# Verilog for Modeling

Module 9

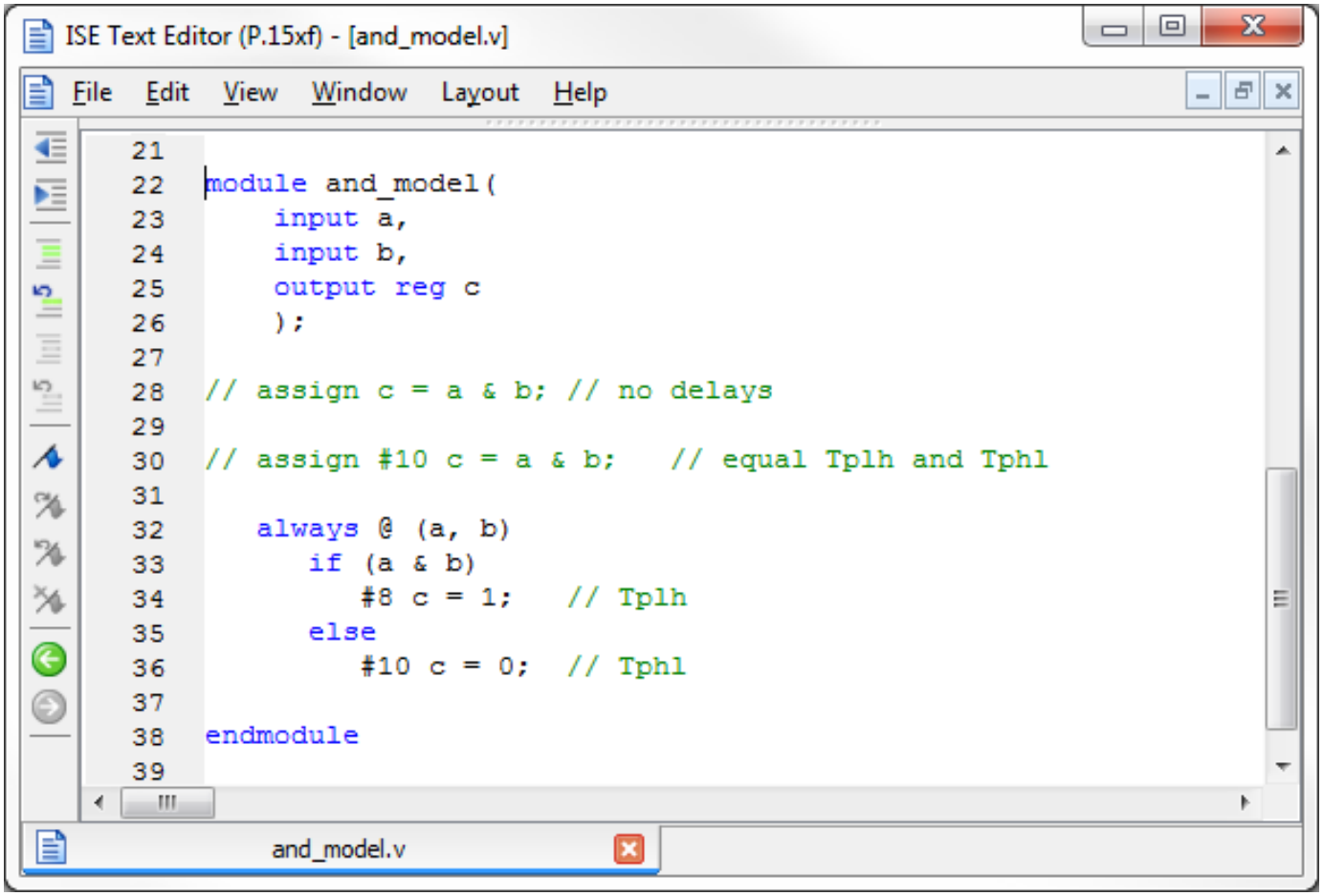
# Overview

- General examples
  - AND model
  - Flip-flop model
  - SRAM Model
- Customizing Models
  - Generics in VHDL
    - DDR SDRAM Model
  - Parameters in Verilog
- Commercial memory models

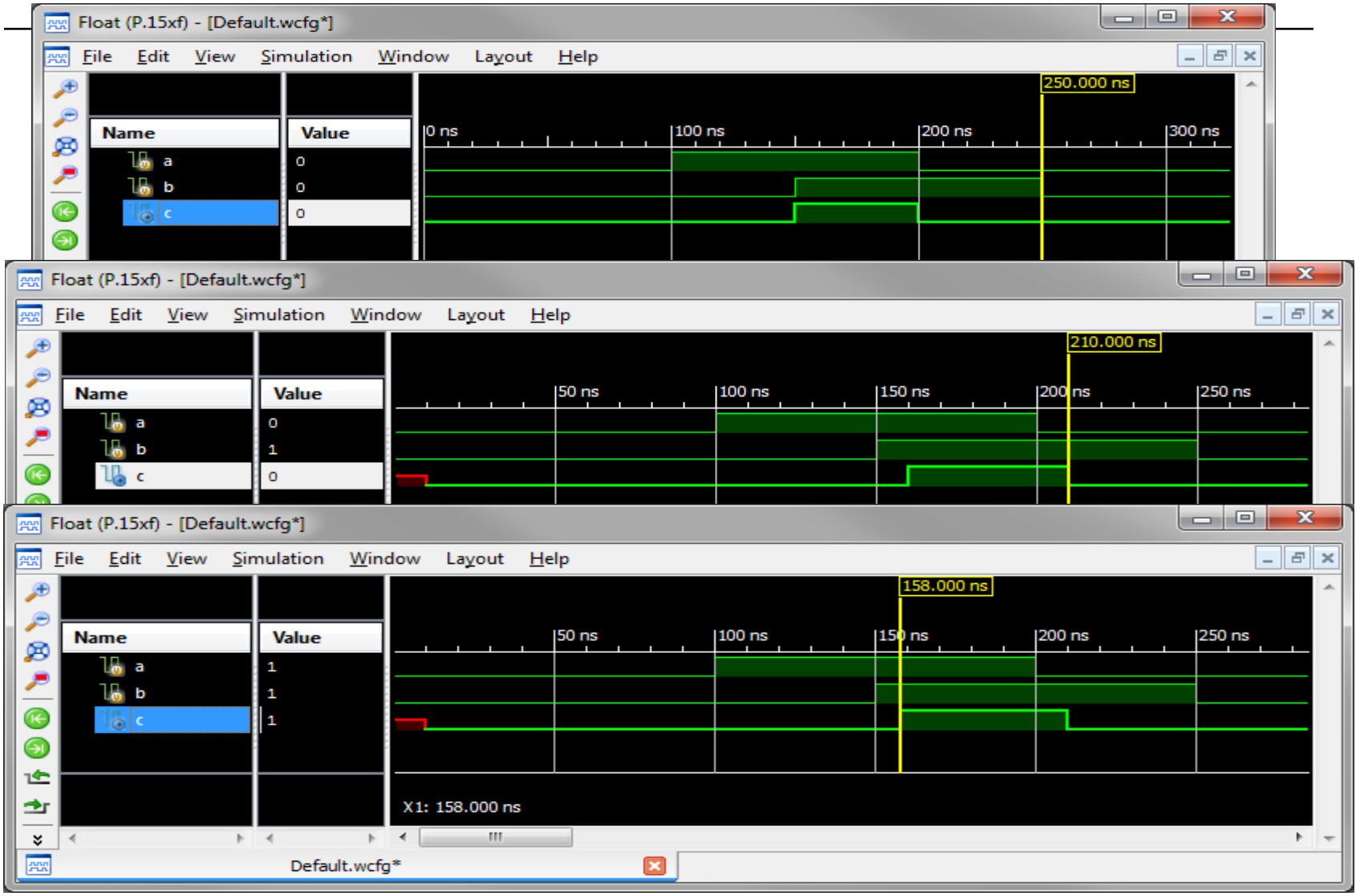
# Verilog for Modeling

- We have covered
  - Verilog for Synthesis
  - Verilog for testing (simulation)
- Now - Verilog for modeling
- Describes the *expected behavior* of a component or device
- Can be used to test other components
  - for example a model of a CPU could be used to test:
    - UART
    - DRAM memory controller
    - cache controller

