Embedded Test and Debug of Full Custom and Synthesisable Microprocessor Cores

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

ARM Ltd
110 Fulbourn Road, Cambridge, CB1 9NJ, England
Peter.Harrod@arm.com

Abstract

This paper compares and contrasts two very different approaches to testing cached CPU macrocells that are typically embedded in a System on Chip (SoC). One uses a test bus to apply functional vectors, while the other uses a combination of scan insertion, memory BIST and test collars. IP protection issues and non-intrusive tracing are also discussed.

1 Introduction

The development of microprocessor macrocell cores in the form of reusable Intellectual Property (IP) presents many challenges, not the least of which is the effective testing of the core once it is deeply embedded in a System on Chip (SoC) design [1]. Alongside the manufacturing test of the core, debugging of application code, IP protection and on-line BIST are other considerations that must be taken into account during the design of the microprocessor core.

Processor cores are typically developed in one of two distinct styles:

- **Full-custom processor cores**, in which the IP provider produces a GDSII layout database of the design (a so-called hard core) and delivers this along with associated deliverables such as test patterns to the licensee of the IP.
- **Synthesisable processor cores,** in which the IP provider delivers an RTL model of the design (a so-called soft core), together with the infrastructure needed for the IP licensee to synthesise the design, do scan insertion and ATPG, place and route, and then integrate the resulting layout into their SoC.

Both types of processor core have been developed at ARM recently and each has used a different approach to developing the embedded test mechanisms and patterns that are needed. This paper will compare and contrast the different approaches to test, using a processor core of each type by way of example.

An example of a full custom microprocessor core is the ARM920T™ core, which is targetted at applications where high performance, small die size and low power are all important. It uses the AMBA bus test methodology, using functional patterns to gain a high fault coverage. On the other hand, the ARM966E-S™ core is a fully synthesisable microprocessor core; most of its logic can be tested using a full scan approach, using an industry-standard ATPG tool to generate the patterns.

2 Full custom processor core test approach

The traditional approach to developing processor cores for licensing as IP blocks has been to deliver the final layout for the processor in the form of a GDSII database. This is known as a hard core; the licensee of the IP is not able to make any changes to it. It is therefore the IP developer's choice as to how to construct the processor, using the most suitable methods available. Typically this might involve hand-crafting a full custom datapath and cache arrays, to take advantage of the speed, power and die-size benefits of such regular blocks, coupled with synthesis and place and route of random control blocks. A functional test approach has typically been used, with the result that a full set of fault graded test vectors form part of the deliverables. A design that is typical of this approach is the ARM920T core, which is shown in Figure 1.

The ARM920T core is a Harvard architecture cached processor core, comprising the ARM9TDMI CPU, separate 16KB instruction and data caches, a Memory Management Unit (MMU) to provide address translation and access permission checks and a write buffer with 16 data and 4 address entries. It is interfaced to other logic blocks in an embedded system via the Advanced Microcontroller Bus Architecture (AMBA) bus [2]. The AMBA bus provides an effective channel for applying bus-based tests to any AMBA-compatible block in an

