ECPS 203

TA: Zhongqi Cheng

Agenda

- Assignment 8
  1. Add timing to your model
  2. Pipelining canny
  3. Parallelize BlurX and BlurY

Add timing to your model

- Suppose you have a function foo() as follows

```
void foo(){ ... }
```

you have measured the run time of foo() on your board, and it is

10 seconds

Add timing to your model

- Back-annotate the 10 seconds into your module

```
SC_MODULE(M1){

    void foo(){ ... }

    void main(){
        foo();
        wait(10,SC_SEC);
    }

    SC_CTOR(Stimulus)
    {
        SC_THREAD(main);
    }
}
```

module M1

place a

wait(10, SC_SEC)

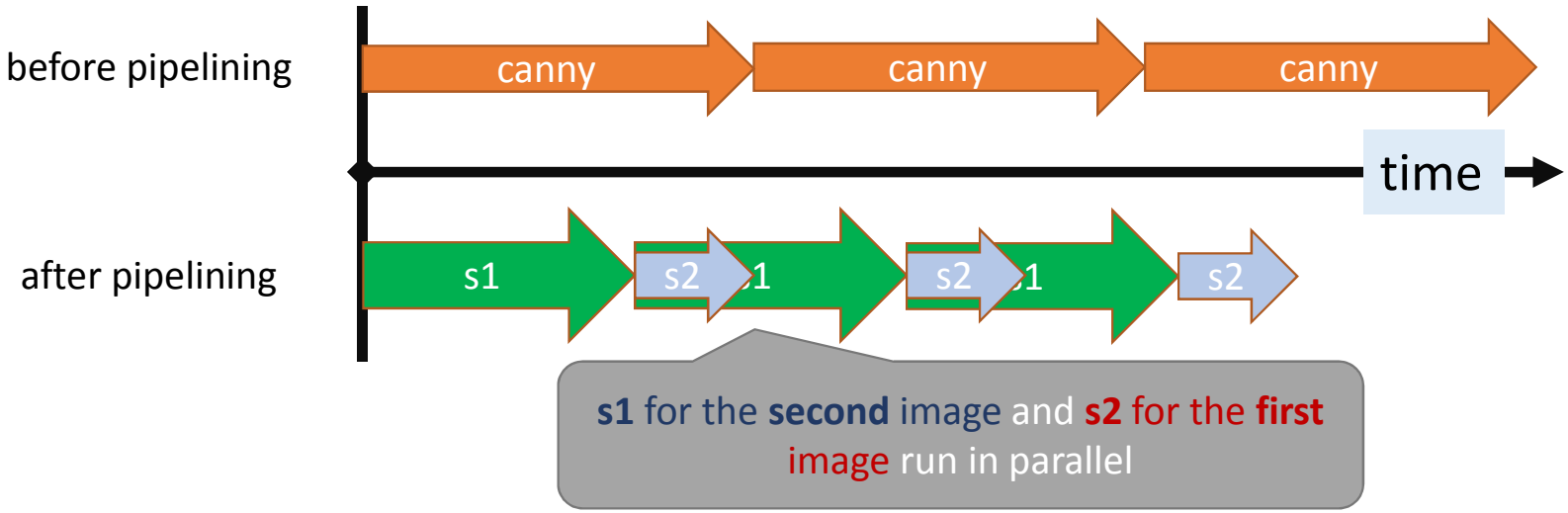
before or after foo()

other time units in SystemC

SC_FS
SC_PS
SC_NS
SC_US
SC_MS
SC_SEC

Design a pipeline

- Pipeline can speedup the application
- An example with a sequence of 3 input images
  - suppose we divide canny into 2 pipeline stages: s1, s2



