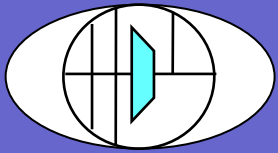


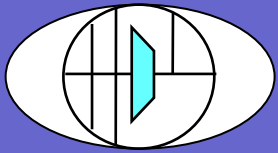
On-Chip Instrument Assertions for OCP SoC Analysis

Dr. Neal Stollon, HDL Dynamics
neals@hldynamics.com



Topic Overview

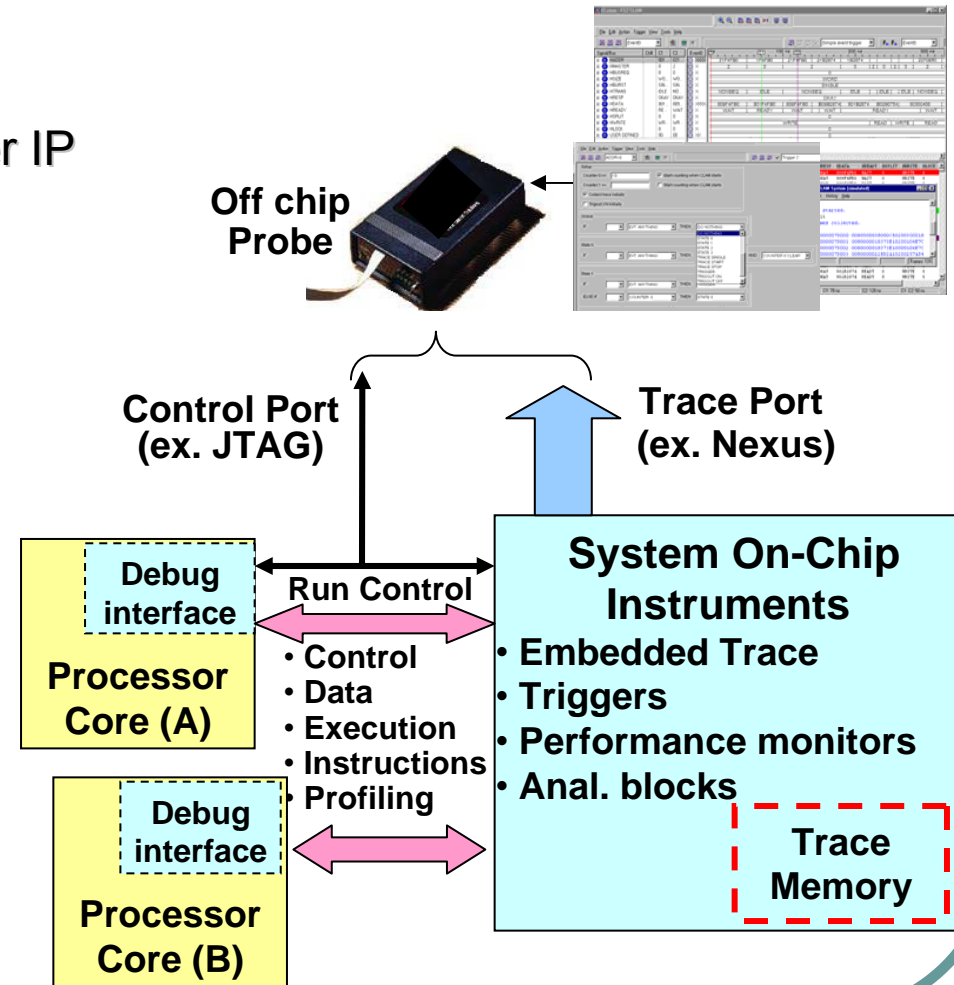
- On-Chip Instrumentation Overview
- Application Example
 - Work with Temento Systems
 - On-Chip Assertions and Bus Trace

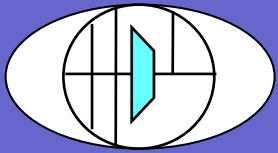


What is On-Chip Instrumentation?

Range of On-Chip Instruments

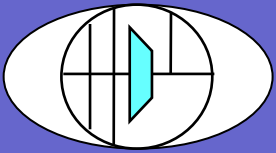
- **Logic Event monitoring**
 - Parallel/Serial/User Trigger IP
 - Assertion Checker IP
- **Processor & bus analysis**
 - Bus Tracer
 - Bus Range Checker IP
 - Processor Run Control
 - Instruction Trace
- **Storage of trace data**
 - Transaction Registers
 - History Registers
 - Trace RAM
- **Post capture tools**
 - Fault Finder
 - IP map back to RTL
 - Profiling tools



