Problem-Set #1 – COE838

Introduction and SystemC

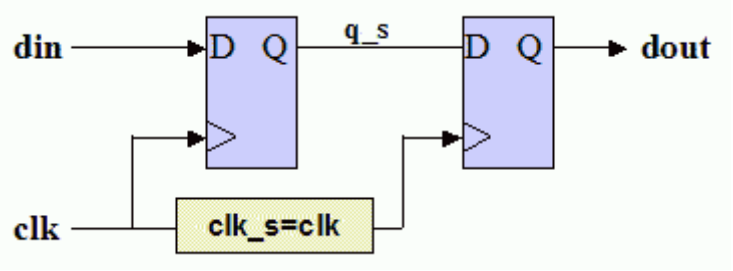
Q-1. Suppose we have four different processors; each does 25% of the application. If we improve two of the processors by 10 times, what would be the overall application speedup?

Q-2. Write a SystemC module to model a D-type flip-flop.

Q-3. Identify the main function of the following SystemC module. Explain the module briefly.

```
SC_MODULE(what) {
   sc_in<bool >  clk;
   sc_out<bool > dout1;
   sc_out<bool > dout2;
   sc_signal<bool> sig1;
   bool var2 ;
     void p1() {     // using signals
                    sig1.write(!sig1.read());
                    dout1.write(sig1.read());
     }
    void p2() {    // using variables
                    var2 = !var2;
                    dout2.write(var2);
    }
   SC_CTOR(what) {
          sig1.write(true);   // init
          var2=false;        // init
          SC_METHOD(p1);
                    sensitive << clk.pos();
                    dont_initialize();
          SC_METHOD(p2);
                    sensitive << clk.pos();
                    dont_initialize();
   }
};
```

Q-4. Model the following two flip-flop hardware, by writing the SystemC code as a module. The module can have multiple METHODS or processes.



Q-5. SystemC code of a counter module is given below. Identify the type of counter and which signal it counts. Justify your answer.

```
#include "systemc.h"
SC_MODULE (which_counter) {
        //-----------Input Ports---------------
        sc_in    <bool> enable, clk, reset;
        //-----------Output Ports---------------
        sc_out   <sc_uint<8> > out;
        //------------Internal Variables--------
        sc_uint<8>  count;

        //-------------Code Starts Here---------
        void counter () {
          if (reset.read()) {
                count = 0 ;
          } else if (enable.read()) {
                count = count + 1;
            }
         out.write(count);
        }

 SC_CTOR(which_counter) {
        SC_METHOD (counter);
        sensitive << clk.pos();
 }
};
```

Q-6. Design and write the SystemC code of a down-counter module that will initialize its count at 1024 and then count the number of input pulses.

Q-7. Design and write the SystemC code for asynchronous read/write random access memory (RAM) module. The memory size is 512 bytes, and the word size of the memory is of 16-bits. The memory has separate data-in and data-out ports of 16-bits each. In addition to memory address signals, the memory module has chip select (cs), output enable (oe) and write enable (we) input signals.