# Graphics and Displays

Lecture 15

Josh Brake
Harvey Mudd College

# Outline

- Video Graphics Array (VGA)
- Liquid Crystal Display (LCD)
- Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI)
- Miscellaneous other displays

# Learning Objectives

By the end of this lecture you will…

- Understand the basic operation of the video graphics array (VGA) protocol and how to implement it on an FPGA.
- Understand how liquid crystal displays (LCDs) work and how to control them with a digital device.
- Explain the basic concepts behind Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI).

# Video Graphics Array (VGA)

# Video Graphics Array (VGA) History

- Introduced in 1987 for the IBM PS/2 computers
- Video information relayed using analog voltages
- 15-pin connector

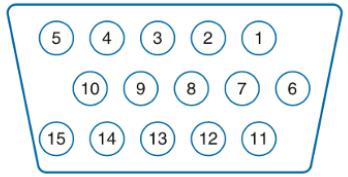
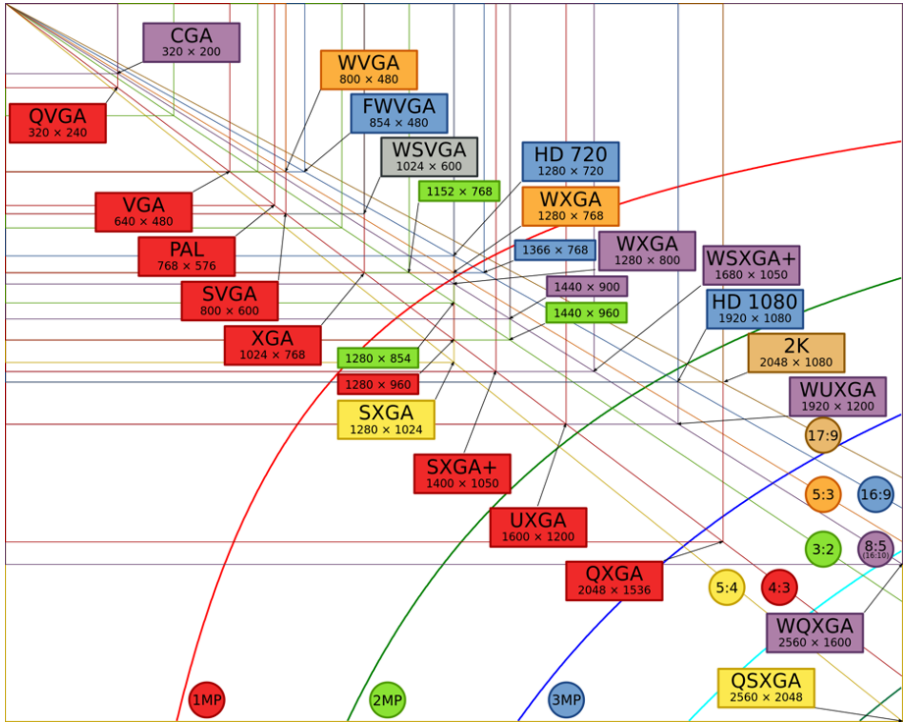



**Figure e9.27 VGA connector pinout**

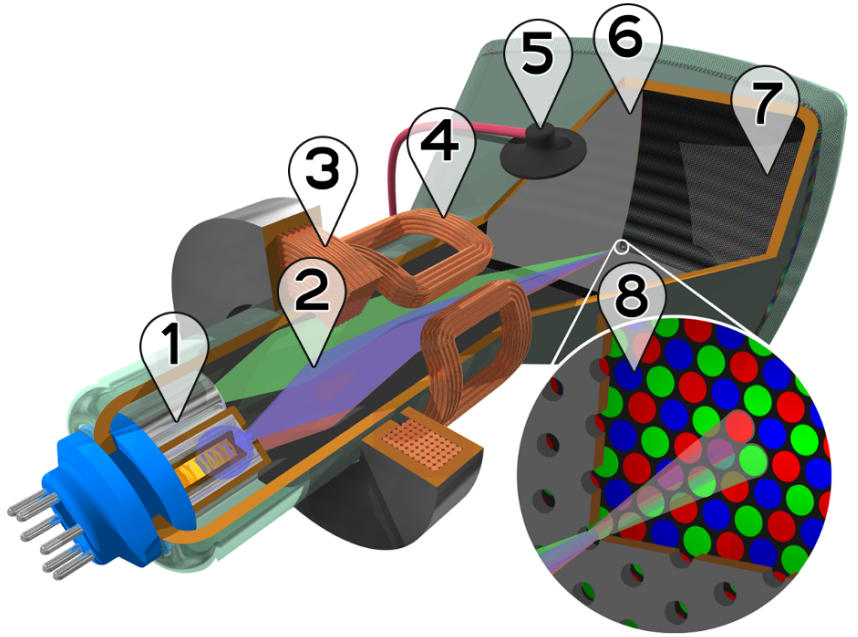
Figure e9.27 in DDCA ARMed Edition




Comparison of standard resolutions including VGA's 640x480 by XXV under CC BY-SA 3.0

# VGA and Cathode Ray Tubes (CRTs)

1. Electron beam emitters
2. Electron beams
3. Focusing coils
4. Deflection coils
5. Connection for final anodes
6. Mask for separating red, green, and blue zones
7. Close-up of phosphor-coated inside of screen



# VGA Timing

- Cathode ray tube work by raster scanning left to right and exciting fluorescent material for each pixel.
- At the end of the line, the gun turns off for the horizontal blanking interval to return to the beginning of the next line.
- After all lines are complete, it turns off for the vertical blanking interval to return to the top left
- Each line begins and ends with a “porch” which is a blank area of zeros where the CRT gun is turned off while it is moving in position for the next line. These porches exist for both horizontal and vertical lines.