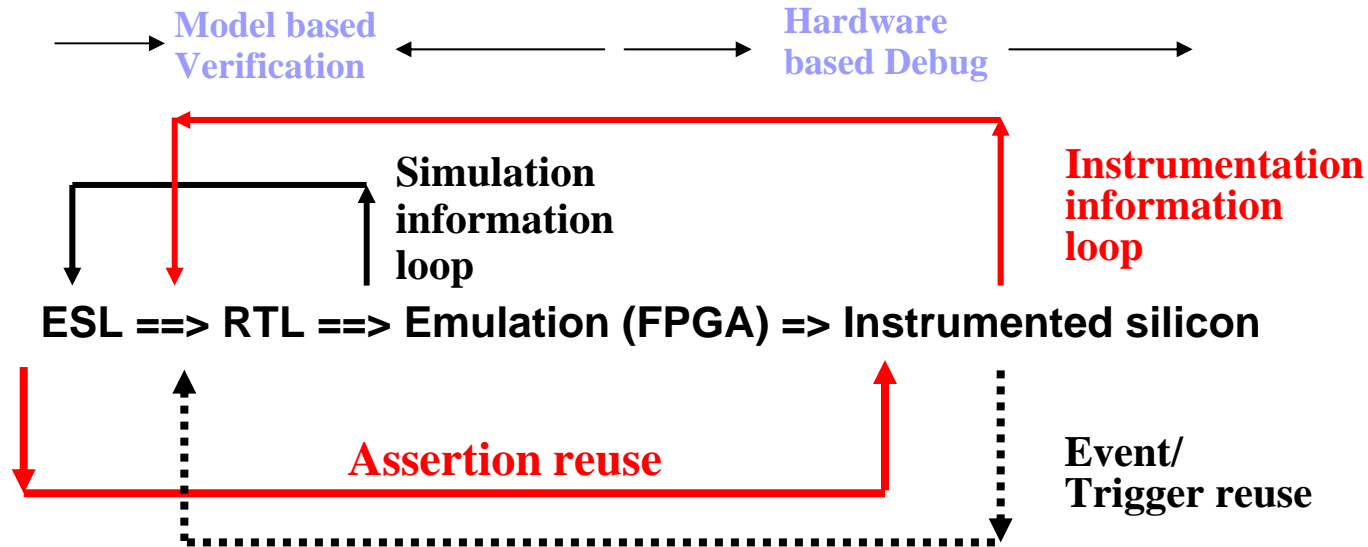


Why On-Chip Instrumentation

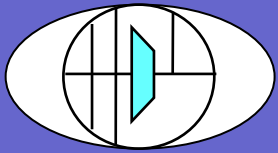
- System and SoC Debug as part of a silicon EDA/Test flow
 - Can't fix what you can't see
 - Instrument hardware = Better SoC analysis
- Improved software analysis
 - Better visualization of operations
 - Localizing problems in software



Assertion Verification with Instrumentation

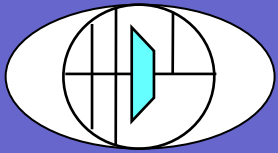


- Many System Assertions are synthesizable
- Synthesizable assertions support debug methods
 - Create Event based Monitoring, Triggering, etc...
 - Reuse Event/Trigger descriptions as assertions
- Instrumentation allows validation of assertions



Industry Efforts supporting OCP

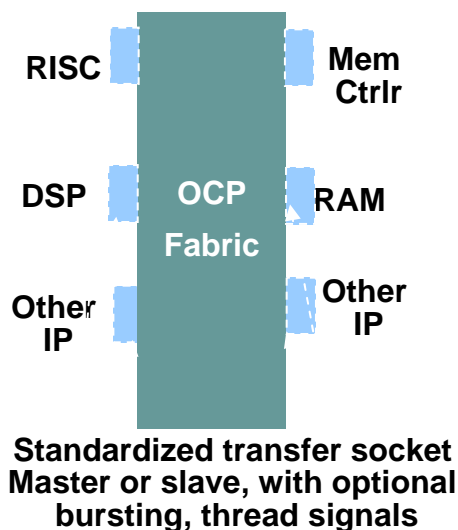
- **Several Ongoing industry driven standards and methodology efforts**
 - **OCP-IP Debug Working Group**
 - **IEEE std 5001 Nexus partnership**
- **On-Chip Instrument vendors**
 - **Temento Systems - Dialite**



OCP Debug Methodology

- OCP-IP On-Chip signal Protocol for SoC IP connection
- **OCP-IP Debug WG announced Debug Standard in 2007**
- Defined On-Chip instrument architecture www.ocpip.org/socket/whitepapers

OCP 2.1 Socket

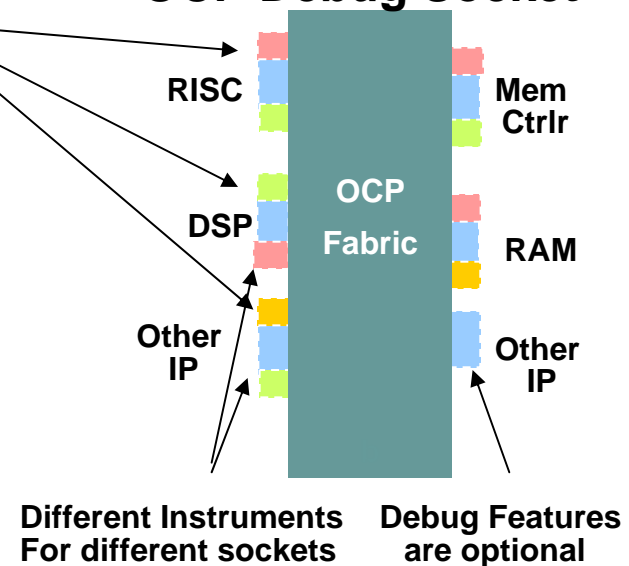


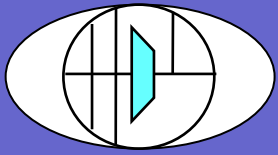
Baseline Instrument

Sockets

1149.1 JTAG
Debug resets
Generic Processor
debug handshakes
Cross Trigger Interfaces
Synchronize Trace/Debug
Trace Triggers
TimeStamp Interfaces
Power Management
Debug Security

