Supplementary

Outline

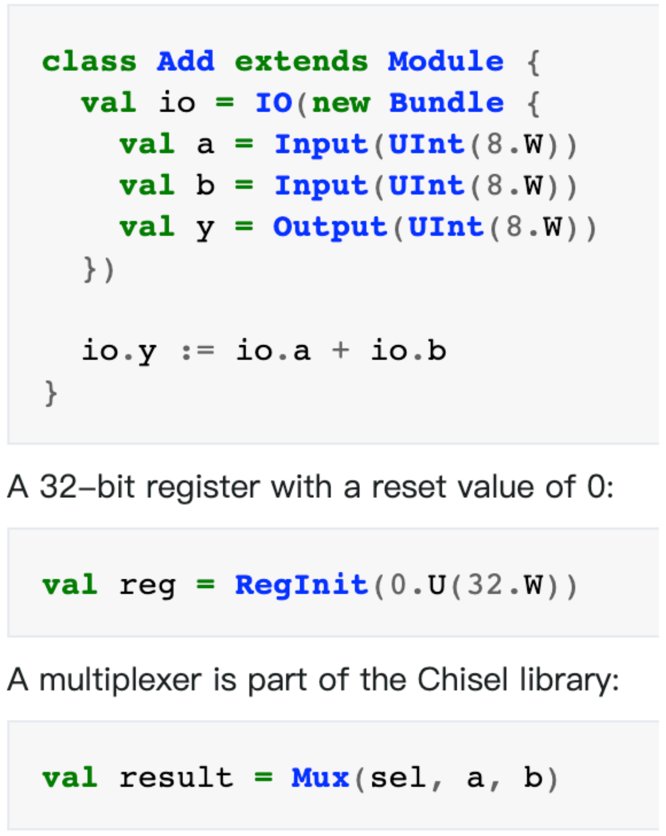
- Tool
  - Chisel
  - Rocket-chip
  - Chipyard
  - Riscv-tests
- HW5

Outline

- Tool
  - Chisel
  - Rocket-chip
  - Chipyard
  - Riscv-tests
- HW5

Chisel

- Constructing hardware in a Scala embedded language
- Open source hardware description language to describe circuits at RTL
- It can be converted to verilog for synthesis
- Not exactly HLS cause there's no EDA tool for it
- Developed by Berkeley



```
class Add extends Module {
  val io = IO(new Bundle {
    val a = Input(UInt(8.W))
    val b = Input(UInt(8.W))
    val y = Output(UInt(8.W))
  })

  io.y := io.a + io.b
}
```

A 32–bit register with a reset value of 0:

```
val reg = RegInit(0.U(32.W))
```

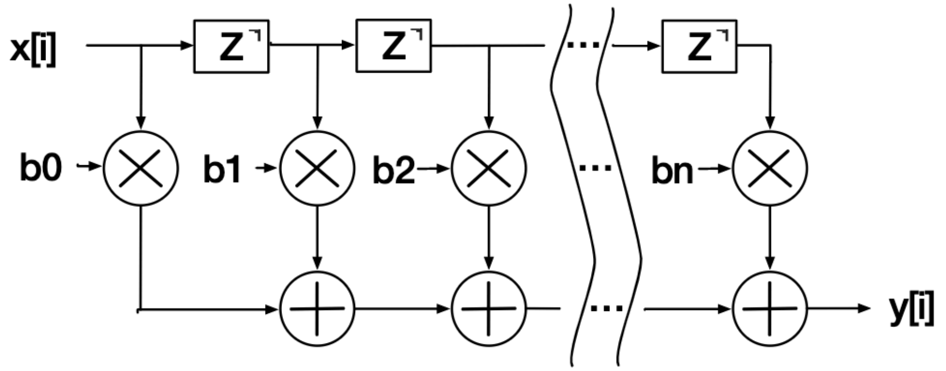
A multiplexer is part of the Chisel library:

```
val result = Mux(sel, a, b)
```

https://www.chisel-lang.org/

Chisel example

- Example of FIRfilter



While Chisel provides similar base primitives as synthesizable Verilog, and *could* be used as such:

```scala
// 3-point moving average implemented in the style of a FIR filter
class MovingAverage3(bitWidth: Int) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(bitWidth.W))
    val out = Output(UInt(bitWidth.W))
  })

  val z1 = RegNext(io.in)
  val z2 = RegNext(z1)

  io.out := (io.in * 1.U) + (z1 * 1.U) + (z2 * 1.U)
}
```

i/o specification

Sequential part

Combinational part

The above can be converted to Verilog using `ChiselStage`:

```scala
import chisel3.stage.{ChiselStage, ChiselGeneratorAnnotation}

(new chisel3.stage.ChiselStage).execute(
  Array("-X", "verilog"),
  Seq(ChiselGeneratorAnnotation(() => new FirFilter(8, Seq(1.U, 1.U, 1.U)))))
```

Alternatively, you may generate some Verilog directly for inspection:

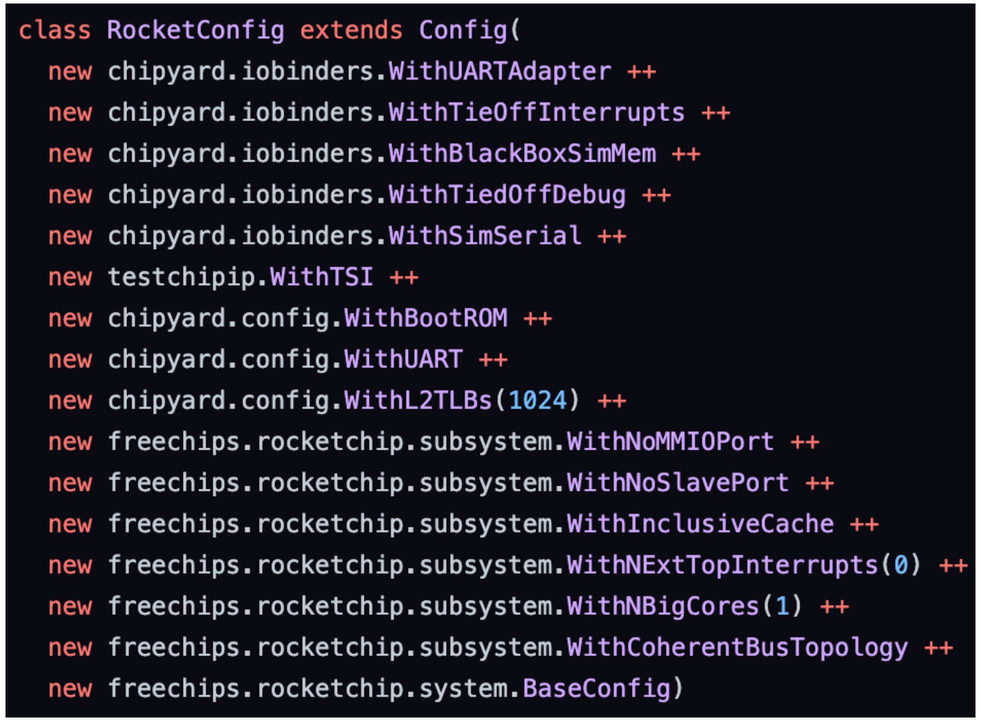
```scala
val verilogString = (new chisel3.stage.ChiselStage).emitVerilog(new FirFilter(8, Seq(0.U, 1.U)))
println(verilogString)
```

Generate Verilog file

https://mybinder.org/v2/gh/freechipsproject/chisel-bootcamp/master

Rocket-chip

- Rocket chip generator is an SoC generator developed at Berkeley and now supported by SiFive
- High extensibility with easy configuration



```
class RocketConfig extends Config(
  new chipyard.iobinders.WithUARTAdapter ++
  new chipyard.iobinders.WithTieOffInterrupts ++
  new chipyard.iobinders.WithBlackBoxSimMem ++
  new chipyard.iobinders.WithTiedOffDebug ++
  new chipyard.iobinders.WithSimSerial ++
  new testchipip.WithTSI ++
  new chipyard.config.WithBootROM ++
  new chipyard.config.WithUART ++
  new chipyard.config.WithL2TLBs(1024) ++
  new freechips.rocketchip.subsystem.WithNoMMIOPort ++
  new freechips.rocketchip.subsystem.WithNoSlavePort ++
  new freechips.rocketchip.subsystem.WithInclusiveCache ++
  new freechips.rocketchip.subsystem.WithNExtTopInterrupts(0) ++
  new freechips.rocketchip.subsystem.WithNBigCores(1) ++
  new freechips.rocketchip.subsystem.WithCoherentBusTopology ++
  new freechips.rocketchip.system.BaseConfig)
```

