



# RISC-V Core IP Products

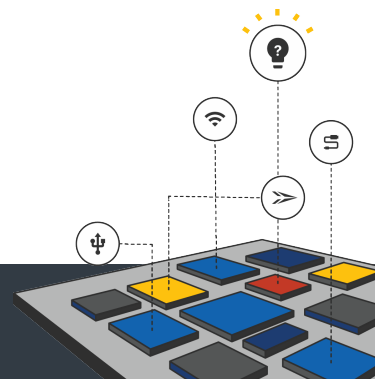
An Introduction to SiFive RISC-V Core IP

Drew Barbier – September 2017  
[drew@sifive.com](mailto:drew@sifive.com)



# SiFive RISC-V Core IP Products

- This presentation is targeted at embedded designers who want to learn more about SiFive's RISC-V Core IP products.
- This presentation will introduce the SiFive RISC-V Core IP products, the E31 Core Complex and the E51 Core Complex



# 3 Part Webinar Series

- RISC-V 101
  - The Fundamentals of RISC-V architecture
  - Recording: <https://info.sifive.com/risc-v-webinar>
- Introduction to SiFive RISC-V Core IP
  - Introduction to the SiFive E31 and E51 Core Complexes
- Getting Started with SiFive RISC-V Core IP
  - November 2017



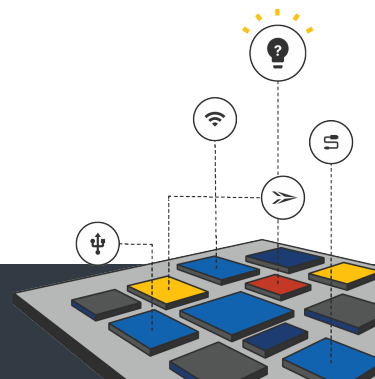
Study



Evaluate



Buy


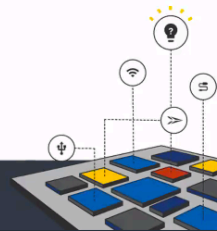


# How to ask questions

[Audio Settings](#) **Q&A** [Chat](#) [Raise Hand](#)

## E31 Core Complex Feature Table

Features	E31 Details
Architecture	RV32IMAC
Modes Supported	Machine and User
Microarchitecture	5+ stages, in order execution
Hardware Multiply/Divide (M)	Multiplier is pipelined, 8 bits per cycle Divide is 1 bit per cycle, with early termination
Atomic Support (A)	Yes
Compressed Support (C)	Yes
PMP Regions	8
Branch Prediction	40-entry BTB, 128-entry BHT, 2-entry RAS
Instruction Subsystem	16KB, 2-way associative Instruction Cache Can re-configure up to 8KB as ITIM
Data Subsystem	Up to 64KB Data Scratchpad (DTIM)
Interrupts	16 Local Interrupts, with vectored addresses
Platform Level Interrupt Controller	255 inputs, 7 priority levels
Core Local Interruptor	1 timer interrupt, 1 SW interrupt
Debug	JTAG Support, 2 programmable breakpoints

13    © 2017 SiFive. All Rights Reserved.



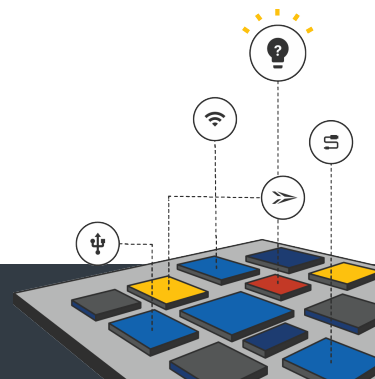
# SiFive Introduction

Jack Kang, VP of Product and Business Development



# Introduction to SiFive

- Founded by the inventors of RISC-V
- Maintain the open-source Freedom Platform
  - <https://github.com/sifive/freedom>
  - Including popular “Rocket” RISC-V Core
- License RISC-V cores
- Build custom SoCs based on the Freedom Platform
- Bring the power of open-source to hardware



# SiFive Products: RISC-V SoCs and RISC-V IPs

Tailored RISC-V Solutions for both Chip and System Designers

## RISC-V Core IP



### Low-power, 32-bit and 64-bit Embedded CPU IP

- Standard RISC-V extensions and privileged modes
- Physical Memory Protection
- Microcontrollers, IOT, Housekeeping cores



### High-performance, Unix-capable, 32-bit and 64-bit CPU IP

- Standard RISC-V extensions and privileged modes
- Virtual Memory Support
- Application Processors, Datacenter Accelerators

## SiFive Freedom SoCs



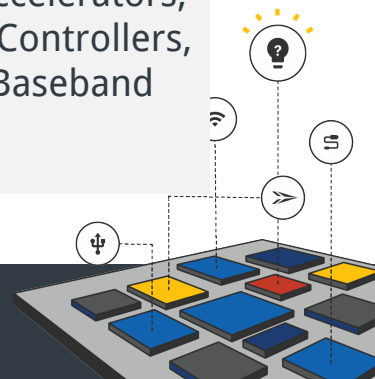
### Low-power, 32-bit microcontrollers

- TSMC 180nm
- Digital and Analog peripherals
- Edge Computing (AI), Embedded, Smart IOT, Wearables



### High-performance, 64-bit multi-core SoCs

- TSMC 28nm
- Cache coherent accelerator support
- High speed peripherals: PCIe Gen3, GbE, DDR3/4
- Datacenter Accelerators, Storage, SSD Controllers, Networking, Baseband



# SiFive changing the IP Business Model

www.sifive.com



Study

Detailed information  
Datasheet  
Power, Perf, Area

Free  
No NDA  
No Registration



Evaluate

FPGA Bitstream  
Evaluation RTL  
Freedom Studio IDE

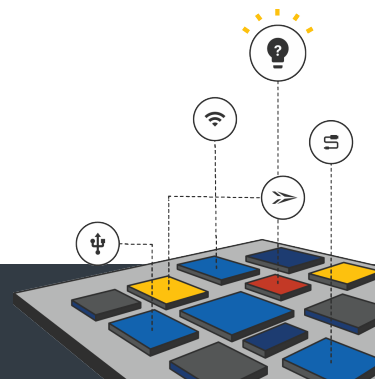
Free Evaluation License



Buy

Configure Online  
Transparent Pricing

7-page license agreement





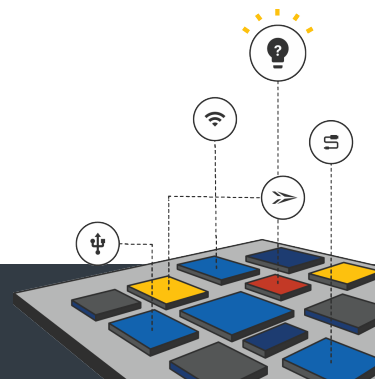
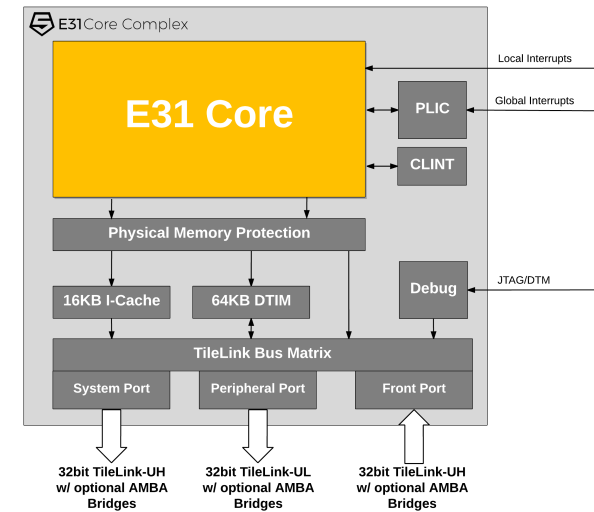
# E-Series Products



