A Debug Probe for Concurrently Debugging Multiple Embedded Cores and Inter-Core Transactions in NoC-Based Systems

Shan Tang and Qiang Xu

EDA&Testing Laboratory

The Chinese University of Hong Kong 香港中文大學

Background

Silicon Debug for Single Core

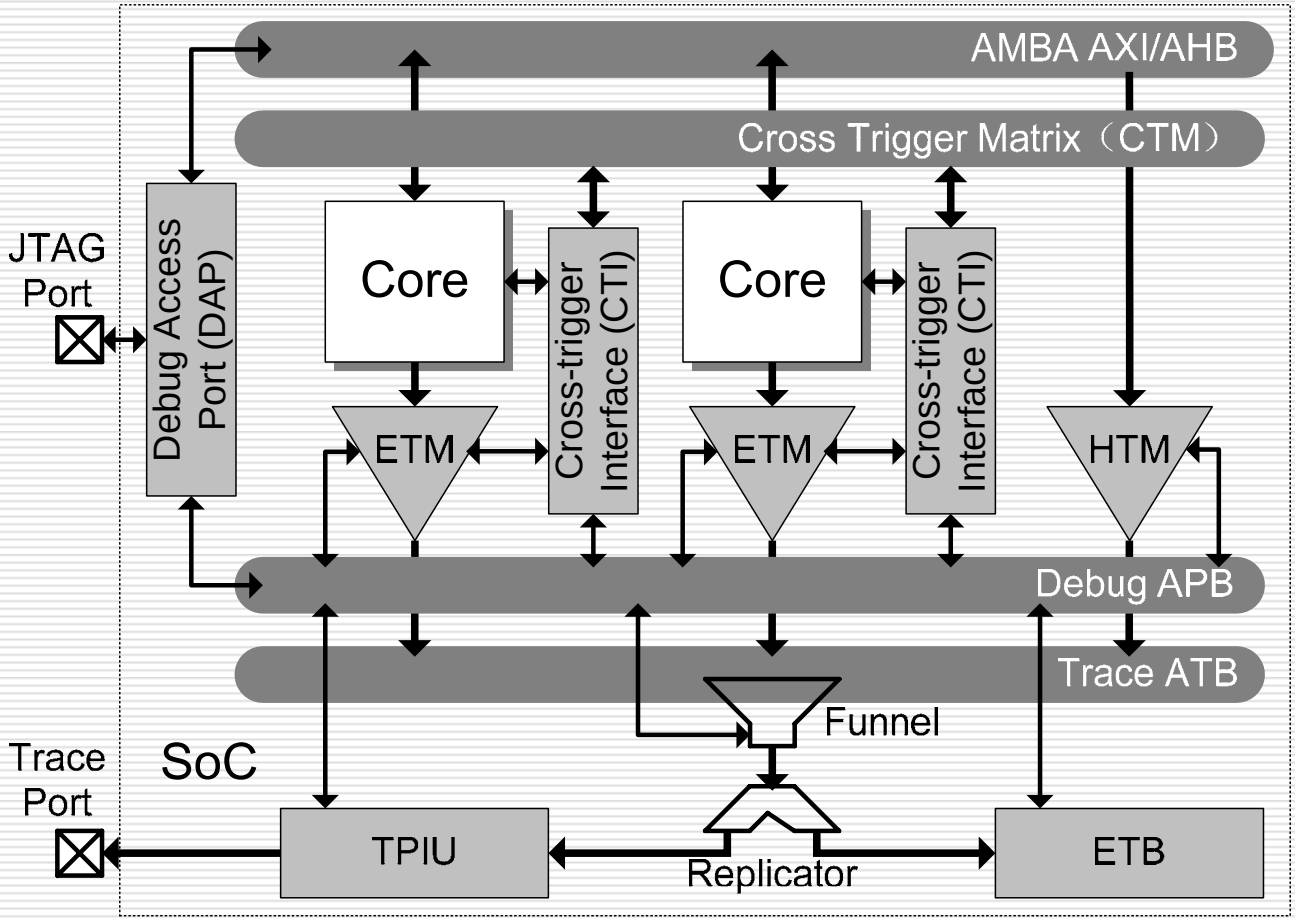
- Run Control Interface (e.g., JTAG)
  - Widely used in practice
  - Not enough for tricky bugs
  - Not applicable for certain real-time applications
- Trace + Trigger
  - Effective in most cases

Challenging yet Well Studied Problem!

