Семинар 5. Проектирование VGA контроллера

Оглавление

Формирование VGA видеосигнала

Существуют различные форматы видеосигналов: аналоговые и цифровые, с прогрессивной и чересстрочной разверткой, с частотой обновления кадра от 24 до 120 Гц, различным разрешением - от CGA (320x200) до QSXGA (2560x2048).

VGA (Video Graphics Adapter) является стандартом для мониторов и видеоадаптеров, разработанным фирмой IBM в 1987 году. Для передачи цветовой информации используется аналоговый сигнал.

Интервалы синхронизации (timings) стандарта VGA опубликованы и стандартизованы организацией VESA. Опишем управление видеосигналом для режима 640x480. Более детальная информация расположена на сайте организации VESA [30].

ЭЛТ VGA-дисплеи используют амплитудную модуляцию, перемещая электронные пучки (катодные лучи) для отображения информации на флуоресцентной поверхности экрана. ЖКИ-дисплеи используют матрицу переключателей, которая может подавать напряжение через небольшое количество жидких кристаллов, при этом изменяя проходимость световых лучей через кристалл. Хотя нижерасположенное описание применительно к ЭЛТ дисплеям, ЖКИ-мониторы используют те же интервалы синхронизации. Цветные ЭЛТ дисплеи используют три электронных пучка (для каждого из цветов – красный, синий, зеленый) для возбуждения поверхности люминофора.

Информация отображается только тогда, когда луч движется в «прямом» направлении (слева направо или сверху вниз) и не отображается во время возврата луча назад налево или в верхний угол дисплея. В связи с этим большое количество времени, которое потенциально могло бы быть использовано, теряется на эти периоды «гашения», когда луч сбрасывается и готовится к новой серии горизонтальной или вертикальной развертки. Размер лучей, частота, на которой луч может быть проходить кадр, частота, на которую электронный луч может быть модулирован, определяют разрешение дисплея. Современные VGA дисплеи могут поддерживать различные разрешения и схема видеоконтроллера определяет разрешение через задание интервалов синхронизации для управления растром. Контроллер должен обеспечить синхронизирующие импульсы 3.3В (или 5В) для установки частоты, на которой сформирован поток через отклоняющую катушку, и он должен гарантировать, что видеоданные поступают на электронную пушку в правильные моменты времени. Растровые дисплеи определяют количество «рядов», которым соответствует число горизонтальных проходов катода через пространство дисплея, и число «колонок», которым соответствует пространство, в котором ряд разбивается на пикселы. Обычно дисплеи используют от 240 до 1200 рядов и от 320 до 1600 колонок. Общие размеры дисплея и количество рядов и колонок определяют размер каждого пиксела.

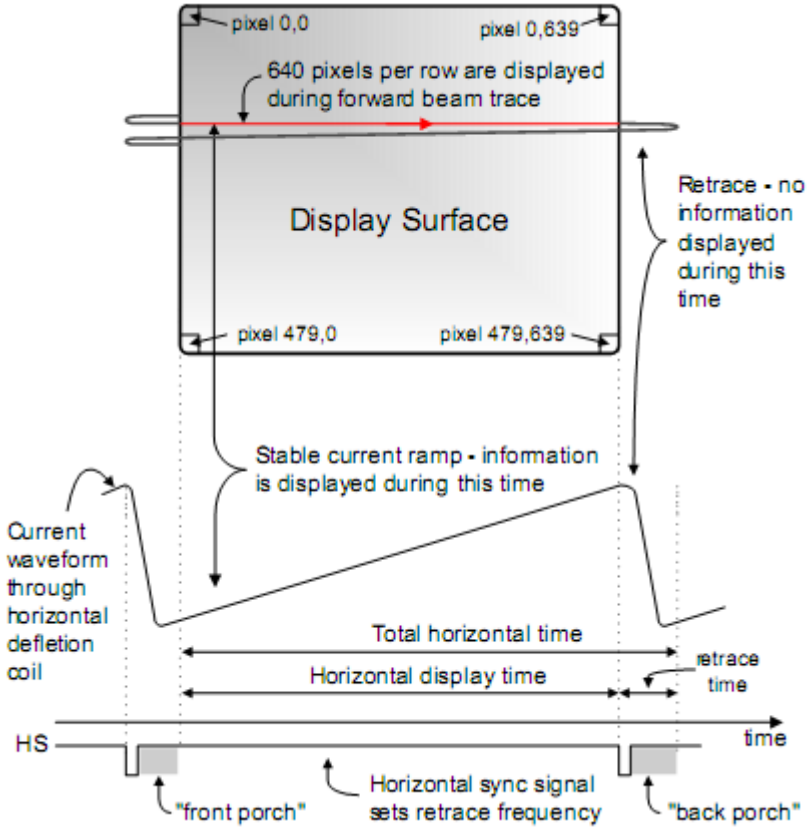


Рисунок 1 – Сигналы стандарта VGA

Видеоданные обычно формируются в обновляемой видеопамяти с кодированием пикселов одним или более байтом (в данном проекте – 3 бита на пиксел). Контроллер должен обращаться к видеопамяти, также как лучи перемещаются вдоль дисплея, выводить видеоданные на дисплей в точности в то время, когда электронный луч пересекает данный пиксел.

