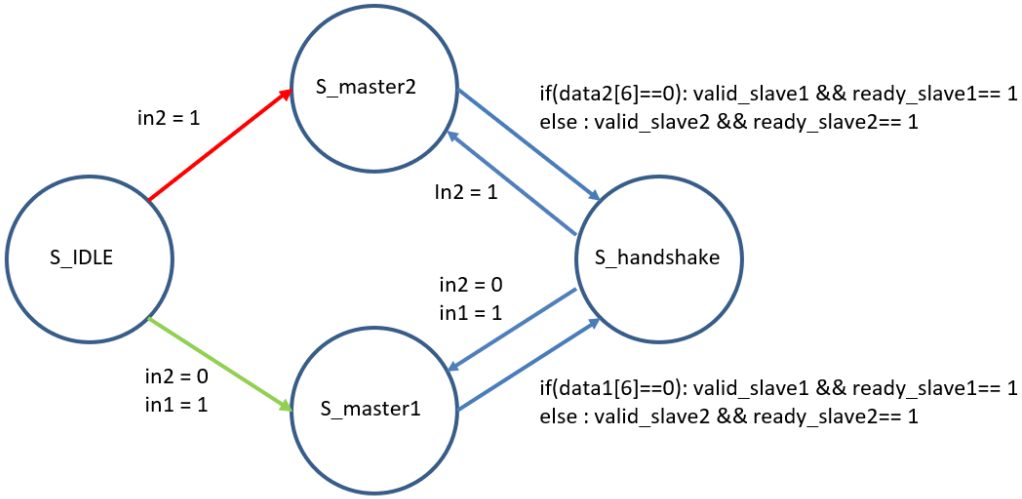
Lab05 Code Review

Finite state machine

Reference code

- Idle state



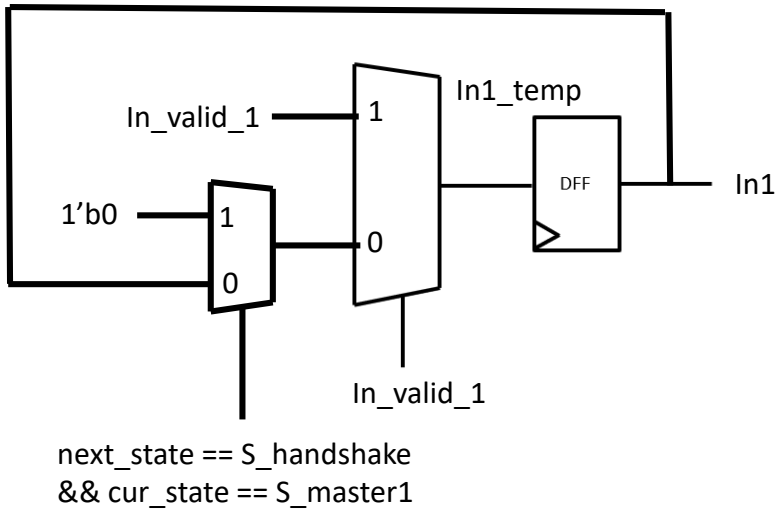
S_master1 and S_master2:
- Set valid_slave
- Set value_out and addr_out

```
always_comb begin
  case(cur_state)
    S_idle :
    if(in2==1)  next_state = S_master2;
    else if(in1 == 1) next_state = S_master1;
    else next_state = cur_state;
```

