Embedded Test and Debug of Full Custom and Synthesisable Microprocessor Cores

Andrew Burdass, Gary Campbell, Richard Grisenthwaite, David Gwilt, Peter Harrod and Richard York

ARM Ltd, Cambridge, UK

Peter.Harrod@arm.com

Purpose

- To compare and contrast test approaches in:
  - A full custom microprocessor core
  - A synthesisable microprocessor core
- To discuss IP protection issues associated with test of embedded cores
- To briefly discuss requirements of embedded debug of real-time systems

Outline

- Introduction to processor core testing
- Example of full custom core : ARM920T
  - Bus-based test methodology and coverage
    - CPU functional vectors
    - Cache and MMU array testing
- Example of synthesisable core : ARM966E-S
  - Test methodology and coverage
    - Scan testing
    - INTEST wrapper
    - Memory BIST
- Embedded core debug
- Conclusions

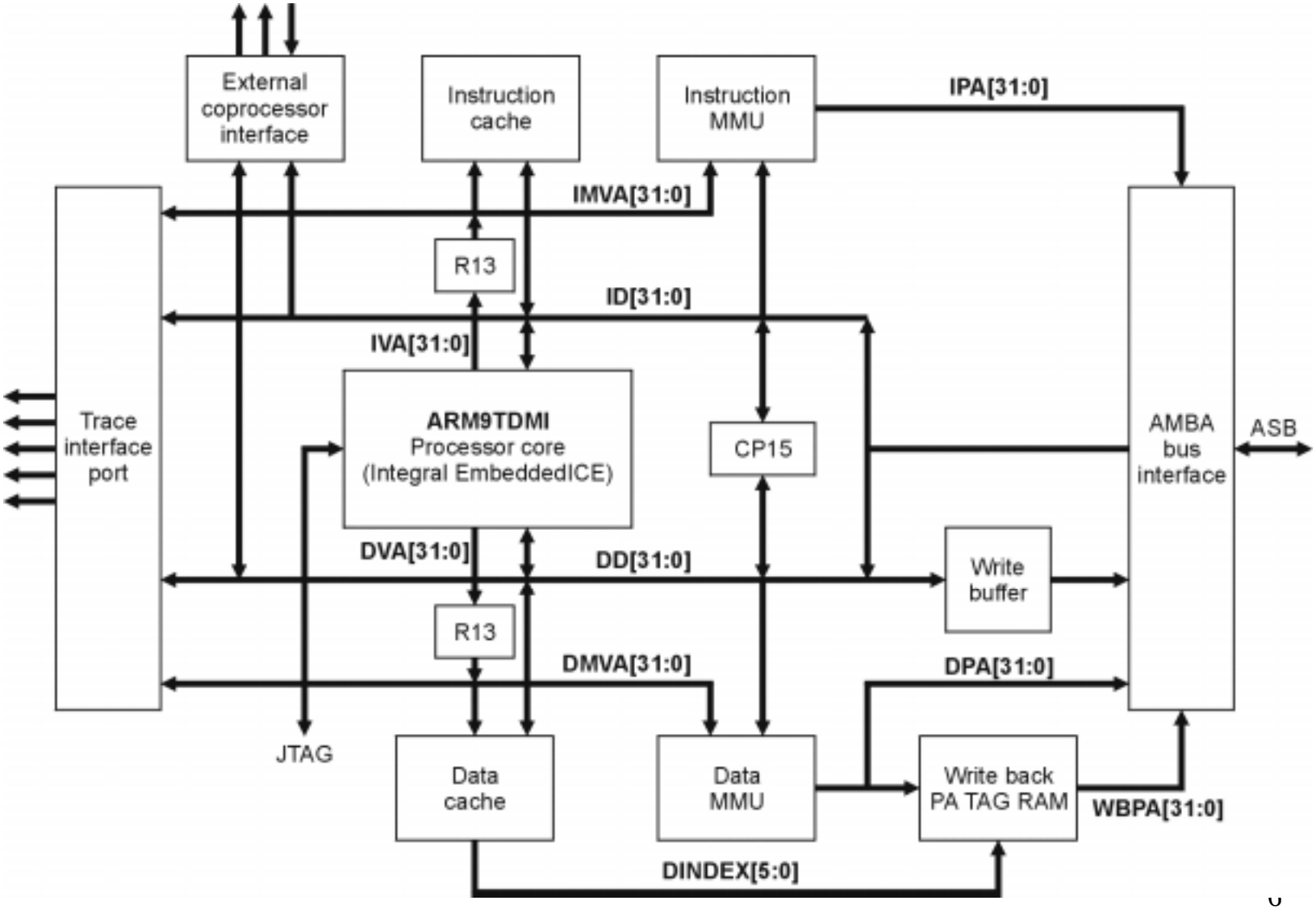
Full custom processor core - Hard IP

- Hard layout is created by IP developer
- Supplied as a database ported to the target process
- Tested using logic block isolation, AMBA TIC
- Test vectors produced by IP developer

Synthesisable CPU core - Soft IP

- Soft IP deliverables:
  - Synthesizable RTL
  - Fully automated synthesis scripts
  - Validation environment
  - Documentation
  - Design Signoff Model (DSM)
- Tested using industry standard scan insertion/ATPG tools
  - Licensee produces test vectors

Full Custom Core Example - ARM920T



AMBA bus-based test

- Reuses on-chip bus as test data channel
  - AMBA ASB
- Reusable test vectors
  - Vectors written in higher level language
  - Compiled for system implementation
- Low logic overhead test interface controller
  - 3 pin interface, 1.5K gates
- Test harness for controlling inputs and observing outputs
  - AMBA bus slave

ARM920T AMBA Test

- Address mapped AMBA bus slave
- Six test modes
  - Instruction Cache, Data Cache
  - Instruction MMU, Data MMU
  - Physical address Tag RAM
  - Functional (core & memory system controllers)
- 4KB of address space required, re-mapped for each test mode

Memory array test modes

- Address mapped burst read and write locations of up to 128 words for
  - Cache CAM & RAM
  - TLB CAM & RAM
  - Physical address Tag RAM
- CAM match - RAM read
- CAM invalidate & dirty

