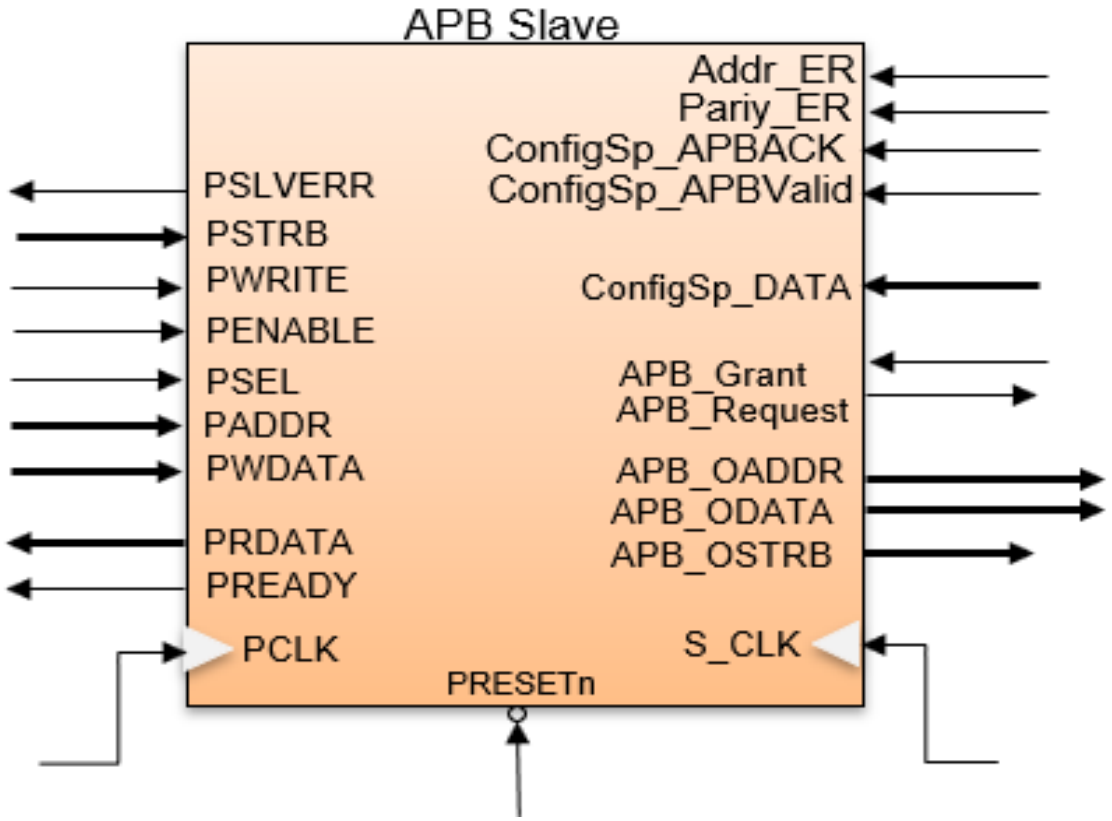
APB SLAVE

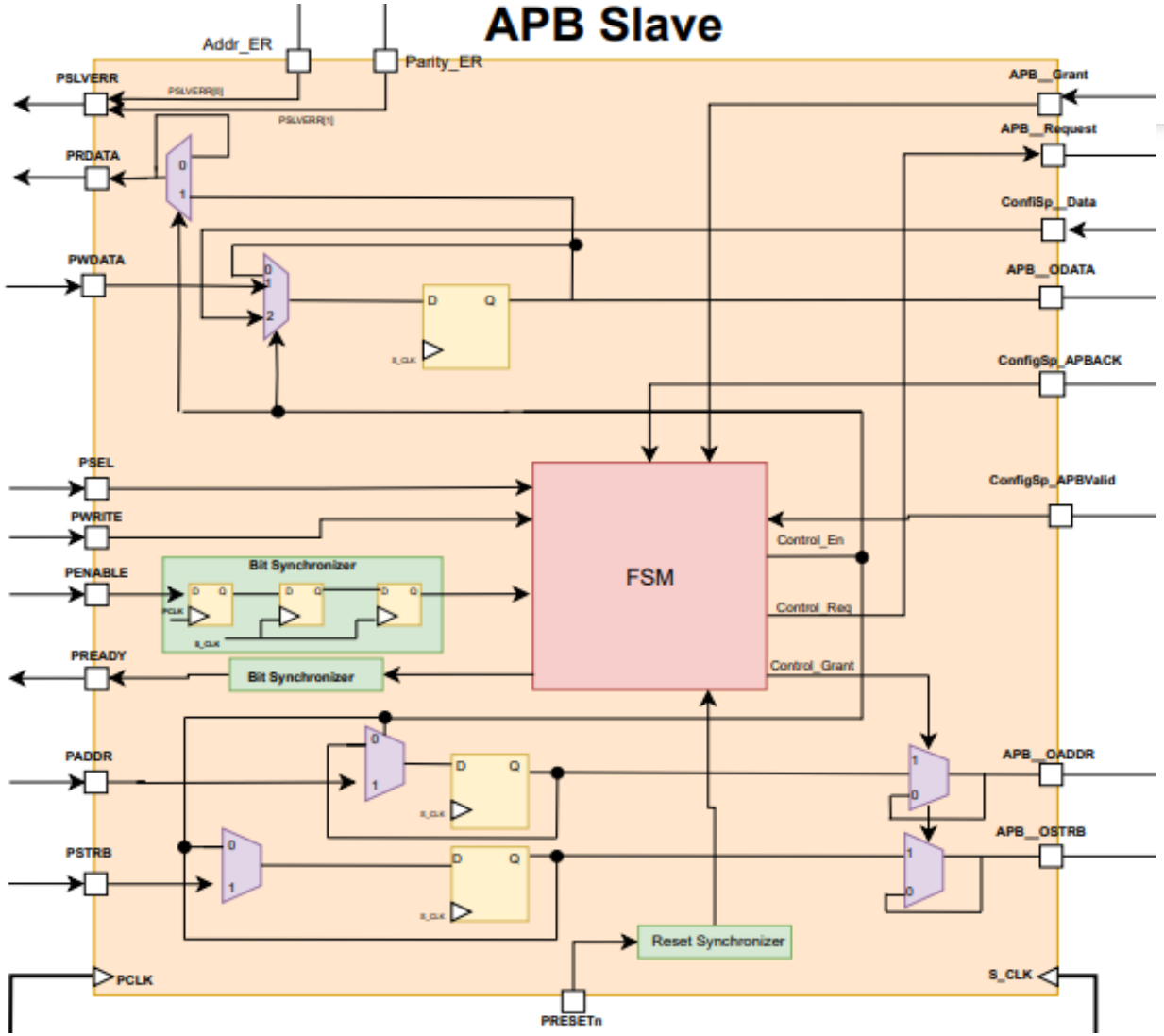
Contents

- Block Diagram of Module
- Main Building Blocks of Module
- RTL
- Output waves

Block Diagram

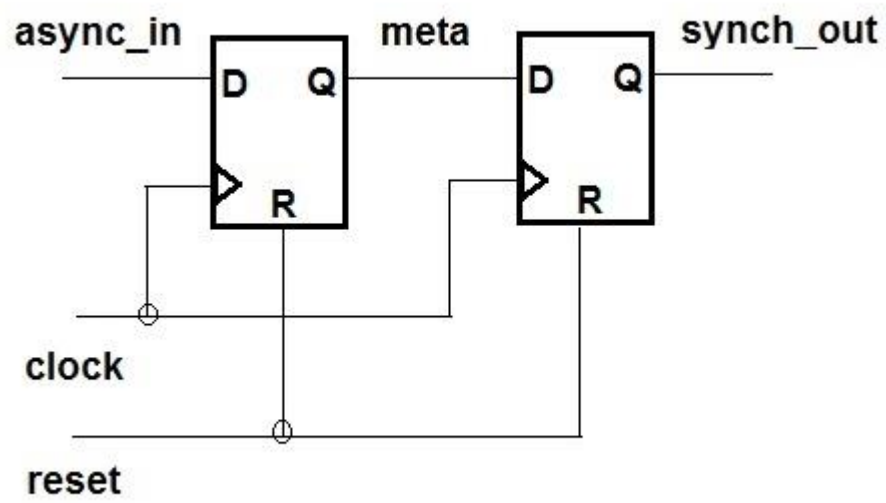


Main Building Blocks



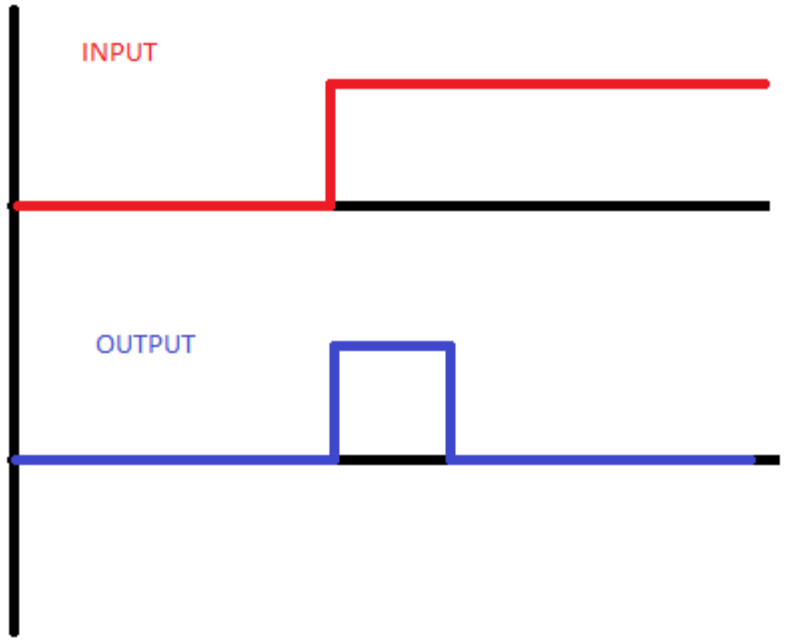
RTL : Bit Synchronizer with Pulse Converter

```
module BIT_SYNC (
input       wire        Destination_CLK,
input       wire        RST,
input       wire        ASYNC_IN,
output      reg         SYNC_OUT
    );
//Destination FlipFlop
reg         FF1;
//Synchronization FlipFlop
reg         FF2;

//flag
reg     Pulse_Conv_flg;

//Sequential always
always @(posedge Destination_CLK, negedge RST) begin
    if(!RST) begin
        FF1         <= 'b0;
        FF2         <= 'b0;
    end
    else begin
        FF1         <= ASYNC_IN;
