1. A) Interconnects
    1. A.1) Vgmn
    2. A.2) LocalCrossbar
2. B) VCI Initiators
    1. B.1) Xcache
3. C) Processors
    1. C.1) Mips
4. D) VCI Targets
    1. D.1) MultiRam
    2. D.2) MultiTty

A) Interconnects

A.1) 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_size = 8)
```

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

A.2) 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

B) VCI Initiators

B.1) Xcache

- functionality: a direct mapping cache controler (separated instruction & data cache)
- Arguments:
    - ♦ `instance name`
    - ♦ `dcache_lines` : number of lines in data cache
    - ♦ `dcache_words` : number of words per line in data cache
    - ♦ `icache_lines` : number of lines in instruction cache
    - ♦ `icache_words` : number of words per line in instruction cache
- Example:

```
my_cache = arch.create('caba:vci_vgmn', "my_cache",
                       dcache_lines = 32,
                       dcache_words = 8,
                       icache_lines = 32,
                       icache_words = 8 )
```

- Defined ports:
  - `dcache`: to the CPU
  - `icache`: to the CPU
  - `vci`: to the VCI micro-network

C) Processors

C.1) Mips

This component is special as it is in fact a wrapped ISS.

- Functionality : a MIPS R3000 micro-processor
- Arguments:
  - `name`
  - `ident`: cpu id
- Example:

```
my_proc = arch.create('caba:iss_wrapper', "my_proc", iss_t = 'common:mipsel', ident = 0 )
```

- Defined ports:
  - `dcache`: to the cache's dcache port
  - `icache`: to the cache's icache port
  - `irq[n]`: interrupt line (0 <= n < 6)

D) VCI Targets

D.1) 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, 0x100, true )
```

D.2) 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", ["TTY0", "TTY1", "TT2"])
```

- Defined ports:
  - `vci`: to the micro-network
  - `irq[n]`: interrupt line (0 <= n < nb of ttys)