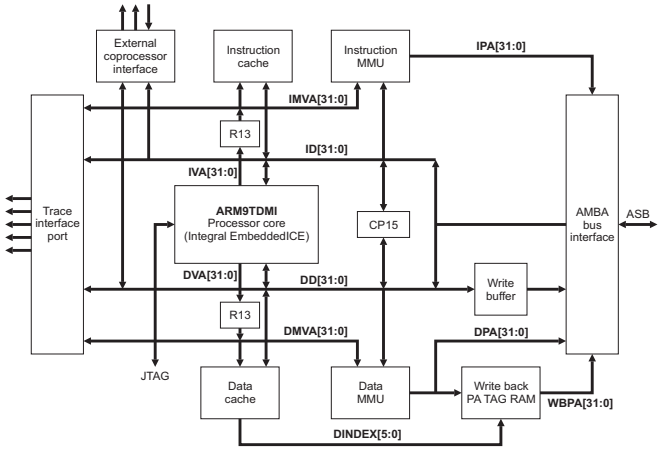

Figure 1 ARM920T Block Diagram

embedded system, which includes the ARM920T macrocell [3]. In a SoC design, a low gate count Test Interface Controller (TIC) is included to provide test access over the normal 32-bit data bus. In test mode the TIC becomes the AMBA bus master – in this way, vectors that are applied to the external pins of the SoC can be converted into internal AMBA System Bus (ASB) transfers. By using the AMBA bus-based test approach, the number of transistors dedicated to DFT on ARM920T core is only 1%.

2.1 ARM920T AMBA Test Modes

In normal functional mode, the ARM920T core is an AMBA bus master and it initiates bus transactions. In test mode, the ARM920T core can be selected as a bus slave and provides address-mapped read and write locations for silicon test. The ARM920T core requires 4KB of address space for test purposes. The entire address space is re-mapped for each test mode, except for the first word, the AMBA test base address, which has a reserved use. By writing different data to the AMBA test base address, one of six test modes can be selected:

- Instruction cache
- Data cache
- Instruction MMU
- Data MMU
- Physical Address Tag RAM
- Functional

The first five of these test modes cover the arrayed sections of the core, while the functional mode covers the CPU and other logic. In the array test modes, the ARM920T core provides address mapped locations for burst reads and writes to the caches, MMUs and PA Tag RAM. High fault coverage of the arrays is obtained by supplementing these burst accesses with some extra targetted tests:

- CAM match RAM read
- Invalidate by modified virtual address
- Invalidate all
- Dirty all (cache only)
- Lock down operations

In functional test mode, the ARM920T core is isolated and the TIC gains access to all inputs and outputs, using three write locations and six read locations. The TIC can then sequence tests by repeatedly writing the inputs, clocking the core and reading the outputs. The 9 read/write locations are bit-mapped to 9 bits of the address, which allows a subset of the I/Os to be tested. This enables the vector count to be reduced in cases where functionality is not used, for instance if the external coprocessor interface is left unconnected in a particular SoC application.

2.2 Predicted fault coverage and vector count

Fault simulation of the non-arrayed sections of the ARM920T core is still being carried out but fault coverage of around 90% is expected using 70K parallel vectors, based upon previous similar designs. However, fault coverage analysis of the arrayed sections (I-Cache, D-Cache, I-MMU, D-MMU and PATag Rams) has been completed and a maximum coverage of close to 100% is achieved. This coverage is achieved using the number of AMBA vectors shown in Table 1. Note that up to 9 AMBA vectors are needed to apply each parallel vector to the ARM920T CPU.

ARM920T Block	AMBA Vectors
I-Cache	52K
D-Cache	52K
I-MMU	7K
D-MMU	7K
PA Tag RAM	5K
CPU + others	630K

Table 1: ARM920T AMBA Vector Counts

Based upon the proportions of transistors in the arrayed and non-arrayed parts of the ARM920T, the overall fault coverage for the whole design is predicted to be around 99.5%. AMBA test thus performs well in both coverage and vector count, giving acceptably high fault coverage whilst keeping the vector count low by accessing the input and output locations for functional test in 32-bit words.

3 Synthesisable core test approach

One drawback of the full custom processor approach is that the layout database has to be translated (or *migrated*) to each new process technology that an IP licensee wishes to use. Each new layout has to be characterised to prove that it will work on that process. An alternative approach is to develop a fully synthesisable processor core. In this case, the RTL of the design can be delivered to the IP licensee, enabling re-synthesis and re-layout to a new process technology to be quite readily achieved. This approach also allows a certain degree of implementor customisation, for example the size of the embedded RAM or the memory BIST algorithm can be modified. A fully synthesisable design also lends itself to full scan test techniques; this simplifies the job of developing high test coverage patterns for the processor core itself, but introduces other issues with respect to embedded test and IP protection.