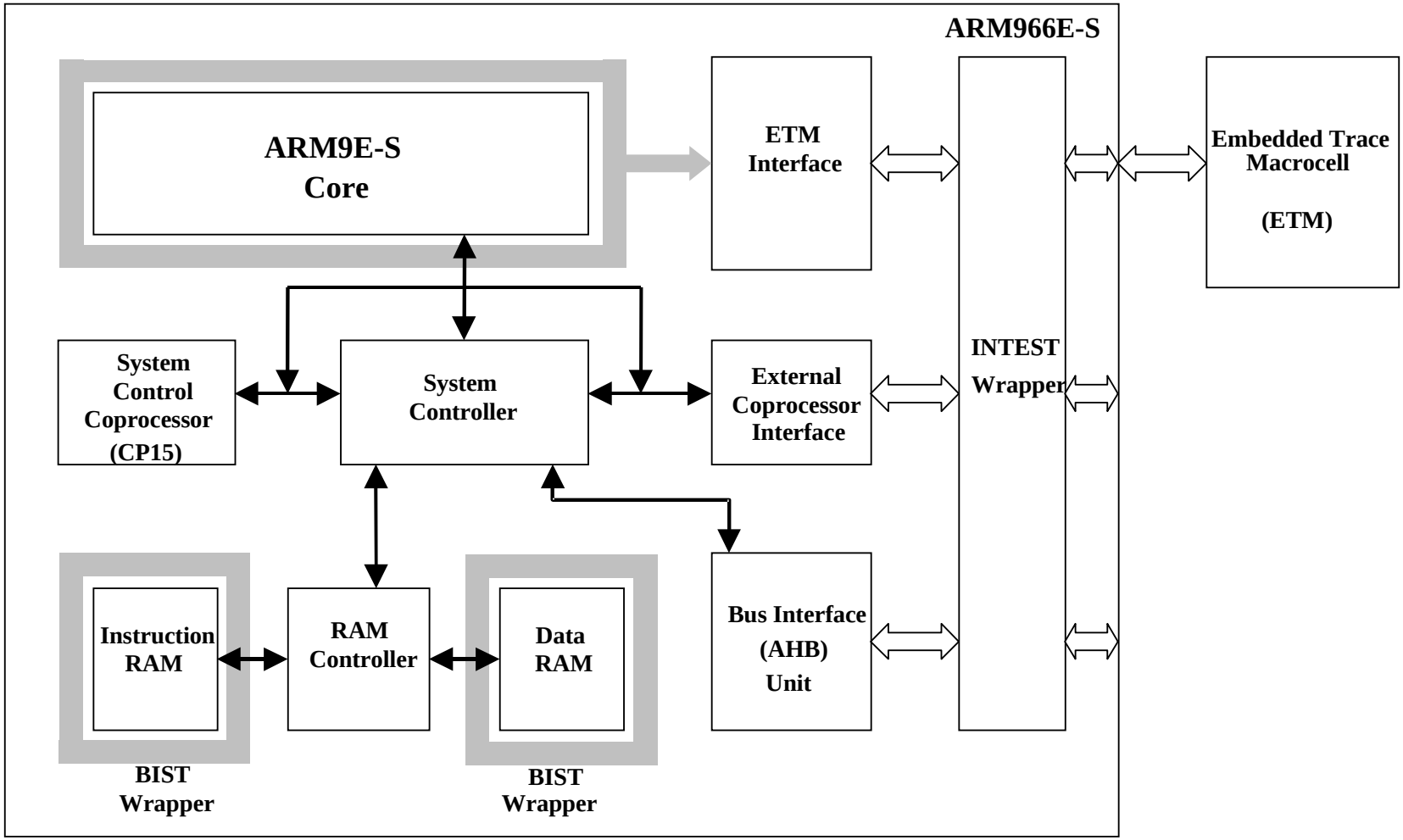
Functional test mode

- Three 32-bit write locations
  - All ARM920T inputs
- Six 32-bit read locations
  - All ARM920T outputs
  - Functionally grouped
- Bit mapped to nine bits of the address
  - Any subset of the macrocell I/O can be tested

ARM920T Test Strategy - Results

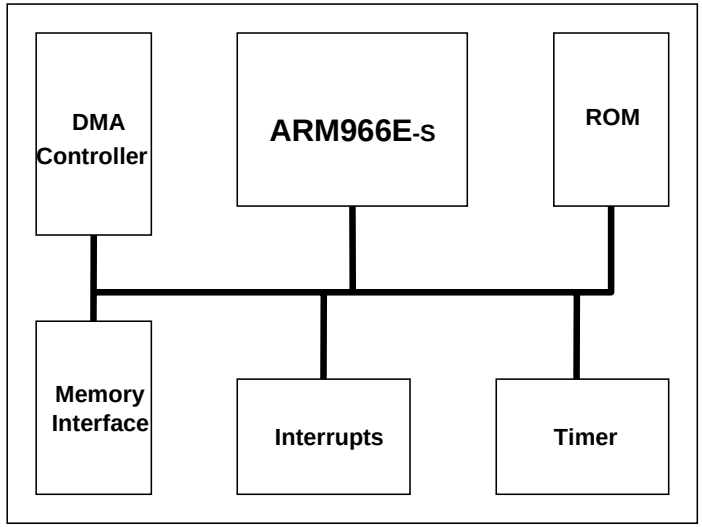
- Low transistor overhead, due to bus-based testing and DFT features of caches and MMUs
- Test logic overhead overall is about 1%
- ~100% fault coverage for arrays; ~90% for other logic
- Bus-based approach reduces vector count compared to serialised vector method

