Lab 10: VGA Graphic Display

National Chiao Tung University
Chun-Jen Tsai
12/19/2017

Lab 10: VGA Graphic Display

- In this lab, you will implement a circuit that shows some graphics using the VGA video interface; your circuit must do the following things:
  - Animates the moon to move across the picture with green-screen removal
  - Adds fireworks animations to the picture

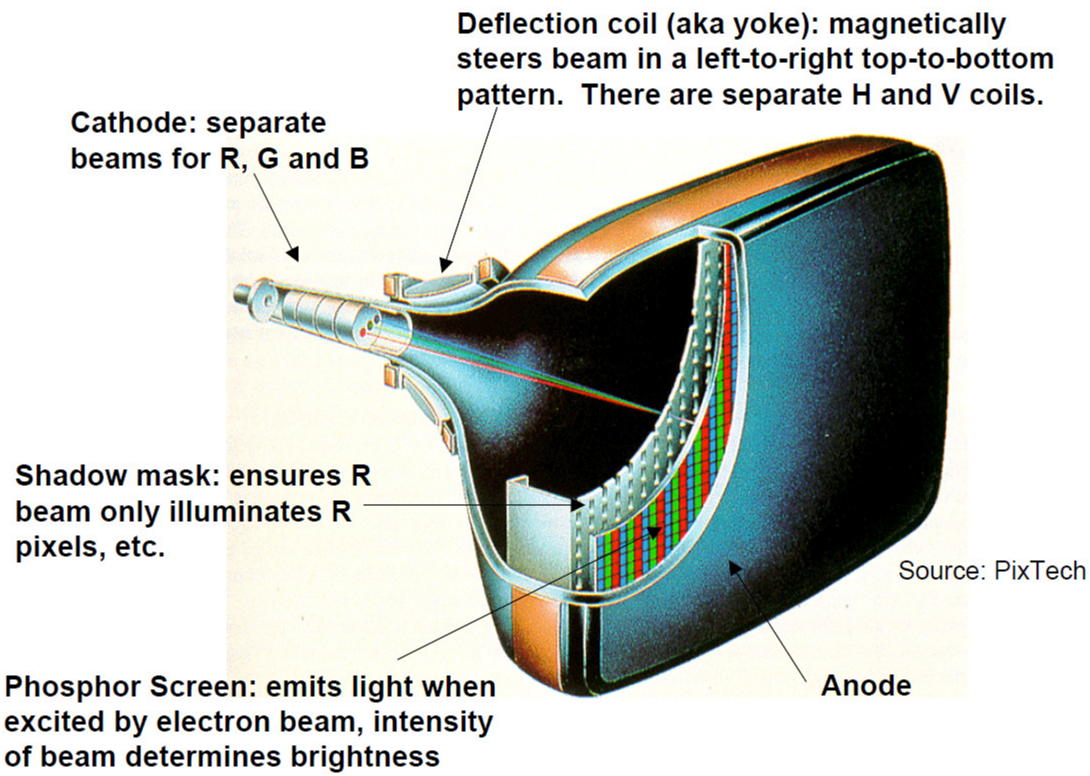
- The deadline of the lab is on 1/2

Lab 10 Setup

- In lab 10, the Arty board will share the LCD monitor with the PC
  - To see the video output of Arty, you must press the "video source" toggle on the LCD monitor to change the video source from "DVI" (for PC) to "VGA" (for Arty)
  - You should press "MENU"→ "Image" → "Aspect Control" and set it to 4:3.

