ECPS 203
Discussion

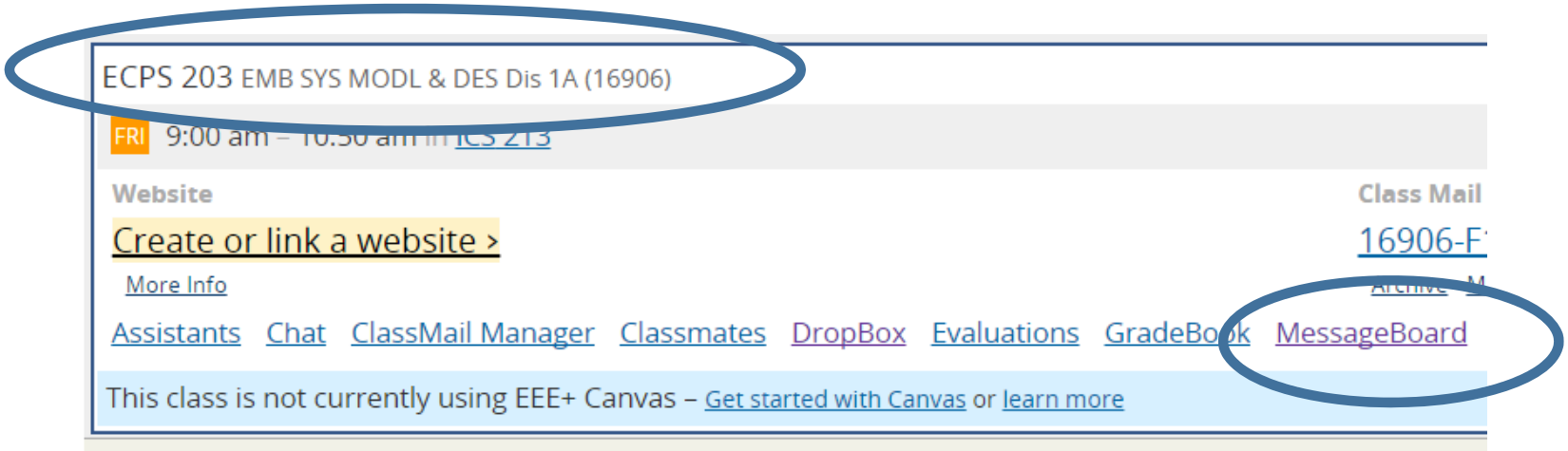
TA: Zhongqi Cheng

Office hour and MessageBoard

- change of office hour

EH 4106	Wednesday, 10am

- Use course messageboard on eee.uci.edu



Agenda

- Assignment 3
  - modules
  - compiling
  - submission

Assignment 3

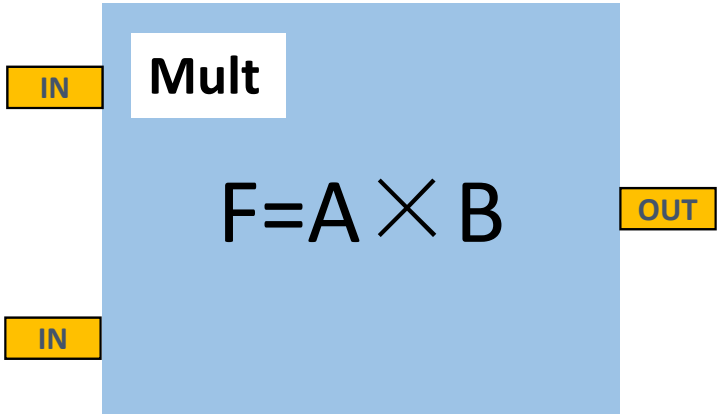
- The purpose of this Assignment

Get involved in SystemC

- **write code**
- **compile code**
- **run simulation**

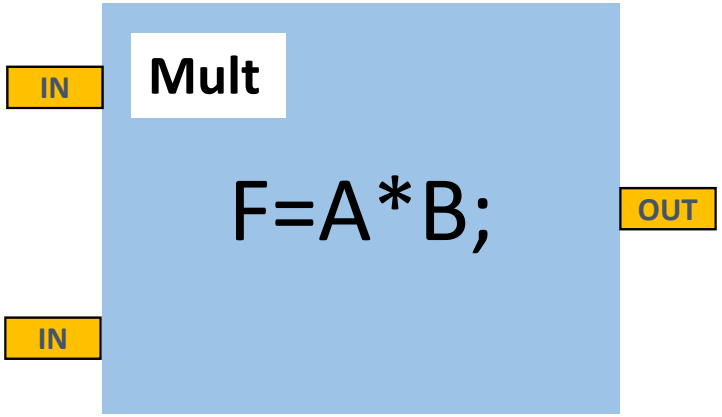
Assignment 3

- Hardware: a multiplier:
  - two inputs: A, B
  - one output: F
  - core algorithm: F=A×B



