IEEE 1149.1 Compliance-enable Pin(s): A Solution for Embedded Microprocessor-based Systems Debug and Test

Gustavo R. Alves
Department of Electrical Engineering
ISEP
Rua de S. Tomé
4200 Porto, Portugal
e-mail: galves@dee.isep.ipp.pt

José M. M. Ferreira
Department of Electrical Engineering
FEUP
Rua dos Bragas
4000 Porto, Portugal
e-mail: jmf@fe.up.pt

Abstract

Microprocessor-based systems are usually debugged with the help of in-circuit emulators and logic analysers. However, these traditional debug tools can not be used when the microprocessor is an embedded core. To overcome this problem we propose the use of an embedded debug and test infrastructure and the IEEE 1149.1 compliance-enable mode to implement the basic functionality provided by an In-circuit emulator and a logic analyser.

Keywords: embedded microprocessor, embedded debug and test infrastructure, IEEE 1149.1 compliance-enable mode, Boundary Scan Test.

1. Introduction

Boards with a microprocessor are usually debugged using an In-circuit Emulator (ICE) and a logic analyser. These tools are able to assist not only in the detection and analysis of problems faced during the prototype validation but also during the integration of software (e.g. the program run by the microprocessor) with hardware, generally considered to be the most troublesome phase. Unfortunately these traditional debug tools can not be used when the microprocessor is an embedded core as there is no physical access to it.

The concept of Design for Testability (DfT) is being adopted by an increasing number of designers in order to solve, earlier in the design phase, the testability problems faced during the prototype validation and production test. Boundary Scan Test (BST) or IEEE 1149.1 standard [1] is also now widely used and supported by several silicon manufacturers. Although BST is primarily used for the structural test of Printed Circuit Boards (PCBs), it can be used for others applications like functional test, circuit emulation or debug [2, 3, 4, 5, 6].

On this paper we describe how to use an IEEE 1149.1 compatible debug and test infrastructure and the IEEE 1149.1 compliance-enable mode to implement the basic functionality provided by an ICE and a logic analyser for debugging embedded microprocessor-based systems. Our approach eliminates the need for physical access, imposed by these traditional debug tools, although it requires the system-on-a-chip to have a standard Test Access Port (TAP) and one or more IEEE 1149.1 compliance-enable pin(s).

2. The debug and test requirements

To write down the requirements for the debug and test infrastructure we first identify the basic functionality provided by ICEs and logic analysers. These tools are used at the PCB level and they are virtually design independent e.g., their use does not imply that the system should be designed in a particular way. However at the chip level, the inclusion of an embedded debug and test infrastructure requires that all implications and aspects related to the interaction between the functional and test logic should be discussed and planned earlier at the specification phase. A brief study shows that when using an ICE it is possible to reset, single step, stop on break point conditions or run the microprocessor [7]. When the microprocessor is stopped it is possible to examine and change the internal registers contents and the internal and external data memory contents. The microprocessor may run the program from an internal or external memory. The logic analyser provides the capabilities needed to observe the program execution flow in real time [8]. Characteristics like the maximum acquisition speed, the memory size, the number of data

inputs, the number of inputs for synchronisation and the trigger options are taken into account when selecting the right logic analyser. Based on the information gathered it is possible to specify the following basic debug modes:

- Start mode.
- Single Step (SS) mode.
- Break Point (BP) mode.
- Real Time (RT) mode.

Start mode

In this mode the microprocessor is in the reset state and it is possible to download the program into the program memory. As the microprocessor is stopped it is also possible to read and change the internal registers and both data memories contents. To implement these capabilities the debug and test infrastructure must:

1) provide control to the embedded microprocessor reset line.
2) provide access and control to program memory, internal registers and both data memories.

SS mode

In this mode it is possible to single step the microprocessor. There are two options for what is considered to be a *single step*: provide *n* clock cycles to the microprocessor, where *n* is specified by the user, or provide clock cycles until the microprocessor program counter changes it's current value. When the microprocessor is stopped it is possible to read and change the internal registers and both data memories contents. To implement these capabilities the debug and test infrastructure must:

1) provide control to the clock line and monitor the address (and possibly the control) bus in order to detect changes in the microprocessor program counter.
2) provide access and control to internal registers and both data memories.

