EECS2070 02
Digilent Basys3 FPGA Board
Part 5

Ref: Digilent Basys3™ FPGA Board Reference Manual

黃稚存

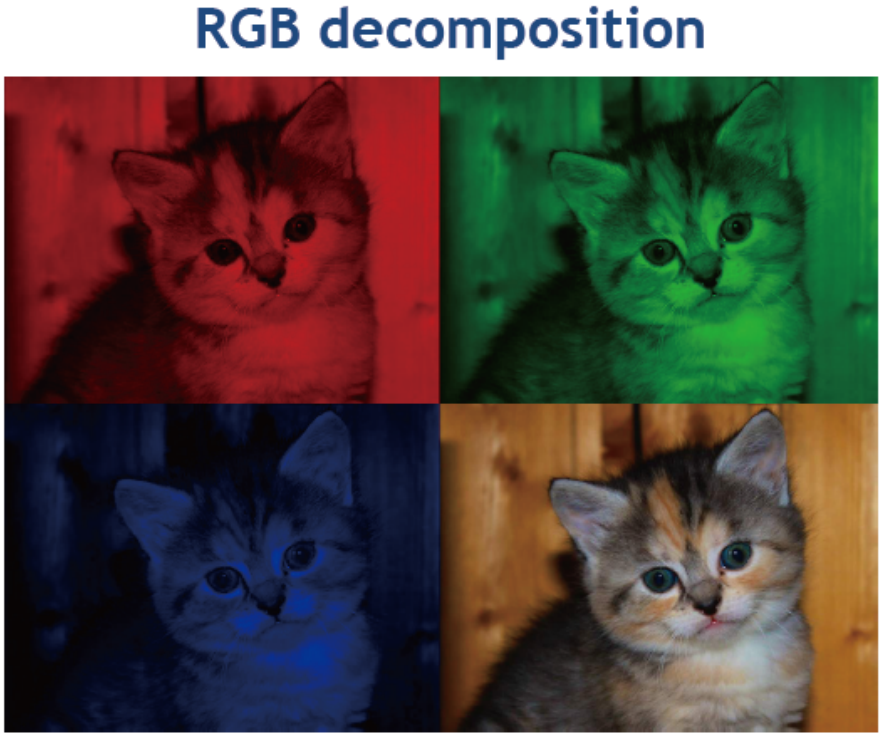
國立清華大學
資訊工程學系

Lecture 10

VGA Port

RGB Bitmap

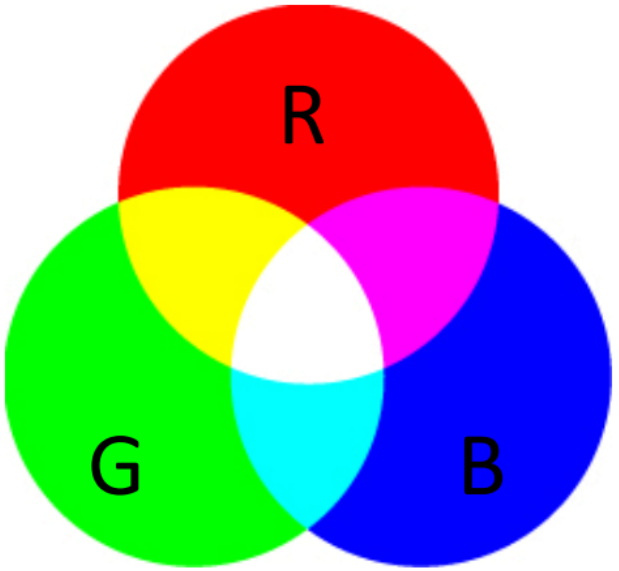
- A digital color image is composed by a lot of pixels
- Each pixel contains three R, G, B values to represent the intensity of these three primary colors



Src: faculty.ycp.edu/~dhovemey/spring2013/cs101/labs/lab11.html

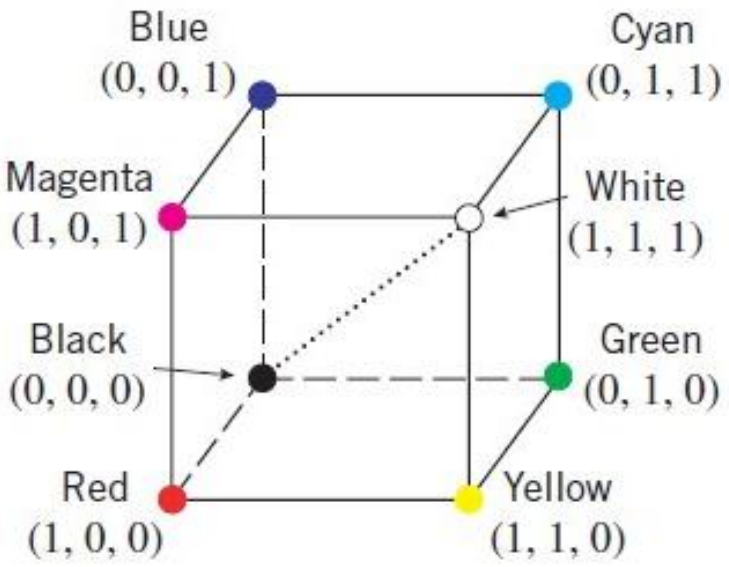
RGB Color Mode

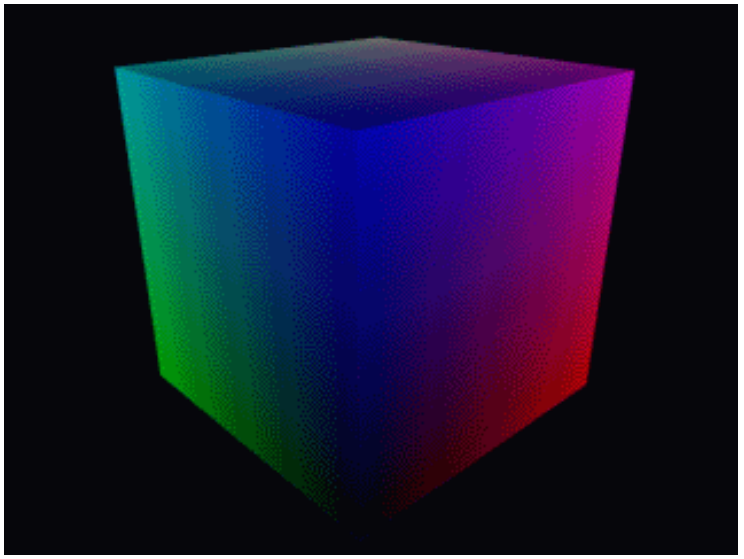
- Three primary colors
  - **Red**
  - **Green**
  - **Blue**
- No one of them can be created as a combination of the other two
- Any other color is a combination of them



RGB Color Cube

- R, G, and B correspond to three axes in 3D space.
- Normalize their values to be between (0,1)

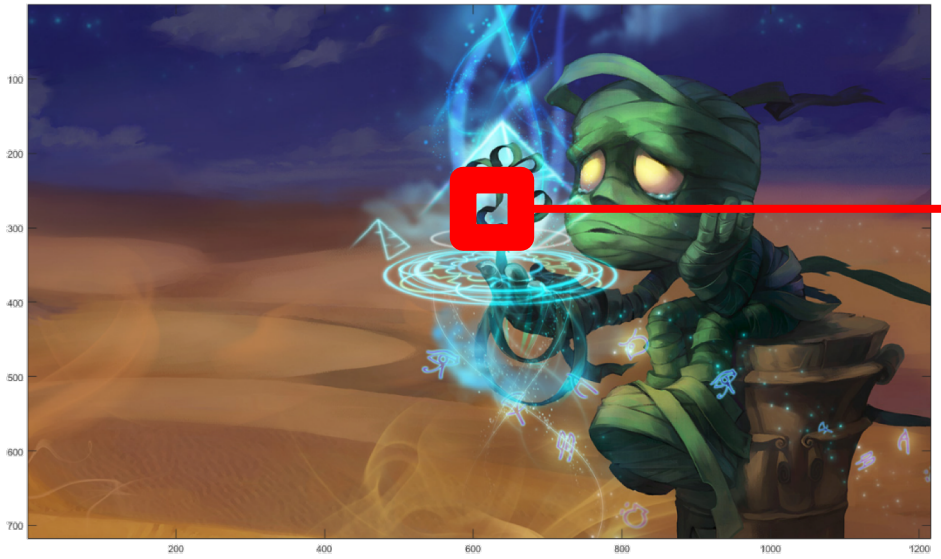




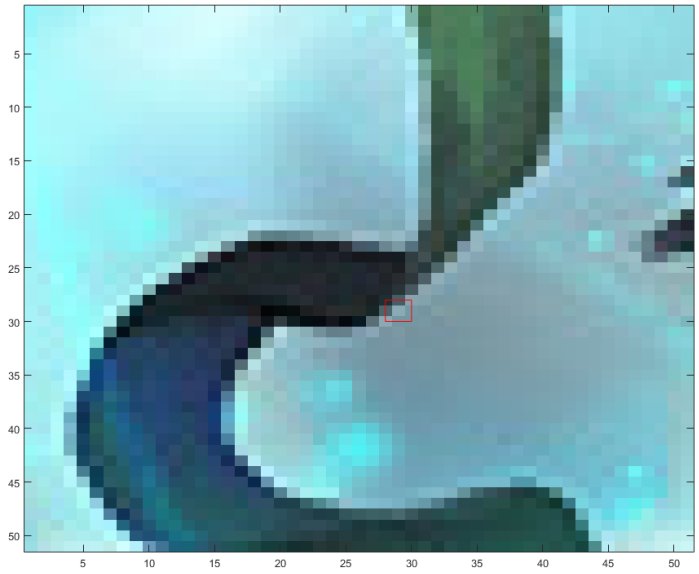
Src: www.chegg.com http://www.ahinson.com/research/res3.htm

RGB Bitmap Example (1/2)

