Gem5 for Arm+SVE and RIKEN Post-K simulator

Yuetsu Kodama and Mitsuhisa Sato

Architecture Development Team,

FLAGSHIP 2020 project

RIKEN Center for Computational Science (R-CCS)

Copyright2018 RIKEN

What is gem5

- **Gem5 is an open-source processor simulator**
  - Detail information is available at http://gem5.org
  - ✓ General-purpose processor simulator
  - ✓ “a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture.”
  - ✓ Gem5 supports multiple CPU models: atomic, in-order, out-of-order(o3)
  - ✓ Gem5 O3 CPU model is a general model based on Alpha 21264.
  - ✓ Gem5 supports multiple ISAs: x86, alpha, SPARC, and arm. Since SVE o3 was not supported, we developed it..
  - ✓ It is useful for evaluating the relative performance by tuning effects such as changing compiler options and rewriting source code.

RIKEN Post-K Simulator

- **Riken Simulator is based on 'gem5'.**
  - ✓ Gem5 supports multiple ISAs: x86, alpha, SPARC, and arm. Since SVE o3 was not supported, we developed it.
    - But currently gem5 supports SVE by arm, so we changed the base code to the arm code in processor level, but we also enhanced the system level, such as cache and memory system.
  - ✓ We tuned the parameters of gem5 for Post-K, described in detail later.
- **How about Accuracy of our simulator?**
  - We compared with Fujitsu's in-house simulator. We found less than 10% difference in 58 kernels (70%) out of 82 kernels in our test set.
  - There are some difference such as combined gather load, etc.

Copyright2018 RIKEN

Open gem5-Arm-SVE

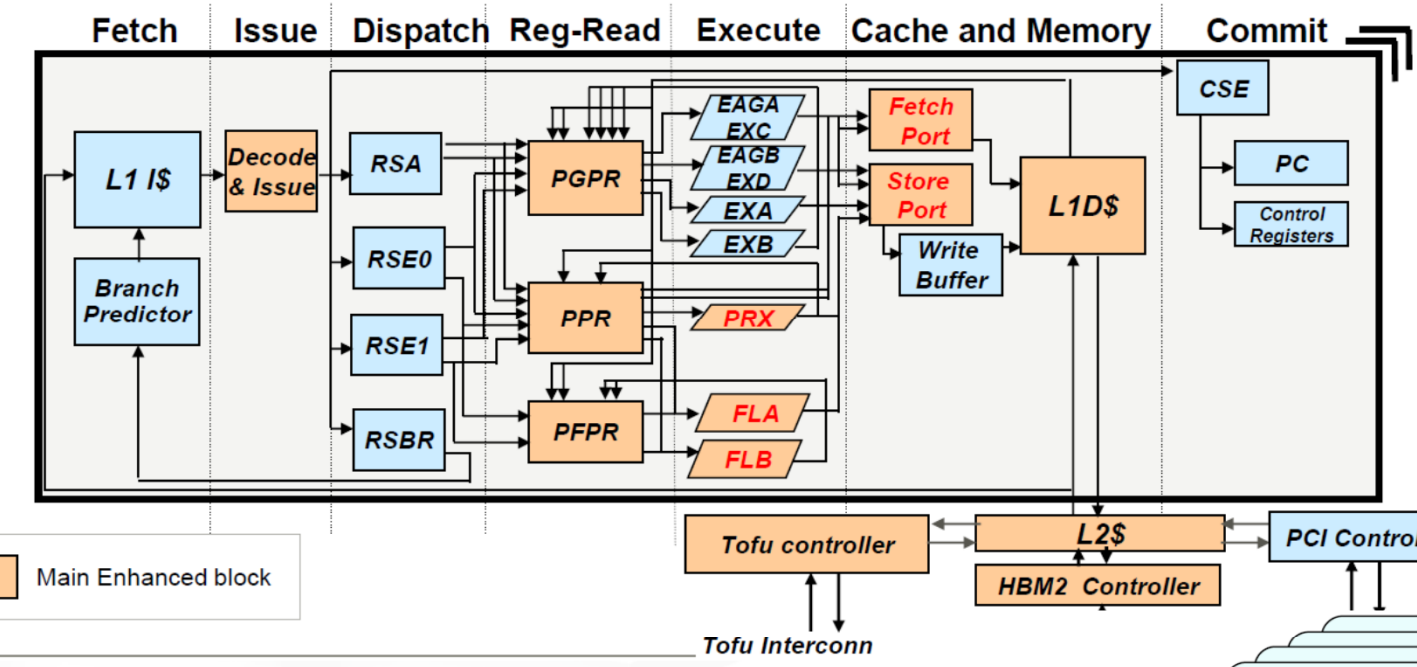
- **RIKEN simulator is based on parameters under NDA with Fujitsu, so we cannot share it with the community.**
- **Open gem5-Arm-SVE is an "open" version of our gem5 by excluding the detailed parameters of A64FX.**
  - By co-operations with Prof. Simon in University of Bristol, we set the "open" parameters and re-build the gem5 simulator.
  - **Note: "Some of parameters in this gem5 are taken from public information about Cavium ThunderX2, but all parameters including these parameters are not relevant to any specific hardware."**
  - The gem5 simulator provided supports the simulation of SVE instructions on an Arm based architectural mode.
  - An out-of-order processor core, similar to that in a current generation Arm HPC processor, however, we also implement a SVE unit, whose default set is 512-bit.

