HW07

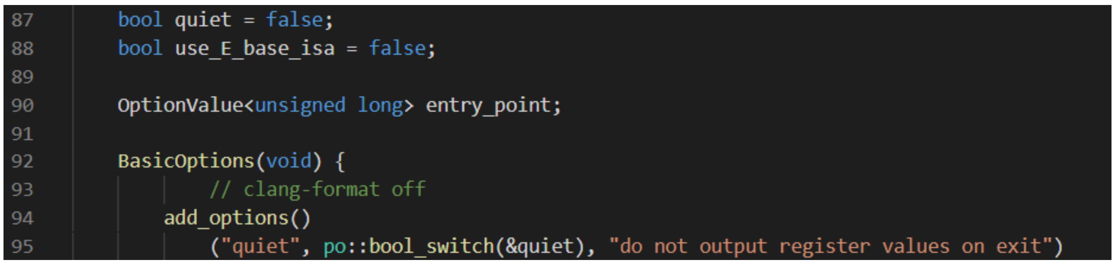
109061641 林庭寬

problem

add two Gaussian Blur modules to the 2-core riscv-vp platform ("tiny32-mc".

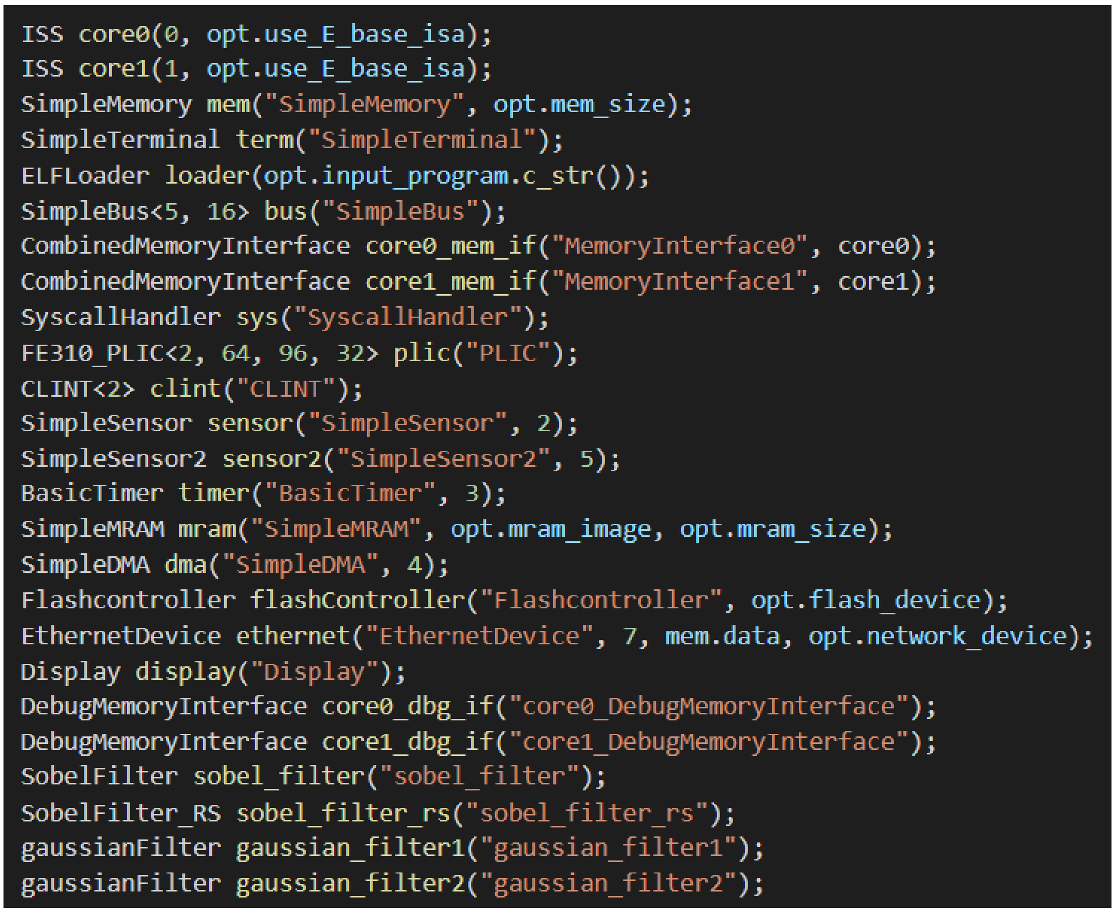
Solution algorithms

we use systemC TLM2.0 and mutex to implement this multicore gaussian blur
we divide the picture in half to use two core process this hw
at the platform we add quiet to do not output register values on exit



