The T9000 Transputer
A Practical Example of the Application of Standard Test Techniques

Graham Frearson, Inmos Ltd, Bristol

Abstract

This paper outlines a practical approach adopted to integrate current Design–For–Test methodologies into a VLSI chip containing several processing elements and embedded RAM. It is shown that optimising design styles to suit the application has resulted in the need for more than one test strategy. Global design rules concerning clocking and reset to allow scan and behavioural test are discussed. Using the T9000 Virtual Channel Processor as an example, the integration of some of these techniques is explained.

1 Introduction

There have been many approaches towards an integrated test methodolgy, each with advantages and disadvantages depending on the design and functionality of a particular chip. A single test rule can encompass the entire chip, enabling the use of ATPG software tools to generate the entire functional test suite. However, with the move to integrate ever more subsystems on a single die such as custom peripheral I/O around an ASIC type core, a single test rule can either compromise parts of the design or else the test coverage itself may be compromised.

When chips are formed from multiple blocks, (on the T9000 transputer there are essentially seven major subsystems), there is a conflict between the need to generate efficient vectors that verify correct manufacture (structural tests) and the very real need to verify functional or systems related operation. This places very different design constraints on the process of DFT. The integration of different test methods, and a global test strategy with minimal impact on design functionality, was fundamental to ensuring that the exercise of generating test vectors was as smooth as possible without compromising test coverage.

The first part of the paper gives an overview of the T9000 and its design methods. The third chapter explains some of the global issues that affected the standard Design For Test (DFT) techniques which could be implemented, the contraints these DFT strategies put on the design and how they are integrated at the chip level. Chapter four looks specifically at the Virtual Channel Processor (VCP) and the problems of integrating test with design functionality. Finally the conclusion gives a summary of the overall effectiveness of the test strategies adopted.

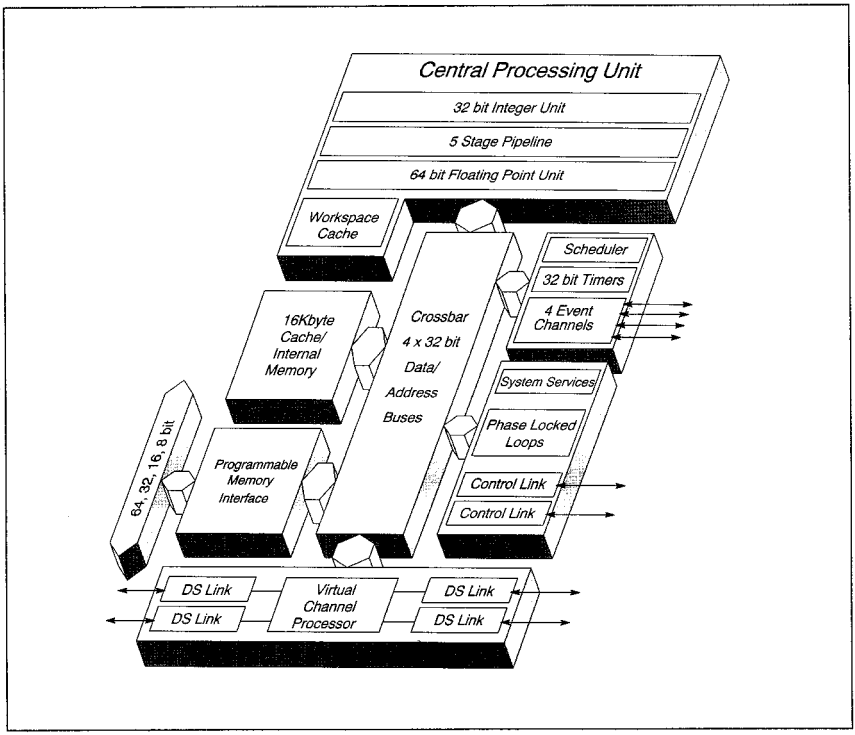
2 T9000 Overview

The T9000 is the latest member of the transputer family and integrates a 32–bit integer dual–pipelined processor, a 64–bit floating point processor, 16Kbytes of cache memory, a communications processor and four serial communication links.