# VGA Timing

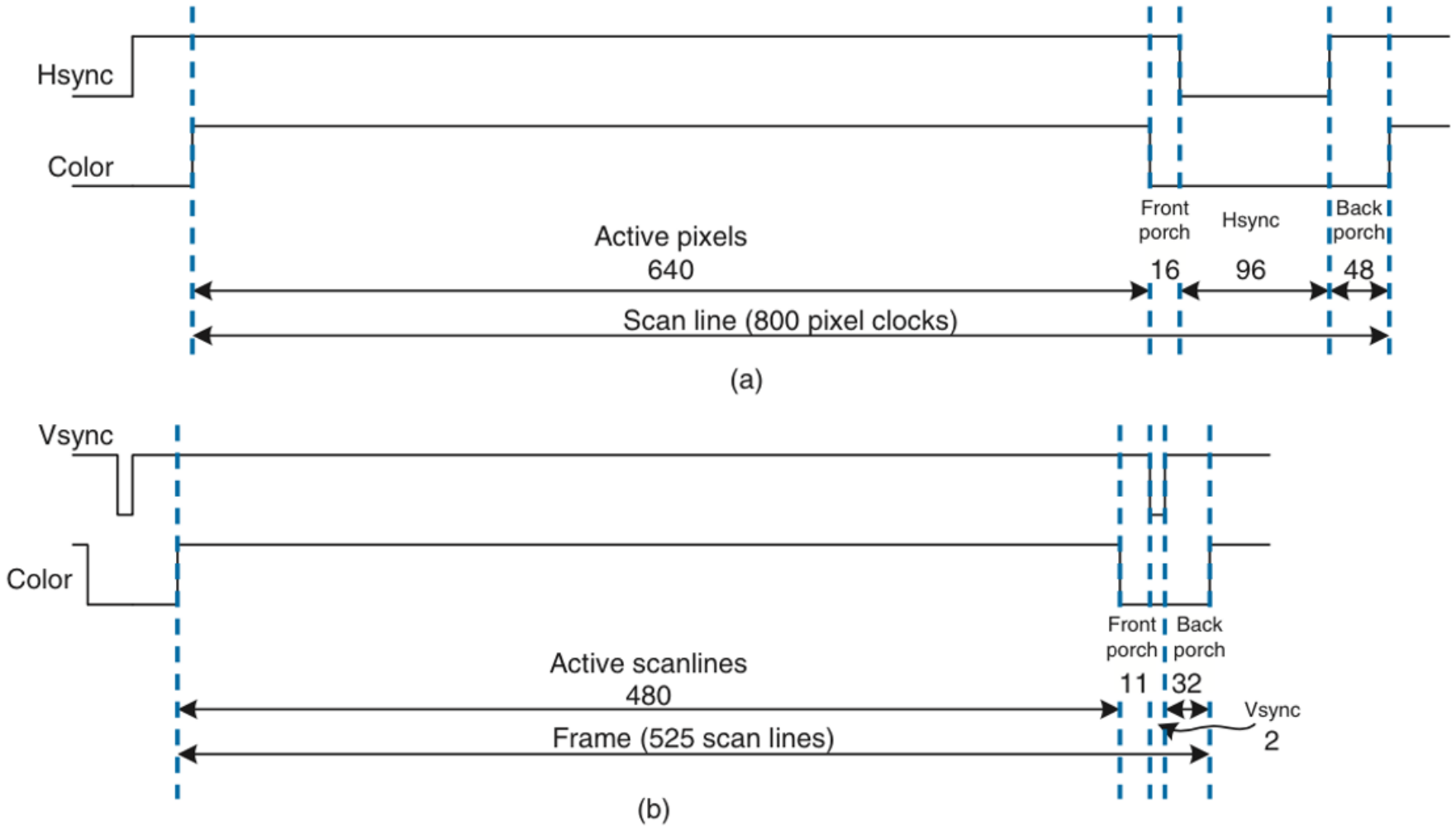



**Figure e9.26 VGA timing: (a) horizontal, (b) vertical**

# Timing

Pixel clock 25.175 MHz

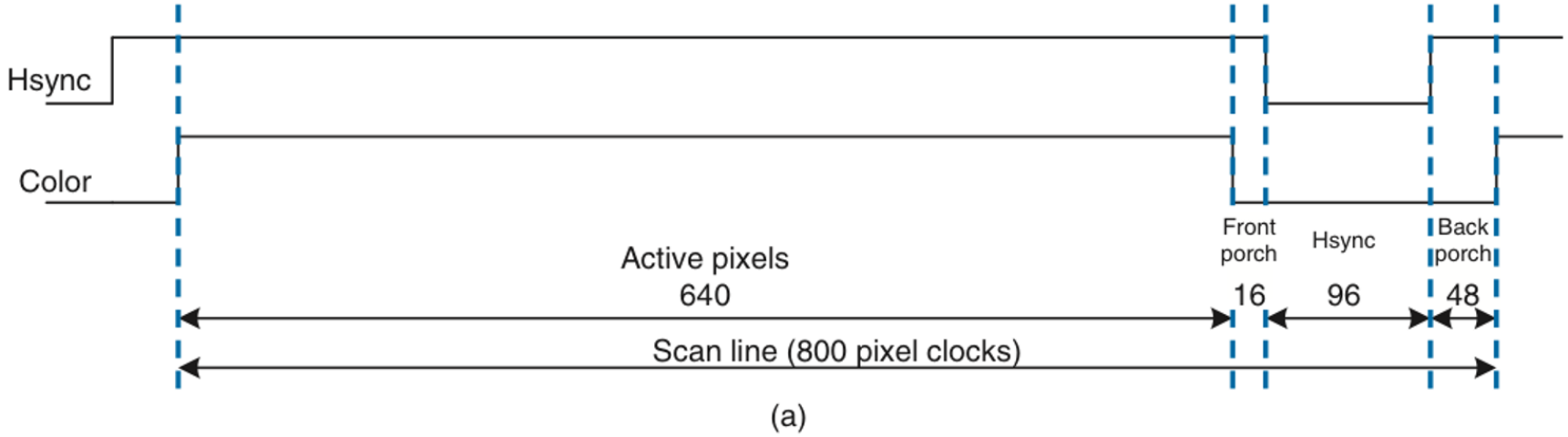
This allows enough time for 800 x 525 pixels (640 x 480 + sync)

- HSync: Clock / 800 = 31.47 KHz (31.77 ms scanline time)
- Vsync = Hsync / 525 = 59.94 Hz (about 60 Hz refresh rate)

# Horizontal Timing

Horizontal Timing: 31.47 KHz = 31.77 µsec

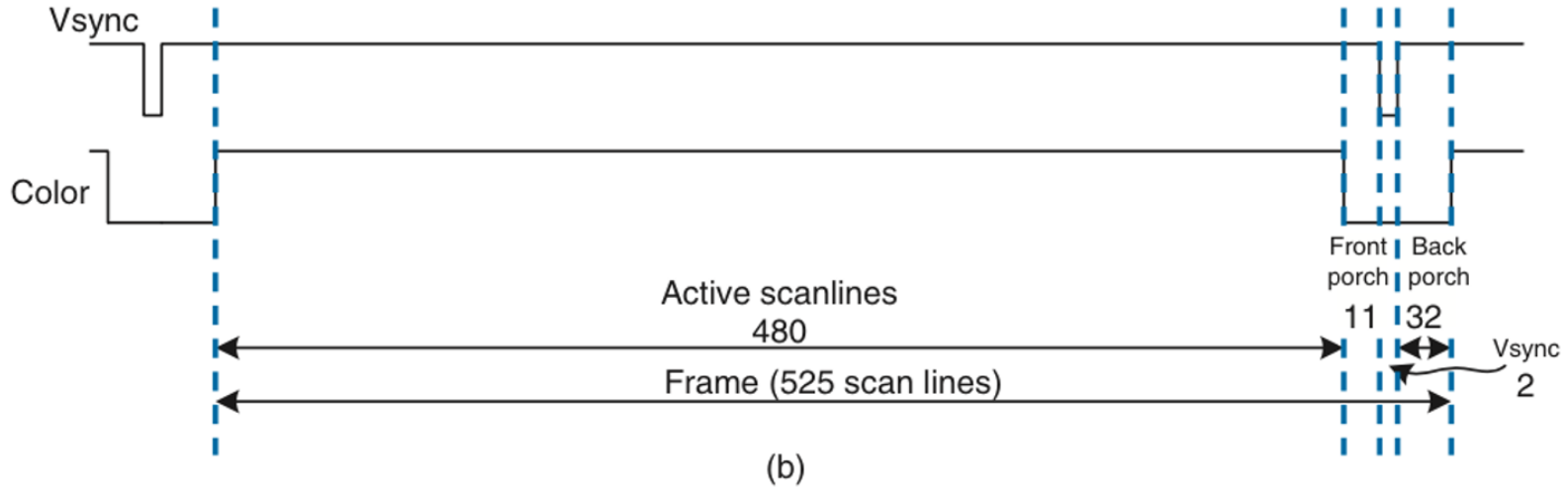
- Hsync goes low for first 3.81 usec (96 pixel clocks)
- Then 1.91 us back porch with no video signal (48 pixels)
- Then 25.42 us active video time (640 pixels)
- Then 0.64 us front porch (16 pixels)



# Vertical Timing

Total frame time is 1/60 Hz = 16.7 ms

- Vertical sync goes low for 60 µsec (two lines)
- Then back porch is 1.02 ms (32 lines)
- Then do the 480 lines of video
- Then front porch of 0.35 ms (11 lines)
- Total of 525 lines



# Color Information

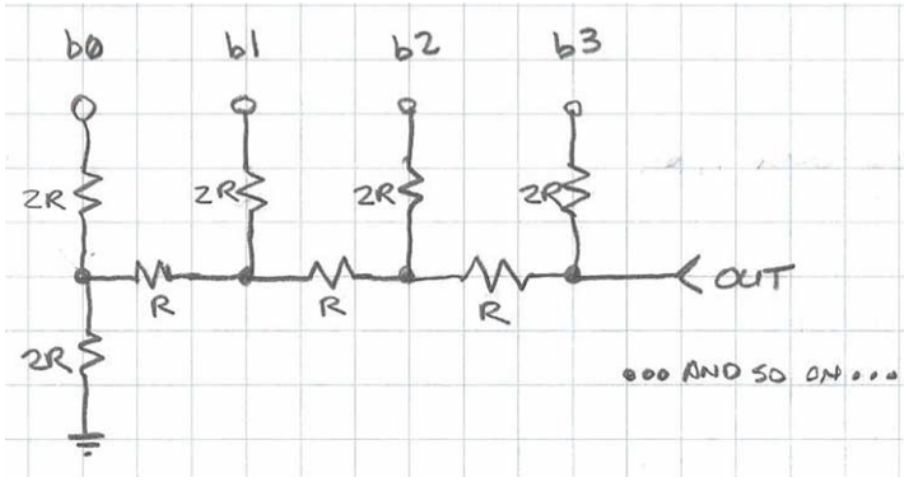
Three analog outputs for R, G, B

On a 0 – 0.7 V scale (more positive is brighter)

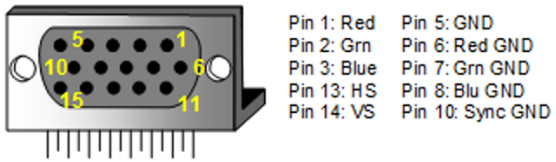
Pixel values should be 0 during front and back porch of horizontal and vertical.

# Cheap DAC

Using 75Ω termination resistance of the VGA display to create 4-bit analog values




https://www.tek.com/blog/tutorial-digital-analog-conversion-r-2r-dac



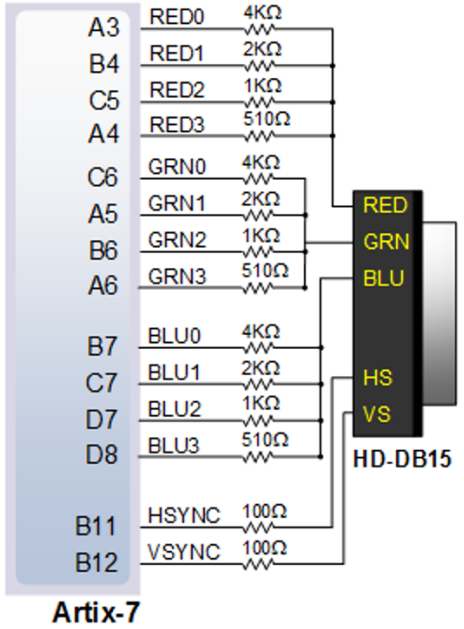




*Figure 11. Nexys4 DDR VGA interface.*

https://digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual

# Clock Generation

Need a 25.175 MHz pixel clock

- Use a square wave from an expensive signal generator
- If your FPGA is clocked fast enough, use a clock divider
- Use a PLL to create the clock on an FPGA

