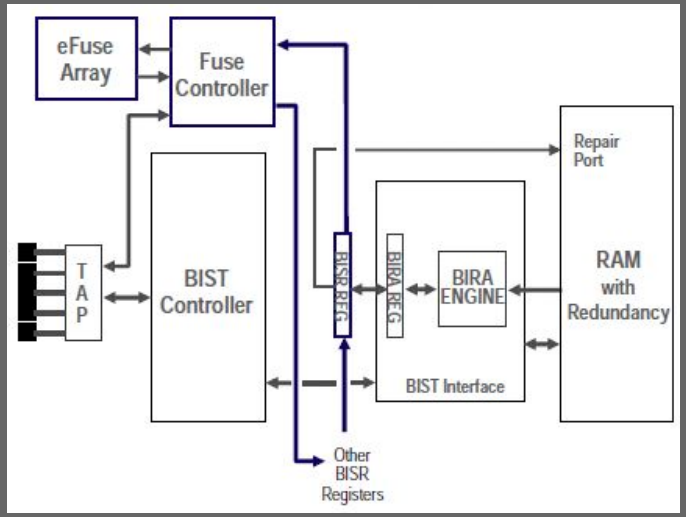
Memory Repair

How Real Chips use Redundancy

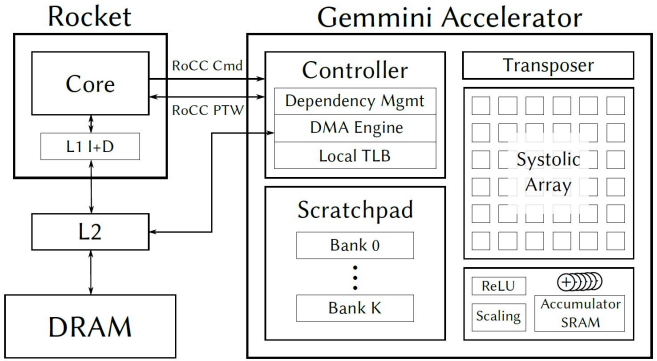


Preface

Being in silicon academia has a very different set of objectives than the silicon industry!

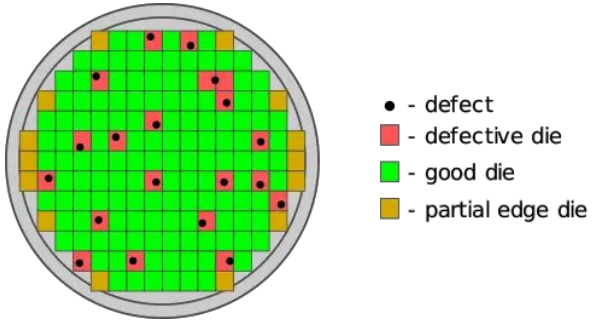
Academia often Prioritizes:

- Novelty
- Experimentation
- Performance
- Simplicity



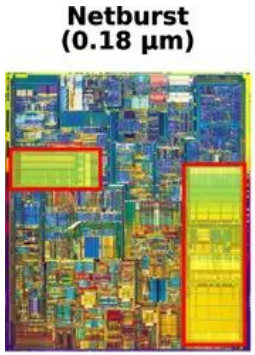
Industry often Prioritizes:

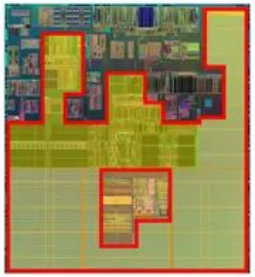
- Yield
- Cost
- Reliability
- Bring Up Time

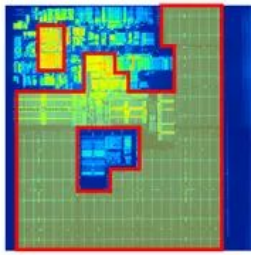


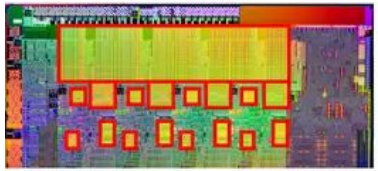
SRAM

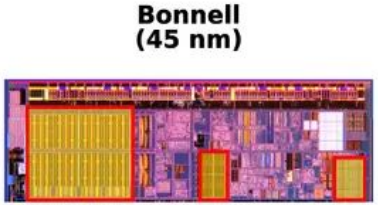
- Modern chips are mostly SRAM
- SRAM cells are tiny!
  - High Density :)
  - Error Susceptible :(
- Statistically likely to see bit errors