Схема VGA-контроллера должна генерировать синхросигналы HS и VS и координировать доставку видеоданных, связанных с частотой пиксельной развертки. Частота пиксельной развертки определяет время, доступное для отображения одного пиксела. Сигнал VS определяет частоту обновления дисплея. Минимальная частота обновления зависит от мощности электронного луча, на практике эта величина находится в диапазоне от 50 Гц до 120 Гц. Количество строк, которые можно вывести при заданной частоте обновления, определяет частоту обновления строк. Для дисплеев с разрешением 640 x 480, использующие частоту пиксельной развертки 25 МГц и обновления экрана 60 +/- 1 Гц, интервалы синхронизации показаны в таблице 3.1. Интервалы для ширины импульса и передняя и задняя площадки гашения развертки основаны на наблюдениях реальных VGA дисплеев.

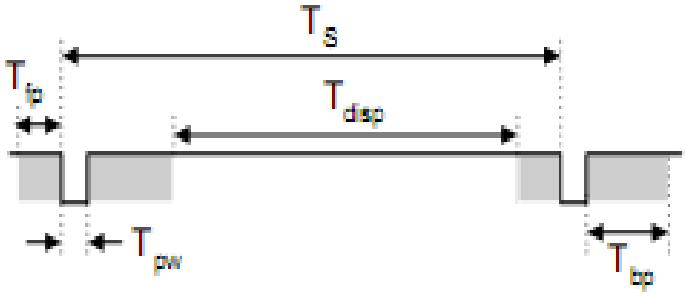
Таблица 1

Обозначение	Параметр	Вертикальная синхронизация			Горизонтальная синхронизация	
		Время	Такты	Строки	Время	Такты
TS	Sync Pulse	16.7ms	416.800	521	32us	800
Tdisp	Display time	15.36 ms	384.000	480	25.6us	640
Tpw	Pulse width	64us	1.600	2	3.84us	96

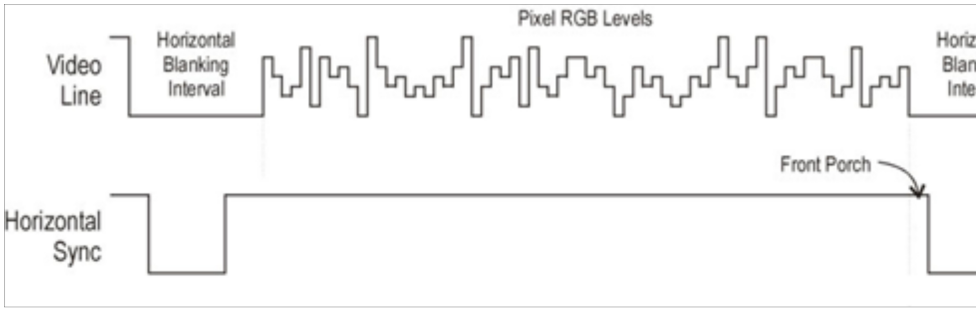
Tfp	Front porch	320us	8.000	10	640ns	16
Tbp	Back porch	928us	23.200	29	1.92us	48

Рисунок – Интервалы синхронизации для VGA дисплея с разрешением 640x480

Виды выходного сигнала в масштабе периодов, соответствующих разверткам по горизонтали и по вертикали, показаны на рисунке:



а)



б)

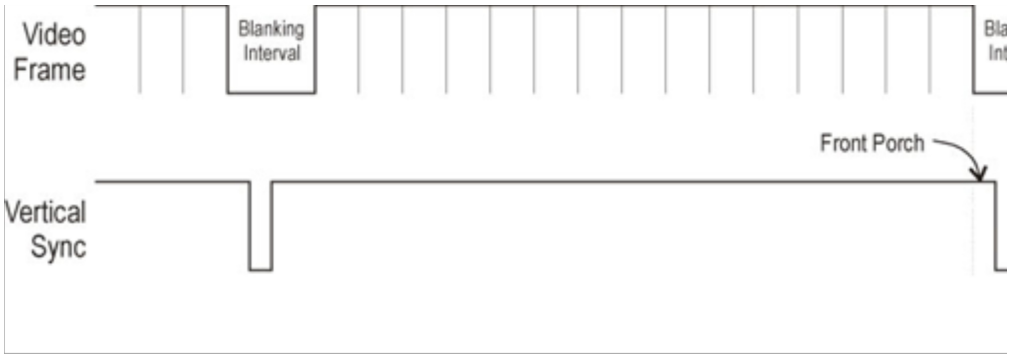


Рисунок – а) Горизонтальная синхронизация; б) Вертикальная синхронизация.

Видеоконтроллер декодирует выходной сигнал счетчика горизонтального синхросигнала для генерации временных интервалов. Этот счетчик может использоваться для размещения любого пиксела в заданном ряду. И аналогично для вертикального синхросигнала. Эти два постоянно запущенных счетчика могут быть использованы для формирования адреса в видеопамяти. Разработчик может использовать их для формирования адресов в видеопамяти или для минимизации логики декодирования для генерации синхросигналов.

Код генератора сигналов `Hsync и Vsync`

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Synchro is
   port (
                clk25: in std_logic;                                        -- ck 25MHz
                HS: out std_logic;                                          --
horizontal synchro signal
                VS: out std_logic;                                          -- verical
synchro signal
                Hcnt: out std_logic_vector (9 downto 0); -- horizontal counter
                Vcnt: out std_logic_vector (8 downto 0)      -- verical counter
                );
end Synchro;

