SESC

SUPER ESCALAR SIMULATOR

Joe Renau, Basilio Fraguela, and Liu Wei

University of Illinois at Urbana-Champaign

http:://sesc.sourceforge.net

March 2002

OUTLINE

- Why Another One?
- Objective
- Mottos
- Coding Style
- Code Organization
- SESC Output
- Simulation Accuracy and Speed
- Todo

WHY ANOTHER ONE?

- Model more details in the out-of-order core
- The need for speed
- After 3 years I still do not understand smt
- To make the new simulator understandable:
  - Clean C++ design
  - Document in source
  - Remove the weird MINT scheduling
  - Common powerful configuration format
  - Facilitate code sharing between projects

OBJECTIVE

- Parameters:
  - Multiprocessor with Release Consistency
  - Multiple types of processors concurrently
  - SMTs
  - Add some “kind-of OS”: TLB, process. . .
- Results:
  - Separate stats generation and processing:
    - The simulator outputs close to raw data
    - Perl scripts process the data
  - Profile applications at inst. level

MOTTOS

- Build the fastest possible simulator
  - Every addition is evaluated
  - Exploit the most common pattern
  - Checks only in debug mode
- Make the code easy to understand and extend
  - Be ashame of bad code
  - Uniform coding style
- Provide as many configurations as possible
  - Add all the reasonable parameters

Coding Style

- Indentation (2 tabstop)
- Braces (Linux Kernel style)

```
if( x ) {
}
```

- Assertions: Use nanassert wherever possible
- Comments, do not over comment. Use doxygen
- Naming convention
- Code replication (rule of 3)

MAIN DIRECTORIES

src main core of the simulator

backend memory sub-system (based on FlexRAM)

os OS support (TLB)

libapp library necessary for applications (ala libsim.a)

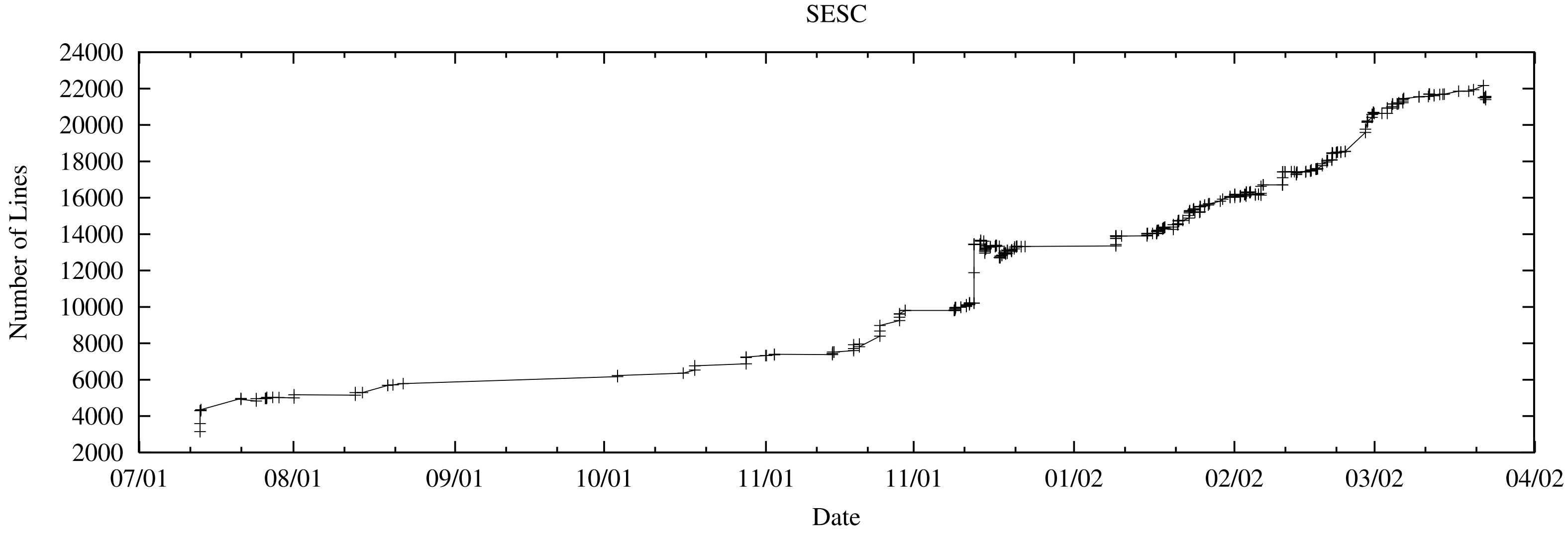
doc documentation files and doxygen output

