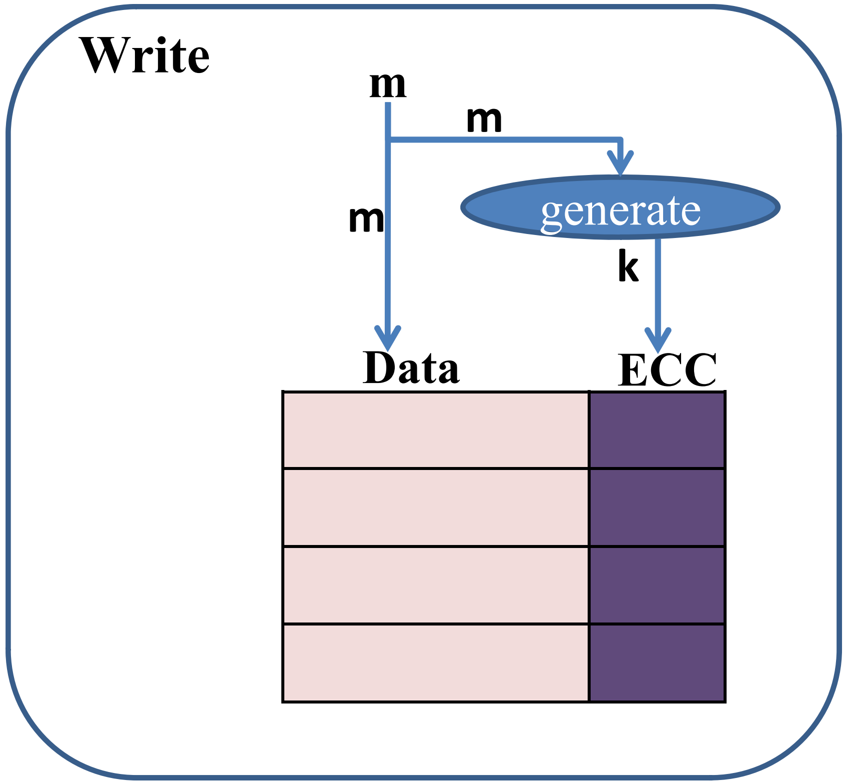
    If protected data is 64 bits ➔ code can protect 56 extra bits

Protecting data from errors

How it works:

Write:

- Generate ECC bits(k) from data bits (m)
- Store data and ECC bits in the array

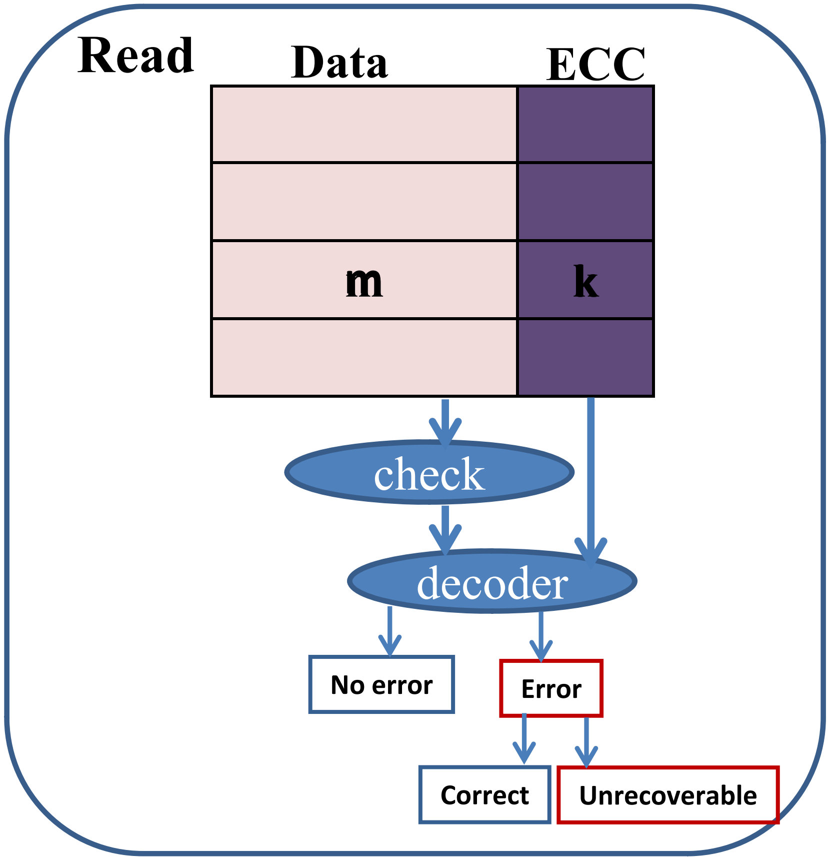


Protecting data from errors

How it works:

Read:

- Read data bits (m) and ECC bits (k) from the array
- Perform error checking
- The decoder indicates:
  - No error
  - Error:
    - Correctable
    - Uncorrectable

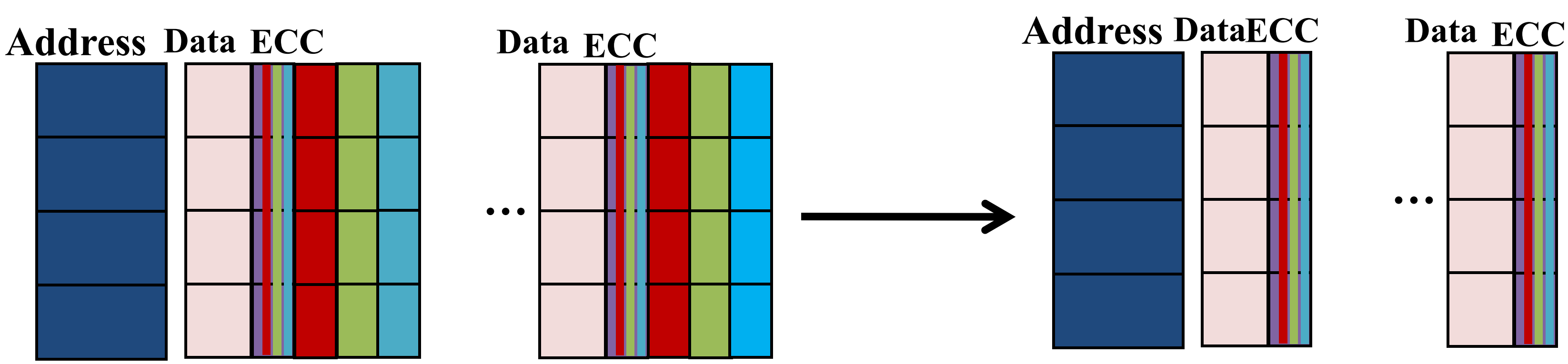


Outline

- Background
- Implicit Storing (IS)
- Redundant- Encoding-of-Attributes (REA)
- IS with REA
- Conclusions

Implicit-Storing (IS)

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



Implicit-Storing (IS)

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

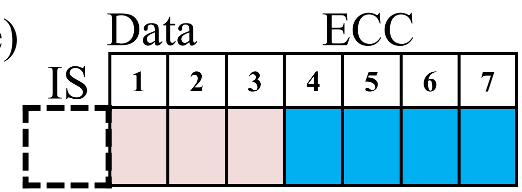
- **How:**
  - Do not save the extra information but encode it in the ECC
  - On writes, extra information is erased using erasure coding (**Data ECC**)
    - **Erasure:** a specific bit position of the data with an unknown value

  - On reads, the extra information is produced using erasure recovery

Example parameters

On a write

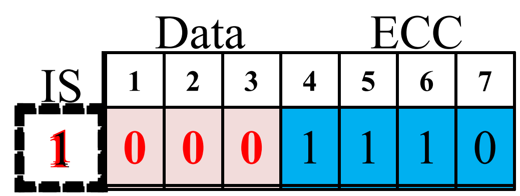
- Assume 3 bit data (1,2,3)
- Protected with 4 bit SECDED code (4,5,6,7)
  - Maximum number of protected bits is 4 (shortened code)
    - $p \leq 2^{k-1} - k$
    - Extra space for Implicit store → 1 bit (IS)



- Parity matrix that produce the ECC check bits [Hsiao 1970]:

