Celerity Simulation Quick Start V2

Bespoke Silicon Group @ University of Washington (http://bjump.org)

1. Checkout Repos

1.1. git clone https://bitbucket.org/taylor-bsg/bsg_celerity_benchmarks.git
1.2. cd bsg_celerity_benchmarks
1.3. Checkout baseline**:**
baseline is a set of coherent versions of dependent repos. Upto now following baselines are defined:
BASELINE=CELERITY_V1_ACTUAL : celertity v1 tapeout
=CELERITY_V1_INTENDED: default, intended v1 tapeout.
=CELERITY_V2 : v2 tapeout, backend fix.
=CELERITY_V3 : v3, with remote-load and atomics.
=LATEST : Warning, latest repo. Not fully test.

1.3.1. Have bsg_**cadenv** access:
make checkout-repos BASELINE=<baseline-name> BSG_CADENV=1
1.3.2. Do not have bsg_**cadenv** access:
make checkout-repos BASELINE=<baseline-name>
1.3.3. If you do not have access to bsg_**cadenv** repository:
a) Setup the VCS license.
b) Setup the VCS_HOME
c) After checkout all of the repos (see Section 2),
in vcs-build/bsg_rocket/rockets/coyote/testing/rtl/Makefile
replace include ../../../../../cad/common/mk/cadenv.mk"
with "export VCS_BIN=<Path To VCS Bin Directory>"

2. Build toolchains and Test

Only need to do once.

2.1. Make sure GCC>=4.8 available. Otherwise:
2.1.1. Checkout comments in **vcs-build/bsg_riscv/Makefile** for how to install gcc-4.8 if not available.
2.1.2. Check the **CC** and **CXX** in following Makefile if gcc is not default.
2.1.2.1. vcs-build/bsg_riscv/Makefile
2.1.2.2. vcs-build/bsg_rocket/rockets/coyote/testing/rtl_five/Makefile
2.1.2.3. vcs-build/bsg_manycore/software/riscv-tools/Makefile

2.2. make build-manycore-tools ## EST time: 12 mins
2.3. make build-rocket-tools ## EST time: 16 mins

2.4. make loopback-test ## EST time: <1 mins

3. Run Benchmarks

3.1. cd vcs-build/bsg_rocket/rockets/coyote/testing/rtl_five
3.2. make run BENCHMARK_0=bsg_rocket_loopback # without waveform
3.3. make run_debug BENCHMARK_0=bsg_rocket_loopback # with waveform
3.4. make soft_run BENCHMARK_0=bsg_rocket_loopback
run without recompiling RTL. Faster.

4. Run different benchmarks

4.1. Benchmark directory: All benchmark hex files should be placed in this directory
bsg_rocket/common/benchmark
4.2. Run different benchmark:
make run BENCHMARK_0=<**benchmark_name**>
without .riscv.hex suffix
4.3. Run on different Rocket:
make run BENCHMARK_<**rocket_num**>=<**benchmark_name**>
rocket_num can be 0, 1, 2, 3, 4
But some benchmark can only run on specific
rocket
4.4. Recompile benchmarks:
cd bsg_rocket/rockets/coyote/testing/rtl_five
./compile_bmark <**benchmark_name>**

Celerity Simulation Quick Start V1

Bespoke Silicon Group @ University of Washington (http://bjump.org)

1. Get code

1.1. mkdir celerity
1.2. cd celerity
1.3. **Manycore repo**: git clone https://bitbucket.org/taylor-bsg/bsg_manycore
1.4. **IP repo**: git clone https://bitbucket.org/taylor-bsg/bsg_ip_cores
1.5. **Top Design repo**:git clone https://bitbucket.org/taylor-bsg/bsg_designs
git checkout Synopsys_Benchmark
1.6. **Package repo**: git clone https://bitbucket.org/taylor-bsg/bsg_packaging
1.7. **RISC-V tools**: git clone https://bitbucket.org/taylor-bsg/bsg_riscv
1.8. **Sim Framework**: git clone https://bitbucket.org/taylor-bsg/bsg_rocket
1.9. **CAD setup**:

1.9.1. For Others:
a) Setup the VCS license.
b) Setup the VCS_HOME
c) Replace the first line:
"include ../../../../../cad/common/mk/cadenv.mk"
in bsg_rocket/rockets/coyote/testing/rtl/Makefile
with "export VCS_BIN=<Path To VCS Bin Directory>"

1.9.2. d) Alternatively, you can modify the rules to use Verilator or Vivado command line simulator.