architecture Behavioral of Synchro is
-- constants for Synchro module
constant HPW:integer:=96;           --Horizontal synchro Pulse Width (pixels)
constant HFP:integer:=16;           --Horizontal synchro Front Porch (pixels)
constant HBP:integer:=48;           --Horizontal synchro Back Porch (pixels)
constant VPW:integer:=2;            --Verical synchro Pulse Width (lines)
constant VFP:integer:=10;           --Verical synchro Front Porch (lines)
constant VBP:integer:=29;           --Verical synchro Back Porch (lines)
constant PAL:integer:=640;          --Pixels/Active Line (pixels)
constant LAF:integer:=480;          --Lines/Active Frame (lines)
-- constants for Synchro module (also used in FieldGen)
constant PLD: integer:=800;  --Pixel/Line Divider
constant LFD: integer:=521;  --Line/Frame Divider
signal intHcnt: integer range 0 to PLD-1; --PLD-1  -- horizontal counter
signal intVcnt: integer range 0 to LFD-1; -- LFD-1  -- verical counter
signal a: integer;

begin
syncro: process (clk25)
begin
if clk25'event and clk25='1' then
        if intHcnt=PLD-1 then
                intHcnt<=0;
                if intVcnt=LFD-1 then intVcnt<=0;
                else intVcnt<=intVcnt+1;
                end if;
        else intHcnt<=intHcnt+1;
        end if;

        -- Generates HS - active low
        if intHcnt=PAL-1+HFP then
                HS<='0';
        elsif intHcnt=PAL-1+HFP+HPW then
                HS<='1';
        end if;
```

```
        -- Generates VS - active low
        if intVcnt=LAF-1+VFP then
                VS<='0';
        elsif intVcnt=LAF-1+VFP+VPW  then
                VS<='1';
        end if;
end if;
end process;

-- mapping itnernal integers to std_logic_vector ports
  Hcnt <= conv_std_logic_vector(intHcnt,10);
  Vcnt <= conv_std_logic_vector(intVcnt,9);
end Behavioral;
```

Формирование аналогового видеосигнала на плате Nexus2

Необходимо 10 выводов для создания VGA порта с 8-битной цветовой гаммой и двумя стандартными сигналами синхронизации (HS – горизонтальная развертка, VS – вертикальная развертка). Сигналы цветов используют схему резистивного делителя, для того чтобы при соединении с согласующими резисторами в 75 Ом VGA дисплея создать 8 уровней сигналов по красному и зеленому каналам и 4 по синему (человеческий глаз менее восприимчив к синему спектру). Схема, показанная на рисунке 2.7, формирует цветовые сигналы, которые поступают в диапазоне от 0 В до 0.7 В.

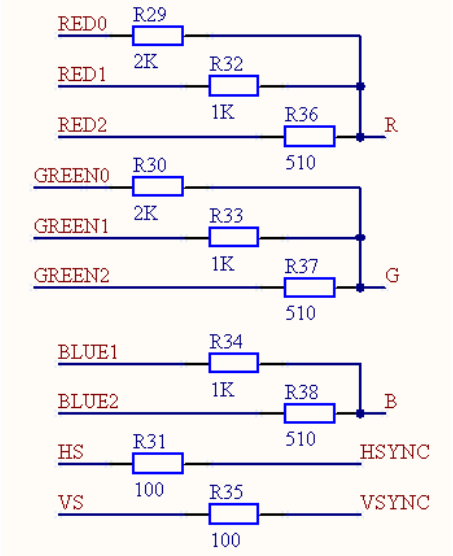


Схема формирования уровня аналогового видеосигнала

Используя такую схему, устройство может выдавать 256 различных цветов – 1 на каждую комбинацию из 8 битов. Для управления синхросигналами и сигналами цветов с верными временными интервалами, соответствующими рабочему состоянию видеоподсистемы, необходимо разработать схему видеоконтроллера внутри кристалла ПЛИС.

HDL-описание системы

Описание системы на языках HDL представляет собой многоуровневую структуру. Дерево проекта представлено на рисунке:

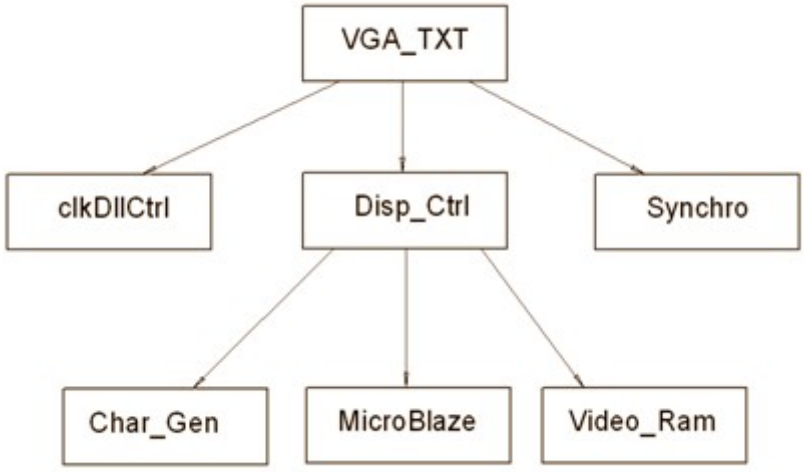


Рисунок - Дерево проекта

Объект VGA_TXT является верхним в иерархии, описание его портов ввода/вывода представляет собой интерфейс всей системы:

```
entity VGA_TXT is
  port (                    clk     : in    std_logic;

