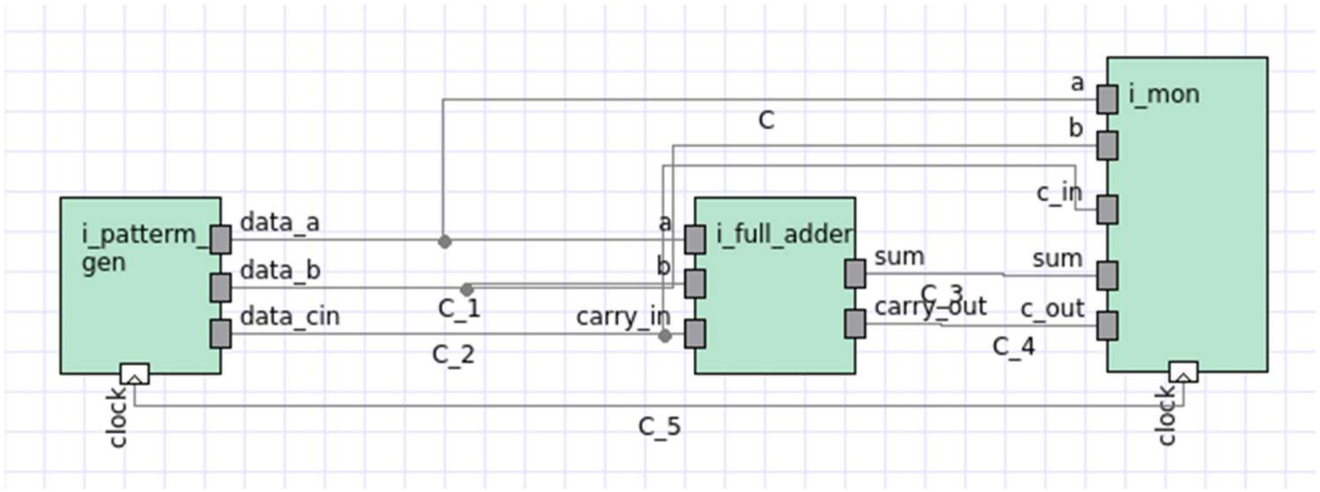
SystemC Exercise 1: Full Adder Implementation

Machine Learning Intelligent Chip Design
2024 Spring

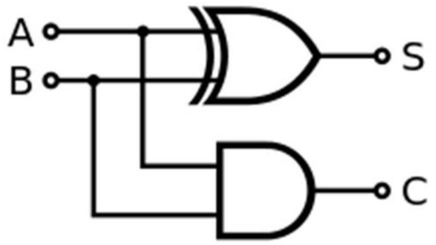
Agenda

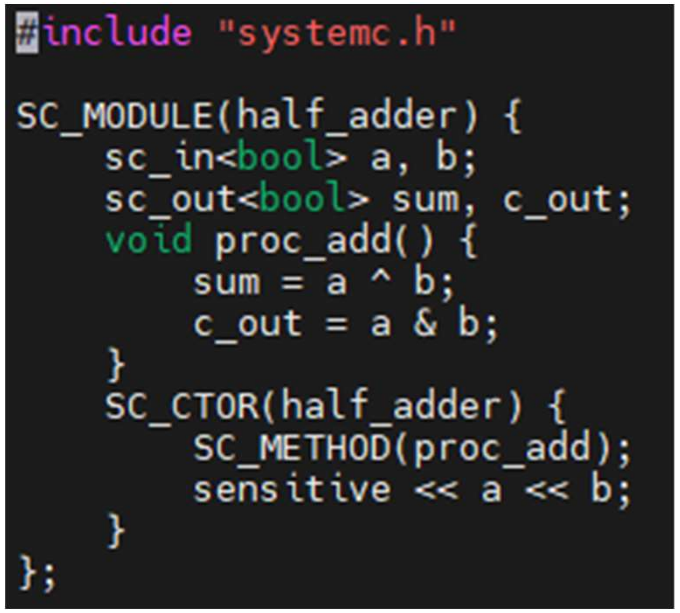
- Check Server connection
- Copy the Lab1 folder
  - /RAID2/COURSE/mlchip/mlchipTA01/sharing/lab1.zip
- Try run the lab1 material
  - **Hello**
  - **Datatype**
  - **Module**
  - **Full Adder**
- Lab1 explanation
  - **Full Adder Implementation**
- Do exercise
  - **Multiply Accumulation Unit**
- Check Result
  - Sign & Leave