1.9.3. For Bespoke Silicon Group Members:
git clone https://bitbucket.org/taylor-bsg/cad
git checkout bsg_tsmc180

2. Build RISC-V toolchain

2.1. Prerequisite

2.1.1. Make sure host machine has **gcc >= 4.8**

2.1.2. Checkout comment in bsg_rocket/Makefile for how to install **gcc-4.8** if not available.

2.2. Toolchain for Berkeley Rocket:

2.2.1. cd bsg_riscv

2.2.2. git checkout bsg_celerity
[check **CC** and **CXX** in Makefile if gcc is not default.]

2.2.3. make checkout-all ## EST time: 1 mins

2.2.4. make build-riscv-tools-newlib ## EST time: 4 mins

2.2.5. make test-spike-hello ## Verify the installation

2.3. Toolchain for Manycore:

2.3.1. cd bsg_manycore/software/riscv-tools
[check **CC** and **CXX** Makefile if gcc is not default.]

2.3.2. make checkout-all ## EST time: 9 mins

2.3.3. make build-riscv-tools ## EST time: 41 mins

3. Run simulation

3.1. cd expobsg_rocket/rockets/coyote/testing/rtl_five
[check **CC** and **CXX** in Makefile if gcc is not default.]

3.2. make run BENCHMARK_0=bsg_rocket_loopback # without waveform

3.3. make run_debug BENCHMARK_0=bsg_rocket_loopback # with waveform

4. Run different benchmarks

4.1. Benchmark directory:
bsg_rocket/common/benchmark

4.2. Run different benchmark:
make run BENCHMARK_0=<**benchmark_name**>
without .riscv.hex suffix

4.3. Run on different Rocket:
make run BENCHMARK_<**rocket_num**>=<**benchmark_name**>
rocket_num can be 0, 1, 2, 3, 4
But some benchmark can only run on specific
rocket

4.4. Recompile benchmarks:
cd bsg_rocket/rockets/coyote/testing/rtl_five
./compile_bmark <**benchmark_name>**

5. Some running time metrix:

Following table shows some benchmarks we have ran on our server.

- Benchmarks with '**bnn**' will activate the Binary Neural Network Accelerator.
- Benchmarks with '**streambuf**' will stream weights that stored in manycore to the BNN
- Benchmarks with '**layers**' means only run for single layer of the BNN
- Benchmarks without '**layers**' means to run the full BNN.

Type following commands will run the benchmarks and generate the running result at a directory named by the date & time:

```
>> cd bsg_rocket/rockets/coyote/testing/rtl_five
>> make run_all
```

Server Configuration:
CPU: Intel(R) Xeon(R) CPU E3-1241 v3 @ 3.50GHz, 4 cores
Mem: 16GB

BENCHMARK_0	BENCHMARK_4/2[1]	TIME[2]	
bsg_rocket_loopback:	NULL	1	mins
bsg_rocket_manycore_loopback:	NULL	1	mins
NULL:	bnn_loopback	1	mins
bsg_rocket_manycore_token_queue:	NULL	3	mins
NULL	bnn_layer_1	8	mins
manycore_streambuf_layer_1	bnn_layer_1_sneakpath	14/908	mins
manycore_streambuf_layer_2	bnn_layer_2_sneakpath	16	mins
manycore_streambuf_layer_3	bnn_layer_3_sneakpath	16	mins
manycore_streambuf_layer_4	bnn_layer_4_sneakpath	23	mins
manycore_streambuf_layer_5	bnn_layer_5_sneakpath	32	mins
manycore_streambuf_layer_6	bnn_layer_6_sneakpath	57	mins
manycore_streambuf_layer_7	bnn_layer_7_sneakpath	202/1771	mins
manycore_streambuf_layer_8	bnn_layer_8_sneakpath	22	mins
manycore_streambuf_single_image	bnn_sneakpath	1141/9893	mins

[1] For Versions after CELERITY_V1, use ‘BENCHMARK_2’; for Versions “CELERITY_V1_*”, use “BENCHMARK_4”.
[2] Time for 16x31/64x64 Configurations.