Lab 10: VGA Graphic Display

National Chiao Tung University
Chun-Jen Tsai
12/09/2016

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
  - Show a city photo and a moon on a VGA monitor
  - Allow the users to use the rotary dial to move the position of the moon

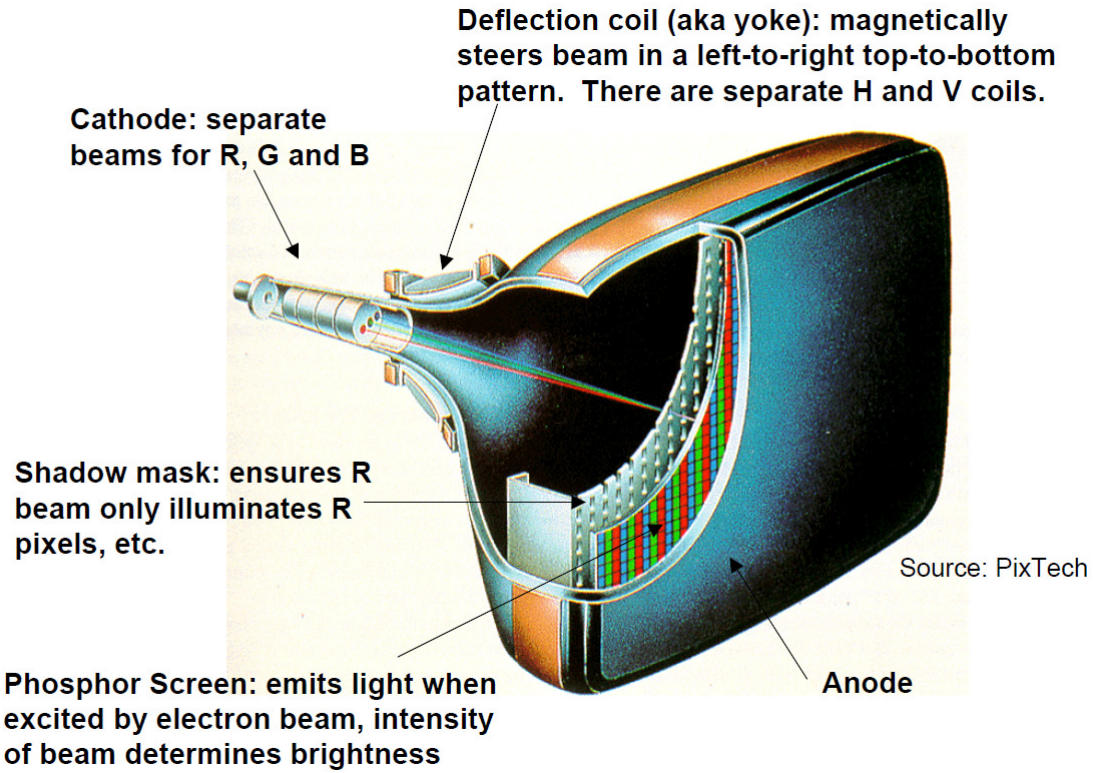
- You will demo the design to your TA during the lab hours on 12/27

Lab 10 Setup

- In lab 10, the Spartan-3E Starter Board will share the LCD monitor with the PC
  - To see the video output of Spartan-3E, you must press the “video source” toggle on the LCD monitor to change the video source from “DVI” (PC) to “VGA” (Spartan-3E)