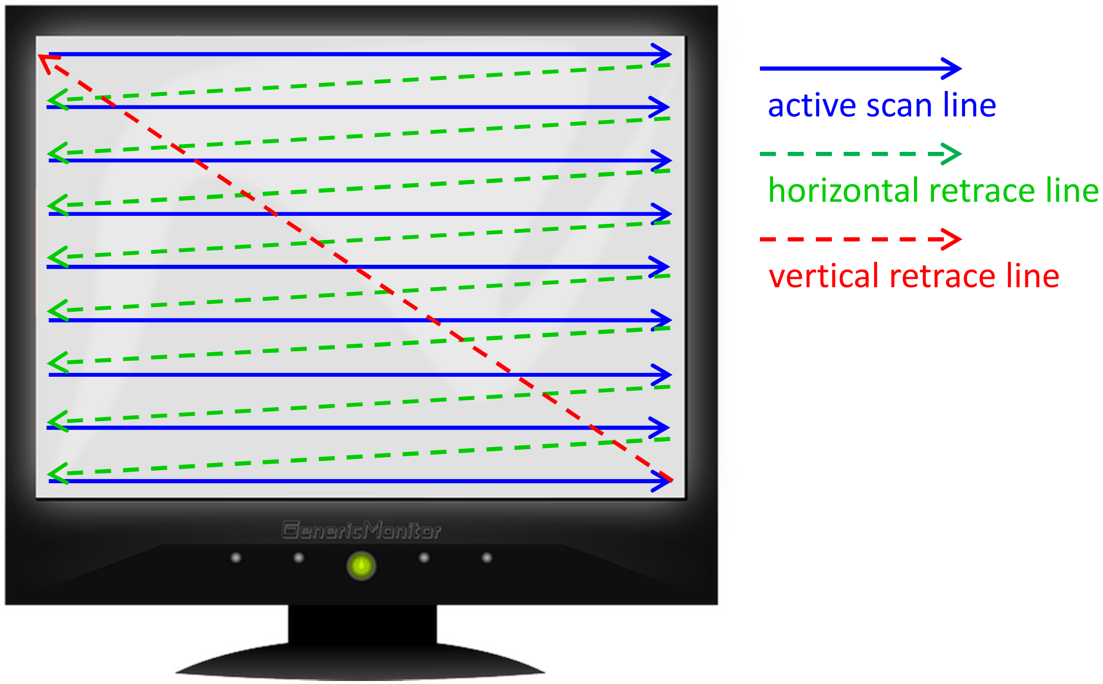
VGA – Old Soldiers Never Die

- VGA stands for Video Graphics Array; it's a video interface originally designed for CRT display



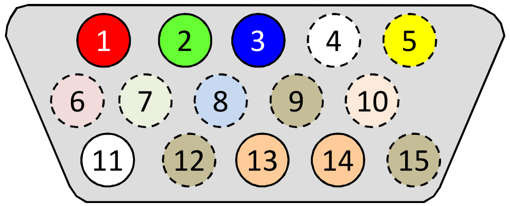
VGA Scanning Pattern

- A VGA screen displays the entire screen pixel-by-pixel:
  - The pixels of the screen are illuminated following a one-dimensional scanning path
  - When pixel at (x, y) are “scanned,” your circuit must tell the screen what RGB color it should display



VGA Interface Pin Assignment

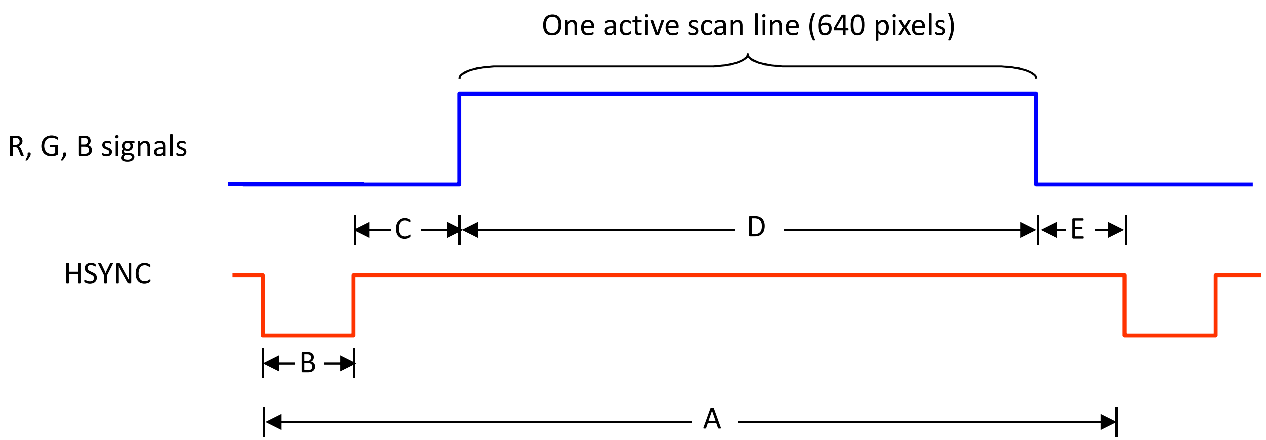
- VGA uses HD15 (a.k.a. D-sub) connectors
  - Pin #1 Red (0 ~ 0.7V)
  - Pin #2 Green (0 ~ 0.7V)
  - Pin #3 Blue (0 ~ 0.7V)
  - Pin #5, 6, 7, 8 Ground
  - Pin #9 Power (for external I^{2}C device)
  - Pin #10 Sync Return (Sync Ground)
  - Pin #4, 11 No connection
  - Pin #12 I^{2}C SDA
  - Pin #15 I^{2}C SCL
  - Pin #13 Horizontal Sync (2.5V)
  - Pin #14 Vertical Sync (2.5V)



HD-15 male

VGA Signal Timing (1/2)