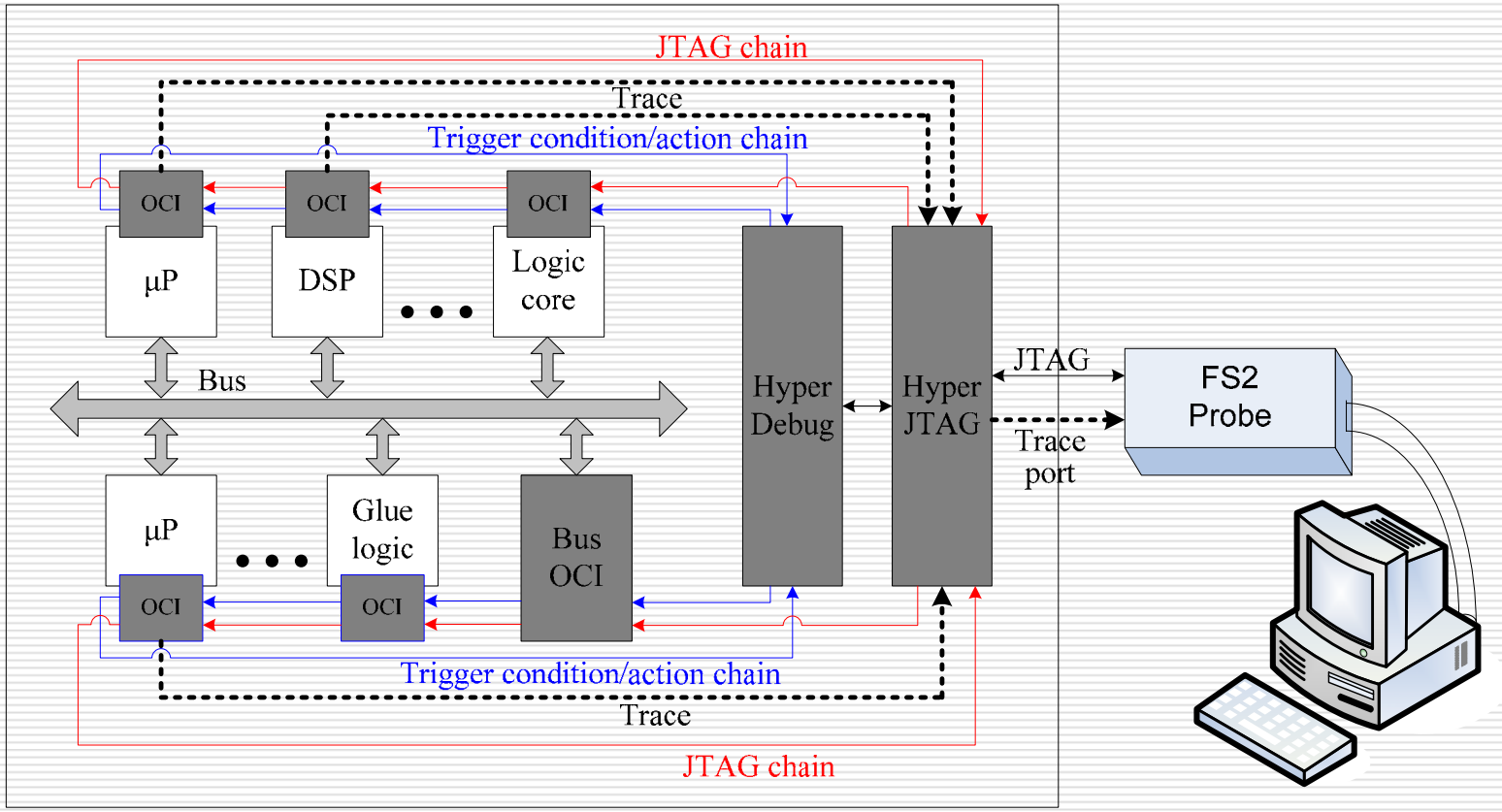
Multi-Core Debug - Requirements

- Concurrent debug access to interacting cores and their transactions
- System-level triggering and trace
- Debug event synchronization for cores from multiple clock domains
- Limited DfD cost in terms of silicon area, routing and device pins.

