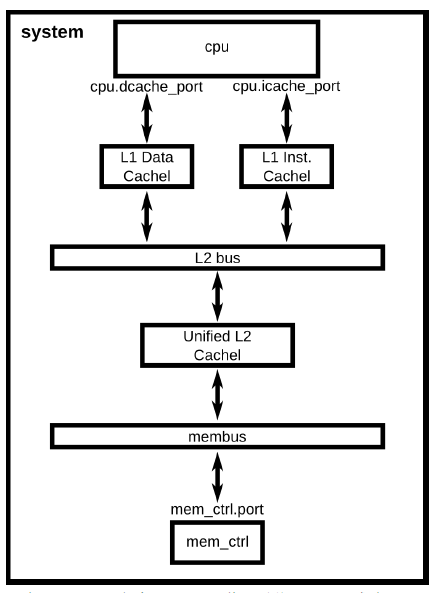
实验三 构建L2cache

**一、实验目的**

1. 掌握构建L2cache的方法。

**二、实验内容**

1. L2cache的逻辑系统架构如图所示：



1. 为了构建三级cache，我们通过修改gem5目录下configs/learning\_gem5/part1/two\_level.py文件能够获得三级cache配置文件。具体代码如下：

# -\*- coding: utf-8 -\*-

# Copyright (c) 2015 Jason Power

# All rights reserved.

#

# Redistribution and use in source and binary forms, with or without

# modification, are permitted provided that the following conditions are

# met: redistributions of source code must retain the above copyright

# notice, this list of conditions and the following disclaimer;

# redistributions in binary form must reproduce the above copyright

# notice, this list of conditions and the following disclaimer in the

# documentation and/or other materials provided with the distribution;

# neither the name of the copyright holders nor the names of its

# contributors may be used to endorse or promote products derived from

# this software without specific prior written permission.

#

# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

# A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

# OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

# LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

# DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

# THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

""" This file creates a single CPU and a two-level cache system.

This script takes a single parameter which specifies a binary to execute.

If none is provided it executes 'hello' by default (mostly used for testing)

See Part 1, Chapter 3: Adding cache to the configuration script in the

learning\_gem5 book for more information about this script.

This file exports options for the L1 I/D and L2 cache sizes.

IMPORTANT: If you modify this file, it's likely that the Learning gem5 book

also needs to be updated. For now, email Jason <power.jg@gmail.com>

"""

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import absolute\_import

# import the m5 (gem5) library created when gem5 is built

import m5

# import all of the SimObjects

from m5.objects import \*

# Add the common scripts to our path

m5.util.addToPath('../../')

# import the caches which we made

from caches import \*

# import the SimpleOpts module

from common import SimpleOpts

# Set the usage message to display

SimpleOpts.set\_usage("usage: %prog [options] <binary to execute>")

# Finalize the arguments and grab the opts so we can pass it on to our objects

(opts, args) = SimpleOpts.parse\_args()

# get ISA for the default binary to run. This is mostly for simple testing

isa = str(m5.defines.buildEnv['TARGET\_ISA']).lower()

# Default to running 'hello', use the compiled ISA to find the binary

# grab the specific path to the binary

thispath = os.path.dirname(os.path.realpath(\_\_file\_\_))

binary = os.path.join(thispath, '../../../',

'tests/test-progs/hello/bin/', isa, 'linux/hello')

# Check if there was a binary passed in via the command line and error if

# there are too many arguments

if len(args) == 1:

binary = args[0]

elif len(args) > 1:

SimpleOpts.print\_help()

m5.fatal("Expected a binary to execute as positional argument")

# create the system we are going to simulate

system = System()

# Set the clock fequency of the system (and all of its children)

system.clk\_domain = SrcClockDomain()

system.clk\_domain.clock = '3GHz'

system.clk\_domain.voltage\_domain = VoltageDomain()

# Set up the system

system.mem\_mode = 'timing' # Use timing accesses

system.mem\_ranges = [AddrRange('2048MB')] # Create an address range

#----------------------------Core1-------------------------------

# Create a simple CPU

system.cpu = TimingSimpleCPU()

