Systemkonstruktion med VHDL

Obligatorisk uppgift nr 1a

**Versionshistorik**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Ansvarig** | **Beskrivning** |
| 0.0 | 2011 | LL | Skapad |
| 1.0 | 2015-04-14 | LL | Uppdaterat enligt mall |
| 1.1 | 2015-05-13 | ML | Granskad och reviderad |
| 2.0 | 2016-05-10 | AN | Uppdaterad till DE1-SoC |
| 2.1 | 2016-08-16 | ML | Granskad och reviderad |

Innehåll

[1 Inledning 2](#_Toc459104163)

[2 Kravspecifikation 2](#_Toc459104164)

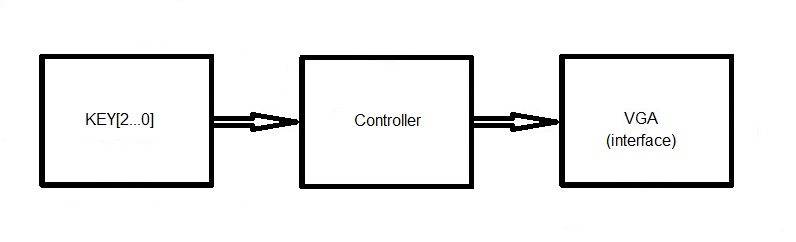
[3 Bilagor 4](#_Toc459104165)

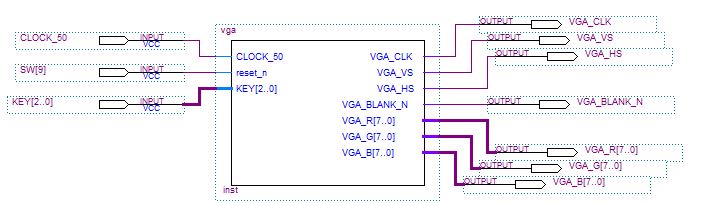
[Bilaga A VHDL-kod från en lång fika rast 4](#_Toc459104166)

[Bilaga B Protokollet som ska verifieras i ModelSim 8](#_Toc459104167)

[Bilaga C Exempel på resultat efter verifiering av VGA-protokollet 10](#_Toc459104168)

[Bilaga D Beskrivning av hur testfallen ska valideras 12](#_Toc459104169)





Figur : Två olika figurer för systemarkitekturen av det inbyggda VGA-systemet (top level).

# 1 Inledning

Nästa kunduppdrag är att förbättra och slutföra ett arbete med en VGA- kontroller. Första uppgiften är att bygga en VGA-prototyp som styrs av några test knappar och där resultatet visas på en VGA-skärm (för att validera konstruktionen). Vidare behövs även en VGA-kabel mellan kortet och skärmen. Se figur 1.

Den föregående konstruktören har flyttat och går inte att få tag på, men han har efterlämnat sin ”kladdiga” kod. Det var ett tag sedan koden skrevs och några kommentarer är skrivna i rött. Som hjälp finns en genomgång av hur pulser skapas i en föregående teorigenomgång.

Det finns mycket att läsa om VGA. Använd 25 Mhz pixel frekvens. Se gärna teori om VGA-protokollet.

Chefen har sagt att konstruktionen ska verifieras tills den är bugg-fri. Därefter ska FPGAn programmeras och då bör konstruktionen fungera första gången. Chefen anser att ingenjören har kompetens att verifiera konstruktioner av denna komplexitet.

Rapporten kommer att kunna återanvändas i senare uppgifter. Det är en mycket bra investering att lära sig och förstå VGA-protokollet.