Hsync   : out   std_logic;
Vsync   : out   std_logic;
                          vgaBlue  : out   std_logic_vector (1 downto 0);
                          vgaGreen : out   std_logic_vector (2 downto 0);
                          vgaRed   : out   std_logic_vector (2 downto 0);

end VGA_TXT;
```

Сигналы Hsync, Vsync, vgaRed, vgaBlue, vgaGreen являются выходом видеоконтроллера, формирующего сигнал в формате VGA с частотой обновления 60 Гц в соответствии со стандартом VESA. К внешнему интерфейсу относятся линии приема и пробуждения дальномера, кнопка управления ОЭУН, элементы настройки режимов работы прибора, а также сигнал активизации затвора фотоаппарата.

Объект clkDllCtrl выполняет функции делителя частоты. На выходе – два импульсных сигнала – 25МГц и 50МГц (входная частота генератора).

Объект Synchro выдает импульсы вертикальной и горизонтальной развертки, а также номер соответствующего пиксела. Описан в соответствии с временными интервалами стандарта VGA.

Модуль ClkDllCtrl

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Library UNISIM;
use UNISIM.vcomponents.all;

entity clkDllCtrl is
  port(ckIn: in std_logic;--50MHz
       ckOut: out std_logic;--50MHz
       ckDivOut: out std_logic);--25MHz
end clkDllCtrl;
```

```
architecture Behavioral of clkDllCtrl is
signal bufck: std_logic;
signal ckDivOut_Int: std_logic := '0';
begin
 Divider:process (ckIn)
  begin
    if ckIn'event and ckIn = '1' then
          ckDivOut_Int <= not ckDivOut_Int;
    end if;
  end process;
 BUFG_inst : BUFG
       port map (
                O => ckDivOut, -- 1-bit output Clock buffer output
                I => ckDivOut_Int -- 1-bit input Clock buffer input
       );
       bufck <= ckIn;
       ckOut <= bufck;
end Behavioral;
```

Реализация контроллера

Рассмотрим HDL-реализацию контроллера видеопамяти.

Для описания поведения системы используются следующие сигналы:

– intVcnt – счетчик строк в развертке видеосигнала;

– intHcnt – счетчик столбцов в развертке видеосигнала;

– Vcnt(9 downto 0), Hcnt(9 downto 0) – их векторное представление;

– HS, VS – сигналы вертикальной и горизонтальной синхронизации (см. рисунки 3.20-3.21);

– write_adr – адрес, для записи кода символа в память VideoRAM;

– write_data – код символа, записываемый в память VideoRAM;

– VideoMemAdr – 12-битный адрес в памяти VideoRAM, по которому производится чтение кода символа в текущей точке;

– VideoMemByte – считанный байт кода символа;

– CharLineAdr – адрес текущего (т.е. выводимого в данный момент времени) символа в памяти CharRAM;

– CharLineByte – байт, содержащий данные о выводимой строке текущего символа;

Синхронизация МКО и микродисплея осуществляется компонентом Synchro, отвечающим за развертку видеосигнала. Счетчики intVcnt и intHcnt пробегают соответственно значения от 0 до (521-1) и от 0 до (800-1). Им соответствуют вектора Vcnt(9 downto 0) и Hcnt(9 downto 0). Компонент Synchro формирует сигналы вертикальной и горизонтальной развертки Hsync и Vsync в соответствии с требованиями стандарта VGA [31].

Для вывода графической информации разработан видеоконтроллер. Размеры знакоместа приняты 8x16 пикселов. Таким образом, по горизонтали символ будет кодироваться 3мя битами, а по вертикали – 4мя. При этом на экране можно разместить 480/16=30 строк по 640/8=80 символов в каждой.

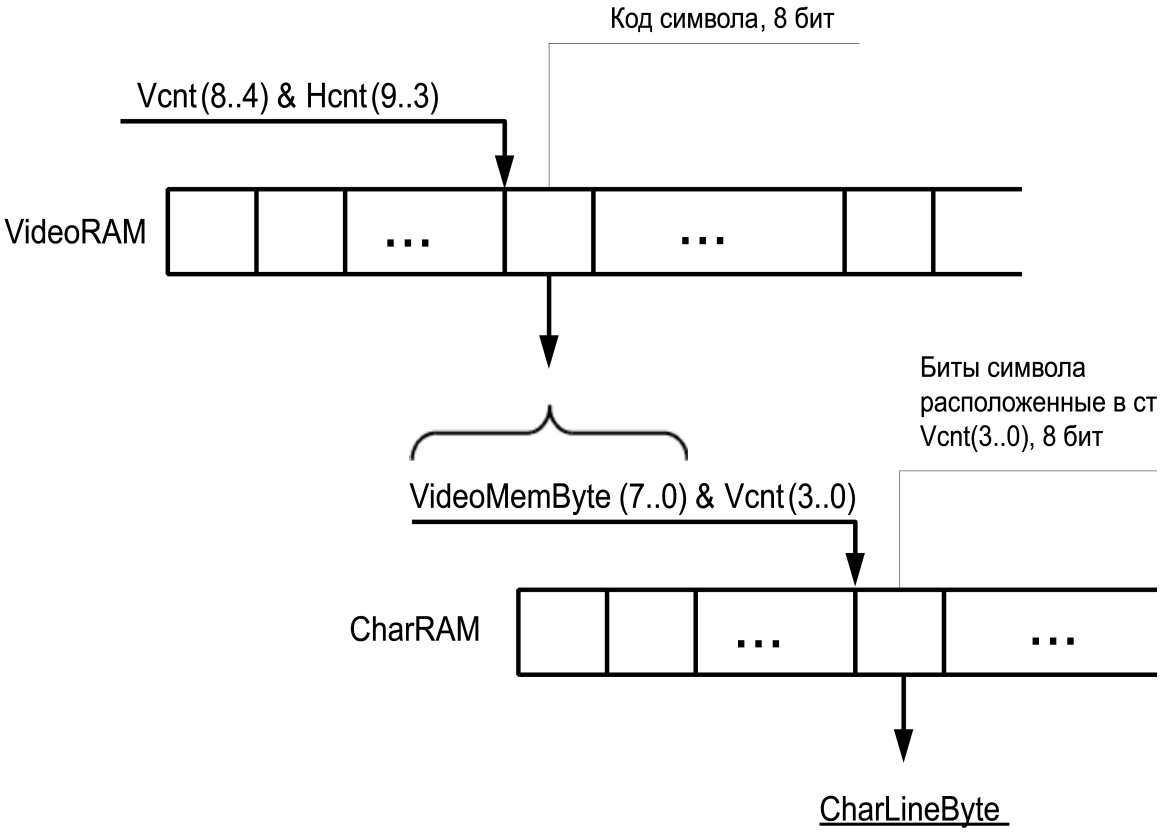
Организованы две области блочной памяти ПЛИС – одна для хранения символов алфавита (компонент Char_Gen_RAM), а другая – видеопамять – для хранения текущего состояния экрана, т.е. выводимых данных (компонент Video_RAM).