Size: 1215*717



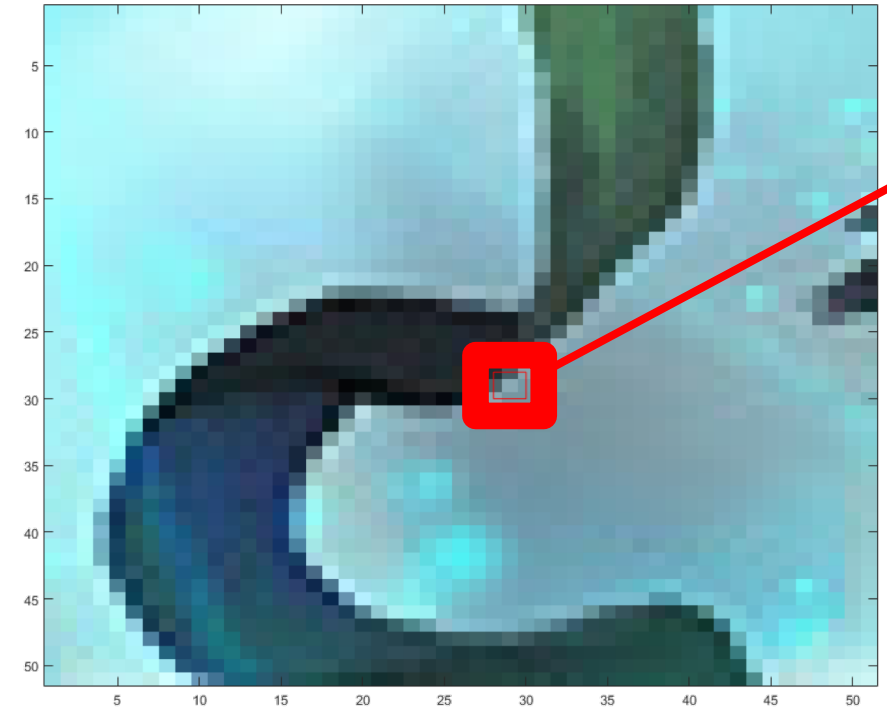
Size: 51*51

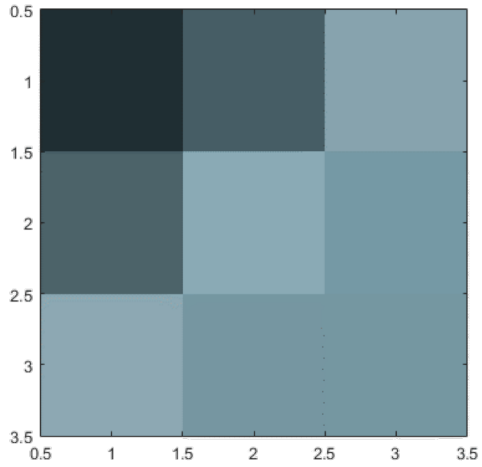


Src: leagueoflegends.com

RGB Bitmap Example (2/2)

- Each pixel is a combination of RGB values
  - 8x3 bits per pixels: 16,777,216 (16M) colors
  - 4x3 bits per pixels: 4096 colors
- The smaller the value, the darker the pixel





(R,G,B) : range from 0 to 255

(31,46,51)	(70,93,101)	(135,163,174)
(76,99,105)	(138,170,181)	(117,153,165)
(141,168,179)	(118,150,161)	(117,151,161)

Src: leagueoflegends.com

VGA (Video Graphics Array)

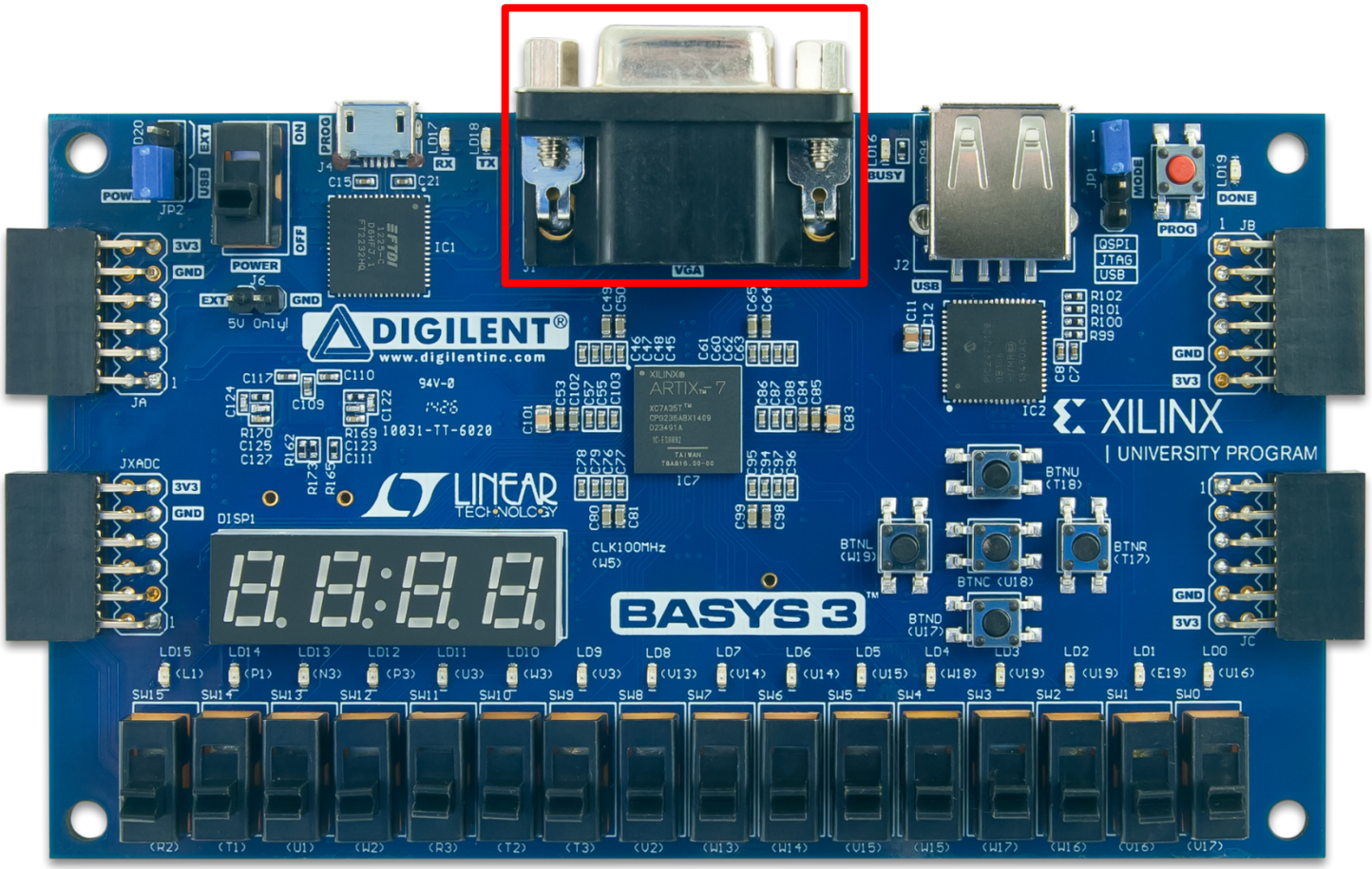
- Introduced by IBM in 1987, and still used today
- Transmitting analog signals
  - Today's HDMI transmits digital signals

Cathode-Ray Tube (CRT) Monitor

VGA Connector

LED Monitor

VGA Port on Basys 3

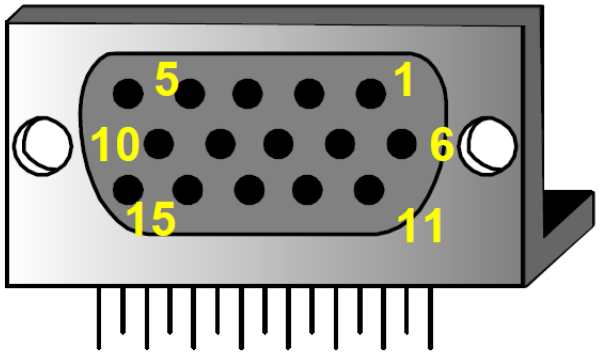


Control Signals on Basys 3

- 14 signals to create a VGA port
  - Red, green, blue (RGB) colors
    - 4 bits per color; 16 levels for each color
    - For one pixel
      - 12 bits
      - 4096 different colors
  - 2 standard sync signals
    - HS: horizontal sync
    - VS: vertical sync

VGA Video Signal

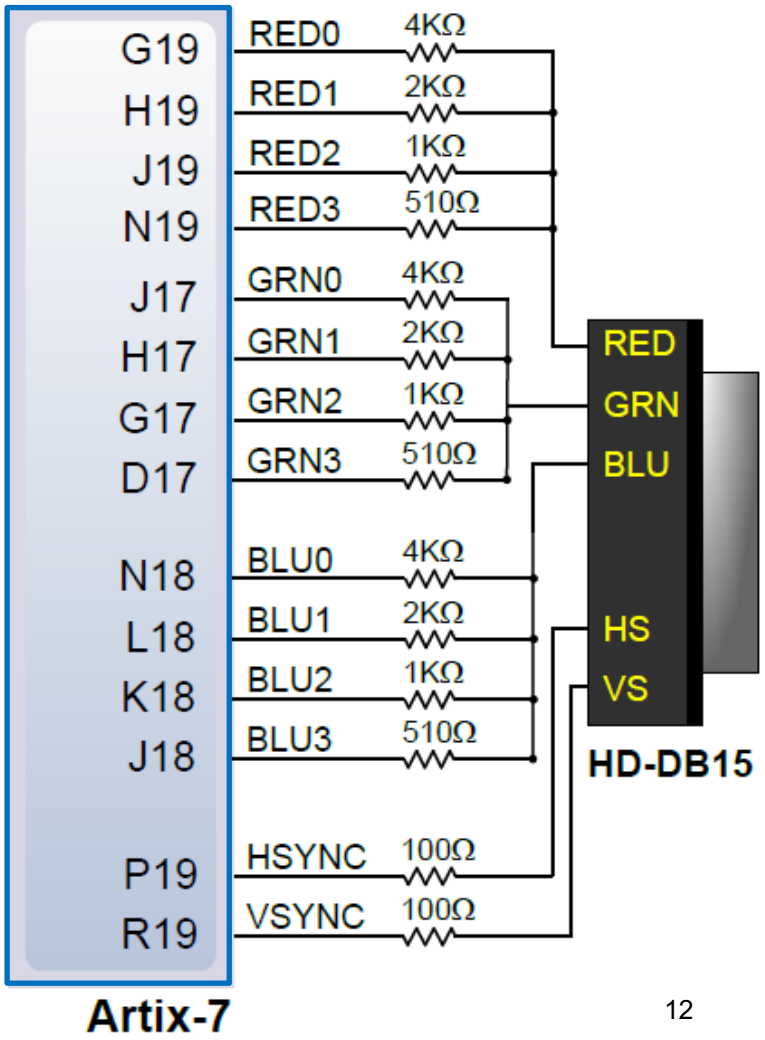
- A VGA video signal contains 5 active signals:
  - Horizontal synchronization (HS, or Hsync)
  - Vertical synchronization (VS, or Vsync)
  - Red (R): 0v ~ 0.7v
  - Green (G): 0v ~ 0.7v
  - Blue (B): 0v ~ 0.7v