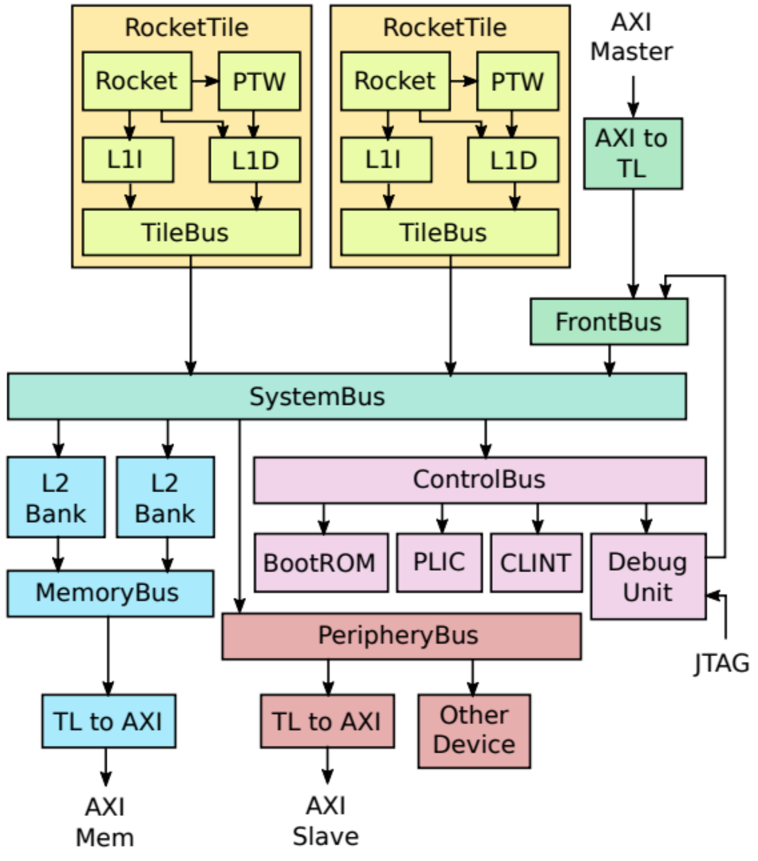
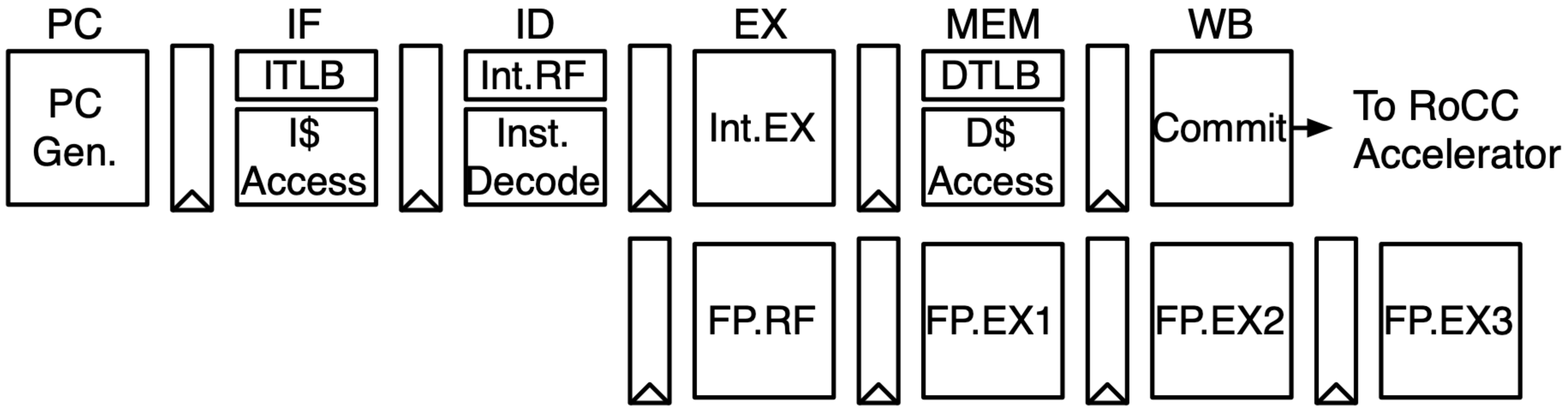


Figure 2: A generic Rocket Chip instance

https://github.com/chipsalliance/rocket-chip

Rocket-core

- Default rocket core is a 5-stage single-issue in-order pipeline
- 64-entry BTB, 256-entry BHT, 2-entry RAS
- MMU supports page-based virtual memory
- IEEE 754-2008-compliant FPU



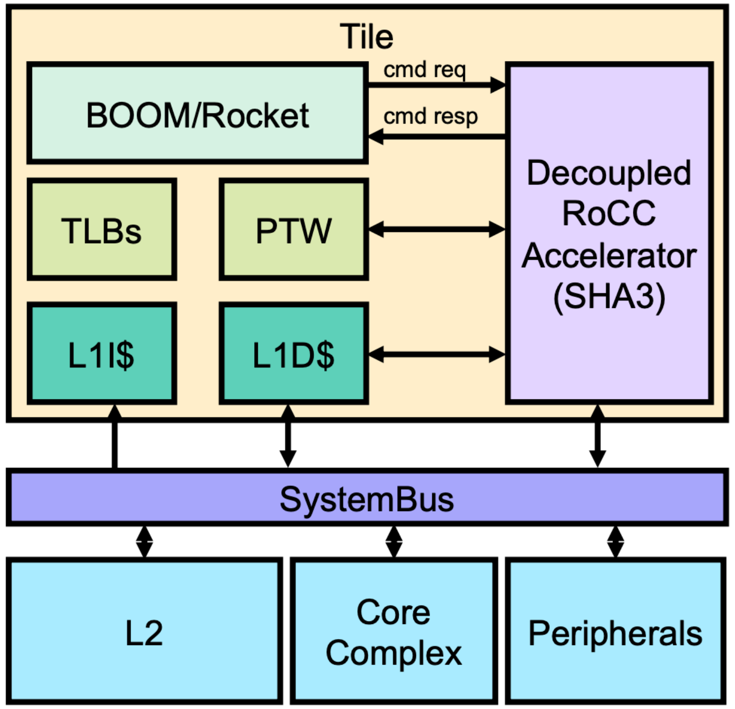
https://riscv.org/wp-content/uploads/2015/01/riscv-rocket-chip-generator-workshop-jan2015.pdf

Chipyard

- Integrate lots of packages, including rocket-chip, firrtl, sifive-cache, gemmini, nvdla, sha3, etc.

https://chipyard.readthedocs.io/en/latest/Chipyard-Basics/index.html

Add a sha3 coprocessor



Execute custom RISC-V instructions for a custom extension

31 25	24 20	19 15	14	13	12	11 7	6 0
funct	rs2	rs1	xd	xs1	xs2	rd	opcode
7	5	5	1	1	1	5	7

```
class TutorialRocketConfig extends Config(
  new WithTop ++
  new WithBootROM ++
  new freechips.rocketchip.subsystem.WithInclusiveCache ++
  new freechips.rocketchip.subsystem.WithNBigCores(1) ++
  new freechips.rocketchip.system.BaseConfig)
```

Modify and add

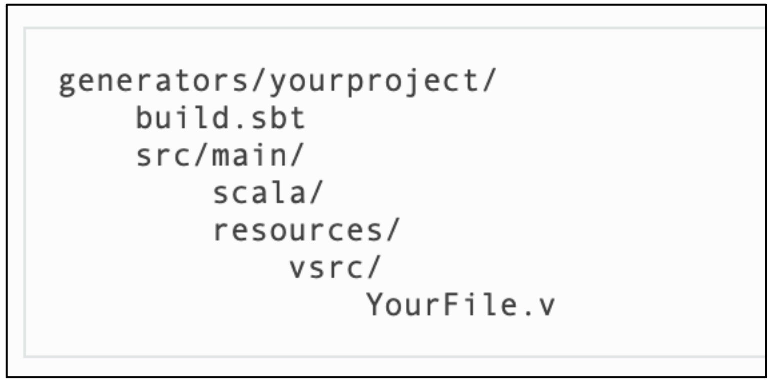
```
class Sha3RocketConfig extends Config(
  new WithTop ++
  new WithBootROM ++
  new freechips.rocketchip.subsystem.WithInclusiveCache ++
  new sha3.WithSha3Accel ++
  new freechips.rocketchip.subsystem.WithNBigCores(1) ++
  new freechips.rocketchip.system.BaseConfig)
```

Expanded C macro

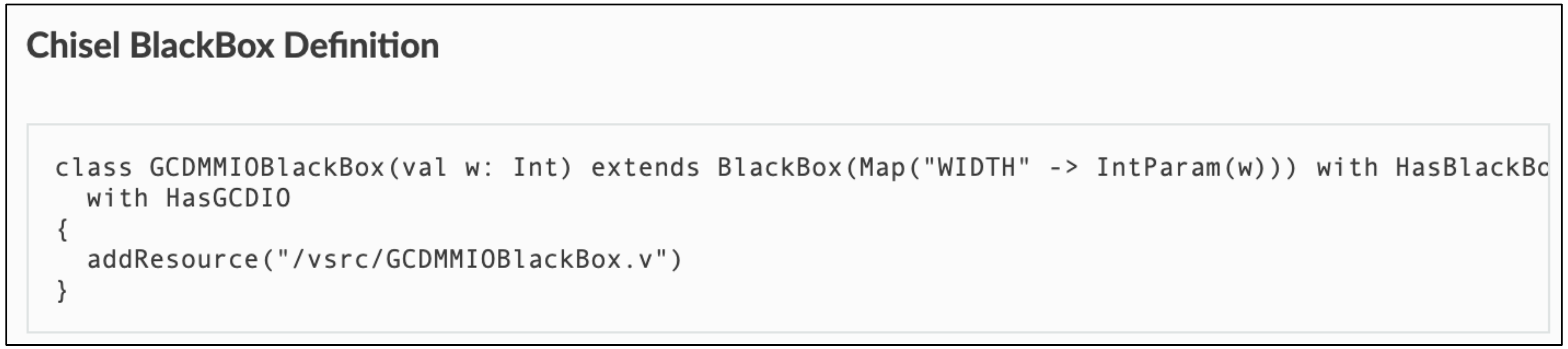
```
                     opcode   rd   rs1              rs2             funct
asm volatile ("custom2 x0, %[msg_addr], %[hash_addr], 0"
        :: [msg_addr] "r" (&input), [hash_addr] "r" (&output));
```

https://fires.im/micro19-slides-pdf/03_building_custom_socs.pdf

Incorporating verilog blocks

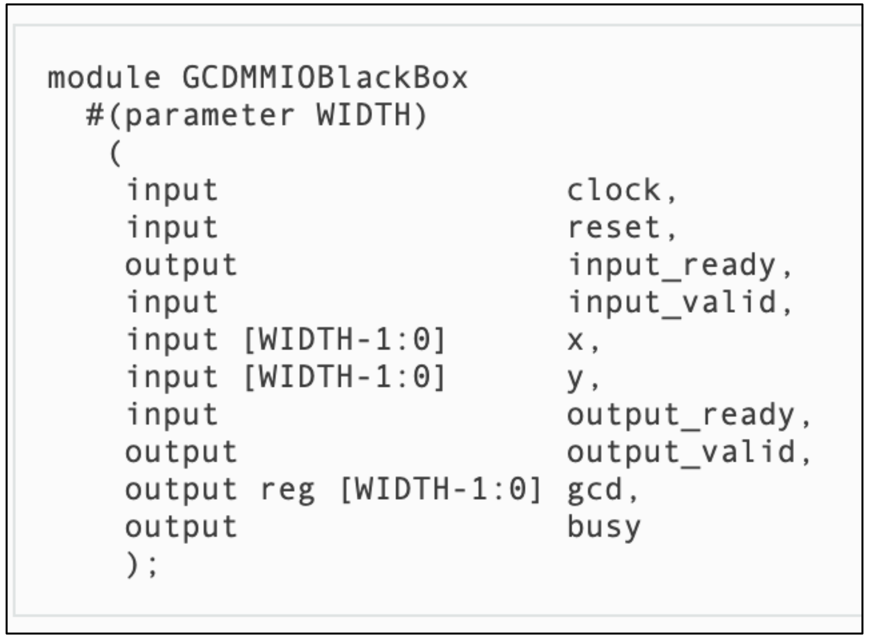


```
generators/yourproject/
    build.sbt
    src/main/
        scala/
        resources/
            vsrc/
                YourFile.v
```