Reference code

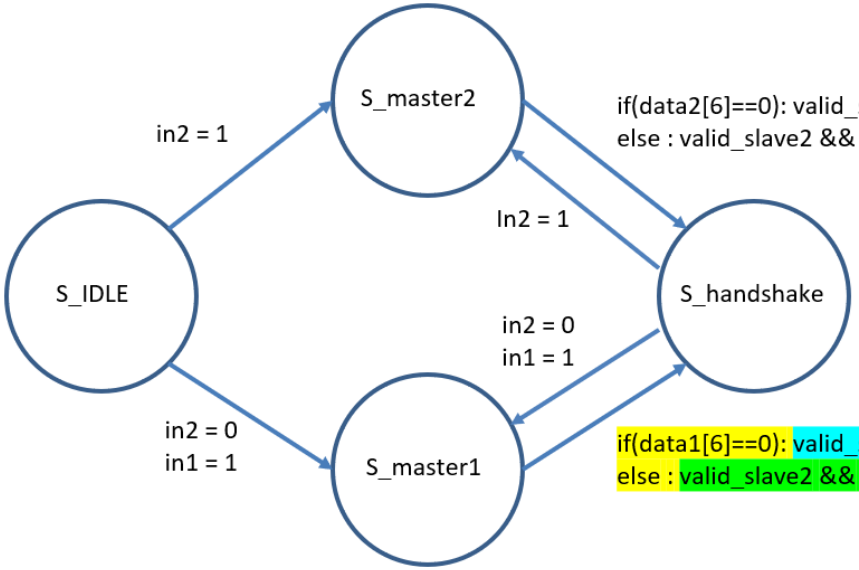
Control signal: in1, in2

```
always_comb begin
  if(in_valid_1) in1_temp = in_valid_1;
  else if(next_state == S_handshake && cur_state == S_master1)
    in1_temp = 1'b0;
  else in1_temp = in1;
end
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    in1 <= 0;
  end else begin
    in1 <= in1_temp;
  end
end
always_comb begin
  if(in_valid_2) in2_temp = in_valid_2;
  else if(next_state == S_handshake && cur_state == S_master2)
    in2_temp = 1'b0;
  else in2_temp = in2;
end
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    in2 <= 0;
  end else begin
    in2 <= in2_temp;
  end
end
```



Reference code

- Master1 state

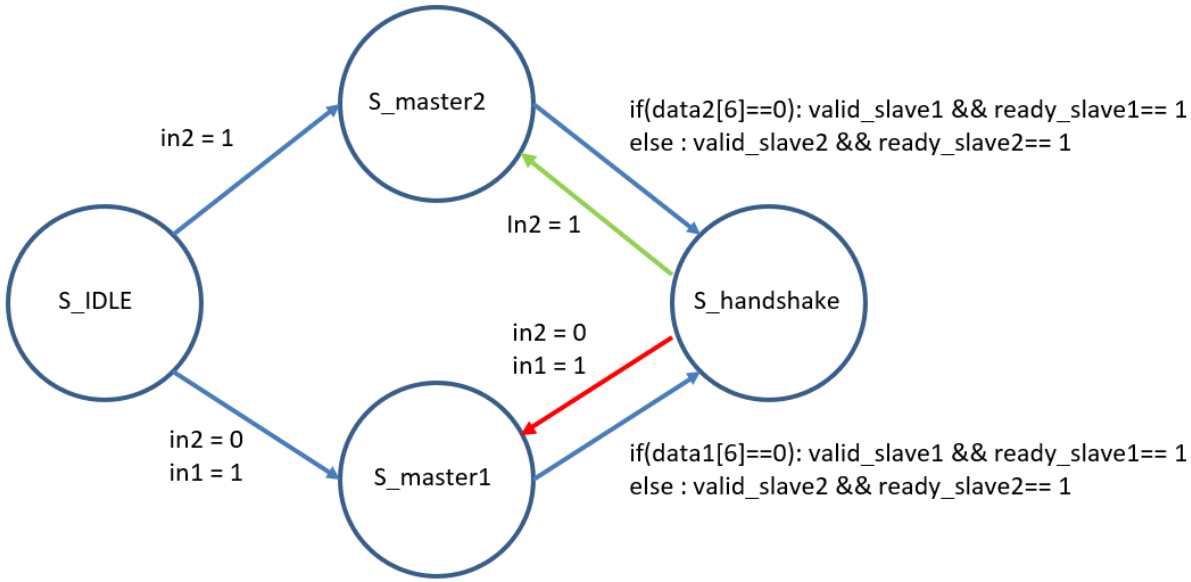


S_master1 and S_master2:
- Set valid_slave
- Set value_out and addr_out

```
S_master1 :
  if(!data1[6]) begin
    if(valid_slave1 && ready_slave1)
      next_state = S_handshake;
    else
      next_state = cur_state;
  end
  else begin
    if(valid_slave2 && ready_slave2)
      next_state = S_handshake;
    else
      next_state = cur_state;
  end
```

