

Video

Prof. Stephen A. Edwards
sedwards@cs.columbia.edu

Columbia University
Spring 2010

Television: 1939 Du Mont Model 181



110 volt 60 cycle power lines. Two Du Mont Teletron are employed in dynamic speaker is used for perfect sound a three-band superheterodyne all wave radio reception. This receiver employs 110 volt 60 cycle power tuning are provided. An individual broadcast sound reproduction.

Model 181 is a high console model and sound entertainment with a selection. The black and white picture is projected on the screen of the 14 inch by 10 inches. The beautifully designed modern design measures 48 3/8 inches high and 16 inches deep. It is completely a

V

SCORE

000

HIGH SCORE

5000

GAME OVER

COIN

PLAY

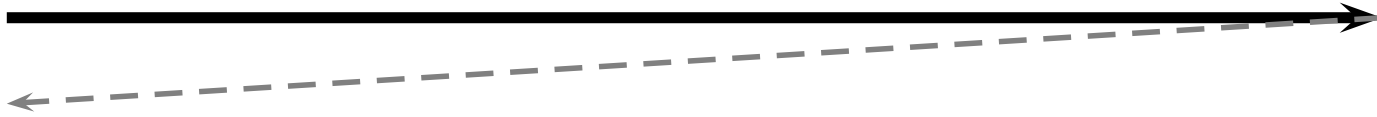
INSERT COIN

© ATARI 1980

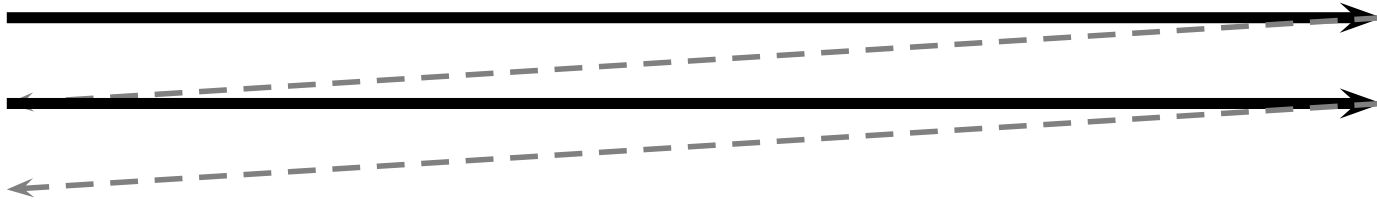