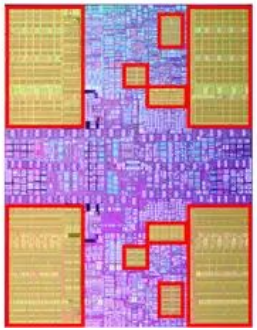


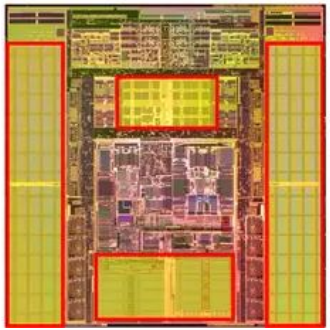






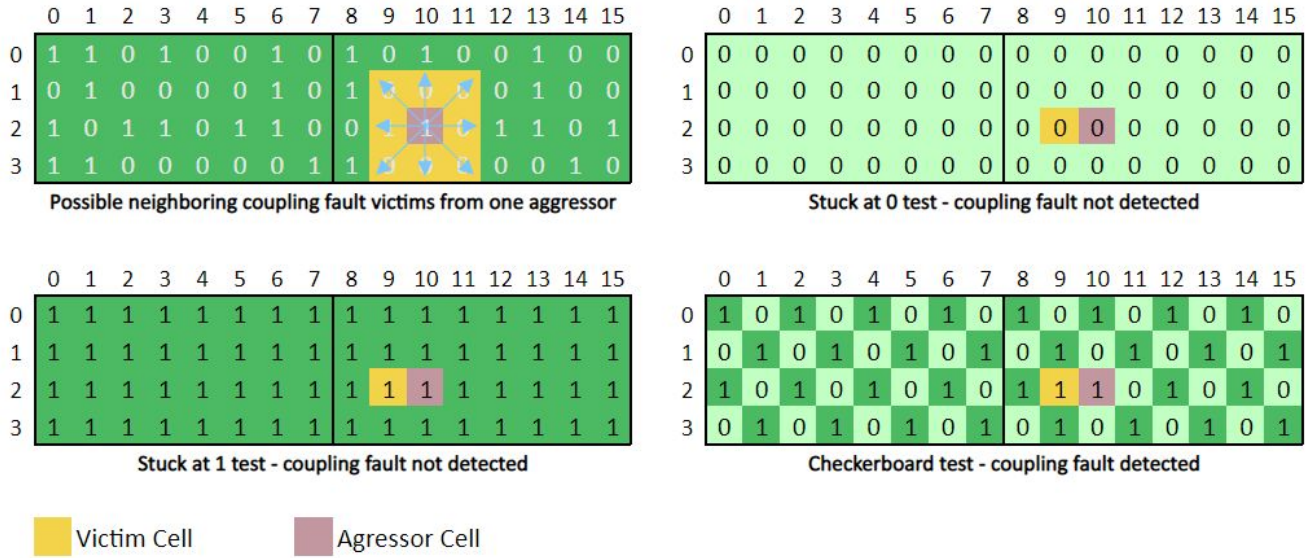






SRAM Failures

- **Stuck-at** faults: stuck at 1 or 0
- **Coupling** faults: writing one impacts another

