ARM-Based SoC Design – TFT LCD
Agenda

- TFT LCD & Driving IC
- Direct Memory Access (DMA)
- System Structure
- Project Overview Compile the Hardware
- Build the Software
- Link the Software Image with Still Images & Program the Board
TFT LCD & Driving IC
- 53.64(H) X 71.52(V)
- A–Si TFT Active Matrix
- 262,144 Color
- 240(H) X 320(V)
- RGB Vertical Stripe
### Electrical Absolute Ratings

\((Ta = 25 \pm 2^\circ C)\)

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital supply voltage</td>
<td>(D_{VDD})</td>
<td>3.0</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Analog supply voltage</td>
<td>(A_{VDD})</td>
<td>4.7</td>
<td>5.0</td>
<td>5.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Gate On voltage</td>
<td>(V_{ON})</td>
<td>15</td>
<td>18</td>
<td>20</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Gate Off voltage</td>
<td>(V_{OFF})</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Common voltage</td>
<td>(V_{CMO(V_{DD})})</td>
<td>3.6</td>
<td>4.0</td>
<td>4.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Digital supply current</td>
<td>(I_{DVDD})</td>
<td>-</td>
<td>0.35</td>
<td>0.9</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Analog supply current</td>
<td>(I_{AVDD})</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Gate supply current</td>
<td>(I_{VON/VOFF})</td>
<td>-</td>
<td>0.3</td>
<td>0.5</td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

#### Input voltage
- Source driver (High): \(V_{HS}\) = 0.8\(D_{VDD}\) - \(D_{VDD}\) V
- Source driver (Low): \(V_{ILS}\) = GND - 0.2\(D_{VDD}\) V

#### Power Dissipation
- White: \(P_W\) = (30) - (40) mW
- Black: \(P_B\) = (35) - (45) mW
- Vertical: \(P_V\) = (35) - (45) mW (1),(2)
Block Diagram

- **DVdd**
- **AVdd**
- **Von, Voff**
- **Gate driver control signals**
- **18-bit video data**
- **Source IC control signals**
- **Gamma voltages**
- **Vcom**

**3.5" Transflective TFT-LCD (240xRGBx320)**

**Driving IC**

**Connector with host**

A: Horizontal direction

B: Vertical direction

**Level Shifter**

**Source IC**

**AVdd**

**DVdd**
Driving IC (LCC3600A)

- Resolution: 240(H) X 320(V)
- Generate Gate Clock and Gate Start signal for Integrated Gate Driver
- Generate signals for VCOM
- 6bits RGB Input data for 260K color display
- Function of Gate clock control
- Maximum Operating Frequency: 20MHz
- Operating Voltage: 3.0V ~ 3.6V
- Package: 64-ELp
Block Diagram

Excalibur

TFT LCD

LCC3600A

- STH
- LD
- INV
- HCLK
- OUTPUT DATA (18 bits)
- STV
- CKV
- REV/REV

For Source Driver

For Gate Driver Circuits

For VCOM
## AC Electrical Characteristics

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLK</td>
<td>Frequency</td>
<td>1/(T_r)</td>
<td>-</td>
<td>5</td>
<td>20</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>High level width</td>
<td>(T_{CH})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>Low level width</td>
<td>(T_{CL})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>INPUT DATA</td>
<td>setup time</td>
<td>(T_{SA})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>hold time</td>
<td>(T_{DH})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>DE</td>
<td>setup time</td>
<td>(T_{SA})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td>hold time</td>
<td>(T_{DH})</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
</tbody>
</table>

![AC Electrical Characteristics Diagram](image_url)
hsync

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Total</td>
<td>T_{v, total}</td>
<td>324</td>
<td>327</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Blank Time</td>
<td>T_{v, blank}</td>
<td>4</td>
<td>7</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Display</td>
<td>T_{v, data}</td>
<td>-</td>
<td>320</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Horizontal Total</td>
<td>T_{h, total}</td>
<td>254</td>
<td>282</td>
<td>512</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Blank Time</td>
<td>T_{h, blank}</td>
<td>14</td>
<td>42</td>
<td>272</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Display</td>
<td>T_{h, data}</td>
<td>-</td>
<td>240</td>
<td>-</td>
<td>MCLK</td>
</tr>
</tbody>
</table>
Output Timing (Vertical)