A recent example of a fully synthesisable processor core is the ARM966E-S core, which is shown in Figure 2. This macrocell, the ARM966E-S core, consists of an ARM9E-S core, Instruction and Data RAM, a bus interface unit (AHB) and a system control coprocessor (CP15). The ARM966E-S core is fully synthesisable and most of its logic can be tested with scan based ATPG tools [4].

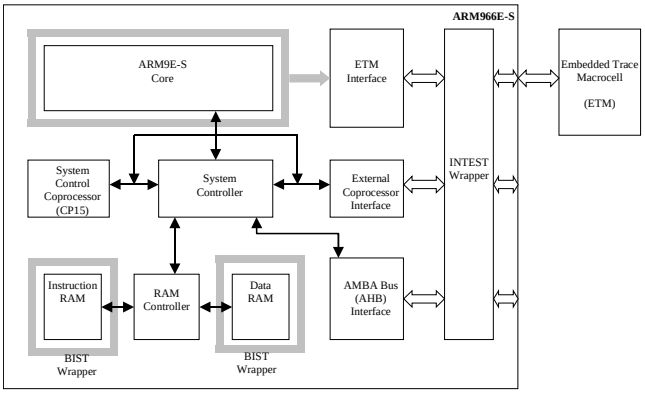

Figure 2: ARM966E-S Synthesisable Macrocell

An INTEST wrapper allows the shadow logic between the macrocell and the rest of the SoC to be tested without having to reveal any IP to a third party.

BIST wrappers for the Instruction and Data RAMs are provided with the macrocell RTL. These wrappers allow a unified and controlled approach to the testing of the RAMs within any ARM966E-S system.

The Embedded Trace Macrocell (ETM) is used to provide real-time, non intrusive Instruction and Data tracing.

3.1 Scan Testing

The ARM966E-S is a fully synchronous design and therefore traditional full scan techniques can be used to achieve high test coverage. Depending on how the macrocell is to be used , either a top-down or embedded scan insertion approach can be used.

The top-down approach would be appropriate in situations where licensees of the ARM966E-S are incorporating it in their own SoC design. In this case, the licensee has full visibility of the IP and can perform scan insertion from the top-level, giving the test tools the greatest flexibility in scan chain creation. Alternatively, the ARM966E-S could be synthesised with scan chains pre-inserted and the test tool could then connect up these chains to those in the rest of the SoC as appropriate.

The embedded scan chain insertion approach would be used when the macrocell is to be supplied to an OEM who is not an ARM licensee and thus does not have access to the IP. The ARM partner will synthesise the ARM966E-S with scan chains inserted and generate ATPG patterns. These will be supplied to the OEM together with a black-box synthesis model of the macrocell. In this case, the ARM966E-S scan chains must remain as separate scan chains within the SoC.

Full scan insertion on the ARM966E-S achieves 96.5% test coverage with a 7-10% overhead. The missing coverage is due to untested logic in the shadows of the RAMs. This shadow logic can be tested either by adding a scan chain wrapper to the RAMs or by using an ATPG tool that can generate vectors using the RAM as a capture/update element. Using the latter approach (which is preferred because it has no effect on performance), the test coverage can be increased to 99.6%. The performance impact of scan insertion was in the range of 0 to 5% depending on whether the target library included scannable flip-flops.

3.2 INTEST Wrapper

The ARM966E-S core is a register based scan tested

design which does not have a register on each I/O on its boundary. Correspondingly, scan vectors generated for the ARM966E-S core in isolation for use when the part is to be embedded in a SoC, are unable to provide test coverage for the regions of logic between the input pins and the first scanned register within the design, and between the last scanned register in the design and the output pins. This is referred to as the internal test shadow. Furthermore, in the SoC the logic between the pins of the ARM966E-S core and the first registers in the SoC also form a test shadow, the external test shadow. The OEMs synthesizing with the black box timing model are therefore unable to generate ATPG vectors to cover this SoC test shadow.

