**中山大学数据科学与计算机学院本科生实验报告**

**（2019学年秋季学期）**

课程名称：**计算机组成原理实验**  任课教师：郭雪梅 助教：汪庭葳、刘洋旗

|  |  |  |  |
| --- | --- | --- | --- |
| 年级&班级 | **2018级1班** | 专业(方向) | **计算机类** |
| 学号 | **18308045** | 姓名 | **谷正阳** |
| 电话 | **13355426001** | Email | [**Guzy0324@163.com**](mailto:Guzy0324@163.com) |
| 开始日期 | **2019.11.13** | 完成日期 | **2019.11.13** |

**一、实验题目**

计算机结构与组成 实验6

1. **实验目的**

1. 掌握存储器的设计原理，并且自己设计一个大小合适的ROM。

2. 掌握存储器的存储和取数的过程，并将存储器的内容通过数码管显示出来。

3. 学会封装IP核以及调用方法。

**三、实验内容**

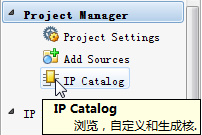
**1.** 实验步骤

（一）．新建工程文件(xc7a35tcpg236-1)，建立存储器系统并仿真

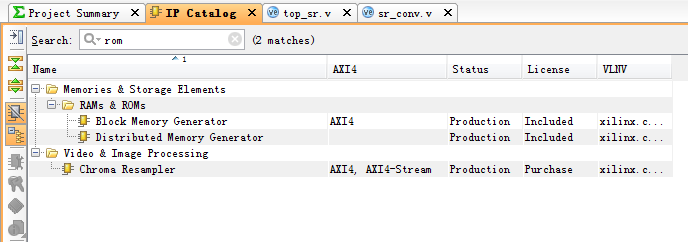
1.**添加调用rom IP核:**

在IP Catalog > Memories & Storage Elements > RAMs & ROMs> Block Memory Generator目录底下，找到系统自带的rom核并添加。