- DE
- STH
- DATA OUTPUT
- LD
- STV
- CKV
- INV
- REV
- REV B
Output Timing (Horizontal)
Direct Memory Access (DMA)
Need to transfer blocks of data from I/O channel to system memory, or memory to custom logic or from one memory location to another

To maximize system performance

Excalibur devices are programmable, it is possible to implement DMA controllers
I/O to Memory DMA in EPXA
Descriptor Based DMA in EPXA
System Structure
Block Diagram

ALTERA Excalibur Device

- ARM922T
- SDRAM Controller
- SDRAM Device
- PLD-to-Stripe Bridge
- Stripe-to-PLD Bridge
- Embedded ARM Stripe

- AHB Slave Decoder
- AHB Master Interface
- AHB Slave Interface
- Default Slave
- AHB Read Data Multiplexer

- DMA Controller
- Register Bank
- Line Buffer
- LCD Driver

FPGA Array

240x320 TFT-LCD
LCD Driver

- Generate timing signal for 240 X 320 image running at 60 frames per second
- Image size and Output enable from AHB Slave
- Begins driving pixels to LCD when enabled
- Pixels are transferred from SDRAM to LCD driver via a line-buffer
- Line buffer is 32 bits, so contains 2 pixels
- The pixels are in a 5–6–5 RGB format
Figure 5. Input Signal Timing Diagram

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Total</td>
<td>$T_{V,\text{TOTAL}}$</td>
<td>324</td>
<td>327</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Blank Time</td>
<td>$T_{V,\text{BLANK}}$</td>
<td>4</td>
<td>7</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Display</td>
<td>$T_{V,\text{DATA}}$</td>
<td>-</td>
<td>320</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Horizontal Total</td>
<td>$T_{H,\text{TOTAL}}$</td>
<td>254</td>
<td>282</td>
<td>512</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Blank Time</td>
<td>$T_{H,\text{BLANK}}$</td>
<td>14</td>
<td>42</td>
<td>272</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Display</td>
<td>$T_{H,\text{DATA}}$</td>
<td>-</td>
<td>240</td>
<td>-</td>
<td>MCLK</td>
</tr>
</tbody>
</table>
- Transmit images from SDRAM to LCD Driver
- Consumes a large amount of memory bandwidth
- Configured by the slave interface
- Base address of frame buffer register
- Size of image to be transferred register
- Enable DMA control register
Line Transfer Timing

hblank

vblank

control state

IDLE

XMIT LINE

IDLE

XMIT LINE

IDLE

hrdata

Begin Line

End Line
- Register bank contains control and status registers
- Also have a slave decoder and a default slave
- And the data and response multiplexer is exist
## Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Mnemonic</th>
<th>Address (offset)</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Address</td>
<td>BUFFER_ADDRESS</td>
<td>00</td>
<td>R/W</td>
</tr>
<tr>
<td>Image dimensions</td>
<td>IMAGE_DIMENSIONS</td>
<td>04</td>
<td>R/W</td>
</tr>
<tr>
<td>Control</td>
<td>CONTROL</td>
<td>08</td>
<td>R/W</td>
</tr>
<tr>
<td>Current address</td>
<td>CURRENT_ADDRESS</td>
<td>0C</td>
<td>R</td>
</tr>
<tr>
<td>Status</td>
<td>STATUS</td>
<td>10</td>
<td>R</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>14</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
</tbody>
</table>
## Buffer Address Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>HADDR</td>
<td>32-bit base address in the frame buffer memory</td>
</tr>
</tbody>
</table>

## Image Dimensions Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>NUM_LINES</td>
<td>The total number of lines</td>
</tr>
<tr>
<td>31:16</td>
<td>NUM_PIXEL_PER_LINE</td>
<td>The number of pixels in each line</td>
</tr>
</tbody>
</table>
### Control Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EN_DMA</td>
<td>DMA enable: 1, disable: 0</td>
</tr>
<tr>
<td>31:1</td>
<td>Reserved</td>
<td>Future use</td>
</tr>
</tbody>
</table>