# E31 Core Complex

## High Performance RISC-V MCU

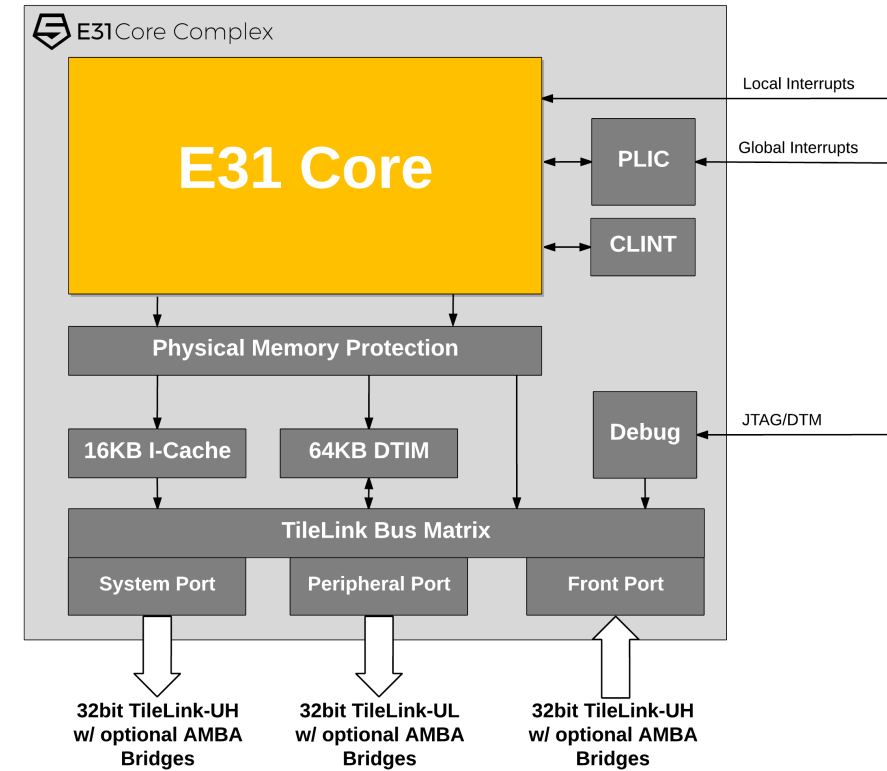
- 32-bit, RV32IMAC RISC-V CPU Core
- In-order, 5-6 stage variable pipeline
- Silicon proven, most widely deployed RISC-V core in the world
- Choice of buses: TileLink, APB, AHB, AXI4
- Smaller area, lower power compared to competing ISAs
- Designed for high performance embedded systems:
  - Smart IOT
  - Wearables
  - Embedded Microcontrollers



# E31 Core Complex

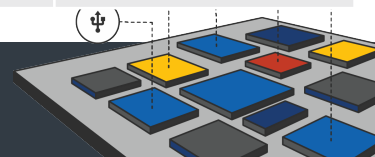
## High Performance RISC-V MCU

- Higher Performance –
  - 320+MHz in TSMC 180G
  - 515.2+ Total DMIPS, 873.6 Total CoreMarks at TSMC180G
- Flexible Memory Architecture
  - I-Cache can be reconfigured into I-Cache + ITIM
  - DTIM for fast on Core Complex Data Access
  - Off Core Complex memory access through System and Peripheral Ports
- Supports Local and Global Interrupts
  - 271 total interrupts
- 8 Region Physical Memory Protection Unit



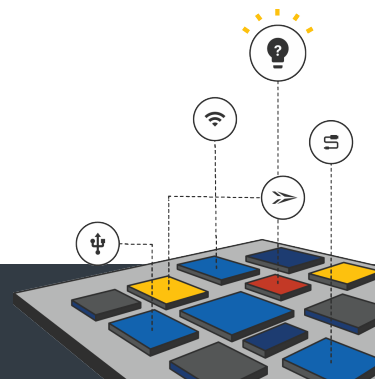
Specification	28nm HPC	55nm LP
E31 Core Complex Area* (with 16KB I\$/16KB DTIM)	0.187 mm <sup>2</sup>	0.593 mm <sup>2</sup>
E31 Core Only Area* (w/o SRAM)	0.0286 mm <sup>2</sup>	0.0507 mm <sup>2</sup>
Frequency	Typical: 1.45 GHz Worst: 900 MHz	Typical: 540 MHz Worst: 320 MHz
Power (DMIPS/mW)	59.4 DMIPS/mW at 1.4 GHz 2300 Total DMIPS	20.8 DMIPS/mW at 500 MHz 805 Total DMIPS

\*Area calculated with 85% utilization



# E31 Core Complex Feature Table

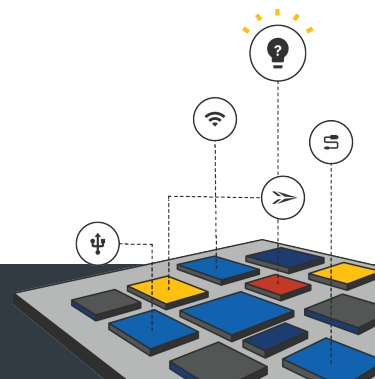
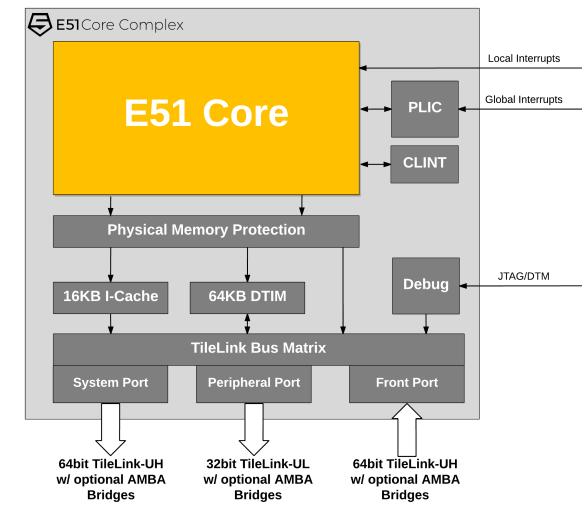
Features	E31 Details
Architecture	RV32IMAC
Modes Supported	Machine and User
Microarchitecture	5+ stages, in order execution
Hardware Multiply/Divide (M)	Multiplier is pipelined, 8 bits per cycle Divide is 1 bit per cycle, with early termination
Atomic Support (A)	Yes
Compressed Support (C)	Yes
PMP Regions	8
Branch Prediction	40-entry BTB, 128-entry BHT, 2-entry RAS
Instruction Subsystem	16KB, 2-way associative Instruction Cache Can re-configure up to 8KB as ITIM
Data Subsystem	Up to 64KB Data Tightly Integrated Memory
Interrupts	16 Local Interrupts, with vectored addresses
Platform Level Interrupt Controller	255 inputs, 7 priority levels
Core Local Interruptor	1 timer interrupt, 1 SW interrupt
Debug	JTAG Support, 2 programmable breakpoints



# E51 Core Complex

## 64-bit RISC-V MCU

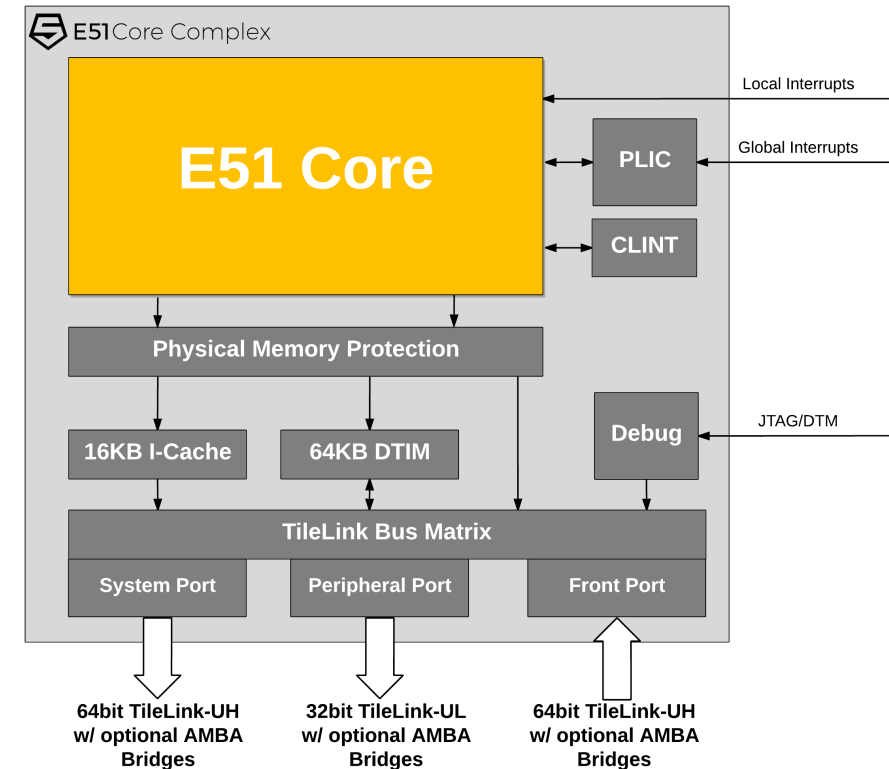
- 64-bit, RV64IMAC RISC-V CPU Core
- In-order, 5-6 stage variable pipeline
- Smallest 64-bit core in the market today
- Choice of buses: TileLink, APB, AHB, AXI4
- 64-bit core makes it ideal for integration into larger 64-bit SoC Systems
  - Example uses: “Minion” Core, Boot Processor, Security Processor
- Other embedded applications that benefit:
  - SSD Controllers
  - Networking Applications



