Electrical and Computer Engineering Department

ENCS5131 Hardware Design Laboratory

Design Verification Part

Experiment No. 2: Advanced SystemVerilog Concepts

MailBox todo

Instructor

Dr. Abdellatif Abu-Issa

Teaching Assistants

Eng.Tala Abahra

Student Name

Leen Aldeek

Student ID

1212391

Section 1

Date

14/10/2025

Implement a mailbox class in SystemVerilog

Description:

In this todo, you are required to implement the functionality of a mailbox using Object-Oriented Programming (OOP) concepts in SystemVerilog and a testbench for it .

---

1. Provide the full implementation of the class my_mailbox

```
// Code your design here
class my_mailbox;
  protected string mail[$];//mail queue
  protected int mail_size;// = 0 for unlimited size
  protected int s; //semaphore value

  //constructor
  function new(int max_size);
  //  mail = new();
    mail_size = max_size;
    s = 1;// s =1 means the shared resourse is free, s = 0 means it is not free
  endfunction

// put method with blocking
    task automatic put(string new_mail);
      forever begin
        wait (s == 1);// wait until resource "mail box" is available to put in
        s = 0; // we put the mail in the mailbox
        //check mailbox space
        if(mail_size == 0 || mail.size() < mail_size) begin
```

```
        mail.push_back(new_mail); //add mail
        s = 1; //release to allow ading other mails if we have capacity
        return;
      end

      s = 1; // relese resource if full
      #1;
    end
  endtask

// try put method(non blocking)
function bit try_put(string new_mail);
      bit success = 0;
      if (s == 1) begin
        s = 0;
        if (mail_size == 0 || mail.size() < mail_size)          begin
            mail.push_back(new_mail);
            success = 1;
         end
         s = 1;
      end
      return success; // 1 if successful, 0 if mailbox full or busy
   endfunction


//get method with blocking
```

```
  task automatic get(output string get_mail);
    forever begin
      wait(s == 1);
      s = 0;

      //check if mailbox has mails to get
      if(mail.size() > 0) begin
        get_mail = mail.pop_front();
        s = 1;
        return;
      end

      s = 1;
      #1;
    end
  endtask

// try get method (non blocking)
 function string try_get();
      string get_mail = "";
      if (s == 1) begin
         s = 0;
         if (mail.size() > 0) begin
            get_mail = mail.pop_front();
         end
         s = 1;
```

```
      end
      return get_mail; // empty string if mailbox empty or busy
    endfunction


//peek method(blocking)---talk a look
  task automatic peek(output string peek_mail);
    forever begin
      wait (s == 1);
      s = 0;

      if(mail.size() > 0) begin
        peek_mail = mail[0];
        s = 1;
        return;
      end

      s = 1;
      #1;
    end
  endtask


// try peek method(non blocking)
 function string try_peek();
      string peek_mail = "";
```

