Video Image Capture with DE2

Christopher Upton

December 1, 2011

The DE2 Package

Special points of interest:

- Green screen
- SDRAM burst read
- SDRAM burst write
- SRAM controller
- Increased Frame Rate
- SRAM video buffer
- SDRAM memory A ^ B

Introduction

Welcome! We are going to capture digital video images onto an Altera DEV board. The required resources for this project include: VGA monitor, D%M camera, and Altera DE2 board. The most enjoyable part of this project is playing with the green screen. I would recommend this course to a friend. In this assignment, we learned how to write and read from SDRAM using a FIFO as a buffer (Sdram_Control_4Port.v). We were able to edit the code provided to meet specific specifications. In addition to editing provided code, we incorporate the SRAM_slave from midterm 2 into this project with little to no modifications. However, the SRAM_master must have been rewritten to meet the projects requirements. The two always block style of our Verilog modules simplified the debugging process. It was our goal to exhibit reasonable quality and reliability of the video image. Pak instructions contributed tremendously to our assignments success as a good professor should. Although most of the assignment was written by Johnny Fan, students were required to “re-write” Johnny’s code. This project

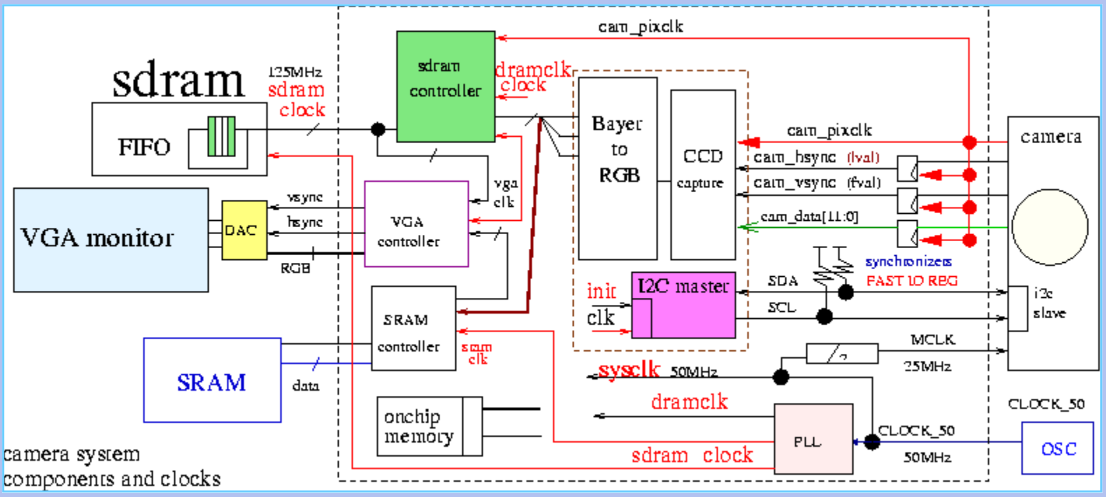


Table of Contents

Part 1 : Play and Record

Were are required to build an SRAM controller (implement as FSM coded in Verilog two-always block style) to be integrated into the original project. This entails that an external event (pushbutton) drive the logic to read/write to/from SRAM. SRAM is to memorize pixel data of 256K pixels.

Video Image Capture with DE2 ...

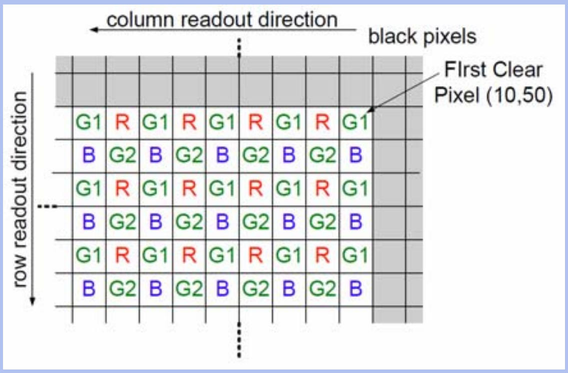
Christopher Upton

December 2, 2011

Part 2 : "We went to Spain!"