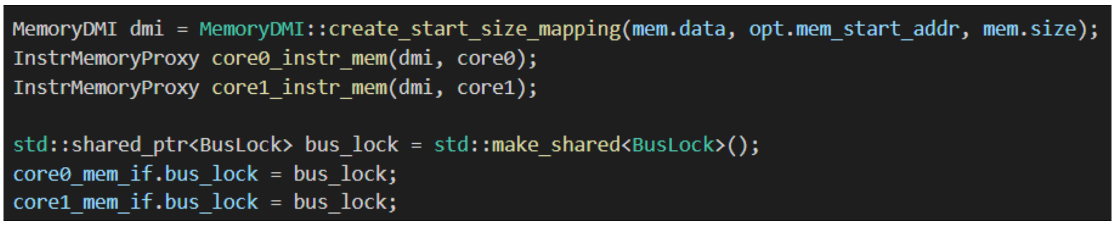
```
    bool quiet = false;
    bool use_E_base_isa = false;

    OptionValue<unsigned long> entry_point;

    BasicOptions(void) {
        // clang-format off
        add_options()
            ("quiet", po::bool_switch(&quiet), "do not output register values on exit")
```

add two core to execute gaussian blur



```
ISS core0(0, opt.use_E_base_isa);
ISS core1(1, opt.use_E_base_isa);
SimpleMemory mem("SimpleMemory", opt.mem_size);
SimpleTerminal term("SimpleTerminal");
ELFLoader loader(opt.input_program.c_str());
SimpleBus<5, 16> bus("SimpleBus");
CombinedMemoryInterface core0_mem_if("MemoryInterface0", core0);
CombinedMemoryInterface core1_mem_if("MemoryInterface1", core1);
SyscallHandler sys("SyscallHandler");
FE310_PLIC<2, 64, 96, 32> plic("PLIC");
CLINT<2> clint("CLINT");
SimpleSensor sensor("SimpleSensor", 2);
SimpleSensor2 sensor2("SimpleSensor2", 5);
BasicTimer timer("BasicTimer", 3);
SimpleMRAM mram("SimpleMRAM", opt.mram_image, opt.mram_size);
SimpleDMA dma("SimpleDMA", 4);
Flashcontroller flashController("Flashcontroller", opt.flash_device);
EthernetDevice ethernet("EthernetDevice", 7, mem.data, opt.network_device);
Display display("Display");
DebugMemoryInterface core0_dbg_if("core0_DebugMemoryInterface");
DebugMemoryInterface core1_dbg_if("core1_DebugMemoryInterface");
SobelFilter sobel_filter("sobel_filter");
SobelFilter_RS sobel_filter_rs("sobel_filter_rs");
gaussianFilter gaussian_filter1("gaussian_filter1");
gaussianFilter gaussian_filter2("gaussian_filter2");
```

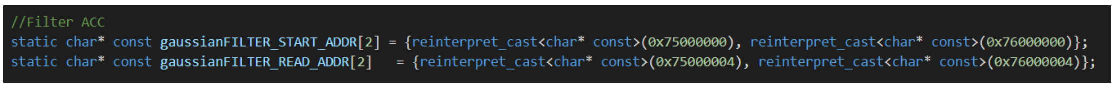
create two DMI and bus



```
MemoryDMI dmi = MemoryDMI::create_start_size_mapping(mem.data, opt.mem_start_addr, mem.size);
InstrMemoryProxy core0_instr_mem(dmi, core0);
InstrMemoryProxy core1_instr_mem(dmi, core1);

std::shared_ptr<BusLock> bus_lock = std::make_shared<BusLock>();
core0_mem_if.bus_lock = bus_lock;
core1_mem_if.bus_lock = bus_lock;
```

