数字电路与数字系统实验

实验七

存储器

计算机科学与技术系

191220008 陈南曈

924690736@qq.com

2020.10.16

一、实验目的

存储器（Memory）是电子设备中的记忆器件，用来存放程序和数据。电子设备中全部信息，包括输入的原始数据、程序、中间运行结果和最终运行结果 都保存在存储器中。

本实验的目的是了解 FPGA 的片上存储器的特性，分析存储器的工作时序和结构，并学习如何设计存储器。

（一）存储器

请在一个工程中完成如下两个存储器。两个存储器的大小均为 16×8，即每个存储器共有 16 个存储单元，每个存储单元都是 8 位的，均可以进行读写。

RAM1：采用下面的方式进行初始化，输出端有输出缓存，输出地址有效后，等时钟信号的上升沿到来时才输出数据。

```
initial
begin
    $readmemh("D:/digital_logic/mem1.txt", ram, 0, 15);
end
```

初始化数值为

```
@0 00
@1 01
@2 02
@3 03
@4 04
@5 05
@6 06
@7 07
@8 08
@9 09
@a 0a
@b 0b
@c 0c
@d 0d
@e 0e
@f 0f
```

RAM2：利用 IP 核设计一个双口存储器，利用.mif 文件进行初始化，十六个单元的初始化值分别为：0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6,0xf7,0xf8,0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff。

此两个物理上完全不同的存储器共用时钟、读写地址和写使能信号，当写使能有效时，在时钟信号的有效沿写入数据；当写使能信号无效时，在时钟信号的有效沿输出数据。适当选择时钟信号和写使能信号，以能够分别对此两个存储器进行读写。

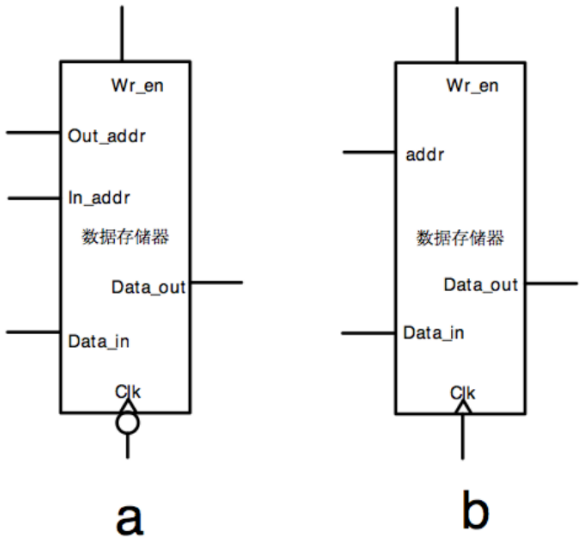
请合理使用 FPGA 开发板的输入/输出资源，完成此两个存储器的设计。由于开发板上输入数量不够，写入时可以只写入 2 位数据。

二、实验原理（知识背景）

1、存储器的结构

存储器是一组存储单元，用于在计算机中存储二进制的数据，如图所示。存储器的端口包括输入端、输出端和控制端口。输入端口包括：读/写地址端口、数据输入端口等；输出端口一般指的是数据输出端口；控制端口包括时钟端和读/写控制端口。

存储器的工作过程如下：



写数据：在时钟（clk）有效沿（上升或下降沿），如果写使能（Wren，也可以没有使能端）有效，则读取输入总线（Data_in）上的数据，将其存储到输入地址线（In_addr）所指的存储单元中。

读数据：存储器的输出可以受时钟和使能端的控制，也可以不受时钟和使能端的控制。如果输出受时钟的控制，则在时钟有效沿，将输出地址所指示的单元中的数据，输出到输出总线上（Data_out）；如果不受时钟的控制，则只要输出地址有效，就立即将此地址所指的单元中的数据送到输出总线上。