OCP Debug Socket

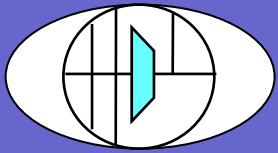




OCP Instrumentation Sockets

Define Instrumentation related signaling between cores and other embedded subsystems

- **Identifies Basic and Extended sets of socket level signals**
- **Leverage other work for IOs (JTAG, Nexus), APIs, etc**
- **Supports debugging of multiple processor cores connected with the OCP interface.**
 - **Different solutions supported using OCP Debug wrapper**
- **Allows designers to distribute instrument signals as part of the system interface scheme.**
- **Ability to create multi-core (heterogeneous processors) instrument hardware and software.**



OCP Instrument Interfaces

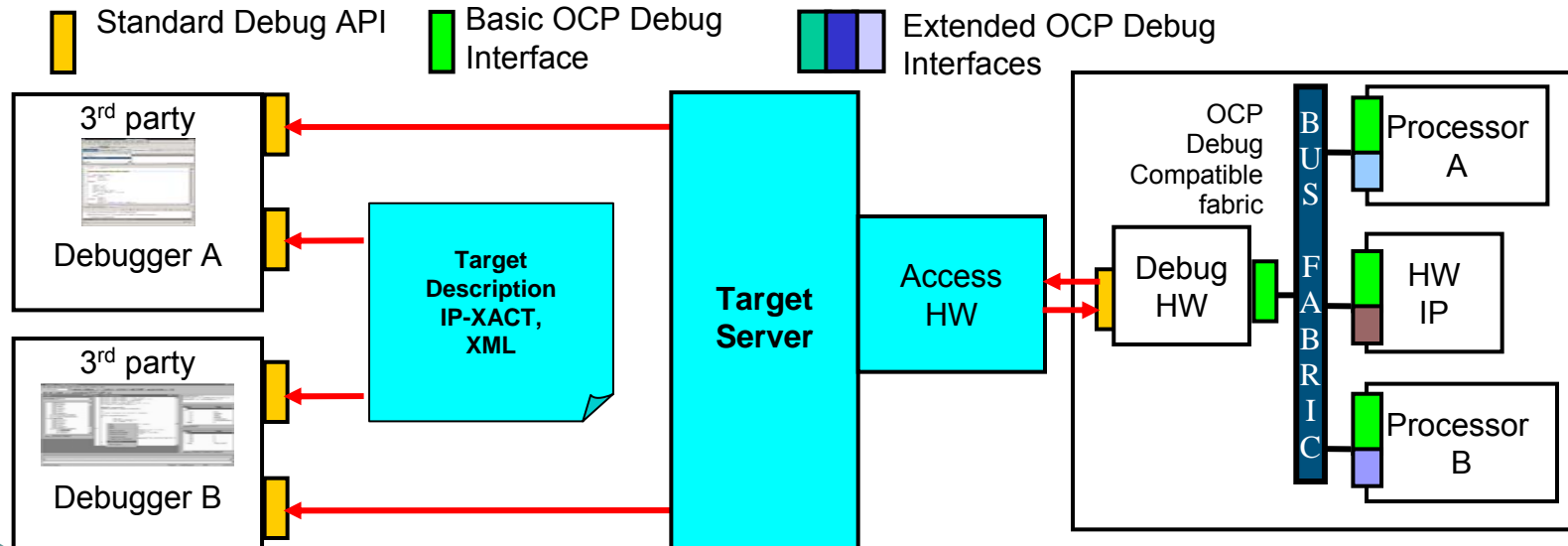
OCP Basic Debug Signals Interface

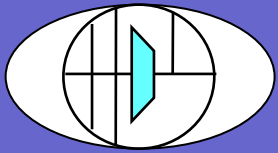
- Debug Control and Run Control for Cores
- Cross-Triggering between Multiple Cores and Events
- Bus traffic observation/System Trace Interface