        FF2         <= FF1;
    end

end
```



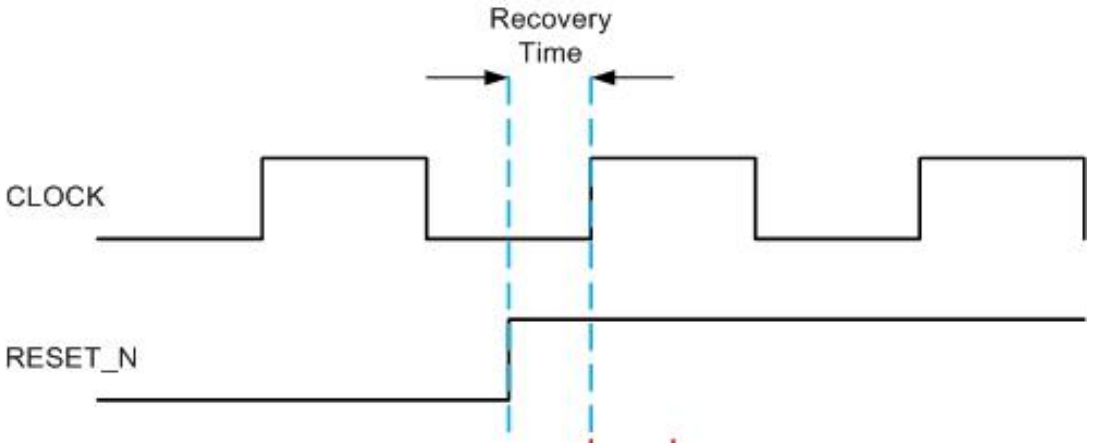
RTL : Bit Synchronizer with Pulse Converter

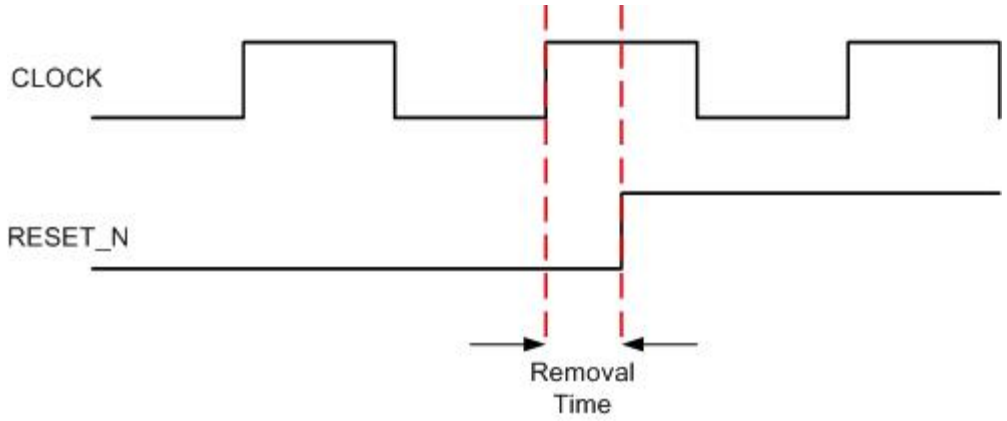
```
//Pulse Converter Logic
always@(posedge Destination_CLK, negedge RST) begin
    if(!RST) begin
        Pulse_Conv_flg      =    'b1;
    end

    else begin

        if(FF2  ==  'b1) begin

            if(Pulse_Conv_flg   ==  'b1) begin
                SYNC_OUT            =   'b1;
                Pulse_Conv_flg      =   'b0;
            end

            else begin
                SYNC_OUT            =   'b0;
                Pulse_Conv_flg      =   'b0;
            end

        end

        else begin
            Pulse_Conv_flg          =   'b1;
            SYNC_OUT                =   'b0;
        end
    end
end


endmodule
```