Raster Scanning



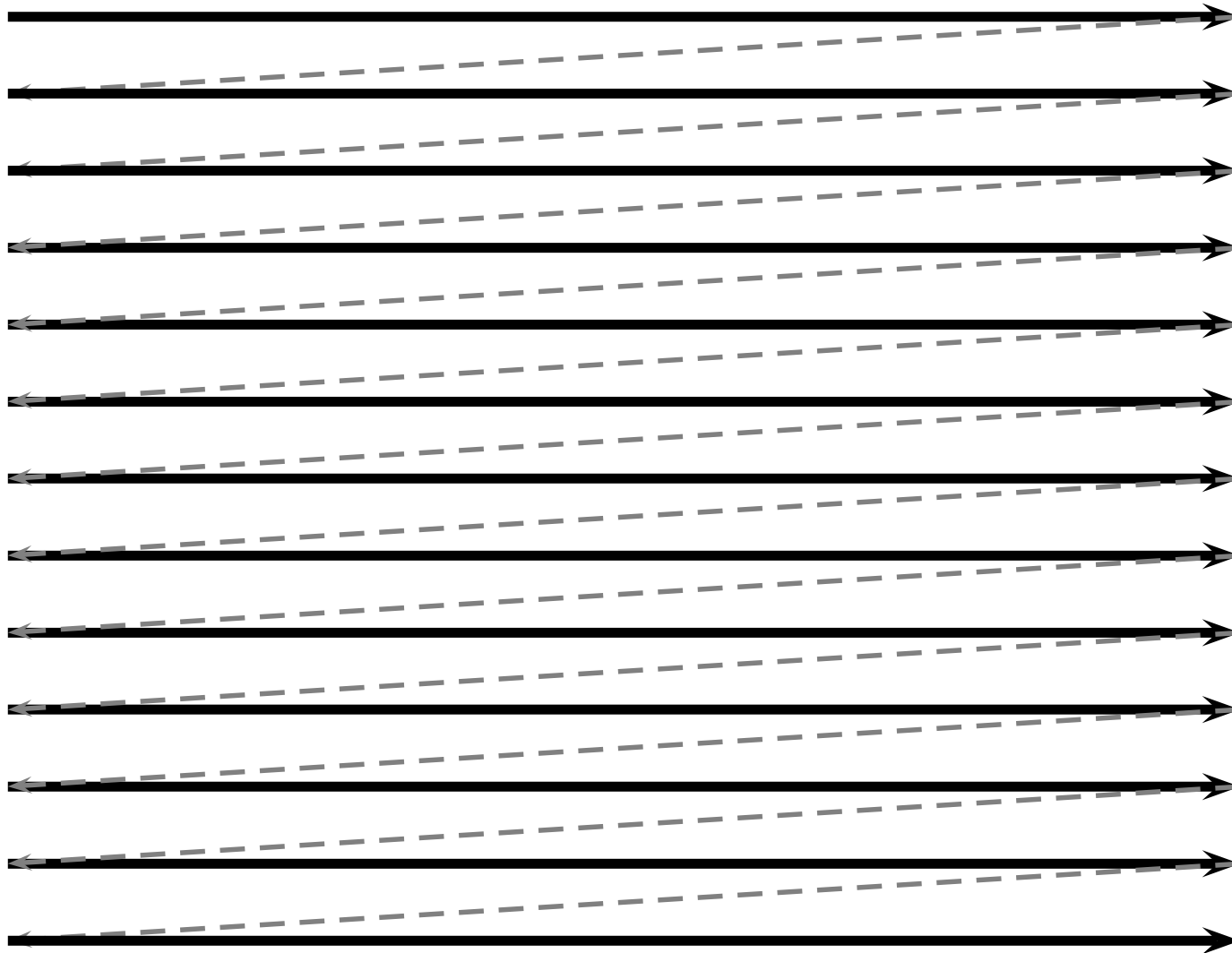
Raster Scanning



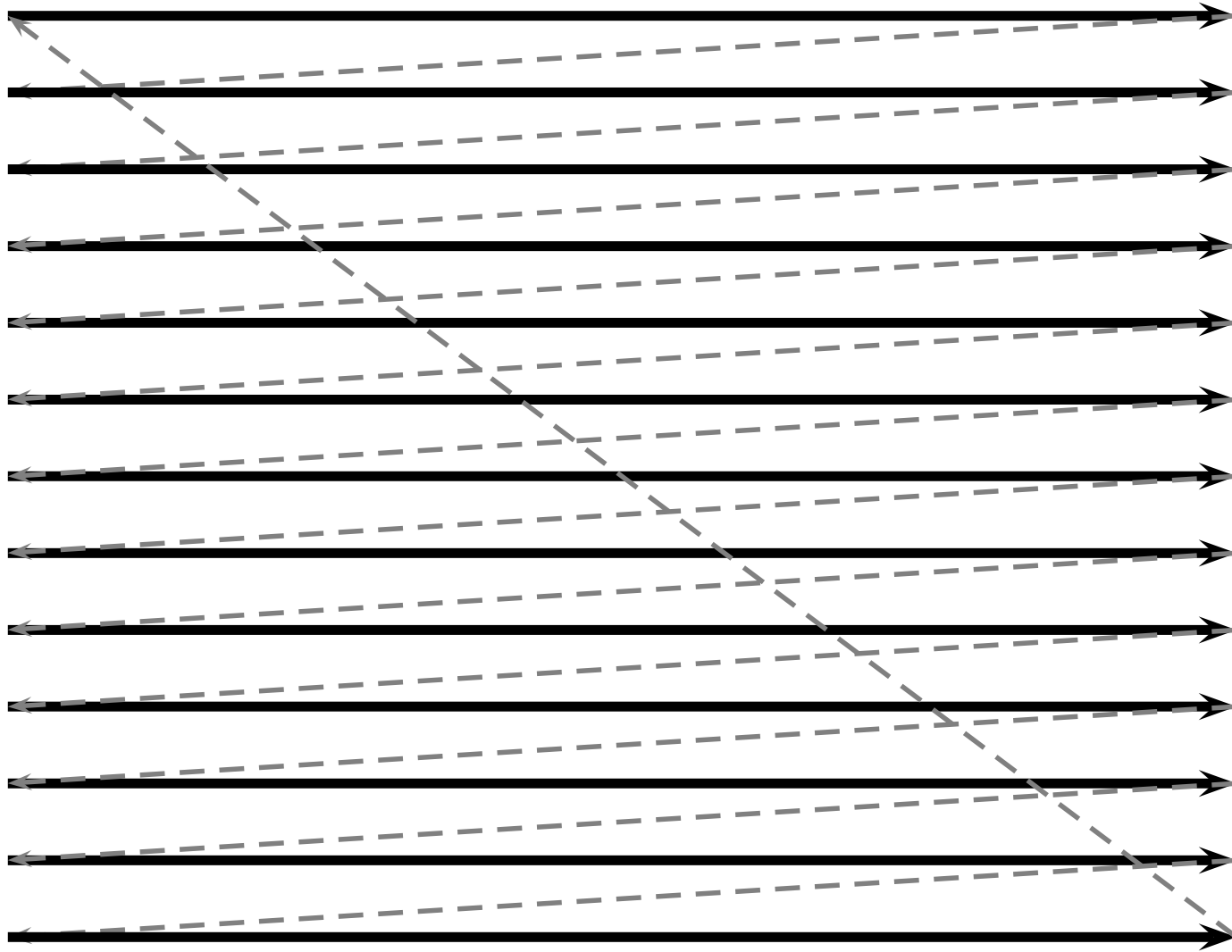
Raster Scanning



Raster Scanning



Raster Scanning



NTSC or RS-170

Originally black-and-white

60 Hz vertical scan frequency

15.75 kHz horizontal frequency

$$\frac{15.75 \text{ kHz}}{60 \text{ Hz}} = 262.5 \text{ lines per field}$$

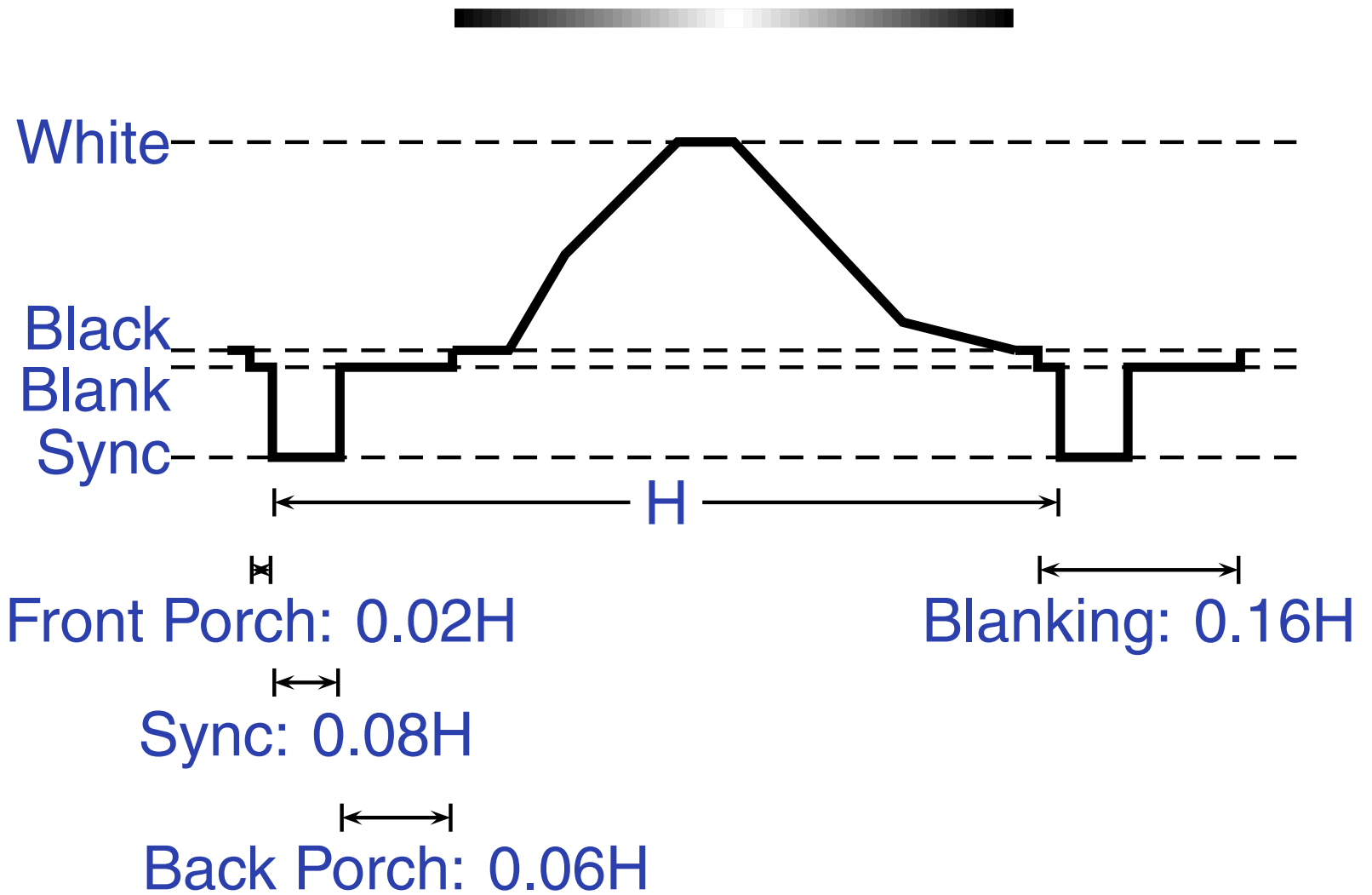
White 1 V