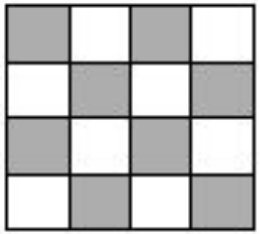


Finding Failures

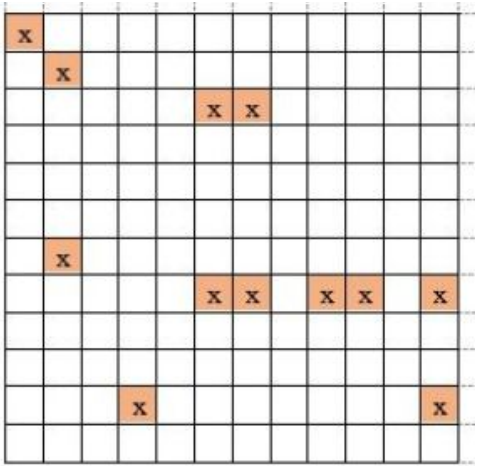
Glossary

BIST: Built In Self Test

- How can we diagnose failures in an SRAM?
  - **BIST**
- Use state machine to write and readback checkerboard pattern
  - Read nearby cells to ensure no coupling
- Find and record any bits that are broken
  - Tells us if SRAM works



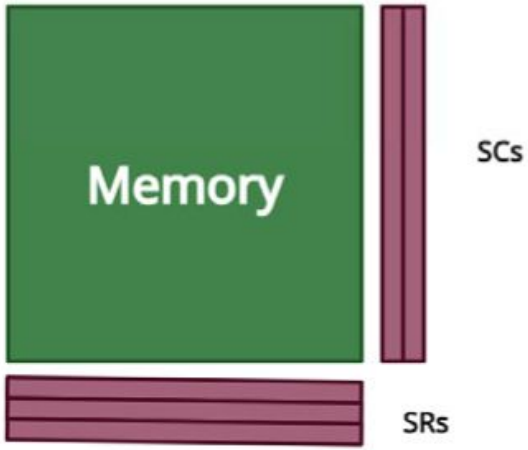
Write		Read	
0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

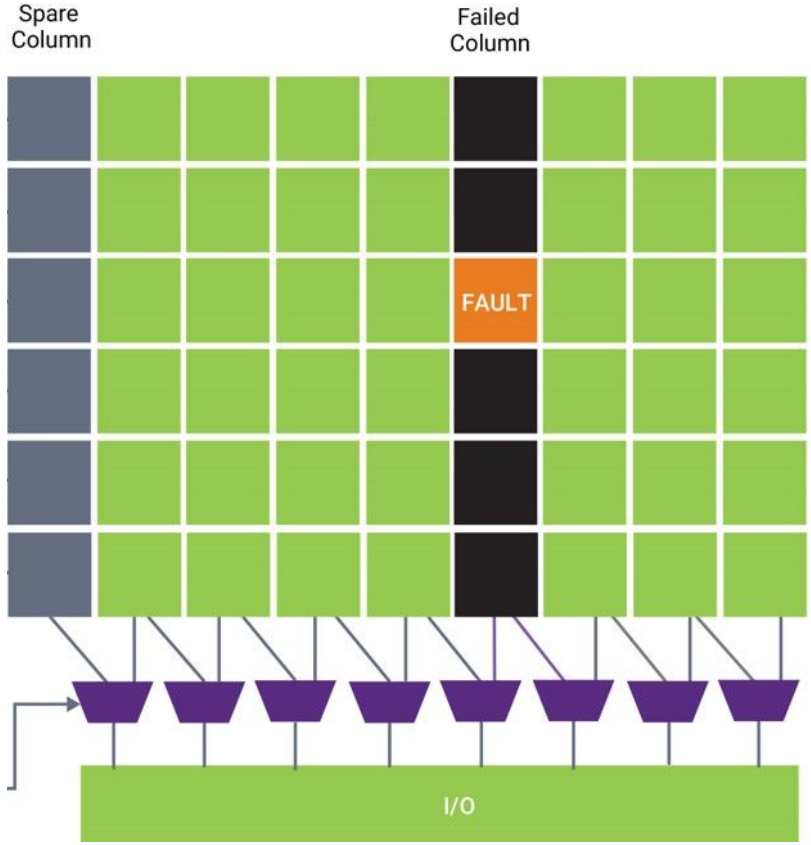


Redundancy

- Finding failures is great, but can we fix them?