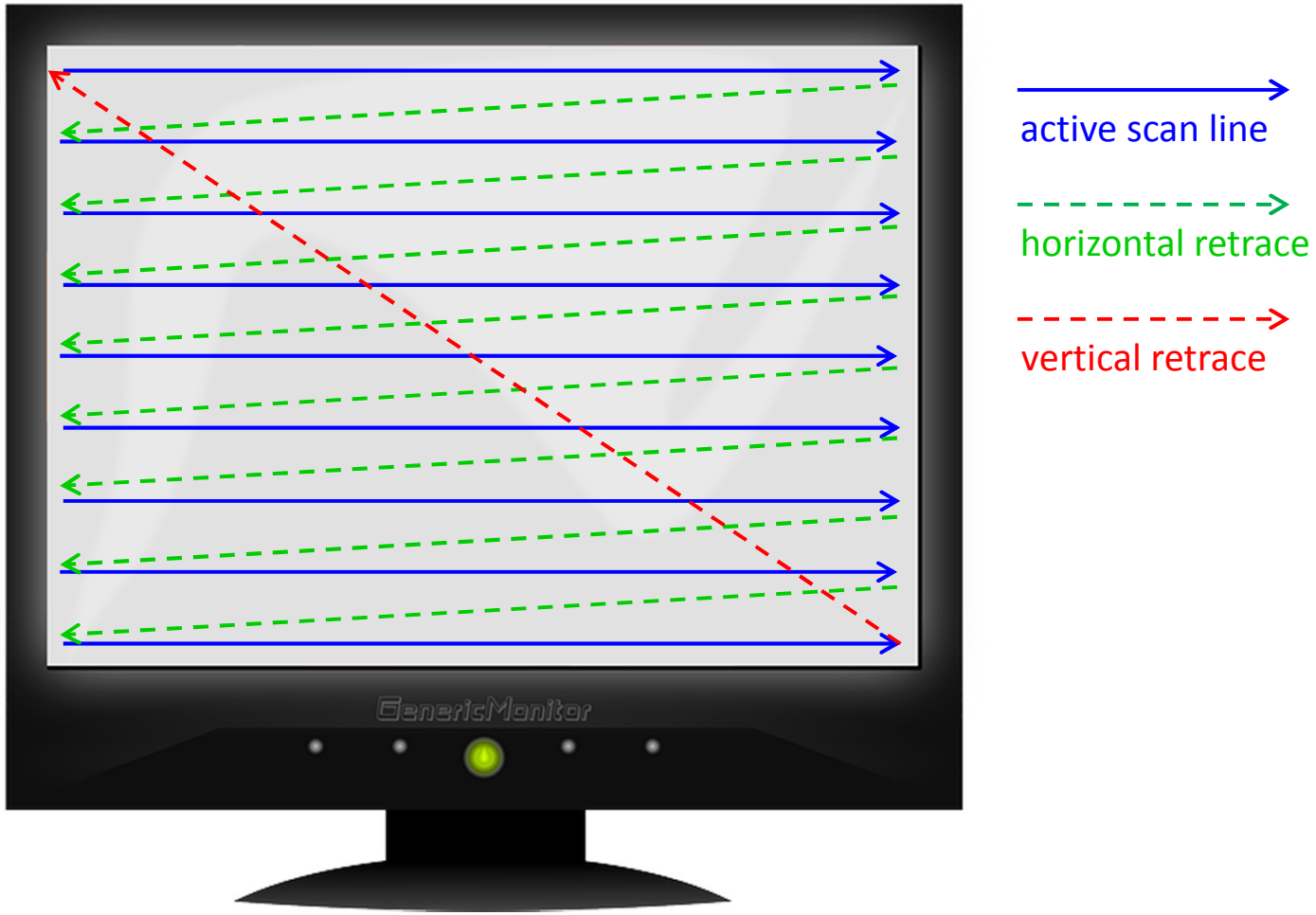
VGA – Old Soldiers Never Die

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



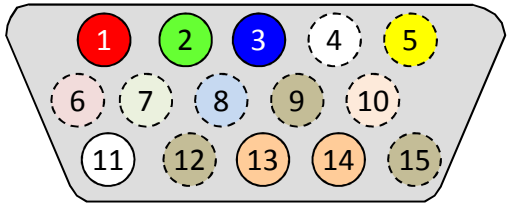
VGA Scanning Pattern

- A VGA display scans the entire screen pixel-by-pixel:



VGA Interface Pin Assignment

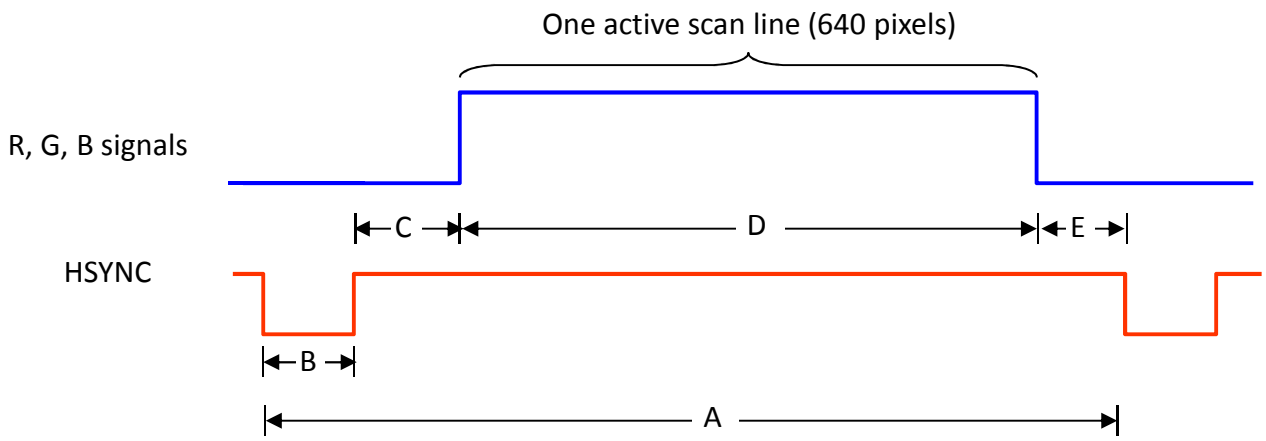
- VGA uses HD15 (a.k.a. D-sub) connectors
  - Pin #1 Red (0 ~ 0.7V)
  - Pin #2 Green (0 ~ 0.7V)
  - Pin #3 Blue (0 ~ 0.7V)
  - Pin #5, 6, 7, 8 Ground
  - Pin #9 Power (for external I^2C device)
  - Pin #10 Sync Return (Sync Ground)
  - Pin #4, 11 No connection
  - Pin #12 I^2C SDA
  - Pin #15 I^2C SCL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



HD-15 male

VGA Signal Timing (1/2)

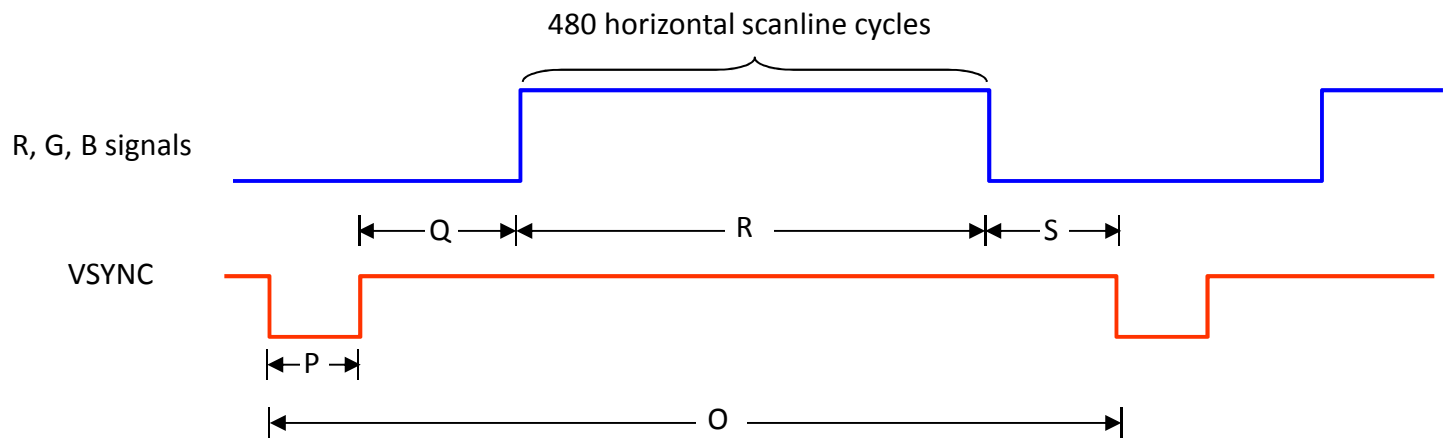
- Horizontal Retrace Cycles (for 640×480@60Hz):
  - 1/31.77μs = 31476 Hz
  - 31476/60 = 525 lines/frame