Black 0.075 V

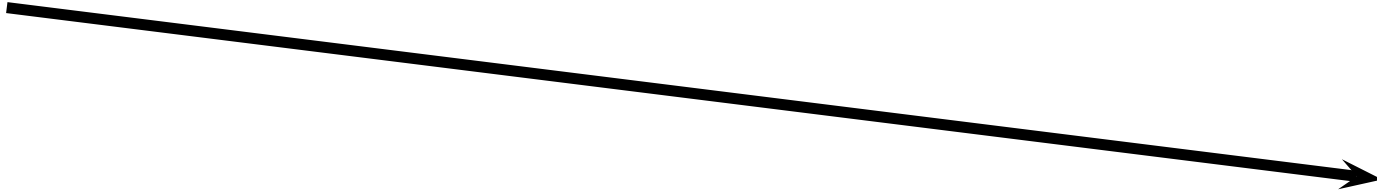
Blank 0 V

Sync – 0.4 V

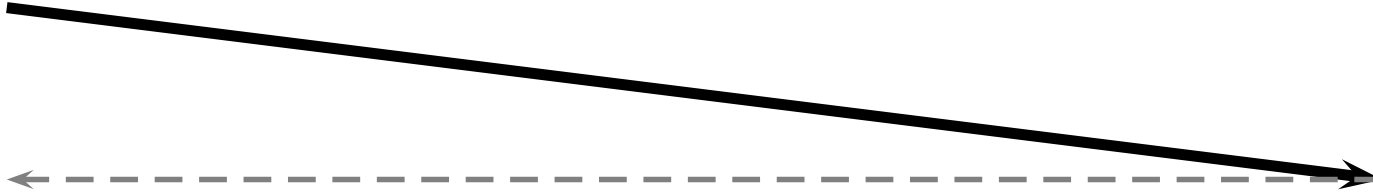
A Line of B&W Video



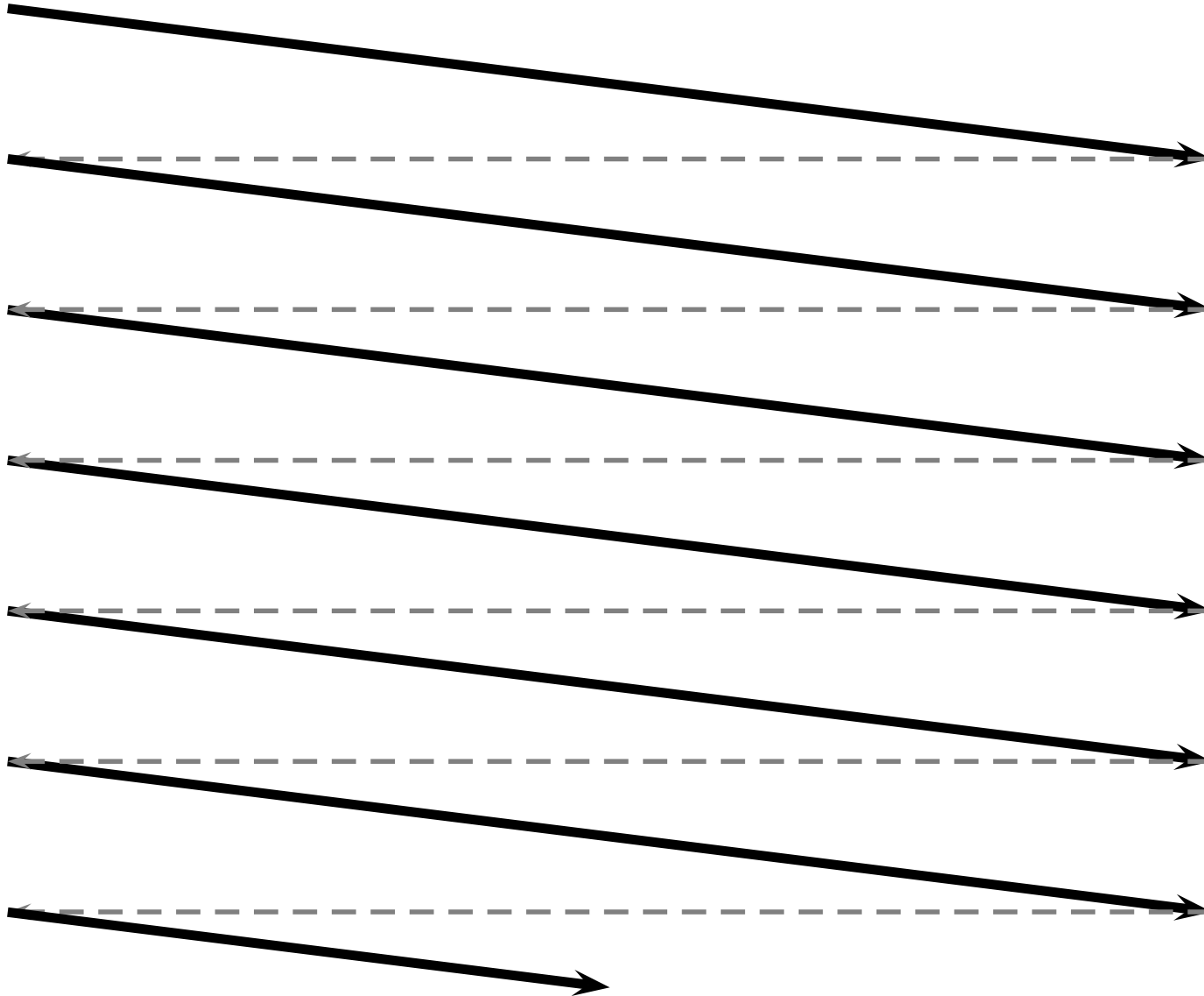
Interlaced Scanning



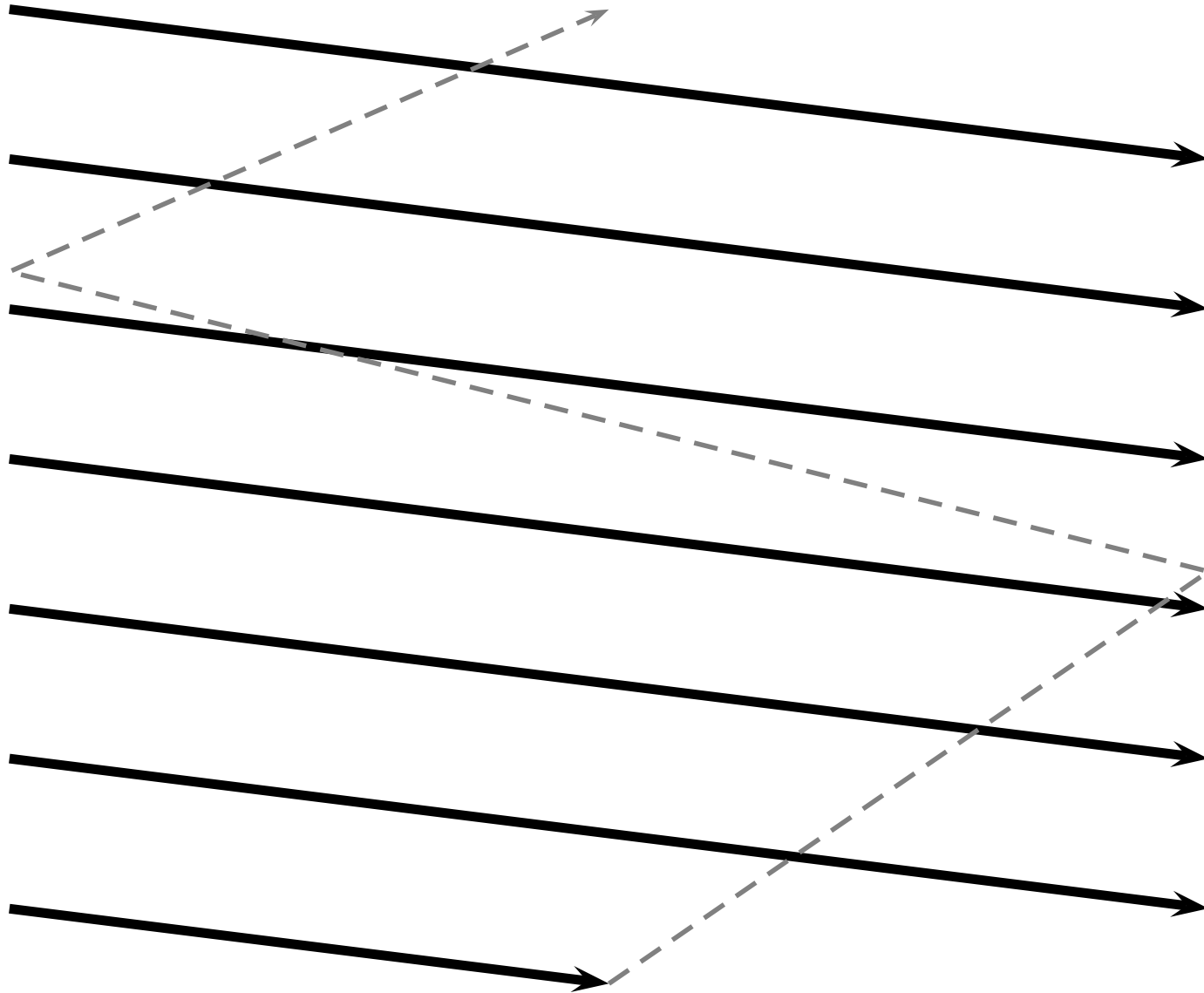
Interlaced Scanning



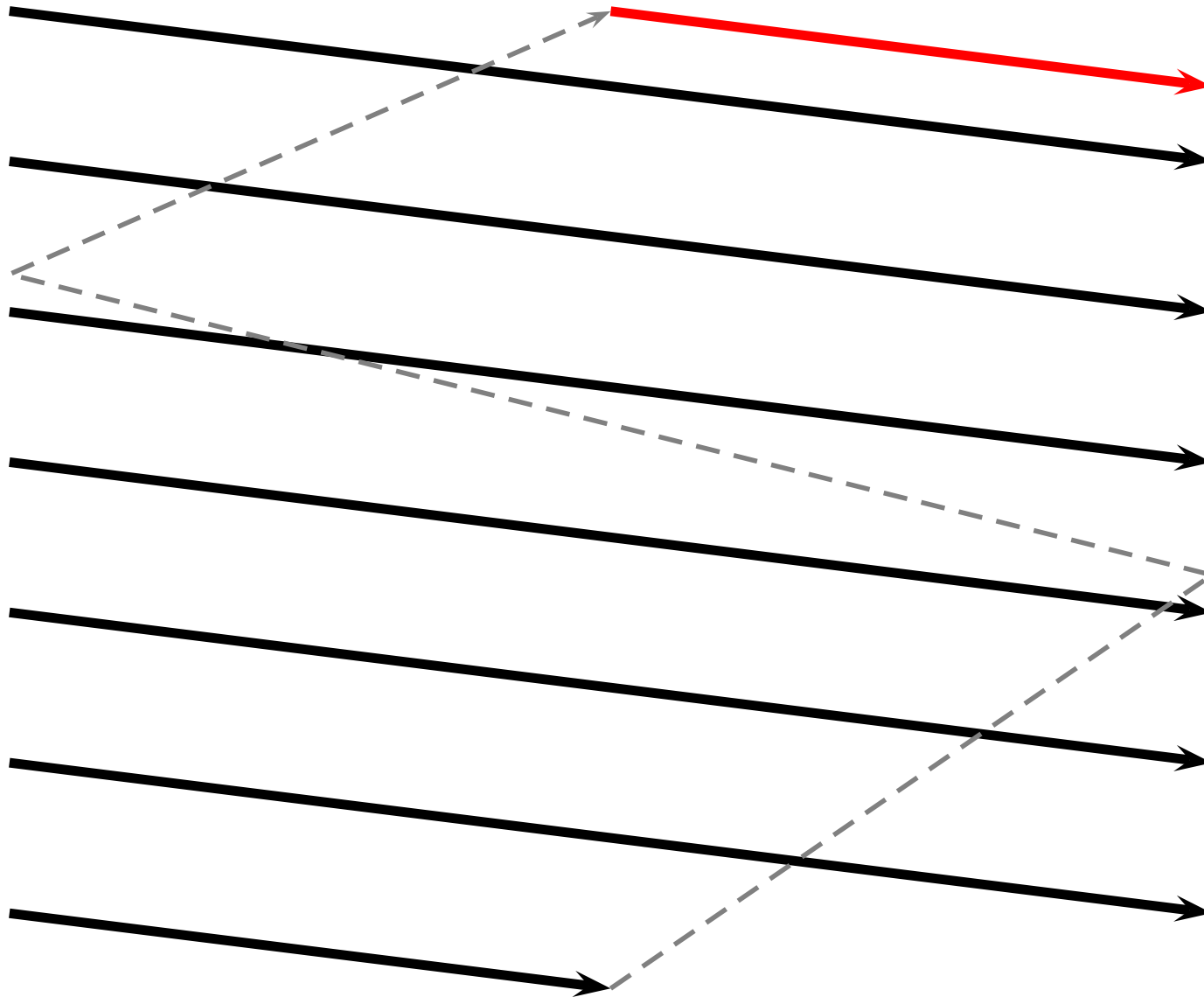
Interlaced Scanning



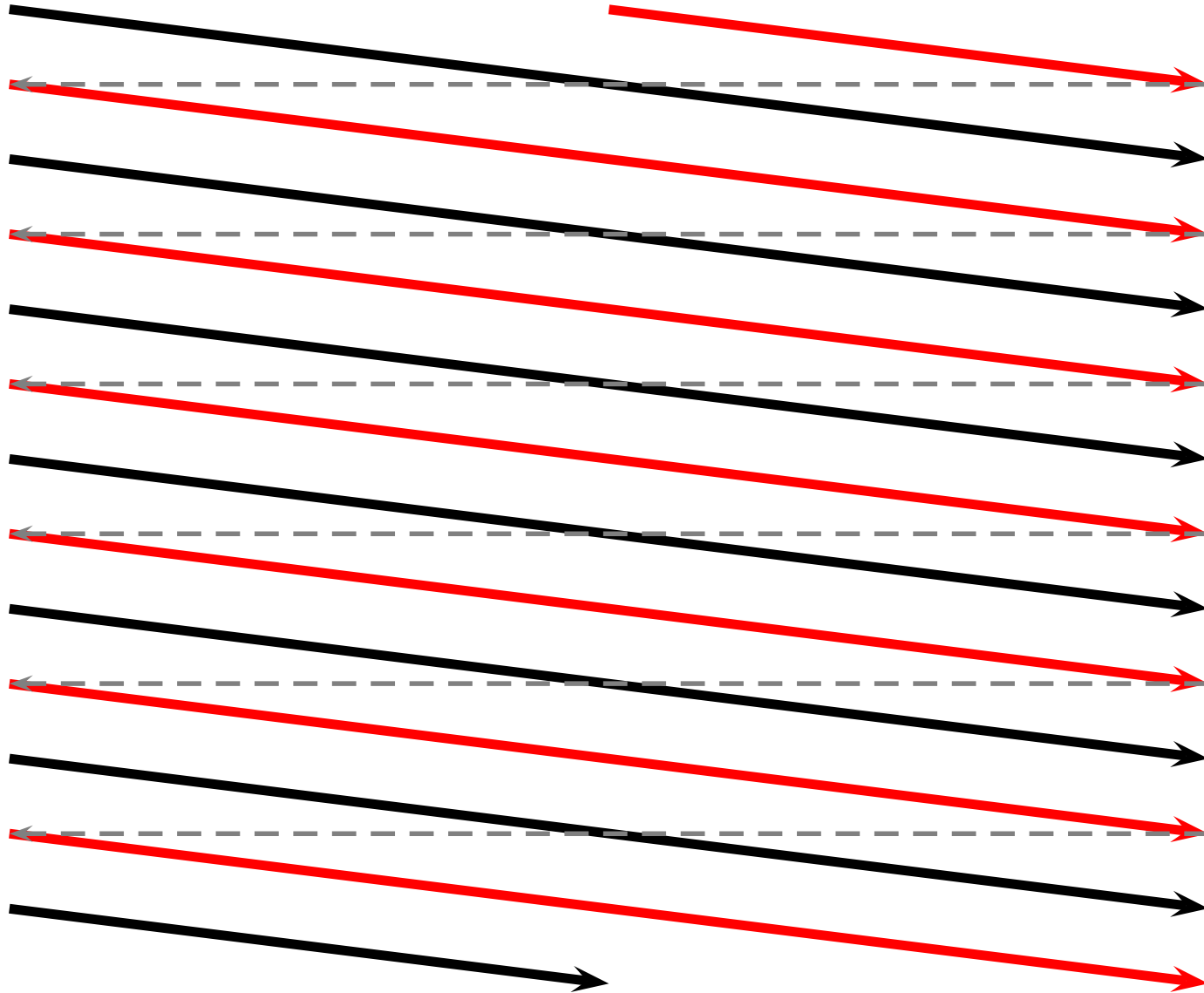
Interlaced Scanning



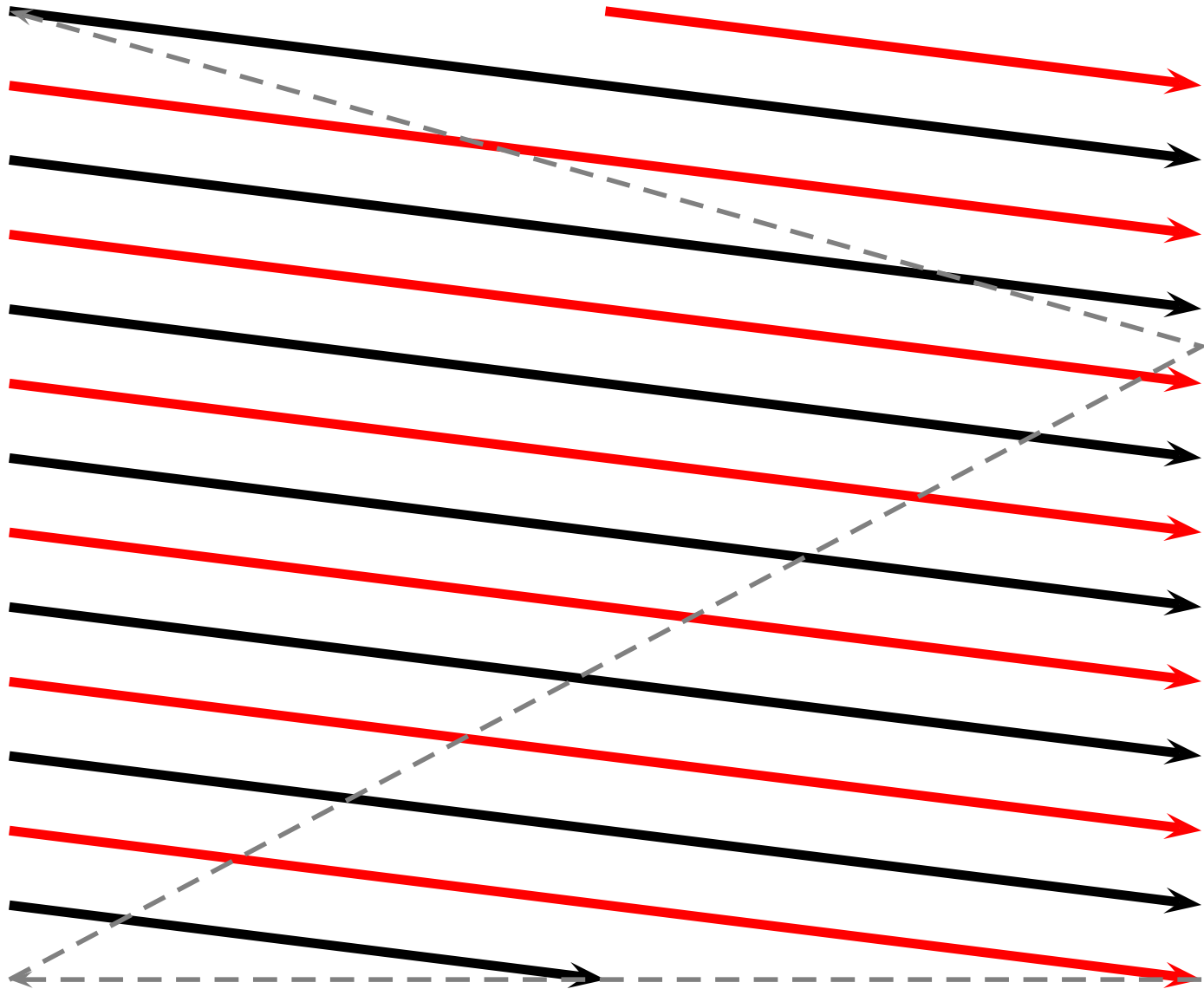