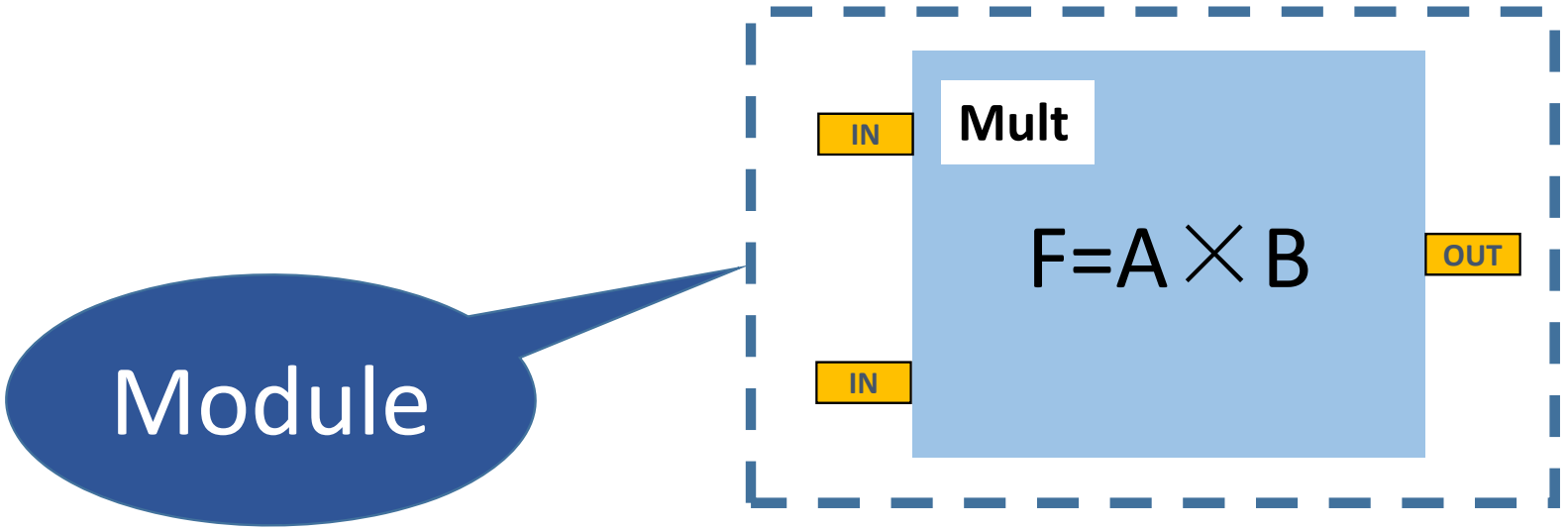
Assignment 3

- Core algoritm:
  - only one line of code in C++
  - F=a*b



Assignment 3

- How to model this component in SystmC?
  - such component is called a module

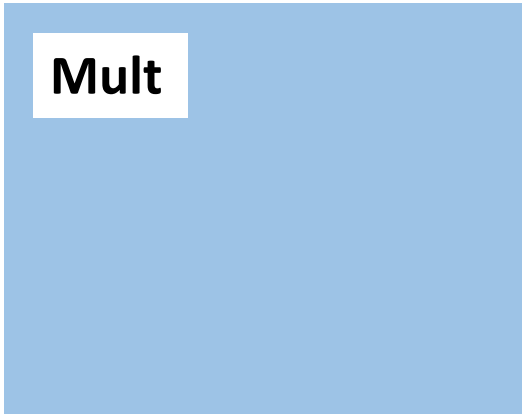


Assignment 3

- Create the module for the multiplier

```
SC_MODULE(Mult)
{

};
```

SC_MODULE represents a module



Assignment 3

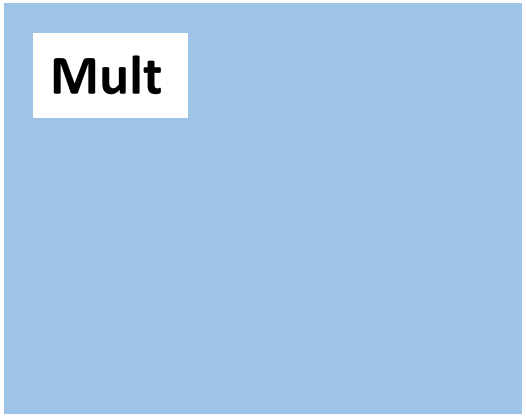
- Create the module for the multiplier