To solve the problem of testing the shadow logic, a synthesis option enables an INTEST wrapper to be included with the ARM966E-S. This wrapper provides a scan chain accessing all of the macrocell pins but which is switched in only during scan testing. The wrapper will be used by the ARM partner to generate vectors that cover the internal test shadow and by the OEM to generate vectors that cover the external test shadow. Figure 3 shows how the INTEST wrapper can be used to test gates e1,e2 and e3 in the external test shadow and gates i1 and i4 in the internal test shadow.

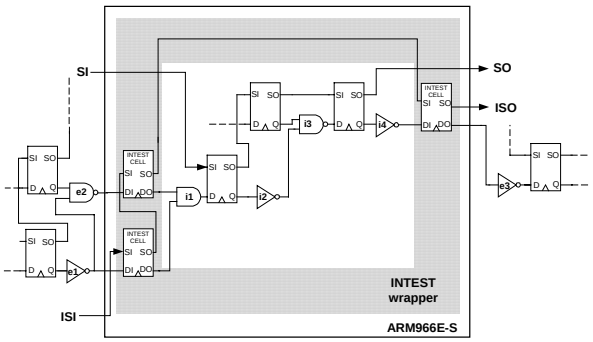

Figure 3: INTEST wrapper testing shadow logic

When an SoC is being developed by an ARM partner whose ATPG tool is allowed visibility inside the macrocell, this wrapper can be optionally removed, so removing a multiplexor gate delay.

Adding the INTEST wrapper to ARM966E-S increases the gate count by about 5.5% but does not impact the maximum internal operating frequency of the device. It does however add a multiplexor gate delay to the input and output paths, resulting in a 0-2% performance impact on the I/O paths. The wrapper achieves close to 100% test coverage of the internal and external shadow logic, which would otherwise be untested.

3.3 Memory BIST

The full scan approach to testing of the core logic on ARM966E-S core meant that a different approach to testing of the embedded RAM blocks was needed which preferably avoided the need for a functional test mechanism. The natural way to do this was to develop an embedded memory BIST controller. However, because ARM processor cores are licensed to multiple semiconductor partners, it was realised that some partners would wish to implement their own BIST algorithms, tailored to their particular RAM implementation. For this reason, ARM's synthesisable products containing RAM are supplied with BIST *wrappers*. The ARM966E-S core's Instruction and Data RAMs are configurable by the system integrator so that the device can be tailored for each particular application. The BIST wrappers are automatically synthesised to the required size. This unified and carefully controlled BIST wrapper technique means that the SoC designer no longer needs to worry about the embedded memory test problem and can concentrate on other system design issues.

A particular feature of the ARM966E-S core's memory BIST implementation is that it is fully controllable via the ARM programmer's model, using coprocessor instructions. This has the added benefit of allowing on-line BIST to be run. Both the start address and test length are programmable to allow small segments of memory to be tested. Hooks in the RTL ensure that the interface to the programmer's model is maintained, even if the 6N BIST algorithm supplied as default with the ARM966E-S is replaced by a manufacturer-specific algorithm.

The two BIST wrappers on the ARM966E-S incur a total area overhead of 4%. The RAM interface RTL is structured so that the BIST paths into the RAM do not cause any degradation in device performance.

4 Debugging Embedded Systems

Being able to debug the application (both software and hardware) is another important issue in the design of embedded systems. The traditional method of debugging an embedded system design has been to use an ICE (In Circuit Emulator). However, trends in system integration and increases in system clock frequency have diminished the usefulness of an intrusive ICE. In many applications, the processor, memory and peripherals are all integrated on the same device, making access for ICE difficult or impossible. The solution to this problem is to provide an embedded debug capability.

