The SoCLib project is a library that contains all needed parts to create a fully working VirtualPrototype.

Most simulation models connect around an on-chip bus. It can be either a NoC or a simpler BUS or a crossbar. SoCLib components are mainly using the VCI on-chip-bus protocol. This makes the components easily interoperable. Moreover, VCI is simple enough to ease integration of new components, without forbidding translation of VCI to other protocols. See On-Chip-Bus/NoC Protocol adapters.

Available models can be split up in categories:

On-Chip-Bus/NoC implementations

Two main types of interconnects are available:

- Virtual interconnects, implementing a typical behavior but without any existing hardware equivalent. This abstraction from an actual implementation makes simulation faster, without making performance evaluation worse.
    - ♦ VciVgmn acts as a typical worm-hole NoC. (M->N communication, switched network)
    - ♦ VciVgsb acts as a classical BUS. (1 at-a-time communication, synchronous response)
- Actual interconnects, which can be implemented in RTL
    - ♦ BUSes
        - ◊ VciPibus : A VCI compliant PIBUS implementation.
        - ◊ VciAvalonBus : A VCI compliant AVALON bus interconnect.
    - ♦ Crossbars:
        - ◊ VciLocalCrossbar : A VCI compliant crossbar.
    - ♦ 2D-meshes:
        - ◊ VciDspin : A VCI compliant DSPIN micro-network.
        - ◊ VirtualDspinNetwork : A VCI compliant DSPIN micro-network with virtual channels.
        - ◊ VciAnoc : A VCI compliant ANOC micro-network.
    - ♦ Ring-based:
        - ◊ VciSimpleRingNetwork : A VCI compliant ring interconnect.
        - ◊ VciLocalRingNetwork : A VCI compliant ring interconnect. (targeting local interconnection, in a clusterized architecture)

On-Chip-Bus/NoC Protocol adapters

- VciPCI : A bridge to the PCI bus

OCB/NoC configuration utilities

- MappingTable : A tool to declare and list all memory segments used in a platform and to define the memory mapping.

Processor + cache

In SoCLib, processor+cache bundles are designed as two distinct entities. This has several advantages:

- Many CPU cores out there are just the same instruction set with variations on the implementation (pipeline stages, cache, coherency, coprocessors, ?)
- Modeling a cache alone is easier
- Modeling an ISS alone is easier
- Instrumentation tools can be factored-out (gdb, profiling, ?)

SoCLib provides different caches, with different features. All caches can be used with all ISSes, some features may just be unavailable in certain configurations (e.g. not all CPU support MMU-aware caches).

Cache models:

- VciXcacheWrapper : A generic, VCI compliant, cache controller for Iss2Api processors
- VciVcacheWrapper : A generic, VCI compliant, cache controller for Iss2Api processors supporting virtual memory mapping

ISS models:

- processors using the Iss2API
    - Mips32
    - Ppc405
    - Arm, with ARM-v6 instruction set (ARM11, Cortex-M0, Cortex-M1)
    - Sparc v8
    - Lattice Mico 32
    - NiosII
    - IssIss2 : This wrapper may be necessary to use the following IssApi-compliant ISSes (IssApi is deprecated. New ISSes should implement the Iss2Api)
        - MicroBlaze
        - ST231
        - TMS320C62

ISS instrumenting tools:

- Tools/Gdb Server : A GDB-server wrapper, for any ISS.
- Tools/Memory Checker : A wrapper providing valgrind-like features, for any ISS.

Memories

- VciRom? : A multi-segment embedded ROM controller
- VciHeterogeneousRom? : A multi-segment embedded ROM controller, with differentiated answers depending on initiator
- VciRam : A multi-segment embedded RAM controller
- VciSimpleRam : A multi-segment embedded RAM controller with parameterized latency

- VciLocks : A memory mapped locks controller (memory with implicit test-and-set)

Memory loading is done through Loader : A binary-file loader (ELF, COFF, plain)

IO Controllers

Character devices:

- VciMultiTty : A memory mapped multi-TTY controller
- VciLogConsole : A memory-mapped text log sink, for debugging purposes
- VciI2cInterface : An I2C bus controller.

Block devices (with DMA):

- VciFdAccess : A file system access controller
- VciBlockDevice : A block device controller

Other:

- VciFrameBuffer : A frame buffer for YUV or RVB image display.

Internal controllers

- VciTimer : A memory mapped timer controller
- VciIcu : A memory mapped interrupt controller
- Mailbox : A mailbox component allows several processors to communicate via an interrupt mechanism

- VciXicu : A memory mapped Hardware interrupt + Timer + IPI controller (all the 3 above controllers in 1)

- VciDma : A DMA engine

- VciMwmrController : A Mwmr channels controller
- VciMwmrControllerLf : Another Mwmr channels controller, with a lock-free software protocol

Dedicated coprocessors, not necessarily connected to On-Chip-Bus

- Tc4200 : A WiMAX LDPC decoder
- Tc4200_enc : a WiMAX LDPC encoder
- trx_ofdm : A FFT and IFFT coprocessor
- FIR128 : A 128-taps Finite Impulse Response filter
- Upsampling : An interpolation component
- Downsampling : A decimation component
- Synchronization : A synchronization component
- Mapping : A mapping component
- Demapping : A demapping component
- FHT : A Fast Hartley Transform component
- Tc1700 : A triple mode turbo decoder (3GPP-LTE, HSPA, WiMAX)

Debugging tools

Simulation controller utilities

- VciSimHelper : A memory-mapped simulation control tool, can call `sc_stop` or `exit` upon specific memory access

VCI debugging

- VciLogger : A VCI spy, useful for debugging network messages

Simulation MoC wrappers

Some components are just syntactical wrappers in order to mix CABA and TLM-DT simulation models:

- VciInitiatorTransactor : A VCI CABA Initiator compliant VCI TLM-DT Initiator.
- VciTargetTransactor : A VCI CABA Target compliant VCI TLM-DT Target.

Component factored-out code library

These are common parts of modules that have been factored-out to ease current and future components writing. Their usage is not mandatory for newly-written components, but is useful.

- VciTargetFsm : A generic CABA submodule for handling the VCI fsm part of a target components, so that you can focus on the functionality
- TtyWrapper : A simulator-side TTY abstraction tool, used by the VciMultiTty component
- ProcessWrapper : A simulator-side fork/exec abstraction tool, with process' stdin/stdout communication
- FbController : A simulator-side framebuffer abstraction tool