# E51 Core Complex

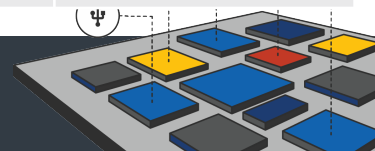
## 64-bit RISC-V MCU

- Better integer performance than 32-bit or smaller architectures
- Supports the RISC-V C Extension
  - Smaller code size than competing 64bit architectures
- Extended memory map
  - Support for up to 40 physical address bits
- Supports Local and Global Interrupts
  - 527 total interrupts



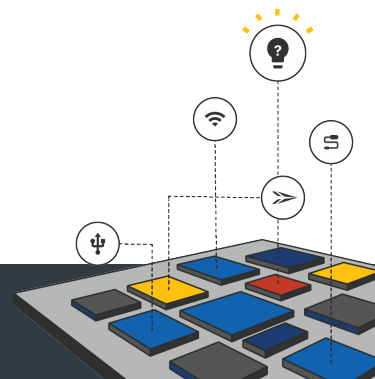
Specification	28nm HPC	55nm LP
E51 Core Complex Area* (with 16KB I\$/16KB DTIM)	0.236 mm <sup>2</sup>	0.704 mm <sup>2</sup>
E51 Core Only Area* (w/o SRAM)	0.0473 mm <sup>2</sup>	0.0877 mm <sup>2</sup>
Frequency	Typical: 1.45 GHz Worst: 900 MHz	Typical: 540 MHz Worst: 320 MHz
Power (DMIPs/mW)	58.0 DMIPs/mW at 1.4 GHz 2585 Total DMIPS	24.7 DMIPs/mW at 500 MHz 905 Total DMIPS

\*Area calculated with 85% utilization



# E51 Core Complex Feature Table

Features	E51 Details
Architecture	RV64IMAC
Modes Supported	Machine and User
Microarchitecture	5+ stages, in order execution
Hardware Multiply/Divide (M)	Multiplier is pipelined, 8 bits per cycle Divide is 1 bit per cycle, with early termination
Atomic Support (A)	Yes
Compressed Support (C)	Yes
PMP Regions	8
Branch Prediction	40-entry BTB, 128-entry BHT, 2-entry RAS
Instruction Subsystem	16KB, 2-way associative Instruction Cache Can re-configure up to 8KB as ITIM
Data Subsystem	Up to 64KB Data Tightly Integrated Memory
Interrupts	16 Local Interrupts, with vectored addresses
Platform Level Interrupt Controller	511 inputs, 7 priority levels
Core Local Interruptor	1 timer interrupt, 1 SW interrupt
Debug	JTAG Support, 2 programmable breakpoints



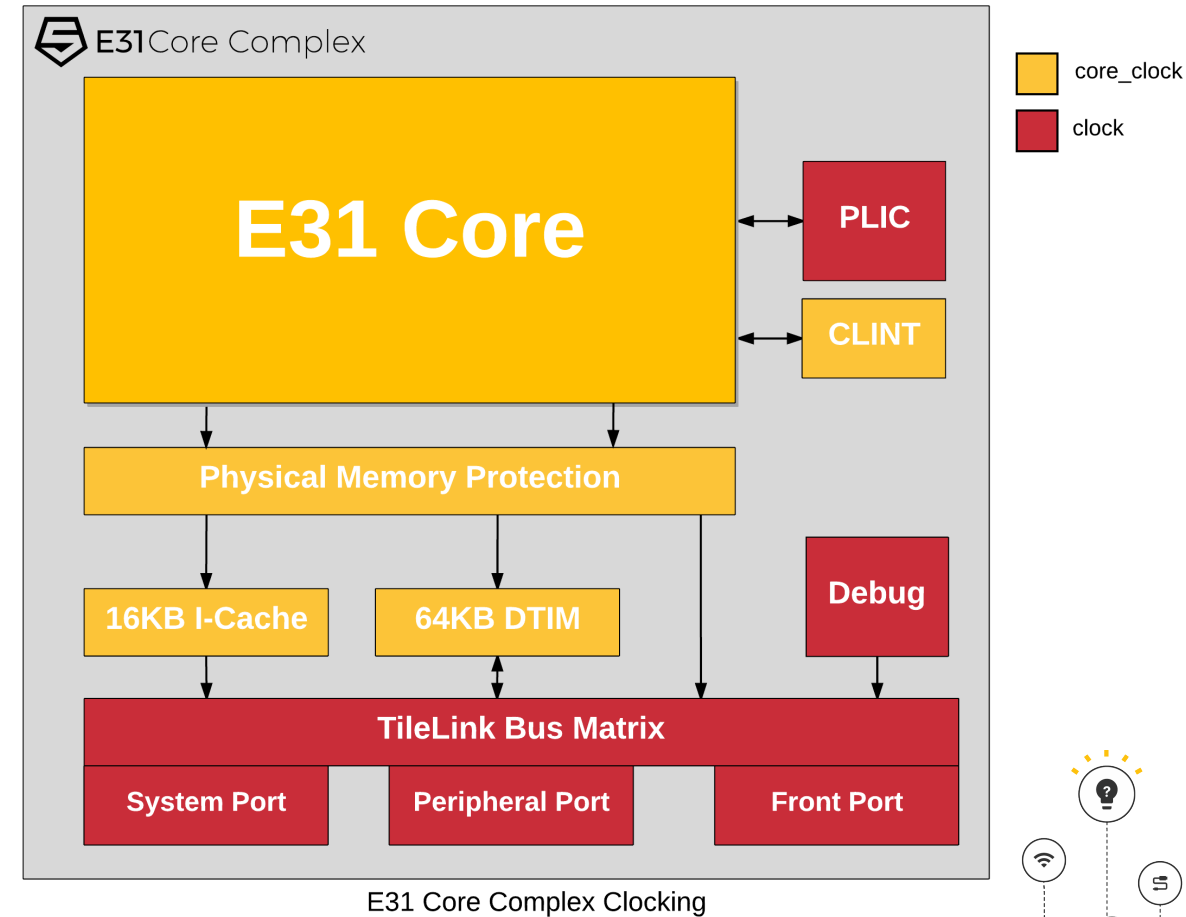


# E-Series Core Complex Features



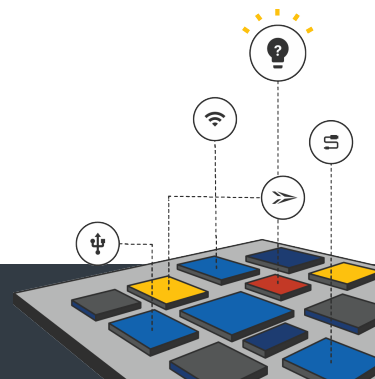
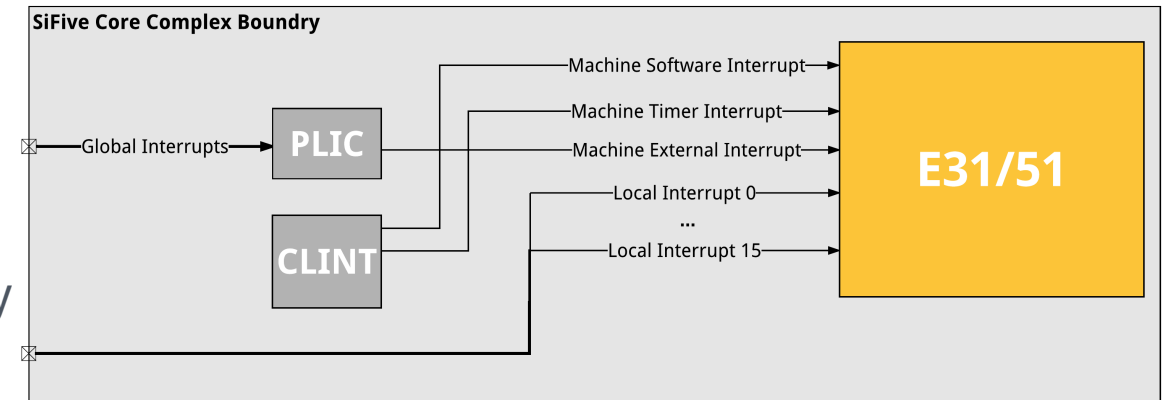
# E-Series Core Complex Clocking