- 对于存储器，其读写时序非常重要，也是实践中容易出错的地方。读取数据时在哪个时间点数据有效，写入数据过多久可以读取这些都要在设计时反复检查和验证。

FPGA 存储器的工作模式有很多，如：真双口 RAM、简单双口 RAM、单口 RAM、ROM 或者 FIFO 缓存等。常见的模式请参照下表。

存储器模式	说明
单口存储器	某一时刻，只读或者只写
简单双口存储器模式	简单双口模式支持同时读写（一读一写）
混合宽度的简单双口存储器模式	读写使用不同的数据宽度的简单双口模式
真双口储存器模式	真双口模式支持任何组合的双口操作：两个读口、两个写口和两个不同时钟频率下的一读口一写口
混合宽度的真双口存储器模式	读写使用不同的数据宽度的真双口模式
ROM	工作于 ROM 模式，ROM 中的内容已经初始化
FIFO 缓冲器	可以实现单时钟或双时钟的 FIFO

2、存储器的实现

Cyclone V 系列 FPGA 内部含有两种嵌入式存储器块：

10Kb 的 M10K 存储器块——这是专用存储器资源块。M10K 存储器块是理想的大存储器阵列，并提供大量独立端口。

64 位存储器逻辑阵列（MLABs）——是一种嵌入式存储器阵列是由双用途逻辑阵列块配置而来的。MLAB 是理想的宽而浅的存储阵列。MLAB 是经过优化的可以用于实现数字信号处理（DSP）应用中的移位寄存器、宽浅 FIFO 缓存和滤波延迟线。每个 MLAB 都由 10 个自适应逻辑块（ALM）组成。在 Cyclone V 系列器件中，你可以将这些 ALM 可配置成 10 个 32×2 模块，从而每 个 MLAB 可以实现一个 32×20 简单双端口 SRAM 模块。

Cyclone V 系列 FPGA 嵌入式存储器资源如图所示，我们可以对应比较一下 DE10-standard 开发平台上配置的 Cyclone V SX C6 的存储器资源。

Variant	Member Code	M10K		MLAB		Total RAM Bit (Kb)
		Block	RAM Bit (Kb)	Block	RAM Bit (Kb)	
Cyclone V GX	C3	135	1,350	291	182	1,532
	C4	250	2,500	678	424	2,924
	C5	446	4,460	678	424	4,884
	C7	686	6,860	1338	836	7,696
	C9	1,220	12,200	2748	1,717	13,917
Cyclone V GT	D5	446	4,460	679	424	4,884
	D7	686	6,860	1338	836	7,696
	D9	1,220	12,200	2748	1,717	13,917
Cyclone V SE	A2	140	1,400	221	138	1,538
	A4	270	2,700	370	231	2,460
	A5	397	3,970	768	480	4,450
	A6	557	5,570	994	621	5,761
Cyclone V SX	C2	140	1,400	221	138	1,538
	C4	270	2,700	370	231	2,460
	C5	397	3,970	768	480	4,450
	C6	557	5,570	994	621	5,761
Cyclone V ST	D5	397	3,970	768	480	4,450
	D6	557	5,570	994	621	5,761

Quartus 会根据用户存储器设计的速度与大小，来自动选择硬件实现时使用的存储器模块的数量与配置。例如，为提供设计性能，Quarrus 可能将可以由 1 块 RAM 实现的存储器设计扩展为由多块 RAM 来实现。

