Character Display Experiment

Technical Email: alinx@aithtech.com Sales Email: rachel.zhou@alinx.com

1 Experiment Introduction

In the HDMI test experiment, the HDMI display principle and display mode are explained. This experiment introduces how to use FPGA to realize character display. Through this experiment, you can get a deeper understanding of HDMI display mode.

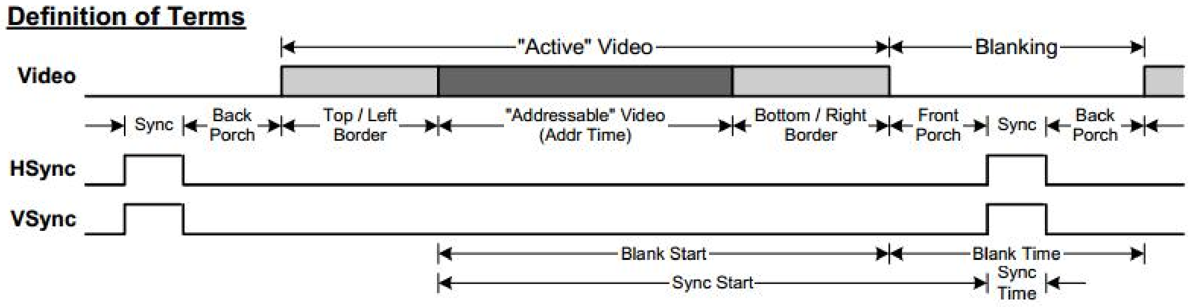
2 Experiment Principle

The experiment converts the characters into octal coe files through the character conversion tool and stores them in the ROM IP core of the single port, and then reads the converted data from the ROM and displays it on the HDMI.

3 Programming

For the timing of HDMI, a counter is used for line synchronization and field synchronization. The line synchronization counter is used to generate line synchronization and line effective pixels, and the field synchronization counter is used to generate field synchronization and field effective pixels.

The "timing_gen_xy" module defines two counters "x_cnt" and "y_cnt" according to the HDMI timing standard and generates "x" coordinates and "y" coordinates of the HDMI display by these two counters. The program uses "vs_edge" and "de_falling" to indicate the field sync start signal and the data valid end signal, respectively. The principle is as shown below:



Signal Name	Direction	Description
rst_n	in	Asynchronous reset input, low reset
clk	in	External clock input
i_hs	in	Line sync signal
i_vs	in	Field sync signal

i_de	in	Data valid signal
i_data	in	Color_bar data
o_hs	out	Output line sync signal
o_vs	out	Output field sync signal
o_de	out	Output data valid signal
o_data	out	Output Data
x	out	Generate X coordinates
y	out	Generate Y coordinate

