# SiFive Core Debug and Trace

SiFive RISC-V 2019 Symposium

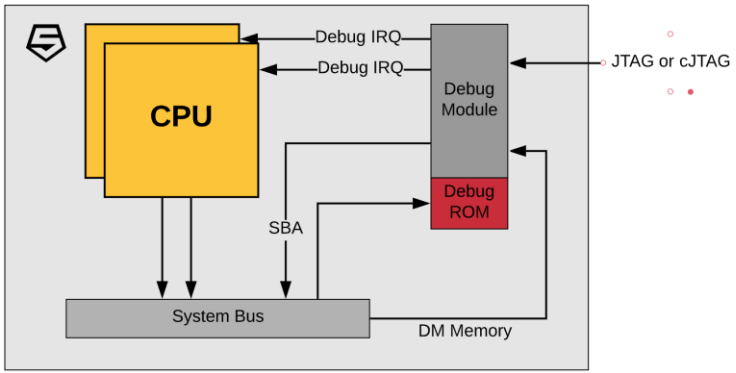
# SiFive Debug and Trace IP

## Access, Observe, Control

SiFive's Debug IP portfolio gives developers the power to efficiently debug SiFive based designs. From simple run control debug, to cross-triggering, to advanced multicore trace solutions, all delivered **pre-integrated** and **verified** together with SiFive's RISC-V Core IP in a single deliverable.

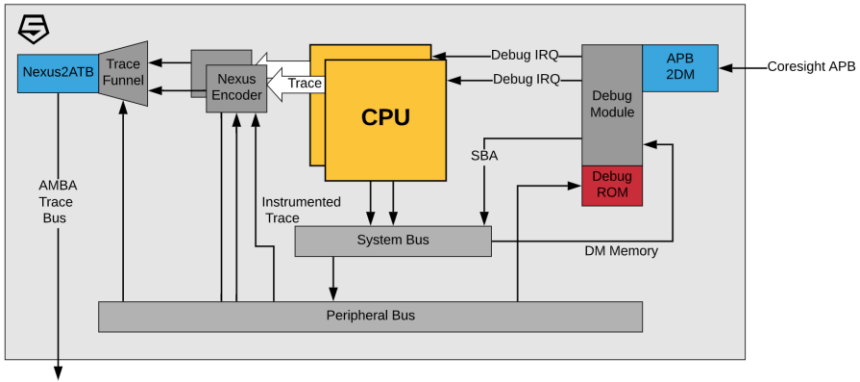
### Basic Debug IP

Everything needed for run-control debug, and more. Included with every Core IP subscription.




### Advanced Debug IP

Enabling Nexus trace, advanced debug control, and Coresight compatibility.




# Basic Debug IP

# Debug Features

- **Standard Run Control – go, halt**
  - When halted by probe, hardware breakpoint, external trigger, or executing an EBREAK software breakpoint instruction, core takes a debug interrupt @ROM code 0x800
  - ROM code reads *hartid* from CSR then writes it to an address which causes hardware to set halted bit for this hart
  - Probe polls target reading dmstatus to detect when hart goes into debug mode.
    - For multiple harts, haltsum registers return status of 32 harts per register
- **Return from Debug Mode: DM/Probe Handshake**
  - The DM has a FLAG byte per hart with 'RESUME' and 'GO' bits.
  - For 'resume', flag is set by probe and ROM code clears flag and executes a *dret* back to user code
  - For 'go', probe first writes an abstract command and optionally puts instructions into PROGBUF.
  - Hardware sets 'go' bit to start sequence. Debug code completes by executing an EBREAK and re-enters ROM.
- **Single step by instruction**
  - Hart has single step mode bit (*step* in dcsr); when set, a return from debug mode executes 1 instruction then returns
  - *stepie* control bit = 0 disables interrupts during single step mode
- **Read/write registers and memory**
  - gpr's are read/written with abstract command; CSRs, fp regs, and memory are read with PROGBUF instructions
  - Fast mode supports reading/writing blocks of memory

