2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

# Modelling of UVC Monitor Class as a Finite State Machine for a Packet-Based Interface

Djordje Velickovic, Verification Engineer, Veriest Solutions

Milos Mitic, Verification Engineer, Veriest Solutions

Veriest

accellera SYSTEMS INITIATIVE

# Agenda

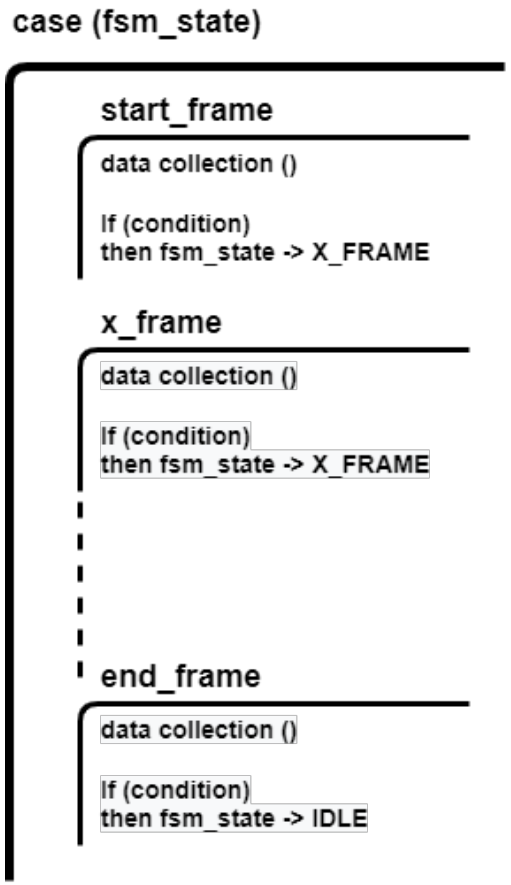
- How was the idea born?
- Concept of modelling of UVC monitor Class as an FSM for a packet-based interface
- Implementation of UVC monitor class for exemplary interface
- Benefits
- Results
- Conclusion

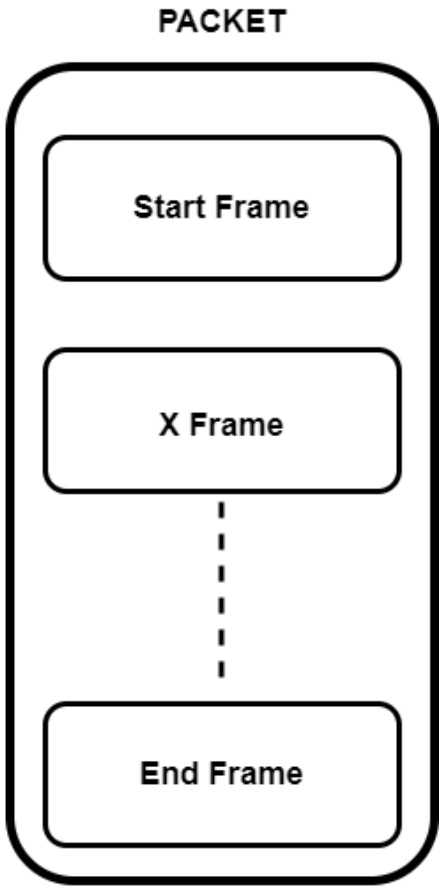
# How was the idea born?

- Best quality VIPs need time and resources
- Packet Based Interfaces could be difficult to model
  - Large Data Packets
  - Interface Signals Behavior
- Packet-based Interface and FSM – Natural connection
- Everybody knows about FSMs
- Development time
- Debug tool

# Concept

- FSM states -> Packet frames
- State “jumping”
- Coding simplicity
- Protocol violations checks
- Knowledge of interface functionality
- Debug benefits