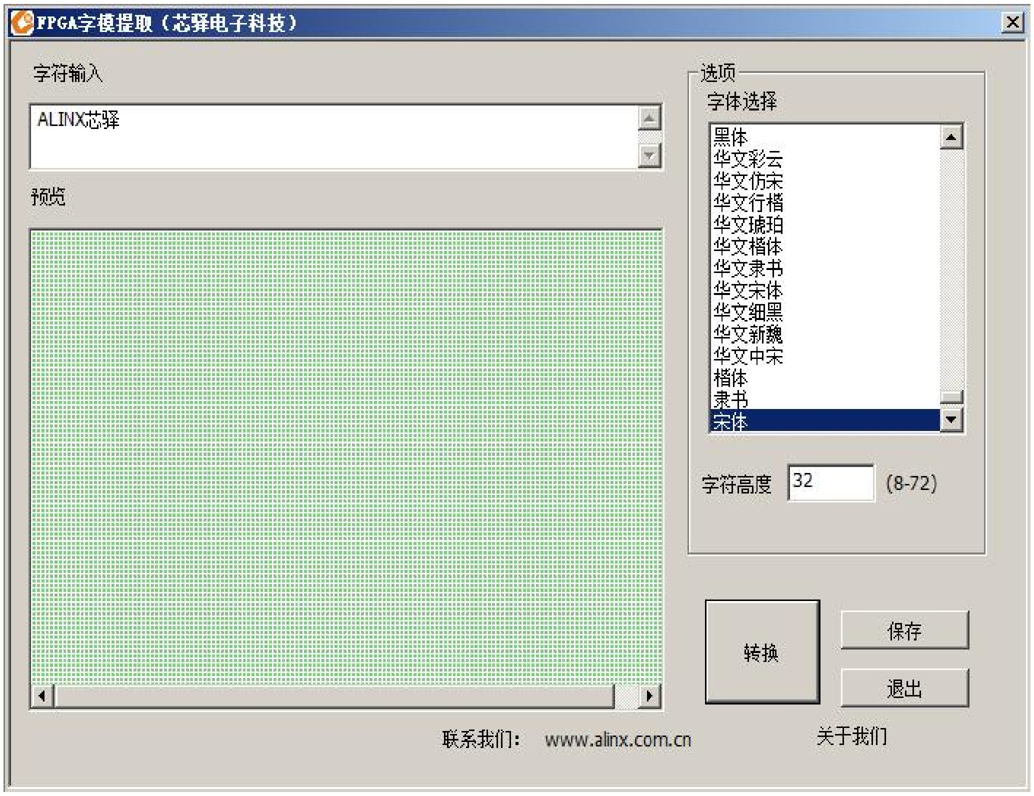
timing_gen_xy Module Port

The following describes how to store the ROMIP of text information. First, you need to generate a .coe file that can be recognized by XILINX FPGA.

1. Find the "FPGA Font Extraction" tool under the Software Tools and Drivers folder.

FPGA字模提取.exe

2. Double-click the .exe file to open the tool



FPGA字模提取（芯驿电子科技）

字符输入

ALINX芯驿

预览

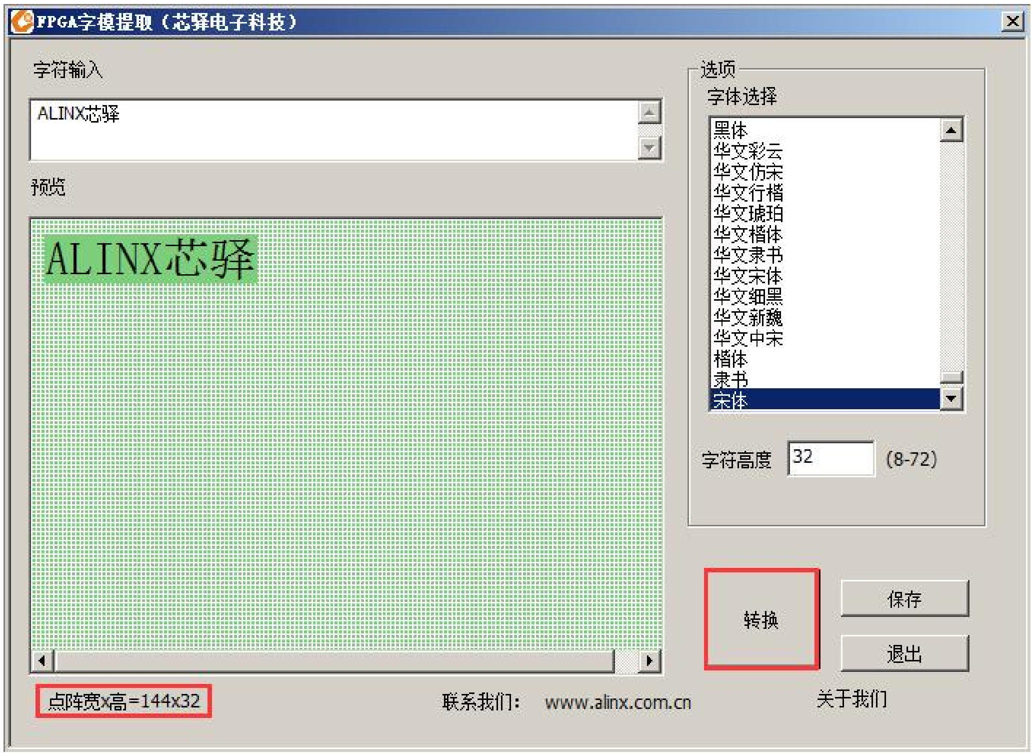
选项

字体选择

字符高度 32 (8-72)

联系我们： www.alinx.com.cn

3. Enter the characters you want to display in the “Character Input box” of the extraction tool. The font and character height can be customized. After the setting is completed, click the “Convert” button. In the lower left corner of the interface, you can see the size of the converted character dot matrix. The width and height of the dot matrix are needed in the program.