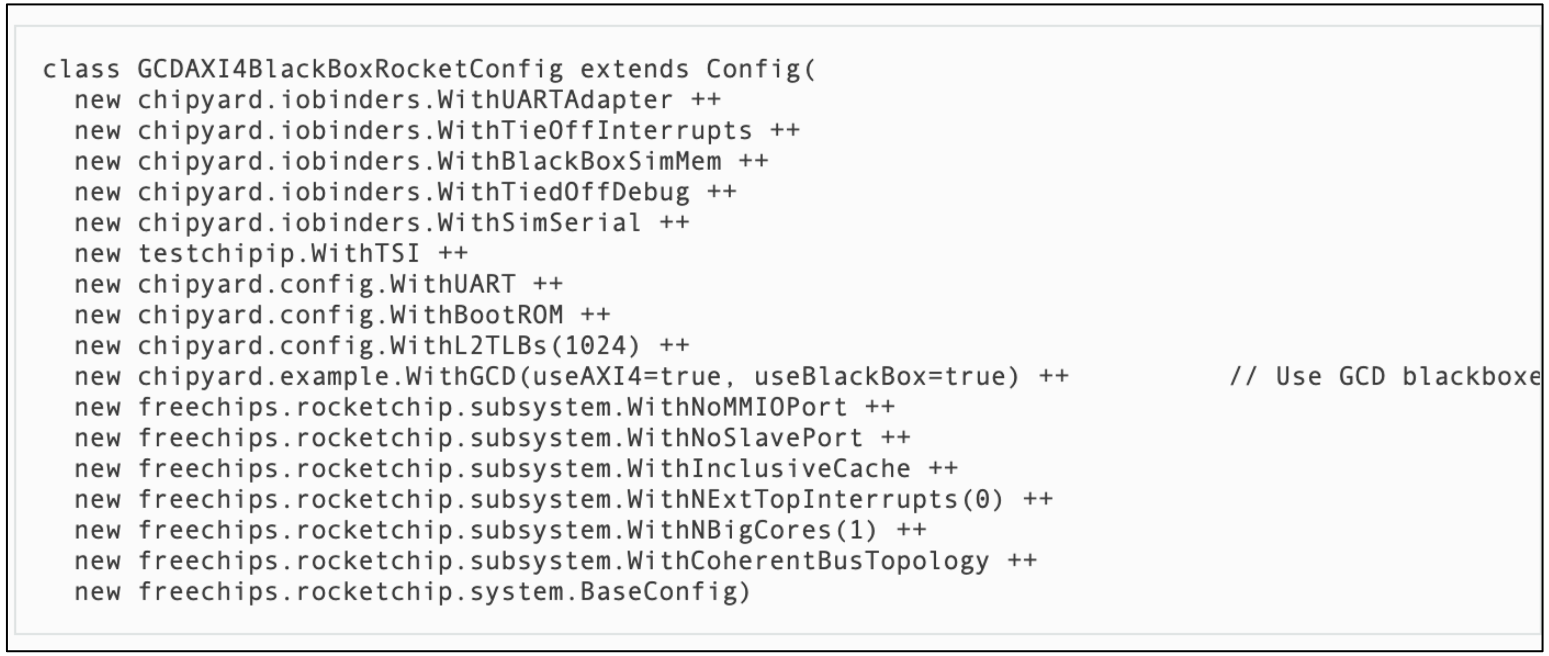


Chisel BlackBox Definition

```
class GCDMMIOBlackBox(val w: Int) extends BlackBox(Map("WIDTH" -> IntParam(w))) with HasBlackBc
  with HasGCDIO
{
  addResource("/vsrc/GCDMMIOBlackBox.v")
}
```



```
module GCDMMIOBlackBox
  #(parameter WIDTH)
   (
    input                 clock,
    input                 reset,
    output                input_ready,
    input                 input_valid,
    input [WIDTH-1:0]     x,
    input [WIDTH-1:0]     y,
    input                 output_ready,
    output                output_valid,
    output reg [WIDTH-1:0] gcd,
    output                busy
    );
```



```
class GCDAXI4BlackBoxRocketConfig extends Config(
  new chipyard.iobinders.WithUARTAdapter ++
  new chipyard.iobinders.WithTieOffInterrupts ++
  new chipyard.iobinders.WithBlackBoxSimMem ++
  new chipyard.iobinders.WithTiedOffDebug ++
  new chipyard.iobinders.WithSimSerial ++
  new testchipip.WithTSI ++
  new chipyard.config.WithUART ++
  new chipyard.config.WithBootROM ++
  new chipyard.config.WithL2TLBs(1024) ++
  new chipyard.example.WithGCD(useAXI4=true, useBlackBox=true) ++          // Use GCD blackboxe
  new freechips.rocketchip.subsystem.WithNoMMIOPort ++
  new freechips.rocketchip.subsystem.WithNoSlavePort ++
  new freechips.rocketchip.subsystem.WithInclusiveCache ++
  new freechips.rocketchip.subsystem.WithNExtTopInterrupts(0) ++
  new freechips.rocketchip.subsystem.WithNBigCores(1) ++
  new freechips.rocketchip.subsystem.WithCoherentBusTopology ++
  new freechips.rocketchip.system.BaseConfig)
```

https://chipyard.readthedocs.io/en/latest/Customization/Incorporating-Verilog-Blocks.html

Riscv-tests

- Contain ISA test (privileged and non-privileged) and some small benchmark tests
- Small benchmark tests
  - Qsort
  - Rsort
  - Dhrystone
  - Median
  - Tower
  - Mm
  - Multiply
  - Mt-matmul
  - …

Outline

- Tool
  - Chisel
  - Rocket-chip
  - Chipyard
  - Riscv-tests
- HW5

Memory hierarchy

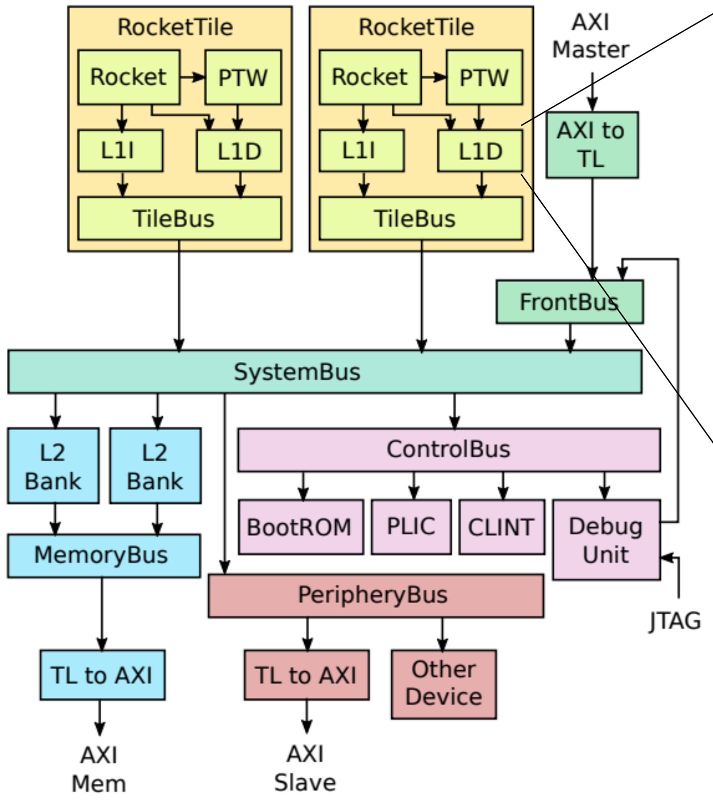
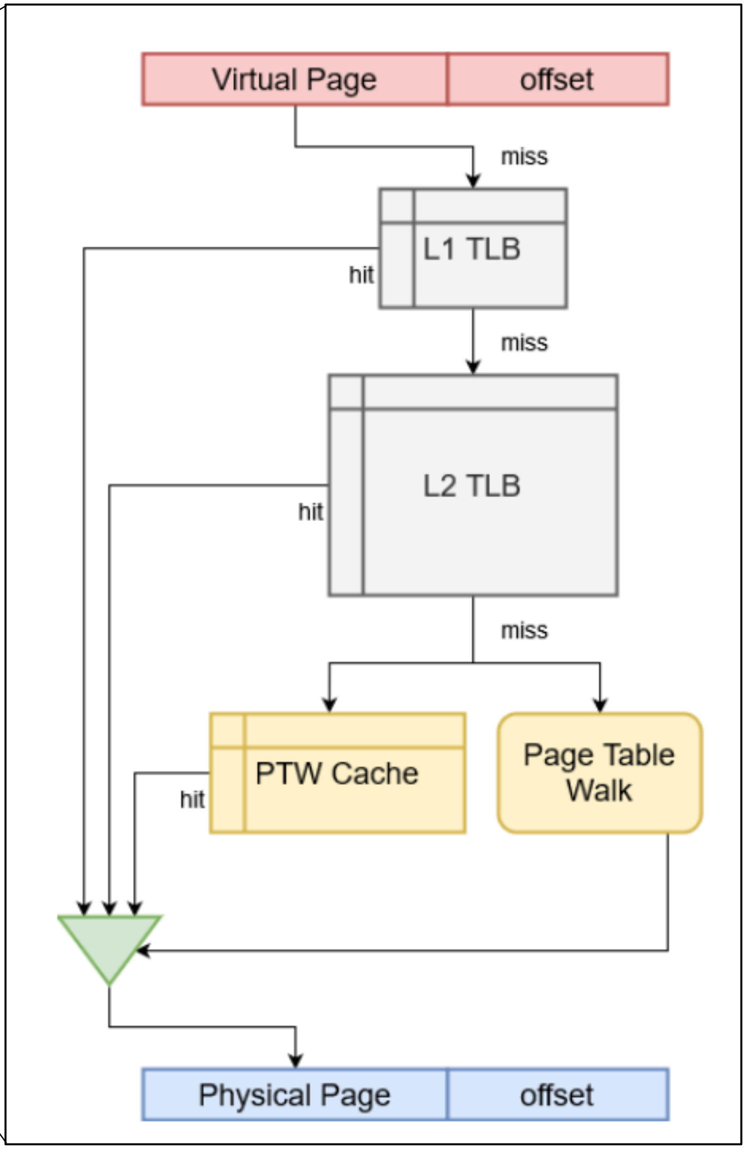


