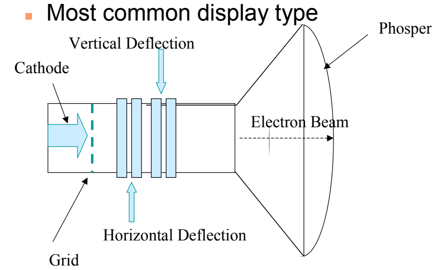
Display Technology

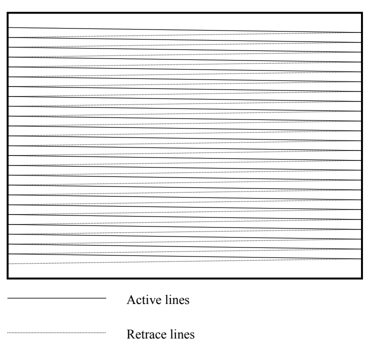
- Display Technology Tree:
  - CRT
  - Flat Panel
    - Active
      - Fluorescent
      - Gas Discharge (plasma)
      - Electroluminescent
      - LEDs
      - Incadescent
    - Passive
      - Liquid Crystal Displays (LCDs)
      - Electromechanical

Cathode Ray Tubes

- Most common display type



Raster Scan



Beam directed in lines (scan lines) across screen.

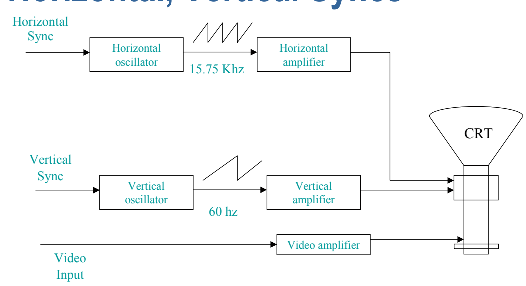
Dots are created by turning the beam on & off.

During *retrace*, the beam is off.

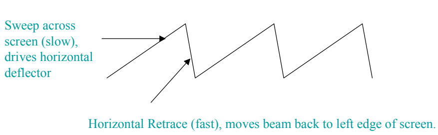
***Horizontal retrace* is when beam returns to left edge**

***Vertical retrace* is when beam returns to upper left corner.**

Horizontal, Vertical Syncs

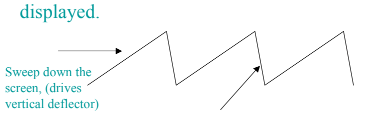


The horizontal sweep controls the number of lines on the screen



Horizontal Retrace (fast), moves beam back to left edge of screen.

The vertical sweep controls how fast each screen is displayed.



Vertical Retrace moves beam back to top left corner

How many lines on screen?

- Let Horizontal sweep = 15.75Khz
- Let Vertical sweep = 60 hz
  - Num lines = horizontal/vertical = 15750/60 = 262.5
- These numbers are for standard North America broadcast television
  - Half line allows next field (next 262.5 lines) to be offset by one half line (even & old fields). Two fields make one frame
  - Gives an interlaced display of 524 lines refreshed at 30 hz
    - Human eye can detect flicker at 45 Hz - don't notice flicker on TV because of image types

How many Dots on screen?

- Video input controls whether beam on or not
- How fast we can turn beam on/off during horizontal trace time
- Monitors use internal clock to sample video signal
  - Monochrome - only one line for video signal
  - Digital RGB - three digital lines (Red, Green, Blue)
    - **Gives 8 colors (a 4th line, an intensity signal, can be used to give 16 colors)**
  - Analog RGB - three analog lines, each driven by 8-bit DAC - gives 256 * 256 * 256 = 2^{24} colors

VGA Timing (640 dots x 480 lines)

Horizontal Sync = 31.5 Khz, Vertical Sync = 60 Hz

Internal Monitor clock (Dot Clock) for latching video signal is 25.175 Mhz

#max dots per line = Dot Clock Freq/ Horizontal Sync

= 25.175 Mhz / 31.5 Khz = 800 Dots

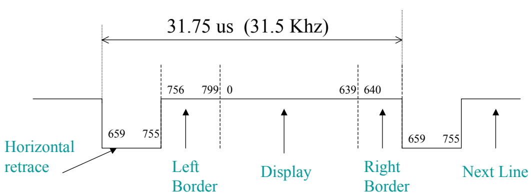
Only can use 640 dot times out of possible 800 for display because we need black areas on left/right edges and time for horizontal retrace.

#max lines per screen = Horizontal Sync/Vertical Sync

= 31.5 Khz / 60 hz = 525 lines

Only 480 lines usable, need blank areas on top/bottom, time for vertical retrace

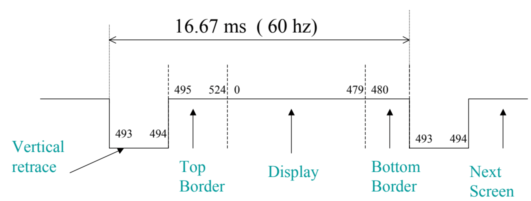
Horizontal Sync Timing



800 dot times per line

Counter can be used to keep track of horizontal screen position.

Vertical Sync Timing



525 line times per screen

Counter can be used to keep track of vertical screen position.

Other resolutions

- 800 x 600
  - Dot clock 36 Mhz
  - Horizontal Sync 35.15 Khz
  - Vertical Sync 56 hz
- 1024 x 768
  - Dot clock 64.142 Mhz
  - Horizontal Sync 48.3 Khz
  - Vertical Sync 60 hz
- Allow about 20% of horizontal trace time for borders, retrace
- 6% to 8% of vertical trace time for borders, retrace

What drives video signal?

