Cache Simulation

ECEN 4593

12/16/2013

Brian Campuzano

Ryan Riley

Source Code

~~~~~~~~~~

include

include

include

include

include

include

include

include

include

include

include "StdTypes.h"

include "MemoryModule.h"
~~~~~~~~~~

define procBusWidth 4

using namespace std; using namespace Valhalla;

int main(int argc, char ** argv) {

//creating a map to store the addresses of the params std::map params; //setting default values

/* *\brief These are the default values for the L1 Cache* / int L1_BLOCK_SIZE = 32; int L1_MEMORY_SIZE = 8192; int L1_ASSOCIATIVITY = 1; int L1_HIT_PENALTY = 1; int L1_MISS_PENALTY = 1;

/* *\brief These are the default values for the L2 Cache* / int L2_BLOCK_SIZE = 64; int L2_MEMORY_SIZE = 65536; int L2_ASSOCIATIVITY = 1; int L2_HIT_PENALTY = 4; int L2_MISS_PENALTY = 6; int L2_TRANSFER_TIME = 6; int L2_TRANSFER_WIDTH = 16;

/* *\brief These are the default values for Main Memory* / int MAIN_MEMORY_SEND_ADDRESS_TIME = 10; int MAIN_MEMORY_READY_TIME = 50; int MAIN_MEMORY_CHUNK_SEND_TIME = 20; int MAIN_MEMORY_ADDRESS_WIDTH = 16;

//initializing params map params["L1_BLOCK_SIZE"] = &L1_BLOCK_SIZE; params["L1_MEMORY_SIZE"] = &L1_MEMORY_SIZE; params["L1_ASSOCIATIVITY"] = &L1_ASSOCIATIVITY; params["L1_HIT_PENALTY"] = &L1_HIT_PENALTY; params["L1_MISS_PENALTY"] = &L1_MISS_PENALTY; params["L2_BLOCK_SIZE"] = &L2_BLOCK_SIZE; params["L2_MEMORY_SIZE"] = &L2_MEMORY_SIZE; params["L2_ASSOCIATIVITY"] = &L2_ASSOCIATIVITY; params["L2_HIT_PENALTY"] = &L2_HIT_PENALTY; params["L2_MISS_PENALTY"] = &L2_MISS_PENALTY; params["L2_TRANSFER_TIME"] = &L2_TRANSFER_TIME; params["L2_TRANSFER_WIDTH"] = &L2_TRANSFER_WIDTH; params["MAIN_MEMORY_SEND_ADDRESS_TIME"] = &MAIN_MEMORY_SEND_ADDRESS_TIME; params["MAIN_MEMORY_READY_TIME"] = &MAIN_MEMORY_READY_TIME; params["MAIN_MEMORY_CHUNK_SEND_TIME"] = &MAIN_MEMORY_CHUNK_SEND_TIME; params["MAIN_MEMORY_ADDRESS_WIDTH"] = &MAIN_MEMORY_ADDRESS_WIDTH;

if(argc > 1){

```
//open and read config file and create a report file
ifstream config(argv[1]);
string line;
if (config.is_open())
  {
cout << "Config filename: " << argv[1] << endl;
int param_count = 0;
while (getline(config,line) )
  {
    string buffer;
    stringstream ss(line);
    vector<string> tokens;
    while (ss >> buffer) tokens.push_back(buffer);
    *params[tokens[0]] = atoi(tokens[1].c_str());
    cout << "Set parameter " << tokens[0] << " to " << tokens[1] << endl;
    param_count++;
  }
config.close();
cout << "Set " << param_count << " parameter(s) from config file " << endl;
  }
if ( L1_ASSOCIATIVITY == -1) L1_ASSOCIATIVITY = L1_MEMORY_SIZE / L1_BLOCK_SIZE;
if ( L2_ASSOCIATIVITY == -1) L2_ASSOCIATIVITY = L2_MEMORY_SIZE / L2_BLOCK_SIZE;
```

}

//Variables for mem operations char op; uint64 address; uint32 byteSize; uint64 time = 0; uint64 refNum = 0; uint64 iCount = 0; uint64 wCount = 0; uint64 rCount = 0; uint32 blockSize = 4; uint64 rTime = 0; uint64 iTime = 0; uint64 wTime = 0; cout << "Creating Main Memory." << endl; MemoryModule * mainMemory = new MemoryModule();

mainMemory->printMemoryModuleSetup(); cout << "Creating L2 Cache." << endl;