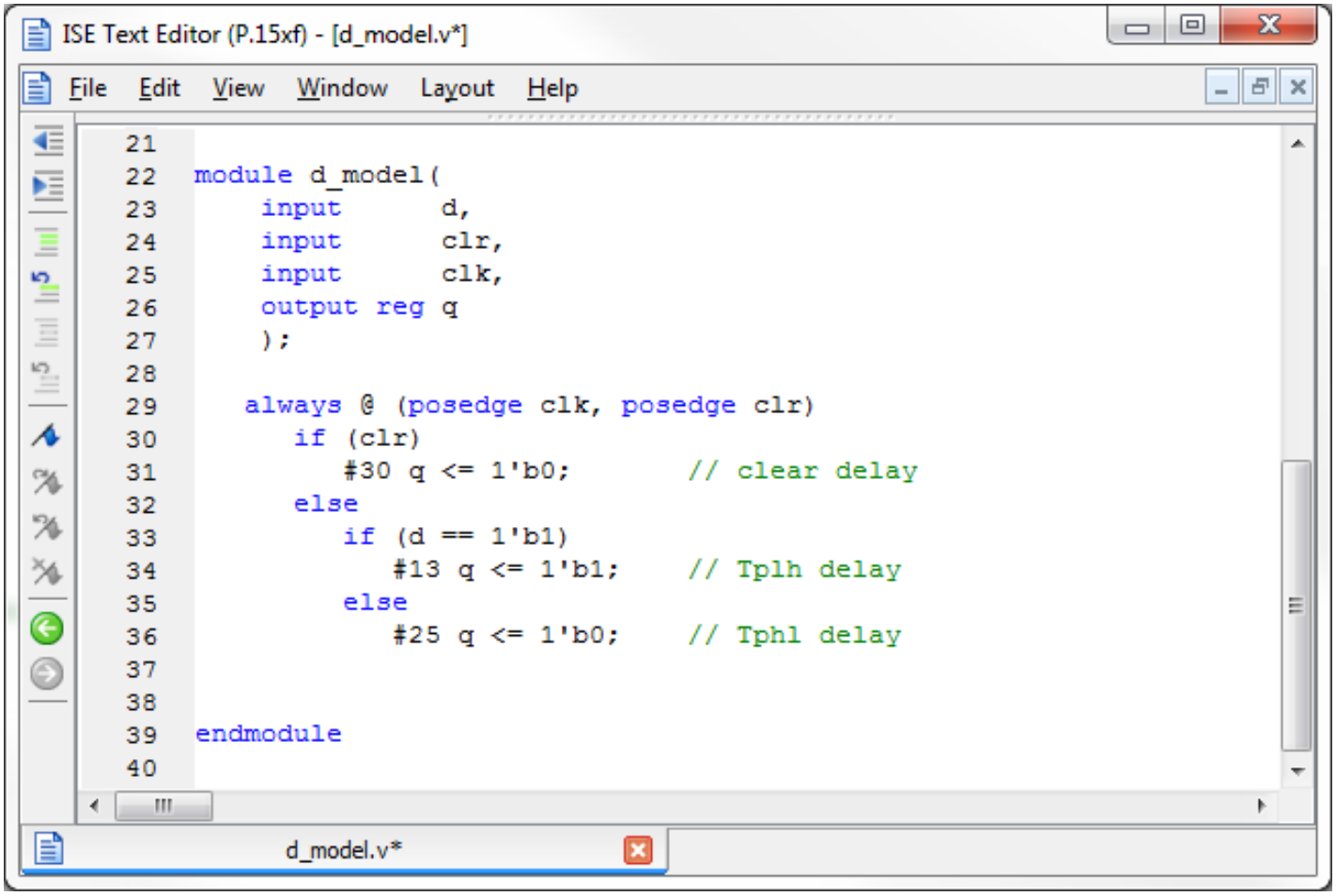
# AND gate model



# Simulation Results



# D flip-flop model

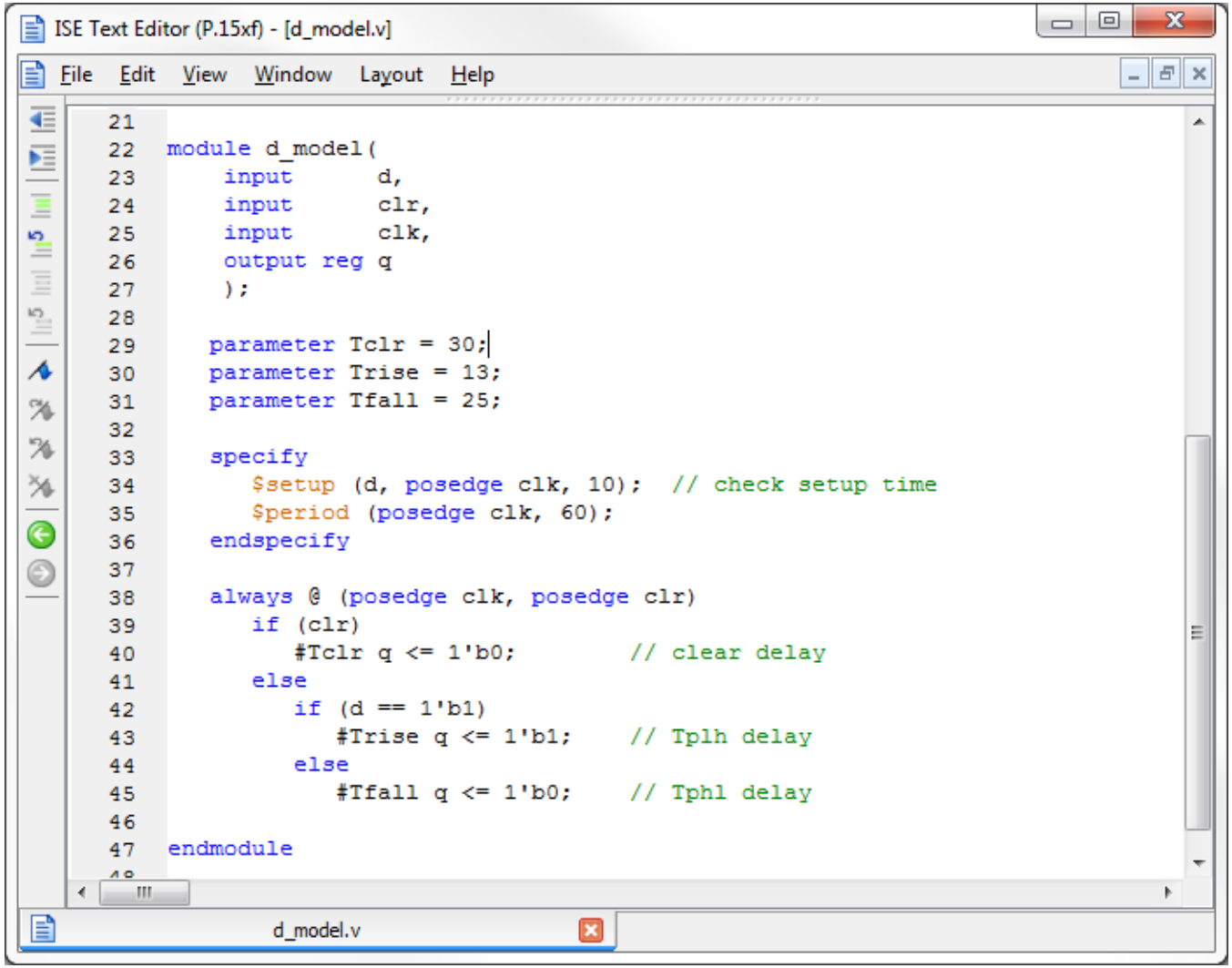



```
21
22  module d_model(
23      input       d,
24      input       clr,
25      input       clk,
26      output reg q
27      );
28
29     always @ (posedge clk, posedge clr)
30        if (clr)
31           #30 q <= 1'b0;        // clear delay
32        else
33           if (d == 1'b1)
34              #13 q <= 1'b1;     // Tplh delay
35           else
36              #25 q <= 1'b0;     // Tphl delay
37
38
39  endmodule
40
```

# Timing Check Tasks in Verilog

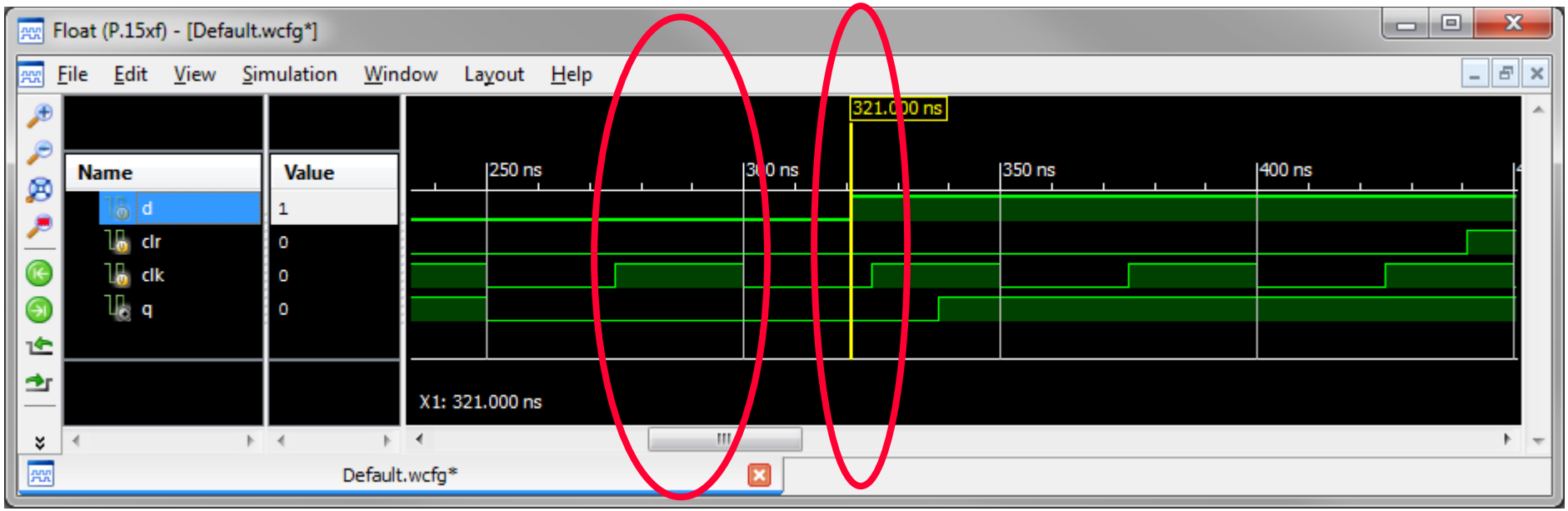
- Specify block can be used to specify setup and hold times for signals
  - specify and endspecify (Use specparam to define parameters in specify block)
- $setup (data, clock edge, limit)
  - Displays warning message if setup timing constraint is not met
  - $setup(d, posedge clk, 10)
- $hold (clock edge, data, limit)
  - Displays warning message if hold timing constraint is not met
  - $hold(posedge clk, d, 2)
- $width (pulse event, limit)
  - Displays warning message if pulse width is shorter than limit
  - $width(posedge clk, 20) – specify start edge of pulse
- $period (pulse event, limit)
  - Check if period of signal is sufficiently long
  - $period(posedge clk, 50)

# Adding setup and period timing checks



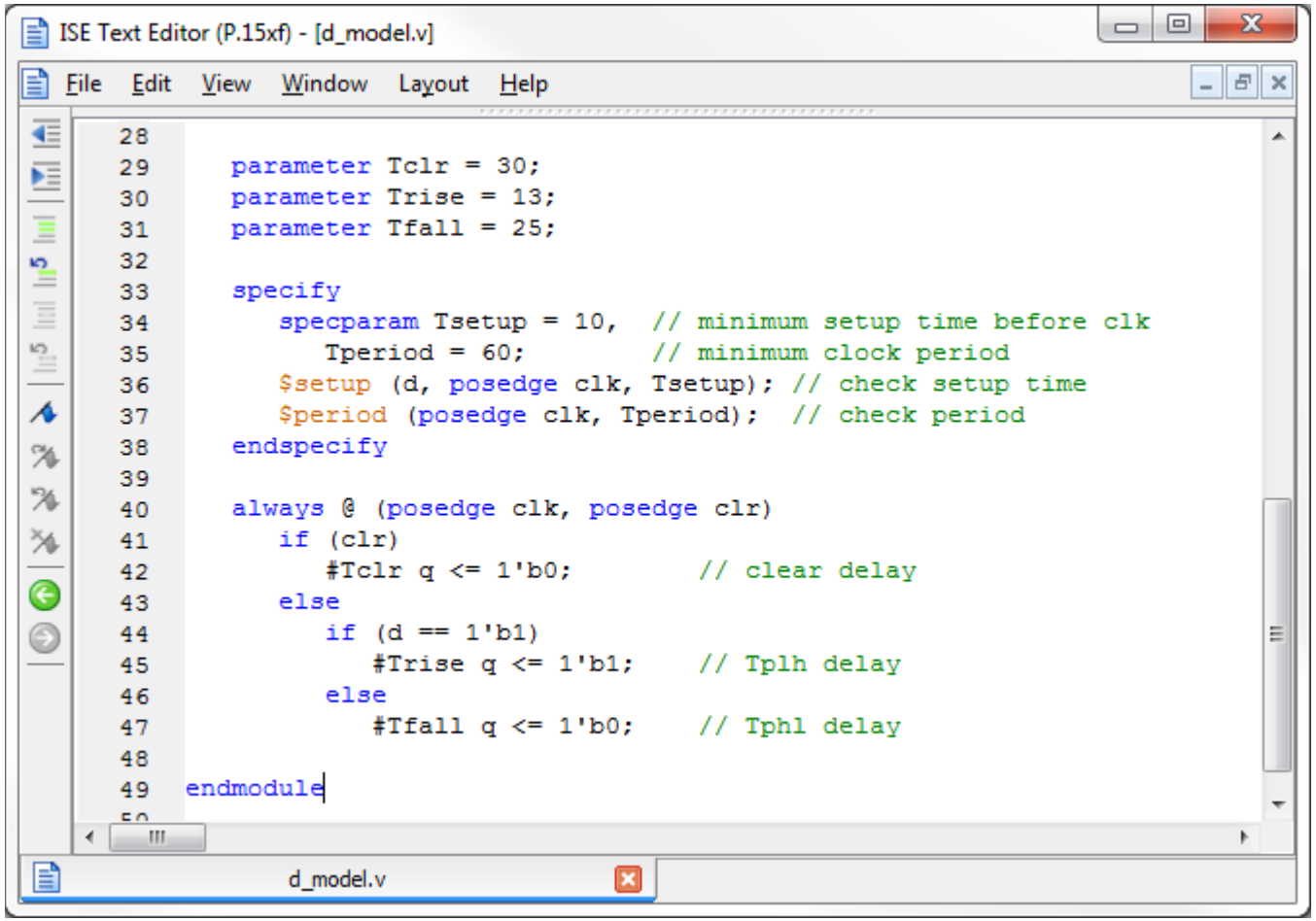

```
21
22  module d_model(
23      input       d,
24      input       clr,
25      input       clk,
26      output reg q
27      );
28
29     parameter Tclr = 30;
30     parameter Trise = 13;
31     parameter Tfall = 25;
32
33     specify
34        $setup (d, posedge clk, 10);  // check setup time
35        $period (posedge clk, 60);
36     endspecify
37
38     always @ (posedge clk, posedge clr)
39        if (clr)
40           #Tclr q <= 1'b0;         // clear delay
41        else
42           if (d == 1'b1)
43              #Trise q <= 1'b1;     // Tplh delay
44           else
45              #Tfall q <= 1'b0;     // Tphl delay
46
47  endmodule
```

