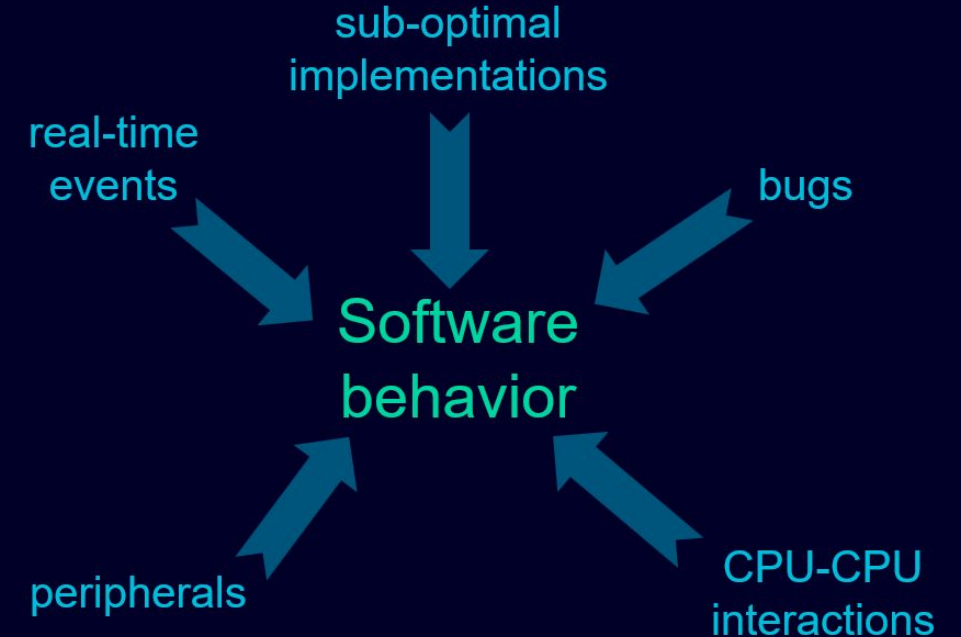


# RISC-V on-chip debug & trace solution: **Tessent™ UltraSight-V**

Devan Sharma, Account Technology Manager, Tessent Embedded Analytics

# Modern day SoCs are complex

- Complex SoC design
  - *Multicore Processor architecture*
  - *Integration of complex IPs*
- Complex Software
  - *Difficult to predict real-time behaviour*
  - *CPU-CPU interactions*
  - *Software optimization*
  - *Huge amount of raw-trace data*



# Traditional SoC debug needs help

The time and costs taken to debug and optimize software executing on complex RISC-V based SoCs are escalating

- Software code complexity
- Designs have multicore and multichips packaging techniques
- Heisenbugs and long-tail problems are difficult to identify & reproduce
- Silent data corruption

**Identifying hardware and real-time software issues  
require more efficient methods to debug, iterate,  
and scale**



Low  
debug  
productivity

# Tessent UltraSight-V

**A comprehensive end to end debug & trace solution for RISC-V based SoCs**

# Tessent UltraSight-V

Run control debug using  
GDB and OpenOCD  
with code instrumentation for  
logging capabilities

Highly-compressed processor  
trace  
(RISC-V E-trace spec)

Fast system memory access  
for ELF file uploads

Compatible with 3<sup>rd</sup> party tools  
including VS Code and  
Lauterbach TRACE32®

Scalable to a complete SoC  
system-level debug,  
optimization, and monitoring  
solution

HW verified by RISC-V core  
vendors, silicon proven down  
to 3nm and over 3GHz

Reduce costs, time taken to debug and optimize software executing on RISC-V SoCs



# Tessent UltraSight-V

## Embedded IPs

- Processor Analytic module
- Enhanced Trace encoder
- Static instrumentation
- Virtual console
- DMA
- Message Engine
- Communicator i.e. JTAG/USB/AXI

**A comprehensive end to end debug & trace solution for RISC-V based SoCs**

# Tessent UltraSight-V

## Embedded IPs

- Processor Analytic module
- Enhanced Trace encoder
- Static instrumentation
- Virtual console
- DMA
- Message Engine
- Communicator i.e. JTAG/USB/AXI

