Pour Ingénieur 3ème année GE option: SMART + Mastères 2 : SSD & IST

Enseignante : Dr. Faten BEN ABDALLAH

TD SoC: Conception Haut niveau

Donner les résultats d'exécution des codes SystemC suivants :

```
// main.cpp

#include "EX1h"

int sc_main(int argc, char* argv[])
{
  sc_clock clock("clock", 15, SC_NS);
  Ex1 my_ex1("my_ex1");
  my_ex3.clk(clock);

  sc_start();
  return 0;
}
```

```
// EX1.h
#include<systemc.h>
SC_MODULE(Ex1)
{
  sc_in_clk clk;
  sc_event my_event;
  int count;
void event_filled()
{
  my_event.notify(2,SC_NS);
  wait();
  cout<<" @P1 "<<sc_time_stamp()<<" count= "<<count<<endl;
  count++;
  my_event.notify(2+count,SC_NS);
  wait();
  cout<<" @P1 "<<sc_time_stamp()<<" count= "<<count<<endl;
  sc_stop();
}

void monitor()
{
      while(true)
      {
wait(my_event);
cout<<" @P2 "<<sc_time_stamp()<<" count= "<<count<<endl;
wait(3,SC_NS);
count++;
cout<<" @P2 "<<sc_time_stamp()<<" count= "<<count<<endl;
      }
}


  SC_CTOR(Ex1): count(0)
  {
    SC_THREAD(event_filled);
    sensitive<<clk.pos();
    dont_initialize();
    SC_THREAD(monitor);
  }
};
```

```
// main.cpp
#include "ex2.h"

int sc_main (int argc, char* argv[]) {

sc_clock clock("clock",1,SC_NS);

 ex2 votreTD("votreTD1");
 votreTD.clock (clock);

cout <<"Starting simulation" <<endl;
sc_start();

  return 0;
}
```

```
// ex2.h
#include <systemc.h>

SC_MODULE(ex2)
{
sc_in_clk clock;
  sc_event  e1,e2;
  int cnt;

  SC_CTOR(ex2)
{
   cnt = 0;
    SC_METHOD(do_test1);
      sensitive << clock.pos();

SC_CTHREAD(do_test2,clock.pos());
  }
void do_test1();
  void do_test2();
};
```

```
// ex2.cpp
#include " ex2.h"
void ex2:: do_test1()
{ switch (cnt) {
      case 0 : cout << "@" << sc_time_stamp() <<" cnt= "<< cnt << endl;
               next_trigger(e1);
               break;
      case 1 : cout << "@" << sc_time_stamp() <<" cnt= "<< cnt << endl;
               next_trigger(10, SC_NS);
               break;
      case 2 : cout << "@" << sc_time_stamp() <<" cnt= "<< cnt << endl;
               next_trigger(e1 | e2);
               break;
      case 3 : cout << "@" << sc_time_stamp() <<" cnt= "<< cnt << endl;
               break;
      default : cout << "@" << sc_time_stamp() <<" cnt= "<< cnt << endl;
               break;
    }
    cnt ++;
 }
void ex2::do_test2()
{
while (true) {
      wait(2);
      cout << "@" << sc_time_stamp() <<" Triggering e1"<<endl;
      e1.notify();
      wait(20);
      cout << "@" << sc_time_stamp() <<" Triggering e2"<<endl;
      e2.notify();
      wait(2);
      cout << "@" << sc_time_stamp() <<" Terminating simulation"<<endl;
      sc_stop();
    }
}
```

```
// main.cpp


#include "Ex3.h"

int sc_main(int argc, char* argv[])
{
sc_clock clock("clock", 2, SC_NS);

Ex3 myEx3("myEx3");

myEx3.clk (clock);

sc_start(12,SC_NS);

return 0;
}
```

```
// Ex3.h
#include <systemc.h>
SC_MODULE(Ex3)
{
  sc_event my_event;
  sc_in_clk clk;
  sc_signal<int> my_count, better_count;
  int count;
SC_CTOR(Ex3): my_count(0), better_count(0),
count(0)
  {
    SC_THREAD(process1);
    sensitive<<better_count;
    SC_THREAD(process2);
    sensitive<<clk.pos();
    SC_METHOD(process3);
    sensitive<<clk.pos();
  }
void process1(); void process2();void process3();
};
```

```
// Ex3.cpp
#include "Ex3.h"
void Ex3::process1()
{
       while(true){
         my_event.notify(3,SC_NS);
         wait();
         cout<<" P1@ "<<sc_time_stamp()<<" better_count= "<<better_count<<endl;
         count++;
         my_count.write(count);
         my_event.notify(my_count,SC_NS);
         wait(count,SC_NS);
         my_count.write(my_count.read()+1);
         cout<<" P1@ "<<sc_time_stamp()<<" my_count= "<<my_count<<endl;
         my_event.notify(count,SC_NS);
         wait();
         cout<<" P1@ "<<sc_time_stamp()<<" better_count= "<<better_count<<endl;
         my_event.notify(better_count.read(),SC_NS);
                 }
}
void Ex3::process2()
{
       while(true){
              wait(2);
              cout<<" P2@ "<<sc_time_stamp()<<" my_count= "<<my_count<<endl;
              better_count.write(better_count.read()+1);
              wait(my_event);
              better_count.write(better_count.read()+1);
              cout<<" P2@ "<<sc_time_stamp()<<" better_count= "<<better_count<<endl;
              wait();
              cout<<" P2@ "<<sc_time_stamp()<<" my_count= "<<my_count<<endl;
              better_count.write(better_count.read()+1);
                  }
}
void Ex3::process3()
{
 if(count<3)
              {next_trigger(my_event);
              cout << " P3@ "<<sc_time_stamp()<<" count= "<< count <<endl; }
              else
              {next_trigger();
              cout <<" P3@ "<< sc_time_stamp()<<" count= "<< count <<endl;}
count++;
}
```