How to future proof your JTAG debugging

This article introduces a JTAG Switcher, which allows you to connect JTAG TAPs of multiple chips with varying I/O voltage levels to one PCB wide JTAG connector.

Richard Copeman, Senior Support Engineer, Lauterbach UK

Introduction

JTAG debugging came about as a branch of the IEEE1149 Boundary Scan and Test specification. This original specification, ratified in 1990, describes a method of interrogating the pins of a device to ensure that it is electrically connected to other devices on a board. It replaced the bed of nails test which became less popular as boards moved to multi-layer PCB with surface mount devices.

Many enhancements to this original piece of technology followed over the years but when you mention JTAG, most people automatically think of hardware-based debugging. The system is not without its flaws for multicore or multichip debugging. This paper highlights one or two and introduces a new paradigm to solve them and to make your JTAG experience more future proof.

Background

For debug purposes, each Integrated Circuit (IC) implements a JTAG Test Access Port (TAP for short). The TAP is used to access several shift registers: one Instruction Register and several Data Registers. One of these Data Registers must be used to implement a BYPASS Register. The contents of the Instruction Register are used to determine which of the available Data Registers is to be used. The JTAG debugger feeds values into these registers via a protocol that is transmitted over a set of pins (see figure 1).

A JTAG interface only requires 4 or 5 pins to implement:

- TCK, the Test Clock
- TMS, the Test Mode Select which controls the JTAG TAP controller state machine
- TDI, Test Data In
- TDO, Test Data Out

The fifth signal is an optional TRST(-) and asserting this will asynchronously reset the JTAG TAP controller state machine.

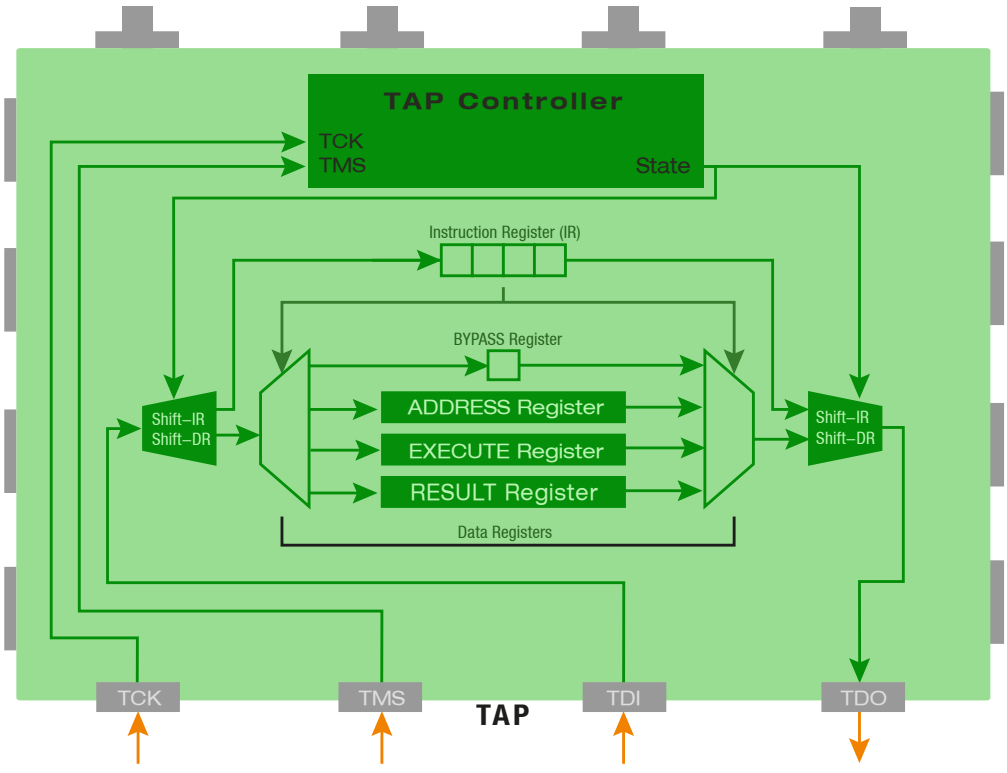


Figure 1: JTAG circuitry with three Data Registers as an example

All a JTAG debug tool does is shift values into the Instruction and Data Registers using TDI, set the state machine using TMS and wait for the device to respond via TDO. For example, a common command might be 'Read Memory'.