Pixel generation

- Use a set of counters running at the pixel clock rate At 25.175 MHz, one counts to 800 and the other to 525

# HDL

```
module vga(input  logic clk, reset,
           output logic vgaclk,          // 25.175 MHz VGA clock
           output logic hsync, vsync,
           output logic sync_b, blank_b, // to monitor & DAC
           output logic [7:0] r, g, b);  // to video DAC

  logic [9:0] x, y;

  // Use a PLL to create the 25.175 MHz VGA pixel clock
  // 25.175 MHz clk period = 39.772 ns
  // Screen is 800 clocks wide by 525 tall, but only 640 x 480 used
  // HSync = 1/(39.772 ns *800) = 31.470 kHz
  // Vsync = 31.474 kHz / 525 = 59.94 Hz (~60 Hz refresh rate)
  pll vgapll(.inclk0(clk), .c0(vgaclk));

  // generate monitor timing signals
  vgaController vgaCont(vgaclk, reset, hsync, vsync, sync_b, blank_b, x, y);

  // user-defined module to determine pixel color
  videoGen videoGen(x, y, r, g, b);

endmodule
```

```
module vgaController
#(parameter HBP     = 10'd48,   // horizontal back porch
            HACTIVE = 10'd640,  // number of pixels per line
            HFP     = 10'd16,   // horizontal front porch
            HSYN    = 10'd96,   // horizontal sync pulse = 60 to move electron gun back to left
            HMAX    = HBP + HACTIVE + HFP + HSYN, //48+640+16+96=800: number of horizontal pixels
            VBP     = 10'd32,   // vertical back porch
            VACTIVE = 10'd480,  // number of lines
            VFP     = 10'd11,   // vertical front porch
            VSYN    = 10'd2,    // vertical sync pulse = 2 to move electron gun back to top
            VMAX    = VBP + VACTIVE + VFP  + VSYN) //32+480+11+2=525: number of vertical pixels

     (input  logic vgaclk, reset,
      output logic hsync, vsync, sync_b, blank_b,
      output logic [9:0] hcnt, vcnt);

 // counters for horizontal and vertical positions
      always @(posedge vgaclk, posedge reset) begin
        if (reset) begin
          hcnt <= 0;
          vcnt <= 0;
        end
        else  begin
          hcnt++;
           if (hcnt == HMAX) begin
```

```
 // compute sync signals (active low)
      assign hsync  = ~( (hcnt >= (HACTIVE + HFP)) & (hcnt < (HACTIVE + HFP + HSYN)) );
      assign vsync  = ~( (vcnt >= (VACTIVE + VFP)) & (vcnt < (VACTIVE + VFP + VSYN)) );
      // assign sync_b = hsync & vsync;
      assign sync_b = 1'b0;  // this should be 0 for newer monitors

      // force outputs to black when not writing pixels
      // The following also works: assign blank_b = hsync & vsync;
      assign blank_b = (hcnt < HACTIVE) & (vcnt < VACTIVE);
endmodule
```

```
module rectgen(input  logic [9:0] x, y, left, top, right, bot,
               output logic inrect);

  assign inrect = (x >= left & x < right & y >= top & y < bot);

endmodule
```

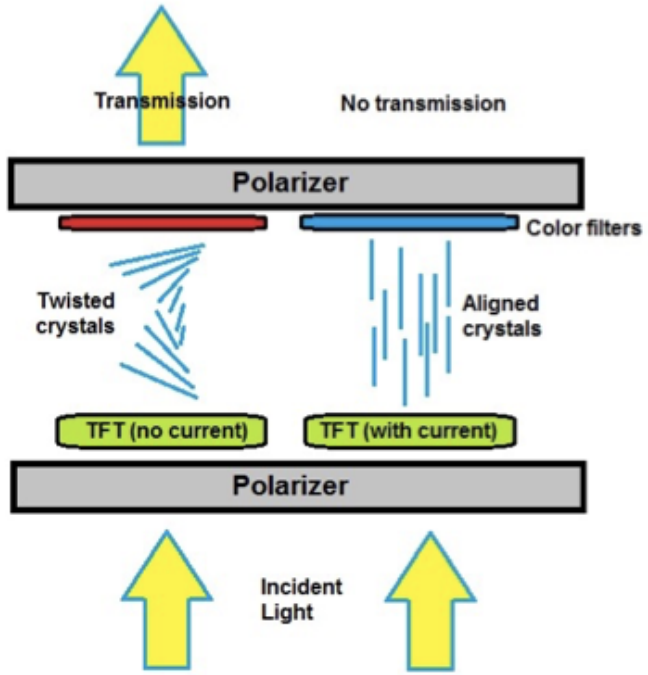
```
module videoGen(input logic [9:0] x, y, output logic [7:0] r, g, b);

  logic pixel, inrect;

  // given y position, choose a character to display
  // then look up the pixel value from the character ROM
  // and display it in red or blue. Also draw a green rectangle.
  chargenrom chargenromb(y[8:3]+8'd65, x[2:0], y[2:0], pixel);
  rectgen rectgen(x, y, 10'd120, 10'd150, 10'd200, 10'd230, inrect);

  assign {r, b} = (y[3]==0) ? {{8{pixel}},8'h00} : {8'h00, {8{pixel}}};
  assign g      = inrect    ? 8'hFF : 8'h00;

endmodule
```

```
module chargenrom(input  logic [7:0] ch,
                  input  logic [2:0] xoff, yoff,
                  output logic       pixel);

  logic [5:0] charrom[2047:0]; // character generator ROM
  logic [7:0] line;            // a line read from the ROM

  // initialize ROM with characters from text file
  initial $readmemb("charrom.txt", charrom);

  // index into ROM to find line of character
  assign line = charrom[yoff+{ch-65, 3'b000}];  // subtract 65 because A
                                                // is entry 0
  // reverse order of bits
  assign pixel = line[3'd7-xoff];

endmodule
```

# Liquid Crystal Display (LCD) Technology

# Liquid Crystal Display (LCD)

- Liquid crystals under electric field used to change the polarization of light traveling through it.
- When not energized, light is transmitted through cross polarizers
- When energized, light does not pass through crossed polarizers

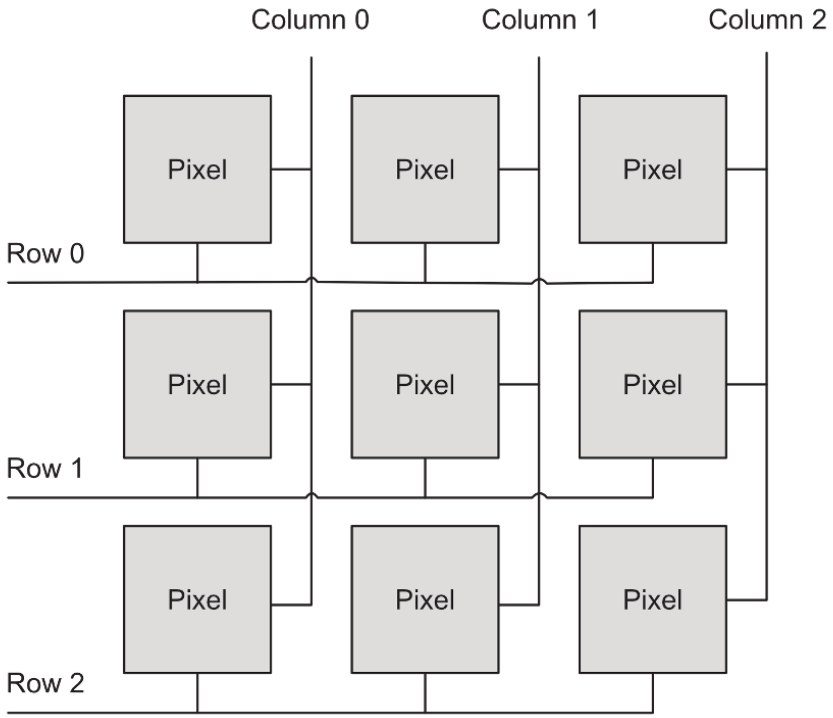



Individual pixel "Diagram of liquid-crystal display" by Spokoyny under CC BY-SA 4.0.

# LCD Matrix

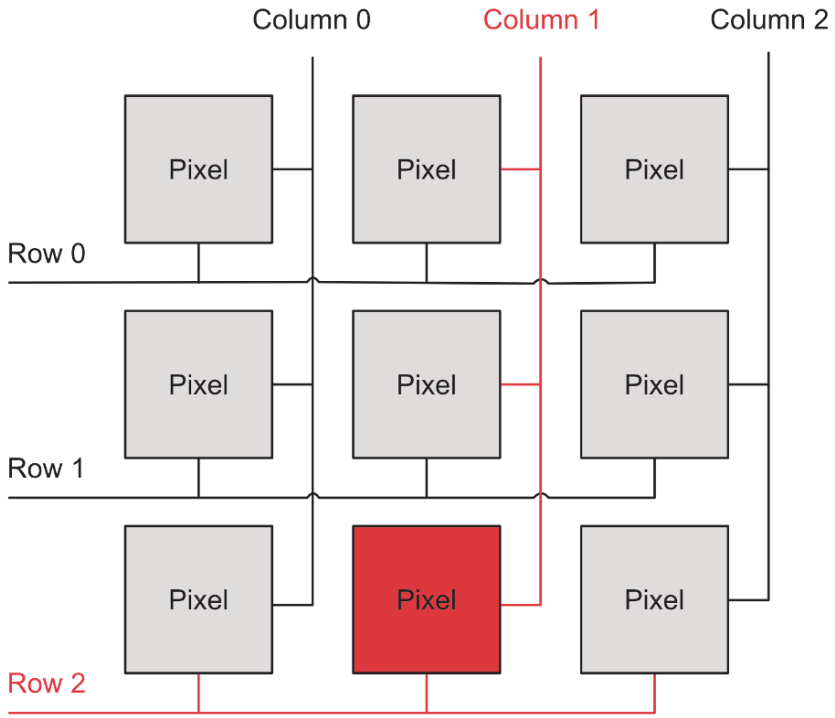
Pixels are arranged in a grid.