Copyright2018 RIKEN

RIKEN Post-K simulator

Copyright2018 RIKEN

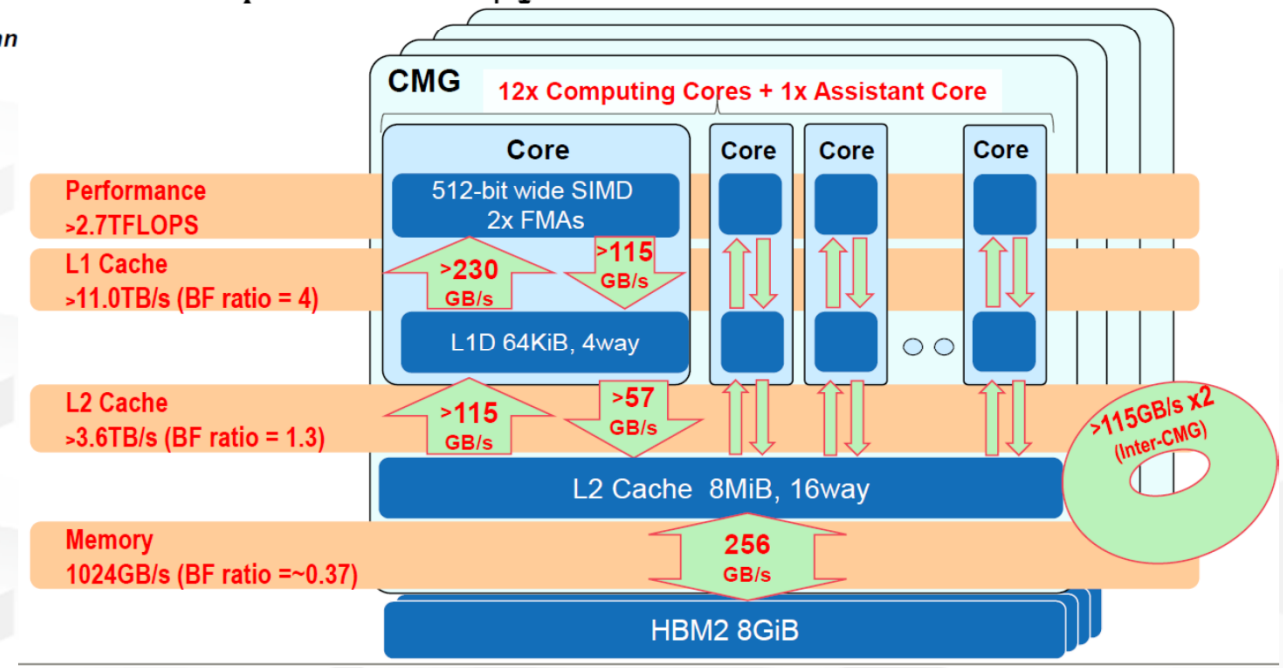
Parameter tuning for A64FX



- latency/throughput of each pipel ine stage.
- Number of function units, and lat ency/throughput of each units.
- Which units are used in each inst ruction.

Both figure from SSKEN_hpcf2018_Y oshidaToshio.pdf

L1 cache size, number of ways, latenc y, throughput for load/store units.
L2 cache size, number of ways, latenc y, throughput, number of banks.
Memory size, latency, throughput, ba nk access scheduling, address mappin g/interleaving for HBM2.



Copyright2018 RIKEN

Gem5 Execution Mode

- Atomic (instruction level: an instruction executed in a cycle)

  ○ Dynamic instruction counts

  ○ Dynamic instruction mix (integer, floating, SIMD, memory, branch, etc.)

  ○ OpenMP multithread execution on a CMG.

  × Execution cycles（execution cycle = number of instruction）

  △ Simulation Speed order is 1M instructions/sec. (it takes 20 minutes for 1 second execution)

- O3 (cycle level: out-of-order pipeline is simulated in a cycle)

  ○ Execution cycles (general out-of-order super scalar execution model, not precise Post-K model). The target difference between gem5 and Post-K is within 10%.