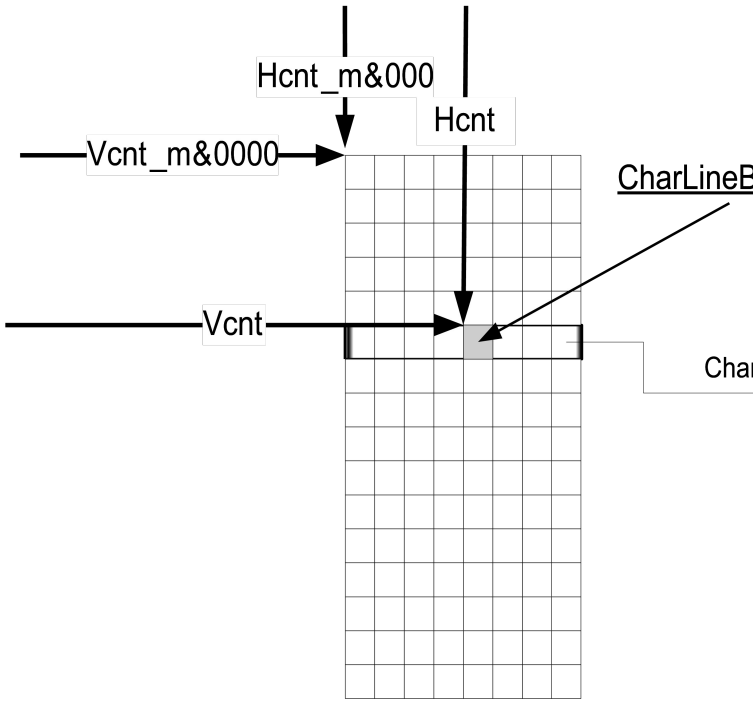
Для первой области в HDL-библиотеке было выбрано 2 блока памяти RAMB16_S4 (обозначены RAM0 и RAM1) – синхронная RAM с 12-битной шиной адреса и 4-битной шиной данных. Первые 8 бит адреса определяют код символа (возможно до 256 различных знаков), который предварительно читается из памяти Video_RAM, а остальные 4 бита

определяются младшими битами сигнала вертикальной синхронизации и обозначают номер строки в развертке символа.

Рисунок – Чтение из памяти значения пиксела, выводимого по адресу (Vcnt, Hcnt)

Рисунок поясняет организацию обращения в память. Для чтения значения выводимого пиксела необходимо обратиться в память хранения символов CharRAM. Адрес, по которому происходит чтение, формируется следующим образом: старшая часть адреса (8 бит)





содержит в себе код выводимого символа, а младшая – младшие биты вектора вертикальной развертки Vcnt. В результате возвращается строка пикселов CharLineByte, при проходе которой выводятся пикселы с адресами [Vcnt, Hcnt_m&Hcnt(2..0)], где младшие биты сигнала горизонтальной развертки пробегают значения байта CharLineByte.

Чтение из памяти описывается следующим образом:

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Library UNISIM;
use UNISIM.vcomponents.all;
entity DispCtrl is
  Port (clk25 : in std_logic;
              clk50: in std_logic;
      Hcnt: in std_logic_vector(9 downto 0);          -- horizontal counter
      Vcnt: in std_logic_vector(8 downto 0);          -- verical counter
      outRed  : out std_logic_vector(2 downto 0);  -- final color
      outGreen: out std_logic_vector(2 downto 0); -- outputs
      outBlue : out std_logic_vector(1 downto 0)
);
end DispCtrl;
architecture Behavioral of DispCtrl is
-- constants for Synchro module
  constant PAL:integer:=640;                --Pixels/Active Line (pixels)
  constant LAF:integer:=480;                --Lines/Active Frame (lines)
  constant for backgroud color /xxxRED_xxxGREEN_xxBLUE
  constant BGColor:std_logic_vector(7 downto 0):="00000000";
- constant for foregroud color /xxxRED_xxxGREEN_xxBLUE
  constant FGColor:std_logic_vector(7 downto 0):="11111111";

  signal VideoMemAdr: std_logic_vector(11 downto 0);
  signal VideoMemByte: std_logic_vector(7 downto 0);
  signal CharLineAdr: std_logic_vector(11 downto 0);
  signal CharLineByte: std_logic_vector(7 downto 0);
  signal CharLineByte_0: std_logic_vector(7 downto 0);
  signal CharLineBit: std_logic;
  signal VideoData: std_logic;
  signal intHcnt: integer range 0 to 800-1; --PLD-1 - horizontal counter
  signal intVcnt: integer range 0 to 521-1; -- LFD-1 - verical counter
  signal write_clk: std_logic;
  signal be: std_logic_vector(1 downto 0);
  signal write_enable: std_logic;
  signal write_en: std_logic_vector(3 downto 0);
  signal mem_write_adr: std_logic_vector(11 downto 0);
  signal write_adr: std_logic_vector(31 downto 0);
  signal write_data: std_logic_vector(31 downto 0);

