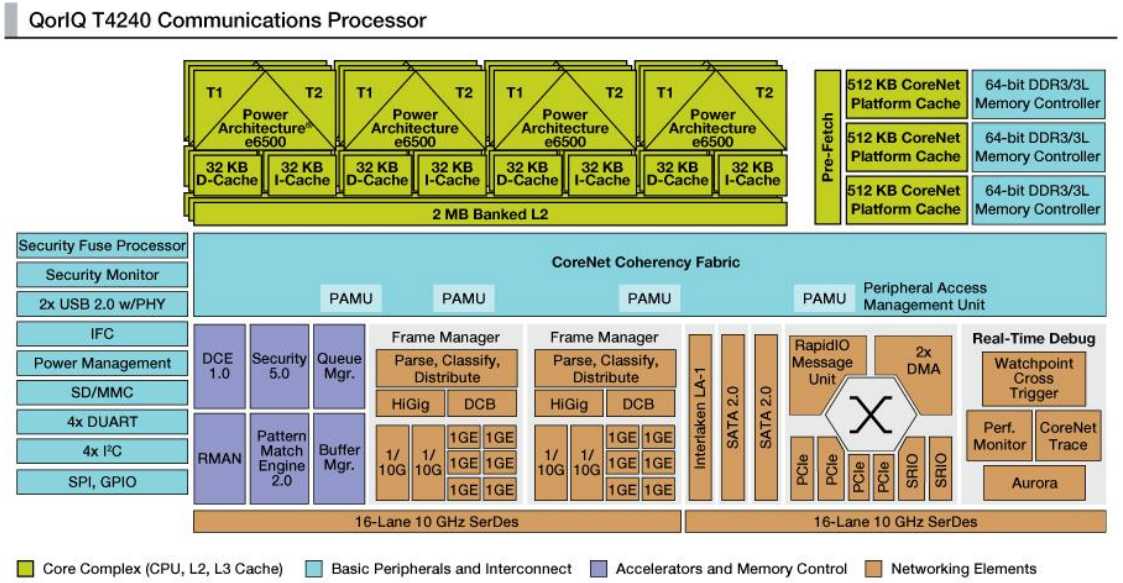
Observation of Multi-Core SoCs

Alexander Weiss

Garching, 15.11.2013

Accemic GmbH & Co. KG

Multi-Core SoC Observation

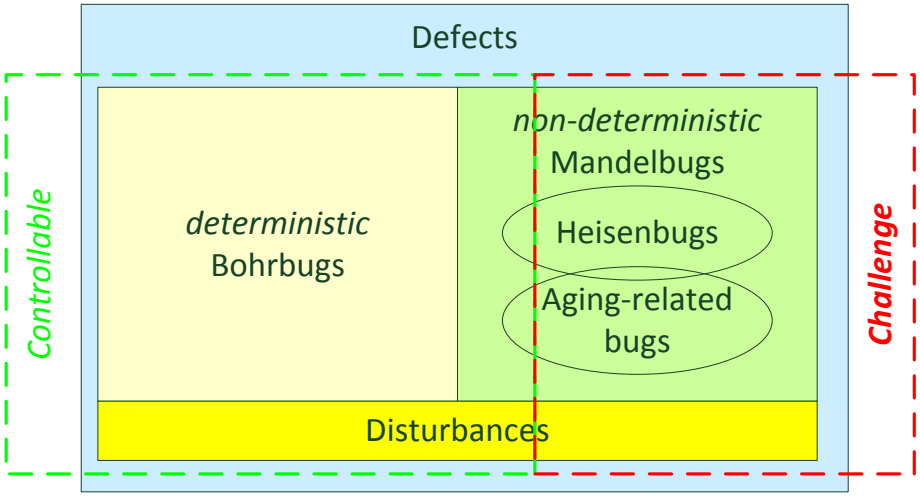


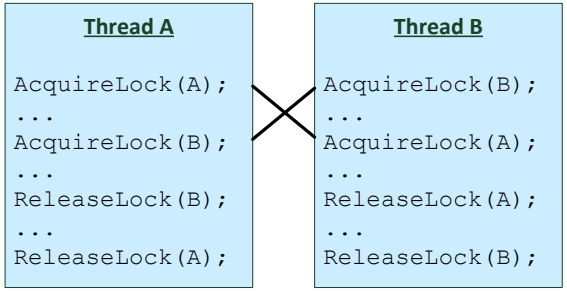
You can design it, but can you debug it?

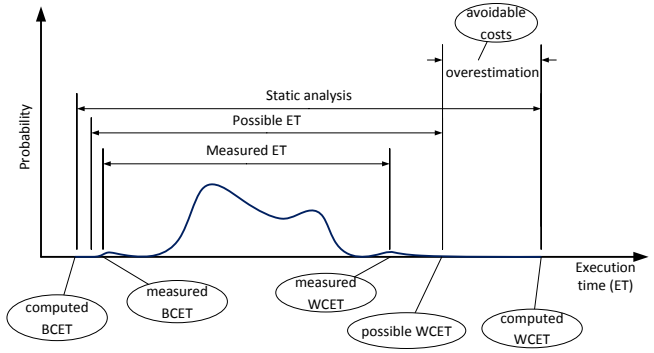
[Martin, Grant; Mayer, Albrecht; "*The challenges of heterogeneous multicore debug* ", Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010]

Observation for

- Debug
- Tests / coverage analysis
- WCET measurement
- Detection of race conditions
- Profiling / optimization