## UVM Verification IPs

- Sequence libraries
- Example tests
- UVM integration environment with virtual interfaces for each communicator IP

**A comprehensive end to end debug & trace solution for RISC-V based SoCs**

# Tessent UltraSight-V

## Embedded IPs

- Processor Analytic module
- Enhanced Trace encoder
- Static instrumentation
- Virtual console
- DMA
- Message Engine
- Communicator i.e. JTAG/USB/AXI

## UVM Verification IPs

- Sequence libraries
- Example tests
- UVM integration environment with virtual interfaces for each communicator IP

## Software



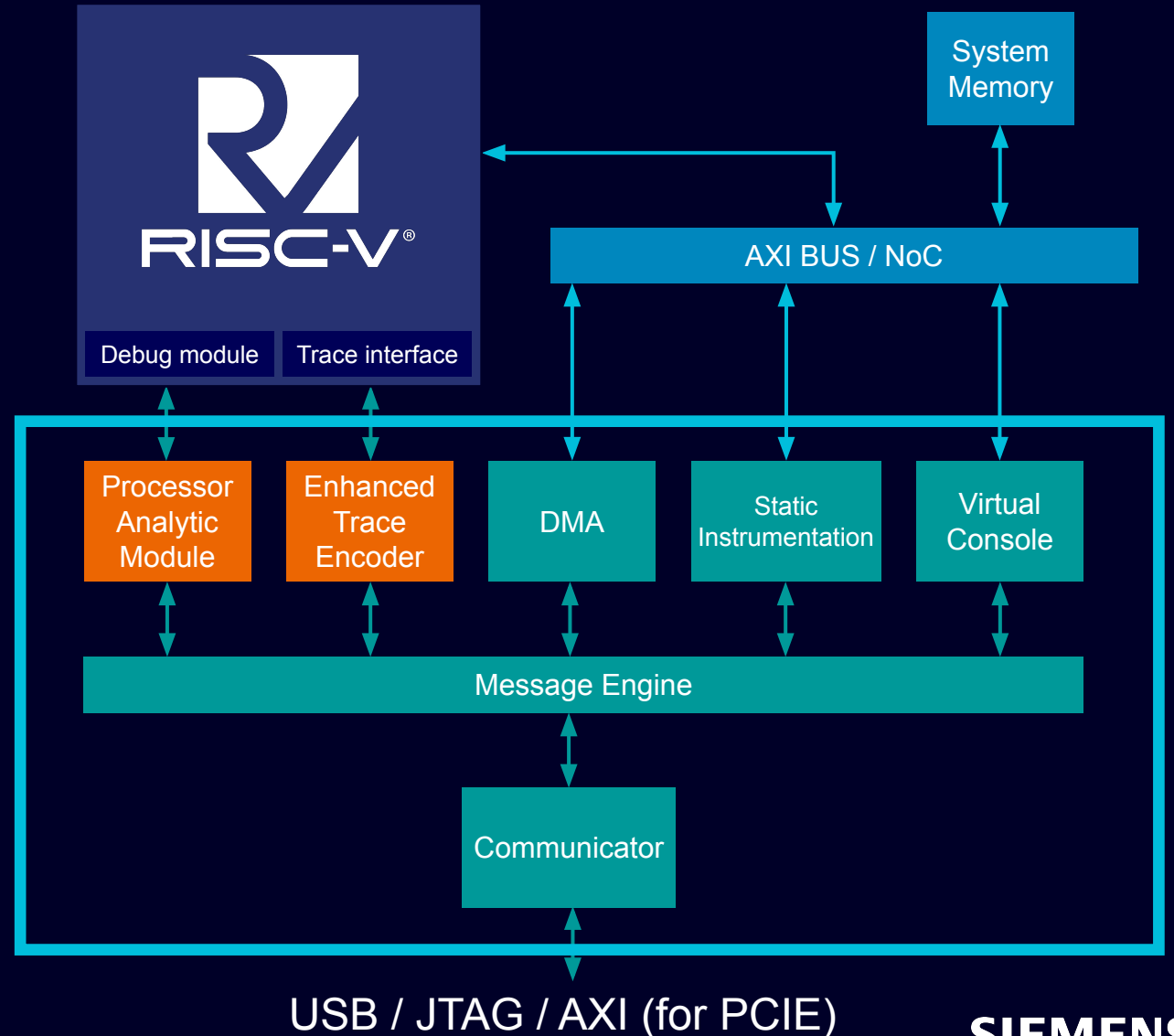
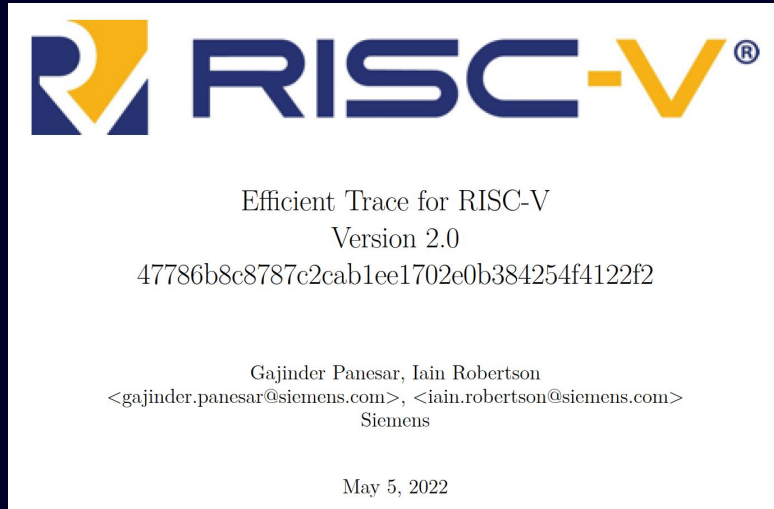
**A comprehensive end to end debug & trace solution for RISC-V based SoCs**