--    COMPONENT Inst_mb02
--    PORT(
--            fpga_0_clk_1_sys_clk_pin : IN std_logic;
--            fpga_0_rst_1_sys_rst_pin : IN std_logic;
--            video_BRAM_WEN_pin : OUT std_logic_vector(0 to 3);
```

```
--              video_BRAM_Addr_pin : OUT std_logic_vector(0 to 31);
--              video_BRAM_Dout_pin : OUT std_logic_vector(0 to 31);
--              video_BRAM_Clk_pin : OUT std_logic;
--              );
--    END COMPONENT;

--attribute box_type : string;
--attribute box_type of Inst_mb02 : component is "user_black_box";

      COMPONENT CHAR_GEN_ROM
      PORT(
              clk50 : IN std_logic;
              address : IN std_logic_vector(11 downto 0);
              data : OUT std_logic_vector(7 downto 0)
              );
      END COMPONENT;
      COMPONENT VIDEO_RAM
      PORT(
              read_clk : IN std_logic;
              read_addr : IN std_logic_vector(11 downto 0);
              read_enable : IN std_logic;
              write_data : IN std_logic_vector(7 downto 0);
              write_addr : IN std_logic_vector(11 downto 0);
              write_clk : IN std_logic;
              write_enable : IN std_logic;
              read_data : OUT std_logic_vector(7 downto 0)
              );
      END COMPONENT;
begin
-- mapping the std_logic_vector ports to internal integers
 intHcnt <= conv_integer(Hcnt);
 intVcnt <= conv_integer(Vcnt);
 VideoData <= CharLineBit;
-- Out RG
 RGB_Out:process (clk25)
 begin
   if clk25'event and clk25 = '1' then
        if intHcnt < PAL and intVcnt < LAF then  -- in the active screen
                if VideoData = '0' then
                        outRed <= BGColor(7 downto 5);
                        outGreen <= BGColor(4 downto 2);
                        outBlue <= BGColor(1 downto 0);
                else
                        outRed <= FGColor(7 downto 5);
                        outGreen <= FGColor(4 downto 2);
                        outBlue <= FGColor(1 downto 0);
                end if;
              else
       outRed <= (others => '0');
       outGreen <= (others => '0');
       outBlue <= (others => '0');
     end if;
```

```
      end if;
    end process;
--    Instant_mb02 : Inst_mb02 PORT MAP(
--            fpga_0_clk_1_sys_clk_pin => clk50,
--            fpga_0_rst_1_sys_rst_pin => '0',
--            video_BRAM_WEN_pin =>  write_en,
--            video_BRAM_Addr_pin => write_adr,
--            video_BRAM_Dout_pin => write_data,
--            video_BRAM_Clk_pin => write_clk,
--    );

  VideoMemAdr<=Vcnt(8 downto 5)&'0'&Hcnt(9 downto 4)&'0';--!!!!!!!!!12bit!!!!!!!!!
      Inst_VIDEO_RAM: VIDEO_RAM PORT MAP(
            read_clk => clk50,
            read_data => VideoMemByte,--8-áèòíûé êîä ñèìâîëà, ïðî÷òåííûé ïî àäðåñó read_addr
            read_addr => VideoMemAdr,
            read_enable => '1',
            write_data => write_data(7 downto 0),--"00000001",--êîä ñèìâîëà, êîòîðûé ïèøåòñÿ
            write_addr => write_adr(13 downto 2),--àäðåñ ñèìâîëà â âèäåîïàìÿòè
            write_clk => clk50,--write_clk,--
            write_enable => write_en(0)
      );

  --Òåñò çàïèñè âèäåîïàìÿòè
      write_data(7 downto 0)<="00001001";
      write_adr(13 downto 2)<=write_adr(13 downto 2)+1 when clk50'event and clk50='1';
      write_en(0)<='1';


      Vcnt_2:process(intVcnt)--âìåñòî 2õ öèôð ïî âåðòèêàëè âûâîäèò îäíó.
      begin
            CASE Vcnt(4) IS
                  WHEN '0' => CharLineAdr<=VideoMemByte(7 downto 0)&'0'&Vcnt(3
downto 1);--âåðõ áóêâ ýòîé ñòðîêè
                  WHEN OTHERS => CharLineAdr<=VideoMemByte(7 downto
0)&'1'&Vcnt(3 downto 1);--0 - âûâîäèò âåðõ ýòîé ñòðîêè; 1 - íèç
            END CASE;
      end process;

      Inst_CHAR_GEN_ROM: CHAR_GEN_ROM PORT MAP(
            clk50 => clk50,
            address => CharLineAdr,
            data => CharLineByte
      );
    muxer: process(intHcnt, intVcnt)
    begin
      CASE Hcnt(3 downto 0) IS
                  WHEN "0000"  => CharLineBit<=CharLineByte(7);
                  WHEN "0001"  => CharLineBit<=CharLineByte(7);
                  WHEN "0010"  => CharLineBit<=CharLineByte(6);
                  WHEN "0011"  => CharLineBit<=CharLineByte(6);
                  WHEN "0100"  => CharLineBit<=CharLineByte(5);
```