	1	2	3	4	5	6	7
4	1	1	0	0	0	0	1
5	1	0	1	0	0	1	0
6	0	1	1	0	1	0	0
7	1	1	1	1	0	0	0

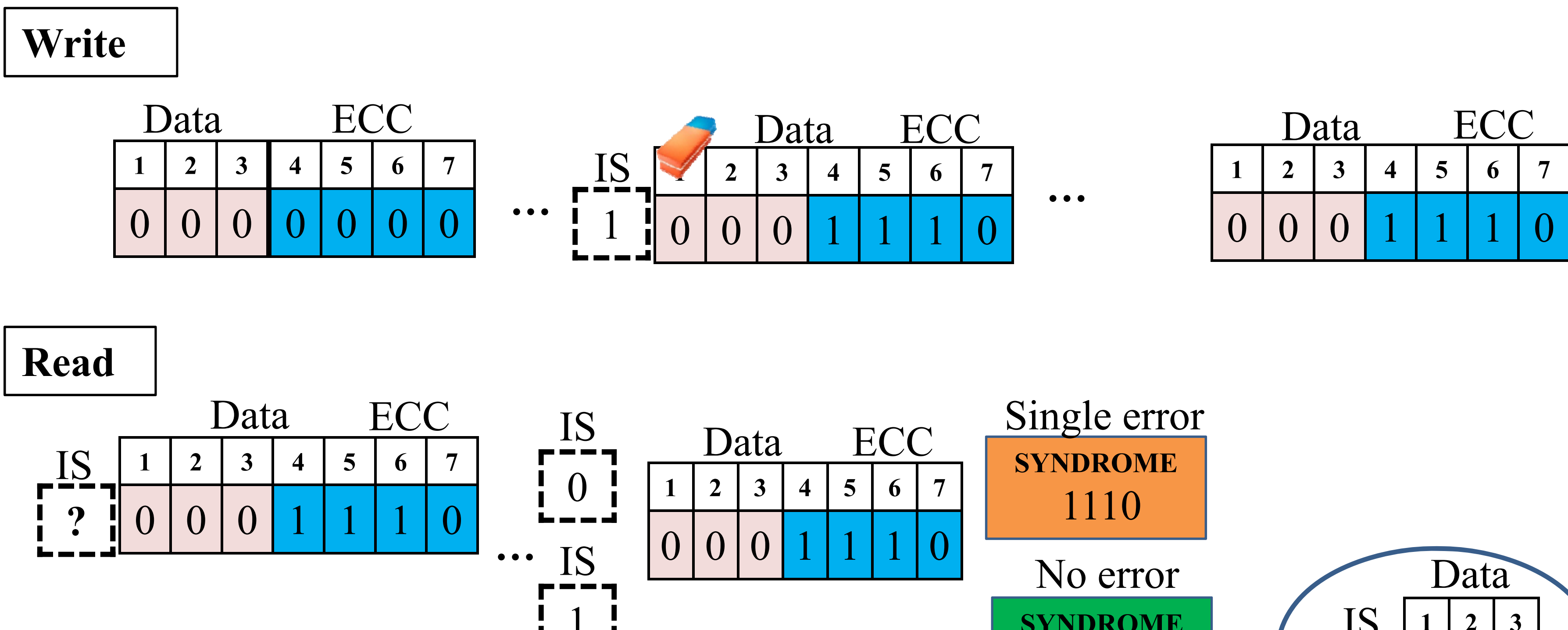
- **P4=1 ^ 2 ^ IS**
- **P5=1 ^ 3 ^ IS**
- **P6= 2 ^ 3 ^ IS**
- **P7=1 ^ 2 ^ 3**

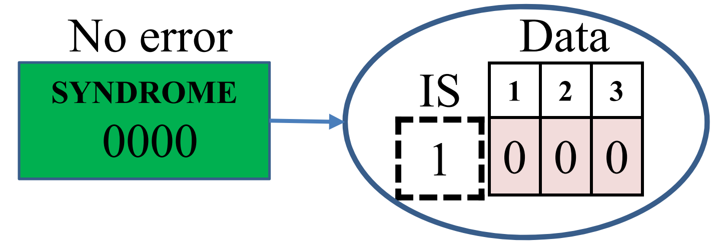


On a read

- A syndrome is produced:
  - Syndrome=Stored ECC ^ Produced ECC
  - Indicates the type of the error
  - Syndrome decoding based on the above parity matrix:
    - Zero Syndrome: No error
    - Odd Syndrome: Odd errors ≥1 → Single error correction
    - Even Syndrome: Even errors ≥2 → Uncorrectable

