System Level Instrumentation using the Nexus 5001-2012 specification

Dr. Neal Stollon, HDL Dynamics nstollon@nexus5001.org

Abstract – The paper addresses issues and challenges of integrating chip-level instrument resources, and their use at the board or rack levels of system integration and will discuss IEEE-ISTO Nexus 5001 as a solution in this context. This level of instrument integration is relevant to a variety of automotive, aerospace, and telecommunications systems. The presentation/paper will provide a summary overview of the Nexus 5001 instrumentation architecture, and discuss interfaces defined in the Nexus 5001-2012 specification, released in 2012, that simplify the integration of Nexus 5001 and other instrumentation subsystems, along with some example systems level architectures for debug and other analysis instrumentation and interfaces for access at the systems level.

Introduction.

Chip level embedded Instrumentation systems have been well established over the last decade as a required feature in all but the most trivial of ICs. Adding instrumentation to a design enables better functional observability and controllability of the systems to address more subtle issues related to integration and environment [1]. Systems integrators at the board and rack level in mission critical applications such as automotive, networking, and military systems have started to adopt and leverage chip level instruments to address the calibration and debug requirements of complex systems. In turn, instrumentation interfaces developed for the individual chip level analysis need to comprehend the larger scale debug challenges of boards and systems. Nexus 5001, an IEEE ISTO standard, along with other proprietary debug and instrumentation subsystems, has provided a comprehensive and effective means for in-silicon analysis of individual processors and SoC subsystems. Nexus 5001 was developed to address the limitations and concerns on instrumentation in complex (multi-core) systems, and has been used extensively in the automotive domain for over a decade. With the introduction of Nexus 5001-2012, the Nexus instrumentation and debug architecture introduces new features that also support the access to and integration of multi-chip systems found in distributed or heterogeneous board environments typical to automotive, aerospace, and telecommunications systems.

A Nexus 5001 Overview

The Nexus 5001 instrumentation standard was originally developed as a set of guidelines, based on best debug tools practices in the semiconductor and test tools industries. The initial 5001 specification was released as an IEEE-ISTO industry standard in 1999. In particular, Nexus 5001 sought to address the bottleneck of instrumentation solutions that were proprietary to a single processor family, and provide a common instrumentation framework that supported diverse processors or cores. Nexus 5001 specification defines a modular framework and guidelines for a message protocol, baseline command set, a register infrastructure, and IO interfaces.

Nexus 5001 Message Packet Protocol

One of differentiating features of Nexus 5001 is that it is based on an instrumentation-centric packet protocol. All Nexus compliant communication transfers are via self-contained packets. Packet messages consist of fields that include information being transferred as a payload. The payload may be program flow information, trace data values, register or memory values being read or written, among others. The packet also includes meta-information, such as type of information,

cores or tools that are the payload source or destination, the register or memory location, timestamps, etc.

Since Nexus 5001 communication is based on packet based messages, it allows for straightforward interleaving of communication with different cores, with minimal switching overheads or delays insetting up communications with different cores. Nexus messages incorporate ID cords that correspond to different instrumentation resources in the system. So, as shown in Figure 1, a Nexus 5001 packet with source/destination to subsystem-A may be followed immediately by a packet associated with subsystem-B.

