

```

// *****
// **
// ** LTC1606.v - LTC1606 16-BIT PARALLEL ADC (LTC1606/LTC1606A GRADES)
// **
// *****
// **
// **          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 : 05/01/2003
// ** Revision History:
// **
// ** 05/01/2003: Initial design
// **
// *****
// **          TABLE OF CONTENTS
// **
// **-----**
// **  DECLARATIONS
// **-----**
// **-----**
// **  INITIALIZATION
// **-----**
// **-----**
// **  CORE LOGIC
// **-----**
// **-----**
// **  TIMING CHECKS
// **-----**
// **
// *****

```

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

```
module LTC1606 (DOUT, BUSY_N, VIN, R_C, CS_N, BYTE, RESET);
```

```

input  [15:00]    VIN;           // analog voltage in
input           R_C;           // read or convert
input          CS_N;          // chip select
input          BYTE;         // data byte select
input          RESET;        // system reset
output [15:00]    DOUT;        // parallel data out
output          BUSY_N;       // conversion status

```

```

// *****
// **  DECLARATIONS
// *****

```

```

reg  [15:00]    VIN_Hold;       // sampled input data
wire [15:00]    DOUT;          // data output bus
reg           BUSY_N;         // conversion status
wire         BYTE_Dlyd;       // byte select delayed
reg  [15:00]    Data;          // data output bus
wire [15:00]    DataDO;        // data output bus
wire           DataOE;        // data output enable
wire           ConvTrigger;    // conversion trigger

integer        t2;           // timing parameter
integer        t3;           // timing parameter
integer        t4;           // timing parameter

```

```

integer          t7;                // timing parameter
integer          t9;                // timing parameter
integer          t12;               // timing parameter

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

initial begin
    VIN_Hold = 16'hXXXX;
end

initial BUSY_N = 1'b1;

initial begin
    t2 = 2500;                // new data valid after R_C low
    t3 = 65;                 // BUSY_N delay from R_C low
    t4 = 2500;               // BUSY_N low time
    t7 = 15;                 // data output float delay
    t9 = 2000;               // previous data valid after R_C
    t12 = 60;                // data access time
end

// *****
// **  CORE LOGIC  **
// *****

assign ConvTrigger = !R_C & !CS_N;

always @(posedge ConvTrigger) begin
    if (BUSY_N == 1) begin
        VIN_Hold = VIN[15:00];
    end
end

always @(posedge ConvTrigger) begin
    if (BUSY_N == 1) begin
        #(t3)          BUSY_N = 1'b0;
        #(t4)          BUSY_N = 1'b1;
    end
end

always @(posedge ConvTrigger) begin
    if (BUSY_N == 1) begin
        #(t9)          Data = 16'hXXXX;
        #(t2-t9)       Data = VIN_Hold[15:00];
    end
end

assign #(t12) BYTE_Dlyd = BYTE;
assign DataDO = BYTE_Dlyd ? {Data[07:00],Data[15:08]} : Data[15:00];

assign #(t12,t7) DataOE = R_C & !CS_N;

bufif1 (DOUT[00], DataDO[00], DataOE);
bufif1 (DOUT[01], DataDO[01], DataOE);
bufif1 (DOUT[02], DataDO[02], DataOE);
bufif1 (DOUT[03], DataDO[03], DataOE);
bufif1 (DOUT[04], DataDO[04], DataOE);
bufif1 (DOUT[05], DataDO[05], DataOE);
bufif1 (DOUT[06], DataDO[06], DataOE);
bufif1 (DOUT[07], DataDO[07], DataOE);
bufif1 (DOUT[08], DataDO[08], DataOE);
bufif1 (DOUT[09], DataDO[09], DataOE);
bufif1 (DOUT[10], DataDO[10], DataOE);
bufif1 (DOUT[11], DataDO[11], DataOE);
bufif1 (DOUT[12], DataDO[12], DataOE);
bufif1 (DOUT[13], DataDO[13], DataOE);
bufif1 (DOUT[14], DataDO[14], DataOE);
bufif1 (DOUT[15], DataDO[15], DataOE);

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

specify

```

```
specparam
    t1 = 40, // R_C/CS_N pulse width - low
    t10 = 5; // R_C to CS_N setup/hold time

$width (negedge R_C, t1);
$width (negedge CS_N, t1);

$setup (R_C, negedge CS_N &&& (RESET==0), t10);
$hold (posedge CS_N &&& (RESET==0), R_C, t10);
endspecify

endmodule
```