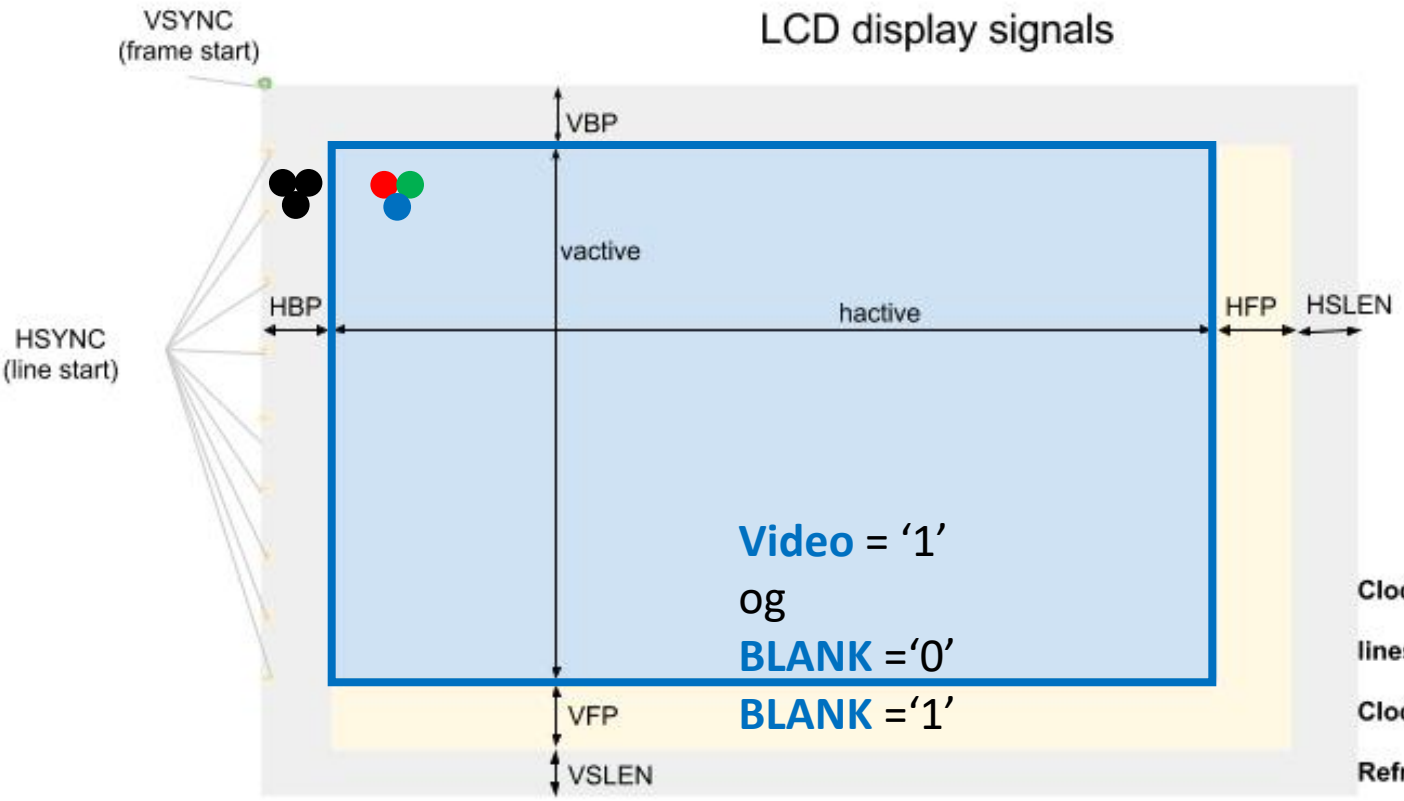
Starthjælp til VGA interface (og digitalt oscilloskop)



Key

vactive: Vertical resolution
hactive: Horizontal resolution

VBP: Vertical Back Porch
VFP: Vertical Front Porch
VSLEN: Vertical sync length

HBP: Horizontal Back Porch
HFP: Horizontal Front Porch
HSLEN: Horizontal sync length

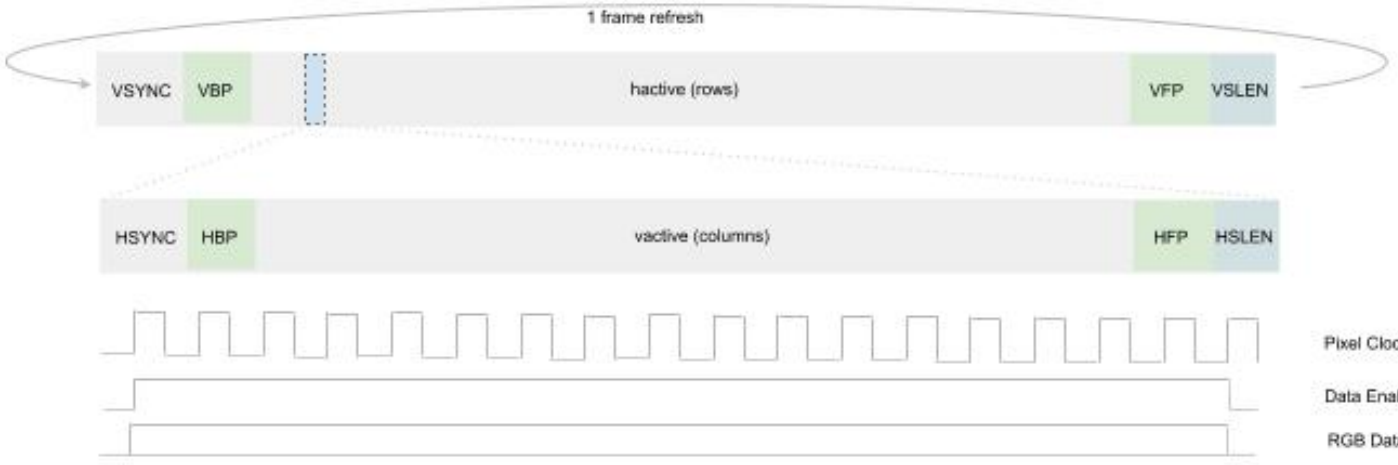
VSYNC: Vertical sync
HSYNC: horizontal sync

Clocks per frame = lines per frame * Clocks per line

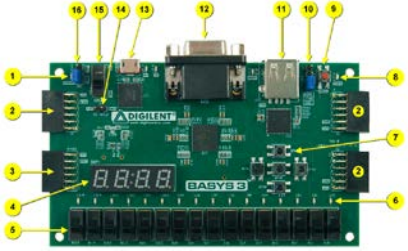
lines / frame = VSYNC + VBP + vactive + VFP + VSLEN

Clocks / line = HSYNC + HBP + hactive + HFP + HSLEN

Refresh rate = Pixel Clock / clocks per frame

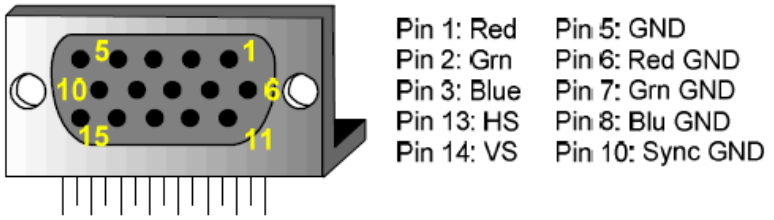


VGA - interface



Uddrag fra Basys3_RM.pdf – læs evt. resten der

the 75-ohm termination resistance of the VGA display to create 16 signal levels each on the red, green, and blue VGA signals. This circuit, shown in Fig 11, produces video color signals that proceed in equal increments between 0V (fully off) and 0.7V (fully on).