# Implementation

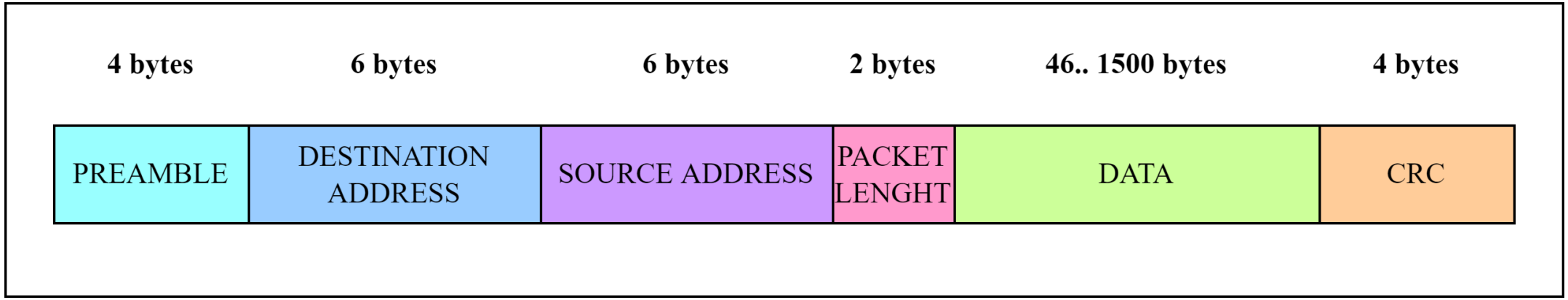
- UVM
- Definition of interface used for showcasing the implementation
- Coding the FSM model inside the monitor class
- Implementation of protocol checkers

# Definition of interface – signal list

| Name | Width | Direction | Description |
|---|---|---|---|
| MRxDV | 1 bit | input | Valid indication |
| MRxD | 4 bits | input | Data bus |
| r_Pro | 1 bit | input | Promiscuous mode control signal - all frames are received regardless of their destination address |
| r_Bro | 1 bit | input | Broadcast mode control signal - all frames containing broadcast addresses are rejected |
| MAC | 48 bits | input | MAC address of the used Ethernet MAC IP Core |
| MRxClk | 1 bit | input | Clock |
| Reset; | 1 bit | input | Reset |

# Definition of interface

- Frames in the packet



# Task for packet detection

- Started in the run phase
- New packet detection
- Create a new item
- Collect address and other control signals
- Call packet collection task

```
01  virtual task detect_packet();
02
03      fsm = IDLE;
04
05      forever begin
06          if (vif.slv_cb.MRxDV == 1'b1) begin
07              fsm = PREAMBLE;
08              item = uvm_eth_vip_item::type_id::create("item");
09              item.MAC= vif.slv_cb.MAC;
10              item.r_Pro = vif.slv_cb.r_Pro;
11              item.r_Bro = vif.slv_cb.r_Bro;
12              collect_packet();
13          end
14      else
15          @vif.slv_cb;
16      end
17  endtask
```

# Task for packet detection

- Collect frame by frame
- Each frame – unique FSM value
- Conditions for “state jumping”
- End of packet condition

```
01  virtual task collect_frame ();
02
03      while(fsm != IDLE) begin
04          if (vif.slv_cb.MRxDV == 1'b1) begin
05              case (fsm)
06                  PREAMBLE:
07                      begin
08                          item.preamble[(preamble_cnt-1)*4 +: 4] = vif.slv_cb.MRxD;
09                          preamble_cnt--;
10                          if (preamble_cnt == 0)begin
11                              fsm =  DEST_ADDR;
12                              preamble_cnt = 8;
13                          end
14                      end
15                  DEST_ADDR:
16                      begin
17                          item.dest_addr[(dest_add_cnt-1)*4 +: 4] = vif.slv_cb.MRxD;
18                          dest_add_cnt--;
.                       .
.                       .
.                       .
52                  CRC:
53                      begin
54                          item.CRC[(crc_cnt-1)*4 +: 4] = vif.slv_cb.MRxD;
55                          crc_cnt--;
56                          if (crc_cnt == 0)begin
57                              fsm =  IDLE;
58                              crc_cnt = 8;
59                              a_port.write(item);
60                          end
61                      end
62              endcase
63          end
64          @vif.slv_cb;
65      end
66  endtask
```

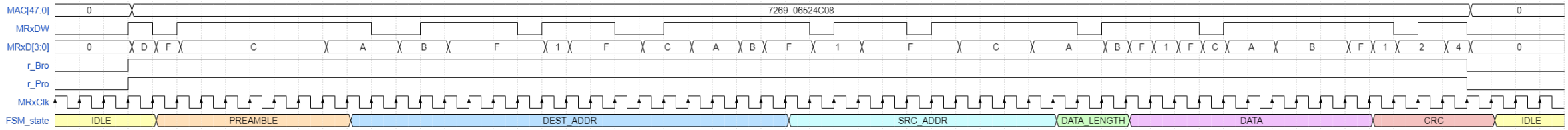
# Task for protocol violations checks

- Started in the run phase
- Tracks each signal behavior separately
- Do not check the value of the signal
- Check if detected activity on the signal is in valid frame of the packet