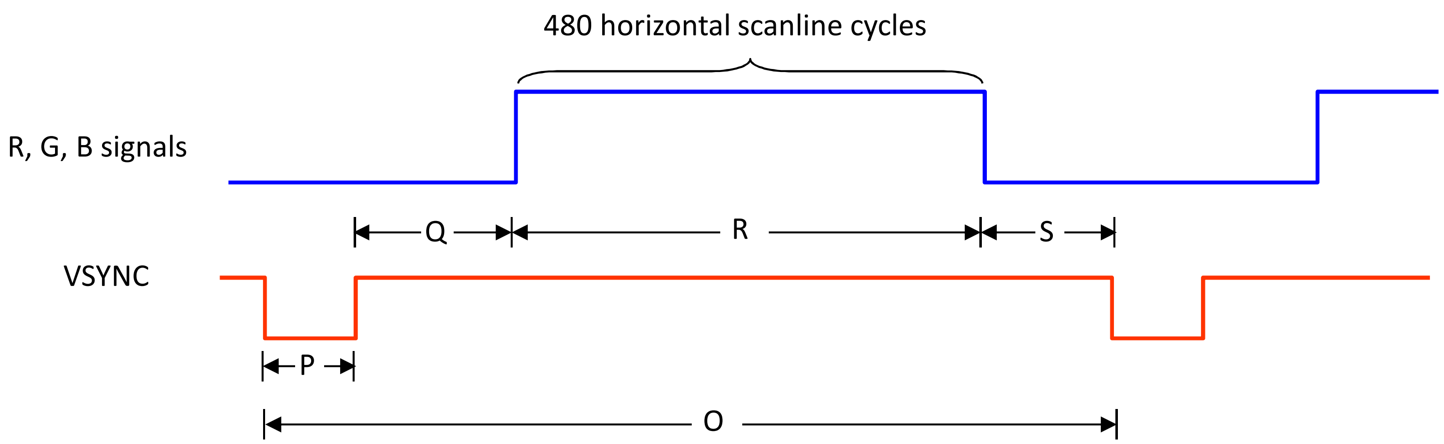
- Horizontal Retrace Cycles (for 640×480@60Hz):
  - $1/31.77\mu s = 31476$ Hz
  - $31476/60 = 525$ lines/frame



Parameters	A	B	C	D	E
Time	31.77μs	3.813μs	1.907μs	25.42μs	0.6356μs

VGA Signal Timing (2/2)

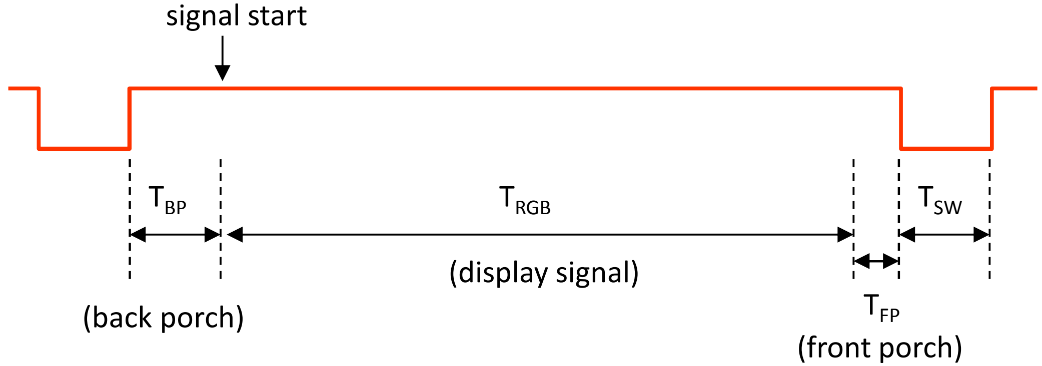
- Vertical Retrace Cycles (for 640x480@60Hz):
  - 1/16.66ms = 60 Hz (frames/second)



Parameters	O	P	Q	R	S
Time	16.68ms	63.56μs	1.049ms	15.25ms	0.3178ms

Sync Timing in Pixel Clocks

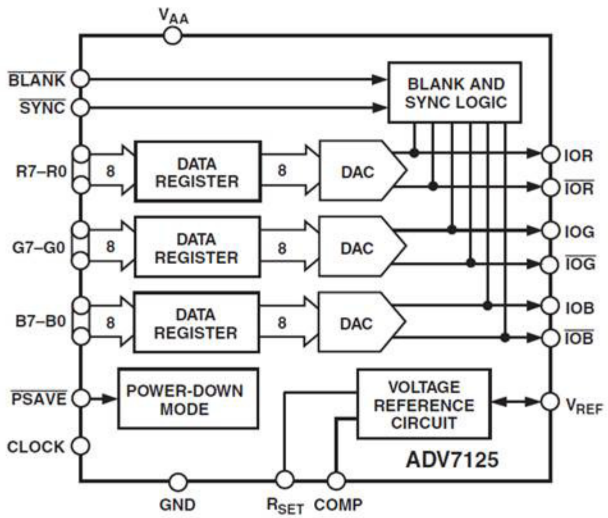
- The Horizontal Sync (HS) and Vertical Sync (VS) signal in pixel clock ticks are as follows:



Format	Sync Type	Clock	Total	T_{RGB}	T_{FP}	T_{SW}	T_{BP}
VGA (4:3 Screen) 640x480@60Hz	HS (pixels)	25.175MHz	800	640	48	96	16
	VS (lines)		525	480	10	2	33
VGA (4:3 Screen) 800x600@72Hz	HS (pixels)	50 MHz	1040	800	56	120	64
	VS (lines)		666	600	37	6	23

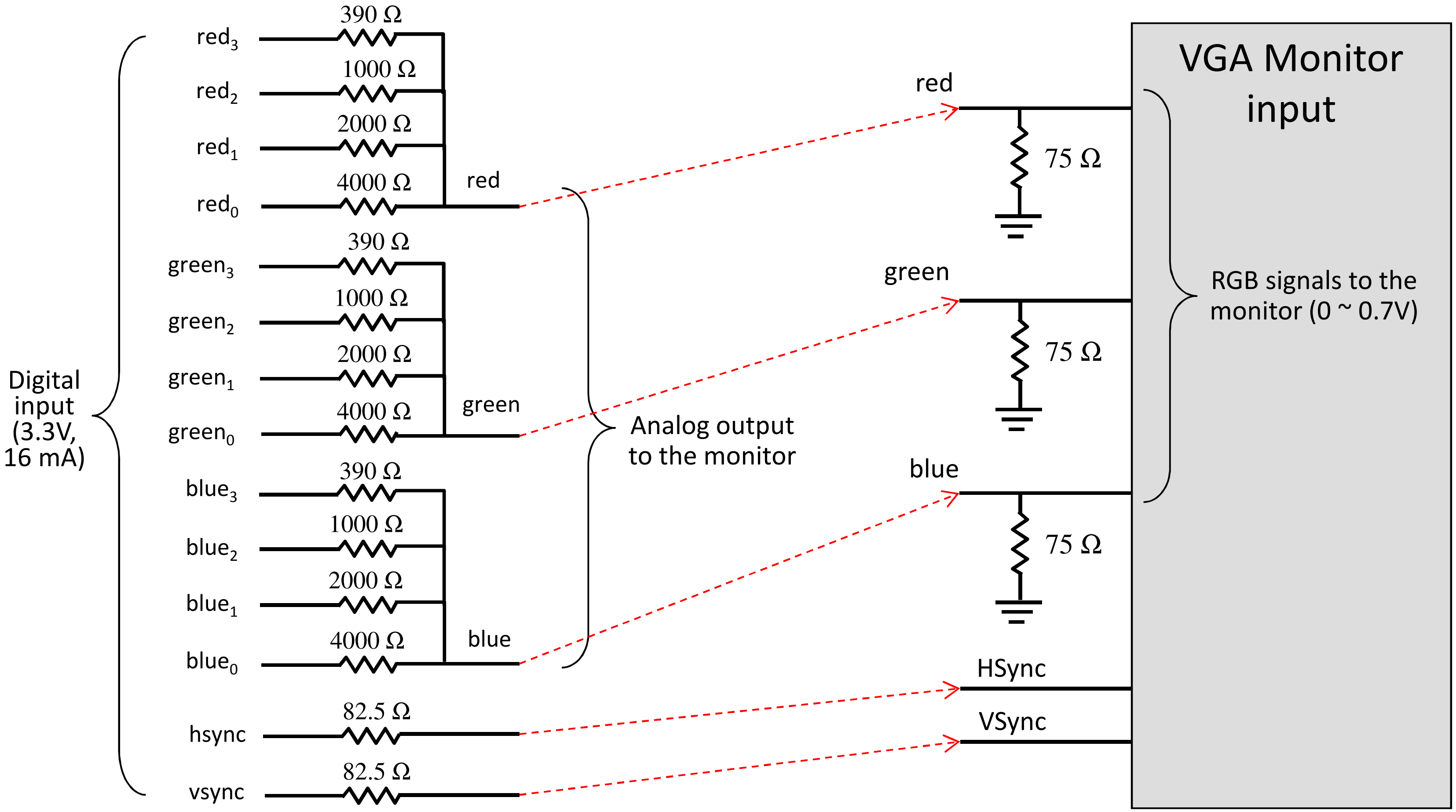
Digital-to-Analog Conversion

- VGA is an analog interface
  - 0v stands for the darkest pixel, 0.7v stands for the brightest
  - The transition from darkest pixel to brightest pixel is not linear
- A video digital-to-analog converter (DAC) IC is usually used to convert digital pictures to analog VGA signals
  - The most popular DAC is the ADV 7125 IC by Analog Devices