Block Diagram



Half Adder



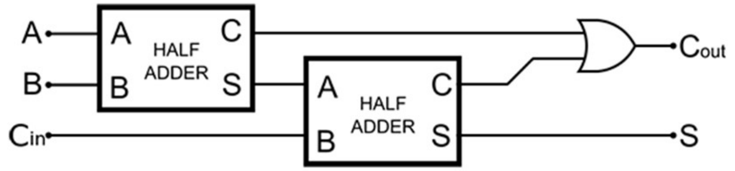


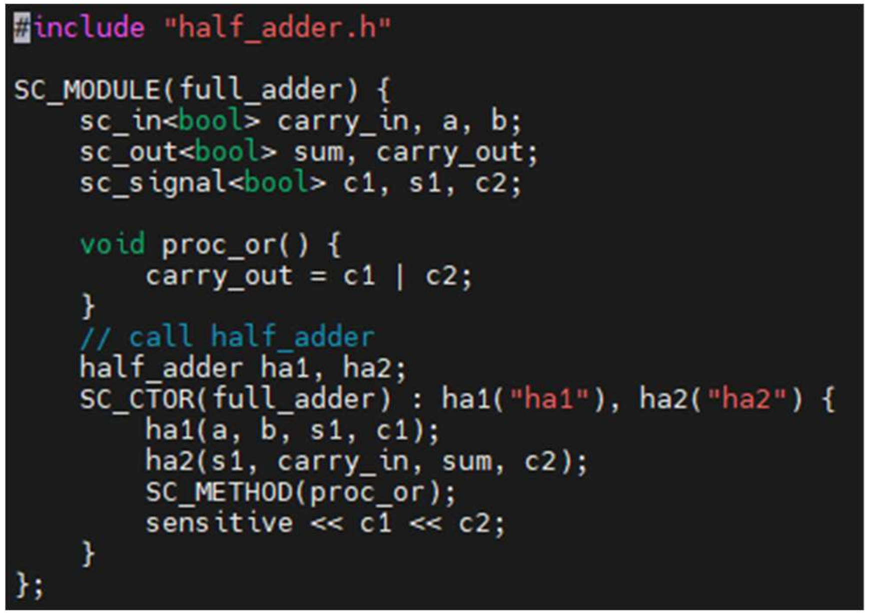
```
#include "systemc.h"

SC_MODULE(half_adder) {
    sc_in<bool> a, b;
    sc_out<bool> sum, c_out;
    void proc_add() {
        sum = a ^ b;
        c_out = a & b;
    }
    SC_CTOR(half_adder) {
        SC_METHOD(proc_add);
        sensitive << a << b;
    }
};
```

