Tricks of the Test Trade: ATPG Methods That Improve Fault Coverage of SoC Devices

Mike Lewis

Leah Clark

Cypress Semiconductor

Purpose

- Improving fault coverage with Scan-Through RAM
  - A new approach for handling shadow logic testing
- It had better be good
  - Your time is valuable
  - It's the last paper of the last User Session of SNUG 2002
  - Cliff Cummings is presenting next door
- We think it is!
  - Obtain a high degree of fault coverage…
  - …on designs containing lots of embedded memories…
  - …with little additional logic

Outline

- Motivation
- Standard Approaches
- Scan-Through RAM Approach
- Empirical Results
- Gotchas
- Extensions
- Conclusion

Motivation

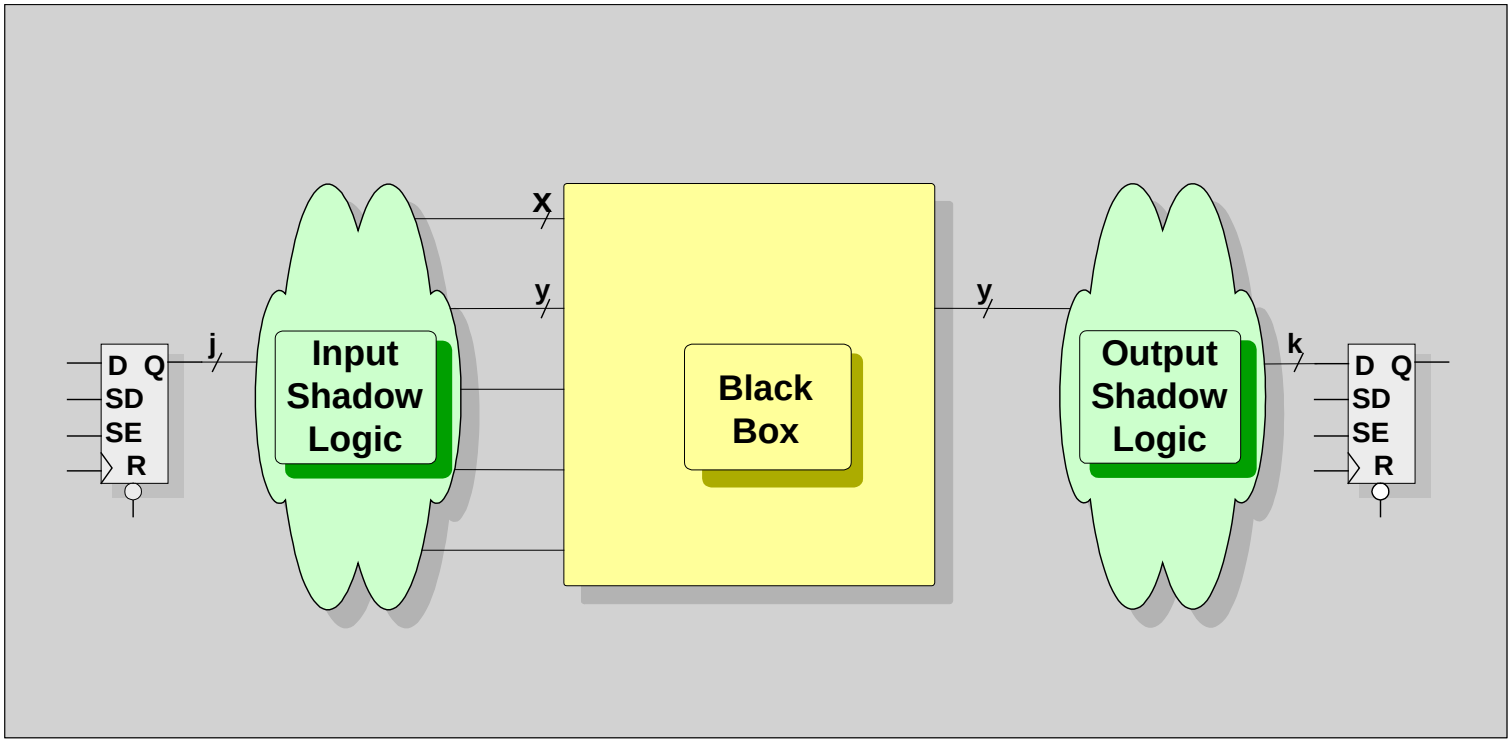
- Shadow Logic
  - Combinatorial logic immediately surrounding embedded memory
  - Problematic for ATPG tools to fault-grade properly
  - Neither observable nor controllable
- Observability

  “A node is observable if you can predict the response on it and propagate the fault effect to the primary outputs where you can measure the response. A primary output is an output that can be directly observed in the test environment."
- Controllability

  “A node is controllable if you can drive it to a specified logic value by setting the primary inputs to specific values. A primary input is an input that can be directly controlled in the test environment.”