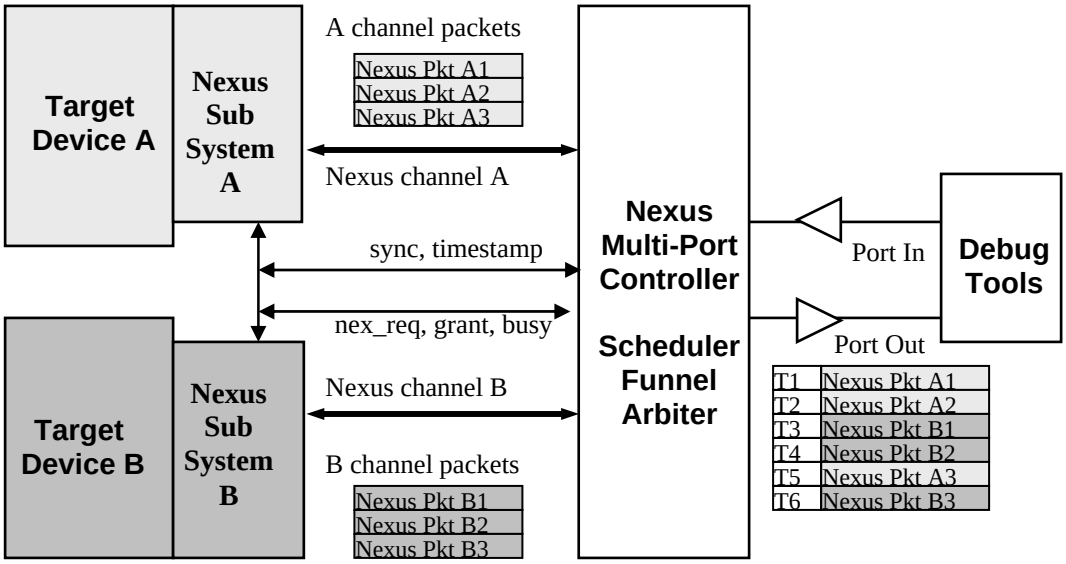

Figure 1: Integrated Nexus 5001 Subsystems with Interwoven Packets

Packets may include either standard commands (with vendor or user defined field options) and custom/user defined command/instructions. Packet types are defined by a TCODE (transaction code) header field. The Nexus 5001 standard defines 33 standard TCODES for common operations including instruction and data trace and data transfer, monitoring, breakpoints and run control for system debug, and importing of data for calibration and port replacement operations. The specifics of the commands are discussed in the Nexus 5001 specification. Commands are typically interpreted in hardware, compliant to the instrumented features of a processor or subsystem.

Packet Example (Direct Branch Trace Message Data)

Vendor defined	1-8 bits	Vendor defined	6 bits
Timestamp	Instruction Count	Message Source	TCODE (0x3)
Optional timestamp to record time event occurred	Payload - Number of instructions since last BTM	For multi processor systems to identify which processor sent message	Message transfer code

Figure 2: An exemplary Nexus 5001 Packet

In the Figure 2 example, a branch trace message (BTM) packet, the packet payload is a single field containing instruction count. Other packet types may include additional fields of variable or fixed length. Note that the message source and timestamp fields are variable in size for different devices, or omitted as appropriate, to provide maximum flexibility.

Nexus 5001 specification reserves a majority of TCODES for user or vendor defined commands that may map to custom instruments or instructions to other debug environments. Using the above packet example as a template, the entire command structure for control of a non-Nexus debug operation may be embedded into the payload field of a Nexus packet.

Nexus 5001 architecture allows several different implementations of an instrumentation interface to be instantiated based on specific debug or calibration requirements and supported interfaces Standard Nexus 5001 commands typically use Nexus defined registers that are defined in the specification. Custom or application specific instruments can implement user defined registers. In either case, registers may be mapped to reserved (debug/calibration) register space or a system memory map.

An example Nexus 5001 architecture is shown in Figure 3. Since, unlike more proprietary debug architectures, Nexus 5001 does not specify a particular implementation or IP, an application specific variety of instrument features, scheduling methods and means for transfer of messages between the Nexus 5001 interfaces and different cores are possible [2].