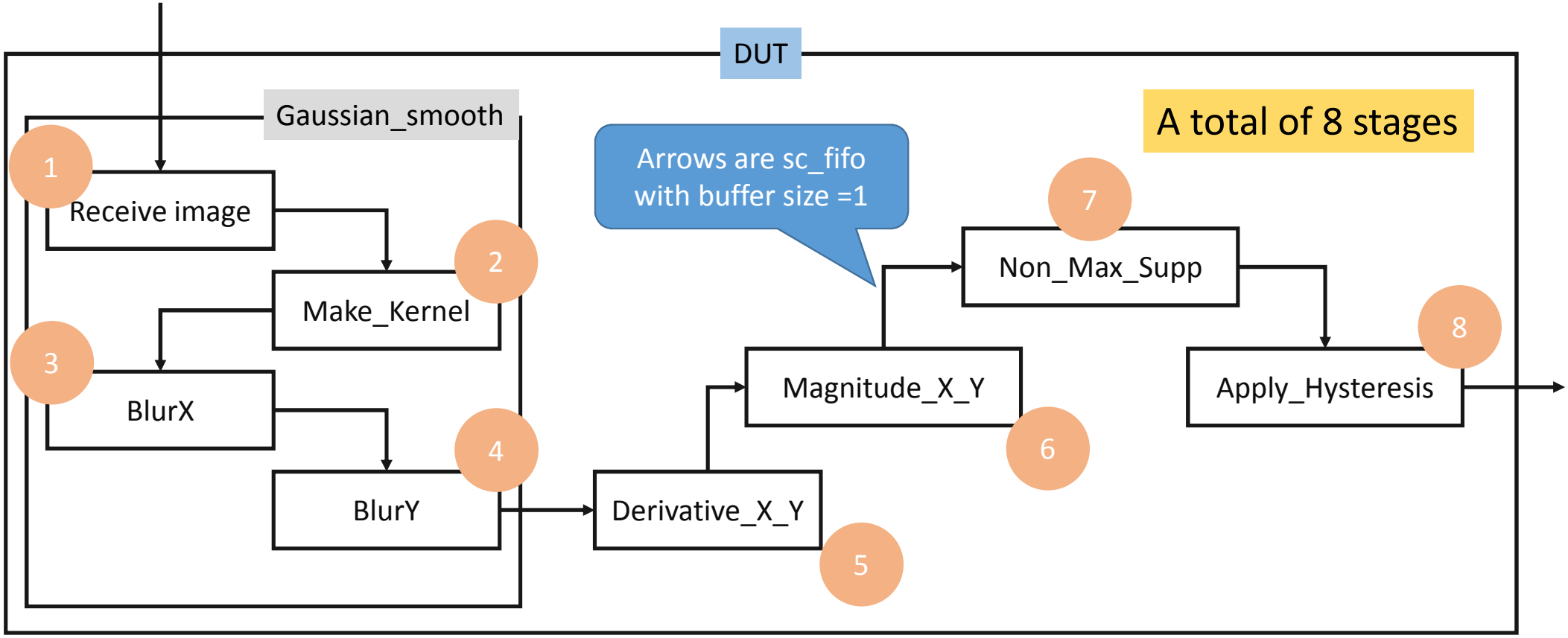
Design a pipeline

- In our canny application, there are following stages
  1. Receive_Image
  2. Make_Kernel
  3. BlurX
  4. BlurY
  5. Derivative_X_Y
  6. Magnitude_X_Y
  7. Non_Max_Supp
  8. Apply_Hysteresis

Design a pipeline

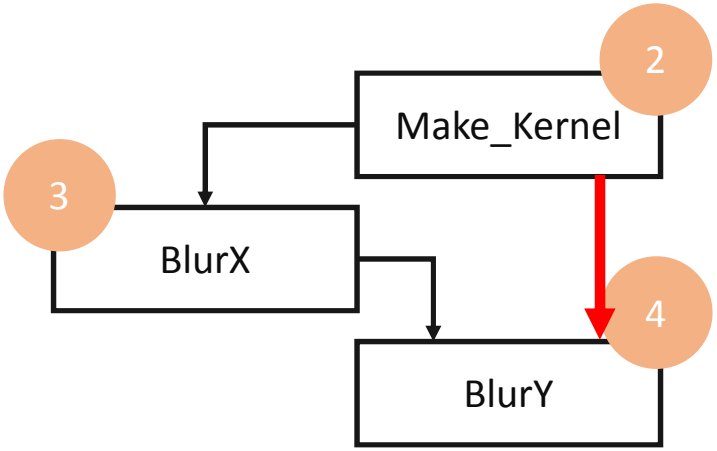
- Between each stage, there should be a buffer. Otherwise the model will not behave as a pipeline
- sc_fifo is the buffer.
- set the buffer size of sc_fifo to 1