Interlaced Scanning



Interlaced Scanning



Interlaced Scanning



Color Television

Color added later: had to be backwards compatible.

Solution: continue to transmit a “black-and-white” signal and modulate two color signals on top of it.

RGB vs. YIQ colorspaces

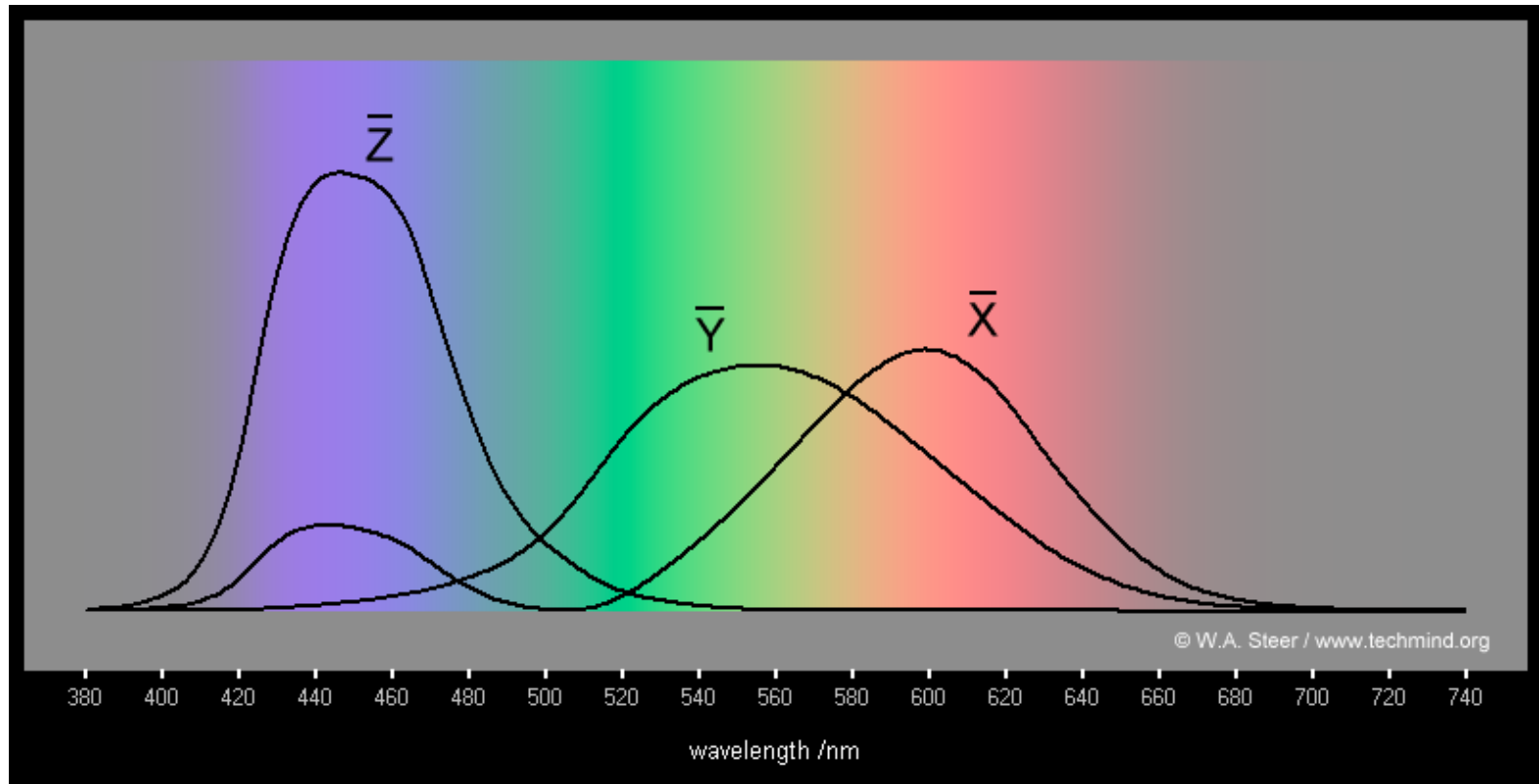
$$\begin{bmatrix} 0.30 & 0.59 & 0.11 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