Multi-Core Debug Architecture - ARM CoreSight

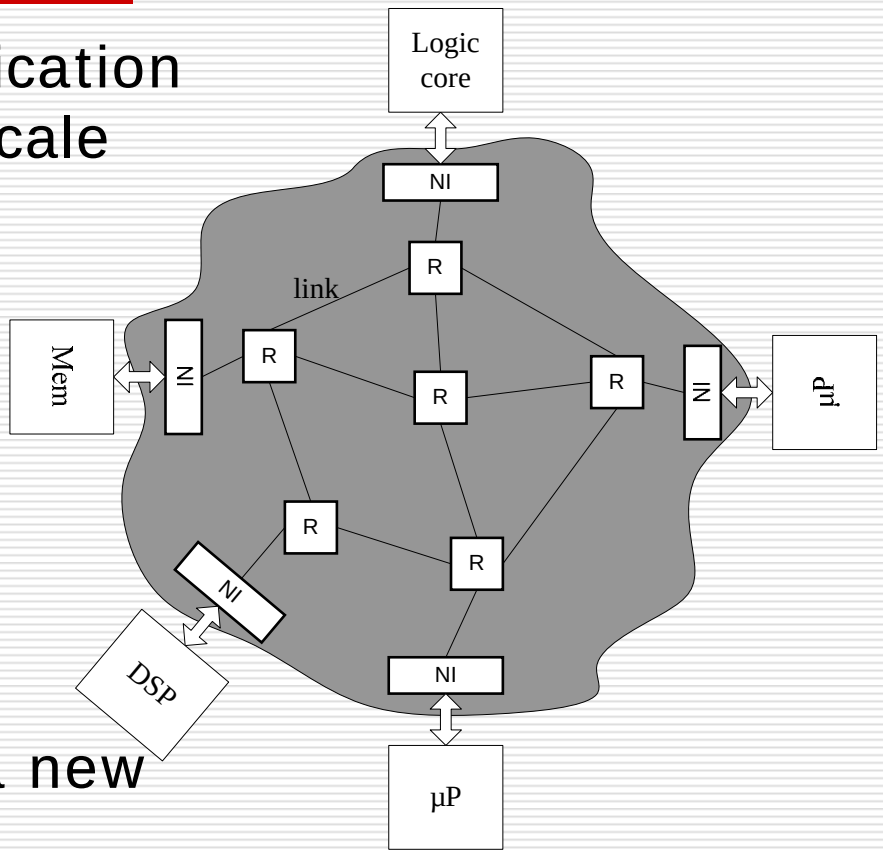


Multi-Core Debug Architecture – First Silicon



Network-on-Chip

- Most promising communication scheme for future giga-scale SoCs
- NoC generally contains:
  - Network interface
  - Router
  - Physical link
- Need debug support as a new design paradigm



NoC Monitoring Service

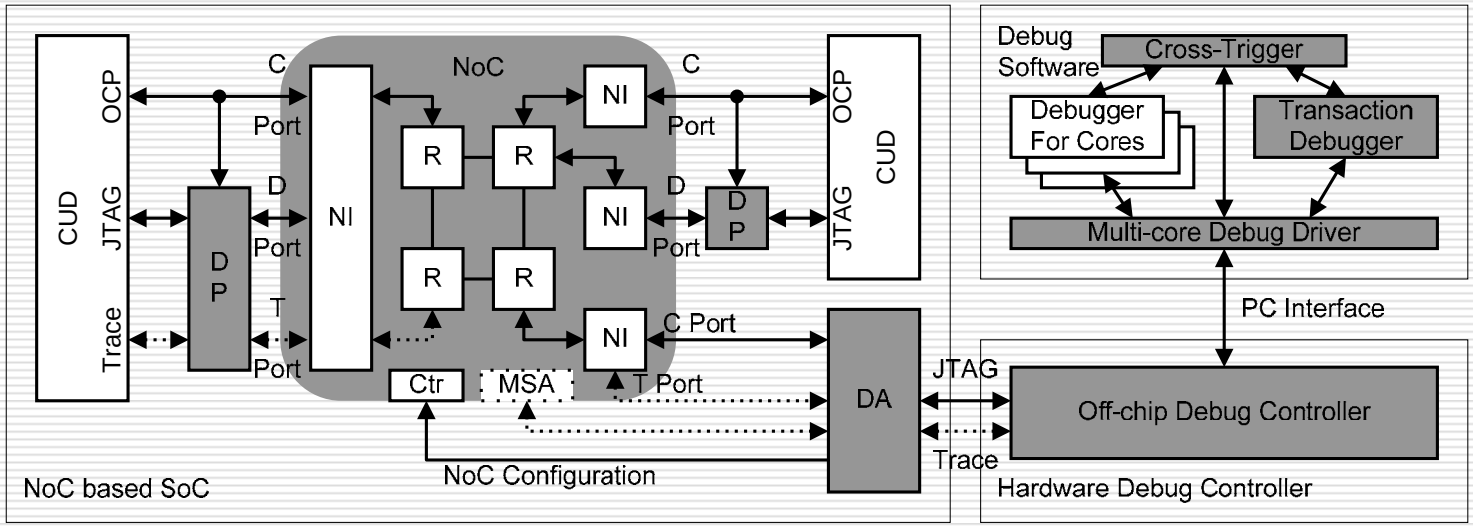
- Ciordas et al. TODAES'05, IES'06
  - Monitoring probe attached at routers
  - Effective identifying bit-level errors
  - Costly in terms of NoC bandwidth at transaction level
  - Monitor instead of Debug

Multi-Core Debug Platform for NoC-Based Systems

Rationale

- How to achieve concurrent debug access?
  - Reuse NoC to transfer debug data
  - Insert debug probe between core and NI
- How to monitor inter-core transactions in NoC-based systems
  - Not shared mechanism – cannot simply listen
- How to deal with the latency problem?
  - Use QoS guaranteed service for debug connections
  - Two-pass debug strategy