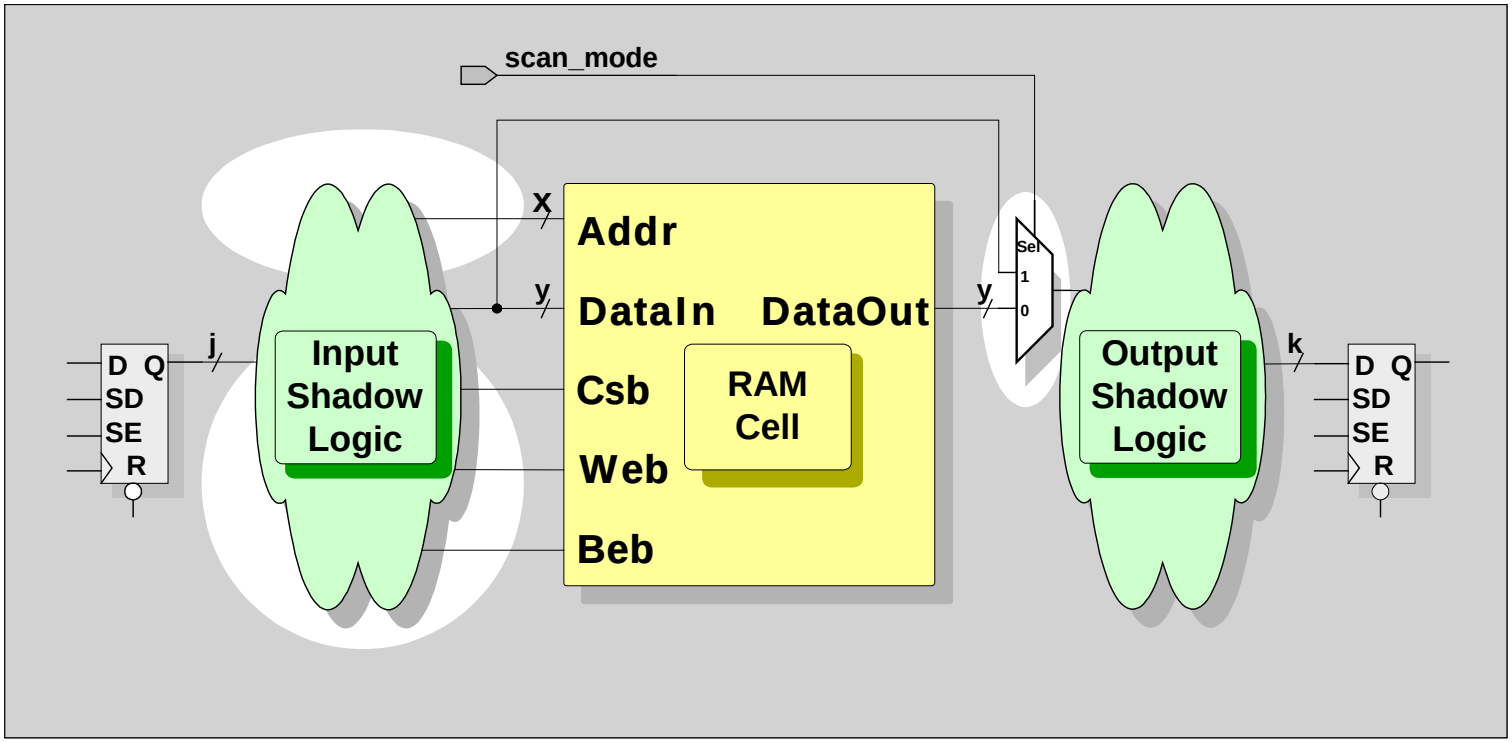
Shadow Logic

- Input shadow logic is not observable
- Output shadow logic is not controllable
- More on-chip memory → Less fault coverage



