Richard Ojo

Lab 3 report

APB slave

```
/*APB Slave module*/
module apb(input PCLK, input PRESET, input [3:0] PWAIT, input PSEL, input PENABLE, input [7:0] PADDR, input
PWRITE,
                    input [31:0] PWDATA, input [31:0] MEMRDATA, output logic PREADY, output logic [31:0] PRDATA,
output logic [7:0] addr, output logic [31:0]  data, output logic ce, output logic wren, output logic rden);

//Counter vars
logic [7:0] count;
logic [7:0] next_count;

// Store the current and next state
enum logic [1:0]{
        IDLE = 2'b00,
        ACCESS = 2'b01,
        WAIT = 2'b10,
        WAIT2 = 2'b11
} next_state, curr_state;

task clearAPB;//Reset APB
        begin
                assign PREADY = 0;
                assign PRDATA = 0;
                assign addr = 0;
                assign ce = 0;
                assign wren = 0;
                assign rden = 0;
                assign data = 0;//Write Data to Memory
                next_count = PWAIT;//If we have a wait value, wait those additional clock cycles until access
        end
endtask: clearAPB

task readyToRW(input read_on, input write_on);//Push values for reading and writting to mem
        begin
                assign wren = write_on;
                assign rden = read_on;
                assign addr = PADDR;
                assign ce = 1;
        end

endtask:readyToRW



//Current State logic
always_ff@(posedge PCLK or negedge PRESET) begin
        if(PRESET)  begin
                curr_state <= IDLE; //If reset -> go to the first state
                count <= 0;
                end
        else begin
                curr_state <= next_state; //go to the next state
                count <= next_count;
                end
```

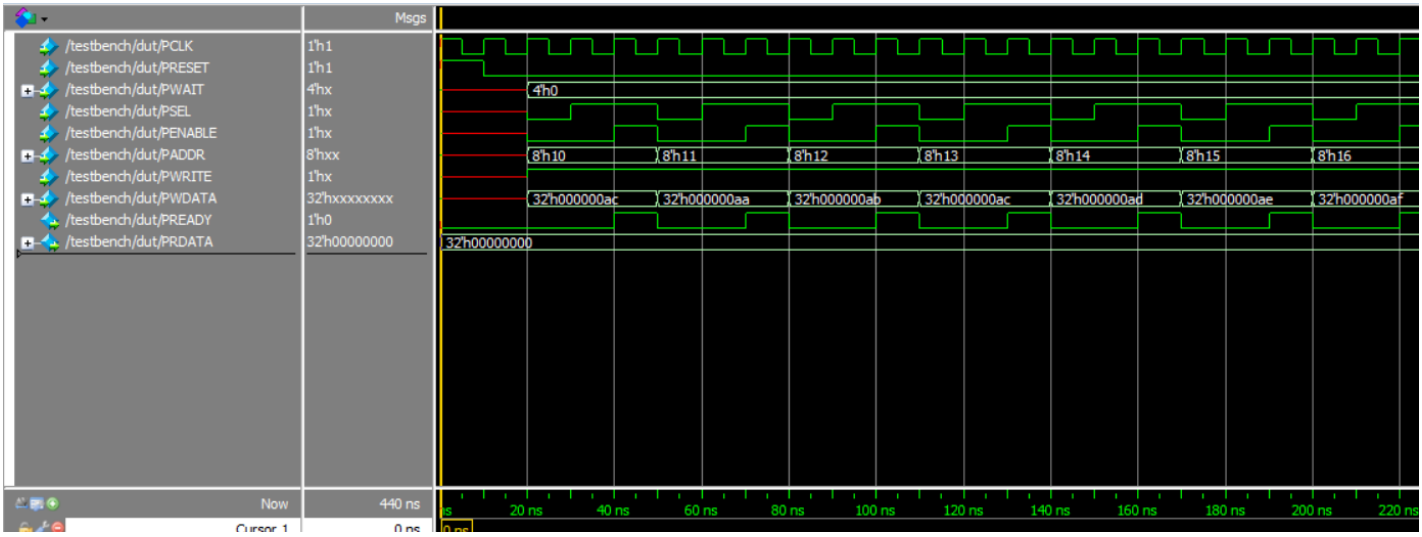
```
end
//Next State Logic
always_comb begin
	case(curr_state)
		IDLE: begin
			clearAPB;

			if(PSEL) begin
				readyToRW(!PWRITE, PWRITE);//Prepare memory to be read or written to
				if(!PWAIT)
					next_state = ACCESS;// ACESS STATE
				else if(PWAIT)
					next_state = WAIT;// WAIT STATE
				else if(!PENABLE)
					next_state = next_state.first;//IDLE STATE
				end
			else
				next_state = next_state.first;//IDLE STATE
		end

		ACCESS: begin
			if(!ce) begin
				readyToRW(!PWRITE, PWRITE); //IF we are performing another transfer
			end
			if(!PSEL && !PENABLE) begin
				clearAPB;
				next_state = next_state.first;//IDLE STATE
			end
			else if(next_count > 4'b0001) begin
				next_state = WAIT;//Wait value is greater than 1
			end
			else if(PSEL && PENABLE && PWRITE && next_count <= 4'b0001) begin // Ready to Write

				assign PREADY = 1;
				assign PRDATA = 0;//Read Data from Memory
				assign data = PWDATA;//Write Data to Memory

				end
			else if(PSEL && PENABLE && PWRITE == 1'b0 && next_count <= 4'b0001) begin // Ready to
Read
				assign PREADY = 1;
				assign PRDATA = MEMRDATA;//Read Data from Memory
				assign data = 0;//Write Data to Memory
				end
			else begin //PREADY == 1 and another transfer
				next_state = ACCESS;
				clearAPB;
				end
			end

		WAIT: begin
			if(!PENABLE) begin
				next_state = WAIT;//Keep waiting
			end
			else if(PENABLE) begin
				next_count = next_count - 1;//1 Clock Cycle Passed
				if(next_count == 0) begin
					next_state = ACCESS; //All Clock Cycles done so time to read/write
				end
				else begin
					next_state = WAIT2;//Keep waiting
				end
```

