

```

// *****
// **
// ** IDT6116SA25.v - IDT6116SA 2K x 8-BIT RAM (-25 SPEED GRADE)
// **
// *****
// **
// **          COPYRIGHT (c) 2003 YOUNG ENGINEERING
// **          ALL RIGHTS RESERVED
// **
// ** THIS PROGRAM IS CONFIDENTIAL AND A TRADE SECRET OF YOUNG ENGINEERING. THE RECEIPT OR
// ** POSSESSION OF THIS PROGRAM DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS
// ** CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN
// ** PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF YOUNG ENGINEERING.
// **
// *****
// ** Revision      : 1.0
// ** Modified Date : 06/05/2003
// ** Revision History:
// **
// ** 06/05/2003: Initial design
// **
// *****
// **          TABLE OF CONTENTS
// **          -----
// **  DECLARATIONS
// **          -----
// **  INITIALIZATION
// **          -----
// **  CORE LOGIC
// **          -----
// **  1.01: Memory Write Logic
// **  1.02: Memory Read Logic
// **  1.03: Memory I/O Buffers
// **          -----
// **  DEBUG LOGIC
// **          -----
// **  2.01: Memory Data Bytes
// **          -----
// **  TIMING CHECKS
// **          -----
// **
// *****

```

```
`timescale 1ns/10ps
```

```
module IDT6116SA25 (IO, A, WE_N, OE_N, CS_N, RESET);
```

```

input  [10:00]    A;                // memory address

input          WE_N;                // memory write enable
input          OE_N;                // memory read enable
input          CS_N;                // memory chip select

input          RESET;               // system reset

inout  [07:00]    IO;               // memory data I/O bus

```

```

// *****
// **  DECLARATIONS
// **          -----
// *****

```

```

wire [07:00]    IO;                // memory data I/O bus

reg  [07:00]    DataDO;              // memory data output
wire          DataOE;               // memory data output enable

reg  [10:00]    A_Dlyd_tAA;          // memory address delayed
reg  [10:00]    A_Dlyd_tOH;          // memory address delayed

wire          OE_N_Dlyd_DO;          // OE_N delayed for output data
wire          CS_N_Dlyd_DO;          // CS_N delayed for output data

```

```

wire          WE_N_Dlyd_OE;          // WE_N delayed for output enable
wire          OE_N_Dlyd_OE;          // OE_N delayed for output enable
wire          CS_N_Dlyd_OE;          // CS_N delayed for output enable

reg  [07:00]   MemoryBlock [0:2047]; // memory block

wire          WriteStrobe_N;         // memory write strobe

integer       tAA;                   // timing parameter
integer       tACS;                   // timing parameter
integer       tCLZ;                   // timing parameter
integer       tOLZ;                   // timing parameter
integer       tCHZ;                   // timing parameter
integer       tOHZ;                   // timing parameter
integer       tOE;                    // timing parameter
integer       tOH;                    // timing parameter
integer       tWHZ;                   // timing parameter

// *****
// **   INITIALIZATION   **
// *****

initial begin
    tAA = 25;                          // address to data access time
    tACS = 25;                          // chip select to data access time
    tCLZ = 5;                            // CS_N to data low-Z delay
    tOLZ = 5;                            // OE_N to data low-Z delay
    tCHZ = 12;                           // CS_N to data high-Z delay
    tOHZ = 10;                           // OE_N to data high-Z delay
    tWHZ = 16;                           // WE_N to data high-Z delay
    tOE = 13;                             // OE_N to data valid delay
    tOH = 5;                              // output hold from address change
end

// *****
// **   CORE LOGIC   **
// *****
// -----
//      1.01:  Memory Write Logic
// -----

assign WriteStrobe_N = WE_N | CS_N;

always @(posedge WriteStrobe_N) begin
    MemoryBlock[A] = IO[07:00];
    DataDO          = IO[07:00];
end

// -----
//      1.02:  Memory Read Logic
// -----

always @(A_Dlyd_tAA or A_Dlyd_tOH or OE_N_Dlyd_DO or CS_N_Dlyd_DO) begin
    if (A_Dlyd_tAA != A_Dlyd_tOH)      DataDO = 8'hXX;
    else if ((OE_N_Dlyd_DO==0) & (CS_N_Dlyd_DO==0)) DataDO = MemoryBlock[A_Dlyd_tAA];
    else                                DataDO = 8'hXX;
end

always @(A) A_Dlyd_tAA <= #(tAA) A[10:00];
always @(A) A_Dlyd_tOH <= #(tOH) A[10:00];

assign #(tOHZ,tOE)  OE_N_Dlyd_DO = OE_N;
assign #(tCHZ,tACS) CS_N_Dlyd_DO = CS_N;

assign #(0,tWHZ)    WE_N_Dlyd_OE = WE_N;
assign #(tOHZ,tOLZ) OE_N_Dlyd_OE = OE_N;
assign #(tCHZ,tCLZ) CS_N_Dlyd_OE = CS_N;

assign DataOE = (OE_N_Dlyd_OE == 0) & (WE_N_Dlyd_OE == 1) & (CS_N_Dlyd_OE == 0);

// -----
//      1.03:  Memory I/O Buffers
// -----

bufif1 (IO[00], DataDO[00], DataOE);
bufif1 (IO[01], DataDO[01], DataOE);
bufif1 (IO[02], DataDO[02], DataOE);