# Debug Features

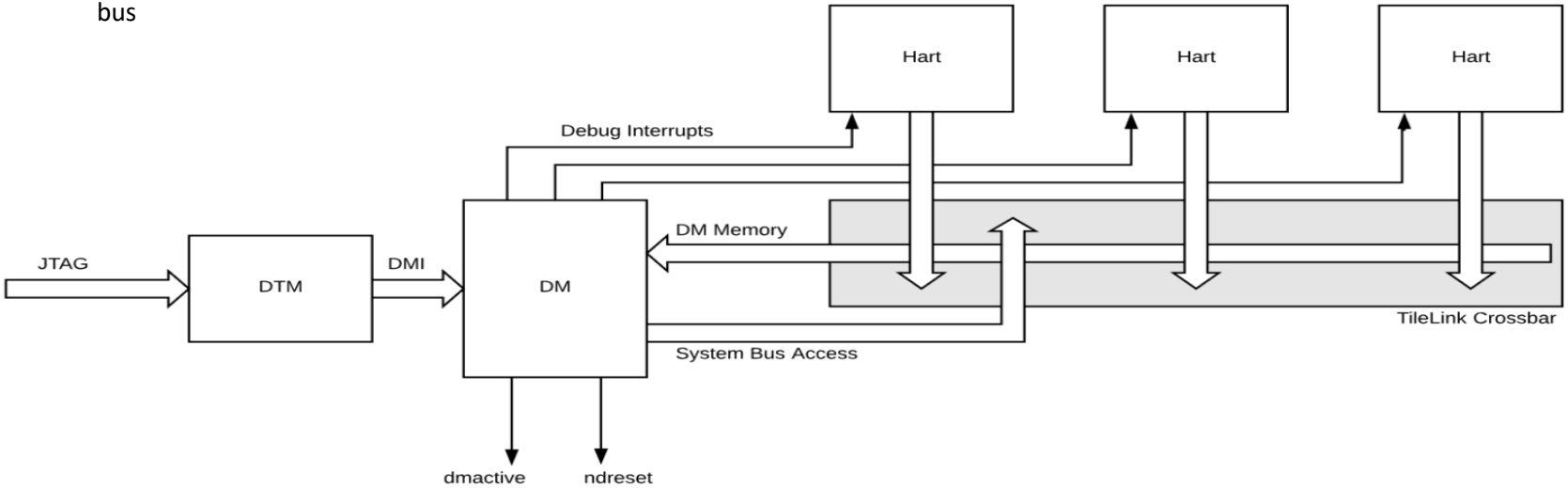
- **SBA – System Bus Access**
  - Optional, allows debugger to access memory directly while core is running
  - Useful for periodic sampling of pertinent variables
  - Can access memory-mapped instrumentation devices like trace buffer, PC sampling
  - Segger supports their RTT – Real Time Transfer – over SBA.
    - Up to 16 virtual terminals
    - Logging
  - Can be used to implement low overhead semi-hosting
- **Multi-hart and heterogeneous core debug control**
  - For multi-core, select hart (hardware thread) first
  - Hardware can be configured with up to 31 halt groups
  - Each group supports synchronous start, stop of all harts defined in group when any one enters debug mode
  - External trigger input/output pairs, go to periphery of core complex
    - Can configure 0 to 16 pairs
    - 2-wire request/acknowledge handshake (CoreSight CTI compatible), can cross clock boundaries
    - If trigger input asserted, will cause group of harts to enter debug mode
    - When hart group halts, external trigger out is asserted to a CTM (CoreSight cross-trig matrix) or customer logic

# Debug Features

- **Hardware Triggers**
  - Usually 2, 4 or 8 implemented, recommend in pairs
  - Virtual address compare
  - Filters: execute, load, store, and U, S, or M modes
  - Break happens before the event
    - instruction doesn't execute on match to the address
    - on data address match, data is not read or written
  - Actions: enter debug mode
    - When trace is included, additional actions are trace on, trace off, trace single
  - Address match on: single address, power-of-2 range (up to 64 bytes), >= address, < address
  - Two adjacent breakpoints can be set up for full address range when **chain** bit set
    Example: trigger0 set to >=lower address, trigger1 set to < upper address
    - Useful for triggering on a data structure access
    - Invalid or non-mapped code range(s)
    - Stack frame overflow or underflow

