

1. Interconnects

1. Vgmn
 2. LocalCrossbar
- ## 2. VCI Initiators
1. Xcache
- ## 3. VCI Targets
1. MultiRam
 2. MultiTty

This page is a little excerpt of all available soclib components, please see documentation on [?SoCLib's main site](#) for a complete list.

Interconnects

Vgmn

- functionality : a generic VCI compliant micro-network
- Arguments:
 - ◆ instance name
 - ◆ min_latency (one-way)
 - ◆ fifo_size
- Example:

```
my_vgmn = arch.create('caba:vci_vgmn', "my_vgmn", min_latency = 10, fifo_depth = 8)
```

- Defined ports:
 - ◆ to_initiator.new() and to_target.new() : local ports, allocated on demand

LocalCrossbar

- functionality : a VCI compliant crossbar interconnect
- Arguments:
 - ◆ instance name
- Example:

```
my_lc = arch.create('caba:vci_local_crossbar', "lc0")
```

- Defined ports:
 - ◆ to_initiator.new() and to_target.new() : local ports, allocated on demand

VCI Initiators

Xcache

- functionality: a n-associative cache controller (separated instruction & data cache) wrapping a given ISS, it can be considered as a CPU with an embedded cache.
- Arguments:
 - ◆ instance name
 - ◆ dcache_sets : number of lines in data cache
 - ◆ dcache_words : number of words per line in data cache
 - ◆ dcache_ways : associativity of the data cache
 - ◆ icache_sets : number of lines in instruction cache

- ◆ `icache_words`: number of words per line in instruction cache
- ◆ `icache_ways`: associativity of the instruction cache
- ◆ `iss_t`: type of ISS to be wrapped in the cache, at least "common:mips32el", "common:mips32eb" and "common:ppc405" are supported
- ◆ `ident`: cpu number
- Example:

```
my_cpu0 = arch.create('caba:vci_xcache_wrapper',
                      'cpu0',
                      ident = 0,
                      icache_ways = 1,
                      icache_sets = 32,
                      icache_words = 8,
                      dcache_ways = 1,
                      dcache_sets = 32,
                      dcache_words = 8,
                      iss_t = "common:mips32el",
                      )
```

- Defined ports:
 - ◆ `irq[n]`: to the CPU interrupts
 - ◆ `vci`: to the VCI micro-network

VCI Targets

MultiRam

- !Mandatory arguments:
 - ◆ `name`
- Example:

```
my_ram = arch.create('caba:vci_ram', "my_ram")
```

- Defined ports:
 - ◆ `vci`: to the micro-network
- Adding a segment:
 - ◆ Arguments: name, base address, size, cacheability

```
my_ram.addSegment( 'boot', 0xbfc00000, 0x1000, true )
```

MultiTty

- functionality: a TTY controler
- Mandatory arguments:
 - ◆ `instance name`
 - ◆ `a list of names`
- Example:

```
my_tty = arch.create('caba:vci_multi_tty', "my_tty_controler", names = ["TTY0", "TTY1", "
```

- Defined ports:
 - ◆ `vci`: to the micro-network
 - ◆ `irq[n]`: interrupt line ($0 \leq n < \text{nb of ttys}$)