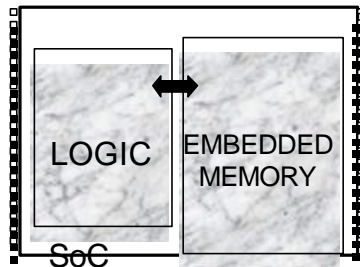


Outline

- Memories today
- Fault Model
- MARCH algorithms

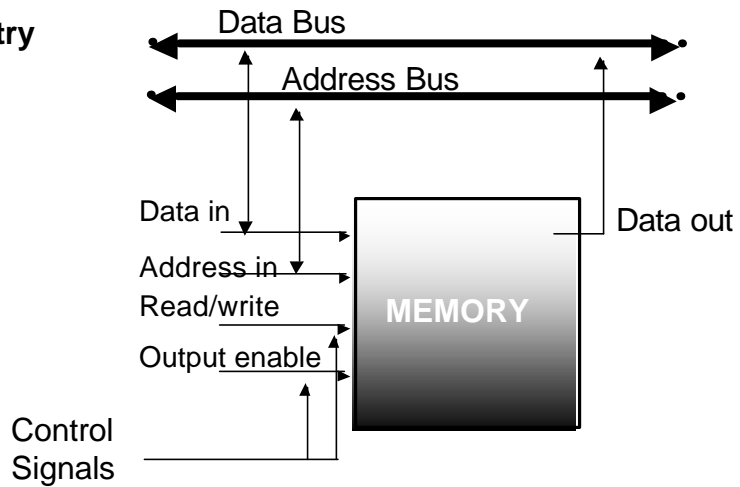
Memory Testing

- Memory is the most dense physical structure
 - Embedded memories begin to dominate physical die area vs. logic
 - Memory arrays can be doubly embedded (ex: microprocessor with cache within a larger chip design)
- Considering the increase density, memory arrays are more sensitive to defects



Memory Organization

- Address Decoder
- Data Decoder
- Control Circuitry

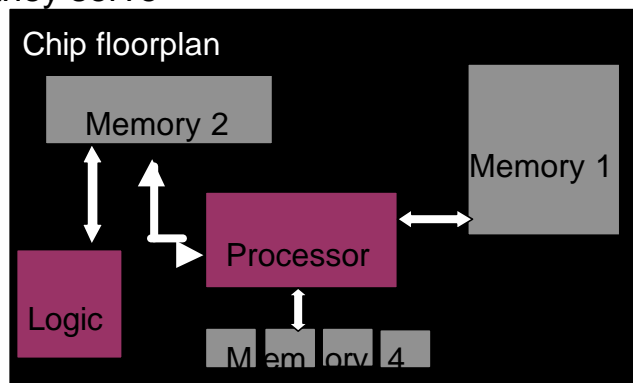


ECE 1767

University of Toronto

Memory Design Concerns

- **Aspect ratio:** square, rectangle, in pieces, number of memory modules. It affects routing, chip area, power dissipation etc
- Recent trend is to have many distributed memory arrays close to the logic they serve

