HW02

109061641 林庭寬

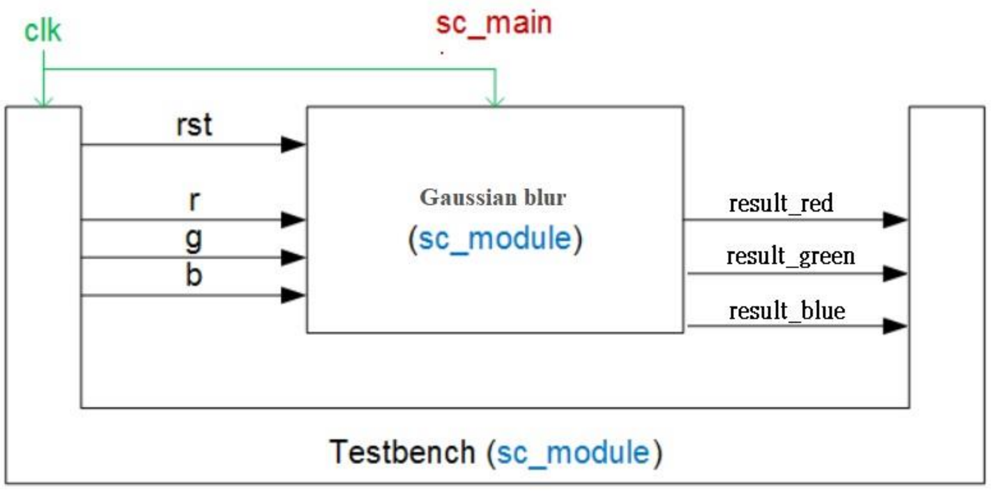
problem

1. Please implement a Gaussian blur filter with SystemC modules connected with SystemC FIFO channels.
2. Please rewrite the parts related to pixel transfer at Input and Calculation processes.