# 2 Kravspecifikation

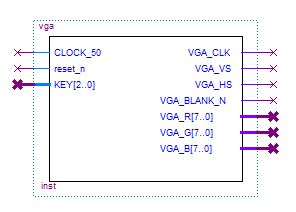
Tabell : Kravspecifikation från kund

|  |  |  |
| --- | --- | --- |
| **Krav id** | **Beskrivning** | **Utfört Ja/nej** |
| **Konstruktionskrav** | | |
| **1** | Skapa ett Quartus-projekt med namnet vhdl2\_uppgift\_1a |  |
| **2a** | Kravspecifikationen är att konstruera en VGA-kontroller, kommentera koden, verifiera med testbänk (före valideringen) och validera koden på DE1-kortet. |  |
| **2b** | VGA-protokollet ska följas och dela 50 MHz klockan i en process för att få 25 MHz. |  |
| **Testprotokoll** | | |
| **3** | Testfall: VGA protokollets styrsignaler testas med ModelSim. Följande signaler ska verifieras: VGA\_HS, VGA\_VS, VGA\_CLK och VGA\_BLANK\_N.  Se bilaga.  Validering på FPGA kortet:  Testfall 1: Rita en röd fyrkant i övre vänster hörn när KEY(0) trycks ned  Testfall 2: Rita en grön fyrkant i nedre vänster hörn när KEY(1) trycks ned  Testfall 3: Rita en blå fyrkant i nedre höger hörn när KEY(2) trycks ned  Dessa färger ska överlappa varandra och de ska blandas i mitten av skärmen. Se figur 14 i bilaga D. |  |
| **VHDL-kod** | | |
| **4** | Regler och riktlinjer för VHDL och C ska följas. |  |
| **Verifierings-/Valideringskrav** | | |
| **5a** | Verifiera med ModelSim. |  |
| **5b** | Validera på DE1-kortet. Fyll i testprotokollet. |  |
| **Rapportkrav** | | |
| **6** | Skriv en rapport (pdf eller word):   1. Framsida med titel, ditt namn, e-post, datum och en kort sammanfattning 2. Innehållsförteckning, sidnumrering och kapitel nummer 3. Kravspecifikation, eget kapitel. 4. Beskriv VGA-protokollet. Ett tydligt pulsdiagram i förhållande till H- och V-räknarna för VGA-signalerna: VGA\_HS, VGA\_VS, VGA\_CLK, VGA\_BLANK\_N. (detta bör göras innan koden skrivs!), eget kapitel. 5. Testprotokoll för verifiering och ett enkelt protokoll för validering (olika protokoll), eget kapitel. 6. Konstruktionsbeskrivning, eget kapitel.    1. System arkitektur    2. Delsystem 7. Verifiering, Eget kapitel.    1. Testbänken, Teststimuli (som bygger på testprotokollet för programmet)    2. Bilder på resultatet från simuleringen med ModelSim (se bilaga C för exempel)    3. Ifyllt testprotokoll för verifiering 8. Validering, beskriv resultatet med ett foto.    1. Ifyllt testprotokoll för validering (ej samma som verifiering) 9. Bilaga    1. VHDL-fil    2. Do-fil    3. Testbänk   Ett dokumentationskrav är också att figurer och tabeller ska ha numrering och beskrivning. Figurbeskrivning under figuren och tabellbeskrivning ovanför tabellen. |  |
| **Leveranskrav** | | |
| **7** | Leveransen ska ske till Itslearning. Leveransen ska vara rapporten och det arkiverade projektet. Namnet på filen ska vara ”förnamn\_efternamn\_vhdl2\_uppgift\_1a”. Sista leveransdag se kursschema. |  |

# 3 Bilagor

## Bilaga A VHDL-kod från en lång fika rast

Koden i bilagan ger bara idéer på hur den kan struktureras. Detta förslag är från en kladd-kopia från en lång kafferast! Se det som ett förslag. In- och utsignaler se figur 2.



Figur : Gränssnitt och utvecklingskort. (Symbol genererad från Quartus)

**Komponent namn:** VGA-kontroller (tre processer, se figur 3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Beskrivning: Dela 50Mhz klockan till 25Mhz för VGA synkpulser och räknare | | |
|  |  | Klockdelare |  |  |  |
|  |  | **Process** |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Beskrivning: Horisontell och vertikal räknare Horisontell räknare (x): 0-799, Vertikal räknare (y): 0-524 | | |
|  |  | Räknare |  |  |  |
|  |  | **Process** |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Beskrivning: Egen process där signalerna från ingångarna går genom två D-vippor för att stabilisera signalerna | | |
|  |  | Metastabilitet |  |  |  |
|  |  | **Process** |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | Syncsignaler |  |  |  | Beskrivning: Syncsignaler och RGB signaler kan kopplas utanför en process.  RGB signalerna sätts till "11111111" när knapp är nedtryckt, och "00000000" när knapp inte är nedtryckt | | |
|  |  | RGB-utgångar |  |  |  |
|  |  |  |  |  |  |
|  |  | **Parallellt** |  |  |  |
|  |  |  |  |  |  |