# Detecting timing violations



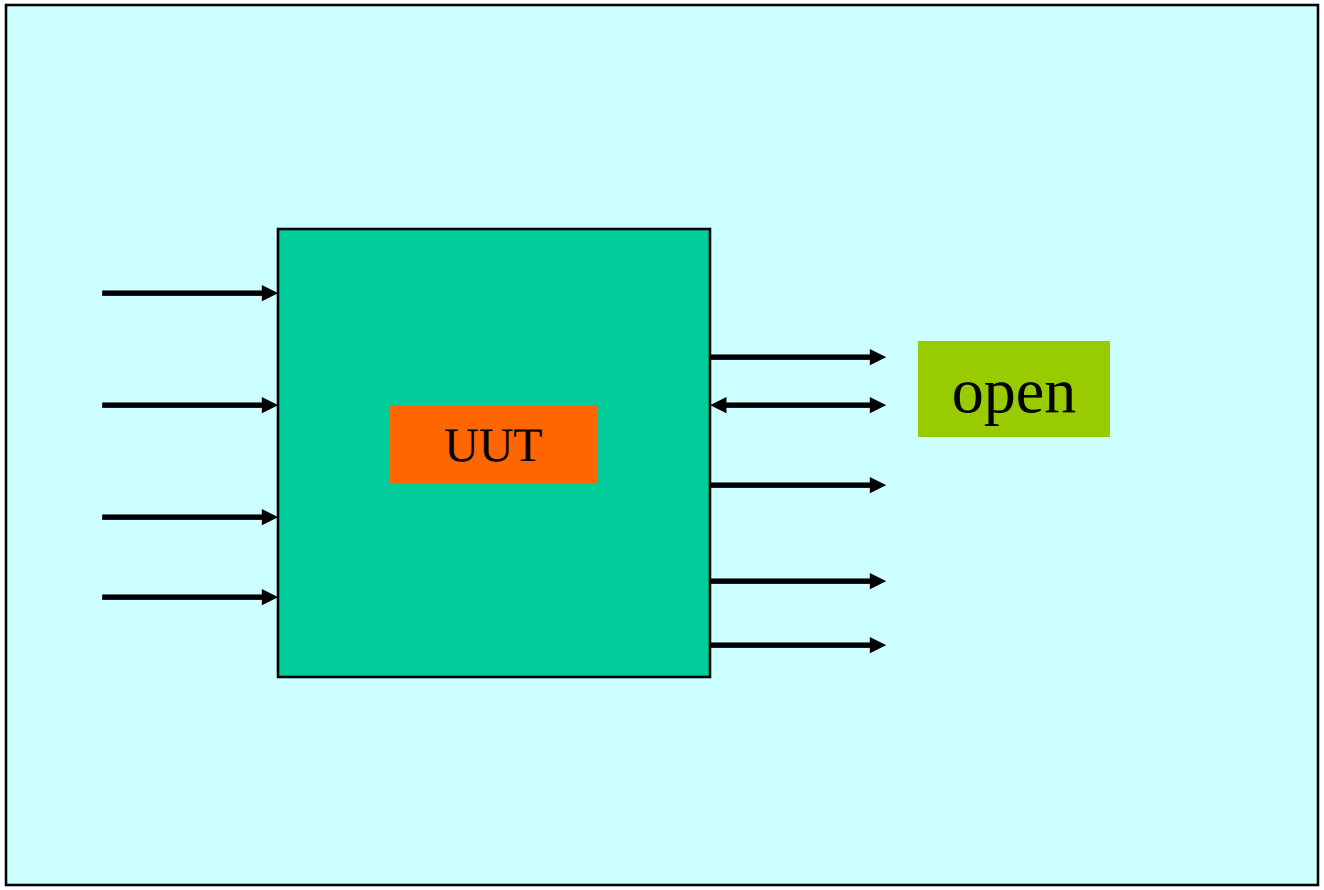

WARNING: at 225 ns: Timing violation in /d_model_tf/uut/ $period( clk:175 ns, :225 ns, 60 ns)
WARNING: at 275 ns: Timing violation in /d_model_tf/uut/ $period( clk:225 ns, :275 ns, 60 ns)
WARNING: at 325 ns: Timing violation in /d_model_tf/uut/ $setup( d:321 ns, clk:325 ns,10 ns)
WARNING: at 325 ns: Timing violation in /d_model_tf/uut/ $period( clk:275 ns, :325 ns, 60 ns)