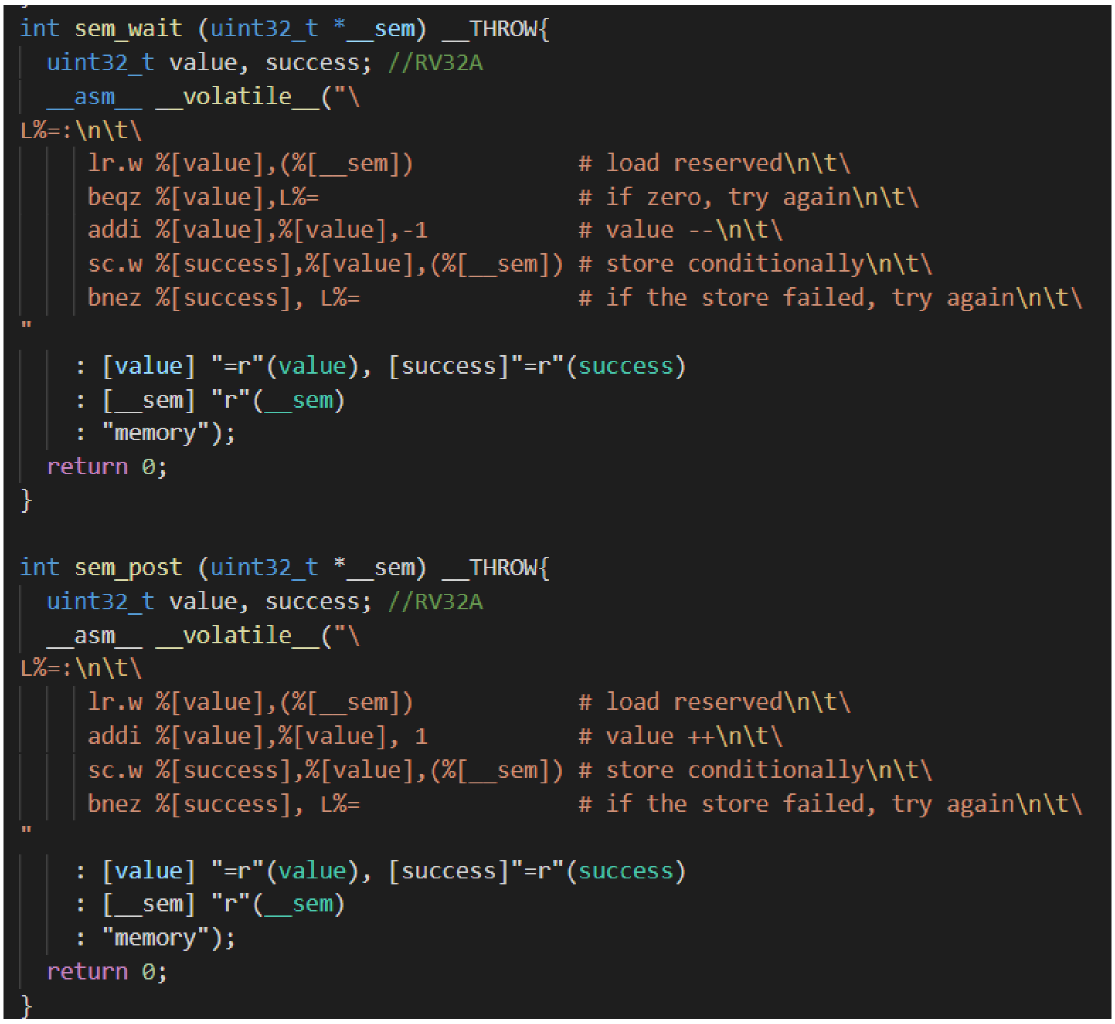
at sw

create two address to multicore



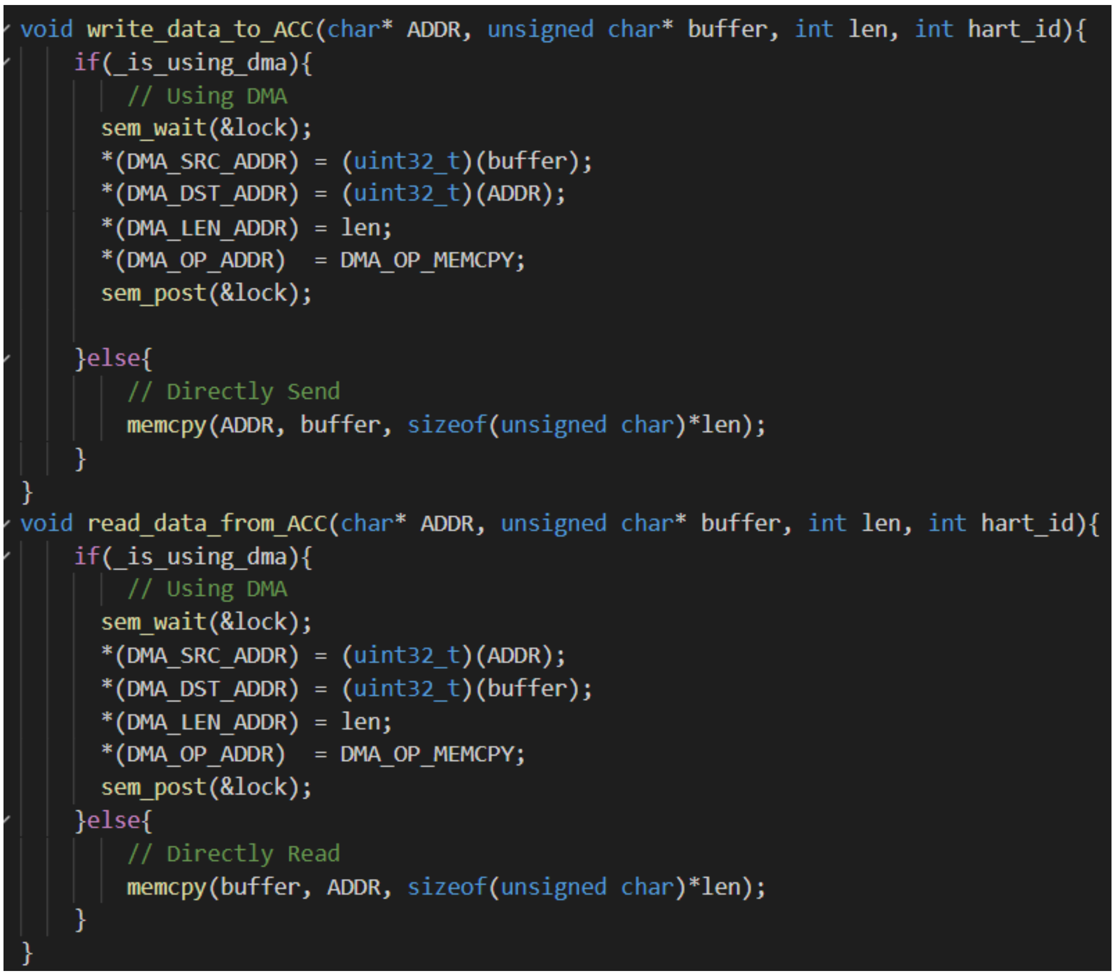
```
//Filter ACC
static char* const gaussianFILTER_START_ADDR[2] = {reinterpret_cast<char* const>(0x75000000), reinterpret_cast<char* const>(0x76000000)};
static char* const gaussianFILTER_READ_ADDR[2]   = {reinterpret_cast<char* const>(0x75000004), reinterpret_cast<char* const>(0x76000004)};
```

create two function sem_wait and sem_post



```
int sem_wait (uint32_t *__sem) __THROW{
  uint32_t value, success; //RV32A
  __asm__ __volatile__("\
L%=:\n\t\
     lr.w %[value],(%[__sem])            # load reserved\n\t\
     beqz %[value],L%=                   # if zero, try again\n\t\
     addi %[value],%[value],-1           # value --\n\t\
     sc.w %[success],%[value],(%[__sem]) # store conditionally\n\t\
     bnez %[success], L%=                # if the store failed, try again\n\t\
"
    : [value] "=r"(value), [success]"=r"(success)
    : [__sem] "r"(__sem)
    : "memory");
  return 0;
}

int sem_post (uint32_t *__sem) __THROW{
  uint32_t value, success; //RV32A
  __asm__ __volatile__("\
L%=:\n\t\
     lr.w %[value],(%[__sem])            # load reserved\n\t\
     addi %[value],%[value], 1           # value ++\n\t\
     sc.w %[success],%[value],(%[__sem]) # store conditionally\n\t\
     bnez %[success], L%=                # if the store failed, try again\n\t\
"
    : [value] "=r"(value), [success]"=r"(success)
    : [__sem] "r"(__sem)
    : "memory");
  return 0;
}
```