### Current Address Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>HADDR</td>
<td>32-bit base address in memory that is currently being read by the DMA controller</td>
</tr>
</tbody>
</table>

### Status Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HB</td>
<td>Horizontal blanking signal from the LCD Driver</td>
</tr>
<tr>
<td>1</td>
<td>VB</td>
<td>Vertical blanking signal from the LCD driver</td>
</tr>
<tr>
<td>31:2</td>
<td>Reserved</td>
<td>Write all 0s to ensure future compatibility</td>
</tr>
</tbody>
</table>
- Compile the Hardware
- Build the Software
- Link the Software Image with Still Images & Program the Board
- Simulate the Reference Design
Directory Structure

dma_lcd_reference_design

- images
  Contains the perl script that converts .bmp to .hex files
- prog_files
  Contains the programming batch files
- quartus
  Contains the Quartus II project
- rtl_sim
  Contains the simulation and testbench files
- software
  Contains the software design files
- source
  Contains the hardware design source files
## Reference Design Files (1 of 2)

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\source\video_system.v</td>
<td>Reference design top-level design file</td>
</tr>
<tr>
<td>\source\video_dma.v</td>
<td>Video DMA driver top-level</td>
</tr>
<tr>
<td>\source\video_dma_controller.vq</td>
<td>DMA controller portion of the design</td>
</tr>
<tr>
<td>\source\slave_interface.v</td>
<td>AHB Slave containing a register bank</td>
</tr>
<tr>
<td>\source\tft_lcd.v</td>
<td>LCD driver</td>
</tr>
<tr>
<td>\source\default_slave.v</td>
<td>AHB default slave</td>
</tr>
<tr>
<td>\source\slave_decoder.v</td>
<td>AHB Slave address decoder</td>
</tr>
<tr>
<td>\source\response_and_data_mux.v</td>
<td>AHB Slave data and response multiplexer</td>
</tr>
<tr>
<td>\software\main.c</td>
<td>The main C file used to test the design</td>
</tr>
<tr>
<td>\software\irq.c</td>
<td>The interrupt service routine</td>
</tr>
<tr>
<td>\software\uartcomm.c</td>
<td>UART driver</td>
</tr>
<tr>
<td>Filename</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td><code>\software\ irq.h,uartcomm.h,uart00.h , int_ctrl00.h</code></td>
<td>Head files for software design</td>
</tr>
<tr>
<td><code>\images\ bmp2perl.pl</code></td>
<td>Perl script used to convert .bmp to .hex files</td>
</tr>
<tr>
<td><code>\prog_files\ prog_hw.bat</code></td>
<td>Batch file used to link software and download code to the target board</td>
</tr>
</tbody>
</table>
Requiements

- Quartus II v4.0 (or later version)
- ARM Development Suite v1.2
- Mentor Graphics ModelSim v5.7 (or later)
Compile the Hardware
Run Quartus II
Create New Project
- Name: Video_system
- Top-level Entity: Video_system
- Target: EPXA4F672C3
Add H/W design files

- Menu > Project > Add/Remove … Click
- Add *.v or *.vqm to project
Video_system
- Stripe
- System_pll
- Default_slave
- Reponse_and_data_mux
- Slave_decoder
- Video_dma
  - Video_dma_controller
  - Tft_lcd
  - Slave_interface
  - Line_buffer
Tools > MegaWizard. Click
ARM-Based Excalibur

- Megafuction: ARM-Based Excalibur
- Output file: Verilog HDL
- Output file: stripe.v
System Setting

- Device Selection
  - Excalibur_ARM
  - EPXA4
- Reset Operation
  - Boot from Flash
- Byte Order
  - Little Endian
- Reserve Pins
  - EBI Automatically Enabled
  - Enable UART Pins
  - Disable Trace Pins
Stripe-to-PLD Interface

- Bridges
  - Stripe to PLD
  - PLD to Stripe
- Interrupts
  - PLD to Stripe interrupt – Mode 3
- Trace/Debug
  - Disable Trace Extensions
Clocks

