Poročilo

Digitalni elektronski sistemi

Generator slike za logični analizator Red Pitaya v jeziku VHDL

Avtor: Jernej Slak
Mentor: izr. prof. dr. Andrej Trost, univ. dipl. inž. el.
Šolsko leto: 2015/16

Kraj: Ljubljana
Datum: 7.6.2016

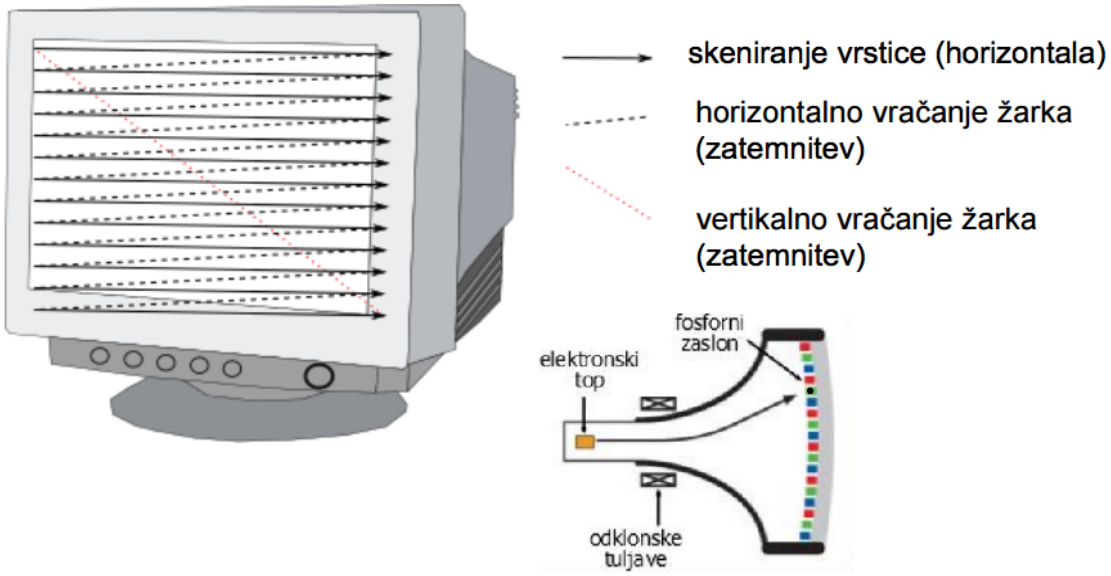
1 Uvod

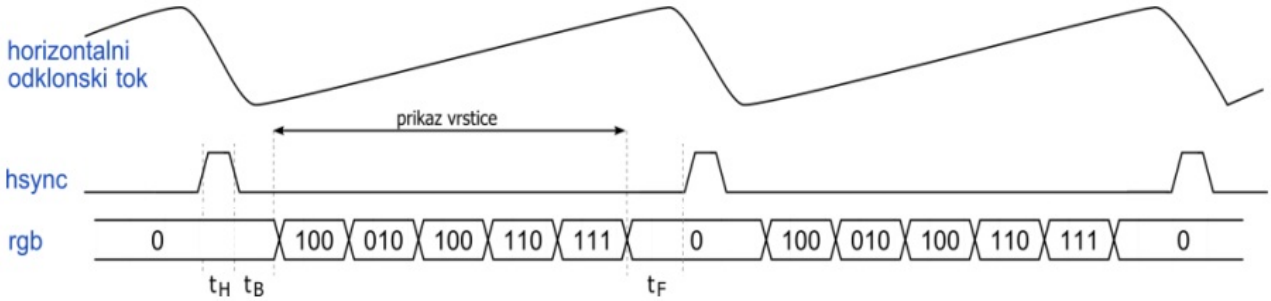
V sklopu laboratorijskih vaj smo izdelovali logični analizator za Red Pitayo. Analizator sestavlja tiskano vezje z vhodi in izhodi ter program, ki omogoča zajem, shranjevanje in posredovanje signalov Red Pitayi, ki jih nato nadaljnje analizira. Projekt je sestavljen iz treh delov: arhitekture, grafike ter logike. Celoten projekt je združeval te tri dele: vezje z vhodi in izhodi (arhitektura), grafični vmesnik (grafika) ter vmesnik za zajem logičnih signalov (logika).

Namen projekta grafične skupine je bil izdelava vmesnika, ki je zmožen prikazovati sličico iz pomnilnika, črke in številke ter grafikon z logičnimi signali.

