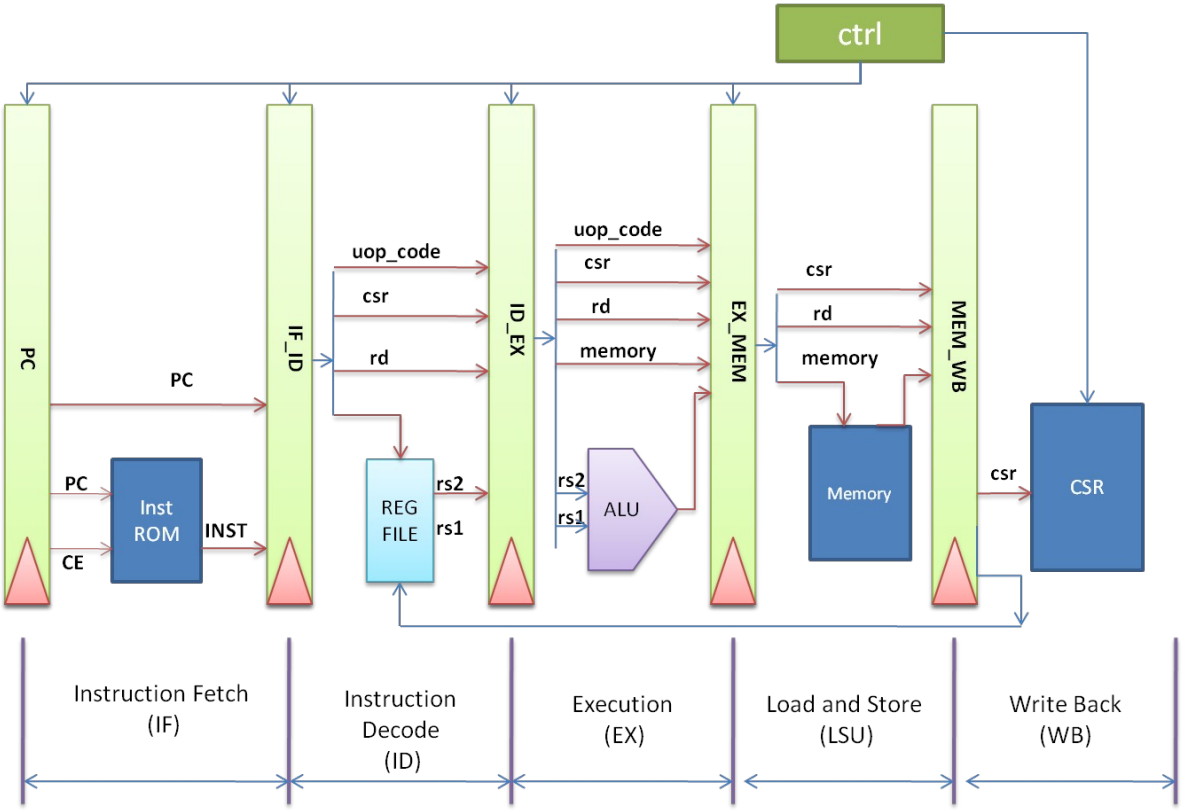
This CPU is a training-target implementation of rv32im, designed to be simple and easy to understand. However, most of the technologies of RISC-V CPU design are still addressed in this practice (named: Kookaburra), including the classic pipeline structure, dealing with data/structure/control hazards, precise exception and interrupt, pipeline stalling and flushing etc. The self-explained Verilog code accompanied with ample comments makes it is easy to expand a variety of new features, for instance, adding i-cache, data-cache units, supporting machine and user privileges as well as applying the branch prediction and etc.