# Tessent UltraSight-V – debug and trace hardware

## Embedded IPs

- Processor Analytic Module
  - provides run control capabilities
- Enhanced Trace Encoder
  - Outputs processor trace (**E-trace**)



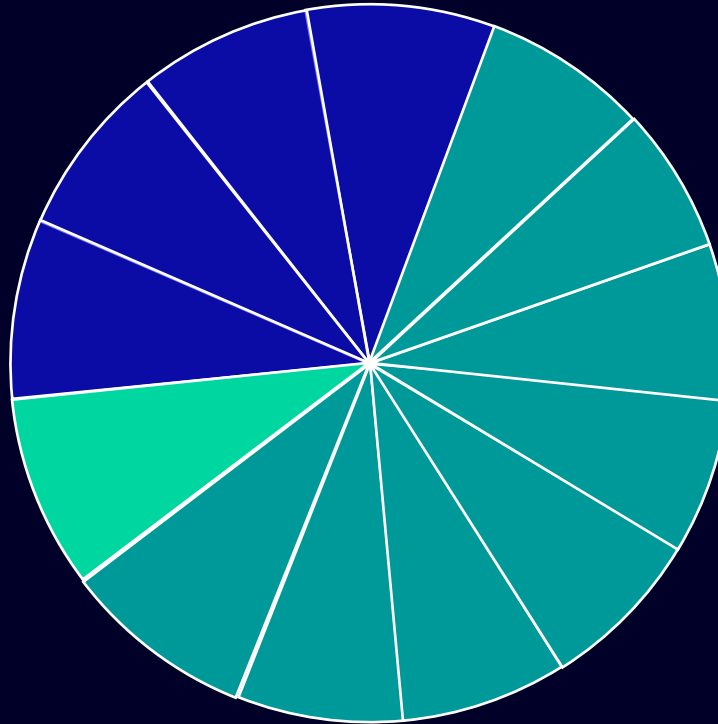
# Tessent UltraSight-V – Instruction trace support