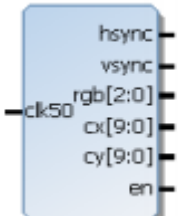
2 VGA komponenta in sinhronizacijski signali

VGA protokol izhaja iz dobe katodnih monitorjev, kjer je bilo krmiljenje prikaza izvedeno z žagastima tokovoma. S pomočjo katodne cevi sta ta tokova ukrivljala žarek elektronov in tako na zaslon izrisovala vrstice. Ko je žarek prišel do konca vrstice, je potreboval določen čas, da se je vrnil na začetek nove vrstice, tačas pa je moral biti izklopljen. Enako je bilo, ko je žarek prišel do dna zaslona ter se vrnil v zgornji levi kot.



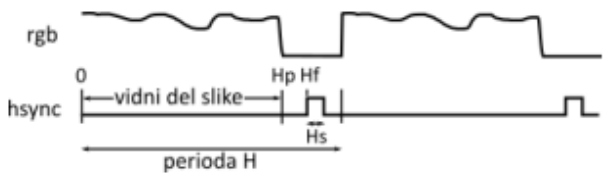


V našem vezju za sinhronizacijo signalov skrbi komponenta banana, ki je narejena za prikaz slike ločljivosti 800x600 točk in frekvence osveževanja 72 Hz Za vhod ima uro s frekvenco 50 MHz kot izhodne signale pa **hsync**, **vsync**, **cx**, **cy**, **rgb** in **en**.



Sinhronizacijska signala **hsync** in **vsync** povesta kdaj žarek potuje na začetek nove vrstice. Števca **hst** in **vst** uporabimo za določanje koordinate trenutne točke. Signal **en** nam pove kdaj sta števca **hst** in **vst** v vidnem delu slike, signal **rgb** pa pove barvo na izhodu.

Z nastavljanjem konstant **H**, **Hp**, **Hf** in **Hs** (ter ekvivalentnih za **vsync**) določamo začetek **hysnc** pulza in njegovo trajanje



VGA 800x600	število period	število vrstic
perioda	H = 1040	V = 666
vidni del	Hp = 800	Vp = 600
začetek pulza	Hf = 856	Vf = 637
trajanje pulza	Hs = 120	Vs = 6

3 Vezje za prikaz sličice

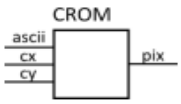
V vezje za prikaz slike na monitorju smo dodali logiko, ki prikaže v vogalu monitorja enobitno sličico. Prikaz sličice deluje tako, da se na osnovi trenutnih koordinat **cx** ter **cy** odločamo kakšna bo vrednost barvnega izhoda **rgb3**.

V vezju za prikaz sličice deklariramo pomnilnik ROM, ki je dvodimenzialna zbirka velikosti 80x20 točk, kot signal **slika**, enobitni signal **tocka** ter konstanti **xpre** ter **ypre** s katerima premikamo pozicijo logotipa na zaslonu. V procesu preverjamo pozicijo na zaslonu: ko smo v območju med [**xpre**, **ypre**] in [**xpre**+80, **ypre**+20] nam signal **tocka** bere zbirko **slika** ter zavzame vrednost '0' ali '1' elementa v zbirki, **rgb3** pa nato na podlagi **tocke** pove barvo na izhodu. Ko smo izven območja pa **rgb3** zavzame vrednost ozadja **rgb2**, ki pride iz VGA komponente (banana).

```
logo: process(cx, cy)
begin
    if (cx >= xpre) and (cy >= ypre) and (cx < (xpre+80)) and (cy < (ypre+20)) then
        tocka <= slika( to_integer(cy - ypre) )( to_integer(cx - xpre));
            if tocka > '0' then
                rgb3 <= "111";
            else
                rgb3 <= "000";
            end if;
     else
        rgb3 <= rgb2;
     end if;

end process;
```

4 Vezje za prikaz besedia



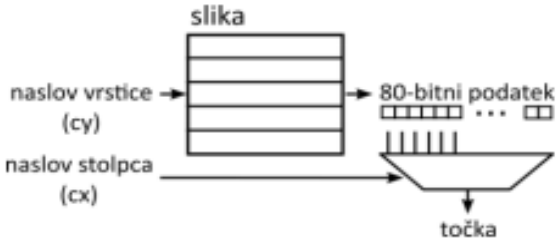
V vezje VGAtest dodamo komponento CROM z vhodi **ascii**, **cx** in **cy** ter izhodom **pix**, ki je odvisen od vseh treh vhodov. Komponenta CROM (Character ROM) vsebuje pomnilnik s 64 sličicami črk (ASCII kode od 0x20 do 0x5F).