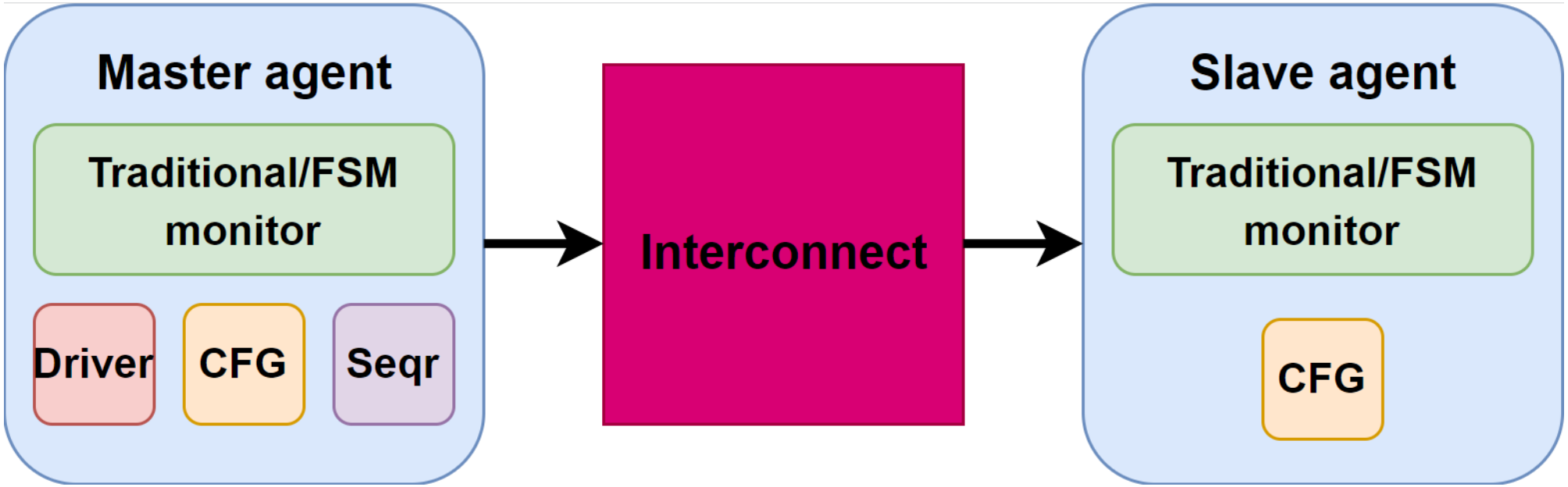
```
01  virtual task check_signals();
02      fork
03          begin
04              forever begin
05                  @(posedge vif.r_Pro, negedge vif.r_Pro);
06                  r_Pro_signal_chk: assert(fsm == IDLE)else
07                      `uvm_error("check_signals", $sformatf("Illegal r_Pro
change detected during packet state : %s",fsm.name));
08              end
09          end
10          begin
11              forever begin
12                  @(posedge vif.r_Bro, negedge vif.r_Bro);
13                  r_Bro_signal_chk: assert(fsm == IDLE)else
14                      `uvm_error("check_signals", $sformatf("Illegal r_Bro
change detected during packet state : %s",fsm.name));
15              end
16          end
17          begin
18              forever begin
19                  @(vif.MAC);
20                  MAC_signal_chk: assert(fsm == IDLE)else
21                      `uvm_error("check_signals", $sformatf("Illegal MAC change
detected during packet state : %s",fsm.name()));
22              end
23          end
24      join_none
25  endtask
```

# Benefits

- Gains in debug
- Usage of modeled packet state on a waveform
- Compare modelled state behavior to interface behavior