- *core\_clock*
  - Main CPU and L1 memory clock
- *clock*
  - Secondary peripheral clock
  - Frequency must be an integer multiple of *core\_clock*
  - 1:1 ratio is acceptable
- *rtc\_toggle*
  - Real Time Clock input as defined by RISC-V Architecture
  - Must run at strictly less than half the rate of *clock*
- $core\_clock \geq clock > (2 \times rtc\_toggle)$



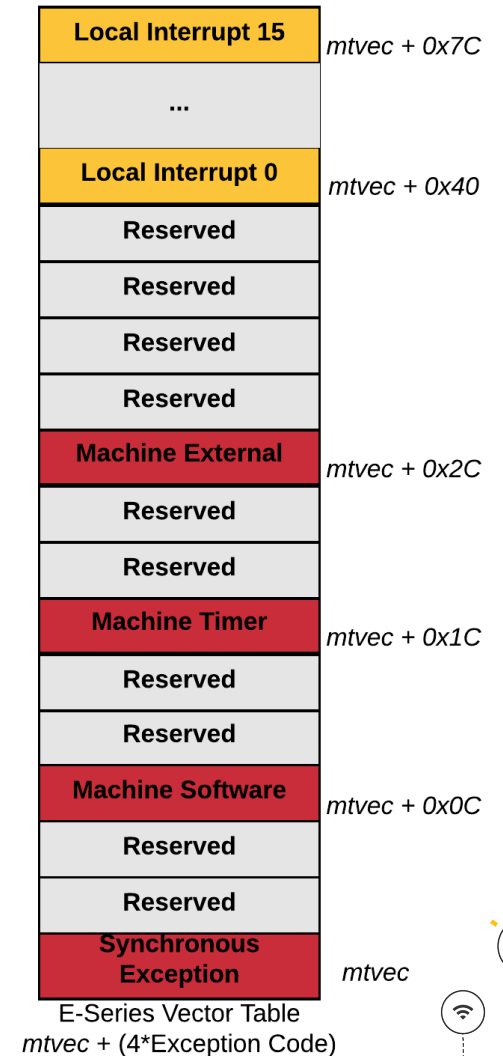
# SiFive RISC-V Core IP Interrupts

- 2 Interrupt Types: Local and Global
- Local Interrupts
  - Connected directly to the core = Lower Latency
  - Can be vectored = Lower Latency
  - Including additional Local Interrupts 0 - 15
- Global Interrupts
  - Interrupts routed through the Platform-Level Interrupt Controller (PLIC)
  - Programmable priority levels and thresholds
  - The PLIC is in *clock* domain
- Overall Core Complex Prioritization of Interrupts:
  - Local Interrupts > Machine External > Machine Software > Machine Timer



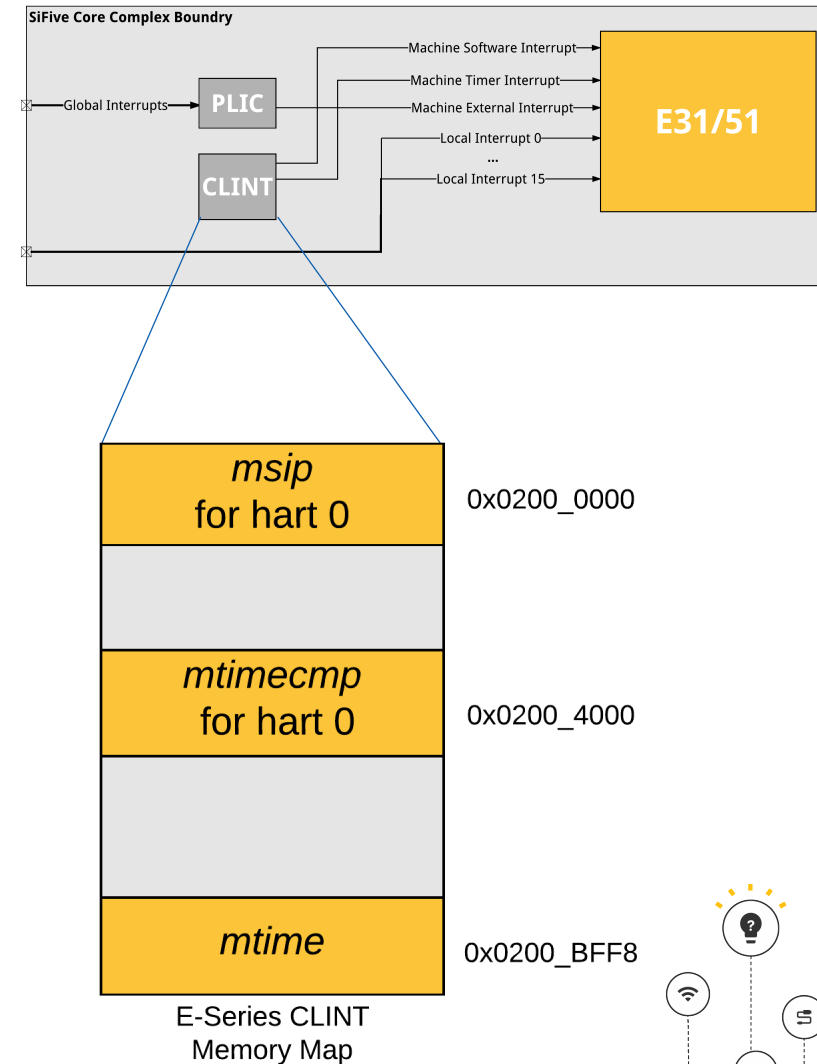
# SiFive Local Interrupts

- Can be Vectored by setting the LSB of *mtvec*
  - Allows for jumping directly to an interrupt handler without first reading *mcause*
- Local Interrupts have a fixed priority scheme
  - The higher the Interrupt ID, The higher the Priority
  - Local Interrupt 15 is the highest priority Interrupt in the system and special due to its position in the vector table - no jump necessary



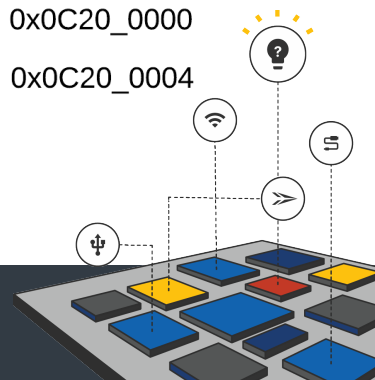
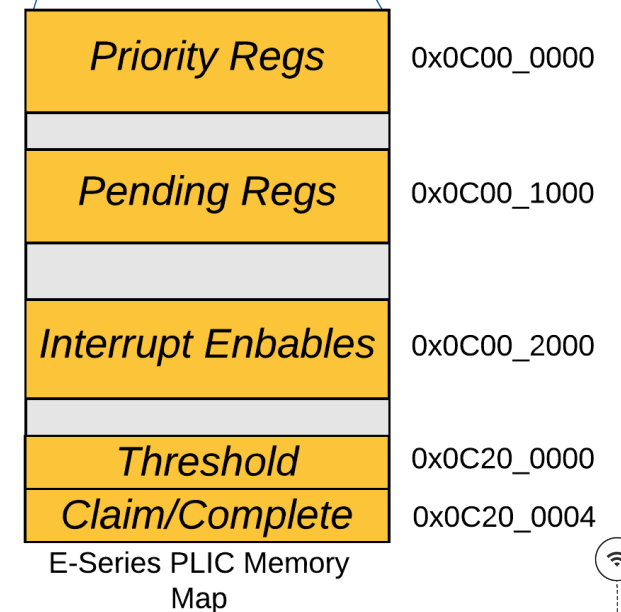
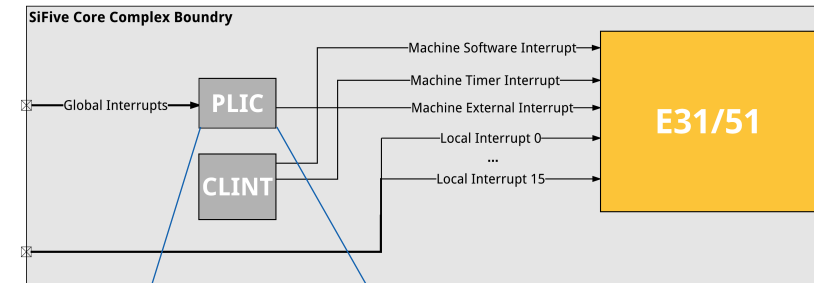
# Core Local Interruptor (CLINT)