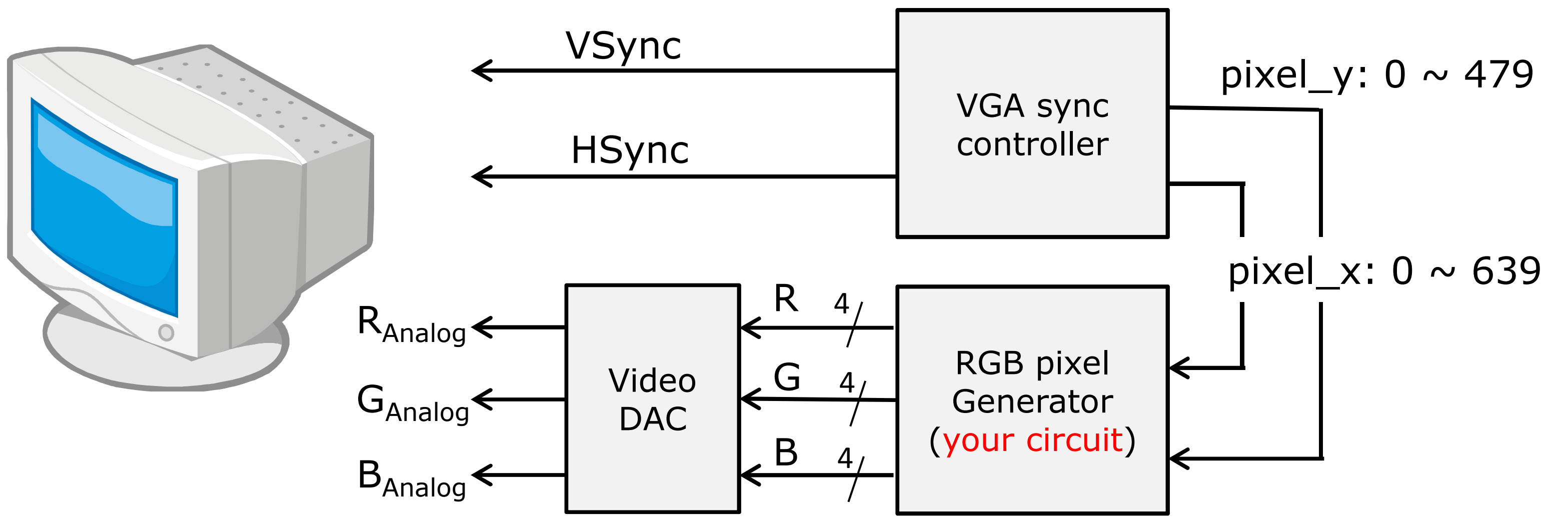
Simple VGA Interface Circuit

- A resister network is used on the I/O board of Arty to provide a 3×4-bit VGA DAC:



VGA Controller for Lab 10

- The Verilog code of a VGA synchronization controller will be provided for this lab:



VGA Sync Controller Interfaces

- The I/O port of the VGA controller module:

```
module vga_sync
(
   input clk,
   input reset,
   output wire oHS,
   output wire oVS,
   output wire visible,
   output wire p_tick,
   output wire [9:0] pixel_x,
   output wire [9:0] pixel_y
);
```

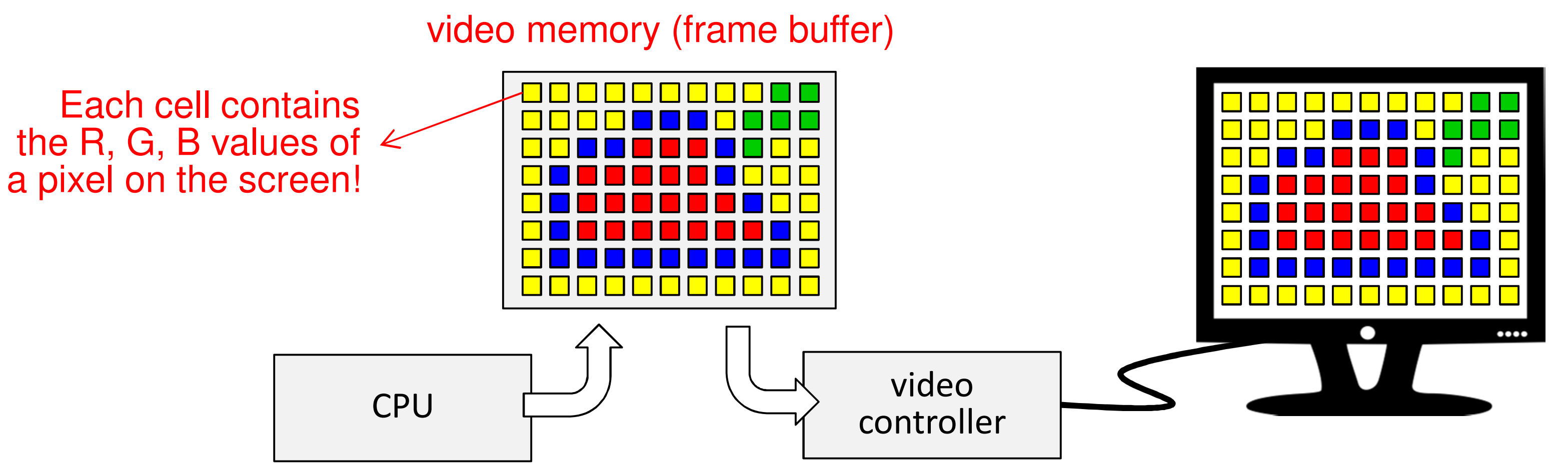
Is it active scan line period or sync period?