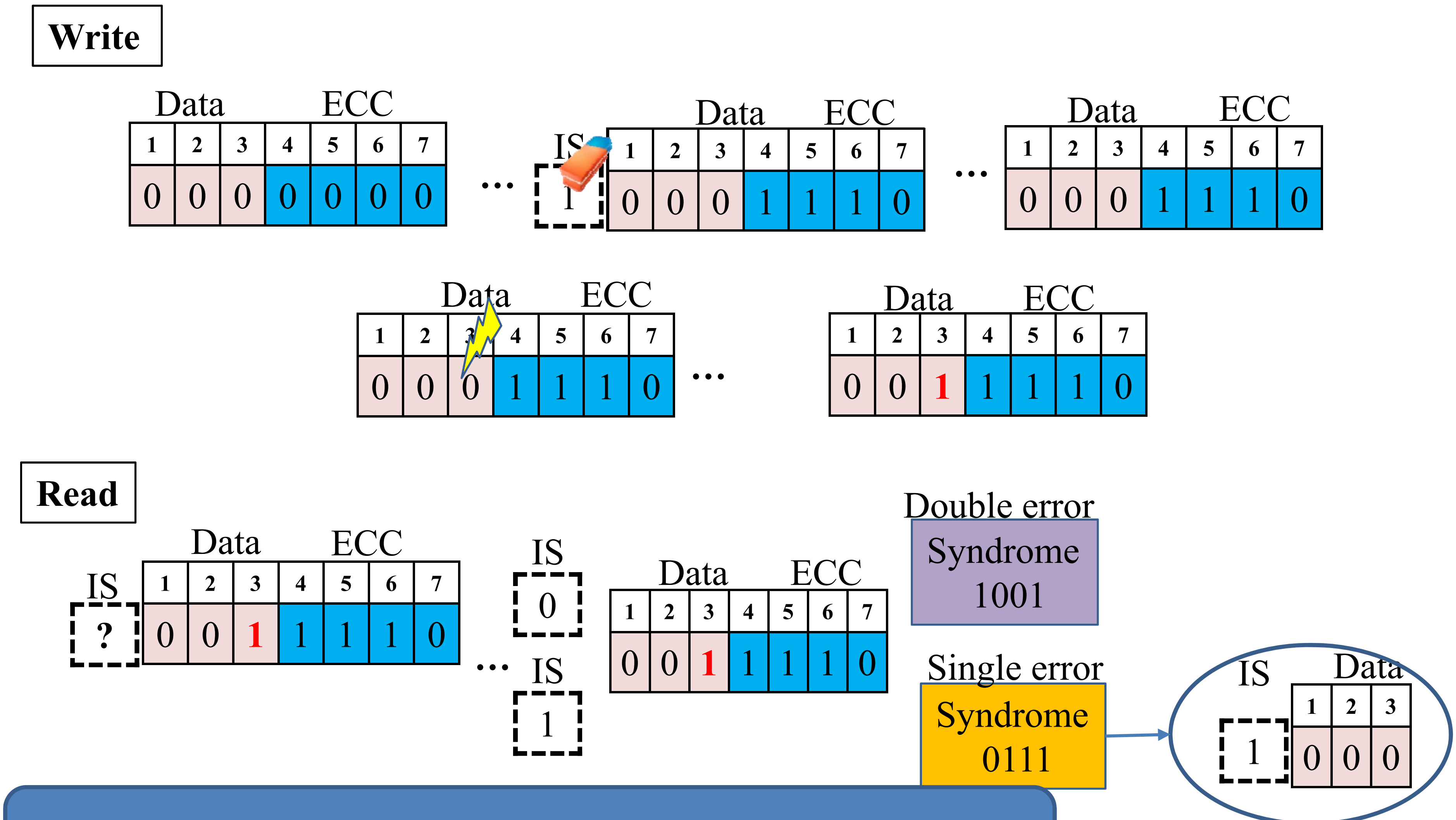
Example of 1 bit Implicit Storing (IS)