huge value to final silicon debug

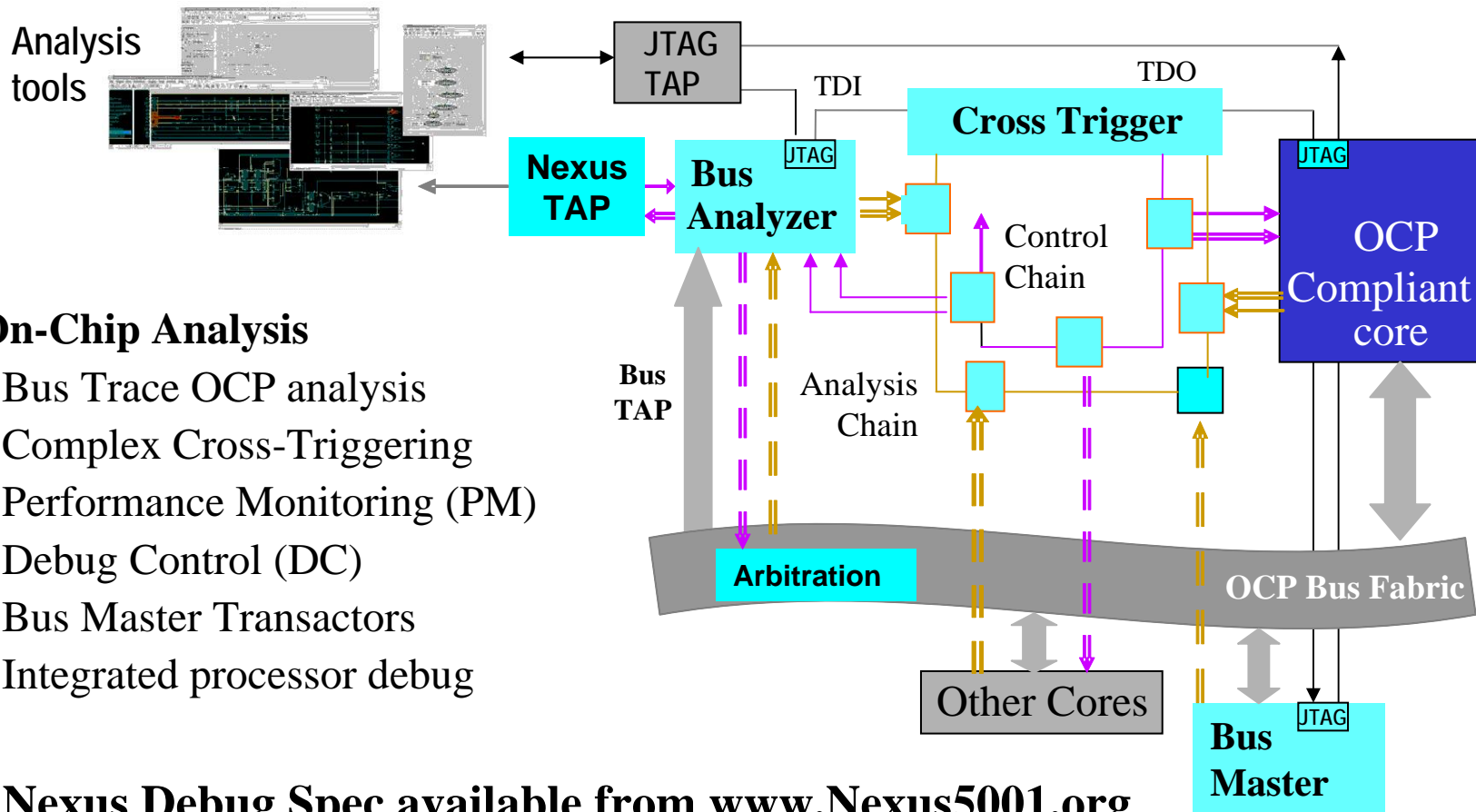
OCP Extended Signals Interface (Special features)

- Performance Monitors, Asynchronous Time-stamps
- Power Monitoring, Security Monitoring





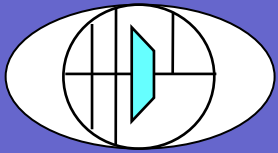
System Debug Instrumentation OCP/Nexus (IEEE 5001) Example



On-Chip Analysis

- Bus Trace OCP analysis
- Complex Cross-Trigging
- Performance Monitoring (PM)
- Debug Control (DC)
- Bus Master Transactors
- Integrated processor debug

- **Nexus Debug Spec** available from www.Nexus5001.org
- **OCP Debug Spec** available from www.ocpip.org



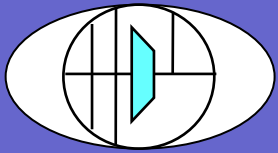
Example Application Solution

- **Work with Temento Systems**
 - **System enhancement of Dialite instrumentation product**
 - **Instrumentation insertion from ESL flow**
 - **Debug data insertion to ESL flow**
 - **Tradeoffs of abstractions and timing vs. resources**