BP mode

In this mode the user may specify a BP condition and run the microprocessor program until the condition becomes true. Although there are many possible BP conditions the following list represents a minimum set:

- BP on program address
- BP on program data
- BP on external data memory address
- BP on external data memory data

When the microprocessor is stopped it is possible to read and change the internal registers and both data memories contents. To implement these capabilities the debug and test infrastructure must:

1) monitor in real time the address and data bus in order to detect the BP condition.
2) provide control to the clock line in order to stop the microprocessor.
3) provide access and control to internal registers and both data memories.

RT mode

In this mode the microprocessor is in a free-running state and it is only possible to monitor and sample the signal lines. A brief survey on several logic analysers trigger options shows that it is generally possible to define an acquisition protocol where the trigger or transition conditions are specified by the user and where the following states commonly exist: start, look for condition (or idle) and store sample. The conditions may be the ones already supported in the BP mode. To implement this capability the debug and test infrastructure must:

1) monitor and sample in real time the address and data bus in order to detect the transition condition.
2) store the samples in a dedicated memory whose contents may be shifted out.

3. The embedded debug and test infrastructure

The scheme presented in figure 1 allows the implementation of the debug and test infrastructure, in the embedded microprocessor core, which may be seen as a special BST infrastructure. From the list of requirements presented at each one of the debug modes (*Start*, *SS*, *BP* and *RT*) it is possible to identify the requirements for the embedded debug and test infrastructure:

a) provide control to the embedded microprocessor reset line.
b) provide control to the embedded microprocessor clock line in order to single step, stop or resume the program execution.
c) provide access and control to program memory, internal registers and both data memories.
d) monitor in real time the address and data bus in order to detect the BP or the transition condition.
e) store the samples in a dedicated memory whose contents may be shifted out through the BS chain.

To implement requirements a) and b) the debug and test infrastructure of the embedded microprocessor must control the reset and clock lines. The control must be independent e.g., when the values present on the reset and clock lines are controlled through the debug and test infrastructure the values of the remaining lines may be controlled by the microprocessor functional circuitry. Also it must be possible to resume the microprocessor normal activity by placing the cell associated with the clock line in a bypass mode, without causing any spikes on the transition that could have hazardous effects. A simple circuit based on a Flip-Flop may be used to fulfil this precaution.

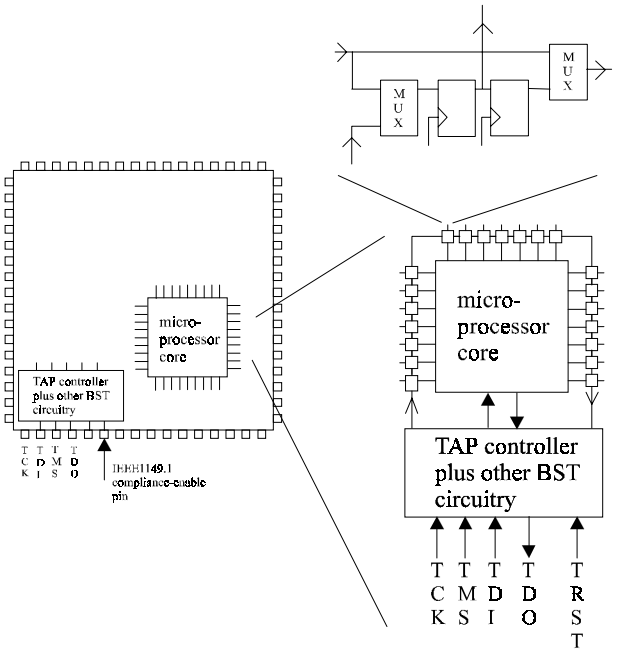

Fig. 1: IEEE 1149.1 compatible device containing an embedded microprocessor core with a debug and test infrastructure.

To implement requirement c) the debug and test infrastructure of the embedded microprocessor must have access and control to the program memory (internal or external to the microprocessor), internal registers and both data memories. The scheme presented in figure 2 allows the implementation of this requirement. The internal registers and memories (program or data) of the embedded microprocessor can be accessed through an internal scan chain. A private BST instruction can be used to place the internal scan chain into the TDI-TDO path of the embedded microprocessor. Other embedded memories (program or data) that are exterior to the microprocessor can be accessed through the BS register, using for instance the EXTEST instruction. If other embedded blocks have access to these memories then possible situations of bus contention should be prevented.