- Redundant Rows and Columns
  - Columns: more bits, less decoding logic
  - Rows: less bits needed, changes addressing



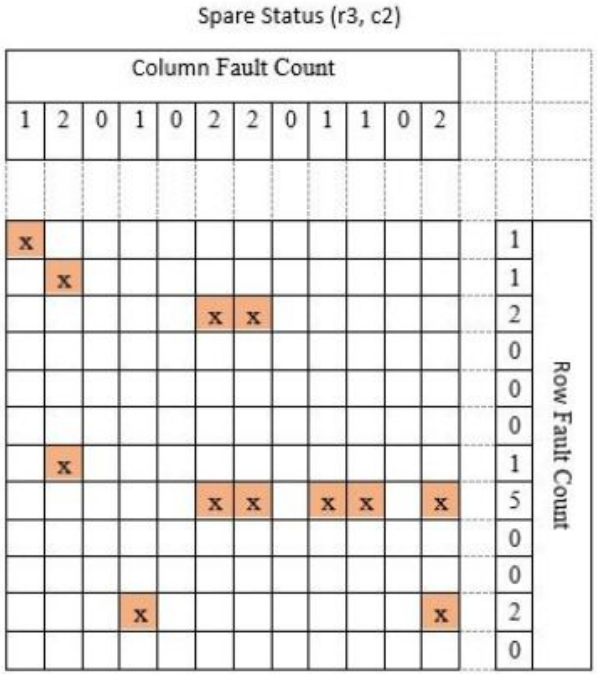


Redundancy Analysis

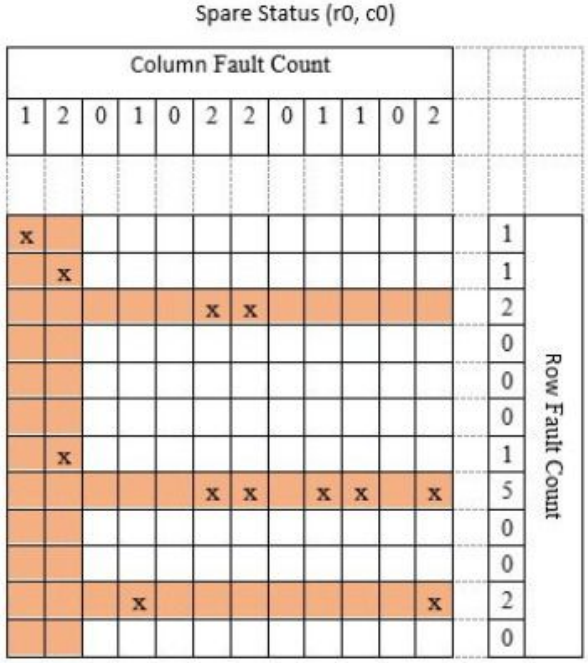
Glossary

BIRA: Built In Redundancy Analysis

- Given # of spare rows and cols, how can we cover the most faults?
- **BIRA** algorithmically assigns spare rows and columns
- Figures out if errors are recoverable, and if so how



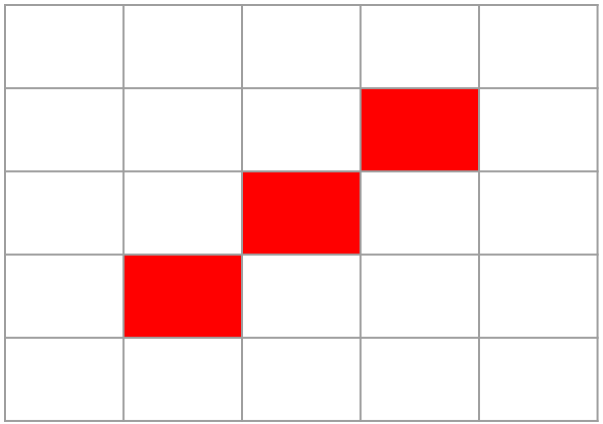
(a) Fault Present in the Memory

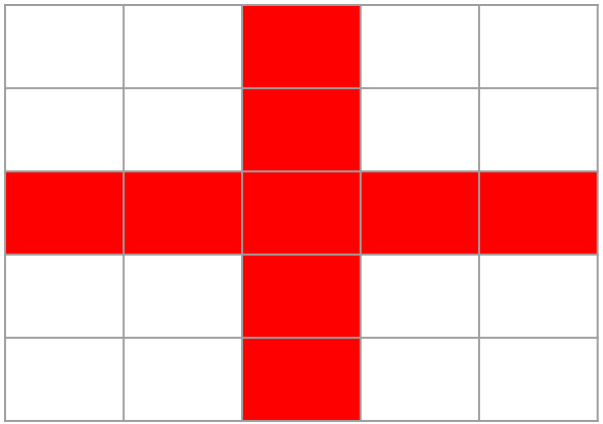


(b) Memory Repaired

Recoverability

With one row and one column:



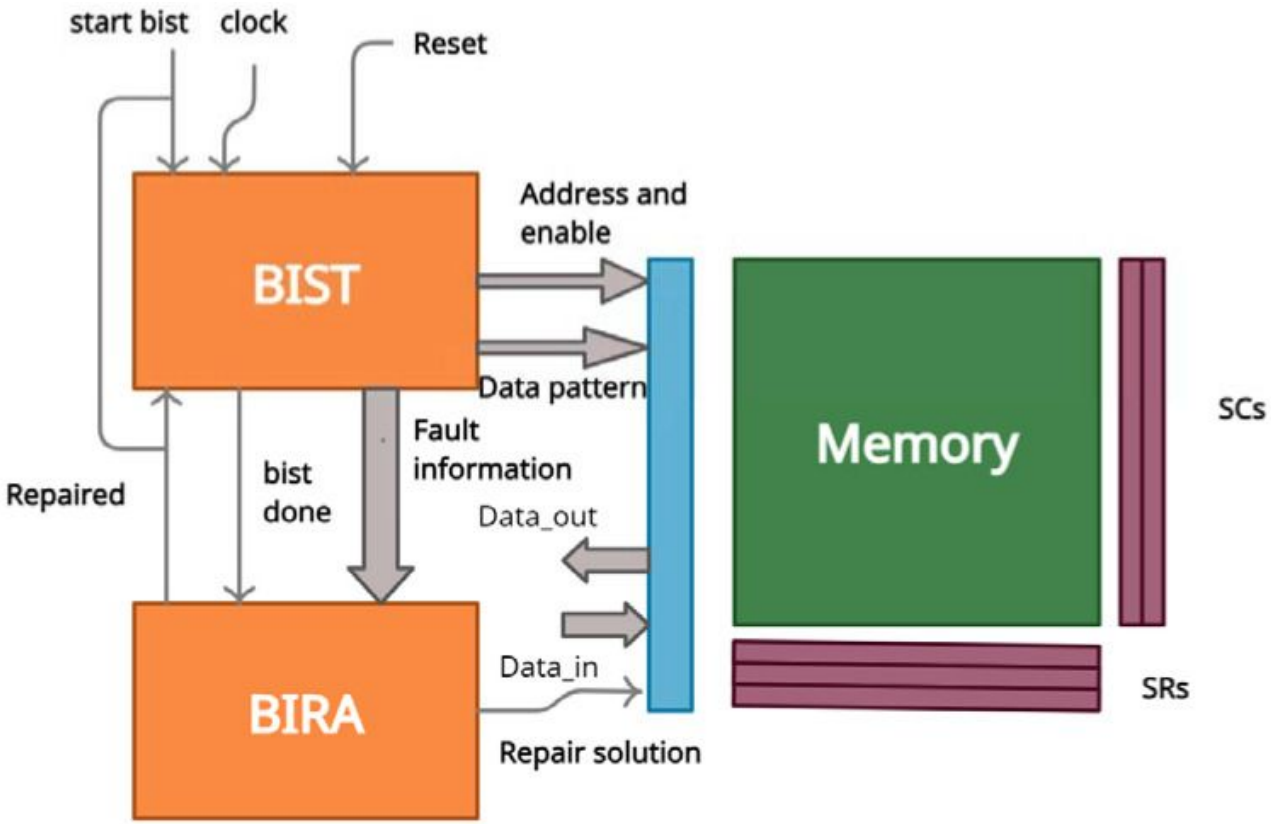


# of errors is not the only factor -- where the errors are is important

Memory Repair

Glossary
BISR: Built In Self Repair

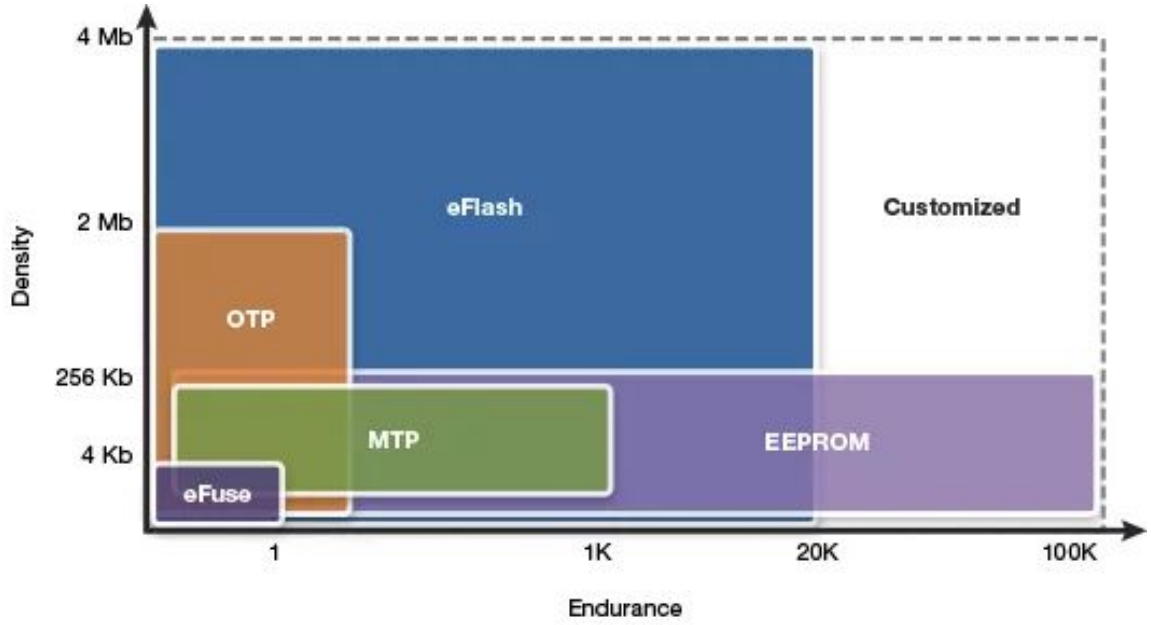
- On Startup, Run **BIST**
  - Access every memory location multiple times
- Use **BIRA** algorithm to assign spares
- Memory is now usable!
- This process is **BISR**



Storing the Results

Glossary
OTP: One-Time Programmable
MTP: Multi-Time Programmable

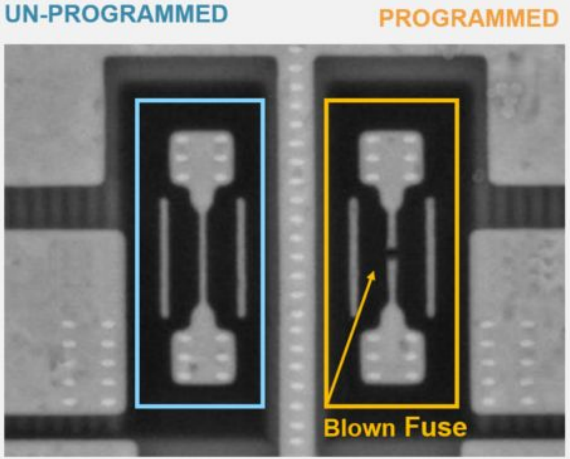
- **BISR** only ever needs to happen once
  - Store Result
- Only need to store a couple address bits per spare row/col
- This makes eFuses a really good option!