  ○ Cache hit rate (L1, L2, memory access)

  △ OpenMP multithread execution on a CMG. (it emulates system call by software and the overhead is 0, so the thread overhead cannot be evaluated.)

  × Simulation Speed order is 100K instructions/sec for a core, and 10K instructions/sec for 10 threads. (it takes 30 hours for 1 second execution by 10 threads)

It cannot simulate whole application. You should extract kernels that executes msec order.

Copyright2018 RIKEN

Current status of RIKEN Simulator

- **We compared with Fujitsu's in-house simulator**
  - We tested 82 kernels, and 58 kernels (70%) were less than 10% error.
  - Several kernels has more than 10% error. This is because of lack of functions in gem5, such as combined gather load, etc.
- **We compared with a test chip in L2 and memory scalability for number of thread by STREAM triad.**
  - The scalability are well simulated
  - We found more than 10% difference in peak bandwidth for L2 and single thread bandwidth for memory.
  - This is because of lack of functions in gem5, such as target L2 software and hardware prefetch, etc. We plan to implement them to reduce the difference.

Copyright2018 RIKEN

Evaluation Example

- We select a nbody program.
- We use the same source but different compiler options of Fujitsu compiler.

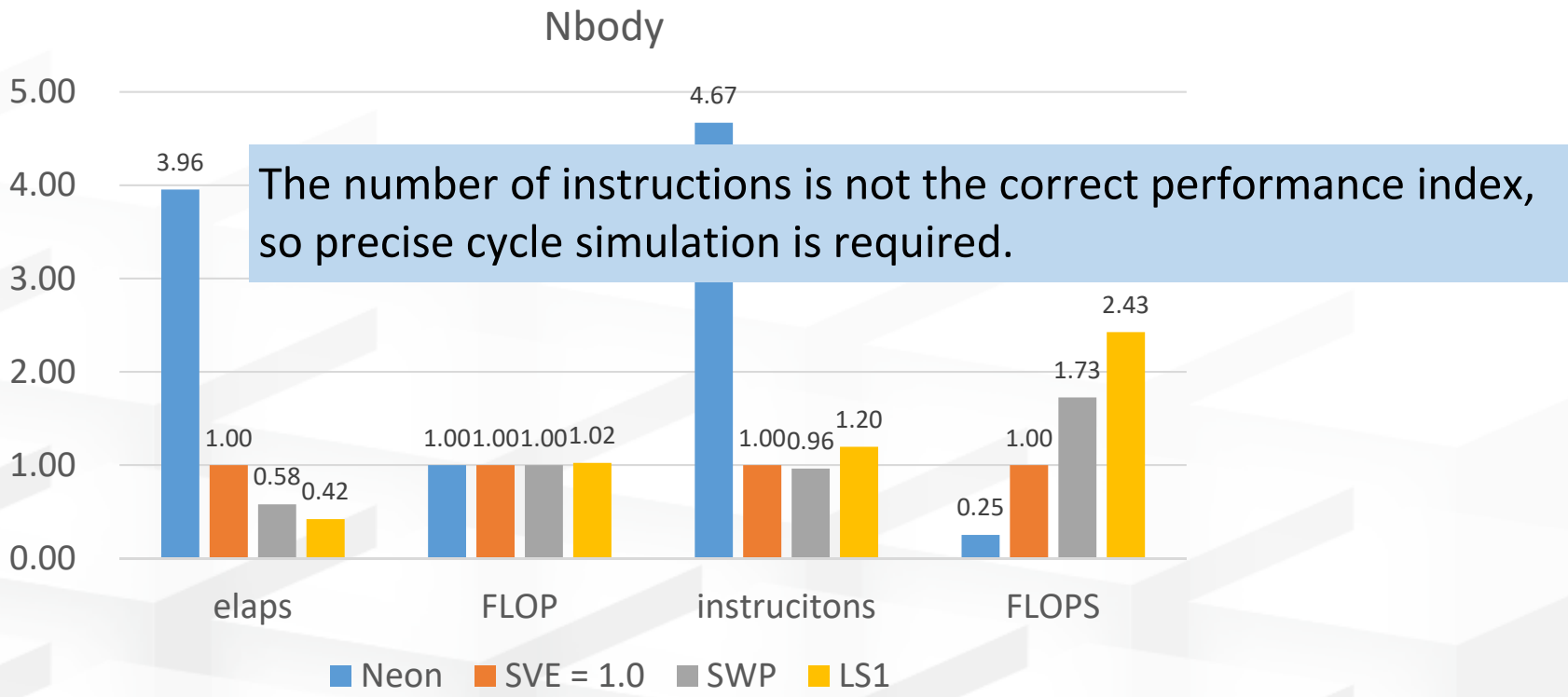
Neon: -Kfast,noswp,nounroll,NOSVE

SVE: -Kfast,noswp,nounroll

SWP: -Kfast

LS1: -Kfast,ocl with loop split point is specified by ocl(optimize control line)

512bit SVE



The number of instructions is not the correct performance index, so precise cycle simulation is required.