## RISC-V Trace mandatory features

- Instruction trace
- Hart to encoder interface
- 'Delta Address' trace mode
- Efficient packet format

## Tessent Embedded Analytics

- Cycle accurate trace



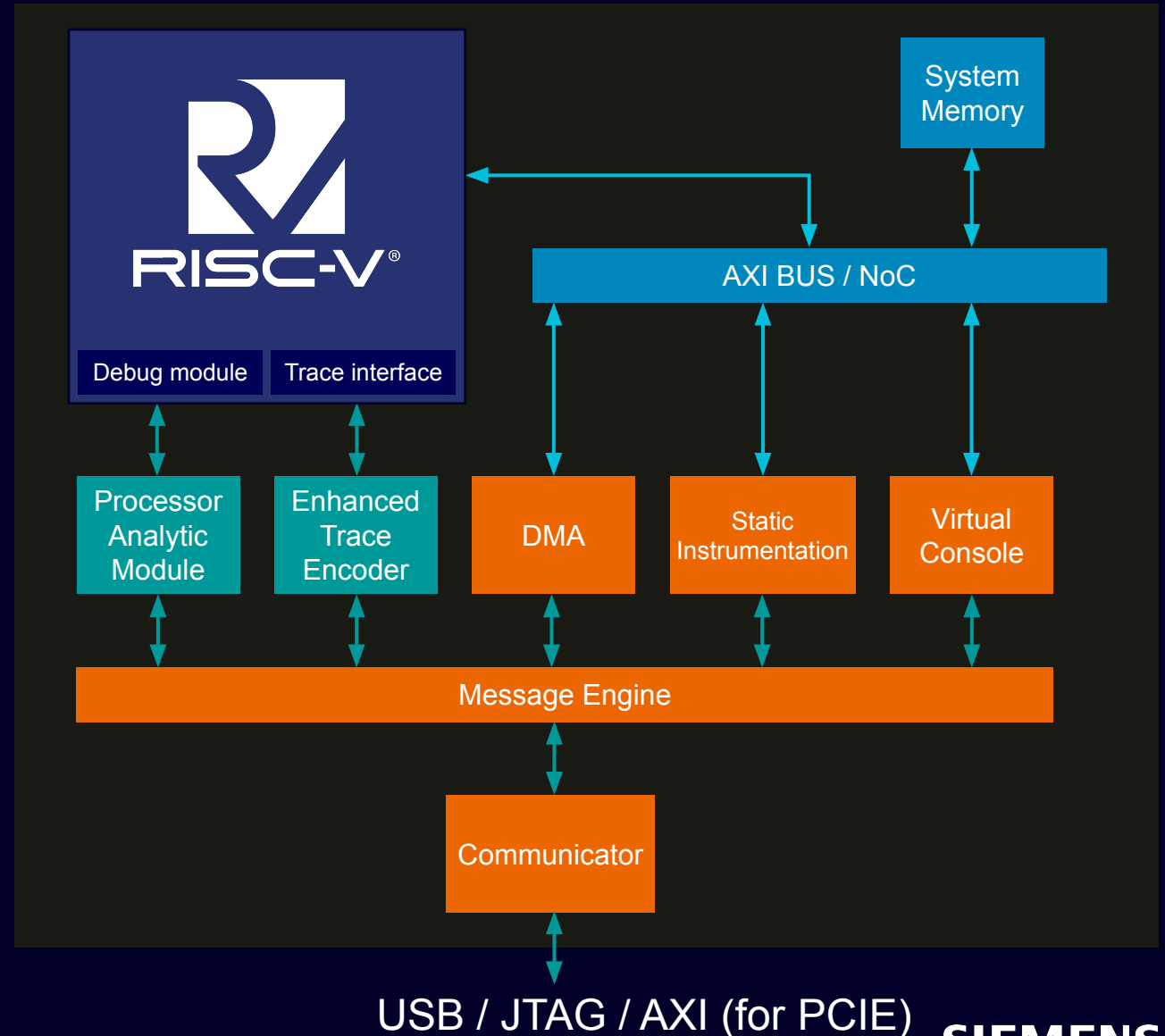
## RISC-V Trace optional features

- Multiple instruction retirement
- Sequentially inferable jump mode
- Implicit return mode
- Branch Prediction mode
- Jump Target Cache mode
- Full Address mode
- Sign-based compression
- Filtering

