



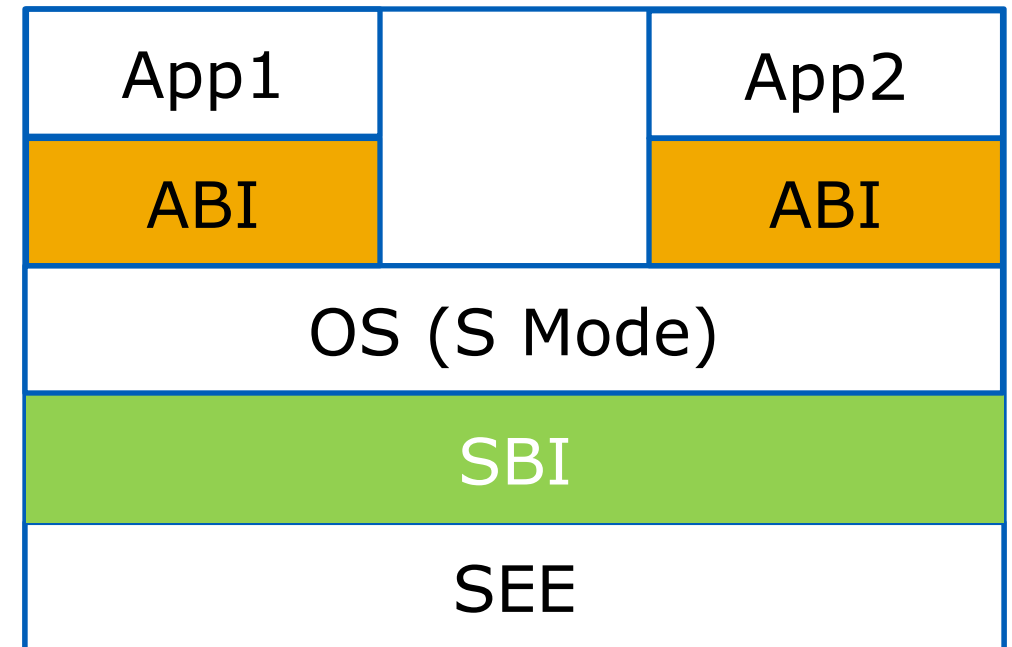
Western Digital®

Supervisor Binary Interface Extension proposal

Atish Patra

Supervisor Binary Interface (SBI)

- Clear separation between Supervisor & Machine Mode
- Helps to run single OS image across different SEE
- Currently provided by Berkeley Boot Loader (BBL)
- Calling convention
 - S mode traps into M mode using **ECALL**
 - Arguments via a0-a2
 - SBI call type via a7
 - a0 is clobbered register
- Documentation available at
 - <https://github.com/riscv/riscv-sbi-doc>



Current status

- Fixed
- No backward compatibility
- Not standardized
- No standard return/error value

Type	Function	Function ID
Timer	sbi_set_timer	0
Console	sbi_console_putchar	1
	sbi_console_getchar	2
IPI	sbi_clear_ipi	3
	sbi_send_ipi	4
Memory Model	sbi_remote_fence_i	5
	sbi_remote_sfence_vma	6
	sbi_remote_sfence_vma_as	7
	id	
Shutdown	sbi_shutdown	8

SBI Extension Proposal - Objective

- Make it extendable & backward compatible
- Shouldn't be a kitchen sink for all hardware abstractions
- Continue to remain minimalistic
- Make it modular to facilitate easy discussion/debate
 - Focus on Legacy (v0.1) & Base SBI
 - Power management/Vendor extension to follow
- Define standard error codes
 - A mapping function between SBI error code & OS specific error
- Ongoing discussion
 - V1
 - <http://lists.infradead.org/pipermail/linux-riscv/2018-October/001974.html>
 - V2
 - <http://lists.infradead.org/pipermail/linux-riscv/2018-November/002171.html>