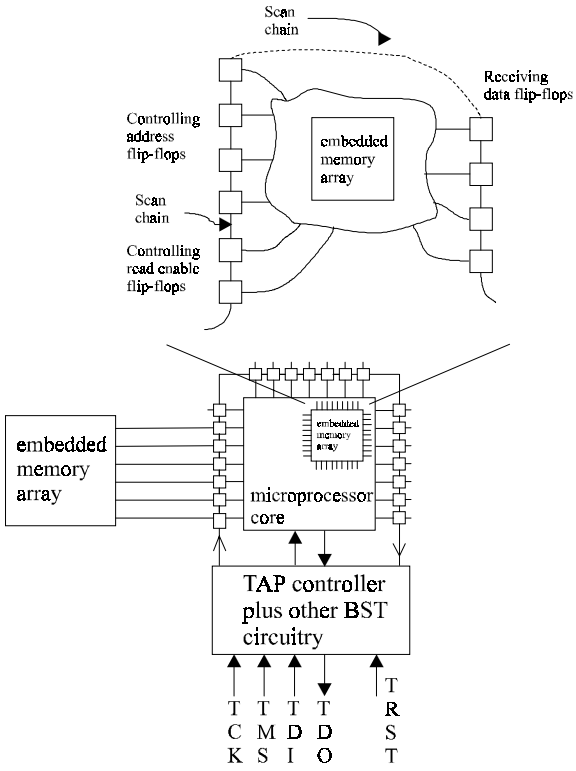

Fig. 2: Access to internal registers and embedded memories (inside or outside the microprocessor core).

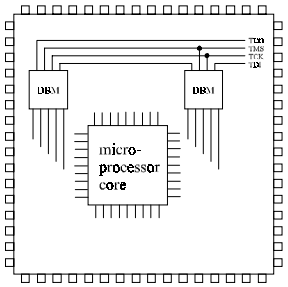

Fig. 3: Embedded DBMs connected to some of the microprocessor lines.

To implement requirements d) and e) it is necessary to embed, as illustrated in figure 3, the core of one or more devices called Digital Bus Monitors (DBMs) or 74SN8994 [9, 10]. The DBM contains a special register that allows users to scan in, through the TAP, a vector and a mask to be compared against the patterns sampled at the 16 input data lines. The vector represents the value expected and the mask contains information regarding which bits are relevant e.g., should be compared. When the comparison results true

an output pin, called Event Qualification Output (EQO) is activated. This pin is connected to the special cell associated with the microprocessor clock line so that it may be stopped immediately after a match is found. The number of DBMs that have to be embedded depends on the microprocessor address and data bus width. If several DBMs are needed then they should be daisy chained e.g., the TDO of the first one should be connected to the TDI of the second one, this way until all of them are connected together. The sampled values are stored in the DBM internal memory (identified as RAM in the block diagram presented in figure 4) whose contents can be scanned out through the TAP. This device also contains a Test Cell Register that allows to form a signature of the RAM contents, thus reducing the number of bits that have to be shifted out in certain cases.

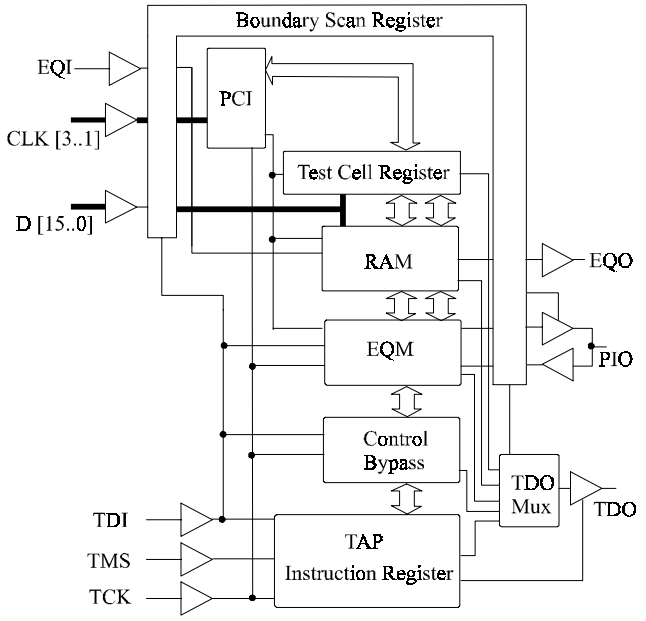

Fig. 4: Block diagram of the DBM.