SC_MODULE(Mult)
{

equivalent

class Mult: public sc_module

SC_MODULE is a macro defined in systemc.h



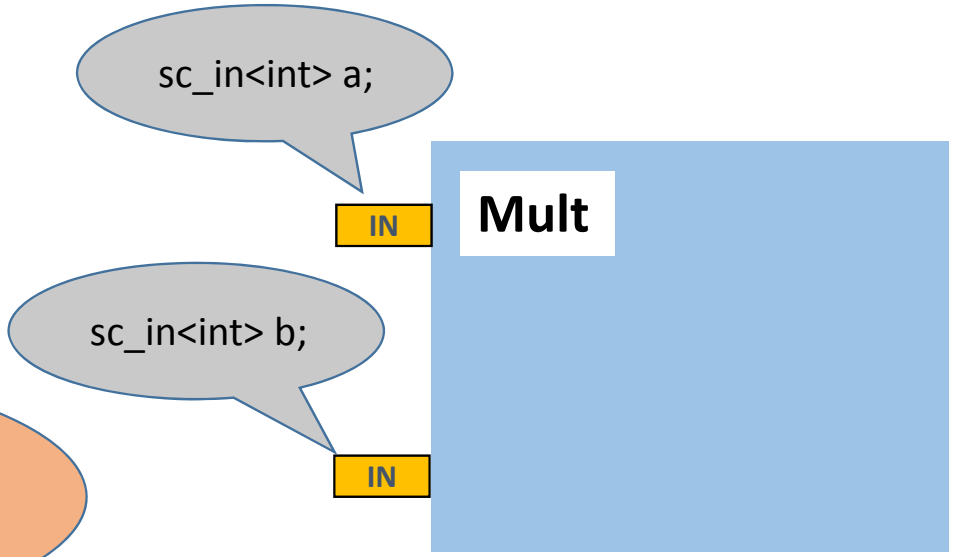
};

Assignment 3

- Add two input ports

```
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;

};
```

Ports: integer inputs

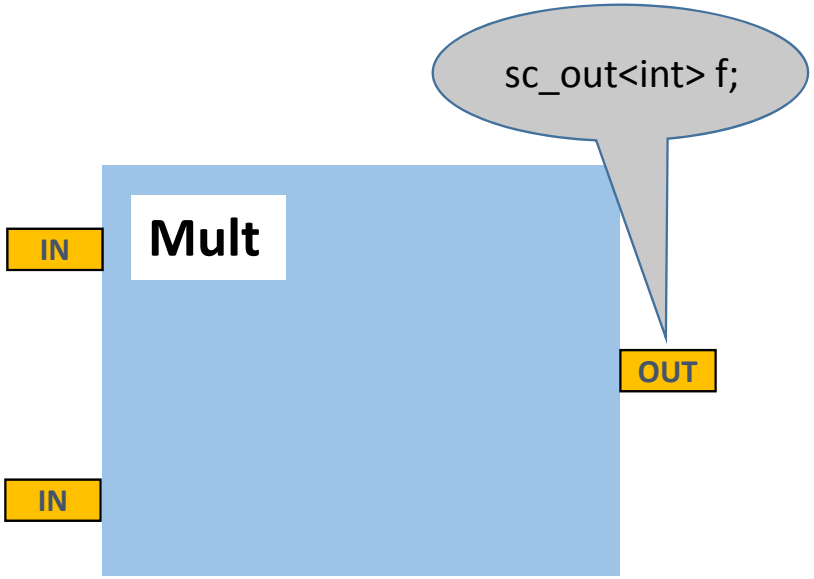


Assignment 3

- Add an output port

```
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;
    sc_out<int> f;

};
```