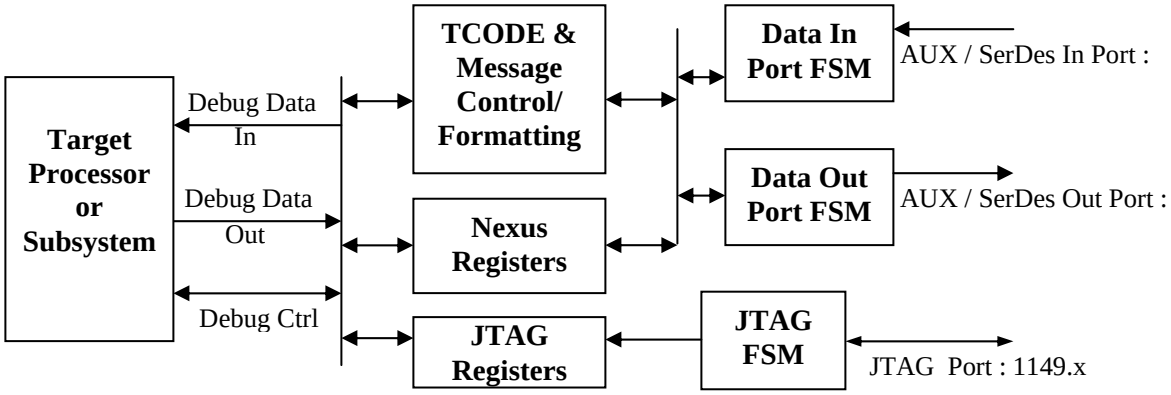

Figure 3: A simple Nexus 5001 Subsystem architecture

The Nexus 5001 standard roughly groups typical debug instrumentation features in terms of four classes of increasing complexity, which are summarized in Table 1Class 1 defines run-control operations. Class 2 defines Instruction trace operations; Class 3 defines data trace and real-time register and memory accesses. Class 4 defines memory substitution, as a processor debug method of replacing on-chip program and data memories with external code and data sourced by the debugger tool. The segmentation of classes are guidelines, and not all systems debug features will or need to fit into a given class, however they are useful for discussion of features with varying bandwidth requirements. Class 1 operations typically have low or non-real time data transfer requirements that may be met via JTAG interfaces. Class 2 operations operate in real time, but due to the intermittent nature of ownership trace and compressed program trace, data transfer may also use lower performance interfaces, including JTAG. Class 3 data access and data trace and Class 4

memory substitution are much more bandwidth intensive and typically require a dedicated trace port.

Nexus	Services	Typical Features
Class 1 Run time Control	Static debugging Breakpoints	Read/Write registers / memory Set / Clear Breakpoints Stop / Start code execution Control entry into / exit from debug mode Stop execution/enter debug mode on hitting a breakpoint Single step instructions Read Nexus device ID Static memory and I/O access
Class 2 Dynamic Debug	Watchpoints Ownership Trace Program Trace	All class 1 features plus: Ownership Trace Messages –process / task ownership trace) Watchpoint Messages – Event Trigger a Nexus message Program Trace Messages – Real time, non intrusive instruction trace Port Replacement of GPIO
Class 3 Data Trace	Data Trace Real-time read/write Transfers	All class 2 features plus: Real Time Data Access – real time Registers / memory read/write Real Time Data Trace Transmission of data used for data acquisition
Class 4 Advanced Debug	Memory Substitution	All class 3 features plus: Watchpoint Triggers – Watchpoint triggering of trace events Memory Substitution –run code from memory in development tool (ROM emulation) Start memory substitution on watchpoint Over-Run Control – Allows Nexus operations to stop core operation if data buffers will overflow.

Table 1 - NEXUS 5001 IMPLEMENTATION CLASSES

Nexus 5001 supports instrumentation packet transfer between a target system and the outside world via two types of interfaces:
1. JTAG Test and Debug/Test Control (DTC) Access Points compliant with IEEE 1149.1 and 1149.7 JTAG standards, with user defined instructions defining Nexus-access operations. JTAG interfaces have provided the default means of control-based debug interfaces, although their use for more data intensive trace and calibration operations are more limited.
2. Dedicated Instrumentation data interfaces, implemented as either parallel FSM-controlled dedicated interfaces (referred to in the Nexus 5001 specification as AUX ports), or Nexus high speed serial (Aurora) interfaces (discussed in the next section). Implementations that have any significant amount of trace data will typically require one of these data interfaces.

Nexus 5001-2012 interfaces

Nexus 5001 has supported 1149.1 JTAG and streaming parallel (AUX) trace ports since its inception. In the 2012 Nexus 5001 specification release, 1149.7 and Aurora SerDes standard interfaces are also supported. These interfaces address latency and comprehensive data instrumentation needs of system debug environments.

IEEE 1149.7 is a downward compatible extension to 1149.1 JTAG that defines a framework to improve JTAG-based debug performance. Based on limitations of 1149.1 for debug purposes, 1149.7 defines extended test and debug modes, background instrumentation capabilities, and reduced pin configurations. 1149.7 features are divided into different classes [3] based on features:

- The class T0 capability provides a backward compatible 1149.1 interface.
- Class T1 capability provides user selectable power control modes (based on power down states and power down after delay) selectable power mode controls
- Class T2 capability provides coupling mechanisms for reduced bypass delay (shorter latency for serial scan operations) for the case of multiple TAPS on a chip or a system
- Class T3 capability provides for parallel data input (TDI) and output (TDO) configurations (aka Star configurations)
- Class T4 supports multiple test/debug configuration functions, including operation in a 2 wire TAP (Star-2) mode. In this configuration, pin operations are dependent on the 1149.7 mode, acting as a control interface during initial access states and then switching to a 2 wire data interface mode during data transfer states.
- Class T5 supports BDX (Background Data Transport) and CDX (custom data transport) transfer modes for higher data channel performance.