Proposed Platform



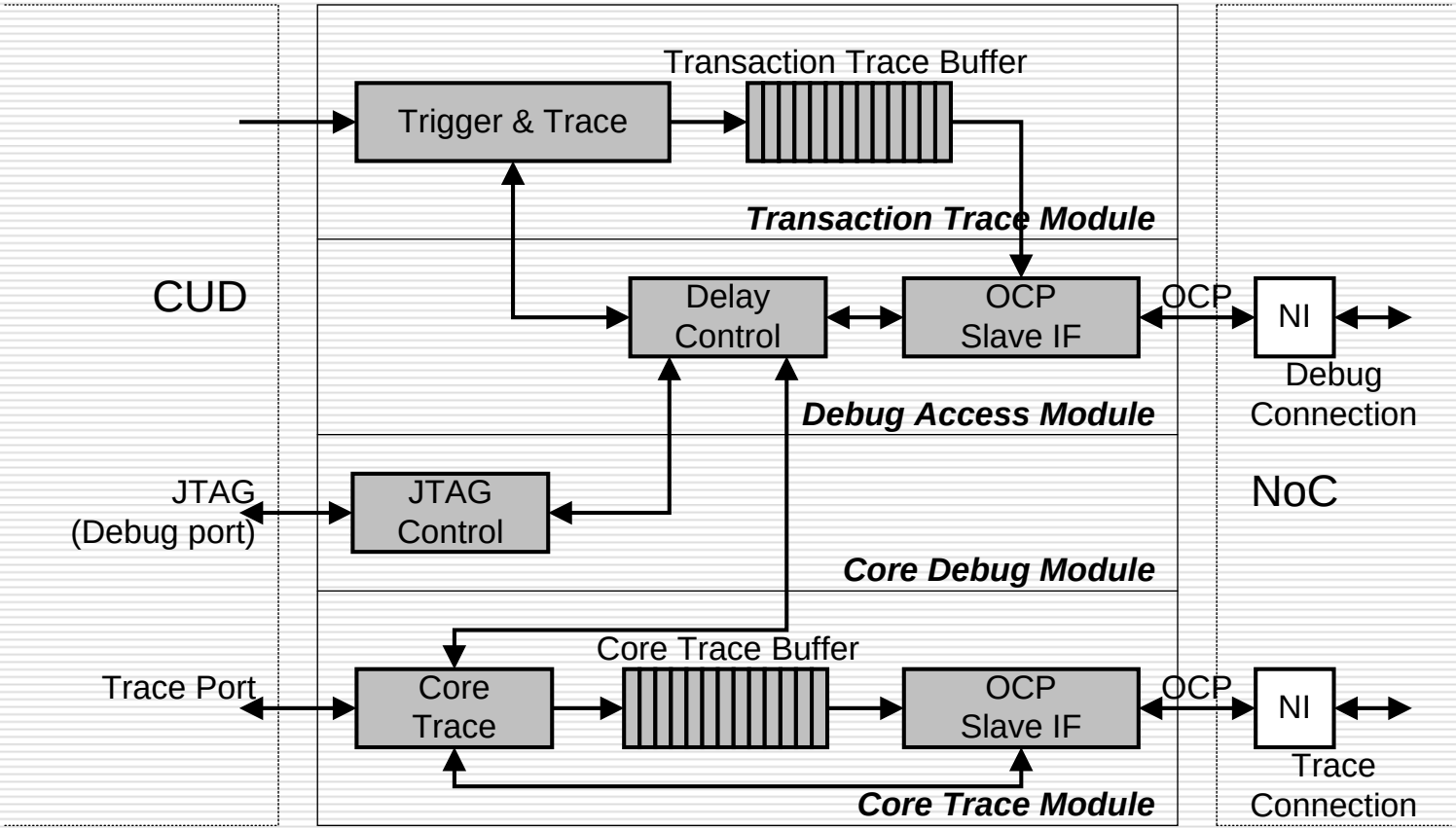
■C port: Communication Port ■D port: Debug Port ■T port: Trace Port ■MSA: Monitor Service Access

- On-Chip Debug Architecture
- Off-Chip Debug Controller
- Supporting Debug Software