integer output



Assignment 3

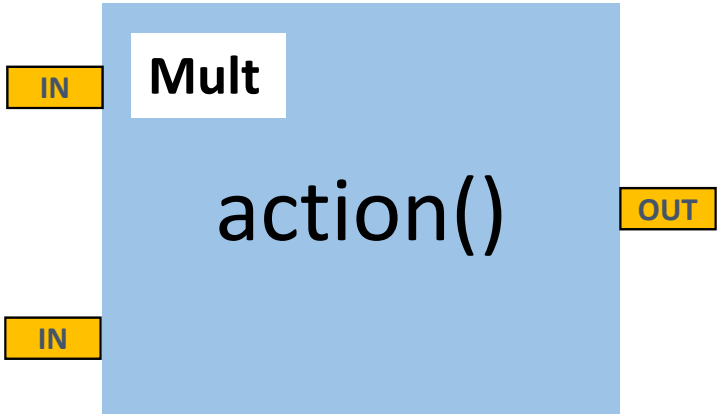
- Design the functionality of the module

```
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;
    sc_out<int> f;

    void action(); //f=a*b

};
```

function definition

```
void Mult::action(){
    f=a*b;
}
```

function declaration

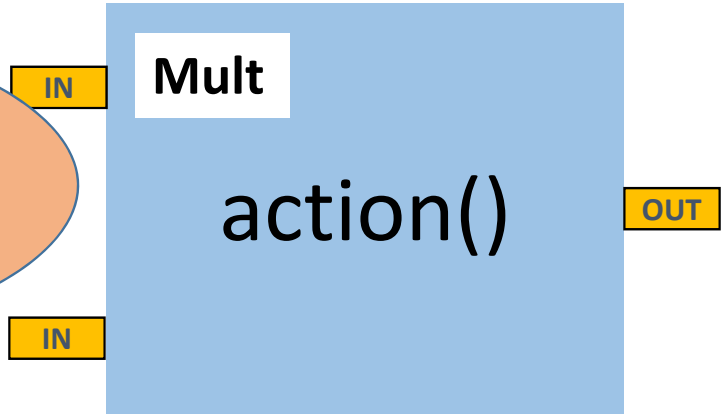


Assignment 3

- Make action() sensitive to input signals

```
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;
    sc_out<int> f;

    void action(); //f=a

    SC_CTOR(Mult)
   {
      SC_METHOD(action);
      sensitive << a << b;
   }
};
```

During the simulation, when **a** or **b** is changed, method action() will be invoked

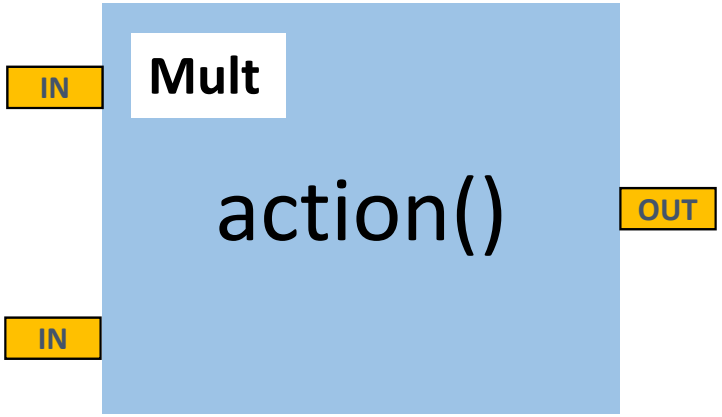


Assignment 3

- Complete code for Mult

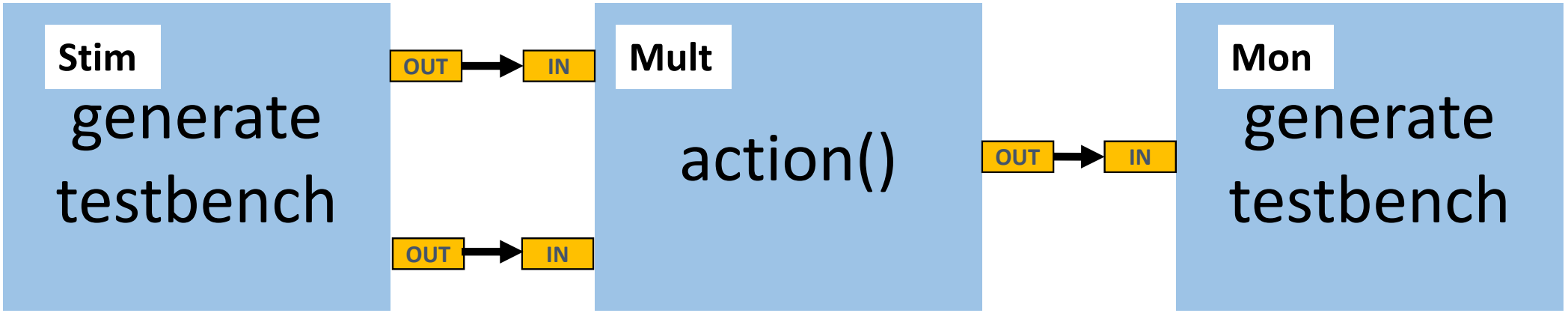
```
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;
    sc_out<int> f;

    void action(); //f=a*b

    SC_CTOR(Mult)
    {
      SC_METHOD(action);
      sensitive << a << b;
    }
};
```

```
void Mult::action(){
   f=a*b;
}
```