Synthesisable Core Example - ARM966E-S



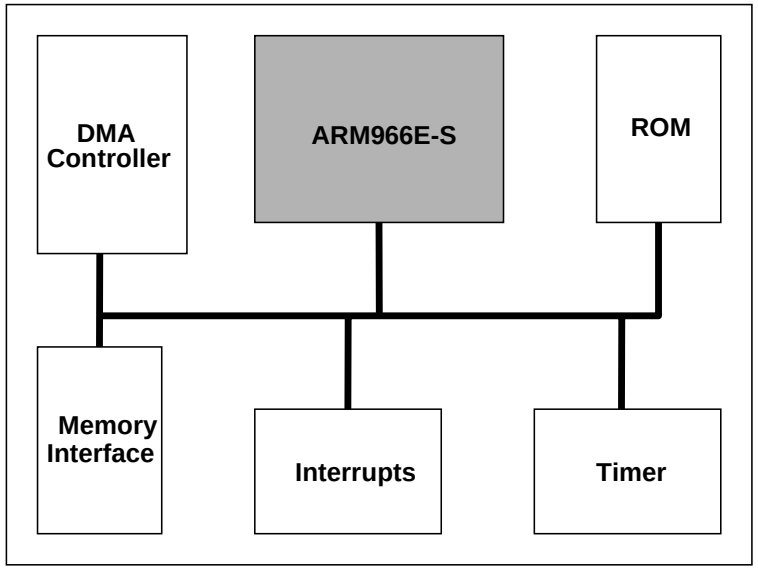
Using IP - By Licensee

- Licensee creates macrocell for use in their own SoCs
- Tools (Synthesis, layout, scan-insertion, ATPG) are allowed IP visibility



Using IP - By OEM

- Licensee creates black-box model of the macrocell for use by the OEM
- Tools do not have visibility of the IP
- DSM allows timing and cycle accurate simulation
- Licensee hardens IP and produces test vectors

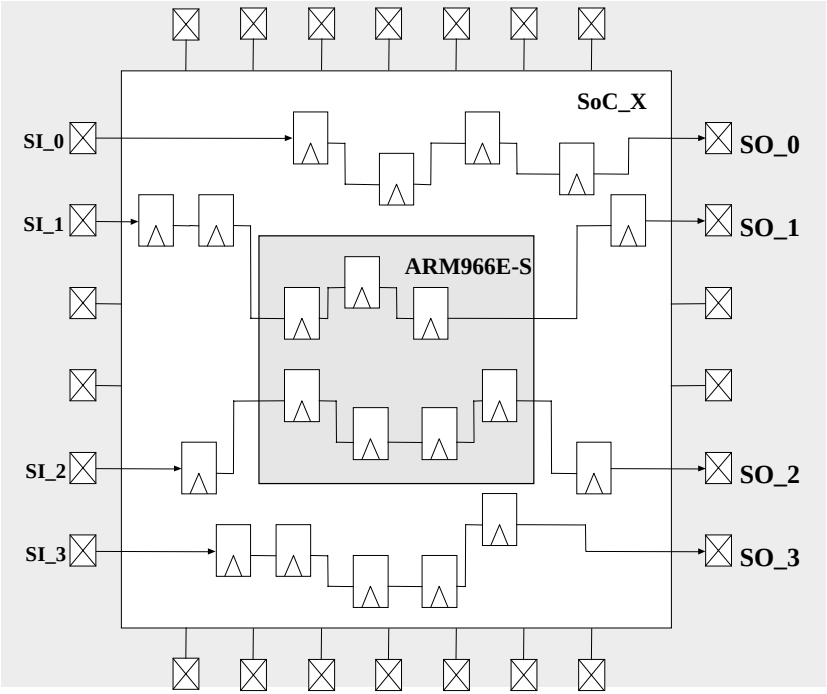


ARM966E-S Test Features

- Scan-insertion/ATPG used to test majority of logic
- INTEST wrapper to resolve shadow logic
- RAM BIST