Pin 1: Red
Pin 2: Grn
Pin 3: Blue
Pin 13: HS
Pin 14: VS

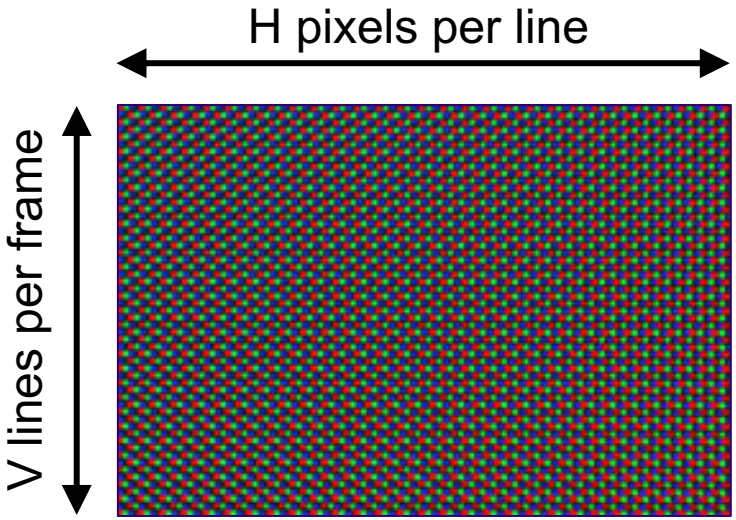
Pin Assignment and The XDC File

```
##VGA Connector
set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]
set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]
set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]
set_property PACKAGE_PIN P19 [get_ports hsync]
set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19 [get_ports vsync]
set_property IOSTANDARD LVCMOS33 [get_ports vsync]
```

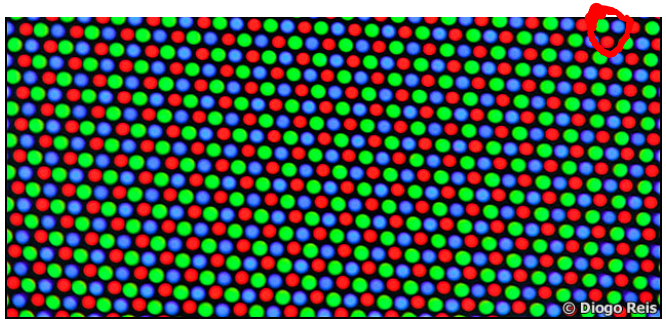


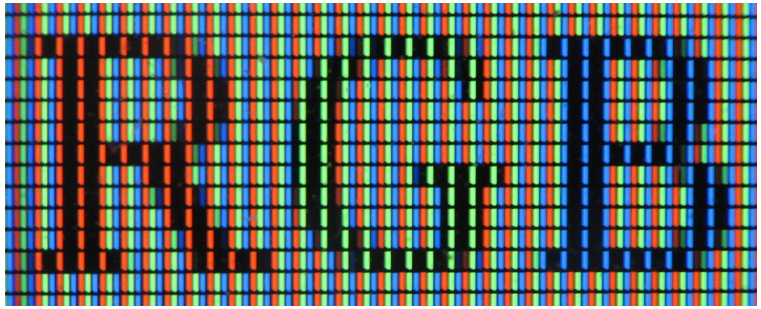
General Video Display Format

- A color video display is a 2-dimensional grid of elements
  - Each element is called a **pixel**
  - Each pixel is made up of red, green, and **blue** (RGB)
  - The relative intensities of RGB determines the **color**



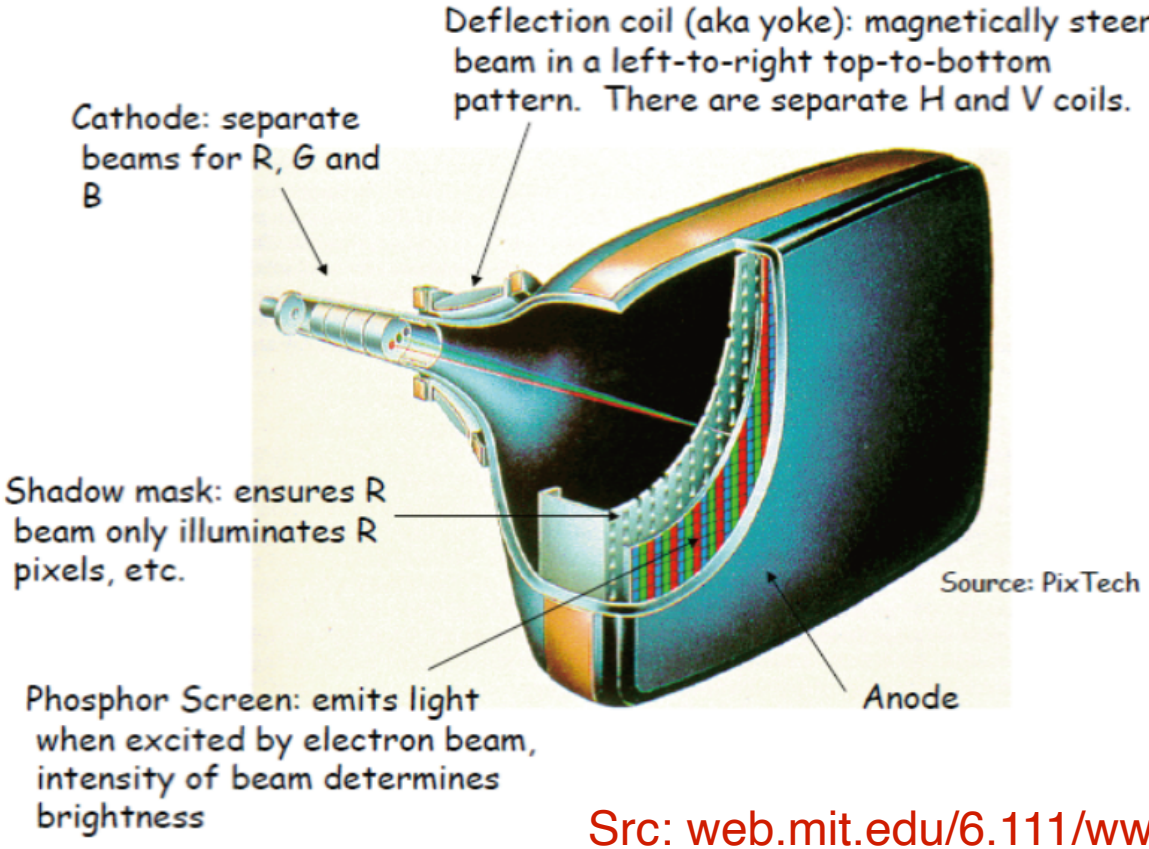
H/V => 4/3, 16/9, 16/10, …

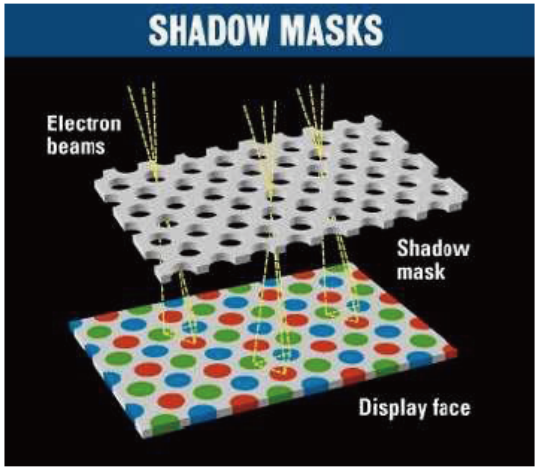




Cathode Ray Tube (CRT)

- A vacuum tube containing one or more electron guns, and a phosphorescent screen to view images