1) Set the Instruction Register (IR) to select the ADDRESS Data Register.
2) The address to read is set in the ADDRESS Data Register.
3) Set the IR to select the EXECUTE Data Register.
4) The command "Read memory" is set to the EXECUTE Data Register.
5) The TAP Controller State Machine is set to Run Test using the TMS signal.
6) The results are written to the RESULT Data Register.
7) Set the IR to select the RESULT Data Register.
8) The result is read from the chip using TDO.

This is a fairly simplistic view but is enough to give the reader an understanding of how the technology works, if not the actual implementation.

Each silicon vendor implements their own command set and the width of the Instruction Register will vary with the complexity of the set of Data Registers. The number and types of available Data Registers is implementation dependent and varies widely.

One of the nice features about JTAG is that it scales well into multichip and multicore scenarios. The output of each device is chained to the input of the next one. By using the Instruction Register to select the BYPASS Data Register in the first and last core, the tool can communicate with the middle core. An example layout can be seen in figure 2.

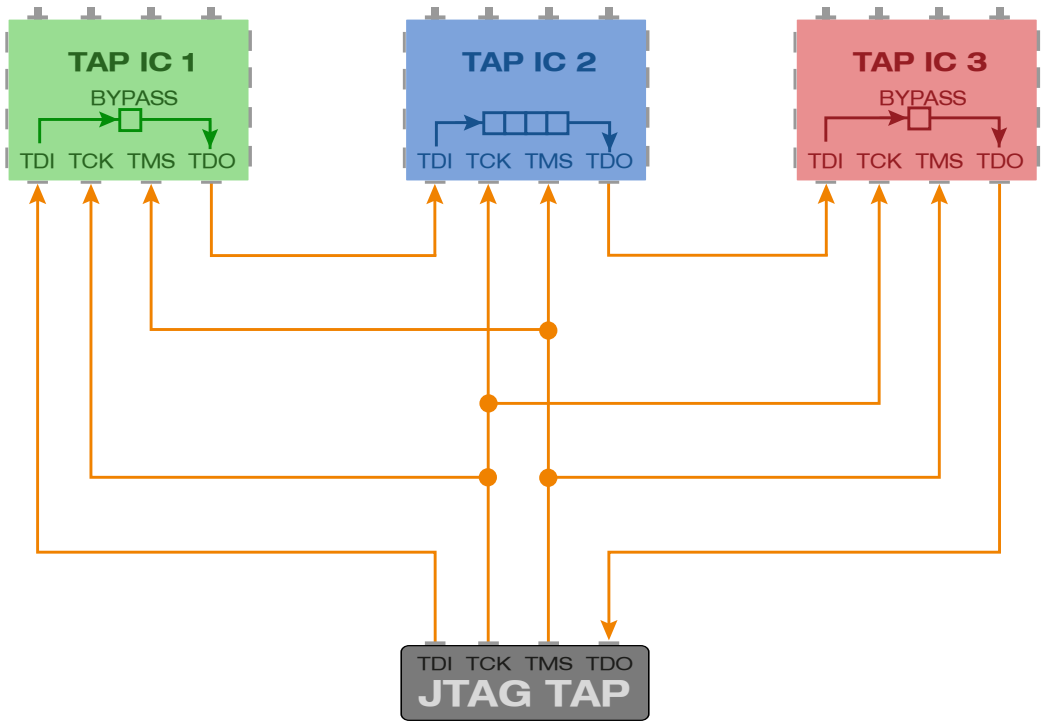


Figure 2: Bypassing TAP IC 1 and TAP IC 3 allows a debugger to communicate with TAP IC 2

JTAG chaining could be done for individual devices on a single board or cores within a multicore chip; the layout and principles are the same. All a tool needs to know is the width of the Instruction Register of each TAP controller in the chain and the number of devices so that any single device can be isolated.

This system is not perfect. Let's take a closer look at each device as shown in figure 3.