Reference code

- Handshake state



S_master1 and S_master2:
- Set valid_slave
- Set value_out and addr_out

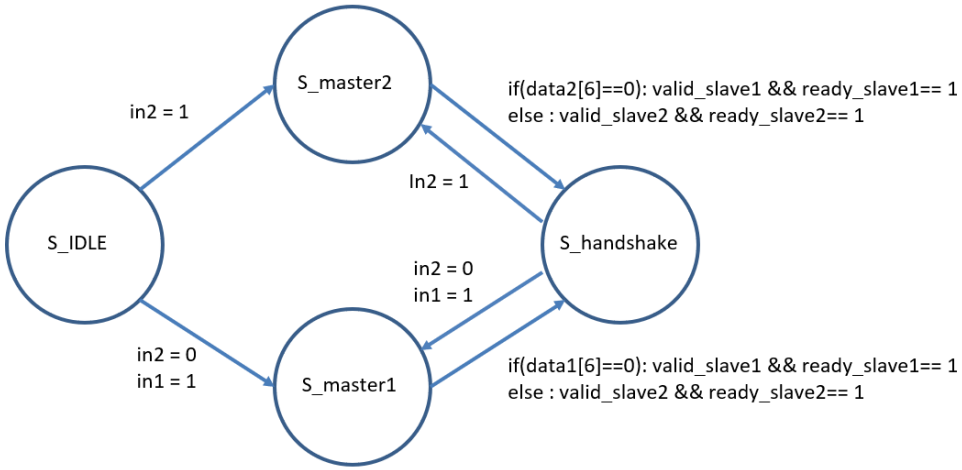
```
S_handshake :
if(in2) next_state = S_master2;
else if(in1) next_state = S_master1;
else next_state = cur_state;
```

Reference code: overall FSM

```
always_comb begin
  case(cur_state)
    S_idle :
      if(in2==1)  next_state = S_master2;
      else if(in1 == 1) next_state = S_master1;
      else next_state = cur_state;
    S_master1 :
      if(!data1[6]) begin
        if(valid_slave1 && ready_slave1)
          next_state = S_handshake;
        else
          next_state = cur_state;
      end
      else begin
        if(valid_slave2 && ready_slave2)
          next_state = S_handshake;
        else
          next_state = cur_state;
      end
    S_master2 :
      if(!data2[6]) begin
        if(valid_slave1 && ready_slave1)
          next_state = S_handshake;
        else
          next_state = cur_state;
      end
      else begin
        if(valid_slave2 && ready_slave2)
          next_state = S_handshake;
        else
          next_state = cur_state;
      end
    S_handshake :
      if(in2) next_state = S_master2;
      else if(in1) next_state = S_master1;
      else next_state = cur_state;
    default:
      next_state = cur_state;
  endcase
end
```

Reference code

- What we have to do in S_master1&2



S_master1 and S_master2:
- Set valid_slave
- Set value_out and addr_out

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    valid_slave1 <= 0;
  end else begin
    valid_slave1 <= valid_slave1_temp;
  end
end
always_comb begin
  if((cur_state==S_master1 ) && !data1[6]) begin
    valid_slave1_temp = 'b1;
  end
  else if((cur_state==S_master2 ) && !data2[6]) begin
    valid_slave1_temp = 'b1;
  end
  else begin
    valid_slave1_temp = 'b0;
  end
end
```

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    addr_out <= 0;
  end else begin
    addr_out <= addr_out_temp;
  end
end
always_comb begin
  if(cur_state==S_master1) begin
    addr_out_temp = data1[5:3];
  end
  else if(cur_state==S_master2) begin
    addr_out_temp = data2[5:3];
  end
  else begin
    addr_out_temp = 'b0;
  end
end
```