三、实验环境/器材等

1) 软件环境：

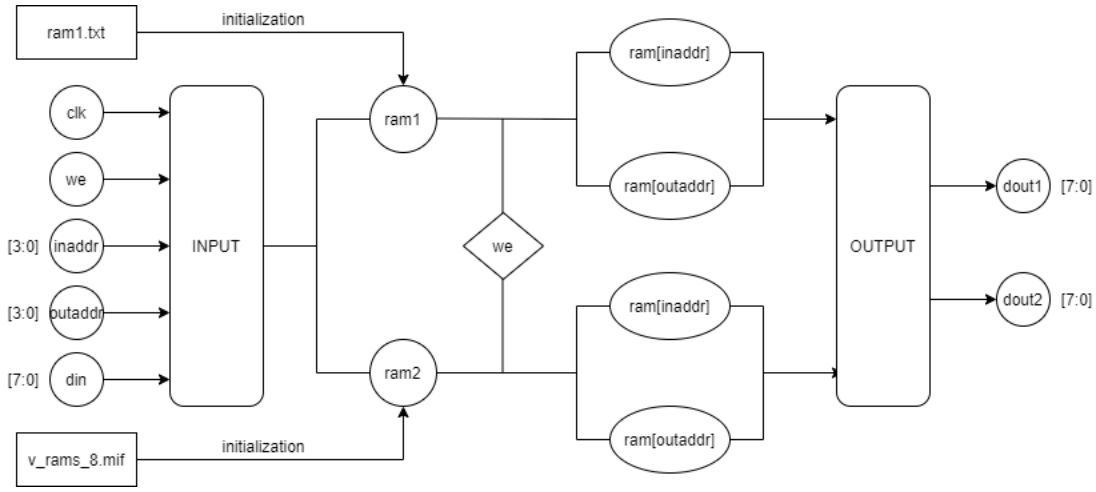
Quartus (Quartus Prime 17.1) Lite Edition

2) 硬件环境：

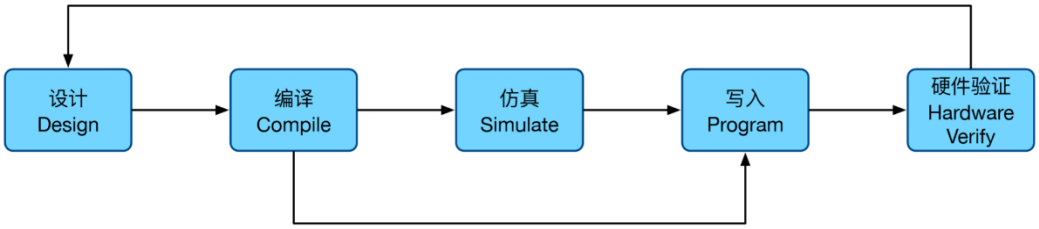
DE10-Standard 开发板
FPGA 部分：
Intel Cyclone V SE 5CSXFC6D6 F31C6N
- 110K 逻辑单元
- 5,761Kbit RAM