# Create an L1 instruction and data cache

system.cpu.icache = L1ICache(opts)

system.cpu.dcache = L1DCache(opts)

# Connect the instruction and data caches to the CPU

system.cpu.icache.connectCPU(system.cpu)

system.cpu.dcache.connectCPU(system.cpu)

# Create a memory bus, a coherent crossbar, in this case

system.l2bus = L2XBar()

# Hook the CPU ports up to the l2bus

system.cpu.icache.connectBus(system.l2bus)

system.cpu.dcache.connectBus(system.l2bus)

# Create an L2 cache and connect it to the l2bus

system.l2cache = L2Cache(opts)

system.l2cache.connectCPUSideBus(system.l2bus)

#----------------------------Core1end-------------------------------

#----------------------------Core2-----------------------------

# Create a simple CPU

system.cpu1 = TimingSimpleCPU()

# Create an L1 instruction and data cache

system.cpu1.icache1 = L1ICache(opts)

system.cpu1.dcache1 = L1DCache(opts)

# Connect the instruction and data caches to the CPU

system.cpu1.icache1.connectCPU(system.cpu1)

system.cpu1.dcache1.connectCPU(system.cpu1)

# Create a memory bus, a coherent crossbar, in this case

system.l2bus1 = L2XBar()

# Hook the CPU ports up to the l2bus

system.cpu1.icache1.connectBus(system.l2bus1)

system.cpu1.dcache1.connectBus(system.l2bus1)

# Create an L2 cache and connect it to the l2bus

system.l2cache1 = L2Cache(opts)

system.l2cache1.connectCPUSideBus(system.l2bus1)

#----------------------------Core2end-----------------------------

# Create a L3 bus

system.l3bus = L2XBar()

# Connect the L2 cache to the l3bus

system.l2cache.connectMemSideBus(system.l3bus)

system.l2cache1.connectMemSideBus(system.l3bus)

# Create an L3 cache and connect it to the membus

system.l3cache = L3Cache(opts)

system.l3cache.connectCPUSideBus(system.l3bus)

# Create a memory bus

system.membus = SystemXBar()

# Connect the L3 cache to the membus

system.l3cache.connectMemSideBus(system.membus)

# create the interrupt controller for the CPU

system.cpu.createInterruptController()

system.cpu1.createInterruptController()

# For x86 only, make sure the interrupts are connected to the memory

# Note: these are directly connected to the memory bus and are not cached

if m5.defines.buildEnv['TARGET\_ISA'] == "x86":

system.cpu.interrupts[0].pio = system.membus.master

system.cpu.interrupts[0].int\_master = system.membus.slave

system.cpu.interrupts[0].int\_slave = system.membus.master

system.cpu1.interrupts[0].pio = system.membus.master

system.cpu1.interrupts[0].int\_master = system.membus.slave

system.cpu1.interrupts[0].int\_slave = system.membus.master

# Connect the system up to the membus

system.system\_port = system.membus.slave

# Create a DDR3 memory controller

system.mem\_ctrl = DDR3\_1600\_8x8()

system.mem\_ctrl.range = system.mem\_ranges[0]

system.mem\_ctrl.port = system.membus.master

# Create a process for a simple "Hello World" application

process = Process()

# Set the command

# cmd is a list which begins with the executable (like argv)

process.cmd = [binary]

# Set the cpu to use the process as its workload and create thread contexts

system.cpu.workload = process

system.cpu.createThreads()

system.cpu1.workload = process

system.cpu1.createThreads()

# set up the root SimObject and start the simulation

root = Root(full\_system = False, system = system)

# instantiate all of the objects we've created above

m5.instantiate()

print("Beginning simulation!")

exit\_event = m5.simulate()

print('Exiting @ tick %i because %s' % (m5.curTick(), exit\_event.getCause()))

**三、调试和心得体会**

通过这次实验，我学会了三级cache的构建。