Design a pipeline

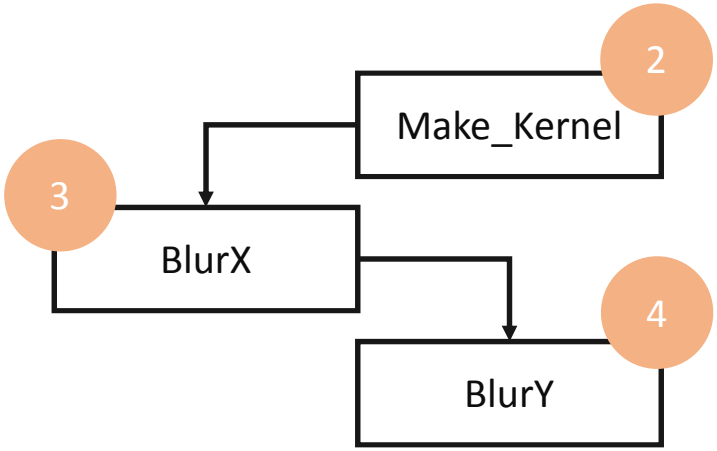


Design a pipeline

- Modifying the model in assignment 6
- In assignment 8, the nth stage should only output to the $(n+1)^{th}$ stage



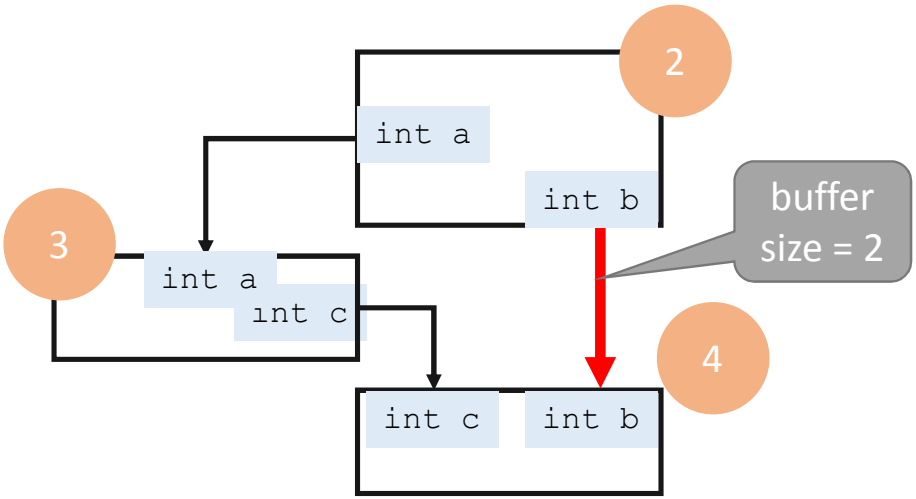
In assignment 6, connections may be across two stages



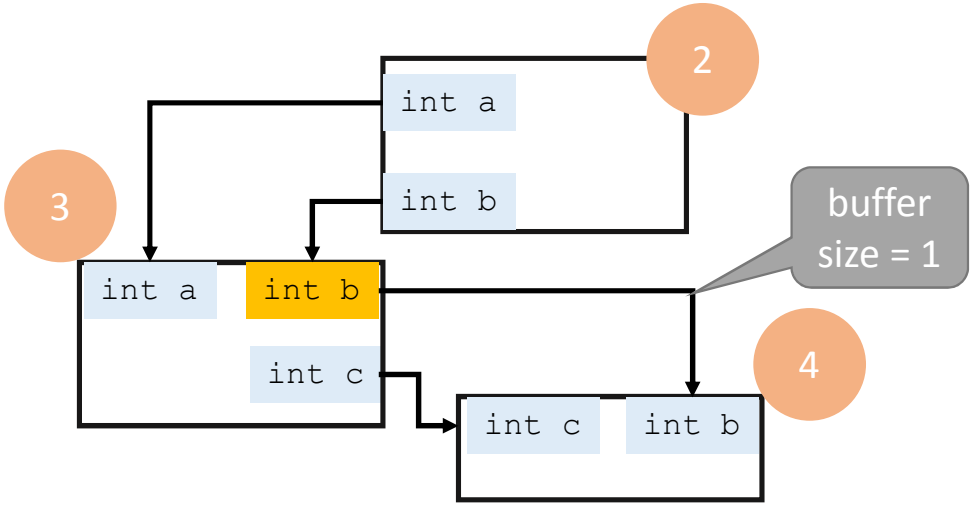
In assignment 8, connection should be only between two neighboring stages