MemoryModule * l2Cache = new MemoryModule("L2", L2_BLOCK_SIZE, L2_MEMORY_SIZE, L2_ASSOCIATIVITY, L2_HIT_PENALTY, L2_MISS_PENALTY, MAIN_MEMORY_SEND_ADDRESS_TIME + MAIN_MEMORY_READY_TIME, MAIN_MEMORY_CHUNK_SEND_TIME, MAIN_MEMORY_ADDRESS_WIDTH, mainMemory, "Memory"); l2Cache-

>printMemoryModuleSetup(); cout << "Creating L1 Data Cache." << endl;

MemoryModule * l1DataCache = new MemoryModule("L1Data", L1_BLOCK_SIZE, L1_MEMORY_SIZE, L1_ASSOCIATIVITY, L1_HIT_PENALTY, L1_MISS_PENALTY, 0, L2_TRANSFER_TIME, L2_TRANSFER_WIDTH, l2Cache, "L2");

l1DataCache->printMemoryModuleSetup(); cout << "Creating L1 Instruction Cache." << endl;

MemoryModule * l1InstCache = new MemoryModule("L1Inst", L1_BLOCK_SIZE, L1_MEMORY_SIZE, L1_ASSOCIATIVITY, L1_HIT_PENALTY, L1_MISS_PENALTY, 0, L2_TRANSFER_TIME, L2_TRANSFER_WIDTH, l2Cache, "L2");

l1InstCache->printMemorySetup(); cout << "After initialization" << endl;

while (scanf("%c %llx %ld\n",&op,&address,&byteSize) == 3) { switch(op) { case 'I': iCount++; break; case 'R': rCount++; break; case 'W': wCount++; break; default: continue; } uint64 remainder = address % blockSize; if(remainder != 0) { address -= remainder; byteSize += remainder; } int bytesToFetch = byteSize;

```
  cout << "--------------------------------------------------------------------------
  cout << "Ref " << refNum;
  cout << ": Addr = " << hex << address;
  cout << ", Type = " << op;
  cout << ", BSize = " << byteSize << endl;
  while (bytesToFetch > 0)
{
  bytesToFetch -= procBusWidth;

  uint64 tempTime;
  switch(op)
    {
    case 'I':
      //Intruction fetch
      tempTime = l1InstCache->checkMemoryEntry(CACHE_READ, address, procBusWidth);
      time += tempTime;
          iTime += tempTime;

      break;
    case 'R':
      tempTime = l1DataCache->checkMemoryEntry(CACHE_READ, address, procBusWidth);
      time += tempTime;
          rTime += tempTime;
      break;
    case 'W':
      tempTime = l1DataCache->checkMemoryEntry(CACHE_WRITE, address, procBusWidth);
      time += tempTime;
          wTime += tempTime;
      break;
    default:
      continue;
    }

  address += procBusWidth;

}
  cout << "Simulated time = " << dec << time << endl;
  refNum++;
}
```

/* uint64 time = l1DataCache->checkMemoryEntry(CACHE_WRITE, 65536, 32); cout << "Time for memory lookup 1: " << time << endl; time = l1DataCache->checkMemoryEntry(CACHE_WRITE, 4096, 32); cout << "Time for memory lookup 2: " << time << endl;

```
time = l1DataCache->checkMemoryEntry(CACHE_READ, 8192, 32);
cout << "Time for memory lookup 3: " << time << endl;
time = l1DataCache->checkMemoryEntry(CACHE_READ, 256, 32);
cout << "Time for memory lookup 4: " << time << endl;

time = l1DataCache->checkMemoryEntry(CACHE_READ, 512, 32);
```

```
cout << "Time for memory lookup 5: " << time << endl;
```

//cout << "L1 instruction cache" << endl; l1InstCache->printMemoryEntries(); cout << "L1 data cache" << endl; l1DataCache->printMemoryEntries(); cout << "L2 cache" << endl; l2Cache->printMemoryEntries(); */

cout << "Test Complete." << endl;