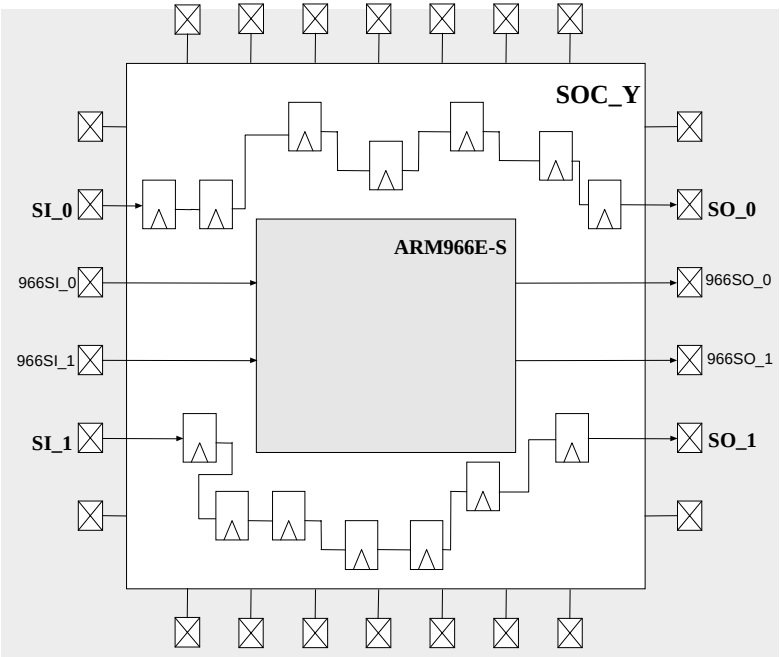
Top Down Scan Insertion

- Test tools have IP visibility
- Scan insertion performed from top level of SoC
- Macrocell synthesized scan ready

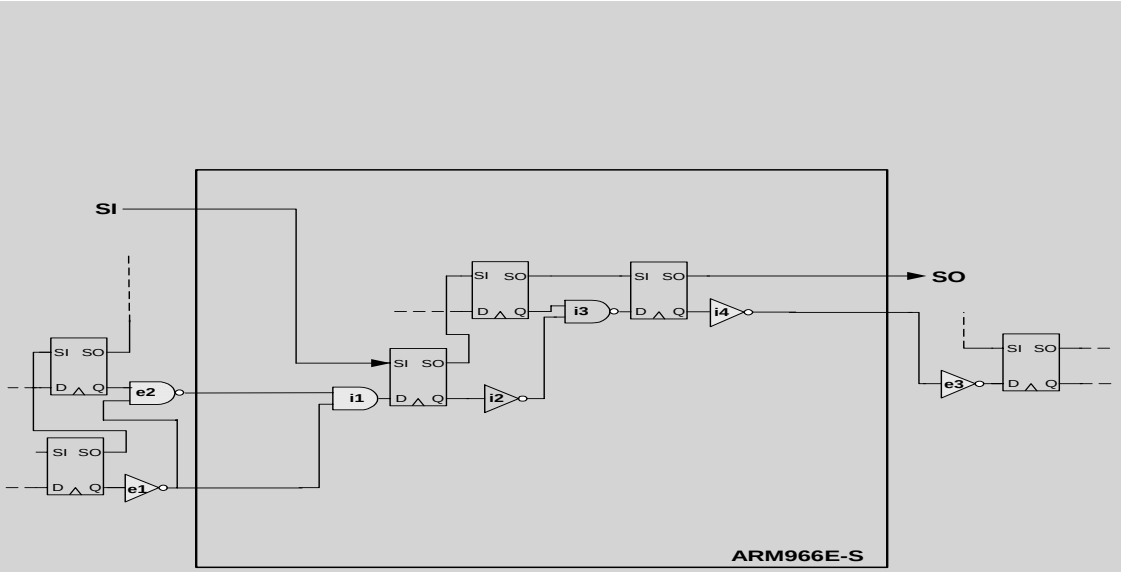


Embedded Scan Insertion

- Test tools do not have IP visibility
- Scan chains and test vectors produced by licensee
- Separate macrocell scan chains