Design a pipeline

- Add a “relaying” variable in the middle stage



In assignment 6, connections may be across two stages



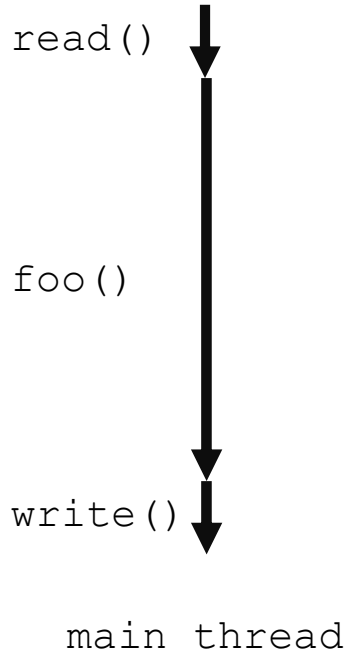
add a “relaying” variable **b** in stage 3

Parallelizing a module

- How to parallelize foo() ?

```
SC_MODULE(M1){

    void foo(){
        for(int i=0;i<LEN;i++)
            array_out[i]=x[i]+1;
    }

    void main(){
        In.read(x);
        foo();
        wait(10,SC_SEC);
        Out.write(array_out)
    }
    SC_CTOR(Stimulus)
    {
         SC_THREAD(main);
         SET_STACK_SIZE
    }
    …
}
```

module M1



Parallelizing a module

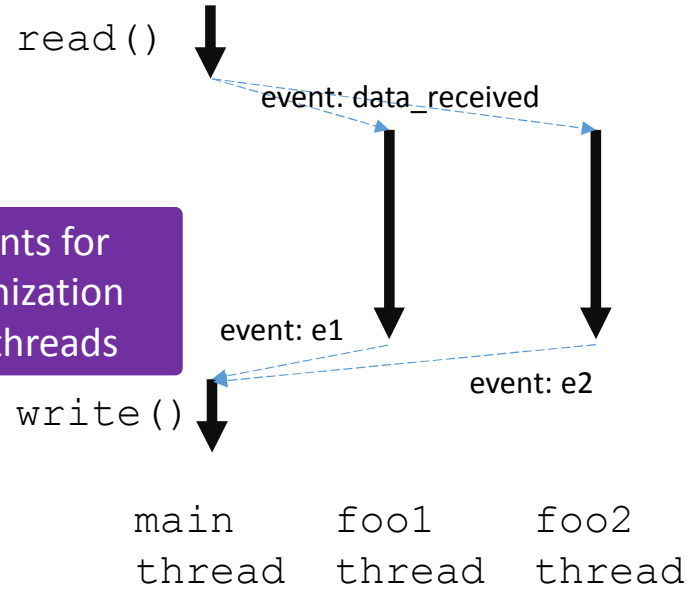
- Split it into two parallel parts

```
SC_MODULE(M1){

    event e1,e2,data_received;
    void foo1(){
        wait(data_received);
        for(int i=0;i<LEN/2;i++)
            array_out[i]=x[i]+1;
        wait(10/2,SC_SEC);
        e1.notify(SC_ZERO_TIME);
    } //sc_zero_time means that
the event is notified immediately

    void main(){
        In.read(x)
        data_received.notify(
SC_ZERO_TIME);
        wait(e1&e2);
        Out.write(array_out)
    }
}
```

```
void foo2(){
    wait(data_received);
    for(int i=LEN/2;i<LEN;i++)
        array_out[i]=x[i]+1;
    wait(10/2,SC_SEC);
    e2.notify(SC_ZERO_TIME)
}
SC_CTOR(Stimulus)
{
    SC_THREAD(main);
    SET_STACK_SIZE
    SC_THREAD(foo1);
    SET_STACK_SIZE
    SC_THREAD(foo2);
    SET_STACK_SIZE
}
```

module M1

use events for synchronization among threads



Parallelizing a module

- in Assignment 8, we parallelize BlurX and BlurY
- each with 4 parallel slices
- Hints:
  1. use events for synchronization
  2. make sure the Blur algorithms still work correctly
  3. remember to change the timing for each parallel slices
  4. don’t forget SC_ZERO_TIME in the notify() function

Submission

- Canny.cpp: source code
- Canny.txt: troubles and result output