benchs sample bench and configuration

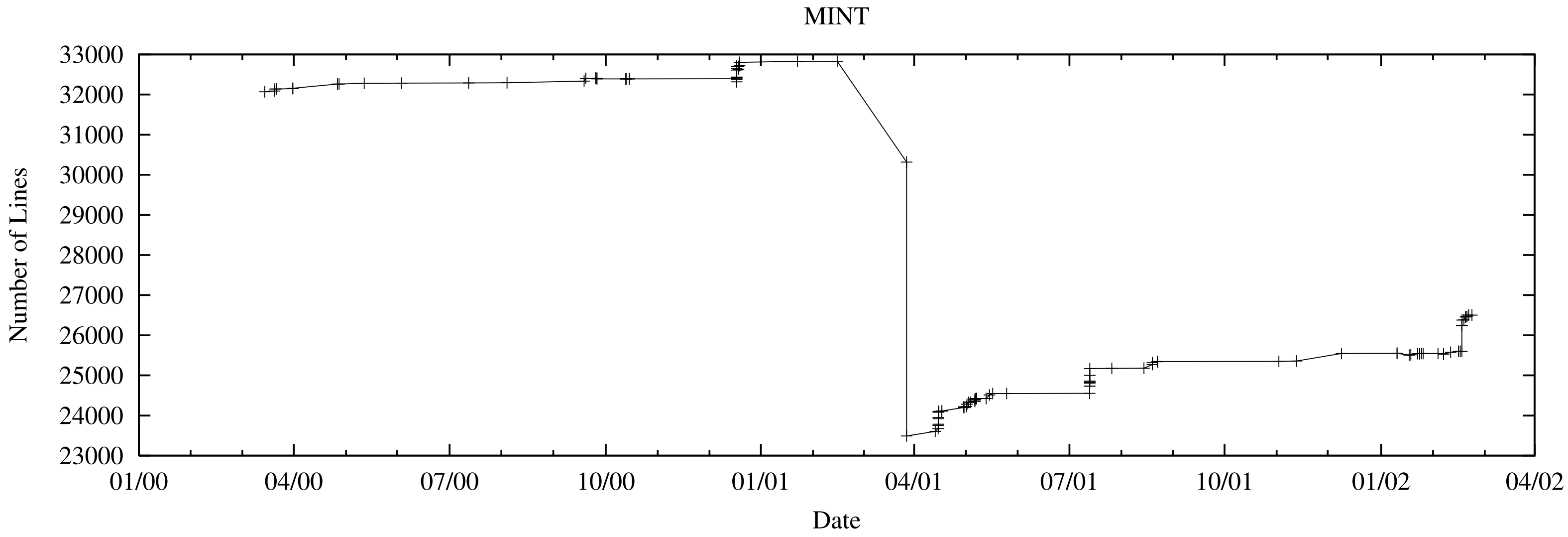
scripts for processing the statistics generated

FlexRAM the FlexRAM architecture (not in sourceforge)

CVS WORK



SESC
Number of Lines
24000
22000
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
07/01
08/01
09/01
10/01
11/01
11/01
01/02
02/02
03/02
04/02
Date



MINT
Number of Lines
33000
32000
31000
30000
29000
28000
27000
26000
25000
24000
23000
01/00
04/00
07/00
10/00
01/01
04/01
07/01
10/01
01/02
04/02
Date

LIBAPP

- Thread support:
  spawn, wait, self, suspend, resume, yield, exit
- Barriers and locks:
  lock/unlock, barrier, psema/vsema, C4 Macros
- Events:
  - preevent: Notified when executed in front-end
  - postevent: Goes through the pipeline
  - memfence : Previous memory operations issued
  - acquire,release : Release Consistency
  - fast_sim_begin, fast_sim_end

Memory Subsystem

```
class DCache:public MemoryObj {
protected:
  void read() {
    MSG("0x%x:ld [0x%x]",req->getIaddr(),req->getVaddr());
    req->goUp(1);
  };

  void write() {
    MSG("time %lld",globalClock);
    req->goUp(1);
  };

  void memFence() { req->goUp(23);  };
  void acquire() {  req->goUp(10);  };
  void release() {  req->goUp(13);  };
};
```

Booting SESC

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

  DummyMemorySystem *cm[NPROC];
  MyProc *core[NPROC];

  osSim = new OSSim(argc, argv, envp);

  for(Pid_t i = 0; i < NPROC; i ++) {
    cm[i]   = new DummyMemorySystem(i);
    core[i] = new MyProc(cm[i], i);
  }

  osSim->boot();

  for(int i = 0; i < NPROC; i ++) {
    delete core[i];
    delete cm[i];
  }
  delete osSim;
}
```

RUNNING SESC

- Compile a program for O32 in SGI (link libapp)
- Check configuration file (sesc.conf)
- ./mtst1 -h0x8000000 crafty <tt.in
- Check output sesc_crafty.xxxxxx
- Process it ../scripts/report.pl sesc_crafty.xxxxxx

Sample Configuration

