RISC-V External Debug
(aka JTAG debugging)

Tim Newsome <tim@sifive.com>
SiFive

Goals

- Debug system that works for everybody
  - Give feedback!
- Working system done by July 1, 2016
  - RISC-V on an FPGA
  - Hardware JTAG debugger
- Specification will be submitted to the RISC-V Foundation
- Open Source release of debugger
- Open Source release of implementations for Rocket Chip and Z-Scale

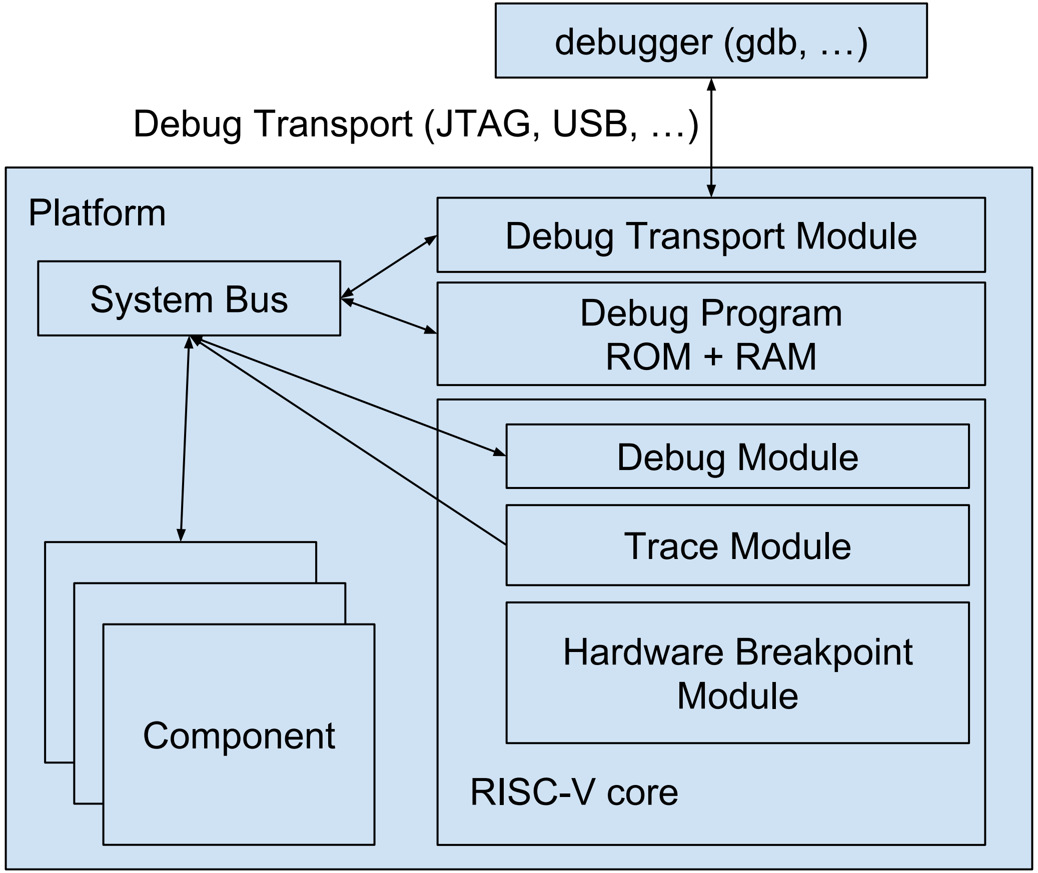
Status

- Specification is mostly complete

Features

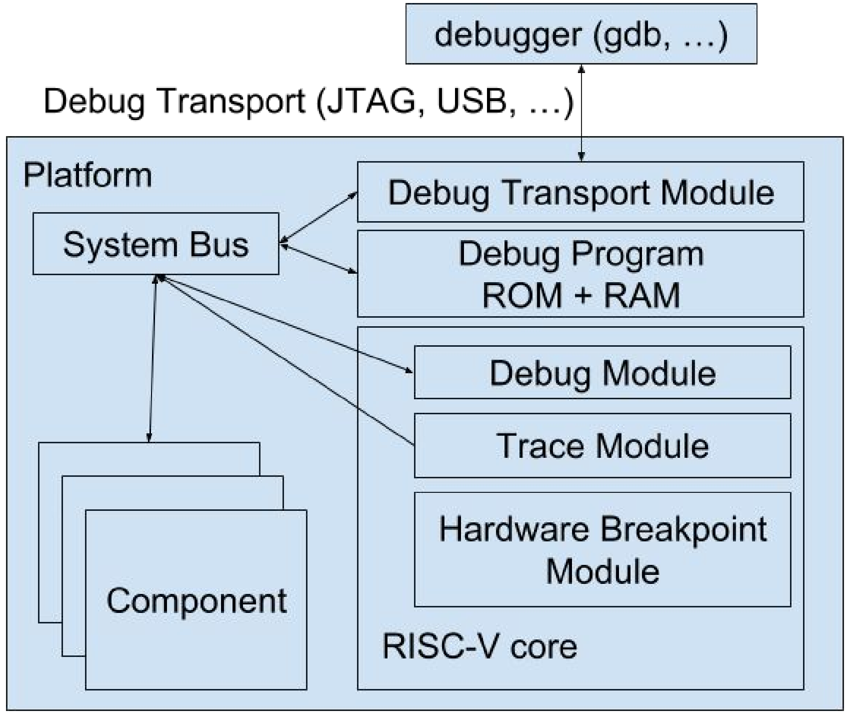
- Perform reads/writes on the system bus
- Framework to debug any component in the platform (halt/freeze/run/step)
- Software breakpoints.
- Access RISC-V registers.