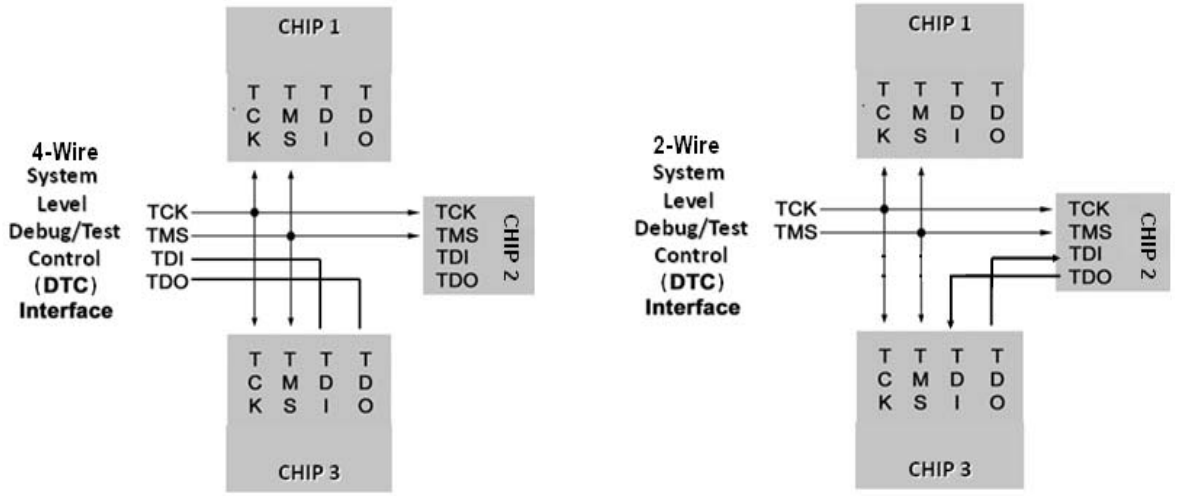

Figure 4: Nexus 5001 2/4-wire parallel-JTAG 1149.7 interfaces

From a system perspective, 1149.7's parallel chip level data interfaces (as opposed to 1149.1's daisy chained serial data interfaces) simplifies and reduces latency and timing of the JTAG interface allows between chips. This parallel operation is supported for chips with both 4 pin and 2 pin JTAG interfaces. Figure 4 shows multi-chip examples of a parallel 1149.7 interface (per T3) supporting both 2 wire (chip 1, 2) and 4 wire (chip 3) JTAG device configurations (T4). In a purely 2 wire interface, 1149.1 TAPs can be supported as serial TDI/TDO connections being driven under 1149.7 control, as shown by the chip 2 and 3 interfaces in the 2 wire example.

1149.7 support for advanced debug features such as Custom Data Transport and Background Data Transport modes are being evaluated for Nexus 5001 applications and no specific guidelines for class T5 features are discussed in the 5001-2012 specification. These configurations have the potential to improve JTAG data transfer utilization and throughput and introduces options for JTAG on-chip initiated transactions that allow for more autonomous debug modes or operation [3].

Aurora SerDes Interface

Aurora defines the link-layer protocol for Gigabit SERDES instrumentation interface over a LVDS connection. Aurora converts parallel data associated different trace operations and serializes it to allow multiple lanes and channels to transfer several concurrent trace flows and other bandwidth intensive data across point-to-point serial links. (Figure 5). It supports transmission via simplex and full duplex channels, allowing both trace and calibration over common channels. In comparison, the AUX ports in previous versions of the Nexus specification are unidirectional interfaces, which increase the required pins when both trace and calibration are required.