- Port
  - sc_in<data type>
  - sc_out<data type>

- SC_METHOD
  - Execution when trigger
  - No Suspend & Resume

Full Adder

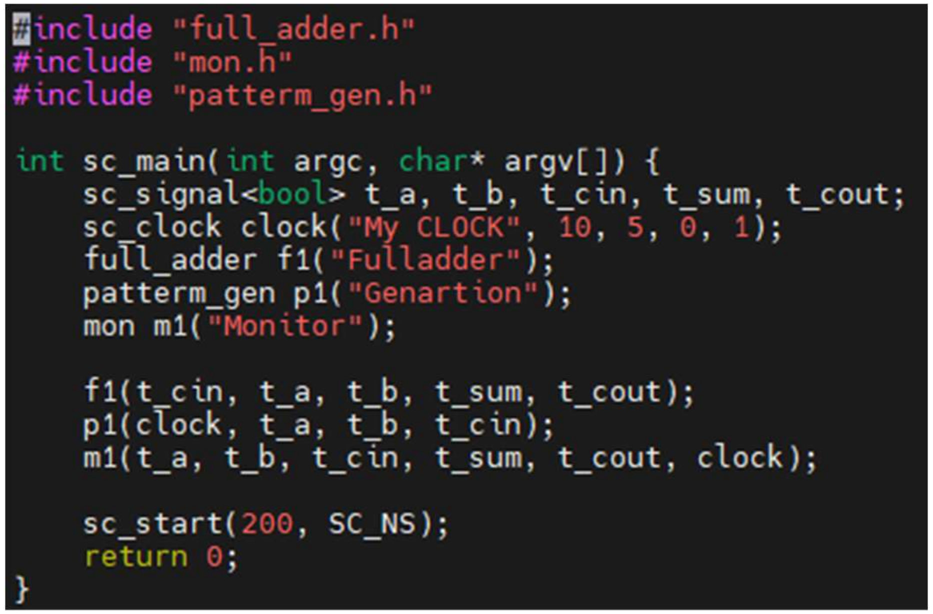




```
#include "half_adder.h"

SC_MODULE(full_adder) {
    sc_in<bool> carry_in, a, b;
    sc_out<bool> sum, carry_out;
    sc_signal<bool> c1, s1, c2;

    void proc_or() {
        carry_out = c1 | c2;
    }
    // call half_adder
    half_adder ha1, ha2;
    SC_CTOR(full_adder) : ha1("ha1"), ha2("ha2") {
        ha1(a, b, s1, c1);
        ha2(s1, carry_in, sum, c2);
        SC_METHOD(proc_or);
        sensitive << c1 << c2;
    }
};
```

- Constructor
  - Initializing / allocating sub-designs
  - Connecting sub-designs
  - Registering processes with the SystemC kernel
  - Providing static sensitivity
  - Miscellaneous user-defined setup
- SC_METHOD
  - Execution when trigger
  - No Suspend & Resume
  - RTL Model

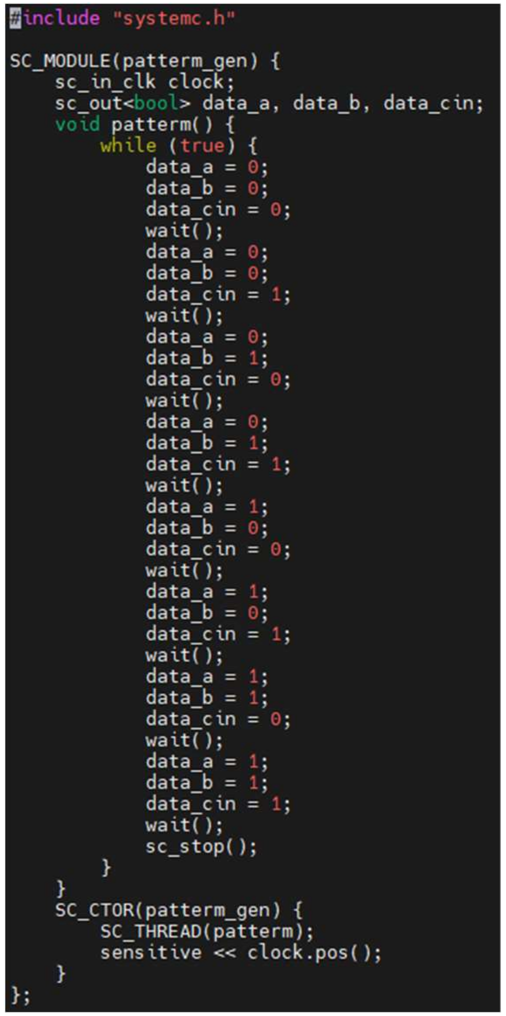
Main



```
#include "full_adder.h"
#include "mon.h"
#include "patterm_gen.h"

int sc_main(int argc, char* argv[]) {
    sc_signal<bool> t_a, t_b, t_cin, t_sum, t_cout;
    sc_clock clock("My CLOCK", 10, 5, 0, 1);
    full_adder f1("Fulladder");
    patterm_gen p1("Genartion");
    mon m1("Monitor");

    f1(t_cin, t_a, t_b, t_sum, t_cout);
    p1(clock, t_a, t_b, t_cin);
    m1(t_a, t_b, t_cin, t_sum, t_cout, clock);

    sc_start(200, SC_NS);
    return 0;
}
```