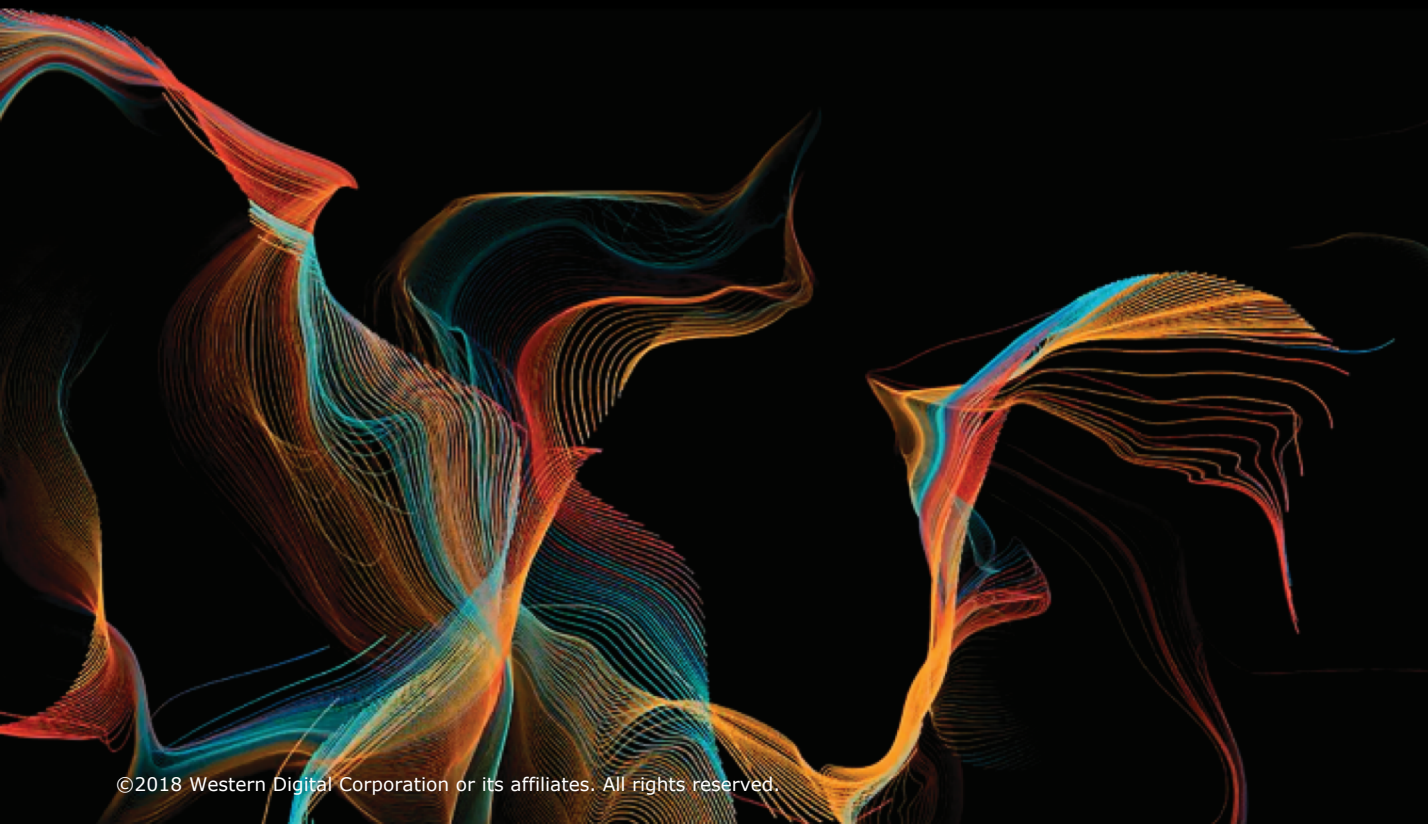
SBI extension proposal – Calling convention

- Function ID numbering scheme
 - Static binding
 - Bit[31:24] = Function Type Set
 - Number Bit[23:0] = Function Number within Function type
 - Dynamic binding
 - DT Based dynamic probing individual functions
- Return Values
 - Just single return value with value (≥ 0) & error (< 0)
 - Return a struct with a value & error (ABI says both a0 & a1 are return values)
 - Error strings in per hart local area?

Function Type	Function ID	Description
Base Function	0x0	Base APIs mandatory for any SBI version
HART PM APIs	0x1	Hart UP/Down/Suspend APIs for per-Hart power management
System PM APIs	0x2	System Shutdown/Reboot/Suspend for system-level power management
Vendor APIs	0xff	Vendor specific APIs

SBI extension proposal - Scope

SBI Extension	Function	Type
Base Function	sbi_get_version sbi_check_api or get_feature()	Mandatory
HART PM APIs	sbi_hart_up sbi_hart_suspend sbi_hart_down sbi_hart_get_state	Optional
System PM APIs	sbi_system_shutdown sbi_system_reset	Optional
Vendor APIs	TODO	Optional



OpenSBI project

- BBL/Coreboot provides separate SBI implementation
- More fragmentations going forward considering vendor specific usage
- Difficult to maintain & track the SBI changes
- OpenSBI to rescue!!
 - Create a open separate SBI implementation project
 - SBI implementation built as static library that any boot loader(both S mode & M mode) can use
 - Provides a reference implementation that potentially can replace BBL as well
 - This reference implementation can produce firmware binary that can be used as in M mode directly
- Recommended License ?
- Any feedback ?

Feedbacks

- `u32 sbi_check_api(unsigned long start_api_id, unsigned long count)`
vs
`u32 sbi_get_version(void)`
- Treat SBI function return & global error separately
 - No global state should be maintained
 - pass by reference
 - return a struct containing err & return value (My choice!!)
 - Should we change all the mandatory existing functions as well to reflect that ?
- May be rename it to a generic interface instead of Supervisor Binary Interface
- Anything else ?

Etrap call

- lui t0,#ecallTableBase #or auipc
- slli t1,a6,2 #3 for 64 bit
- add t0,t0,t1
- jr,nnn(t0) #lo bits of ecallTableBase, if not 4k aligned
- # called function does the mret