Y baseband 4 MHz “black-and-white” signal

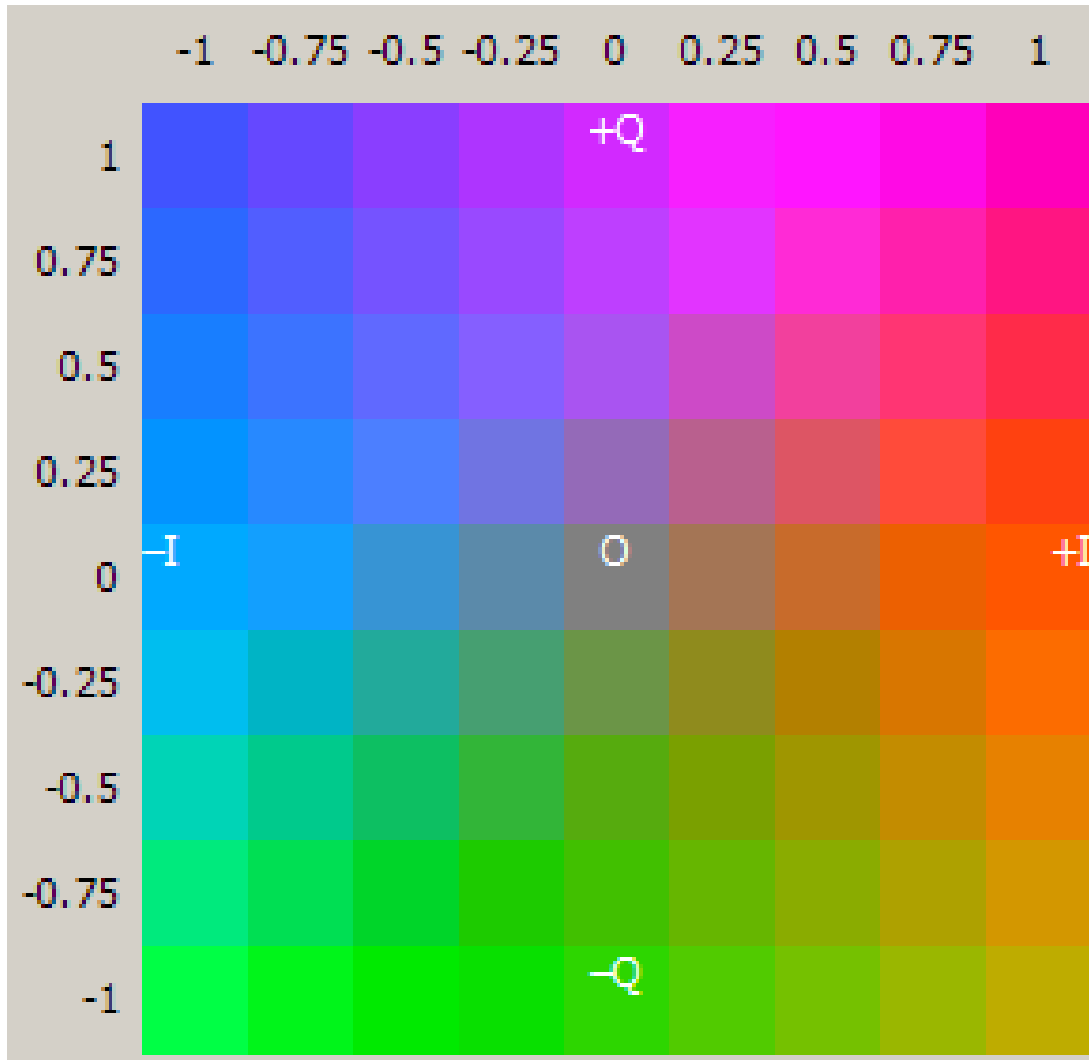
I as 1.5 MHz, Q as 0.5 MHz at 90°:

modulated at 3.58 MHz

CIE Color Matching Curves



YIQ color space with $Y=0.5$



International Standards

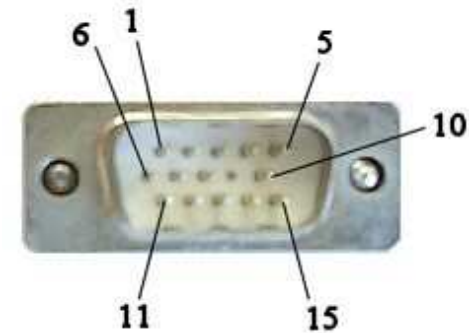
	lines	active lines	vertical res.	aspect ratio	horiz. res.	frame rate
NTSC	525	484	242	4:3	427	29.94 Hz
PAL	625	575	290	4:3	425	25 Hz
SECAM	625	575	290	4:3	465	25 Hz

PAL: Uses YUV instead of YIQ, flips phase of V every other line

SECAM: Transmits the two chrominance signals on alternate lines; no quadrature modulation

Computer Video: VGA

1	2	3	4	5
Red	Green	Blue	ID2	GND
6	7	8	9	10
RGND	GGND	BGND	(+5V)	GND
11	12	13	14	15
ID0	ID1	hsync	vsync	ID3



ID2	ID0	ID1	
-	-	GND	Monochrome, < 1024×768
-	GND	-	Color, < 1024×768
GND	GND	-	Color, ≥ 1024×768

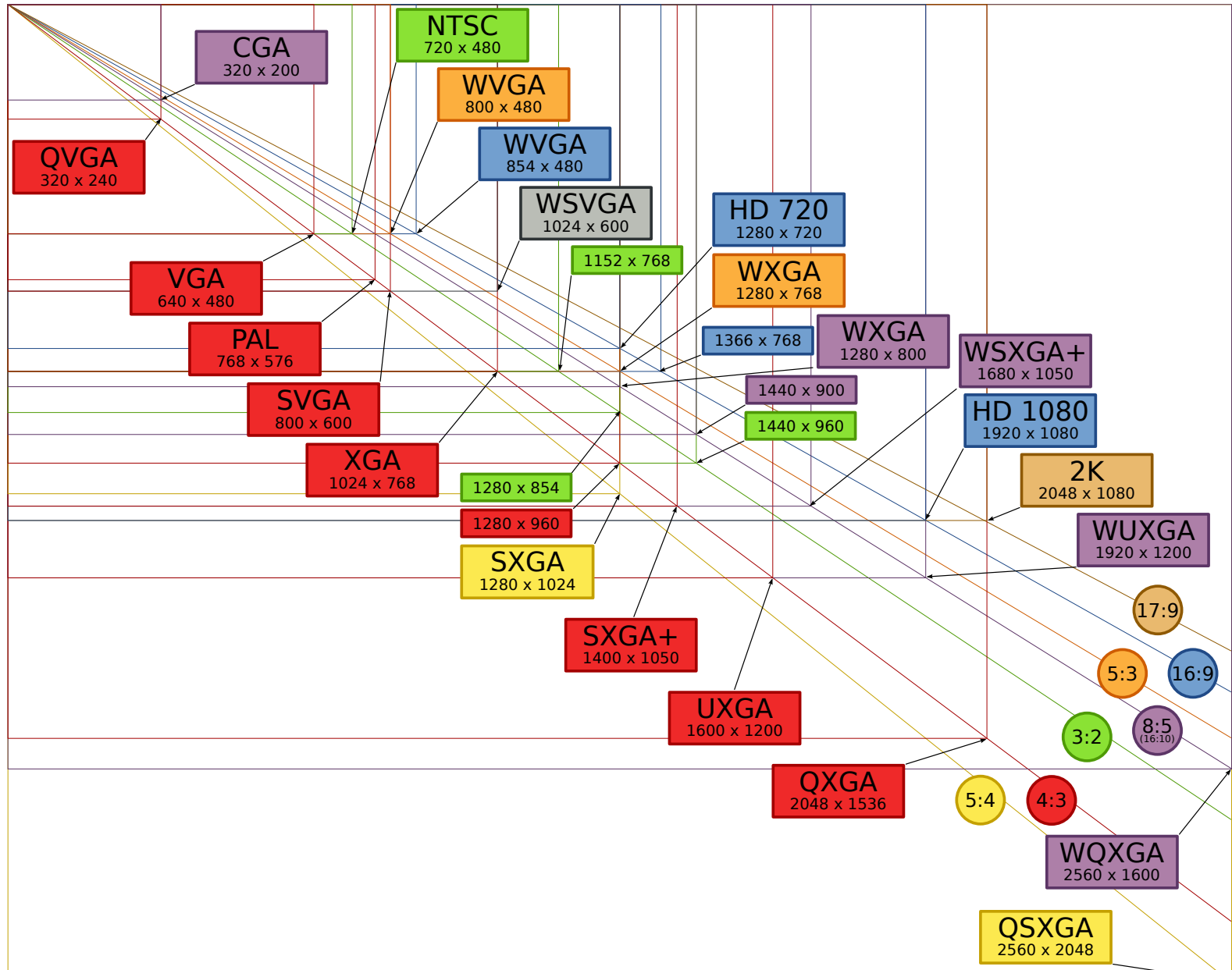
DDC1 ID2 Data from display
vsync also data clock