- Used to generate Software and Timer Interrupts
  - Contains the RISC-V *mtime* and *mtimecmp* memory mapped CSRs
  - The *msip* memory mapped CSR can be used to generate Machine Software Interrupts
- Multi-Core scalable and consistent register map across SiFive RISC-V Core IP



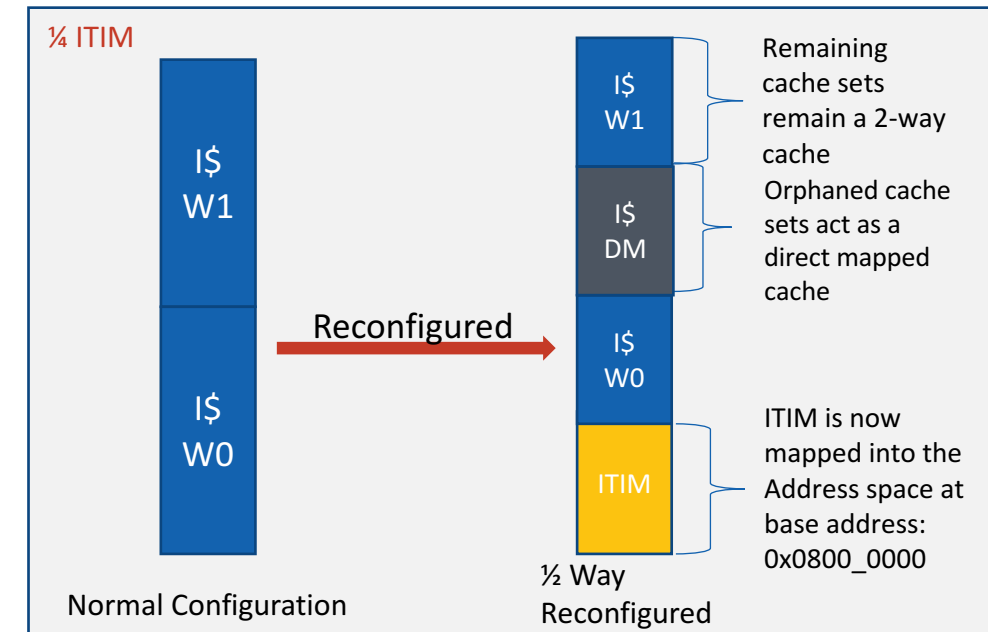
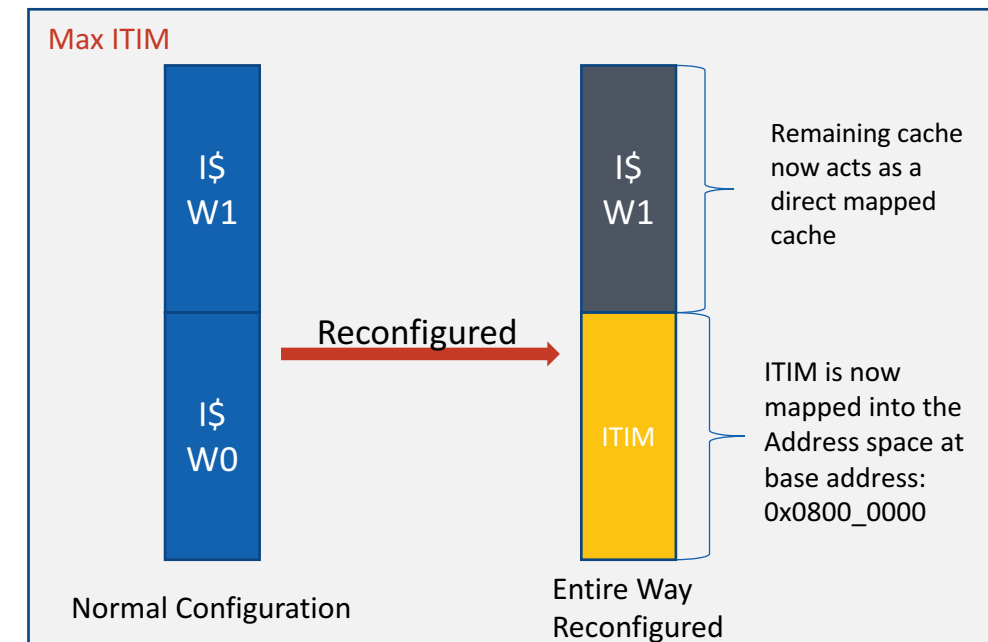
# Global Interrupts and the PLIC

- The PLIC handles the majority of the Core Complex's Interrupts
- Priority Registers
  - 4B registers containing 3-bit interrupt priority
  - 1 is lowest priority, 7 is the highest, 0 disables
- Pending and Enable Registers
  - Bit packed registers
- Threshold Register
  - Only interrupts with Priority > Threshold will trigger an interrupt
- Claim/Complete
  - Returns the ID of the highest pending interrupt
  - Interrupt completion is signaled to the PLIC by software writing the ID back to this register



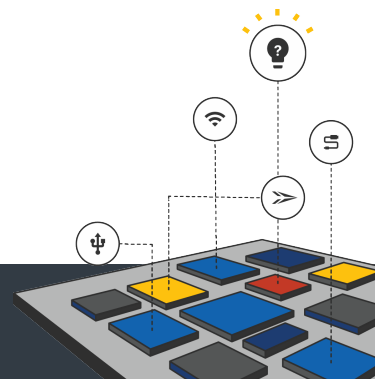
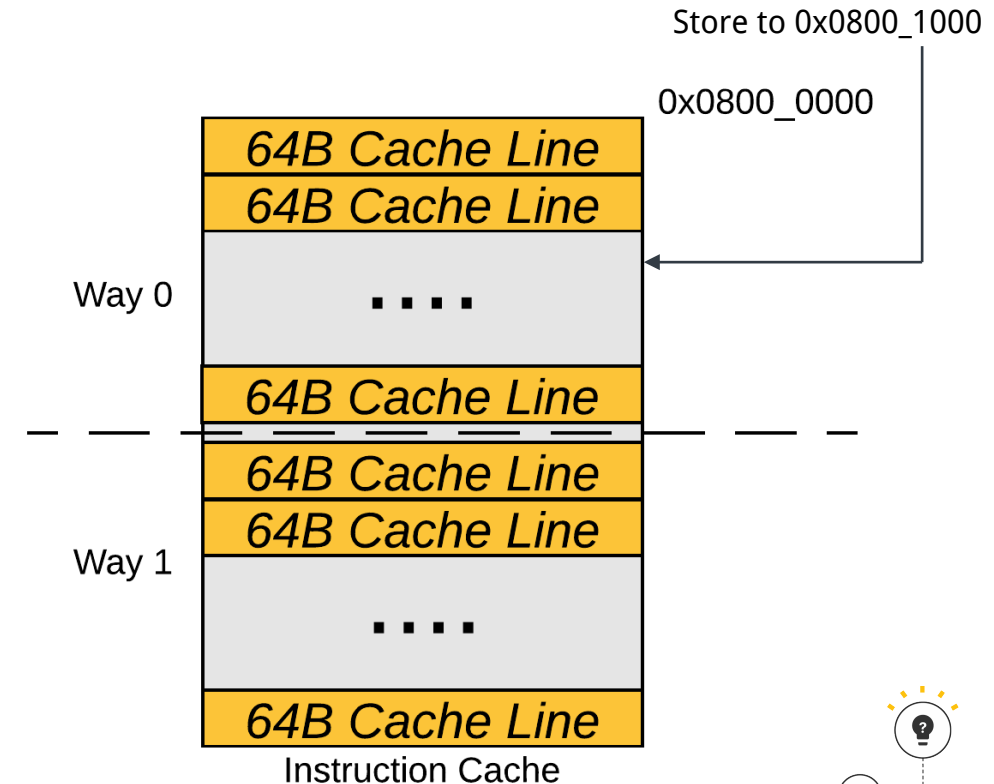
# I-Cache Re-Configurability

