SoC Level Analytics, Trace & Debug for RISC-V Designs

Rupert Baines, CEO, UltraSoC
rupert.baines@ultrasoc.com
RISC-V Technical Symposium Moscow
20 May 2019
• Overview
• Processor Trace
• Algorithm
• Holistic System
• Demo System
• Summary
UltraSoC overview

• Embedded analytics
  • On-chip hardware monitors delivered as silicon intellectual property (SIP)
  • Supporting debug in-lab, & safety and security in-life

• Silicon-proven with multiple customers
• Founded 2009: VC-funded
• 35 employees; 40+ patents; HQ Cambridge UK
UltraSoC: partners and customers

Tier-1 Automotive

Custom uP Server

ARMv8 Server

SSD Controller

Western Digital
Microsemi
Imagination
HISILICON
kraftway
C-SKY
Alibaba
Movidius
Intel
Codasip
HiSilicon
SiFive
imperas
Sondrel
Cadence
NetSpeed Systems
Lauterbach Development Tools

Piranha Systems
Baysand
Mentor
LeCroy
MIPS
Advanced debug/monitoring for the whole SoC

Interconnect (AXI, ACE, ACE-lite, OCP, NoC)

 Portfolio of Analytic Modules
 Flexible & Scalable Message Fabric
 Family of Communicators

System Block
UltraSoC IP

30 May 2019
Software tools for data-driven insights

Eclipse based UltraDevelop 2 IDE

- Single step & breakpoint CPU code
- Real-time HW Data
- Instruction trace
- SW & HW in one tool
- Control
- Multiple CPUs
- Configuration

Third Party Tool Vendor Partnerships

- arm
- cadence
- CEVA
- Green Hills Software
- eclipse
- IMU Systems
- Imperas
- Mentor
- Lauterbach Development Tools
- Synopsys
- Teledyne Lecroy

30/05/2019
UltraSoC creates value both in-lab and in-life

Lab test » Field trial » In Life

UltraSoC detects threats and hazards an order of magnitude faster than any other solution – radically increasing security and safety

UltraSoC accelerates innovation and maximizes profitability
Faster TTM, higher quality, lower cost & higher margin

UltraSoC allows rapid optimization of application SW: improving performance, reducing TCO

2 Sept 2018

Commercial in Confidence
Actionable Insights across the whole SoC

UltraSoC delivers actionable insights

With system-wide understanding

From rich data across the whole SoC

UltraSoC enables full visibility of SoC

Debug, optimization, analytics
UltraSoC has the only commercial development environment for RISC-V

- Includes run control and trace
- Heterogeneous, massively multicore
- FPGA demonstrator, Eclipse IDE (gdb, gcc, openOCD, Imperas MPD)

- Silicon proven solution
- Partnerships with leading core vendors
- RISC-V Foundation member since 2016
  - Chair of trace group, member/contributor debug group
RISC-V Ecosystem
RISC-V Ecosystem
• In complex systems understanding program behavior is not easy
• Software often does not behave as expected
  • Interactions with other cores’ software, peripherals, realtime events, poor implementation or some combination of all of the above
  • Hiring better software engineers is not always an option
  • But usually because engineers write code with bugs in
• Using a debugger is not always possible
  • Realtime behavior is affected
• Providing visibility of program execution is important
  • This needs to be done without swamping the system with vast amounts of data
• One method of achieving this is via Processor Trace
Standardization

• **Debug**
  - Run-control, halt, single step etc
  - Ratified by Foundation
  - Supported by all core vendors
  - Support from standard tools (GDB etc)

• **Trace**
  - Working Group has “working consensus” for first release (instruction trace)
  - Supported by most core vendors (SweRV, SiFive, Andes etc)
  - Supported by open source (Boom and –soon – Pulp)
  - Commercial encoder IP (UltraSoC)
  - Open source encoder soon (ETH)
  - Support from tools (Lauterbach, IAR etc)
Branch vs Cycle-Accurate Trace

• Branch trace tracks execution from a known start address and sends messages about the deltas taken by the program
  • Jump, call, return and branch type instructions; interrupts and exceptions
  • Instructions between the deltas can be assumed to be executed sequentially

• Cycle-accurate trace tracks execution per-cycle
  • Required for real-time code optimization
Trace Encoder Ingress Port

<table>
<thead>
<tr>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivalid</td>
<td>Instruction has retired or trapped (exception)</td>
</tr>
<tr>
<td>iexception</td>
<td>Exception</td>
</tr>
<tr>
<td>interrupt</td>
<td>0 if the exception was synchronous; 1 if interrupt</td>
</tr>
<tr>
<td>cause [CAUSELEN-1:0]</td>
<td>Exception cause</td>
</tr>
<tr>
<td>tval[XLEN-1]</td>
<td>Exception data</td>
</tr>
<tr>
<td>priv[PRIVLEN-1:0]</td>
<td>Privilege mode during execution</td>
</tr>
<tr>
<td>iaddr [XLEN-1:0]</td>
<td>The address of the instruction</td>
</tr>
<tr>
<td>instr[ILEN-1:0]</td>
<td>The instruction</td>
</tr>
</tbody>
</table>