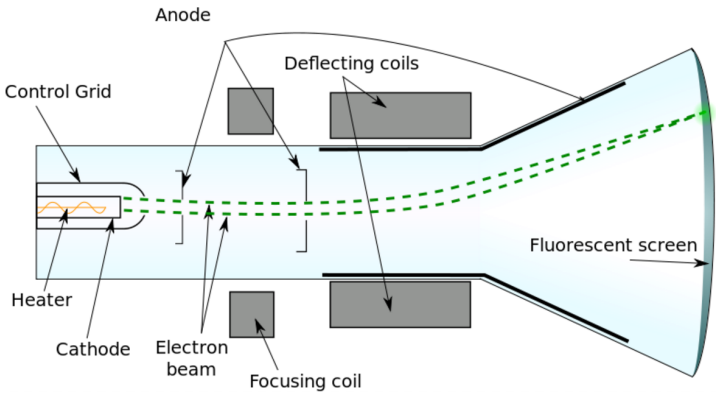


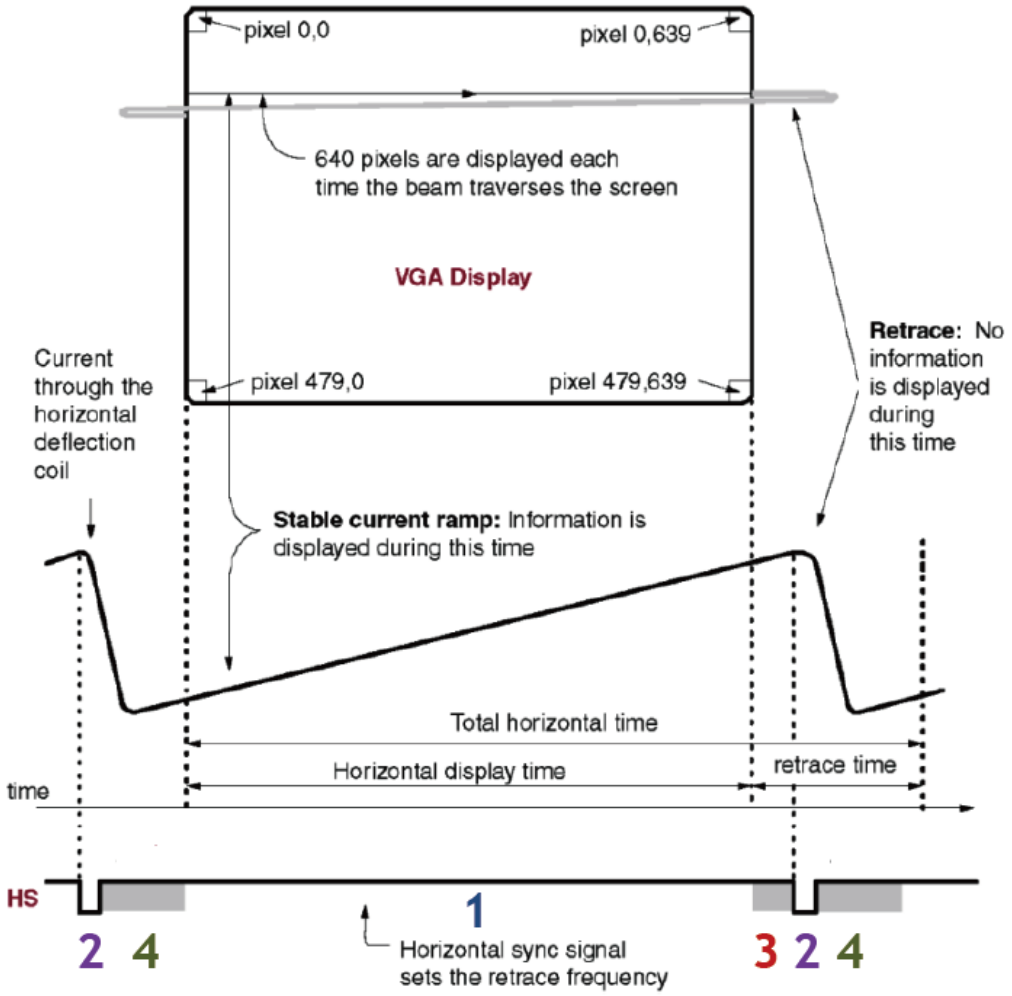


Src: web.mit.edu/6.111/www/f2008/ and www.comp.dit.ie

VGA System Timing (1/4)

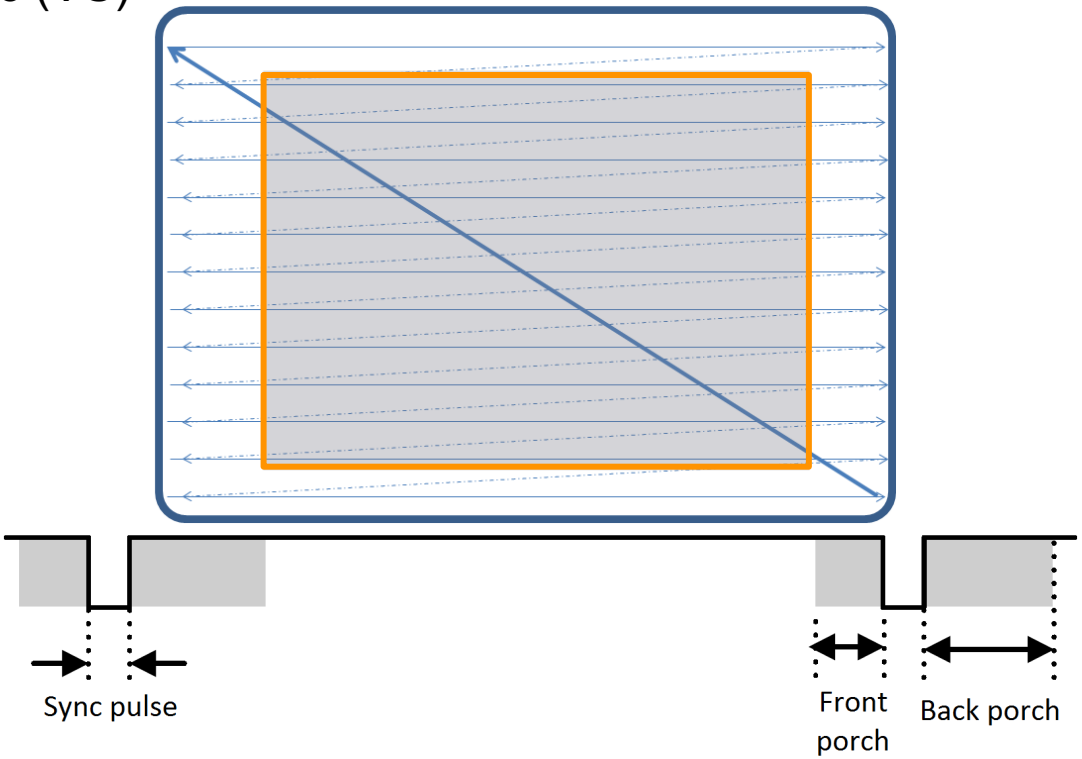
- Displays 640 x 480 pixels
- Scans the screen back and forth
- Four components (horizontal)
  - 1. Screen display time
  - 2. Sync pulse
  - 3. Front porch
  - 4. Back porch
- Retrace time = **2 + 3 + 4**
- Total display time = **1 + 2 + 3 + 4**





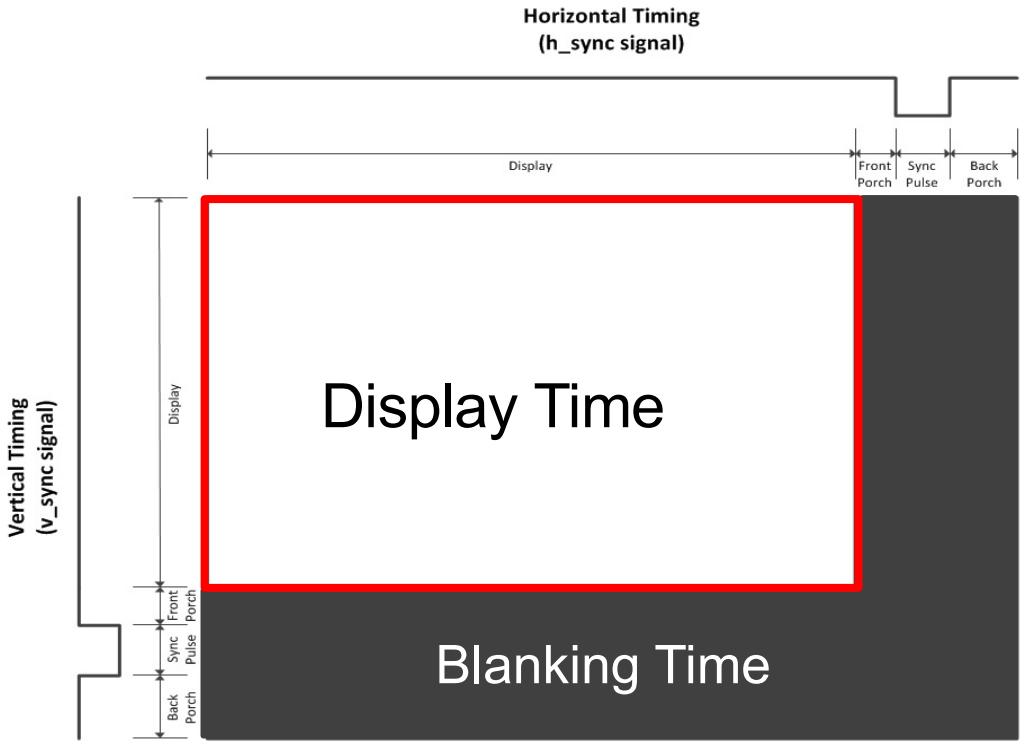
VGA System Timing (2/4)

- Scans through each horizontal lines first
- Pull back the pointer to the beginning of the line at the end
- Two control signals
  - Hsync (HS)
  - Vsync (VS)

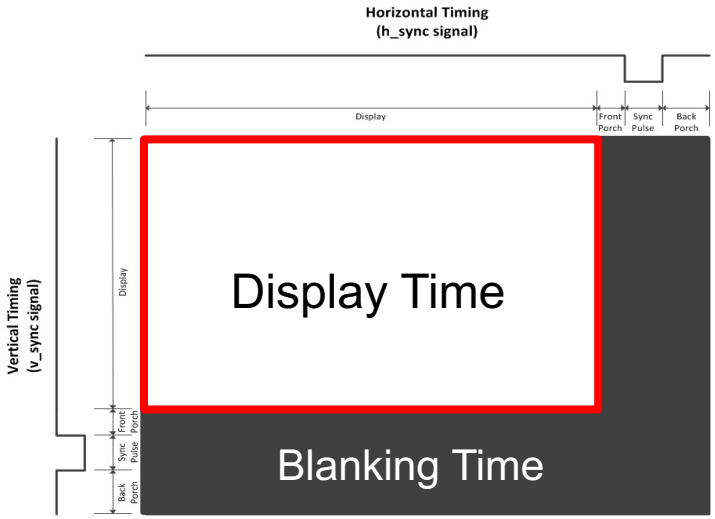


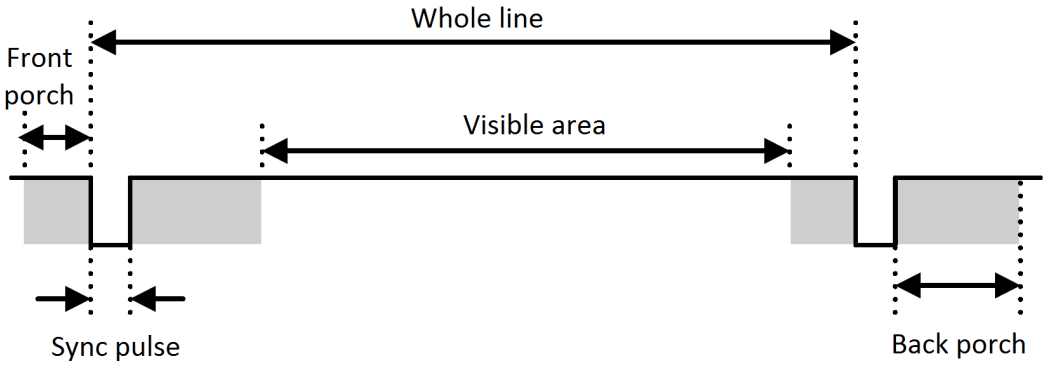
VGA System Timing (3/4)