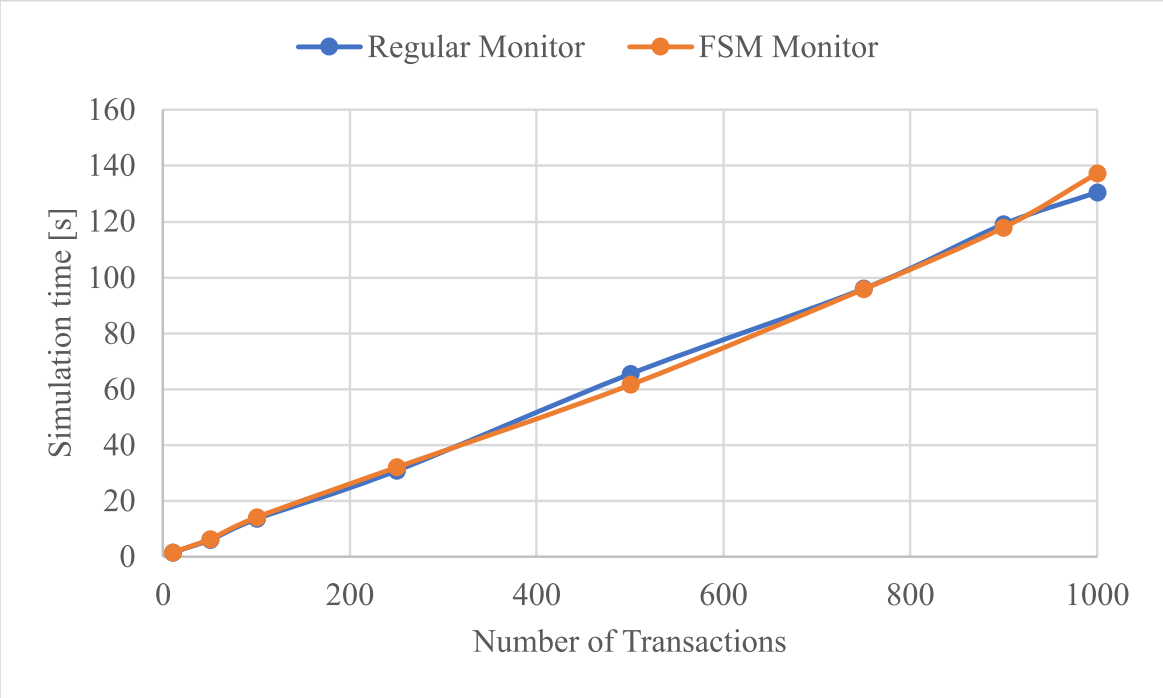


# Test environment

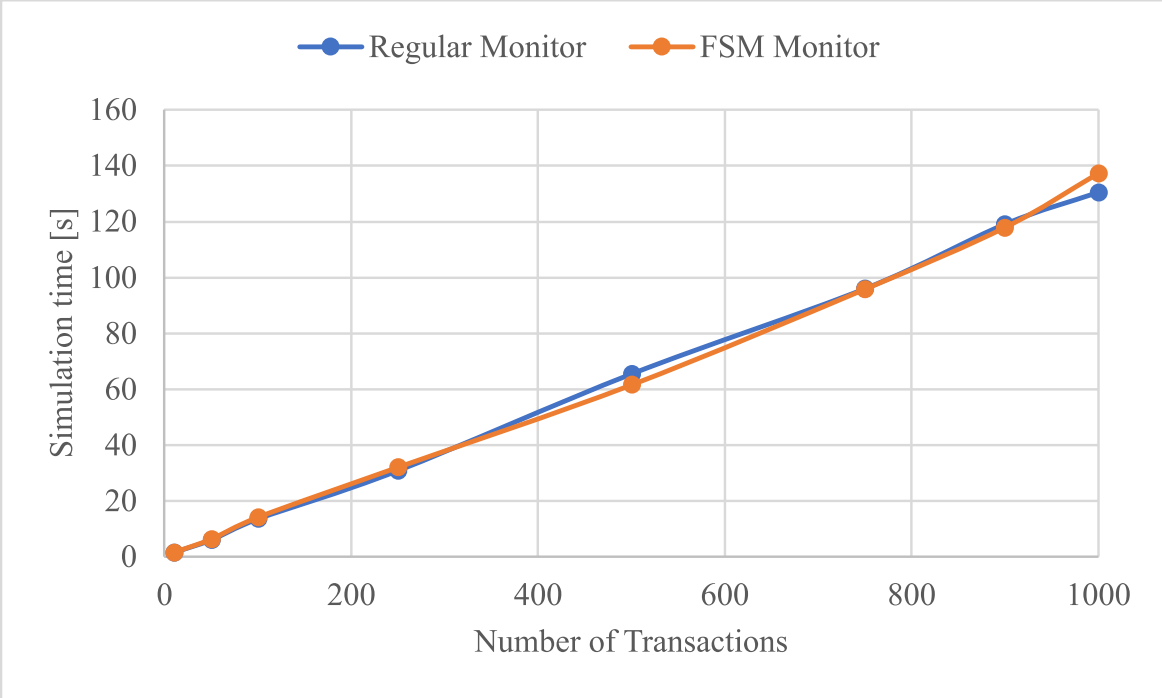


# Traditional vs. FSM monitor results

## CPU usage




## Simulation time



# Conclusion – Why FSM monitor is good?

- Standardized monitor coding
- Easier implementation based on a well-known concept
- Lower threshold for Interface behavior expertise
- Significant boost in debug
- On par or better performance when compared to the traditional approach

# Questions?

2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

# Thank you for your attention.

accellera
SYSTEMS INITIATIVE