Available as part of current Dialite product release

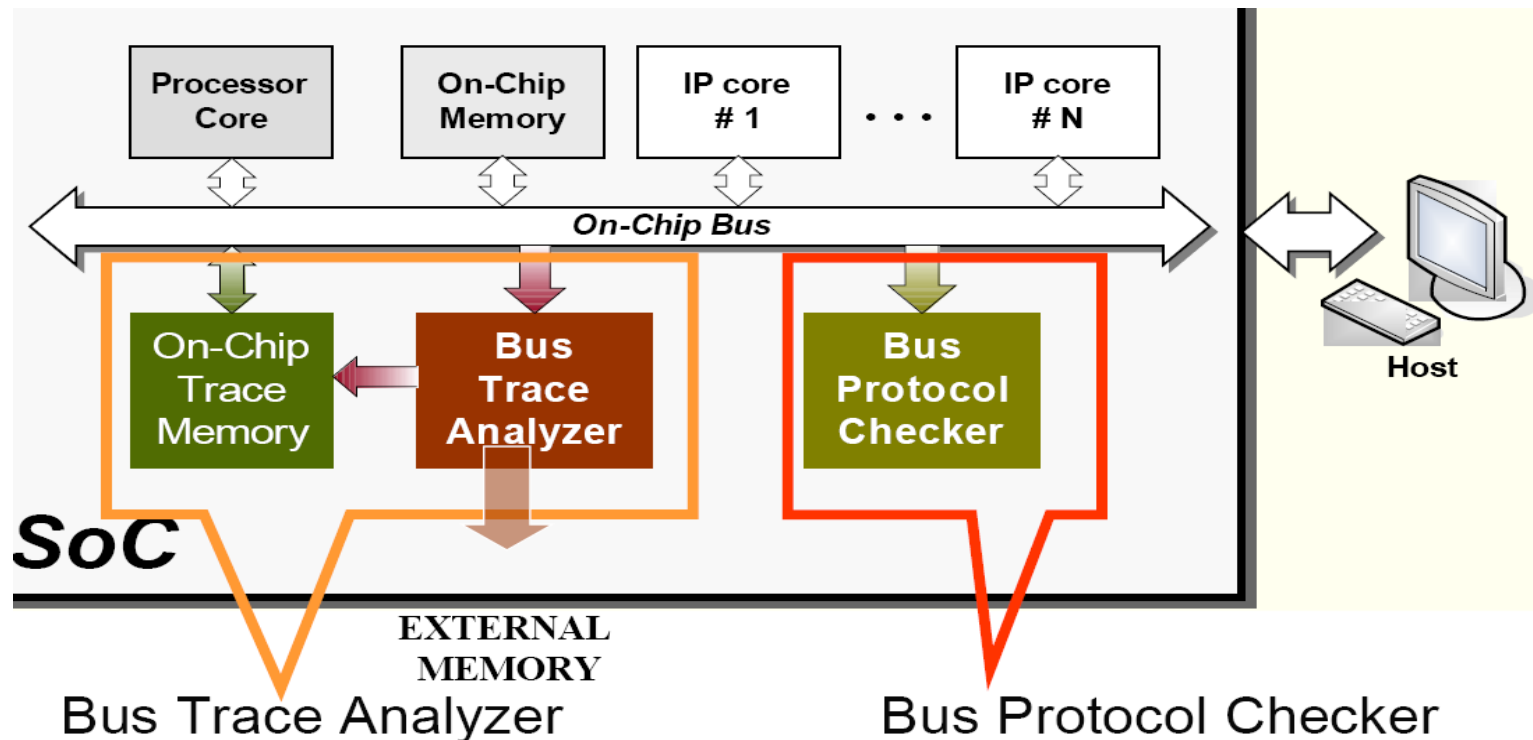
Contact Author for details

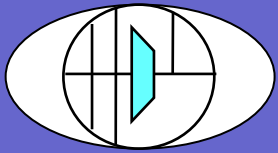
HDL Dynamics is USA representative for Temento Systems