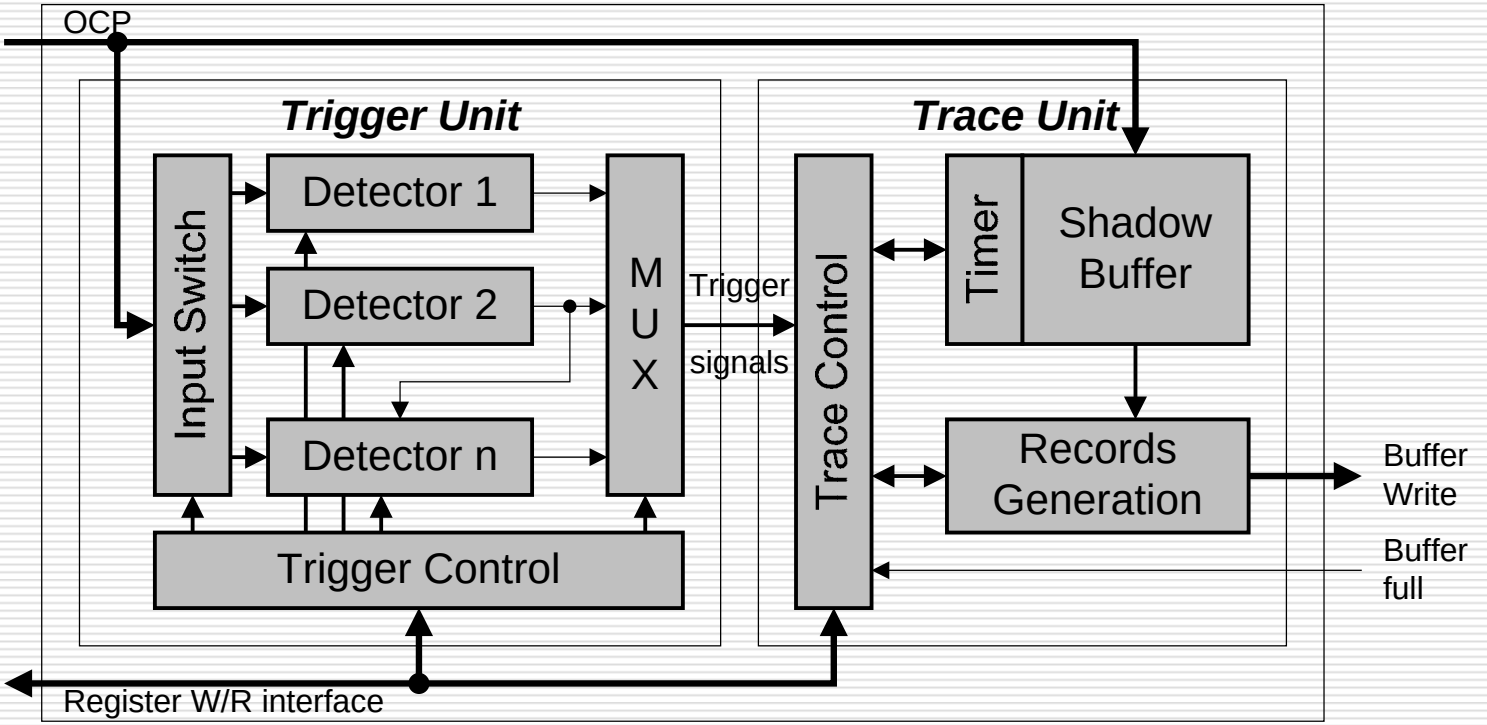
On-Chip Debug Architecture

- Core-Level Debug Probe
  - Between core and NI
  - Monitor transactions
  - Control/observe core's debug interface
- System-level Debug Agent
  - JTAG (+ trace port)
  - Controlled by off-chip debug controller

Debug Probe Design

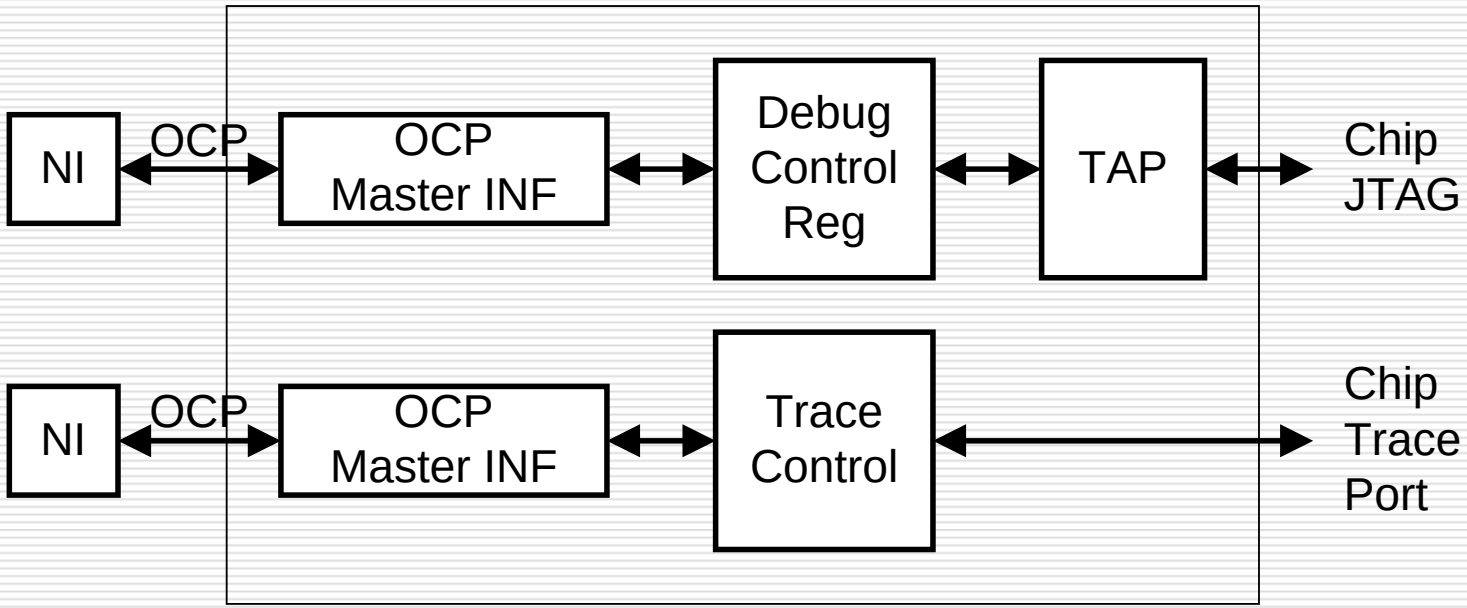


Trigger & Trace Unit Design



Debug Agent

- TAP controlled by off-chip debug controller
- Build debug connections between DA and DPs



Control On-Chip Debug Registers through DA

Off-Chip Debug Controller	**Debug Agent**
Write 'DEBUG_REG' command into IR;	Use DEBUG_REG_DATA register as DR;
Write operation	
Shift in 'write' command to DR: :WR:ADDR:DATA:-:	Write specified register;
Read operation	
Shift in 'read' command to DR: :RD:ADDR:-:-: Do {Shift out the contents of DR;} While (READ_VALID) != '1'; The DATA field in RD is the valid data;	Read specified register; Set READ_VALID when data is ready;