- Every dot clock time for visible pixels, need to determine value of video signal
  - For monochrome, 1 bit per pixel (black or white, on or off)
  - For Digital RGB, 4 bits per pixel (R,G,B, Intensity)
  - For Analog RGB, N bits per R,G,B value where each RGB value can have 2^N distinct values
- The memory that defines the screen contents is called the display memory or frame buffer.

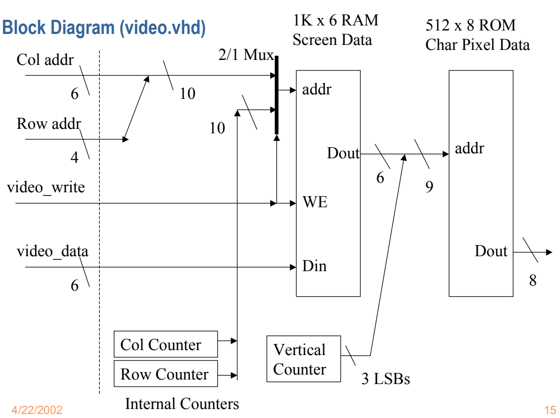
Display Memory Characteristics

- Accessed at high data rates
- Will need to accessed by two sources
  - CPU which will be doing read/writes to random locations
  - Video signal driver which will be reading memory locations in a fixed pattern (the scan pattern)
- Can use either SRAM, DRAM, SDRAM or specialty graphics memory to implement the graphics memory.
- Graphics memory usually on same board as rest of video logic

Altera Video VHDL Model

- View screen as a character oriented display
  - 40 columns x 16 rows (640 characters)
  - 1024 x 6 RAM holds contents of display
- Each RAM location specifies a character via a 6-bit code (allows 64 different characters)
- Each character is defined as a 8x8 pixel pattern in RAM.
  - Each pixel in RAM actually displayed as two pixels on screen (16 x 16)
  - 40 columns * 16 = 640 horizontal pixels
  - 16 rows * 16 = 256 vertical pixels

Block Diagram (video.vhd)



Pixel Generation

- The 8-bits from the Char ROM are muxed to the Red, Green outputs to form the pixel output
  - Red + Green => Yellowish background
  - Blue output in 'video.vhd' is currently grounded
- Every pixel out of Char ROM is used as two screen pixels to save memory costs
  - Not enough RAM in 10K20 part to have true bit-mapped display.

Video Ram vs Character ROM

- 6-bit value in Video RAM specifies upper 6 bits of address of Character ROM
- Data in Character ROM defines an 8X8 pixel pattern (expanded externally to 16 x 16)
  - Pixel pattern defines how character looks on screen
  - TCGROM.MIF file defines initial values of character ROM
- A 'MIF' file is how you specify the initial contents of a RAM in Altera
  - 'vidram.mif' specifies initial Screen contents

Character ROM instantiation

From 'video.vhd':

```
tiny_char_gen_rom: lpm_rom
    GENERIC MAP ( lpm_widthad => 9,
      lpm_numwords => "512",
      lpm_outdata => "UNREGISTERED",
      lpm_address_control => "UNREGISTERED",
-- Reads in mif file for character generator data
       lpm_file => "tcgrom.mif",
       lpm_width => 8)
    PORT MAP ( address => rom_address, q => rom_data);
```

tcgrom.mif

```
Depth = 512;
Width = 8;
Address_radix = oct;
Data_radix = bin;
% Character Generator ROM Data %
Content
  Begin
000  : 00111100 ; %    ****     %
001  : 01100110 ; %   **  **    %
002  : 01101110 ; %   ** ***    %
003  : 01101110 ; %   ** ***    %
004  : 01100000 ; %   **        %
005  : 01100010 ; %   **   *    %
006  : 00111100 ; %    ****     %
007  : 00000000 ; %             %

010  : 00011000 ; %     **      %
011  : 00111100 ; %    ****     %
012  : 01100110 ; %   **  **    %
013  : 01111110 ; %   ******    %
014  : 01100110 ; %   **  **    %
015  : 01100110 ; %   **  **    %
016  : 01100110 ; %   **  **    %
017  : 00000000 ; %             %
```

Only first few lines are shown.

Parameter setup

Comment

Contents

Note that address is specified in OCTAL, data is in binary.

Upper 6 bits of address is used to specify what the character is; lower 3 bits define the 8 scan lines

Video RAM instantiation

From 'video.vhd':

```
format_ram: lpm_ram_dq
    GENERIC MAP (lpm_widthad => 10,
      lpm_outdata => "UNREGISTERED",
      lpm_indata => "REGISTERED",
      lpm_address_control => "UNREGISTERED",
-- Reads in mif file for data display format\
      lpm_file => "vidram.mif",
      lpm_width => 6)
   PORT MAP (data => video_data,
                  address => format_address,
                  we => we,
                  inclock => notclock,
                  q => format_data);
```


vidram.mif

```
Depth = 1024;
Width = 6;
Address_radix = bin;
Data_radix = oct;
% Character Format ROM Data %
Content
  Begin
[0000000000..1111111111] : 40;
End;
```

Note that initial contents of Video RAM is '40' (octal). What is character '40'? Turns out to be the 'space' character as defined by 'tcgrom.mif'. So initial screen content is blank. Edit this file to change what the initial screen contents will be.