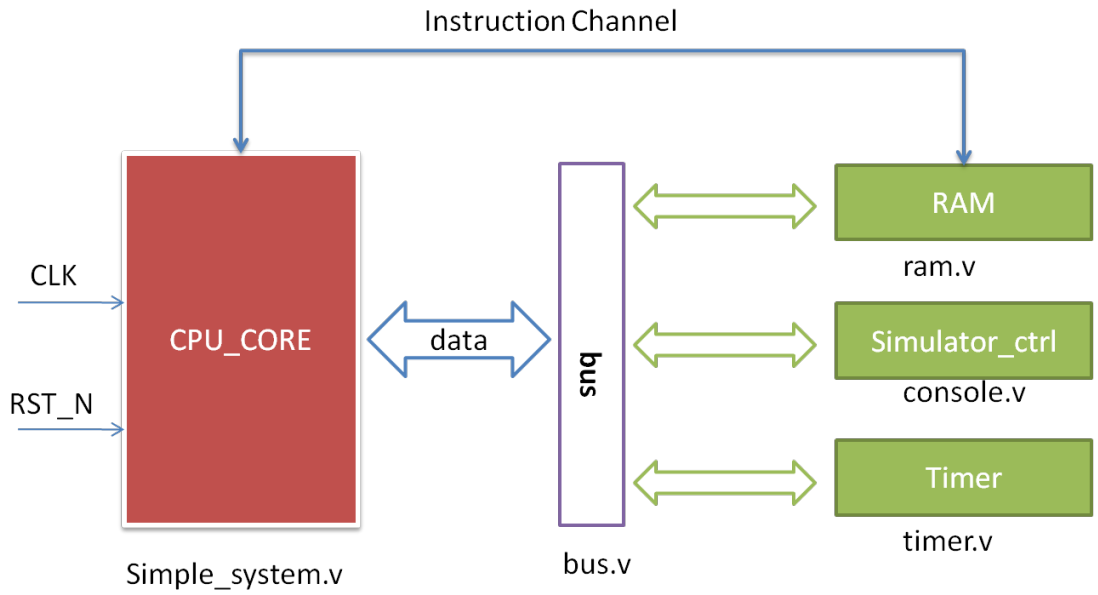
1. The micro-architecture

The implementation did not dwell on the micro-architecture design, it has adopted the typical 5 stages pipeline (instruction fetch, instruction decoder, execute, LSU and write back), with the control unit in charge of coordinating all the components as well as handling the exception and interruptions.



2. Test bench

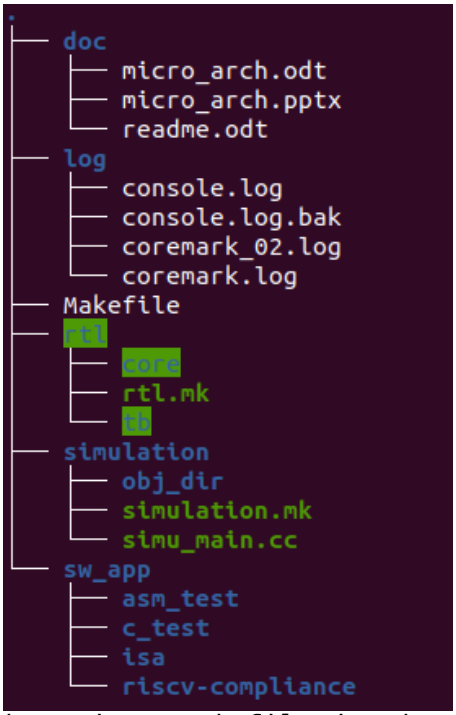
The test bench consists of the CPU core, the data bus as well as some other necessary peripheral devices, including a dual-port RAM (one port for instruction fetching, the other for data access), the simulator ctrl (acting as the console to display the debug information), and a timer to produce interrupts. The bus, simulator ctrl and the timer are from the open-source project ibex provided by lowRISC.



3. verification

3.1 requirements
To run the simulation, the two components (verilator and rv32 gcc toolchain) are necessary, these are accessible from their official website.

3.1 Compile the CPU and test bench



there is a makefile in the top directory that can be used to build the whole test bench.

```
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git$ make all
make -C ./rtl -f rtl.mk all
make[1]: Entering directory '/var/cpu_testbench/Cookabarra_git/rtl'
===============compile RTL into cpp files, start=========================
verilator --cc -trace -Wno-style   -I/var/cpu_testbench/Cookabarra_git/rtl/core/include --top-module simple_system /var/cpu_testbench/Cookabarra_git/rtl/tb/ram.v /var/cpu_testbench/Cookabarra_git/rtl/tb/simple_system.v /var/cpu_testbench/Cookabarra_git/rtl/tb/timer.v /var/cpu_testbench/Cookabarra_git/rtl/tb/console.v  /var/cpu_testbench/Cookabarra_git/rtl/tb/bus.v /var/cpu_testbench/Cookabarra_git/rtl/core/ifu/ifu.v
 /var/cpu_testbench/Cookabarra_git/rtl/core/ifu/tf_id.v      /var/cpu_testbench/Cookabarra_git/rtl/core/dec/id.v        /var/cpu_testbench/Cookabarra_git/rtl/core/dec/id_ex.v /var/cpu_testbench/Cookabarra_git/rtl/core/exu/ex.v      /var/cpu_testbench/Cookabarra_git/rtl/core/exu/div.v        /var/cpu_testbench/Cookabarra_git/rtl/core/exu/ex_mem.v     /var/cpu_testbench/Cookabarra_git/rtl/core/lsu/mem.v       /var/cpu_testbench/Cookabarra_git/rtl/core/lsu/mem_wb.v /var/cpu_testbench/Cookabarra_git/rtl/core/wb/gpr.v      /var/cpu_testbench/Cookabarra_git/rtl/core/wb/csr.v           /var/cpu_testbench/Cookabarra_git/rtl/core/ctrl/ctrl.v     /var/cpu_testbench/Cookabarra_git/rtl/core/core_top.v        Vsimple_system.cpp
==================compile RTL into cpp files, end =========================
===============add rtl object files into cpp files, start=====================
make -C ./obj_dir -f Vsimple_system.mk
make[2]: Entering directory '/var/cpu_testbench/Cookabarra_git/rtl/obj_dir'
/usr/bin/perl /usr/local/share/verilator/bin/verilator_includer -DVL_INCLUDE_OPT=include Vsimple_system.cpp Vsimple_system__Dpi_Export__0.cpp Vsimple_system___024root__DepSet_h3a5c9f22__0.cpp Vsimple_system___024root__DepSet_ha17093c5__0.cpp Vsimple_system__Dpi.cpp Vsimple_system__Trace__0.cpp Vsimple_system__Trace__0.cpp Vsimple_system__ConstPool_0.cpp Vsimple_system___024root__Slow.cpp Vsimple_system___024root__DepSet_ha17093c5__0__Slow.cpp Vsimple_system__Syms.cpp Vsimple_system__Trace__0__Slow.cpp > Vsimple_system__ALL.cpp
g++  -I.  -MMD -I/usr/local/share/verilator/include -I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -DVM_TRACE_FST=0 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable -Wno-shadow      -std=c++14 -Os -c -o Vsimple_system__ALL.o Vsimple_system__ALL.cpp
echo "" > Vsimple_system__ALL.verilator_deplist.tmp
```