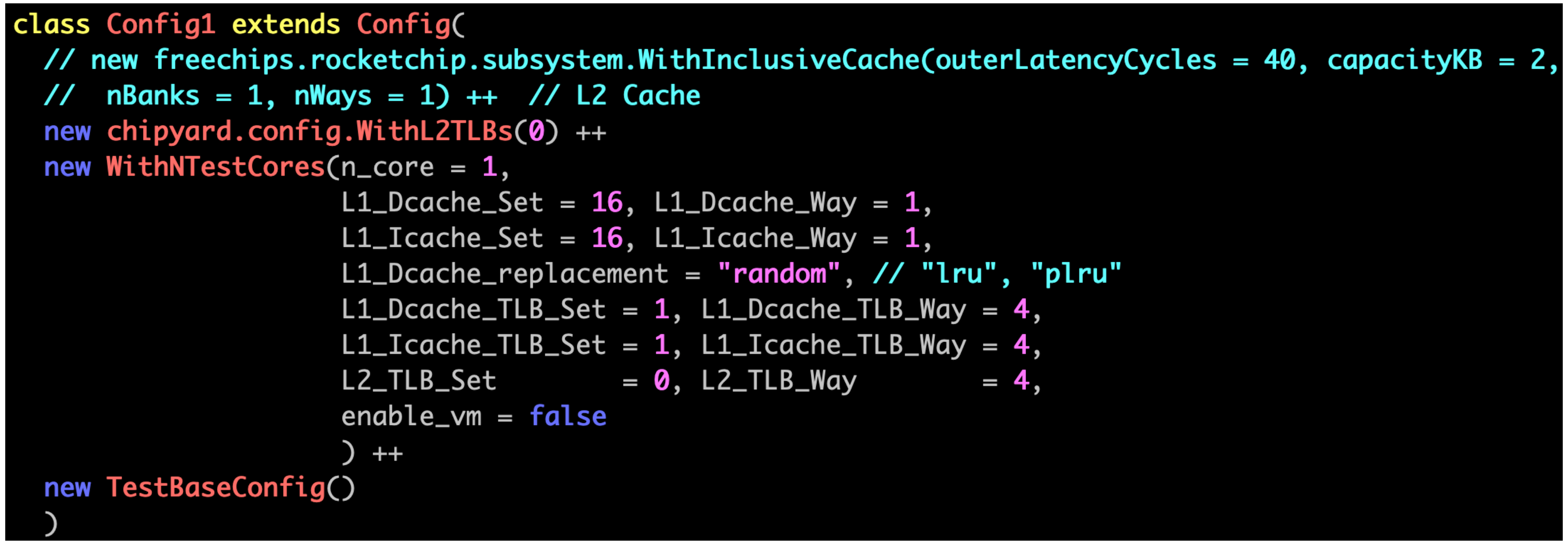
Figure 2: A generic Rocket Chip instance



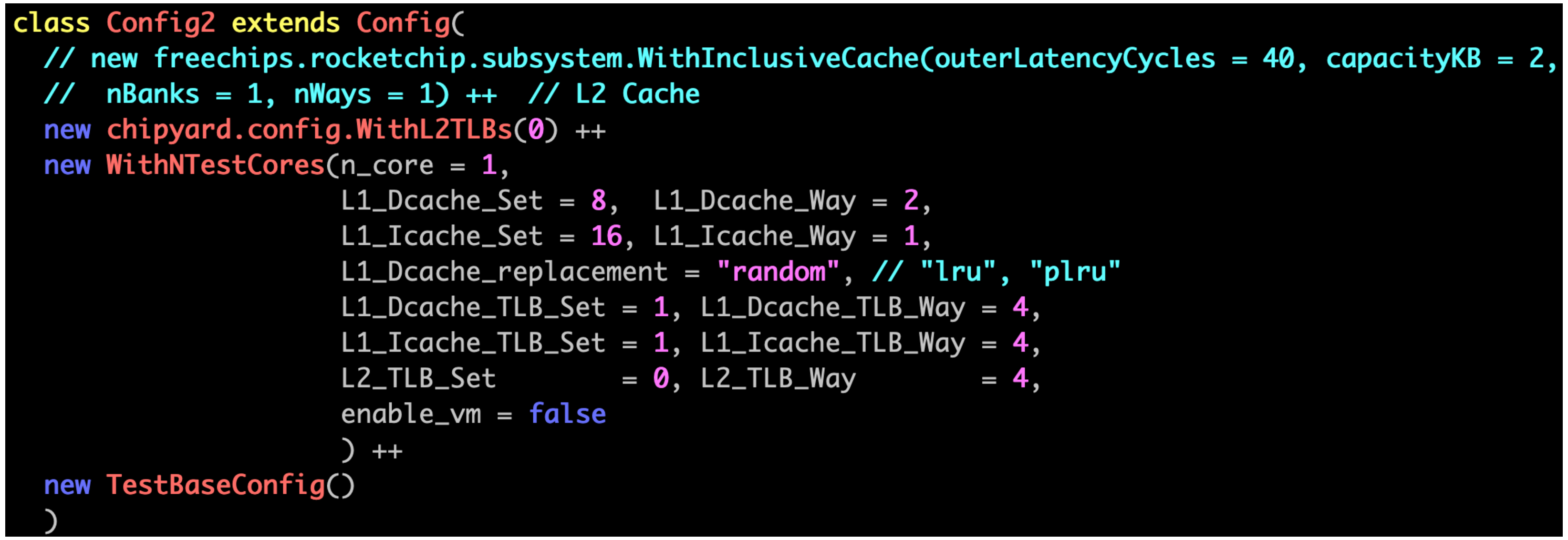
https://arxiv.org/pdf/2009.07723.pdf

Configurable cache SoC

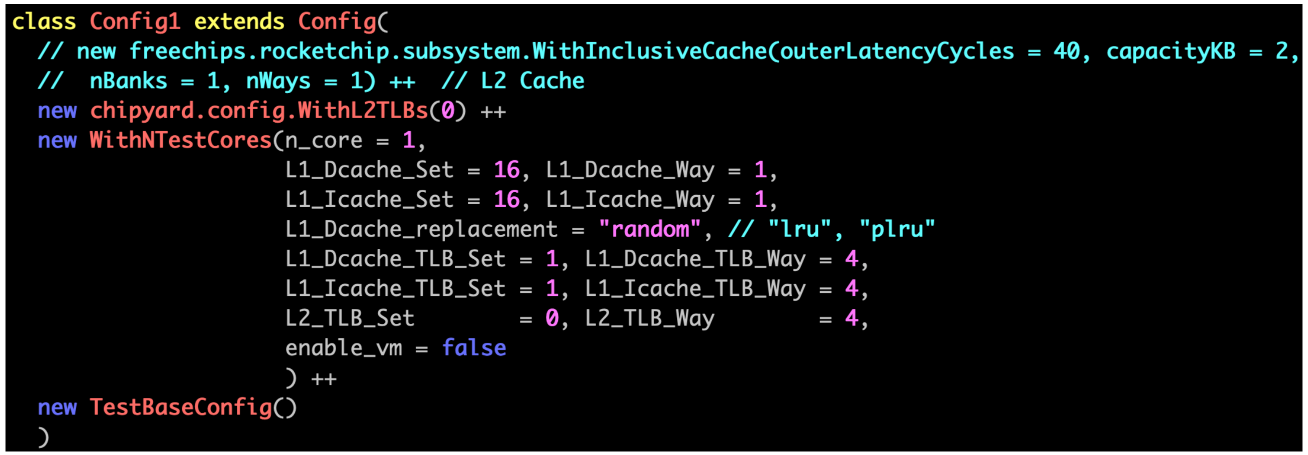
/root/emulator/system/test.scala



```
class Config1 extends Config(
  // new freechips.rocketchip.subsystem.WithInclusiveCache(outerLatencyCycles = 40, capacityKB = 2,
  //  nBanks = 1, nWays = 1) ++  // L2 Cache
  new chipyard.config.WithL2TLBs(0) ++
  new WithNTestCores(n_core = 1,
                     L1_Dcache_Set = 16, L1_Dcache_Way = 1,
                     L1_Icache_Set = 16, L1_Icache_Way = 1,
                     L1_Dcache_replacement = "random", // "lru", "plru"
                     L1_Dcache_TLB_Set = 1, L1_Dcache_TLB_Way = 4,
                     L1_Icache_TLB_Set = 1, L1_Icache_TLB_Way = 4,
                     L2_TLB_Set        = 0, L2_TLB_Way        = 4,
                     enable_vm = false
                     ) ++
  new TestBaseConfig()
  )
```



```
class Config2 extends Config(
  // new freechips.rocketchip.subsystem.WithInclusiveCache(outerLatencyCycles = 40, capacityKB = 2,
  //  nBanks = 1, nWays = 1) ++  // L2 Cache
  new chipyard.config.WithL2TLBs(0) ++
  new WithNTestCores(n_core = 1,
                     L1_Dcache_Set = 8,  L1_Dcache_Way = 2,
                     L1_Icache_Set = 16, L1_Icache_Way = 1,
                     L1_Dcache_replacement = "random", // "lru", "plru"
                     L1_Dcache_TLB_Set = 1, L1_Dcache_TLB_Way = 4,
                     L1_Icache_TLB_Set = 1, L1_Icache_TLB_Way = 4,
                     L2_TLB_Set        = 0, L2_TLB_Way        = 4,
                     enable_vm = false
                     ) ++
  new TestBaseConfig()
  )
```



```
class Config1 extends Config(
  // new freechips.rocketchip.subsystem.WithInclusiveCache(outerLatencyCycles = 40, capacityKB = 2,
  //  nBanks = 1, nWays = 1) ++  // L2 Cache
  new chipyard.config.WithL2TLBs(0) ++
  new WithNTestCores(n_core = 1,
                     L1_Dcache_Set = 16, L1_Dcache_Way = 1,
                     L1_Icache_Set = 16, L1_Icache_Way = 1,
                     L1_Dcache_replacement = "random", // "lru", "plru"
                     L1_Dcache_TLB_Set = 1, L1_Dcache_TLB_Way = 4,
                     L1_Icache_TLB_Set = 1, L1_Icache_TLB_Way = 4,
                     L2_TLB_Set        = 0, L2_TLB_Way        = 4,
                     enable_vm = false
                     ) ++
  new TestBaseConfig()
  )
```