Figur : Förslag på mjukvaruarkitektur

Information som kan vara bra att veta:

1. En signal kan endast tilldelas ett värde i en och samma process. Däremot kan alla andra processer läsa signalen.
2. Horisontalräknare (x) ska räkna från 0 till 799, även om den synliga bilden endast är 640 pixlar bred.
3. Vertikalräknare (y) ska räkna från 0 till 524, även om den synliga bilden endast är 480 pixlar hög. Några klockpulser måste finnas utanför den synliga bilden för synkpulserna.
4. Vertikalräknare (y) uppdateras när horisontalräknare (x) är på 707.
5. Signalen VGA\_BLANK\_N ska vara ’0’ när räknarna är utanför bilden. Principen är att inget ritas när elektronstrålen sveper tillbaka över skärmen. Det kan beskrivas som att en penna hålls upp mellan ny rad (x), eller ny sida (y).
6. Olika sidor på internet kan specificera andra tider för VGA\_HS, och VGA\_VS. Detta på grund av att VGA-skärmen automatiskt kalibrerar sig till de signaler som skickas.
7. Den externa krets (ADV7123) som är kopplad till VGA\_R, VGA\_G och VGA\_B tar en klockcykel på VGA\_CLK för att göra omvandlingen från digital till analog.
8. Om en utgång inte blir tilldelad ett värde vid varje klockpuls, kommer en latch att användas. Denna kommer då att spara det tidigare värdet på utgången. Latchar måste ha en klockpuls för att kunna uppdateras, och ibland kan detta leda till att en signal räknas som en klocka.

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

LIBRARY work;

ENTITY vga IS

PORT

(

CLOCK\_50 : in std\_logic;

reset\_n : in std\_logic;

KEY : in std\_logic\_vector(2 downto 0);

VGA\_CLK : out std\_logic;

VGA\_VS : out std\_logic;

VGA\_HS : out std\_logic;

VGA\_BLANK\_N : out std\_logic;

VGA\_R : out std\_logic\_vector(7 downto 0);

VGA\_G : out std\_logic\_vector(7 downto 0);

VGA\_B : out std\_logic\_vector(7 downto 0)

);

END vga;

ARCHITECTURE rtl OF vga IS

signal x\_counter : unsigned(9 downto 0);

signal y\_counter : unsigned(9 downto 0);

---------- could use a record for x/y counters ----------

type HV\_type is -- Horizontal/Vertical type

record

H : integer range 0 to 1023; -- Horizontal (x) signal

V : integer range 0 to 525; -- Vertical (y) signal

end record;

signal counter : HV\_type;

-- Access it by typing:

-- counter.H <= counter.H + 1;

---------- Might be overkill to use a record ----------

-- More signals are needed

begin

------------------ Clock divider process ------------------

clock\_divider:process(CLOCK\_50, reset\_n)

begin

if reset\_n = '0' then

clk\_25 <= '0';

elsif rising\_edge(CLOCK\_50) then

-- Divide the clock by two

clk\_25 <= not clk\_25;

end if;

end process;

------------------ Clock divider process ------------------

------------------ Metastability process ------------------

Metastability:process(CLOCK\_50, reset\_n)

begin

if reset\_n = '0' then

-- Something might need to be reset

elsif rising\_edge(CLOCK\_50) then

-- There should be a double D-flip-flop for the inputs

-- first\_flip\_flop <= key;

-- second\_flip\_flop <= first\_flip\_flop;

end if;

end process;