Solution algorithms

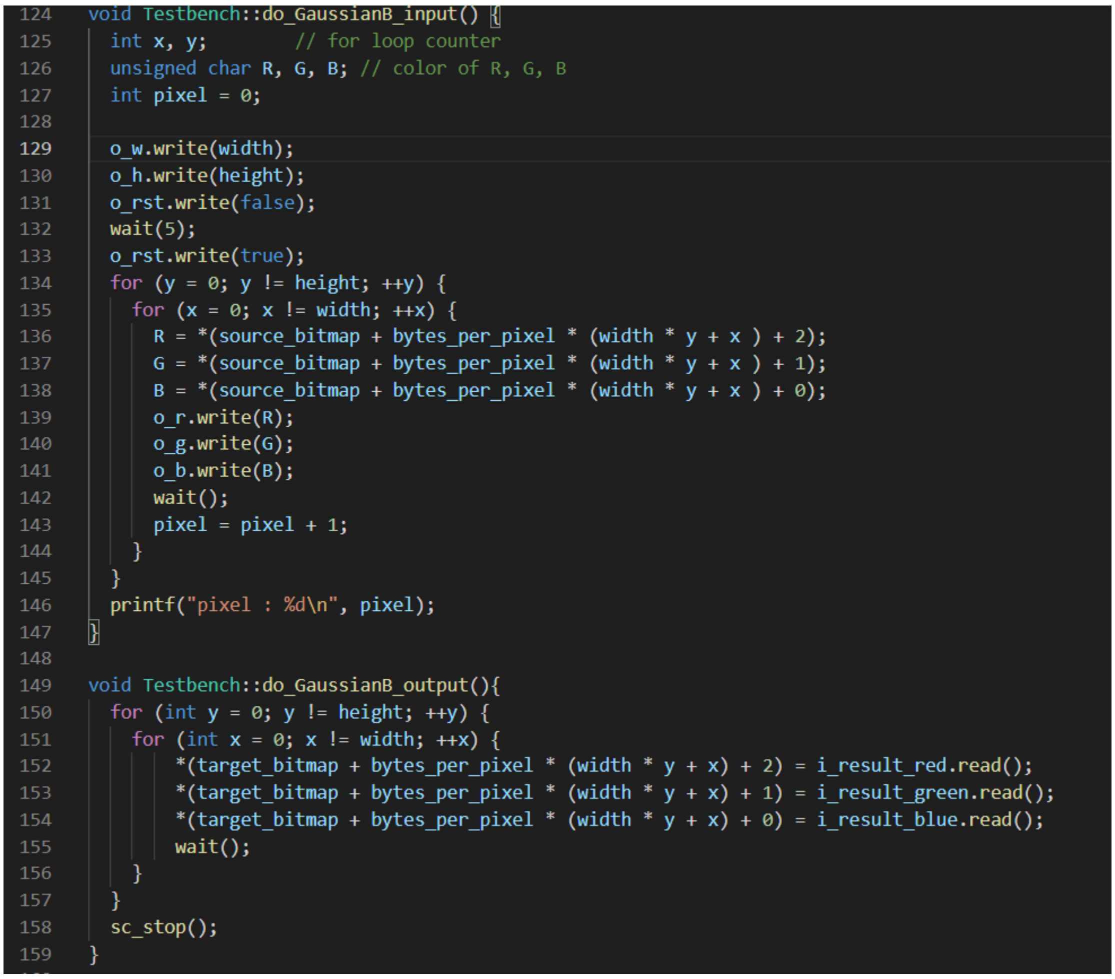
SystemC processes

A Gaussian blur with FIFO interface
Architecture shown below:



Gaussian_Blur.cpp do filter with sc_module and Testbench.cpp do read/write file with sc_module, after that FIFO will process next operation

At part 2

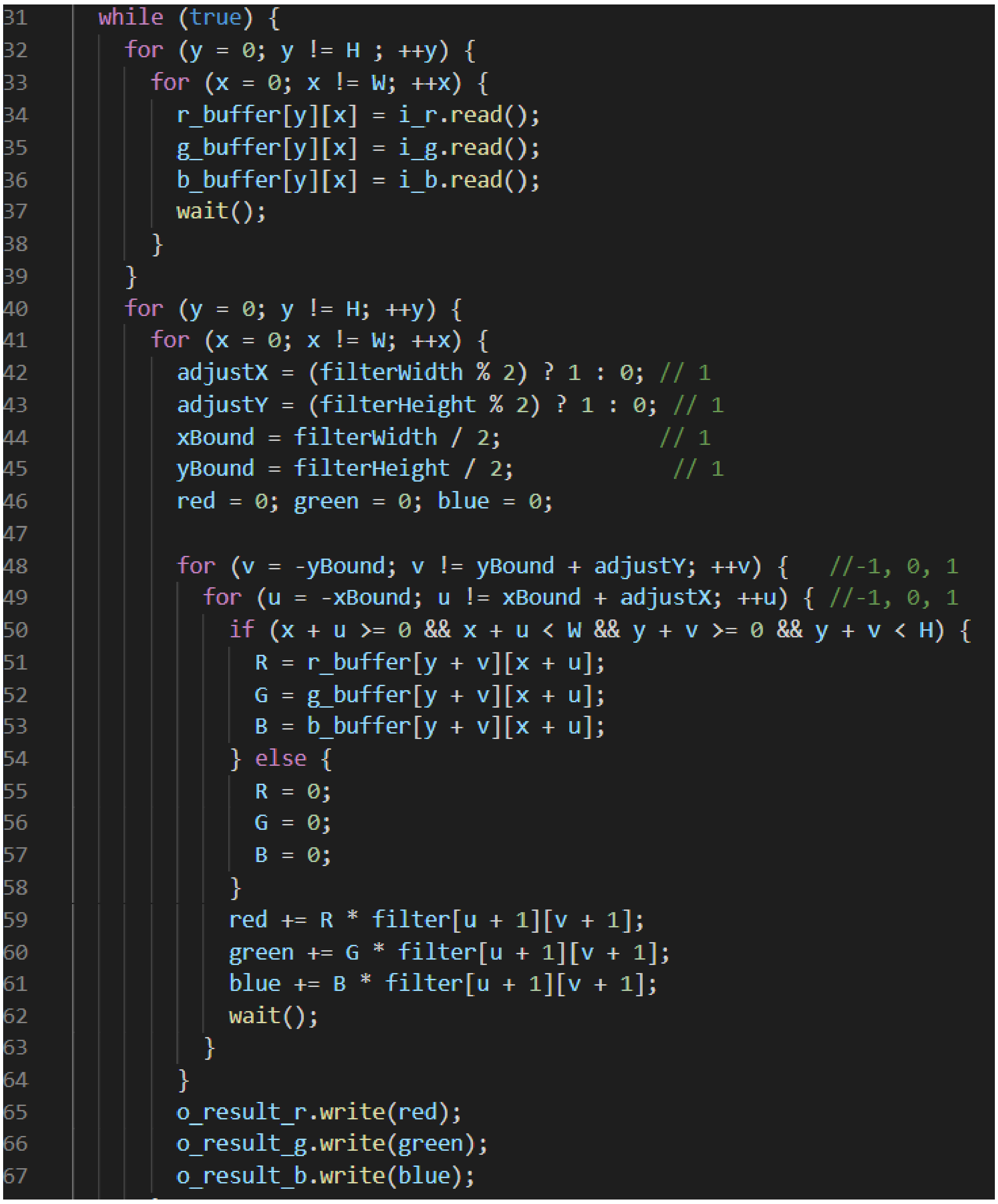


```
void Testbench::do_GaussianB_input() {
  int x, y;        // for loop counter
  unsigned char R, G, B; // color of R, G, B
  int pixel = 0;

  o_w.write(width);
  o_h.write(height);
  o_rst.write(false);
  wait(5);
  o_rst.write(true);
  for (y = 0; y != height; ++y) {
    for (x = 0; x != width; ++x) {
      R = *(source_bitmap + bytes_per_pixel * (width * y + x ) + 2);
      G = *(source_bitmap + bytes_per_pixel * (width * y + x ) + 1);
      B = *(source_bitmap + bytes_per_pixel * (width * y + x ) + 0);
      o_r.write(R);
      o_g.write(G);
      o_b.write(B);
      wait();
      pixel = pixel + 1;
    }
  }
  printf("pixel : %d\n", pixel);
}

void Testbench::do_GaussianB_output(){
  for (int y = 0; y != height; ++y) {
    for (int x = 0; x != width; ++x) {
        *(target_bitmap + bytes_per_pixel * (width * y + x) + 2) = i_result_red.read();
        *(target_bitmap + bytes_per_pixel * (width * y + x) + 1) = i_result_green.read();
        *(target_bitmap + bytes_per_pixel * (width * y + x) + 0) = i_result_blue.read();
        wait();
    }
  }
  sc_stop();
}
```

