Evolution of self-hosted Tracing with Arm® CoreSight

Suzuki K Poulse <suzuki.poulse@arm.com>
Agenda

- CPU tracing
- Introduction to CoreSight
- CoreSight Tracing subsystem
- Linux driver support
- CoreSight with perf
- The evolution
CPU Tracing

- Record information about the CPU’s execution
- Normally with very less overhead
- Use cases
  - Non-invasive Software Debugging
  - Performance optimization - Dynamic feedback
What is Arm® CoreSight?

- Architecture for Debug and Trace for Arm
  - Programming interface
  - Physical interface
  - System architecture

- Supports external and self-hosted modes

- Often confused between:
  - CoreSight Architecture
  - CoreSight IP suite
CoreSight tracing subsystem
CoreSight tracing subsystem

- Trace Sources
  - Embedded Trace Macrocells (ETM)
  - SystemTrace Macrocells (STM)

- Trace Links
  - Funnels
  - Replicators
  - Hardware buffering (TMC-ETF)

- Trace Sinks
  - Trace Port Interface Unit (TPIU)
  - Embedded Trace Buffer (ETB, TMC-ETB)
  - Embedded Trace Router (TMC-ETR)
Embedded Trace Macrocells - ETM

- CPU trace unit for monitoring execution
- Generates compressed trace data
  - Meta data (Configuration, Context)
  - Program execution (e.g., branches, exceptions, speculation)
  - Cycle count and Time Stamps

```
ID:12; I_ADDR_CTX_L_64IS0 : Address & Context, Long, 64 bit, IS0.; Addr=0x0000FFFFB161326C; Ctxt: AArch64,EL0, NS;
ID:12; I_ATOM_F1 : Atom format 1.; E
ID:12; I_CCNT_F1 : Cycle Count format 1.; Count=0x0
ID:12; I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEEEE
```

- Decoded with program text
Embedded Trace Macrocells - Continued

- Trace filtering based on:
  - Address range (inclusive/exclusive)
  - Context based filtering (process or VM)
  - 16bit decrementing counters (with chaining)
  - External inputs (CoreSight Cross Trigger, PMU events)
  - Sequencer state machine
Sinks - Embedded Trace Router TMC-ETR (v1)

- Route trace data to System RAM
- Buffer modes - Circular Buffer mode, Software FIFO
  - Read, Write pointers
  - No overflow interrupt :-(
- Memory addressing modes - IOMMU is recommended
  - Contiguous memory mode
  - Scatter-Gather(SG) mode
CoreSight support in Linux

- Introduced in Linux v3.19 (2014)
- Maintained by Mathieu Poirier (Linaro)
- Device tree bindings describe the connections
- Clocks and Power management
- Controlled via sysfs
  - /sys/bus/coresight/devices/<device>/
  - Trace data exposed via misc device
CoreSight support in Linux - Perf

- Introduced in v4.6 (2016) - named cs_etm
- Open CoreSight Trace Decoder (OpenCSD) library
- perf tool support - v4.16
- Supports address filtering
- ETR sink support - v4.20
- Uses AUX TRACE interface
- Restricted to single threaded processes
$ perf record -e cs_etm/@020070000.etr,cycacc/u --per-thread ./test
$ perf report -D -i perf.data --stdio