Prøv her http://tinyurl.com/y6mf4uuq

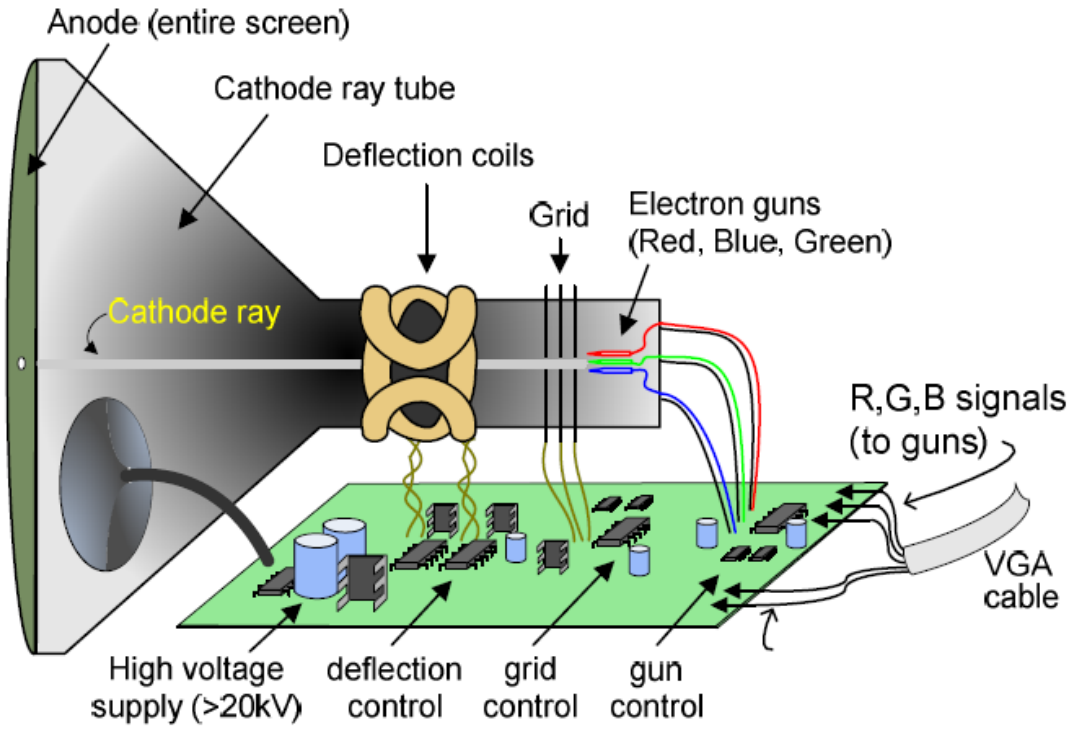
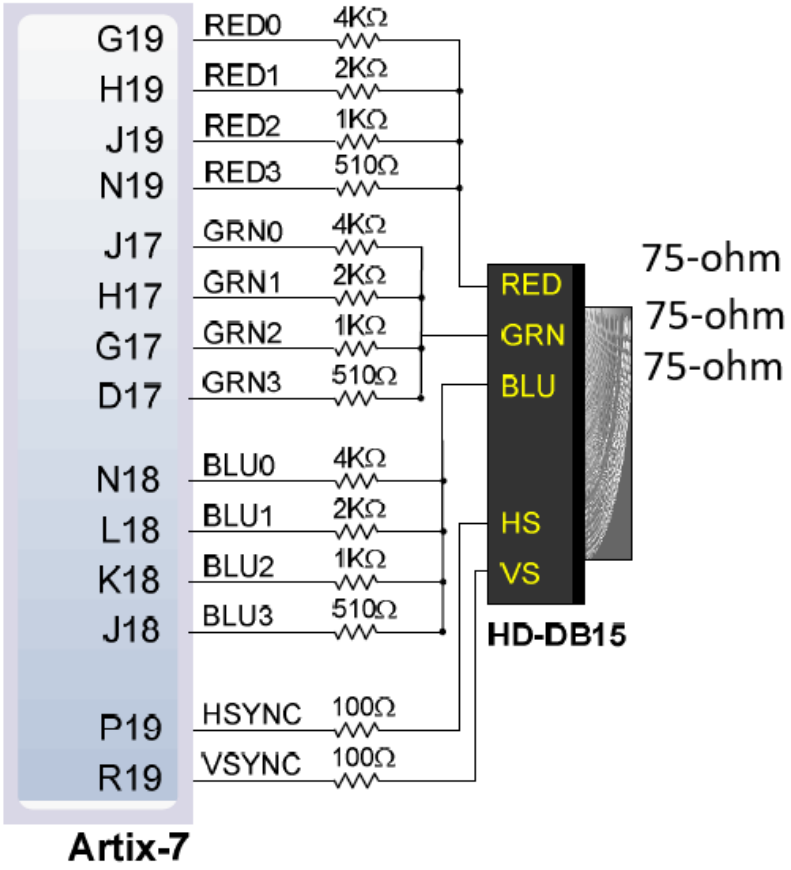


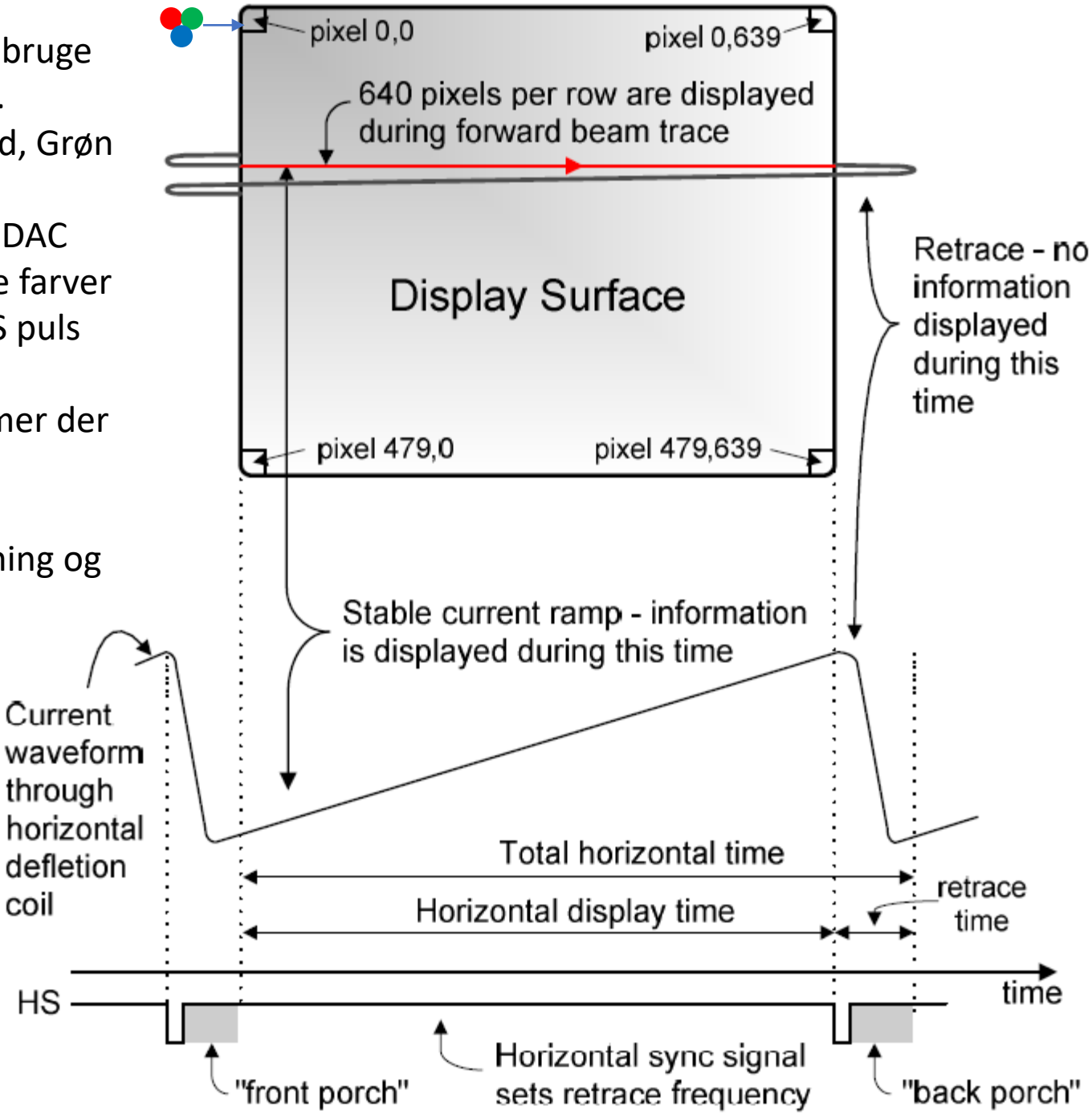
Figure 12. Color CRT display



Den VGA opløsning vi ønsker at bruge benytter 480 linjer af 640 pixels.
En pixel består af de 3 farver Rød, Grøn og Blå.
Hver pixel kan styres af en 4-bit DAC hvilket giver 2^{12}=4096 forskellige farver
For hver linje kommer der en HS puls

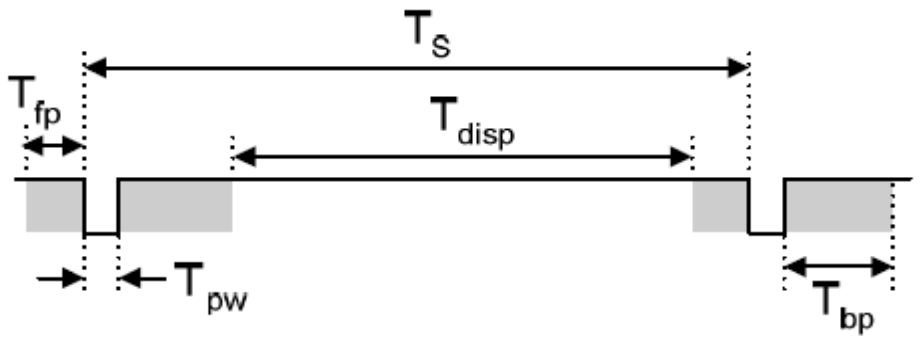
Efter 640 linjer (+Porches) kommer der så en VS puls.

Det hele er et spørgsmål om timing og tællere.



Timing i tabelform
og i VHDL-koden

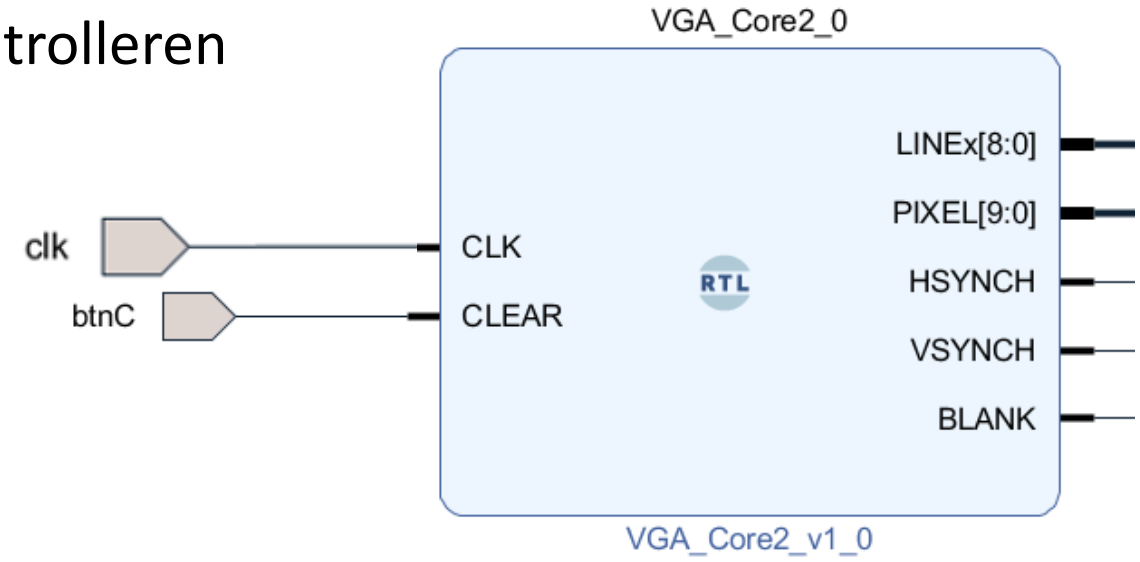
```
constant HR:  integer:=640;--Horizontal Resolution
constant HFP: integer:= 16;--Horizontal Front Porch
constant HBP: integer:= 48;--Horizontal Back Porch
constant HRet:integer:= 96;--Horizontal retrace
constant VR:  integer:=480;--Vertical Resolution
constant VFP: integer:= 10;--Vertical Front Porch
constant VBP: integer:= 33;--Vertical Back Porch
constant VRet:integer:=  2;--Vertical Retrace
```



Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Figure 14. Signal timings for a 640-pixel by 480 row display using a 25MHz pixel clock and 60Hz vertical refresh

VGA_Core2 implementerer display controlleren



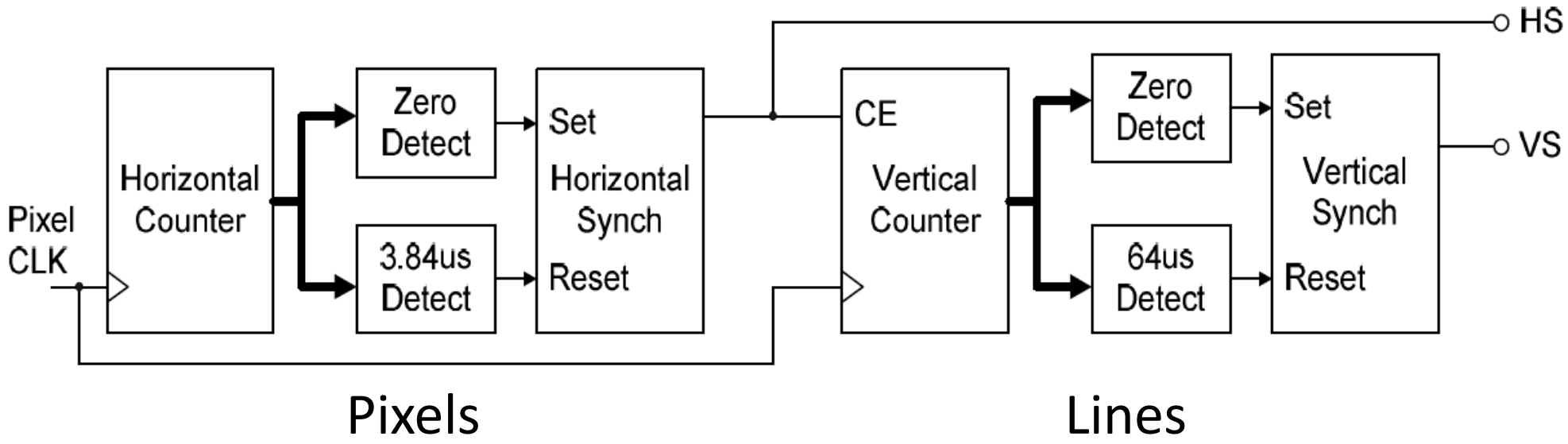
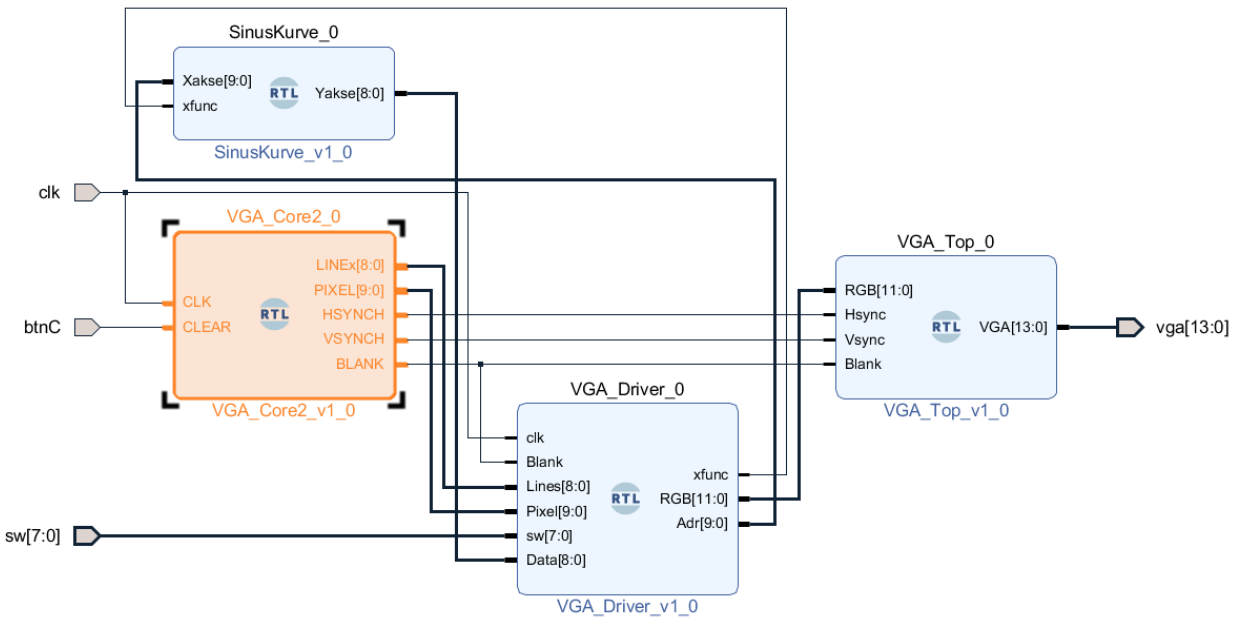
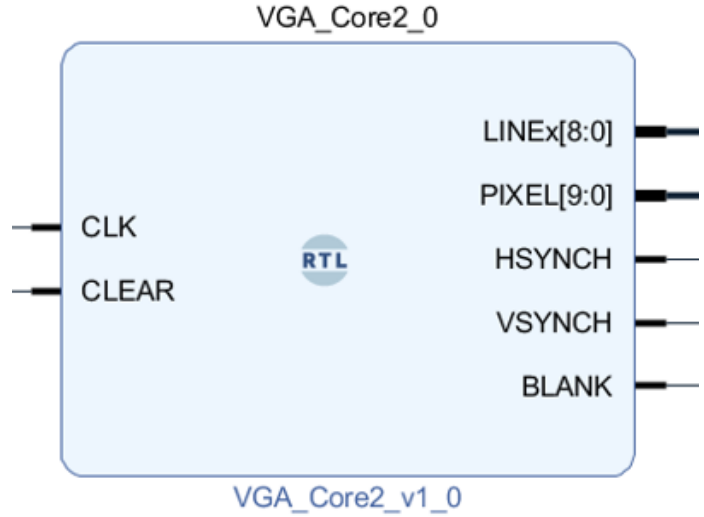


Figure 15. VGA display controller block diagram

VGA_Core2 sørger for timing (HS og VS) samt at holde styr på hvor den aktive Pixel er






```
 7  entity VGA_Core2 is
 8      Port (  CLK :      in STD_LOGIC;  -- 100 MHz clock
 9              CLEAR :    in STD_LOGIC;
10              LINEx :    out STD_LOGIC_VECTOR (8 downto 0); -- y_control
11              PIXEL :    out STD_LOGIC_VECTOR (9 downto 0); -- x_control
12              HSYNCH :   out STD_LOGIC; -- h_s : out STD_LOGIC;
13              VSYNCH :   out STD_LOGIC; -- v_s : out STD_LOGIC;
14              BLANK :    out STD_LOGIC  -- video_on : out STD_LOGIC;
15              );
16  end VGA_Core2;
```

```
entity VGA_Core2 is
    Port (  CLK :      in STD_LOGIC;  -- 100 MHz clock
            CLEAR :    in STD_LOGIC;
            LINEx :    out STD_LOGIC_VECTOR (8 downto 0); -- y_control
            PIXEL :    out STD_LOGIC_VECTOR (9 downto 0); -- x_control
            HSYNCH :   out STD_LOGIC; -- h_s : out STD_LOGIC;
            VSYNCH :   out STD_LOGIC; -- v_s : out STD_LOGIC;
            BLANK :    out STD_LOGIC  -- video_on : out STD_LOGIC;
            );
end VGA_Core2;

architecture Behavioral of VGA_Core2 is
    -- Video Parameters
    constant HR:  integer:=640;--Horizontal Resolution
    constant HFP: integer:= 16;--Horizontal Front Porch
    constant HBP: integer:= 48;--Horizontal Back Porch
    constant HRet:integer:= 96;--Horizontal retrace
    constant VR:  integer:=480;--Vertical Resolution
    constant VFP: integer:= 10;--Vertical Front Porch
    constant VBP: integer:= 33;--Vertical Back Porch
    constant VRet:integer:=  2;--Vertical Retrace
    --sync counter
    signal counter_h, counter_h_next: integer range 0 to 800;
    signal counter_v, counter_v_next: integer range 0 to 525;
    --mod 4 counter
    signal counter_mod4,counter_mod4_next: std_logic_vector(1 downto 0):="00";
    --State signals
    signal h_end, v_end:std_logic:='0';
    --Output Signals(buffer)
    signal hs_buffer,hs_buffer_next:std_logic:='0';
    signal vs_buffer,vs_buffer_next:std_logic:='0';
    --pixel counter
    signal x_counter, x_counter_next:integer range 0 to 900;
    signal y_counter, y_counter_next:integer range 0 to 900;
    --video_on_off
    signal video:std_logic;
begin
```