FPGA字模提取（芯驿电子科技）

字符输入: ALINX芯驿

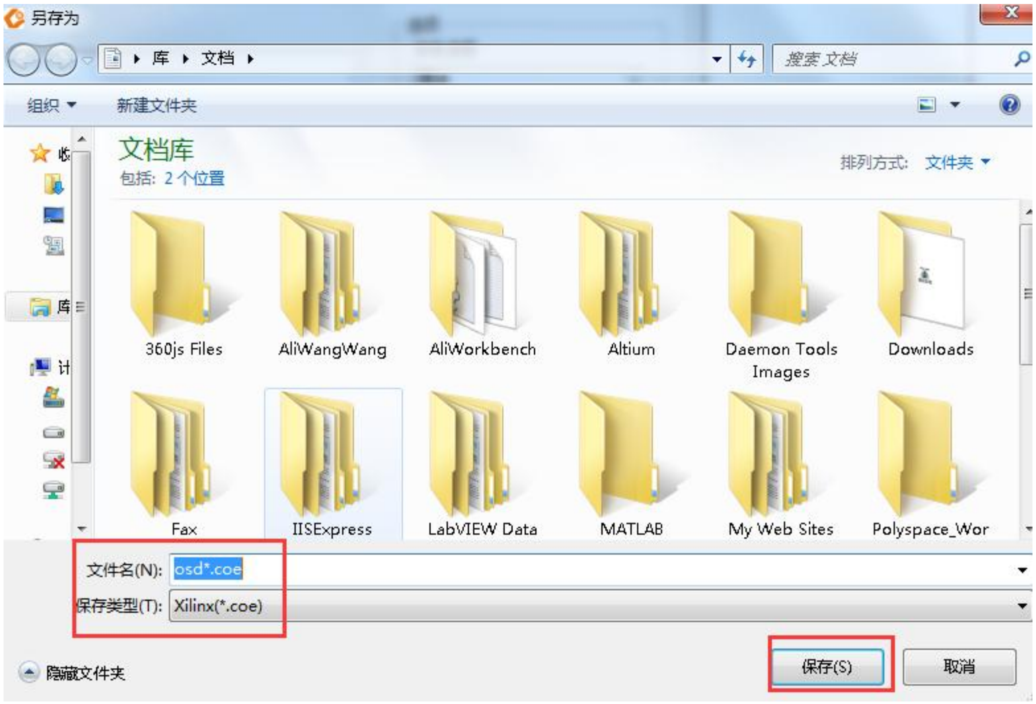
字体选择: 宋体

字符高度 32 (8-72)

点阵宽x高=144x32

联系我们: www.alinx.com.cn

4. Click the "Save" button to save the file to the source file directory of this routine. Note that you should select Xilinx (*.coe) under the save type and click the "Save" button.



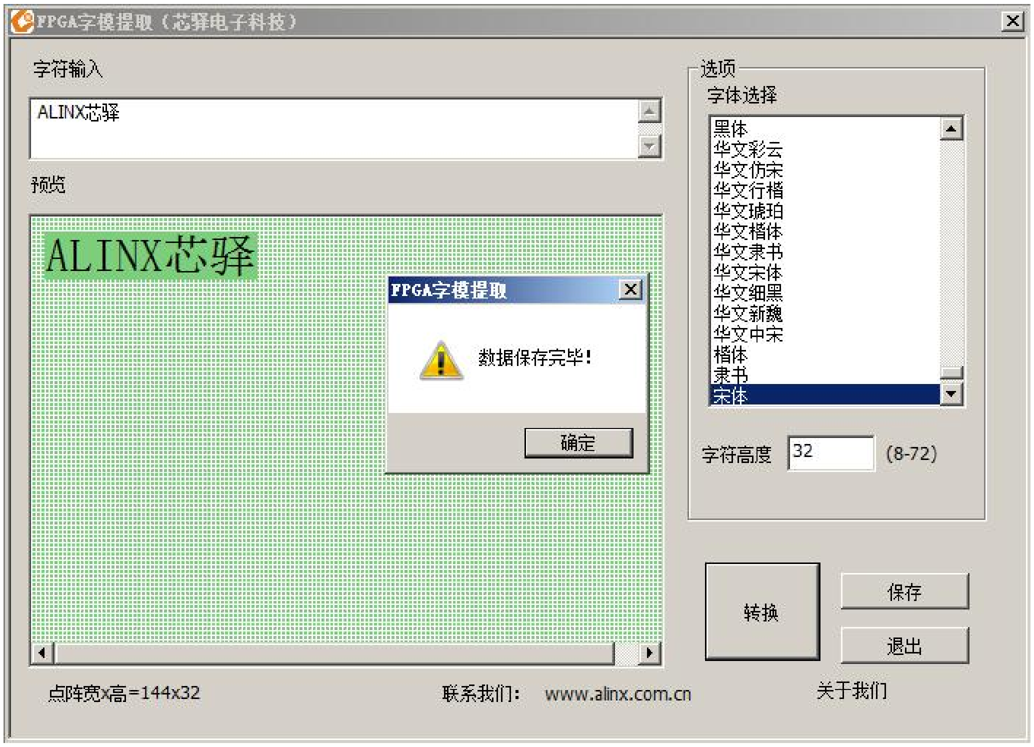
另存为

库 ▸ 文档 ▸

文件名(N): osd*.coe

保存类型(T): Xilinx(*.coe)

5. Back to the character extraction tool interface, the following dialog box appears to indicate that the save is complete, click OK, and exit.