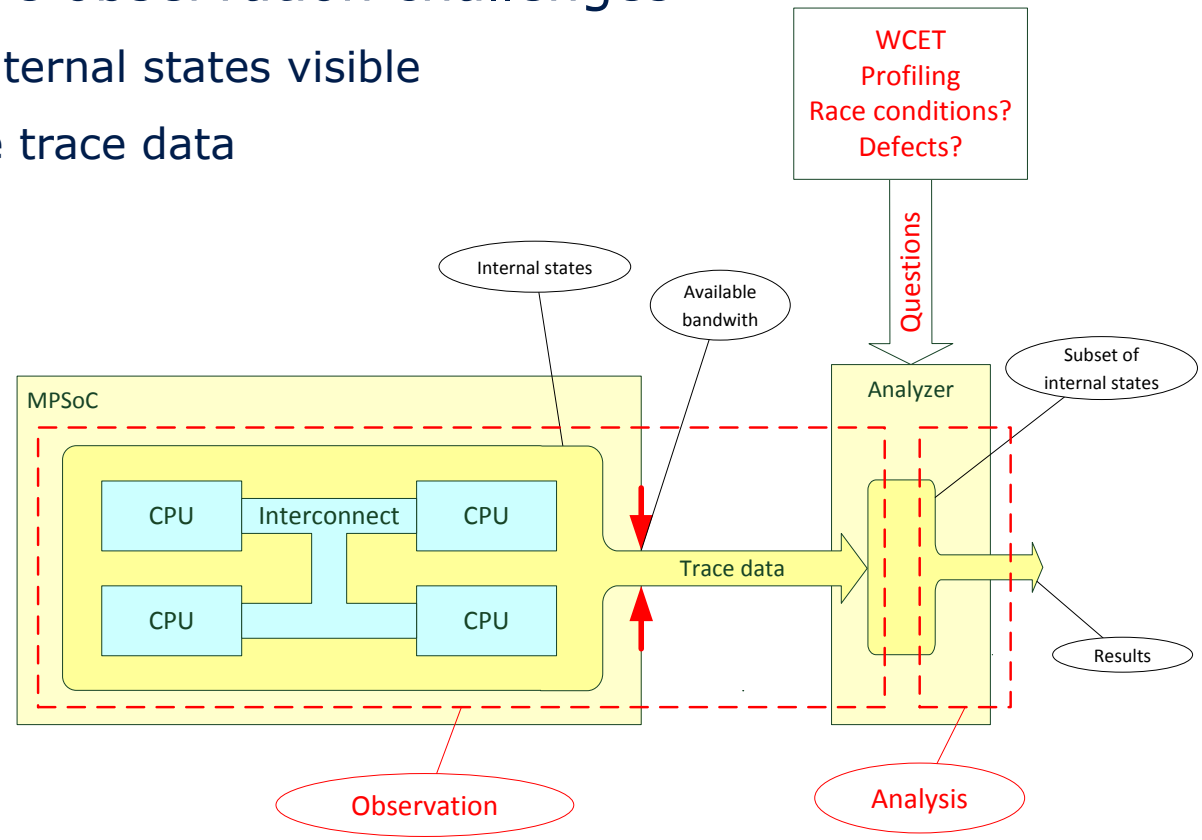






Multi-Core observation challenges

- Make internal states visible
- Analyze trace data



What we want to know

- CPUs
  - Executed instructions
  - Clock cycles / instruction
  - Data access (value, address, direction)
  - Cache
  - CPU register
  - Events
- Bus system / Bus master peripherals
  - Bus master data access (value, address, direction)
  - Events (timeouts, concurrent access, splitted transfers, errors)
- Vector clock
  - Temporal assignment of CPUs and bus master operations

Shared resource	Mechanism
System bus	Contention by multiple cores Contention by other device - IO, DMA, etc. Contention by coherency mechanism traffic
Bridges	Contention by other connected busses
Memory bus and controller	Concurrent access
Memory (DRAM)	Interleaved access by multiple cores causes address set-up delay Delay by memory refresh
Shared cache	Cache line eviction Contention due to concurrent access Coherency: Read delayed due to invalidated entry Coherency: Delay due to contention by coherency mechanism read requested by lower level cache Coherency: Contention by coherency mechanism on this level
Local cache	Coherency: Read delayed due to invalidated entry Coherency: Contention by coherency mechanism read
TLBs	Coherency overhead
Addressable devices	Overhead of locking mechanism accessing the memory I/O Device state altered by other thread/application Interrupt routing overhead Contention on the addressable device - e.g. DMA, Interrupt controller, etc. Synchronous access of other bus by the addressable device (e.g. DMA)
Pipeline stages	Contention by parallel hyperthreads
Logical units	Contention by parallel applications
	Other platform-specific effects, e.g. BIOS Handlers, Automated task migration, Cache stashing, etc.

UNDESIRED MECHANISMS AFFECTING THE TEMPORAL DETERMINISM from: Kotaba et al, "Multicore In Real-Time Systems – Temporal Isolation Challenges Due To Shared Resources", 2013

Other requirements

- Real-time capability
- Non-intrusiveness
- Concurrent observation of multiple CPUs / Busses / Peripherals
- State specific observation focus
- Observation of mass-produced SoCs
- Unlimited time observation
- Low latency
- Intuitive to use

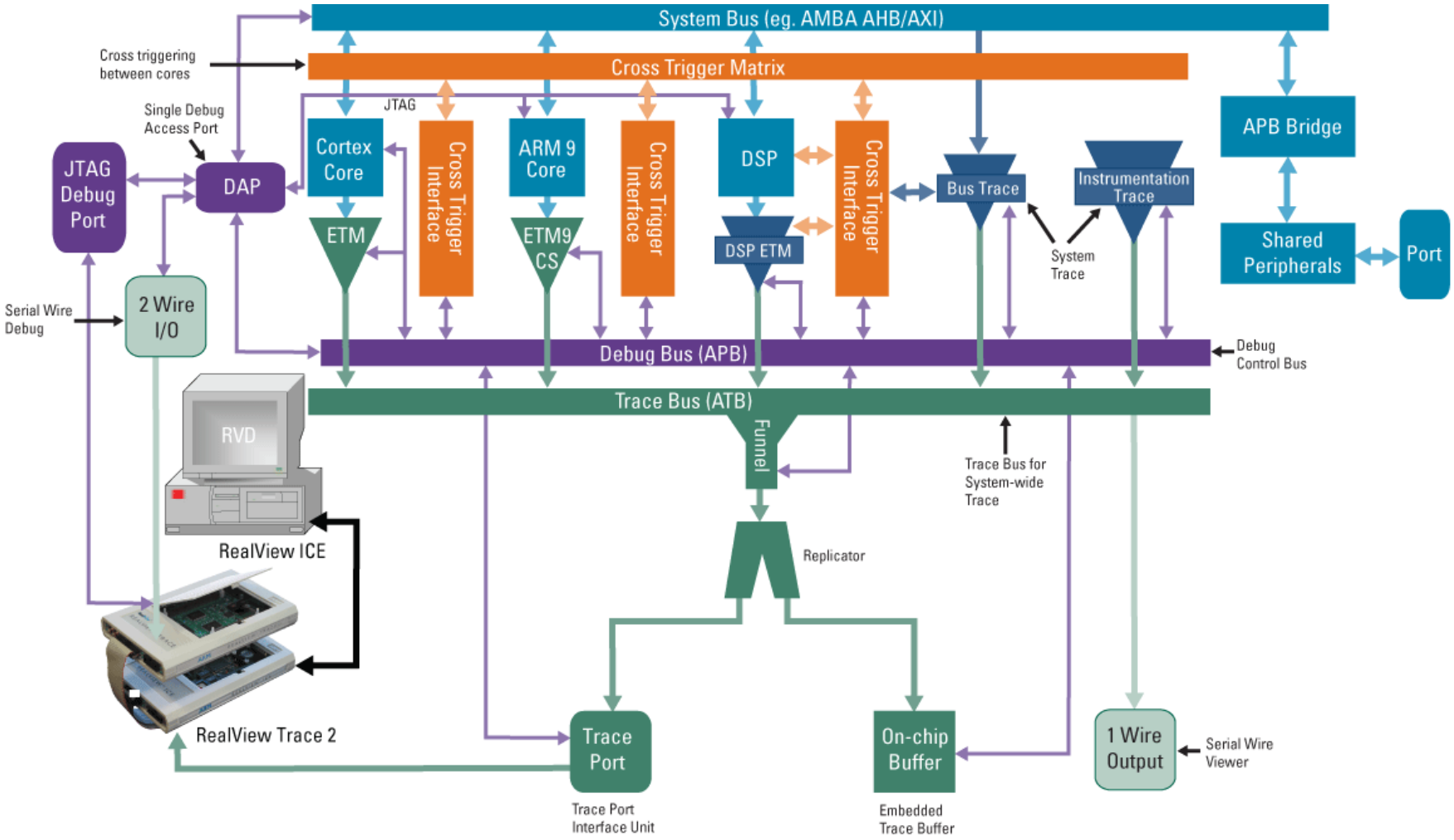
Software Instrumentation

+ Easy to implement / low cost for tests
– Additional resources / different behavior
– Changing observation focus requires code recompilation
– Temporal assignment of different CPUs processes is very limited
– *Questionable approach: removing instrumentation from production code*

