```
// ****************************************************************************************************
// **                                                                                                **
// **   AD1674.v - AD1674 12-BIT PARALLEL ADC (J/K/A/B GRADES)                                       **
// **                                                                                                **
// ****************************************************************************************************
// **                                                                                                **
// **                              COPYRIGHT (c) 2000 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  : 02/28/2000                                                                  **
// **   Revision History:                                                                            **
// **                                                                                                **
// **   02/28/2000:  Initial design                                                                  **
// **                                                                                                **
// ****************************************************************************************************
// **                                       TABLE OF CONTENTS                                        **
// ****************************************************************************************************
// **--------------------------------------------------------------------------------------------------**
// **   DECLARATIONS                                                                                 **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   INITIALIZATION                                                                               **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   CORE LOGIC                                                                                   **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   TIMING CHECKS                                                                                **
// **--------------------------------------------------------------------------------------------------**
// **                                                                                                **
// ****************************************************************************************************


`timescale 1ns/10ps

module AD1674 (DOUT, STS, VIN, R_C, A0, CE, CS_N, B12_8, RESET);

   input  [11:00]       VIN;                            // analog voltage in

   input                R_C;                            // read or convert

   input                A0;                             // byte address
   input                CE;                             // chip enable
   input                CS_N;                           // chip select
   input                B12_8;                          // 12-bit/8-bit mode

   input                RESET;                          // system reset

   output [11:00]       DOUT;                           // parallel data out

   output               STS;                            // conversion status


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

   reg  [11:00]         VIN_Hold;                       // sampled input data

   wire [11:00]         DOUT;                           // data output bus

   reg                  STS;                            // conversion status

   reg  [11:00]         Data;                           // data output bus
   reg  [11:00]         DataDO;                         // data output bus
   wire                 DataOE;                         // data output enable

   wire                 DataOE1;                        // data output enable

   reg                  DataOE_L;                       // data output enable
   reg                  DataOE_M;                       // data output enable
   reg                  DataOE_H;                       // data output enable
```

```
   reg                 Conv12Bit;                  // conversion type
   reg                 ConvRunning;                // conversion running
   wire                ConvTrigger;                // conversion trigger

   integer             tC;                         // timing parameter
   integer             tDS;                        // timing parameter
   integer             tHS;                        // timing parameter
   integer             tHL;                        // timing parameter
   integer             tDD;                        // timing parameter

   integer             tHDR;                       // timing parameter


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

   initial begin
      VIN_Hold = 12'hXXXX;
   end

   initial STS = 1'b0;

   initial begin
      Conv12Bit   = 1'b0;
      ConvRunning = 1'b0;
   end

   initial begin
      tDS = 200;                                   // STS delay from R_C
      tHS = 600;                                   // STS delay after data valid
      tHL = 150;                                   // data output float delay
      tDD = 150;                                   // data access time

      tHDR = 25;                                   // data valid after R_C low
   end


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

   assign ConvTrigger = !R_C & CE & !CS_N;

   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         if (A0 == 0)   VIN_Hold = VIN[11:00];
         else           VIN_Hold = {4'h0,VIN[07:00]};
      end
   end

   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         if (A0 == 0)   Conv12Bit = #(tDS) 1'b1;
         else           Conv12Bit = #(tDS) 1'b0;
      end
   end

   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         #(tHL)         ConvRunning = 1'b1;
         #(tC-tHL-tHS)  ConvRunning = 1'b0;
      end
   end

   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         if (A0 == 0)   tC = 10000;
         else           tC =  8000;
      end
   end

   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         #(tDS)         STS = 1'b1;
         #(tC-tDS)      STS = 1'b0;
      end
   end
```

```
   always @(posedge ConvTrigger) begin
      if (STS == 0) begin
         fork
            #(tHDR)     Data = 12'hXXX;
            #(tC-tHS)   Data = VIN_Hold[11:00];
         join
      end
   end

   always @(B12_8 or A0 or Data) begin
      casex ({B12_8,A0})
         3'b00 : DataDO = {Data[11:04],4'h0};
         3'b01 : DataDO = {8'h00,Data[03:00]};

         3'b1x : DataDO = {Data[11:00]};
      endcase
   end

   assign #(tDD) DataOE1 = R_C & CE & !CS_N;
   assign DataOE = DataOE1 & !ConvRunning;

   always @(B12_8 or A0 or DataOE) begin
      casex ({B12_8,A0})
         2'b00 : begin DataOE_H = DataOE; DataOE_M = DataOE; DataOE_L = 1'b0;   end
         2'b01 : begin DataOE_H = 1'b0;   DataOE_M = DataOE; DataOE_L = DataOE; end

         2'b1x : begin DataOE_H = DataOE; DataOE_M = DataOE; DataOE_L = DataOE; end
      endcase
   end

   bufif1 (DOUT[00], DataDO[00], DataOE_L);
   bufif1 (DOUT[01], DataDO[01], DataOE_L);
   bufif1 (DOUT[02], DataDO[02], DataOE_L);
   bufif1 (DOUT[03], DataDO[03], DataOE_L);
   bufif1 (DOUT[04], DataDO[04], DataOE_M);
   bufif1 (DOUT[05], DataDO[05], DataOE_M);
   bufif1 (DOUT[06], DataDO[06], DataOE_M);
   bufif1 (DOUT[07], DataDO[07], DataOE_M);
   bufif1 (DOUT[08], DataDO[08], DataOE_H);
   bufif1 (DOUT[09], DataDO[09], DataOE_H);
   bufif1 (DOUT[10], DataDO[10], DataOE_H);
   bufif1 (DOUT[11], DataDO[11], DataOE_H);


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

   wire TimingCheckEnable1 = !RESET & (R_C == 0);
   wire TimingCheckEnable2 = !RESET &              (CS_N == 0);
   wire TimingCheckEnable3 = !RESET & (R_C == 0) & (CS_N == 0);
   wire TimingCheckEnable4 = !RESET & (R_C == 1) & (CS_N == 0);

   specify
      specparam
         tHEC = 50,                                    // CE pulse width - high
         tHRL = 50,                                    // R_C pulse width - low
         tHRH = 150,                                   // R_C pulse width - high
         tSSC = 50,                                    // CS_N to CE setup time
         tHSC = 50,                                    // CS_N to CE hold time
         tSRC = 50,                                    // R_C to CE setup time
         tHRC = 50,                                    // R_C to CE hold time
         tHAC = 50,                                    // A0 to CE hold time
         tSSR = 50,                                    // CS_N to CE setup time
         tSAR = 50,                                    // A0 to CE setup time
         tHAR = 50;                                    // A0 to CE hold time

      $width (posedge CE,  tHEC);
      $width (negedge R_C, tHRL);
      $width (posedge R_C, tHRH);

      $setup (CS_N, posedge CE &&& (TimingCheckEnable1==1), tSSC);
      $setup (R_C,  posedge CE &&& (TimingCheckEnable2==1), tSRC);
      $setup (CS_N, posedge CE &&& (TimingCheckEnable4==1), tSSR);
      $setup (A0,   posedge CE &&& (TimingCheckEnable4==1), tSAR);

      $hold  (posedge CE &&& (TimingCheckEnable1==1), CS_N, tHSC);
      $hold  (posedge CE &&& (TimingCheckEnable2==1), R_C,  tHRC);
```

```
      $hold  (posedge CE &&& (TimingCheckEnable3==1), A0,   tHAC);
      $hold  (negedge CE &&& (TimingCheckEnable4==1), A0,   tHAR);
   endspecify

endmodule
```