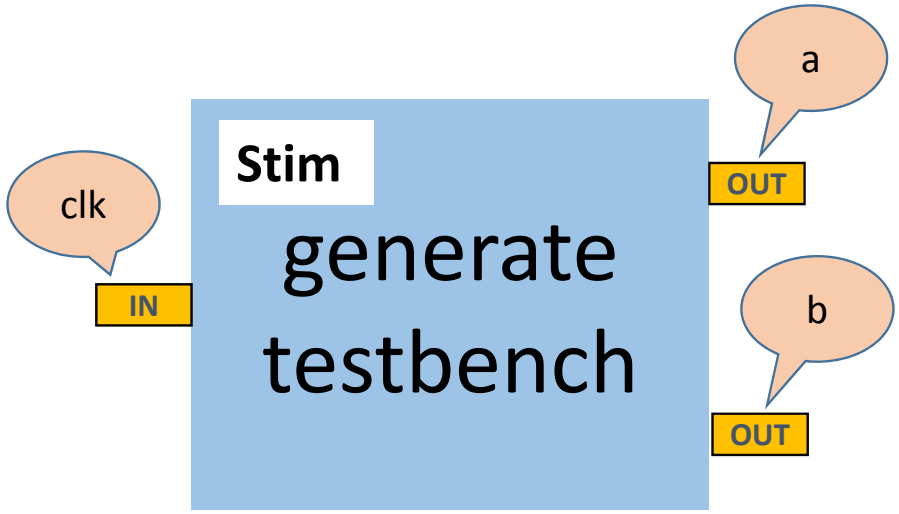
Assignment 3

- Test the Multiplier
- Stimulus: generates the testbench
- Monitor: display the result



Assignment 3

- Stimulus is also a module
  - name: Stim
  - two outputs to Mult:
    - a
    - b
  - One input
    - clock signal clk
    - clock signal is common in digital circuits