- **Clk_ref**
  - 25 MHz

- **AHB1/AHB2**
  - Enable PLL1
  - AHB1: 200 MHz
  - AHB2: 100 MHz

- **SDRAM Controller**
  - Enable PLL2
  - 133MHz

- **Serial Programming**
  - Enable But Not Used
Memory Map

- Registers
  - Default: 0x7FFFC000; 16K
- SDRAM0:
  - 0x20000000; 64M; 16bit
- SDRAM1: OFF
- EBI0
  - 0x40000000; 16M
- EBI1, EBI2, EBI3: OFF
- PLD0
  - 80000000; 2G
- PLD1, PLD2, PLD3: OFF
- Synchronous
- Wait Cycles: 8
- CS Polarity: Active Low
- Data Width: 16 Bits
- Bus Clock Divide: 1
Display file to create

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stripe.v</td>
<td>Variation File</td>
</tr>
<tr>
<td>stripe_bb.v</td>
<td>Verilog Black Box Module</td>
</tr>
<tr>
<td>stripe.tmpl</td>
<td>Template File</td>
</tr>
<tr>
<td>stripe.bs</td>
<td>Block Symbol File</td>
</tr>
<tr>
<td>stripe.h</td>
<td>C Header File</td>
</tr>
<tr>
<td>stripe.s</td>
<td>Assembly Header File</td>
</tr>
</tbody>
</table>
- Designed PLL
- Generate 60MHz and 50MHz clock
- Megawizard™
- AMBA Bus Interface
- HREADY, HRESP[1:0] Signal generation
case (state)
    address_phase :
        if (hsel)
            if (htrans == 2'b10 | htrans == 2'b11)
                begin
                    state <= error_phase;
                    hresp <= 2'b01; // error response
                    hready <= 0;
                end
            else begin
                state <= address_phase;
                hresp <= 2'b00; // okay response
                hready <= 1;
            end
        else begin
            state <= address_phase;
            hresp <= 2'b00; // okay response
            hready <= 1;
        end
endcase

error_phase : begin
    state <= address_phase;
    hresp <= 2'b01; // error response
    hready <= 1;
end
Address decoder block
- Address = 32x’80000000 : hsel_slave_iface = 1;
- Address /= 32x’80000000 : hsel_default_slave = 1;
assign current_base_address = 32'h80000000

always @(hbusreq, haddr)
if ((hbusreq) & (haddr[31:10] == current_base_address[31:10]))
begin // 8000000xx
  hsel_slave_iface <= 1;
  hsel_default_slave <= 0;
end else if ((hbusreq) & (haddr[31:10] != current_base_address[31:10]))
begin // not 8000000xx
  hsel_slave_iface <= 0;
  hsel_default_slave <= 1;
end
- Video_dma to Stripe Data Interface Block
- Hsel_slave_iface = 1 : video_dma hrdata trans
Response_and_data_mux block

// hrdata mux
always @(hsel_slave_iface_delay,hssel_default_slave,hrdata_slave_iface,hresp_slave_iface,hresp_default_slave)

if (hsel_slave_iface_delay) begin
    hrdata <= hrdata_slave_iface;
    hresp <= hresp_slave_iface;
end else begin
    hresp <= hresp_default_slave;
    hrdata <= 0; // we are idle or have a default slave hit
end
- AMBA interface
- DMA controller
- TFT LCD controller
- DPRAM (line_buffer)
- AMBA interface
- Set Register (DMA controller, VGA Driver)
case (state)  
    address :  
        if ((hsel) & (htrans == 2'b10)) begin  
            if (hwrite)  
                internal_write <= 1;  
            else  
                internal_write <= 0;  
            state <= data;  
        end else begin  
            internal_write <= 0;  
            state <= address;  
        end  
    data :  
        // Remain in data state on burst transfers  
        if (htrans == 2'b11) begin  
            if (hwrite)  
                internal_write <= 1;  
            else  
                internal_write <= 0;  
            state <= data;  
        end else begin  
            internal_write <= 0;  
            state <= address;  
        end  
    default : begin  
        internal_write <= 0;  
        state <= address;  
    end  
endcase
AHB Transaction consist of address and data phase
Slave_interface block