DDC2 ID1 I²C SDA
ID3 I²C SLC

VGA Timing

Mode	Resolution	Vertical	Horizontal	Pixel Clock
VGA	640×350	70 Hz	31.5 kHz	25.175 MHz
VGA	640×400	70 Hz	31.5 kHz	25.175 MHz
VGA	640×480	59.94 Hz	31.469 kHz	25.175 MHz
SVGA	800×600	56 Hz	35.2 kHz	36 MHz
SVGA	800×600	60 Hz	37.8 kHz	40 MHz
SVGA	800×600	72 Hz	48.0 kHz	50 MHz
XGA	1024×768	60 Hz	48.5 kHz	65 MHz
SXGA	1280×1024	61 Hz	64.2 kHz	110 MHz
HDTV	1920×1080i	60 Hz		
UXGA	1600×1200	60 Hz	75 kHz	162 MHz
UXGA	1600×1200	85 Hz	105.77 kHz	220 MHz
WUXGA	1920×1200	70 Hz	87.5 kHz	230 MHz

Video Standards



Detailed VGA Timing

640 × 480, “60 Hz”

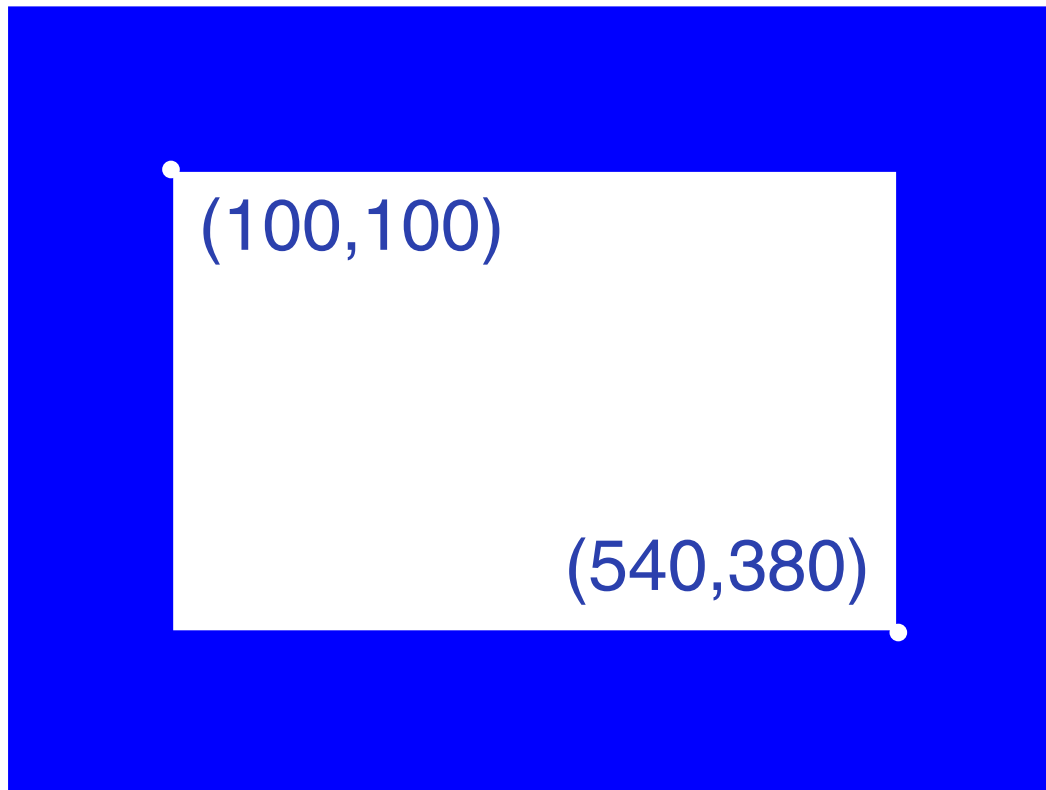
25.175 MHz Dot Clock
31.469 kHz Line Frequency
59.94 Hz Field Frequency

pixels	role	lines	role
8	Front Porch	2	Front Porch
96	Horizontal Sync	2	Vertical Sync
40	Back Porch	25	Back Porch
8	Left border	8	Top Border
640	Active	480	Active
8	Right border	8	Bottom Border
<hr/>		<hr/>	
800	total per line	525	total per field

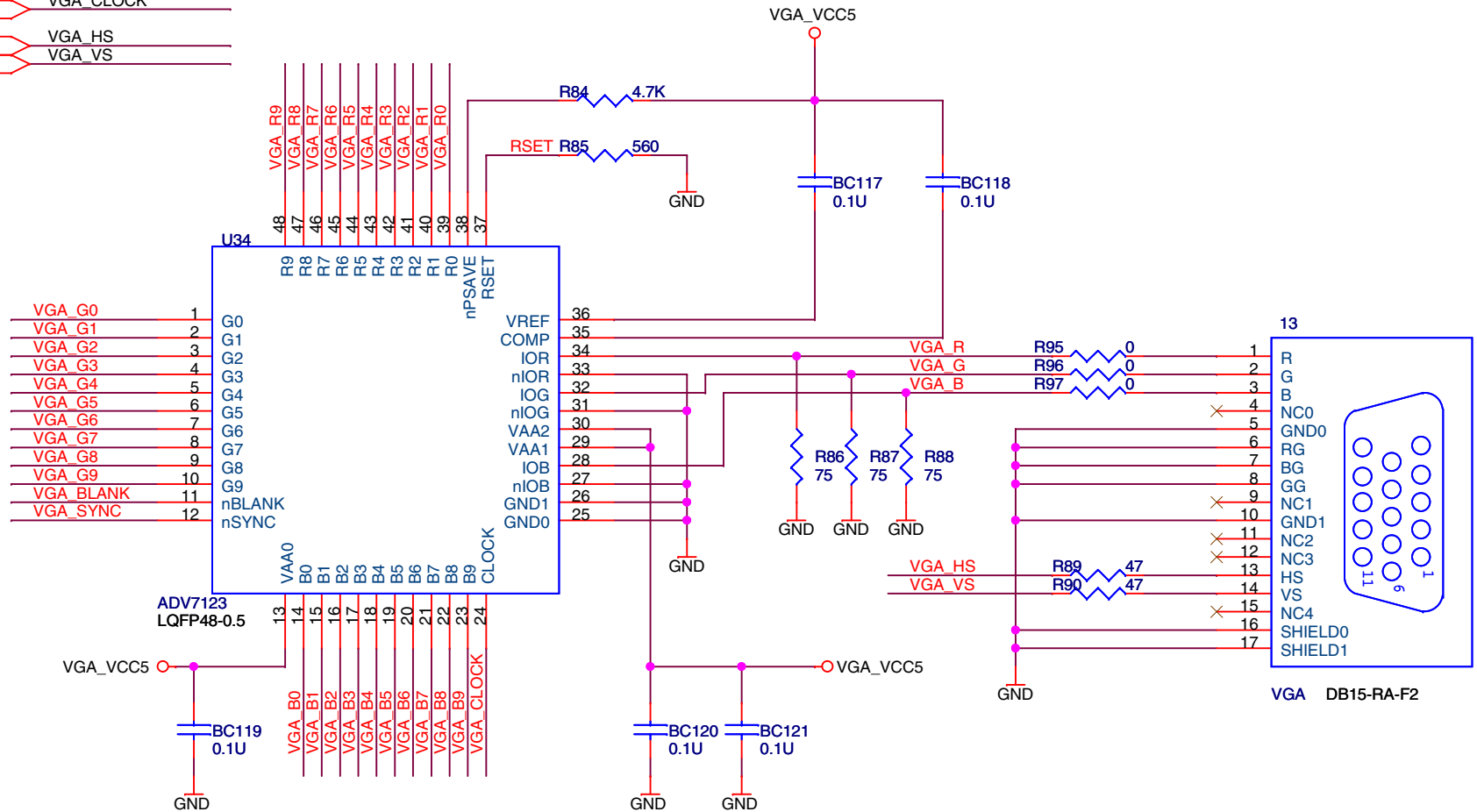
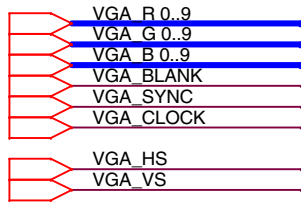
Active-low Horizontal and Vertical sync signals.

