

```

// *****
// **
// ** DAC8412.v - DAC8412 12-BIT PARALLEL DAC (VDD = +5.0V)
// **
// *****
// **
// **          COPYRIGHT (c) 2002 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 : 01/15/2002
// ** Revision History:
// **
// ** 01/15/2002: Initial design
// **
// *****
// **          TABLE OF CONTENTS
// **          -----
// **  DECLARATIONS
// **          -----
// **  INITIALIZATION
// **          -----
// **  CORE LOGIC
// **          -----
// **  TIMING CHECKS
// **          -----
// **
// *****

```

`timescale 1ns/10ps

```

module DAC8412      ( VOUTA,
                    VOUTB,
                    VOUTC,
                    VOUTD,

                    DB00,
                    DB01,
                    DB02,
                    DB03,
                    DB04,
                    DB05,
                    DB06,
                    DB07,
                    DB08,
                    DB09,
                    DB10,
                    DB11,

                    A0,
                    A1,

                    R_W,
                    CS_N,

                    LDAC_N,
                    RESET_N,

                    RESET
                );

input  A0;           // input register address
input  A1;           // input register address

input  R_W;         // read/write enable
input  CS_N;        // chip select

input  LDAC_N;      // DAC update strobe

```

```

input          RESET_N;                // DAC register reset

input          RESET;                  // system reset

inout         DB00;                    // parallel data I/O bus
inout         DB01;                    // parallel data I/O bus
inout         DB02;                    // parallel data I/O bus
inout         DB03;                    // parallel data I/O bus
inout         DB04;                    // parallel data I/O bus
inout         DB05;                    // parallel data I/O bus
inout         DB06;                    // parallel data I/O bus
inout         DB07;                    // parallel data I/O bus
inout         DB08;                    // parallel data I/O bus
inout         DB09;                    // parallel data I/O bus
inout         DB10;                    // parallel data I/O bus
inout         DB11;                    // parallel data I/O bus

output [11:00] VOUTA;                  // DAC A voltage output
output [11:00] VOUTB;                  // DAC B voltage output
output [11:00] VOUTC;                  // DAC B voltage output
output [11:00] VOUTD;                  // DAC B voltage output

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

reg [11:00]    DACA_InputLatch;        // DAC A input data latch
reg [11:00]    DACB_InputLatch;        // DAC B input data latch
reg [11:00]    DACC_InputLatch;        // DAC C input data latch
reg [11:00]    DACD_InputLatch;        // DAC D input data latch

reg [11:00]    VOUTA;                  // DAC A output voltage
reg [11:00]    VOUTB;                  // DAC B output voltage
reg [11:00]    VOUTC;                  // DAC C output voltage
reg [11:00]    VOUTD;                  // DAC D output voltage

wire [11:00]   DataDI;                 // parallel data bus - input
reg [11:00]   DataDO;                 // parallel data bus - output
wire          DataOE;                 // parallel data bus - output enable

integer       tDZ;                    // timing parameter
integer       tCSD;                   // timing parameter

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

initial begin
    DACA_InputLatch= 12'hXXX;
    DACB_InputLatch= 12'hXXX;
    DACC_InputLatch= 12'hXXX;
    DACD_InputLatch= 12'hXXX;
end

initial begin
    VOUTA = 12'hXXX;
    VOUTB = 12'hXXX;
    VOUTC = 12'hXXX;
    VOUTD = 12'hXXX;
end

initial begin
    tDZ = 200;                          // data to hi-Z delay
    tCSD = 320;                          // CS_N to data delay
end

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

assign DataDI = {DB11,DB10,DB09,DB08,DB07,DB06,DB05,DB04,DB03,DB02,DB01,DB00};

always @(DataDI or A0 or A1 or R_W or CS_N or RESET_N) begin
    if (RESET_N == 0) begin
        DACA_InputLatch = 12'h800;
        DACB_InputLatch = 12'h800;
    end
end