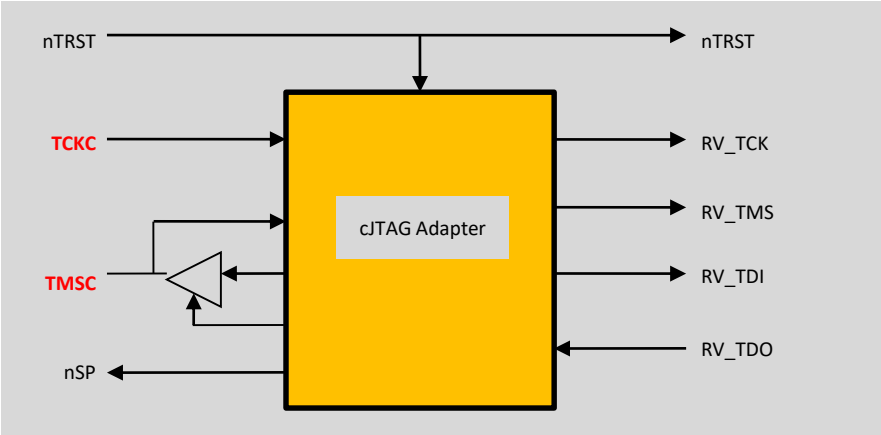
# Debug Block Diagram

- **DTM – Debug Transport Module**
  - Converts comm format (e.g. JTAG) into simple R/W bus
  - cJTAG (2-wire) front-end to JTAG - optional
  - Could have a parallel input like APB or TileLink in future
- **DMI – Debug Module Interface –** the simple R/W bus
  - 7-bit addr, 32-bit data, 8 control bits
- **DM – Debug Module**
  - Handles most functions related to debug
  - Slave to both DMI, and Hart(s) via TileLink/DM Mem bus
- **Hart – Hardware Thread**
  - One or more cpus controlled from one DM
  - Access to DM is shared, debug one core at a time
- **Debug Interrupts**
  - via probe, DM asserts debug int to get control of Hart
- **Debug ROM** (not shown)
  - a debug interrupt vectors the cpu to execute ROM code
- **SBA** (optional) – provides debug access to shared memory, bypasses cores, and can operate while cores run



# cJTAG Support

**Optional standalone converter from the 2-wire 1149.7 protocol (also called cJTAG) to 4-wire 1149.1.**




**cJTAG System Block Diagram**

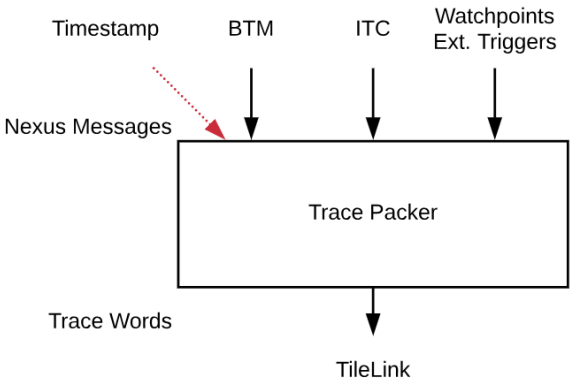

# Advanced Debug IP

# Instruction Trace Features

- **Trace protocol based on Nexus standard**
  - IEEE-ISTO 5001 standard since 2003, expanded in 2012 for SERDES trace sink
  - Established standard, protocol includes message types for extensions and customization
  - Well supported by the tools ecosystem such as Lauterbach, Ashling, Green Hills
- **Instruction trace messages**
  - Compresses trace by capturing only changes in program flow, using BTMs – Branch Taken Messages
  - **Direct Branch:** records number of half-words executed (I-CNT) since last branch, for branches where the destination is statically known
  - **Indirect Branch:** for branches with destinations that are determined at runtime
  - **Synchronization:** records full address for destination of an interrupt, or periodically as reference for other branches
  - **Ownership trace**: Message generated when instrumented code writes to the memory-mapped ITC - Instrumented Trace Component
    - Primarily used to record RTOS task ID when a context switch occurs
- **Timestamp** (optional)
  - Choice of internal (fixed frequency) or external timestamp, synchronized to TLclock
  - Width is parameterizable, typically 40 or 48 bits
  - Each Nexus message can include timestamp, or turned off to save bandwidth