Konstanter til timingen

Bemærk at alle variable, som skal tildeles en F/F er erklæret to gange som **counter_h**. **counter_h** er den aktuelle værdi **counter_h_next** er næste værdi

```
43  begin
44      ----------------------------------------clk register
45      -- Dette er den eneste clk-styrede process (alle F/F laves her)
46      -- De andre processer laver "ren" kombinatorisk logik
47      -- Man undgår en masse problemer ved at skille F/F og Logik
48      process(clk,CLEAR)
49      begin
50          if CLEAR ='1' then
51            counter_h    <=0;
52            counter_v    <=0;
53            hs_buffer    <='0';
54            hs_buffer    <='0';
55            counter_mod4 <="00";
56          elsif Rising_edge(clk) then
57                counter_h  <= counter_h_next;
58                counter_v  <= counter_v_next;
59                x_counter  <= x_counter_next;
60                y_counter  <= y_counter_next;
61                hs_buffer  <= hs_buffer_next;
62                vs_buffer  <= vs_buffer_next;
63                counter_mod4 <= counter_mod4_next;
64         end if;
65      end process;
66      ----------------------------------------video on/off
67      -- Når video='1' er "elektronstrålen" inden for skærmen
68      video <= '1' when      (counter_v >= VBP) and (counter_v < VBP + VR)
69                        and (counter_h >= HBP) and (counter_h < HBP + HR) else
70               '0';
71
72      -------mod 4 counter deler 100 Mhz til 25 MHz
73      counter_mod4_next<= counter_mod4+1;
```

Hver gang der kommer en clk-puls **counter_h_next** overført til **counter_h** (den nye aktuelle værdi)

Det samme gælder for resten af **_next** variablene

På den måde undgår man får ”rodet” logiske kredse og F/F sammen på en uønsket måde.

```
72      -------mod 4 counter deler 100 Mhz til 25 MHz
73      counter_mod4_next<= counter_mod4+1;
74      ---------------------------------------- Horizontal Counter
75      process(counter_h,counter_mod4,h_end)
76      begin
77          counter_h_next<=counter_h;          -- Default:  Next<=Current
78          if counter_mod4= "11" then
79              if h_end='1' then
80                  counter_h_next<=0;
81              else
82                  counter_h_next<=counter_h+1;
83              end if;
84          end if;
85      end process;
86      ----------------------------------------end of Horizontal scanning
87      h_end<= '1' when counter_h=799 else
88              '0';
89      ----------------------------------------Vertical Counter
90      process(counter_v,counter_mod4,h_end,v_end)
91      begin
92          counter_v_next <= counter_v;         -- Default:  Next<=Current
93          if counter_mod4= "11" and h_end='1' then
94              if v_end='1' then
95                  counter_v_next<=0;
96              else
97                  counter_v_next<=counter_v+1;
98              end if;
99         end if;
100     end process;
101     ----------------------------------------end of Vertical scanning
102     v_end<= '1' when counter_v=524 else
103             '0';
```

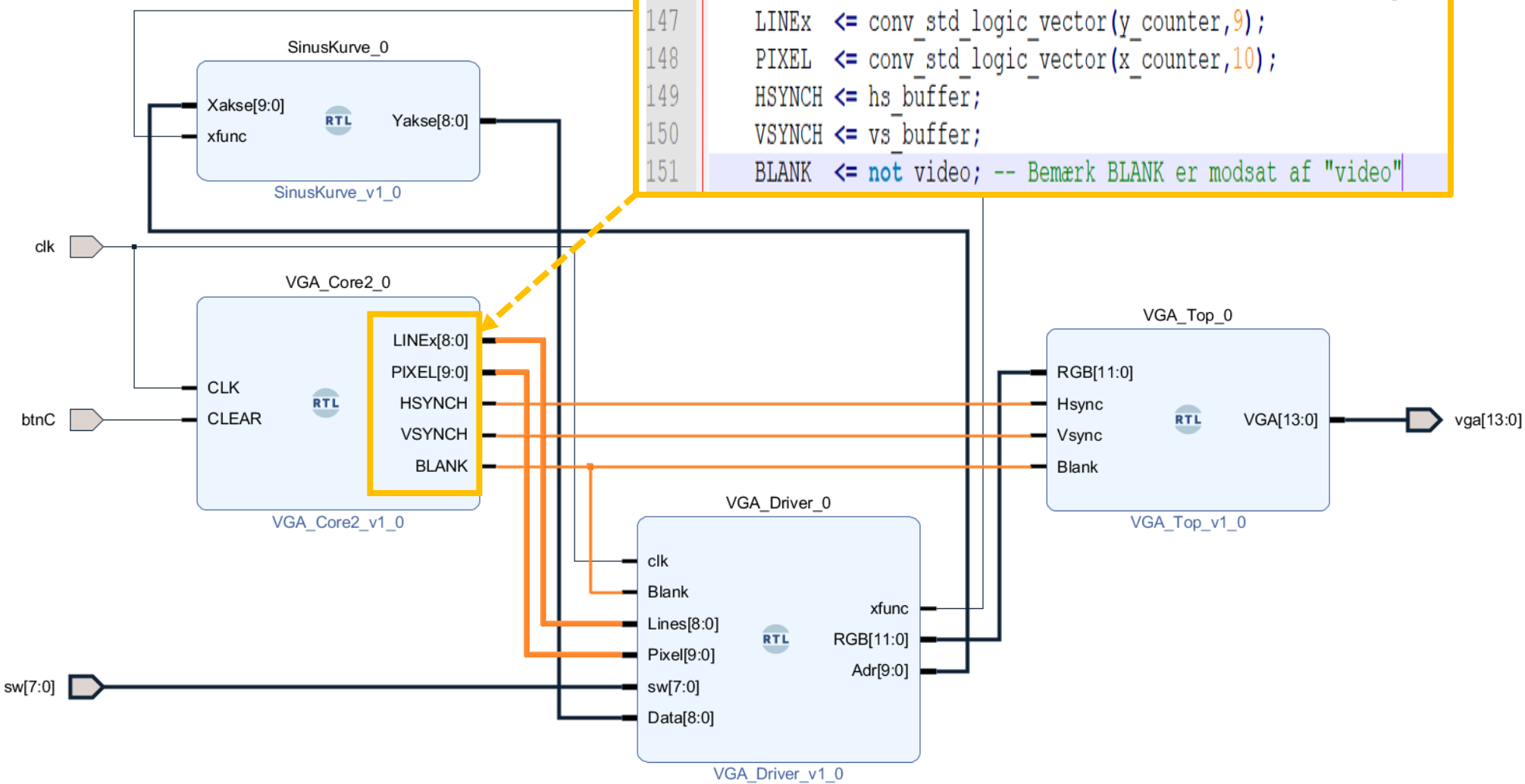
De følgende processer laver kun ren kombinatorisk logik:
counter_h_next er som udgangspunkt det samme som **counter_h**
Hvis betingelserne er tilstede vil **_next** enten blive talt op eller nulstillet

```
104     ------------------------------------------------pixel x counter
105     process(x_counter,counter_mod4,h_end,video)
106     begin
107         x_counter_next<=x_counter;      -- Default:  Next<=Current
108         if video = '1' then
109             if counter_mod4= "11" then
110                 if x_counter= 639 then
111                     x_counter_next<=0;
112                 else
113                     x_counter_next<=x_counter + 1;
114                 end if;
115             end if;
116         else
117             x_counter_next<=0;
118         end if;
119     end process;
120     ------------------------------------------------pixel y counter
121     process(y_counter,counter_mod4,h_end,counter_v)
122     begin
123         y_counter_next<=y_counter;      -- Default:  Next<=Current
124         if counter_mod4= "11" and h_end='1' then
125             if counter_v >32 and counter_v <512 then
126                y_counter_next<=y_counter + 1;
127             else
128                y_counter_next<=0;
129             end if;
130         end if;
131     end process;
```

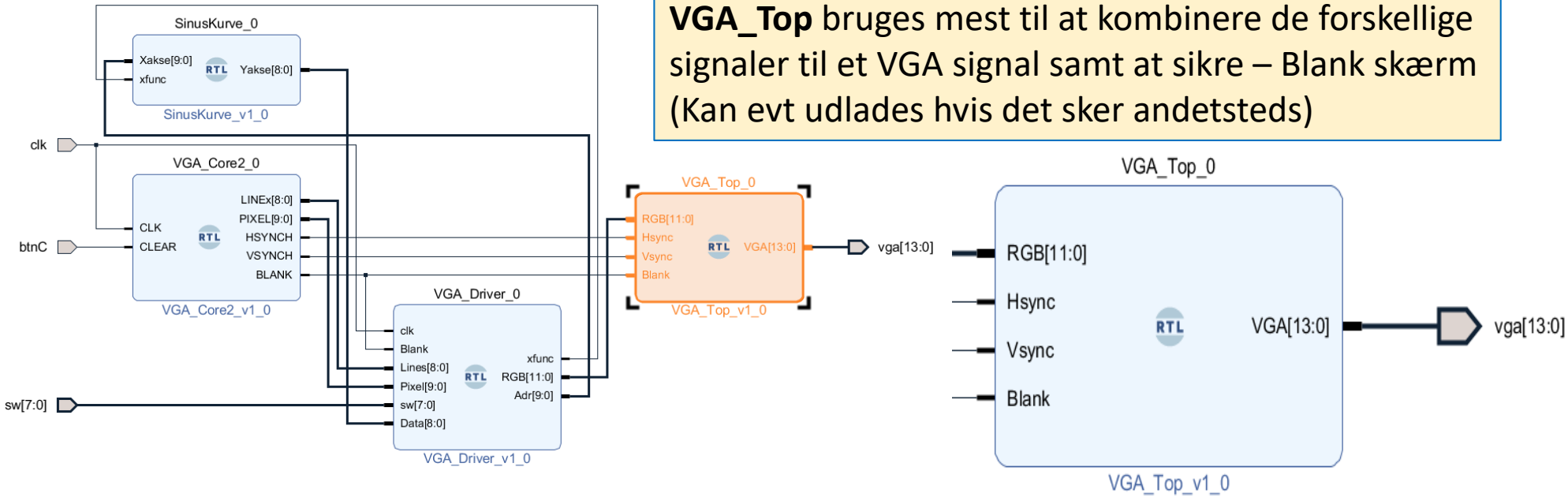
```
--    constant HR:  integer:=640;--Horizontal Resolution
--    constant HFP: integer:= 16;--Horizontal Front Porch
--    constant HBP: integer:= 48;--Horizontal Back Porch
--    constant HRet:integer:= 96;--Horizontal retrace
--    constant VR:  integer:=480;--Vertical Resolution
--    constant VFP: integer:= 10;--Vertical Front Porch
--    constant VBP: integer:= 33;--Vertical Back Porch
--    constant VRet:integer:=  2;--Vertical Retrace
   ------------------------------------------------buffer
    hs_buffer_next<= '1' when counter_h < HR+HFP+HBP else
                     '0';
    vs_buffer_next<= '1' when counter_v < VR+VFP+VBP else
                     '0';
    ------------------------------------------------outputs
    LINEx  <= conv_std_logic_vector(y_counter,9);
    PIXEL  <= conv_std_logic_vector(x_counter,10);
    HSYNCH <= hs_buffer;
    VSYNCH <= vs_buffer;
    BLANK  <= not video; -- Bemærk BLANK er modsat af "video"
end Behavioral;
```