- Digitalized control
- Signal timing for a 640-pixel by 480 rows display using a 25MHz pixel clock
- Higher resolution can be achieved via different timing spec



VGA System Timing (3/3)





Parameter	Ver. Sync		Hor. Sync	
	Lines	Time (ms)	Pixels	Time (μs)
Visible Area	480	15.3	640	25
Front Porch	10	0.3	16	0.64
Sync Pulse	2	0.064	96	3.8
Back Porch	33	1.05	48	1.9
Whole Line	525	16.7	800	32

Pixel Clock

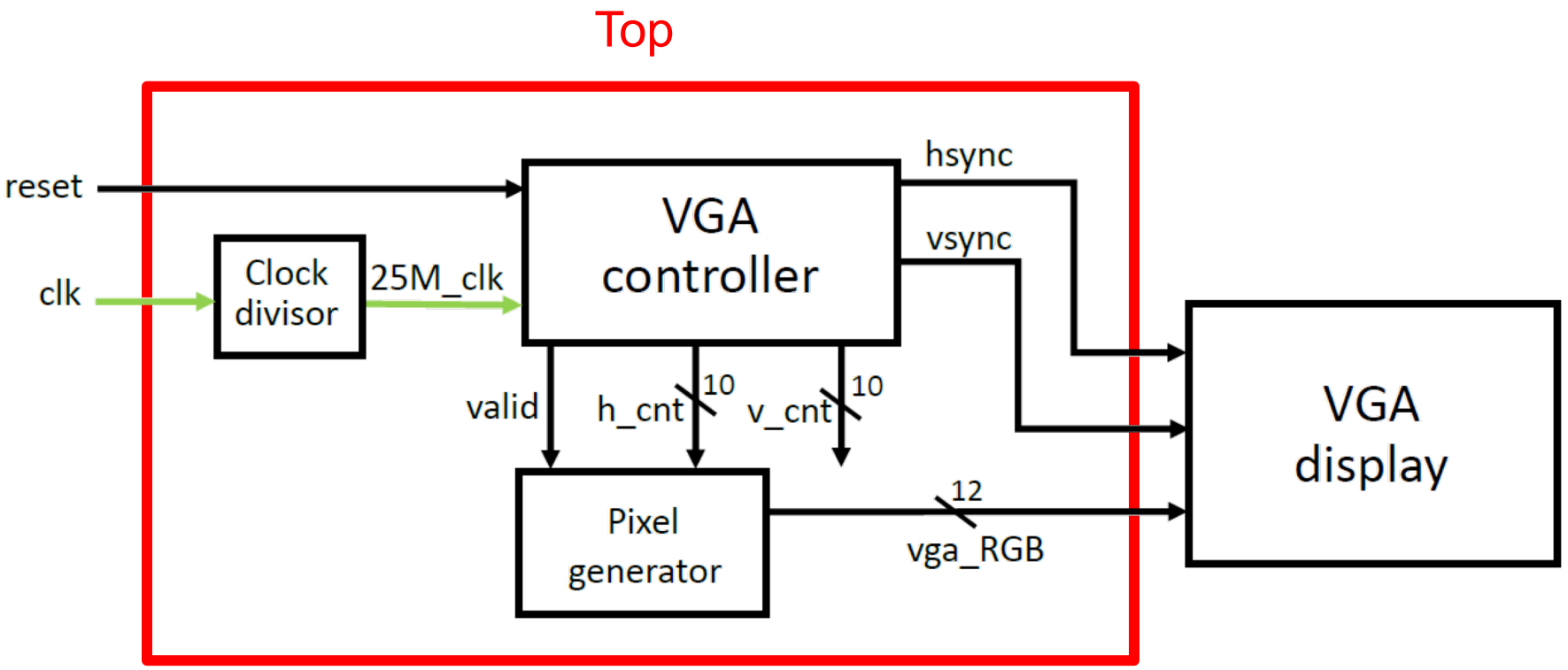
- The clock frequency used to display one pixel on the screen
  - Use a clock divider to generate it
- Clock frequency depends on number of pixels and refresh rate
- For 640 x 480 pixels with 60Hz refresh rate, the clock frequency is:

  800 x 525 x 60 (frame/sec) = 25M (pixel/sec)

- The 25MHz clock frequency can be generated by the 100MHz FPGA frequency (divide-by-four)

Demo 1: Monitor Test

Demo 1: Block Diagram



Demo 1: Top Module

```
module top(
    input clk,
    input rst,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output hsync,
    output vsync
     );
     wire clk_25MHz;
     wire valid;
     wire [9:0] h_cnt; //640
     wire [9:0] v_cnt;  //480

      clock_divisor clk_wiz_0_inst(
       .clk(clk),
       .clk1(clk_25MHz)
      );

    pixel_gen pixel_gen_inst(
        .h_cnt(h_cnt),
        .valid(valid),
        .vgaRed(vgaRed),
        .vgaGreen(vgaGreen),
        .vgaBlue(vgaBlue)
      );

      vga_controller   vga_inst(
       .pclk(clk_25MHz),
       .reset(rst),
       .hsync(hsync),
       .vsync(vsync),
       .valid(valid),
       .h_cnt(h_cnt),
       .v_cnt(v_cnt)
      );

endmodule
```

Demo 1: Clock Divisor

```
module clock_divisor(clk1, clk);
input clk;
output clk1;
reg [1:0] num;
wire [1:0] next_num;

always @(posedge clk) begin
  num <= next_num;
end

assign next_num = num + 1'b1;
assign clk1 = num[1];
endmodule
```

Demo 1: Pixel Generator

```
module pixel_gen(
    input [9:0] h_cnt,
    input valid,
    output reg [3:0] vgaRed,
    output reg [3:0] vgaGreen,
    output reg [3:0] vgaBlue
    );

        always @(*) begin
        if(!valid)
                {vgaRed, vgaGreen, vgaBlue} = 12'h0;
         else if(h_cnt < 128)
                {vgaRed, vgaGreen, vgaBlue} = 12'h000;
         else if(h_cnt < 256)
                {vgaRed, vgaGreen, vgaBlue} = 12'h00f;
         else if(h_cnt < 384)
                {vgaRed, vgaGreen, vgaBlue} = 12'hf00;
         else if(h_cnt < 512)
                {vgaRed, vgaGreen, vgaBlue} = 12'h0f0;
         else if(h_cnt < 640)
                {vgaRed, vgaGreen, vgaBlue} = 12'hfff;
         else
                {vgaRed, vgaGreen, vgaBlue} = 12'h0;
    end
endmodule
```

VGA Controller (1/3)

```
module vga_controller (
  input wire pclk, reset,
  output wire hsync, vsync, valid,
  output wire [9:0] h_cnt,
  output wire [9:0] v_cnt
);
  reg [9:0] pixel_cnt;
  reg [9:0] line_cnt;
  reg hsync_i, vsync_i;
```

```
  parameter HD = 640;
  parameter HF = 16;
  parameter HS = 96;
  parameter HB = 48;
  parameter HT = 800;
  parameter VD = 480;
  parameter VF = 10;
  parameter VS = 2;
  parameter VB = 33;
  parameter VT = 525;
  parameter hsync_default = 1'b1;
  parameter vsync_default = 1'b1;
```

VGA Controller (2/3)

```
always @(posedge pclk)
  if (reset)
    pixel_cnt <= 0;
  else
    if(pixel_cnt < (HT - 1))
      pixel_cnt <= pixel_cnt + 1;
    else
      pixel_cnt <= 0;

always @(posedge pclk)
  if (reset)
    hsync_i <= hsync_default;
  else
    if ((pixel_cnt >= (HD + HF - 1))
       && (pixel_cnt < (HD + HF + HS
       - 1)))
      hsync_i <= ~hsync_default;
    else
      hsync_i <= hsync_default;
```

```
always @(posedge pclk)
  if (reset)
    line_cnt <= 0;
  else
    if (pixel_cnt == (HT - 1))
      if (line_cnt < (VT - 1))
        line_cnt <= line_cnt + 1;
      else
        line_cnt <= 0;

always @(posedge pclk)
  if (reset)
    vsync_i <= vsync_default;
  else
    if ((line_cnt >= (VD + VF - 1)) &&
      (line_cnt < (VD + VF + VS - 1)))
      vsync_i <= ~vsync_default;
    else
      vsync_i <= vsync_default;
```