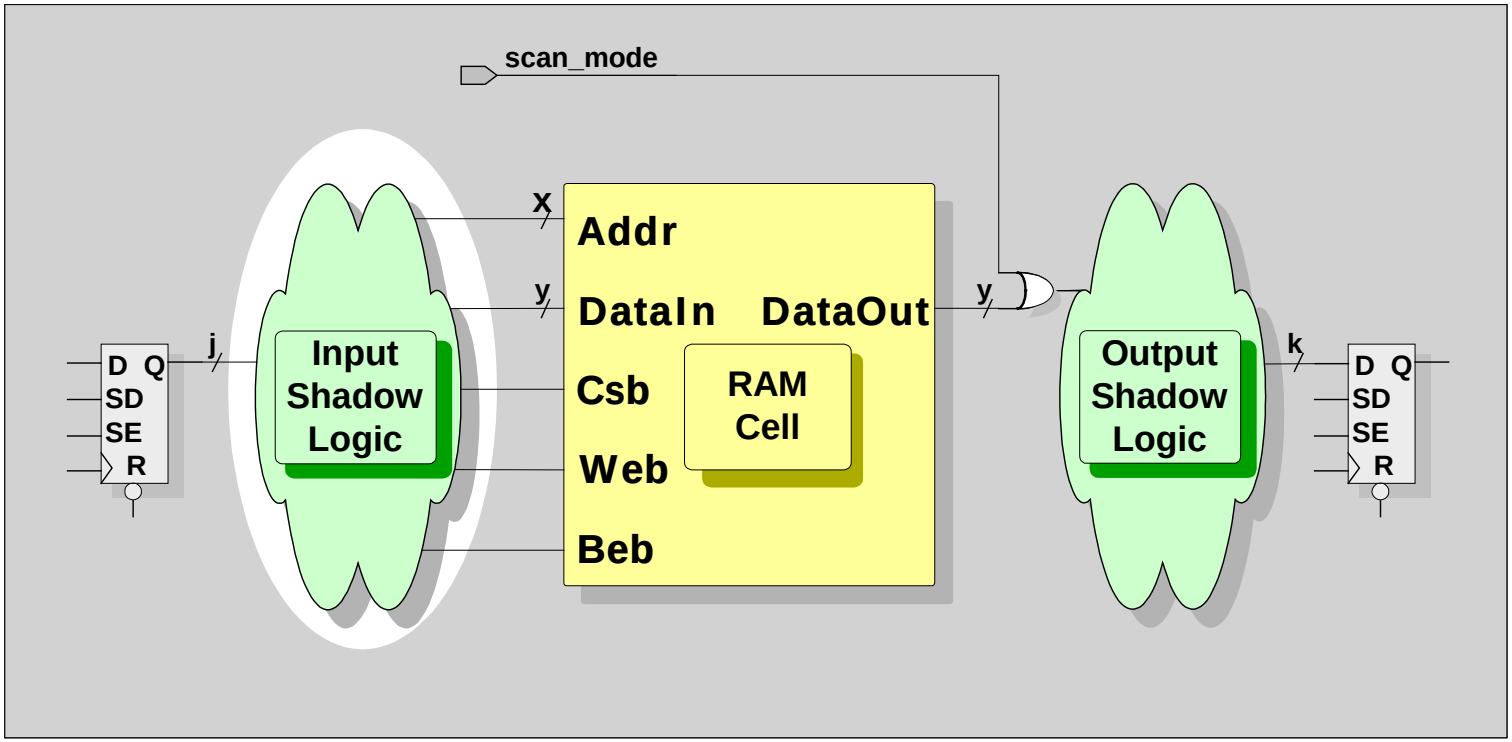
Mux Bypass

- Pro: Fully covers input and output datapath
- Con: Wide muxes, static delay on read data path, some fault escapes



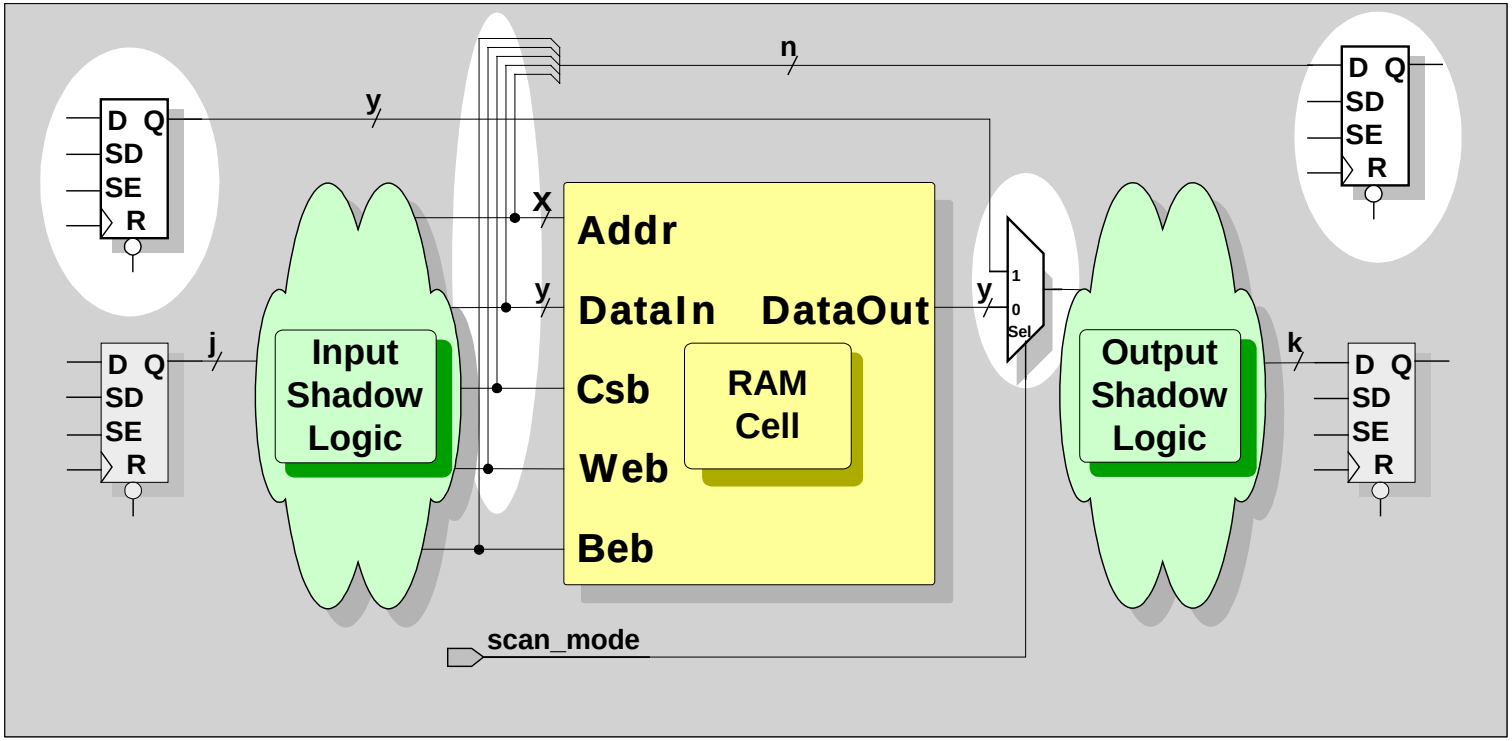
Forced Controllability

- Pro: Covers output logic, little real estate required
- Con: No coverage of input logic, small but real static delay on read data path