# Using Specparam



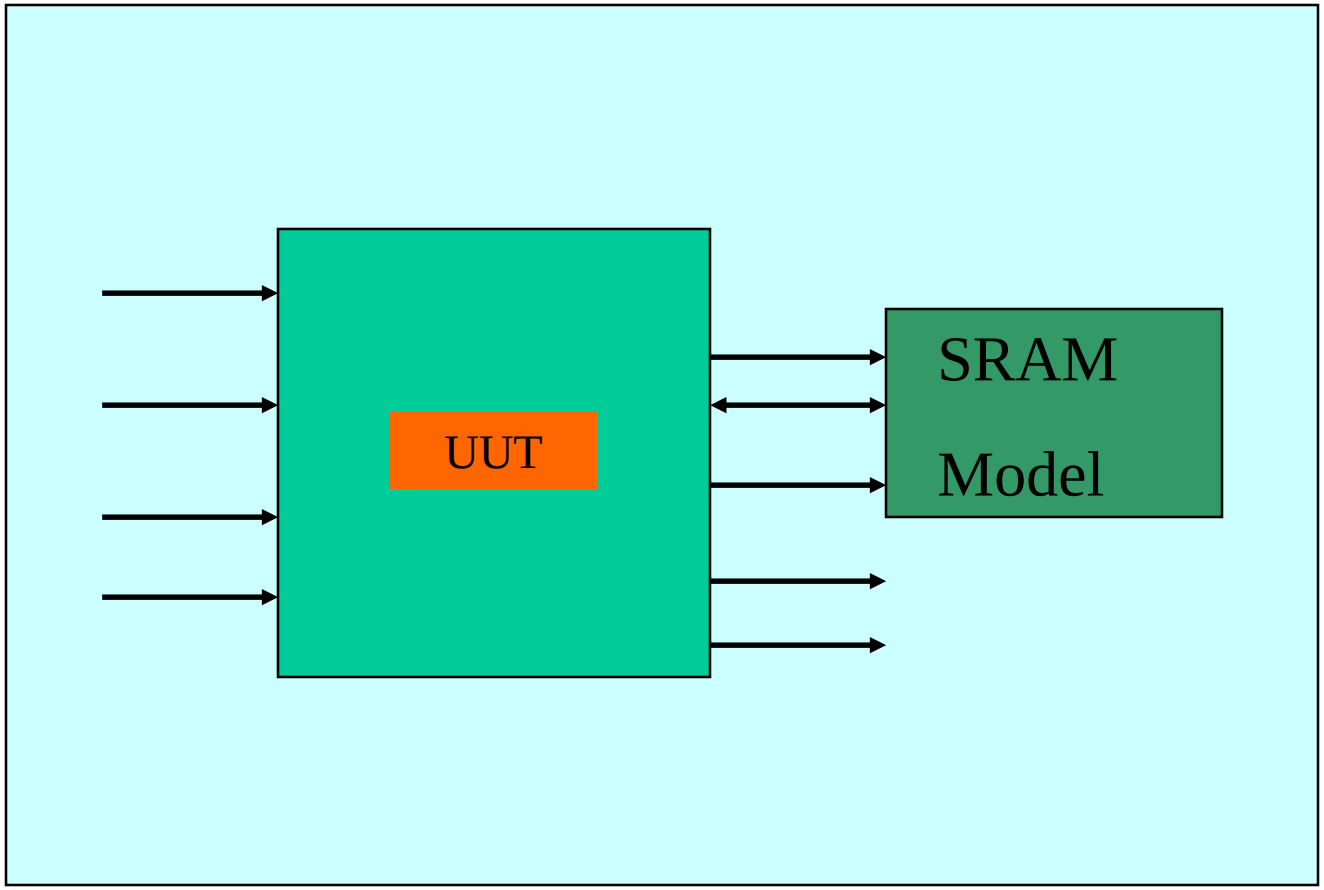
# Test Bench – no models

- SRAM connections are open

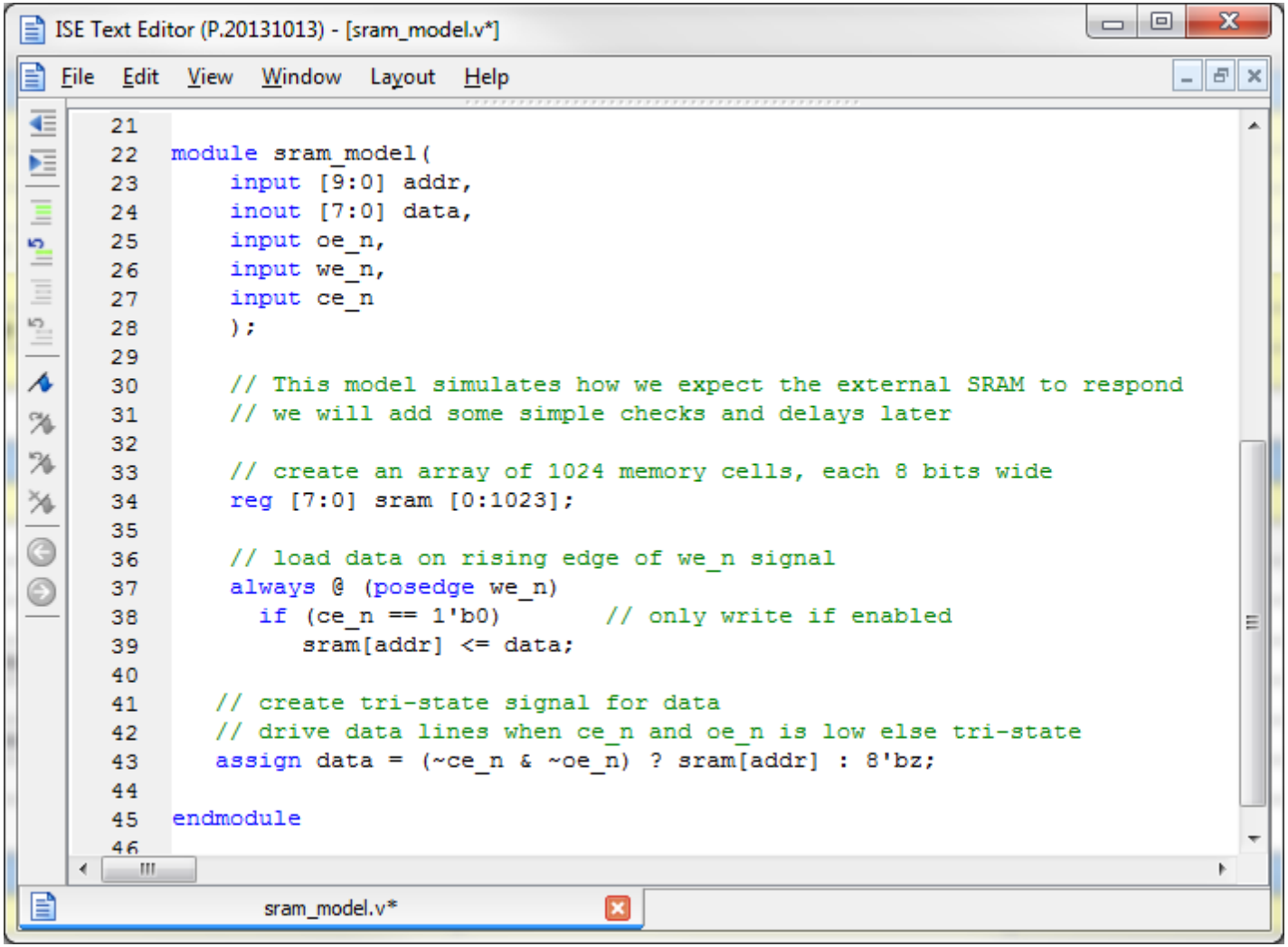


# Adding the SRAM model

- New testbench

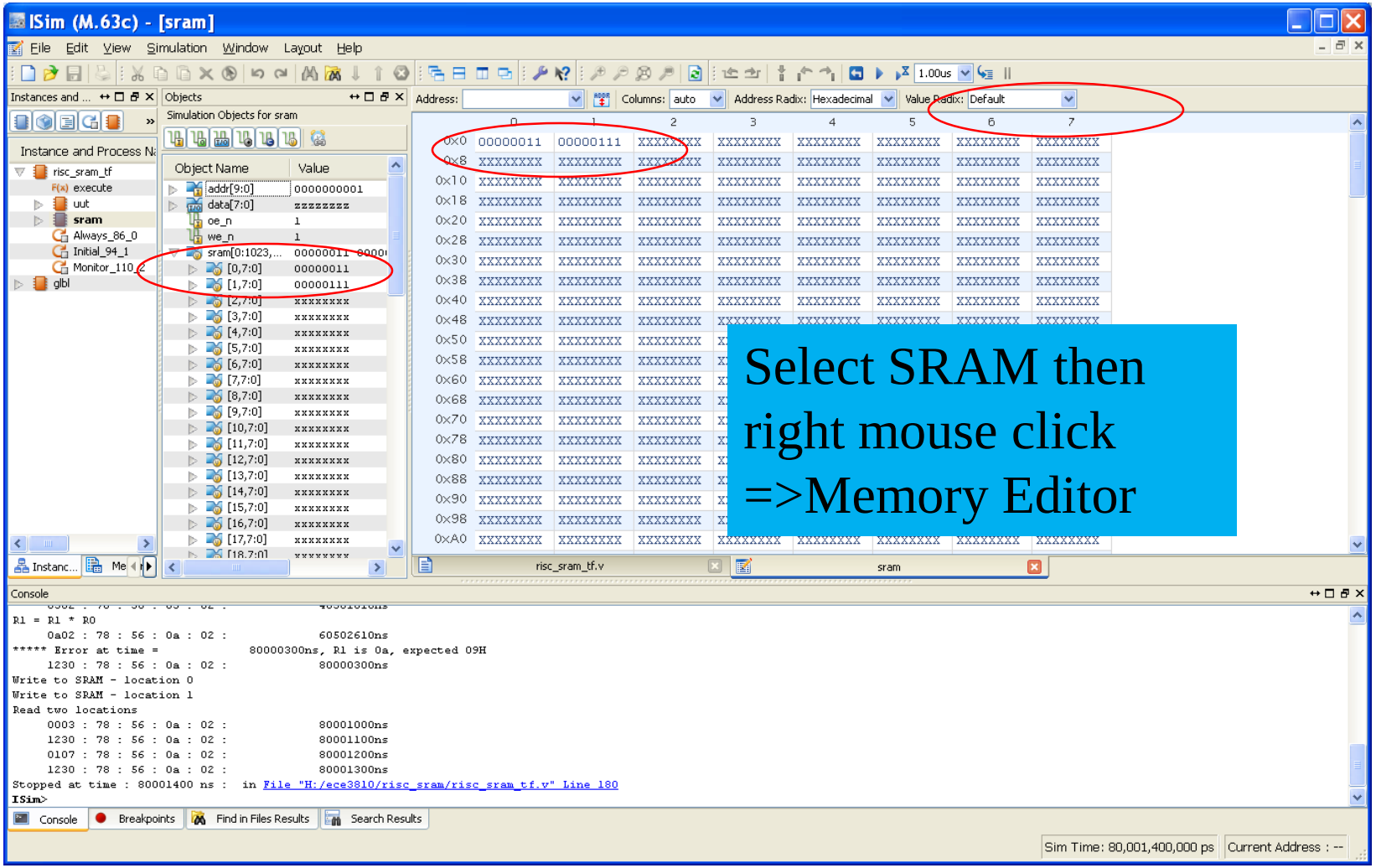


# Very Simple SRAM Model

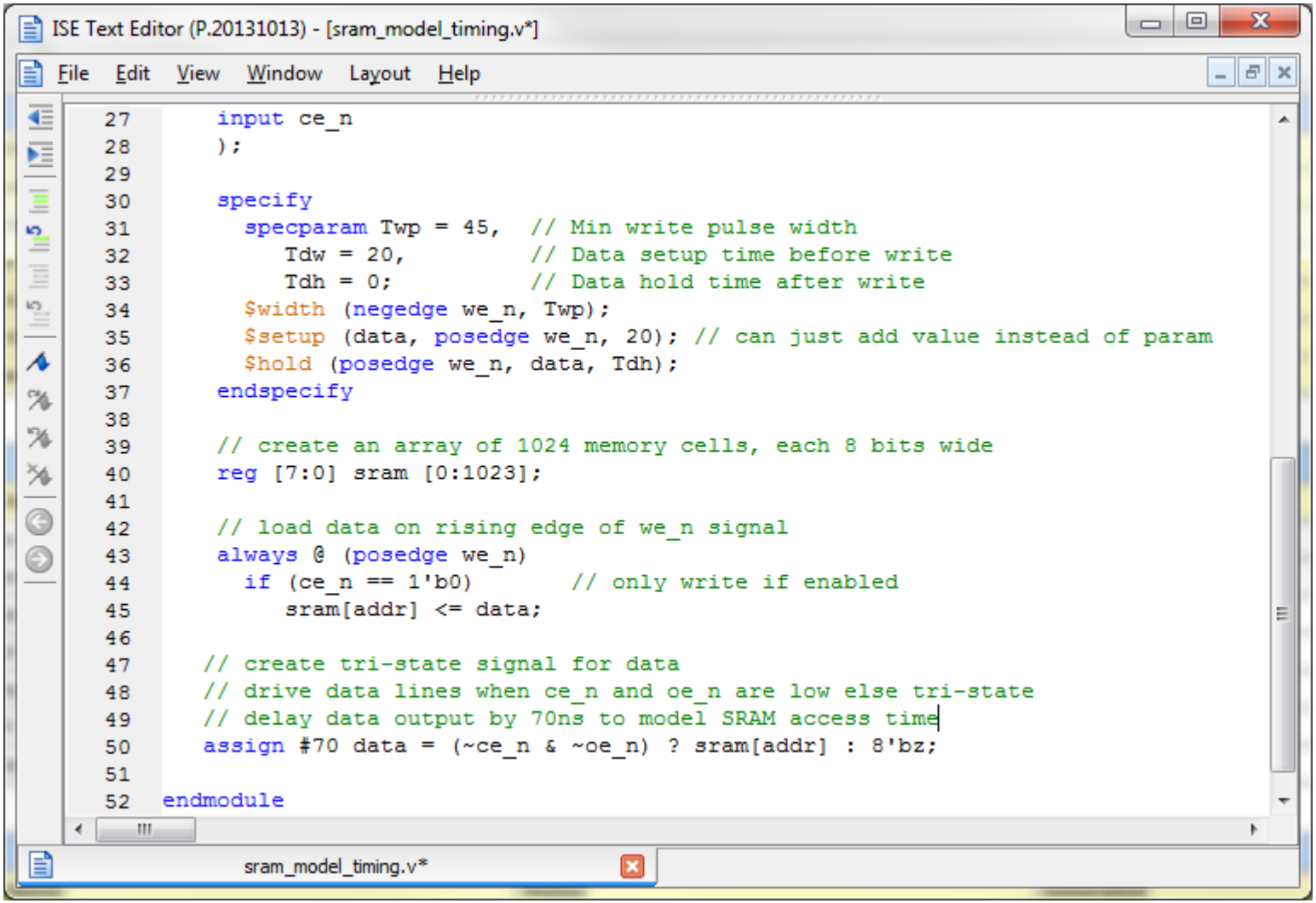



```
21
22  module sram_model(
23      input [9:0] addr,
24      inout [7:0] data,
25      input oe_n,
26      input we_n,
27      input ce_n
28      );
29
30      // This model simulates how we expect the external SRAM to respond
31      // we will add some simple checks and delays later
32
33      // create an array of 1024 memory cells, each 8 bits wide
34      reg [7:0] sram [0:1023];
35
36      // load data on rising edge of we_n signal
37      always @ (posedge we_n)
38        if (ce_n == 1'b0)        // only write if enabled
39           sram[addr] <= data;
40
41     // create tri-state signal for data
42     // drive data lines when ce_n and oe_n is low else tri-state
43     assign data = (~ce_n & ~oe_n) ? sram[addr] : 8'bz;
44
45  endmodule
46
```