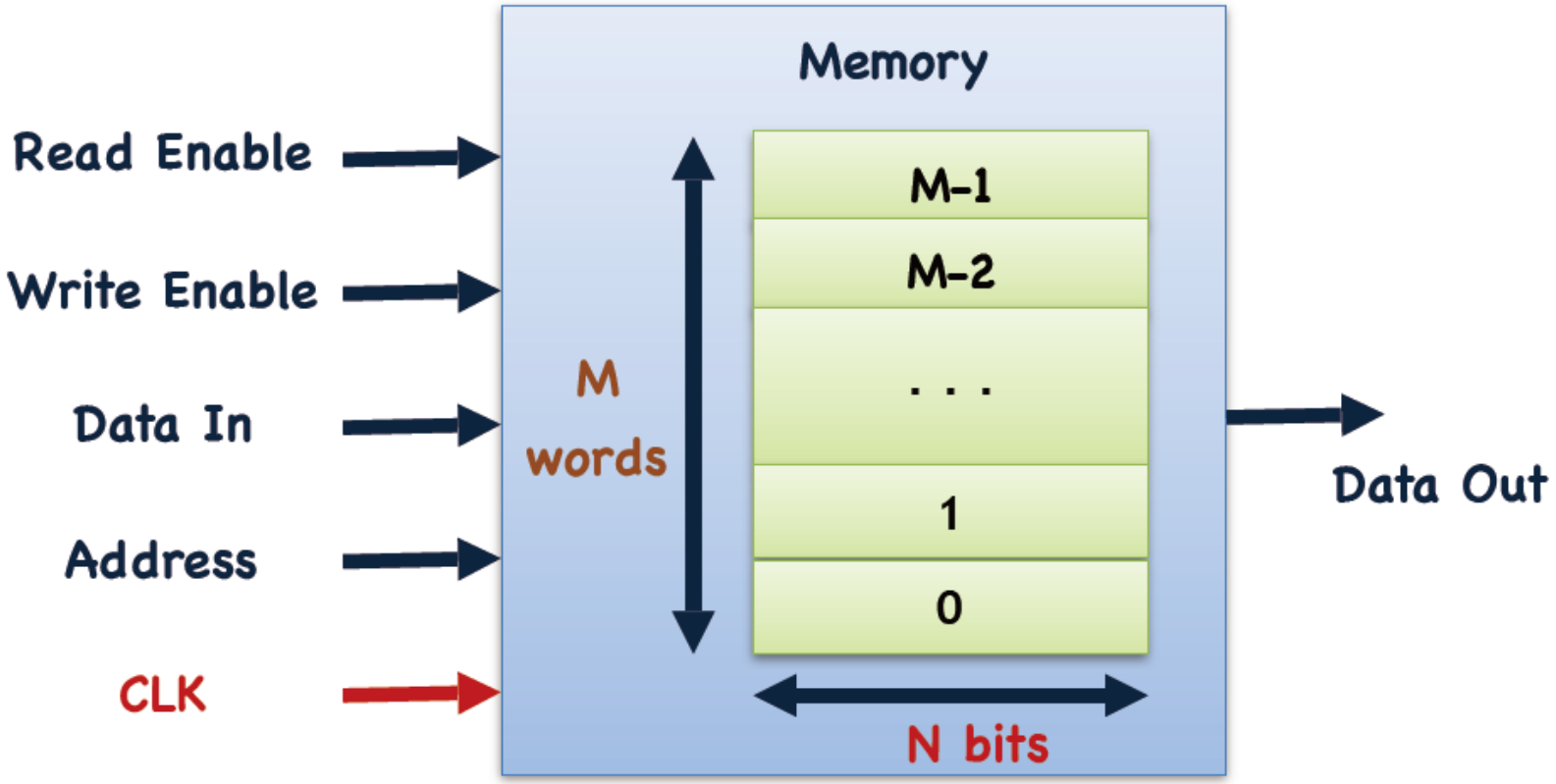
VGA Controller (3/3)

```
  assign hsync = hsync_i;
  assign vsync = vsync_i;
  assign valid = ((pixel_cnt < HD) && (line_cnt < VD));
  assign h_cnt = (pixel_cnt < HD) ? pixel_cnt : 10'd0;
  assign v_cnt = (line_cnt < VD) ? line_cnt : 10'd0;

endmodule
```

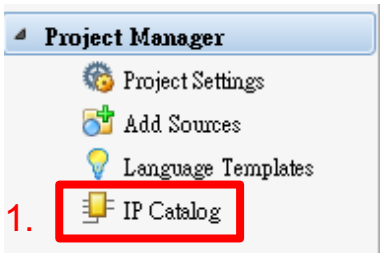
Memory Array (as Frame Buffer)

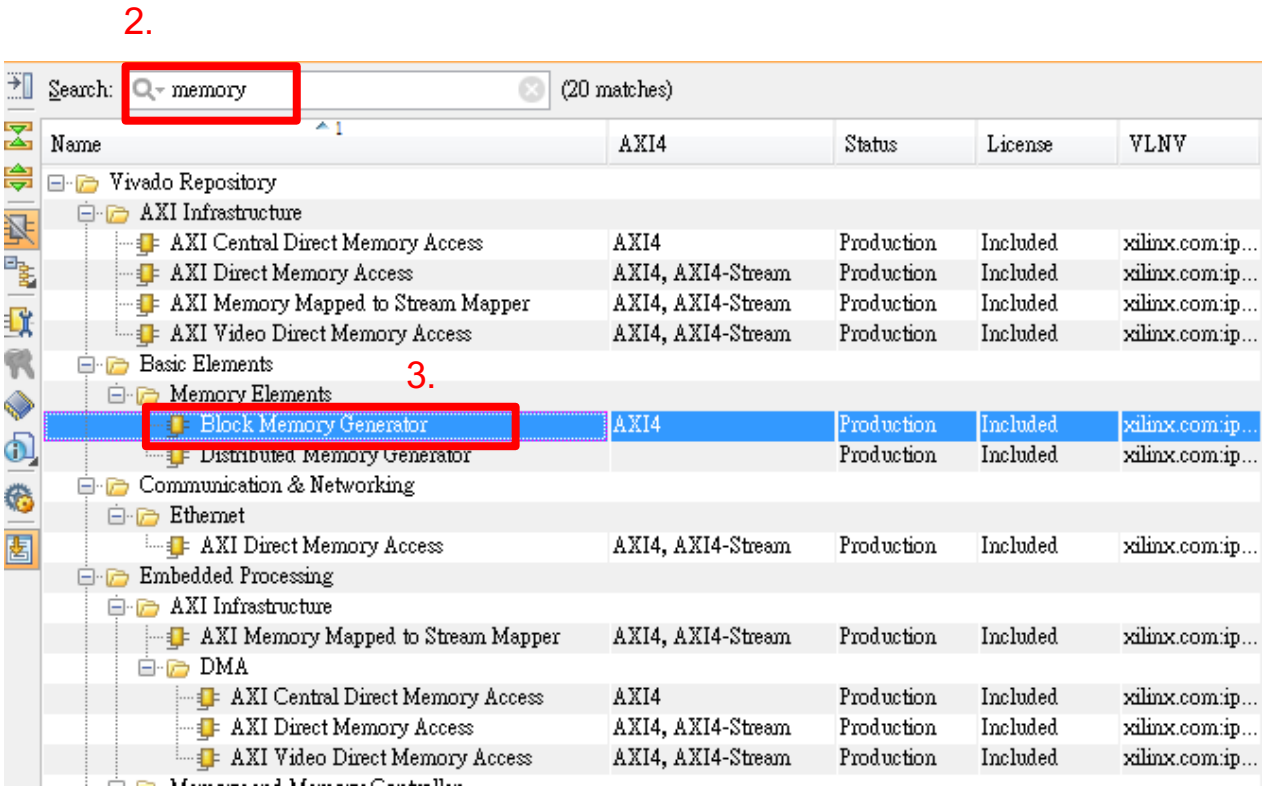
- Two dimensional storage elements
- Stores M numbers of words
- Each element is N-bit wide



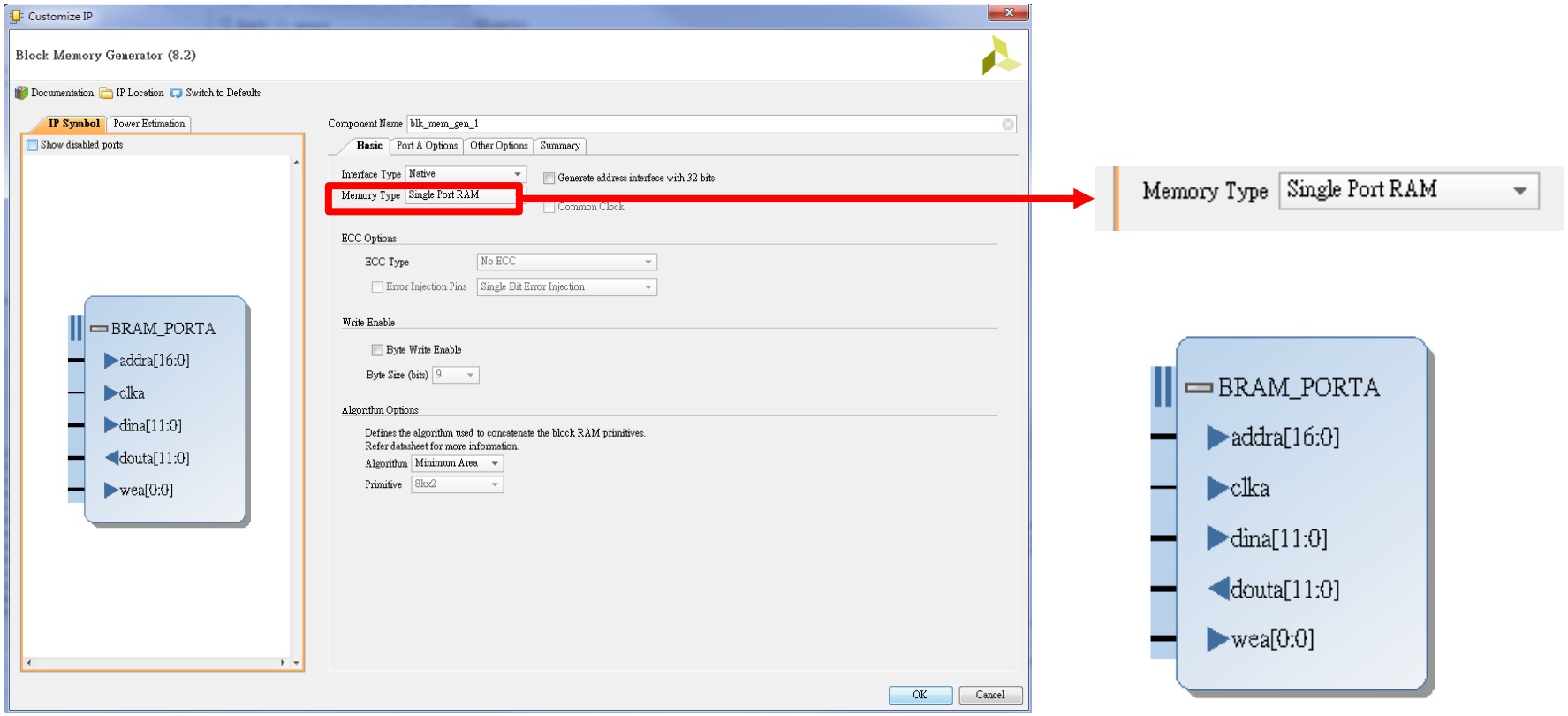
Memory IP (1/5)

- Alternative way to create the memory to store the image data.