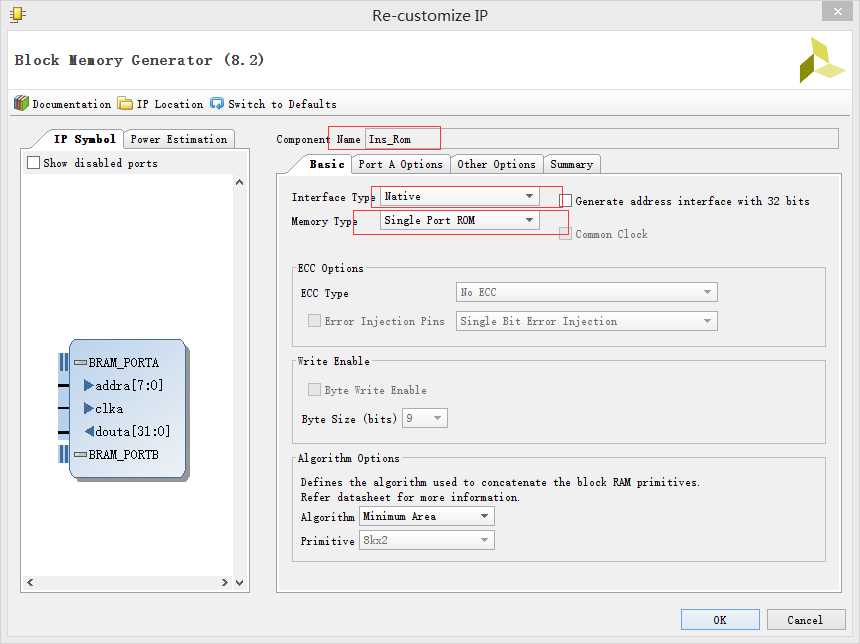
点击Flow Navigator中的IP Catalog，打开窗口添加IP核。



Block Memory为块存储设备，这里需要的是Block Memory Generator

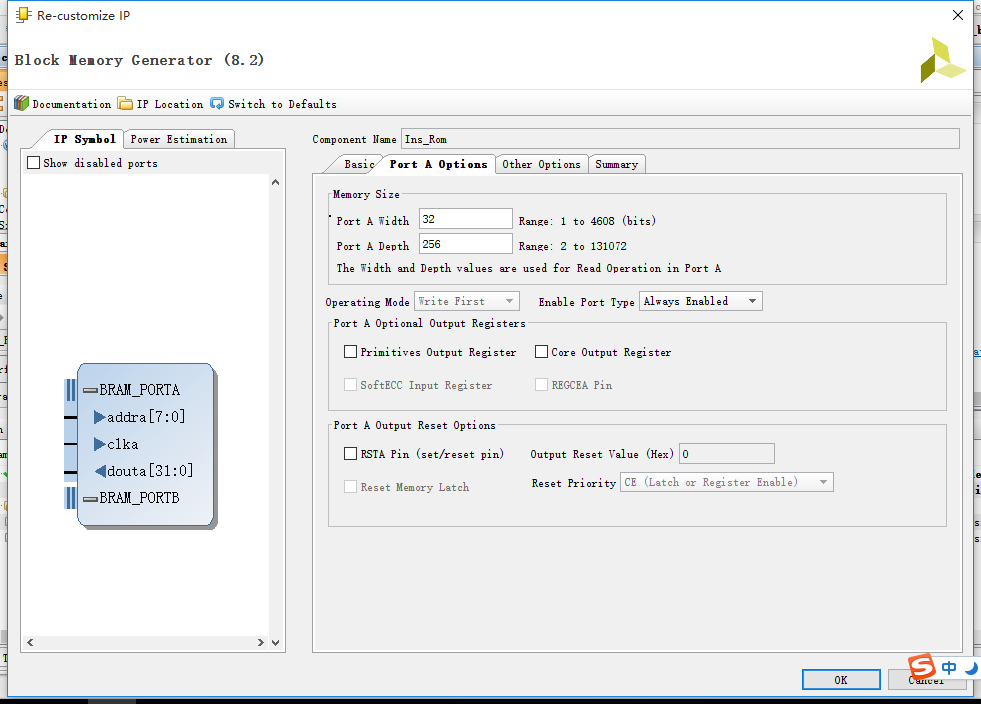


双击之后得到Basic一栏的基本属性：



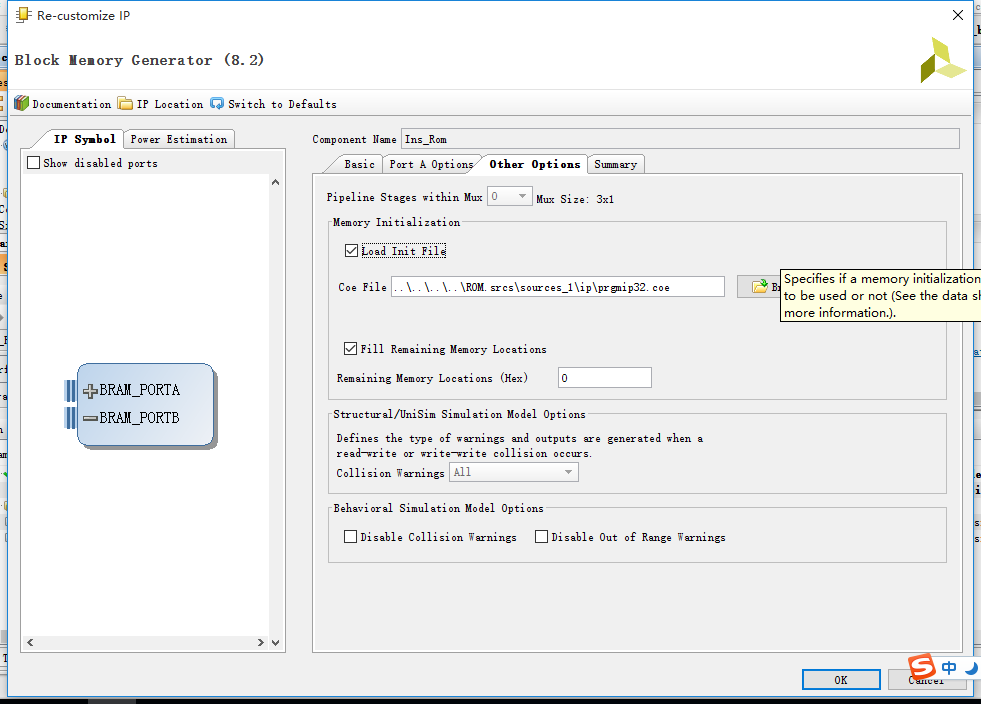
Component Name命名为：Ins\_Rom。Native、单端口、ROM不要ECC校验、最小面积算法。