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

VGA Signal Timing (2/2)

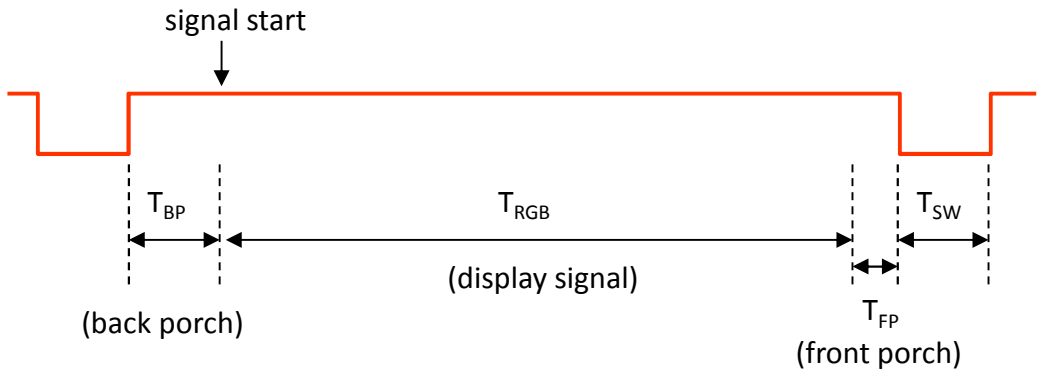
- Vertical Retrace Cycles (for 640x480@60Hz):
  - 1/16.66ms = 60 Hz (frames/second)



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

Sync Timing in Pixel Clocks

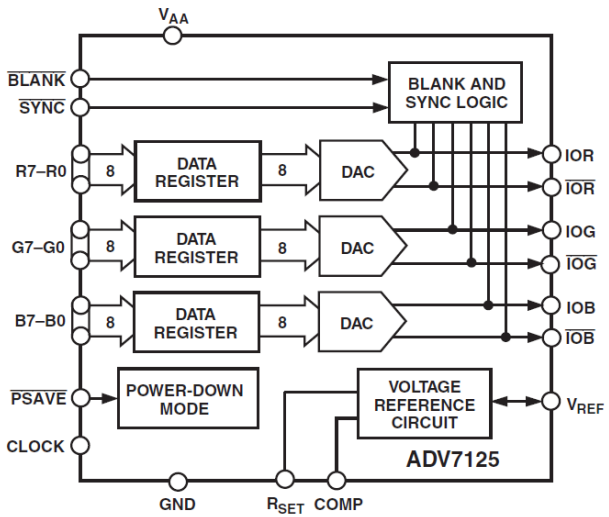
- The Horizontal Sync (HS) and Vertical Sync (VS) signal in pixel clock ticks are as follows:



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

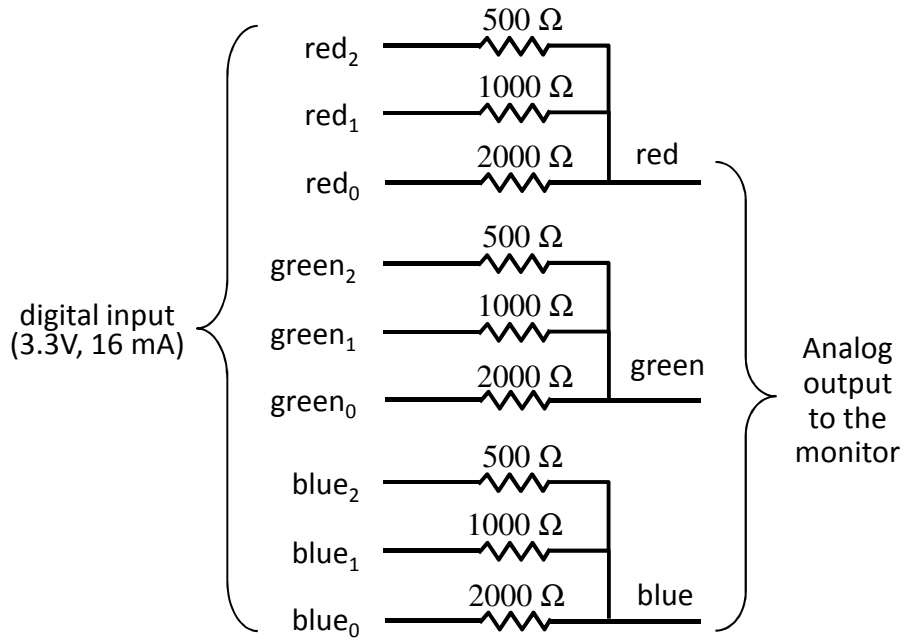
Digital-to-Analog Conversion

- 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 DAC IC is usually used for converting digital picture to analog VGA signals
  - The most popular DAC is the ADV 7125 IC by Analog Devices