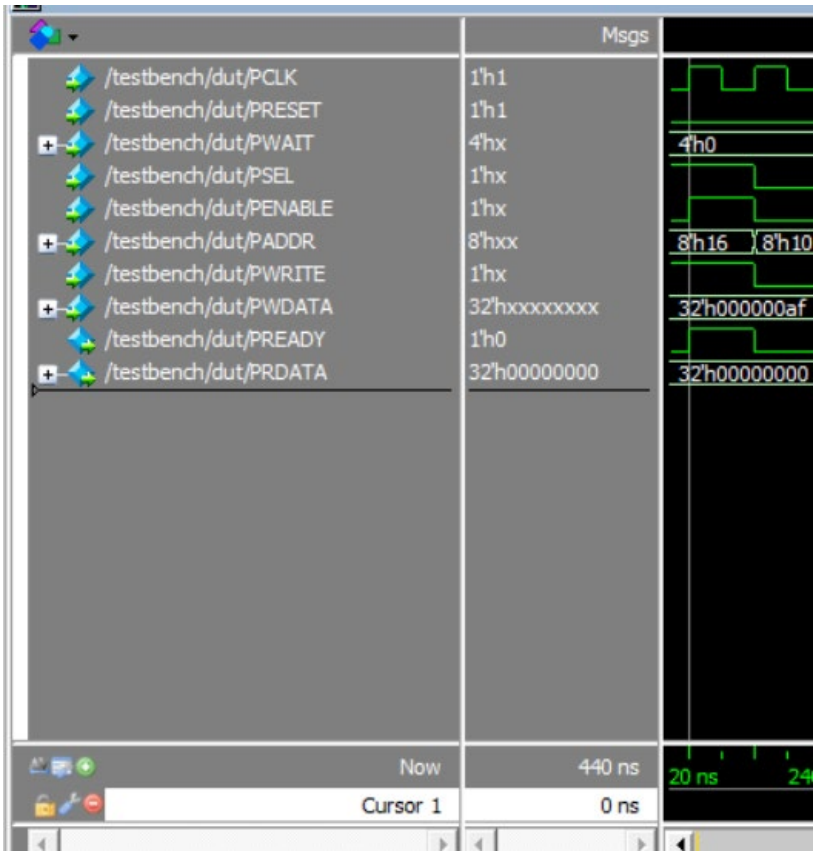
```
                end
        end

        WAIT2: begin//This is a 2nd wait state for waits > 2 cycles
                if(!PENABLE) begin
                        next_state = WAIT;//Keep waiting
                end
                else if(PENABLE) begin
                        next_count = next_count - 1;//1 Clock Cycle Passed
                        if(next_count == 0) begin
                                next_state = ACCESS; //All Clock Cycles done so time to read/write
                        end
                        else begin
                                next_state = WAIT;//Keep waiting
                        end
                end
        end
    endcase
end
endmodule: apb
```

Wave Forms

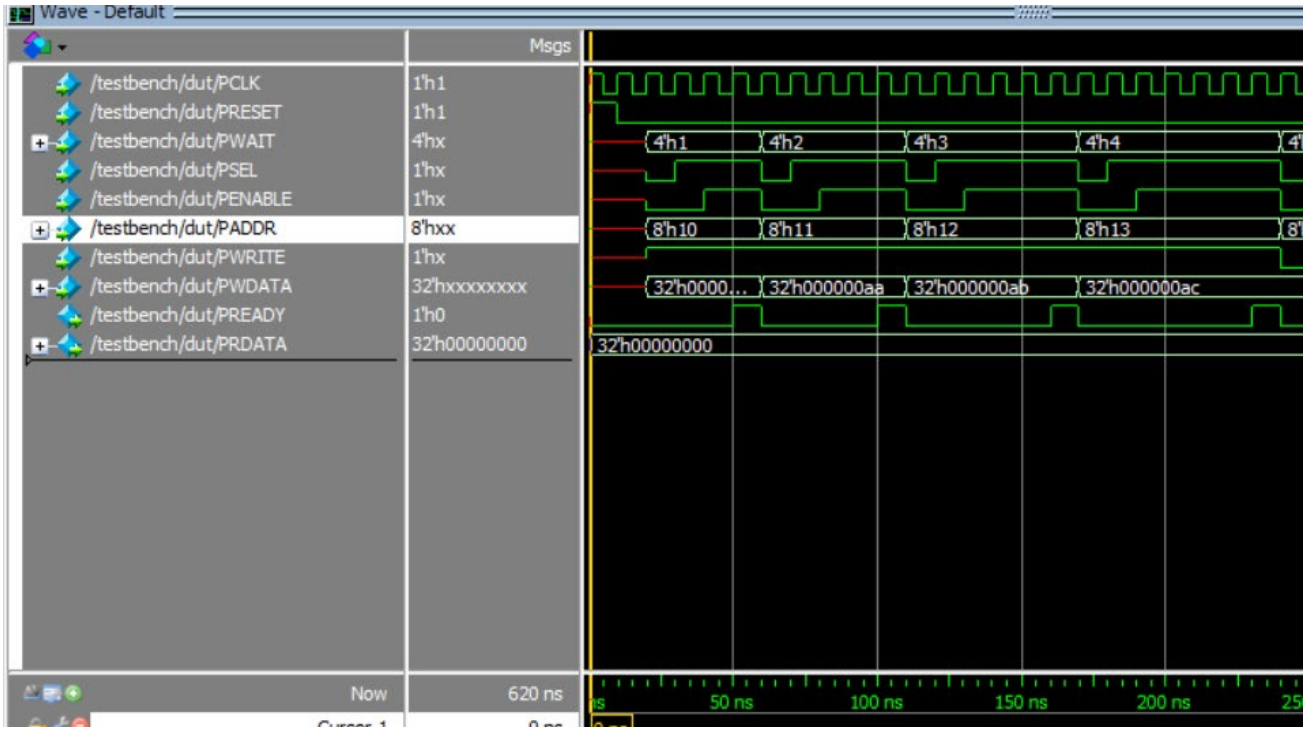
***Wave forms are from 2 runs (1 with wait states and 1 without)**