接着设置Port A Option的基本属性



数据宽度16位，深度要256个数据，地址线8根。

接着设置option选项属性：



在LOAD INIT FILE 项上打勾，点击LOAD FILE，要你选择一个文件。  
这个文件就是你要放在ROM中的数据，文件命名prgmip16.coe，文本文件。你可以自己建一个文本文件，然后把后缀名改为.COE即可。(文件中放入自己的学号或你想显示任意内容)。

RST & Initialization：使用coe文件可以给ROM输入初值，格式如下：

memory\_initialization\_radix=16;

memory\_initialization\_vector=

0108,0304,

0506,0907,

0001,0004,

0005,0009,

0007,0005,

0000,0005,

8888,8888,

9999,9999,

Aaaa,aaaa,

Bbbb,bbbb，

1234,5678，

12ab,cdef;

可以直接用文本编辑器写好COE文件，radix=16进制定义，这里是16进制，第二行照抄，第三行开始是数据（我们定义的16位数据，rom数据宽度），数据间隔用逗号，最后一个数据分号。COE文件最后会生成MIF文件用于初始化ROM。

点击ok，生成rom ip核相应的文件。

## 2.添加设计源文件

`timescale 1ns / 1ps

module rom\_top(Clk,Rst,data);

input Clk;//系统时钟

input Rst;//高电平复位

output [15:0] data;//输出数据

reg [7:0]addr;//rom地址

always@(posedge Clk or negedge Rst)

if(Rst)

addr<=8'd0;

else if(addr == 8'd255)

addr<=8'd0;

else

addr<=addr+1'b1;