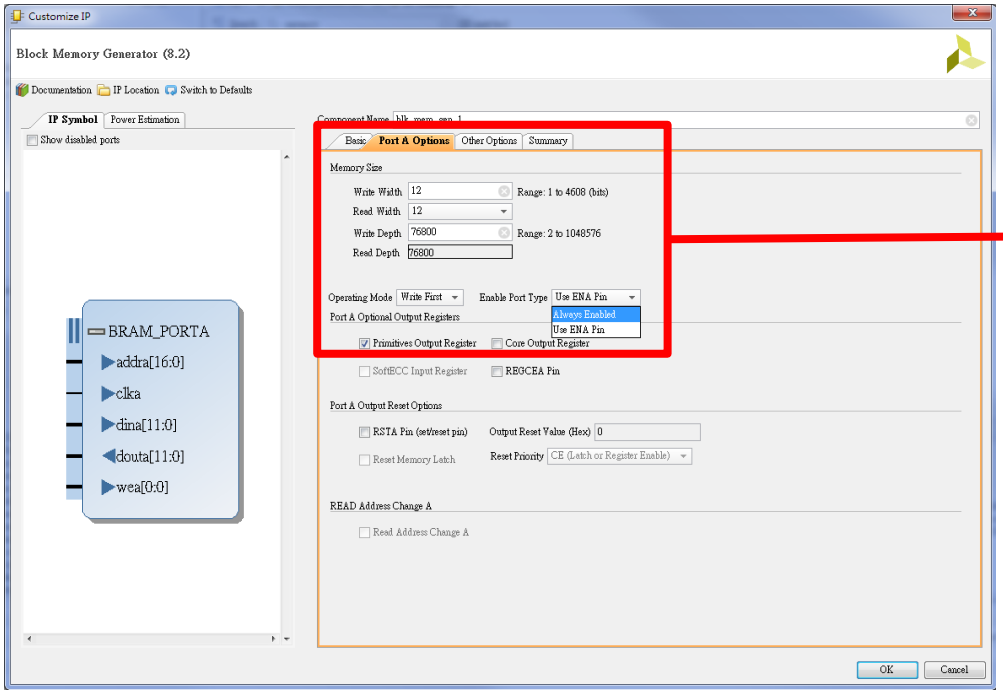


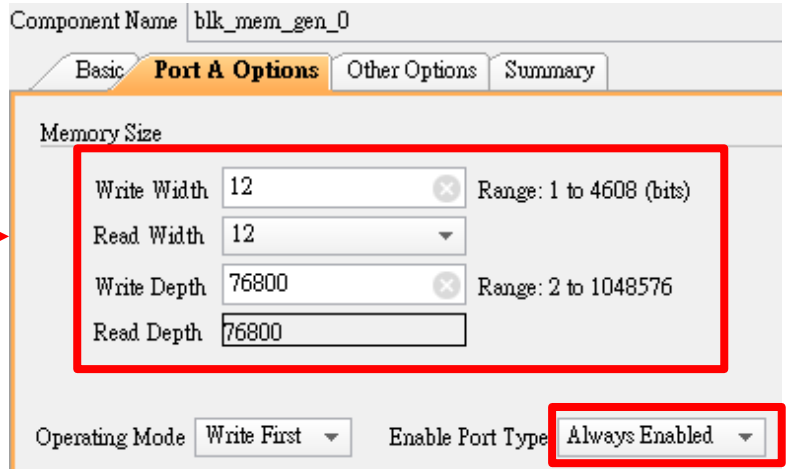


Memory IP (2/5)


Customize IP
Block Memory Generator (8.2)
Documentation
IP Location
Switch to Defaults
IP Symbol
Power Estimation
Show disabled ports
BRAM_PORTA
addra[16:0]
clka
dina[11:0]
douta[11:0]
wea[0:0]
Component Name
blk_mem_gen_1
Basic
Port A Options
Other Options
Summary
Interface Type
Native
Generate address interface with 32 bits
Memory Type
Single Port RAM
Common Clock
ECC Options
ECC Type
No ECC
Error Injection Pins
Single Bit Error Injection
Write Enable
Byte Write Enable
Byte Size (bits)
9
Algorithm Options
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.
Algorithm
Minimum Area
Primitive
8kx2
OK
Cancel
Memory Type
Single Port RAM
BRAM_PORTA
addra[16:0]
clka
dina[11:0]
douta[11:0]
wea[0:0]

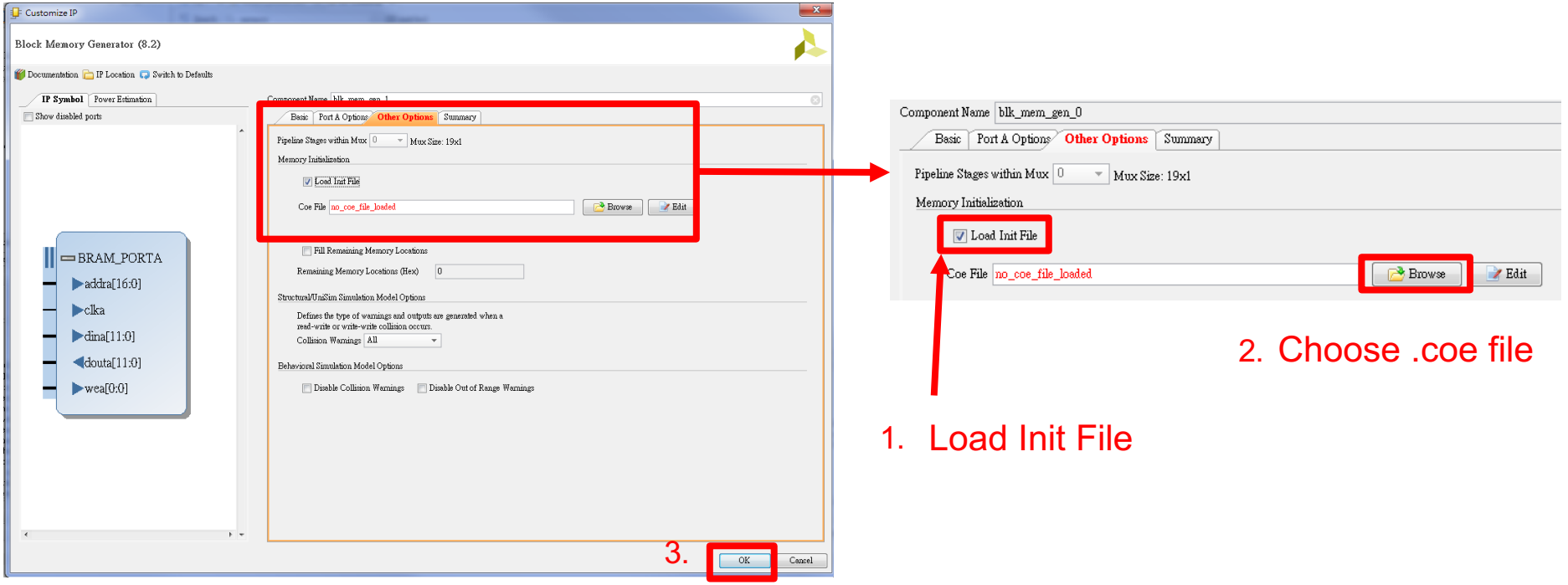
Memory IP (3/5)



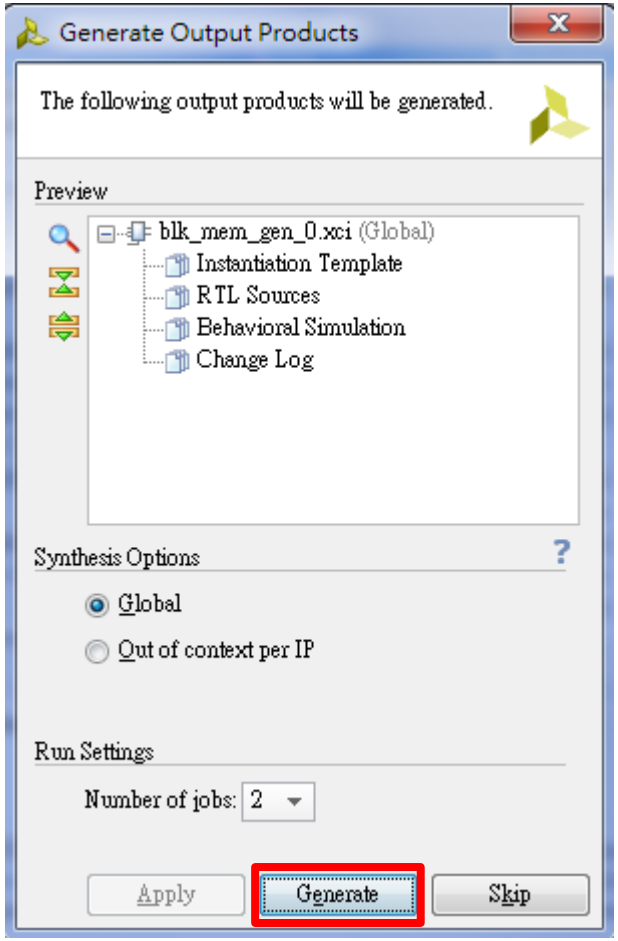


RGB : 12 bits
320x240 : 76800

Memory IP (4/5)