Some constraints

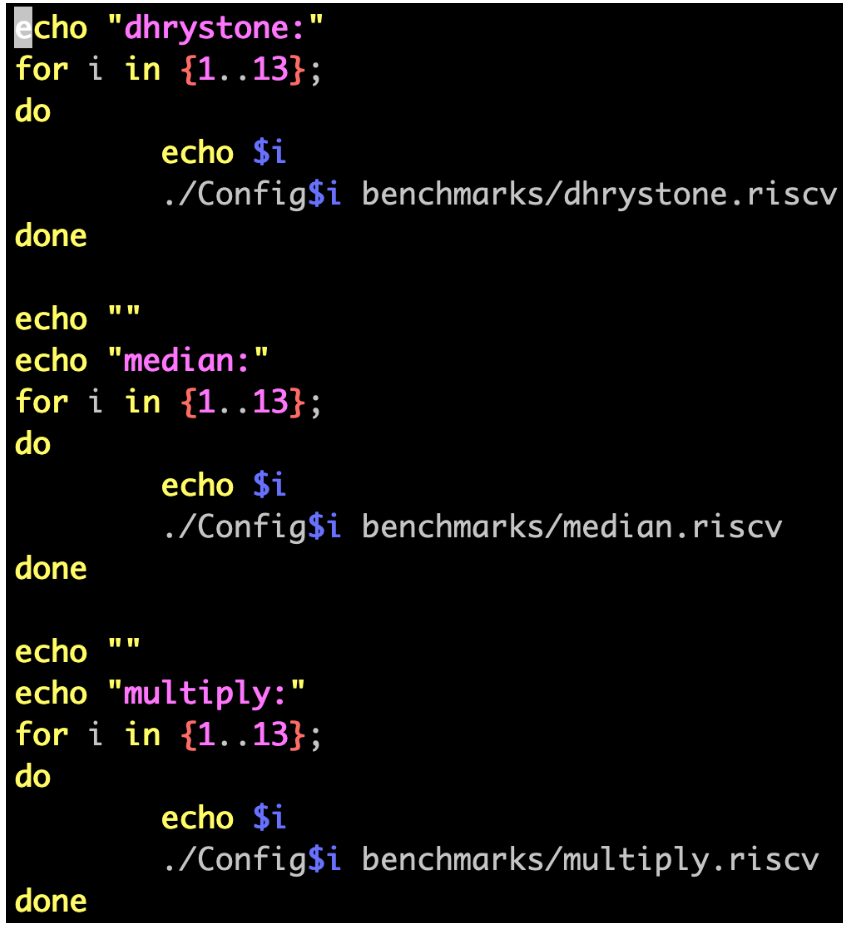
- Block size (L1$,L2$): 64 bytes
  - Not configurable, related to bus width
- L2 inclusive cache at least 2 ways
  - nBanks*nWays*nSets*64 = capacity
- L1, L2 TLB at least 4 ways
  - May not be used in this homework because of the small size of benchmarks
- L2_TLB_Set * L2_TLB_Way = chipyard.config.WithL2TLBs(…)
  - Should specify both explicitly
- Replacement policy only affects L1D$
- Way, Set, Bank should be 2, 4, 8, 16, …

Some other things

- If L2$ is not enabled, when L1 misses, the outer access latency is determined by bus width and L1 cache blocks, and hence, it is determined by the number of beats which is 8 cycles.
- If L2$ is enabled, L1 to L2 is still 8 cycles (perhaps), and L2 to outer memory is 40 cycles (by default).
- So when L2 is enabled, the cycle count increases but with more realistic modeling.

Part1

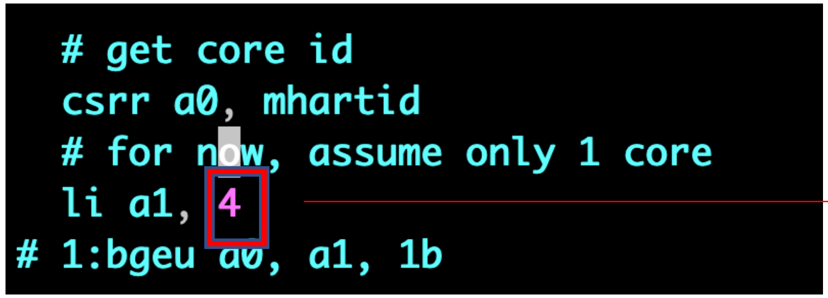
- You can collect part1 info by running ./test.sh



```
echo "dhrystone:"
for i in {1..13};
do
        echo $i
        ./Config$i benchmarks/dhrystone.riscv
done

echo ""
echo "median:"
for i in {1..13};
do
        echo $i
        ./Config$i benchmarks/median.riscv
done

echo ""
echo "multiply:"
for i in {1..13};
do
        echo $i
        ./Config$i benchmarks/multiply.riscv
done
```

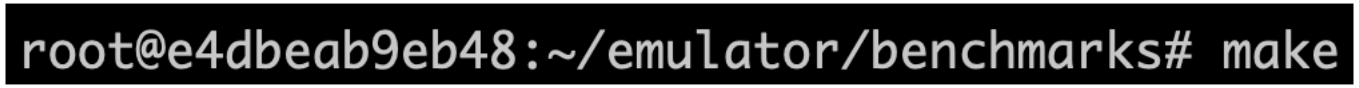
Part1

- As for 1, 2, 4 cores setting, you should change the number of the available cores in /root/emulator/benchmarks/common/crt.S and recompile the program



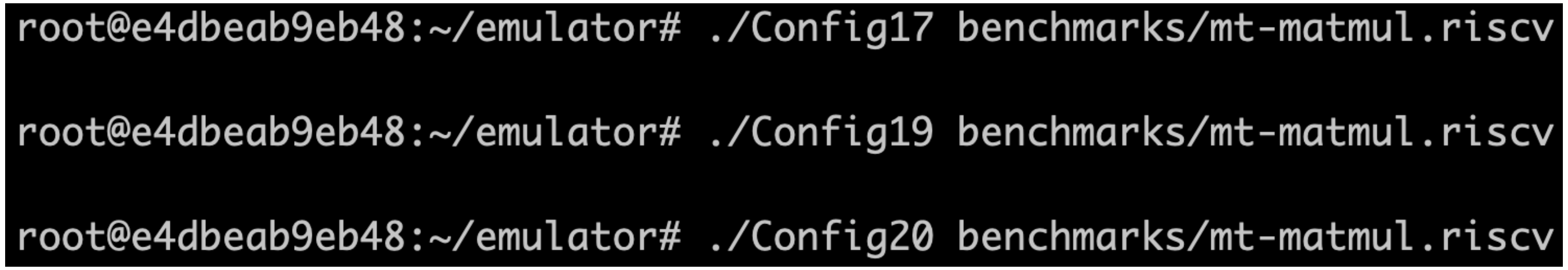
```
  # get core id
  csrr a0, mhartid
  # for now, assume only 1 core
  li a1, 4
# 1:bgeu a0, a1, 1b
```

1, 2, 4



```
root@e4dbeab9eb48:~/emulator/benchmarks# make
```

Recompile the program (It will recompile not only mt-matmul, but the whole benchmarks)



```
root@e4dbeab9eb48:~/emulator# ./Config17 benchmarks/mt-matmul.riscv

root@e4dbeab9eb48:~/emulator# ./Config19 benchmarks/mt-matmul.riscv

root@e4dbeab9eb48:~/emulator# ./Config20 benchmarks/mt-matmul.riscv
```

1

2

4

Part2

L1 I$ D$ available settings:

(set, way)
(16, 1)
(8, 2)
(4, 4)
(2, 8)

L1 D$ replacement available settings:

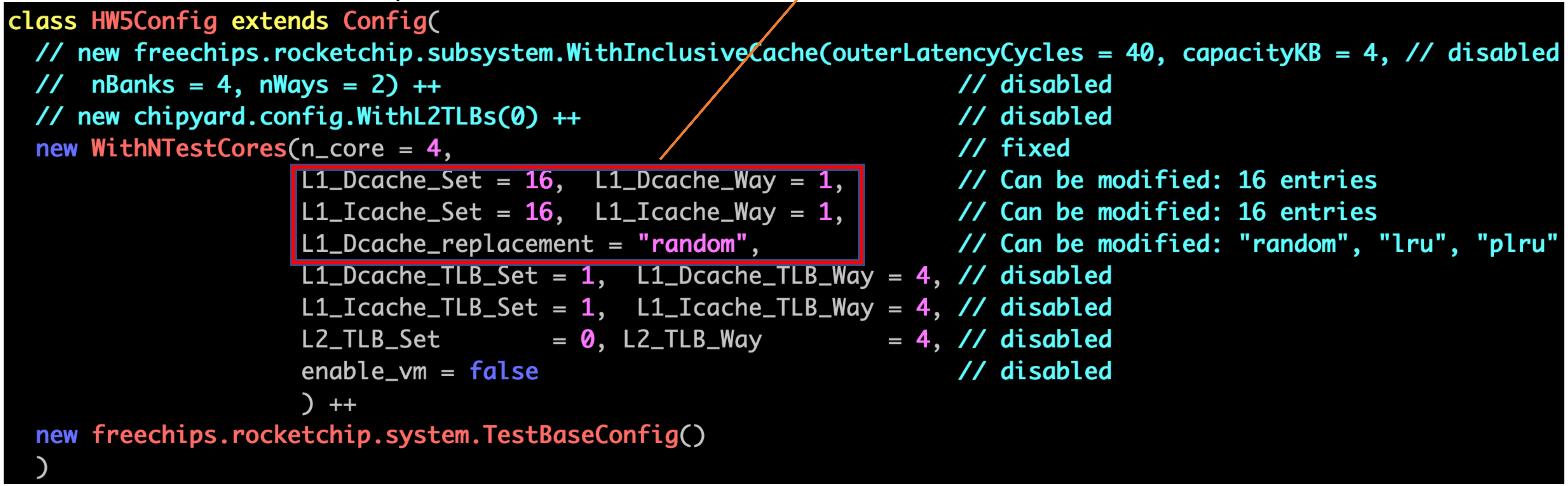
random
lru
plru

Some combinations may not work !!