RTL : Reset Synchronizer

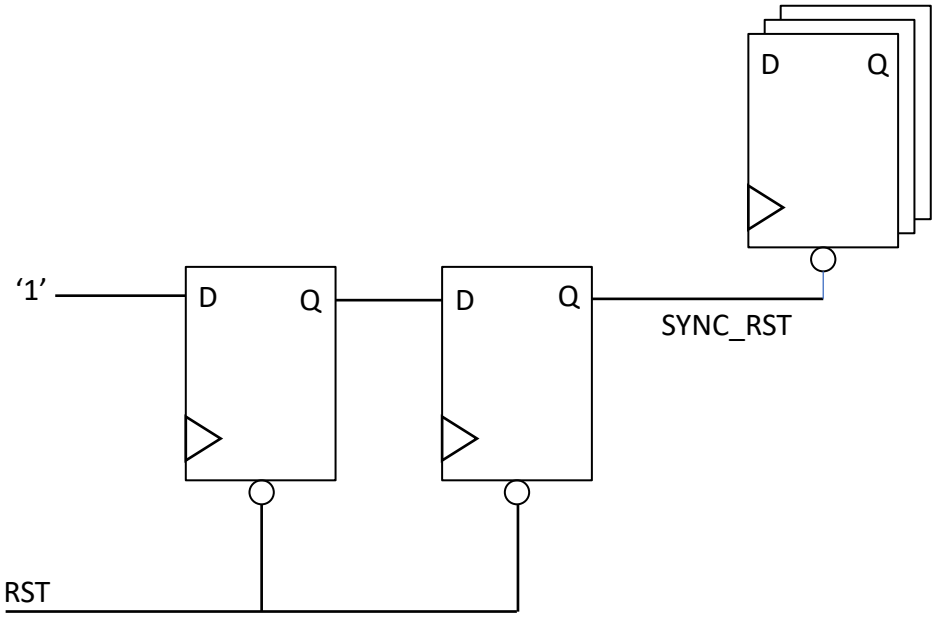
- We have a problem with the Asynchronous reset de-assertion as it **may violate recovery time** which may get all the flip flops onto a **metastable state.**
- $T_{Recovery}$: it is the minimum time for sequential cells that the active low set or reset signal **must be de-asserted before the active edge** of the clock, to allow the circuit to be recovered from the effect of the asynchronous control signal and the next clock edge can trigger the circuit.

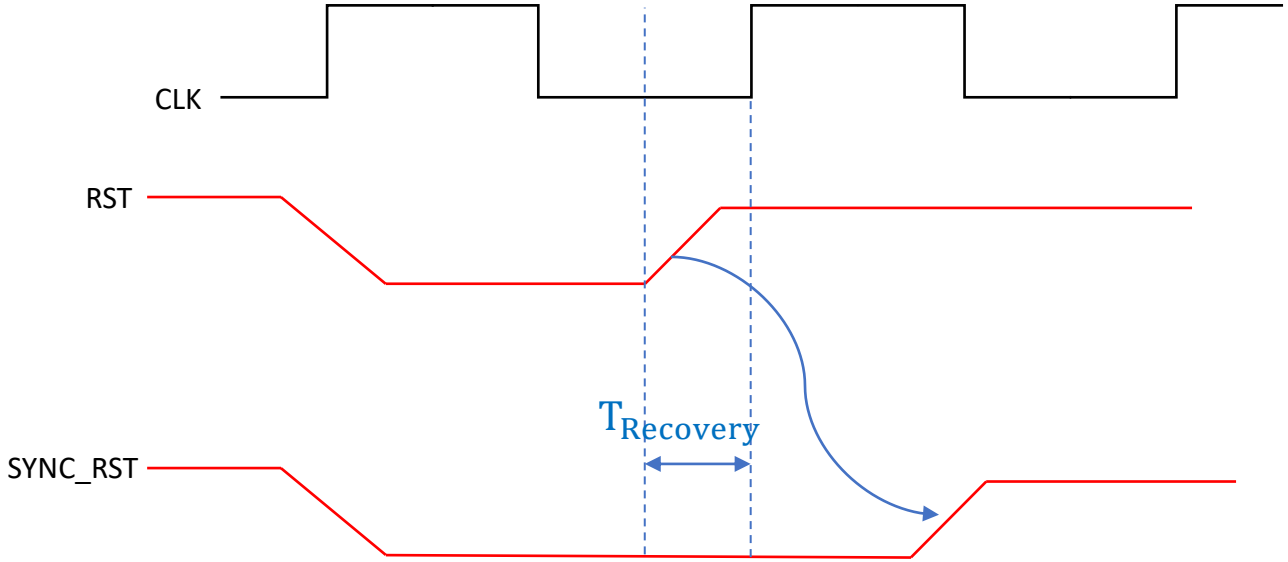




RTL : Reset Synchronizer

- **A reset synchronizer synchronizes the de-assertion of reset with respect to the clock domain.**