Figure 1. shows the internal block diagram of the T9000. The chip contains seven major subsystems, each of which contains several minor subsystems as shown in the diagram. The major subsystems pass control signals via retimed interfaces. Data flow to external memory and cache rams is via the 4x32 bit crossbar arbitrating between external memory accesses, cache–bank accesses and cache–line refill operations. A more detailed overview of the T9000 subsystems is provided elsewhere [1].

The size of the design meant that meant that a range of different design styles, tools and therefore constraints were used. A problem is that often the constraints required to make a chip testable are incompatible with the design specification. Full scan insertion offers the most effective test coverage, but the area, performance and power overheads can prove too great.

0-8186-3502-9/93 $3.00 © 1993 IEEE



The design was concurrently engineered using top–down and bottom–up approaches. The top–down designs started with an architectural definition or high–level behavioural description in VHDL. The high–level behavioural models were then gradually refined until either a register level description was derived or until Synopsys synthesis tools were able to translate the design directly into the standard cell library. The latter approach could then be placed and routed using Cadence software. This had the advantage that floorplanning could be carried out early in the design, and any changes to the behavioural model could be implemented and the logic re–synthesised very quickly. Where the behavioural description was refined to a register transfer type model, a bottom–up approach was used to connect custom datapath elements to the top–down control logic. This had the advantage that the custom datapath elements could be laid out efficiently by hand, characterised in circuit simulations, then modelled as a behavioural unit.

Another advantage of keeping parts of the design as two separate blocks – control and datapath, meant that microcode ROMs could be implemented efficiently. The use of microcode has many advantages in a design :

1. ROM and PLA contents can be generated late in the design cycle allowing for subtle changes as the design progresses.
2. Once the approximate size of the ROM is known, all timing parameters for microcode state transitions are a constant, allowing easier characterisation of datapath elements.
3. The testing of a ROM is a well understood process that requires relatively few, simply generated, vectors for the overall functionality compared with random logic.

In some cases the top–level behaviour was described in OCCAM, the native language of the transputer. This description could then be converted into VHDL to enable the synthesis tools to take the design further towards implementation, with the final gate–level model simulated against the OCCAM model to prove correctness. Where some parts of the design already existed, such as the Data–Strobe links, VHDL models were created to reflect the current design so that high–level simulations could be used to verify interface behaviour.

In the final stages of verification, all the major subsystems were simulated together on a hardware accelerator. This performs the acid test of the interface specification by checking physical connectivity, protocol behaviour, and subsystem state dependencies and interlocks. An IKOS hardware accelerator was used which had the capacity to model virtually the entire T9000 (the cache rams were modelled separately), and had a sufficiently fast simulation time to enable millions of cycles of code to be run on the T9000 as part of the final design verification prior to fabrication. This was an important stage and emphasised the need to run top–level simulations as early as possible in the design cycle.

3 Design For Test Integration

In order to successfully integrate a variety of test methods across the chip, global standards had to be imposed which unified the boundaries of the subsytems involved. In test mode the part was to be entirely synchronous and so the first area to address was the clock architcture.

3.1 Clock Architecture

The T9000 has a mixture of custom and ASIC styles, the majority being synchronous. Custom logic has the advantage that it has the flexibility to change clock regimes to maximise cycle times, use precharge cycles or catchup slow cycles by delaying clocks. ASIC design usually has a rigid clocking regime dictated by the type of latches used. For purposes of test, the global test clock architecture is conformant with the areas of synthesised logic . In order to integrate the two different clock regimes when in test mode the global clocking had to be defined as this was based upon a feature peculiar to the T9000. In its normal mode of operation, the T9000 has a four phase internal clock sourced by the on–chip phase locked loop (PLL), clocked externally from a single 5MHz clock.

The relation between the external clock and internal phase is analogue in nature and so is difficult to model using logic simulation, the exact delays depending on the process. When stable, the phase of the PLL, and hence the internal system clocks, maintains a fixed relation to the external clock. As the clock rate is increased however, the fixed analogue delay may cause the internal clocks to stradle a clock boundary relative to the tester clock. The degree to which this occurs is again process dependent. There is a also a significant 'start–up' period during which the PLL gradually 'locks on' to the external clock, which is in the order of tens of milliseconds.