```
                WHEN "0101"  => CharLineBit<=CharLineByte(5);
                WHEN "0110"  => CharLineBit<=CharLineByte(4);
                WHEN "0111"  => CharLineBit<=CharLineByte(4);
                WHEN "1000"  => CharLineBit<=CharLineByte(3);
                WHEN "1001"  => CharLineBit<=CharLineByte(3);
                WHEN "1010"  => CharLineBit<=CharLineByte(2);
                WHEN "1011"  => CharLineBit<=CharLineByte(2);
                WHEN "1100"  => CharLineBit<=CharLineByte(1);
                WHEN "1101"  => CharLineBit<=CharLineByte(1);
                WHEN "1110"  => CharLineBit<=CharLineByte(0);
                WHEN "1111"  => CharLineBit<=CharLineByte(0);
                WHEN OTHERS =>  CharLineBit <= '0';
        END CASE;
 end process;
end Behavioral;
```

Любой знак (см. рисунок) кодируется в памяти двумя последовательностями шестнадцатеричных чисел. Память организована двумя блоками памяти RAMB_S4, имеющими 4-битные шины данных. «Старшая» часть символа хранится в памяти CharRAM1, «младшая» – в CharRAM0.

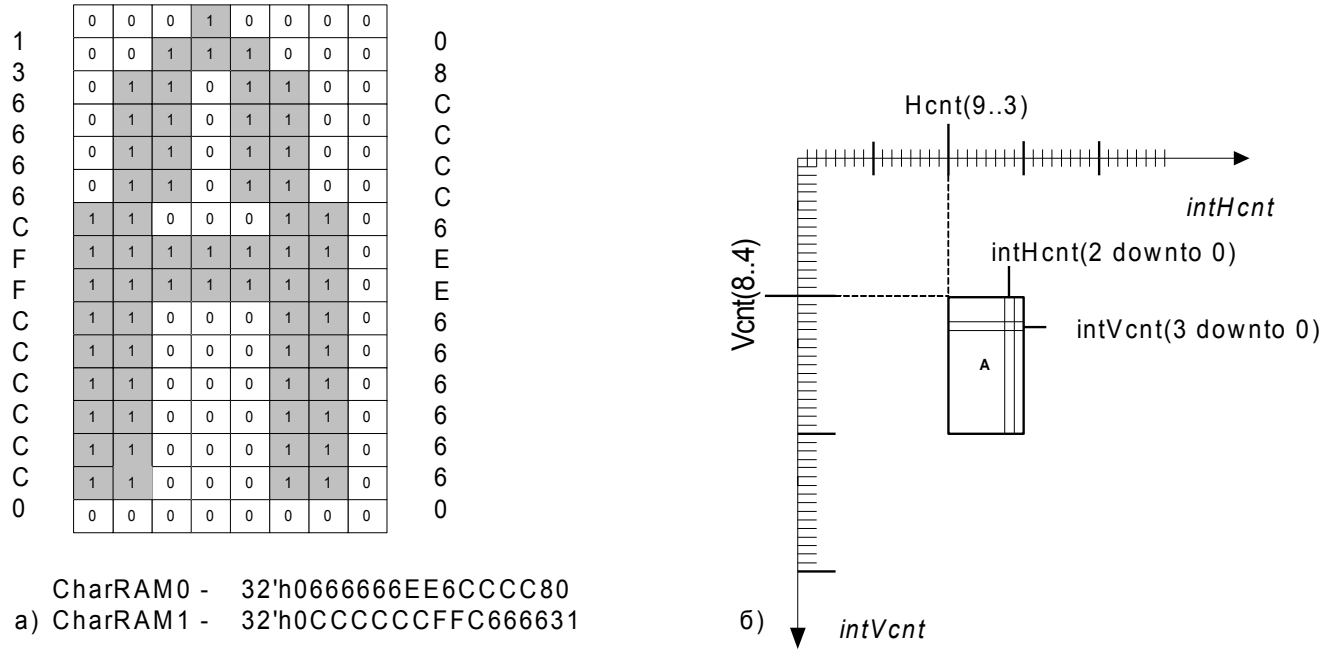


Рисунок - а) Кодирование символа; б) Адресация символа «А»

Область видеопамяти (VideoRAM) использует 2 блока памяти RAMB16_S4_S4 – синхронная двухпортовая RAM с 12-битной шиной адреса и 4-битными шинами данных. Один порт предназначен для записи символов из рабочей программы в память, а второй – для чтения символов из памяти и вывода на дисплей. Для адресации матрицы пикселов 640*480 требуется 10 бит по горизонтали и 9 бит по вертикали. С учетом размеров символа (8*16) для адресации 1 символа достаточно 7 и 5 (в сумме 12) старших битов (см. рисунок 3б). Чтение из памяти VideoRAM отражено на рисунке 3.11. Коды выводимых символов хранятся в памяти построчно.

Запись в память VideoRAM производится из основной программы системы.

Отрисовку символов, выводимых на VGA-дисплей, необходимо предварительно сформировать в памяти CharRAM. Память представлена двумя блоками памяти RAMB16_S4. Пример кодировки символов алфавита представлен ниже:

```
defparam CHARACTER_GEN_ROM0 .INIT_00 =
```

```
256'h0EF3333366333FFE0FF00008C63333FE0333333333333BF7000000000000000;
  defparam CHARACTER_GEN_ROM1 .INIT_00 =
256'h037600000000673107766631000066310000000000463100000000000000000;
  defparam CHARACTER_GEN_ROM0 .INIT_01 =
256'h00000088CC6333FF0EF33333FE0033FE0EF33333FE0000FF03333FF333333FF7;
  defparam CHARACTER_GEN_ROM1 .INIT_01 =
256'h06663311000006770376666677666673077000007766667700000 7FE66673100;
  defparam CHARACTER_GEN_ROM0 .INIT_02 =
256'hCC00CCCCCCCCCCCC0EF33333333333FE0EF3333FF33333FE0EF33337E73333FE;
  defparam CHARACTER_GEN_ROM1 .INIT_02 =
256'h0000000000000000037666666666667303760003766666730376666737666673;
  defparam CHARACTER_GEN_ROM0 .INIT_03 =
256'h0000000000000000CCC000000000CCC0CCC000000000000000000FFF000000;
  defparam CHARACTER_GEN_ROM1 .INIT_03 =
256'h0000000000000000011100000000011101110000000000000000000777000000;
```

и т.д.

Для автоматизации процесса отрисовки символов алфавита была разработана программа в среде Delphi, генерирующая файл с описанием кодировки алфавита по входным рисункам символов. В программе необходимо выделить ячейки, соответствующие битам символа, ввести код, или адрес, символа в памяти CharRAM и нажать кнопку генерации:

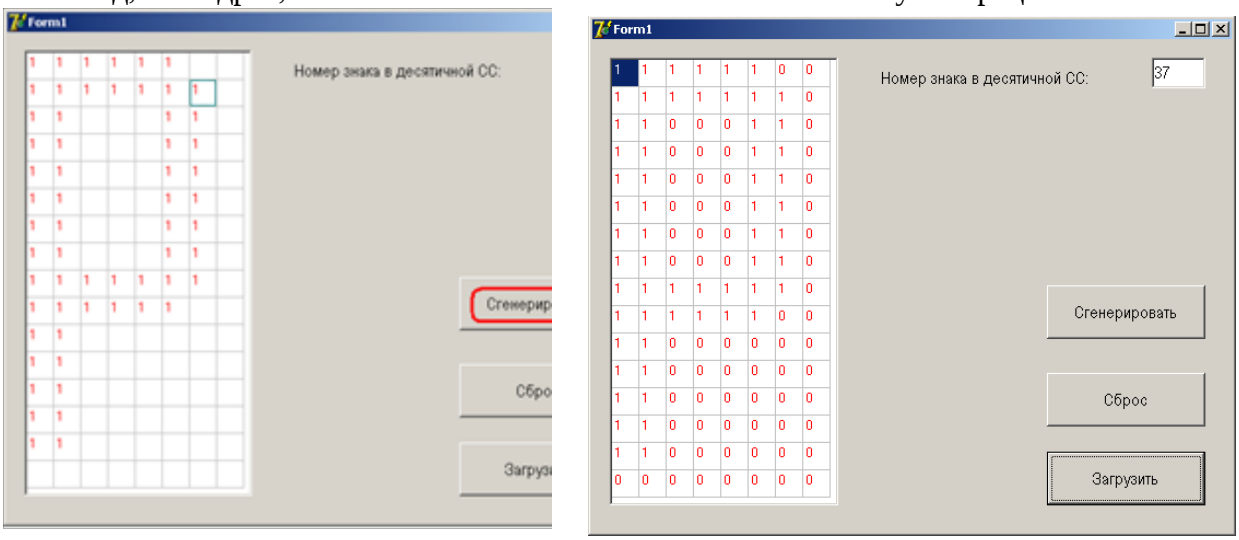


Рисунок – Пример генерации буквы «Р»

Архитектура проекта

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
--use ieee.std_logic_unsigned.ALL;
use ieee.std_logic_signed.ALL;
entity VGA_TXT is
  port (       clk     : in    std_logic;
                           Hsync    : out   std_logic;
                           vgaBlue  : out   std_logic_vector (1 downto 0);
                           vgaGreen : out   std_logic_vector (2 downto 0);
                           vgaRed   : out   std_logic_vector (2 downto 0);
                           Vsync    : out   std_logic
);
end VGA_TXT;
```

```
architecture BEHAVIORAL of VGA_TXT is
  signal clk25   : std_logic;
  signal clk50  : std_logic;
  signal Hcnt   : std_logic_vector (9 downto 0);
  signal Vcnt   : std_logic_vector (8 downto 0);

  component clkDllCtrl
    port ( ckIn     : in    std_logic;
           ckOut    : inout std_logic;
           ckDivOut : inout std_logic);
  end component;

      COMPONENT DispCtrl
      PORT(
              clk25 : IN std_logic;
              clk50 : IN std_logic;
              Hcnt : IN std_logic_vector(9 downto 0);
              Vcnt : IN std_logic_vector(8 downto 0);
              outRed : OUT std_logic_vector(2 downto 0);
              outGreen : OUT std_logic_vector(2 downto 0);
              outBlue : OUT std_logic_vector(1 downto 0)
              );
      END COMPONENT;
 component Synchro
    port ( clk25 : in    std_logic;
           HS      : out   std_logic;
           VS      : out   std_logic;
           Hcnt    : out   std_logic_vector (9 downto 0);
           Vcnt    : out   std_logic_vector (8 downto 0));
  end component;

begin

  CLK_MGMT : clkDllCtrl
    port map (ckIn=>clk,
              ckDivOut=>clk25,
              ckOut=>clk50);
  RGB_CNTL : DispCtrl
    port map (
                                clk25 => clk25,
                                clk50 => clk50,
                                Hcnt => Hcnt,
                                Vcnt => Vcnt,
                                outRed => vgaRed,
                                outGreen => vgaGreen,
                                outBlue => vgaBlue
                                );

  SYNC_CNTL : Synchro
    port map (clk25=>clk25,
              Hcnt(9 downto 0)=>Hcnt(9 downto 0),
              HS=>HSync,
```

```
            Vcnt(8 downto 0)=>Vcnt(8 downto 0),
             VS=>VSync);
end BEHAVIORAL;
```