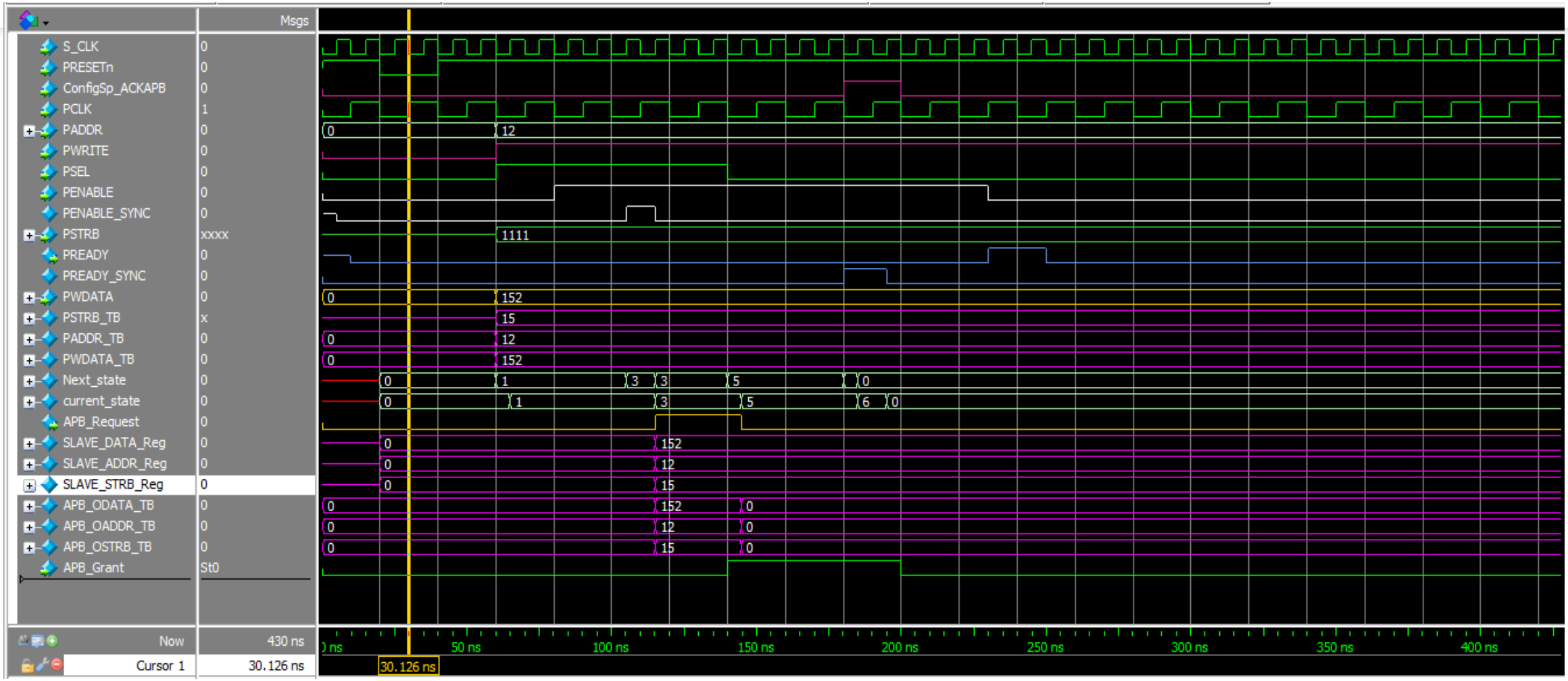


RTL : Reset Synchronizer

```
module RESET_SYNC (
input       wire        Destination_CLK,
input       wire        RST,
output      reg         SYNC_RST
    );

//Destination FlipFlop
reg         FF1;


//Sequential always
always @(posedge Destination_CLK, negedge RST) begin
    if(!RST) begin
        FF1                 <= 'b0;
        SYNC_RST            <= 'b0;
    end
    else begin
        FF1                 <= 'b1;
        SYNC_RST            <= FF1;
    end

end


endmodule
```