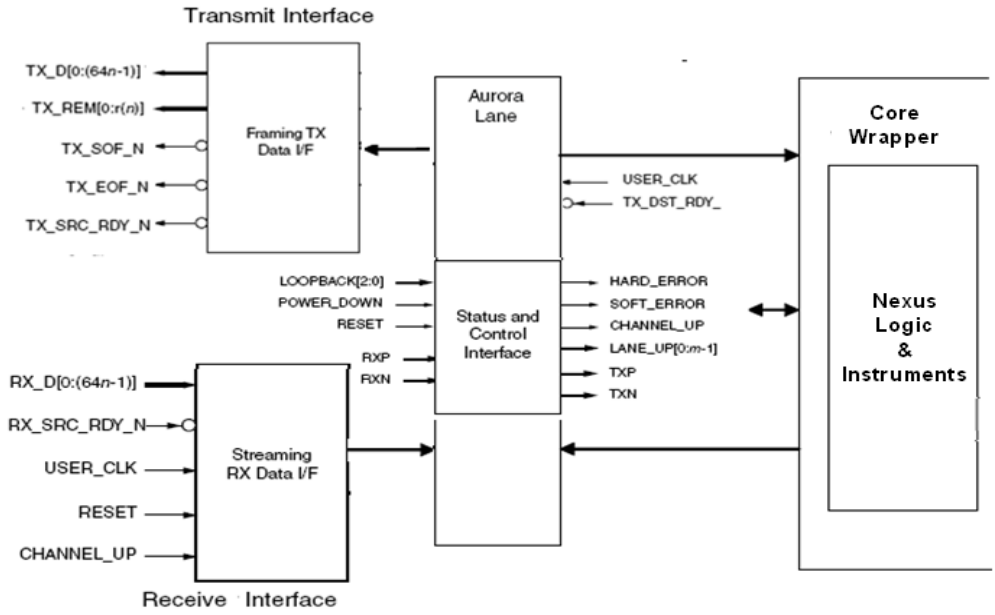

Figure 5: Nexus 5001 Aurora Serdes Interface

The Aurora protocol is a scalable low-latency protocol that manages partitioning of parallel data into different lanes of a channel and the associated serialization and de-serialization operations. The protocol was developed by and its specifications are openly distributed and licensed by Xilinx [4]. The goals for Aurora were to provide a transparent and flexible framing interface to high-speed serial links, that are compatible with logic constrained FPGA implementations.

The Aurora protocol incorporates the steps of data padding, clock compensation, CRC insertion, Link Layer Framing, and 8B/10B Encoding. Aurora high bandwidth transmission is limited only by SERDES data rate capability. With common and typical physical interfaces supporting transfer bandwidth well above 3 Gbs/lane, Aurora supports an extended number of lanes (8 lines per the

Nexus-2012 specification) that can be bonded into a single or multiple trace channels for high aggregate bandwidth needed for data trace in multi-core architectures. In addition to significantly improving the data transfer bandwidth, IP from Xilinx and others allow formatting and aggregation logic of channels for required configurations. It provides support for vender/user defined frame size, expedited messaging, and flexible framing via built-in flow control logic.

Systems Level Nexus Integration

Higher performance and reduced pin standard interfaces allow for the introduction of Nexus 5001 instrumentation into systems with pin and wire limited applications such as mobile devices as well as complex networking and computing systems that require large amount of debug data to provide a useful view of system operations. An example multicore hierarchical system, using 1149.7 and Aurora Gigabit SERDES interfaces in a system level configuration, shown in figure 6.