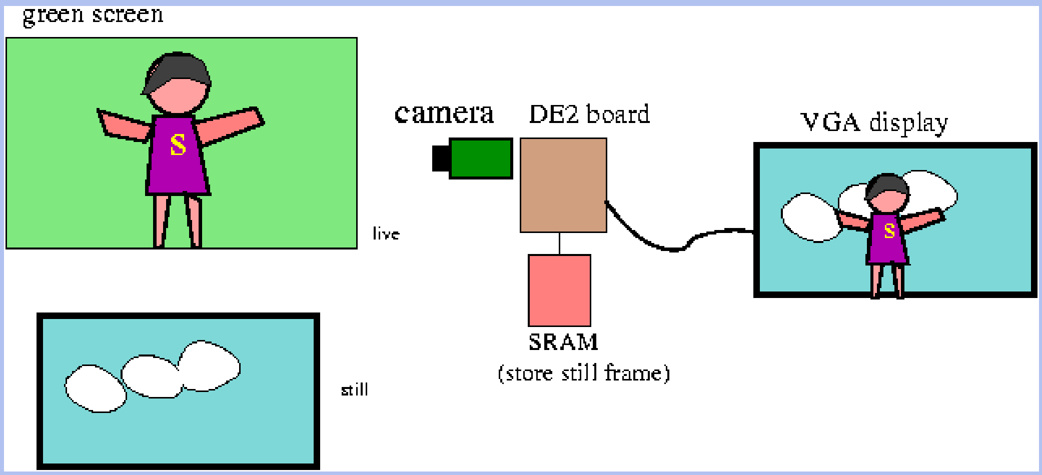
——-—-— Measure the "green" screen —————

GREEN[4:0]	RED[4:0]	BLUE[4:0]
01101	01011	01000
10011	01111	01100
01011	01001	00111
10001	01110	01100
10001	01101	01010
01000	00111	00101
10000	01110	01010



green_SCREEN = GREEN > RED > BLUE

green_SCREEN ? SRAM : SDRAM



Part III : Performance

2. Frame rate: improve frame rate from 10Hz to higher
3. Swapping the role of SDRAM controller with SRAM controller, i.e., SRAM is the buffer, SDRAM records one frame.

Major Project Components:

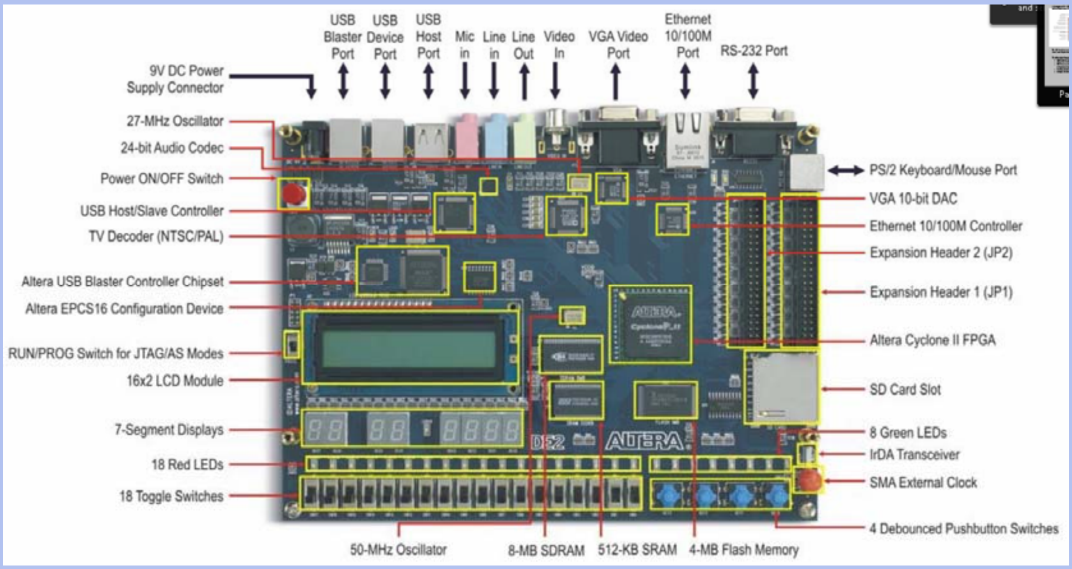
- D5M video camera module with a I2C (slave) interface.
- A I2C master module.
- A Bayer pattern to RGB converter.
- VGA controller for VGA converter.
- An SDRAM (SDR) memory chip and burst/multi-bank controller.
- SRAM controller.

Video Image Capture with DE2 ...

Christopher Upton

December 2, 2011

User Guide



SWITCH COMBINATION	RESULT
KEY[0]	Reset
KEY[1]	Take_Picture —- write compressed pixel data to SRAM
KEY[2]	Read_SRAM —- new pixel every 40ns to VGA
KEY[3]	Video_Camera —- SDRAM
!SW[0] & !SW[1]	Increase exposure time
SW[0] & !SW[1]	Decrease exposure time
SW[3]	Writing to SDRAM memory location A
!SW[3]	Writing to SDRAM memory location B
SW[4]	Reading from SDRAM memory location A
!SW[4]	Reading from SDRAM memory location B
SW[5]	Use SRAM as video camera buffer

Video Image Capture with DE2 ...