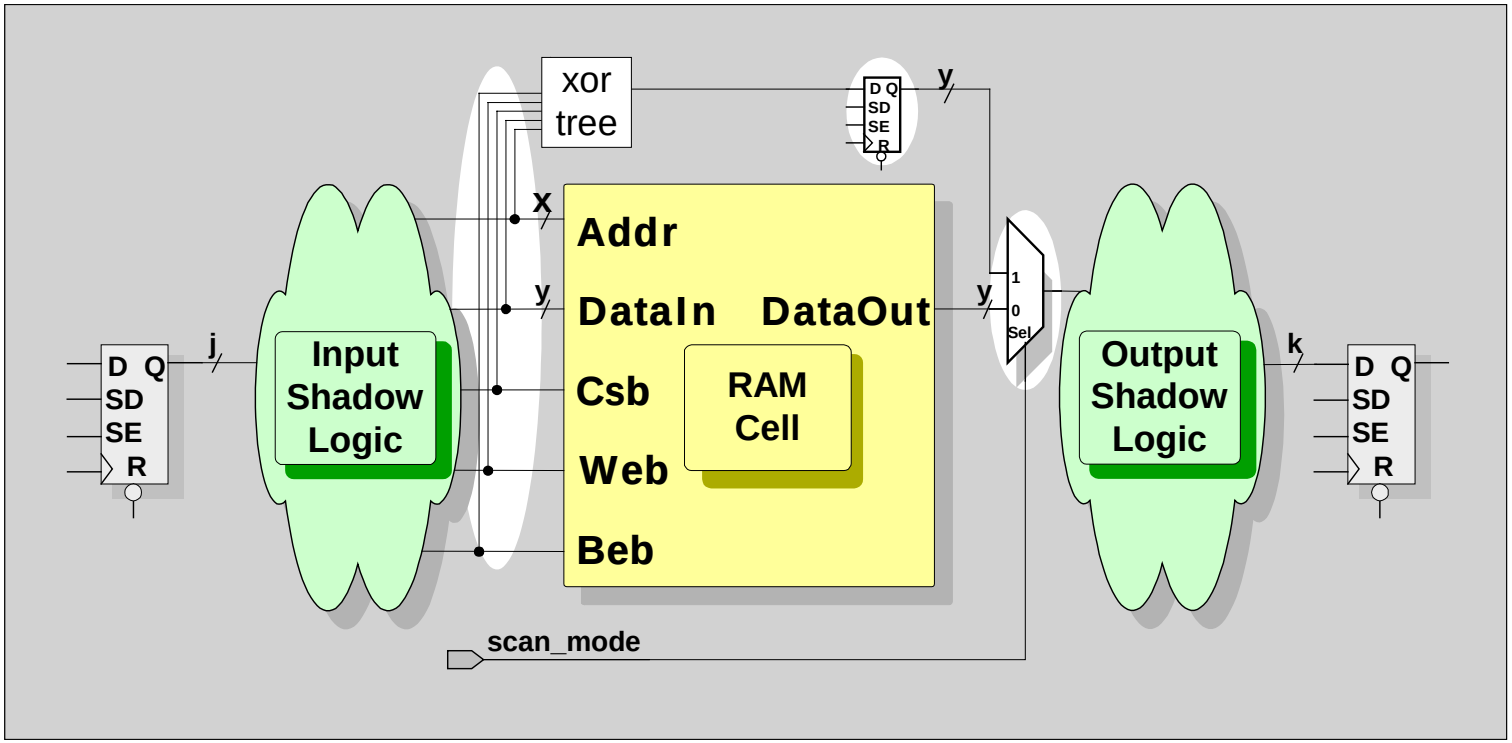
Wrapper or Register Collar

- Pro: Full coverage of shadow logic
- Con: Large routing and area overhead, static delay on read data bus, increased scan chain depth



Smart Wrapper

- Pro: Full coverage, xor tree less expensive than FFs
- Con: Still some area/routing overhead, and still have static delay on read data bus