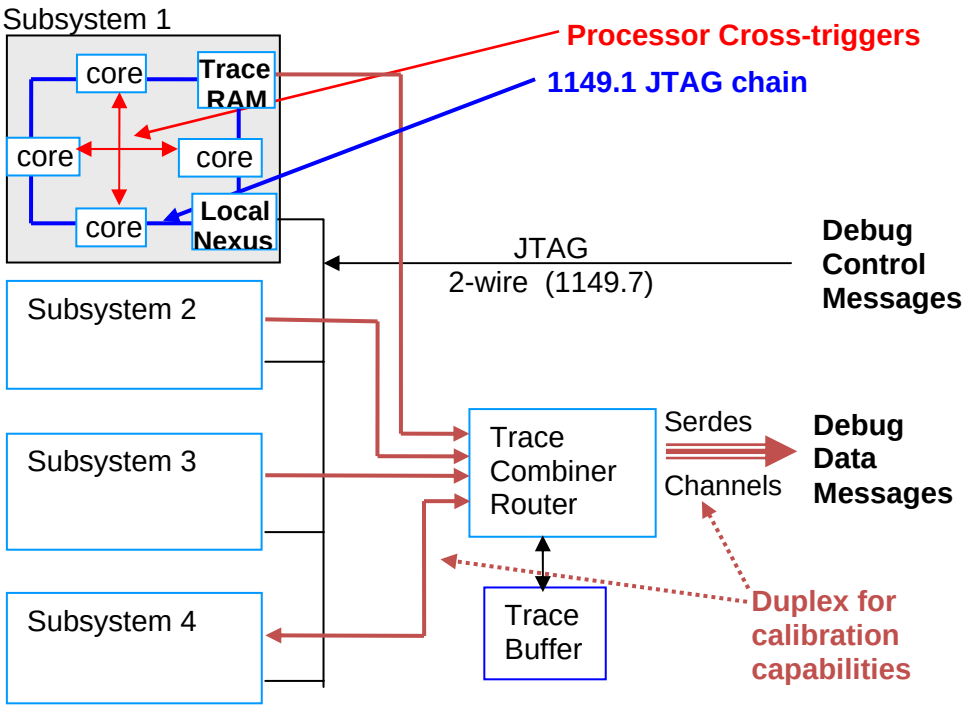

Figure 6: A system level Nexus 5001 configuration

A parallel 1149.7 interconnection allows configuration of all subsystems (cores within the subsystem may be integrated at chip or board level) for break/trace/watch points and for common synchronization of trigger and run control signals. JTAG accesses to the subsystem may be addressed in various configurations, included in serial or parallel JTAG. They also generally are compatible with JTAG variants such as P1687/IJTAG. Trace data within a subsystem is locally stored and aggregated and packetized by a "Local" Nexus block (refer to figure 3 as an example) for transmission over an Aurora or AUX channels (refer to figure 1 as an example). A dedicated trace combiner/router block may be used as an intermediate interface to more effectively reformat and buffer packets into available channel resources for transmission to external analysis tools. For subsystems requiring calibration, the flow is similar, with duplex channels and bidirectional resources for transferring calibration data into the device(s).

The example in figure 6, may be used a template for a range of architectural configurations. Since Nexus 5001 protocols and logic are core agnostic, subsystems may consist of homogeneous or heterogeneous architectures. A decomposition of a "Local" Nexus 5001 block shown in Figure 6 is shown in figure 7 to consist of different instrument subsystems that share a common set of external interfaces. The Nexus 5001 instrumentation for the different cores in figure 7 are segmented into different instrumentation clusters based on the required features supported by different cores. Buffering of the different instrumentation clusters allows concurrent capture of data from different parts of the subsystem, based on core specific or common trigger conditions. The common Nexus logic at the interface would typically include prioritization or arbitration of packet generation and transmission to the external interfaces. As in prior versions of the Nexus specification, buffer overflow may be addressed by either halting of core(s) until buffer overflow is relieved, or by notification to the tools that overflow has invalidated data transfers.

.