All pixels on a row and column are connected.



# Selecting Pixel in LCD

- To select a given pixel, choose its row and column
- *Passive* matrices use liquid crystal for each pixel.
- *Active* matrices add thin-film transistor to actively maintain its state while other pixels are addressed



# LCD Timing

Same ideas as in VGA, but now we are just dealing with selecting individual pixels instead of scanning electron beam.

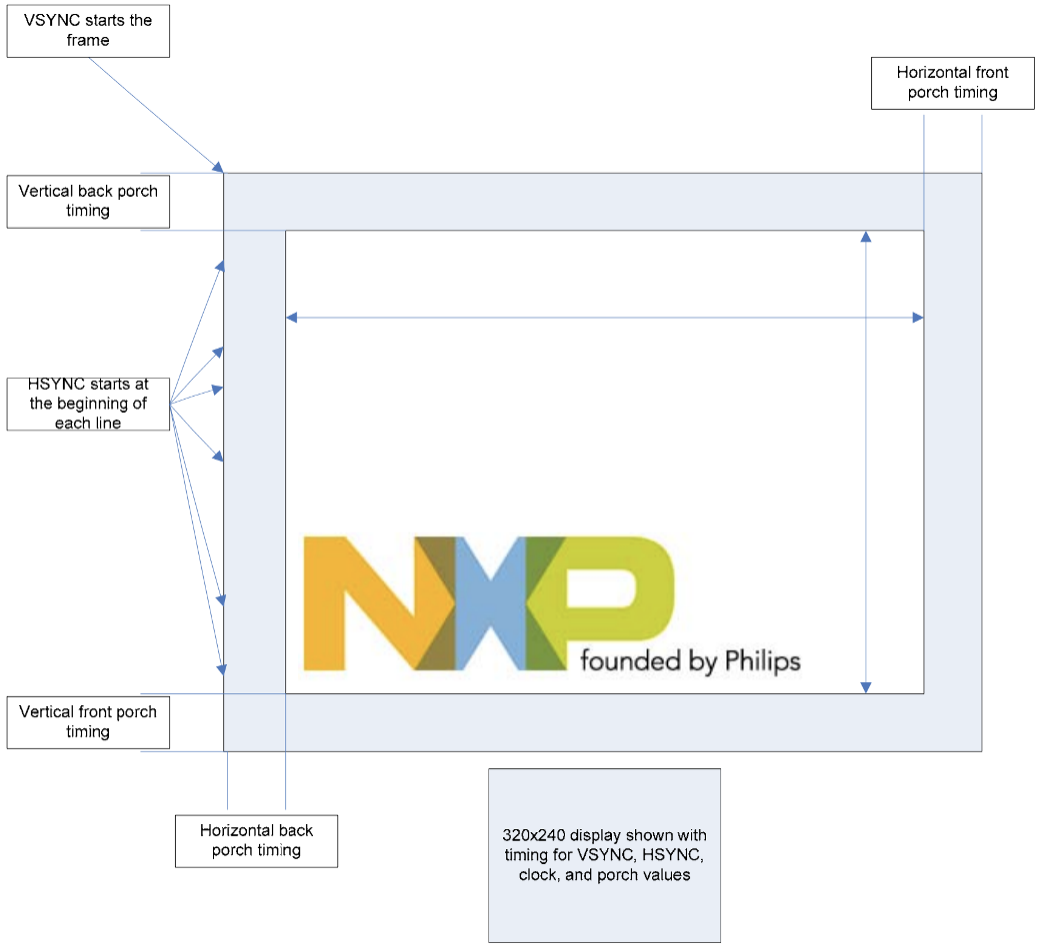
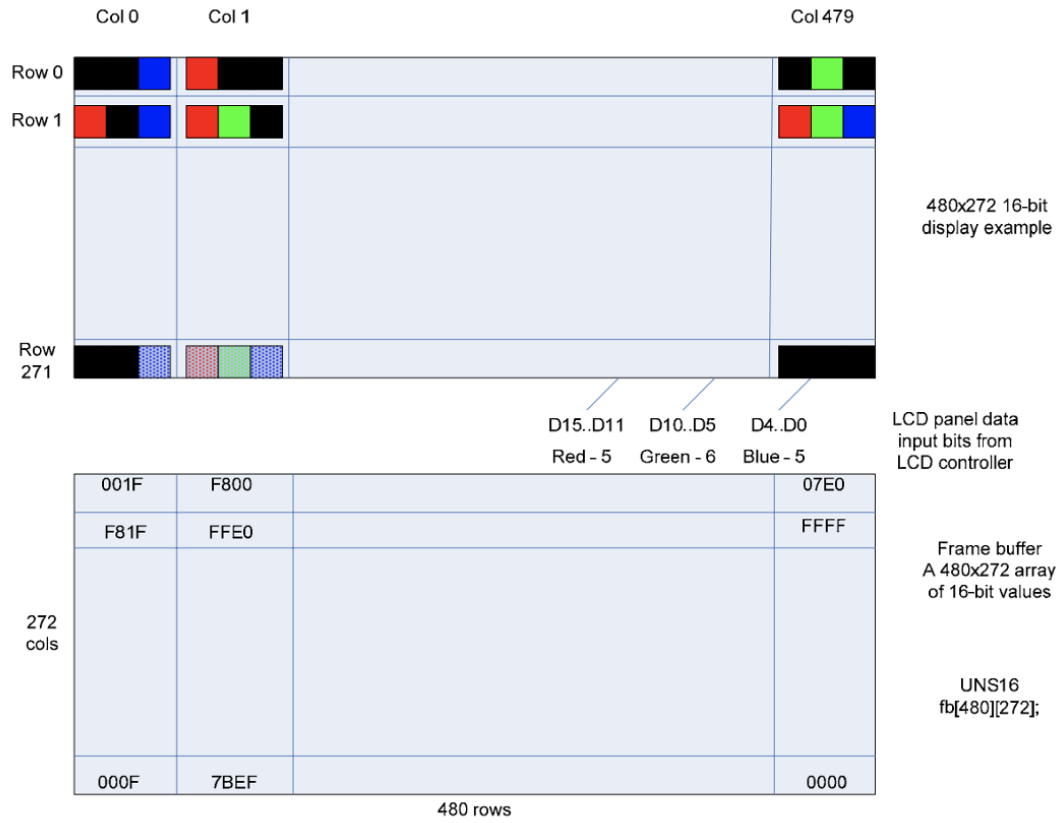



Figure from Introduction to graphics and LCD technologies

# Frame Buffer

- The frame buffer is an array of memory used to store the data to be displayed
- Double buffered displays have two buffers so that the display can be updated without directly writing to the display.