internal_address <= haddress[4:2];

if (internal_write)
  case (internal_address)
    case (internal_address)
      3'b000 : buffer_address_reg <= hwdata;
      3'b001 : image_dimensions_reg <= hwdata;
      3'b010 : control_reg <= hwdata;
      endcase
  endcase

if ((hsel) & (~hwrite))
  case (haddress[4:2])
    case (haddress[4:2])
      3'b000 : hrdata <= buffer_address_reg;
      3'b001 : hrdata <= image_dimensions_reg;
      3'b010 : hrdata <= control_reg;
      3'b011 : hrdata <= current_address;
      3'b100 : hrdata <= status;
      default : hrdata <= 0;
      endcase
  endcase
<table>
<thead>
<tr>
<th>Name</th>
<th>Mnemonic</th>
<th>Address (offset)</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Address</td>
<td>BUFFER_ADDRESS</td>
<td>00</td>
<td>R/W</td>
</tr>
<tr>
<td>Image dimensions</td>
<td>IMAGE_DIMENSIONS</td>
<td>04</td>
<td>R/W</td>
</tr>
<tr>
<td>Control</td>
<td>CONTROL</td>
<td>08</td>
<td>R/W</td>
</tr>
<tr>
<td>Current address</td>
<td>CURRENT_ADDRESS</td>
<td>0C</td>
<td>R</td>
</tr>
<tr>
<td>Status</td>
<td>STATUS</td>
<td>10</td>
<td>R</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>14</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
</tbody>
</table>
- DPRAM (Dual Port RAM)
- Image Data Storage
- MegaWizard™
**Video_DMA_controller**

- AMBA interface (PLD to Stripe bridge)
- DMA controller
- DPRAM Interface
- Image Data Store
always@(posedge clk)
begin
  dma_hblank <= hblank;
end

start_of_line <= (!dma_hblank) & (hblank) & (vblank)) ? 1 : 0;

always @(posedge clk)
begin
  case control_state
    idle :
      begin
        if ([start_of_line] & [en_dma])
          control_state <= begin_line;
        masterbusreq <= 1;
        else
          masterbusreq <= 0;
      end
    begin_line :
      begin
        control_state <= transmit_line;
        masterbusreq <= 1;
        masterhlock <= 1;
        irq <= 0;
      end
    end_line :
      begin
        control_state <= idle;
        start_transfer <= 0;
        masterlock <= 0;
        if (line_counter == num_line)
          line_counter <= 0;
        irq <= 1;
      end
    transmit_line :
      begin
        masterhlock <= 1;
        irq <= 0;
        if ([burst_counter == num_pixel_line] & [burst_beat_counter == 5])
          begin
            control_state <= end_line;
            masterbusreq <= 0;
          end
        else
          begin
            masterbusreq <= 1;
            control_state <= transmit_line;
          end;
      end
  end
end
- TFT LCD Controller
- DPRAM Interface
  - Image Data Read
if (line_cnt < 320) begin
  if (pixel_cnt < 42) begin
    lcd_de_sig <= 0;
    pixel_cnt <= pixel_cnt + 1;
    rdaddr_to_dpram_sig <= 0;
    hblank <= 0;
  end else if (pixel_cnt < 282) begin
    lcd_de_sig <= 1;
    pixel_cnt <= pixel_cnt + 1;
    rdaddr_to_dpram_sig <= rdaddr_to_dpram_sig + 1;
    half_word_sel <= (~ half_word_sel);
    hblank <= 1;
  end else begin
    lcd_de_sig <= 0;
    pixel_cnt <= 0;
    line_cnt <= line_cnt + 1;
    rdaddr_to_dpram_sig <= 0;
    hblank <= 0;
  end
end else if (line_cnt < 327) begin
  if (pixel_cnt < 282 )
    pixel_cnt <= pixel_cnt + 1;
  else begin
    pixel_cnt <= 0;
    line_cnt <= line_cnt + 1;
  end
end else if (pixel_cnt < 282) begin
  lcd_de_sig <= 0;
  rdaddr_to_dpram_sig <= 0;
  vblank <= 0;
end else begin
  line_cnt <= 0;
  lcd_de_sig <= 0;
  rdaddr_to_dpram_sig <= 0;
  vblank <= 0;
end
else begin
  lcd_de <= lcd_de_sig;
end