- Execute arbitrary instructions on a halted RISC-V core.
- Use different debug transports. (Only JTAG is specified, 1149.7 coming.)
- Use debug transport for something else (eg. serial port).
- Debug code from very first instruction executed.
- Hardware breakpoints/trace triggers.
- Trace core execution to on- or off-chip RAM. (May not be implemented.)

Overview


debugger (gdb, …)
Debug Transport (JTAG, USB, …)
Platform
Debug Transport Module
System Bus
Debug Program
ROM + RAM
Debug Module
Trace Module
Hardware Breakpoint
Module
Component
RISC-V core

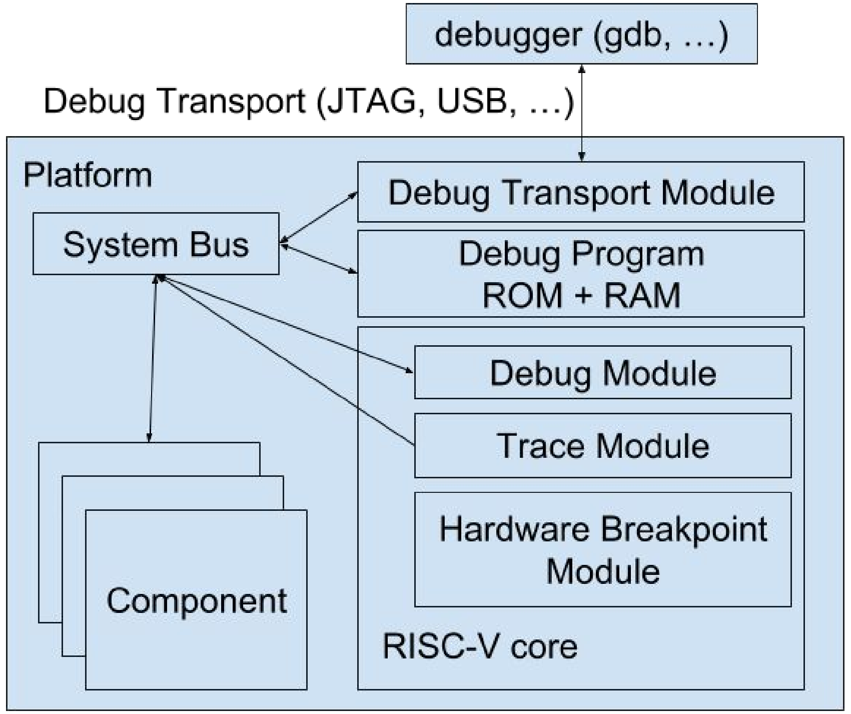
Debug Transport Module

- Provides access to the System Bus
- Implements a message queue
- Optional authentication