FPGA字模提取（芯驿电子科技）

字符输入: ALINX芯驿

FPGA字模提取: 数据保存完毕！

字体选择: 宋体

字符高度 32 (8-72)

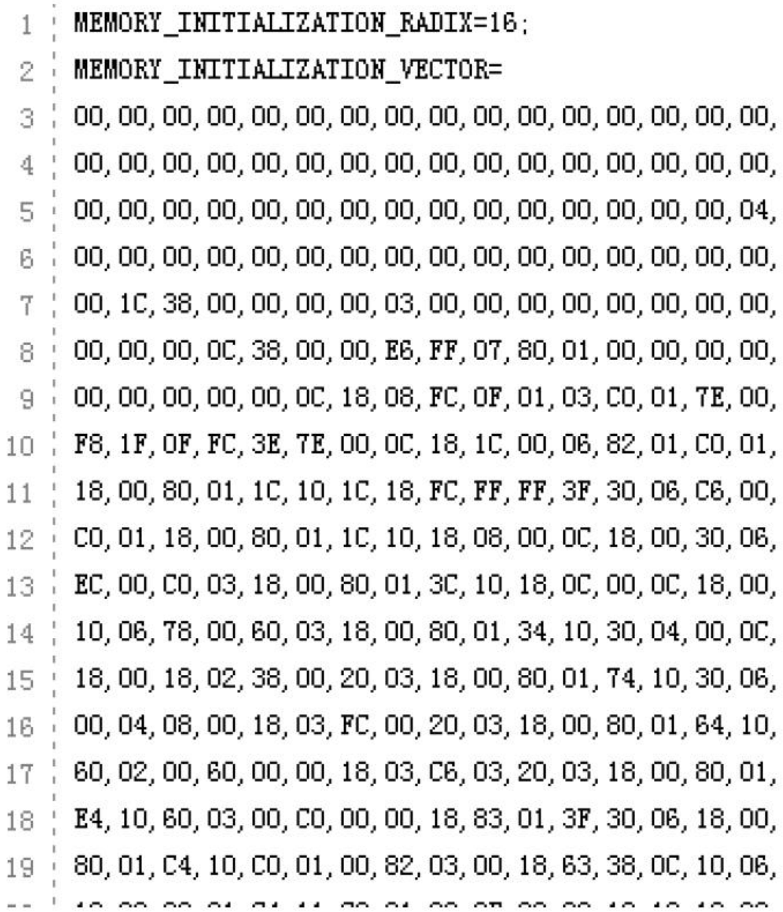
点阵宽x高=144x32

联系我们： www.alinx.com.cn

Returning to the "osd_display" module program, the display size of the character is set according to the width and height of the character dot matrix, and the character can be displayed at any position on the screen through the X and Y coordinates generated by the "timing_gen_xy" module:

```
always@(posedge pclk)

begin

        if(pos_y >= 12'd9 && pos_y <= 12'd9 + OSD_HEGIHT - 12'd1 && pos_x >= 12'd9 && pos_x  <= 12'd9 +
OSD_WIDTH - 12'd1)

                region_active <= 1'b1;

        else

                region_active <= 1'b0;

end
```

In addition, it has been described in the above way that the characters are converted into a dot matrix and stored in the “Rom IP” core. When the generated “.coe” file is found, it can be seen as follows:

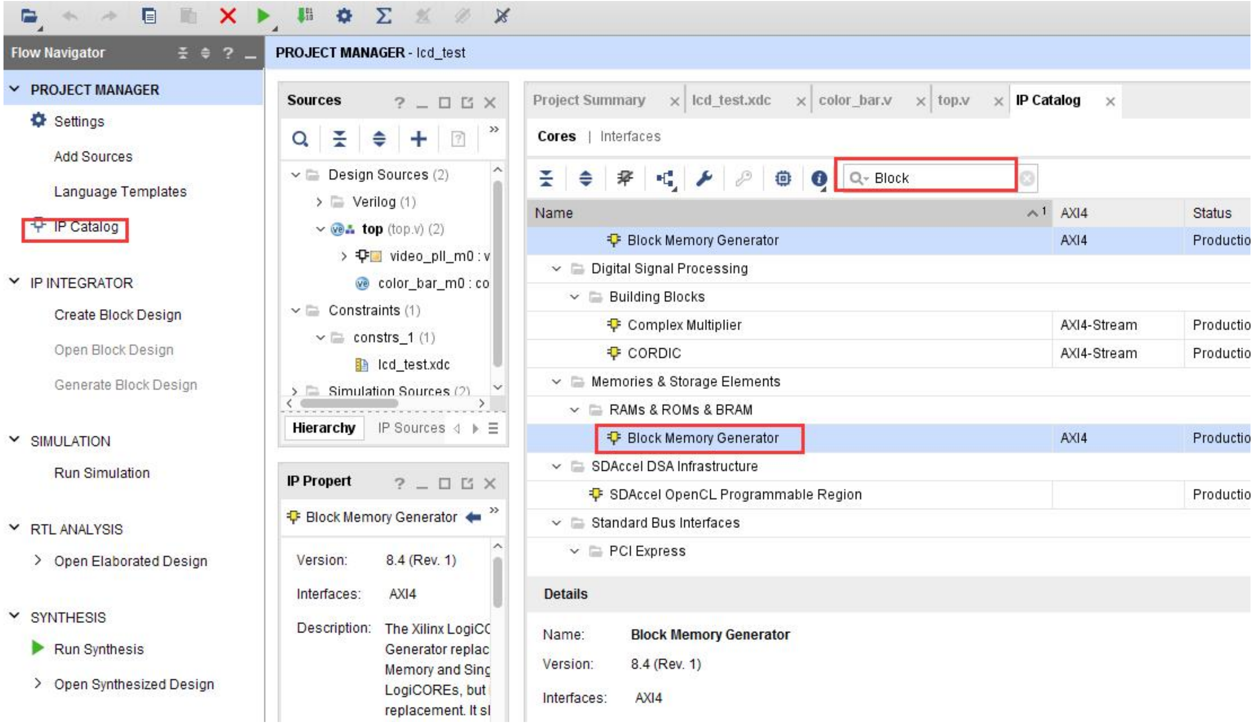


```
MEMORY_INITIALIZATION_RADIX=16;
MEMORY_INITIALIZATION_VECTOR=
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,04,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,1C,38,00,00,00,00,03,00,00,00,00,00,00,00,00,
00,00,00,0C,38,00,00,E6,FF,07,80,01,00,00,00,00,
00,00,00,00,00,0C,18,08,FC,0F,01,03,C0,01,7E,00,
F8,1F,0F,FC,3E,7E,00,0C,18,1C,00,06,82,01,C0,01,
18,00,80,01,1C,10,1C,18,FC,FF,FF,3F,30,06,C6,00,
C0,01,18,00,80,01,1C,10,18,08,00,0C,18,00,30,06,
EC,00,C0,03,18,00,80,01,3C,10,18,0C,00,0C,18,00,
10,06,78,00,60,03,18,00,80,01,34,10,30,04,00,0C,
18,00,18,02,38,00,20,03,18,00,80,01,74,10,30,06,
00,04,08,00,18,03,FC,00,20,03,18,00,80,01,64,10,
60,02,00,60,00,00,18,03,C6,03,20,03,18,00,80,01,
E4,10,60,03,00,C0,00,00,18,83,01,3F,30,06,18,00,
80,01,C4,10,C0,01,00,82,03,00,18,63,38,0C,10,06,
```

The converted characters have become 8bit value files, so the settings are as follows when reading the mif file from the Rom IP core:

```
always@(posedge pclk)

begin

        if(region_active_d0 == 1'b1)

                if(q[osd_x[2:0]] == 1'b1)

                        v_data <= 24'hff0000;

                else

                        v_data <= pos_data;

        else

                v_data <= pos_data;

end
```

The process of calling a single-port Rom IP core is the same as calling other IP cores. Open the "IP Catalog" and search for "Block":