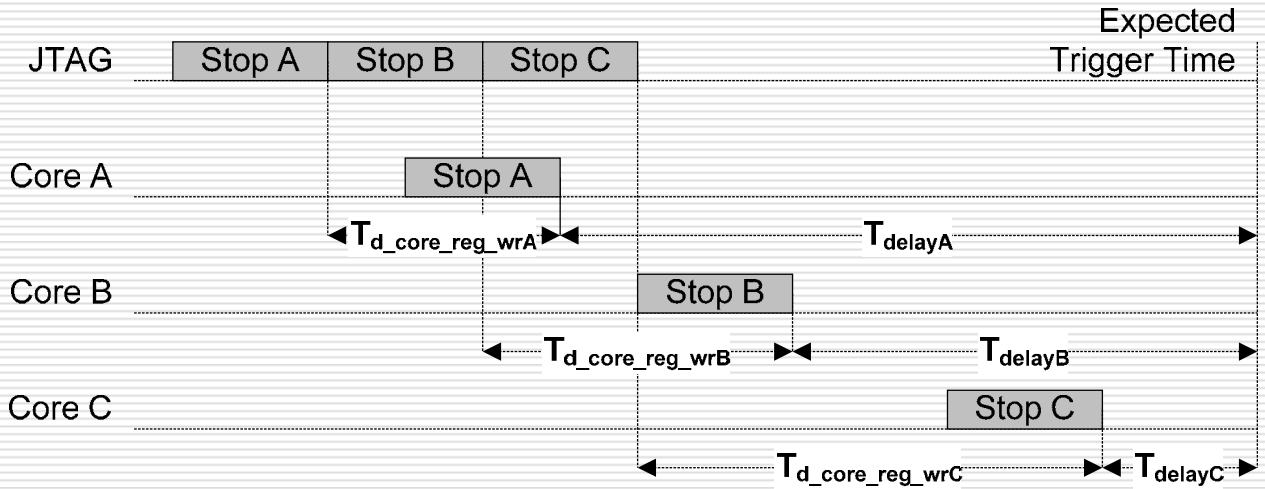
Supporting Debug Software

- Provide GUI or command line interface

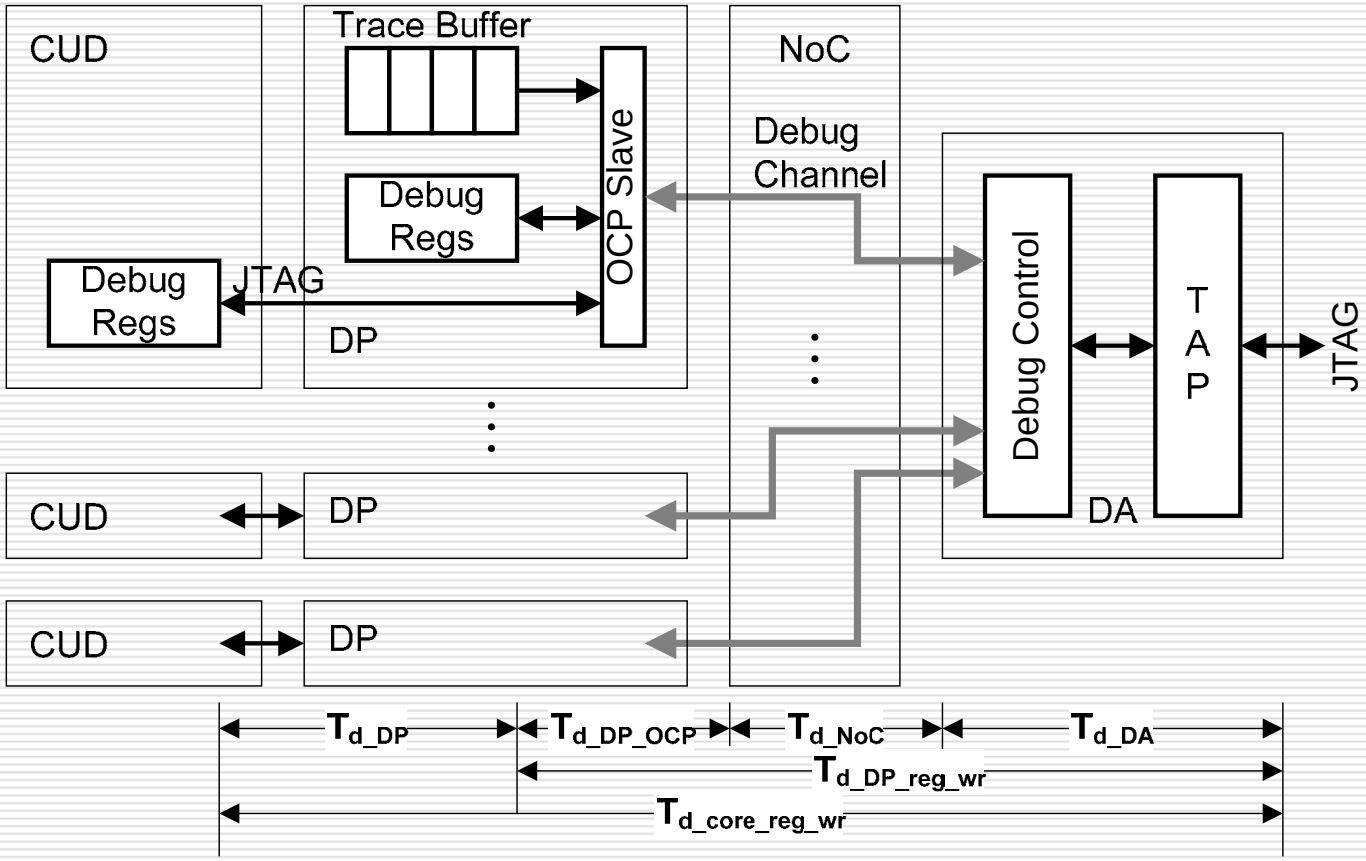
- 3-layer architecture:
  - Cross debugger
  - Core debugger and Transaction debugger
  - Multi-core debug driver (PC interfaces)