To make synchronising the part to a tester derived stimulus faster and more reliable, a 'times one mode' is used which bypasses the PLL and generates all internal system clocks directly from clocks provided by the tester. This has several advantages :

- The start–up period of test pattern execution is quicker as the PLL lock–on time is avoided.
- The static operation of the part can be tested simply by stopping the clocks.
- The input stimuli can be sourced with a fixed, known relationship to internal clocks, and the internal clocks are guaranteed to be running at the same speed as the tester (the PLL speed is frequency programmable).
- The part can have critical timing paths measured by changing the period of the tester clocks for the relevant cycle using on–the–fly timeset switching methods.

The clocks used throughout the chip are all derived from the distributed PLL (or times–one mode) clocks and have the same meaning in each subsytem. This is so that each subsystem has

the same start and end of cycle time which is important for interface signals that have to cross subsytem boundaries. If signals internal to a subsytem are generated on non–standard clock edges, they must be retimed before they are passed across the interface. There are four main clocks :

1. Capture Clock – This marks the end of the logic evaluation time.
2. Release Clock – This marks the start of signal propagation and logic evaluation.
3. Shift Clock – The clock which is synonomous with the Capture Clock but marks the end of a shift cycle. It only runs under test logic control.
4. Update Clock – A clock coincident with the Release Clock but under test logic control.

These four internal clocks constitute a two–phase clocking scheme, although their meaning is modified according to the global mode of chip operation. The global mode has three main categories:

1. Reset – During reset certain clocks must run to initialise the part into a known state.
2. Scan – Scan and certain scan type tests require a different sequence of clocks to generate the required test conditions for scan tests and setting up other test conditions.
3. Functional – The part defaults to its 'normal' mode of operation having performed a reset. This mode uses a combination of the above modes.

Again a unified strategy exists which means the global mode is common to all subsysyems. Reset is controlled either by an external reset pin or via a software reset. Functional operation occurs immediately after reset by default, or on return from any special test conditions. Scan operation is invoked by a separate subsystem – the Test Access Port (TAP). Each clock mode has a priority – the highest being reset and the lowest being functional.

A typical latch that enables all three modes to be differentiated is shown in Figure 2.

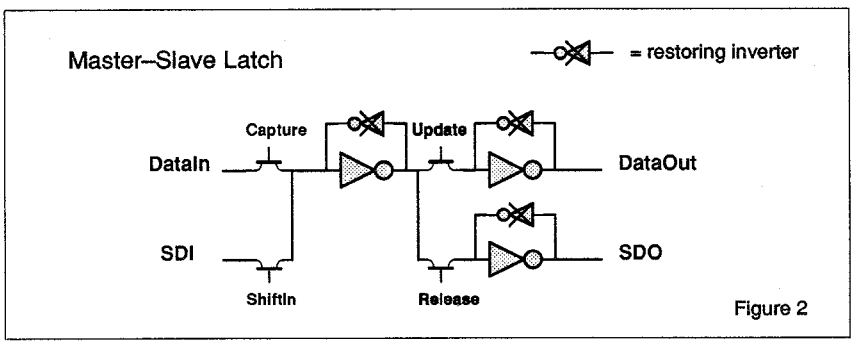


Figure 2

3.2 Reset Conditions

Reset has several operations to perform in order to make the chip testable. The first is to ensure that on power–up, all metastable states are cleared and any form of contention that could cause shorts between internal power rails is removed. All this should happen within one cycle of the application of the reset signal.

The reset function is performed using the network of scan chains that exist in all the major subsystems and more specifically in the elements that require initialising. For the majority of initialisation, shifting zeros through the scan chains is sufficient to leave the logic in a known condition. This has the advantage that no additional asynchronous reset transisitors have to be included in the latch cells. There are exceptions to this however. If there is a possibility of large tri–stateable bus drivers being in contention, whose enables are sourced from a scan latch output near the end of a long scan chain, it may take several hundred cycles before the drivers are turned off, by which time it may be too late to prevent the collapse of the internal supply and latchup. In these circumstances an asynchronous reset is provided. Another exception stems from the fact that on the

T9000 there are four levels of reset. Some forms of reset only affect certain subsytems, and if one of these is implemented by scan, it must be guaranteed that a resetting subsystem does not randomly toggle an interface signal to a functional subsytem. Again this is achieved by providing an asynchronous signal that holds all interface signals static during reset.