Resultatet er HS, VS, BLANK samt LINEx og PIXEL signaler

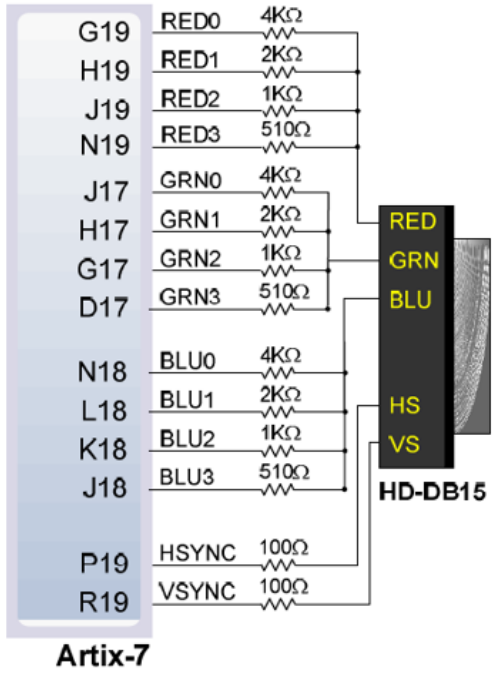
```
141   ----------------------------------------------------buffer
142   hs_buffer_next<= '1' when counter_h < HR+HFP+HBP else
143                    '0';
144   vs_buffer_next<= '1' when counter_v < VR+VFP+VBP else
145                    '0';
146   ----------------------------------------------------outputs
147   LINEx  <= conv_std_logic_vector(y_counter,9);
148   PIXEL  <= conv_std_logic_vector(x_counter,10);
149   HSYNCH <= hs_buffer;
150   VSYNCH <= vs_buffer;
151   BLANK  <= not video; -- Bemærk BLANK er modsat af "video"
```



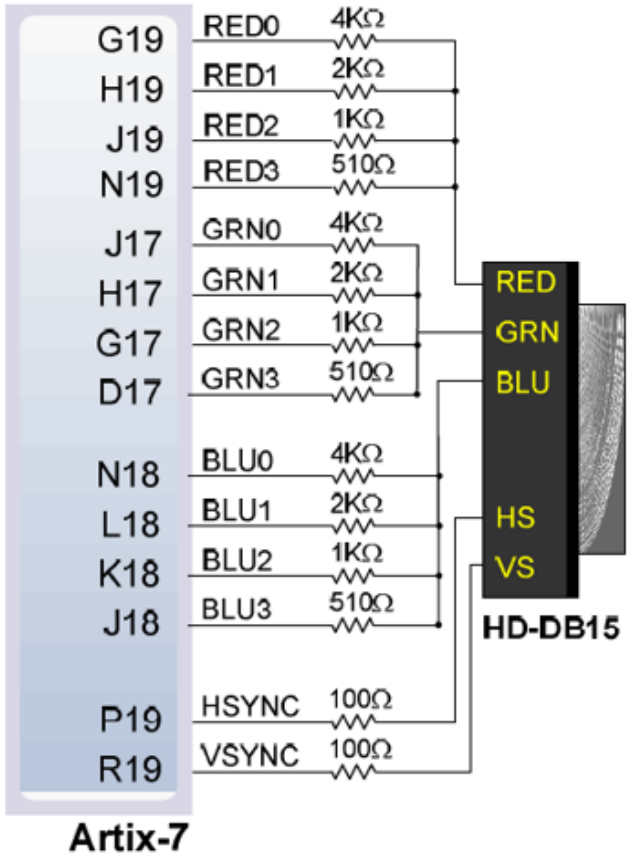
VGA_Top bruges mest til at kombinere de forskellige signaler til et VGA signal samt at sikre – Blank skærm (Kan evt udlades hvis det sker andetsteds)




```
6  entity VGA_Top is
7      Port ( RGB : in STD_LOGIC_VECTOR (11 downto 0);
8             Hsync : in STD_LOGIC;
9             Vsync : in STD_LOGIC;
10            Blank : in STD_LOGIC;
11            VGA : out STD_LOGIC_VECTOR (13 downto 0));
12 end VGA_Top;
13
14 architecture Behavioral of VGA_Top is
15
16 begin
17   -- Når Blank='1' skal Green Blue Red GBR slukkes helt med 000 hex
18     VGA <= Vsync & Hsync &   RGB    when Blank='0' else
19            Vsync & Hsync & X"000" ;
20 end Behavioral;
```