vblank <= 1;

assign lcd_data = (~half_word_sel) ? rddata_from_dpram[15:0] : rddata_from_dpram[31:16];
assign lcd_mclk = lcd_mclk_sig;
assign current_pixels = pixel_cnt;
assign current_lines = line_cnt;
assign rdaddr_to_dpram = {1'b0, rdaddr_to_dpram_sig[31:1]};
Figure 5. Input Signal Timing Diagram

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Total</td>
<td>$T_{V_{\text{total}}}$</td>
<td>324</td>
<td>327</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Blank Time</td>
<td>$T_{V_{\text{blank}}}$</td>
<td>4</td>
<td>7</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Vertical Display</td>
<td>$T_{V_{\text{data}}}$</td>
<td>-</td>
<td>320</td>
<td>-</td>
<td>Lines</td>
</tr>
<tr>
<td>Horizontal Total</td>
<td>$T_{H_{\text{total}}}$</td>
<td>254</td>
<td>282</td>
<td>512</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Blank Time</td>
<td>$T_{H_{\text{blank}}}$</td>
<td>14</td>
<td>42</td>
<td>272</td>
<td>MCLK</td>
</tr>
<tr>
<td>Horizontal Display</td>
<td>$T_{H_{\text{data}}}$</td>
<td>-</td>
<td>240</td>
<td>-</td>
<td>MCLK</td>
</tr>
</tbody>
</table>
Device Settings

Device & Pin Options

Dual-Purpose Pins

- General
- Configuration
- Programming Files
- Unused Pins

Options:
- Auto-reset configuration after error
- Release clear before time-out
- Enable user-supplied startup clock (CLKUSR)
- Enable device-wide reset (DEV_CLRn)
- Enable device-wide output enable (DEV_OE)
- Enable INIT_DONE output

- Generate compressed JTAG code
- Auto usermode

JTAG user code (32-bit hexadecimal): FFFFFF

Description:
Directs the device to restart the configuration process automatically if a data error is encountered. If this option is turned off, you must externally direct the device to restart the configuration process if an error occurs.

Reset

OK Cancel

Device & Pin Options

Dual-Purpose Pins

- General
- Configuration
- Programming Files
- Unused Pins

Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve unused pins individually, click the "Assign Pins" button in the Device page of the Settings dialog box.

- Reserve all unused pins:
  - As inputs, hi-stated
  - As outputs, driven ground
  - As outputs, driven an unspecified signal

Description:
Reserves all unused pins on the target device in one of these states: as inputs that are hi-stated, as outputs that drive ground, or as outputs that drive an unspecified signal.

Reset

OK Cancel
### Assign Pin to signals & Synthesis

### Compile All H/W Design

<table>
<thead>
<tr>
<th>No.</th>
<th>To</th>
<th>Location</th>
<th>I/O Bank</th>
<th>I/O Standard</th>
<th>General Function</th>
<th>Special Function</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bled</td>
<td>PIN_B18</td>
<td>3</td>
<td>LVTTL</td>
<td>Column I/O</td>
<td>DATA2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>clock25</td>
<td>PIN_W6</td>
<td>13</td>
<td>LVTTL</td>
<td>Dedicated Clock</td>
<td>CLK2p</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>lcd_data[0]</td>
<td>PIN_AA21</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>lcd_data[1]</td>
<td>PIN_AA22</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>lcd_data[2]</td>
<td>PIN_AA23</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>lcd_data[3]</td>
<td>PIN_AA24</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>lcd_data[4]</td>
<td>PIN_AA25</td>
<td>8</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td>LVDSR&gt;02n</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>lcd_data[5]</td>
<td>PIN_AA26</td>
<td>8</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td>LVDSR&gt;02p</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>lcd_data[6]</td>
<td>PIN_AB17</td>
<td>10</td>
<td>LVTTL</td>
<td>Column I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>lcd_data[7]</td>
<td>PIN_AB18</td>
<td>10</td>
<td>LVTTL</td>
<td>Column I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>lcd_data[8]</td>
<td>PIN_AB19</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>lcd_data[9]</td>
<td>PIN_AB20</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>lcd_data[10]</td>
<td>PIN_AB21</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>lcd_data[12]</td>
<td>PIN_AB23</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>lcd_data[13]</td>
<td>PIN_AB24</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>lcd_data[14]</td>
<td>PIN_AB25</td>
<td>8</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td>LVDSR×INCLK1n</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>lcd_data[15]</td>
<td>PIN_AB26</td>
<td>8</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td>LVDSR×INCLK1p</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>lcd_de</td>
<td>PIN_AA19</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>lcd_ml</td>
<td>PIN_AA20</td>
<td>9</td>
<td>LVTTL</td>
<td>Row I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>sys_clk</td>
<td>PIN_VS</td>
<td>13</td>
<td>LVTTL</td>
<td>Dedicated Clock</td>
<td>CLK1p</td>
<td></td>
</tr>
</tbody>
</table>