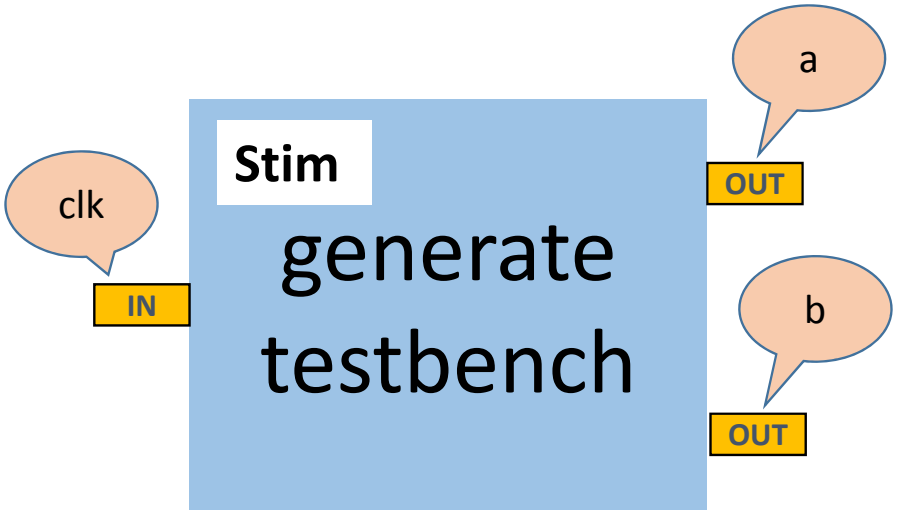


Assignment 3

- Code

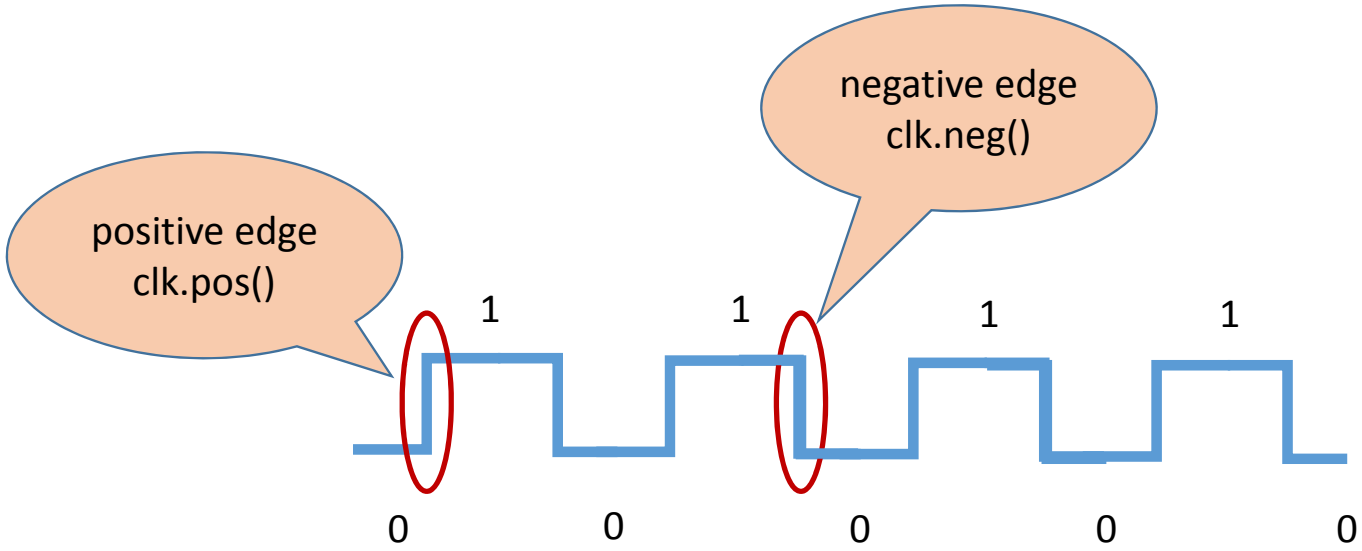
```
SC_MODULE(Stim)
{
   sc_in<bool> clk;
   sc_out<int> a;
   sc_out<int> b;

   void action();

   SC_CTOR(Stim)
   {
      SC_THREAD(action);
      sensitive << clk.pos();
   }
};
```

positive edge
of clock signal



Assignment 3

- Clock signal



Assignment 3

- wait() function and SC_THREAD()

```
SC_MODULE(Stim)
{
   sc_in<bool> clk;
   sc_out<int> a;
   sc_out<int> b;

   void action();

   SC_CTOR(Stim)
   {
      SC_THREAD(action);
      sensitive << clk.pos();
   }
};
```

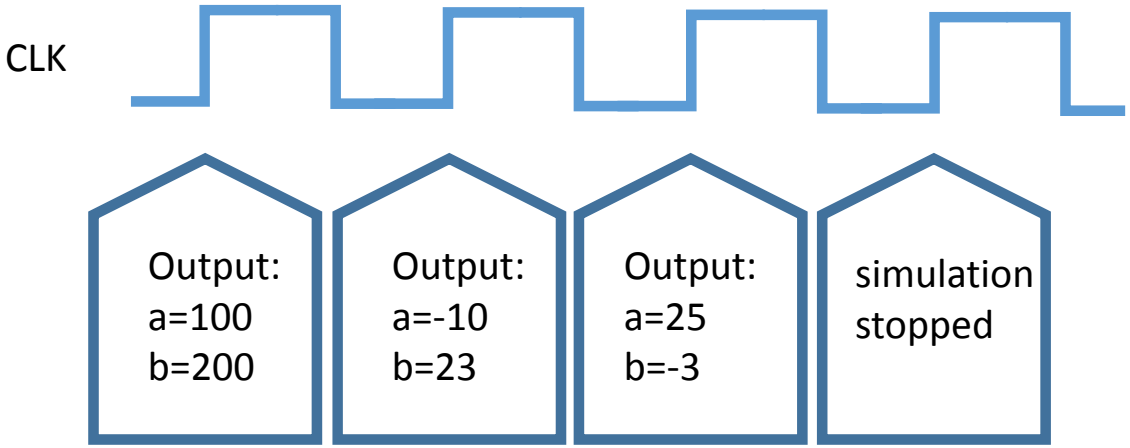
create a thread for action () during simulation

```
void Stim::action(){
   wait();
   a = 100;
   b = 200;
   wait();
   a = -10;
   b = 23;
   wait();
   a = 25;
   b = -3;
   wait();
   sc_stop();
}
```

wait(): pause the execution of this function;
resume when any sensitivity signal is triggered