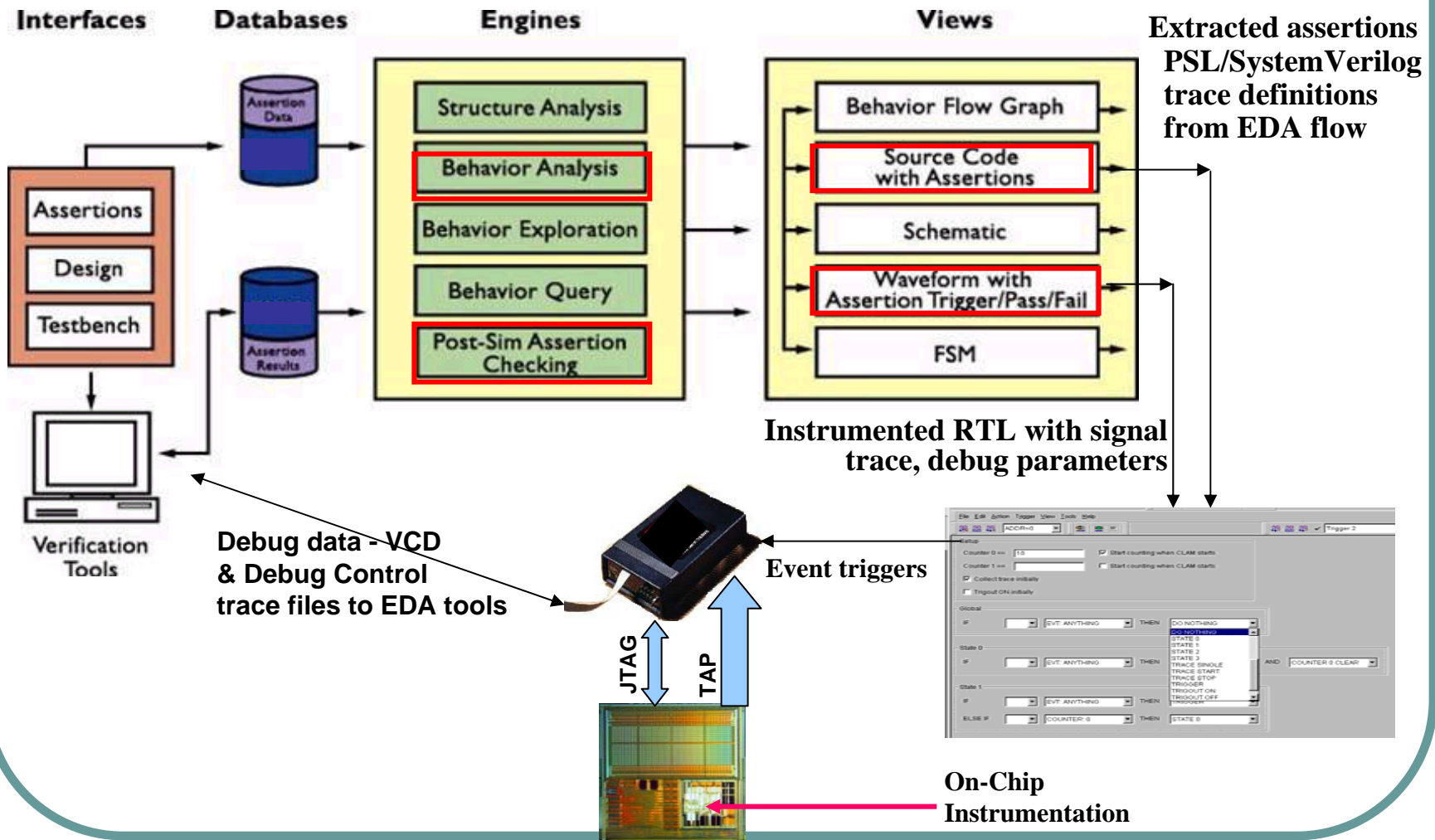
OCP Bus Trace and Analysis

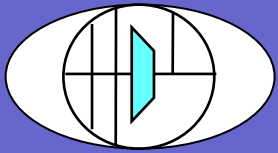
- Record On-Chip Bus Transactions using instrumentation
 - with tradeoff between accuracy and depth of recording
- Check protocols using assertions





Dialite Chip Instrumentation flow

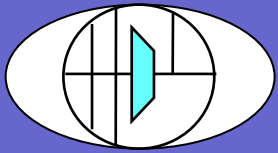




Dialite instruments

•Types of on-chip instruments available

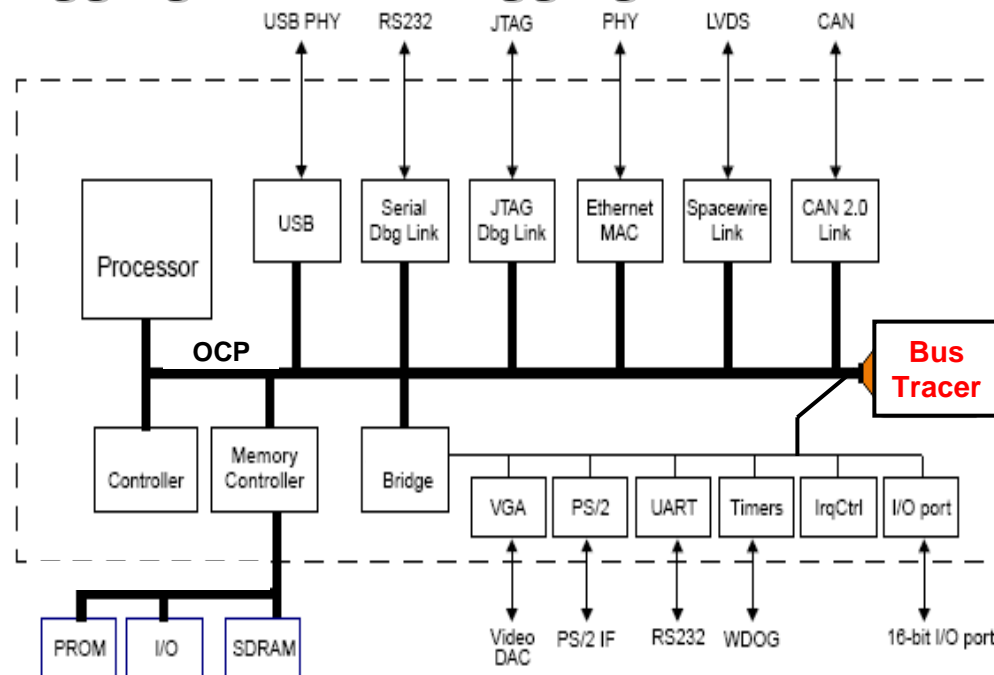
Logic Debug	Debug Switches	Interactive control and setting signals to desired values
Logic Debug	Virtual Debug LEDs	Interactive observing of internal signal values
Logic Debug	Debug Logic Module	Custom triggering on specific event (FSM sequences, assertions...)
Trigger/Analysis	Glitch Detector	Signal glitch tracking and system synchronization
Trigger/Analysis	Logic Equation Module	Monitor/Trigger on real-time adjustable logic equation of two signals
Trigger/Analysis	Parallel Trigger	Monitor/Trigger on Pattern recognition of parallel signals
Trigger/Analysis	Serial Trigger	Monitor/Trigger on pattern recognition of serial/sequential signals
Trigger/Analysis	Counter Marker	Trigger after a fixed duration or fixed occurrence of an event
Trace	History Register	Trigger based sequential recording into RAM or registers
Trace	Transaction Register	Trigger based selective recording of key events into RAM/registers
System Analysis	Bus Range Monitoring	Capturing signal value inside or outside a given range
System Analysis	Bus Trace Analyzer	Monitoring and compressed data trace compression computing.
System Analysis	Bus Protocol Monitor	Check bus transactions based on defined rules of bus behaviors
System Analysis	Traffic Analyzer	Analyze bus traffic metrics (DMA, stalls, latencies) over long periods
Verify Properties	Assertion Checker	Hardware verification of PSL/SVA system properties
Logic Debug	HDL Fault Finder	Monitoring events by inserting Watchpoints and Breakpoints
Stimulate Signals	Pseudo Random Gen.	Inject pseudo random signal sequences for robustness checking