HPS 部分：
Dual-core ARM Cortex A9
- 925MHz
- 1GB DDR

四、程序代码或流程图



五、实验步骤/过程



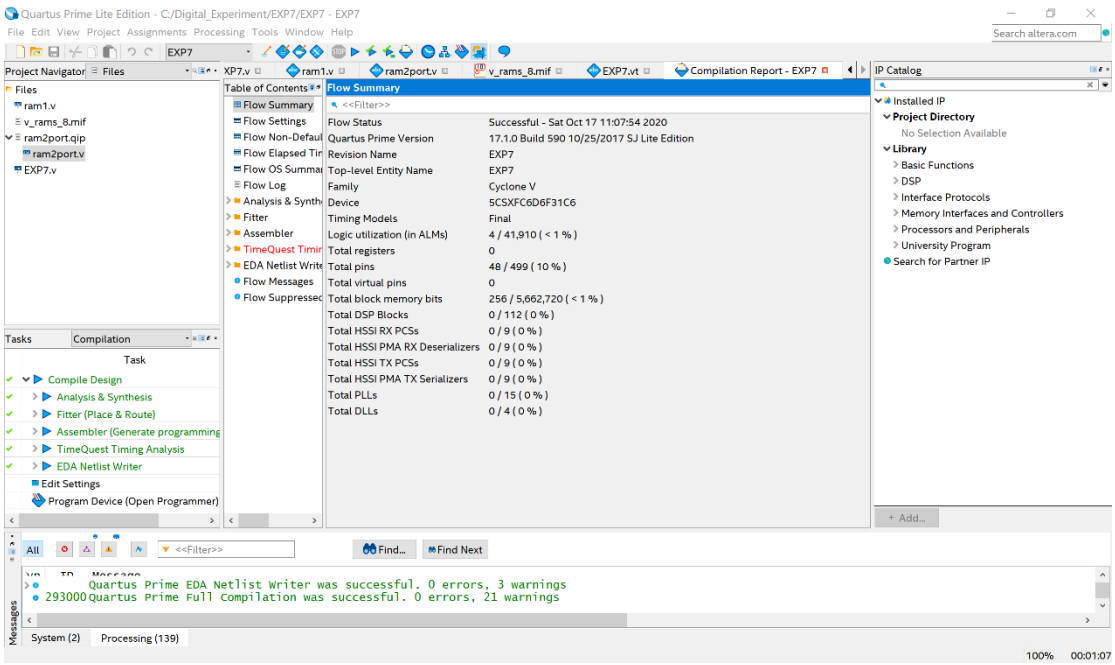
设计：

```
module EXP7(clk, we, inaddr, outaddr, din, dout1, dout2, HEX0, HEX1);
    input clk;
    input we;
    input [3:0] inaddr;
    input [3:0] outaddr;
    input [7:0] din;

    output [7:0] dout1;
    output [7:0] dout2;
    output reg [6:0] HEX0;
    output reg [6:0] HEX1;

    ram1 myram1(clk, we, inaddr, outaddr, din, dout1);
    ram2port myram2(.clock(clk), .data(din), .wraddress(inaddr), .rdaddress(outaddr),

always @(*)
begin
    case (dout1[1:0])
        2'b00: HEX0 = 64;
        2'b01: HEX0 = 121;
        2'b10: HEX0 = 36;
        2'b11: HEX0 = 48;
        default: HEX0 = 127;
    endcase

    case (dout2[1:0])
        2'b00: HEX1 = 64;
        2'b01: HEX1 = 121;
        2'b10: HEX1 = 36;
        2'b11: HEX1 = 48;
        default: HEX1 = 127;
    endcase
end

endmodule


module ram1(clk, we, inaddr, outaddr, din, dout);
    input clk;
    input we;
    input [3:0] inaddr;
    input [3:0] outaddr;
    input [7:0] din;

    output reg [7:0] dout;

    reg [7:0] ram1 [15:0];

initial
begin
    $readmemh("C:/Digital_Experiment/EXP7/ram1.txt", ram1, 0, 15);
end

always @(posedge clk)
begin
    if (we)
        ram1[inaddr] <= din;
    else
        dout <= ram1[outaddr];
end

endmodule
```