- By default Core Complexes boot with all L1 Instruction Cache ways enabled
- The Instruction Cache allows mapping of all *or part* of a Cache Way into the memory space dynamically
  - This memory space is called Instruction Tightly Integrated Memory (ITIM)
- Use ITIM for deterministic timing
- When re-configuring entire cache way
  - The other way acts as a direct mapped cache
- When re-configuring part of a cache way
  - Cache still behaves as a 2 way cache with the remaining matching sets in the cache
  - The orphaned cache set then act as a direct mapped cache



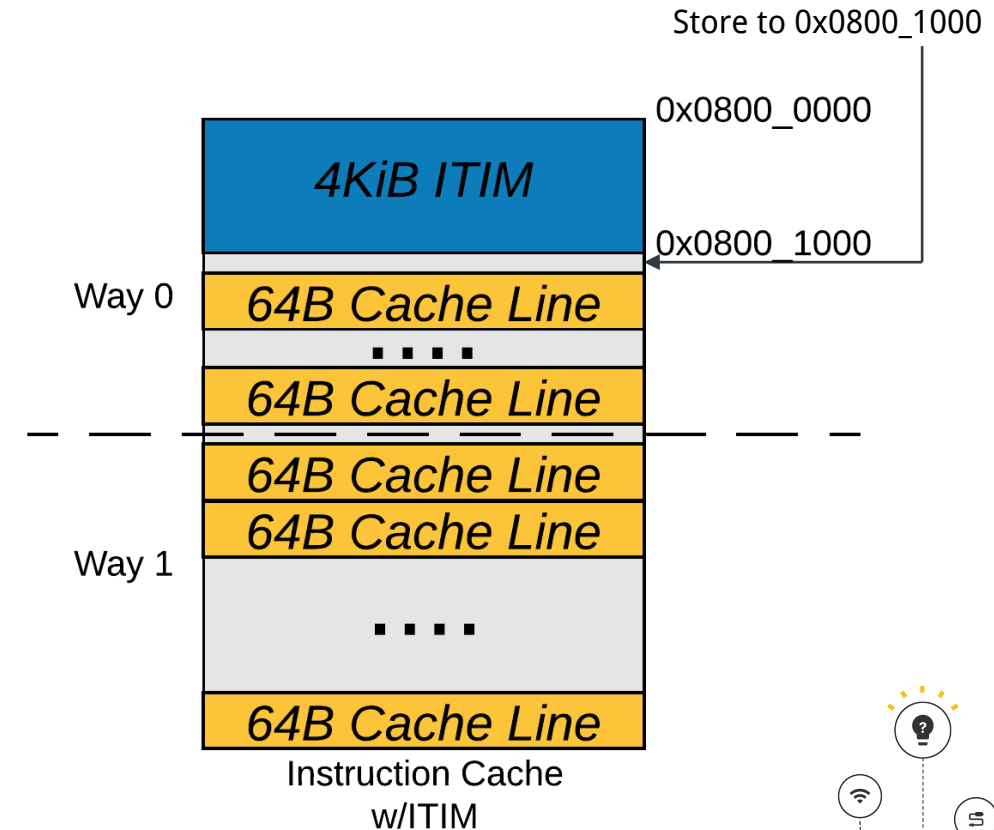
# Instruction Cache Re-Configurability How To

- Create an ITIM by storing to it
  - ITIM size is determined by the address written on allocation, rounded up to the nearest cache line (64B)
  - Software should not make assumptions about ITIM contents on creation
- Deallocate ITIM
  - Store 0x00 to the first byte after the ITIM region returns the entire ITIM space back to the Instruction Cache



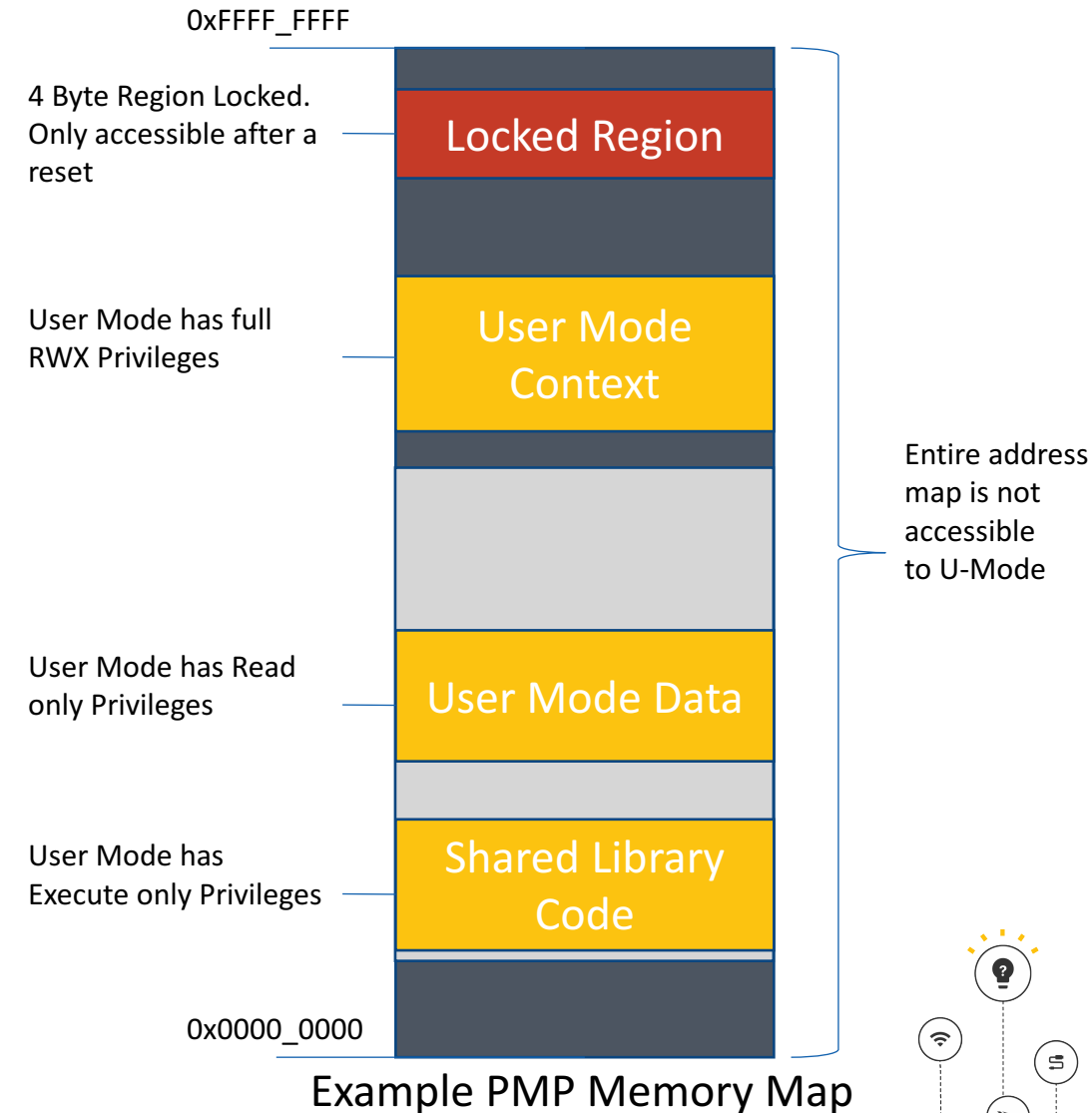
# Instruction Cache Re-Configurability How To

- Create an ITIM by storing to it
  - ITIM size is determined by the address written on allocation, rounded up to the nearest cache line (64B)
  - Software should not make assumptions about ITIM contents on creation
- Deallocate ITIM
  - Store 0x00 to the first byte after the ITIM region returns the entire ITIM space back to the Instruction Cache