# Viewing Contents of Array

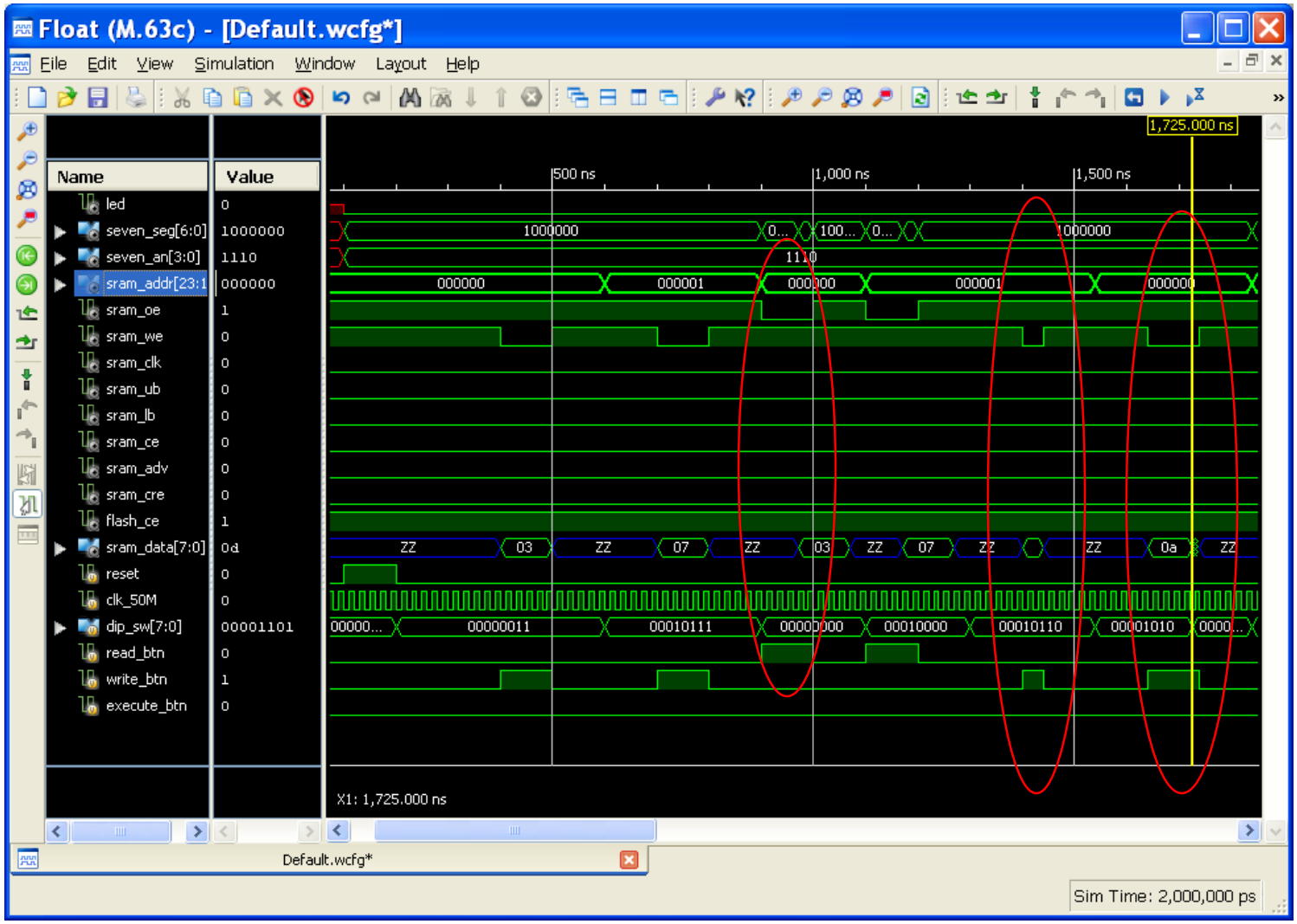


# Adding Timing Checks and Delay

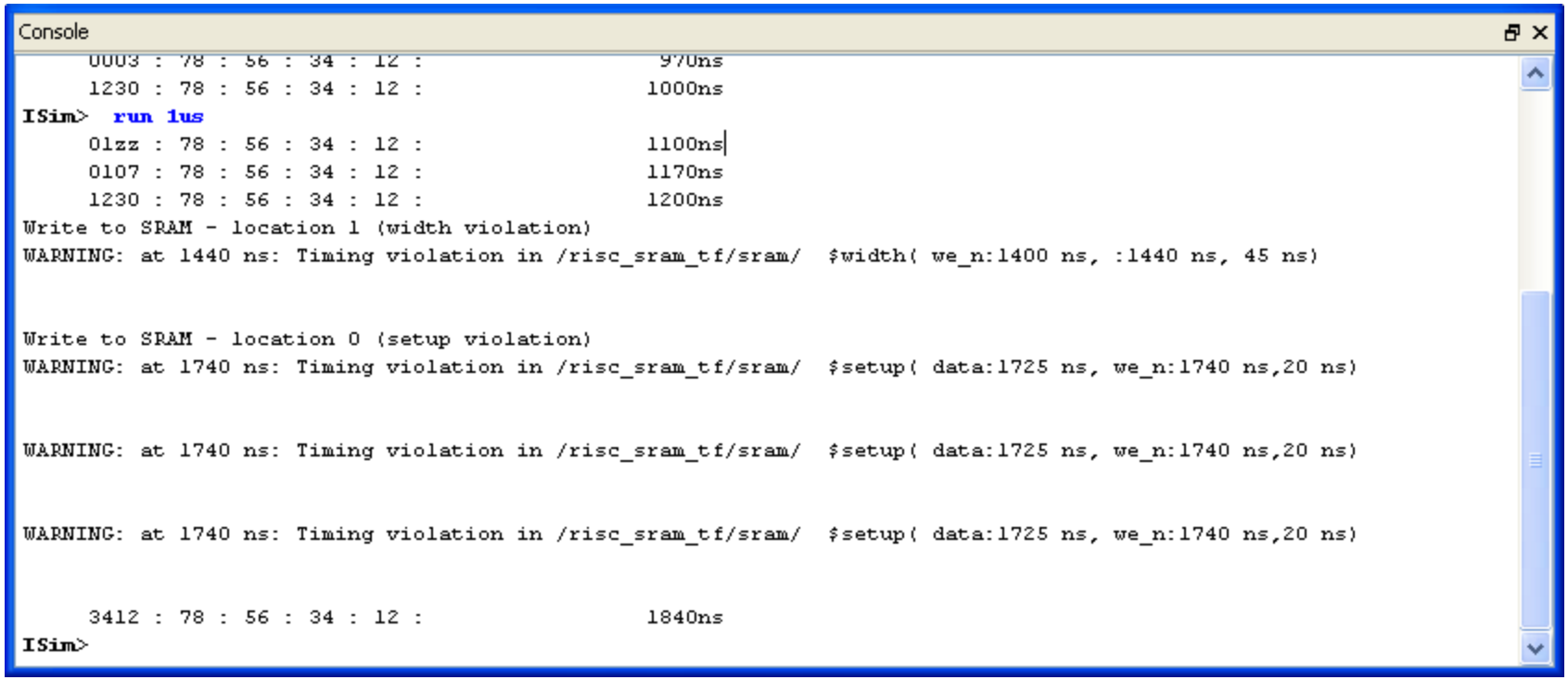



```
27      input ce_n
28      );
29
30      specify
31        specparam Twp = 45,   // Min write pulse width
32           Tdw = 20,          // Data setup time before write
33           Tdh = 0;           // Data hold time after write
34        $width (negedge we_n, Twp);
35        $setup (data, posedge we_n, 20); // can just add value instead of param
36        $hold (posedge we_n, data, Tdh);
37      endspecify
38
39      // create an array of 1024 memory cells, each 8 bits wide
40      reg [7:0] sram [0:1023];
41
42      // load data on rising edge of we_n signal
43      always @ (posedge we_n)
44        if (ce_n == 1'b0)       // only write if enabled
45           sram[addr] <= data;
46
47     // create tri-state signal for data
48     // drive data lines when ce_n and oe_n are low else tri-state
49     // delay data output by 70ns to model SRAM access time
50     assign #70 data = (~ce_n & ~oe_n) ? sram[addr] : 8'bz;
51
52  endmodule
```

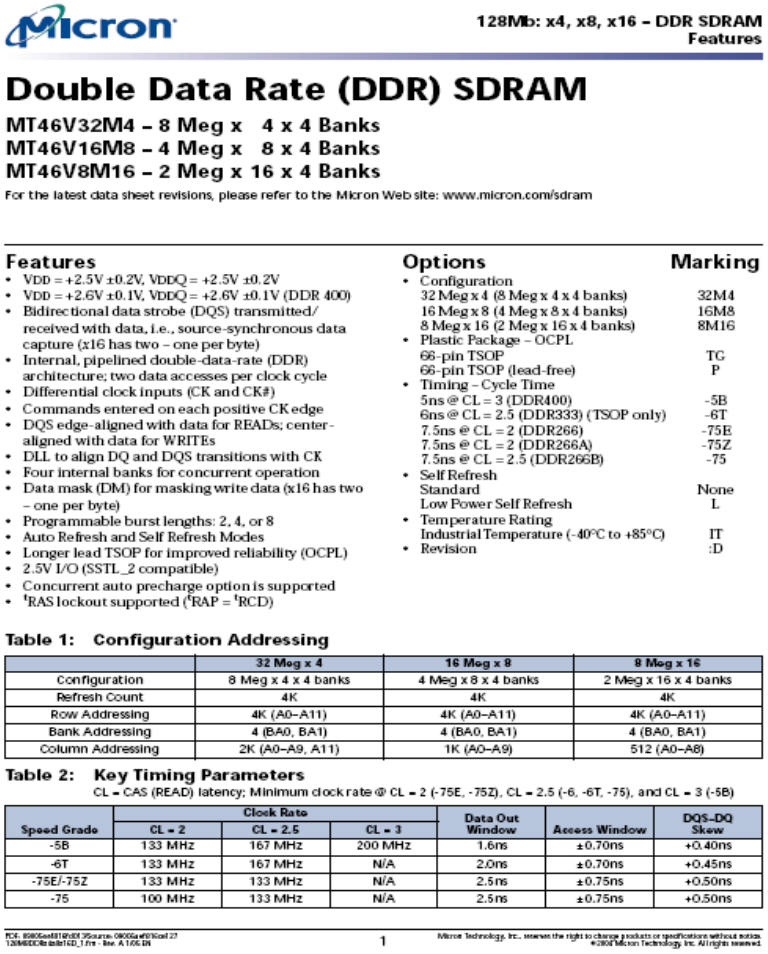
# Testing with Test Bench



# Warning Messages



# DDR SDRAM Model



128Mb: x4, x8, x16 – DDR SDRAM
Features

## Double Data Rate (DDR) SDRAM

**MT46V32M4 – 8 Meg x 4 x 4 Banks**
**MT46V16M8 – 4 Meg x 8 x 4 Banks**
**MT46V8M16 – 2 Meg x 16 x 4 Banks**

For the latest data sheet revisions, please refer to the Micron Web site: www.micron.com/sdram

### Features

- VDD = +2.5V ±0.2V, VDDQ = +2.5V ±0.2V
- VDD = +2.6V ±0.1V, VDDQ = +2.6V ±0.1V (DDR 400)
- Bidirectional data strobe (DQS) transmitted/ received with data, i.e., source-synchronous data capture (x16 has two – one per byte)
- Internal, pipelined double-data-rate (DDR) architecture; two data accesses per clock cycle
- Differential clock inputs (CK and CK#)
- Commands entered on each positive CK edge
- DQS edge-aligned with data for READs; center-aligned with data for WRITEs
- DLL to align DQ and DQS transitions with CK
- Four internal banks for concurrent operation
- Data mask (DM) for masking write data (x16 has two – one per byte)
- Programmable burst lengths: 2, 4, or 8
- Auto Refresh and Self Refresh Modes
- Longer lead TSOP for improved reliability (OCPL)
- 2.5V I/O (SSTL_2 compatible)
- Concurrent auto precharge option is supported
- $^{t}$RAS lockout supported ($^{t}$RAP = $^{t}$RCD)

### Options / Marking

| Options | Marking |
|---|---|
| • Configuration | |
| 32 Meg x 4 (8 Meg x 4 x 4 banks) | 32M4 |
| 16 Meg x 8 (4 Meg x 8 x 4 banks) | 16M8 |
| 8 Meg x 16 (2 Meg x 16 x 4 banks) | 8M16 |
| • Plastic Package – OCPL | |
| 66-pin TSOP | TG |
| 66-pin TSOP (lead-free) | P |
| • Timing – Cycle Time | |
| 5ns @ CL = 3 (DDR400) | -5B |
| 6ns @ CL = 2.5 (DDR333) (TSOP only) | -6T |
| 7.5ns @ CL = 2 (DDR266) | -75E |
| 7.5ns @ CL = 2 (DDR266A) | -75Z |
| 7.5ns @ CL = 2.5 (DDR266B) | -75 |
| • Self Refresh | |
| Standard | None |
| Low Power Self Refresh | L |
| • Temperature Rating | |
| Industrial Temperature (-40°C to +85°C) | IT |
| • Revision | :D |

**Table 1: Configuration Addressing**

| | 32 Meg x 4 | 16 Meg x 8 | 8 Meg x 16 |
|---|---|---|---|
| Configuration | 8 Meg x 4 x 4 banks | 4 Meg x 8 x 4 banks | 2 Meg x 16 x 4 banks |
| Refresh Count | 4K | 4K | 4K |
| Row Addressing | 4K (A0–A11) | 4K (A0–A11) | 4K (A0–A11) |
| Bank Addressing | 4 (BA0, BA1) | 4 (BA0, BA1) | 4 (BA0, BA1) |
| Column Addressing | 2K (A0–A9, A11) | 1K (A0–A9) | 512 (A0–A8) |

**Table 2: Key Timing Parameters**

CL = CAS (READ) latency; Minimum clock rate @ CL = 2 (-75E, -75Z), CL = 2.5 (-6, -6T, -75), and CL = 3 (-5B)

| Speed Grade | Clock Rate | | | Data Out Window | Access Window | DQS–DQ Skew |
|---|---|---|---|---|---|---|
| | CL = 2 | CL = 2.5 | CL = 3 | | | |
| -5B | 133 MHz | 167 MHz | 200 MHz | 1.6ns | ±0.70ns | +0.40ns |
| -6T | 133 MHz | 167 MHz | N/A | 2.0ns | ±0.70ns | +0.45ns |
| -75E/-75Z | 133 MHz | 133 MHz | N/A | 2.5ns | ±0.75ns | +0.50ns |
| -75 | 100 MHz | 133 MHz | N/A | 2.5ns | ±0.75ns | +0.50ns |

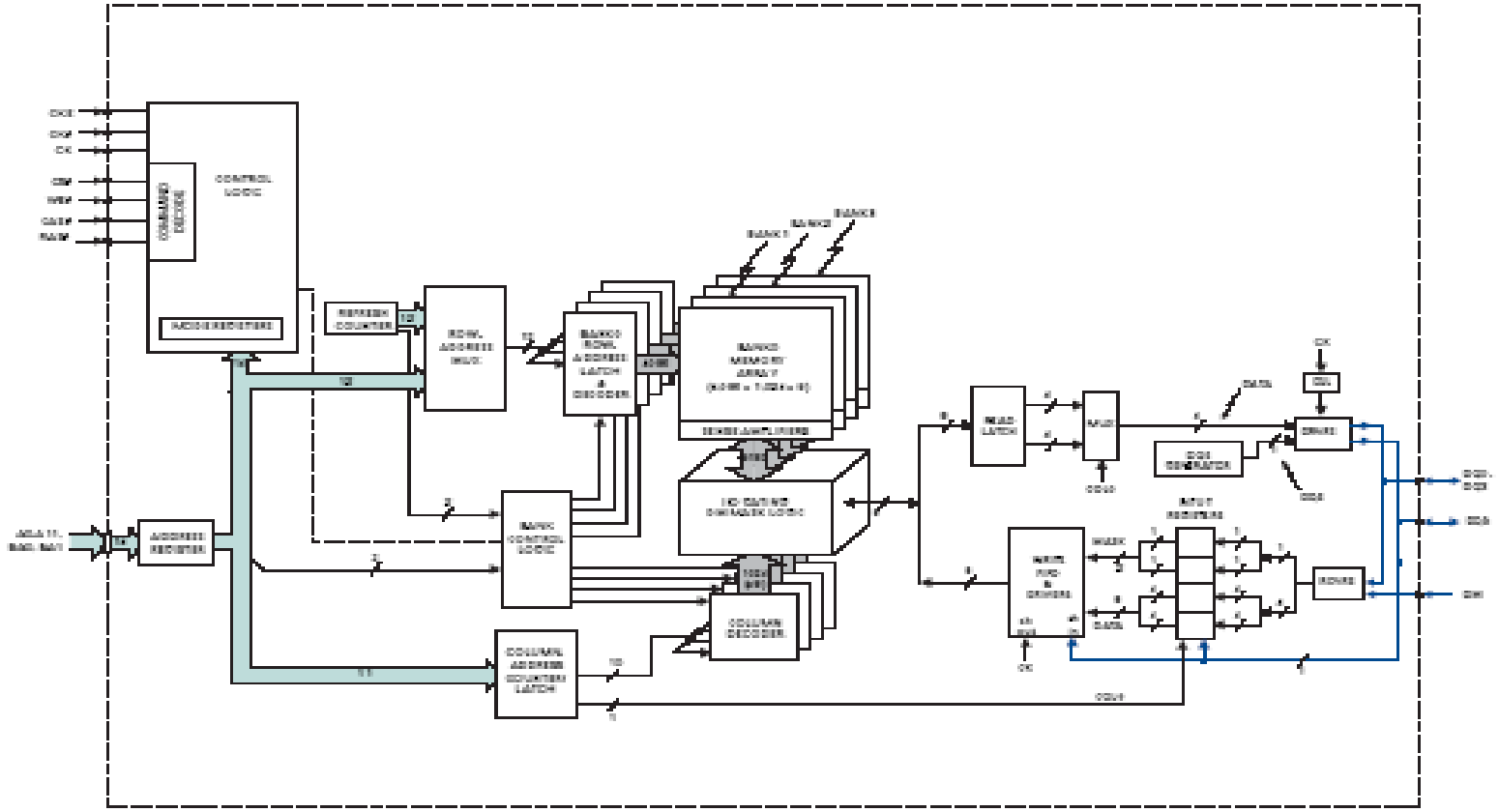
1

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2004 Micron Technology, Inc. All rights reserved.