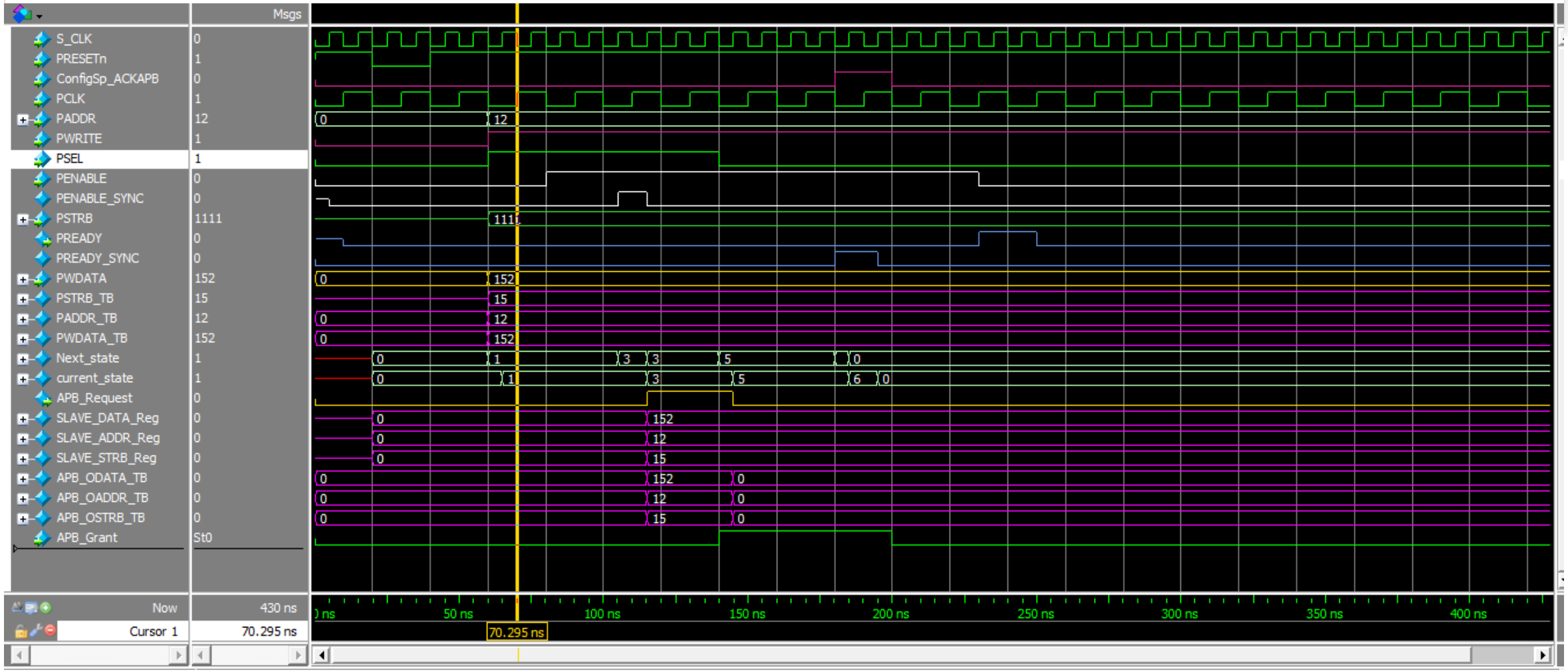
❑ Write Transaction : 1. Reset

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



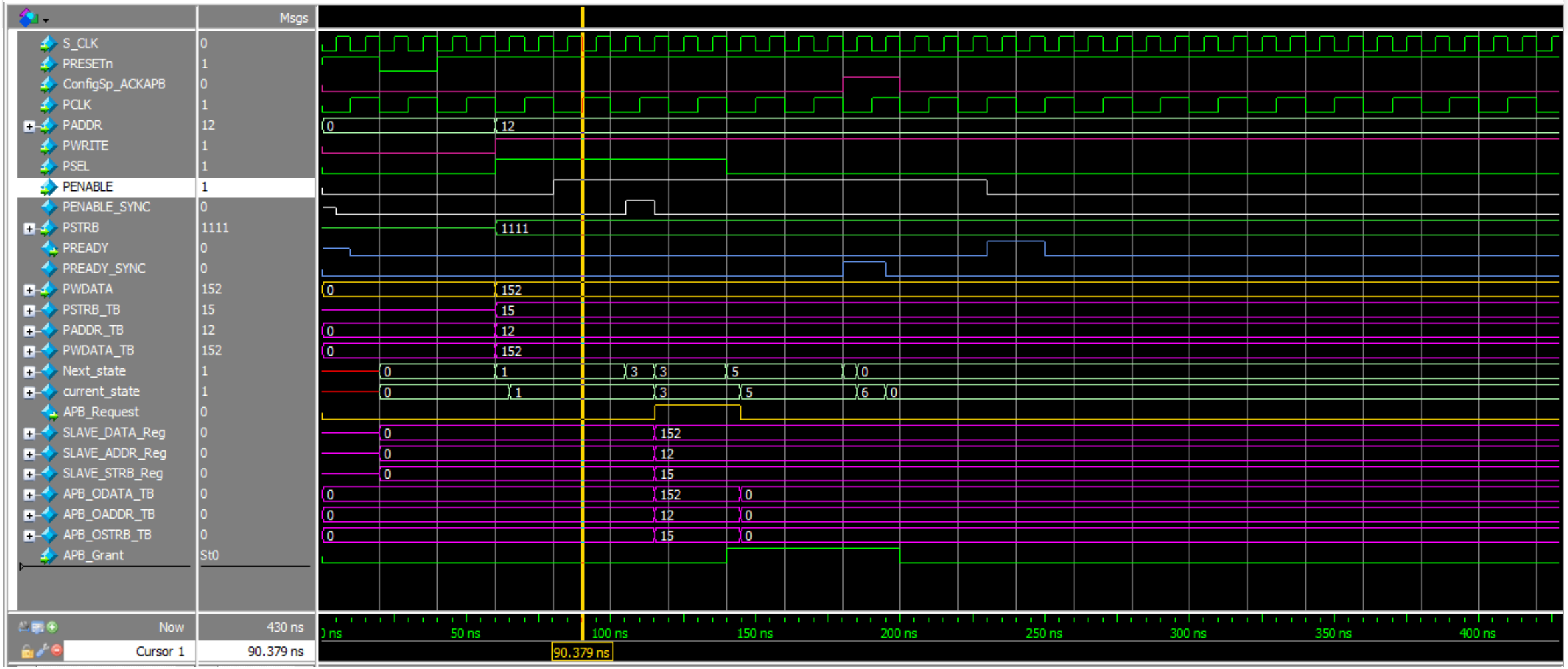
❑ Write Transaction : 2. Setup Phase

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



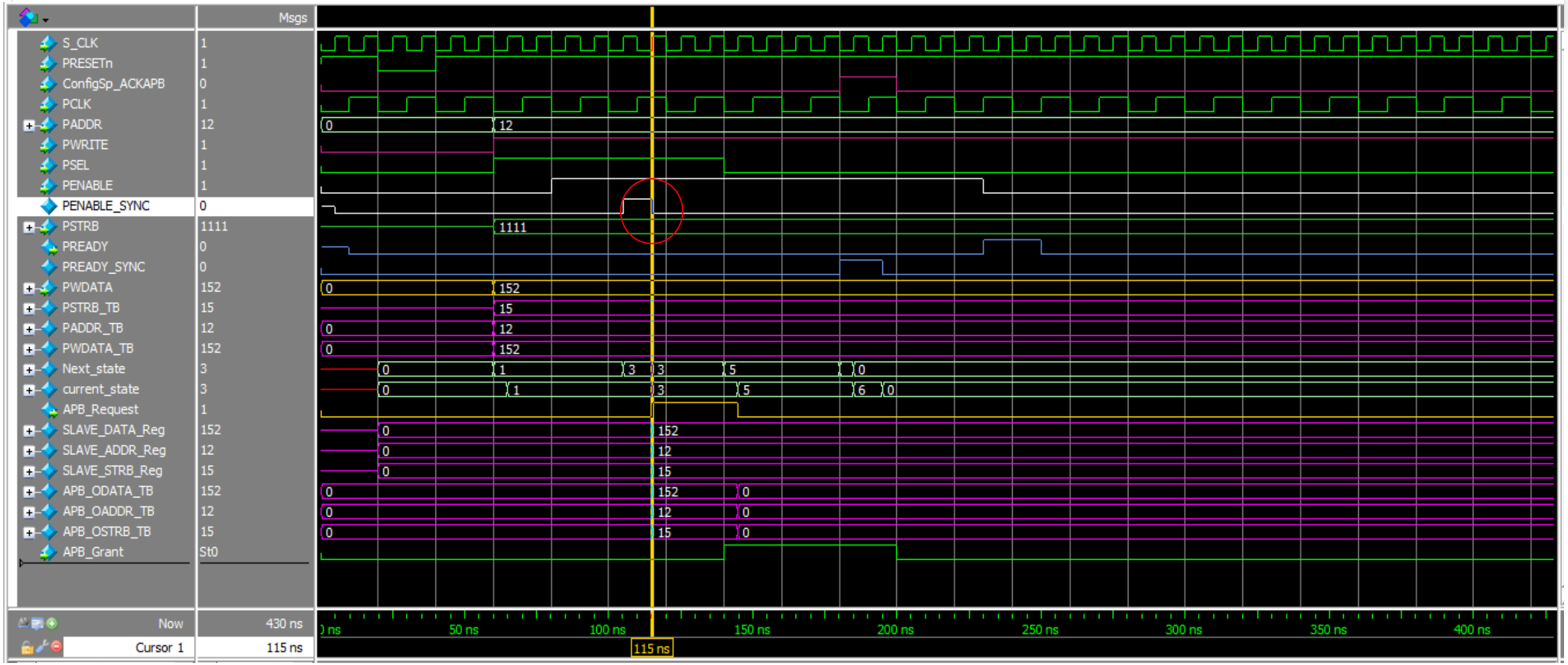
