```
// ********************************************************************************************************
// **                                                                                                    **
// **   LTC1609.v - LTC1609 16-BIT SERIAL ADC (LTC1609/LTC1609A 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 - External Data Clock Read During Conversion is NOT Supported.       **
// **                                                                                                    **
// ********************************************************************************************************
// **                                       TABLE OF CONTENTS                                            **
// ********************************************************************************************************
// **--------------------------------------------------------------------------------------------------**
// **   DECLARATIONS                                                                                     **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   INITIALIZATION                                                                                   **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   CORE LOGIC                                                                                       **
// **--------------------------------------------------------------------------------------------------**
// **--------------------------------------------------------------------------------------------------**
// **   TIMING CHECKS                                                                                    **
// **--------------------------------------------------------------------------------------------------**
// **                                                                                                    **
// ********************************************************************************************************


`timescale 1ns/10ps

module LTC1609 (DATA, SYNC, DATACLK, BUSY_N, VIN, R_C, TAG, CS_N, SB_BTC, EXT_INT, RESET);

   input  [15:00]       VIN;                            // analog voltage in

   input                R_C;                            // read or convert

   input                TAG;                            // external serial data input

   input                CS_N;                           // chip select

   input                SB_BTC;                         // binary/two's complement select
   input                EXT_INT;                        // external/internal clock select

   input                RESET;                          // system reset

   inout                DATACLK;                        // serial data clock input/output

   output               DATA;                           // serial data out
   output               SYNC;                           // external clock sync pulse

   output               BUSY_N;                         // conversion status


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

   reg  [15:00]         VIN_Hold;                       // sampled input data - current
   reg  [15:00]         VIN_Last;                       // sampled input data - previous

   reg  [15:00]         DataShifter;                    // data output shift register
   reg                  ShiftEnable;                    // data shift enable

   reg                  TAG_Hold;                       // sampled TAG input

   reg                  DATA;                           // serial output data
```

```
   reg                 SYNC;                           // external clock sync output
   reg                 BUSY_N;                         // conversion status

   reg                 DATACLK_DO;                     // data clock output
   wire                DATACLK_OE;                     // data clock output enable

   wire                ConvTrigger;                    // conversion trigger

   reg                 SyncEnable;                     // sync pulse generate enable

   integer             t2;                             // timing parameter
   integer             t3;                             // timing parameter
   integer             t8;                             // timing parameter
   integer             t9;                             // timing parameter
   integer             t10;                            // timing parameter
   integer             t11;                            // timing parameter
   integer             t17;                            // timing parameter
   integer             t18;                            // timing parameter


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

   initial begin
      VIN_Hold = 16'h0000;
      VIN_Last = 16'h0000;

      DataShifter = 16'h0000;
      ShiftEnable = 1'b0;