Ready to get next pixel?

* When "`visible`" is false, the RGB output to the monitor MUST be all zeros!

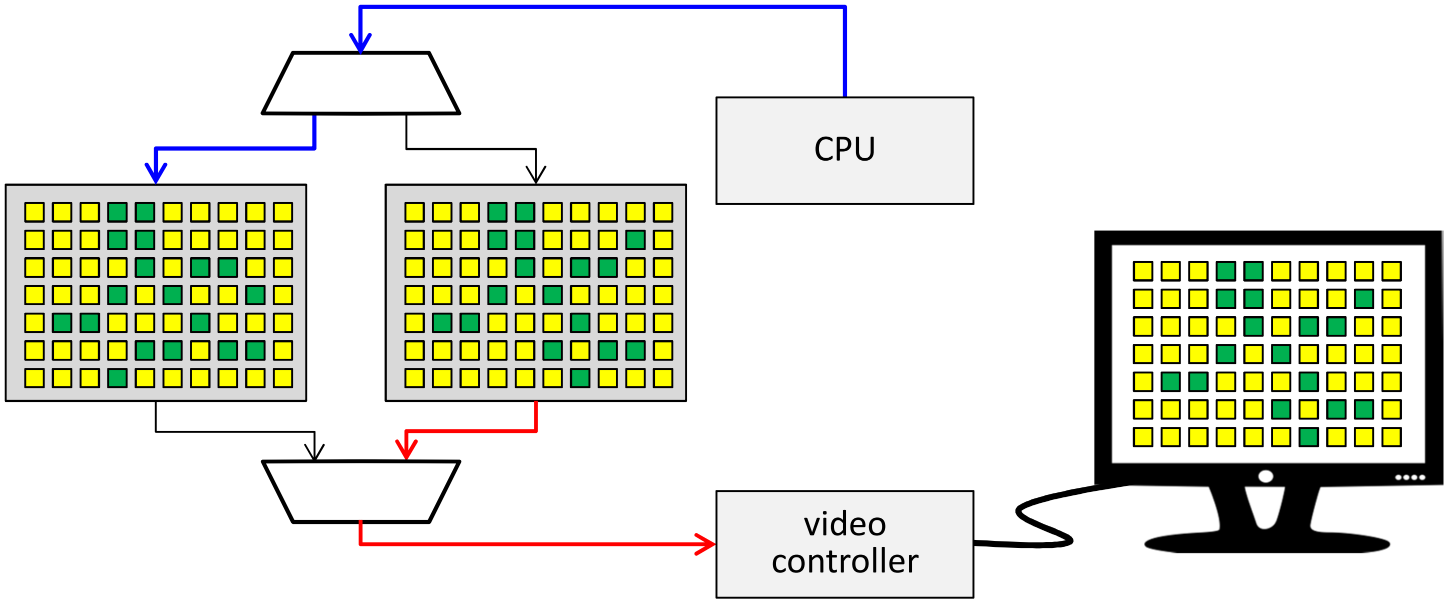
Computer Graphics and Video Buffer

- Processors are too slow to directly generate pixel data for video display
  - Old arcade games are built using dedicated circuit to produce graphics
- A break-through idea that enables software-based computer graphics is the concept of frame buffers (FB):



Double-Frame Buffering

- Computer systems usually use a single-port memory for video frame buffer
  - Most memory bandwidth will be used by the video controller
  - CPU cannot make significant update to the content of the frame buffer without interrupting the video controller
- Double frame buffers can be used to solve this problem



Cycle Stealing from the Retrace Time

- Some systems do not have enough memory for double-buffering, other tricks must be used to modify the FB without interfering the video controller
- During the horizontal or vertical retrace periods, the video controller is NOT reading the FB
- FB data modifications during the sync periods will not corrupt video

Good News for Lab 10

- In lab 10, you do not have to modify the video FB
  - The video FB is only used to store the background image
  - All foreground objects are generated by your circuit on-the-fly

- For example, the moving moon in the sky is done by:

A 320x240-pixel background image stored in the 12-bit SRAM.

A 64x40-pixel foreground image stored in the same SRAM. The green pixels are transparent pixels that shall not be displayed.

You must replace the green pixels by the background pixels.

