



```
#!/usr/bin/env python

import dsx
import soclib

def _cluster(arch, no,
             ncpu, nram,
             icache_lines, icache_words,
             dcache_lines, dcache_words,
             with_gdb = False ):
    crossbar = arch.create('caba:vci_local_crossbar', 'lc%d'%no)

    for i in range(ncpu):
        cn = arch.cpu_num
        arch.cpu_num += 1
        if with_gdb:
            proc = dict( iss_t='common:gdb_iss', gdb_iss_t = 'common:mips32el' )
        else:
            proc = dict( iss_t='common:mips32el' )
        cpu = arch.create('caba:vci_xcache_wrapper', 'cpu%d_%d'%(no,i),
                        ident = cn,
                        icache_sets = icache_lines,
                        icache_words = icache_words,
                        icache_ways = 1,
                        dcache_sets = dcache_lines,
                        dcache_words = dcache_words,
                        dcache_ways = 1,
                        **proc )

        crossbar.to_initiator.new() // cpu.vci

    for i in range(nram):
        ram = arch.create('caba:vci_ram', 'ram%d_%d'%(no,i))
        base = 0x10000000*(1+no)+0x1000000*i
        ram.addSegment('cram%d_%d'%(no,i), base, 0x100000, True)
        ram.addSegment('uram%d_%d'%(no,i), base + 0x400000, 0x100000, False)
        ram.vci // crossbar.to_target.new()
        if i == 0 and no == 0:
            ram.addSegment('boot', 0xbfc00000, 0x800, True)
            ram.addSegment('excep', 0x80000080, 0x800, True)

    return crossbar

def ClusteredNoirqMulti(
    cpu_count = [1],
    ram_count = [1],
    min_latency = 10,
    with_gdb = False,
    icache_lines = 32,
    icache_words = 8,
    dcache_lines = 32,
    dcache_words = 8 ):
    arch = soclib.Architecture(
        cell_size = 4,
        plen_size = 9,
        addr_size = 32,
        rerror_size = 1,
        clen_size = 1,
        rflag_size = 1,
        srcid_size = 8,
        pktid_size = 1,
        trdid_size = 1,
        wrplen_size = 1
    )

    arch.cpu_num = 0
```

```

arch.create('common:mapping_table',
            'mapping_table',
            addr_bits = [5,3],
            srcid_bits = [4,4],
            cacheability_mask = 0xc00000)
arch.create('common:loader', 'loader')

vgmn = arch.create('caba:vci_vgmn', 'vgmn0', min_latency = min_latency, fifo_depth = 8)

for no, (ncpu, nram) in enumerate(zip(cpu_count, ram_count)):
    lc = _cluster(
        arch, no,
        with_gdb = with_gdb,
        ncpu = ncpu, nram = nram,
        icache_lines = icache_lines, icache_words = icache_words,
        dcache_lines = dcache_lines, dcache_words = dcache_words)
    vgm_n.to_initiator.new() // lc.initiator_to_up
    vgm_n.to_target.new() // lc.target_to_up
    # si cluster == 0
    #   ajouter tg a l'adresse 0x16400000
    # si cluster == 3
    #   ajouter ramdac a l'adresse 0x46400000

tty = arch.create('caba:vci_multi_tty', 'tty', names = ['tty0'])
tty.addSegment('tty', 0x95400000, 0x10, False)
tty.vci // lc.to_target.new()

return arch

```