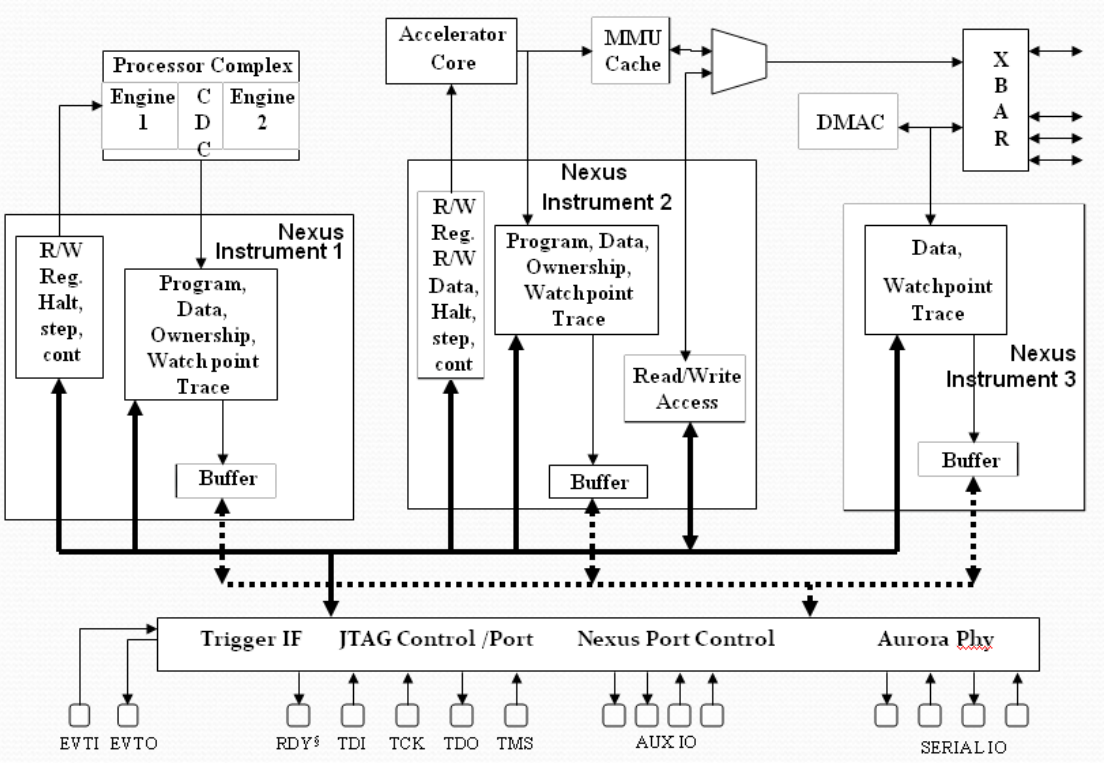

Figure 7: Nexus multi-core instrument subsystems

The Nexus 5001 specification defines several interface configurations for use with the 1149.7 and Aurora resources. These allow for compatibility with existing JTAG and debug interfaces. As the 1149.7 class (T0) includes 1149.1 operations, the 1149.7 interface is the familiar 4 wire JTAG TAP. The 1149.7 specification allows a port to initialize the DTC in pass through mode, which can then be enabled into the 1149.7 4 or 2-pin modes, Figure 8 shows a configuration that incorporates both parallel AUX ports and Serial Aurora interfaces for data transport. The AUX ports and Aurora interfaces ports may be sourced by different chips or subsystems, under either fixed or programmable control, further increasing the instrumentation interface flexibility.

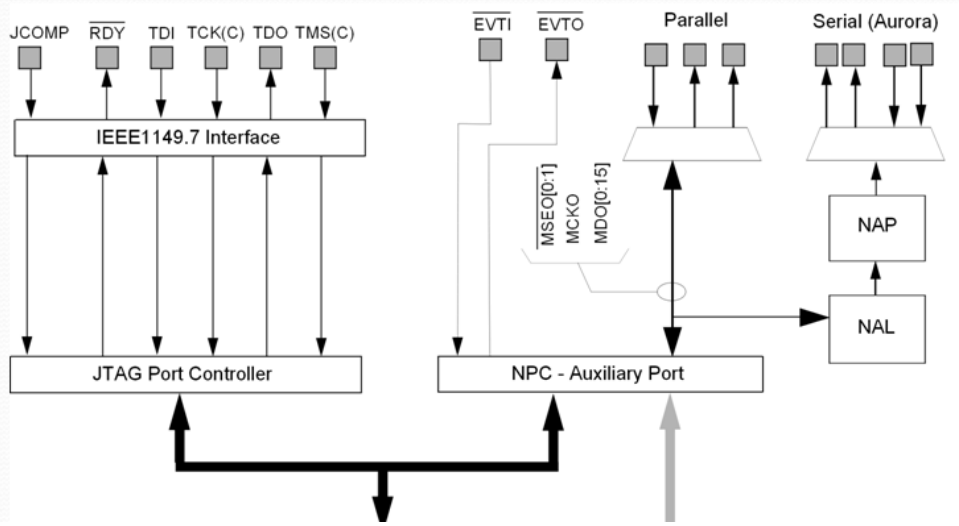

Figure 8: A legacy Nexus-2012 interface including a parallel AUX ports

Figure 9 shows a configuration that exclusively uses Serial Aurora interfaces for data message packets. This provides higher throughput on a per pin basis and enables more flexible and configurable configuration and transmission of data packets in more complex data driven systems. Trace output can go to either the physical interface (via NAL/NAP/pins) or system memory.