Sending Frame Buffer Content to VGA

- The frame buffer in the SRAM can be connected to the VGA controller as follows:

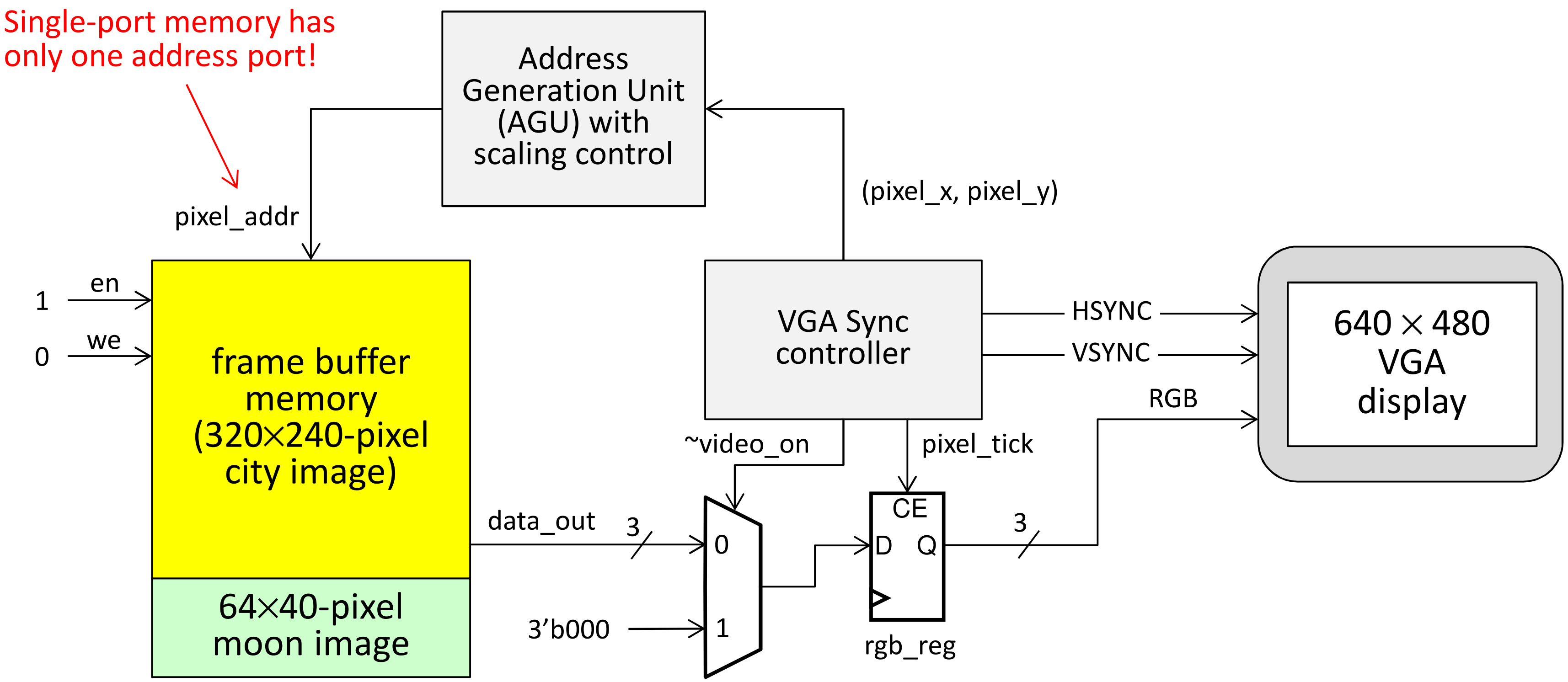
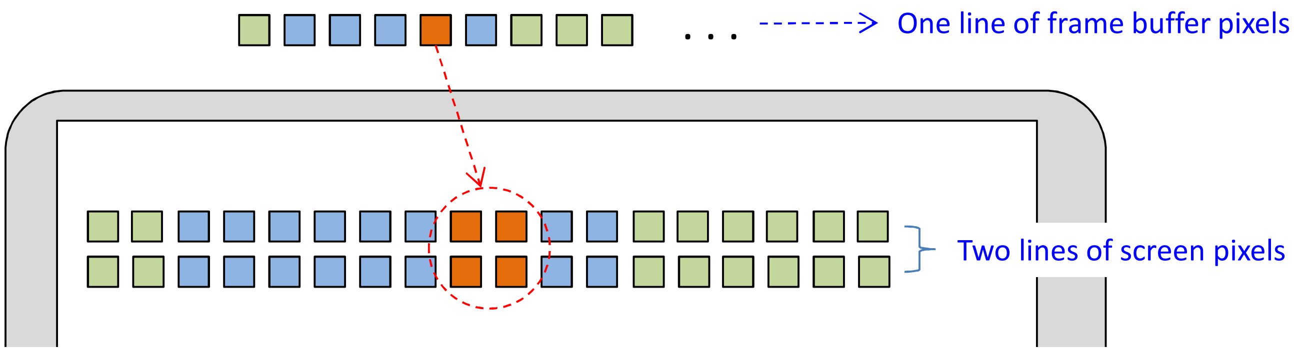


Image Scaling

- The size of our frame buffer has 320×240 pixels, but the VGA resolution is of 640×480 pixels → we must blow up the picture by 2× in each dimension
- A simple scaling algorithm can be used:
  - Each pixel in the frame buffer is used repeatedly for four pixels on the screen