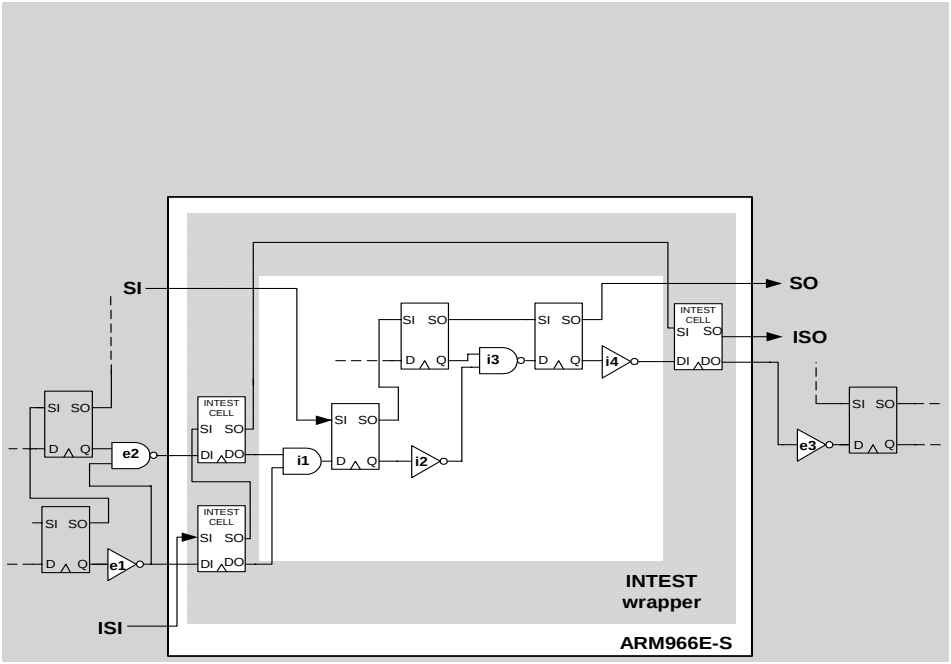


INTEST Wrapper (1)



- Embedded (black-box) usage
- Macrocell scan chains generated by licensee. Top level scan chains created by OEM
- Internal and External shadow logic cannot be tested => Reduced test coverage

INTEST Wrapper (2)



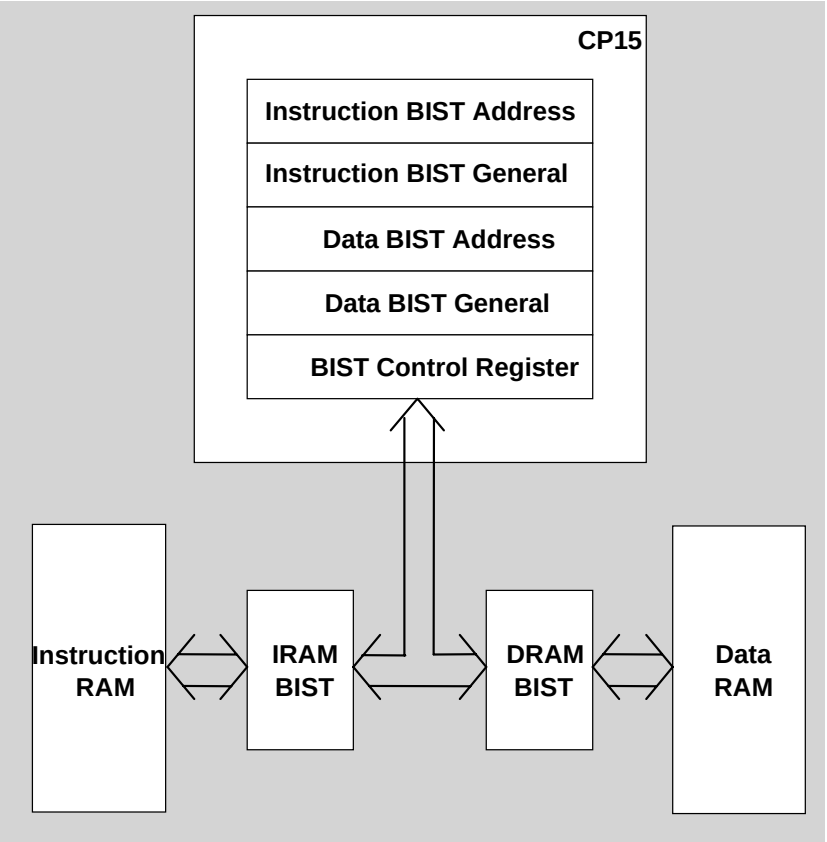
- Synthesis option creates INTEST scan chain around the boundary of the macrocell
- Resolves Internal and External shadows
- OEM allowed visibility of INTEST wrapper