In this example, each device has a different voltage level. A lot of extra work would need to be done on the PCB design in this scenario. The normal solution is to use a number of dual voltage buffers to try to get everything to the same level. When this is coupled with the requirement to include or exclude devices from the JTAG scan chain, the resulting switches or jumpers necessary to implement this can often lead to long stub lines on some or all of the signals. This can be disastrous, especially for TCK, and results in bad edges, poor signal integrity and an overall reduction in the possible maximum speed of the entire JTAG interface to all devices.

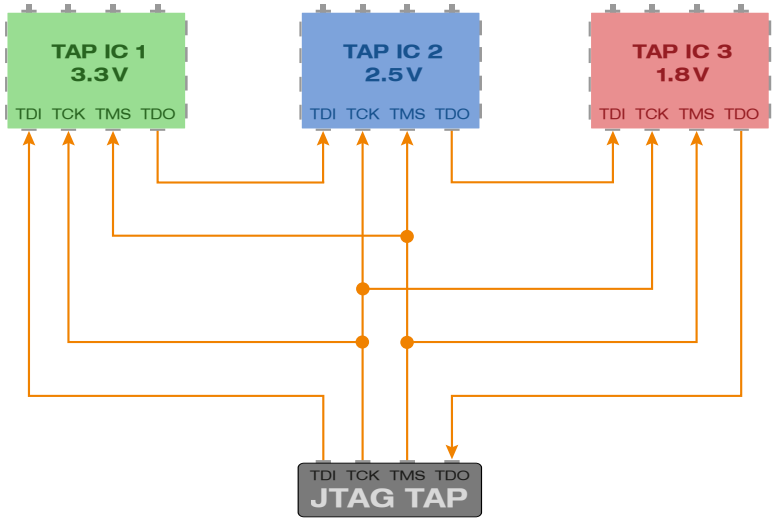
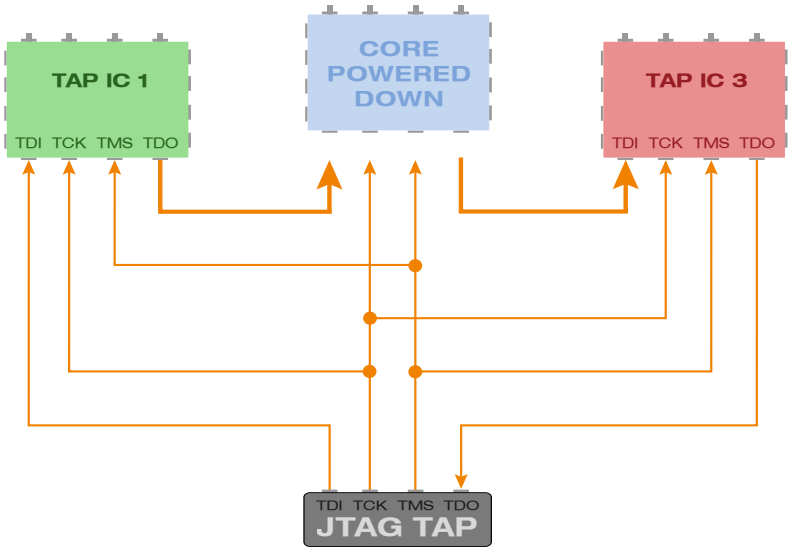


Figure 3: Different voltage levels require additional PCB design work



As embedded designs strive for even greater efficiency and prolonged battery life, devices are regularly put into low power or sleep modes when they are not currently active. An example can be seen in figure 4.

As TAP IC 2 powers down the JTAG chain is broken; there is no connection between the TDO of TAP IC 1 and the TDI of TAP IC 3. In such a configuration it is impossible to debug any of the devices.

Figure 4: Power down or sleep modes brake the JTAG chain

The JTAG Switcher

By placing a JTAG Switcher between the board JTAG header and the individual devices, these problems can be overcome. The JTAG Switcher would be implemented in an FPGA that can cope if devices present different voltage levels. The design will also allow it modify which TAP ICs are presented to the debug tool, changing which TAP ICs are included in the debug chain at any given point in time. A conceptual block diagram is shown in figure 5.