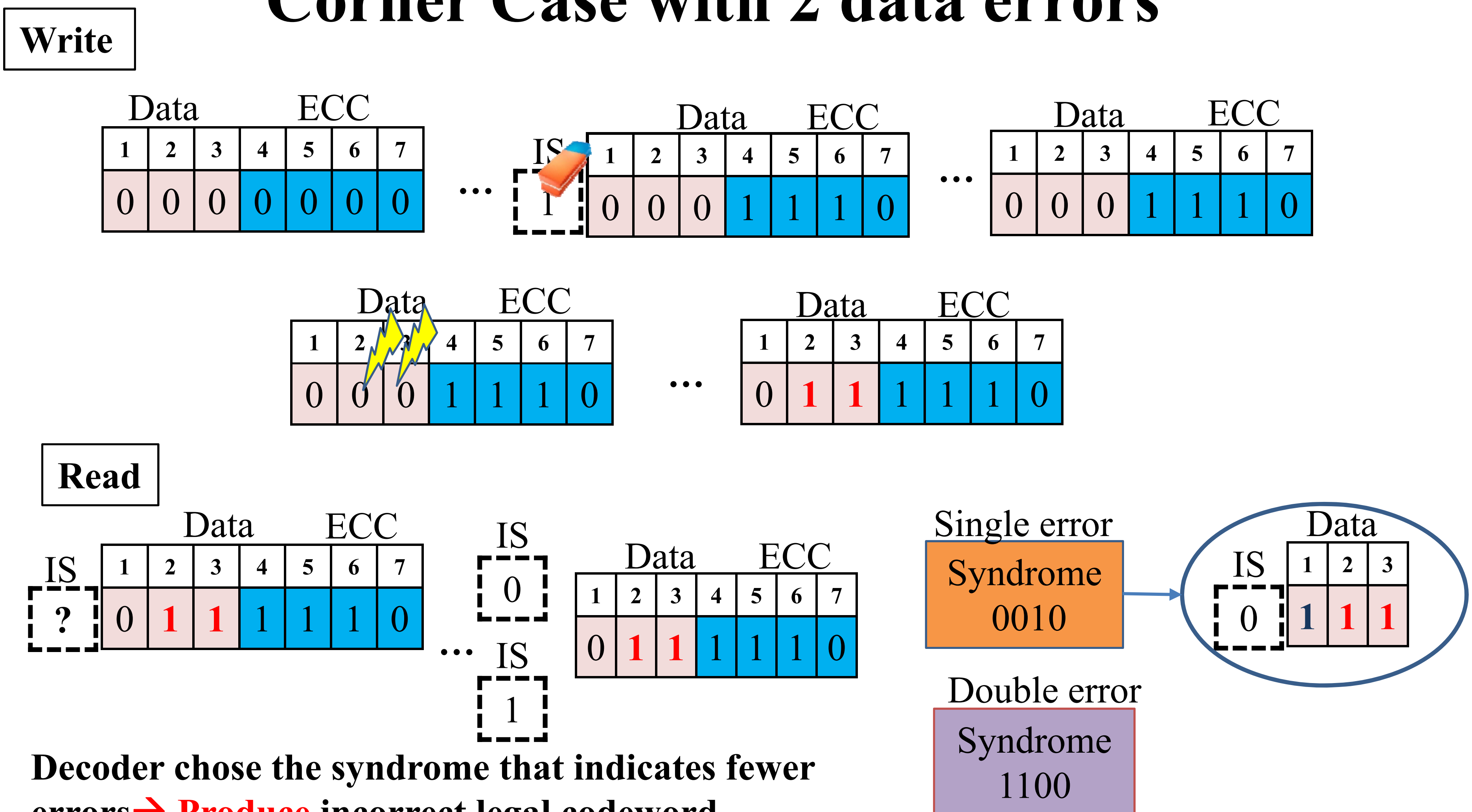
- **On a write:**
  - **Data and IS are encoded in the ECC code**
  - **Then IS erased**
- **On a read:**
  - **Produce the implicit bit with two decodings instead of one**
  - **One assumes IS=0 and the other assumes IS=1**
  - **Infer implicit bit from codeword with fewer errors**