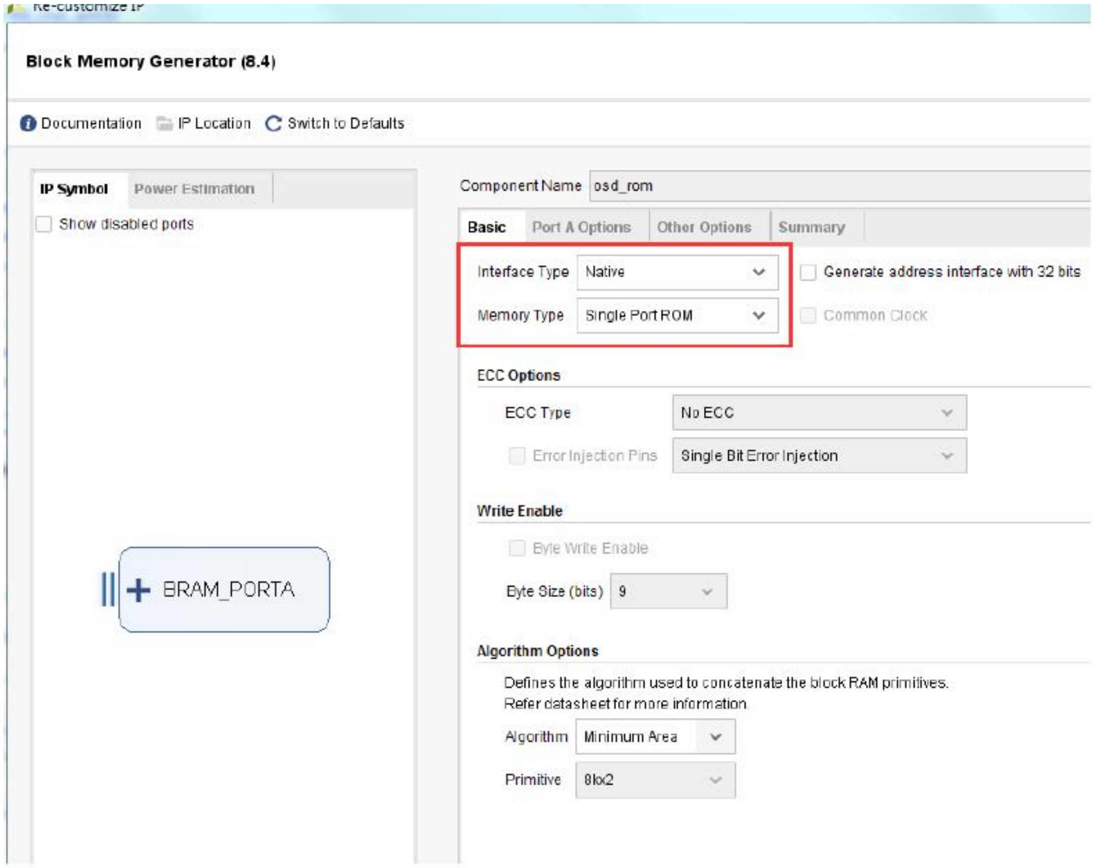


Name	AXI4	Status
Block Memory Generator	AXI4	Productio
Digital Signal Processing		
Building Blocks		
Complex Multiplier	AXI4-Stream	Productio
CORDIC	AXI4-Stream	Productio
Memories & Storage Elements		
RAMs & ROMs & BRAM		
Block Memory Generator	AXI4	Productio
SDAccel DSA Infrastructure		
SDAccel OpenCL Programmable Region		Productio
Standard Bus Interfaces		
PCI Express		

Version: 8.4 (Rev. 1)
Interfaces: AXI4
Description: The Xilinx LogiCC Generator replac Memory and Sing LogiCOREs, but replacement. It sl

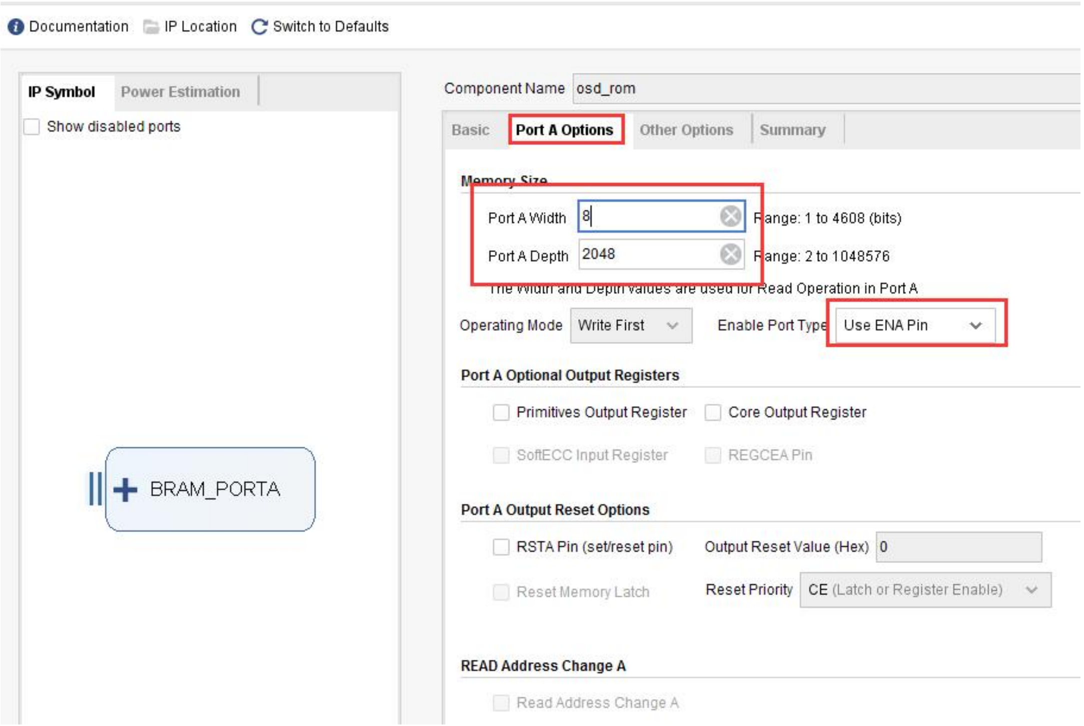
Name: Block Memory Generator
Version: 8.4 (Rev. 1)
Interfaces: AXI4