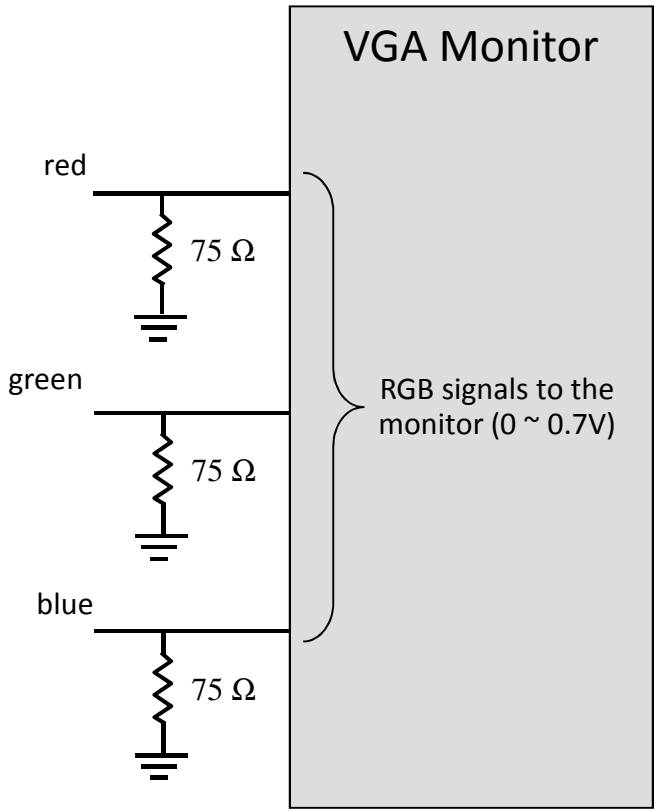


Simple VGA Interface Circuit

- A resister network can be used as a 3×3-bit VGA DAC:

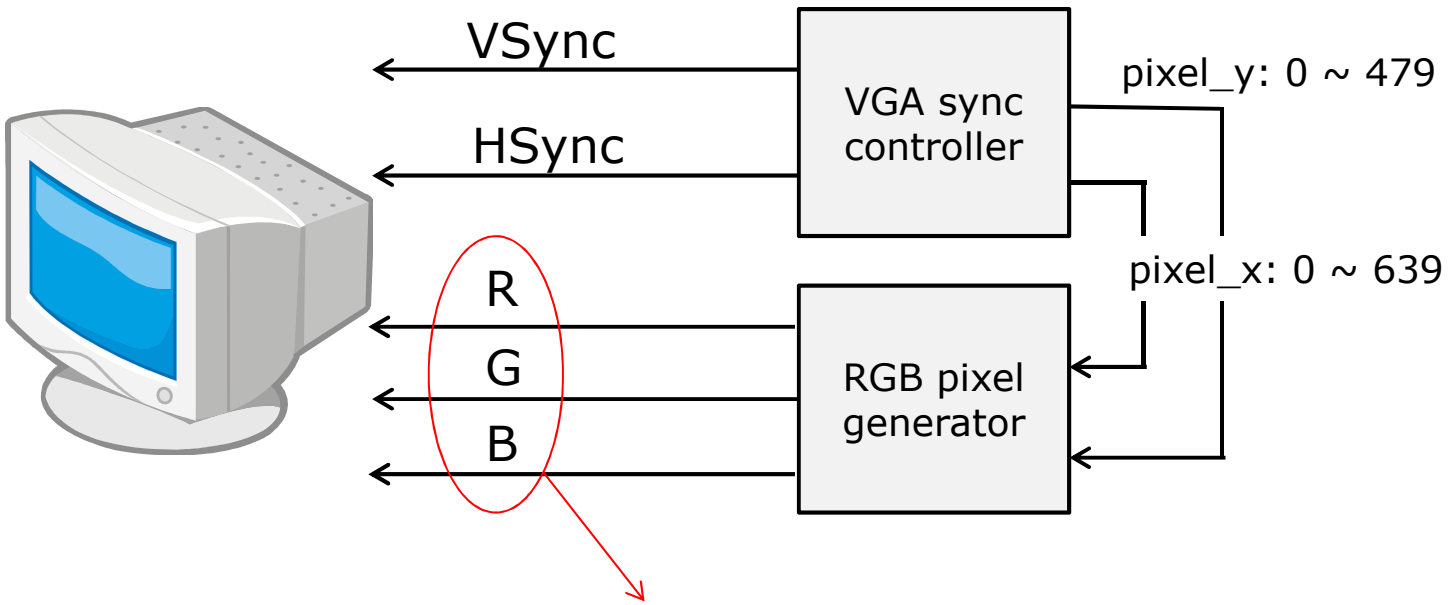


digital-to-analog conversion



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
);
```

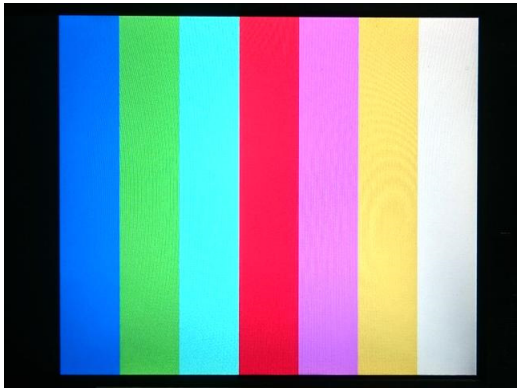
Is it active scan line period or sync period?

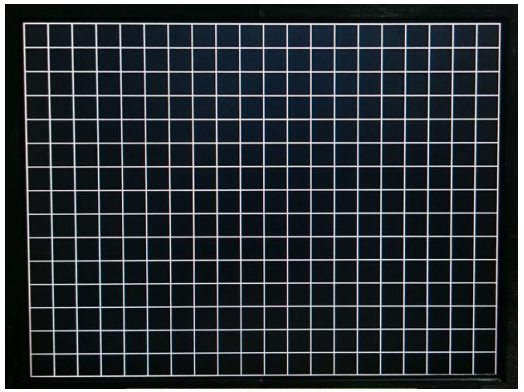
Ready to get next pixel?

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

What's in Lab 10 Sample Code

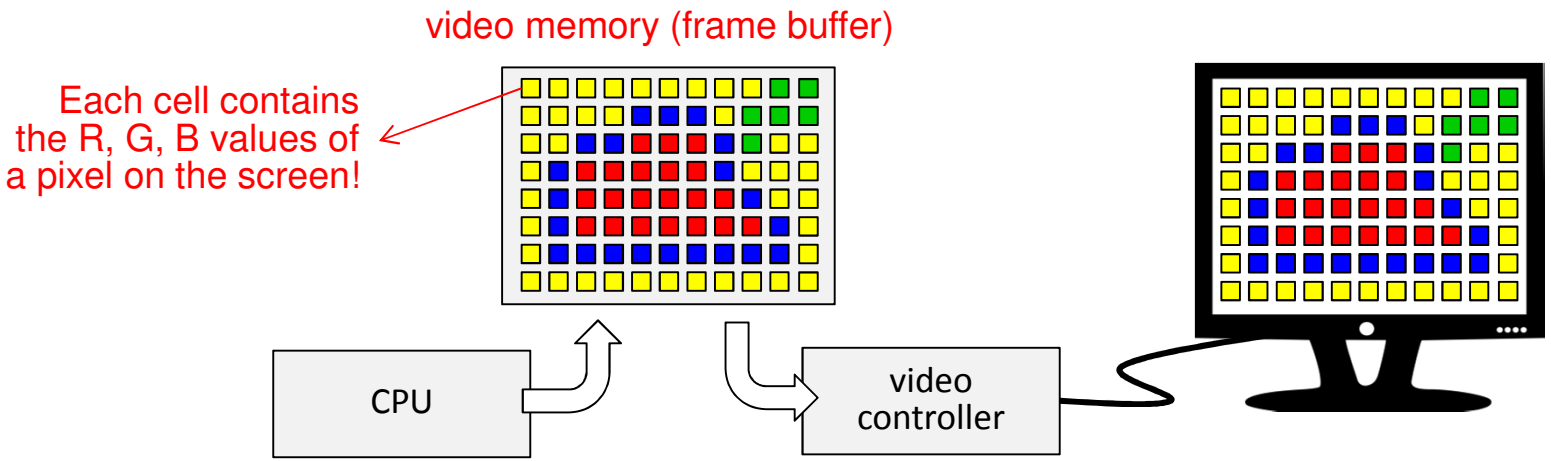
- In Lab 10, a sample circuit that shows eight different video test screens will be provided
  - Users use the rotary dial to select different screens





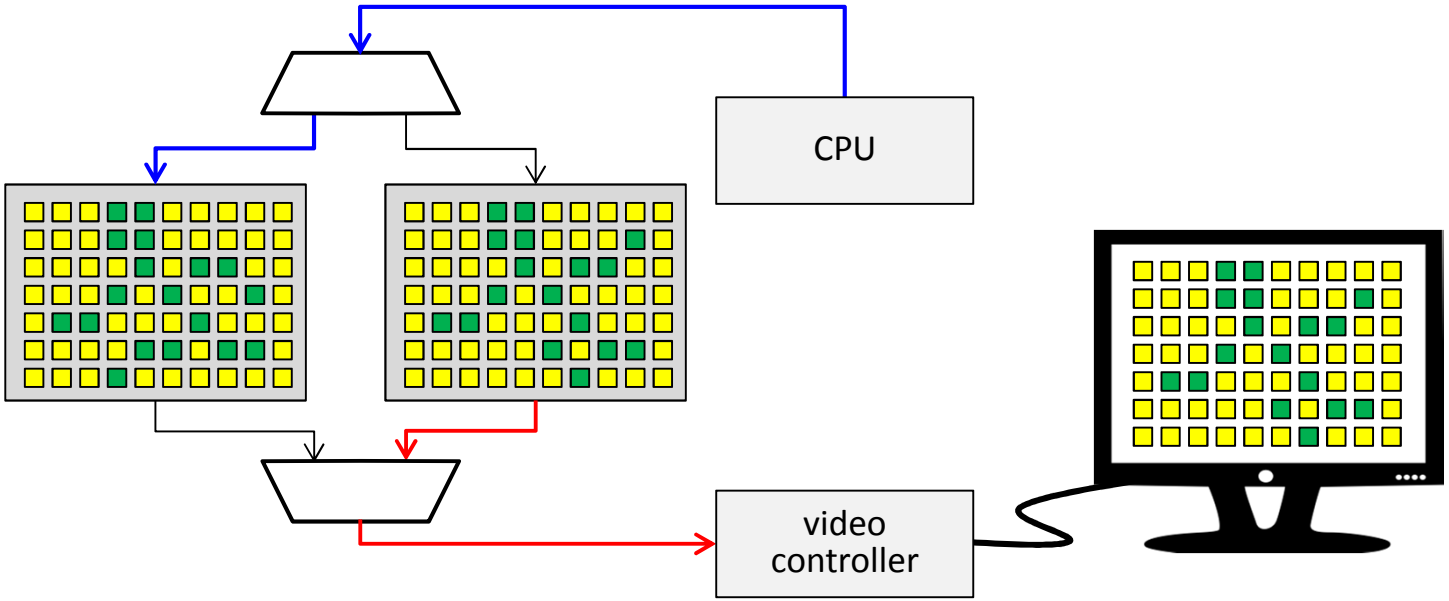

Computer GUI and Video Buffer

- 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:



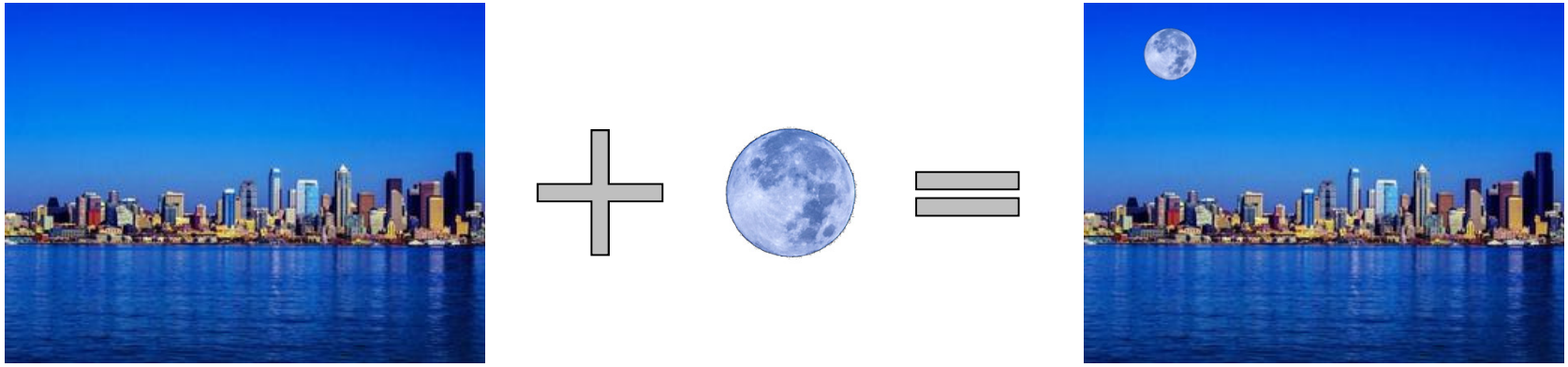
Double-Frame Buffering

- We usually use a single-port memory device 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



What You Need to Do for Lab10?

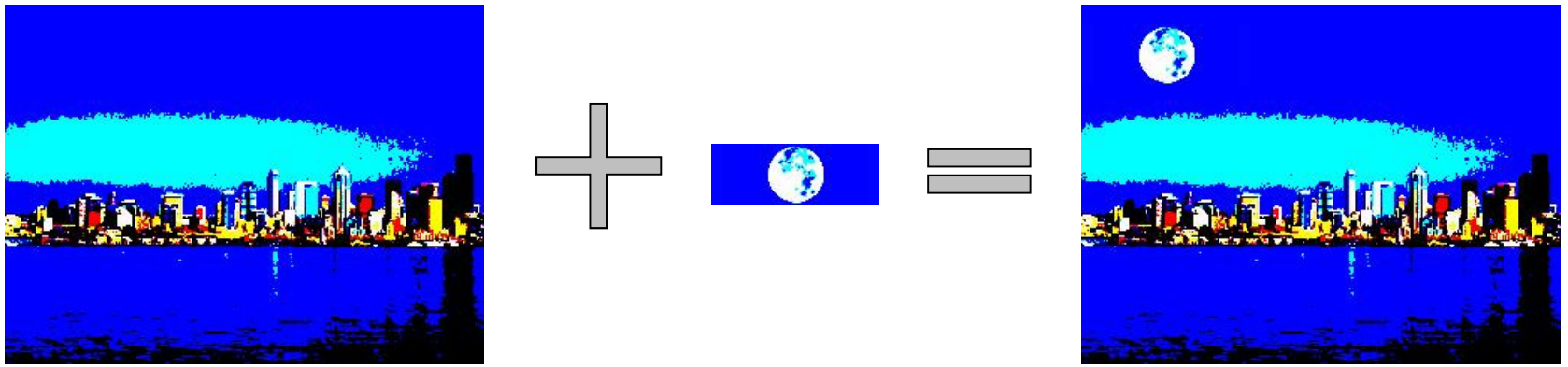
- Color images of a city scene and a moon are stored in the SRAM of the sample code of lab10.
- You must “paste” the moon to the sky of the city scene, and let the user to move the moon horizontally using the rotary dial:



Color Reduction for 3-Bit VGA

- Unfortunately, we have a 3-bit VGA interface, so we must convert the color depth to 1-bit per channel
  - The images stored in the SRAM are already color reduced!