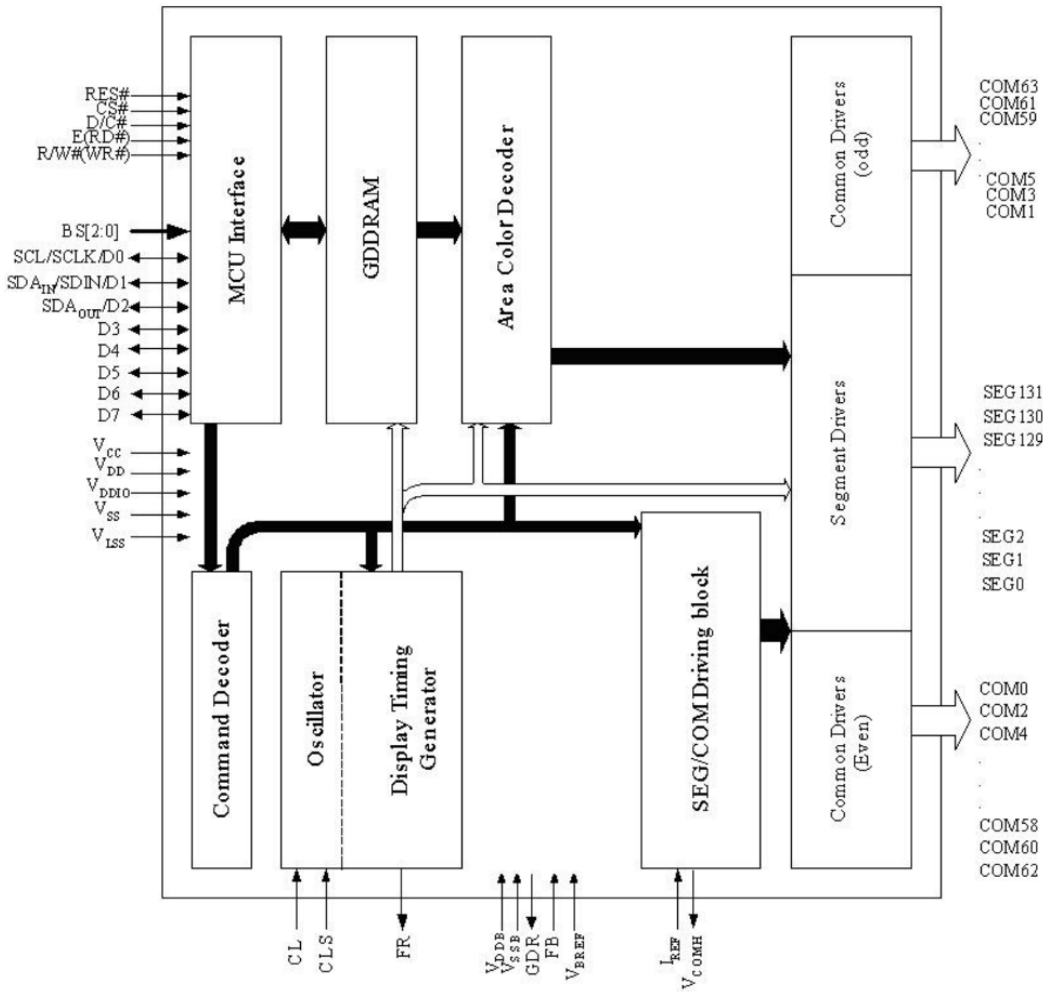



Frame Buffer 16bpp from Introduction to graphics and LCD technologies

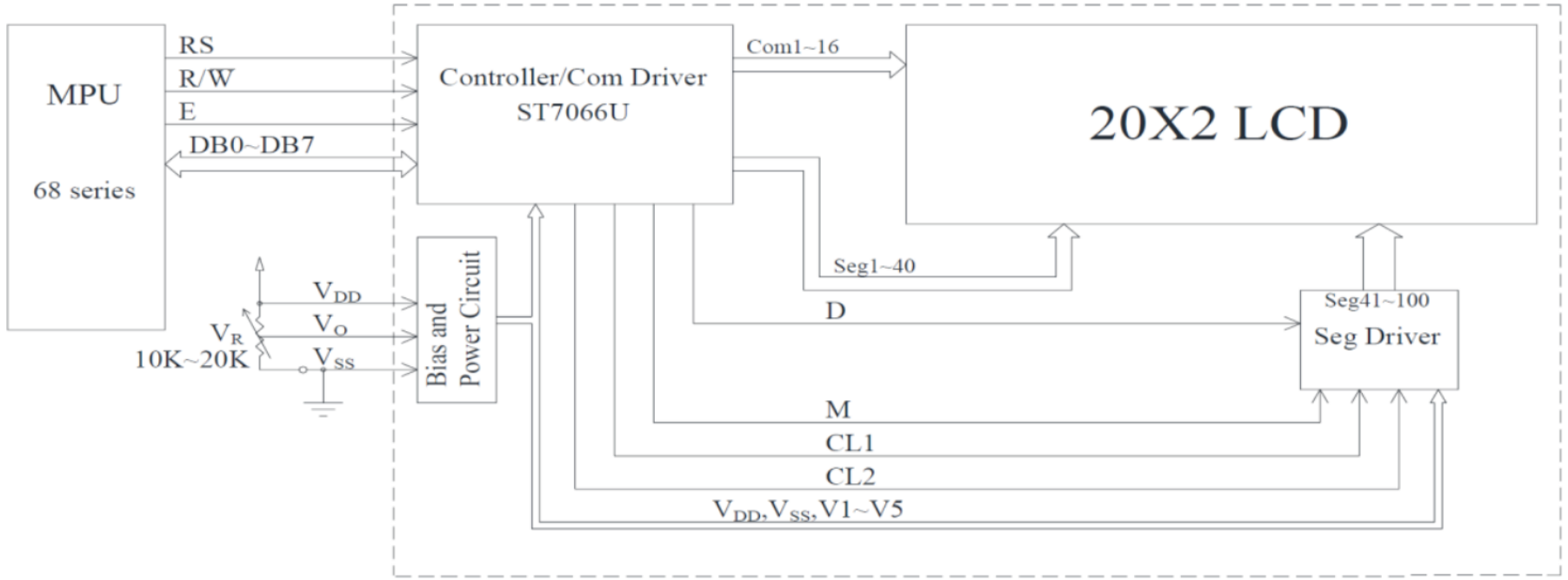
# SSD1305 Driver

Time multiplexing to drive array

Figure 4-1 : SSD1305 Block Diagram



# Character LCD


CFAH2002A TMI JTDatasheet

| Pin No. | Symbol | Level | Function |
|---|---|---|---|
| 1 | $V_{SS}$ | 0v | Ground |
| 2 | $V_{DD}$ | 5.0v | Supply Voltage for Logic |
| 3 | $V_O$ | (variable) | Supply Voltage for LCD |
| 4 | RS | H/L | H: Data<br>L: Instruction Code |
| 5 | R/W | H/L | H: Read<br>L: Write |
| 6 | E | H, H ⟶ L | Chip Enable Signal |
| 7 | DB0 | H/L | Data Bus Line |
| 8 | DB1 | H/L | Data Bus Line |
| 9 | DB2 | H/L | Data Bus Line |
| 10 | DB3 | H/L | Data Bus Line |
| 11 | DB4 | H/L | Data Bus Line |
| 12 | DB5 | H/L | Data Bus Line |
| 13 | DB6 | H/L | Data Bus Line |
| 14 | DB7 | H/L | Data Bus Line |
| 15 | A | - | Power Supply for B/L (+) |
| 16 | K | - | Power Supply for B/L (-) |

**Example e9.10** LCD CONTROL

Write a program to print "I love LCDs" to a character display.

**Solution:** The following program writes "I love LCDs" to the display by initializing the display and then sending the characters.

```
#include "EasyPIO.h"

int LCD_IO_Pins[] = {14, 15, 4, 17, 22, 24, 9, 11};

typedef enum {INSTR, DATA} mode;
#define RS 7
#define RW 8
#define E  25

char lcdRead(mode md) {
    char c;
    pinsMode(LCD_IO_Pins, 8, INPUT);
    digitalWrite(RS,(md == DATA));      // Set instr/data mode
    digitalWrite(RW, 1);                // Read mode
    digitalWrite(E, 1);                 // Pulse enable
    delayMicros(10);                    // Wait for LCD response
    c = digitalReads(LCD_IO_Pins, 8);   // Read a byte from parallel port
    digitalWrite(E, 0);                 // Turn off enable
    delayMicros(10);
    return c;
}
```