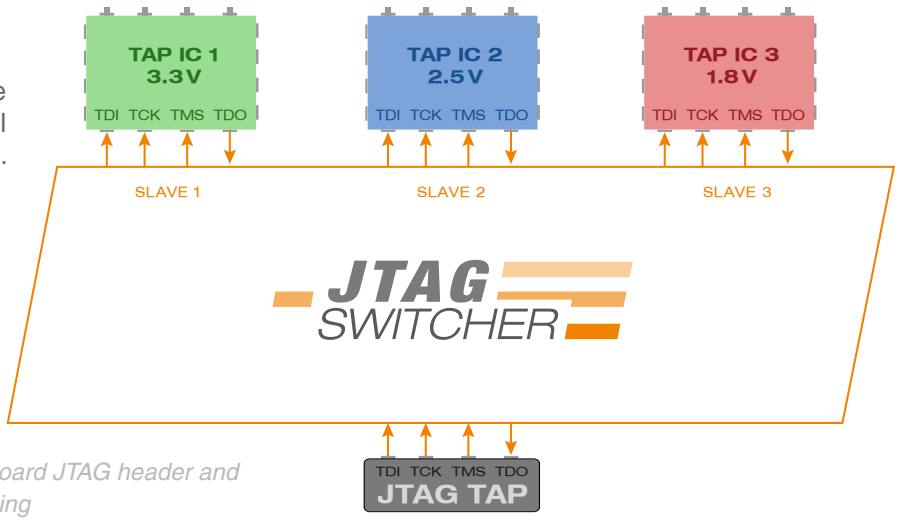


Figure 5: A JTAG Switcher placed between the board JTAG header and the individual devices allows a seamless debugging

In the case where the target system comprises of several discrete boards, the JTAG Switcher could be housed in a separate PCB (or box) to allow the debug tool to connect through it to all of the target boards. Any of the target boards could be removed from the JTAG chain at any point, if required.

The JTAG Switcher appears as a new device in the JTAG chain and must be positioned first. Debug tools can communicate with it via standard JTAG protocol in order to configure it.

Conceptually, the inside of the JTAG Switcher looks like shown in figure 6.