# Extensions to Instruction Trace

- **Multiple message sources**
  - **BTMs** – CPU instruction trace
  - **ITC –** Instrumented Trace Component (optional)
    - A set of memory-mapped Stimulus registers that, when written to, generate messages that includes register index + 32 bit data value
    - Writes are non-blocking
    - Supports byte, half-word, word writes
    - 32 mapped addresses; writes to upper half include a timestamp with the index and data written
  - **Watchpoints –** Core watchpoints
    - **Actions:** start trace, stop trace, insert Program Trace Sync message
    - **Edge mode** – transition from no-match to match; start or stop trace
    - **Range mode** – trace while watchpoint true, don't trace when false
    - **Use:** record markers in trace, w/ timestamp
      - Can be used for precise point-to-point timing of code
      - Watchpoints can also be variable address matches
  - **External Triggers**
    - Up to 8 external triggers supported for controlling trace
    - **Actions:** start or stop trace, record Program Trace Sync messages
      - Provides inserting markers into trace based on external events



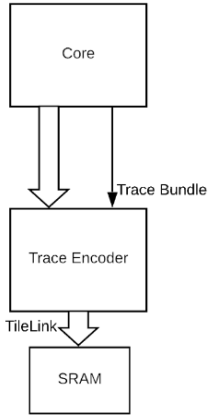

***Trace Packer Merges Messages from Multiple Sources and Compresses the Data***

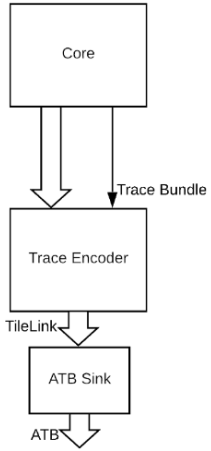
# Trace Flow & Configurations

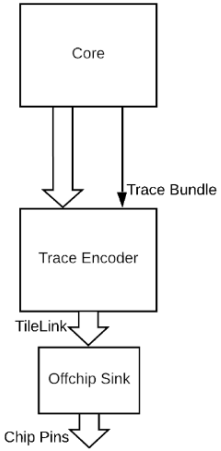
- **Trace Bundle** – also called the trace ingress port
  Uses Trace WG Ingress specification of signals
  from core required for instruction tracing
- **Trace Encoder** (TE) - logic that takes core signals and forms Nexus branch taken messages
- **TileLink** – point-to-point connection from the TE to SRAM and other trace sinks.
- **SBA** (System Bus Access) – provides access to system physical memory including TE control port and trace SRAM.

## Trace Sink Options

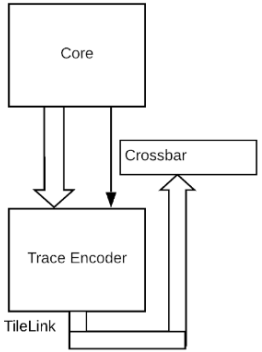
- Local dedicated SRAM, readable from DM->SBA
- ATB Sink, a bridge to ATB format, for CoreSight compatibility
- Off-chip port – dedicated clock/data pins for trace streaming
- Main memory – bridge to TileLink Crossbar to route trace to dedicated region of main memory. Uses a DMA block to create circular buffer.