blk\_mem\_gen\_0 rom (

.clka(Clk), // input wire clka

.ena(1'b1), // input wire ena数据输出允许

.addra(addr), // input wire [7 : 0] addra

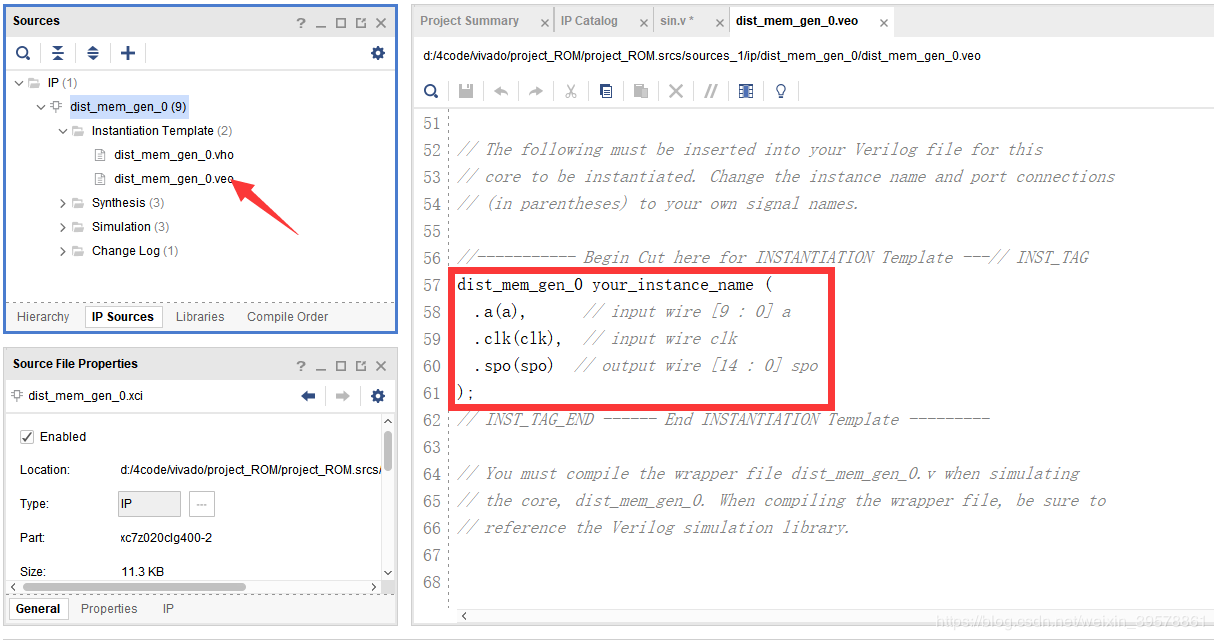
.douta(data) // output wire [15 : 0] douta

);

endmodule

ROM例化文件的方法：

ip核的例化，在source下面有一个IP Source选项卡点击进去，再点击Instantiation Template里的.veo文件就是verilog例化文件了,默认名称blk\_mem\_gen\_0.veo.

如图所示  
 

把实例化代码复制到top文件中，在此基础上进行修改。

编写好源代码后，进行综合，在Flow Navigator选项卡选择synthesis，点击 “run synthesis”，综合后可以在下面的message一栏查看综合信息，是否有错误，不要有critical warning和error。

**3. 功能仿真：**

接下来进行testbench仿真，在source中右击simulation source 选择add or create simulation souce 点击Next  点击“+”，选择“create file”打开“create source file”填写相应的文件名，点击ok，回到“add or create simulation souce”再点击finish。完成创建testbench仿真文件，编写测试文件，测试文件编写后，点击 “run synthesis”，综合后在下面的message一栏查看综合信息，是否有critical warning和错误。

一切没有问题，点击run simulation、Run behavioral simulation就可以进行功能仿真了。对应的代码如下

测试文件的代码：

`timescale 1ns / 1ps

`define clk\_period 10

module rom\_tb;

// Inputs

reg Clk;

reg Rst;

// Outputs

wire [15:0] dout;

rom\_top rom\_top(

.Clk(Clk),

.Rst(Rst),

.data(dout));

initial

begin

Clk = 1;

Rst=0;

end

always #(`clk\_period/2)Clk = ~Clk;

initial begin

Rst=1'b1;

#(`clk\_period\*3);

Rst=1'b0;

#(`clk\_period\*1000);

$stop;

end

endmodule

**（二）加入数码管显示，修改文件**

**1.添加数码管设计源文件文件display**

module display(clk,data,sm\_wei,sm\_duan);

input clk;

input [15:0] data;

output [3:0] sm\_wei;

output [6:0] sm\_duan;

//----------------------------------------------------------

//分频

//位控制

reg [3:0]wei\_ctrl=4'b1110; always @(posedge clk)

wei\_ctrl <= {wei\_ctrl[2:0],wei\_ctrl[3]}; //段控制

reg [3:0]duan\_ctrl;

always @(wei\_ctrl)

case(wei\_ctrl)

4'b1110:duan\_ctrl=data[3:0];

4'b1101:duan\_ctrl=data[7:4];

4'b1011:duan\_ctrl=data[11:8];

4'b0111:duan\_ctrl=data[15:12];

default:duan\_ctrl=4'hf;

endcase

//----------------------------------------------------------

//解码模块

reg [6:0]duan;

always @(duan\_ctrl)

case(duan\_ctrl)