Assignment 3

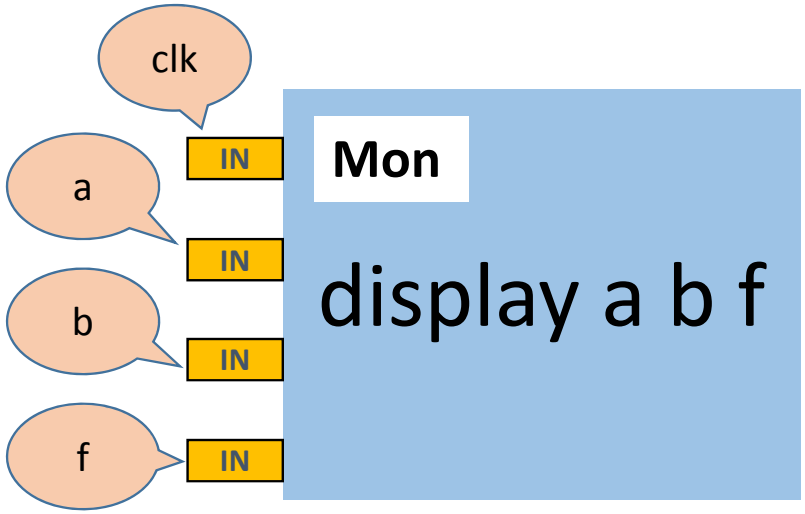
- In this example:




```
void Stim::action(){
   wait();
   a = 100;
   b = 200;
   wait();
   a = -10;
   b = 23;
   wait();
   a = 25;
   b = -3;
   wait();
   sc_stop();
}
```

Assignment 3

- Monitor
  - Design it yourself
  - Sensitive to negative edge of clock: clk.neg()
  - take a,b and f as input
  - display them
  - for example, to display “a”,
    printf("a=%",a->read());

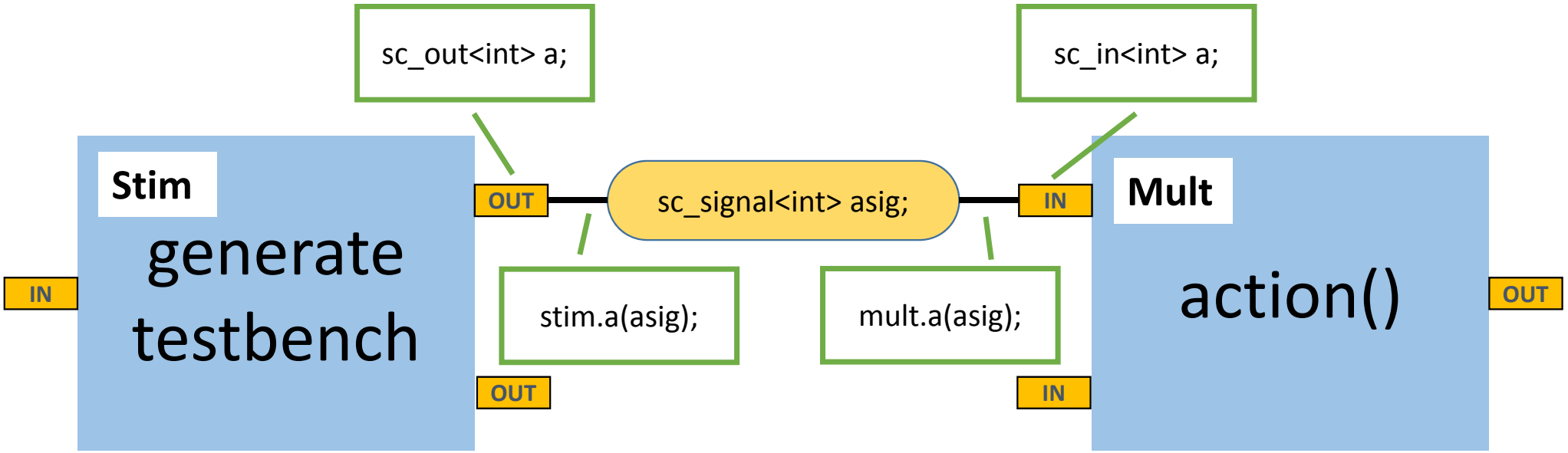


Assignment 3

- Top module:
  - Connects Stim, Mult and Mon
  - find the complete code in https://eee.uci.edu/17f/16905/DAC15_SystemC_Training.pdf, page 27-28