# DDR SDRAM Model (cont'd)

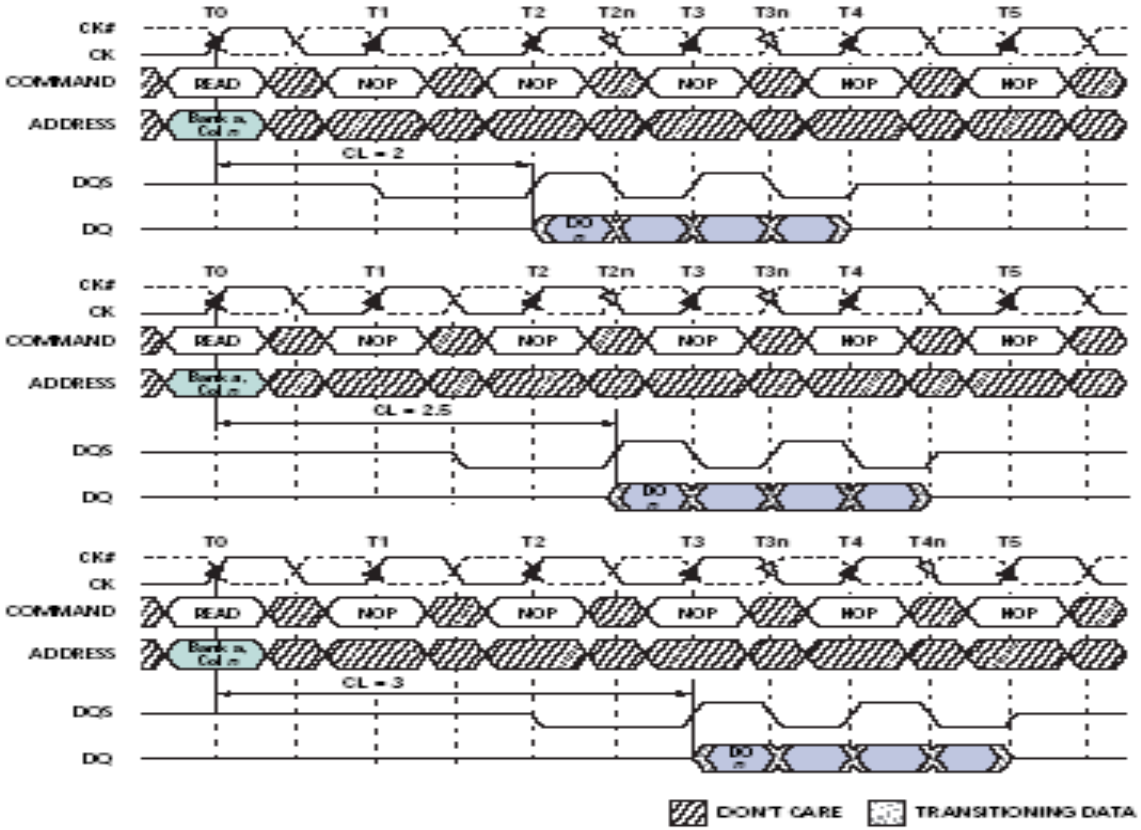
**Functional Block Diagrams**

**Figure 1: 32 Meg x 4**



# DDR SDRAM Model (cont'd)

**Figure 11: READ Burst**




Notes: 1. DO *n* = data-out from column *n*.
2. BL = 4.
3. Three subsequent elements of data-out appear in the programmed order following DO *n*.
4. Shown with nominal $^{t}$AC, $^{t}$DQSCK, and $^{t}$DQSQ.

# DDR SDRAM Model (cont'd) - VHDL

edit - mt46v8m16.vhd

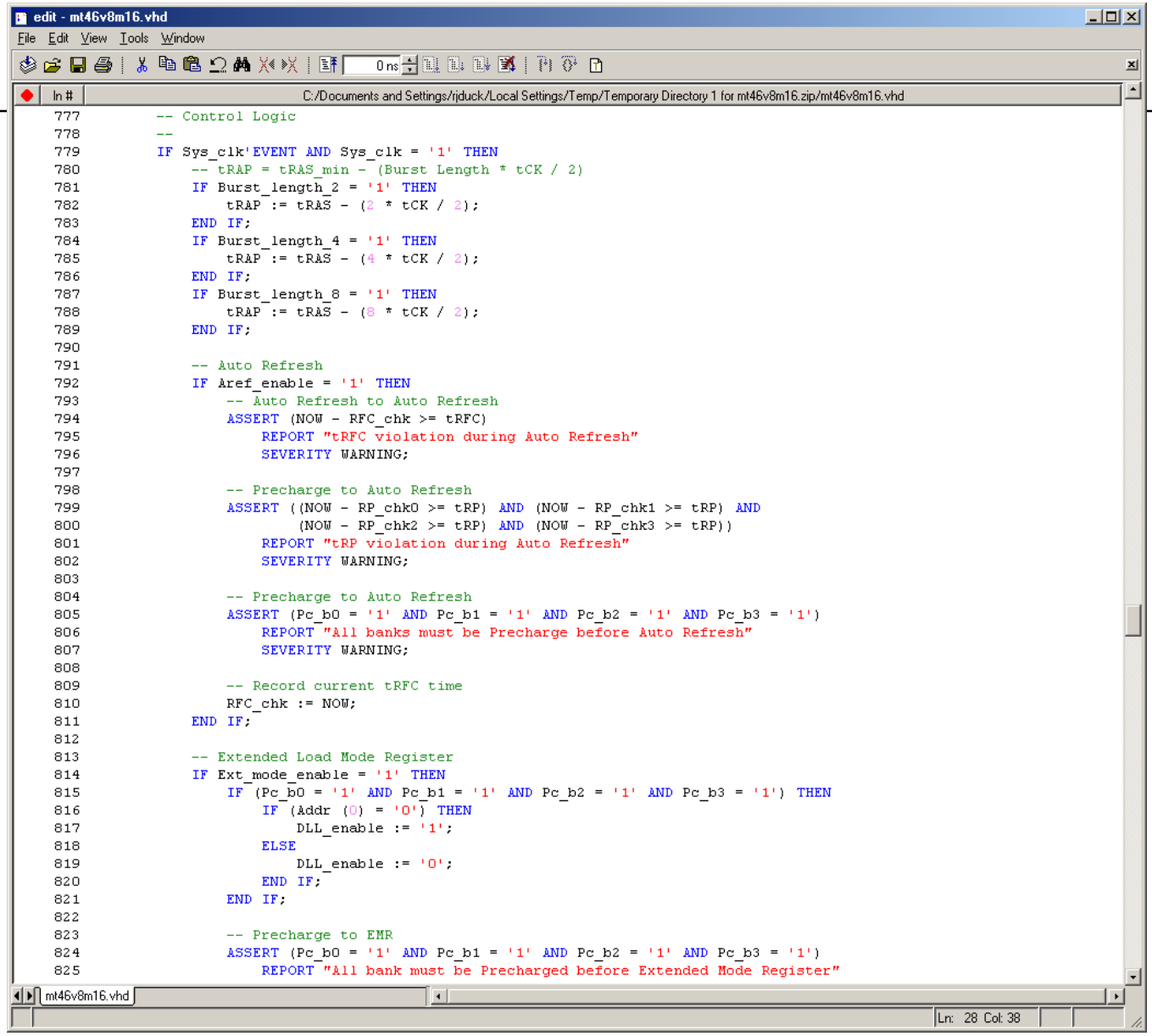
File Edit View Tools Window

C:/Documents and Settings/rjduck/Local Settings/Temp/Temporary Directory 1 for mt46v8m16.zip/mt46v8m16.vhd

```
41 LIBRARY IEEE;
42     USE IEEE.STD_LOGIC_1164.ALL;
43     USE IEEE.STD_LOGIC_UNSIGNED.ALL;
44     USE IEEE.STD_LOGIC_ARITH.ALL;
45
46 ENTITY MT46V8M16 IS
47     GENERIC (                                          -- Timing for -75Z CL2
48         tCK        : TIME    :=  7.500 ns;
49         tCH        : TIME    :=  3.375 ns;          -- 0.45*tCK
50         tCL        : TIME    :=  3.375 ns;          -- 0.45*tCK
51         tDH        : TIME    :=  0.500 ns;
52         tDS        : TIME    :=  0.500 ns;
53         tIH        : TIME    :=  0.900 ns;
54         tIS        : TIME    :=  0.900 ns;
55         tMRD       : TIME    := 15.000 ns;
56         tRAS       : TIME    := 40.000 ns;
57         --tRAP       : TIME    := 20.000 ns;
58         tRC        : TIME    := 65.000 ns;
59         tRFC       : TIME    := 75.000 ns;
60         tRCD       : TIME    := 20.000 ns;
61         tRP        : TIME    := 20.000 ns;
62         tRRD       : TIME    := 15.000 ns;
63         tWR        : TIME    := 15.000 ns;
64         addr_bits  : INTEGER := 12;
65         data_bits  : INTEGER := 16;
66         cols_bits  : INTEGER :=  9
67     );
68     PORT (
69         Dq     : INOUT STD_LOGIC_VECTOR (data_bits - 1 DOWNTO 0) := (OTHERS => 'Z');
70         Dqs    : INOUT STD_LOGIC_VECTOR (1 DOWNTO 0) := "ZZ";
71         Addr   : IN    STD_LOGIC_VECTOR (addr_bits - 1 DOWNTO 0);
72         Ba     : IN    STD_LOGIC_VECTOR (1 DOWNTO 0);
73         Clk    : IN    STD_LOGIC;
74         Clk_n  : IN    STD_LOGIC;
75         Cke    : IN    STD_LOGIC;
76         Cs_n   : IN    STD_LOGIC;
77         Ras_n  : IN    STD_LOGIC;
78         Cas_n  : IN    STD_LOGIC;
79         We_n   : IN    STD_LOGIC;
80         Dm     : IN    STD_LOGIC_VECTOR (1 DOWNTO 0)
81     );
82 END MT46V8M16;
83
84 ARCHITECTURE behave OF MT46V8M16 IS
85     -- Array for Read pipeline
86     TYPE Array_Read_cmnd IS ARRAY (8 DOWNTO 0) OF STD_LOGIC;
87     TYPE Array_Read_bank IS ARRAY (8 DOWNTO 0) OF STD_LOGIC_VECTOR (1 DOWNTO 0);
88     TYPE Array_Read_cols IS ARRAY (8 DOWNTO 0) OF STD_LOGIC_VECTOR (cols_bits - 1 DOWNTO 0);
89
```