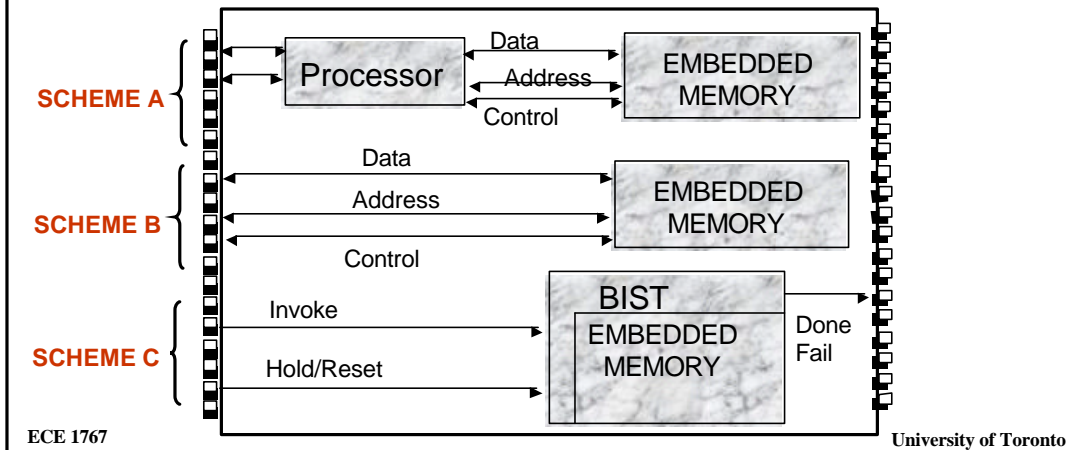


ECE 1767

University of Toronto

Embedded Memory Testing Methods

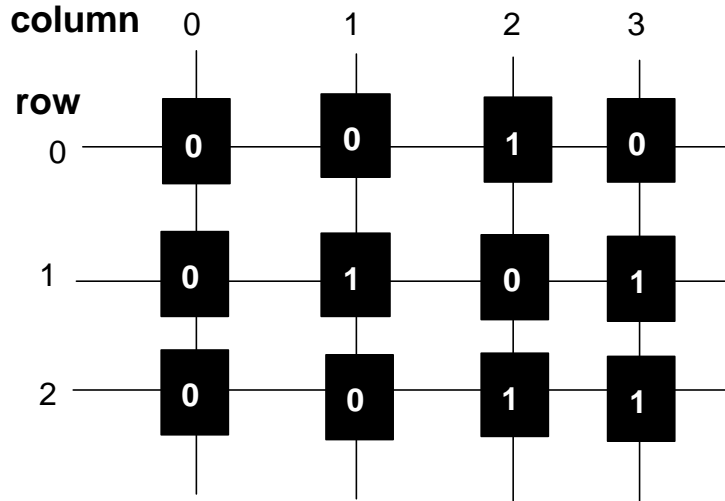
- ◆ **Scheme A: Embedded Microprocessor Access**
- ◆ **Scheme B: Direct Memory Access**
- ◆ **Scheme C: Memory Build-in Self-Test**



Embedded Memory Testing Methods

- **SCHEME A:** uses embedded microprocessor to test memory. These vectors consume “test vector memory”
- **SCHEME B:** accesses memory directly via I/O package pins. Signal pins needs to be reserved/borrowed and it requires a route-intensive bus structure to different memories
- **SCHEME C:** includes tester function right into the silicon. Little chip interface but once BIST is build it cannot be changed. BIST mechanism can be shared between different memory modules. BIST can operate memory at-speed.

Memory Layout



Failure Modes and Fault Model

- **Failure modes:**

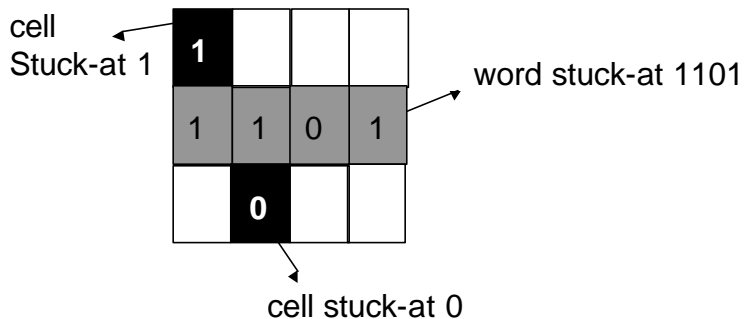
- ▲ Data Storage
- ▲ Data Delivery
- ▲ Data Recovery
- ▲ Address Recovery
- ▲ Data Retention
- ▲ Data Decode
- ▲ Address Delivery
- ▲ Address Decode

- **Fault models:**

- Stuck at faults
- Bridges

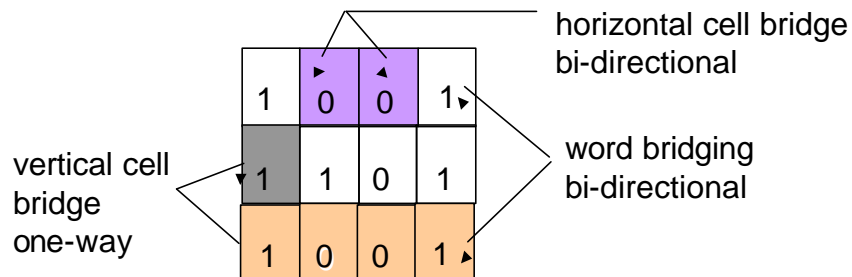
Stuck-at Fault Model

- We can have a single cell or a group of cells stuck to some constant value “0” or “1”
 - ◆ **Fault exercise and detection:** Write both 0 and 1 and read 0 and 1, respectively