To enable this solution the DBM core should be made available, as a debuggability building block, to all Electronic Design Automation (EDA) tools. The debug and test infrastructure formed by the BST infrastructure of embedded microprocessor and the embedded DBMs may be accessed through the device TAP, by adding one or more IEEE 1149.1 compliance-enable pins. When one or more steady-state logic patterns, called "compliance-enable" patterns, are applied to those pins then compliance to the IEEE 1149.1 standard is enabled and the device test infrastructure acts as a normal BST infrastructure. When compliance to the standard is disabled, the device TAP acts as an interface to the embedded debug and test infrastructure and this way instructions and data may be shifted in to it through the device TDI-TDO.

4. Conclusion

The inclusion of an embedded debug and test infrastructure in the original circuit implies an added cost expressed in additional silicon and the extra IEEE 1149.1 compliance-enable pins. The device TAP pins may be used for board production test and internal circuits debug. The embedded debug and test infrastructure includes the special microprocessor BST infrastructure, the embedded DBMs and possibly the BST infrastructure of other embedded blocks. The special microprocessor BST infrastructure allows independent control of the reset and clock lines and also access and control to program memory, internal registers and both data memories contents. The embedded DBMs allow monitoring and sampling in real time the address and data bus in order to detect the BP or the transition condition. The samples can be stored in a dedicated memory whose contents may be shifted out through the embedded debug and test infrastructure.

The proposed debug and test infrastructure can be first implemented in a PCB containing the DBMs and the circuit that will be integrated using the appropriated CAD tools and core libraries. The embedded BS chain is the same as the board BS chain and it can be accessed through the device TAP, as long as IEEE 1149.1 compliance-enable pins are used. This solution allows a straightforward approach to the embedded circuit debug and test.

References

[1] IEEE Standard Test Access Port and Boundary-Scan Architecture, Oct. 1993, IEEE Std. 1149.1 (Includes IEEE Std. 1149.1a).

[2] M. F. Lefébvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," in *International Test Conference*, pp. 294-303, IEEE Computer Society Press, 1990.

[3] Richard M. Sedmak, "Boundary-Scan: Beyond Production Test," in *International Test Conference*, pp. 415-420, IEEE Computer Society Press, 1994.

[4] K. Sievert, Y. Manoli, A. Both and R. Lerch, "On-chip Emulation and Debugging for Embedded Microcontrollers using the IMS ScanDebugger," in *European Design & Test Conference User Forum*, pp. 229-233, IEEE Computer Society Press, 1995.

[5] Andy Halliday, Greg Young and Al Crouch, "Prototype Testing simplified by Scannable Buffers and Latches," in *International Test Conference*, pp. 174-181, IEEE Computer Society Press, 1989.

[6] Jerry Katz, "A Case-Study in the use of Scan in microSPARC™ testing and debug," in *International Test Conference*, pp. 70-75, IEEE Computer Society Press, 1994.

[7] Bruce Erickson, "Selecting the Right Debugging Tool," in *Electronic Design*, pp. 83-98, Oct. 1995.

[8] Thomas R. Blakeslee and Jan Liband, "Real-Time Debugging Techniques: Hardware-Assisted Real-Time Debugging Techniques for Embedded Systems," in *Embedded Systems Programming*, vol. 8, nº 4, 1995.

[9] Texas Instruments, *SCOPE™ Logic Products: Application and Data Manual*, 1994.

[10] Lee Whetsel, "An IEEE 1149.1 Based Logic/Signature Analyser in a Chip," in *International Test Conference*, pp. 869-878, IEEE Computer Society Press, 1991.