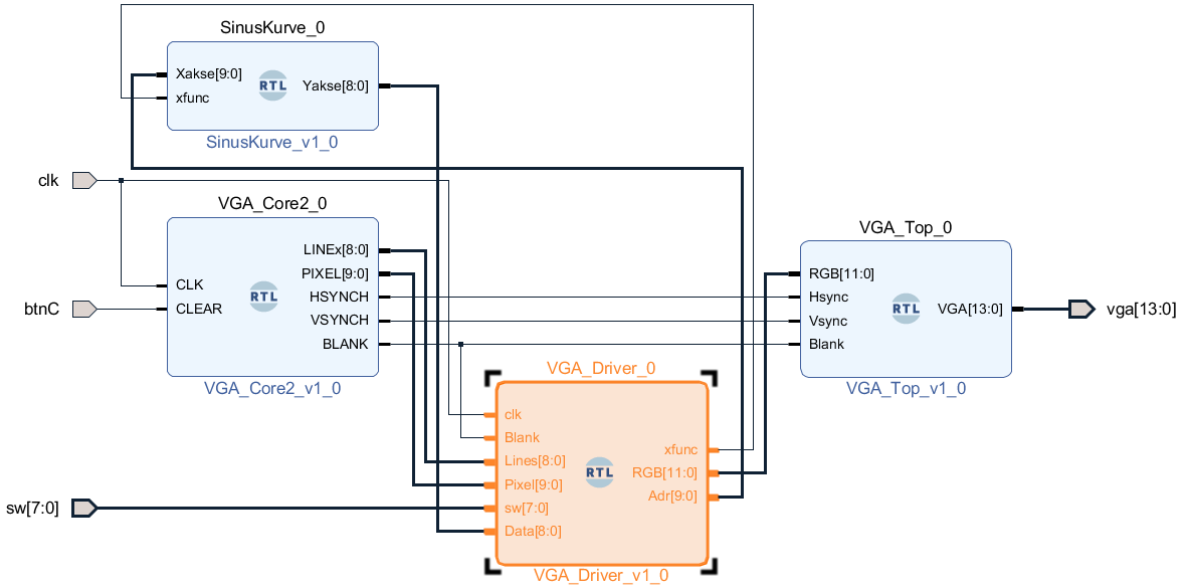


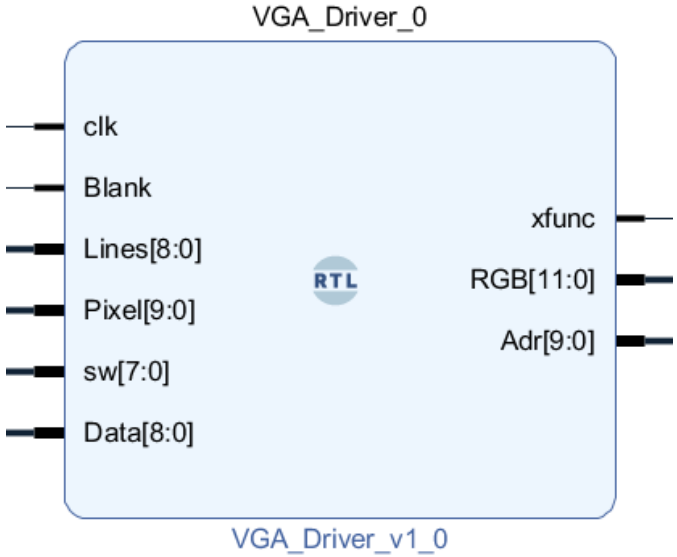
Uddrag af **Basys3_Master_VGA.XDC**




```
#VGA Connector
set_property -dict { PACKAGE_PIN N18    IOSTANDARD LVCMOS33 } [get_ports {vga[0]}]
#Blue[0]
set_property -dict { PACKAGE_PIN L18    IOSTANDARD LVCMOS33 } [get_ports {vga[1]}]
#Blue[1]
set_property -dict { PACKAGE_PIN K18    IOSTANDARD LVCMOS33 } [get_ports {vga[2]}]
#Blue[2]
set_property -dict { PACKAGE_PIN J18    IOSTANDARD LVCMOS33 } [get_ports {vga[3]}]
#Blue[3]
set_property -dict { PACKAGE_PIN J17    IOSTANDARD LVCMOS33 } [get_ports {vga[4]}]
#Green[0]
set_property -dict { PACKAGE_PIN H17    IOSTANDARD LVCMOS33 } [get_ports {vga[5]}]
#Green[1]
set_property -dict { PACKAGE_PIN G17    IOSTANDARD LVCMOS33 } [get_ports {vga[6]}]
#Green[2]
set_property -dict { PACKAGE_PIN D17    IOSTANDARD LVCMOS33 } [get_ports {vga[7]}]
#Green[3]
set_property -dict { PACKAGE_PIN G19    IOSTANDARD LVCMOS33 } [get_ports {vga[8]}]
#Red[0]
set_property -dict { PACKAGE_PIN H19    IOSTANDARD LVCMOS33 } [get_ports {vga[9]}]
#Red[1]
set_property -dict { PACKAGE_PIN J19    IOSTANDARD LVCMOS33 } [get_ports {vga[10]}]
#Red[2]
set_property -dict { PACKAGE_PIN N19    IOSTANDARD LVCMOS33 } [get_ports {vga[11]}]
#Red[3]

set_property -dict { PACKAGE_PIN P19    IOSTANDARD LVCMOS33 } [get_ports {vga[12]}]
#Hsync
set_property -dict { PACKAGE_PIN R19    IOSTANDARD LVCMOS33 } [get_ports {vga[13]}]
#Vsync
```

VGA_Driver – her dannes skærmbilledet



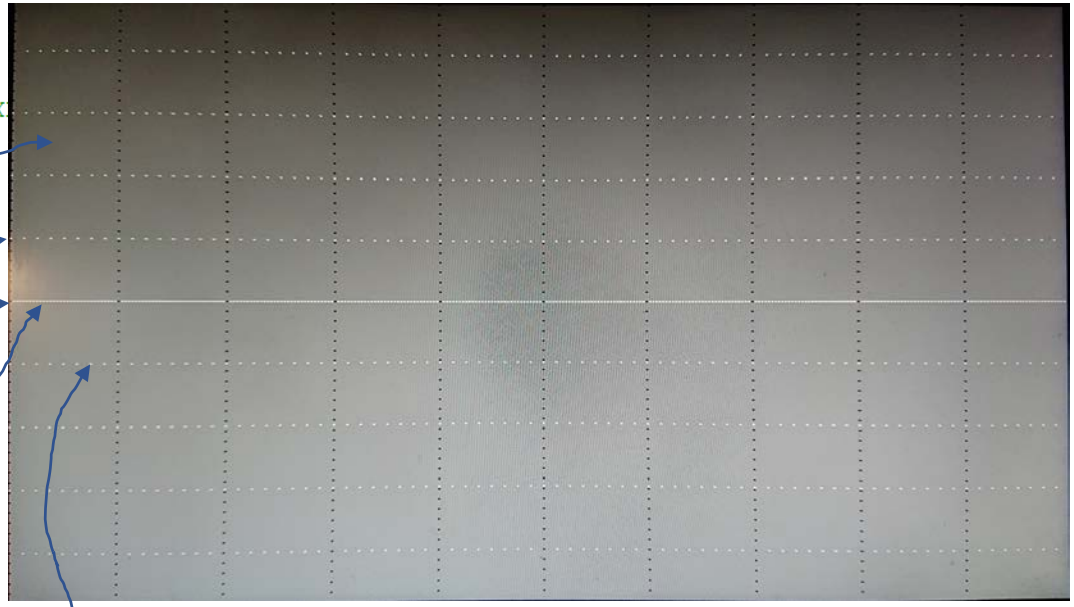



```
14
15  entity VGA_Driver is
16      Port ( clk :    in  STD_LOGIC;
17             Blank :  in  STD_LOGIC;
18             Lines :  in  STD_LOGIC_VECTOR ( 8 downto 0);
19             Pixel :  in  STD_LOGIC_VECTOR ( 9 downto 0);
20             xfunc :  out STD_LOGIC;
21             RGB:     out STD_LOGIC_VECTOR (11 downto 0);
22             sw :     in  STD_LOGIC_VECTOR ( 7 downto 0);
23             Adr :    out STD_LOGIC_VECTOR ( 9 downto 0);
24             Data :   in  STD_LOGIC_VECTOR ( 8 downto 0));
25  end VGA_Driver;
26
```

VGA_Driver – her dannes skærmbilledet

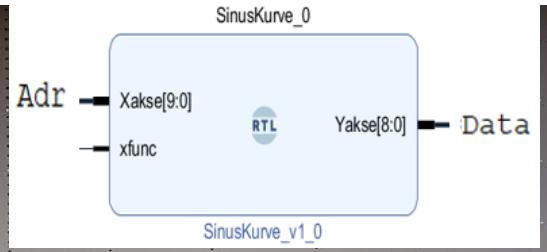
Grå baggrund med streger vandret og lodret
Bemærk hvordan man får en ”stiplet” linje

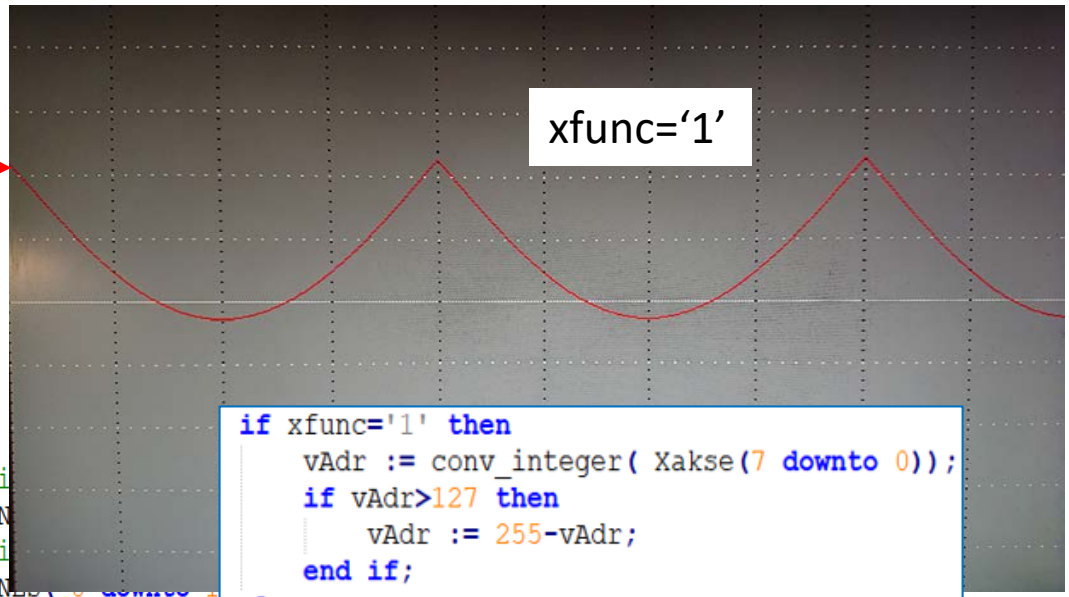
```
architecture Behavioral of VGA_Driver is
begin
    Adr <= Pixel;
    xfunc <= sw(3);

    process( Clk, Lines, Pixel, Blank)
        variable IntLINE: integer;
    begin
        if falling_edge( clk) then
            RGB <= x"000";  -- Sort skærm hvis BLANKx='1'
            IntLINE := conv_integer( Lines);
            if Blank='0' then
                -------------------------- Lav et grid på skærmen  (oversk
                RGB <= x"888";  -- Gråagtig baggrund
                if Sw(4)='1' then
                    if IntLINE=240 and PIXEL(0)='0' then
                        RGB <= x"FFF";  -- Hvid pixel - Midterlinje
                    end if;
                end if;
                ------------------------------------------------
                if Sw(5)='1' then
                    case IntLINE is
                        when 190|140|90|40 =>
                            if  PIXEL(2 downto 0)= "000" then
                                RGB <= x"FFF";  -- Hvid pixel
                            end if;
                        when 240 =>
                            if PIXEL(0 downto 0)= "0" then
                                RGB <= x"FFF";  -- Hvid pixel - Midterlinje
                            end if;
                        when 290|340|390|440 =>
                            if  PIXEL(2 downto 0)= "000" then
                                RGB <= x"FFF";  -- Hvid pixel
                            end if;
                        when others => NULL;
                    end case;
                    if PIXEL( 5 downto 0) = "000000" and LINES(2 downto 0) = "000" then
                        RGB <= x"000";  -- Sort pixel
                    end if;
                end if;
```



```
67          ------------------------------------------
68          if Sw(2)='1' then
69              if (LINES)=Data then
70                  RGB <= x"F00";
71              end if;
72          end if;
73  --      if Sw(3)='1' then
74  --          if (LINES)=AD2(11 downto 3) then
75  --              RGB <= x"0F0";
76  --          end if;
77  --      end if;
78          ------------------------------------------
79          case Sw(7 downto 6) is
80              when "01" => ----------------------Vis
81
82                  ------------Blue-----
83                  RGB(0)  <= LINES(7);
84                  RGB(1)  <= PIXEL(2);
85                  RGB(2)  <= PIXEL(5);
86                  RGB(3)  <= PIXEL(8);
87                   ------------Green-----
88                  RGB(4)  <= PIXEL(0);
89                  RGB(5)  <= PIXEL(3);
90                  RGB(6) <= PIXEL(6);
91                  RGB(7) <= PIXEL(9);
92                  ------------Red-----
93                  RGB(8)  <= LINES(6);
94                  RGB(9)  <= PIXEL(1);
95                  RGB(10)  <= PIXEL(4);
96                  RGB(11)  <= PIXEL(7);
97              when "10" => ----------------------Vi
98                  RGB <=    PIXEL( 9 downto 2) & LIN
99              when "11" => ----------------------Vi
100                 RGB <=    PIXEL( 8 downto 5) & LIN
101                 if LINES=PIXEL then
102                     RGB <= x"FFF";
103                 end if;
104 --              if PIXEL=AD1(11 downto 3) then
105 --                  RGB <= x"444";
106 --              end if;
```