Customize IP
Block Memory Generator (8.2)
Documentation
IP Location
Switch to Defaults
IP Symbol
Power Estimation
Show disabled ports
BRAM_PORTA
addra[16:0]
clka
dina[11:0]
douta[11:0]
wea[0:0]
Basic
Port A Options
Other Options
Summary
Pipeline Stages within Mux
Mux Size: 19x1
Memory Initialization
Load Init File
Coe File
no_coe_file_loaded
Browse
Edit
Fill Remaining Memory Locations
Remaining Memory Locations (Hex)
Structural/UniSim Simulation Model Options
Collision Warnings
All
Behavioral Simulation Model Options
Disable Collision Warnings
Disable Out of Range Warnings
OK
Cancel
3.
Component Name
blk_mem_gen_0
1. Load Init File
2. Choose .coe file

Memory IP (5/5)


Generate Output Products
The following output products will be generated.
Preview
blk_mem_gen_0.xci (Global)
Instantiation Template
RTL Sources
Behavioral Simulation
Change Log
Synthesis Options
Global
Out of context per IP
Run Settings
Number of jobs: 2
Apply
Generate
Skip

Picture Format Translation (1/2)

kanahei.jpg

PicTrans.exe

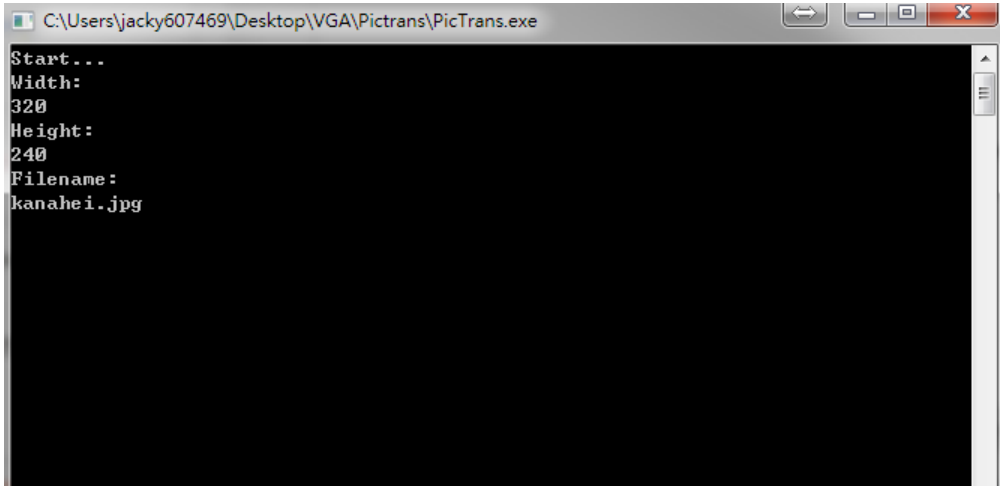
out.coe

kanahei.coe

```
1   memory_initialization_radix=16;
2   memory_initialization_vector=
3   FFF,
4   FFF,
5   FFF,
6   FFF,
7   FFF,
8   FFF,
9   FFF,
10  FFF,
11  FFF,
12  FFF,
    ⋮   ⋮
47834  CAA,
47835  200,
47836  200,
47837  311,
47838  B99,
47839  FDD,
47840  422,
```

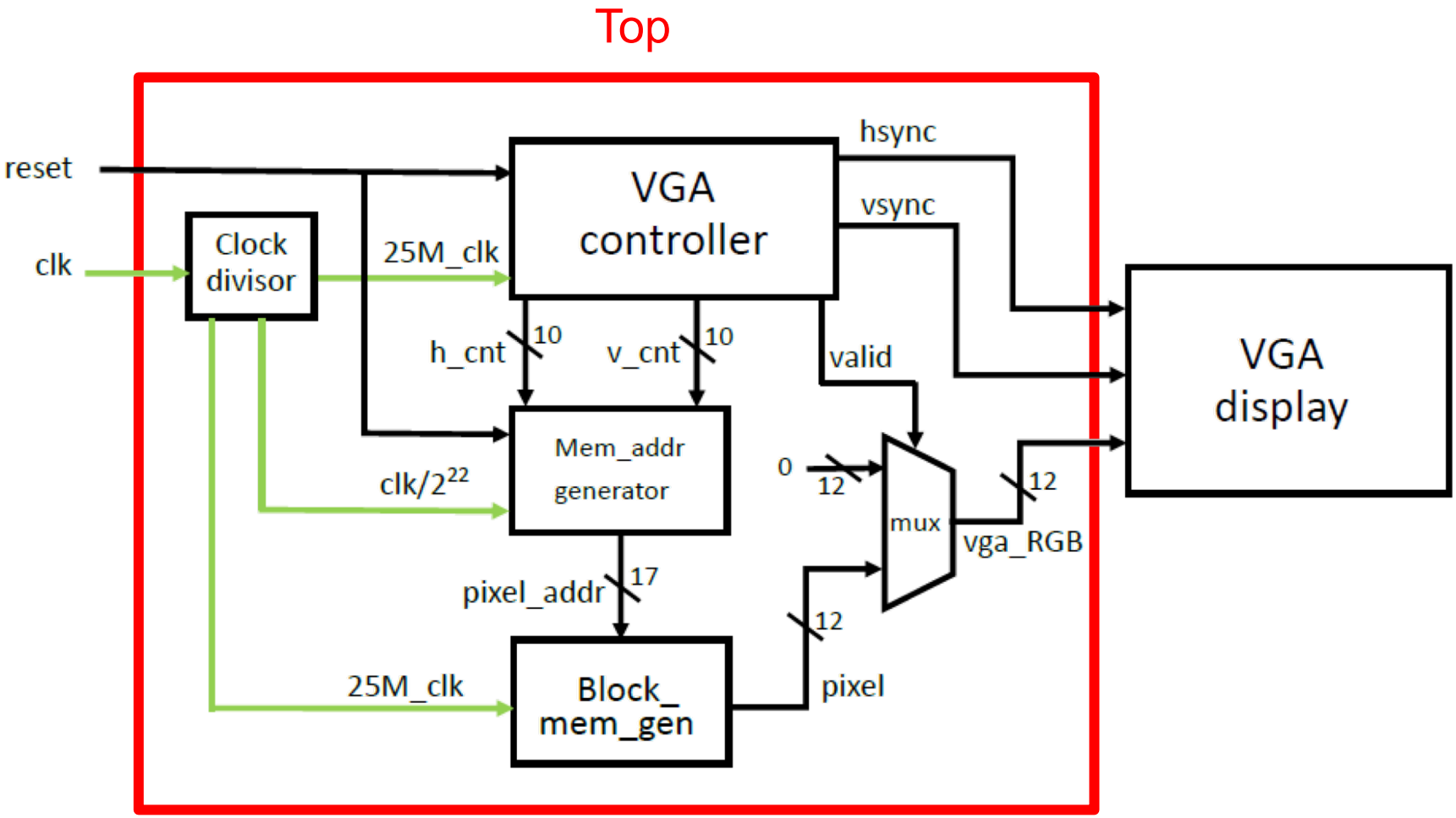
Picture Format Translation (2/2)

- PicTrans.exe:
  - Convert a *.jpg file to a bit map file
- Input:
  - Image (*.jpg)
  - The width of the output file
  - The height of the output file
- Output:
  - out.coe



Demo 2: Video (Advertising Lightbox)

Demo 2: Block Diagram



Demo 2: Top Module

```
module top(
    input clk,
    input rst,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output hsync,
    output vsync
     );

     wire [11:0] data;
     wire clk_25MHz;
     wire clk_22;
     wire [16:0] pixel_addr;
     wire [11:0] pixel;
     wire valid;
     wire [9:0] h_cnt; //640
     wire [9:0] v_cnt;  //480

assign {vgaRed, vgaGreen, vgaBlue}
        = (valid==1'b1) ? pixel:12'h0;
```

```
clock_divisor clk_wiz_0_inst(
  .clk(clk),
  .clk1(clk_25MHz),
  .clk22(clk_22)
);

mem_addr_gen mem_addr_gen_inst(
.clk(clk_22),
.rst(rst),
.h_cnt(h_cnt),
.v_cnt(v_cnt),
.pixel_addr(pixel_addr)
);


blk_mem_gen_0 blk_mem_gen_0_inst(
  .clka(clk_25MHz),
  .wea(0),
  .addra(pixel_addr),
  .dina(data[11:0]),
  .douta(pixel)
);
```

```
vga_controller   vga_inst(
  .pclk(clk_25MHz),
  .reset(rst),
  .hsync(hsync),
  .vsync(vsync),
  .valid(valid),
  .h_cnt(h_cnt),
  .v_cnt(v_cnt)
);

endmodule
```

Demo 2: Clock Divisor

```
module clock_divisor(clk1, clk, clk22);
input clk;
output clk1;
output clk22;
reg [21:0] num;
wire [21:0] next_num;

always @(posedge clk) begin
  num <= next_num;
end

assign next_num = num + 1'b1;
assign clk1 = num[1];
assign clk22 = num[21];
endmodule
```

Demo 2: Memory Address Generator

```
module mem_addr_gen(
    input clk,
    input rst,
    input [9:0] h_cnt,
    input [9:0] v_cnt,
    output [16:0] pixel_addr
    );

    reg [7:0] position;

    assign pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;  //640*480 --> 320*240

    always @ (posedge clk or posedge rst) begin
        if(rst)
            position <= 0;
        else if(position < 239)
            position <= position + 1;
        else
            position <= 0;
    end

endmodule
```

Size of Frame Buffer Matters

- 640 x 480 = 307,200 pixels in total
  - 640 x 480 x 12 = 3,686,400 bits!!
  - Artix-7 35T on Basys 3 has
    - 5,200 slices: each slice has four 6-input LUTs and 8 flip-flops
    - 1,800 Kbits of fast block RAM (BRAM)
- => Reduced display resolution
  - 320 x 240 x 12 => 921,600 bits
- => Or using some smart compression technique
  - We leave it to you