3.3 Scan Test

A proven effective technique for testing large areas of random logic is scan. All the 'random' logic on the T9000 is generated using Synopsys' synthesis tool and so it was decided to incorporate scan test into all these blocks. A scheme based on the IEEE P1149.1 Test Access Port was adopted to provide the necessary control. This had the advantage that there was a specification already in place to which all subsytems could adhere.

The TAP is implemented as per the IEEE P1149.1 specification [2] in terms of its states and control, with some minor modifcations to enable performance testing without the need for area–consuming boundary–type latches. The performance test instruction modifies the controller states to provide a shift/update then capture on consecutive cycles, which also allows debug of the design by allowing the current state of any scanned logic to be serially shifted out, analysed and shifted back in to resume running without destroying the current state of the machine.

It rapidly became apparent however that embedding fully scanned logic into a partially–scanned environment would create some major headaches for both designers and the ATPG software. The in–house ATPG software, in common with some forms of commercially available software, needs access to all input and output bristles of the module in order to generate a test vector. This is fine so long as the module is tested in isolation, but once it is embedded in a larger circuit, stray signals that may be undefined during test must be accounted for. To cope with signals from non–scanned blocks entering a scan region, a latch similar to a boundary latch is used whereby for normal functional operation the latch is bypassed but during scan test, the latch sources the signal from data held within. In a similar way, to ensure adequate fault coverage, all signals passing out from scanned logic that were not direct scan latch outputs go via a similar latch. A typical 'scan–node' is shown in Figure 4.

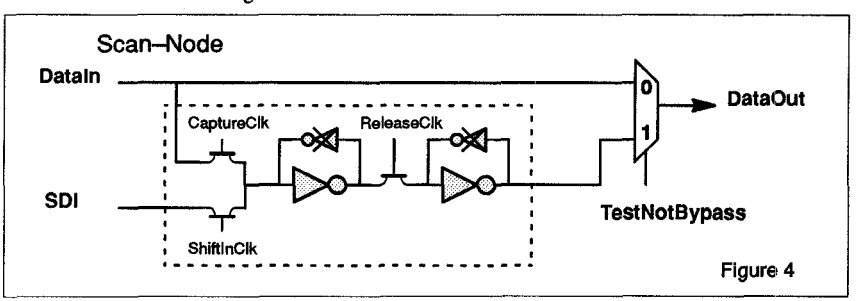


Figure 4

Because scan test is the primary production test for structural defects, the method for getting scan vectors in and out of the chip was investigated at an early stage. Many commercial VLSI testers have a serial scan option which enables scan data to be held more efficiently in vector memory, allowing it to be serially shifted out to a single pin such as TDI. The problem with this, however, is that each scan vector takes N cycles to shift in, where N is the *total* number of scan latches in the entire design. An alternative approach is to load a set of shorter scan chains in parallel.

On the T9000 the simultaneous loading of mutiple scan chains is achieved by adding a 'Parallel Scan' instruction to the TAP. When this instruction is loaded, normal functional input pins are used as scan chain inputs and normal functional outputs are multiplexed to scan chain outputs. Using this method, data can be shifted in and out of 30 scan chains simultaneously. This not only makes efficient use of tester memory but has the added advantage of reducing the vector depth

required for structural tests, and gives a corresponding reduction in test time. The time to reset the part is also reduced as the scan chains are shorter and hence require fewer cycles to flush with zeros.

To aid design debug, systems that are not scanned have scan–nodes on critical signals such that under TAP control, the functional operation of the T9000 can be stoppped and its state scanned out. This has been particularly useful in isolating speed–critical paths. Similar scan–nodes are placed on the periphery of blocks which generate a large number of control signals. Using times one mode clocks, certain cycles can be speeded up to stress these paths. The insertion of scan–nodes then provides a much finer granularity in identifying the position of critical signals.