------------------ Metastability process ------------------

------------------ Counters process ------------------

process(clk\_25, reset\_n)

begin

if reset\_n = '0' then

-- clear counter signals

x\_counter <= (others => '0');

y\_counter <= (others => '0');

elsif rising\_edge(clk\_25) then

-- counters

-- increment x\_counter (counter.H) every clock pulse

------------ x\_counter ------------

if x\_counter >= 799 then

x\_counter <= (others => '0');

else

x\_counter <= x\_counter + 1; -- increment x\_counter

end if;

------------ x\_counter ------------

-- increment y\_counter (counter.V) when x\_counter is 707

------------ y\_counter ------------

if x\_counter = 707 then

if y\_counter = 525 then -- is this the correct value?

y\_counter <= (others => '0');

else

y\_counter <= y\_counter + 1; -- increment y\_counter

end if;

end if;

------------ y\_counter ------------

end if;

end process;

------------------ Counters process ------------------

------------------ Concurrent statements ------------------

-- Output divided clock to VGA\_CLK

VGA\_CLK <= clk\_25;

------- Sync pulses -------

VGA\_HS <= '0' when x\_counter > (What was it...) and x\_counter < (Have to check) else '1';

VGA\_BLANK\_N <= '1' when x\_counter < (Hmm.. what value...) and y\_counter < (Forgot...) else '0';

VGA\_VS <= '0' when y\_counter = (This was active only for one pulse, but which one) else '1';

------- Sync pulses -------

------- RGB signals for validation -------

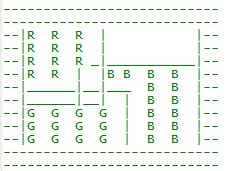
VGA\_R <= (others => '1') when (Check the x/y chart below) and KEY(0) = (pressed) else (others => '0');

VGA\_G <= (others => '1') when (Check the x/y chart below) and KEY(1) = (pressed) else (others => '0');

VGA\_B <= (others => '1') when (Check the x/y chart below) and KEY(2) = (pressed) else (others => '0');

------- RGB signals for validation -------

------------------ Concurrent statements ------------------



-- Red x from 0 to 360, y from 0 to 320

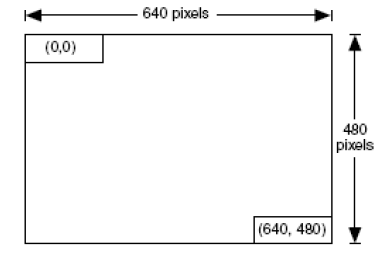
-- Green x from 0 to 410, y from 280 to 480

-- Blue x from 300 to 640, y from 240 to 480

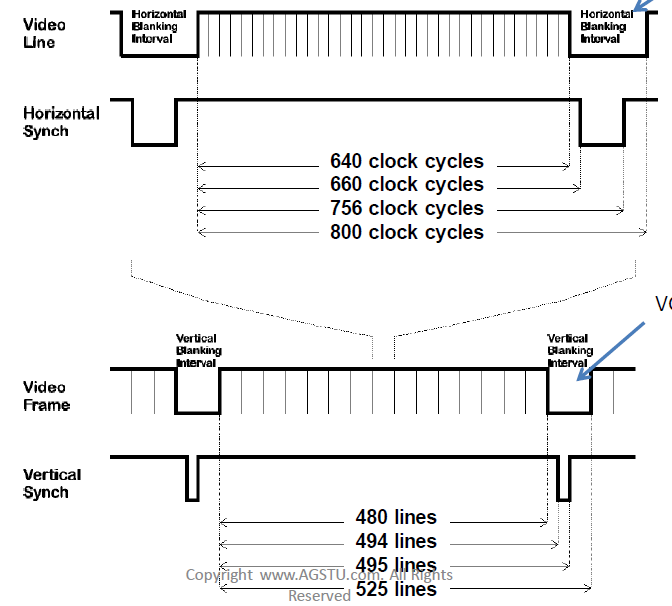
end rtl;