Example of IS in the presence of data error



IS correct the error and infers correctly the IS bit

Corner Case with 2 data errors



- **Decoder chose the syndrome that indicates fewer errors→ Produce incorrect legal codeword**
- **Data are faulty and the IS is not the right ☹**

Without IS uncorrectable
With IS miscorrected data

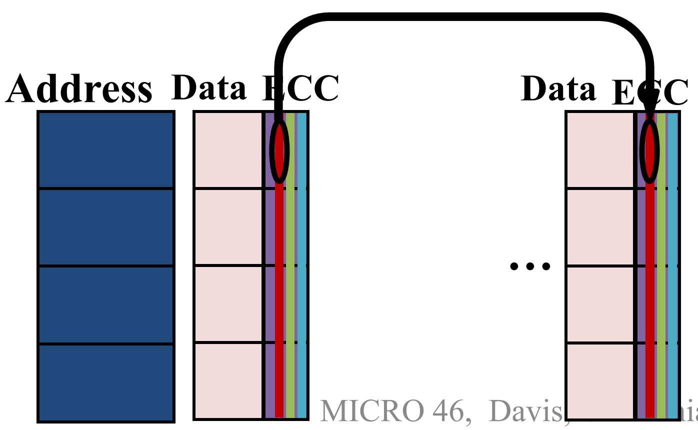
Outline

- Background
- Implicit Storing (IS)
- Redundant- Encoding-of-Attributes (REA)
- IS with REA
- Conclusions

Redundant Encoding of Attributes (REA)

- The granularity for ECC protection is often smaller than the granularity of block transfer
    - e.g. ECC code protects 64 bit data, and the block size is 512 bits

- On writes encode the same information in multiple codewords of a block
  – Correlated words: encode same attribute information



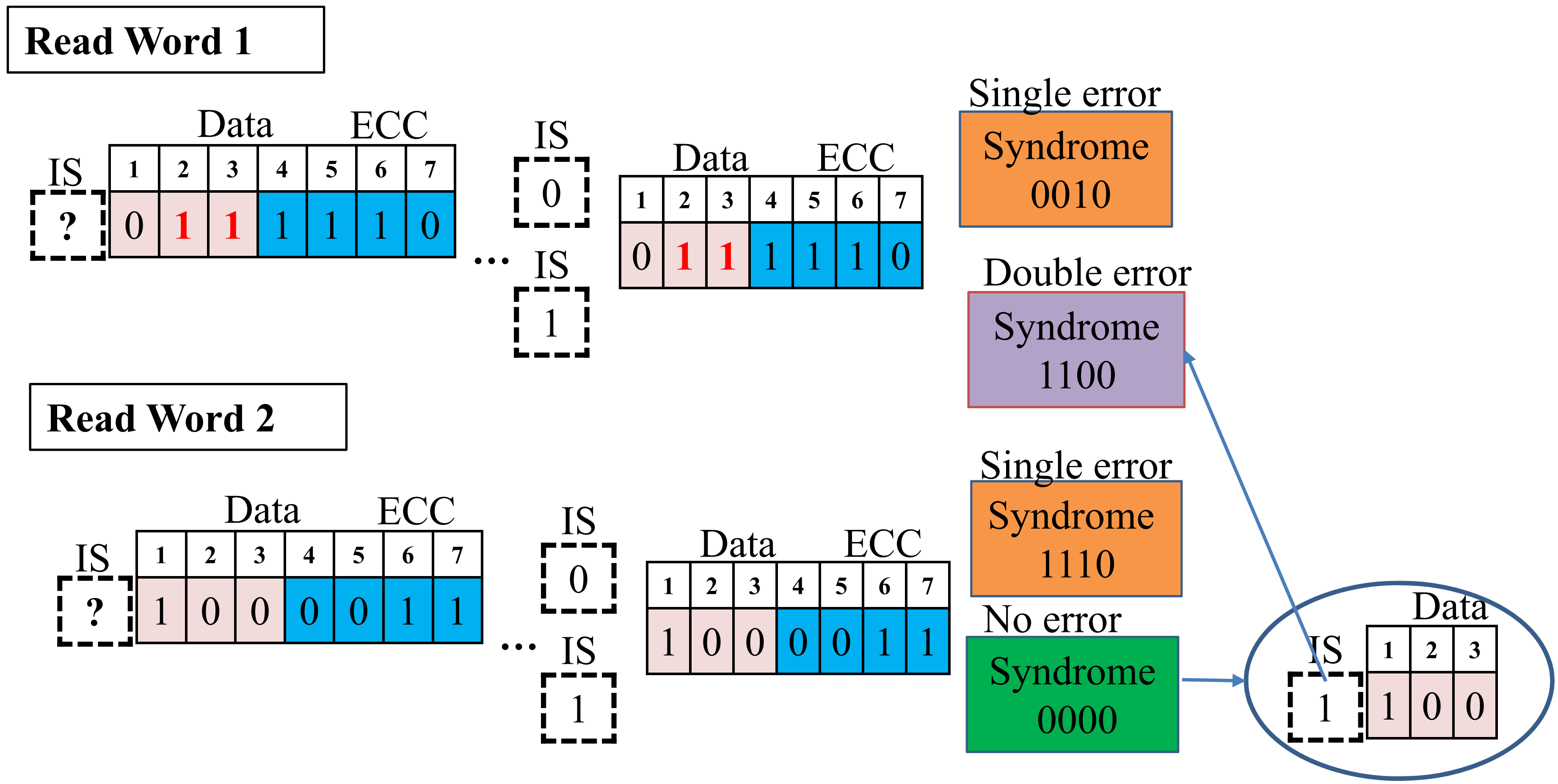
Redundant Encoding of Attributes (REA)

