On-Chip Instrumentation and In-Silicon Debug Tools for SoC

Dr. Neal Stollon
HDL Dynamics
neals@hdldynamics.com
So What do we mean by On-Chip Instrumentation and In-Silicon Debug?
What will this talk cover

• An Overview of “What is On-chip debug”

• A quick example of different types of On Chip Debug Instruments

• A survey of industry standards efforts.
Some quick definitions

Some Quick Definitions

• Verification - process of analyzing, fixing errors, and optimizing a model or design
• Debug - process of analyzing, fixing errors, and optimizing a product (chip, board or box)
• EDA - Any tool used to manage the tedium and complexity of analysis and verification of a design.
• Instrumentation - any logic (on chip or off) used to manage the tedium and complexity of analysis and debug of a product
Verification vs. Debug Efforts

Pre-silicon – EDA based analysis
Focus on Architecture Bugs

Instruction Level/Bus Functional

RTL Diagnostics

System Initialization

Post-silicon – In System centric analysis
Focus on System Bugs

RTOS Integration

Application Software

Multi-core Integration issues

Point were Hardware is “assumed working”

Modeling and Verification Abstraction

System Platform
System Analysis Focus
Core IP

ESL Hardware Simulation
Hardware Prototype/Emulation
Software Debugger System platform

Copyrighted and Trademarked 2007, HDL Dynamics. All rights reserved.
Advantages of Debug Solutions

• On-chip Instrumentation (OCI) facilitates embedded design success
  – It’s difficult to fix what you can not see
  – Compliments other analysis for more complete verification
  – OCI provides real-time analysis of what is happening in the chip
    • Measure events and operations on core, buses, caches, peripherals . . .
• Enables core and system debug and optimization
  – Visibility, synchronization and control of all system components
  – Speeds up getting system software working and optimized
    • Inter-core communications and interfaces

Better Debug = Faster time to Market = $$
SoC Debug Evolution

Embedded Debug Complexity keeps increasing
- Gates increase geometrically - Pins increase linearly
- Significant debug difficulties for leading architectures
- More complex debug needs Instrumentation on the chip
Consider a Typical SoC Design

- **Complex processors**
  - Multithreaded, superscalar, etc.
- **Complex bus fabric options**
  - lots of data transfers
  - Differing bus interfaces
- **More than one core**
  - multiple processors
  - other blocks of IP
  - Differing clock domains
- **IO constrained**
  - Embedded operations of interest not accessible by outside world
- **Problems show up in real time**
  - hard to emulate or reproduce

SO HOW DO WE ADDRESS SoC DEBUG?!?
1. Instrument the processor

Processor instrumentation types

Run control
- start, stop, single-step
- HW/SW breakpoints

Trace
- Instruction and data
- Complex Triggering
- Trace compression
- Performance statistics

Processor Debug SW
- Commercial Debuggers
- GNU/GDB

System Bus Fabric
- OCP/AMBA

Other cores
- OCP/AMBA

Processor Core
- Debug Block

Other IP (security, video, ...)
2. Instrument the Other IP

Add debug control and trace

Logic Trace
- Capture logic sequences
- Complex triggering

Bus Trace
- Special features for on-chip bus properties
- Detect abnormal sequences
- Correlate transfers and responses
- Make Bus Performance measurements

Viewing and analysis similar to EDA based simulation
- Ex. Waveform views
3. Synchronize and control

Cross Trigger blocks
- Synchronized on chip control
- Trigger on system information
- Complex trigger actions
- Direct on chip configuration actions

Bus Master Emulation
- Direct control of bus operations
- Analyze bus properties
- Latency Measurements
- Access peripherals
4. Controlling Instrumentation

JTAG interfaces
- JTAG is standard on chip IF
  Independent of system speed
- Simple RD/WR protocol
- Instruments on separate or
  merged JTAG chains.