- Include the header of related module
- Instance module
    - Initialize parameter
- Elaboration stage
    - Connectivity for the model is established
- sc_start(value,sc_time_unit)
    - The value is the execution time

Pattern_Gen

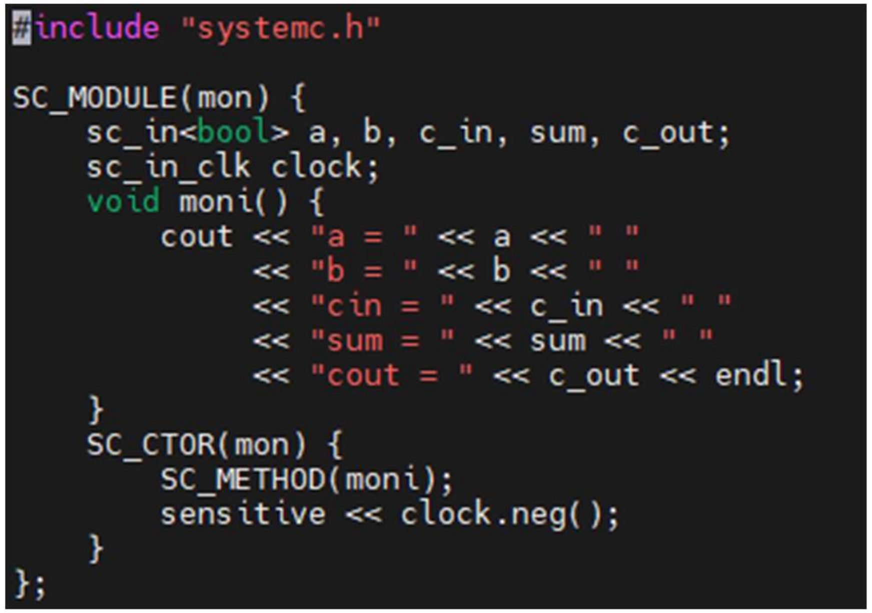


```
#include "systemc.h"

SC_MODULE(patterm_gen) {
    sc_in_clk clock;
    sc_out<bool> data_a, data_b, data_cin;
    void patterm() {
        while (true) {
            data_a = 0;
            data_b = 0;
            data_cin = 0;
            wait();
            data_a = 0;
            data_b = 0;
            data_cin = 1;
            wait();
            data_a = 0;
            data_b = 1;
            data_cin = 0;
            wait();
            data_a = 0;
            data_b = 1;
            data_cin = 1;
            wait();
            data_a = 1;
            data_b = 0;
            data_cin = 0;
            wait();
            data_a = 1;
            data_b = 0;
            data_cin = 1;
            wait();
            data_a = 1;
            data_b = 1;
            data_cin = 0;
            wait();
            data_a = 1;
            data_b = 1;
            data_cin = 1;
            wait();
            sc_stop();
        }
    }
    SC_CTOR(patterm_gen) {
        SC_THREAD(patterm);
        sensitive << clock.pos();
    }
};
```

- SC_THREAD
  - Always execute
  - Can Suspend & Resume
  - Sensitive by wait()
  - Behavioral model

- sc_stop
  - Stop simulation by user

Monitor



```
#include "systemc.h"

SC_MODULE(mon) {
    sc_in<bool> a, b, c_in, sum, c_out;
    sc_in_clk clock;
    void moni() {
        cout << "a = " << a << " "
             << "b = " << b << " "
             << "cin = " << c_in << " "
             << "sum = " << sum << " "
             << "cout = " << c_out << endl;
    }
    SC_CTOR(mon) {
        SC_METHOD(moni);
        sensitive << clock.neg();
    }
};
```

- Sensitive list
    - Check the result by negative clock