❑ Write Transaction : 3. Access Phase

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



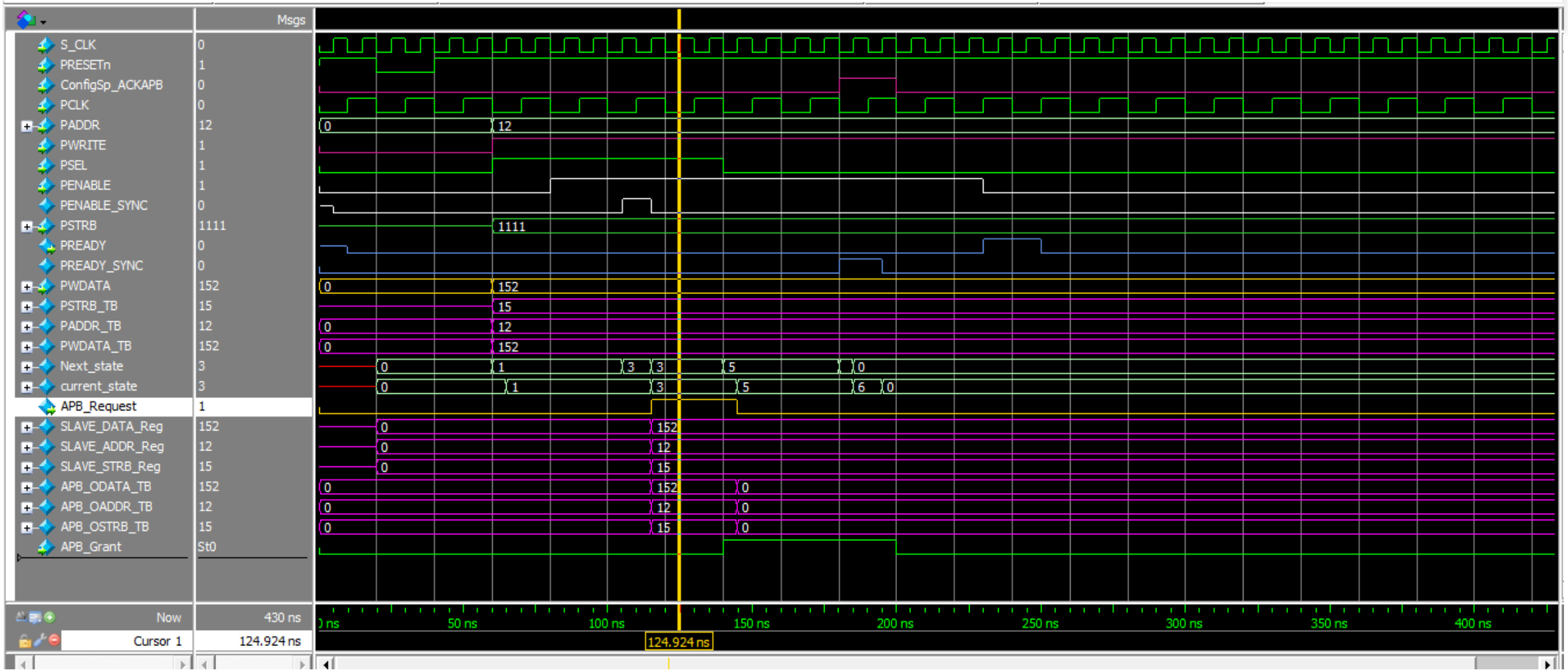
❑ Write Transaction : 4. Storage Data on S_CLK

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



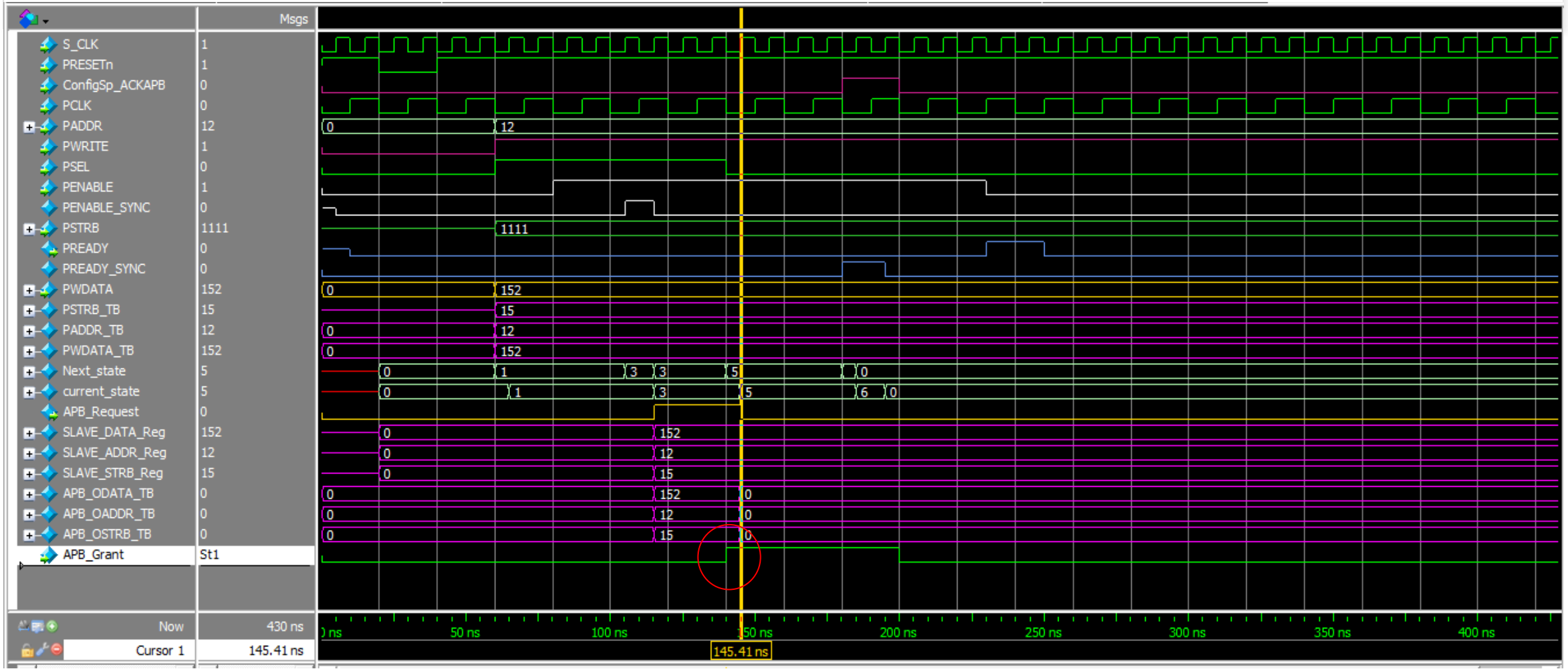
❑ Write Transaction : 5. Waiting for Grant Access