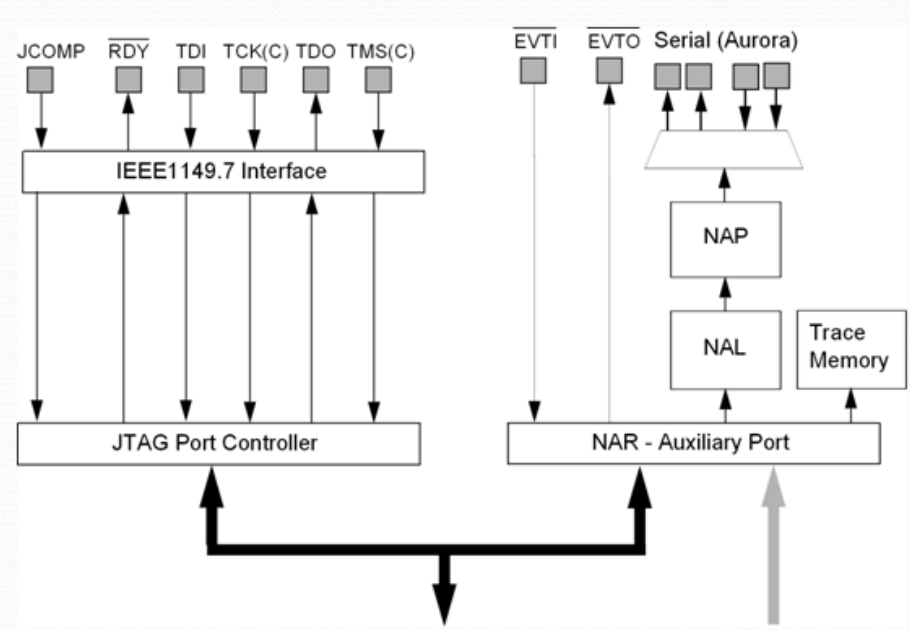

Figure 9: A Nexus-2012 Interface Supporting 1149.7 and Aurora SERDES

Summary

Nexus 5001 was conceived and developed to address multi-core debug issues [5] and has supported industry standard 1149.1 JTAG and parallel trace and calibration ports since its initial release in 1999. Nexus 5001 and other on chip instrumentation approaches are receiving increased interest from the system level community, based on the emerging need for extended and comprehensive

debug access at the highest systems levels. Nexus 5001 has been used extensively in the US automotive electronics since its release and has progressively been adopted into other domains. With the release of IEEE 5001-2012, Nexus 5001 also now supports the emerging IEEE 1149.7 JTAG interface, which enables systems level features such as 2-wire interfaces and parallel chip and board level debug configurations. IEEE 5001-2012 also specifies high speed trace and calibration using industry standard Aurora SerDes interfaces, allowing improved real time trace bandwidth needed to support multi-core architectures. Unlike proprietary instrumentation systems, Nexus 5001 is core and processor agnostic. It defines the protocol, interface, and infrastructure, but allows the implementation and configuration of the system to be defined by vendors and users. The combination allows diverse subsystems to be interconnected for debug and analysis purposes with a minimized pin interfaces and application specific flexibility at the chip, board and systems levels.

Nexus 5001 is supported by an industry consortium, the 5001 Nexus Forum [6], which provides resources for both technical and business support of the IEEE 5001 standard. In complex environments where system reliability and accessibility is critical, Nexus 5001 in conjunction with other debug instrumentation solutions provide real advantages in helping to observe, control, and understand operations of complex computing systems.

Neal Stollon is chairman of the 5001 Nexus Forum, which provides industry support for Nexus 5001 and related instrumentation standardization efforts. He has a Ph.D in EE from Southern Methodist University and is a licensed P.E., and has a decade of experience in debug architectures and instrumentation issues, on top of another decade or so of processor and SoC experience at TI, LSI Logic, MIPS, and elsewhere. Dr. Stollon is CTO at HDL Dynamics, providing systems analysis and consulting on embedded IP and instrumentation solutions for digital systems He is also the author of the book “On Chip Instrumentation: Design and Debug for Systems on Chip ”. He may be reached at neals@hdldynamics.com

References

[1] “Processor and System Bus On-Chip Instruments”, *Embedded Systems Conf (ESC) Proceedings*, April 2003

[2] “Nexus Based Multi-Core Debug”, DesignCon 2006 Proceedings, Jan 2006 http://www.designcon.com/media/2006/3-WP1--Neal_Stollon.pdf

[3] “IEEE 1149.7: Expanding and improving JTAG”, Nov. 2008 http://www.embedded-computing.com/articles/id/?3687

[4] Xilinx Aurora website http://www.xilinx.com/products/design_resources/conn_central/grouping/aurora.htm

[5] “Multi-core analysis made easy with the Nexus 5001 debug spec” March, 2008 http://www.embedded.com/design/multicore/206901752

[6] Nexus 5001 Forum Website http://www.nexus5001.org