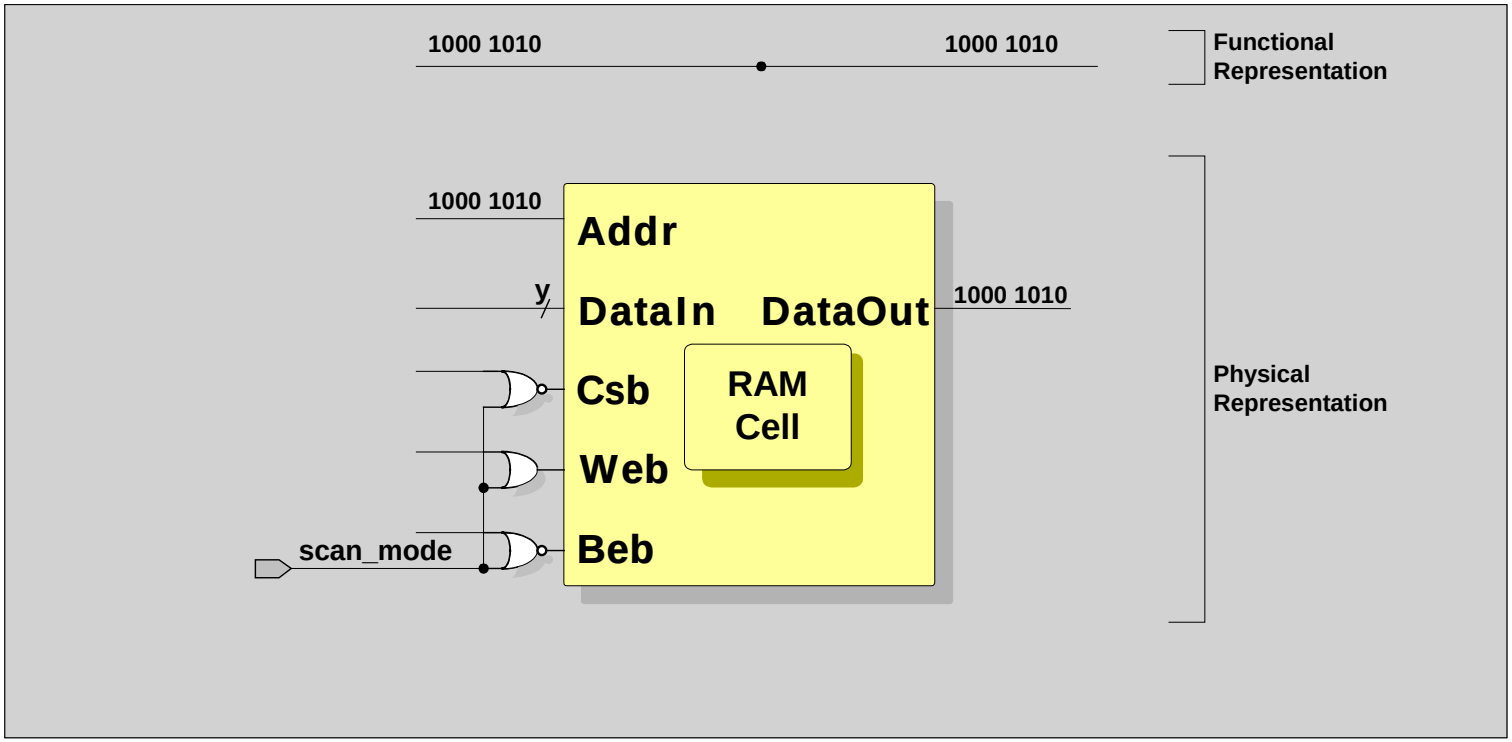
Do Nothing

Scan-through RAM Approach

- STR Philosophy
  - Adds a small amount of DFT Logic
  - Holds memory cells in read mode during ATPG
  - Initialization sequence preloads memory cells
  - Memory outputs are deterministic
  - Behavioral model of memory cells in place of black box
  - Memory outputs are known by ATPG tool
  - Flow fully verified in silicon using EZ-USB FX2 (USB 2.0 Integrated Peripheral/Microcontroller)

STR Implementation: DFT Logic

- Simple 2-input logic gates hold RAM in read mode
- STR area increase: function of number of instances
- STR function: DataOut = f(Addr)



STR Implementation: ATPG RAM Models

- Test Compiler (Legacy) Flow
  - Start with the .lib models (see paper for a template)
  - Models must be modified to represent a “wire” or simple function
  - Mapping of address to data pins is arbitrary
  - Example of mapping algorithm in paper
- TetraMax Flow
  - TetraMax uses *special* Verilog models of library cells in its ATPG algorithm instead of .lib models. Be sure to refer to the TetraMax manual for details on how to create these models for use during ATPG.

STR Implementation: ATPG ROM Models

- "Wire" function model falls apart for ROM cells
  - ROM array is pre-defined
  - Typically contains data of a random nature
  - Inefficient to build a .lib model
- Different model needed
  - Synthesizable Verilog model generated