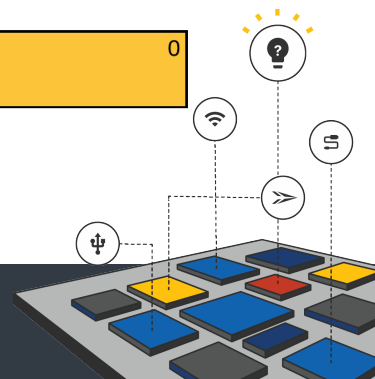
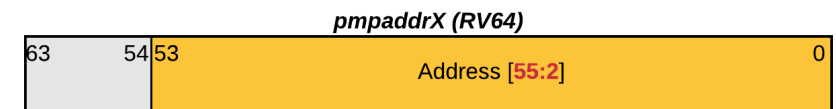
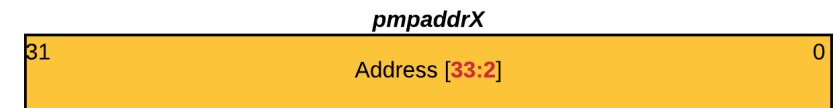
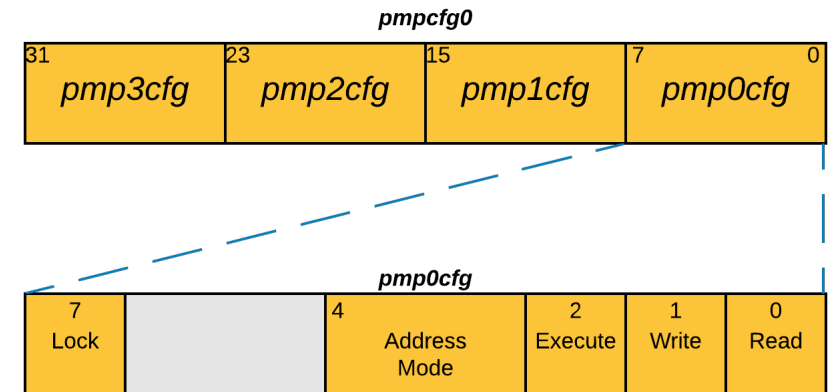
# Physical Memory Protection (PMP)

- Protects up to 8 regions of memory
- Enforces access permissions on U-Modes only
- Ability to Lock a region
  - A locked region enforces permissions on all accesses, including M-Mode
  - Only way to unlock a region is a Reset
- Minimum region size of 4 bytes
- Failed accesses generate a Load, Store, or Instruction access exception



# PMP Register Descriptions

- Each region has *pmpXcfg* field in a *pmpcfgX* register
  - Set R/W/X permissions
  - Region lock bit
  - Address matching scheme (next slide)
- *pmpaddrX* encodes:
  - Most significant 32 bits of a 34 bit physical address for RV32
  - Most significant 54 bits of a 56 bit physical address for RV64



# PMP Addressing Modes

- Top of Range (TOR)
  - Uses *pmpaddrX* and *pmpaddrX-1* to define a specific address range
  - If region 0 is set to TOR, address 0x00 is the lower bound
- Naturally Aligned Power of 2 Region
  - Uses first zero bit of *pmpaddrX* to encode region size (unary encoding)
    - Region Size =  $2^{(\text{zero bit position}+3)}$
  - Region Base = most significant bits above first 0
- Enables fast context switching
  - Single write to *pmpaddrX* is sufficient to map in a new Context

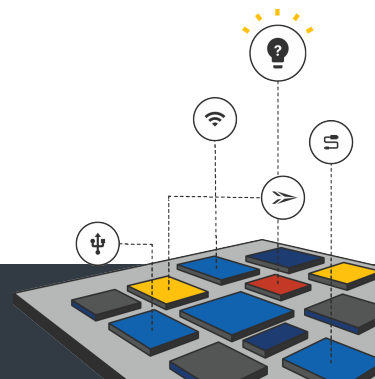
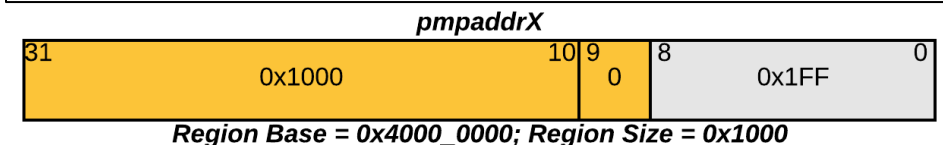
## PMP NAPOT Pseudocode

```
//encodes base and address into RISC-V PMP NAPOT Address format
get_pmp_napot_addr(base, size) {
    napot_size = ((size/2)-1)
    //napot_size = 0x7FFF

    pmp_addr= (base + napot_size)>>2
    //pmp_addr = 0x1000_01FF

    return pmp_addr;
}

...
region_base=0x40000000
region_size=0x1000
write_csr(pmpaddr1, get_pmp_napot_addr(region_base, region_size))
```



# Core Complex Debug

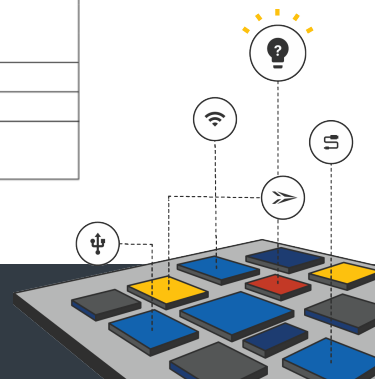
- 2 Hardware Breakpoints
  - Instruction and Data address matches are supported
- Hardware Performance Monitors
  - 2x 40bit event counting CSRs: *mhpmpcounter3(h)* and *mhpmpcounter4(h)*
  - 2x Event Selector CSRs: *mhpevent3* and *mhpevent4*
  - Counters will increment each time a selected event occurs
  - Can select multiple events per counter

SiFive E3x *mhpeventX*

Bit mask of counted events	7 Event Class
----------------------------	---------------

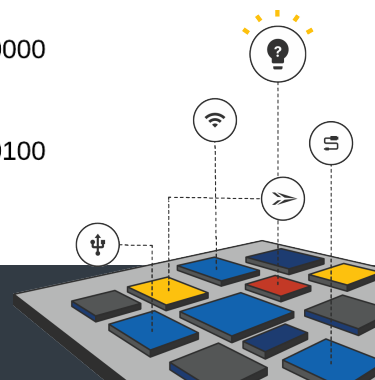
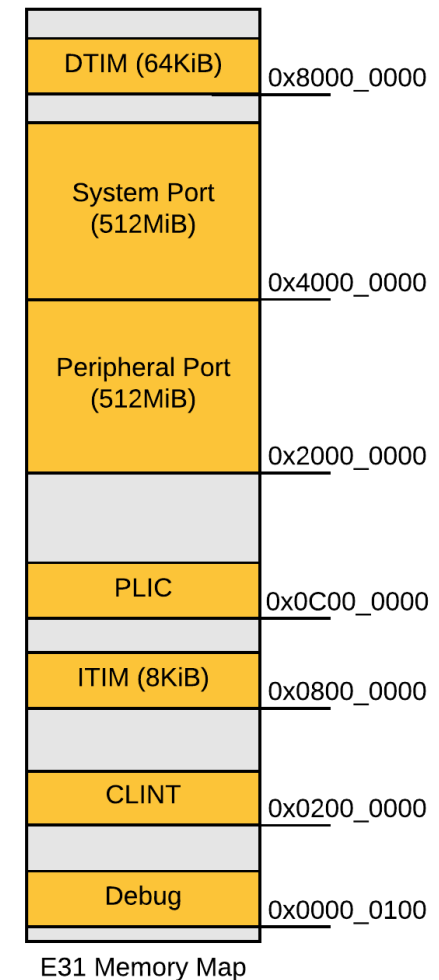
Machine Hardware Performance Monitor Event Register	
Instruction Commit Events, <i>mhpeventX</i> [7:0] = 0	
Bits	Meaning
8	Exception taken
9	Integer load instruction retired
10	Integer store instruction retired
11	Atomic memory operation retired
12	System instruction retired
13	Integer arithmetic instruction retired
14	Conditional branch retired
15	JAL instruction retired
16	JALR instruction retired
17	Integer multiplication instruction retired
18	Integer division instruction retired
Microarchitectural Events, <i>mhpeventX</i> [7:0] = 1	
Bits	Meaning
8	Load-use interlock
9	Long-latency interlock
10	CSR read interlock
11	Instruction cache/ITIM busy
12	Data cache/DTIM busy
13	Branch direction misprediction
14	Branch/jump target misprediction
15	Pipeline flush from CSR write
16	Pipeline flush from other event
17	Integer multiplication interlock
Memory System Events, <i>mhpeventX</i> [7:0] = 2	
Bits	Meaning
8	Instruction cache miss
9	Memory-mapped I/O access