7-bitnemu vektorju **ascii** priredimo del elementa iz zbirke **reg** (v našem primeru odvisne od koordinat **cx** in **cy**), na podlagi katere se v komponenti CROM priredi vrednost enobitnega izhoda **pix**.

Dolžina vsake črke je 8 točk, zato naš interval, dolg 64 točk, razdelimo na 8 delov. Kje v intervalu se nahajamo izvemo iz koordinate **cx**. Vrednost koordinate pretvorimo v številko intervala tako, da koordinato delimo z 8, kar v dvojiškem zapisu pomeni da ji odstranimo zadnje tri bite, nakar dobljeno dvojiško vrednost pretvorimo v celo število.

Prek pogojnega stavka določamo barvo izhoda **rgb3** glede na vrednost **pix** in s tem izrisujemo znak.

```
besedilo: process(cx, cy)
begin
    if (cx < 64) and (cy < 8) then
        ascii <= reg(to_integer(unsigned(cx(9 downto 3))))(6 downto 0);
        if pix = '1' then
            rgb3 <= "111";
        else
            rgb3 <= "000";
        end if;
    end if;
end process;
```



Za namene vaje smo ustvarili zbriko osmih znakov ter jih nato izrisali na zaslon.

D	E	S		2	0	1	6
0x44	0x45	0x53	0x20	0x32	0x30	0x31	0x36

```
type vrstica is array(0 to 7) of std_logic_vector(7 downto 0);
signal reg: vrstica := (X"44", X"45", X"53", X"20", X"32", X"30", X"31", X"36");
```

5 Vezje za prikaz grafikona

V komponenti VGAtest deklariramo signale **d0**, **d0z**, **adr**, **stev**, poljubno konstanto **N** ter zbirko **data** velikost 16x8 bitov.

Enobitni signal **d0** nam določa kdaj se na grafu izriše logična 1 ali logična 0, **d0z** predstavlja zakasnjen podatek, ki ga uporabimo za zaznavanje fronte, števec **stev** ter konstanta **N** pa določata skalo našega grafikona. V podatkovnem pomnilniku **data** je 16 8-bitnih podatkov. 3-bitni signal **adr** nam določa pozicijo na grafikonu, njegova celoštevilska vrednost nam zato določa tudi kateri element v zbirki beremo. Z vrednostjo **cy** določamo kateri bit elementa beremo, tako da je vsaki vrstici dodeljen svoj bit.

```
d0 <= data(to_integer(adr))(to_integer(cy(5 downto 3)));
```

indeks elementa bit elementa

```
p50: process(clk50)
begin
    if rising_edge(clk50) then
        d0z <= d0;
        stev <= stev + 1;

        if cx = 16 then
            adr <= "000";
            stev <= 0;
        end if;

        if stev = N then
            adr <= adr + "001";
            stev <= 0;
        end if;
```

S **cx** še preverjamo x koordinato vrstice, med 0 ter 16 izpišemo črko D ter številko od 0 do 7 (odvisno od vertikalnega položaja **cy**).

```
ascii <= "1000100" when cx < 8 else std_logic_vector("0110000" + cy(5 downto 3));
 v CROM se pix priredi vrednost glede na cx, cy in ascii
//

if cx < 16 then --izris oznake signala [Dx]
    rgb3 <= "000";
    if pix = '1' then
        rgb3 <= "111";
    end if;
```

Od **cx** = 16 dalje izrisujemo signal. **Rgb3** ki izrisuje signal je določen z **d0** in **cy** po logiki:

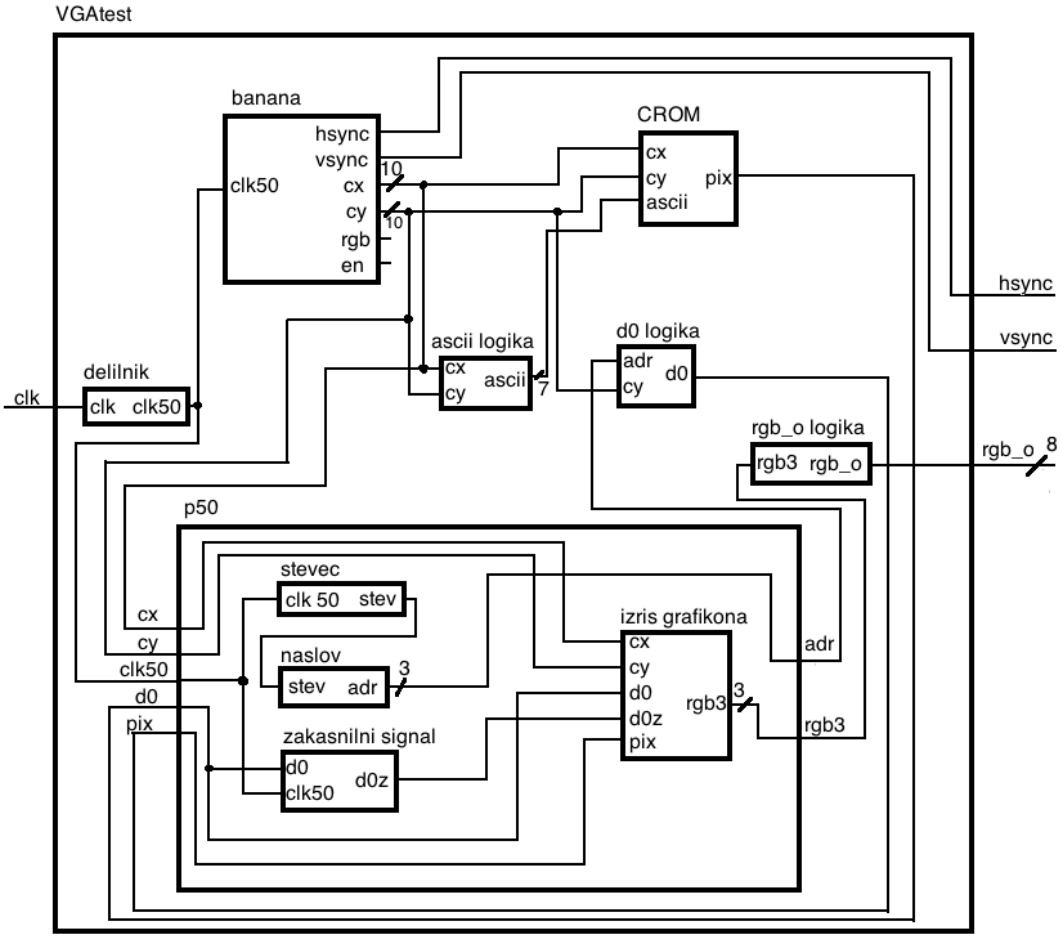
```
d0 = 1 && cy = 1 --> rgb3 = "111" /izris enice
d0 = 0 && cy = 6 --> rgb3 = "111" /izris ničle
(d0 0->1 ali 1->0) && (1 < cy < 6) --> rgb3 = "111" / izris fronte
v vseh ostalih primerih rgb3 = "000"
```

Za zaznavo fronte se poslužujemo zakasnilnega signala **d0z**, ki za **d0** zaostaja za eno urino periodo. S funkcijo xor preverimo, kdaj sta si različna, takrat namreč dobimo logično enico.

```
else --izris signala
    rgb3 <= "000";
    if (cy(2 downto 0) = 1) and (d0 = '1') then --izris logicne 1
        rgb3 <= "111";
    elsif (cy(2 downto 0) = 6) and (d0 = '0') then --izris logicne 0
        rgb3 <= "111";
    elsif (cy(2 downto 0) > 1) and (cy(2 downto 0) < 6) then --izris fronte
        if (d0 xor d0z) = '1' then
            rgb3 <= "111";
        end if;
```

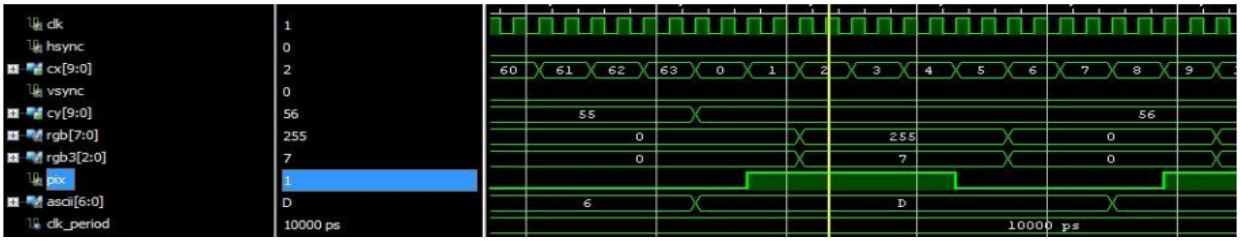
6 Povzetek delovanja vezja in blokovna shema