Challenge: A white rectangle

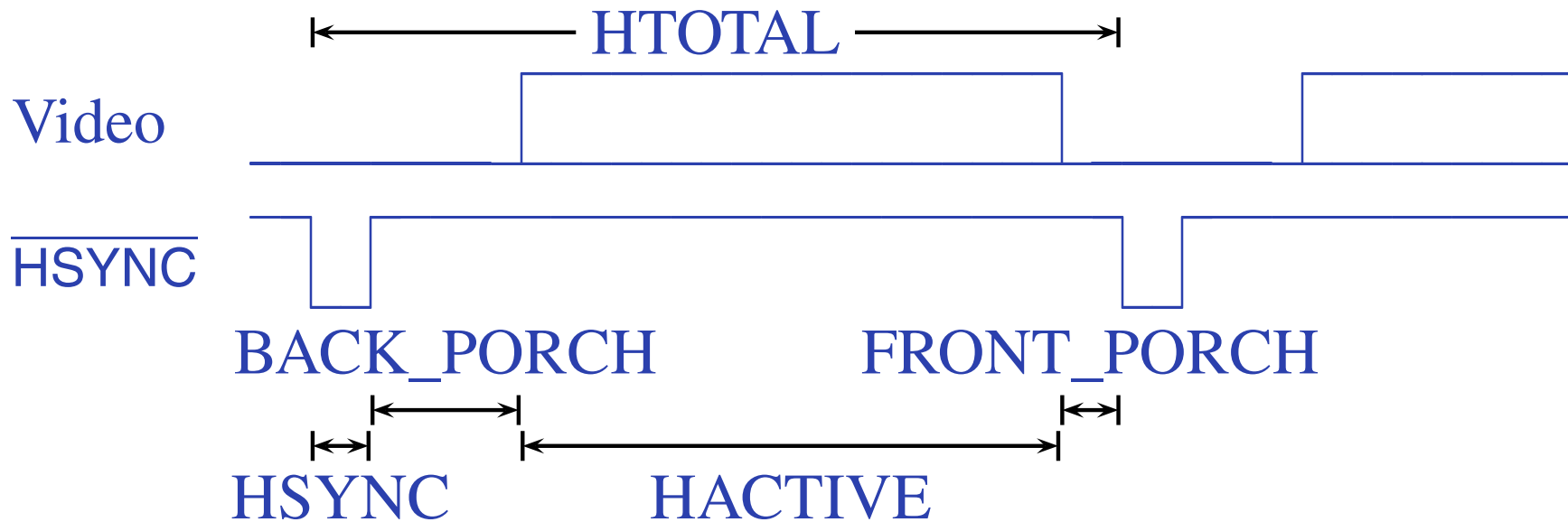
Let's build a VHDL module that displays a 640×480 VGA raster with a white rectangle in the center against a blue background.



Video on the DE2



Horizontal Timing



For a 25.175 MHz pixel clock,

HSYNC	96 pixels
BACK_PORCH	48
HACTIVE	640
FRONT_PORCH	16
<hr/>	
HTOTAL	800

Implementation: Interface

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_vga_raster is

    port (
        reset : in std_logic;
        clk    : in std_logic;           -- Should be 25.125 MHz
        VGA_CLK,                          -- Clock
        VGA_HS,                          -- H_SYNC
        VGA_VS,                          -- V_SYNC
        VGA_BLANK,                       -- BLANK
        VGA_SYNC : out std_logic;        -- SYNC
        VGA_R,                            -- Red[9:0]
        VGA_G,                            -- Green[9:0]
        VGA_B : out unsigned(9 downto 0) -- Blue[9:0]
    );

end de2_vga_raster;
```

Constants

architecture rtl of de2_vga_raster is

-- Video parameters

constant HTOTAL : integer := 800;

constant HSYNC : integer := 96;

constant HBACK_PORCH : integer := 48;

constant HACTIVE : integer := 640;

constant HFRONT_PORCH : integer := 16;

constant VTOTAL : integer := 525;

constant VSYNC : integer := 2;

constant VBACK_PORCH : integer := 33;

constant VACTIVE : integer := 480;

constant VFRONT_PORCH : integer := 10;

constant RECTANGLE_HSTART : integer := 100;

constant RECTANGLE_HEND : integer := 540;

constant RECTANGLE_VSTART : integer := 100;

constant RECTANGLE_VEND : integer := 380;

Signals

```
-- Signals for the video controller

-- Horizontal position (0-800)
signal Hcount : unsigned(9 downto 0);

-- Vertical position (0-524)
signal Vcount : unsigned(9 downto 0);

signal EndOfLine, EndOfField : std_logic;

signal vga_hblank, vga_hsync,
    vga_vblank, vga_vsync : std_logic; -- Sync. signals

-- rectangle area
signal rectangle_h, rectangle_v, rectangle : std_logic;

begin
```

Counters

```
HCounter : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' or EndOfLine = '1' then Hcount <= (others => '0');
    else Hcount <= Hcount + 1;
    end if; end if;
end process HCounter;
```

```
EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';
```

```
VCounter: process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then Vcount <= (others => '0');
    elsif EndOfLine = '1' then
      if EndOfField = '1' then Vcount <= (others => '0');
      else Vcount <= Vcount + 1;
      end if; end if; end if;
end process VCounter;
```

```
EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';
```


Horizontal signals

```
HSyncGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' or EndOfLine = '1' then
      vga_hsync <= '1';
    elsif Hcount = HSYNC - 1 then
      vga_hsync <= '0';
    end if;
  end if;
end process HSyncGen;

HBlankGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      vga_hblank <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH then
      vga_hblank <= '0';
    elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
      vga_hblank <= '1';
    end if;
  end if;
end process HBlankGen;
```

Vertical signals

```
VSyncGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then vga_vsync <= '1';
    elsif EndOfLine = '1' then
      if EndOfField = '1' then vga_vsync <= '1';
      elsif Vcount = VSYNC - 1 then vga_vsync <= '0';
      end if;
    end if;
  end if;
end process VSyncGen;

VBlankGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then vga_vblank <= '1';
    elsif EndOfLine = '1' then
      if Vcount = VSYNC + VBACK_PORCH - 1 then
        vga_vblank <= '0';
      elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
        vga_vblank <= '1';
      end if; end if; end if;
    end process VBlankGen;
```

The Rectangle

```
RectangleHGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' or Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART then
      rectangle_h <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND then
      rectangle_h <= '0';
    end if; end if;
  end process RectangleHGen;
```

```
RectangleVGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then rectangle_v <= '0';
    elsif EndOfLine = '1' then
      if Vcount = VSYNC + VBACK_PORCH - 1 + RECTANGLE_VSTART then
        rectangle_v <= '1';
      elsif Vcount = VSYNC + VBACK_PORCH - 1 + RECTANGLE_VEND then
        rectangle_v <= '0';
      end if; end if; end if;
    end process RectangleVGen;
```

```
rectangle <= rectangle_h and rectangle_v;
```

Output signals

```
VideoOut: process (clk, reset)
begin
  if reset = '1' then
    VGA_R <= "0000000000"; VGA_G <= "0000000000"; VGA_B <= "0000000000";
  elsif clk'event and clk = '1' then
    if rectangle = '1' then
      VGA_R <= "1111111111"; VGA_G <= "1111111111"; VGA_B <= "1111111111";
    elsif vga_hblank = '0' and vga_vblank = '0' then
      VGA_R <= "0000000000"; VGA_G <= "0000000000"; VGA_B <= "1111111111";
    else
      VGA_R <= "0000000000"; VGA_G <= "0000000000"; VGA_B <= "0000000000";
    end if;
  end if;
end process VideoOut;

VGA_CLK <= clk;
VGA_HS <= not vga_hsync;
VGA_VS <= not vga_vsync;
VGA_SYNC <= '0';
VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
```