Write transfer with no wait states

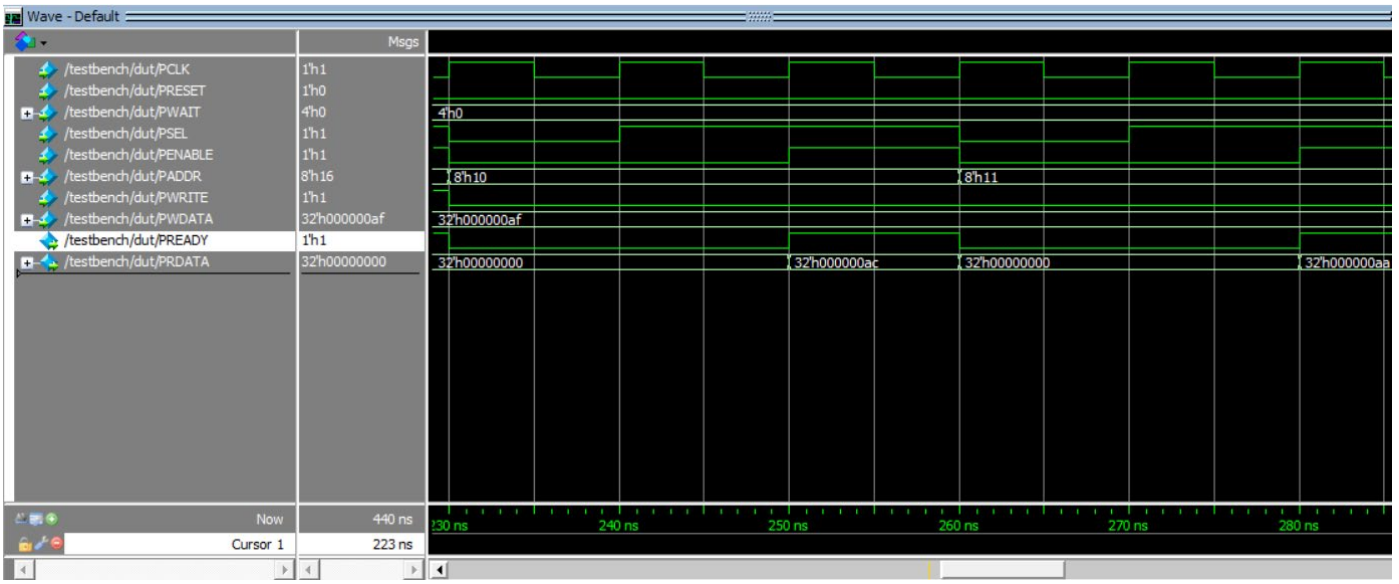


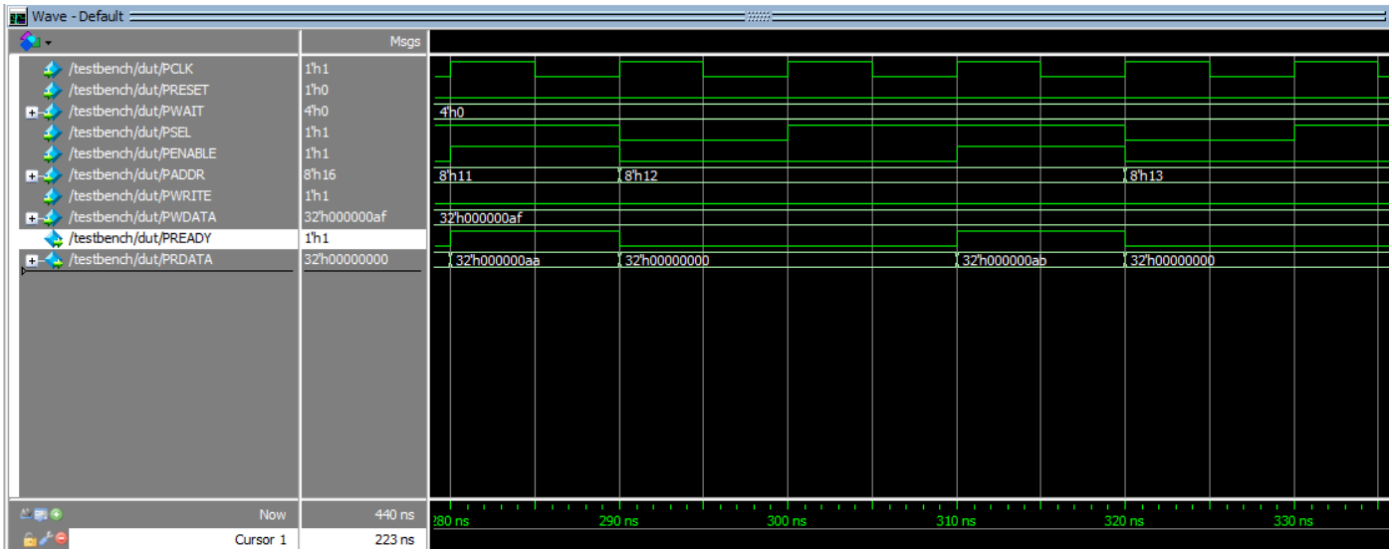


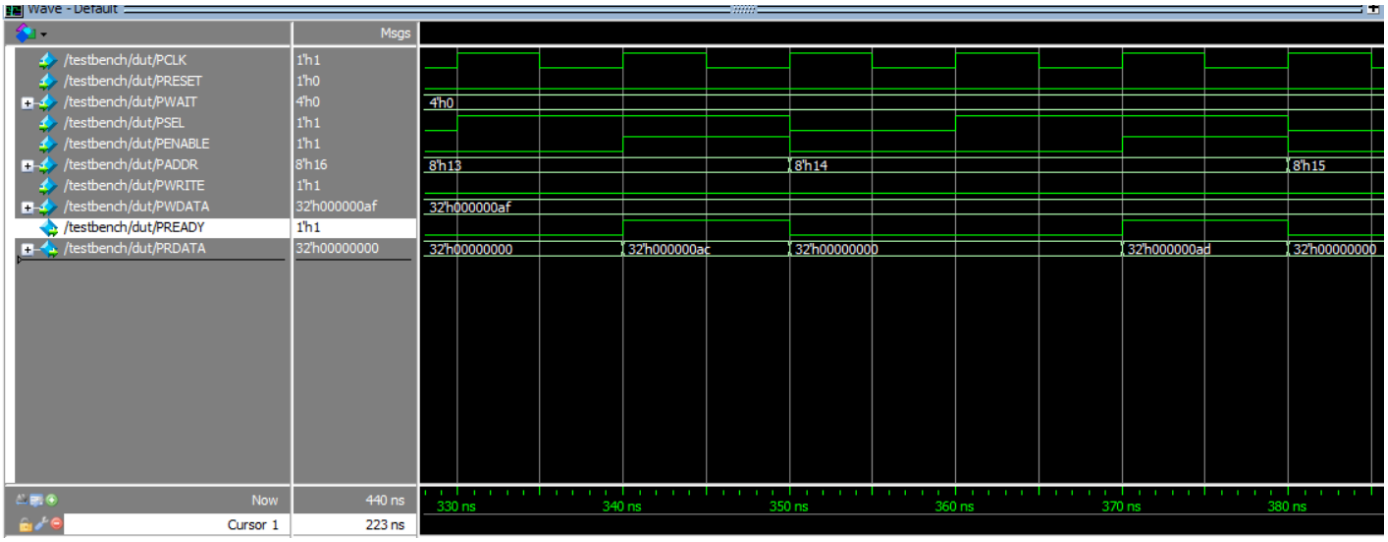
Write transfer with wait states

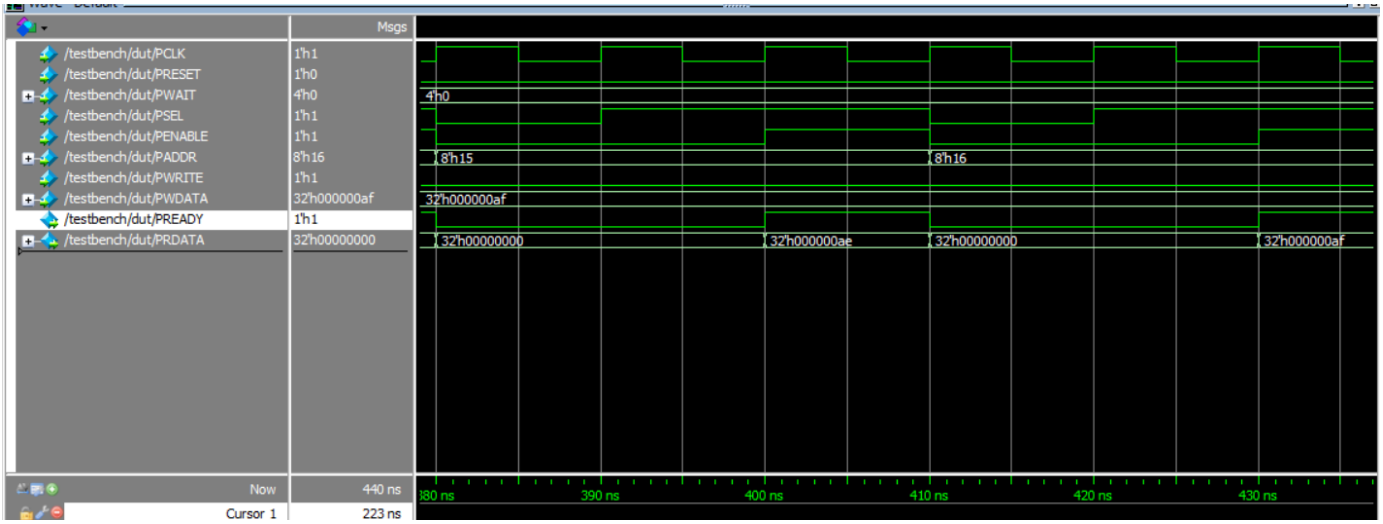