... CoreSight ETM Trace data: size 58832 bytes
  Idx:0; ID:12;  I_ASYNC : Alignment Synchronisation.
  Idx:12; ID:12;  I_TRACE_INFO : Trace Info.; INFO=0x1; CC_THRESHOLD=0x100
  Idx:48; ID:12;  I_ASYNC : Alignment Synchronisation.
  Idx:60; ID:12;  I_TRACE_INFO : Trace Info.; INFO=0x1; CC_THRESHOLD=0x100
  Idx:67; ID:12;  I_TRACE_ON : Trace On.
  Idx:68; ID:12;  I_ADDR_CTX_L_64IS0 : Address & Context, Long, 64 bit, IS0.; Addr=0x0000FFFFB160FF80; Ctxt: AArch64,EL0, NS;
  Idx:78; ID:12;  I_ATOM_F1 : Atom format 1.; E
  Idx:80; ID:12;  I_CCNT_F1 : Cycle Count format 1.; Count=0x0
  Idx:81; ID:12;  I_EXCEPT : Exception.; Data Fault; Ret Addr Follows;
  Idx:83; ID:12;  I_ADDR_L_64IS0 : Address, Long, 64 bit, IS0.; Addr=0x0000FFFFB161326C;
  Idx:92; ID:12;  I_CCNT_F2 : Cycle Count format 2.; Count=0x10c
  Idx:94; ID:12;  I_ADDR_CTX_L_64IS0 : Address & Context, Long, 64 bit, IS0.; Addr=0xFFFF000008082400; Ctxt: AArch64,EL1, NS;
  Idx:105; ID:12;  I_TRACE_ON : Trace On.
  Idx:106; ID:12;  I_ADDR_CTX_L_64IS0 : Address & Context, Long, 64 bit, IS0.; Addr=0x0000FFFFB161326C; Ctxt: AArch64,EL0, NS;
  Idx:117; ID:12;  I_ATOM_F1 : Atom format 1.; E
  Idx:118; ID:12;  I_CCNT_F1 : Cycle Count format 1.; Count=0x0
  Idx:119; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:120; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:121; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:122; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:123; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:124; ID:12;  I_ATOM_F6 : Atom format 6.; EEEEEEEEEEEEEEEEEEEEE
  Idx:125; ID:12;  I_CCNT_F3 : Cycle Count format 3.; Count=0x101
...
CoreSight perf - Instruction trace

```
$ perf inject -i perf.data -o perf-insn.data --inject=i1000il

$ perf report -D -i perf-insn.data --stdio

0x3ec0 [0x168]: PERF_RECORD_SAMPLE(IP, 0x2): 5373/5373: 0xffffb1613184 period: 169 addr: 0
... branch stack: nr:13
..... 0: 0000ffffb1613474 -> 0000ffffb1613140 0 cycles P 0
..... 1: 0000ffffb16135f8 -> 0000ffffb1613460 0 cycles P 0
..... 2: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
..... 3: 0000ffffb161364c -> 0000ffffb16136c4 0 cycles P 0
..... 4: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
..... 5: 0000ffffb161364c -> 0000ffffb16136c4 0 cycles P 0
..... 6: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
..... 7: 0000ffffb161364c -> 0000ffffb16136c4 0 cycles P 0
..... 8: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
..... 9: 0000ffffb161364c -> 0000ffffb16136c4 0 cycles P 0
..... 10: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
..... 11: 0000ffffb161364c -> 0000ffffb16136c4 0 cycles P 0
..... 12: 0000ffffb16136d0 -> 0000ffffb16135f0 0 cycles P 0
... thread: test:5373
...... dso: /lib/aarch64-linux-gnu/ld-2.19.so
```
CoreSight perf - What next?

- Work to do:
  - Tracing multi-threaded process
  - Support for using the resource selectors
  - ACPI support

- System Challenges
  - Double buffering
  - Virtualization
  - Independent trace sessions
  - Power management

© 2018 Arm Limited
CoreSight Hardware enhancements

- CoreSight ETR Architecture
  - New buffer mode Software FIFO2 with overflow interrupt
    - Better buffer management for perf
    - Support for Trace over Functional I/O
  - Restore read/write pointers
    - IOMMU is recommended
    - Standalone scatter-gather component (CoreSight Address Translation Unit)

- Arm v8.4-A Selfhosted-Trace extensions
  - Virtualization support for CPU Tracing
CoreSight Software hacks!

Software hack for the old TMC-ETR SG mode for save-restore
CoreSight System Guidelines

- CoreSight Base System Architecture
  - Power management by OS
  - Ready for use by OS
  - Trace Unit standards - Trace Generation Levels
    - Level0 - Minimal system (legacy)
    - Level1 - v8.4 Trace extensions
- Trace Capture Levels
  - Level0: Legacy support - ETR, ETB
  - Level1: Independent Trace buffer for each PE
  - Level2: Independent ETR compliant buffer + TG Level 1 + IOMMU
Conclusion

- CoreSight tracing system is evolving driven by self-hosted
  - Component Architecture (ETM, ETR) and thus the hardware
  - System design guildelines
- Towards out-of-the-box tracing capability
- Sometimes software hacks can give new powers to the hardware
Appendix - Coresight BSA Trace Capture Level 2