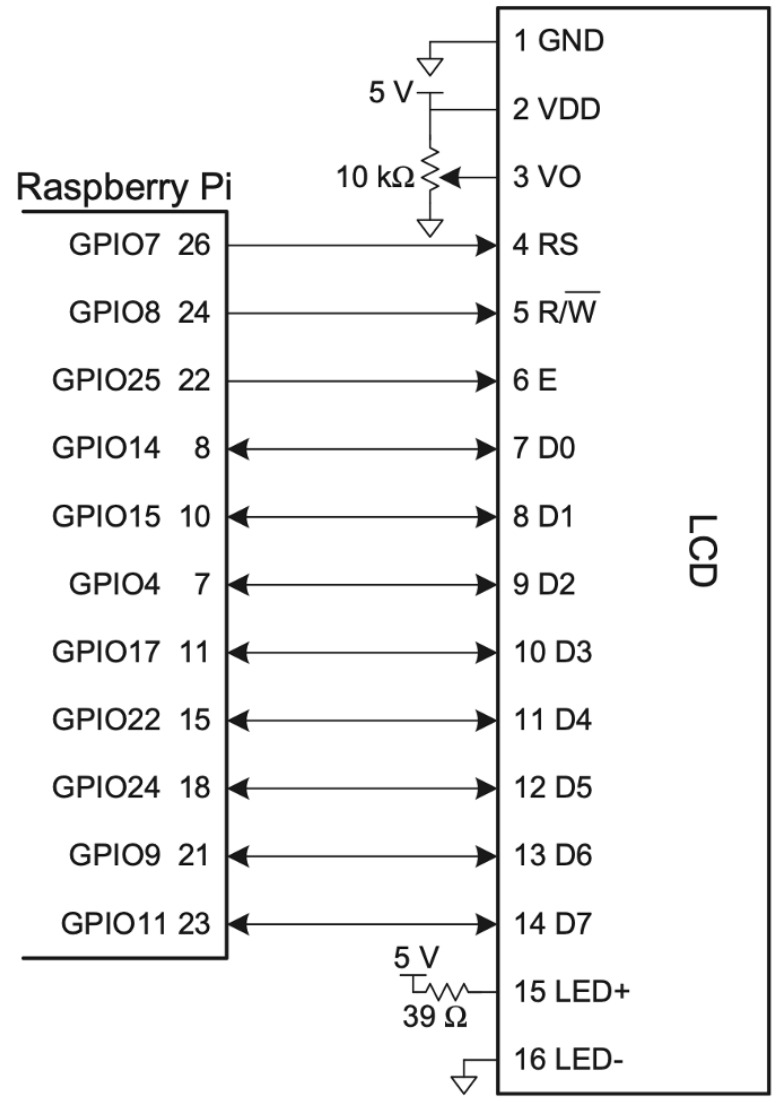


**Table e9.8 LCD initialization sequence**

| Write | Purpose | Wait (µs) |
|---|---|---|
| (apply $V_{DD}$) | Allow device to turn on | 15000 |
| 0x30 | Set 8-bit mode | 4100 |
| 0x30 | Set 8-bit mode again | 100 |
| 0x30 | Set 8-bit mode yet again | Until busy flag is clear |
| 0x3C | Set 2 lines and 5×8 dot font | Until busy flag is clear |
| 0x08 | Turn display OFF | Until busy flag is clear |
| 0x01 | Clear display | 1530 |
| 0x06 | Set entry mode to increment cursor after each character | Until busy flag is clear |
| 0x0C | Turn display ON with no cursor | |

```
void lcdBusyWait(void) {
    char state;
    do {
        state = lcdRead(INSTR);
    } while (state & 0x80);
}

void lcdWrite(char val, mode md) {
    pinsMode(LCD_IO_Pins, 8, OUTPUT);
    digitalWrite(RS, (md == DATA));      // Set instr/data mode. OUTPUT=1, INPUT=0
    digitalWrite(RW, 0);                 // Set RW pin to write  (aka: 0)
    digitalWrites(LCD_IO_Pins, 8, val);  // Write the char to the parallel port
    digitalWrite(E, 1); delayMicros(10); // Pulse E
    digitalWrite(E, 0); delayMicros(10);
}

void lcdClear(void) {
    lcdWrite(0x01, INSTR); delayMicros(1530);
}
```

# Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI)

# Why Use DVI?

LCDs are inherently digital and converting from digital to analog and then back again wastes time, energy, and reduces fidelity.

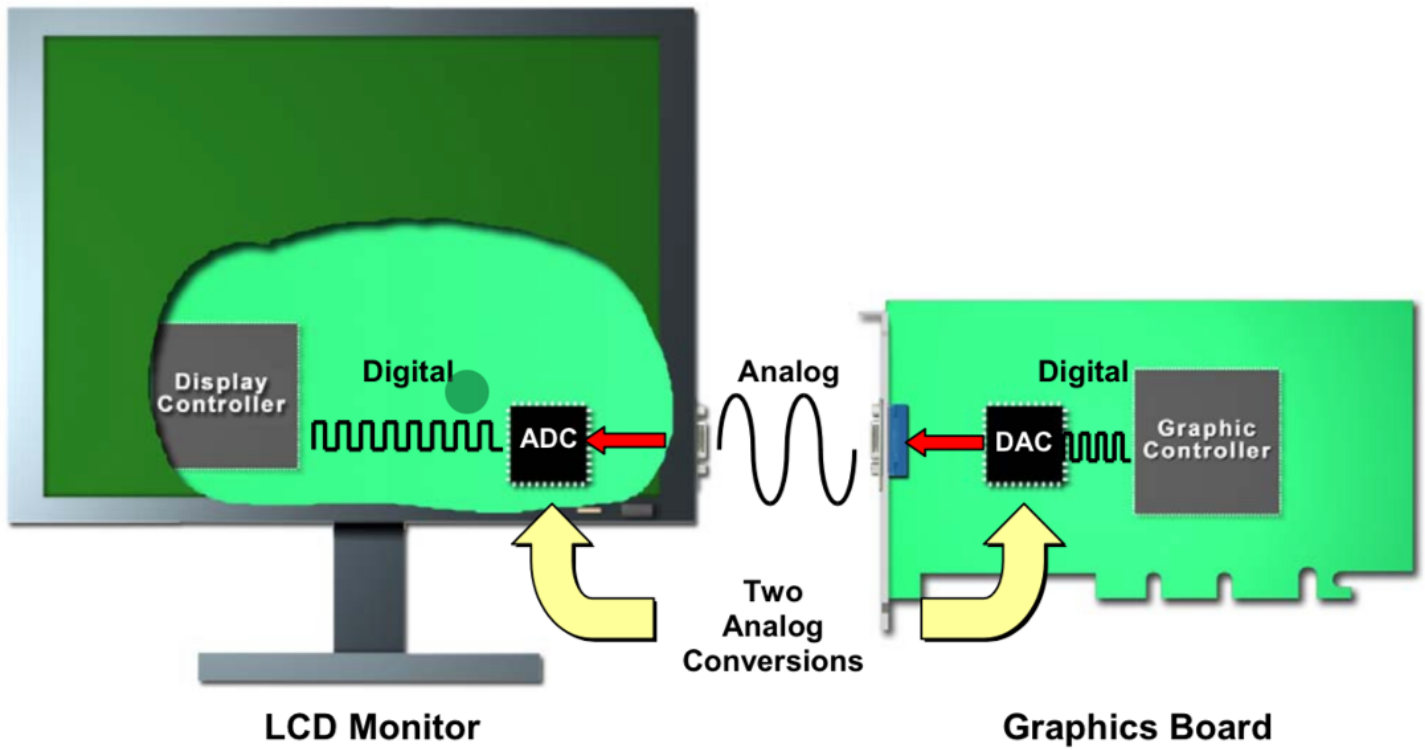



**Figure 2: Analog Conversions**

Figure from DVI and TMDS Extensions - Silicon Image White Paper.

# DVI History

- Developed in 1998 by the Digital Display Working Group (DDWG). Composed of Fujitsu, Compaq, HP, IBM, Intel, NEC, and Silicon Image
- DVI 1.0 spec released in April 1999
- Transition minimized DC-balanced signaling (TMDS) is key enabling cost-effective digital data at high rate and also allowing bandwidth to be doubled with a second link.

# Why not use Low-Voltage Differential Signaling (LVDS)?

- LVDS's speed (and thus resolution) limited by cable length. DVI supports up to 15 meter cable length.
- No universal connector solution
- LVDS can only support up to QXGA (2048 x 1536)