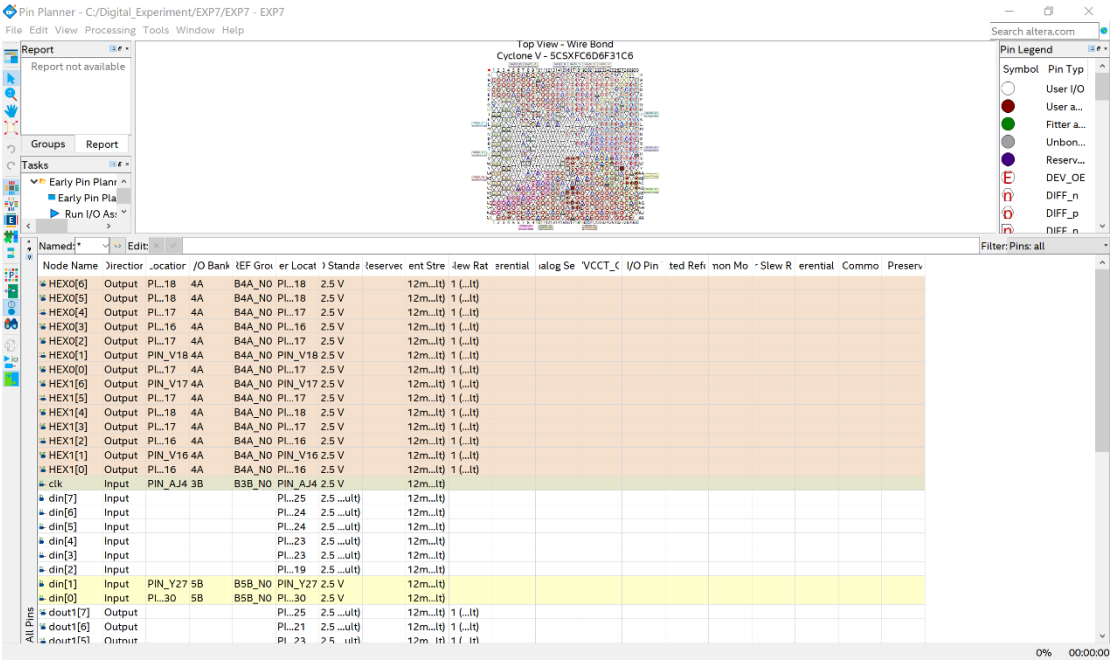
测试：

```
initial
begin
// code that executes only once
// insert code here --> begin
    clk = 0; we = 1; din = 0; inaddr = 0; outaddr = 0; #5;
                     din = 8'b00000001; inaddr = 1; outaddr = 0; #5;
                     din = 8'b00000010; inaddr = 4; outaddr = 5; #5;
                     din = 8'b00000100; inaddr = 2; outaddr = 3; #5;
                     din = 8'b00001000; inaddr = 3; outaddr = 7; #5;
                     din = 8'b00010000; inaddr = 5; outaddr = 6; #5;
                     din = 8'b00100000; inaddr = 7; outaddr = 2; #5;
                     din = 8'b01000000; inaddr = 0; outaddr = 1; #5;
                     din = 8'b10000000; inaddr = 6; outaddr = 4; #5;
                     din = 8'b00000001; inaddr = 1; outaddr = 0; #5;
                     din = 8'b00000010; inaddr = 4; outaddr = 5; #5;
                     din = 8'b00000100; inaddr = 2; outaddr = 3; #5;
                     din = 8'b00001000; inaddr = 3; outaddr = 7; #5;
                     din = 8'b00010000; inaddr = 5; outaddr = 6; #5;
                     din = 8'b00100000; inaddr = 7; outaddr = 2; #5;
                     din = 8'b01000000; inaddr = 0; outaddr = 1; #5;
                     din = 8'b10000000; inaddr = 6; outaddr = 4; #5;
    $stop;
end

always
begin
    #2 clk = ~clk;
    #2 we = ~we;
end
```

