Sensonics Devices

SR2300
Processor Overview

SR2300 is a RISC V ISA implementation. It implements all I, M, C Instructions. The F (Floating Point) Instructions have not been implemented, instead a full suite of POSIT (32,2) and (16,2) instructions have been implemented including many that do not have equivalent F Instructions in RISC V ISA (e.g., circular, and hyperbolic functions and their inverses, square root, n^{th} root, exponent, logarithm, magnitude, phase etc). These instructions have been carved out in the custom instruction opcode space. Detailed error characterization can be found in www.sensonics.in

The F instruction opcodes have been left untouched so that, for example, one could exploit the software trap mechanism to implement all Float Instructions in firmware. Both A (atomic) and POSIT Instructions are optional and can be incorporated or removed almost mechanically. This flexibility follows directly from our design methodology.

There are, additionally, Big Endian Load / Store Instructions implemented to give seamless support to big and little endian data.

The severe opcode bandwidth limitations for 16b instructions prevents having RISC V ISA compliant compressed instruction variants of the POSIT Instructions.

The full Instruction Execute Engine (IEE) has been implemented and is available as Gate Level Verilog. One can for example load binaries into the Program Memory (PMEM) and execute programs. A user-friendly high-level programming interface for is under development.

Interrupts and Exception handler is not part of IEE and neither is Design For Test (DFT) or debug circuitry. These are SOC or microcontroller functionalities and therefore kept loosely specified and not implemented to allow maximum freedom for SOC integration and keep silicon footprint the lowest.

Secure Boot has also not yet been implemented but will soon be.

This is a ground up design and therefore quite easily reconfigurable at instruction level of granularity. This is intended. It allows SR2300 to be actually used as a Logical “Swiss-knife” as it were to quickly implement different categories of intellectual properties. The variations even inconsistent ones, perhaps even across vendors can be encapsulated under a single development framework by clever usage of the “mvendorid”, “marchid” and “mimpid” registers.