Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

Events

- Events are constructs that allow hand shake between two processes
  - Producer-consumer model where consumer is blocked until producer produces
  - Can block untill an event is triggered
  - Or block on a mailbox
- Triggered using -> operator.
  - Triggering unblocks all waiting processes
- @ is used to wait for an event

Event - example

Example 7-32 Producer–consumer synchronized with an event

```
program automatic mbx_evt;

  event handshake;

  class Producer;
    task run;
      for (int i=1; i<4; i++) begin
        $display("Producer: before put(%0d)", i);
        mbx.put(i);
        @handshake;
        $display("Producer: after  put(%0d)", i);
      end
    endtask
  endclass
```

Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

Event - example

```
  class Consumer;
    task run;
      int i;
      repeat (3) begin
        mbx.get(i);
        $display("Consumer: after  get(%0d)", i);
        ->handshake;
      end
    endtask
  endclass
...
endprogram
```

Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

Event Variables

Merging Events

When one event variable is assigned to another, both merge into one event variable.

Executing -> on either one of the events affects processes waiting on either event variable.

```
event a, b;
a = b;
-> a; // also triggers b
-> b; // also triggers a
```