Off-Chip Debug Controller

- Translation layer between debuggers and on-chip debug architecture
  - Schedule debug commands/data transfer
- All debug resources in DPs and CUDs are mapped into addressable registers



Debug Access Delay



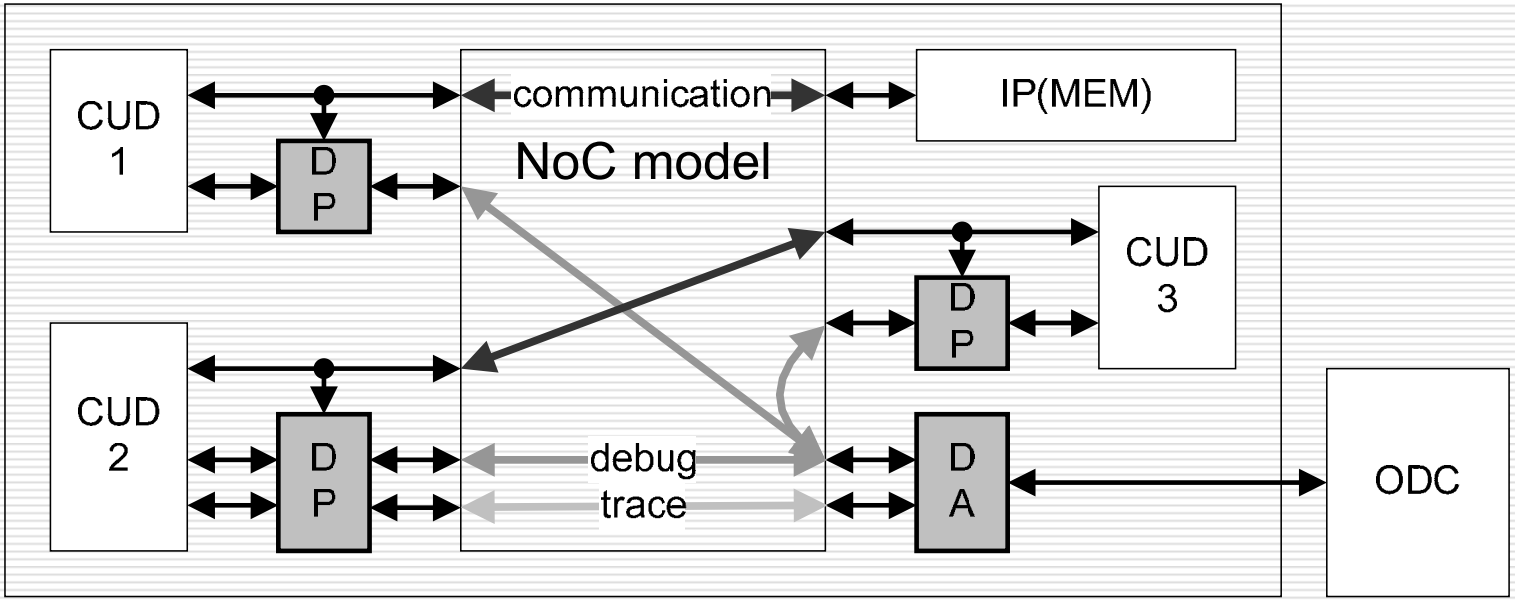
Experimental Results

A Multi-Core Debug Example

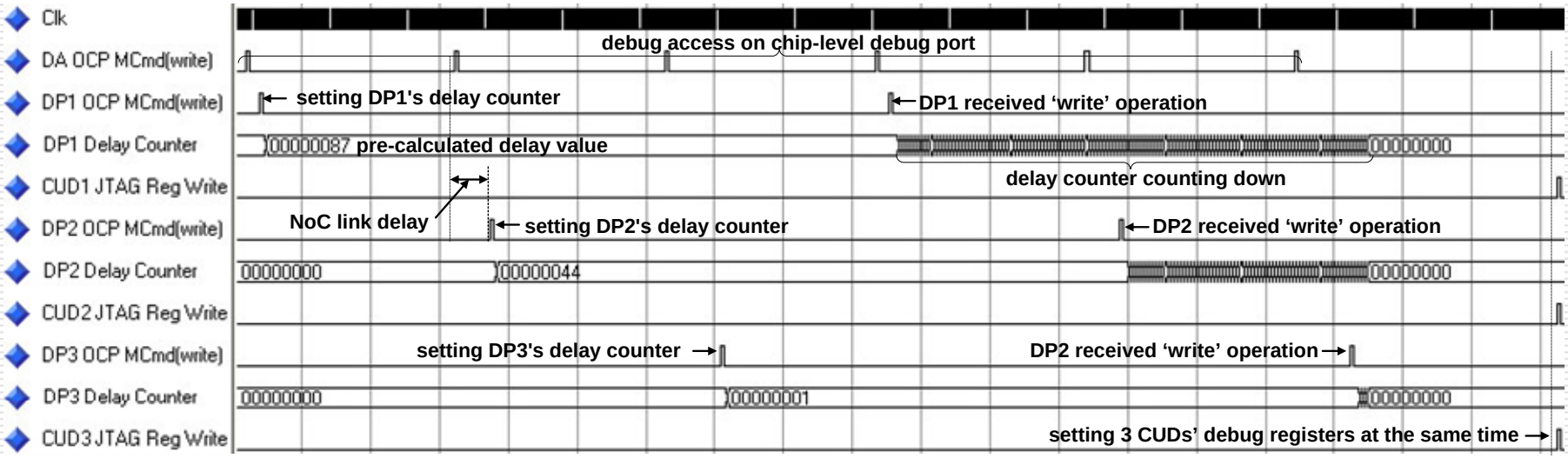
- DP detects a transaction trigger and stop three interacting cores