RAM BIST (1)

- BIST Wrappers included with ARM966E-S RAM interfaces
- Automatically synthesize to the correct size
- Advantages:
  - No licensee effort required
  - No performance degradation
  - Unified and controlled

RAM BIST (2)

- BIST controlled via CP15 Registers
- BIST programmed to run on entire array or smaller segment
- Simultaneous BIST
- Core access disabled during BIST



RAM BIST (3)

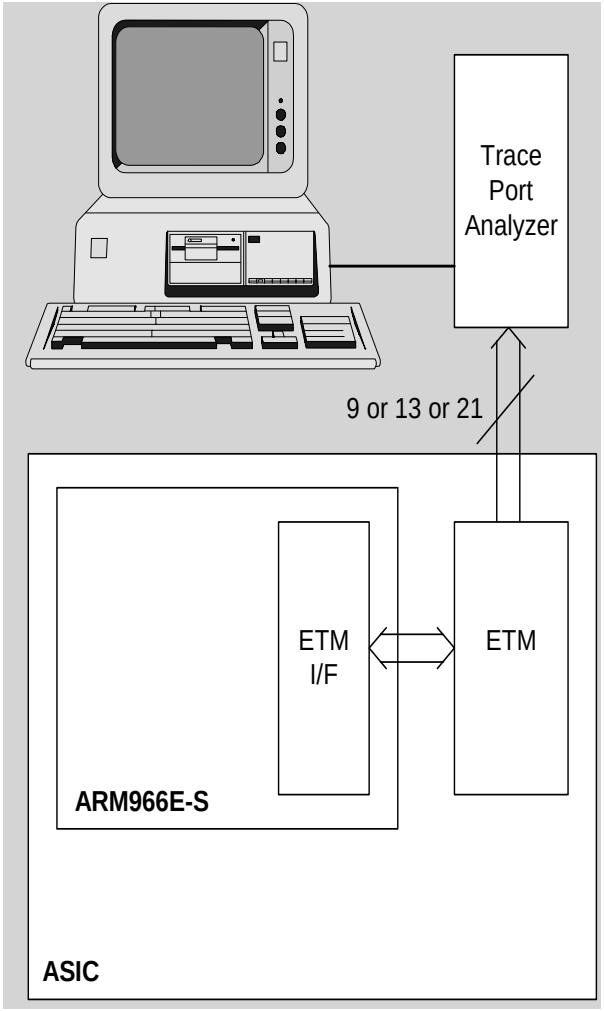
- 6N algorithm
- Algorithm can be replaced by licensee preferred algorithm
- Licensee RAM already containing BIST can be hooked up to CP15 control

Code Tracing and Debug

- Trace requirements evolving:
  - Deeply embedded processor and memory
  - Increase in clock frequency
  - Real time trace required
  - Non-intrusive

Embedded Trace Macrocell

- Full Instruction tracing.
- 3 ETM sizes allow variable trigger resources and data tracing
- Trade off between debug facilities and cost



ARM966E-S Test Features Summary

Test Feature	Test Coverage	Area Overhead	Performance Overhead
Scan Insertion	96.5-99.6%	7-10%	0-5%
INTEST Wrapper	~100%	5.5%	0-2%
RAM BIST	98-99%	4%	0%

Test strategy comparison

- Good coverage of full custom core with:
  - Functional AMBA bus-based vectors
    - Benefit of reusability
  - Test features incorporated in Cache and MMU arrays
- Good coverage of synthesisable core with:
  - Full scan of core logic
  - Memory BIST for RAM arrays
  - INTEST wrapper for testing shadow logic in cases where IP cannot be revealed

Conclusions

- Different test approaches can both yield good results
- Tradeoffs in area, complexity, time-to-market, IP protection, performance and power impact, vector count - all have to be considered at design stage
- Requirements of P1500 must be considered
- Embedded trace macrocell provides for effective debug of real-time applications