- Configure the HW5.scala file
- To keep it simple, only L1 I$ and L1 D$ will be configured

Can be modified

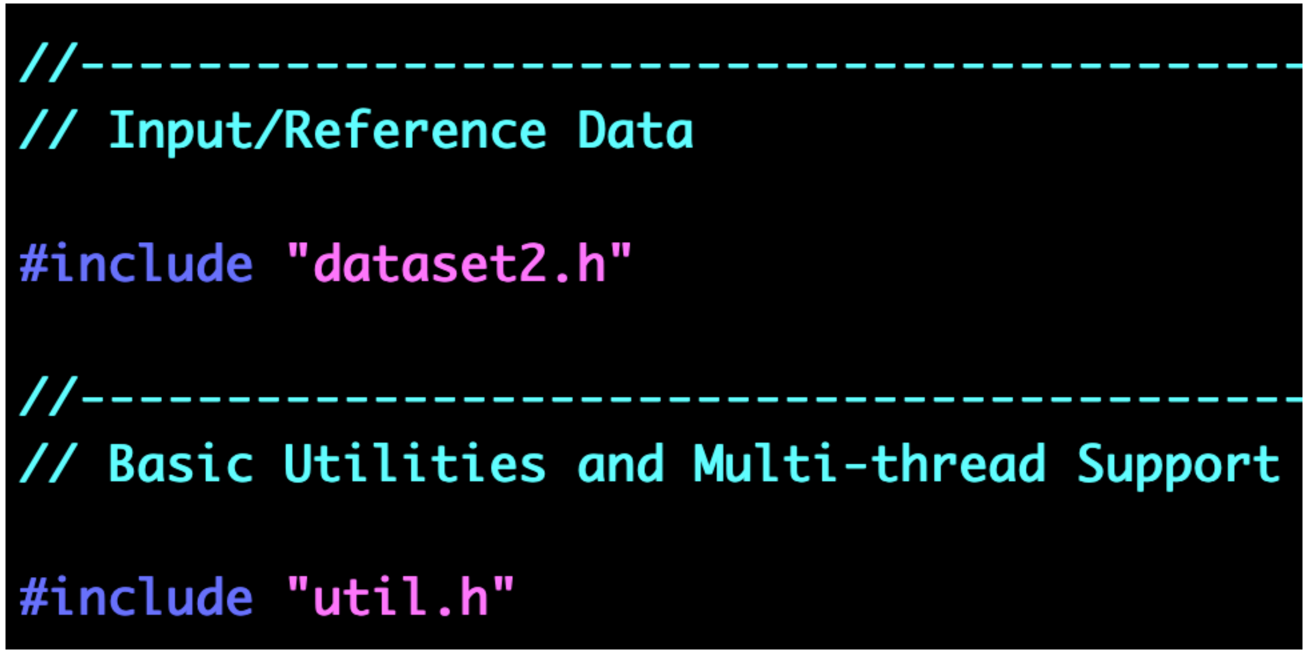
/root/emulator/system/HW5.scala



```
class HW5Config extends Config(
  // new freechips.rocketchip.subsystem.WithInclusiveCache(outerLatencyCycles = 40, capacityKB = 4, // disabled
  //  nBanks = 4, nWays = 2) ++                                     // disabled
  // new chipyard.config.WithL2TLBs(0) ++                           // disabled
  new WithNTestCores(n_core = 4,                                    // fixed
                     L1_Dcache_Set = 16,  L1_Dcache_Way = 1,        // Can be modified: 16 entries
                     L1_Icache_Set = 16,  L1_Icache_Way = 1,        // Can be modified: 16 entries
                     L1_Dcache_replacement = "random",              // Can be modified: "random", "lru", "plru"
                     L1_Dcache_TLB_Set = 1,  L1_Dcache_TLB_Way = 4, // disabled
                     L1_Icache_TLB_Set = 1,  L1_Icache_TLB_Way = 4, // disabled
                     L2_TLB_Set        = 0,  L2_TLB_Way        = 4, // disabled
                     enable_vm = false                              // disabled
                     ) ++
  new freechips.rocketchip.system.TestBaseConfig()
 )
```

Change the testdata to 64x64 in mt_matmul.c

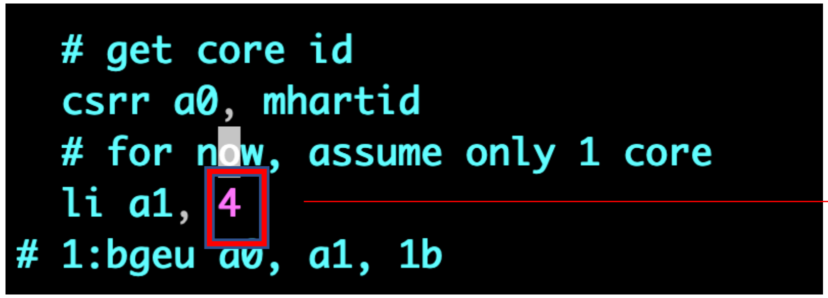
/root/emulator/benchmarks/mt-matmul/mt_matmul.c



```
//--------------------------------------
// Input/Reference Data

#include "dataset2.h"

//--------------------------------------
// Basic Utilities and Multi-thread Support

#include "util.h"
```

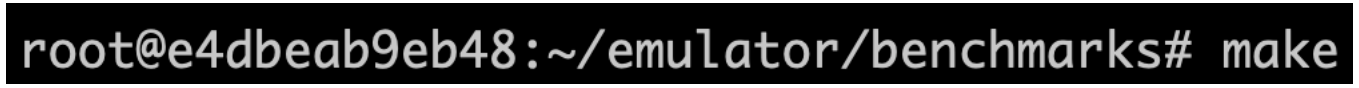
dataset.h → 32x32
dataset2.h → 64x64

Also change number of cores to 4



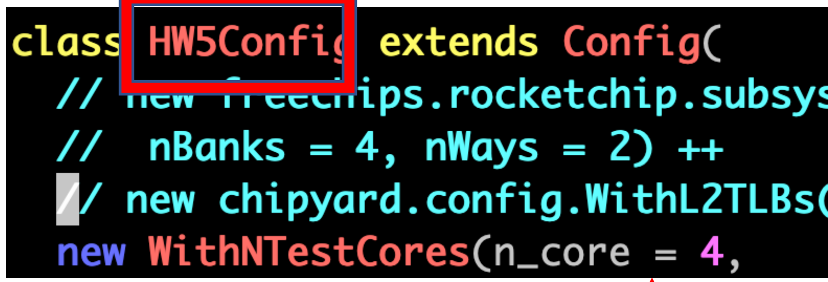
```
  # get core id
  csrr a0, mhartid
  # for now, assume only 1 core
  li a1, 4
# 1:bgeu a0, a1, 1b
```

4



```
root@e4dbeab9eb48:~/emulator/benchmarks# make
```

compile

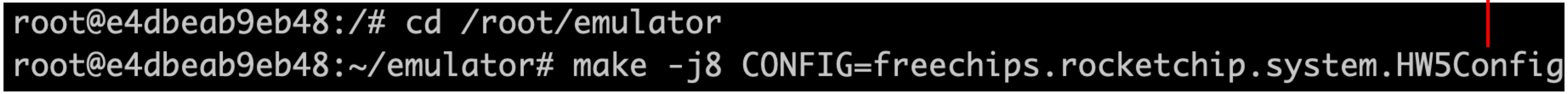


```
class HW5Config extends Config(
  // new freechips.rocketchip.subsys
  //   nBanks = 4, nWays = 2) ++
  // new chipyard.config.WithL2TLBs(
  new WithNTestCores(n_core = 4,
```

Class name

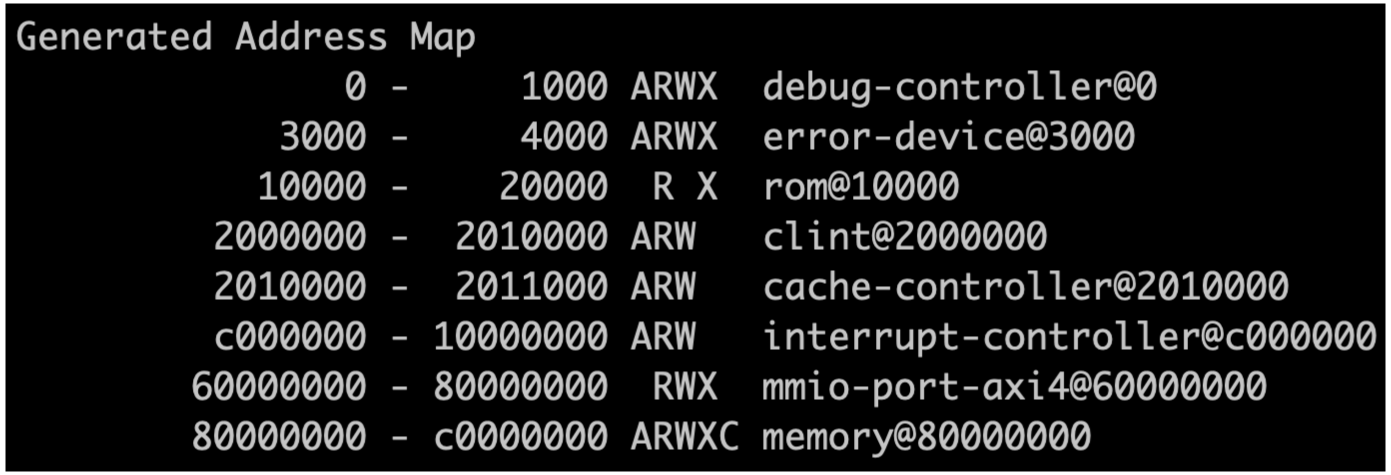
Compile the target configuration

rm -rf generated-src (if you build the emulator before)



```
root@e4dbeab9eb48:/# cd /root/emulator
root@e4dbeab9eb48:~/emulator# make -j8 CONFIG=freechips.rocketchip.system.HW5Config
```

- We can see the address mapping

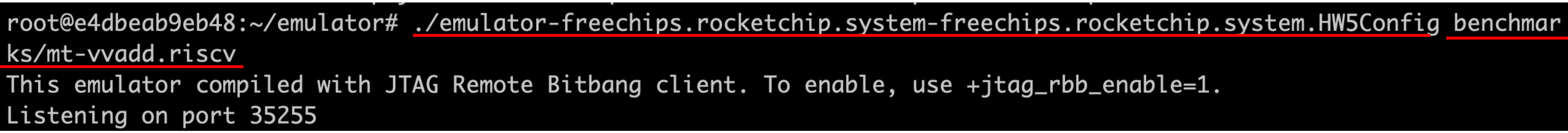


```
Generated Address Map
               0 -     1000 ARWX  debug-controller@0
            3000 -     4000 ARWX  error-device@3000
           10000 -    20000  R X  rom@10000
         2000000 -  2010000 ARW   clint@2000000
         2010000 -  2011000 ARW   cache-controller@2010000
         c000000 - 10000000 ARW   interrupt-controller@c000000
        60000000 - 80000000  RWX  mmio-port-axi4@60000000
        80000000 - c0000000 ARWXC memory@80000000
```