In many applications, an embedded debug unit that is accessed via the IEEE 1149.1 port is sufficient. This

allows breakpoints to be set, instructions to be single-stepped and registers to be read, all via the small pin count TAP port. Using suitable software, debug information can be displayed on a host computer. ARM has integrated this kind of embedded debug unit into products such as the ARM7TDMI™ core. For more critical real-time applications, additional hardware is needed for instruction and data tracing and to enable real-time monitors to be implemented.

4.1 Embedded Instruction and Data Tracing

In real-time systems, debugging information must be collected without stopping the processor. Adding trace hardware is an effective way of achieving this. It is essential that debugging of a system can be carried out at its full rated frequency and this is only possible if the trace mechanism is integrated on-chip. For example, ARM's Embedded Trace Macrocell (ETM) can be interfaced to both the ARM966E-S core and the ARM920T core to provide full instruction tracing and variable bandwidth data tracing. The data trace bandwidth depends on the number of pins that can be allocated to trace on the SoC. The minimum number of pins for the real-time trace port is five, while wider versions are possible, to provide greater data trace capacity. A small FIFO inside the ETM allows full instruction tracing to be achieved using the narrowest port under worst-case conditions. The ETM has been designed from the outset to be tested using full scan and high fault coverage has been achieved, with virtually no logic in a test shadow.

4.2 Debug Monitors

Debugging of real-time systems requires breakpoints that do not cause the processor to stop, in order to preserve the real-time behaviour of the system. For example, ARM has developed a new mode for the on-chip breakpoint hardware called Monitor Mode. When this is used in conjunction with a real-time debug monitor, it allows for foreground tasks to be debugged while allowing continued servicing of critical real-time interrupts. ARM has developed an open protocol for an RTM (Real-time Monitor), allowing hardware and software breakpoints to be freely mixed.

5 Test strategy comparison

Although very different test strategies have been used for the ARM920T and ARM966E-S processor macrocells, both achieve acceptably high test coverage.

The ARM920T uses the AMBA bus-based test approach together with DFT features built in to the cache and MMU arrays to achieve high test coverage with a low vector count and low logic overhead. Although functional test vector generation and fault simulation are time consuming, these AMBA vectors are then reusable on all ARM920T-based designs.

In contrast, the ARM966E-S uses a combination of full scan insertion and ATPG for the macrocell core and RAM BIST to achieve a similarly high fault coverage. Scan insertion does however incur a 7-10% area overhead. The INTEST wrapper allows third parties to test the shadow logic between the ARM966E-S and the rest of the SoC whilst maintaining IP protection. The INTEST wrapper adds 5.5% area overhead and impacts performance by 0-2% in those applications that require it.

6 Conclusions

It has been shown that both functional test and full scan with memory BIST are approaches to CPU macrocell test that are capable of producing high fault coverage.

Just as both full custom and fully synthesisable design styles are capable of producing processor cores with competitive performance, power consumption and die size attributes, these two different approaches to test can be equally successful.

In both cases, the tradeoffs involved in each test strategy need to be carefully considered during the design process. For embedded designs, particular care must be taken to ensure good test coverage of those parts of the design that are not covered either by ATPG tools or by functional patterns. The INTEST wrapper technique described here is expected to fit well into a P1500 compliant test environment.

7 References

1. P. Harrod, "Testing Reusable IP – A Case Study", 1999 IEEE International Test Conference, Atlantic City, September 1999.

2. D. Flynn, "AMBA: Enabling Reusable On-Chip Designs", IEEE Micro, Vol. 17, No. 4, July/Aug. 1997, pp20-27.

3. D. Gwilt, "Silicon Test in the ARM920T Microprocessor", Proceedings of Microprocessor Test and Verification Workshop (MTV '99), Atlantic City, 1999.

4. A. Burdass, G. Campbell, R. Grisenthwaite and R. York, "Testing Embedded Synthesizable IP – A Case Study", Proceedings of TECS 2000 Workshop, Montreal, May 2000.