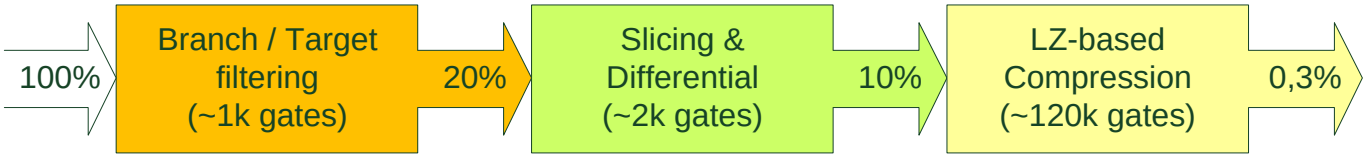
Original source code	Instrumented code (Statement / Condition Coverage)

`void foo()`
`{`
`  bool found=false;`
`  for (int i=0; (i<100) && (!found); ++i)`
`  {`
`  if (i==50) break;`
 	
`  if (i==20) found=true;`
 	
`  if (i==30) found=true;`
 	
`  }`
`  printf("foo\n");`
 	
`}`
	`char inst[15];`
	`void foo()`
	`{`
	`  bool found=false;`
	`  for (int i=0;((i<100)?inst[0]=1:inst[1]=1,0) &&`
	`  ((!found)?inst[2]=1:inst[3]=1,0); ++i)`
	`  {`
	`  if ((i==50?inst[4]=1:inst[5]=1,0))`
	`  { inst[6]=1; break;`
	`  }`
	`  if ((i==20?inst[7]=1:inst[8]=1,0))`
	`  { inst[9]=1; found=true;`
	`  }`
	`  if ((i==30?inst[10]=1:inst[11]=1,0))`
	`  { inst[12]=1; found=true;`
	`  }`
	`  inst[13]=1;`
	`  }`
	`  printf("foo\n");`
	`  inst[14]=1;`
	`}`

Multi-Core SoC Observation – State of the Art



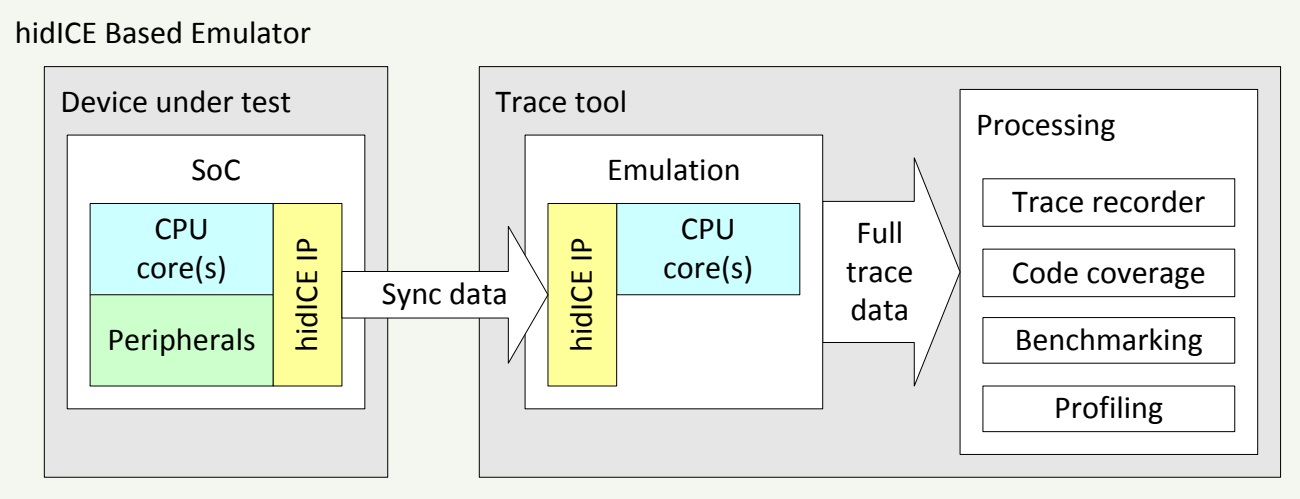
- Huang/Kao/Yang
  (National Sun Yat-Sen University Taiwan)
  - SYS-SIP SoC Development Infrastructure
  - three stages lossless instruction trace compression



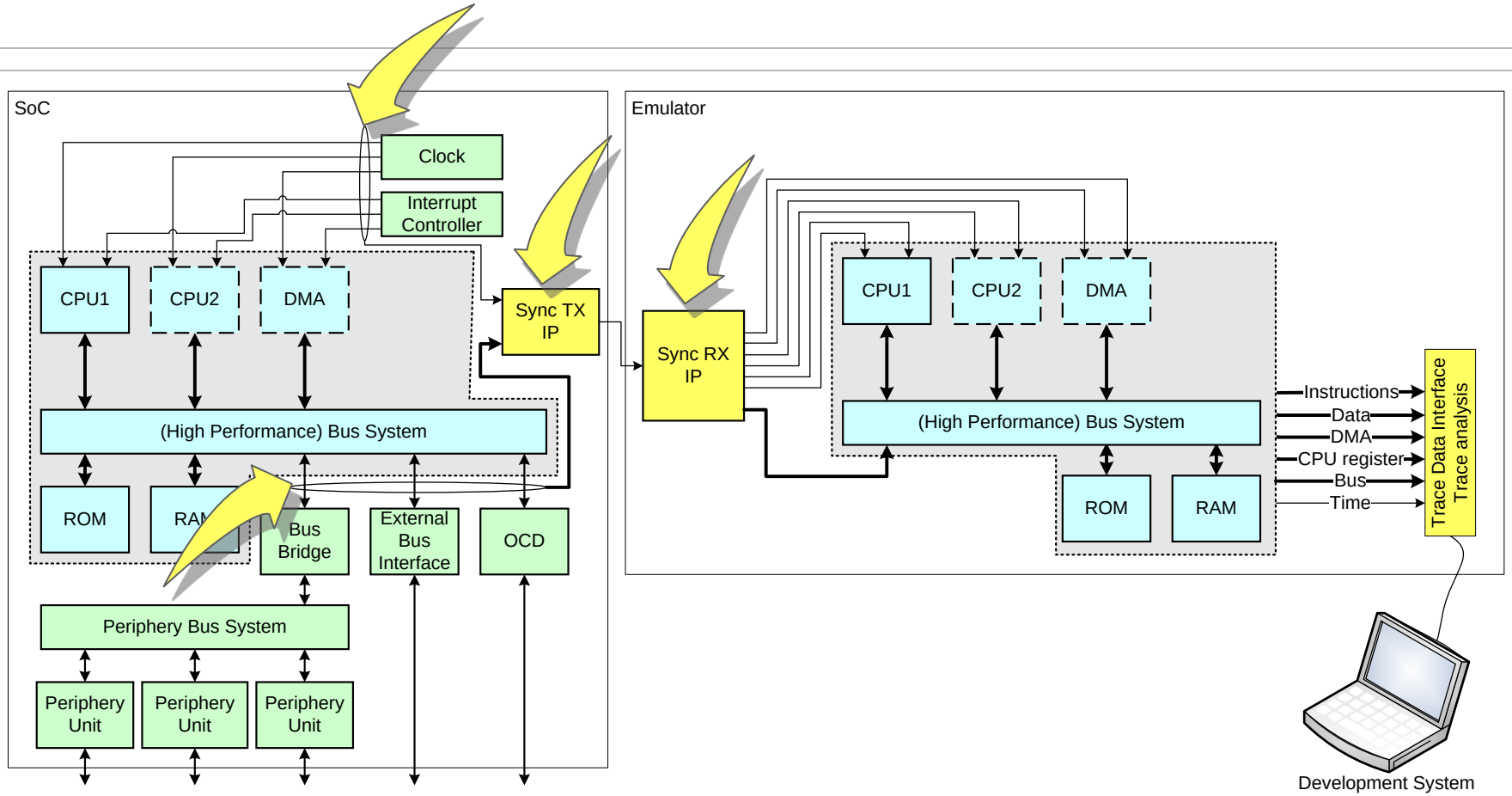
Fu-Ching Yang; ,"SYS-SIP SoC Development Infrastructure", Dissertation, National Sun Yat-Sen University, 2009
Fu-Ching Yang; Yi-Ting Lin; Chung-Fu Kao; Ing-Jer Huang; , "*An On-Chip AHB Bus Tracer With Real-Time* Compression *and Dynamic Multiresolution Supports for SoC*", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.19, no.4, pp.571-584, April 2011