Run the test

Generated binary

Test program

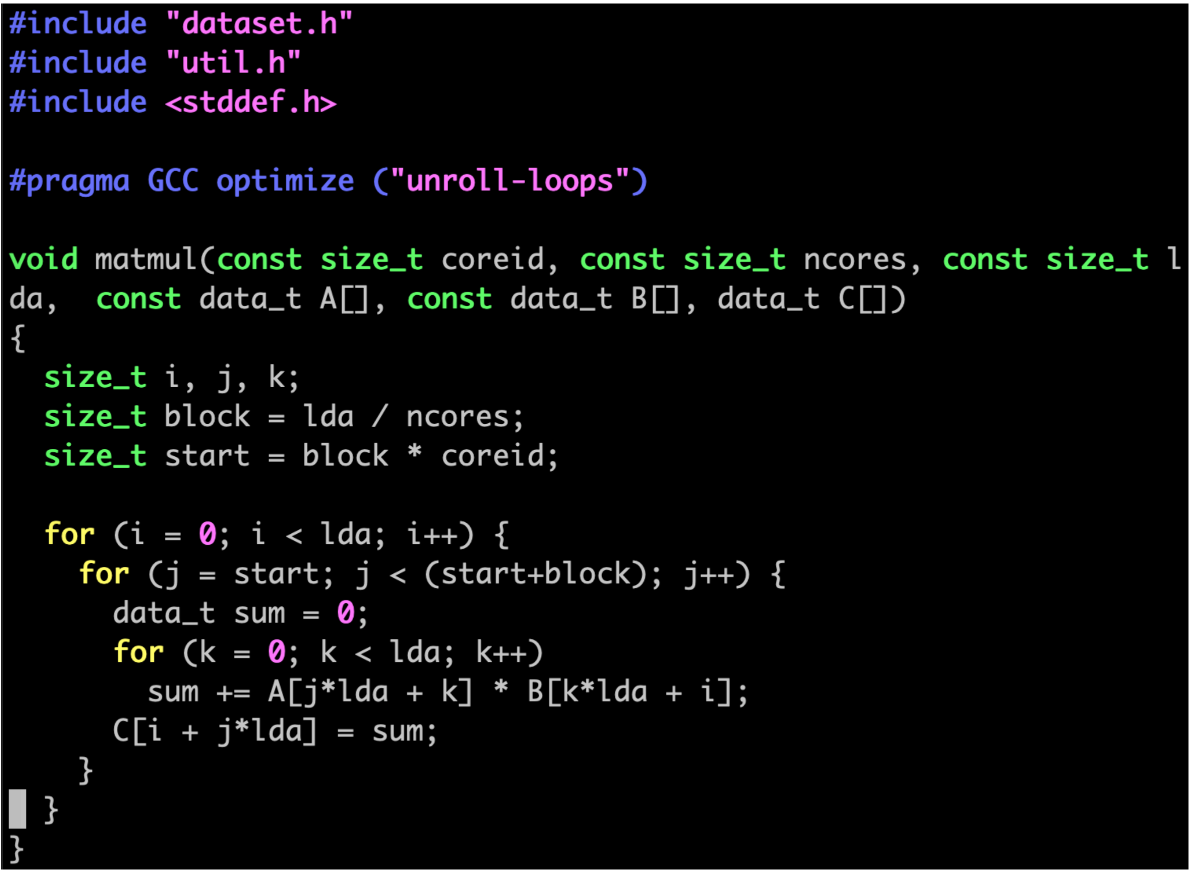


```
root@e4dbeab9eb48:~/emulator# ./emulator-freechips.rocketchip.system-freechips.rocketchip.system.HW5Config benchmar
ks/mt-vvadd.riscv
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb_enable=1.
Listening on port 35255
```

mt-matmul

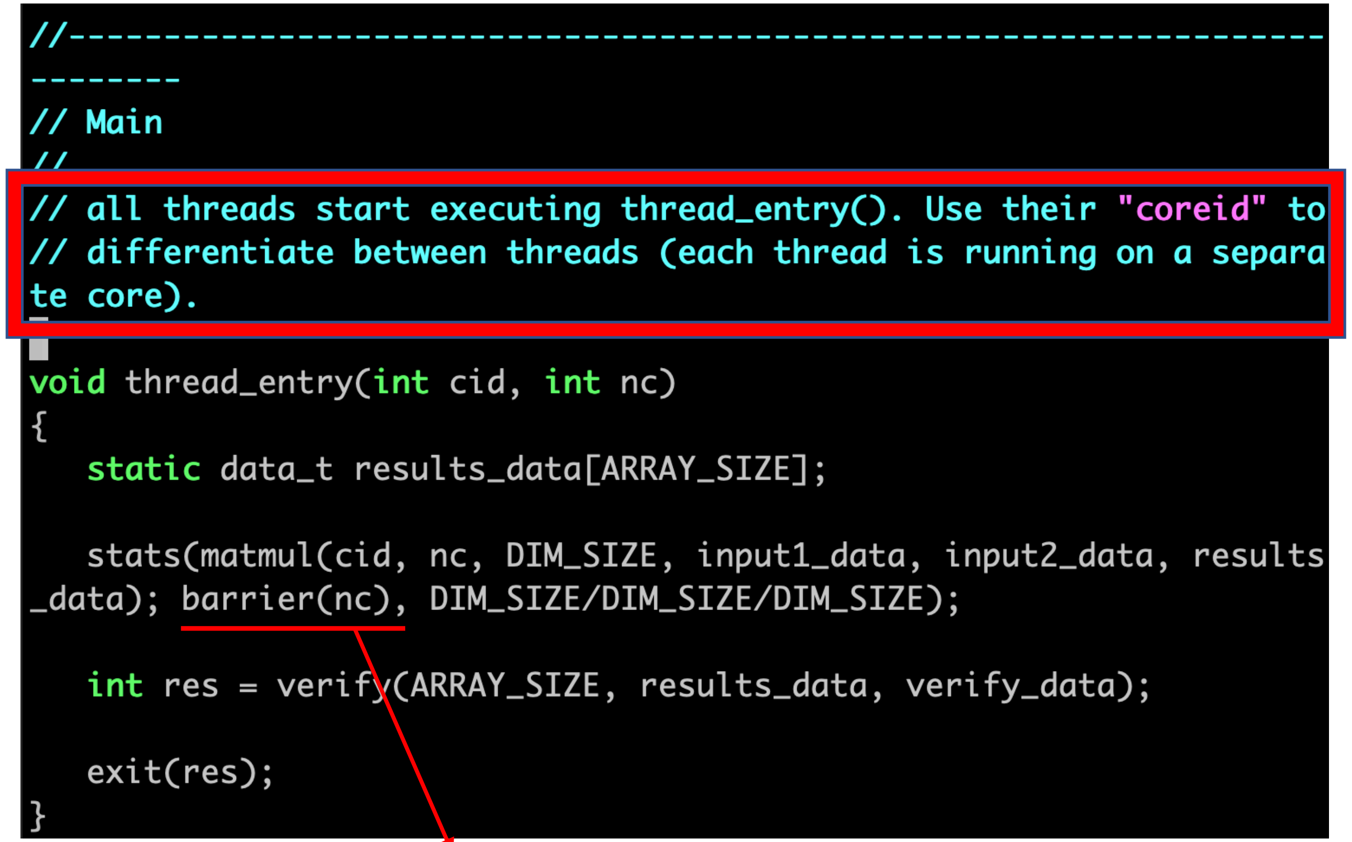
/root/emulator/benchmarks/mt-matmul/

matmul.c



```
#include "dataset.h"
#include "util.h"
#include <stddef.h>

#pragma GCC optimize ("unroll-loops")

void matmul(const size_t coreid, const size_t ncores, const size_t l
da,  const data_t A[], const data_t B[], data_t C[])
{
  size_t i, j, k;
  size_t block = lda / ncores;
  size_t start = block * coreid;

  for (i = 0; i < lda; i++) {
    for (j = start; j < (start+block); j++) {
      data_t sum = 0;
      for (k = 0; k < lda; k++)
        sum += A[j*lda + k] * B[k*lda + i];
      C[i + j*lda] = sum;
    }
  }
}
```

mt-matmul.c



```
//-----------------------------------------------------------------
--------
// Main
//
// all threads start executing thread_entry(). Use their "coreid" to
// differentiate between threads (each thread is running on a separa
te core).

void thread_entry(int cid, int nc)
{
   static data_t results_data[ARRAY_SIZE];

   stats(matmul(cid, nc, DIM_SIZE, input1_data, input2_data, results
_data); barrier(nc), DIM_SIZE/DIM_SIZE/DIM_SIZE);

   int res = verify(ARRAY_SIZE, results_data, verify_data);

   exit(res);
}
```

You may use inline assembly if you think compiler is not doing well
https://gcc.gnu.org/onlinedocs/gcc/RISC-V-Options.html

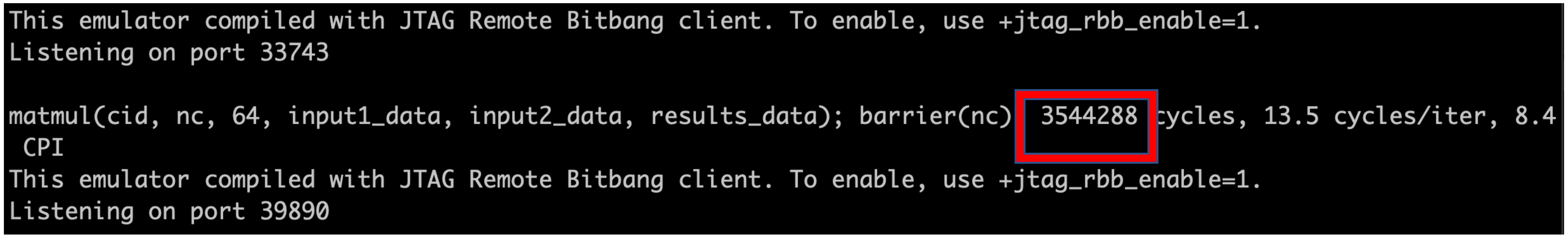
You may also want to use barrier synchronization in your matmul.c when needed.