Assignment 3

- Two ports are connected via a channel



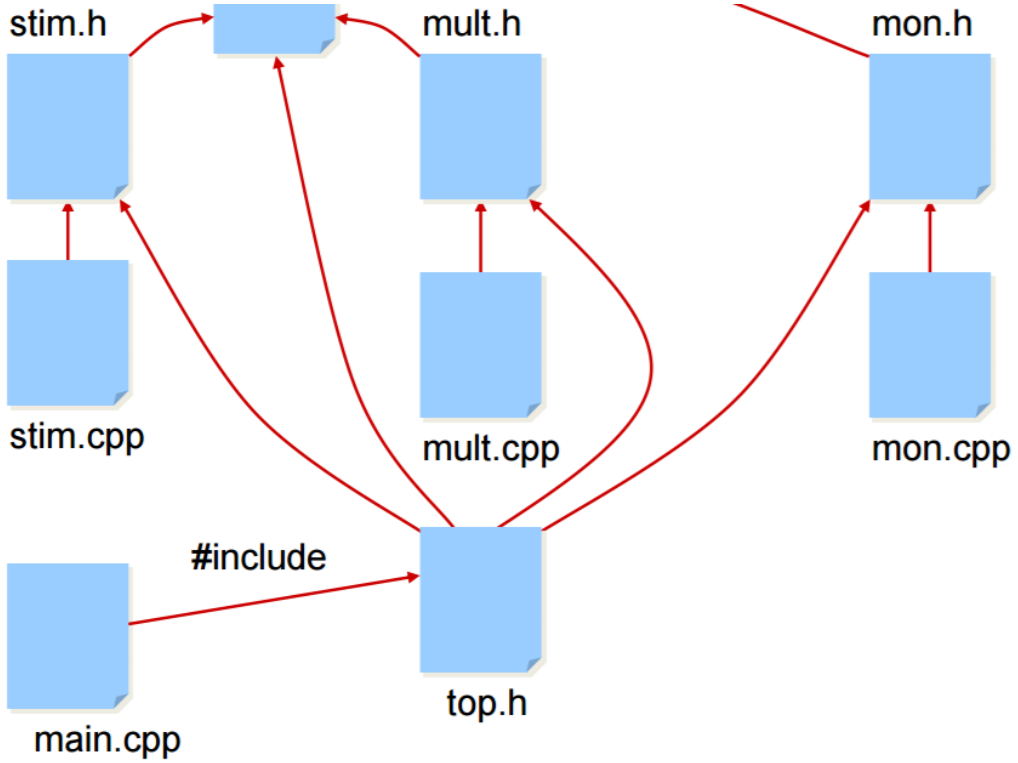
Assignment 3

- sc_main: the entrance to our SystemC code

```
int sc_main(int argc, char* argv[])
{
   Top top("top");
   sc_start();
   return 0;
}
```

Assignment 3

- File structure for our project
  - stim.h
  - stim.cpp
  - mult.h
  - mult.cpp
  - mon.h
  - mon.cpp
  - top.h
  - main.cpp
- SystemC.h is included in the library.



Assignment 3

- write module definition in .h file
- write function definition in .cpp file

```
#include “mult.h“

void Mult::action(){
   f=a*b;
}
```

mult.cpp

```
#include "systemc.h"
SC_MODULE(Mult)
{
    sc_in<int> a;
    sc_in<int> b;
    sc_out<int> f;

    void action(); //f=a*b

    SC_CTOR(Mult)
   {
      SC_METHOD(action);
      sensitive << a << b;
   }
};
```

mult.h

Assignment 3

- compile your code with a Makefile
- we already created a Makefile for you
  - get it from

~ecps203/public/Makefile

- to compile your code, type “make” in the command line
- it will generate an executable: hw3
- to run it, either:
  - ./hw3
  - or, make test

Assignment 3

- Submission
- create a package using the tarball
- gtar cvzf hw3.tar.gz README.txt Makefile *.cpp *.h
- README.txt: describe the troubles you cannot resolve
- and then use the ~ecps203/bin/turnin.sh script