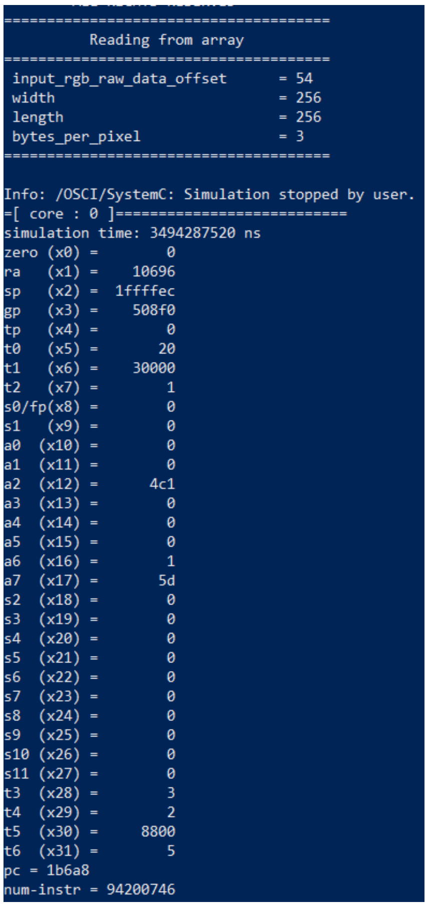
at read data and write data add above two function



```c
void write_data_to_ACC(char* ADDR, unsigned char* buffer, int len, int hart_id){
    if(_is_using_dma){
        // Using DMA
      sem_wait(&lock);
      *(DMA_SRC_ADDR) = (uint32_t)(buffer);
      *(DMA_DST_ADDR) = (uint32_t)(ADDR);
      *(DMA_LEN_ADDR) = len;
      *(DMA_OP_ADDR)  = DMA_OP_MEMCPY;
      sem_post(&lock);

    }else{
        // Directly Send
        memcpy(ADDR, buffer, sizeof(unsigned char)*len);
    }
}
void read_data_from_ACC(char* ADDR, unsigned char* buffer, int len, int hart_id){
    if(_is_using_dma){
        // Using DMA
      sem_wait(&lock);
      *(DMA_SRC_ADDR) = (uint32_t)(ADDR);
      *(DMA_DST_ADDR) = (uint32_t)(buffer);
      *(DMA_LEN_ADDR) = len;
      *(DMA_OP_ADDR)  = DMA_OP_MEMCPY;
      sem_post(&lock);
    }else{
        // Directly Read
        memcpy(buffer, ADDR, sizeof(unsigned char)*len);
    }
}
```

