Gem5 Learning Notes

Name: Haoxuan Zhang

Research Project Fall 2020
Learning Notes

1. Learning Notes for O3 CPU

1.1 O3 CPU Attributes

a) From Base CPU:

protected:

1. instCnt
2. _cpuId
3. socketId
4. taskID
5. _pid
6. bool _switchedOut
7. _cacheLineSize
8. `std::vector<BaseInterrupts*>interrupts;`
9. `std::vector<ThreadContext *>threadContexts;`
10. `Trace::InstTracer * tracer;`
11. `Cycles previousCycle;`
12. `CPUState previousState;`

public:

1. `ThreadID numThreads;`
2. `System *system;`

b) From FullO3CPU:

protected:

1. EventFunctionWrapper tickEvent; The tick event used for scheduling CPU ticks. What is a CPU tick?
2. EventFunctionWrapper threadExitEvent;The exit event used for terminating all ready-to-exit threads
3. **typename CPUPolicy::Fetch fetch;** This is a class which needs Impl specified
4. **typename CPUPolicy::Decode decode;** This is a class which needs Impl specified
5. **typename CPUPolicy::IEW iew;** The issue/execute/writeback stages.
6. **typename CPUPolicy::Commit commit;** The commit stage.
7. **PhysRegFile regFile;** The physical Register File
8. **typename CPUPolicy::FreeList freeList;** What is this?
9. **typename CPUPolicy::RenameMap renameMap[Impl::MaxThreads];** The rename map
10. **typename CPUPolicy::RenameMap commitRenameMap[Impl::MaxThreads];** The commit rename map
11. **typename CPUPolicy::ROB rob;** The re-order buffer.
12. **std::list<ThreadID>activeThreads;** Active Threads List
13. **std::unordered_map<ThreadID, bool>exitingThreads;** This is a list of threads that are trying to exit. Each thread id is mapped to a boolean value denoting whether the thread is ready to exit.
14. **Scoreboard scoreboard;** The scoreboard
15. **TimeBuffer<TimeStruct>timeBuffer;** The main time buffer to do backwards communication.
16. **TimeBuffer<FetchStruct>fetchQueue;** The fetch stage's instruction queue.

17. **TimeBuffer<DecodeStruct>decodeQueue;** The decode stage's instruction queue.
18. **TimeBuffer<RenameStruct>renameQueue;** The rename stage's instruction queue.
19. **TimeBuffer<IEWStruct>iewQueue;** The IEW stage's instruction queue.
20. **ActivityRecorder activityRec;** The activity recorder; used to tell if the CPU has any activity remaining or if it can go to idle and deschedule itself.

private:

1. **System *system;** Pointer to the system.
2. **std::map <ThreadID,unsigned>threadMap;** Mapping for system thread id to cpu id
3. **std::vector<ThreadID>tids;** Available thread ids in the cpu
4. lots of Stats at the end of the class declaration

public:

1.
```
enum Status {
      Running,
      Idle,
      Halted,
      Blocked,
      SwitchedOut
};
```
2. `BaseTLB *itb;`
3. `BaseTLB *dtb;`
4. `Status _status;`
5. `int instcount;` with flag `NDEBUG` Count of total number of dynamic instructions in flight.
6. `std::list<DynInstPtr> instList;` List of all the instructions in flight.
7. `std::queue<ListIt> removeList;` List of all the instructions that will be removed at the end of this cycle.
8. `bool removeInstsThisCycle;` Records if instructions need to be removed this cycle due to being retired or squashed.

1.2 Sept 19 2020

a) What is LSQ request in Full O3 CPU?

b) if there are multiple lanes dispatching at the same time. how should they register themselves in the reorder buffer?

c) DefaultIEW is initializing width in the way: issueWidth(params->issueWidth) Maybe changing such parameters could change the configuration. next problem is how to change the bandwidth of the memory

d) It seems that the SimObject is polymorphic so I could convert a point to SimObject to a pointer to the derived class.

e) seems that all the parameters in the python classes are declared not in __init__() but in the class itself.

f) what is a probe?

g) every sim_object has a name and the plan is to split the name and check whether the name "cpu" or something is in the list of strings. If so, the object is found.