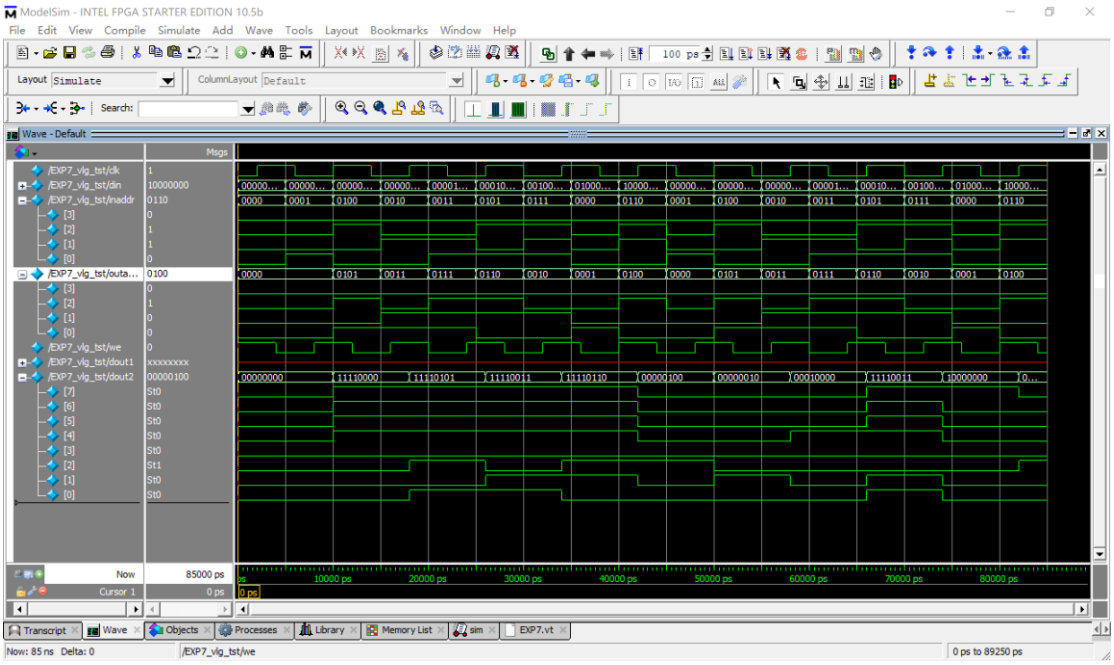
编译：



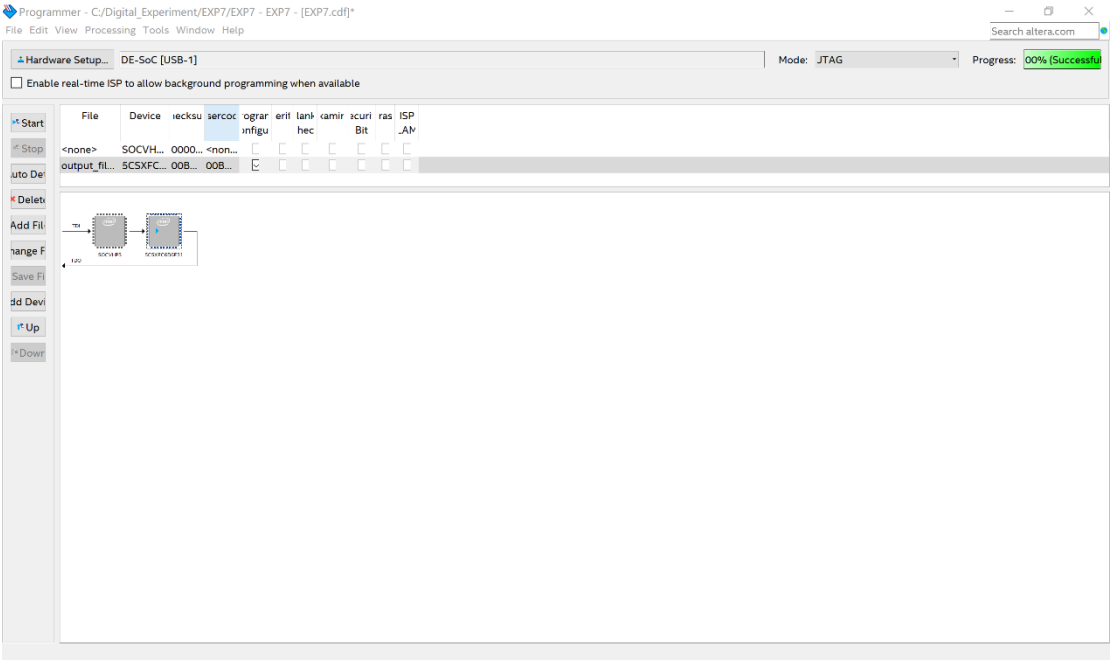
引脚分配:


Pin Planner - C:/Digital_Experiment/EXP7/EXP7 - EXP7
File Edit View Processing Tools Window Help
Top View - Wire Bond
Cyclone V - 5CSXFC6D6F31C6
Pin Legend
Report not available
Groups Report
Tasks
Node Name
HEX0[6] Output
HEX0[5] Output
HEX0[4] Output
HEX0[3] Output
HEX0[2] Output
HEX0[1] Output
HEX0[0] Output
HEX1[6] Output
HEX1[5] Output
HEX1[4] Output
HEX1[3] Output
HEX1[2] Output
HEX1[1] Output
HEX1[0] Output
clk Input
din[7] Input
din[6] Input
din[5] Input
din[4] Input
din[3] Input
din[2] Input
din[1] Input
din[0] Input
dout1[7] Output
dout1[6] Output
All Pins