hidICE (*hidden ICE*)

- Emulate the SoC core region and access trace data from there
- Easy access to **full** trace data



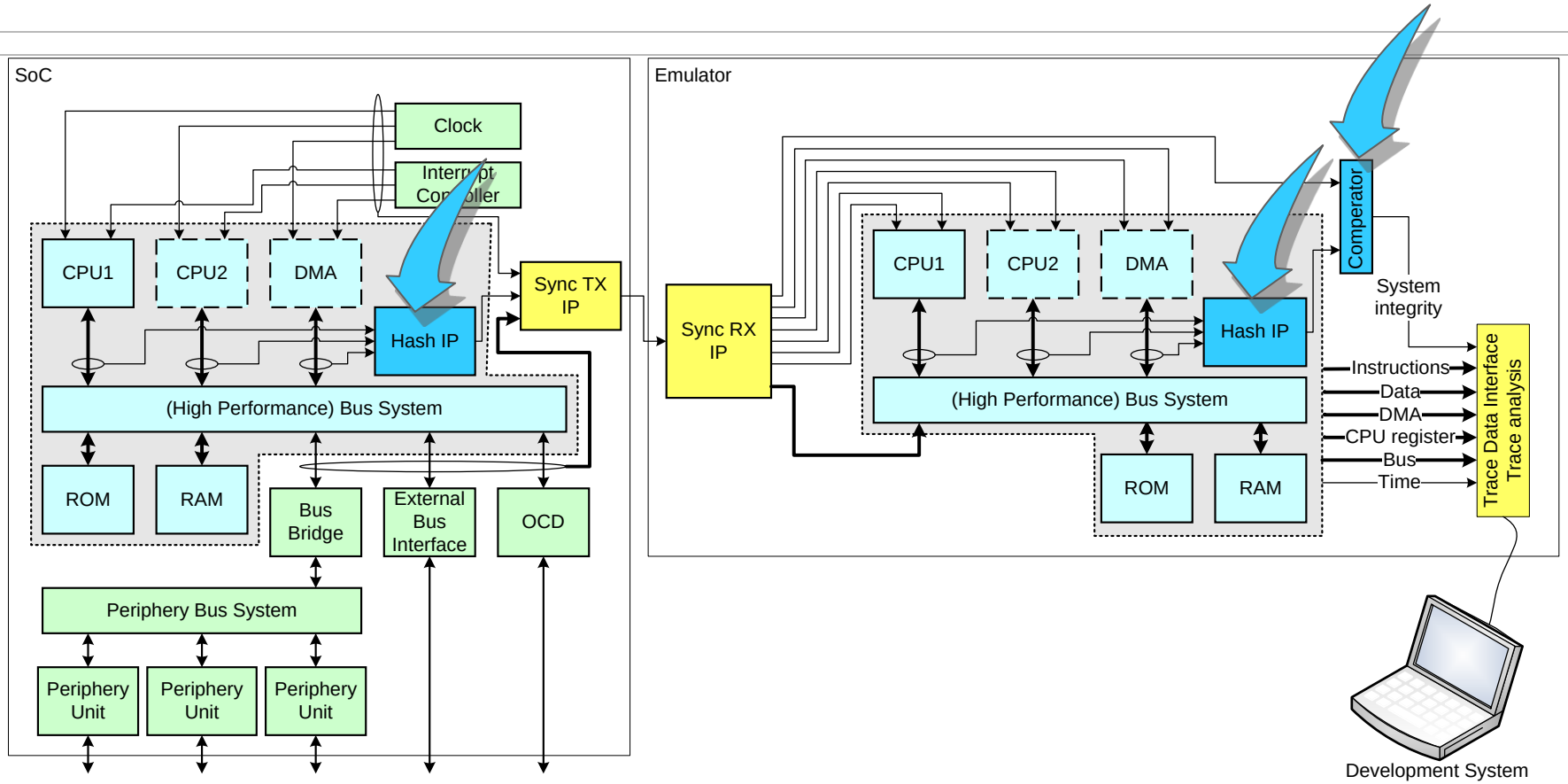
Multi-Core SoC Observation – hidICE



Synchronization

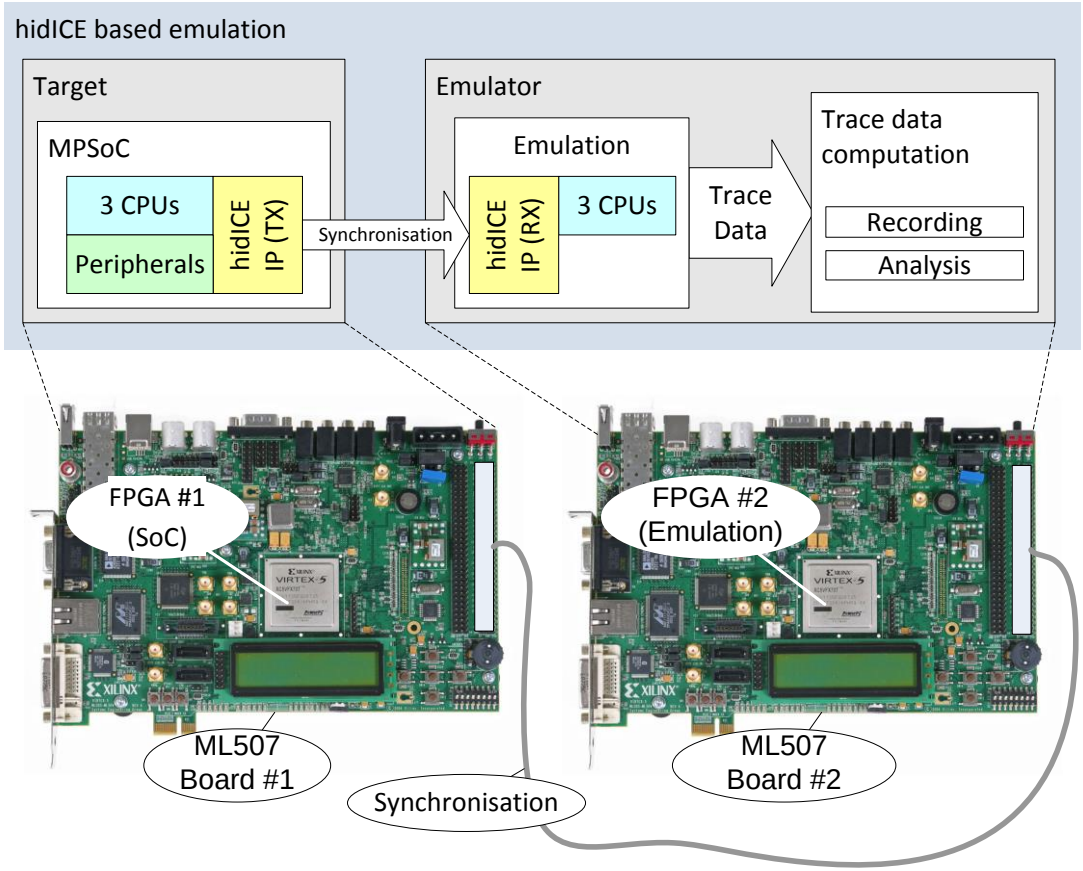
Signals to transmit | Serialization | Deserialization

Multi-Core SoC Observation – hidICE



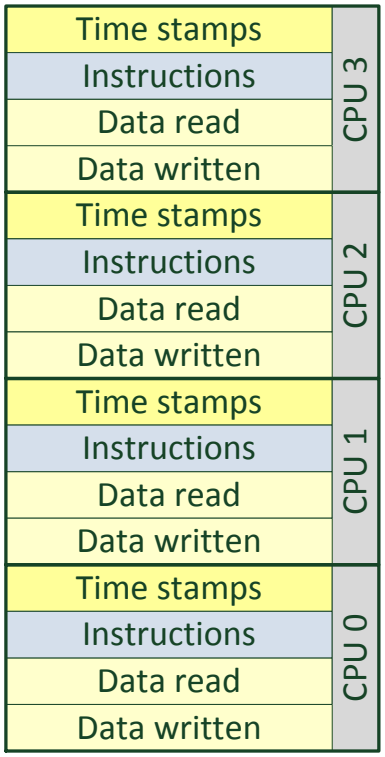
Synchronization			System Integrity Control	