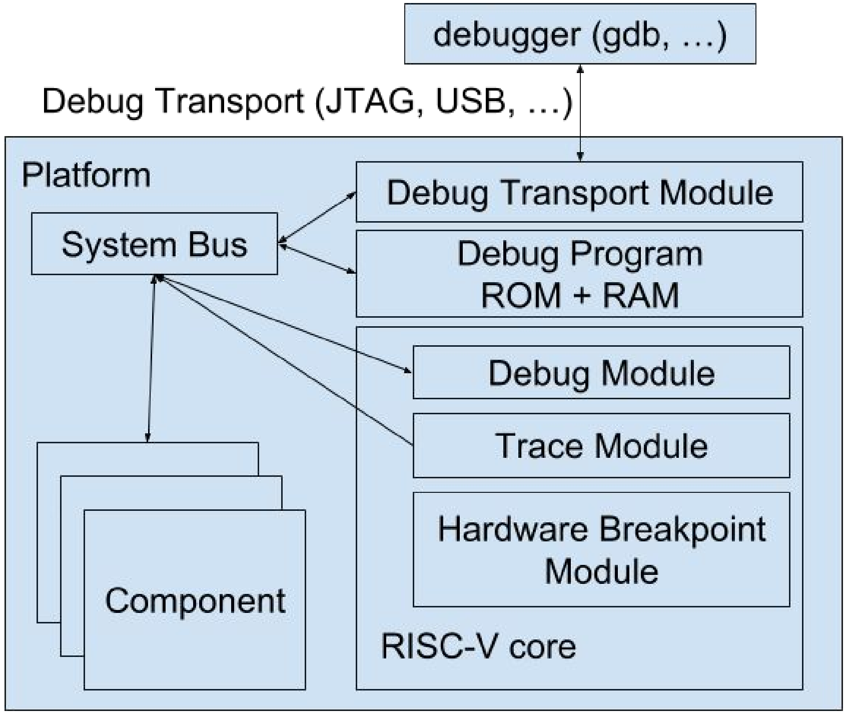
Generic Component Debugging

- Abstract
  - freeze
  - halt
  - step
  - run
- May send messages through message queue
- Optional authentication



RISC-V Control and Status Registers

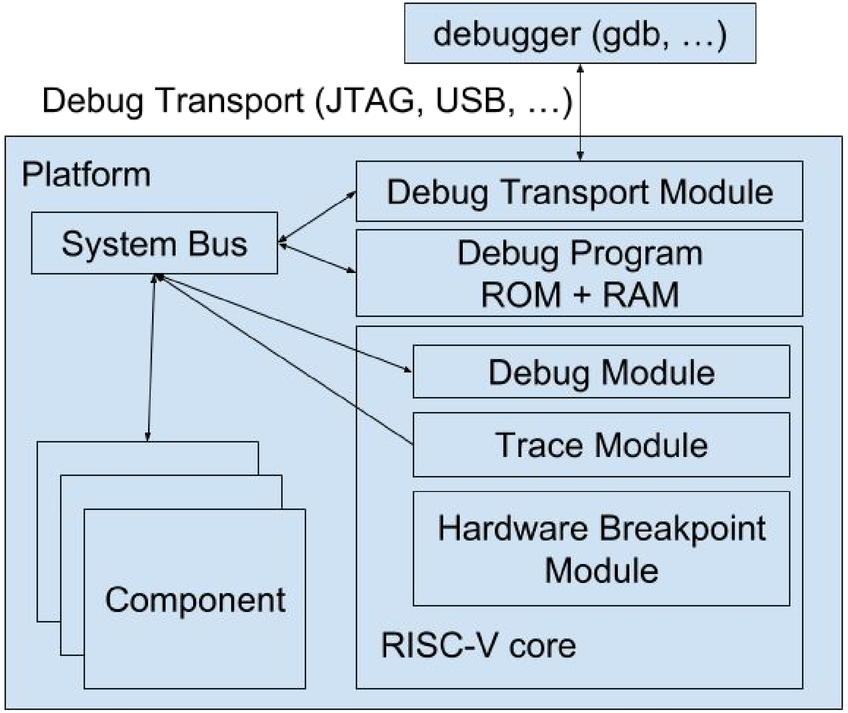
- CSRs are exposed on the System Bus
- New CSRs:
  - Debug Control and Status (dcsr)
  - Debug PC (dpc)
  - Debug Mailbox 0 (dmbox0)
  - Debug Mailbox 1 (dmbox1)
  - Debug State (dstate)
  - PC Sample (pcsample)
- Bus accesses to dmbox[01] set bits in dstate