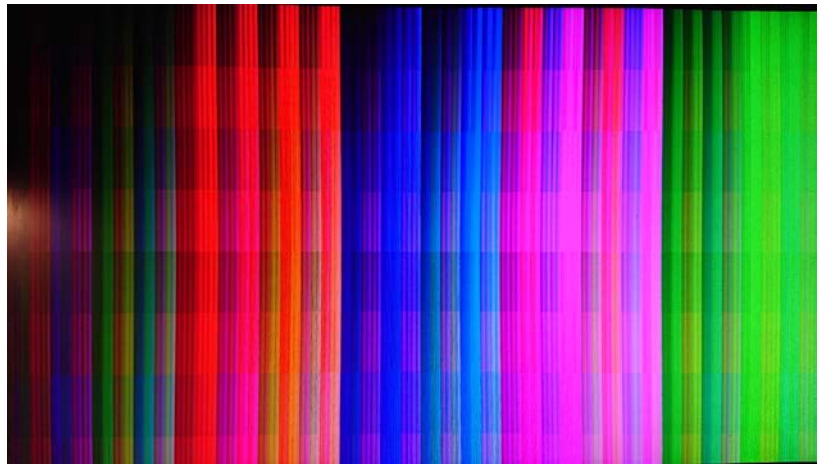
```
Adr <= Pixel;
xfunc <= sw(3);
```

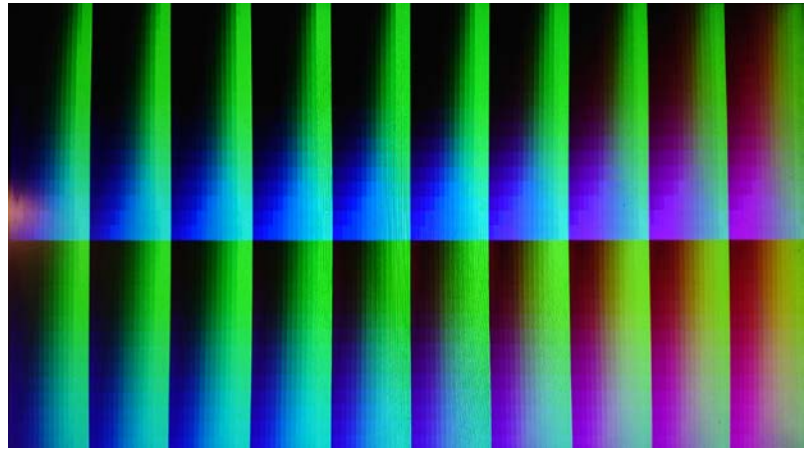


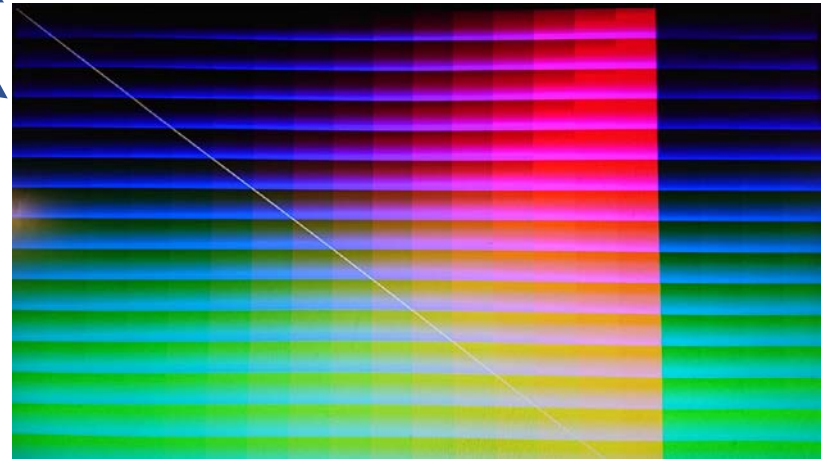



```
if xfunc='1' then
    vAdr := conv_integer( Xakse(7 downto 0));
    if vAdr>127 then
        vAdr := 255-vAdr;
    end if;
else
   vAdr := conv_integer( Xakse);
end if;
```

```
79              case Sw(7 downto 6) is
80                  when "01" => ----------------------Vis et testbillede --------
81                      ------------Blue-----
82                      RGB(0)  <= LINES(7);
83                      RGB(1)  <= PIXEL(2);
84                      RGB(2)  <= PIXEL(5);
85                      RGB(3)  <= PIXEL(8);
86                       ------------Green-----
87                      RGB(4)  <= PIXEL(0);
88                      RGB(5)  <= PIXEL(3);
89                      RGB(6) <= PIXEL(6);
90                      RGB(7) <= PIXEL(9);
91                      ------------Red-----
92                      RGB(8)  <= LINES(6);
93                      RGB(9)  <= PIXEL(1);
94                      RGB(10)  <= PIXEL(4);
95                      RGB(11)  <= PIXEL(7);
96                  when "10" => ----------------------Vis et testbillede --------
97                      RGB <=    PIXEL( 9 downto 2) & LINES( 7 downto 4);
98                  when "11" => ----------------------Vis et testbillede --------
99                      RGB <=    PIXEL( 8 downto 5) & LINES( 8 downto 1);
100                     if LINES=PIXEL then
101                         RGB <= x"FFF";
102                     end if;
103 --                  if PIXEL=AD1(11 downto 3) then
104 --                      GBR <= x"444";
105 --                  end if;
106                 when others => --------------------Vis indholdet af Mem --------
107                     -- Bemærk at PIXEL er koblet til adrb (adresse bus til mem)
108 --                  if (LINES-112)=doutb then
109 --                      GBR <= x"00F";
110 --                  end if;
111             end case;
112         end if;
113     end if;
114 end process;
115
116 end Behavioral;
```