Signals to transmit	Serialization	Deserialization	Hash calculation	Hash check

MPSoC implementation (3 x SPARC V8 / LEON3)



MPSoC, 4 CPUs, 1 GHz
Cycle accurate instruction + data trace
4 x 14 Bit x 1 GHz => approx. **28 Gbit/s**
(+ timestamps)
(+ bus master trace)
(+ peak bandwidth)

Embedded trace

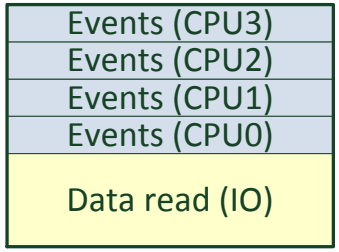


MPSoC (4 CPUs)

hidICE

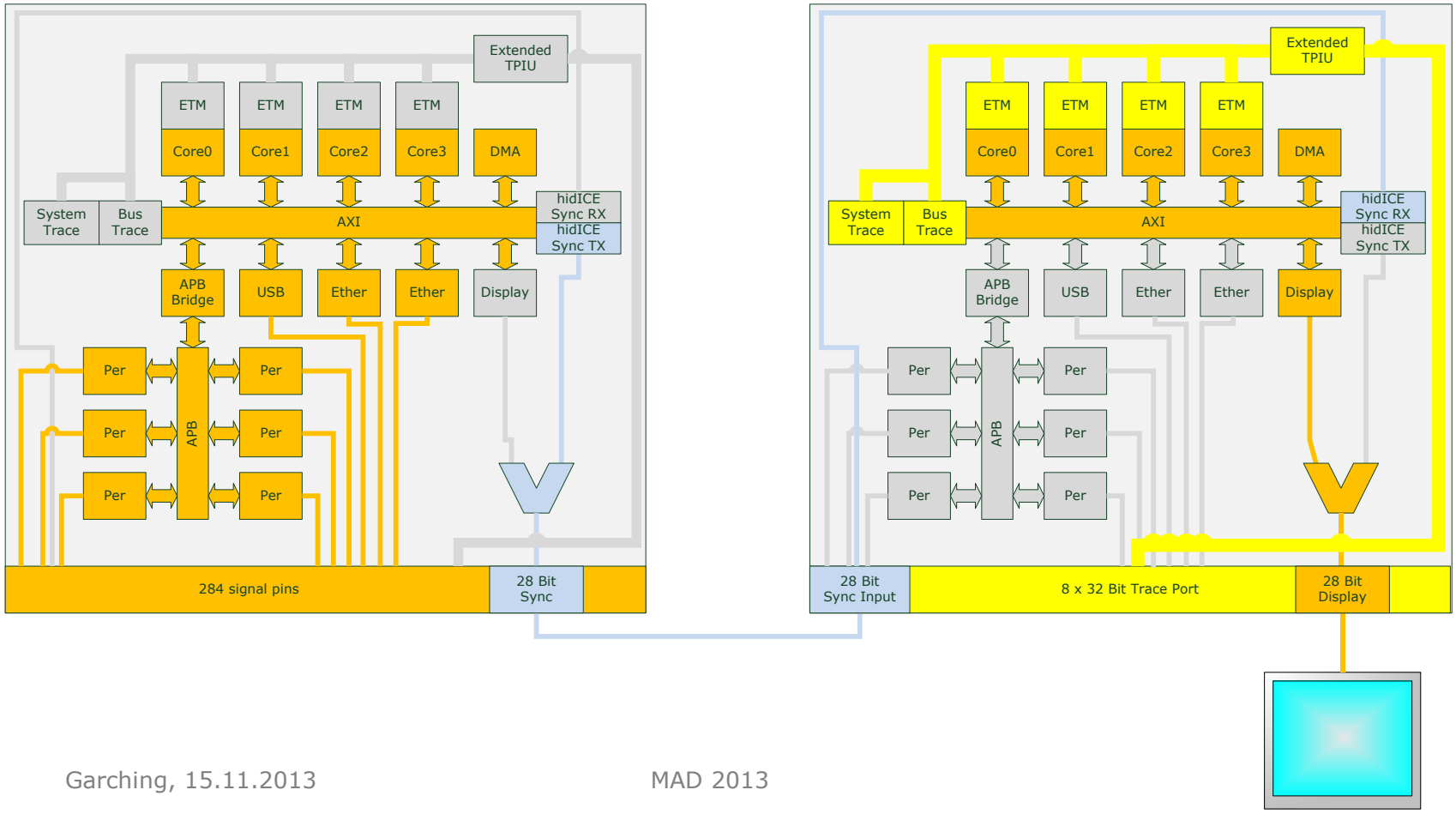
MPSoC, 4 CPUs, 1 GHz
1 x USB2.0, 2 x Gbit Ethernet,
some low speed peripherals