- For cores retiring N instructions per clock cycle the interface is replicated N times
The Encoder sends a packet containing one of the following:

1. Update – a branch map with or without a differential destination address/next address
2. Update – a full destination/next address and branch map
3. Update – a differential destination/next address with no branch or instruction related fields.
4. Synchronize - a context with or without a full current address

The above ensures an efficient packing to reduce data being routed on and subsequently transported off-chip.
Instruction Trace Algorithm

- Formats 0 and 1 send branch map and address
- Format 2 is address only
- Format 3 is a sync packet
  - Subformat 0 for when starting or resume from halt. No *ecause, *interrupt and *tval
  - Subformat 1 for exception. All fields present
  - Subformat 2 for context change. No *address, *ecause, *interrupt and *tval.
• Controlling when trace is generated is important
  • Helps reduces volume of trace data
• Filters are required
• Using filters the following trace examples are available:
  • Trace in an address range
  • Start trace at an address end trace at an address
  • Trace particular privilege level
  • Trace interrupt service routines
• Other examples
  • Trace for fixed period of time
  • Start trace when external (to the encoder) event detected
  • Stop trace when an external (to the encoder) event detected
Table shows encoding efficiency of the algorithm

- Does not include any overhead for encapsulating into messages or routing
- Different program types will have different overheads

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Instructions</th>
<th>Packets</th>
<th>Payload Bytes</th>
<th>Bits per instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhrystone</td>
<td>215015</td>
<td>1308</td>
<td>5628</td>
<td>0.209</td>
</tr>
<tr>
<td>hello_world</td>
<td>325246</td>
<td>2789</td>
<td>10642</td>
<td>0.262</td>
</tr>
<tr>
<td>median</td>
<td>15015</td>
<td>207</td>
<td>810</td>
<td>0.432</td>
</tr>
<tr>
<td>mm</td>
<td>297038</td>
<td>644</td>
<td>2011</td>
<td>0.054</td>
</tr>
<tr>
<td>mt-matmul</td>
<td>41454</td>
<td>344</td>
<td>953</td>
<td>0.184</td>
</tr>
<tr>
<td>mt-vvadd</td>
<td>61072</td>
<td>759</td>
<td>2049</td>
<td>0.268</td>
</tr>
<tr>
<td>multiply</td>
<td>55016</td>
<td>546</td>
<td>1837</td>
<td>0.267</td>
</tr>
<tr>
<td>pmp</td>
<td>425</td>
<td>7</td>
<td>39</td>
<td>0.734</td>
</tr>
<tr>
<td>qsort</td>
<td>235015</td>
<td>2052</td>
<td>8951</td>
<td>0.305</td>
</tr>
<tr>
<td>rsort</td>
<td>375016</td>
<td>683</td>
<td>2077</td>
<td>0.044</td>
</tr>
<tr>
<td>spmv</td>
<td>70015</td>
<td>254</td>
<td>1154</td>
<td>0.132</td>
</tr>
<tr>
<td>towers</td>
<td>15016</td>
<td>72</td>
<td>237</td>
<td>0.126</td>
</tr>
<tr>
<td>vvadd</td>
<td>10016</td>
<td>111</td>
<td>316</td>
<td>0.252</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
<td>0.252</td>
</tr>
</tbody>
</table>
Demo System Architecture

• Zynq ZC706 FPGA platform
  • ARM
    • Plus RV32 RISCV
    • Plus custom logic
  • Demo shows:
    • Bus state
    • Traffic
    • Performance histogram
    • Memory
    • Processor control
    • Bus deadlock detection
    • RISC-V Processor trace
Software tools for data-driven insights

Eclipse based UltraDevelop 2 IDE

- Single step & breakpoint CPU code
- Control
- Multiple CPUs
- Configuration
- SW & HW in one tool
- Real-time HW Data
- Instruction trace

Third Party Tool Vendor Partnerships

- arm
- Cadence
- CEVA
- Green Hills Software
- Eclipse
- Imperas
- Mentor
- Synopsys
- Teledyne Lecroy

30/05/2019
<table>
<thead>
<tr>
<th>Feature</th>
<th>Standard RISC-V</th>
<th>UltraSoC Trace Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Filters</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Counters</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Timestamps</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Comparators</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GPIO</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Data trace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interval timer</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Multiple retirement</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit return mode</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Whole system solution</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Branch prediction</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Cycle-accurate tracing</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
• RISC-V eco-system maturing
  • Development tools & infrastructure available
  • Standardisation moving fast
  • Both commercial and open-source

• Determining Program behavior is not always possible using source level debugging

• Understanding program behavior in-field and realtime is needed

• An efficient Trace scheme provides this

• Couple this with a holistic non-intrusive monitoring infrastructure provides the means of understanding complete SoC behavior