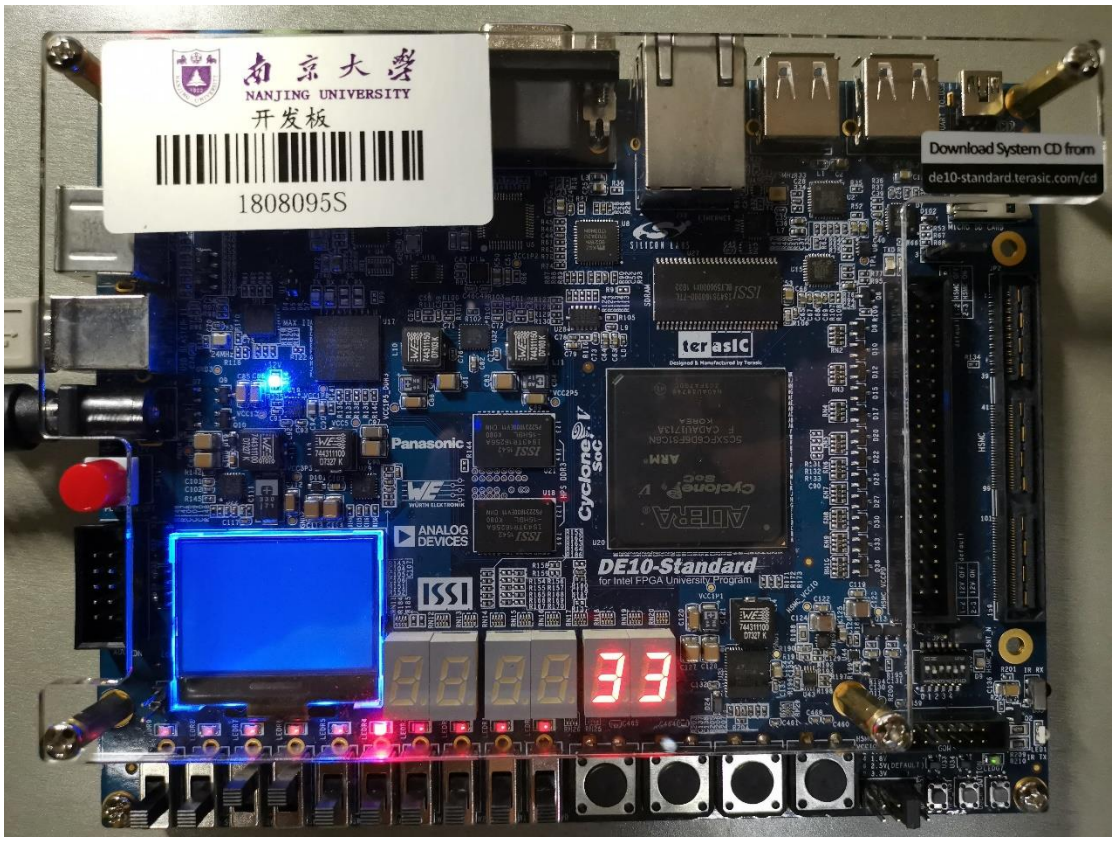
仿真:


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
/EXP7_vlg_tst/clk
/EXP7_vlg_tst/din
/EXP7_vlg_tst/inaddr
/EXP7_vlg_tst/outa...
/EXP7_vlg_tst/we
/EXP7_vlg_tst/dout1
/EXP7_vlg_tst/dout2
Now
Cursor 1
Transcript
Wave
Objects
Processes
Library
Memory List
sim
EXP7.vt

写入：


Programmer - C:/Digital_Experiment/EXP7/EXP7 - EXP7 - [EXP7.cdf]*
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
DE-SoC [USB-1]
Mode: JTAG
Progress:
Enable real-time ISP to allow background programming when available
Start
Stop
Add Fil
Save Fi
Up
File
Device
<none>
SOCVH...
output_fil...
5CSXFC...