Then set it to ROM in the Basic column as shown in the figure below:



Component Name: osd_rom
Interface Type: Native
Memory Type: Single Port ROM
ECC Type: No ECC
Error Injection Pins: Single Bit Error Injection
Byte Size (bits): 9
Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.
Algorithm: Minimum Area
Primitive: 8kx2

The settings in the PortA Options column are as follows:



Documentation IP Location Switch to Defaults

IP Symbol | Power Estimation

□ Show disabled ports

Component Name osd_rom

Basic | Port A Options | Other Options | Summary

Memory Size

Port A Width 8 Range: 1 to 4608 (bits)

Port A Depth 2048 Range: 2 to 1048576

The Width and Depth values are used for Read Operation in Port A

Operating Mode Write First Enable Port Type Use ENA Pin

Port A Optional Output Registers

□ Primitives Output Register □ Core Output Register

□ SoftECC Input Register □ REGCEA Pin

Port A Output Reset Options

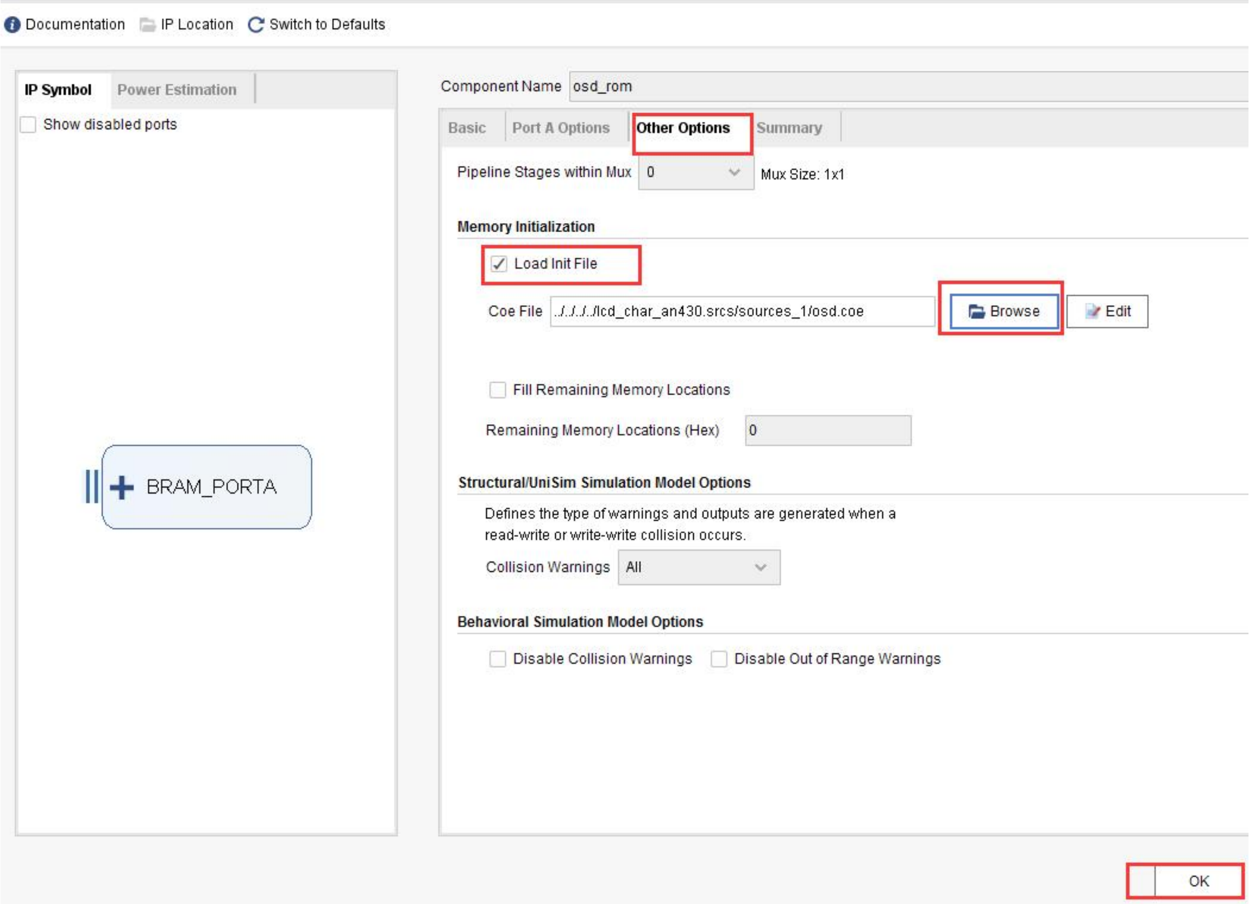
□ RSTA Pin (set/reset pin) Output Reset Value (Hex) 0