# Tessent UltraSight-V – efficient debugging

## Embedded IPs

- DMA
  - Fast read/write to memory system
- Static Instrumentation
  - Minimally invasive logging
- Virtual console
  - Replaces UART interfaces
- Messaging infrastructure
  - Scalable solution
- Communicators
  - Supports USB/JTAG/AXI (for PCIE)

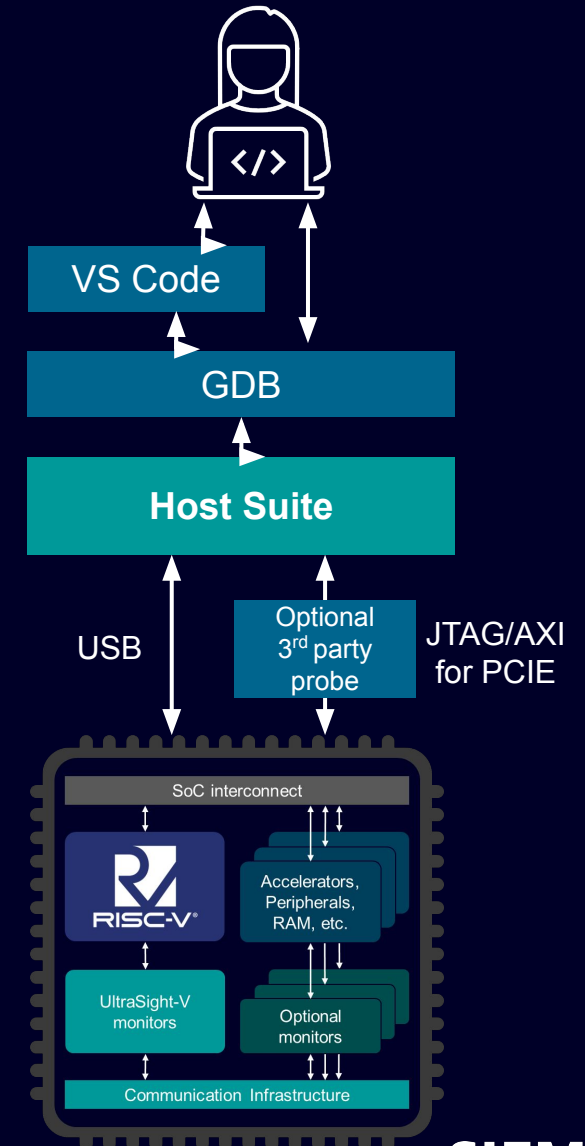


# Tessent UltraSight-V – software environment

Flexible platform for host-driven applications leveraging Tessent UltraSight-V and Embedded Analytics monitors

## Effective RISC-V debug and trace

- Runtime control and trace using GDB and OpenOCD
- Integration with common IDEs (e.g. VS Code, Eclipse) or CLI
- Supports USB, JTAG, and AXI for PCIE
- Supports RISC-V extensions and custom instructions



# Tessent UltraSight-V – RISC-V run control and trace

## IDE Support

- Compatible with Visual Studio Code

## Run-control for multi-core

- Comprehensive support of RISC-V is enabled through GDB and OpenOCD

## E-trace

- Instruction trace reconstruction to ASM and C/C++ via GDB

Reconstructed E-trace

The screenshot displays the Visual Studio Code IDE interface for RISC-V development. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar contains icons for Explorer, Search, Run and Debug, and Extensions. The main editor area is divided into several panels:

- VARIABLES:** Shows local variables like `left`, `x_bar`, `i`, `y`, `dir`, and `colour`.
- Registers:** Shows CPU registers like `zero`, `ra`, `sp`, and `rp`.
- WATCH:** A section for watching variables.
- CALL STACK:** Shows the current call stack, including `disp_fractal(uint32_t colour)` in `te.c`.
- BREAKPOINTS:** A list of breakpoints set in the code.
- Source Code:** The main editor shows the `te.c` file with a function `disp_fractal` and a `main` function. A breakpoint is set at line 97.
- RECONSTRUCTED E-TRACE:** A panel at the bottom right showing the reconstructed instruction trace in assembly and C/C++ format. It includes details like `Breakpoint 4, disp_fractal (colour=<optimized out>) at source/application/te/te.c:97` and the corresponding assembly instructions.