- The granularity for ECC protection is often smaller than the granularity of block transfer
    - e.g. ECC code protects 64 bit data, and the block size is 512 bits
- On writes encode the same information in multiple codewords of a block
    - Correlated words: encode same attribute information
- On reads when there is an error decode the correlated codewords to detect and correct the error

IS + REA=IREA

- How it works:
  - When one syndrome has no error: business as usual
  - Otherwise with errors in both syndromes
    - Read multiple correlated locations and produce their codewords
    - The decoder uses many codewords to determine data and implicit bit
- Changes:
  - Extend generate and check units to consider attributes
  - In case of an error need to read and generate syndromes of correlated locations
  - Need new decoder that uses correlated location codes as inputs to decide reaction

IREA: Example of a word with 2 data errors



With IS miscorrected data ☹
With IREA uncorrectable ☺

Some key design implications

- No changes in the SRAM macros and DIMMs
- Changes limited in the cache and memory controllers
- Required changes are minimal, handful of gates

What else discussed in the paper

- How Implicit Storing and Redundant Encoding of Attributes reacts in the presence of errors in correlated words
- Discuss Error Code Tagging (ECT) [Gumpertz 1983]
  - ECT useful for encoding attributes that are available at write and read time
  - Explain differences with IS
  - How to combine ECT + REA=EREA
- Temporal and Spatial reliability analysis for single bit transient errors
- Discuss performance overheads of IREA and EREA
- Discuss selective use of IS and REA
- Area, Delay and Scalability analysis

Summary and Conclusions (1)

- Many techniques to improve performance, reliability, availability, security, energy rely on extra information stored in memory
- Propose: Implicit Storing and Redundant Encoding of Attributes
- Implicit Storing: extend the logical capacity of a memory array without increasing its physical capacity
- Save extra information
  - without area and energy overheads
  - with minimal performance impact
- IS causes reduction in the code strength

Summary and Conclusions (2)

- Redundant encoding of Attributes: redundantly encode the same attributes in multiple codewords
- REA can minimize the reduction of the code strength
- Applicable to both IS and ECT
- Minimal impact on performance
- Future work: Applications and detailed analysis of correlated errors

Acknowledgments

- “Eurocloud” and “Harpa” FP7 Projects
- HIPEAC FP7 Network of Excellence
- Spanish Ministry of Science and Innovation

Thanks!