□ Reset Memory Latch Reset Priority CE (Latch or Register Enable)

READ Address Change A

□ Read Address Change A

Add the osd.coe file as shown below, and click the "OK" button when finished:



Documentation IP Location Switch to Defaults

IP Symbol | Power Estimation

□ Show disabled ports

Component Name osd_rom

Basic | Port A Options | Other Options | Summary

Pipeline Stages within Mux 0 Mux Size: 1x1

Memory Initialization

☑ Load Init File

Coe File ../../../lcd_char_an430.srcs/sources_1/osd.coe Browse Edit

□ Fill Remaining Memory Locations

Remaining Memory Locations (Hex) 0

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings All

Behavioral Simulation Model Options

□ Disable Collision Warnings □ Disable Out of Range Warnings

OK

The Rom IP core is instantiated in the "osd_display" module as follows:

```
osd_rom osd_rom_m0

(

        .address(osd_ram_addr[15:3]),

        .clock(pclk),

        .q(q)

);
```

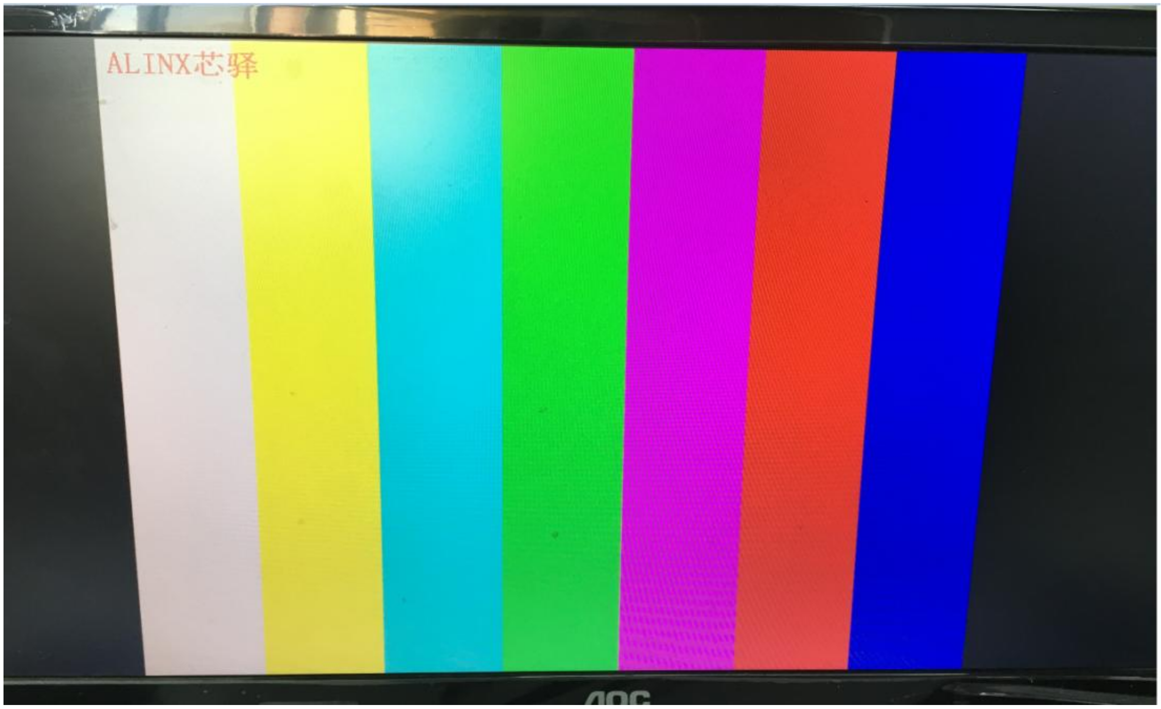
Signal Name	Direction	Description
rst_n	in	Asynchronous reset input, low reset
pclk	in	External clock input
i_hs	in	Line sync signal
i_vs	in	Field sync signal
i_de	in	Data valid signal
i_data	in	Color_bar data
o_hs	out	Output line sync signal
o_vs	out	Output field sync signal
o_de	out	Output data valid signal
o_data	out	Output Data

osd_display Module Port

Several resolution timing parameters are preset in the program, including 2 LCD screens, to prepare for subsequent LCD verification tests.

4 Experiment Result

Connect the FPGA development board and display. Refer to the "HDMI Test Experiment" tutorial for the connection method. Note that the connectors of the development board should not be hot-swapped. Download the test program and you can see the characters on the display with the color bar as the background. The development board is used as the HDMI output device and can only be displayed through an HDMI display device. Do not attempt to display it through the HDMI interface of the notebook computer, because the notebook is also an output device.


ALINX芯驿
AOC