```
R <= (R_8_bit > 128)? 1 : 0;
G <= (G_8_bit > 128)? 1 : 0;
B <= (B_8_bit > 128)? 1 : 0;
```



Sending Frame Buffer Content to VGA

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

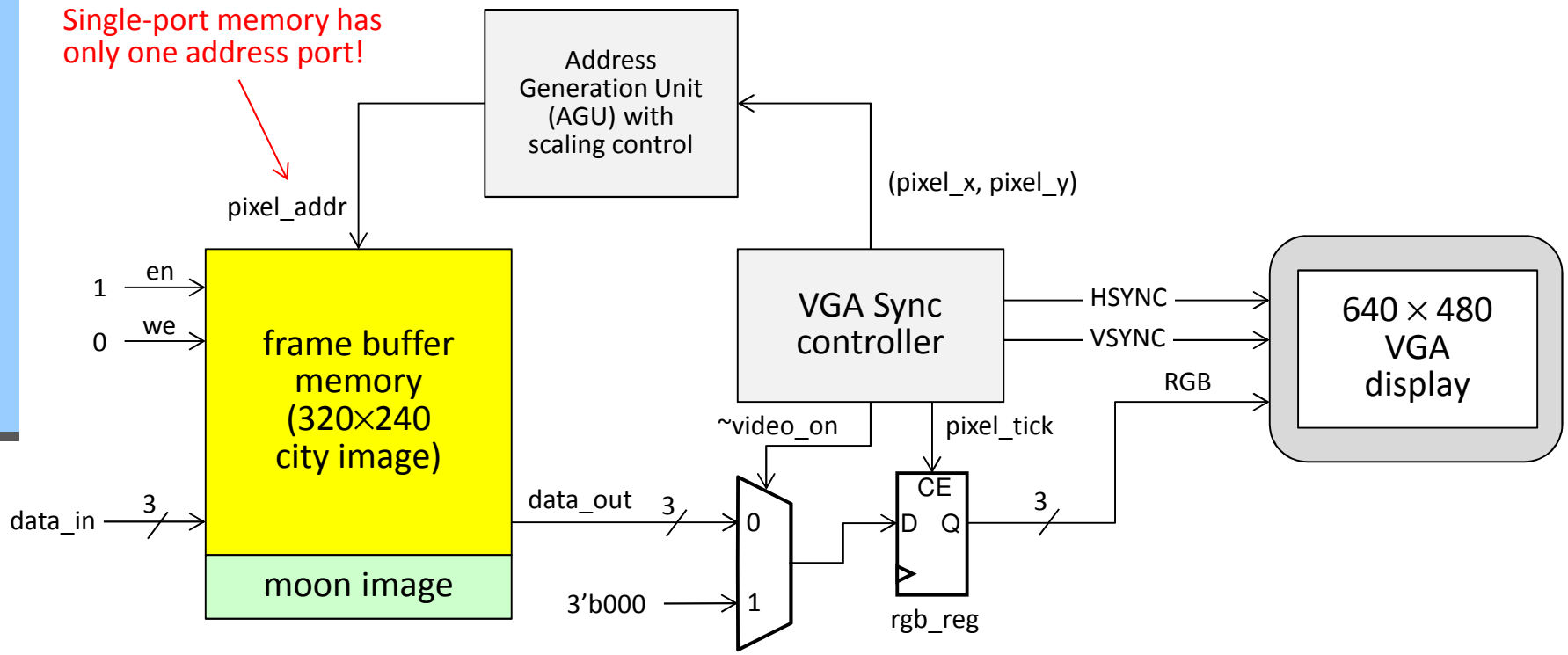
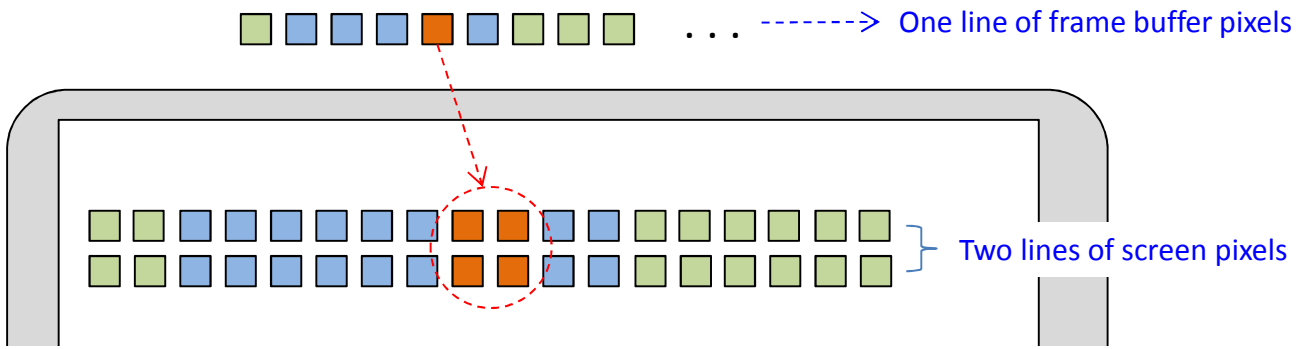


Image Scaling

- 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

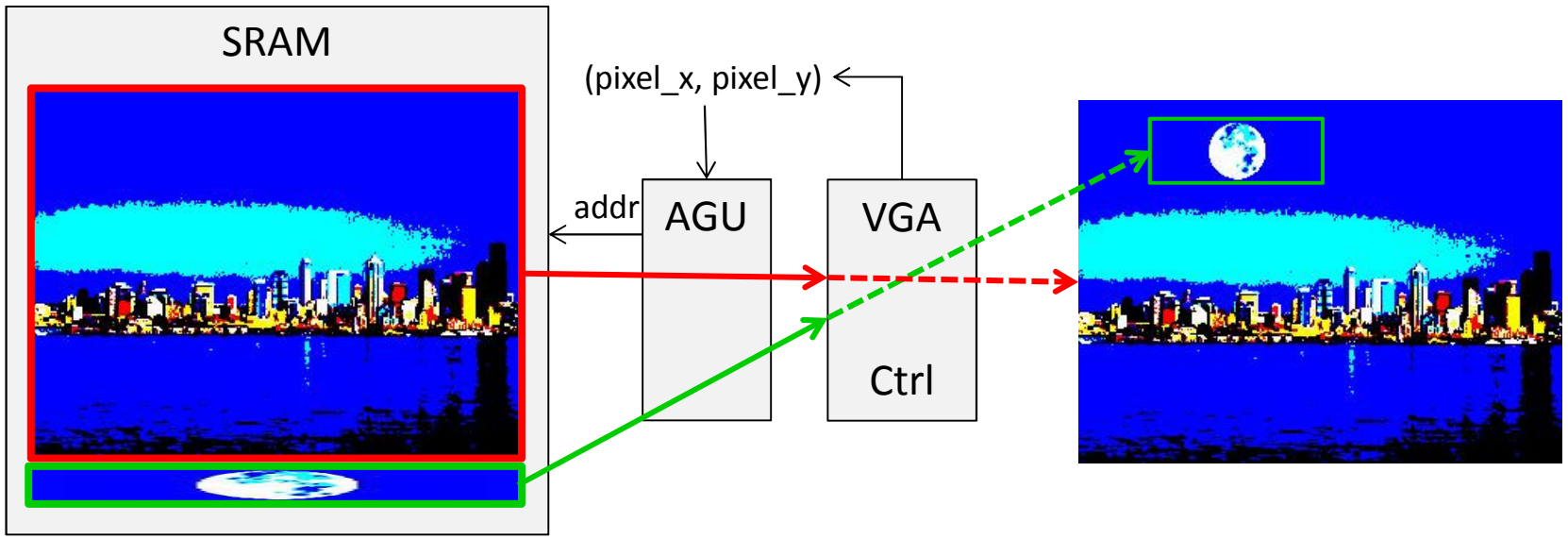
```
// ---------------------- AGU ---------------------------
always @ (posedge clk) begin
  if (reset)
    pixel_addr <= 0;
  else
    // Scale up a 320x240 image for the 640x480 display.
    // (pixel_x, pixel_y) ranges from (0,0) to (639, 379)
    pixel_addr <= (pixel_y >> 1) * VBUF_W + (pixel_x >> 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 3-bit SRAM with 320×240 + 112 × 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
  - 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, read the moon image, and write to the video frame buffer at the same time!