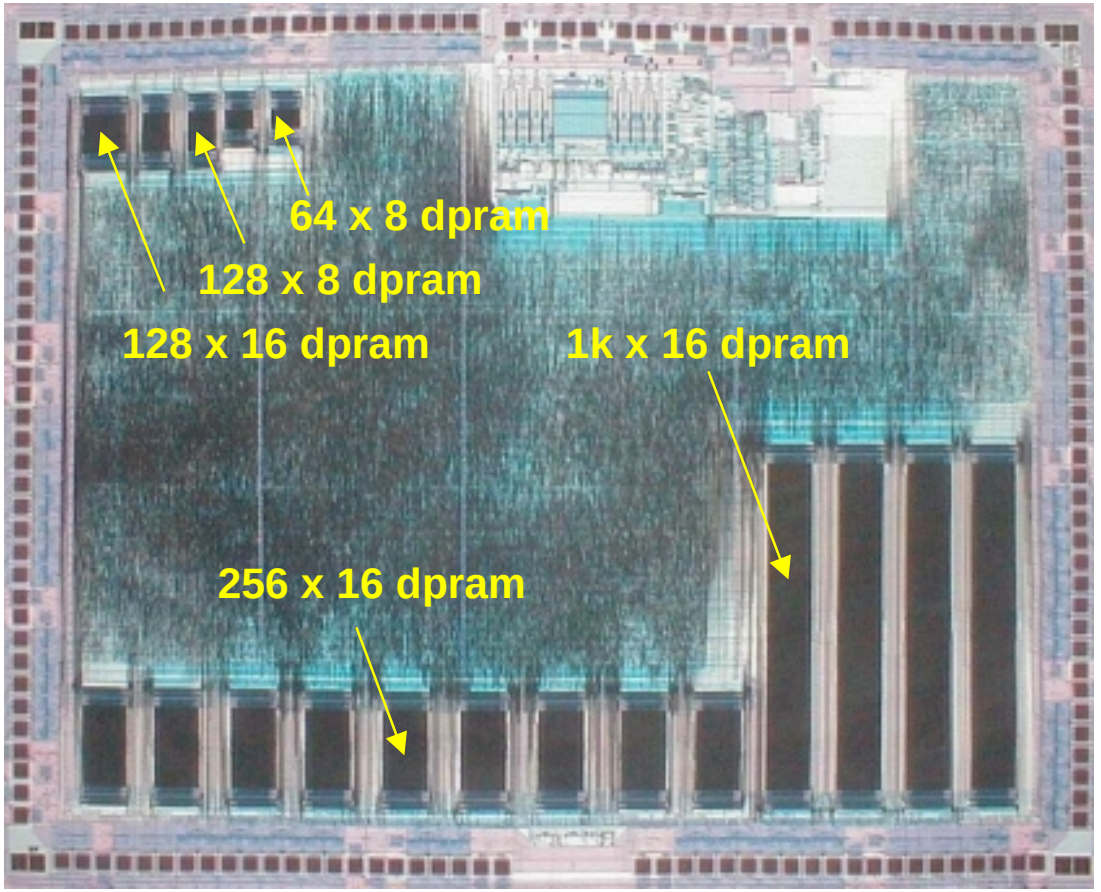
    Large case statement maps the address to data
  - Model compiled with Design Compiler
  - .db file describes the function for ATPG
  - Example in paper
- Flow fully verified in silicon

STR Implementation: ATPG Simulation

- Memories must be preloaded with the desired data
  - Verilog model performs the initialization of the memory cell using the `$readmemh` system task
  - Script can develop .dat files in `$readmemh`-compatible format (See Appendix 2)
  - `$readmemh` saves on simulation cycles
  - Example of Verilog preload code in paper
- **IMPORTANT:** You must have the capability to access your memory cells on the tester (e.g. via a RAM access test mode)

Empirical Results

- Cypress' EZ-USB FX2 is a USB 2.0 microcontroller comprising:
  - 100k logic gates
  - 13k of dual-port RAM
  - STR on-chip logic
- CAD tools used
  - Synopsys' Test Compiler v2000.05-1
  - Cadence's Verilog-XL



Empirical Results

- RAM test approaches compared
  - Do Nothing

    No additional area, but no additional coverage either
  - Partial RAM collar

    Real estate hog, good additional coverage
  - STR

    Acceptable level of area overhead, good additional coverage

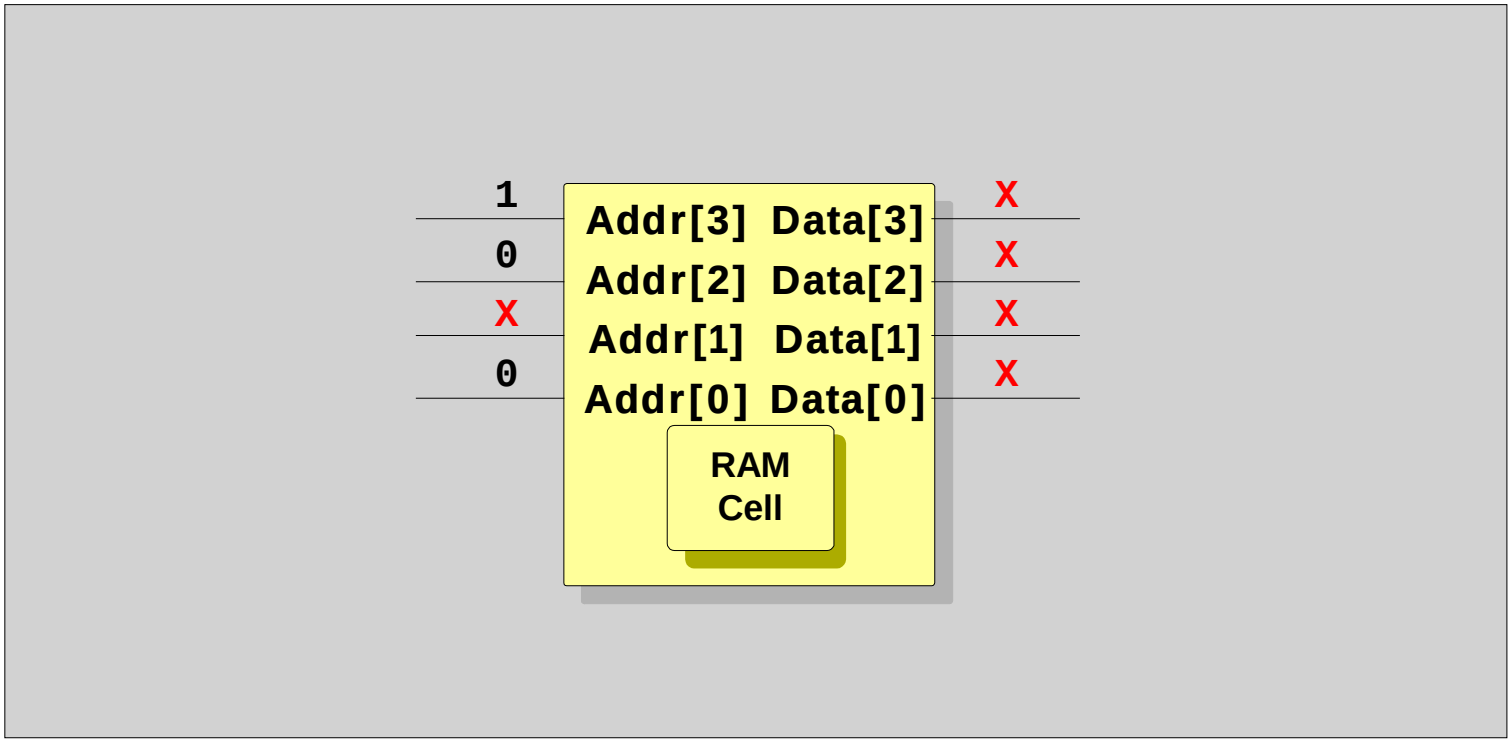
ATPG Method	Die Size	Fault Coverage
Nothing	0.0%	84%
Partial RAM Collar	4.6%	93%
STR	**0.5%**	**93%**