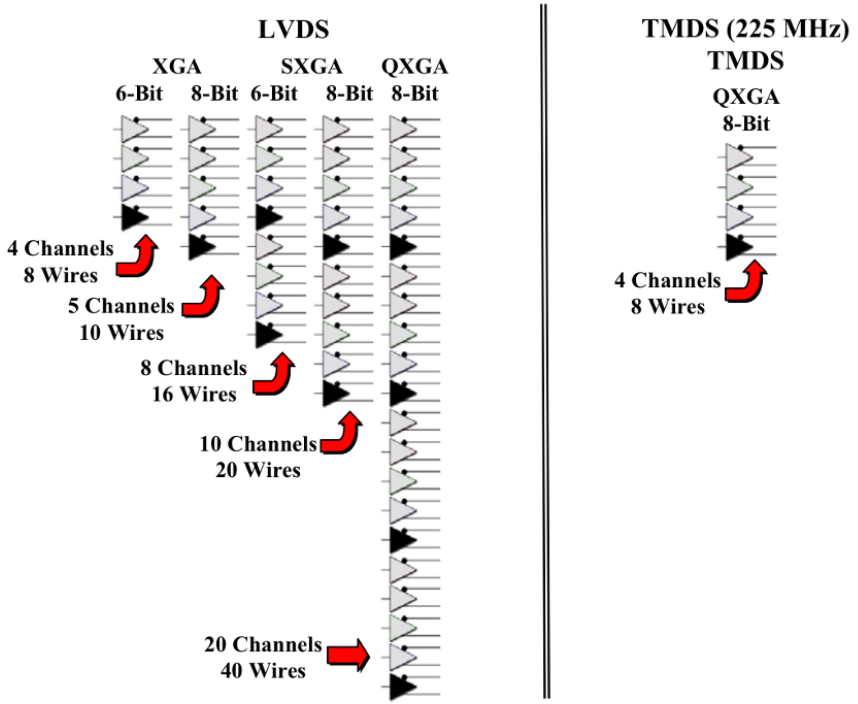



Figure 1: LVDS vs. TMDS

DVI and TMDS Extensions - Silicon Image White Paper

# DVI Components

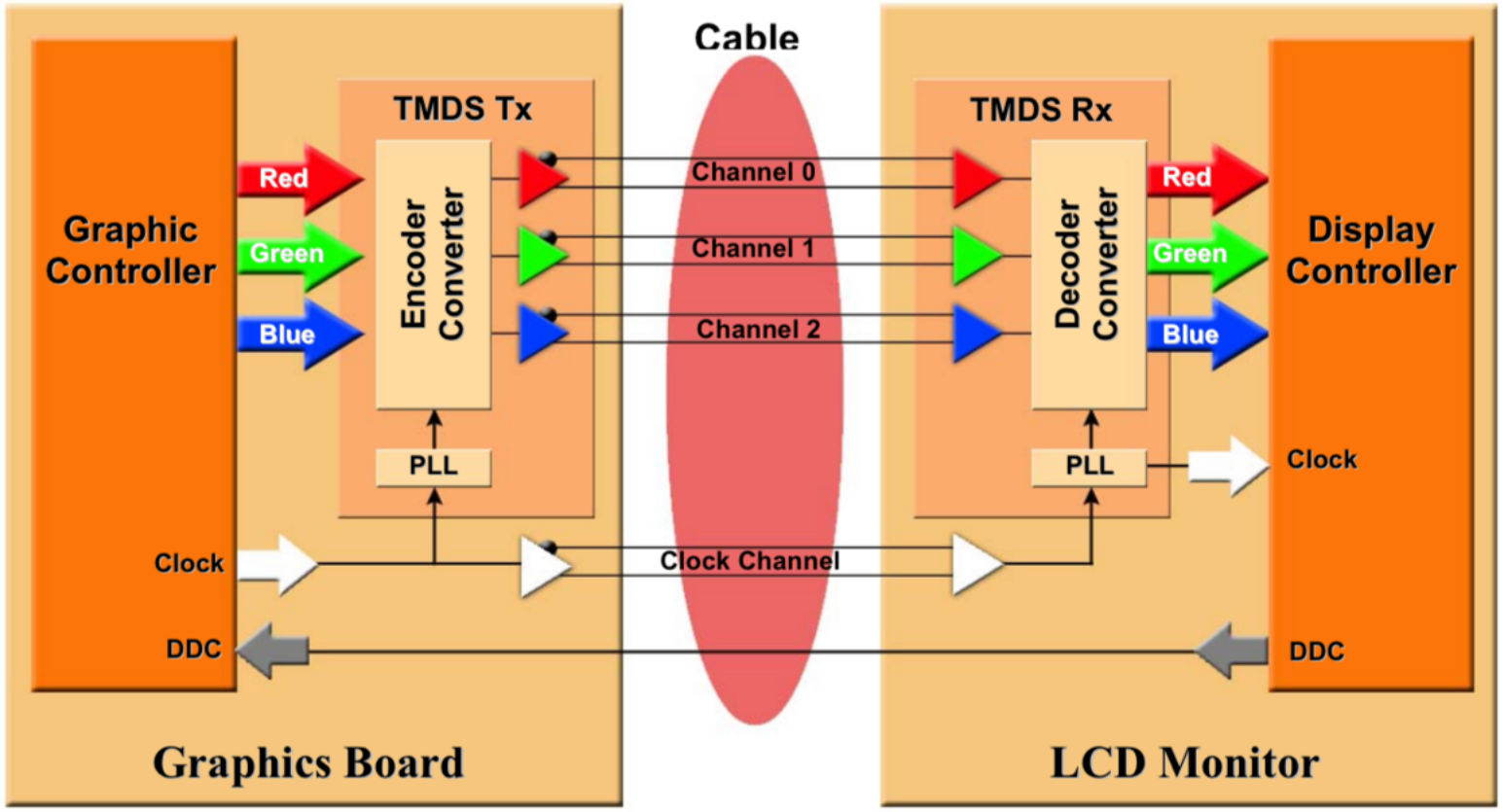



**Figure 5: Single-Link DVI**

Figure from DVI and TMDS Extensions - Silicon Image White Paper

# DVI Components

- TMDS transmitter
  - Prepares 24 bits of parallel data (8 bits for each color channel) for serial transmission by encoding and serializing it
  - 4 channels: clock, R, G, and B.
- TMDS receiver
  - Converts from serial data stream to parallel output
- DVI connector
- DVI cable

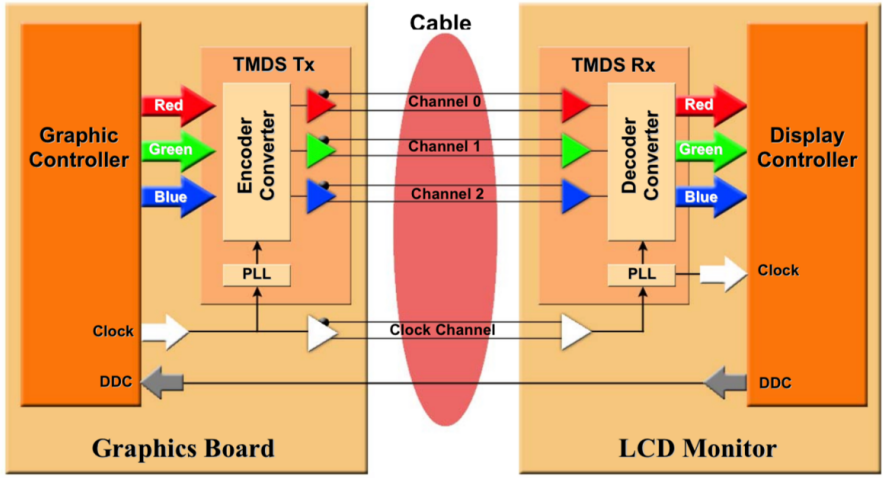



Figure 5: Single-Link DVI

Figure from DVI and TMDS Extensions - Silicon Image White Paper.

# Video Signal

- Uses transition-minimized differential signaling
- TMDS = transition-minimized differential signaling
- 4 TMDS differential pairs of interest
  - clock +/-
  - data0 +/-
  - data1 +/-
  - data2 +/-
- Data lines used for RGB color signals

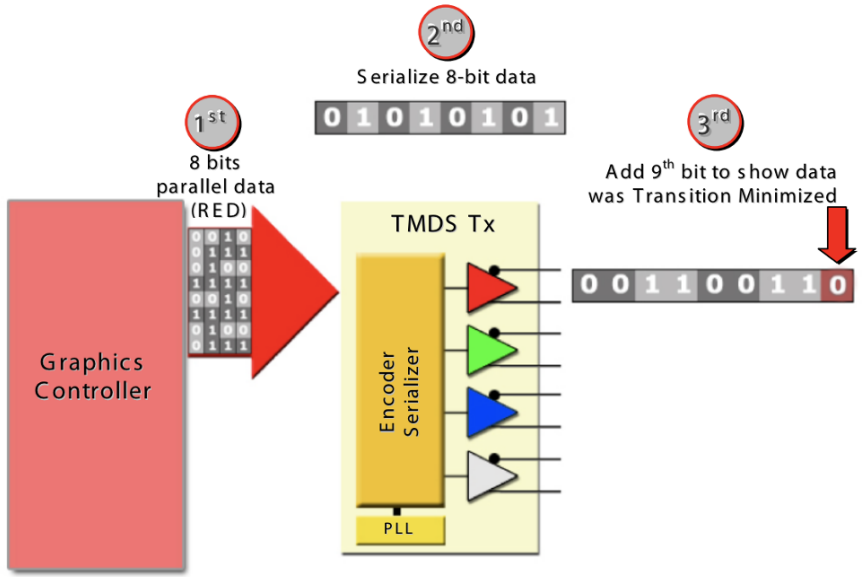
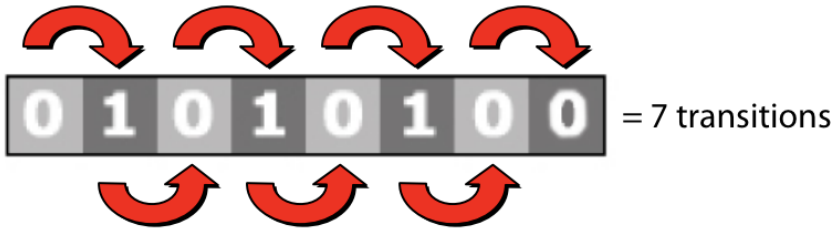



Figure 7: TMDS Transmitter



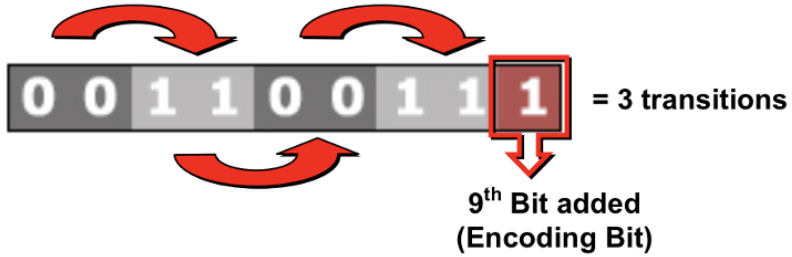




Figure 9: Minimizing Transitions

# TMDS Signal: DC-balancing

- Use two lines instead of one where they are opposites of each other
- Common mode noise is rejected


Figure 16: Differential Signal


Figure 17: Differential Signal (Two Wires)

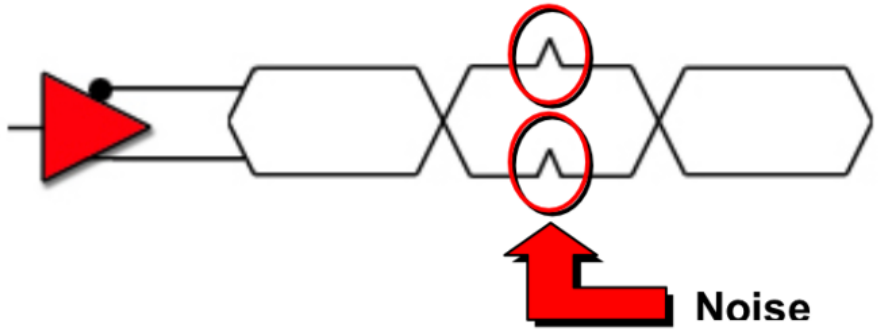



Figure 18: Noise on Line

Figures from DVI and TMDS Extensions - Silicon Image White Paper.