4 Testing the Virtual Channel Processor

With the above test architecture adopted throughout the chip, it was left to each subsystem to make best use of these features. In some cases an ad–hoc approach to testing has been found to be more appropriate, usually when area constraints were tight. An example is the cache–ram. They each have a set of ROM patterns at the far end of the bit lines from the sense amps. This enables a very quick check of address decoder and bit–line integrity to be carried out. By setting a test mode via a TAP scan chain, the arbitration logic of the crossbar can be bypassed, enabling the cache's data and tag regions to be accessed as a standard RAM array (although some pipelining is still involved). Another mode enables Vdd bouncing of the RAM array for testing data retention.

The CPU's instruction decoder/grouper has BIST circuitry that generates weighted pseudo–random instruction groups, and then compresses the response into a signature allowing more than a million instruction groupings to be tested at full system speed using a minimal amount of tester vector memory.

A subsystem which makes full use of integrating the various test techniques is the Virtual Channel Processor (VCP) and this is discussed in detail next.

4.1 VCP functional overview

The VCP is essentially a two way DMA engine for the transfer of asynchronous messages between two processors. The three main components are an input DMA engine controlled by the Packet Input Controller (PIC), an output DMA engine controlled by the Packet Output Controller (POC), and a command processor (CPROC), as well as a queuing mechanism for output messages, an output request coordinator and a block transfer unit. The three main components are microcoded processors which all act independently, yet maintain common information about the state of a particular message using a 'Virtual Link Control Block – VLCB'. Each processor has an interlock mechanism that prevents more than one process from updating the VLCB at a time.

The PIC and POC use microcoded control of their relevant datapaths for passing information *about* the message around and each controls a DMA engine for moving message *data* to and from main memory. Each DMA engine in turn is split into control and datapath elements.

The CPROC interfaces to the CPU to interpret the relevant message commands such as 'IN', 'OUT', 'EnableChannel' and 'DisableChannel'. Generally, once the CPU has passed an instruction to the VCP, the process which is currently running deschedules and it is up to the PIC or POC to reschedule the process by communicating with the Scheduler.

A 'packet' (a message or part message of 32 bytes) which arrives before an 'IN' instruction has been performed on that channel is placed in an area of memory known as the packet buffer. If there is a packet already in the packet buffer when the CPROC receives an 'IN' instruction on that channel, the Buffer Transfer Unit (BTU), does a block move to transfer the packet data from the buffer to an area specified by the process requesting the INput.

Each of the four links can output a message that is either at low or high priority, and either an acknowledge or data packet. There can therefore be 16 lists of messages queued to send at any one time. The lists are linked together via the VLCBs of the particular channels in the list with

the front and back pointers of all 16 lists being held in the 'Queues' block. The Output Request Coordinator decides which message to pass to the POC for the next output and it is up to the POC itself to dequeue the message and unlink the VLCB from the list.

The mode of operation of the VCP is determined by a set of configuration registers which can be set up at initialisation or reconfigured during run time.

4.2 ROM signature analysis

The three microcode ROMs take up a significant proportion of the area of the VCP, and their density makes them more process sensitive to structural defects, so a pseudo–BIST scheme was adopted for each ROM. The ROMs are 'folded' into a four–way array with the outputs being selected via a multiplexer. The multiplexer is switched via a set of conditional inputs which arrive late in the evaluation cycle to enable four–way branching of the microcode, with the remaining address bits being derived from the current state.

To provide adequate test structures for the ROMs, a scheme had to be devised to take into account the additional multiplexing and address generation. The method used is shown in Fig 6. The signature latch is an extension to the normal latches for the outputs from the ROM, built up as a Multiple Input Signature Register (MISR). This will typically have in the order of 60 inputs. When the TAP instruction 'SignatureTest' is loaded, a signature clock is enabled which latches the function of current ROM output XORed with the adjacent bit in the previous cycle. The 'SigTest' signal also breaks the scan chain to feed back the shifted ROM output data with the appropriate polynomial function applied to provide a maximal irreducible sequence with the minimum of aliasing. The test address for each ROM is generated using an LFSR with the SigTest signal switching the address to be sourced from the state to LFSR outputs.

The above scheme has two main advantages over a fixed BIST scheme that terminates itself after a given number of cycles. The first is that the test logic is independent of the height of the ROM – again useful when the number of ROM states may change late in the design. The second is that for debug and process characterisation, the address for a single location can be shifted in, the ROM data captured in the output latch and then shifted out. This direct addressing mode can be useful for the quick location of structural defects. One further advantage for design debug is that any given microcode state can be serially shifted in to set up a specified condition in the datapath before returning to normal functional operation.