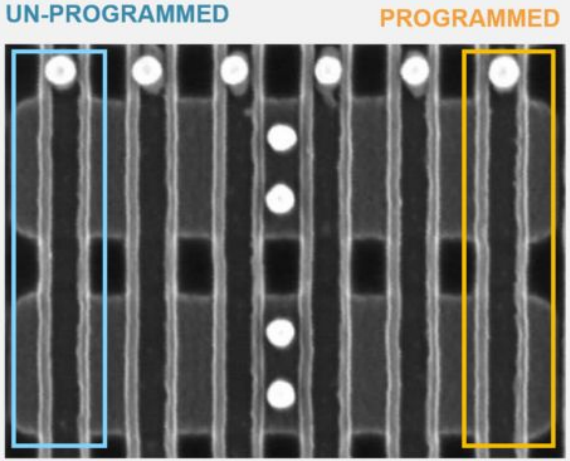
Efuses

- **Efuses** store a “1” by melting a tiny bit of metal layer in the chip
  - Apply a specific voltage for a specific time
- For security applications, can use **Anti-Fuse**
  - Same thing but changes the conductivity of a dielectric
  - Not viewable by microscope

eFuse OTP
(electrical-fuse)



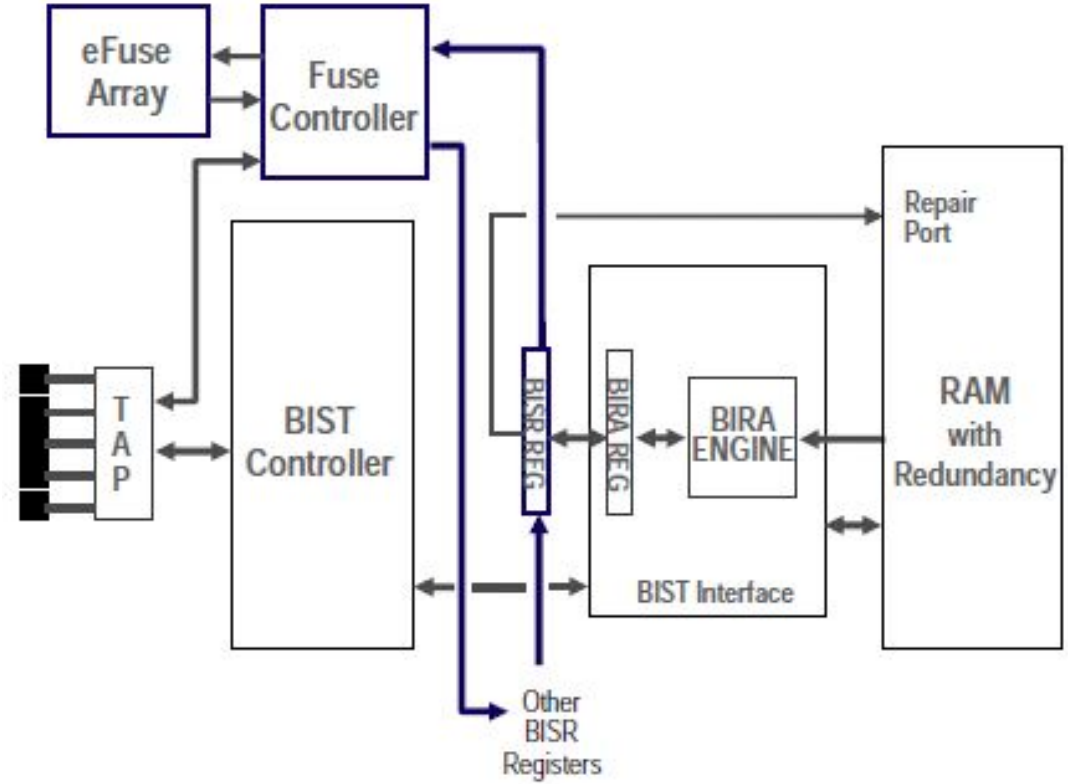
Anti-Fuse OTP
(Secure OTP)



BISR

Full Picture of **BISR** with storage:

- BIST finds faults
- BIRA evaluates solvability with spares
- eFuse FSM writes spare assignment to eFuse array
- Success or Failure reported to debug interface
  - JTAG TAP controller
  - Chip moves on in bringup



Conclusion

- Lots of complexity goes into fault tolerance
  - All this for no new features, just better yields

- As designers, we should think about post-silicon bringup
  - Whole world of DFT to be explored

Design For Test: features that enable easier debug and repair at bring-up