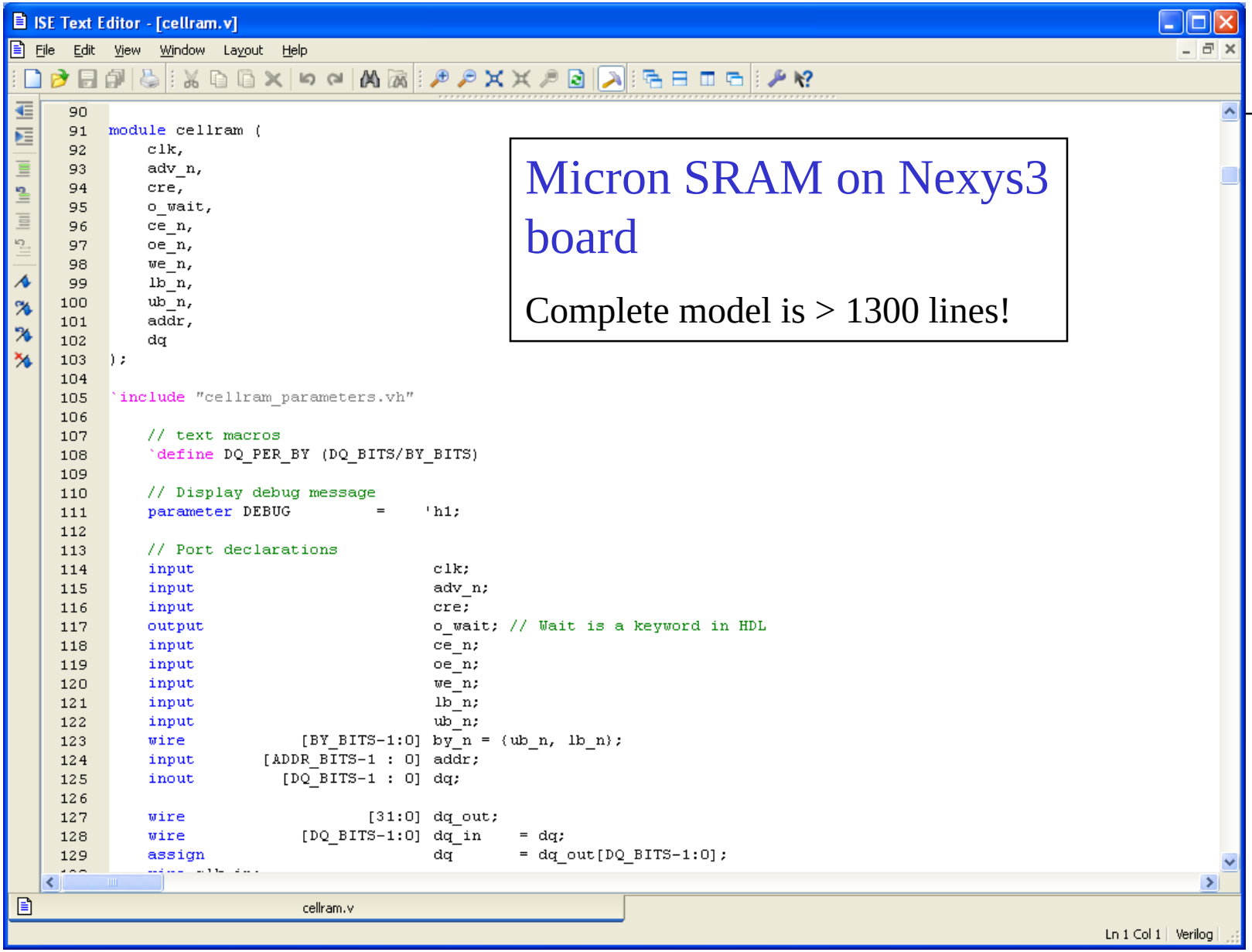
mt46v8m16.vhd

Ln: 28 Col: 38


edit - mt46v8m16.vhd
File Edit View Tools Window
0 ns
ln #
C:/Documents and Settings/rjduck/Local Settings/Temp/Temporary Directory 1 for mt46v8m16.zip/mt46v8m16.vhd
777 -- Control Logic
778 --
779 IF Sys_clk'EVENT AND Sys_clk = '1' THEN
780 -- tRAP = tRAS_min - (Burst Length * tCK / 2)
781 IF Burst_length_2 = '1' THEN
782 tRAP := tRAS - (2 * tCK / 2);
783 END IF;
784 IF Burst_length_4 = '1' THEN
785 tRAP := tRAS - (4 * tCK / 2);
786 END IF;
787 IF Burst_length_8 = '1' THEN
788 tRAP := tRAS - (8 * tCK / 2);
789 END IF;
790
791 -- Auto Refresh
792 IF Aref_enable = '1' THEN
793 -- Auto Refresh to Auto Refresh
794 ASSERT (NOW - RFC_chk >= tRFC)
795 REPORT "tRFC violation during Auto Refresh"
796 SEVERITY WARNING;
797
798 -- Precharge to Auto Refresh
799 ASSERT ((NOW - RP_chk0 >= tRP) AND (NOW - RP_chk1 >= tRP) AND
800 (NOW - RP_chk2 >= tRP) AND (NOW - RP_chk3 >= tRP))
801 REPORT "tRP violation during Auto Refresh"
802 SEVERITY WARNING;
803
804 -- Precharge to Auto Refresh
805 ASSERT (Pc_b0 = '1' AND Pc_b1 = '1' AND Pc_b2 = '1' AND Pc_b3 = '1')
806 REPORT "All banks must be Precharge before Auto Refresh"
807 SEVERITY WARNING;
808
809 -- Record current tRFC time
810 RFC_chk := NOW;
811 END IF;
812
813 -- Extended Load Mode Register
814 IF Ext_mode_enable = '1' THEN
815 IF (Pc_b0 = '1' AND Pc_b1 = '1' AND Pc_b2 = '1' AND Pc_b3 = '1') THEN
816 IF (Addr (0) = '0') THEN
817 DLL_enable := '1';
818 ELSE
819 DLL_enable := '0';
820 END IF;
821 END IF;
822
823 -- Precharge to EMR
824 ASSERT (Pc_b0 = '1' AND Pc_b1 = '1' AND Pc_b2 = '1' AND Pc_b3 = '1')
825 REPORT "All bank must be Precharged before Extended Mode Register"
mt46v8m16.vhd
Ln: 28 Col: 38

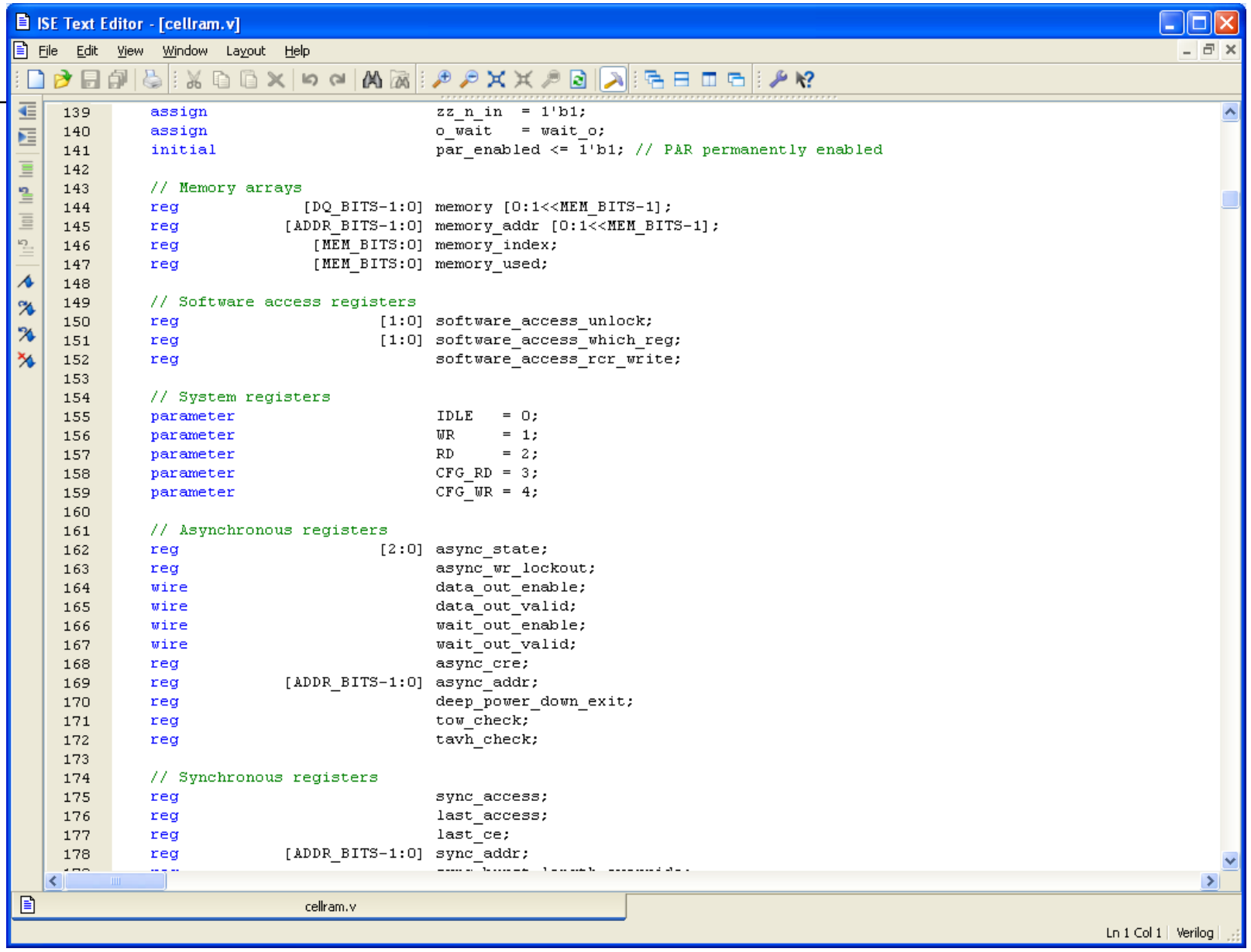
# ISSI SRAM – Verilog Model (partial)