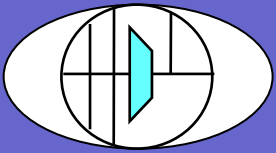


“Simple” OCP Bus Architecture

- Debug of complex design composed of differing processors and IPs
- Different levels of verification required

Signal Debugging **HDL Debugging** **Assertion Debugging**

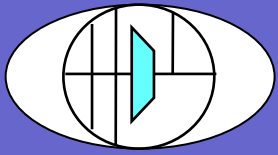




Bus Instrumentation Assertions

Example of supported properties

```
    assert always {sig1} ==> {sig2};
    assert always {sig1;sig2} -> {sig3; sig4};
    assert always {sigA} ==> {req; busy; gnt};
assert always {sigA} ==> {req & !busy & !gnt; !req & busy & !gnt; !req & !busy &
                        gnt};
    assert always {sigA} ==> {req; busy; busy; busy; gnt};
    assert always {sigA} ==> {req; busy[*3]; gnt};
    assert never (sig1 -> (sig2 before sig3));
    assert never (sig1 -> sig2 );
    assert always req -> next ack;
    assert always req -> (req until_ grant);
    assert always sig -> (sig1 before sig2);
    assert always sig -> (sig1 before_ sig2);
    assert always {sig1} -> {sig2[*2]; sig3};
    assert always (grant -> prev(req));
sequence s1 = {req; !req; req; !req}; assert always req -> s1;
```

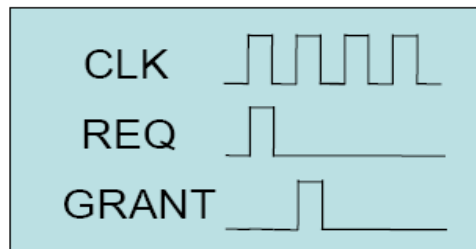


Assertion Based Bus Instrumentation Triggering

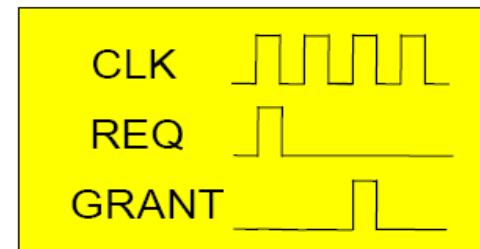
PSL assertion example

```
assert always req -> next grant;
```

This assertion means that whenever the HDL signal req is true, the HDL signal grant must be true in the following cycle. A cycle is specified using a given clock signal.



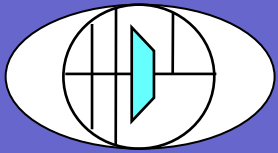
Transaction OK



Error (grant arrives too late)

```
assert always req -> next (ack until grant);
```

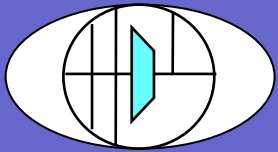
This asserts that whenever req is true, ack is true in the following cycle and ack remains true until the first subsequent cycle in which grant is true.



Dialite Debug Cockpit Instrument Configuration/Insertion

The screenshot displays the Dialite Debug Cockpit software interface. The main window is titled "DLI DEMO STRATIX_47 - DLI Dialite - Instrumentation - [DLI DEMO STRATIX.prg]". The interface is divided into several panes:

- Left Pane:** A tree view showing the project structure, including "Instruments" (IRQ_TRIGGER, BRIDGE_DATA, PWM_ST, PWM_HR_2, PWM_COMMAND, PWM_PRG, MEMORY_TRIGGER, MEMORY_LEM, ADDRESS_BUS_CHECKER, MEMORY_TRANSACTIONS, BRIDGE_TA, BRIDGE_DEBUG, PWM_ULM, PWM_HR, BRIDGE_COUNTER_TRIGGER, ACK_COUNTER), "State machines", and "Sources" (PWM.vhd, regulator.vhd, PWM_Module.vhd, bridge.vhd, sdram_pll_bb.v, delay_reset_block_bb.v, ref_32_system_bb.v, standard_32.v).
- Middle Pane:** A configuration window for "DLI DEMO STRATIX: PWM_Module" showing "interface" and "internal" components. The "interface" section includes "command[6:0]", "regul_done", "regul_to_pwm[6:0]", "reset", and "pwm_synchro". The "internal" section includes "regul_to_pwm_in[6:0]" and "pwm_synchro".
- Right Pane:** A diagram titled "Instrument integration" showing various instrument icons (IRQ_TRIGGER, BRIDGE_DATA, ADDRESS_BUS_CHECKER, MEMORY_TRIGGER, BRIDGE_TA, PWM_ULM, PWM_HR, PWM_ST, PWM_HR_2, BRIDGE_DEBUG, ACK_COUNTER, PWM_COMMAND, PWM_PRG) connected to a central "MEMORY_LEM" component.
- Bottom Pane:** A log window titled "Design loading" showing the following messages:
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\delay_reset_block_bb.v
 - Preprocessing file ..\OriginalProject\Blackboxes\sdram_pll_bb.v
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\sdram_pll_bb.v
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\bridge.vhd
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\PWM_Module.vhd
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\regulator.vhd
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\TmpSource\PWM.vhd
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\ref\PWM_ULM_Regul_Monitoring.vhd
 - Loading file C:\Sebastien\DLI_DEMO_STRATIX_47\1b20\DLI_DEMO_STRATIX\HDL\refAck_counter.v