if(argc == 3){ int L1SizeCost = ((L1_MEMORY_SIZE)/4096)*100; int L2SizeCost = ((L2_MEMORY_SIZE)/65536)*50; int L1AssociativityCost = (log2(L1_ASSOCIATIVITY) * (L1_MEMORY_SIZE/4096)) * 100; int L2AssociativityCost = (log2(L2_ASSOCIATIVITY) * (L2_MEMORY_SIZE/65536)) * 50; int MemReadyCost = ((50 / MAIN_MEMORY_READY_TIME) - 1) * 200; int MemChunkSizeCost = (log2(MAIN_MEMORY_ADDRESS_WIDTH) - log2(16)) * 100; int baseMemCost = 75; int l1iCost = L1SizeCost + L1AssociativityCost; int l1dCost = L1SizeCost + L1AssociativityCost; int l2Cost = L2SizeCost + L2AssociativityCost; int memCost = baseMemCost + MemReadyCost + MemChunkSizeCost; int totalCost = memCost + l2Cost + l1iCost + l1dCost;

```
std::stringstream str;

ofstream outfile;
std::string s = argv[1];
cout << s << endl;
std::string delimiter = "/";

std::string token;
token = s.substr(s.find(delimiter)+1, std::string::npos);

str << argv[2] <<"."<< token.c_str();

outfile.open(str.str().c_str());

outfile << "-------------------------------------------------------------------------
outfile << "\t" << str.str().c_str() << "\t Simulation Results\n";
outfile << "-------------------------------------------------------------------------

outfile << "\t Memory system: \n";
outfile <<"\t      Dcache size = " <<  L1_MEMORY_SIZE << " : ways = " << L1_ASSOCIATI
outfile <<"\t      Icache size = " <<  L1_MEMORY_SIZE << " : ways = " << L1_ASSOCIATI
outfile <<"\t      L2-cache size = " <<  L2_MEMORY_SIZE << " : ways = " << L2_ASSOCIA
outfile <<"\t      Memory ready time = " <<  MAIN_MEMORY_READY_TIME << " chunksize = "

outfile << "\t Execute time = " << dec << time << ";    Total refs = " << refNum << "\
outfile << "\t Number of reference types: [Percentage]\n\t      Reads = " << rCount <
outfile << "\t      Writes = " << wCount << "     " << "[" << fixed << setprecision(2)
outfile << "\t      Inst = " << iCount << "     " << "[" << fixed << setprecision(2) <
outfile << "\t      Total = " << wCount + iCount + rCount << "\n\n" << endl;

outfile << "\t Total cycles for activities: [Percentage]\n\t      Reads = " << rTime <
outfile << "\t      Writes = " << wTime << "     " << "[" << fixed << setprecision(2)
outfile << "\t      Inst = " << iTime << "     " << "[" << fixed << setprecision(2) <<
outfile << "\t      Total = " << wTime + iTime + rTime << "\n\n" << endl;

outfile << "\t Average cycles per activity: \n\t      Read = " << fixed << setprecisio

outfile << "\n\n\t Memory Level: L1i \n";
outfile <<"\t      Hit Count = " <<  l1InstCache->hits() << "  " << "Miss Count = " <<
outfile <<"\t      Total Requests =  " <<  l1InstCache->hits() + l1InstCache->misses()
outfile <<"\t      Hit Rate = " <<  "[" << fixed << setprecision(2) << (float) (((floa
outfile <<"\t      Kickouts = " << l1InstCache->kicks()  << ";  Dirty kickouts = " <<

outfile << "\n\n\t Memory Level: L1d \n";
outfile <<"\t      Hit Count = " <<  l1DataCache->hits() << "  " << "Miss Count = " <<
outfile <<"\t      Total Requests =  " <<  l1DataCache->hits() + l1DataCache->misses()
outfile <<"\t      Hit Rate = " <<  "[" << fixed << setprecision(2) << (float) (((floa
outfile <<"\t      Kickouts = " << l1DataCache->kicks()  << ";  Dirty kickouts = " <<

outfile << "\n\n\t Memory Level: L2 \n";
outfile <<"\t      Hit Count = " <<  l2Cache->hits() << "  " << "Miss Count = " <<  l2
outfile <<"\t      Total Requests =  " <<  l2Cache->hits() + l2Cache->misses() <<endl;
outfile <<"\t      Hit Rate = " <<  "[" << fixed << setprecision(2) << (float) (((floa
outfile <<"\t      Kickouts = " << l2Cache->kicks()  << ";  Dirty kickouts = " << l2Ca

outfile << "\n\n\n\t L1 cache cost (Icache $" << l1iCost << ") + (Dcache $" << l1dCost
outfile << "\t L2 cache cost = $" << l2Cost << ";  Memory cost = $" << memCost << "\n"
outfile << "\t Total cost = $" << totalCost << endl;

outfile.close();


}

return 0;

} ~~~~~~~~~~~
```