HW06

109061641 林庭寬

problem

In this homework we will Cross-compile Gaussian Blur to RISC-V VP platform.

Solution algorithms

we use systemC and TLM2.0 to implement this gaussian blur
at the platform we set fifo channel

```
sc_fifo<unsigned char> i_r;
sc_fifo<unsigned char> i_g;
sc_fifo<unsigned char> i_b;
sc_fifo<int> o_result_r;
sc_fifo<int> o_result_g;
sc_fifo<int> o_result_b;
```

TLM read and write

```
case tlm::TLM_READ_COMMAND:
  // cout << "READ" << endl;
  switch (addr) {
    case GAUSSIAN_BLUR_RESULT_ADDR:
      buffer.uc[0] = o_result_r.read();
      buffer.uc[1] = o_result_g.read();
      buffer.uc[2] = o_result_b.read();
      buffer.uc[3] = 0;
      break;
    default:
      std::cerr << "READ Error! gaussianFilter::blocking_transport: address 0x"
                << std::setfill('0') << std::setw(8) << std::hex << addr
                << std::dec << " is not valid" << std::endl;
    }
  data_ptr[0] = buffer.uc[0];
  data_ptr[1] = buffer.uc[1];
  data_ptr[2] = buffer.uc[2];
  data_ptr[3] = buffer.uc[3];
  break;
case tlm::TLM_WRITE_COMMAND:
  // cout << "WRITE" << endl;
  switch (addr) {
    case GAUSSIAN_BLUR_R_ADDR:
      i_r.write(data_ptr[0]);
      i_g.write(data_ptr[1]);
      i_b.write(data_ptr[2]);
      break;
    default:
      std::cerr << "WRITE Error! gaussianFilter::blocking_transport: address 0x"
                << std::setfill('0') << std::setw(8) << std::hex << addr
                << std::dec << " is not valid" << std::endl;
  }
  break;
```

set address in main.cpp

```
80      addr_t gaussianFilter_start_addr = 0x75000000;
81      addr_t gaussianFilter_size = 0x01000000;
82      addr_t gaussianFilter_end_addr = gaussianFilter_start_addr + gaussianFilter_size - 1;
```

create new bus port and socket

```
186     bus.ports[14] = new PortMapping(opt.gaussianFilter_start_addr, opt.gaussianFilter_end_addr);
```

```
211     bus.isocks[14].bind(gaussian_filter.tsock);
```

at the software

check start address and read address

```
5   static char* const gaussianFILTER_START_ADDR = reinterpret_cast<char* const>(0x75000000);
6   static char* const gaussianFILTER_READ_ADDR  = reinterpret_cast<char* const>(0x75000004);
```

we also use direct mapping access to read and write data

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

Experimental results

```
=====================================
          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
```

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 in C and systemC. I think riscv is very useful to application. I derive much benefit in this class, thanks.