Copyright2018 RIKEN

Open Gem5-Arm-SVE

Copyright2018 RIKEN

Open Gem5-Arm-SVE

- **Open gem5-Arm-SVE is an “open” version of our gem5.**
- Note: **Some of parameters in this gem5 are taken from public information about Cavium ThunderX2, but all parameters including these parameters are not relevant to any specific hardware.”**

- **The timings from these simulations will not be representative of any specific processor, either current generation or future generation. It is designed purely to provide insight on how to simulate SVE instructions.**
- **It is not tuned in L2 and memory performance yet, so it supports multi-core facility, but you had better to use a single core simulator currently.**

Sample run

- **Afternoon hands on, arm will provide more detailed demo using gem5, here we show very simple sample.**
- **You can use docker image of gem5 and gcc for SVE as follows.**

```
$ docker load < riken-gem5-env2.tar.gz
$ docker run --name kodama --it -u r-sim -w /home/r-sim riken-tutorial2
$ cp /opt/samples/daxpy_C.c .
$ cp /opt/samples/mytime.h .
$ make –f /opt/samples/Makefile_C daxpy_C.axf
$ ls
daxpy_C.axf  daxpy_C.c  daxpy_C.o  daxpy_C.s  mytime.h
$ gem5-atomic -c daxpy_C.axf
gem5 Simulator System.  http://gem5.org
…
elaps : 0.000625 (ms)
GFLOPS : 3.276800
$ gem5-o3 -c daxpy_C.axf
…
elaps : 0.000445 (ms)
GFLOPS : 4.602247
```

Binary should be statically linked.

Binary is specified by -c

This time and GFLOPS are not correct.

This time and GFLOPS are correct.

Sample run (cont)

```
$ ln -s daxpy_C.c daxpy_neon.c
$ diff /opt/samples/Makefile_C /opt/samples/Makefile_C_neon
< CFLAGS        = -O3 -march=armv8-a+sve -fopenmp
---
> CFLAGS        = -O3 -march=armv8-a -fopenmp
$ make -f /opt/samples/Makefile_C_neon daxpy_neon.axf
$ gem5-o3 -c daxpy_neon.axf
…
elaps : 0.001029 (ms)
GFLOPS : 1.990282
$ gem5-o3 -c daxpy_neon.axf -o “1024 1000”
elaps : 0.370236 (ms)
GFLOPS : 5.531607
$ gem5-o3 -c daxpy_C.axf -o “1024 1000”
elaps : 0.126590 (ms)
GFLOPS : 16.178213
```

Arguments are specified by -o

SVE is 2.9 times faster than neon

Sample run (cont)

```
$ gem5-o3 -c daxpy_C.axf -o “1024 1000”
command line: /opt/gem5/fs_sim_riken/build/ARM/gem5.opt /opt/gem5/fs_sim_riken/
configs/example/se.py --cpu-type=O3_ARM_riken_3 --caches --l2cache -c daxpy_C.axf -o
'1024 1000'
GFLOPS : 16.178213
$ /opt/gem5/fs_sim_riken/build/ARM/gem5.opt /opt/gem5/fs_sim_riken/configs/exam
ple/se.py--arv-sve-vl=1 --cpu-type=O3_ARM_riken_3 --caches --l2cache -c daxpy_C.axf -o
'1024 1000‘
GFLOPS : 4.158300
$ /opt/gem5/fs_sim_riken/build/ARM/gem5.opt /opt/gem5/fs_sim_riken/configs/exam
ple/se.py--arv-sve-vl=2 --cpu-type=O3_ARM_riken_3 --caches --l2cache -c daxpy_C.axf -o
'1024 1000‘
elaps : 0.248568 (ms)
GFLOPS : 8.239194
$ /opt/gem5/fs_sim_riken/build/ARM/gem5.opt /opt/gem5/fs_sim_riken/configs/exam
ple/se.py --arm-sve-vl=8 --cpu-type=O3_ARM_riken_3 --caches --l2cache -c daxpy_C.axf
-o '1024 1000'
GFLOPS : 31.408634
```

512bit SVE is 3.9 times fater than 128bit SVE

128bit SVE

256bit SVE is 2.0 times faster than 128bit SVE

1024bit SVE is 7.6 times faster than 128bit SVE

Same binary can be run with different vector length:
vector length agnostic programming