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    value_out <= 0;
  end else begin
    value_out <= value_out_temp;
  end
end
always_comb begin
  if(cur_state==S_master1) begin
    value_out_temp = data1[2:0];
  end
  else if(cur_state==S_master2) begin
    value_out_temp = data2[2:0];
  end
  else begin
    value_out_temp = 'b0;
  end
end
```

Reference code

- What we have to do in S_handshake

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    handshake_slave1 <= 0;
  end else begin
    handshake_slave1 <= handshake_slave1_temp;
  end
end
always_comb begin
  if(next_state == S_handshake && cur_state == S_master1  && !data1[6] ) begin
    handshake_slave1_temp = 'b1;
  end
  else if (next_state == S_handshake && cur_state == S_master2  && !data2[6] )begin
    handshake_slave1_temp = 'b1;
  end
  else begin
    handshake_slave1_temp = 'b0;
  end
end
```

Handshake signal only pull high at the first cycle of S_handshake

Recommended coding style

```
2'b01: begin
        in2_comb = in2_seq;
        data_2_comb = data_2_seq;
        if(!data_1_seq[6]) begin
            state_comb = ((valid_slave1) && (ready_slave1)) ? 3 : 1;
            handshake1_comb = ((valid_slave1) && (ready_slave1)) ? 1 : 0;
            handshake2_comb = handshake_slave2;

            valid_slave1_comb = 1;
            valid_slave2_comb = valid_slave2;

            addr_comb         = data_1_seq[5:3];
            value_comb        = data_1_seq[2:0];

            in1_comb          = ((valid_slave1) && (ready_slave1)) ? 0 : 1;
            data_1_comb       = ((valid_slave1) && (ready_slave1)) ? 0 : data_1_seq;
        end
        else if(data_1_seq[6]) begin
            state_comb = ((valid_slave2) && (ready_slave2)) ? 3 : 1;
            handshake2_comb = ((valid_slave2) && (ready_slave2)) ? 1 : 0;
            handshake1_comb = handshake_slave1;

            valid_slave2_comb = 1;
            valid_slave1_comb = valid_slave1;

            addr_comb         = data_1_seq[5:3];
            value_comb        = data_1_seq[2:0];

            in1_comb          = ((valid_slave2) && (ready_slave2)) ? 0 : 1;
            data_1_comb       = ((valid_slave2) && (ready_slave2)) ? 0 : data_1_seq;
        end
```

Put all signals along with FSM in one always block

⇒ Not recommended!

1. Hard to debug when facing multiple driven issue
2. Need to write same statements several times to avoid latch

Recommended coding style

```
always_comb begin
  if(in_valid_1) in1_temp = in_valid_1;
  else if(next_state == S_handshake && cur_state == S_master1)
    in1_temp = 1'b0;
  else in1_temp = in1;
end
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    in1 <= 0;
  end else begin
    in1 <= in1_temp;
  end
end
always_comb begin
  if(in_valid_1) data1_slave_temp = data_in_1[6];
  else data1_slave_temp = data1_slave;
end

always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    data1_slave <= 0;
  end else begin
    data1_slave <= data1_slave_temp;
  end
end
```