   end

   initial begin
      SyncEnable = 1'b0;
   end

   initial begin
      DATA   = 1'b0;
      SYNC   = 1'b0;
      BUSY_N = 1'b1;
   end

   initial begin
      DATACLK_DO = 1'b0;
   end

   always @(EXT_INT) begin
      SyncEnable  = 1'b0;
      ShiftEnable = 1'b0;
   end

   initial begin
      t2  = 80;                                        // R_C/CS_N to BUSY_N delay
      t3  = 3000;                                      // BUSY_N low time
      t8  = 260;                                       // R_C low to DATACLK delay
      t9  = 150;                                       // internal DATACLK period
      t10 = 15;                                        // DATA valid setup time
      t11 = 40;                                        // DATA valid hold time
      t17 = 50;                                        // external DATACLK to SYNC delay
      t18 = 50;                                        // external DATACLK to DATA delay
   end


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

   assign ConvTrigger = !R_C & !CS_N;

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

   always @(posedge ConvTrigger) begin
      if (BUSY_N == 1) begin
         #(t2)          BUSY_N = 1'b0;
```

```
      #(t3)          BUSY_N = 1'b1;
   end
end

always @(posedge ConvTrigger) begin
   if (BUSY_N == 1) begin
      TAG_Hold = TAG;
   end
end

always @(posedge ConvTrigger) begin
   if (BUSY_N == 1) begin
      if (EXT_INT == 0) begin
         #(t8-t10)   DATA = VIN_Last[15];
         #(t9)       DATA = VIN_Last[14];
         #(t9)       DATA = VIN_Last[13];
         #(t9)       DATA = VIN_Last[12];
         #(t9)       DATA = VIN_Last[11];
         #(t9)       DATA = VIN_Last[10];
         #(t9)       DATA = VIN_Last[09];
         #(t9)       DATA = VIN_Last[08];
         #(t9)       DATA = VIN_Last[07];
         #(t9)       DATA = VIN_Last[06];
         #(t9)       DATA = VIN_Last[05];
         #(t9)       DATA = VIN_Last[04];
         #(t9)       DATA = VIN_Last[03];
         #(t9)       DATA = VIN_Last[02];
         #(t9)       DATA = VIN_Last[01];
         #(t9)       DATA = VIN_Last[00];
         #(t9)       DATA = TAG_Hold;
      end
   end
end

always @(posedge ConvTrigger) begin
   if (BUSY_N == 1) begin
      SyncEnable = 1'b0;
      #(t2+t3);
      SyncEnable = 1'b1;
   end
end

always @(posedge ConvTrigger) begin
   if (BUSY_N == 1) begin
      ShiftEnable = 1'b0;
   end
end

always @(negedge BUSY_N) begin
   DataShifter = VIN_Last[15:00];
end

always @(posedge BUSY_N) begin
   DataShifter = VIN_Hold[15:00];
end

always @(posedge DATACLK) begin
   if ((R_C == 1) & (CS_N == 0) & (EXT_INT == 1)) begin
      if (SyncEnable) begin
         SYNC <= #(t17) 1'b1;

         SyncEnable  <= 1'b0;
         ShiftEnable <= 1'b1;
      end
      else begin
         SYNC <= #(t17) 1'b0;
      end
   end
end

always @(posedge DATACLK) begin
   if ((CS_N == 0) & (EXT_INT == 1)) begin
      if (ShiftEnable) begin
         DataShifter[15:01] <= DataShifter[14:00];
         DataShifter[00]    <= TAG;

         DATA <= #(t18) DataShifter[15];
      end
   end
```

```verilog
   end

   always @(posedge ConvTrigger) begin
      if (BUSY_N == 1) begin
         #(t8)          DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
         #(t9/2)        DATACLK_DO = 1'b1;
         #(t9/2)        DATACLK_DO = 1'b0;
      end
   end

   assign DATACLK_OE = (EXT_INT == 0);

   bufif1 (DATACLK, DATACLK_DO, DATACLK_OE);


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

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

   specify
      specparam
         t1  = 40,                                     // R_C/CS_N pulse width - low
         t12 = 50,                                     // DATACLK period
         t13 = 20,                                     // DATACLK pulse width - high
         t14 = 20,                                     // DATACLK pulse width - low
         t15 = 15,                                     // R_C/CS_N to DATACLK setup time
         t16 = 10,                                     // R_C to CS_N setup time
         t19 = 10,                                     // CS_N to DATACLK setup time
         t24 = 10;                                     // TAG to DATACLK hold time

      $width (negedge R_C,  t1);
      $width (negedge CS_N, t1);
      $width (posedge DATACLK, t13);
      $width (negedge DATACLK, t14);

      $setup (R_C,  negedge CS_N &&& (RESET==0), t16);
      $setup (R_C,  posedge DATACLK &&& (TimingCheckEnable1==1), t15);
      $setup (CS_N, posedge DATACLK &&& (TimingCheckEnable2==1), t15);
      $setup (CS_N, posedge DATACLK &&& (TimingCheckEnable3==1), t19);

      $hold  (posedge DATACLK &&& (RESET==0), TAG, t24);

      $period (posedge DATACLK &&& (RESET==0), t12);
   endspecify

endmodule
```