Synchronization bandwith: **< 4 GBit**
- timestamps included
- bus master trace included
- peak bandwith included



MPSoC (4 CPUs)

Draft: hidICE for quad core SoC



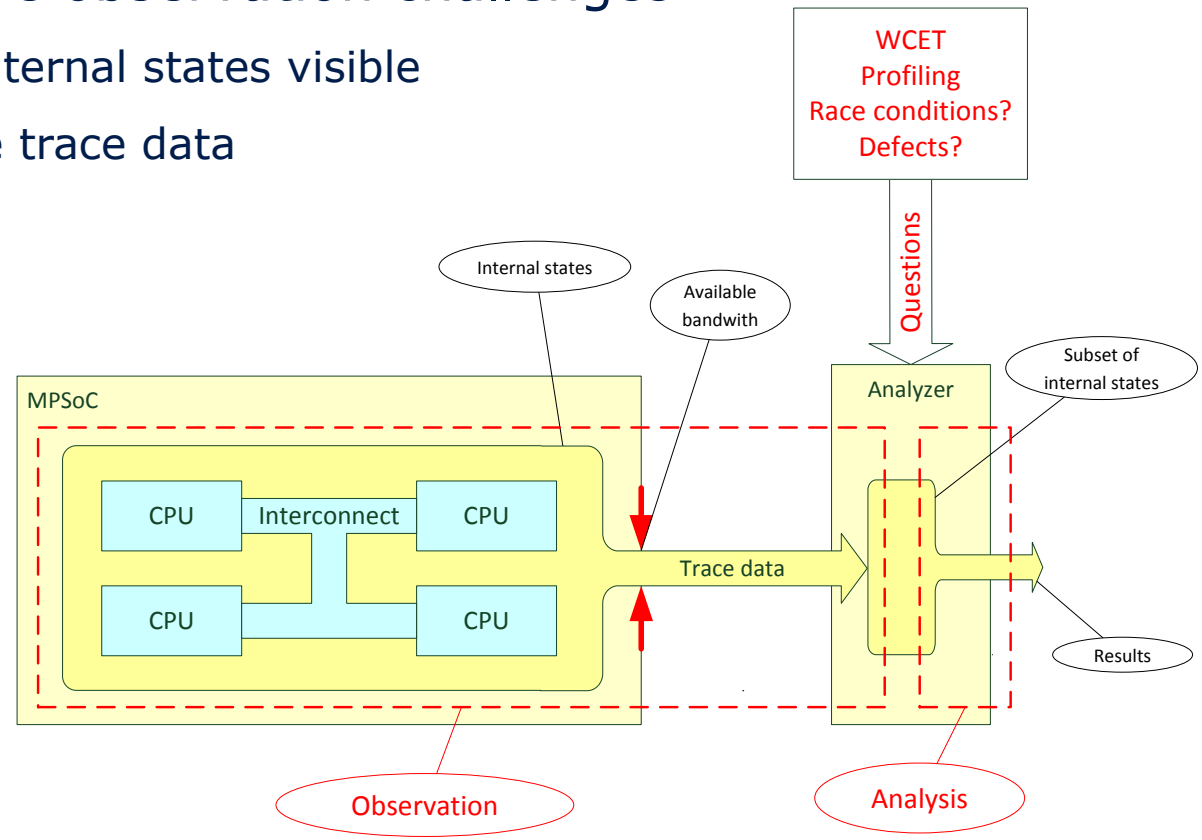
hidICE summary

+ Cycle accurate instruction and data trace from all CPUs
+ Cycle accurate data trace from all bus masters
+ Long-time observation
+ Low latency
+ Low intrusiveness (port replacement)

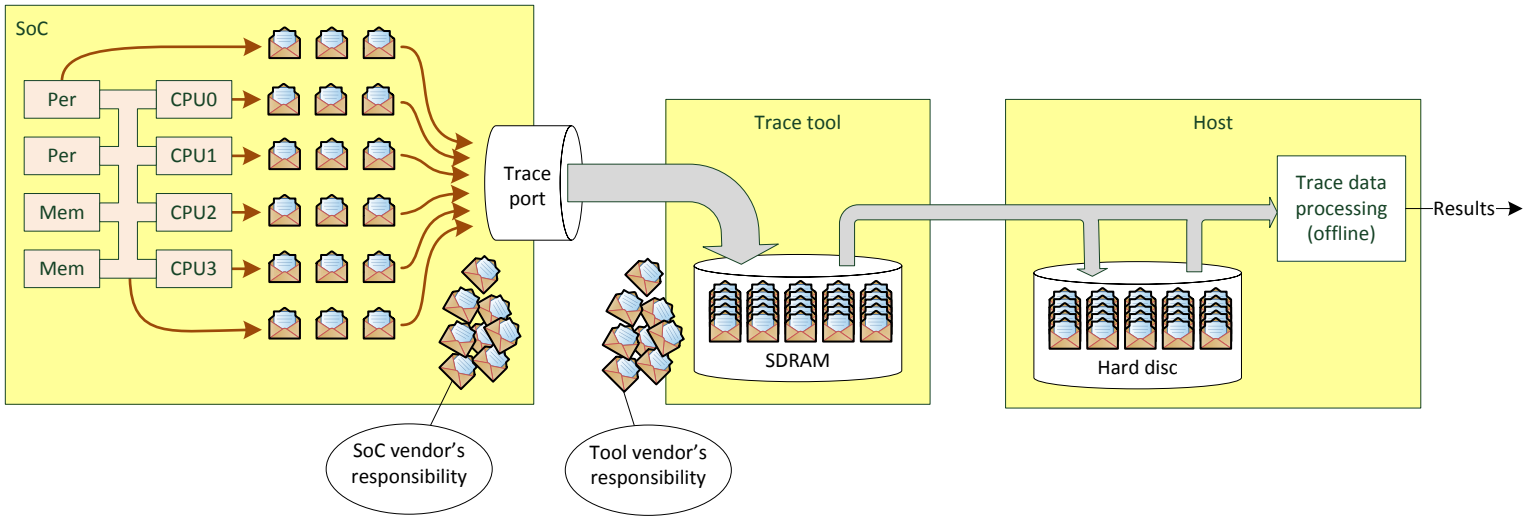
- Implementation effort (e.g. clock domain synchronization, correct implementation of the emulation)
- "All or none" – no partial trace
- Not applicable for SoCs with high I/O bandwidth