However...
- Slow
- Instrument control is limited
- Requires on-chip trace buffers

Under Processor control
- Memory mapped debug
- Simple SW control
- Debug blocks as peripherals

However...
- Debug SW changes operation
- Tricky for core to debug itself
5. Improving debug performance

Add Additional interfaces
- Mainly for deep trace
- Allow higher speed access
- Reduce need for on chip buffers
- Several configurable options
  - Nexus Interface,
  - MIPI Narrow Band Trace port
  - Serdes IO
  - parallel ports, etc.
- Requires specialized probes or Logic Analyzers
  - High IO, High Capacity buffers
Key Open issues in On Chip Debug

• Merging debug operations for several cores
  – Control/data transfer via a single JTAG compliant TAP

• Global control signals for multi-core cross triggering and synchronous actions (go, halt and breaks)

• Multi-core trace with timestamps

• Probes and tool API’s to support multi-core trace

• Handle multiple instantiations of source level debuggers

• Standard ways to measure on chip activity
  – buses, caches, execution cores, co-processors, interrupts, peripheral device events, . . .
On Chip Debug Standards Efforts
On Chip Debug Standards

• Standards will enable building better solutions
  – 1149.1 – JTAG are most notable and widespread
  – Several efforts in standards driven progress for SoC Debug

• Design for Debug is coming on its own as a “ility”
  – Many variants on theme needed to address range of silicon debug requirements, time to market
  – Need to encompass proprietary solutions - varying levels of focus, completeness and depth

• Several standards efforts are ongoing
  – Nexus
  – OCP-IP
  – MIPI
  – SPIRIT
  – 1149.6 (AJTAG) , P4687 (IJTAG)
The Debug Standards Landscape

IO Level of Integration
- JTAG – different flavors
- Nexus 5001
- MIPI NIDnT

Protocol Level of Integration
- Nexus 5001
- Multicore IP & probe APIs
- Other Proprietary

Instrument Level
- Many vendor/IP solutions
  - MIPS EJTAG, PDtrace
  - ARM ETM, CoreSight

System Level Debug Communications
(SoC level run control, Cross-triggers, etc.)

Core Analyzers
- Core A
- Core B

Multicore Trace formats

Bus Analyzers

Bus Fabric

Debug Software

Tool Level of Integration
- SW/EDA Tool level APIs
- HW tools level APIs

SPIRIT

MIPI

IJTAG

AJTAG

Nexus

OCP
Nexus Overview

• The Nexus 5001™ Forum is a program of the IEEE/ISTO

• Focused on embedded system and processor debug interface standard and applications

• The Nexus standard defines classes of standard on-chip features, auxiliary pins, transfer protocol, connectors and API for communication between an embedded instrumentation and a host computer

• IEEE-ISTO 5001-1999

• IEEE-ISTO 5001-2003
Nexus - IEEE 5001

- Focus has been on implementing vendor neutral high performance trace and calibration solution
  - Moving out of traditional Automotive IC focus
  - Many solutions have been implemented in silicon and tools
- 2007 Spec Updates are ongoing
  - SerDes IO for debug – Aurora protocol
  - Convergence with P1049.7 – 2 wire JTAG

www.nexus5001.org
Nexus Port Configurations

- Aux Port only: Auxiliary Input and Output ports
- Combined JTAG/Aux Port
  - Commands and Responses go in and out of the JTAG port, including Read and Write accesses
  - Auxiliary Out is used primarily for trace output

- Aux Input Port
  - MDI: Message Data In
  - MSEI: Message Start End Input
  - MCKI: Message Input Clock
  - EVTI

- Aux Output Port
  - MDO: Message Data Out
  - MCKO: Message Output Clock
  - MSEO: Message Start/End Output
  - EVTO: Event Output

- JTAG
  - TDI: JTAG Input Data
  - TDO: JTAG Output Data
  - TCK: JTAG Clock
  - TMS: Sequencing Control