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    valid_slave1 <= 0;
  end else begin
    valid_slave1 <= valid_slave1_temp;
  end
end
always_comb begin
  if((cur_state==S_master1 ) && !data1[6]) begin
    valid_slave1_temp = 'b1;
  end
  else if((cur_state==S_master2 ) && !data2[6]) begin
    valid_slave1_temp = 'b1;
  end
  else begin
    valid_slave1_temp = 'b0;
  end
end
```

Put only one signal in one always block

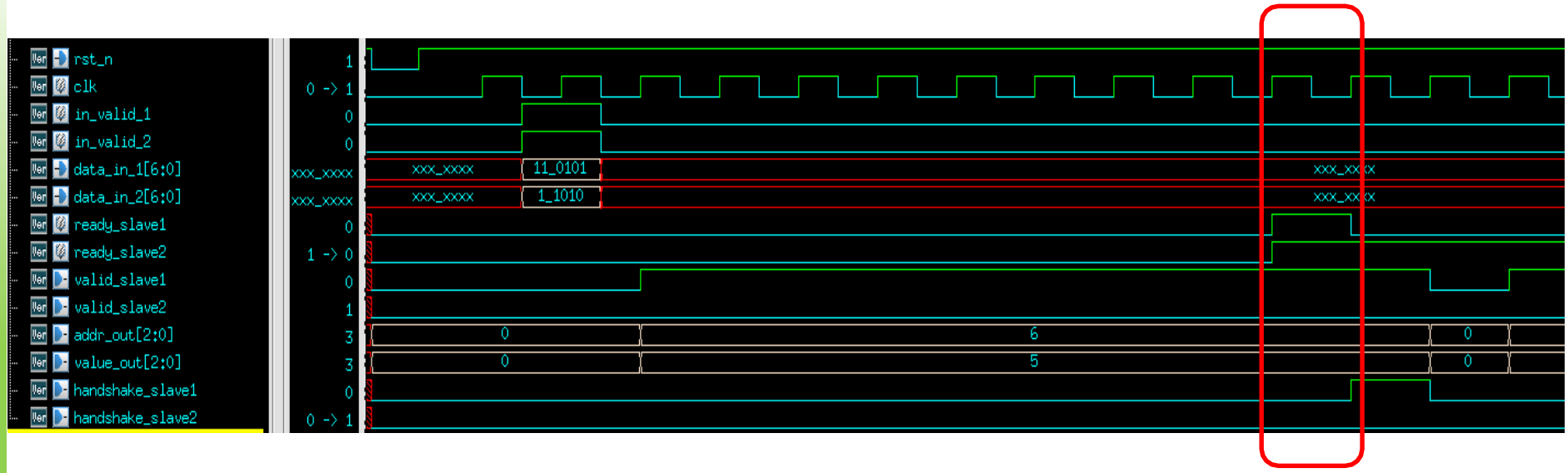
Or

Separate the signals according to in which state they are going to change

⇒ Recommended!

Think one always block as a circuit with one output.

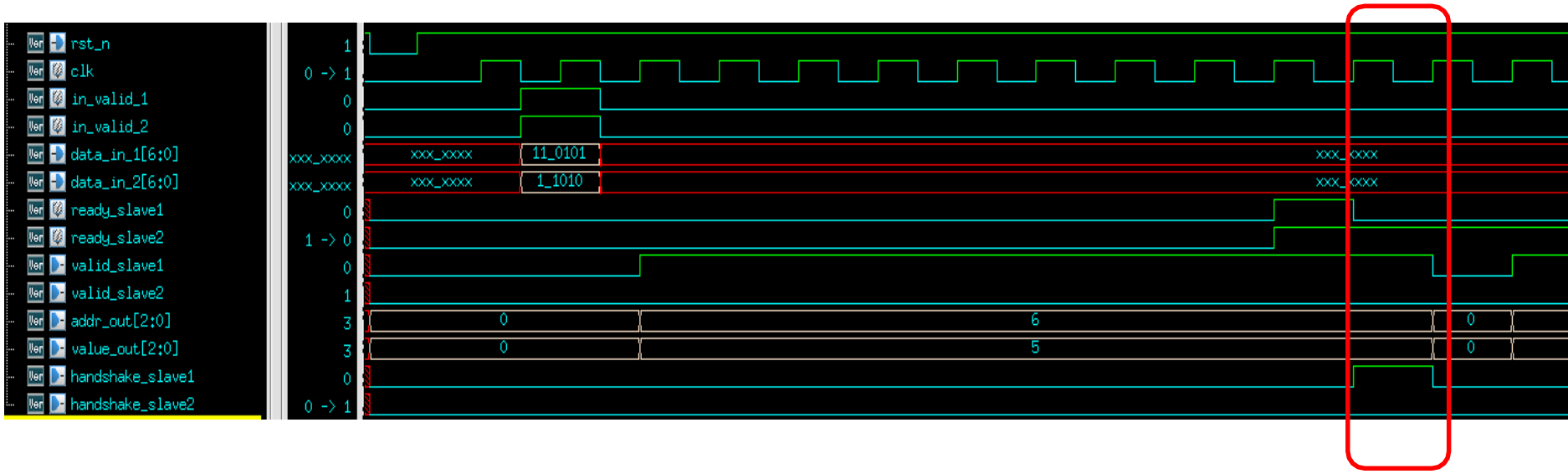
Frequently Asked Problem



Write

Handshake!

Frequently Asked Problem



Done!

output handshake signal