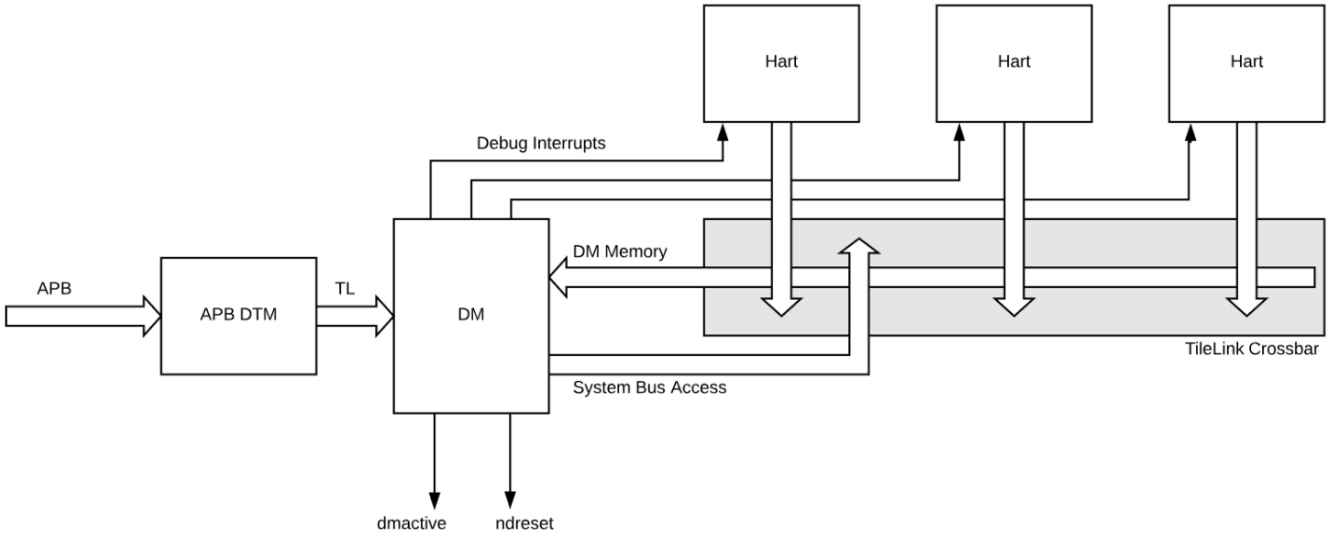

**November 2019**




**November 2019**

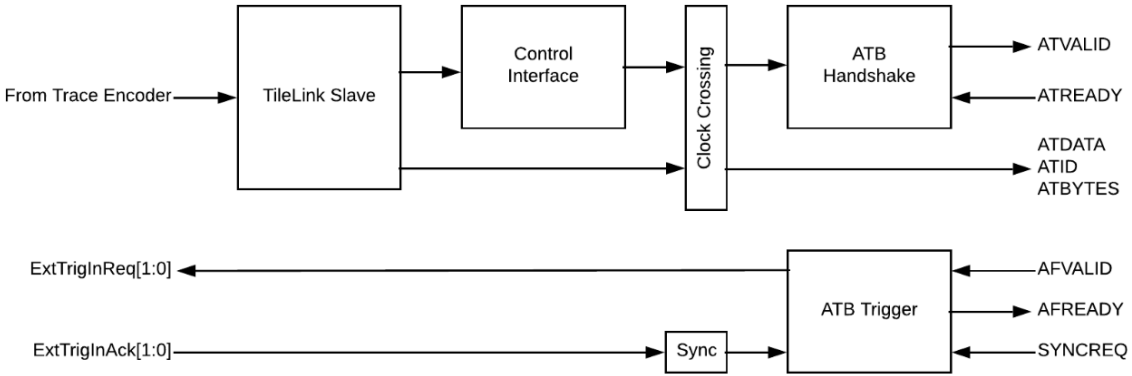
# SiFive Debug: CoreSight Compatibility

- **APB DTM – APB Debug Transport Module is alternative to JTAG connection to Debug Module (DM)**
- **Allows ARM DAP to control SiFive cores**
- **User can insert an AXI or AHB-to-APB bridge if needed**
- **APB Debug Port occupies 4K bytes of APB memory space.**
  - DMI addresses are word granular so each register occupied 4 bytes in the APB space