Aux Port
Combined JTAG/Aux Port
OCP-IP Debug Overview

- OCP is neutral On Chip Protocol for SoC IP connection
- Debug Working Group Defining instrumentation related signaling between cores and other embedded subsystems
- Leverage other work for IOs (JTAG, Nexus), APIs, etc.
- Defined subset architecture for on chip debug
  - Debug Proposal is in review by OCP Standards group

OCP 2.1 Socket

- Standardized Bus Transfer Master and Slave sockets
- RISC
- DSP
- Other IP
- Mem Ctrlr
- RAM

OCP Debug Socket

- Debug Sockets
- Debug handshakes
- Cross Triggers
- Trace Synchronizers
- Trace Triggers
- Power Management
- Debug Security

Socket Specific Features

RISC
DSP
Mem Ctrlr
RAM
Other IP
Other IP
OCP-IP Systems Debug

SoC Debug focus on
5 Hardware and 2 Software Interfaces

SoC Instrument connections
1. CORE-INTERFACE
2. BUS-INTERFACE
3. CROSS-TRIGGER INTERFACE
4. PIN-INTERFACE CONTROL
5. PIN-INTERFACE DATA

System Debug SW Issues
A. Debugger Tools API
B. EDA APIs

www.ocpip.org
MIPI T&DWG

- MIPI - Mobile Industry Processor Interface
- Debug and Test Interface specification
  - A low and high (>1Gbit/s) mode trace port for real-time trace.
  - System Trace Module (STM) for software debug
  - Connector Recommendations

- System Trace Protocol (STP) Specification
- Parallel Trace Interface (PTI) Specification
- Basic Debug Connector Recommendation

www.mipi.org
1149.7 – AJTAG

- Superset of IEEE 1149.1
- Addressing high performance Two wire JTAG solution
  - New protocol with additional levels of control hierarchy
    - SW mode switches from standard and advanced protocol
    - Non Serial mode of operations
    - Better JTAG utilization – less time in idling, pause states

For more information contact Gary Swoboda, g-swoboda@ti.com
SPIRIT Debug WG

SPIRIT Industry Consortium for EDA tools architectural exchange
– IP-XACT standard - human and machine-independent XML format
– Describes configuration / integration data for IP
  • Debug extensions for core registers, buses instances, connectivity, timing, etc.

IP vendor

SoC

System designer

Software developer

Emulator
Reads debug access descriptions from IP-XACT files

EDA / ESL Tools

Debugger
Reads programmer’s model from IP-XACT files

Component, I/O, IP memory map.

Topology, System Memory Map

www.spiritconsortium.org
Lessons Learned since 2002

- JTAG remains key – JTAG as the catchall debug interface is pervasive norm
- Keep stuff small – IC designers have sensitivity to adding gate count for debug IP
- Cross Triggering is key – Coordinate multicore debug
- Customers like standards… sometimes … still lots of proprietary buses, IP debug approaches, etc.
- Custom(er) is king – SoC architectures vary, therefore debug requirements and solutions do as well
- Work with your Eco-system - Debug does not stand-alone – close ties to EDA flow, IP, test, SW are needed …
- It is all about the tools – Interfaces / SW tools have significant and ongoing effort to use new debug features
Summing it up

• Multi-core SoC requires next wave in debug
  – Industry at large is to recognizing needs for On Chip Debug
  – Standards efforts on Architectures, interfaces, tools

• IC debug is still running on 15 year old JTAG concepts
  – Several efforts to re-vitalize JTAG (1149.7, P1687, …)

• Debug fills a hole in SoC design/verification flow
  – Analysis not be otherwise provided
  – Not just debug - multi-core optimization, performance analysis, etc.

• Value of on-chip Debug
  More design visibility + better Real Time Analysis
  Faster time to market, better silicon quality, Better ease of use
Next Stage Debug Challenges
Integrating ESL and Debug

SLD Virtual SoC

RTL Physical SoC