## Bilaga B Protokollet som ska verifieras i ModelSim

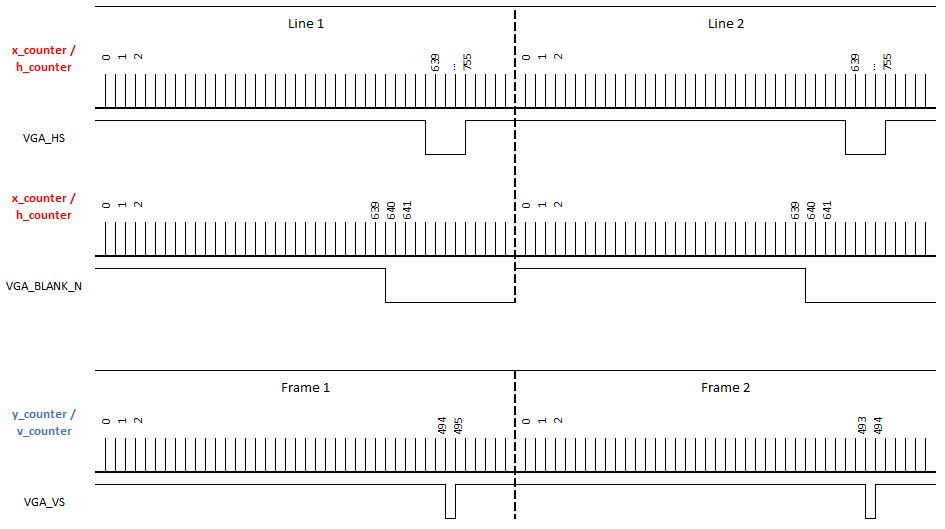
Läs manualen och om det behövs mera information, fråga kollegor eller leta på webben. Det är viktigt att förstå hur protokollet fungerar, därefter kan konstruktionen påbörjas. Fråga handledare om det behövs!



Figur : Översikt, Display time är 640 \*480 Pixlar, Blanking time är (525-480) 45 rader efter sista vertikala raden och (800-640) 160 pixlar per rad.



Figur : VGA-protokollet.



Figur : Visar på protokollet på ett annat sätt

**En förutsättning för att klara av att hantera och använda protokollet är att verkligen förstå hur det fungerar! Fortsätt att analysera tills förståelse av protokollet är 100 %! Se figurerna 4-6 ovan.**

## Bilaga C Exempel på resultat efter verifiering av VGA-protokollet

Följande figurer (figur 7-13) visar olika signaler efter verifiering i ModelSim.



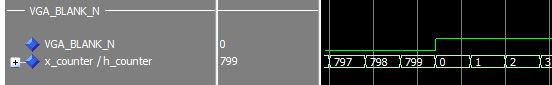
Figur . Fallande flank VGA\_HS, horisontell synkpuls



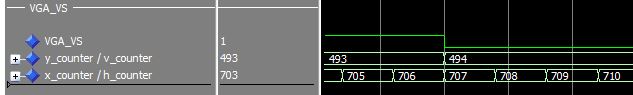
Figur . Stigande flank VGA\_HS



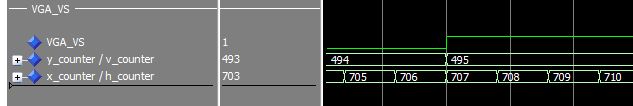
Figur . Fallande flank VGA\_BLANK\_N, räknare utanför synlig bild



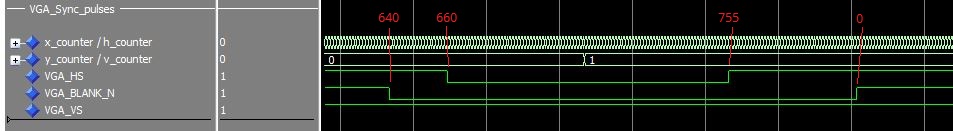
Figur . Stigande flank VGA\_BLANK\_N



Figur . Fallande flank VGA\_VS, vertikal synkpuls



Figur . Stigande flank VGA\_VS



**Figur 13. Övergång till nästa rad**

## Bilaga D Beskrivning av hur testfallen ska valideras

Figur 14 visar var röd, grön och blå färg ska ritas ut på skärmen enligt testfallen. I mitten ska färgerna blandas med varandra.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  | X=360 | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Y=240 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Y=280 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Y=320 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | X=300 | |  |  | X=410 | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Figur 14. Validering av VGA-komponenten. I mitten ska färgerna blandas med varandra**