I read only one time per pixel in R G B, and use two thread with input and output to implement this testbench.



```
    while (true) {
      for (y = 0; y != H ; ++y) {
        for (x = 0; x != W; ++x) {
          r_buffer[y][x] = i_r.read();
          g_buffer[y][x] = i_g.read();
          b_buffer[y][x] = i_b.read();
          wait();
        }
      }
      for (y = 0; y != H; ++y) {
        for (x = 0; x != W; ++x) {
          adjustX = (filterWidth % 2) ? 1 : 0; // 1
          adjustY = (filterHeight % 2) ? 1 : 0; // 1
          xBound = filterWidth / 2;            // 1
          yBound = filterHeight / 2;            // 1
          red = 0; green = 0; blue = 0;

          for (v = -yBound; v != yBound + adjustY; ++v) {   //-1, 0, 1
            for (u = -xBound; u != xBound + adjustX; ++u) { //-1, 0, 1
              if (x + u >= 0 && x + u < W && y + v >= 0 && y + v < H) {
                R = r_buffer[y + v][x + u];
                G = g_buffer[y + v][x + u];
                B = b_buffer[y + v][x + u];
              } else {
                R = 0;
                G = 0;
                B = 0;
              }
              red += R * filter[u + 1][v + 1];
              green += G * filter[u + 1][v + 1];
              blue += B * filter[u + 1][v + 1];
              wait();
            }
          }
          o_result_r.write(red);
          o_result_g.write(green);
          o_result_b.write(blue);
```

First read data from fifo to r g b buffer, and use data in the buffer to do Gaussian blur, then output fifo in the end.

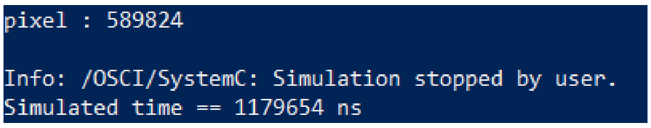
Experimental results

Before filter

After filter

Number of pixel:

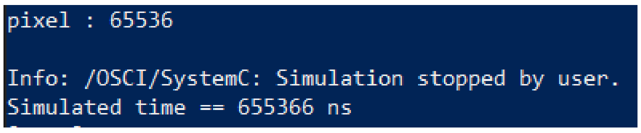
use first way



```
pixel : 589824

Info: /OSCI/SystemC: Simulation stopped by user.
Simulated time == 1179654 ns
```

use second way:



```
pixel : 65536

Info: /OSCI/SystemC: Simulation stopped by user.
Simulated time == 655366 ns
```

Discussions and conclusions

Before this homework I do lab01~02 to learn cmake and the architecture of systemC,and this homework I learn about filter architecture and coding in C and systemC, and how to implement fifo architecture. In part 2 use send in

batch of image pixels from Input to Calculation, it can reduce number of memory access (589824 ➔ 65536) and execution time (1179654ns ➔ 655366ns).
I derive much benefit in this class, thanks.