Gotchas

- Synopsys model generation
  - Working with .lib files is slightly more fun than buying a house in Silicon Valley
  - Cannot define function as a bus – must do bit-by-bit
  - Template is in the paper
  - Avoid by using TetraMax
- Vector depth limitations
  - Can be vector intensive to pre-load RAMs
  - Use parallel vector capability of tester
  - Whatever you do, work with your test engineer

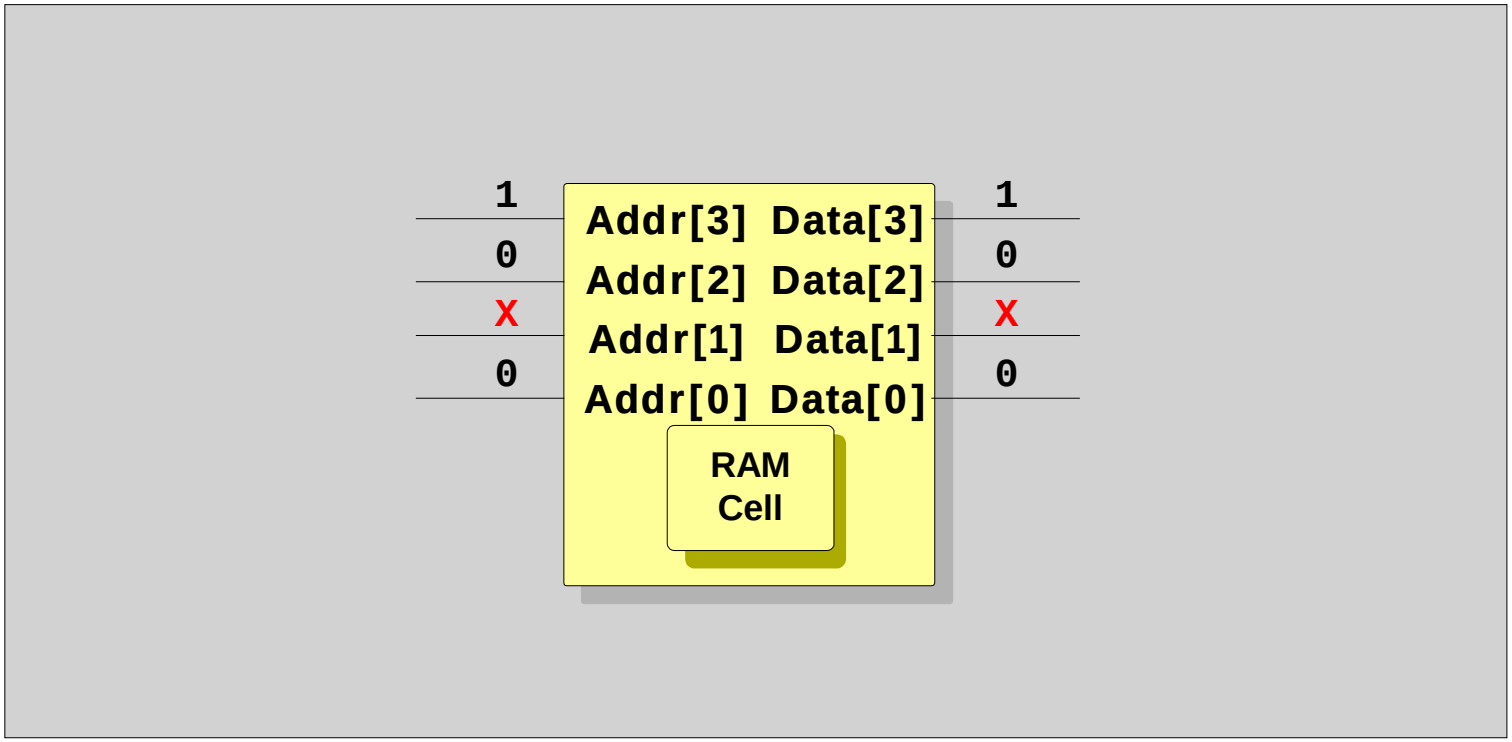
Gotchas

- Unknown values on address bus during simulation
  - In some cases, unknown values may exist on the address bus
  - Verilog model can't determine which cell to access
  - The entire output data bus is forced to unknown