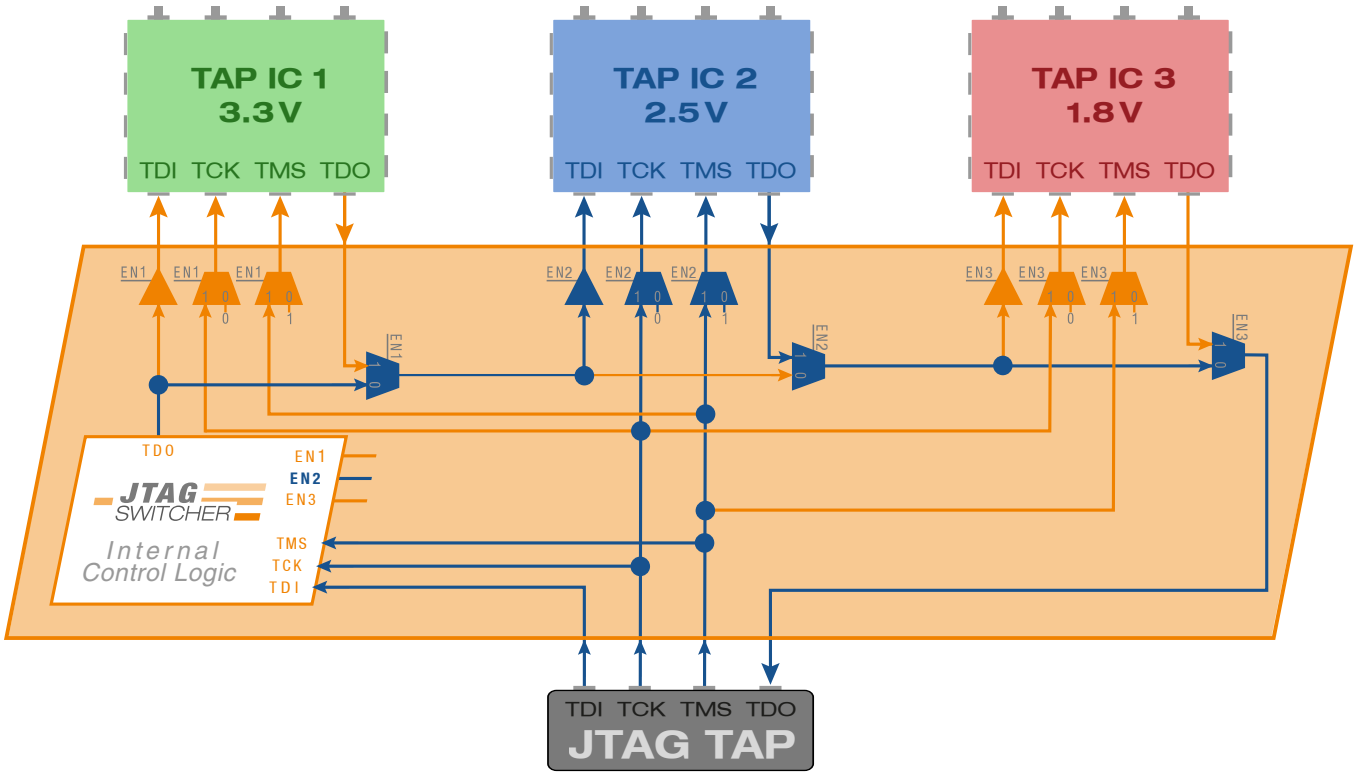


Figure 6: EN2 is active, this allows a debugger to communicate with TAP IC 2

The JTAG Switcher manages a configurable number of internal control bits: EN[n..1] and a TAP IC is excluded from the chain if the corresponding control bit is set to 0.

By excluding a TAP IC:

- TDI is tri-stated
- TCK is gated and kept at 0
- TMS is kept at 1
- TDO from this TAP IC is ignored

If all control bits are set to 0, only the JTAG Switcher's internal control logic will appear in the JTAG chain.

It was decided to ensure that the JTAG Switcher conformed to IEEE1149.1 so that no extra pins are required for its operation and any JTAG tools can chose to ignore it if they don't have any knowledge of the internal control logic.

The JTAG Switcher is designed to be transparent: once it has been configured it will just appear as another inert TAP on the JTAG chain. It can be configured at startup (or even hard-wired at design time) and left to silently get on with its business.

It is imperative that the JTAG Switcher be placed first in the JTAG chain as this will allow the chain to always recover in the event that any other TAP IC becomes inaccessible.

It was also important that the TCK signal to an excluded TAP IC be gated so that excluded TAP ICs do not see it toggling. Some ICs exhibit unusual and unwanted behavior if the TCK signal is allowed to toggle.

The JTAG Switcher will change the status of the EN control bits only whilst the affected TAP IC is in the Run Test/Idle state. This has two advantages: A JTAG transaction is not halted mid-way through completion. All excluded TAP ICs are in a consistent and known state. In the JTAG Switcher a new configuration can be prepared over many TCK cycles. But the final activation takes place once the TAP controller state machine of the JTAG Switcher reaches the Run Test/Idle state.

Open Source

The JTAG Switcher is provided by Lauterbach free-of-charge to anyone who wishes to use, play with or study it. It has been released under an MIT Open Source License and is available from https://www.lauterbach.com/jtag_switcher.html. The package includes the VHDL source files, full documentation and some pre-built examples for various Lattice and Altera FPGAs. It also includes example configuration scripts for the Lauterbach TRACE32 debugger.

It is hoped that by providing the JTAG Switcher as an Open Source package, it will become more widely adopted within the embedded community and act as a springboard for engineers to modify it in new and interesting ways to create real-world solutions to common (and some uncommon) problems. Because the API is open and fully documented, it should be possible to adapt any JTAG based tool to work with the JTAG Switcher in order to leverage the advantages that this technology can provide.

It is envisaged that developers may add a small FPGA to their development boards to improve multichip JTAG based systems, or they may decide to create a small stand-alone unit to allow them to connect multiple development boards together to form a single, adaptable JTAG chain. It is hoped that silicon designers will include this in multi-core devices to provide an extra layer of functionality.