```

```

        DACC_InputLatch = 12'h800;
        DACD_InputLatch = 12'h800;
    end
    else begin
        case ({A1,A0, R_W,CS_N})
            4'b00_00 : DACA_InputLatch = DataDI[11:00];
            4'b01_00 : DACB_InputLatch = DataDI[11:00];
            4'b10_00 : DACC_InputLatch = DataDI[11:00];
            4'b11_00 : DACD_InputLatch = DataDI[11:00];
        endcase
    end
end
end

+ n
always @(LDAC_N or RESET_N or DACA_InputLatch or DACB_InputLatch or DACC_InputLatch or DACD_InputLatch) begin
    if (RESET_N == 0) begin
        VOUTA = 12'h800;
        VOUTB = 12'h800;
        VOUTC = 12'h800;
        VOUTD = 12'h800;
    end
    else if (LDAC_N == 0) begin
        VOUTA = DACA_InputLatch[11:00];
        VOUTB = DACB_InputLatch[11:00];
        VOUTC = DACC_InputLatch[11:00];
        VOUTD = DACD_InputLatch[11:00];
    end
end

always @(A0 or A1 or DACA_InputLatch or DACB_InputLatch or DACC_InputLatch or DACD_InputLatch) begin
    case ({A1,A0})
        2'b00 : DataDO = DACA_InputLatch[11:00];
        2'b01 : DataDO = DACB_InputLatch[11:00];
        2'b10 : DataDO = DACC_InputLatch[11:00];
        2'b11 : DataDO = DACD_InputLatch[11:00];
    endcase
end

assign #(tCSD,tDZ) DataOE = R_W & !CS_N;

bufif1 (DB00, DataDO[00], DataOE);
bufif1 (DB01, DataDO[01], DataOE);
bufif1 (DB02, DataDO[02], DataOE);
bufif1 (DB03, DataDO[03], DataOE);
bufif1 (DB04, DataDO[04], DataOE);
bufif1 (DB05, DataDO[05], DataOE);
bufif1 (DB06, DataDO[06], DataOE);
bufif1 (DB07, DataDO[07], DataOE);
bufif1 (DB08, DataDO[08], DataOE);
bufif1 (DB09, DataDO[09], DataOE);
bufif1 (DB10, DataDO[10], DataOE);
bufif1 (DB11, DataDO[11], DataOE);

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

wire TimingCheckEnable1 = !RESET & (R_W == 0);
wire TimingCheckEnable2 = !RESET & (R_W == 1);

specify
    specparam
        tRCS = 170, // CS_N pulse width - low
        tWCS = 150, // CS_N pulse width - low
        tRDH = 20, // R_W to CS_N hold time
        tWDS = 20, // data to CS_N setup time
        tLS = 70, // LDAC_N to CS_N setup time
        tLH = 50, // LDAC_N to CS_N hold time
        tLDW = 180, // LDAC_N pulse width - low
        tRESET = 150; // RESET_N pulse width - low

    $width (negedge CS_N &&& (R_W==1), tRCS);
    $width (negedge CS_N &&& (R_W==0), tWCS);
    $width (negedge LDAC_N, tLDW);
    $width (negedge RESET_N, tRESET);

    $setup (DB00, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
    $setup (DB01, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);

```

```
$setup (DB02, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB03, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB04, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB05, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB06, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB07, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB08, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB09, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB10, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);
$setup (DB11, negedge CS_N &&& (TimingCheckEnable1==1), tWDS);

$setup (LDAC_N, negedge CS_N &&& (RESET==0), tLS);
$setup (LDAC_N, posedge CS_N &&& (RESET==0), tLS);

$hold (posedge CS_N &&& (TimingCheckEnable2==1), R_W, tRDH);
$hold (posedge CS_N &&& (RESET==0), LDAC_N, tLH);
endspecify

endmodule
```