Bridging Fault Model

- Bridges can be 0 Ohm or resistive or diodic. They can cause change in one or multiple cells
 - ◆ **Fault exercise and detection:** write and read alternating or complementing patterns such as 0-F, 5-A etc. Diodic bridges may be sensitive to address writing order

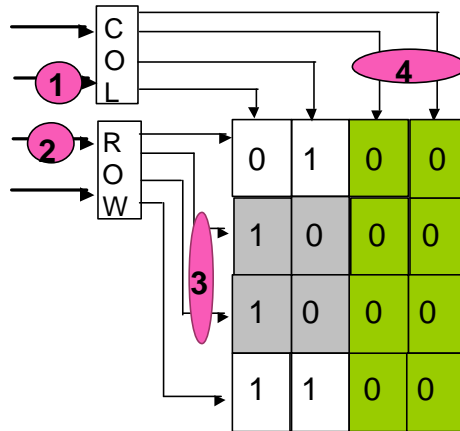


Decode Fault Model

- Stuck-at and bridge faults can cause to always choose wrong address, select multiple address etc

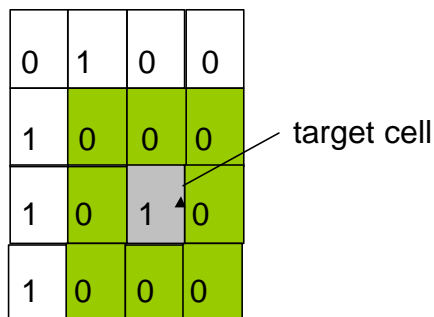
1, 2: stuck at faults result choosing always a wrong address

3,4: bridges result always choosing multiple columns and rows



Data Retention Fault

- Leakage or bridging may cause the stored data value to degrade over time
 - Fault exercise and detection:** put data in target memory cell and surround with complementary values



Algorithmic Test Generation

- **Observation:** memory testing needs certain pattern sequences to exercise and detect the different fault failure modes
- **Algorithmic Test Generation:** research area that tries to identify such pattern sequences for memory testing
 - ◆ Usually called MARCH tests
 - ◆ We will examine MARCH tests for a bit-slice memory (sake of simplicity)

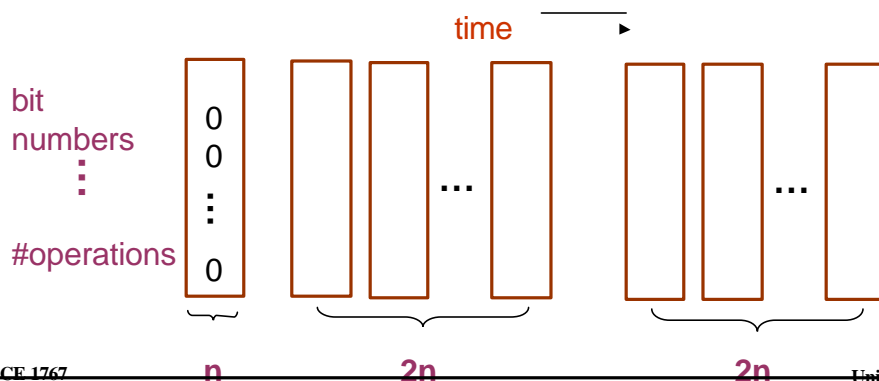
5N MARCH TEST

Consider following sequence of operations:

Address(0) → Address(MAX): WRITE(0)

Address(0) → Address(MAX): READ(0) WRITE(1)

Address(MAX) → Address(0): READ(1) WRITE(0)



Algorithmic Test Generation

- **5N MARCH** detects:
 - ▲ Single stuck-at faults
 - ▲ 1-step neighbor bridge faults
 - ▲ Address decode faults
- Consider **14N MARCH**:
 - Address(0) → Address(MAX): WRITE(5)
 - Address(0) → Address(MAX): READ(5) WRITE(A) READ(A)
 - Address(0) → Address(MAX): READ(A) WRITE(5) READ(5)
 - Address(MAX) → Address(0): READ(5) WRITE(A) READ(A)
 - Address(MAX) → Address(0): READ(A) WRITE(5) READ(5)
 - Address(MAX) → Address(0): READ(5)
- ◆ **Homework:** detects stuck-at, bridges, address decode, data decode and access time faults (why?)