# Tessent UltraSight-V – Efficient debugging

## Fast code upload

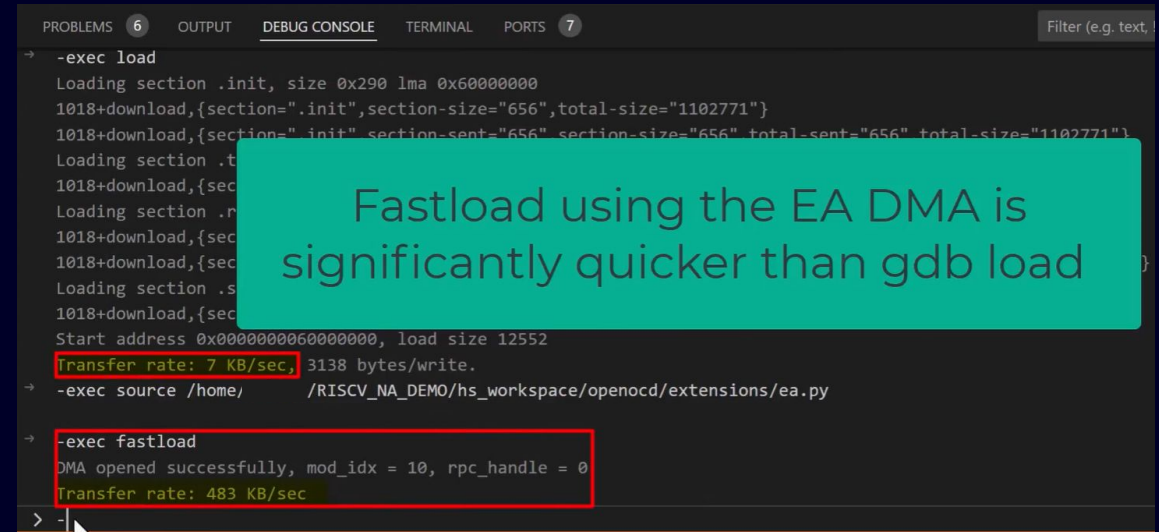
- Uses DMA
- Can be **70-100x faster** than normal GDB load

## Minimal overhead logging

- printf style debugging with timestamps
- Can take equivalent of **-5 instructions** (compared to printf at -100 instructions)

## Virtual Console

- Interact with embedded software using UART-like capabilities (e.g. telnet)



The screenshot shows the 'DEBUG CONSOLE' tab with the following content:

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS 7 Filter (e.g. text, ...)
```

```
→ -exec load
Loading section .init, size 0x290 lma 0x60000000
1018+download,{section=".init",section-size="656",total-size="1102771"}
1018+download,{section=".init" section-sent="656" section-size="656" total-sent="656" total-size="1102771"}
Loading section .text
1018+download,{section=".text",section-size="1024",total-size="1102771"}
Loading section .rodata
1018+download,{section=".rodata",section-size="1024",total-size="1102771"}
Loading section .sdata
1018+download,{section=".sdata",section-size="1024",total-size="1102771"}
Start address 0x00000000, load size 12552
Transfer rate: 7 KB/sec, 3138 bytes/write.
→ -exec source /home/.../RISCV_NA_DEMO/hs_workspace/openocd/extensions/ea.py
→ -exec fastload
DMA opened successfully, mod_idx = 10, rpc_handle = 0
Transfer rate: 483 KB/sec
> -|
```

A teal text box is overlaid on the screenshot with the text: "Fastload using the EA DMA is significantly quicker than gdb load".

## Instrument code with printf-style logging macros