```
foo  = 4
cpucore[0:31] = 'r10' # iacoma3
[r10]
fetchWidth  = 4
issueWidth  = $(foo)
branchDelay = $(foo)+2  # 6 minimum (deps enforced)
maxBranches = 4   # outstanding branches limit
maxIRequests= 2
iALUUnits   = 2
iFPUnits    = 2
iBJUnits    = 1
LDSTQueue   = 16  # Unified LD/ST buffer queue
winsize     = 32
bb4Cycle    = 1   # max BB fetch per cycle
bpred       = 'r10kBPred'

<../benchs/branchModels.conf>  # Include different branch models
```

REPORT FORMAT

- By default is sesc_bench.xxxxxx
- Includes the input configuration file
- Homogeneous way to report data (GStats.h)

```
KEY:field1=value1:field2=value2:...
```

- Example:

```
ProcessId(0):totalTime=358100288:waitTime=0:spawnTime=0
Proc(0):clockTicks=358100289
BPred_RAS(0):nHit=      13
BPred_RAS(0):nMiss=     0
cache(0):readHit=       6788685
```

Report Output

- Simulating an R10K at 180Mhz
- Crafty native execution 109msec

```
# Bench : ./mtst1 -h0x800000 ../benchs/crafty
# File  : sesc_crafty.EMomhe    :       Mon Mar 25 21:27:33 2002
      Sim. Speed          Exe Time           Sim Time (180MHz)
      1137.96 KIPS        17.88 secs       105.50 msec
Proc  Type          Total             RAS            BPred     BTB
   0  2bit        80.819% ( 99.91% of  10.05%)  84.02%  77.40%
File  IPC     Cycles     Busy       Control      Window  Struct  Memory
   0  1.07  18990598  26.79%  29.29%  18.45%  15.42%   0.20%   9.85%
```

PROFILING

Function	nCalls	call(clk)	IPC	Acum(%)	Time(%)	Ctrl(%)
SCVQScores_al	998	1161690.1	1.19	23.34%	23.34%	25.79%
chan_v_eval	1568385	578.9	0.43	41.62%	18.28%	72.29%
_doscan	903173	581.6	1.56	52.20%	10.58%	68.07%
skipto	22804	9285.8	1.05	56.46%	4.26%	78.42%
hmm_tied_read	47	4469093.4	1.83	60.69%	4.23%	62.13%
root_chan_v_	386348	488.0	0.60	64.48%	3.80%	63.28%

....

Simulation Accuracy I

Simulating an R10K at 180Mhz (SGI Origin 200 IP27):

Bench	Native	Simulated	Error	SimTime
matrix	78ms	80ms	+2%	18s
crafty	109ms	105ms	-3%	18s
mcf	791ms	792ms	+0%	142s
mp3dec	643ms	619ms	-5%	160s
lat_mem	2405ms	2532ms	+5%	230s

SIMULATION ACCURACY II

Simulating an R4400 at 150Mhz (SGI IP22):

Bench	Native	Simulated	Error	SimTime
matrix	291ms	282ms	-4%	27s
crafty	265ms	271ms	+4%	43s
mcf	2438ms	2422ms	-1%	174s
mp3dec	2347ms	2521ms	+7%	185s
lat_mem	4005ms	4129ms	+3%	279s

Simulation Speed

- Crafty simulation in a Pentium-III 1GHz

1137KIPS SESC simulating an R10K (with backend)

178KIPS smt simulating an R10K (no backend)

2608KIPS mint in-order

$\approx$**150KIPS** simplescalr sim-outorder

FULL SPECINT

- Assuming 1000KIPS we can run:
- 175.vpr Ref in ≈30h
- 197.parser Ref in ≈90h
- Reduced Data sets (Minnesota) in ≈30min

EVEN FASTER

- Parallelization:
  - Low ≈ 2: 2 threads likely to be balanced
  - High ≈ 4: Several threads in multiprocessors
- Configuration Value Profile $\approx 20\% - 40\%$
  - Convert variables to constants
  - The compiler simplifies the code
  - Run-time verification

Close to 3MIPS in our dual athlon

DO YOU WANNA RUN IT?

- Add http://sesc.sourceforge.net to your bookmark
- Subscribe to sesc-news@lists.sourceforge.net
- Read all the files in doc directory
- See the ./src/main.cpp example
- Understand the callbacks (DInst.h is easy)
- Understand the configuration files (*.conf)

TODO

- More advance branch predictor: Multi-level, multi-cycle
- Add I-Cache (soon)
- Document the code with doxygen (starting)
- Create sample codes and document them
- Type a hackers guide (started)
- Energy model?
- Give more flexibility to profiling
- Use gcc instead of MIPSPro to generate benchmarks