Vezje se nahaja v komponenti VGAtest, v kateri vstopa zunanji signal **clk**, izstopajo pa signali **hsync, vsync** ter **rgb_o**. Komponenta VGAtest v sebi vsebuje še komponenti CROM in banana. Komponenta banana skrbi za sinhronizacijo izpisa na zaslon, CROM pa vsebuje pomnilnik s znaki črk. V VGAtest je tudi logika za določitev signalov **ascii, d0** in **rgb_o.** Glavni proces v VGAtest se imenuje p50. Narekuje ga ura **clk50** s frekvenco 50 MHz. V procesu se vsako periodo priredijo signali **stev, adr** in **d0z**. Glavni del p50 je izris grafikona. Iz procesa p50 izhaja signal **adr**, ki določi **d0** ter najbolj pomemben signal **rgb3**, ki se pretvori v signal **rgb_o**.

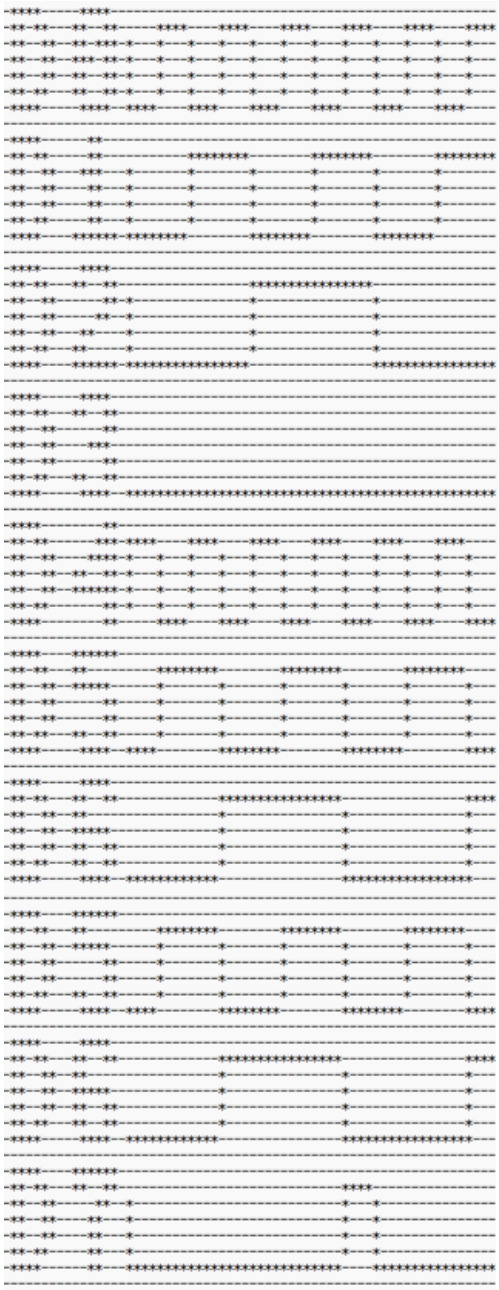


7 Časovni potek signalov in končna slika grafikona

Na spodnji sliki je prikazan izsek iz simulacije, ko preidemo iz risanja grafikona signala D6 na zadnji grafikon D7. Pri tem **ascii** signal spremeni vrednost iz 6 na D ter nato na 7. V primeru da bi izrisali še grafikon D8, bi **ascii** držal vrednost 7 do pričetka izrisovanja grafikona D8. Z modro je označen izhodni signal pix komponente CROM, ki je določa vrednost **rgb3**, ko le-ta izrisuje črke in številke.



Končna slika naše zbirke grafikonov:



8 Zaključek in možnosti nadgradnje

Vezje bi lahko nadgradili tako, da bi povečali ločjivost izrisovanja. S tem bi lahko lepše prikazali tudi sinusne signale. V trenutni verziji je vezje bolj primerno za prikazovanje diskretnih signalov.

Skalo lahko zaenkrat spreminjamo samo v horizontalni smeri z nastavitvijo parametra N. Lahko bi dodali funkcijo tudi za vertikalni razteg, vendar bi bila pomankljivost tega, da ne bi izrisalo (v našem primeru) vseh 8 grafikonov.

9 Viri

- navodila za vaje na spletni strani predmeta, http://lniv.fe.uni-lj.si/des.html
- http://en.wikipedia.org/wiki/Video_Graphics_Array