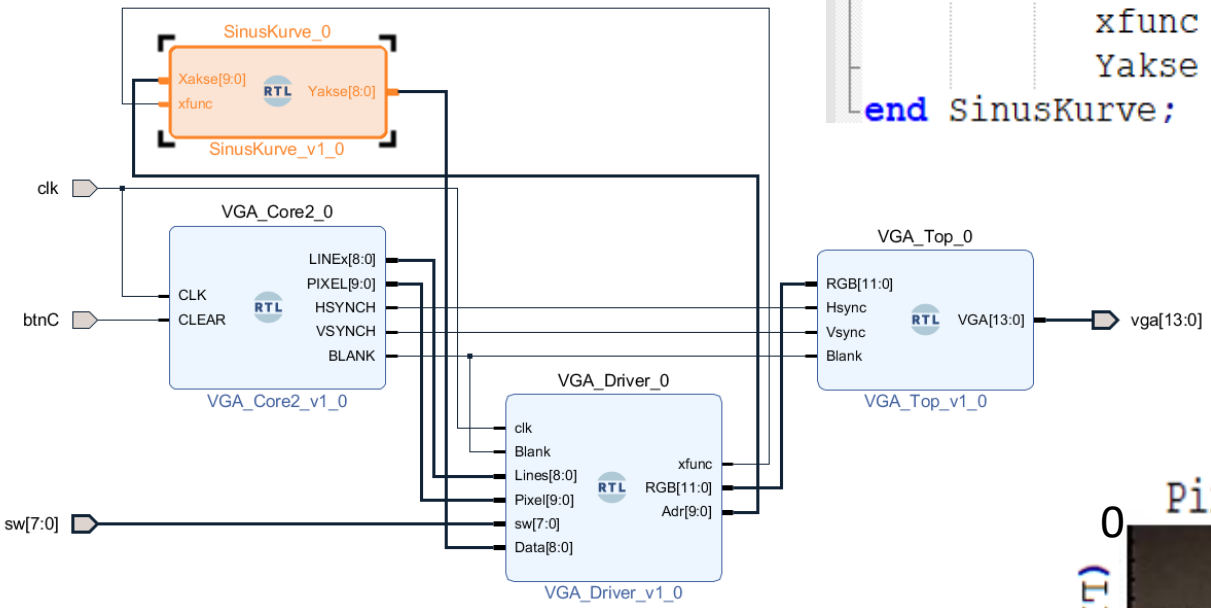


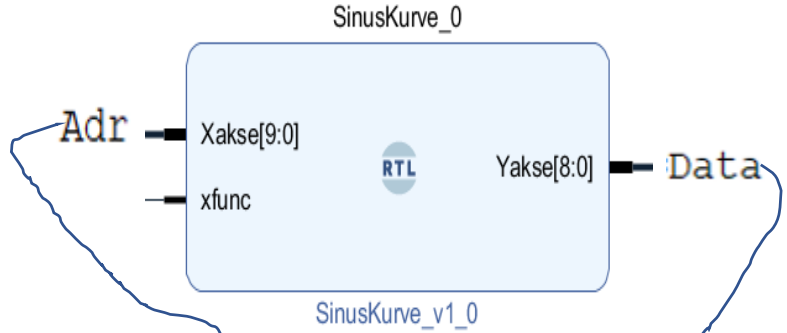




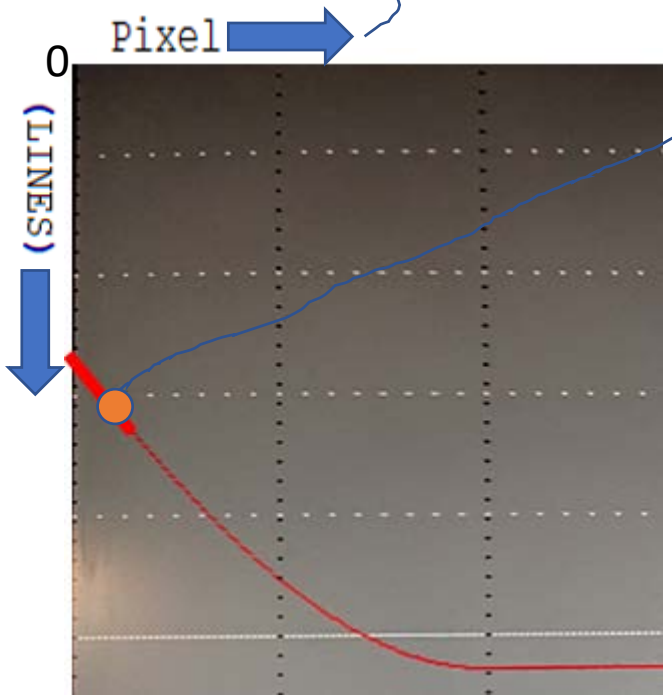
SinusKurve – Er lavet af mangel på bedre (RAM/ROM komponenter)

```
entity SinusKurve is
    Port ( Xakse : in STD_LOGIC_VECTOR (9 downto 0);
           xfunc : in STD_LOGIC;
           Yakse : out STD_LOGIC_VECTOR (8 downto 0));
end SinusKurve;
```



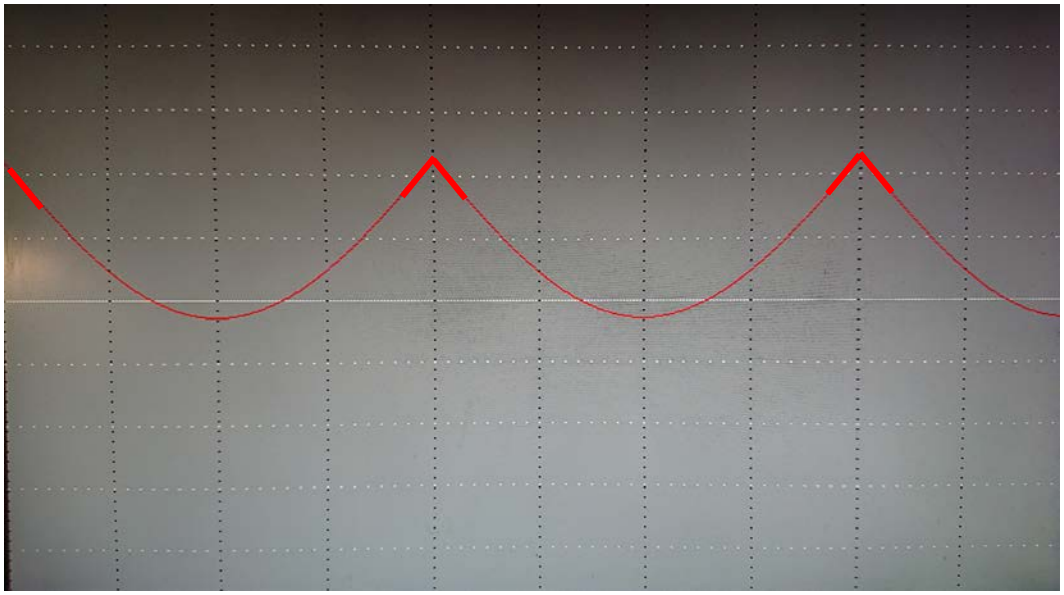


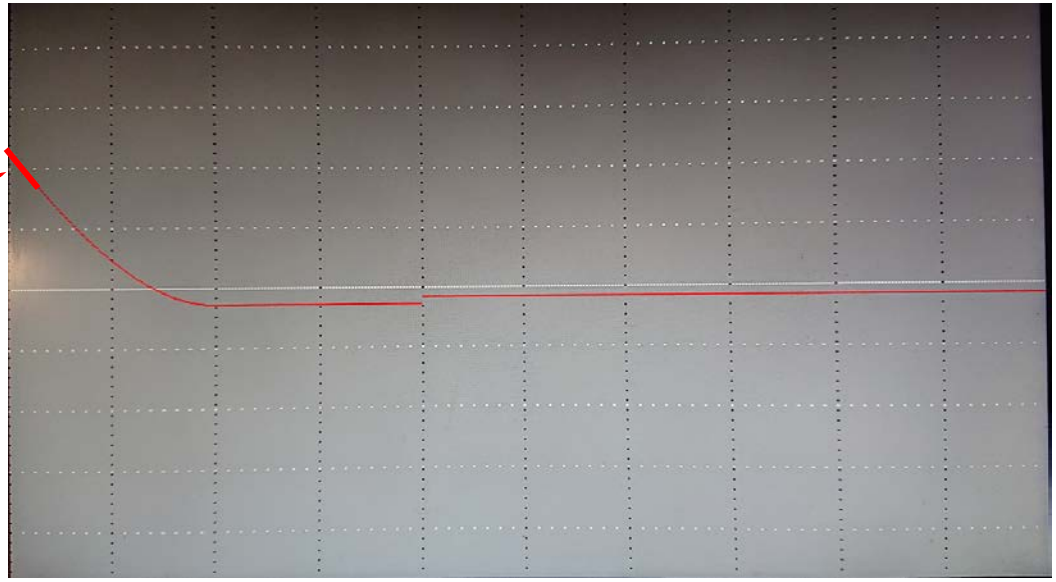

```
architecture Behavioral of VGA_Driver is
begin
    Adr <= Pixel;
    xfunc <= sw(3);

    if Sw(2)='1' then
        if (LINES)=Data then
            RGB <= x"F00";
        end if;
    end if;
```



```vhdl
entity SinusKurve is
    Port ( Xakse : in STD_LOGIC_VECTOR (9 downto 0);
           xfunc : in STD_LOGIC;
           Yakse : out STD_LOGIC_VECTOR (8 downto 0));
end SinusKurve;

architecture Behavioral of SinusKurve is
    signal Sin: integer := 0;
begin
    process( Xakse)
        variable vAdr: integer;
        variable vSin: integer;
    begin
        if xfunc='1' then
            vAdr := conv_integer( Xakse(7 downto 0));
            if vAdr>127 then
                vAdr := 255-vAdr;
            end if;
        else
           vAdr := conv_integer( Xakse);
        end if;

        case vAdr is
            when 0  => vSin := 127;
            when 1  => vSin := 129;
            when 2  => vSin := 130;
            when 3  => vSin := 132;
            when 4  => vSin := 133;
            when 5  => vSin := 135;
            when 6  => vSin := 136;
            when 7  => vSin := 138;
            when 8  => vSin := 139;
            when 9  => vSin :=141;
            when 10 => vSin :=143;
            when 11 => vSin :=144;
            when 12 => vSin :=146;
            when 99 =>  vSin :=246;
            when 100 => vSin :=247;
            when 101 => vSin :=247;
            when 102 => vSin :=248;
            when 103 => vSin :=248;
            when 104 => vSin :=249;
            when 105 => vSin :=249;
            when 106 => vSin :=249;
            when 107 => vSin :=250;
            when 108 => vSin :=250;
            when 109 => vSin :=251;
            when 110 => vSin :=251;
            when 111 => vSin :=251;
            when 112 => vSin :=252;
            when 113 => vSin :=252;
            when 114 => vSin :=252;
            when 115 => vSin :=252;
            when 116 => vSin :=253;
            when 117 => vSin :=253;
            when 118 => vSin :=253;
            when 119 => vSin :=253;
            when 120 => vSin :=253;
            when 121 => vSin :=254;
            when 122 => vSin :=254;
            when 123 => vSin :=254;
            when 124 => vSin :=254;
            when 125 => vSin :=254;
            when 126 => vSin :=254;
            when 127 => vSin :=254;
            when others => vSin  := 255;
        end case;

        Yakse <= conv_std_logic_vector( vSin,9);
    end process;
end Behavioral;
```

```
entity SinusKurve is
    Port ( Xakse : in STD_LOGIC_VECTOR (9 downto 0);
           xfunc : in STD_LOGIC;
           Yakse : out STD_LOGIC_VECTOR (8 downto 0));
end SinusKurve;

architecture Behavioral of SinusKurve is
    signal Sin: integer := 0;
begin
    process( Xakse)
        variable vAdr: integer;
        variable vSin: integer;
    begin
        if xfunc='1' then
            vAdr := conv_integer( Xakse(7 downto 0));
            if vAdr>127 then
                vAdr := 255-vAdr;
            end if;
        else
           vAdr := conv_integer( Xakse);
        end if;

        case vAdr is
            when 0  => vSin := 127;
            when 1  => vSin := 129;
            when 2  => vSin := 130;
            when 3  => vSin := 132;
            when 4  => vSin := 133;
            when 5  => vSin := 135;
            when 6  => vSin := 136;
            when 7  => vSin := 138;
            when 8  => vSin := 139;
            when 9  => vSin :=141;
            when 10 => vSin :=143;
            when 11 => vSin :=144;
            when 12 => vSin :=146;
```





Forslag til forberinger / opgaver:

1) Lav en ”rigtig” sinuskurve ved hjælp af de data som ligger lagret i SinusKurve

2) Sørg for at den er placeret symmetrisk omkring midten (den hvide streg)

3) Brug SPI interfacet som er lavet til PmodAD2 til få vist de aktuelle analoge værdier (2 forskellige værdier) som vandrette streger i selvvalgte farver.

4) Overvej sample-rate af de analoge værdier.