Christopher Upton

December 2, 2011

Sdram_Control_4Port.v

```
…………
input switches;
…………
//      Auto Read/Write Control
always@(posedge CLK or negedge RESET_N)
begin
      if(!RESET_N)
      begin
            mWR             <=      0;
            mRD             <=      0;
            mADDR           <=      0;
            mLENGTH         <=      0;
      end
      else
      begin

            if( (mWR==0) && (mRD==0) && (ST==0) &&
                  (WR_MASK==0)      &&      (RD_MASK==0) &&
                  (WR1_LOAD==0)     &&      (RD1_LOAD==0) &&
                  (WR2_LOAD==0)     &&      (RD2_LOAD==0) )
            begin
                  //      Write Side 1
                  if( (write_side_fifo_rusedw1 >= rWR1_LENGTH) && (rWR1_LENGTH!=0) )
                  begin
                        if (SW[3])
                              mADDR      <=      rWR1_ADDR  + 22'h200000;
                        else
                              mADDR  <=  rWR1_ADDR;
                        mLENGTH      <=      rWR1_LENGTH;
                        WR_MASK      <=      2'b01;
                        RD_MASK      <=      2'b00;
                        mWR          <=      1;
                        mRD          <=      0;
                  end
                  //      Write Side 2
                  else if( (write_side_fifo_rusedw2 >= rWR2_LENGTH) && (rWR2_LENGTH!=0) )
                  begin
                        if (SW[3])
                              mADDR      <=      rWR2_ADDR + 22'h200000;
                        else
                              mADDR      <=      rWR2_ADDR;
                        mLENGTH      <=      rWR2_LENGTH;
                        WR_MASK      <=      2'b10;
                        RD_MASK      <=      2'b00;
                        mWR          <=      1;
                        mRD          <=      0;
                  end
                  //      Read Side 1
                  else if( (read_side_fifo_wusedw1 < rRD1_LENGTH) )
                  begin
                        if (SW[4])
                              mADDR      <=      rRD1_ADDR + 22'h200000;
                        else
                              mADDR      <=      rRD1_ADDR;
                        mLENGTH      <=      rRD1_LENGTH;
                        WR_MASK      <=      2'b00;
                        RD_MASK      <=      2'b01;
                        mWR          <=      0;
                        mRD          <=      1;
                  end
                  //      Read Side 2
                  else if( (read_side_fifo_wusedw2 < rRD2_LENGTH) )
                  begin
                        if (SW[4])
                              mADDR      <=      rRD2_ADDR + 22'h200000;
                        else
                              mADDR      <=      rRD2_ADDR;
                        mLENGTH      <=      rRD2_LENGTH;
                        WR_MASK      <=      2'b00;
                        RD_MASK      <=      2'b10;
                        mWR          <=      0;
                        mRD          <=      1;
                  end
            end
            if(mWR_DONE)
            begin
                  WR_MASK      <=      0;
                  mWR          <=      0;
            end
            if(mRD_DONE)
            begin
                  RD_MASK      <=      0;

                  mRD          <=      0;
            end
      end
end

endmodule
```

Video Image Capture with DE2 ...