3.2 Test

```
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/helloworld$ make clean
rm -f -f /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o helloworld.o  /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.d /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.d helloworld.d /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.d helloworld.elf helloworld.vmem helloworld.bin helloworld.dump
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/helloworld$ make all
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding  -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding  -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding  -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o helloworld.o helloworld.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding  -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.S
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding  -T /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/link.ld /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o helloworld.o  /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o -o helloworld.elf
riscv32-unknown-elf-objcopy -O binary helloworld.elf helloworld.bin
srec_cat helloworld.bin -binary -offset 0x0000 -byte-swap 4 -o helloworld.vmem -vmem   # 0x00100000
riscv32-unknown-elf-objdump --disassemble-all helloworld.elf > helloworld.dump
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/helloworld$
```

you can run the hellworld test case via the following command:
 ./tb ./sw_app/c_test/helloworld/helloworld.vmem

3.3 Coremarks

Tested with the coremark(downloaded from https://github.com/eembc/core-mark.git)

```
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/coremark$ make clean
rm -f -f /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o core_main.o core_list_join.o core_matrix.o core_state.o core_util.o core_portme.o  /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.d /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.d core_main.d core_list_join.d core_matrix.d core_state.d core_util.d core_portme.d /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.d core_main.elf core_main.vmem core_main.bin core_main.dump
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/coremark$ make all
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_main.o core_main.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_list_join.o core_list_join.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_matrix.o core_matrix.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_state.o core_state.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_util.o core_util.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o core_portme.o core_portme.c
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -MMD -c -I/var/cpu_testbench/Cookabarra_git/sw_app/c_test/common  -o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.S
riscv32-unknown-elf-gcc           -march=rv32im -mabi=ilp32 -static -mcmodel=medany -Wall -O2 -fvisibility=hidden -nostdlib -nostartfiles -ffreestanding -fno-builtin-printf -fno-builtin-malloc -DITERATIONS=1000 -DPERFORMANCE_RUN=1 -T /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/link.ld /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/xprintf.o /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/util.o core_main.o core_list_join.o core_matrix.o core_state.o core_util.o core_portme.o  /var/cpu_testbench/Cookabarra_git/sw_app/c_test/common/crt0.o -o core_main.elf
riscv32-unknown-elf-objcopy -O binary core_main.elf core_main.bin
srec_cat core_main.bin -binary -offset 0x0000 -byte-swap 4 -o core_main.vmem -vmem   # 0x00100000
riscv32-unknown-elf-objdump --disassemble-all core_main.elf > core_main.dump
shawnliu@shawnliu-Aspire-TC-780:/var/cpu_testbench/Cookabarra_git/sw_app/c_test/coremark$
```

follow the command below to run the coremark:
 ./tb ./sw_app/c_test/coremark/core_main.vmem

```
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 413265041
Total time (secs): 82
Iterations/Sec   : 12
Iterations       : 1000
Compiler version : GCC8.2.0
Compiler flags   : -O2 -fno-common -funroll-loops -finline-functions --param max-inline-insns-auto=20 -falign-functions=4 -falign-jumps=4 -falign-loops=4
Memory location  : STATIC
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xd340
Correct operation validated. See readme.txt for run and reporting rules.
```

coremark/Mhz = 1000*1000000.0/413265041.0 = 2.42

The formula to calculate the coremark/Mhz:

```
coremark/Mhz = (iterations/secs) / freq_in_MHz
             = iterations/(cycles/freq_in_Hz) / freq_in_MHz
             = iterations/(cycles/(freq_in_MHz*1000000)) / freq_in_MHz
             = iterations * freq_in_MHz*1000000 /cycles / freq_in_MHz
             = iterations * 1000000 /cycles
```

Note: Seen a lot of nop instruction were inserted into the pipeline when there is a branch or jump happened, the performance can be expected to improve significantly by introducing a branch prediction unit.