Date
Page

Arti tyagi- System Verilog 2023

⊙ Mailboxes

- A mailbox is a communication mechanism That allows message to be exchanged between process.
- Data can be sent to a mailbox by one process and retrived by another.

- mailbox is a built-in class that provides methods:

I.> new();-
∴ Create a mailbox.

II.> put();-
∴ place a message in a mailbox

III.> get() or peek():-
∴ Retrive a message from a mailbox.

NOTE: put() and get() method of mailbox is Blocking nature.

IV.> num():-
∴ Retrive the number of message in the mailbox.

V.> try_put():-
Try to place a message in a mailbox without Blocking.

VII> try_get() or try_peek:-
∴ Try to retrive a message from a mailbox without Blocking.

- mailbox behaves as FIFO [FIRST-IN , FIRST-OUT].

◉ SYNTAX OF mailBOX :

mailbox mailbox_name;

◉ mailbox Types:

i.) Bounded mailbox

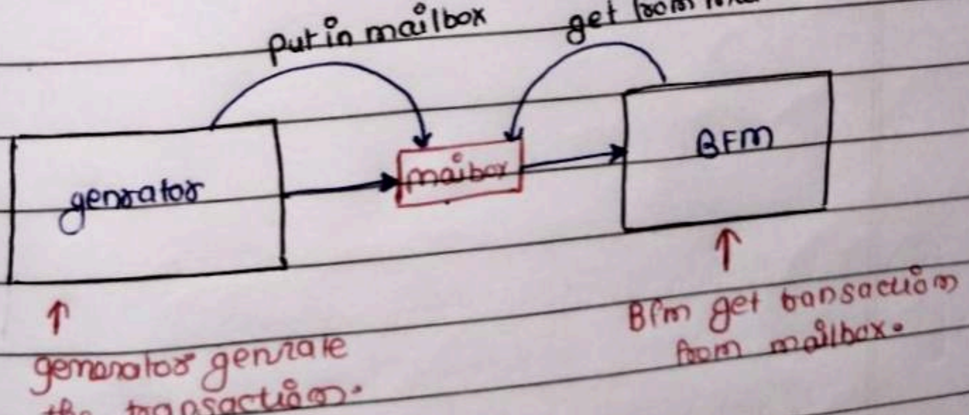
- number of entries is determined when the mailbox is created.
- put () will be blocked if the mailbox is full.

ex- mailbox mbox = new (5);

∴ here mailbox of depth = 5

ii.) UnBounded mailbox :

- No restrictions placed on size of mailbox.
- put() will never block.
- eg:- mailbox mbox = new();


put in mailbox
get from mailbox
generator
mailbox
BFM
generator generate the transaction.
BFM get transaction from mailbox.

CODE1: Mailbox is used for communication between generator and BFM(bus functional module)

Example for mailbox :

Step 1: generator genarate the sample packet and put into the mailbox.

Code for : Sample_packet

```
class sample_pkt;
rand bit [7:0]addr;
rand bit[7:0]data;

function void post_rand();
        $display("packet generated");
        $display("packet::
addr=%0d,Data=%0d",addr,data);
endfunction
endclass
```

Step2:
Code for Genrator class: generator generate the transaction packet

```
mailbox mbox=new();
class generator;
sample_pkt pkt;

task run();
        repeat(2)begin
        pkt = new();
        pkt.randomize();
        mbox.put(pkt);
        $display("generator::packet put into mailbox");
        #5;
        end
endtask
endclass
```

Step3:

Code for Bfm : gets the sample packet from generator

```
class pkt_bfm;
sample_pkt pkt;

task run();
            repeat(2)begin
            mbox.get(pkt);
            $display("bfm::packet get from mailbox");
            $display("packet:: addr=%0d,Data=%0d",pkt.addr,pkt.data);
            #10;
            end
endtask
endclass
```

Step4:

Code for Top_module

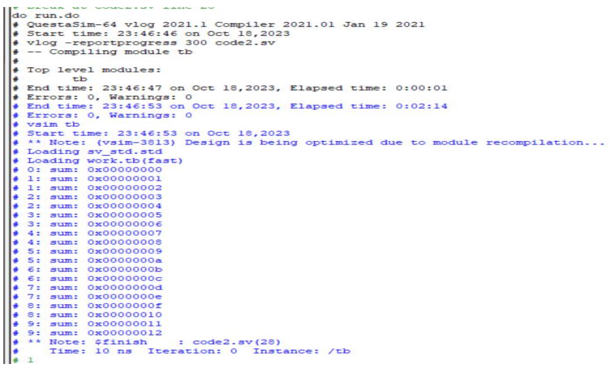
```
`include "sample_pkt.sv"    //Forward declaration
`include "pkt_bfm.sv"
`include "pkt_gen.sv"
module top;
generator gen;
pkt_bfm bfm;
initial begin
            gen=new();
            bfm=new();
            fork
            gen.run();
            bfm.run();
            join
end
endmodule
```

CODE2: mailbox methods

```
module tb;
mailbox mbox1;
mailbox mbox2;
int sum;

initial begin
        sum=0;
        forever begin
        mbox2.get(sum);
        $display("%0t: sum: 0x%8x",$time,sum);
        #1;
        sum++;
        mbox1.put(sum);
        end
end

initial begin
        forever begin
        #1;
        mbox2.put(sum);
        mbox1.get(sum);
        sum++;
        $display("%0t: sum: 0x%8x",$time,sum);
        end
end
initial begin
#10;
$finish();
end
endmodule
```



```
do run.do
# QuestaSim-64 vlog 2021.1 Compiler 2021.01 Jan 19 2021
# Start time: 23:46:46 on Oct 18,2023
# vlog -reportprogress 300 code2.sv
# -- Compiling module tb
#
# Top level modules:
#       tb
# End time: 23:46:47 on Oct 18,2023, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
# End time: 23:46:53 on Oct 18,2023, Elapsed time: 0:02:14
# Errors: 0, Warnings: 0
# vsim tb
# Start time: 23:46:53 on Oct 18,2023
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.tb(fast)
# 0: sum: 0x00000000
# 1: sum: 0x00000001
# 1: sum: 0x00000002
# 2: sum: 0x00000003
# 2: sum: 0x00000004
# 3: sum: 0x00000005
# 3: sum: 0x00000006
# 4: sum: 0x00000007
# 4: sum: 0x00000008
# 5: sum: 0x00000009
# 5: sum: 0x0000000a
# 6: sum: 0x0000000b
# 6: sum: 0x0000000c
# 7: sum: 0x0000000d
# 7: sum: 0x0000000e
# 8: sum: 0x0000000f
# 8: sum: 0x00000010
# 9: sum: 0x00000011
# 9: sum: 0x00000012
# ** Note: $finish    : code2.sv(28)
#    Time: 10 ns  Iteration: 0  Instance: /tb
# 1
```