Christopher Upton

December 2, 2011

SRAM_slave.v

```
module SRAM_slave (sys_clk, areset_n, CE_N, OE_N, WE_N, data_drive,
        slave_DONE_read, slave_DONE_write, slave_READ,
        slave_WRITE);

input sys_clk;
input areset_n;
input slave_READ;
input slave_WRITE;
output reg CE_N;
output reg OE_N;
output reg WE_N;
output reg data_drive;
output reg slave_DONE_read;
output reg slave_DONE_write;
reg [7:0] wait_here;
reg [4:0] Sstate, Snext;

parameter       S0 = 5'b00000,
                S1 = 5'b00001,
                S2 = 5'b00010,
                S3 = 5'b00011,
                S4 = 5'b00100,
                S5 = 5'b00101,
                S6 = 5'b00110,
                S7 = 5'b00111,
                S8 = 5'b01000,
                S9 = 5'b10000;

always @(*)
begin
    case (Sstate)
        S0:     begin
                CE_N = 1;
                OE_N = 1;
                WE_N = 1;
                data_drive = 0;
                slave_DONE_read = 0;
                slave_DONE_write = 0;
                if (slave_WRITE)
                    Snext = S6;
                else if (slave_READ)
                    Snext = S1;
                else
                    Snext = S0;
                End

// READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ

        S1:         begin
                CE_N = 0;
                OE_N = 1;
                WE_N = 1;
                data_drive = 0;
                slave_DONE_read = 0;
                slave_DONE_write = 0;
                Snext = S2;
                end

        S2:         begin
                CE_N = 0;
                OE_N = 0;
                WE_N = 1;
                data_drive = 0;
                slave_DONE_read = 1;
                slave_DONE_write = 0;
                Snext = S3;
                end

        S3:         begin
                CE_N = 0;
                OE_N = 1;
                WE_N = 1;
                data_drive = 0;
                slave_DONE_read = 0;
                slave_DONE_write = 0;
                if (slave_WRITE)
                    Snext = S6;
                else if (slave_READ)
                    Snext = S1;
                else
                    Snext = S0;
                end

// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE

        S6:         begin
                CE_N = 0;
                OE_N = 1;
                WE_N = 1;
                data_drive = 1;
                slave_DONE_read = 0;
                slave_DONE_write = 0;
                Snext = S7;
                end

        S7:         begin
                CE_N = 0;
                OE_N = 1;
                WE_N = 0;
                data_drive = 1;
                slave_DONE_read = 0;
                slave_DONE_write = 1;
                Snext = S8;
                end

        S8:         begin
                CE_N = 0;
                OE_N = 1;
                WE_N = 1;
                data_drive = 1;
                slave_DONE_read = 0;
                slave_DONE_write = 0;
                if (slave_WRITE)
                    Snext = S6;
                else if (slave_READ)
                    Snext = S1;
                else
                    Snext = S0;
                end

        default:        begin
                    CE_N = 1;
                    OE_N = 1;
                    WE_N = 1;
                    data_drive = 0;
                    slave_DONE_read = 0;
                    slave_DONE_write = 0;
                    Snext = S0;
                    end
        endcase
end

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) begin Sstate <= S0; end
    else Sstate <= Snext;
end

endmodule
```

Video Image Capture with DE2 ...

Christopher Upton

SRAM_master.v

December 2, 2011

```
module SRAM_master (sys_clk, areset_n, slave_READ, slave_WRITE, SRAM_ADDR, SW,
            slave_DONE_read, slave_DONE_write, KEY_2, KEY_1, H_Cont, V_Cont);

input [17:0] SW;
input sys_clk;
input areset_n;
input slave_DONE_read;
input slave_DONE_write;
input KEY_1;
input KEY_2;
input [12:0]H_Cont;
input [12:0]V_Cont;

output reg slave_READ;
output reg slave_WRITE;
output reg [17:0] SRAM_ADDR;
    reg [32:0] PRE_SRAM_ADDR;
    reg [2:0] Mstate, Mnext;

parameter
            M1 = 3'b001,
            M2 = 3'b010,
            M0 = 3'b000;

always @(*)
begin
    case (Mstate)
        M0:         begin
                slave_READ = 0;
                slave_WRITE = 0;
                if (!KEY_1)
                    Mnext = M2;
                else if (!KEY_2)
                    Mnext = M1;
                else
                    Mnext = M1;
                end

        M1:         begin
                slave_READ = 1;
                slave_WRITE = 0;
                if (SW[5] && slave_DONE_read)
                    Mnext = M2;
                else if (!KEY_2)
                    Mnext = M1;
                else if (!KEY_1)
                    Mnext = M2;
                else
                    Mnext = M1;
                end

        M2:         begin
                slave_READ = 0;
                slave_WRITE = 1;
                if (SW[5] && slave_DONE_write)
                    Mnext = M1;
                else if (!KEY_1)
                    Mnext = M2;
                else
                    Mnext = M1;
                end

        default:    begin
                    slave_READ = 0;
                    slave_WRITE = 0;
                    Mnext = M0;
                    end
    endcase
end

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) SRAM_ADDR <= 0;
    else
        begin
        if (SW[5] && slave_DONE_write)
            begin
                if ((H_Cont == 13'b0) && (V_Cont == 13'b0))
                    SRAM_ADDR <= 0;
                else
                    SRAM_ADDR <= SRAM_ADDR + 1;
            end
        else if (slave_DONE_read || slave_DONE_write)
            if ((H_Cont == 13'b0) && (V_Cont == 13'b0))
                begin
                SRAM_ADDR <= 0;
                PRE_SRAM_ADDR <= 0;
                end
            else if (PRE_SRAM_ADDR > 32'h03ffff)
                begin
                PRE_SRAM_ADDR <= PRE_SRAM_ADDR + 1;
                SRAM_ADDR <= 0;
                end
            else
                begin
                PRE_SRAM_ADDR <= PRE_SRAM_ADDR + 1;
                SRAM_ADDR <= SRAM_ADDR + 1;
                end
        end
end

always @ (posedge sys_clk, negedge areset_n)

    if (!areset_n) Mstate <= M0;
    else Mstate <= Mnext;

endmodule
```