Debug Memory

- Accessible through System Bus
- Not cached
- Shared between all cores

ROM	1KB, but more than 50% nops. “functions”: ● entry ● exit ● send_x8 ● receive_x8 ● instruction_loop
RAM	At least 8 bytes. 16 bytes recommended.



Halt

1. Core
   a. Saves PC to Debug PC
   b. Jumps to Debug ROM
      i. Writes x1 to mbox0
      ii. Writes x9 to mbox1
      iii. Signals debugger
      iv. Waits for read to mbox0
2. Debugger
   a. Notices message or polls dstate
   b. Reads PC from Debug PC
   c. Reads x9 from mbox1
   d. Writes 2 `j instruction_loop` instructions to Debug RAM
   e. Reads x1 from mbox0
3. Core
   a. Sees that mbox0 was read
   b. Jumps to Debug RAM
   c. Jumps to instruction_loop
   d. Writes x8 to mbox0
   e. Signals debugger
   f. Waits for write to mbox0
4. Debugger
   a. Notices message or polls dstate
   b. Reads x8 from mbox0

Write x13

1. Debugger
    a. Writes new value to mbox1
    b. Writes `mov x13, x8` to mbox0
2. Core
    a. Sees mbox0 was written
    b. Writes instruction in mbox0 to Debug RAM
    c. Writes mbox1 to x8
    d. Jumps to Debug RAM
    e. Executes `mov`
    f. Jumps back to instruction_loop
    g. Signals debugger

Small Debug Program

Pretend there exist some cache access CSRs, and Debug RAM is 24 bytes.

1.
```
        mov     x9, zero
loop:   csrw    CACHE_INDEX, x9
        csrr    x8, CACHE_DATA
        jal     send_x8
        addi    x9, x9, 1
        j       loop
```
2. Execute program
3. When reading the last value in send_x8, change code to `j instruction_loop`

Hardware Breakpoints

- Up to 4095 breakpoints supported (but 4 is more typical)
- Each breakpoint may support:
  - Exact address match
  - Address range match
  - Masked address match
  - Exact data match
  - Data range match
  - Masked data match
  - Trigger on load
  - Trigger on store
  - Trigger on execute

Triggered hardware breakpoints may:

- Cause Debug Exception
- Enter Debug Mode
- Start tracing
- Stop tracing
- Emit single trace sequence

Trace Data

Data consists of sequences of 4-bit packets:

- Nop
- PC, Count, data
- Branch Not Taken
- Branch Taken
- Trace Disable
- Trace Enabled, Version
- Privilege Level, Details
- Load Address, Count, data
- Store Address, Count, data
- Load Data, Count, data
- Store Data, Count, data
- Timestamp, Count, data

Count indicates the number of data packets to follow.

Missing data is filled in with previous value (address) or by sign extending (data).

Trace Features

- Output to System Bus, internal RAM (not specced), or external port (not specced)
- Control over which sequences are output
- Use Hardware Breakpoint Module to start/stop trace

Trace Uses

- Simply reconstruct all PC values
- Watch all writes to some area of memory
- Reconstruct all processor state (assuming enough bandwidth and a smart decoded)

Questions? Feedback?

Missing features?
Bits in the wrong order?
Debug hardware that should be supported?

Let me know: Tim Newsome <tim@sifive.com>

Work-in-progress spec will be posted to hw-dev@lists.riscv.org later today.