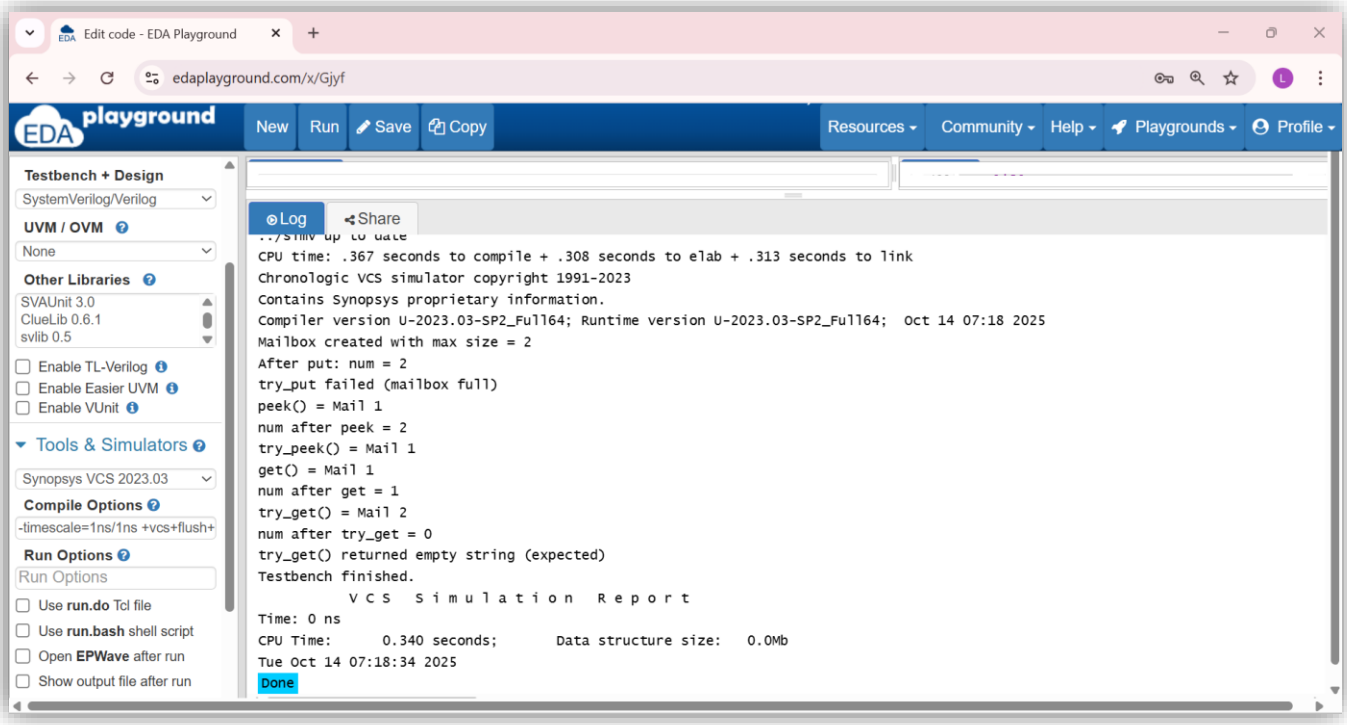
```
        if (s == 1) begin
          s = 0;
          if (mail.size() > 0) begin
            peek_mail = mail[0]; // first mail without removing
          end
          s = 1;
        end
        return peek_mail; // empty string if mailbox empty or busy
      endfunction


//mails number method
   task automatic num(output int mail_number);
    wait(s == 1);
    s = 0;
    mail_number = mail.size();
    s = 1;

endtask
endclass


//get method with priority
task automatic get(output string get_mail);
  forever begin
    wait (s == 1);
```

```
    s = 0;
    if(MAIL.size() > 0) begin
      int index = 0;
      mp max = MAIL[0].prior;
      for(int i = 1; i < MAIL.size(); i++) begin
        if(MAIL[i].prior > max) begin
          max = MAIL[i].prior;
          index = i;
        end
      end
      get_mail = MAIL[index].message;
      MAIL.delete(index);
      s = 1;
      return;
    end
    s = 1;
    #1;
  end
endtask
endclass
```

2. Write a small testbench that demonstrates each function.

```
// testbench
module tb;
  my_mailbox mb;
  string mail_out;
  bit success;
  int count;

  initial begin
    mb = new(2);
    $display("Mailbox created with max size = 2");

    mb.put("Mail 1");
    mb.put("Mail 2");
    mb.num(count);
    $display("After put: num = %0d", count);

    success = mb.try_put("Mail 3");
    if (success)
      $display("try_put success");
    else
      $display("try_put failed (mailbox full)");

    mb.peek(mail_out);
```

```
    $display("peek() = %s", mail_out);
    mb.num(count);
    $display("num after peek = %0d", count);

    mail_out = mb.try_peek();
    $display("try_peek() = %s", mail_out);

    mb.get(mail_out);
    $display("get() = %s", mail_out);
    mb.num(count);
    $display("num after get = %0d", count);

    mail_out = mb.try_get();
    $display("try_get() = %s", mail_out);
    mb.num(count);
    $display("num after try_get = %0d", count);

    mail_out = mb.try_get();
    if (mail_out == "")
      $display("try_get() returned empty string (expected)");
    else
      $display("try_get() = %s", mail_out);

    $display("Testbench finished.");
  end
endmodule
```



3. **Bonus (Optional): Create a subclass priority_mailbox that allows each mail to have a priority field and always returns the highest-priority mail first**

```
//mail priority
class mailbox_priority extends my_mailbox;
//mail priority
  typedef enum {high = 2, MEDIUM = 1, low = 0} mp;

  typedef struct{

    string message;
    mp prior;

  } priority_mail;
```

```
  local priority_mail MAIL[$];


//constructor
  function new(int max_size);
    super.new(max_size);
  endfunction
//put method with priority
  task automatic put(string str,int pri);
    priority_mail m;
    m.message = str;
   m.prior = mp'(pri); // cast int to enum mp
    forever begin
      wait (s == 1);
      s = 0;


      if(mail_size == 0 || MAIL.size() < mail_size) begin
        MAIL.push_back(m);
        s = 1;
        return;
      end


      s = 1;
      #1;
    end
  endtask
```

```
//get method with priority
task automatic get(output string get_mail);
  forever begin
    wait (s == 1);
    s = 0;
    if(MAIL.size() > 0) begin
      int index = 0;
      mp max = MAIL[0].prior;
      for(int i = 1; i < MAIL.size(); i++) begin
        if(MAIL[i].prior > max) begin
          max = MAIL[i].prior;
          index = i;
        end
      end
      get_mail = MAIL[index].message;
      MAIL.delete(index);
      s = 1;
      return;
    end
    s = 1;
    #1;
  end
endtask
endclass
```