Read transfer with no wait states

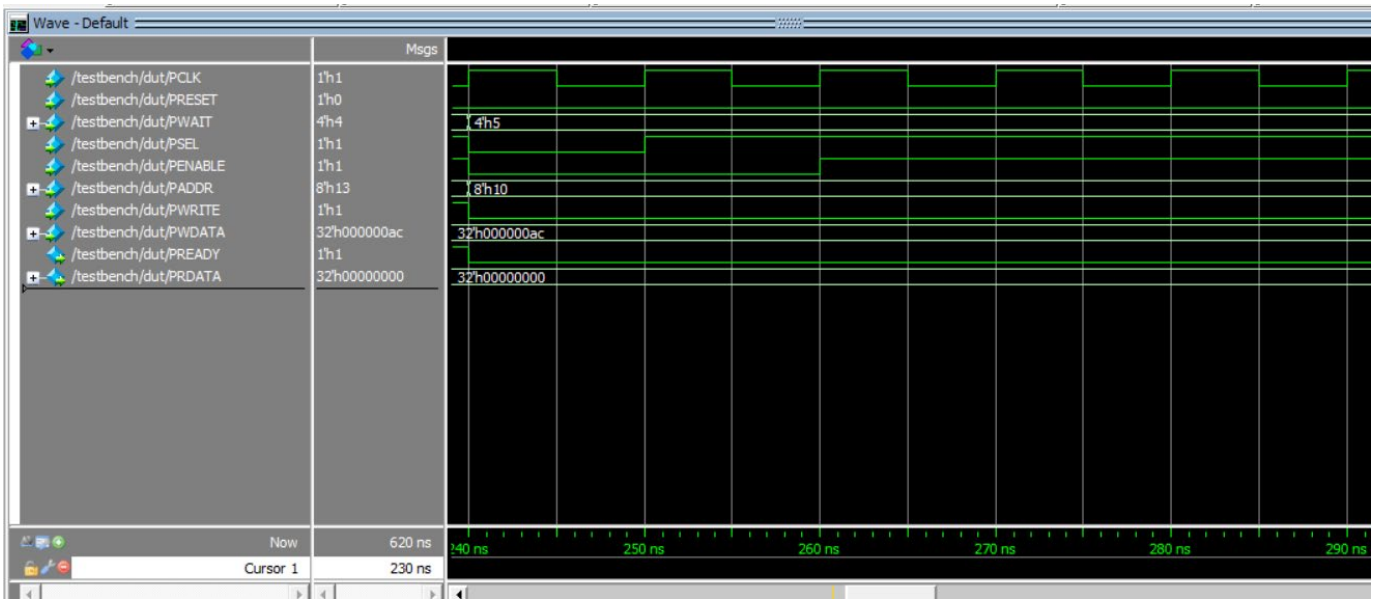


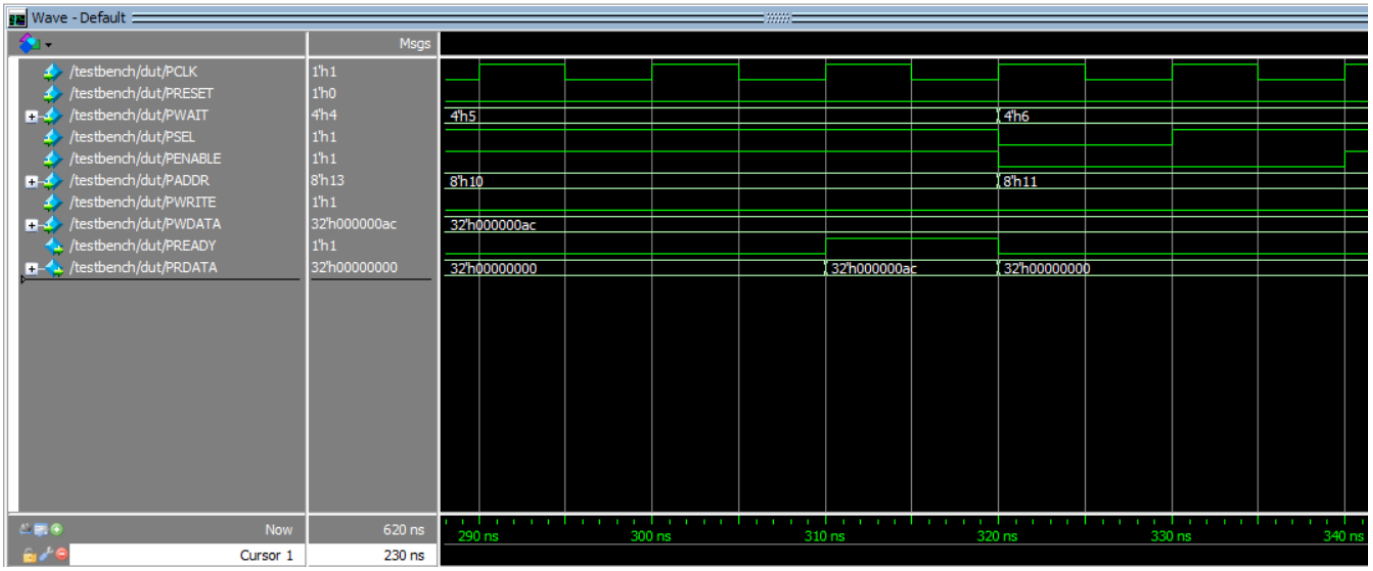


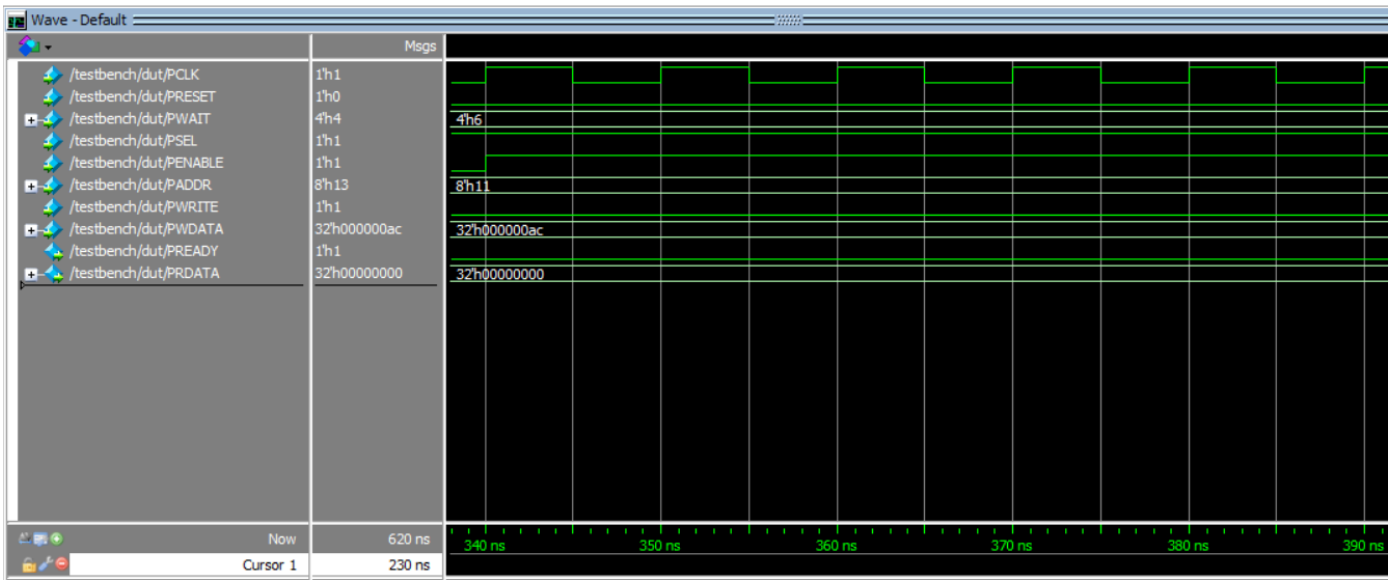

Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h0
1'h1
1'h1
8'h16
1'h1
32'h000000af