Evaluation

- We will put your matmul.c in matmul/ and make under benchmark/
- We will also make CONFIG=freechips.rocketchip.system.HW5Config under emulator/
- We will score based on the cycle count
- Correctness is a must

Default mt-matmul (64x64) status

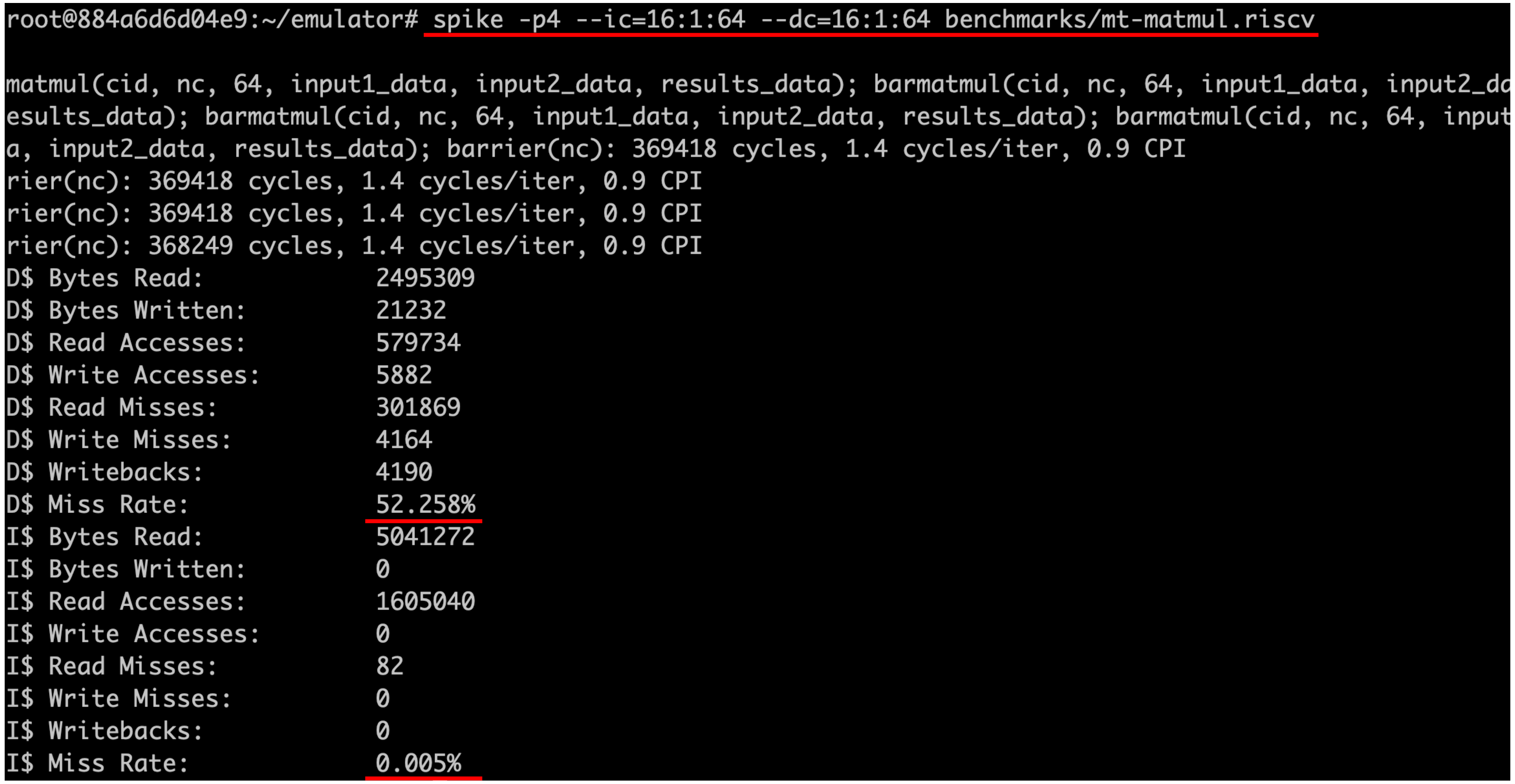
5~7 minutes



```
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb_enable=1.
Listening on port 33743

matmul(cid, nc, 64, input1_data, input2_data, results_data); barrier(nc)  3544288 cycles, 13.5 cycles/iter, 8.4
 CPI
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb_enable=1.
Listening on port 39890
```

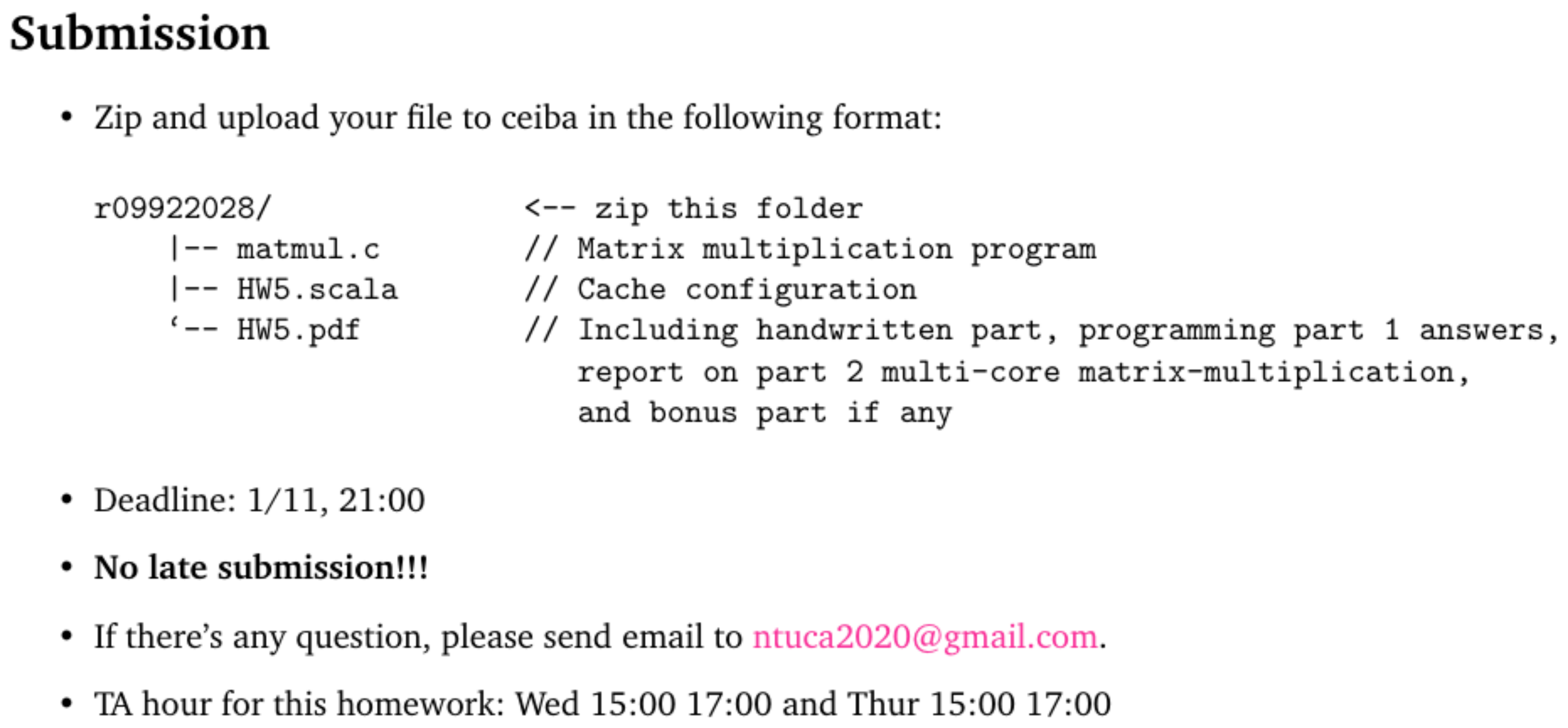
You may want to test using spike before cycle accurate testing

No failed message



```
root@884a6d6d04e9:~/emulator# spike -p4 --ic=16:1:64 --dc=16:1:64 benchmarks/mt-matmul.riscv

matmul(cid, nc, 64, input1_data, input2_data, results_data); barmatmul(cid, nc, 64, input1_data, input2_da
esults_data); barmatmul(cid, nc, 64, input1_data, input2_data, results_data); barmatmul(cid, nc, 64, input
a, input2_data, results_data); barrier(nc): 369418 cycles, 1.4 cycles/iter, 0.9 CPI
rier(nc): 369418 cycles, 1.4 cycles/iter, 0.9 CPI
rier(nc): 369418 cycles, 1.4 cycles/iter, 0.9 CPI
rier(nc): 368249 cycles, 1.4 cycles/iter, 0.9 CPI
D$ Bytes Read:            2495309
D$ Bytes Written:         21232
D$ Read Accesses:         579734
D$ Write Accesses:        5882
D$ Read Misses:           301869
D$ Write Misses:          4164
D$ Writebacks:            4190
D$ Miss Rate:             52.258%
I$ Bytes Read:            5041272
I$ Bytes Written:         0
I$ Read Accesses:         1605040
I$ Write Accesses:        0
I$ Read Misses:           82
I$ Write Misses:          0
I$ Writebacks:            0
I$ Miss Rate:             0.005%
```

Submission format and deadline



Submission

- Zip and upload your file to ceiba in the following format:

```
r09922028/                <-- zip this folder
    |-- matmul.c          // Matrix multiplication program
    |-- HW5.scala         // Cache configuration
    `-- HW5.pdf           // Including handwritten part, programming part 1 answers,
                             report on part 2 multi-core matrix-multiplication,
                             and bonus part if any
```

- Deadline: 1/11, 21:00
- **No late submission!!!**
- If there's any question, please send email to ntuca2020@gmail.com.
- TA hour for this homework: Wed 15:00 17:00 and Thur 15:00 17:00

Reference

- Chisel
  - https://en.wikipedia.org/wiki/Chisel_(programming_language)
  - https://www.chisel-lang.org/
- Rocket-chip
  - https://github.com/chipsalliance/rocket-chip
  - https://chipyard.readthedocs.io/en/latest/Generators/Rocket-Chip.html
  - https://riscv.org/wp-content/uploads/2015/01/riscv-rocket-chip-generator-workshop-jan2015.pdf
- Chipyard
  - https://chipyard.readthedocs.io/en/latest/Chipyard-Basics/index.html
  - https://fires.im/micro19-slides-pdf/03_building_custom_socs.pdf
- Dhrystone: https://en.wikipedia.org/wiki/Dhrystone