67
Build the Software
Add Software files

- Browse the software folder
- Add below files
  - Armc_start.s
  - Retarget.c
  - Uartcomm.c
  - Irq.c
  - Main.c
  - Exception.c
## Software Build Settings

### Settings - video_system

<table>
<thead>
<tr>
<th>Category</th>
<th>General</th>
<th>C/C++ Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Software Build Settings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Embedded processor architecture: ARM926</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software package: ADS Standard Tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Buildroot: Linux ENTRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output file format: Executable (File + elf)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output file name: video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Command line:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--build-root=Linux ENTRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--arch=arm926</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--output=video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--elf-program-file=Video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--elf-executable-file=Video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--elf-machine-browser=video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--elf-machine-browser=video.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--elf-machine-browser=video.bin</td>
</tr>
</tbody>
</table>

### Programming file generation

- Board: |
  - Film: Include Binary Image File (Code) |
  - Video_system.js |
  - Passive configuration: Include programmable logic: Partial SDF (post) |

### C/C++ Compiler

- Optimization: |
  - Level: High, Goal: None |
  - Process commands: separate multiple directives with space |
  - Additional include directories: separate multiple directories with space |
  - Source: |
  - Link: |
  - Toolset: |
  - Hardware Settings: 62 4 1 Hardware
Software Build Settings

- General
- File
- Libraries
- Device
- Timing/Requirements & Options
- EDA Tool Settings
- Compiler Process
- Analysis & Synthesis Settings
- Filter Settings
- Timing Analyzer
- Design Assistant
- SignalTap Logic Analyzer
- Signal Simulation Settings
- Simulator
- Software Build Settings
- EDIF Compiler
- EDIF
- Target Directories
- SDK/Stage Registration
- Hardware Settings

- Assembler
- Preprocessor definitions (separate multiple definitions with commas)

- Additional include directories (separate multiple directories with commas)
- Generate debug information
- Keep local symbols in symbol table
- Support sections

- Command-line options:
  - g-keep-1

- Linker
- Select options for the linker. You can use the scroll bar at the bottom to add other command-line options. Note: The availability of some options is determined by the current Software toolset and your output file format used in the CRP file.

- Object/library modules (separate multiple modules with commas)
- Additional library directories (separate multiple directories with commas)

- Link type
- Simple
- Create symbol name address
- Read only base address
- Read/write base address 0x0000
- Custom link script

- Command-line options:
  - first_mmc_start (setenv): "setenv \"0x0000 \"\"
Build Software

- Build All Software files
- Menu Processing > Start Software Build
int main(void)
{
    volatile int *DMA = (int*) EXC_PLD_BLOCKO_BASE;

    // initialize the video driver
    *DMA = EXC_SDRAM_BLOCKO_BASE + 0x00100000;  // base address of frame buffer
    *(DMA+1) = 0x00f00140;  // image size (240X320)
    *(DMA+1) = 0x028001e0; // image size (640x480)
    *(DMA+1) = 0x03200258; // image size (800x600)
    *(DMA+1) = 0x04000300; // image size (1024x768)