1'h1
32'h00000000
8'h13
8'h14
8'h15
32'h000000af
32'h00000000
32'h000000ac
32'h00000000
32'h000000ad
32'h00000000
Now
440 ns
Cursor 1
223 ns
330 ns
340 ns
350 ns
360 ns
370 ns
380 ns

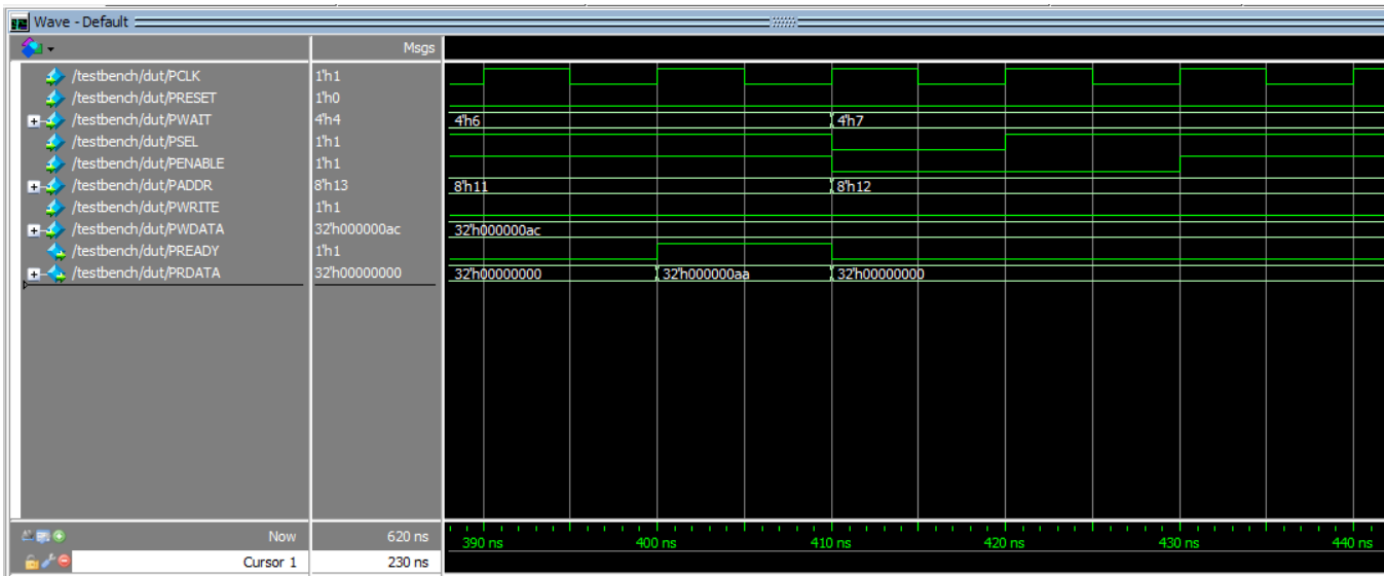

Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h0
1'h1
1'h1
8'h16
1'h1
32'h000000af
1'h1
32'h00000000
4'h0
8'h15
8'h16
32'h000000af
32'h00000000
32'h000000ae
32'h00000000
32'h000000af
Now
440 ns
Cursor 1
223 ns
380 ns
390 ns
400 ns
410 ns
420 ns
430 ns