Table 3.2: *mhpevent* Register Description



# Core Complex Memory Map and Interfaces

- Compatible memory maps across all SiFive RISC-V Core IP
- 3 Ports on each Core Complex
  - Peripheral Port (TL-UL)
    - Efficient low bandwidth bus, supports Atomics
  - System Port (TL-UH)
    - Higher bandwidth bus with burst support
  - Front Port (TL-UH)
    - Allows other masters to access Core Complex DTIM and ITIM address space
- AMBA bridge options
  - Can bridge any port to APB, AHB, or AXI





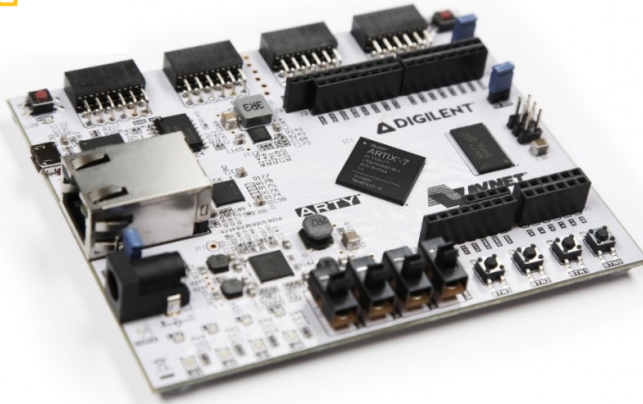
# Evaluate Today

[www.sifive.com](http://www.sifive.com)





# Evaluate – FPGA Bitstreams

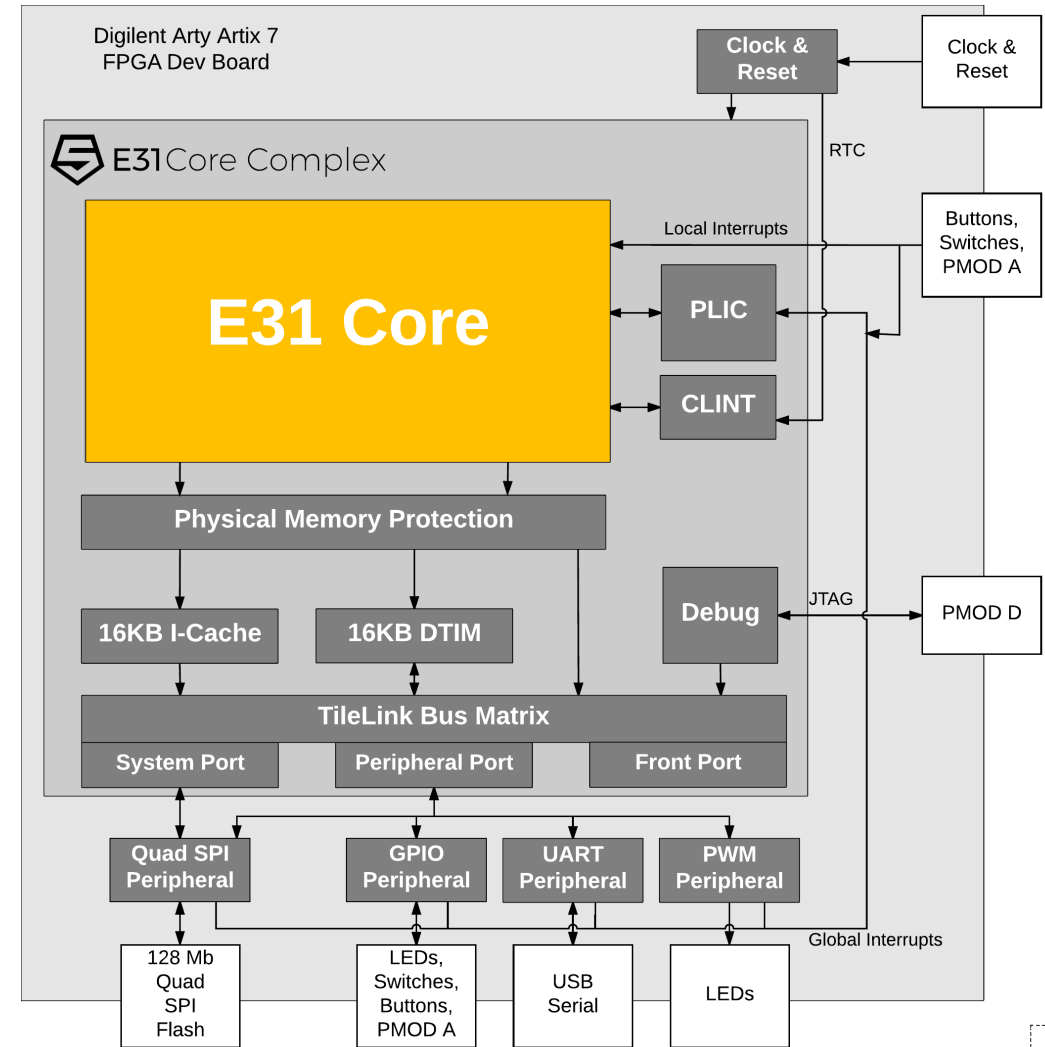


- Prebuilt FPGA Bitstreams
- Low cost \$99 FPGA Dev Kit
- Fast! ~65MHz



FreedomStudio

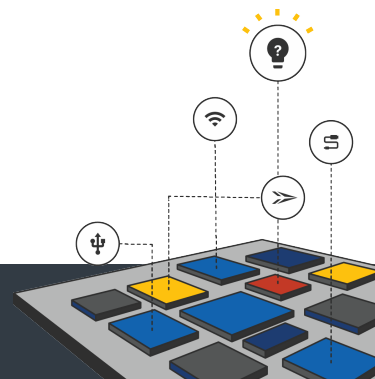
Evaluate code size, tools, CPU performance





# Evaluate – Verilog RTL

- Free, Instant Access
- Fully functional, synthesizable, Verilog RTL
  - Synthesize into your SOC
  - Simulate in your environment
  - Determine your PPA for your process node
- Evaluation License automatically sent electronically
- Countersign electronically
- Instant access in your Developer Dashboard





# Questions



# 3 Part Webinar Series

- RISC-V 101
  - The Fundamentals of RISC-V architecture
  - Recording: <https://info.sifive.com/risc-v-webinar>
- Introduction to SiFive RISC-V Core IP
  - Introduction to the SiFive E31 and E51 Core Complexes
- Getting Started with SiFive RISC-V Core IP
  - November 2017



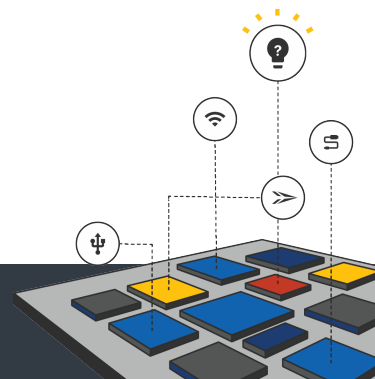
Study



Evaluate

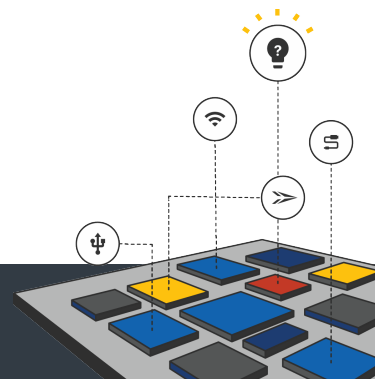


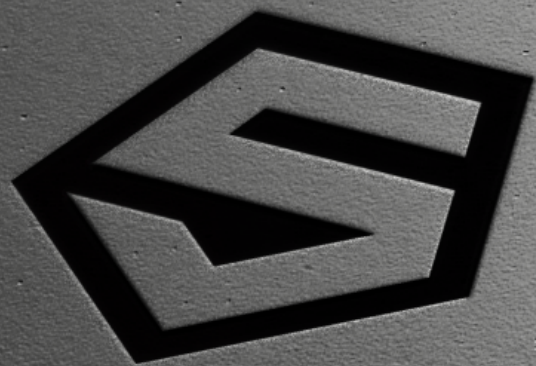
Buy



# Resources

- <https://riscv.org/>
  - RISC-V Specifications
  - Links to the RISC-V mailing lists
  - Workshop proceedings
- GitHub
  - <https://github.com/sifive/>
  - <https://github.com/>
- <https://www.sifive.com/>
  - RISC-V Core IP and Development Boards
  - RISC-V Tools
  - Forums





*SiFive*