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



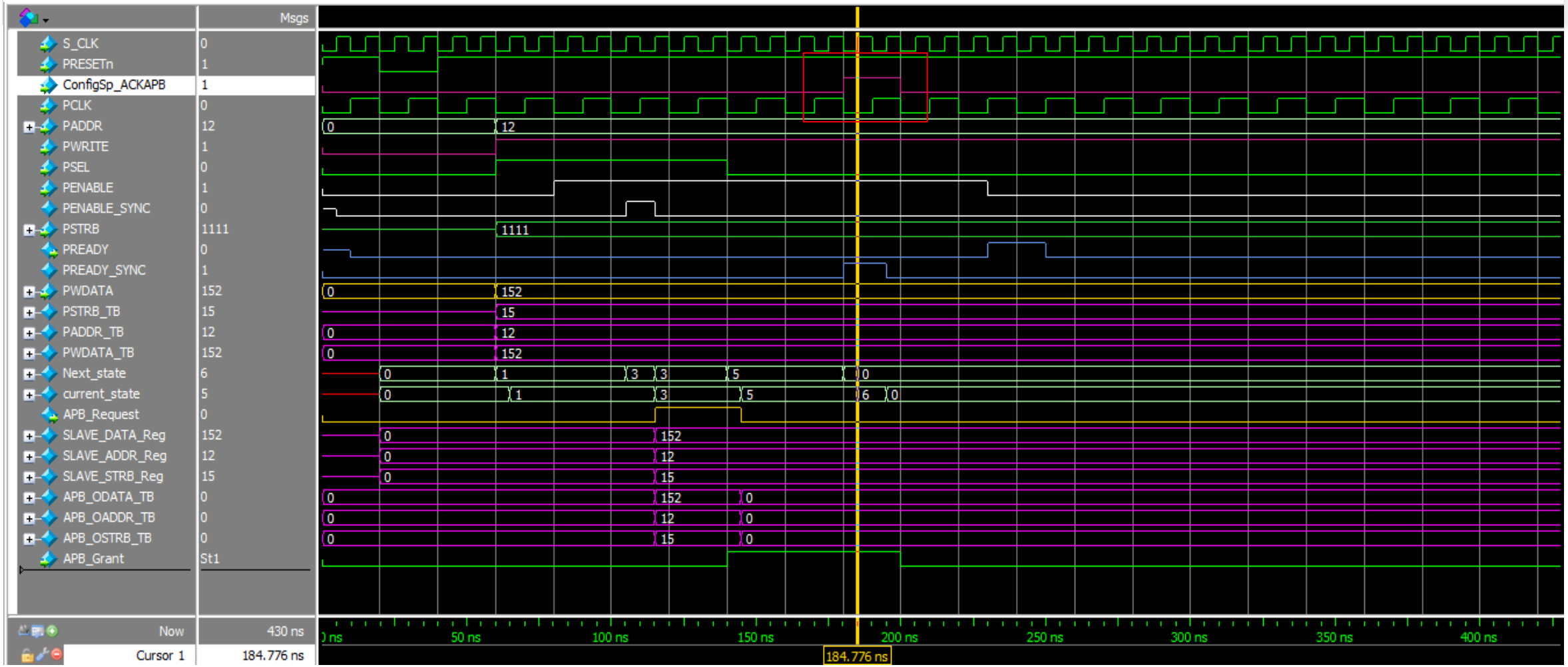
❑ Write Transaction : 6. Grant Access Approved

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



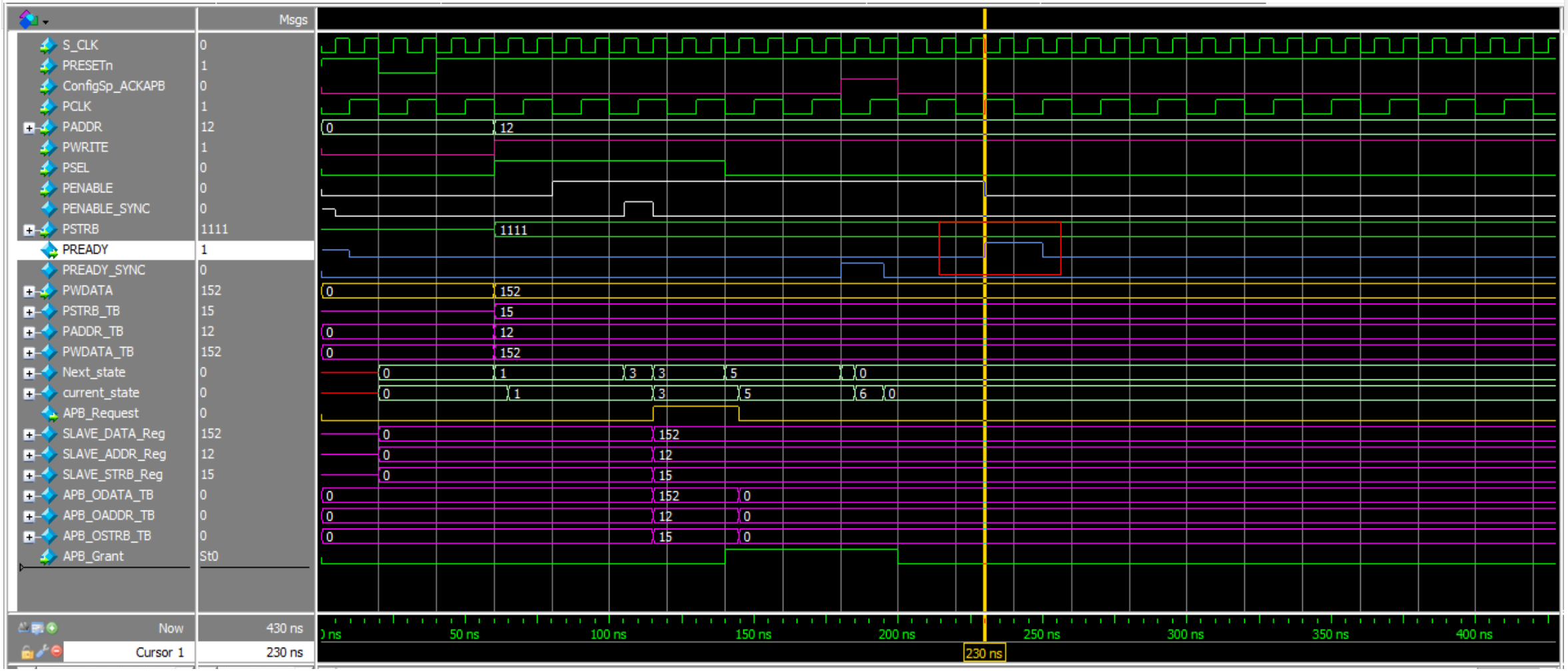
❑ Write Transaction : 7. Config Space ACK came