4'h0:duan=7'b100\_0000;//0

4'h1:duan=7'b111\_1001;//1

4'h2:duan=7'b010\_0100;//2

4'h3:duan=7'b011\_0000;//3

4'h4:duan=7'b001\_1001;//4

4'h5:duan=7'b001\_0010;//5

4'h6:duan=7'b000\_0010;//6

4'h7:duan=7'b111\_1000;//7

4'h8:duan=7'b000\_0000;//8

4'h9:duan=7'b001\_0000;//9

4'ha:duan=7’b000\_1000;//a

4'hb:duan=7'b000\_0011;//b

4'hc:duan=7'b100\_0110;//c

4'hd:duan=7'b010\_0001;//d

4'he:duan=7'b000\_0111;//e 4'hf:duan=7'b000\_1110;//f

// 4'hf:duan=7'b111\_1111;//不显示

default : duan = 7'b100\_0000;//0

endcase

//----------------------------------------------------------

assign sm\_wei = wei\_ctrl;

assign sm\_duan = duan;

endmodule

2.**在rom\_top文件中添加调用数码管显示所读出的rom内容。**

每个时钟周期从rom取出数据，然后送给数码管显示，在top文件中加入时钟分频（系统所提供时钟100MHZ太快），取出和显示一致，大家可以调整合适的分频数据。因为top文件中对时钟分频后的clkin送给数码管显示位扫描，故display中可以去掉分频部分。参考文件：

`timescale 1ns / 1ps

module rom\_top(Clk,Rst,seg,sm\_wei);

input Clk;//系统时钟

input Rst;//高电平复位

// output [15:0] data;//输出数据

output [6:0] seg;//段码

output [3:0] sm\_wei;//哪个数码管

wire [15:0] data;

reg [7:0]addr;//rom地址

integer clk\_cnt;

reg clkin;//1.33秒一个时钟周期，以此为cpu的时钟周期 10^8/0.75\*10^8=4/3=1.33 always @(posedge Clk)

if(clk\_cnt==32'd75\_000\_000)

begin

clk\_cnt <= 1'b0;

clkin <= ~clkin;

end

else

clk\_cnt <= clk\_cnt + 1'b1;

always@(posedge clkin or negedge Rst)

if(Rst)

addr<=8'd0;

else if(addr == 8'd255)

addr<=8'd0;

else

addr<=addr+1'b1;

blk\_mem\_gen\_0 rom (

.clka(clkin), // input wire clka

.ena(1'b1), // input wire ena

.addra(addr), // input wire [7 : 0] addra

.douta(data) // output wire [15 : 0] douta

);

display dis

(.clk(clkin),

.data (data),

.sm\_duan(seg),

.sm\_wei(sm\_wei));

endmodule

3. 添加约束文件

set\_property PACKAGE\_PIN U7 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN U5 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN U8 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN W6 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN W7 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN U2 [get\_ports {sm\_wei[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {sm\_wei[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {sm\_wei[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {sm\_wei[3]}]

set\_property PACKAGE\_PIN W5 [get\_ports Clk]

set\_property PACKAGE\_PIN U18 [get\_ports Rst]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[1]}]

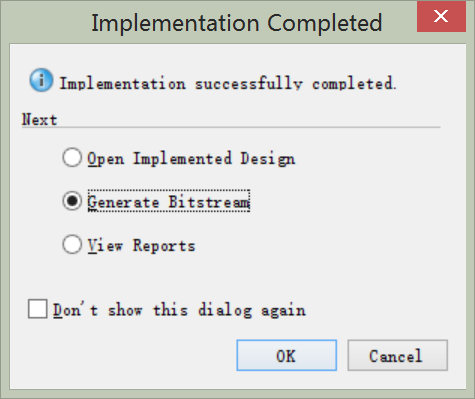
set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports Clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports Rst]