硬件验证：


南京大学
NANJING UNIVERSITY
开发板
1808095S
Download System CD from
de10-standard.terasic.com/cd
terasIC
Panasonic
ANALOG DEVICES
DE10-Standard
for Intel FPGA University Program

六、测试方法

Test Bench

任给 we、inaddr、outaddr 赋值，观察 dout 随 clk、we、inaddr、outaddr 的变化情况

```
initial
begin
// code that executes only once
// insert code here --> begin
    clk = 0; we = 1; din = 0; inaddr = 0; outaddr = 0; #5;
                     din = 8'b00000001; inaddr = 1; outaddr = 0; #5;
                     din = 8'b00000010; inaddr = 4; outaddr = 5; #5;
                     din = 8'b00000100; inaddr = 2; outaddr = 3; #5;
                     din = 8'b00001000; inaddr = 3; outaddr = 7; #5;
                     din = 8'b00010000; inaddr = 5; outaddr = 6; #5;
                     din = 8'b00100000; inaddr = 7; outaddr = 2; #5;
                     din = 8'b01000000; inaddr = 0; outaddr = 1; #5;
                     din = 8'b10000000; inaddr = 6; outaddr = 4; #5;
                     din = 8'b00000001; inaddr = 1; outaddr = 0; #5;
                     din = 8'b00000010; inaddr = 4; outaddr = 5; #5;
                     din = 8'b00000100; inaddr = 2; outaddr = 3; #5;
                     din = 8'b00001000; inaddr = 3; outaddr = 7; #5;
                     din = 8'b00010000; inaddr = 5; outaddr = 6; #5;
                     din = 8'b00100000; inaddr = 7; outaddr = 2; #5;
                     din = 8'b01000000; inaddr = 0; outaddr = 1; #5;
                     din = 8'b10000000; inaddr = 6; outaddr = 4; #5;
    $stop;
end

always
begin
    #2 clk = ~clk;
    #2 we = ~we;
end
```

八、实验中遇到的问题及解决办法

1、mif 文件无法输入十六进制数字

解决办法：mif 文件初始默认为输入十进制数字，在 View → Memory Radix 中可将其改为十六进制（Hexadecimal）

2、未设置 din 的高位时，高位会被自动安排奇怪的值

解决办法：只选取低位进行读取，避免高位值因未定义而产生的不确定性带来不便

3、HEX1 总是比 HED0 满一个时钟周期

解决办法：发现是创建 RAM 时产生的缓冲区导致的。

九、实验得到的启示

1、用 txt 或 mif 文件进行初始化十分方便，省去了在代码中赋值的繁杂部分，也使得初始化的修改更便捷，是一个很好的功能。

2、为了一个目的，可以尝试使用不同的方式来达到，如：存储器的实现。

十、思考题

1、如果将图 2 中存储器实现部分改为图 1

```
always @(posedge clk)
      if (we)
         ram[inaddr] <= din;
      else
         dout <= ram[outaddr];
```

```
module ram #(
  parameter RAM_WIDTH = 32,
  parameter RAM_ADDR_WIDTH = 10
)(
    input clk,
    input we,
    input [RAM_WIDTH-1:0] din,
    input [RAM_ADDR_WIDTH-1:0] inaddr,
    input [RAM_ADDR_WIDTH-1:0] outaddr,
    output [RAM_WIDTH-1:0] dout
);

   reg [RAM_WIDTH:0] ram [(2**RAM_ADDR_WIDTH)-1:0];

   always @(posedge clk)
      if (we)
         ram[inaddr] <= din;

   assign dout = ram[outaddr];

endmodule
```

该存储器的行为是否会发生变化?

会发生改变，原来的 dout 会一直从 ram[outaddr]读数据，但现在只有当 we=0 时才能读数据，即读写不能同时进行

十一、意见和建议

1、讲义中 7.3.2 存储器初始化里，突然跳到 mif 文件的生成步骤，再回到 browse，这一部分有点跳跃，弄了好几次才明白之前的 IP 核生成对话框不能点 OK，需要生成 mif 文件后才能点 OK。