1. set DP trigger condition;
2. start DP and cores at the same time;
3. while (!triggered) check trigger status;
4. read out timestamp of the trigger event;
5. calculate relative trigger time for each core;
6. reset debug environment;
7. start DP and cores at the same time;
8. stop cores at trigger time;
9. hand over the control to the debuggers;

Simulation Environment

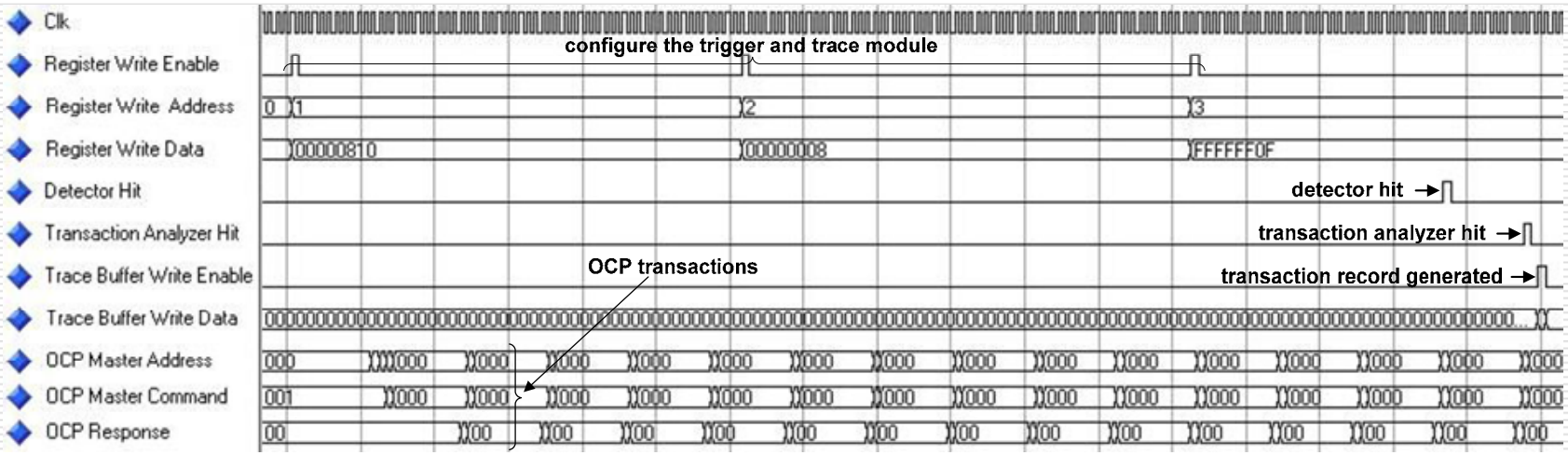


Simulation Results – Multi-core Concurrent Debug



- Pre-calculated delay can be inserted
- Multiple cores can be concurrently debugged

Simulation Results – Transaction Trace



- Configurable trigger and trace conditions
- Transactions are recorded when trigger event happens

DfD Area Cost – Debug Probe

Reference	Area (μm^2)	Percentage
Debug Probe Top	188059.0	100.0%
Transaction Trace Module		
Trigger and Trace Top	19692.1	10.5%
Input Switch	481.8	0.3%
Detector	543.3	0.3%
Transaction Analyzer	364.4	0.2%
MUX	14.3	0.0%
Record Generation	4629.7	2.5%
Shadow Buffer	9197.9	4.9%
Trigger and Trace Control	4456.3	2.4%
Transaction Trace Buffer (32x128)	120689.9	64.2%
Debug Access Module		
Delay Control	5384.8	2.9%
OCP Slave INF	2020.7	1.1%
Core Debug Module		
JTAG INF	3912.9	2.1%
Core Trace Module		
Core Trace INF	3099.6	1.6%
Core Trace Buffer	31225.6	16.6%
OCP Slave INF	2024.0	1.1%

Future Work

- Verify the proposed debug platform in-field
- Introduce DfD units inside NoC to locate the exact NoC error
- NoC without QoS connections for debug?
- ...

Conclusion

Q & A

Thank you!