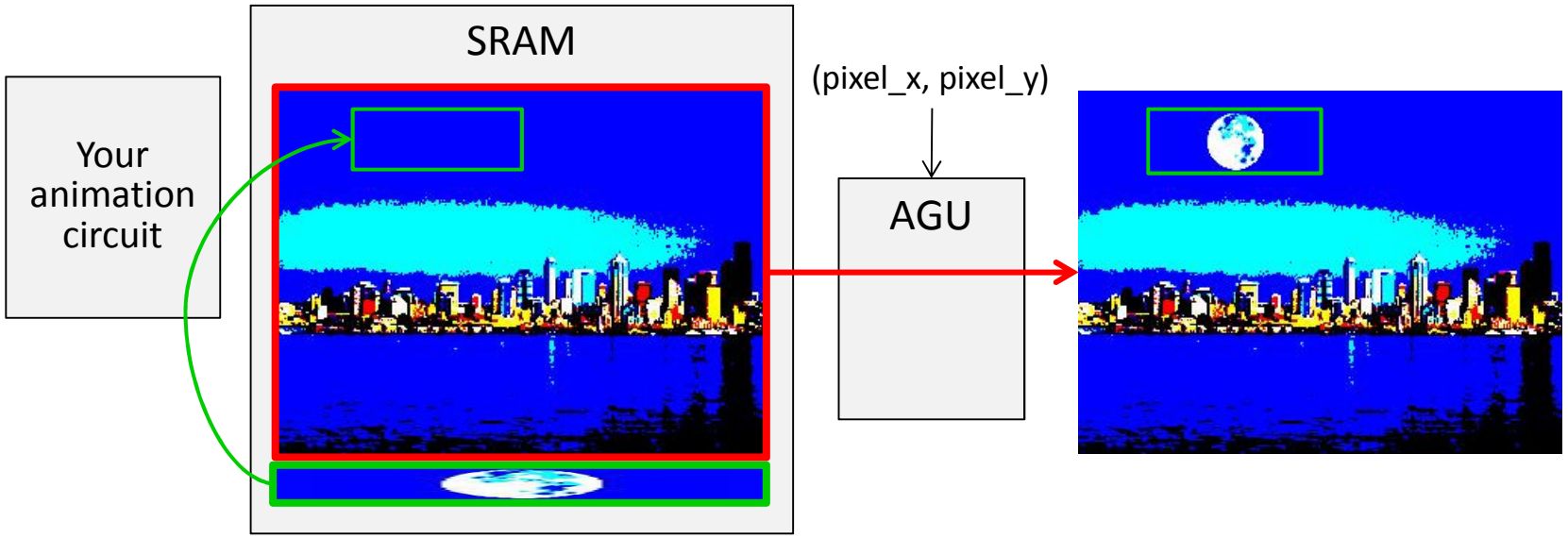
The Simple Solution for Lab 10

- A simple way to perform the animation is to read from the moon image if (pixel_x, pixel_y) falls in the moon area; otherwise, read from the city image



The Complex Solution for Lab 10

- A more flexible but complex way to do lab 10 is to copy the moon image to the appropriate location in the frame buffer



Accessing SRAM while the video controller is scanning the frame buffer may corrupt the VGA output!!!

Cycle Stealing from the Retrace Time

- During the horizontal or vertical retrace periods, the video controller is NOT reading the SRAM
  - SRAM data copying during sync period will not corrupt video
- Copy one line of the moon image takes 112 cycles to read and 112 cycles to write
  - Horizontal retrace period for each scan line takes 160 pixel clock cycles (equal 320 system clock cycles) → good enough!
  - When $0 \leq$ pixel_y < 479 and video_on is false, we are in the horizontal retrace period

Requirements for Lab 10

- For lab 10, you must implement the complex solution to get full credit; if you just do the simple solution, you only get partial credit
- The rotary dial controls the position of the moon; you must implement seven different positions (relative to the 320×240 city image) as follows:

  The upper-left corner of the moon image must be placed at `(pos×32, 8)`, where `pos = 0, 1, …, 6` is controlled by the rotary dial.

(0, 0)

(319, 239)

References

- Chapter 6 of *Spartan-3E Starter Kit Board User Guide, UG230 (v1.0)*.