    *(DMA+2) = 0x00000001;  // start driver

    while(1);
    // continuously loop and service vga
    // interrupts in the irq.c file
}

<table>
<thead>
<tr>
<th>Name</th>
<th>Mnemonic</th>
<th>Address (offset)</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Address</td>
<td>BUFFER_ADDRESS</td>
<td>00</td>
<td>R/W</td>
</tr>
<tr>
<td>Image dimensions</td>
<td>IMAGE_DIMENSIONS</td>
<td>04</td>
<td>R/W</td>
</tr>
<tr>
<td>Control</td>
<td>CONTROL</td>
<td>08</td>
<td>R/W</td>
</tr>
<tr>
<td>Current address</td>
<td>CURRENT_ADDRESS</td>
<td>0C</td>
<td>R</td>
</tr>
<tr>
<td>Status</td>
<td>STATUS</td>
<td>10</td>
<td>R</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>14</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
</tbody>
</table>
### Buffer Address Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>HADDR</td>
<td>32-bit base address in the frame buffer memory</td>
</tr>
</tbody>
</table>

### Image Dimensions Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>NUM_LINES</td>
<td>The total number of lines</td>
</tr>
<tr>
<td>31:16</td>
<td>NUM_PIXEL__PER_LINE</td>
<td>The number of pixels in each line</td>
</tr>
</tbody>
</table>
### Control Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EN_DMA</td>
<td>DMA enable: 1, disable: 0</td>
</tr>
<tr>
<td>31:1</td>
<td>Reserved</td>
<td>Future use</td>
</tr>
</tbody>
</table>

### Current Address Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>HADDR</td>
<td>32-bit base address in memory that is currently being read by the DMA controller</td>
</tr>
</tbody>
</table>

### Status Register Format

<table>
<thead>
<tr>
<th>Data Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HB</td>
<td>Horizontal blanking signal from the LCD Driver</td>
</tr>
<tr>
<td>1</td>
<td>VB</td>
<td>Vertical blanking signal from the LCD driver</td>
</tr>
<tr>
<td>31:2</td>
<td>Reserved</td>
<td>Write all 0s to ensure future compatibility</td>
</tr>
</tbody>
</table>
void CIrqHandler(void)
{
    volatile int irqID;

    irqID = *INT_ID(EXC_INT_CTRL00_BASE);

    switch (irqID)
    {
        case PLDINT1_IRQ_PRI:
            vga_irq_handler();
            break;
        case UART_IRQ_PRI:
            uart_irq_handler();
            break;
        default:
            /* This shouldn't happen, but let's trap it in case */
            printf("Unknown irq %#x", irqID);
            break;
    }

    return;
}

void vga_irq_handler(void)
{
    volatile int *VGA = (int *) EXC_PLD_BLOCK0_BASE;

    // update the VGA driver's buffer base address.
    // This also clears the irq.
    *(VGA + 2) = 1;
}

// This should not happen, but let's trap it in case
// printf("Unknown irq %#x", irqID);
// break;
Link the Software Image with Still Images & Program the Board
- Browse Prog_files folder
  - Prog_hw.bat
    - makeproglob -b
      video_system.o ..\quartus\video_system.sbd ..\quartus\video_system.sbi ..\quartus\video_dma.hex
      image_data.hex
    - armlink -ro 0 -o video_system.elf
      video_system.o %QUARTUS_ROOTDIR%\libraries\software\boot\LIBBOOT_XA_ADS.A
    - fromelf -i32 video_system.elf -o video_board.hex
    - exc_flash_programmer video_board.hex -g
- Altera provide Perl Script
- Convert 24-bit .bmp into 16-bit .hex for storage in SDRAM
- Usage
  - Perl bmp2hex.pl input_file.bmp
  - Default output file: image_data.hex
- Check Power
- Connect ByteBlaster cable between PC and Target Platform
- SoC-Entry II Dip-Switch S1 & S6 are all off in the bottom-center placed
- Turn on Target Board Power
- Run \%Prog_files\%Prog_hw.bat
- LCD & Driving IC
- DMA and Structure
- System component
- H/W Design and Software Design
- Link images and Download Board