Dialite Debug Cockpit Results Display and Analysis

GTKWave - D...leon3_stratixii_demoDLI_AMBA_TRACER.vcd

File Edit Search Time Markers View Help

From: 0 To: 92 Maximum Time : 48450 - Current Time : 23

Signals

- DLI.HCLK=1
- DLI.HADDR[31:0]=0000023C
- DLI.HRDATA[31:0]=12BFFFF5
- DLI.HWDATA[31:0]=ZZZZZZZZ
- DLI.HWRITE=0
- DLI.HSIZE[2:0]=2
- DLI.HBURST[2:0]=1
- DLI.HPROT[3:0]=E
- DLI.Master0_HBUSREQ=0
- DLI.HRESP[1:0]=0
- DLI.HTRANS[1:0]=0
- DLI.HREADY=1
- DLI.Slave2_HSEL=0
- DLI.Slave1_HSEL=1
- DLI.Slave0_HSEL=0

Waves

Bus Tracer Waveform

```
119 when "00" => null; -- idle
114 when "01" =>
117 if r.hwwrite = '0' then v.penable := '1';
118 else v.pndata := abhi.hwdata; end if;
119 v.psel := '1'; v.state := "10";
121 when others =>
124 if r.penable = '0' then v.psel := '1'; v.penable := '1'; end if;
122 v.state := "00"; v.hready := '1';
123 end case;
```

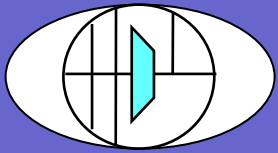
Time	Address	Read Data	Write Data	Direction	Size	Burst	Protection E...	Bus Reques...	Response	Transfer Type	Ready	Slave
0	0x0000022C	-	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
1	0x00000230	0xDA21000A	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
2	0x00000234	0xDA21000A	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
3	0x00000238	0xD600A018	-	WRITE	32	SINGLE	DATA ACC...	0	OKAY	SEQUENTIAL	1	1
4	0x00000238	-	0x76458020	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
5	0x80000318	0x8210600F	-	READ	32	SINGLE	DATA ACC...	0	OKAY	SEQUENTIAL	1	0
6	0x0000023C	-	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	0	0
7	0x0000023C	-	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	0	0
8	0x0000023C	0x00000008	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
9	0x0000023C	0x12BFFFF5	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
10	0x0000023C	0x12BFFFF5	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
11	0x0000023C	0x12BFFFF5	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0
12	0x00000240	0x12BFFFF5	-	READ	32	INCR	OPCODE F...	0	OKAY	IDLE	1	0

HDL Fault Finder Stack

Bus Tracer Stack

Traffic Analyzer

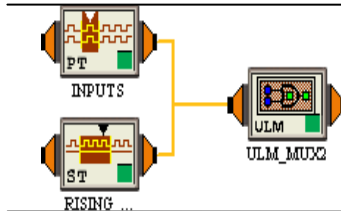
Traffic analyzer statistics: Min: 0% Max: 100% Average: 17.3% Status flags: Overflow Empty



Dialite Instrumentation Summary



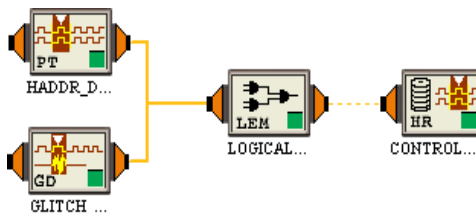
Assertion Checker : Check OCP transactions
Switch/Leds : Check current state of external IO



Capture OCP bus transactions during execution including boot, reset, etc.



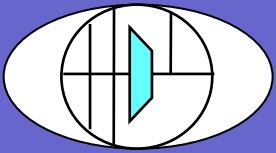
Isolate OCP transactions that access memory (instruction fetch, RAM/register read/write)



Concentrate on control signals associated to each OCP transaction (write strobe, protection signal, selection signal, ...)



Check internal signal behaviour according to HDL code in order to find casual errors or bugs



OCP Instrumentation Summary

- On-Chip Instrumentations provide important analysis/verification
- Growing recognition of value of standards based integration
- Design flows need to include
 - Value from consistency, reuse in debug and verification
 - Standards based integration makes the difference
- OCP On-Chip Instrumentation IP and solutions are available
 - Instrumentation Tools, Interfaces for chip event monitoring, etc.
 - Assertions simplify bus level debug and analysis
- HDL Dynamics is working with Temento and others to develop Instrumentation based Debug solutions