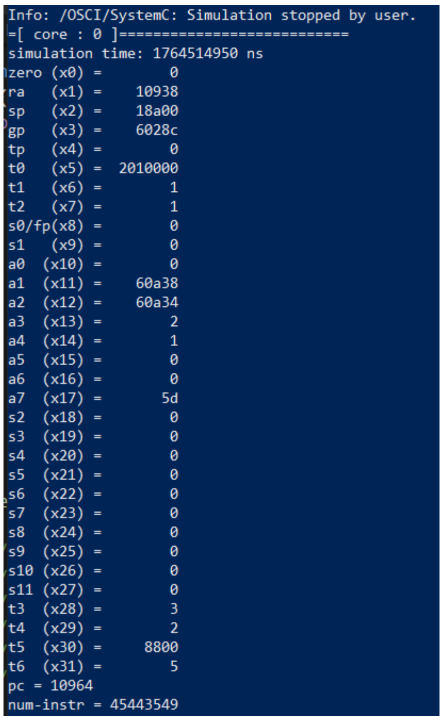
Experimental results

one core

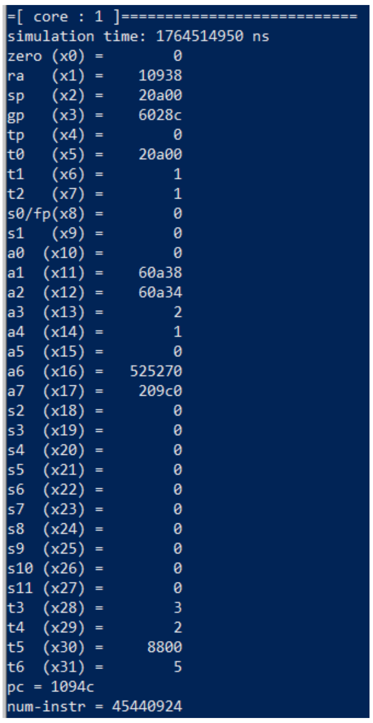


```
=====================================
          Reading from array
=====================================
 input_rgb_raw_data_offset      = 54
 width                          = 256
 length                         = 256
 bytes_per_pixel                = 3
=====================================

Info: /OSCI/SystemC: Simulation stopped by user.
=[ core : 0 ]===========================
simulation time: 3494287520 ns
zero (x0) =         0
ra   (x1) =     10696
sp   (x2) =   1ffffec
gp   (x3) =     508f0
tp   (x4) =         0
t0   (x5) =        20
t1   (x6) =     30000
t2   (x7) =         1
s0/fp(x8) =         0
s1   (x9) =         0
a0  (x10) =         0
a1  (x11) =         0
a2  (x12) =       4c1
a3  (x13) =         0
a4  (x14) =         0
a5  (x15) =         0
a6  (x16) =         1
a7  (x17) =        5d
s2  (x18) =         0
s3  (x19) =         0
s4  (x20) =         0
s5  (x21) =         0
s6  (x22) =         0
s7  (x23) =         0
s8  (x24) =         0
s9  (x25) =         0
s10 (x26) =         0
s11 (x27) =         0
t3  (x28) =         3
t4  (x29) =         2
t5  (x30) =      8800
t6  (x31) =         5
pc = 1b6a8
num-instr = 94200746
```

two core



```
Info: /OSCI/SystemC: Simulation stopped by user.
=[ core : 0 ]===========================
simulation time: 1764514950 ns
zero (x0) =         0
ra   (x1) =     10938
sp   (x2) =     18a00
gp   (x3) =     6028c
tp   (x4) =         0
t0   (x5) =   2010000
t1   (x6) =         1
t2   (x7) =         1
s0/fp(x8) =         0
s1   (x9) =         0
a0  (x10) =         0
a1  (x11) =     60a38
a2  (x12) =     60a34
a3  (x13) =         2
a4  (x14) =         1
a5  (x15) =         0
a6  (x16) =         0
a7  (x17) =        5d
s2  (x18) =         0
s3  (x19) =         0
s4  (x20) =         0
s5  (x21) =         0
s6  (x22) =         0
s7  (x23) =         0
s8  (x24) =         0
s9  (x25) =         0
s10 (x26) =         0
s11 (x27) =         0
t3  (x28) =         3
t4  (x29) =         2
t5  (x30) =      8800
t6  (x31) =         5
pc = 10964
num-instr = 45443549
```



```
=[ core : 1 ]===========================
simulation time: 1764514950 ns
zero (x0) =         0
ra   (x1) =     10938
sp   (x2) =     20a00
gp   (x3) =     6028c
tp   (x4) =         0
t0   (x5) =     20a00
t1   (x6) =         1
t2   (x7) =         1
s0/fp(x8) =         0
s1   (x9) =         0
a0  (x10) =         0
a1  (x11) =     60a38
a2  (x12) =     60a34
a3  (x13) =         2
a4  (x14) =         1
a5  (x15) =         0
a6  (x16) =    525270
a7  (x17) =     209c0
s2  (x18) =         0
s3  (x19) =         0
s4  (x20) =         0
s5  (x21) =         0
s6  (x22) =         0
s7  (x23) =         0
s8  (x24) =         0
s9  (x25) =         0
s10 (x26) =         0
s11 (x27) =         0
t3  (x28) =         3
t4  (x29) =         2
t5  (x30) =      8800
t6  (x31) =         5
pc = 1094c
num-instr = 45440924
```

	One core	Two core
Simulation time	3494287520ns	1764514950ns
Num instruciton	9400746	45440924 * 2

Discussions and conclusions

Before this homework I do lab08 to learn the architecture of riscv, and this homework I learn about riscv architecture and implement multicore in C and systemC. I think riscv is very useful to application. I derive much benefit in this class, thanks.