Multi-Core observation challenges

- Make internal states visible
- Analyze trace data



State of the Art:
Trace data recording and offline processing

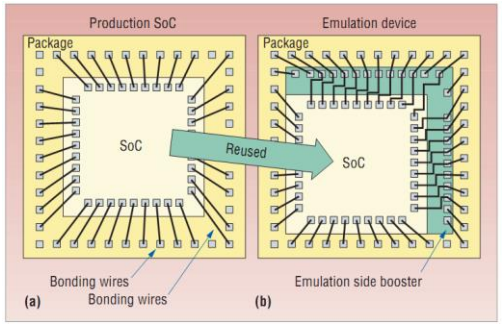


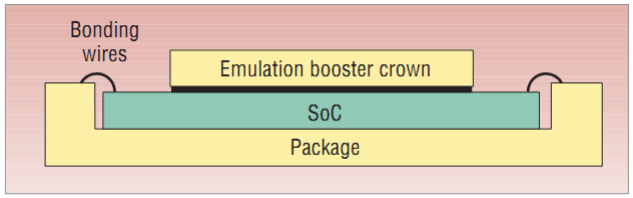
Limitations:

- Trace bandwidth >> Processing bandwith
- High latency in detection of SoC internal states state
- Limited in multiple observation focusses

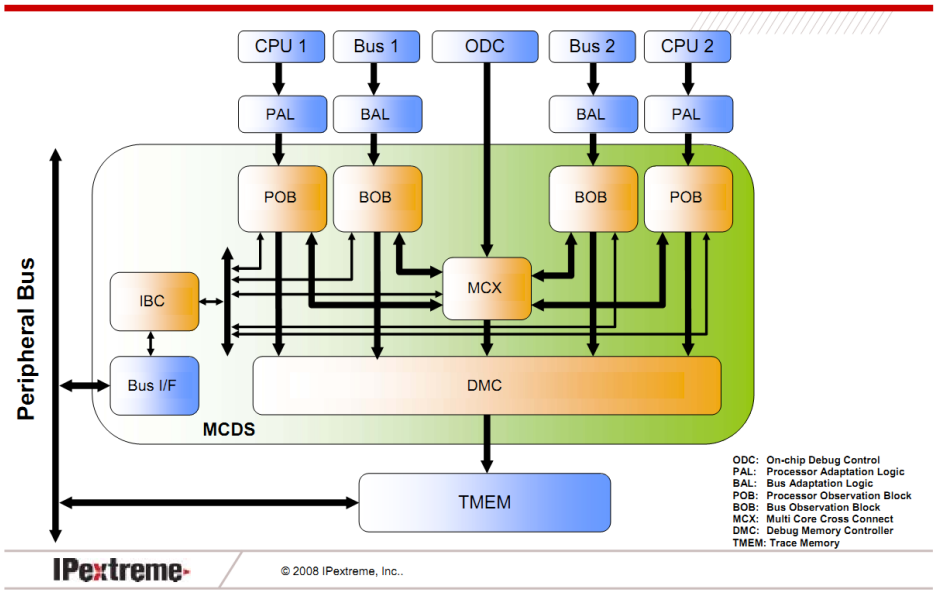
Infineon MCDS

- Combine SoC and trace tool



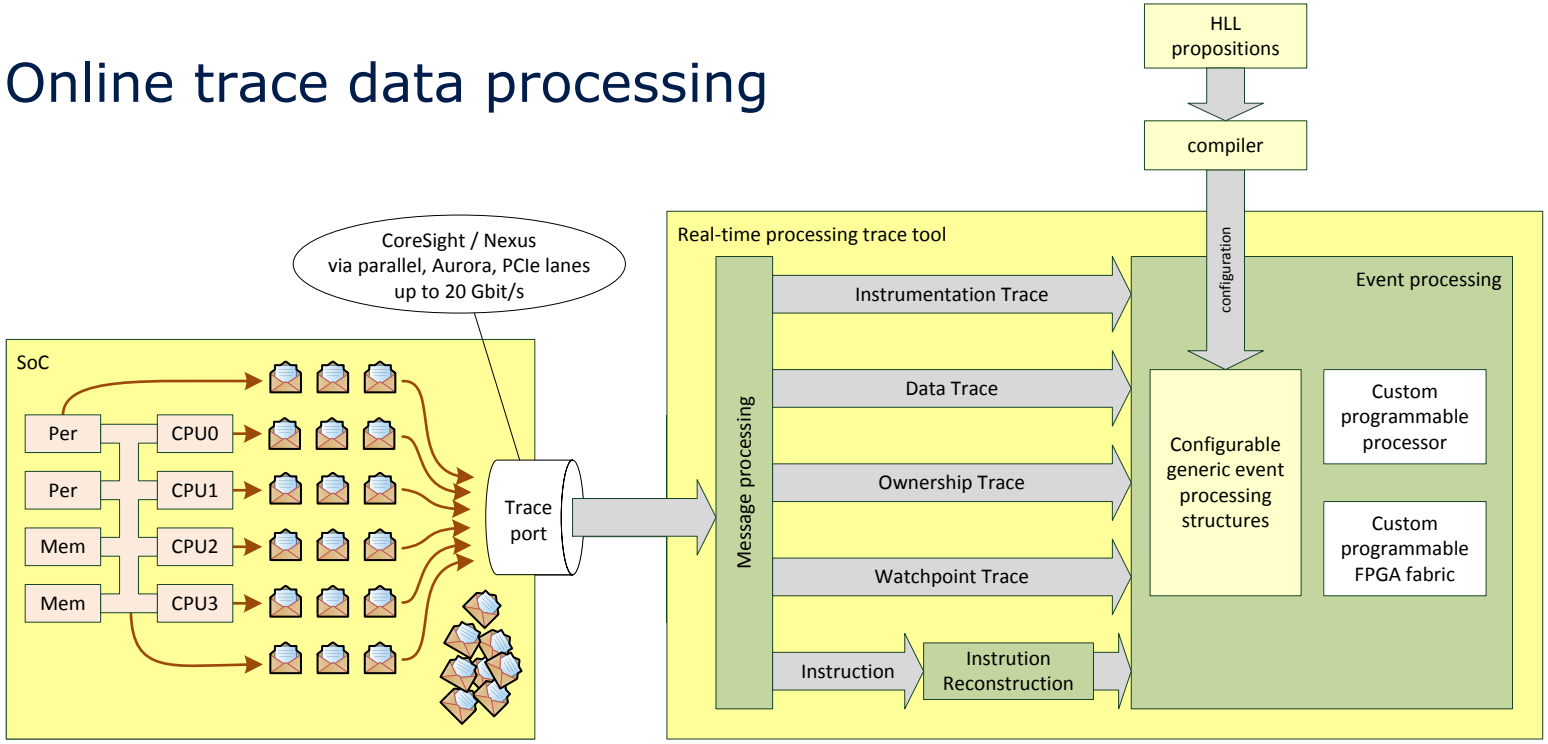


Mayer, A.; Siebert, H.; McDonald-Maier, K.D.; , "*Boosting Debugging Support for Complex Systems on Chip*" Computer , vol.40, no.4, pp.76-81, April 2007