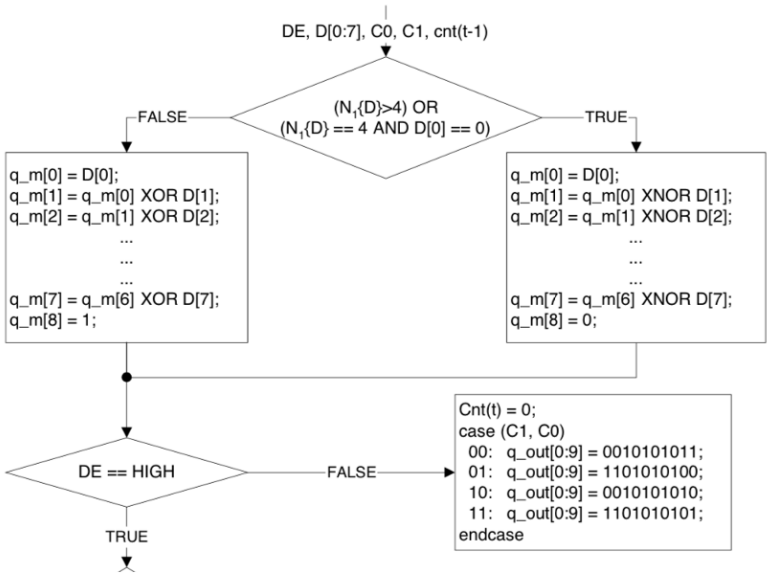
# TMDS Algorithm

| | |
|---|---|
| D, C0, C1, DE | The encoder input data set. D is eight-bit pixel data, C1 and C0 are the control data for the channel, and DE is data enable |
| cnt | This is a register used to keep track of the data stream disparity. A positive value represents the excess number of "1"s that have been transmitted. A negative value represents the excess number of "0"s that have been transmitted. The expression cnt{t-1} indicates the previous value of the disparity for the previous set of input data. The expression cnt(t) indicates the new disparity setting for the current set of input data. |
| q_out | These 10 bits are the encoded output value. |
| $N_1\{x\}$ | This operator returns the number of "1"s in argument "x" |
| $N_0\{x\}$ | This operator returns the number of "0"s in argument "x" |

*Table 3-1 Encoding Algorithm Definitions*

Algorithm details from Digital Visual Interface DVI Rev. 1.0 from Digital Display Working Group

# TMDS Algorithm



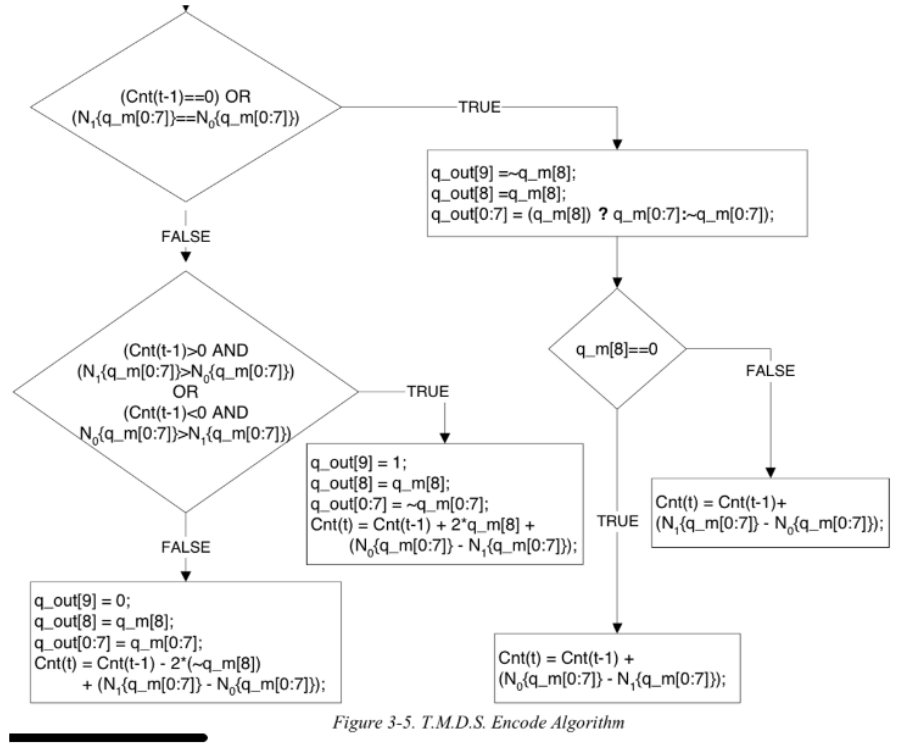




*Figure 3-5. T.M.D.S. Encode Algorithm*

Algorithm details from Digital Visual Interface DVI Rev. 1.0 from Digital Display Working Group

# DVI Pinout

DVI pinout contains three main pieces

- TMDS signals
- Plug & Play signals
- Analog signals

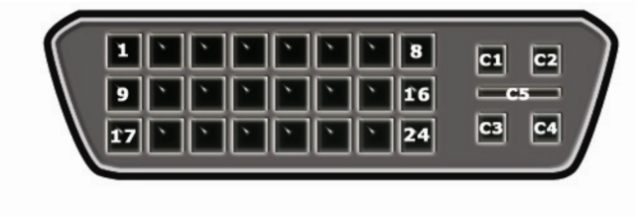



Figure 19: DVI-I Signal Pins

| Pin | Signal | Pin | Signal | Pin | Signal |
|---|---|---|---|---|---|
| 1 | Data 2 - | 9 | Data 1 - | 17 | Data 0 - |
| 2 | Data 2 + | 10 | Data 1 + | 18 | Data 0 + |
| 3 | Shield (2 & 4) | 11 | Shield (1 & 3) | 19 | Shield (0 & 5) |
| 4 | Data 4 - | 12 | Data 3 - | 20 | Data 5 - |
| 5 | Data 4 + | 13 | Data 3 + | 21 | Data 5 + |
| 6 | Clock DDC | 14 | Power +5V | 22 | Shield Clock |
| 7 | Data DDC | 15 | Ground | 23 | Clock + |
| 8 | Analog Vertical Sync | 16 | Hot Plug | 24 | Clock - |
| C1 | Analog Red | | | | |
| C2 | Analog Green | | | | |
| C3 | Analog Blue | | | | |
| C4 | Analog Horizontal Sync | | | | |
| C5 | Analog Ground | | | | |

TMDS
PLUG & PLAY
ANALOG

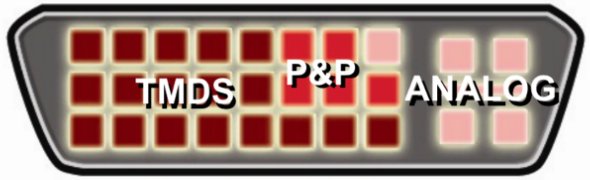



Figure 20: TMDS, Plug & Play and Analog Signals

# HDMI is just DVI+

- 4 TMDS signals
- Display Data Channel (DDC) two way communication including - HDCP signal
- CEC data line
- Hot Plug Detection (HPD)
- +5V power

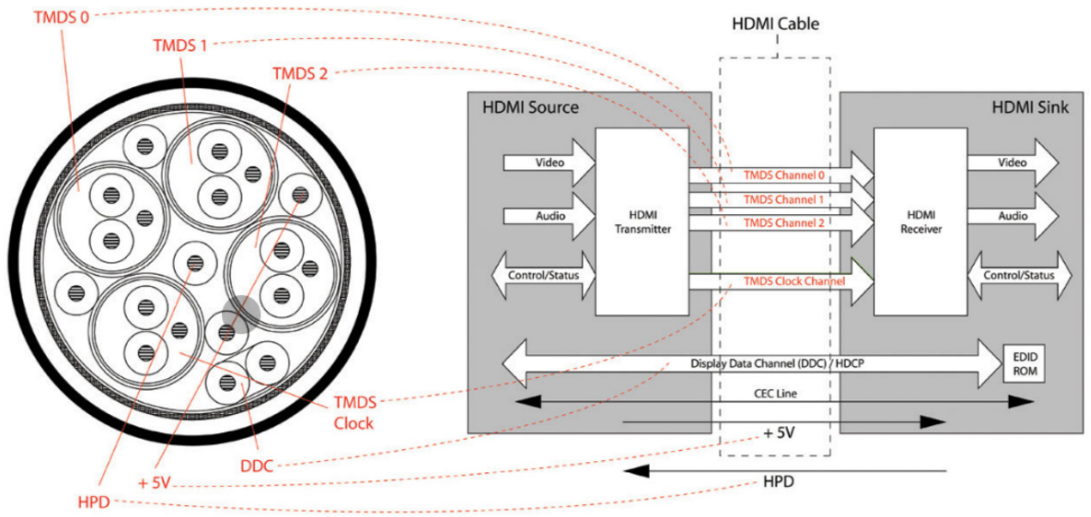



Fig.1 Components inside a HDMI Cable

Figure from HDMI Demystified

# Summary

- VGA was one of the first major graphics display technologies developed but was optimal for analog displays like CRTs
- Digital displays like LCDs are better served by a new interface
- Pixels are addressed in row-column array and time-multiplexed
- Active displays enable each pixel’s current value to be stored while others are being updated
- Digital Visual Interface (DVI) is a standard display technology for many devices today (HDMI is built on top of the key technology behind it)
  - Transition-minimized DC-balanced signaling (TMDS)

# Resources and Further Reading

- HDMI from fpga4fun.com
  - DVI and TMDS Extensions - Silicon Image White Paper
  - HDMI Demystified
  - Digital Visual Interface DVI Rev. 1.0 from Digital Display Working Group
- HDMI Made Easy from Analog Devices