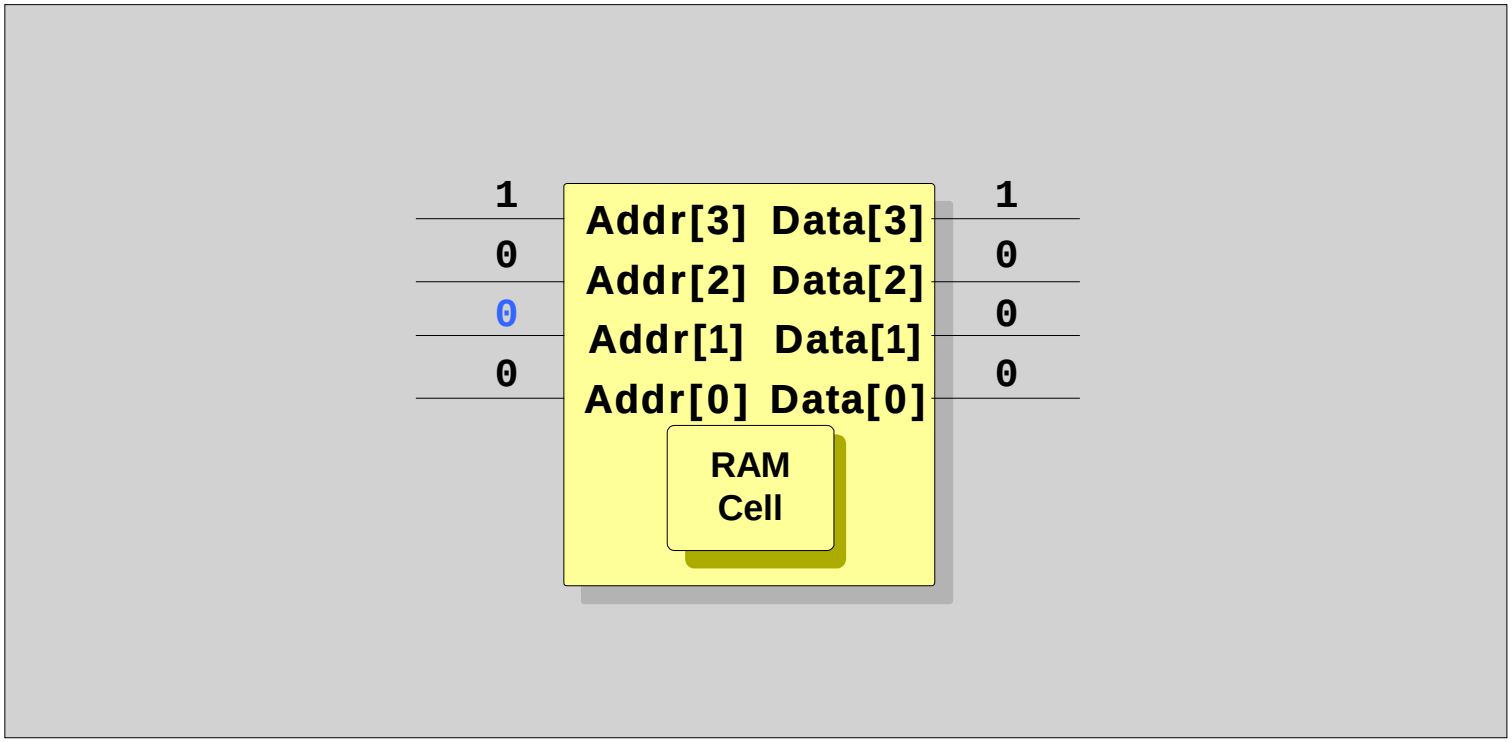
Gotchas

- Unknown values on address bus during simulation
  - .lib model represents a wire model
  - Only unknown input propagates to unknown output
  - Leads to ATPG simulation mismatches



Gotchas

- Unknown values on address bus during simulation
  - `write_test_input_dont_care_value = "0"`
    `(or “1”)`



Extensions

- Dual-Port RAMs
  - Diagrams in presentation show single-port only for clarity
  - The FX2 project used only dual-port RAMs
  - DFT logic applies to both ports equally
  - .lib specifies same exact function for both ports
- Combined read/write data bus
  - Control logic forces the data bus to be an output during scan_mode
- Shared address bus/shared tri-state data bus
  - Multiple RAM instances create a larger memory space
  - MSBs decode the CS/OE to the desired RAM instance
  - Fault coverage loss occurs if every instance has the same image
  - Different data patterns in each RAM instance will improve coverage

Conclusion

- STR Pros
  - Very low area and routing penalty
  - Excellent increase in fault coverage
  - Models and flow are re-usable from project to project
- STR Cons
  - Requires access to memory cells
  - High initial development investment
    - Initial development time will be less with TetraMax, since the .lib hurdles are (re)moved

Method	Static delay on read data?	Static delay on input signals?	Increased scan depth?	Increased gate count?	Increased routing?	Increased Coverage?
Mux Bypass	Y	N	0	2	2	2
Forced Control	Y	N	0	1	1	1
Full Wrapper	Y	N*	3	3	3	3
Smart Wrapper	Y	N*	2	2	2	3
Do Nothing	N	N	0	0	0	0
STR	N	Y	0	1	0	3

N*: In these cases, there is no static delay per se, but there is additional loading placed on these signals.