```

```

bufif1 (IO[03], DataDO[03], DataOE);
bufif1 (IO[04], DataDO[04], DataOE);
bufif1 (IO[05], DataDO[05], DataOE);
bufif1 (IO[06], DataDO[06], DataOE);
bufif1 (IO[07], DataDO[07], DataOE);

// *****
// **   DEBUG LOGIC   **
// *****
// -----
//      2.01:  Memory Data Bytes
// -----

wire [07:00] MemoryByte00 = MemoryBlock[00];
wire [07:00] MemoryByte01 = MemoryBlock[01];
wire [07:00] MemoryByte02 = MemoryBlock[02];
wire [07:00] MemoryByte03 = MemoryBlock[03];
wire [07:00] MemoryByte04 = MemoryBlock[04];
wire [07:00] MemoryByte05 = MemoryBlock[05];
wire [07:00] MemoryByte06 = MemoryBlock[06];
wire [07:00] MemoryByte07 = MemoryBlock[07];

wire [07:00] MemoryByte08 = MemoryBlock[08];
wire [07:00] MemoryByte09 = MemoryBlock[09];
wire [07:00] MemoryByte0A = MemoryBlock[10];
wire [07:00] MemoryByte0B = MemoryBlock[11];
wire [07:00] MemoryByte0C = MemoryBlock[12];
wire [07:00] MemoryByte0D = MemoryBlock[13];
wire [07:00] MemoryByte0E = MemoryBlock[14];
wire [07:00] MemoryByte0F = MemoryBlock[15];

// *****
// **   TIMING CHECKS   **
// *****

wire TimingCheckEnable1 = (WE_N == 0) & !RESET;

specify
  specparam
    tCW = 17,           // CS_N to end of write
    tAW = 17,           // A valid to end of write
    tAS = 0,            // A write setup time
    tWP = 15,           // write pulse width
    tWR = 0,            // write recovery time
    tDW = 13,           // write data setup time
    tDH = 0;            // write data hold time

  $width (negedge CS_N &&& TimingCheckEnable1, tCW);
  $width (negedge WE_N, tWP);

  $setup (A[00], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[01], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[02], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[03], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[04], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[05], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[06], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[07], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[08], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[09], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[10], posedge WriteStrobe_N &&& (RESET==0), tAW);
  $setup (A[00], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[01], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[02], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[03], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[04], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[05], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[06], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[07], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[08], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[09], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (A[10], negedge WriteStrobe_N &&& (RESET==0), tAS);
  $setup (IO[00], posedge WriteStrobe_N &&& (RESET==0), tDW);
  $setup (IO[01], posedge WriteStrobe_N &&& (RESET==0), tDW);
  $setup (IO[02], posedge WriteStrobe_N &&& (RESET==0), tDW);
  $setup (IO[03], posedge WriteStrobe_N &&& (RESET==0), tDW);
  $setup (IO[04], posedge WriteStrobe_N &&& (RESET==0), tDW);

```

```
$setup (IO[05], posedge WriteStrobe_N &&& (RESET==0), tDW);
$setup (IO[06], posedge WriteStrobe_N &&& (RESET==0), tDW);
$setup (IO[07], posedge WriteStrobe_N &&& (RESET==0), tDW);
```

```
$hold (posedge WriteStrobe_N &&& (RESET==0), A[00], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[01], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[02], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[03], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[04], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[05], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[06], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[07], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[08], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[09], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), A[10], tWR);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[00], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[01], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[02], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[03], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[04], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[05], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[06], tDH);
$hold (posedge WriteStrobe_N &&& (RESET==0), IO[07], tDH);
```

```
endspecify
```

```
endmodule
```