Sample Code of the AGU with Scalar

- The address generation unit and the RGB register can be coded as follows:

You can change this to a combinational block, but the critical path will be longer

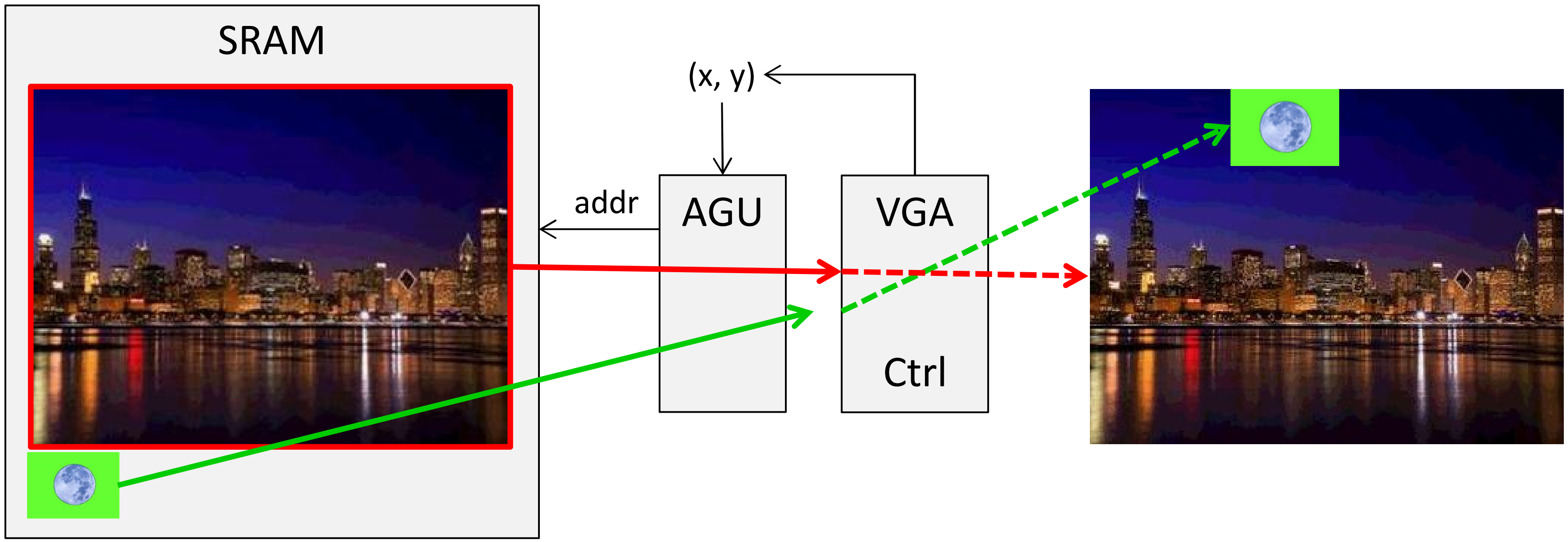
```
// --------------------- AGU ---------------------------
always @ (posedge clk) begin
  if (reset)
    pixel_addr <= 0;
  else
    // Scale up a 320x240 image for the 640x480 display.
    // (pixel_x, pixel_y) ranges from (0,0) to (639, 479)
    pixel_addr <= (pixel_y >> 1) * VBUF_W + (pixel_x >> 1);
end

// ----------------- RGB register ----------------------
always @(posedge clk) begin
  if (pixel_tick) rgb_reg <= rgb_next;
end

always @(*) begin
  if (~video_on)
    rgb_next = 3'b000; // Sync period, must set rgb to zeros
  else
    rgb_next = data_out; // RGB value at (pixel_x, pixel_y)
end
```

Video Frame Buffer for Lab 10

- In lab 10, we declare a 12-bit SRAM with 320×240 + 64 × 40 cells to store the city image and the moon image
  - The first 320×240 cells are also used as the video frame buffer
  - Address 0 is the start address of the video buffer (i.e., the city image)
  - Address 320×240 is the start address of the moon image
- The SRAM has only one port → the circuit cannot read from the video frame buffer and read the moon image at the same clock cycle!

The Sample Code in Lab 10

- A simple way to overlay the moon image on the city image is to read from the moon image if (x, y) falls in the moon area; otherwise, read from the city image:



Green Screen Technique

- Green screens are often used in video production to overlay foreground objects on a background shot at a different location
  - Green pixels of the foreground images will be replaced by the background image pixels at the same coordinates

What You Need to Do for Lab10?

- First, you must replace the green pixels of the moon image by the co-located background pixels (60 points)
- Secondly, you must use your imaginations to create some fireworks animations in the sky area (40 points)

Lab 10 sample code

Green-screen blending

Pretty fireworks

Your tasks for lab 10!