

Make Code As Sequential As Possible

A static construct is widely used because of its elegant looks but it becomes very problematic during the debugging stage. When an event is emitted its associated temporal expression becomes true, and can trigger other events, TCMs and on clauses.

The data valid (rx_dv) and carrier rise (crs) go up with the first byte of packet, carrier falls with the end of the packet. Thus, at the beginning of the packet the signals packet_start and carrier_rise are asserted together, and the same goes for the signals packet_end and carrier_fall at the end of the packet. Whenever a packet starts the event packet_start is emitted, therefore leading to the execution of the on clause, which is just like a method that does not return a value. The other events cause the execution of the rest of the on clauses.

```
<
unit monitor {
  event clk is rise('clk')@ sim;

  event packet_start is true('rx_dv' === '1') @ clk;
  event carrier_rise is true('crs' === '1') @ clk;

  event packet_end is true('rx_dv' === '0') @ clk;
  event carrier_fall is true('crs' === '0') @ clk;

  !unpredictable : uint;

  on packet_start {
    unpredictable = 1;
  };

  on packet_end {
    unpredictable = 2;
  };

  on carrier_rise {
    unpredictable = 3;
  };

  on carrier_fall {
    unpredictable = 4;
  };
};
>
```

The risks in this kind of coding is that each of the four events with its corresponding on clause is an independent process, or a separate thread of execution. The reason for this is simple: since each process runs separately, their order of execution is totally random. For example, consider the field unpredictable. Since at the end of the packet, the events packet_end and carrier_fall are emitted simultaneously, there is no way of knowing which on clause will be executed first. Therefore, after the packet ends, the value of unpredictable is in fact unpredictable, since it can be either 2, when on packet_end is last, or 4, when on carrier_fall is last. The behavior of a code like this can be consistent for months, and then change suddenly for no apparent reason, which makes this kind of bugs particularly catastrophic.

The solution is to replace the static parts with a sequential code. This time, the four events are replaced with a TCM, and the on clauses are replaced with methods that do not return a value:

```
<'
unit ei_gbe_scoreboard is {
  //...
  event clk is rise('clk')@ sim;

  packet_check()@ clk is {
    while(TRUE) {
      wait true('rx_dv'==1 and 'crs' == 1);
      packet_start();
      carrier_rise();
      wait true('rx_dv'==0 and 'crs'==0);
      carrier_fall();
      packet_end();
    };
  };

  !predictable : uint;

  packet_start() is {
    predictable = 1;
  };

  packet_end() is {
    predictable = 2;
  };

  carrier_rise() is {
    predictable = 3;
  };

  carrier_fall() is {
    predictable = 4;
  };
};
>
```

Predictable can be seen, true to its name, has a predictable value of 2 after the packet ends. Once again, this code is certainly safer and easier to debug. Using this type of sequential, independent events and threads can be minimized in the code.