# SiFive Trace Compliance with ARM CoreSight

- **Instrumented Trace Component**
  - Equivalent of ARM ITM for capturing mem-mapped writes into trace
- **Timestamp Unit – Trace supports selection of external source, in this case the CoreSight timestamp**
- **Watch – Match signals from CPU trigger/watchpoints**
  - Provides markers into trace for instruction or data address matches; subset of ARM DWT (Data Watchpoint & Trace) block
- **External Triggers – up to 8 trigger inputs for controlling trace; req/ack signal pairs, compatible w/ARM CTI**
- **APB input to DTM** (Debug Transport Module)
- **Diagram below**
  - Trace Sink option for CoreSight ATB (ARM Trace Bus)
  - Bridge logic from Trace Encoder to ATB signals
  - Bridge logic from 2 triggers [1:0] request and acknowledge handshake signals to the ATB trigger signals
  - Control register: trace on/off control + field allows software to set DeviceID used on ATB for identifying trace source.




# Debug and Trace IP - Roadmap Items

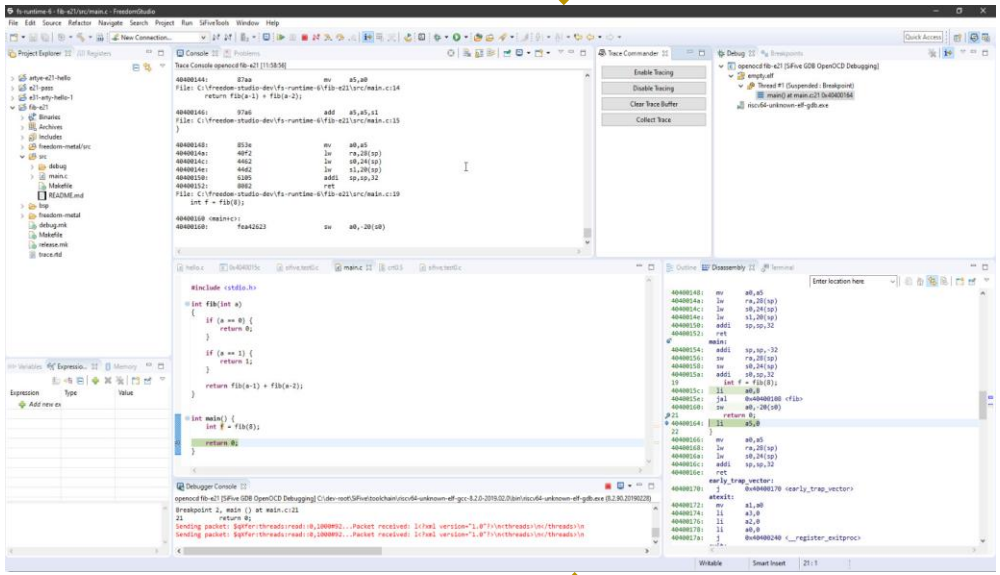
# Trace Roadmap

- **November**
  - Support for Multiple sinks
    - SRAM Trace buffer
    - Stream to system memory
    - Off-chip trace port with stall signal – 2, 4, or 8 data channels w/DDR clock
    - Reconfigurable at runtime
- **Future Enhancements**
  - Extended Trace Encoder filters; “Call-Return” and “Exception” tracing
  - User definable trace signals
  - User definable sampling signals
  - Tools for code performance and coverage analysis

# SiFive Freedom Studio – Get Started Fast with our Eclipse Based IDE

SiFive Freedom-E-SDK

SiFive RISC-V GCC Toolchain



Eclipse Plug-Ins

Simulators

GNU GDB Debugger

OpenOCD

Segger J-Link OB

Olimex Probe

Target

Target

# Support from Major Third-Party Tool Providers

**Support for SiFive RISC-V Trace and SiFive/ARM Integrated Debug**

- Lauterbach
- IAR
- Segger




# End