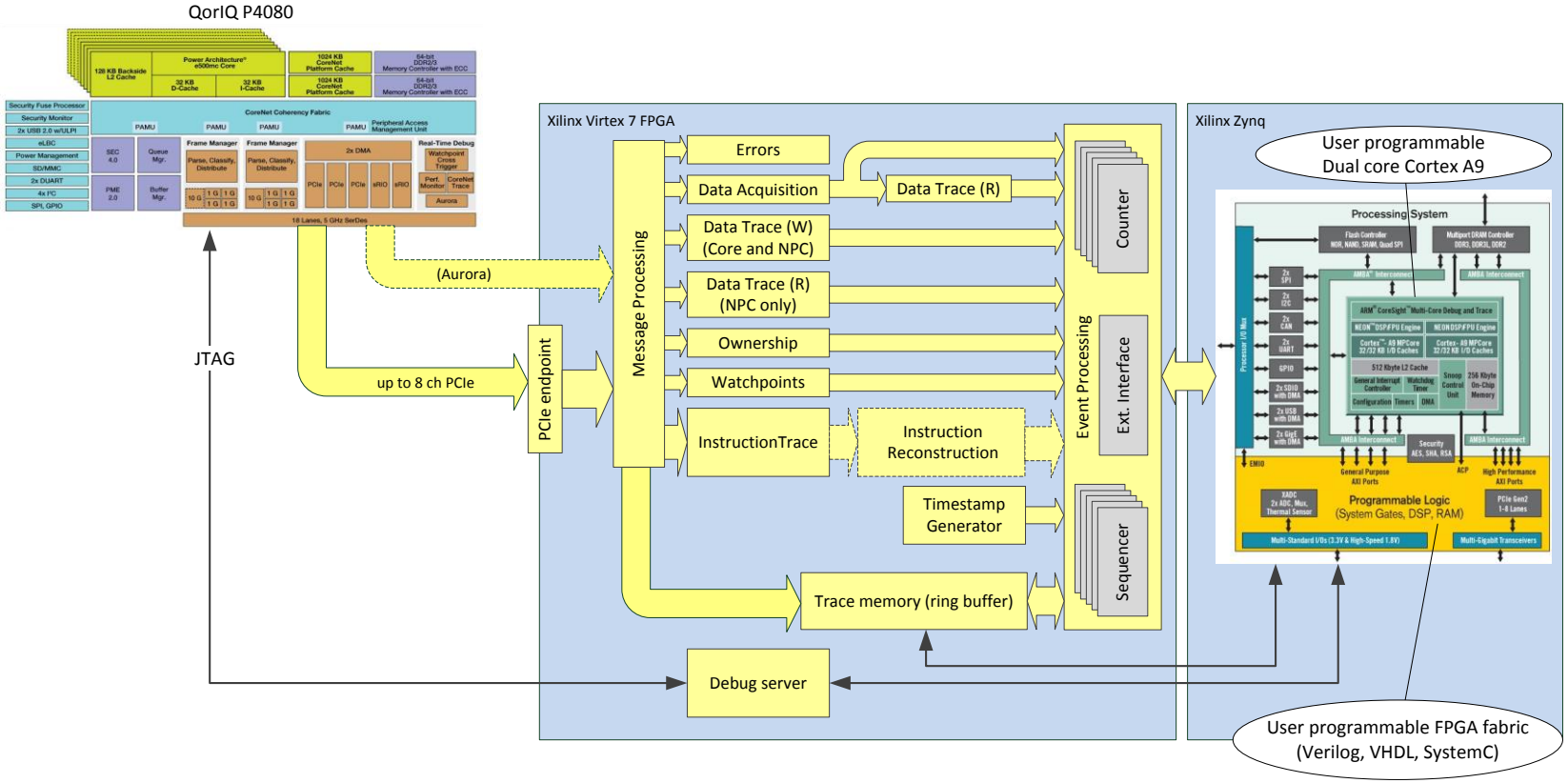
Source: www.ipextreme.com

Online trace data processing



- Processing bandwidth >= trace bandwith
- µs latency in detection of SoC internal states state
- Multiple observation focusses

Online trace processing implementation example
P4080 (8 cores e500mc @1.3 GHz)



Online trace processing

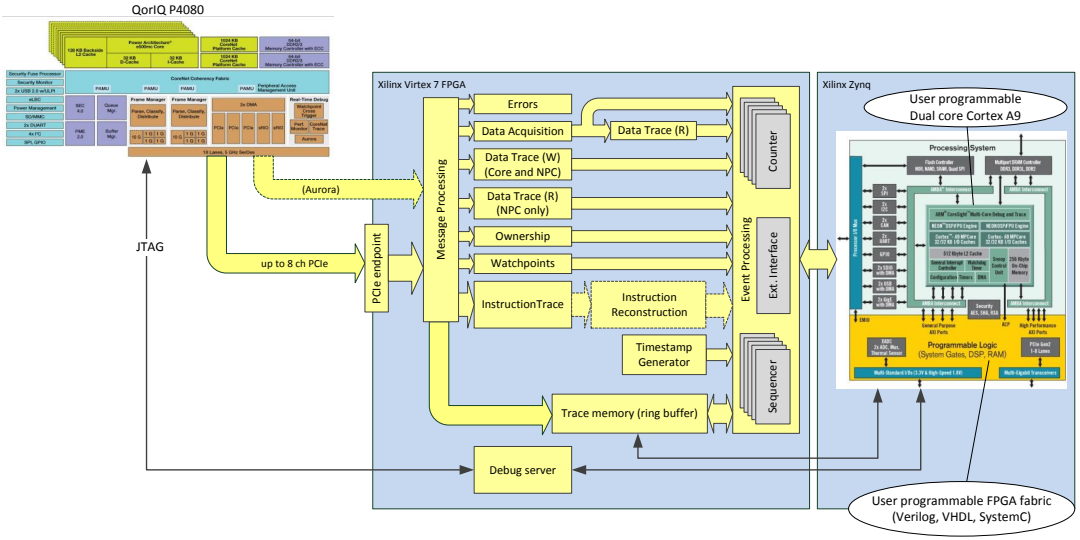
- High implementation effort (algorithms, FPGA resources) for
  - Transport protocol decoding (ARM CoreSight TPIU)
  - Detection of message boundaries (ARM CoreSight)
  - Online Reconstruction of direct branches (Nexus / ARM)
  - Runtime Verification infrastructure

Our wish to SOC vendors: More trace bandwidth!

Online trace processing advantages

+ Real-time capability / unlimited observation time
+ Low latency of state specific changes of observation focus
+ Parallel observation of multiple hotspots
+ Low latency of state specific triggering of the SoC (e.g. tests)
+ Immediate access to the runtime information by the developer
+ Instruction / basic blocks level continuous profiling
+ Basic block level continuous WCET measurement
+ Automatic detection of race conditions (e.g. “Happened before” algorithm)
+ Extension of SoC internal debug / profiling structures
+ Combination of Trace debugging and Run/Stop debugging
+ The “swiss army knife” in debugging: Online Runtime Verification

Questions?



Contact:
Alexander Weiss
Accemic GmbH & Co. KG
Franz-Huber-Str. 39
83088 Kiefersfelden

aweiss@accemic.com
+49 8033 6039790

In cooperation with

TECHNISCHE UNIVERSITÄT DARMSTADT

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÄT ZU LÜBECK