4.3 Control Logic Scan

Control logic in the VCP which is not microcoded was mastered in VHDL then synthesised by Synopsys before being auto–placed and routed. A post–synthesis, pre–layout step was to chain all the latches in the logic together to form a scannable logic block. Larger blocks of logic were split amongst multiple scan chains. An in–house ATPG software tool based on the PODEM algorithm [3] was used to generate test vectors for such logic but this is where several problems were experienced.

The ATPG software assumed it had control over all primary inputs to the design as is common with commercially available software. This could be done by simulating the logic in a harness, tying primary inputs to known values, but when the logic is embedded with the VCP, those inputs are no longer accessible. A solution to this is to place a scan–node on all primary inputs and outputs of such areas of logic. However, if the logic is butted against a similar scanned logic block, it seemed wasteful to include scan–nodes on signals which were already direct latch outputs or inputs, and so modifications were made to the ATPG software. This allowed test vector generation of larger areas of logic, taking into account multiple scan chains across two or more modules, buffering or reset logic in series with the scan chain, and the inaccessibility of direct clocks when they were hidden by local clock buffers.

4.4 Datapath registers

Where datapath elements pass data sourced from memory read or write operations, it is not usually necessary to scan the datapath, so the register elements can be optimised to be both speed and area efficient. Simple behavioural tests of their function cover most manufacturing (structural) faults. However, a few areas of the datapath – such as the CPROC datapath – which contain a large number of constants and multiplexers, are scan tested by a smaller set of scan vectors than would be required in a functional (behavioural) type of test.

4.5 Non–scan test features

For some parts of the design, the extra delays introduced by scan elements are too great. Other parts could not accomodate the constraints scan–based test imposed. Scan requires synchronous designs. Where non–synchronous techniques are required to push the performance of a circuit, such as in the Data–Strobe links, it may be impossible to use full scan. In these remaining areas, behavioural tests are relied on to provide adequate test coverage. Test stimuli are then run on the IKOS fault simulator to show the coverage that has been achieved, iterating the stimulus until the desired coverage has been achieved. Such a non–scanned area is the configuration block of registers. To fascilitate testing, these are all readable *and* writable even though the specification may only be read (or write).

Sample latches are placed where signals stradle two different frequency clock environments, such as happens at the link interfaces. Where possible, different frequency clocks are all derived from a single source which is ultimately the system clock and so is synchronous to the tester. The only other sources of clocks are the Data–Strobe links' clock regeneration, or the Data–Strobe links' high–speed output clock. The latter is normally derived from a separate PLL. In test mode, in a similar way to the system PLL, the link PLL is bypassed and the clocks are derived from the tester. This enables the two sets of clocks to be skewed against each other to detect gross errors in the sample latches interfacing the signals between the two systems.

The Input DMA has a set of registers to allow four packets of data, one from each datalink, to be stored local to the VCP prior to writing it to the cache. These 4x32 byte registers are directly accessible via the datalinks. Serial data into the links during a functional test is analogous to scan test, so these registers are area optimised for functional operation only.

5 Conclusion

The T9000 is a complete microcomputer on a chip. As such it encompasses a CPU, embedded cache RAM, a sophisticated control and monitoring system, an FPU and more unnusually, a communications processor. These units are normally a challenge to integrate at the system level because of the complex interfaces, protocol behaviour and interlocks required as they manipulate data to and from internal cache and external memory. To unify this system for test adds an extra dimension as embedded ram, analogue PLL's, asynchronous and scanned logic all require different test methods.

On the T9000, certain rules had to be imposed in order to integrate test at a system level. These rules concerning clocking, reset, and scan test based around a TAP conforming to the P1149.1 specification, ensured that a global strategy has successfully unified a number of standard design–for–test techniques on a single chip.

6 References

[1] The T9000 Transputer Reference Manual. 1993 edition

[2] IEEE Standard Test Access Port and Boundary–Scan Architecture, 1990

[3] Goel, P., "An implicit Enumeration Algorithm to Generate Tests for Combinatorial Logic Circuits", IEEE Trans. Comput., vol. C–30, no. 3, March 1981, pp. 215–222