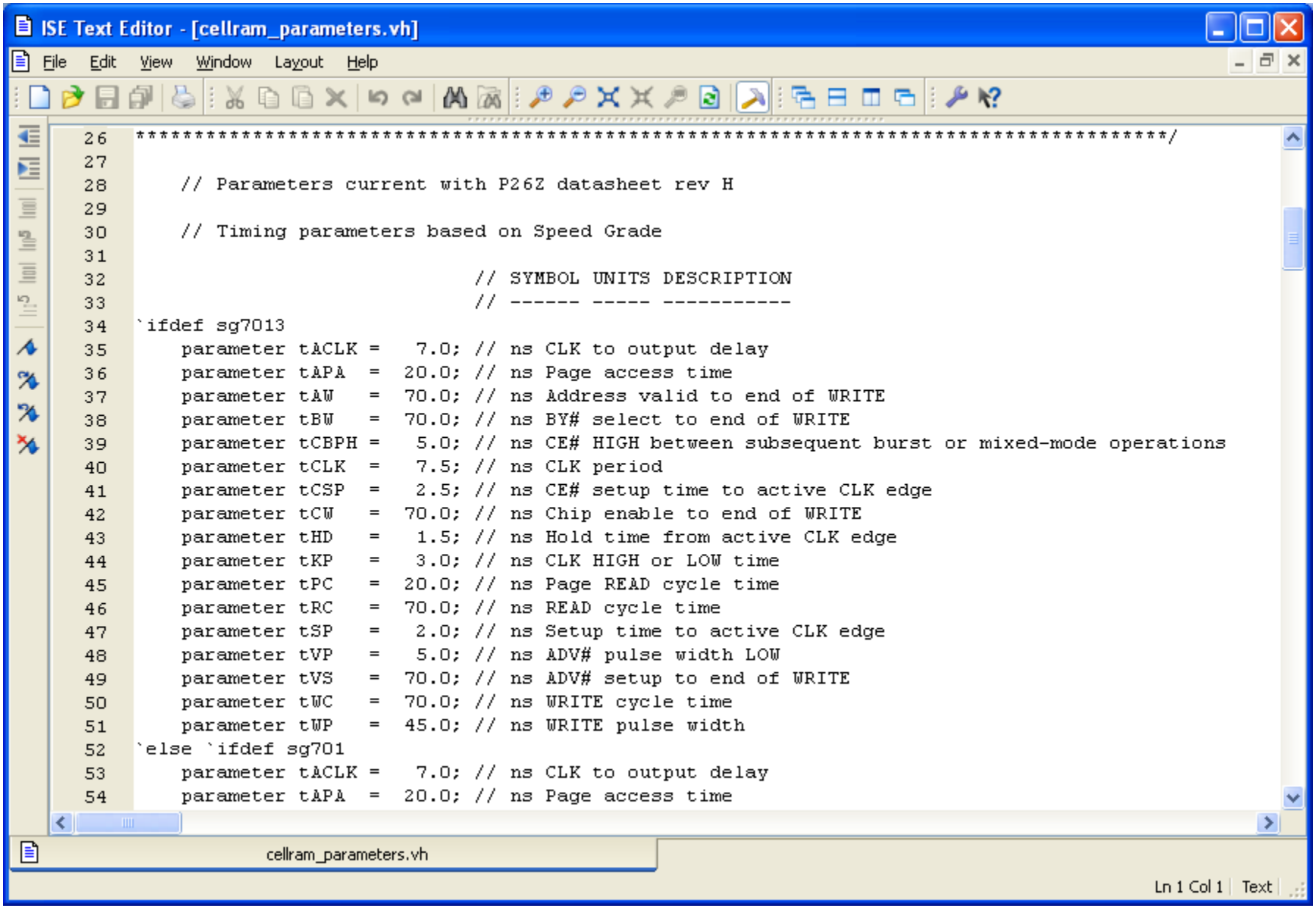
```
// IS61LV25616 Asynchronous SRAM, 256K x 16 = 4M; speed: 10ns.
// Note; 1) Please include "+define+ OEb" in running script if you want to check
//          timing in the case of OE_ being set.
//       2) Please specify access time by defining tAC_10 or tAC_12.

// `define OEb
`define tAC_10
`timescale 1ns/10ps

module IS61LV25616 (A, IO, CE_, OE_, WE_, LB_, UB_);

parameter dqbits = 16;
parameter memdepth = 262143;
parameter addbits = 18;
parameter Toha  = 2;

parameter Tsa   = 2;

`ifdef tAC_10
  parameter Taa   = 10,
            Thzce = 3,
            Thzwe = 5;
`endif

`ifdef tAC_12
  parameter Taa   = 12,
            Thzce = 5,
            Thzwe = 6;
`endif

input CE_, OE_, WE_, LB_, UB_;
input [(addbits - 1) : 0] A;
inout [(dqbits - 1) : 0] IO;

wire [(dqbits - 1) : 0] dout;
reg  [(dqbits/2 - 1) : 0] bank0 [0 : memdepth];
```



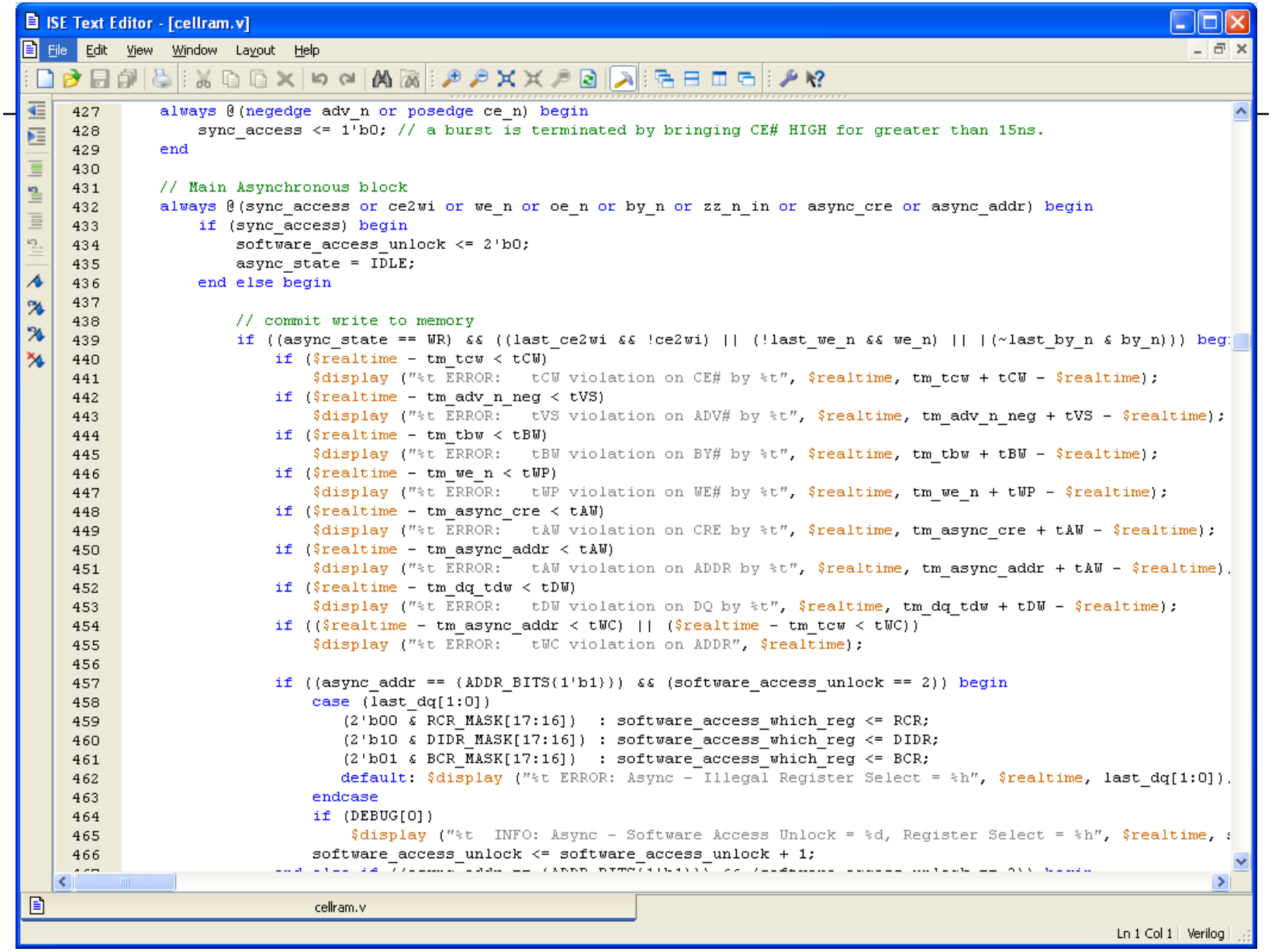

# Micron SRAM on Nexys3 board

Complete model is > 1300 lines!


ISE Text Editor - [cellram.v]
File Edit View Window Layout Help
assign zz_n_in = 1'b1;
assign o_wait = wait_o;
initial par_enabled <= 1'b1; // PAR permanently enabled
// Memory arrays
reg [DQ_BITS-1:0] memory [0:1<<MEM_BITS-1];
reg [ADDR_BITS-1:0] memory_addr [0:1<<MEM_BITS-1];
reg [MEM_BITS:0] memory_index;
reg [MEM_BITS:0] memory_used;
// Software access registers
reg [1:0] software_access_unlock;
reg [1:0] software_access_which_reg;
reg software_access_rcr_write;
// System registers
parameter IDLE = 0;
parameter WR = 1;
parameter RD = 2;
parameter CFG_RD = 3;
parameter CFG_WR = 4;
// Asynchronous registers
reg [2:0] async_state;
reg async_wr_lockout;
wire data_out_enable;
wire data_out_valid;
wire wait_out_enable;
wire wait_out_valid;
reg async_cre;
reg [ADDR_BITS-1:0] async_addr;
reg deep_power_down_exit;
reg tow_check;
reg tavh_check;
// Synchronous registers
reg sync_access;
reg last_access;
reg last_ce;
reg [ADDR_BITS-1:0] sync_addr;
cellram.v
Ln 1 Col 1 Verilog




```
26  ********************************************************************************************/
27
28      // Parameters current with P26Z datasheet rev H
29
30      // Timing parameters based on Speed Grade
31
32                              // SYMBOL UNITS DESCRIPTION
33                              // ------ ----- -----------
34  `ifdef sg7013
35      parameter tACLK =   7.0; // ns CLK to output delay
36      parameter tAPA  =  20.0; // ns Page access time
37      parameter tAW   =  70.0; // ns Address valid to end of WRITE
38      parameter tBW   =  70.0; // ns BY# select to end of WRITE
39      parameter tCBPH =   5.0; // ns CE# HIGH between subsequent burst or mixed-mode operations
40      parameter tCLK  =   7.5; // ns CLK period
41      parameter tCSP  =   2.5; // ns CE# setup time to active CLK edge
42      parameter tCW   =  70.0; // ns Chip enable to end of WRITE
43      parameter tHD   =   1.5; // ns Hold time from active CLK edge
44      parameter tKP   =   3.0; // ns CLK HIGH or LOW time
45      parameter tPC   =  20.0; // ns Page READ cycle time
46      parameter tRC   =  70.0; // ns READ cycle time
47      parameter tSP   =   2.0; // ns Setup time to active CLK edge
48      parameter tVP   =   5.0; // ns ADV# pulse width LOW
49      parameter tVS   =  70.0; // ns ADV# setup to end of WRITE
50      parameter tWC   =  70.0; // ns WRITE cycle time
51      parameter tWP   =  45.0; // ns WRITE pulse width
52  `else `ifdef sg701
53      parameter tACLK =   7.0; // ns CLK to output delay
54      parameter tAPA  =  20.0; // ns Page access time
```


ISE Text Editor - [cellram.v]
File Edit View Window Layout Help
427 always @(negedge adv_n or posedge ce_n) begin
428 sync_access <= 1'b0; // a burst is terminated by bringing CE# HIGH for greater than 15ns.
429 end
430
431 // Main Asynchronous block
432 always @(sync_access or ce2wi or we_n or oe_n or by_n or zz_n_in or async_cre or async_addr) begin
433 if (sync_access) begin
434 software_access_unlock <= 2'b0;
435 async_state = IDLE;
436 end else begin
437
438 // commit write to memory
439 if ((async_state == WR) && ((last_ce2wi && !ce2wi) || (!last_we_n && we_n) || |(~last_by_n & by_n))) beg
440 if ($realtime - tm_tcw < tCW)
441 $display ("%t ERROR: tCW violation on CE# by %t", $realtime, tm_tcw + tCW - $realtime);
442 if ($realtime - tm_adv_n_neg < tVS)
443 $display ("%t ERROR: tVS violation on ADV# by %t", $realtime, tm_adv_n_neg + tVS - $realtime);
444 if ($realtime - tm_tbw < tBW)
445 $display ("%t ERROR: tBW violation on BY# by %t", $realtime, tm_tbw + tBW - $realtime);
446 if ($realtime - tm_we_n < tWP)
447 $display ("%t ERROR: tWP violation on WE# by %t", $realtime, tm_we_n + tWP - $realtime);
448 if ($realtime - tm_async_cre < tAW)
449 $display ("%t ERROR: tAW violation on CRE by %t", $realtime, tm_async_cre + tAW - $realtime);
450 if ($realtime - tm_async_addr < tAW)
451 $display ("%t ERROR: tAW violation on ADDR by %t", $realtime, tm_async_addr + tAW - $realtime).
452 if ($realtime - tm_dq_tdw < tDW)
453 $display ("%t ERROR: tDW violation on DQ by %t", $realtime, tm_dq_tdw + tDW - $realtime);
454 if (($realtime - tm_async_addr < tWC) || ($realtime - tm_tcw < tWC))
455 $display ("%t ERROR: tWC violation on ADDR", $realtime);
456
457 if ((async_addr == {ADDR_BITS{1'b1}}) && (software_access_unlock == 2)) begin
458 case (last_dq[1:0])
459 (2'b00 & RCR_MASK[17:16]) : software_access_which_reg <= RCR;
460 (2'b10 & DIDR_MASK[17:16]) : software_access_which_reg <= DIDR;
461 (2'b01 & BCR_MASK[17:16]) : software_access_which_reg <= BCR;
462 default: $display ("%t ERROR: Async - Illegal Register Select = %h", $realtime, last_dq[1:0]).
463 endcase
464 if (DEBUG[0])
465 $display ("%t INFO: Async - Software Access Unlock = %d, Register Select = %h", $realtime,
466 software_access_unlock <= software_access_unlock + 1;
cellram.v
Ln 1 Col 1 Verilog