```
static void setup( void )
{
    USR_FUNCTION_ENTRY( setup );
    USR_LOGGING_INFO_STRING( "start me up" );

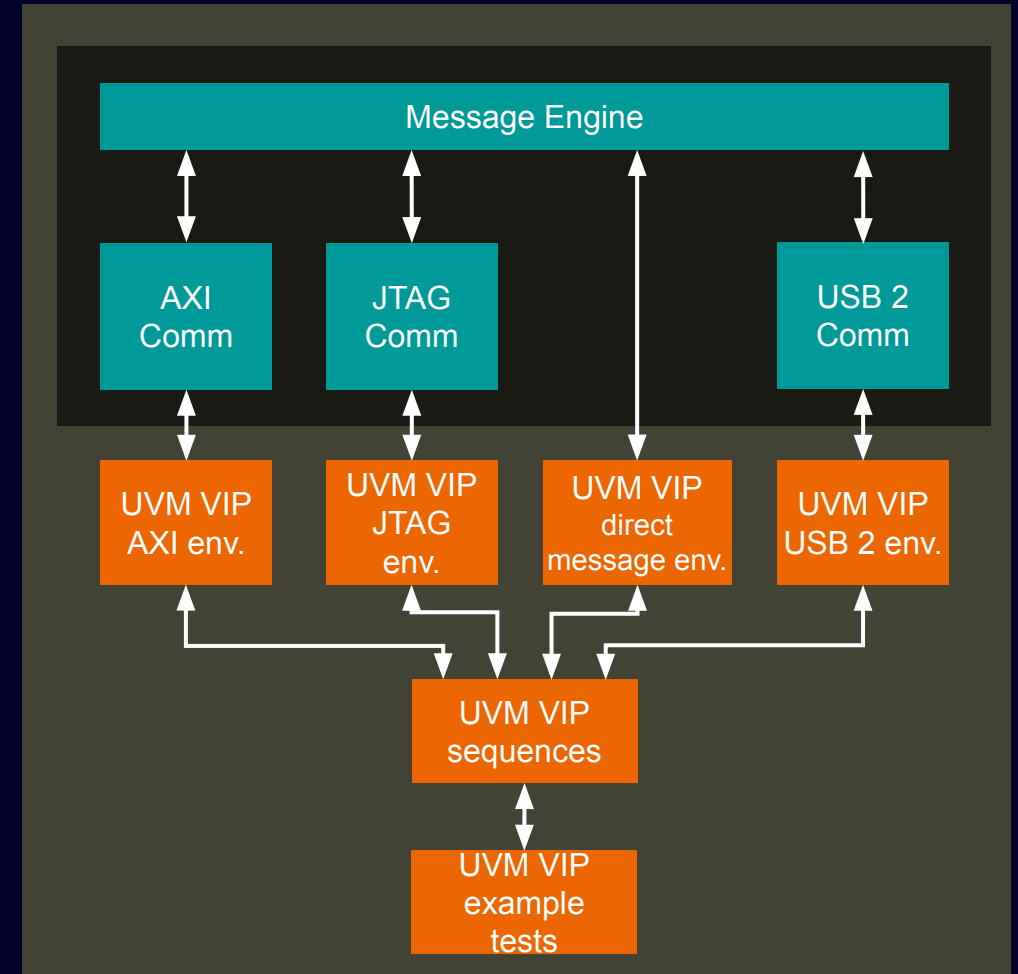
    smb_write_count_gap = 0;
    UST_LOGGING_DEBUG_PARAM3( "Pixel write y=%d, x=%d, colour=%u", y, x, colour );
}
```



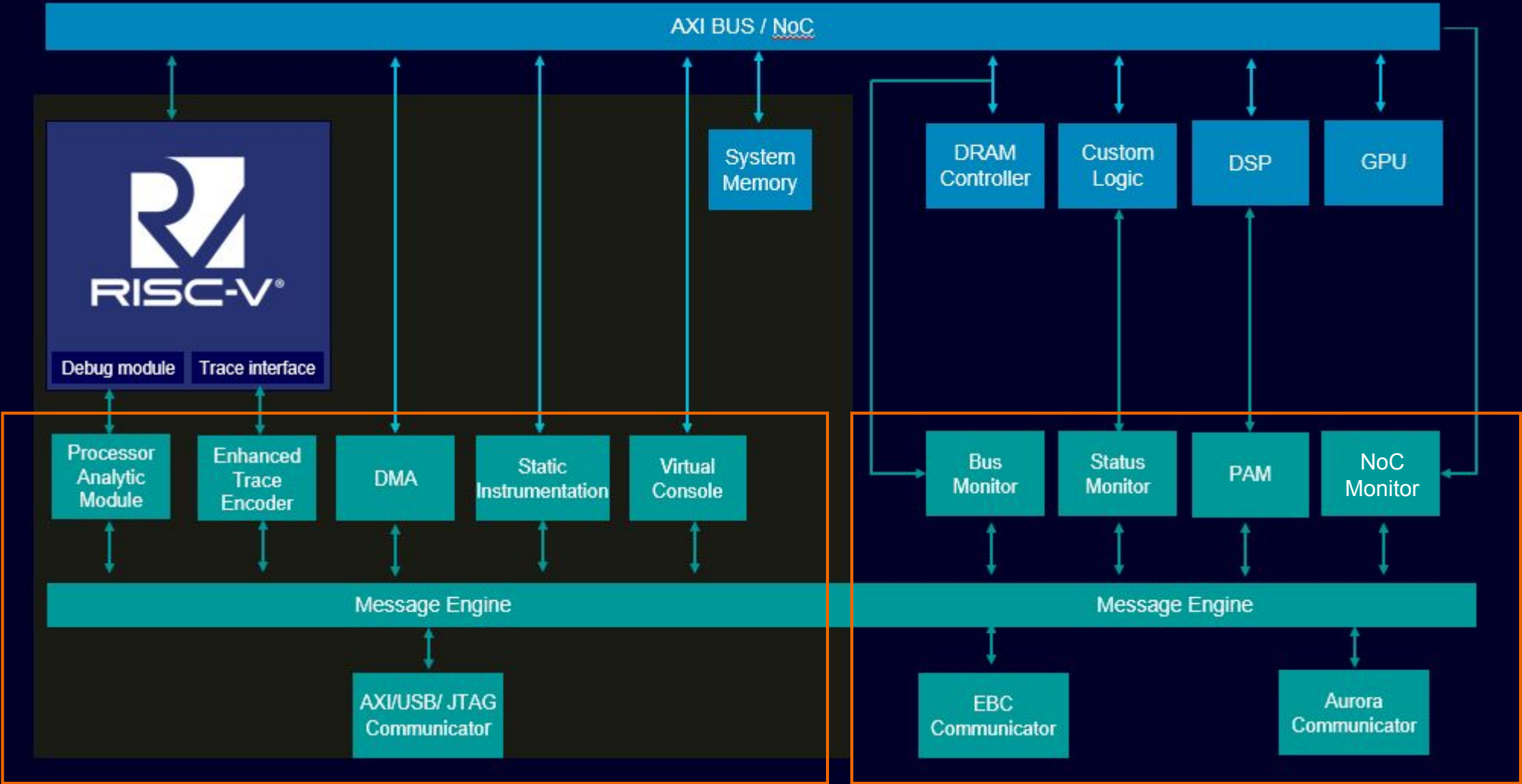
# Tessent UltraSight-V - UVM Verification IP

Ensures effective and comprehensive verification

- Verifies that UltraSight-V modules are correctly connected to each other and the SoC components
- UVM integration environment with virtual interfaces for each communicator IP
- Sequence libraries
- Example tests



# Scalability of Tessent UltraSight-V for complete SoC visibility



Tessent Ultrasight-V

Scalable solution for SoCs visibility

Thank you!