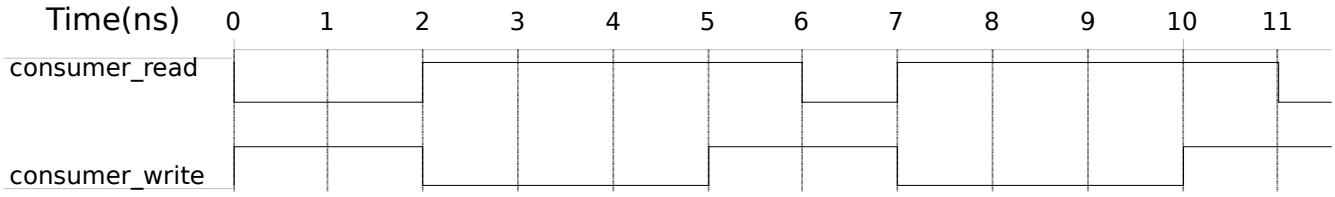
Bonus Question:

You are given a simple SystemC design (`bonus.cpp`) created by a young engineer at a chip design company. The engineer needs your help in understanding the timing waveforms. Please:

1. Execute the design. You can follow the same procedure as in the lab session, but please change the name of the design file from `simple_fifo.cpp` to `bonus.cpp`.
2. Review the waveforms in `gtkwave` and explain it. Make sure that you explain all transitions.

For your convenience, the first 10ns of the waveform is attached:



The `bonus.cpp` is also provided on the next pages (and also separately for you to run).

```
#include <systemc.h>

class write_if : virtual public sc_interface {
public:
    virtual void write(char) = 0;
    virtual void reset() = 0;
};

class read_if : virtual public sc_interface {
public:
    virtual void read(char &) = 0;
    virtual int num_available() = 0;
};

class fifo : public sc_channel, public write_if,
             public read_if {
public:
    fifo(sc_module_name name) : sc_channel(name),
                                num_elements(0),
                                first(0) {}

    bool reading;
    bool writing;

    void write(char c) {
        if (num_elements == max)
            wait(read_event);

        writing = 1;
        wait(1, SC_NS);
        data[(first + num_elements) % max] = c;
        ++ num_elements;
        writing = 0;
        write_event.notify(1, SC_NS);
    }

    void read(char &c) {
        if (num_elements == 0)
            wait(write_event);

        reading = 1;
        wait(2, SC_NS);
        c = data[first];
        -- num_elements;
        first = (first + 1) % max;
        reading = 0;
        read_event.notify(1, SC_NS);
    }

    void reset() { num_elements = first = 0; }

    int num_available() { return num_elements;}

private:
    enum e { max = 2 };
    char data[max];
    int num_elements, first;
    sc_event write_event, read_event;
};

//class producer : public sc_module {
SC_MODULE(producer) {
 public:
    sc_port<write_if> out;

    SC_CTOR(producer){
        SC_THREAD(main);
```

```
    }

    void main() {
        const char *str =
            "Visit www.systemc.org and see what"
            "SystemC can do for you today!\n";

        while (*str)
            out->write(*str++);
    }
};

SC_MODULE(consumer) {
 public:
    sc_port<read_if> in;

    SC_CTOR(consumer) {
        SC_THREAD(main);
    }

    void main() {
        char c;
        cout << endl << endl;

        while (true) {
            in->read(c);
            cout << c << flush << "\n";

              if (in->num_available() == 1)
              cout << "<1>" << flush << "\n";
              if (in->num_available() == 9)
              cout << "<9>" << flush << "\n";

              cout << "time used:" << sc_time_stamp() << "\n";
        }
    }
};

//class top : public sc_module {
SC_MODULE(top) {
   public:
     fifo *fifo_inst;
     producer *prod_inst;
     consumer *cons_inst;

     //     top(sc_module_name name) : sc_module(name) {
     SC_CTOR(top) {
       fifo_inst = new fifo("Fifo1");

       prod_inst = new producer("Producer1");
       prod_inst->out(*fifo_inst);

       cons_inst = new consumer("Consumer1");
       cons_inst->in(*fifo_inst);
     }
};

int sc_main (int argc , char *argv[]) {
   top top1("Top1");

   sc_trace_file *tf;
   tf = sc_create_vcd_trace_file("trace");
   sc_trace(tf, top1.fifo_inst->reading, "consumer_read");
   sc_trace(tf, top1.fifo_inst->writing, "consumer_write");

   sc_start(50, SC_NS);
   cout << "time used: " << sc_time_stamp() << "\n";
```

```
   sc_close_vcd_trace_file(tf);
   return 0;
}
```