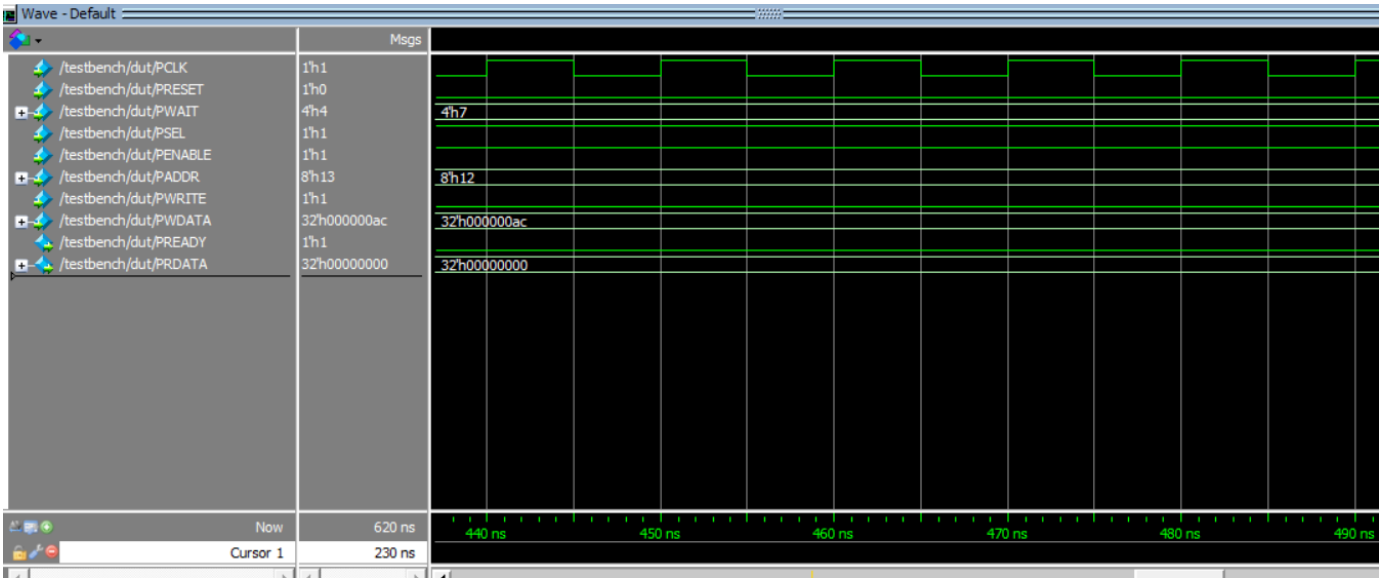
Read transfer with wait states

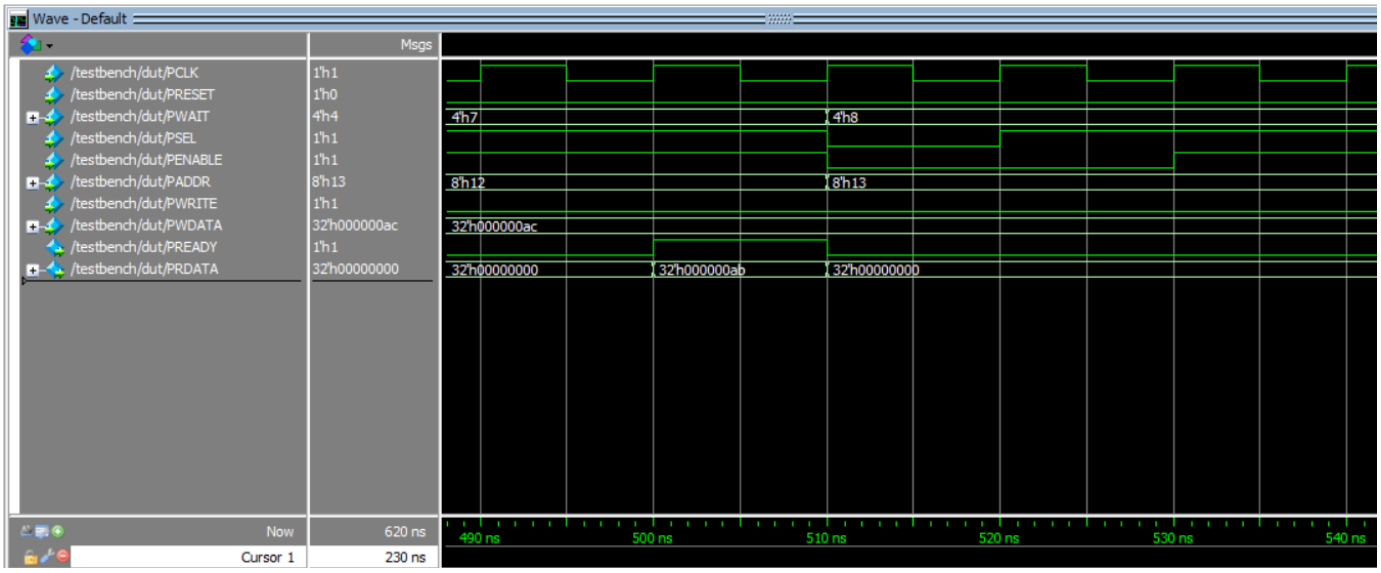


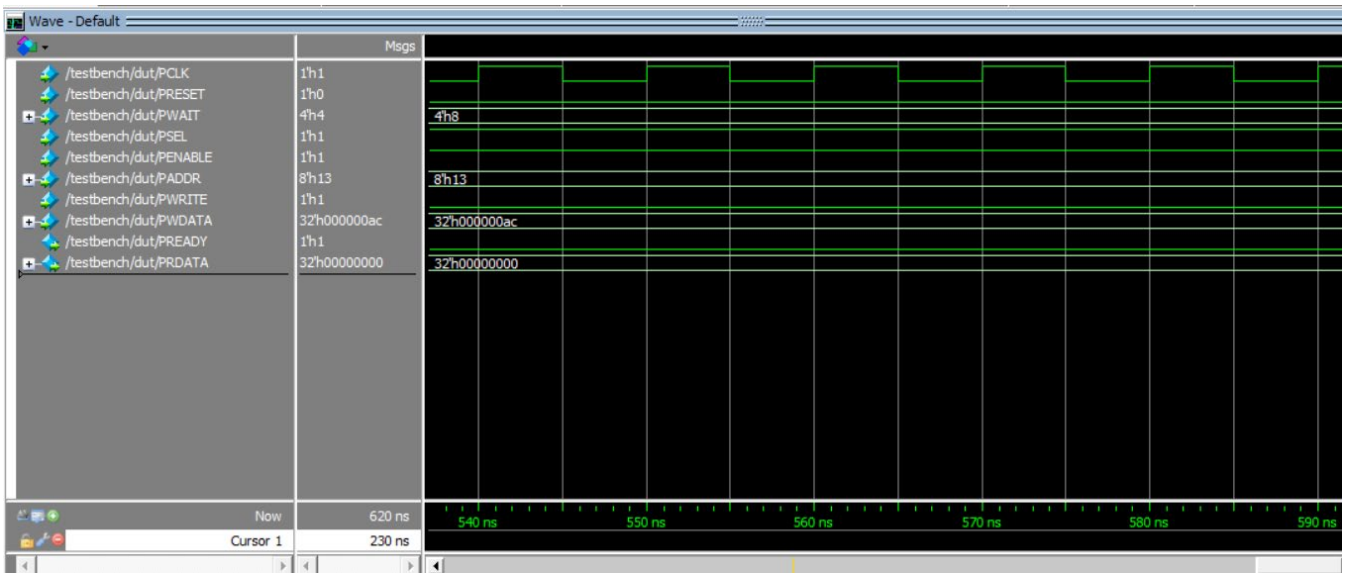


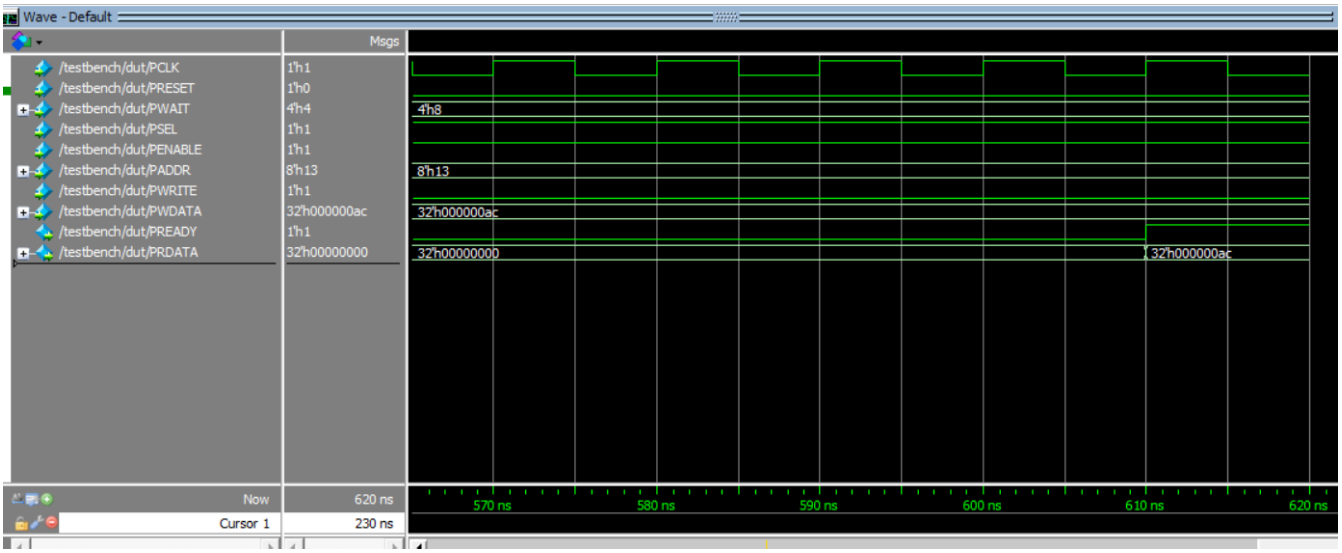

Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h6
8'h11
32'h000000ac
32'h00000000
Now
620 ns
Cursor 1
230 ns
340 ns
350 ns
360 ns
370 ns
380 ns
390 ns


Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h6
4'h7
8'h11
8'h12
32'h000000ac
32'h00000000
32'h000000aa
32'h00000000
Now
620 ns
Cursor 1
230 ns
390 ns
400 ns
410 ns
420 ns
430 ns
440 ns


Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h7
8'h12
32'h000000ac
32'h00000000
Now
620 ns
Cursor 1
230 ns
440 ns
450 ns
460 ns
470 ns
480 ns
490 ns


Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h7
4'h8
8'h12
8'h13
32'h000000ac
32'h00000000
32'h000000ab
32'h00000000
Now
620 ns
Cursor 1
230 ns
490 ns
500 ns
510 ns
520 ns
530 ns
540 ns


Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h8
8'h13
32'h000000ac
32'h00000000
Now
620 ns
Cursor 1
230 ns
540 ns
550 ns
560 ns
570 ns
580 ns
590 ns


Wave - Default
Msgs
/testbench/dut/PCLK
/testbench/dut/PRESET
/testbench/dut/PWAIT
/testbench/dut/PSEL
/testbench/dut/PENABLE
/testbench/dut/PADDR
/testbench/dut/PWRITE
/testbench/dut/PWDATA
/testbench/dut/PREADY
/testbench/dut/PRDATA
1'h1
1'h0
4'h4
1'h1
1'h1
8'h13
1'h1
32'h000000ac
1'h1
32'h00000000
4'h8
8'h13
32'h000000ac
32'h00000000
32'h000000ac
Now
620 ns
Cursor 1
230 ns
570 ns
580 ns
590 ns
600 ns
610 ns
620 ns