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



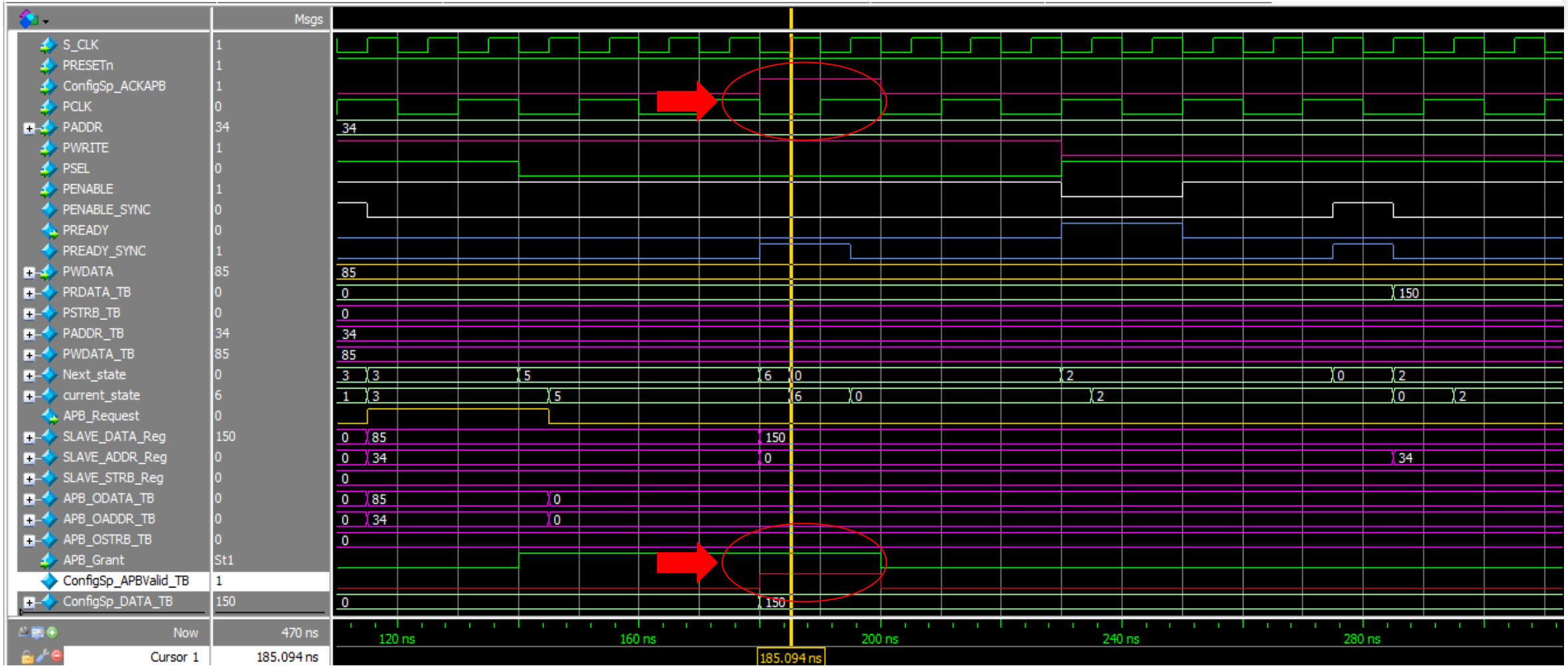
❑ Write Transaction : 8. Transaction Completed

(Data = ‘d152, Address = d’12, Strobes = ‘b1111)



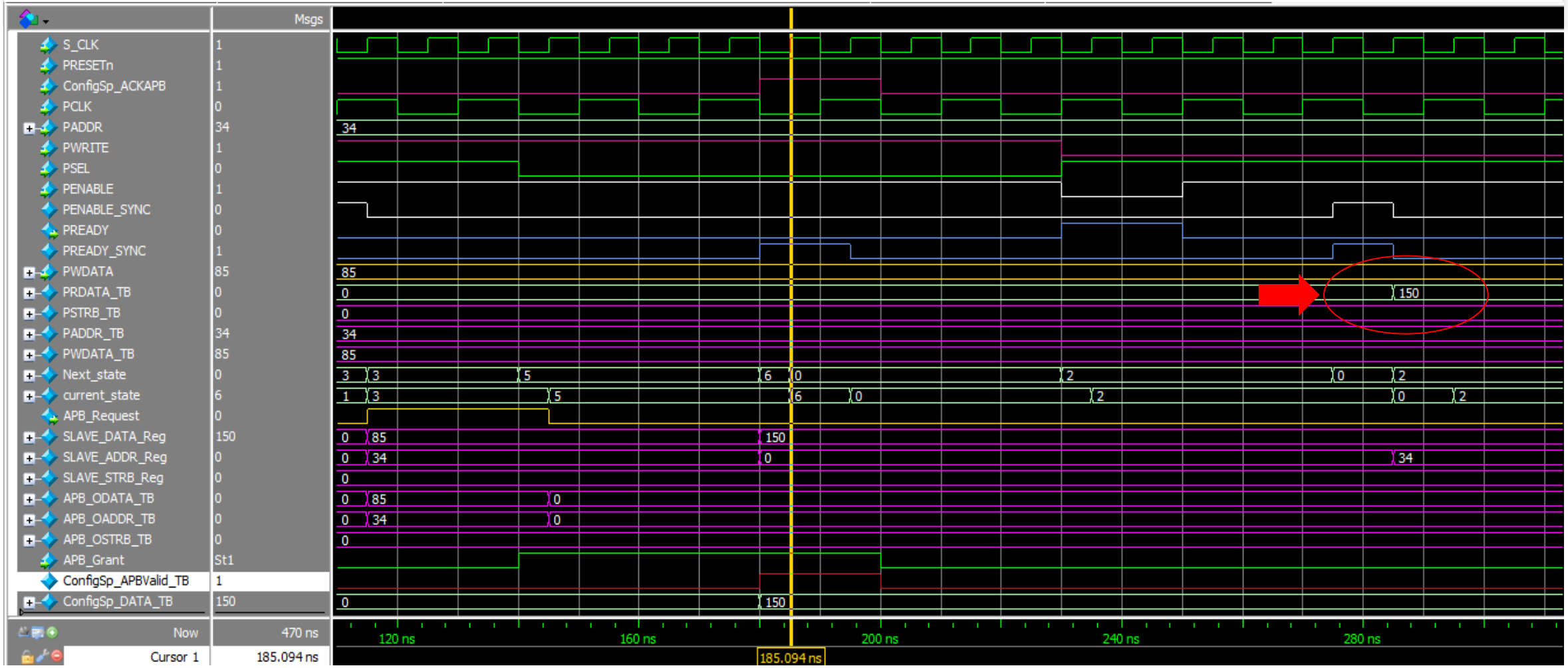
❑ Read Transaction : 7. CS ACK and Valid Came

(Data = ‘d85, Config Space data = ‘d150 Address = d’34, Strobes = ‘b0000)



❑ Read Transaction : 8. Reading CS Data (‘d150)

(Data = ‘d85, Config Space data = ‘d150 Address = d’34, Strobes = ‘b0000)




Any Questions ?