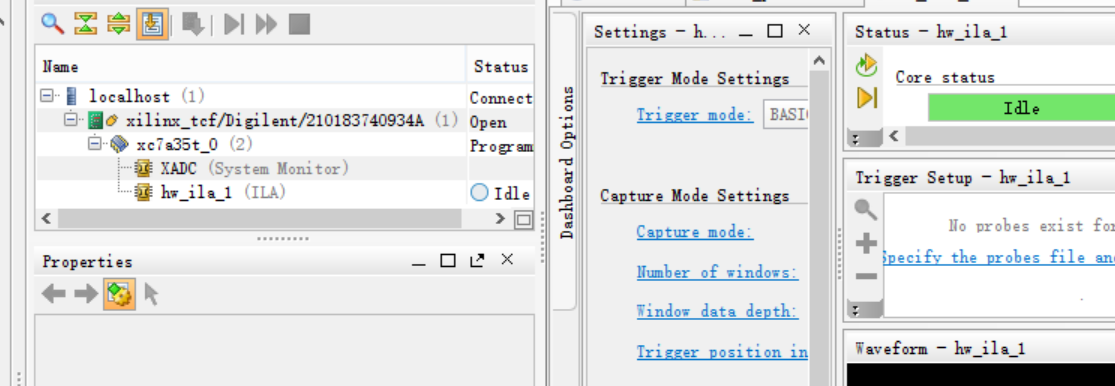
**4. 生成比特流**

 点击生成比特流，见下图：



**5. 下载程序**

 （1）与硬件相连



 （2）下载程序

**6. 实物演示**

**（三）拓展**

设计rom 选择数据宽度8位，取出数据后组合成16位后再送去数码管显示，修改有关文件，FPGA实现。

**2.** 实验原理

本次实验主要是模拟数据在存储器中的存取过程，先初始化ROM存储器中的内容，再通过开关选择相应的地址，将对应的存储器中内容读出来，并通过显示管显示。实验的原理图如下图1所示：



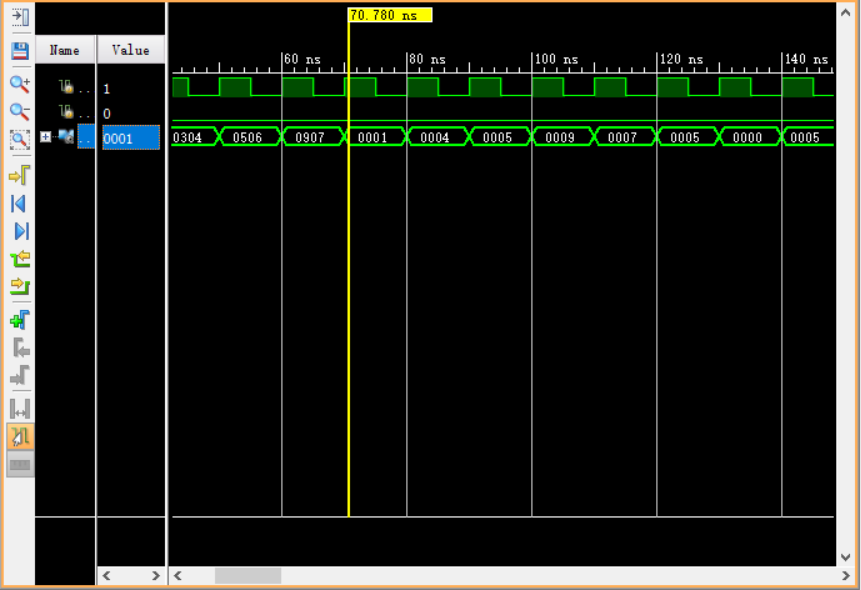
图1存储器原理图

我们使用的IP核，ins\_ROM核是系统自带的，通过该IP核我们实现数据的存取，通过8位地址开关的选择，将ROM中对应的16位数据取出来并送往display。通过display可以将ROM中取出的16位数据在数码管上显示出来。

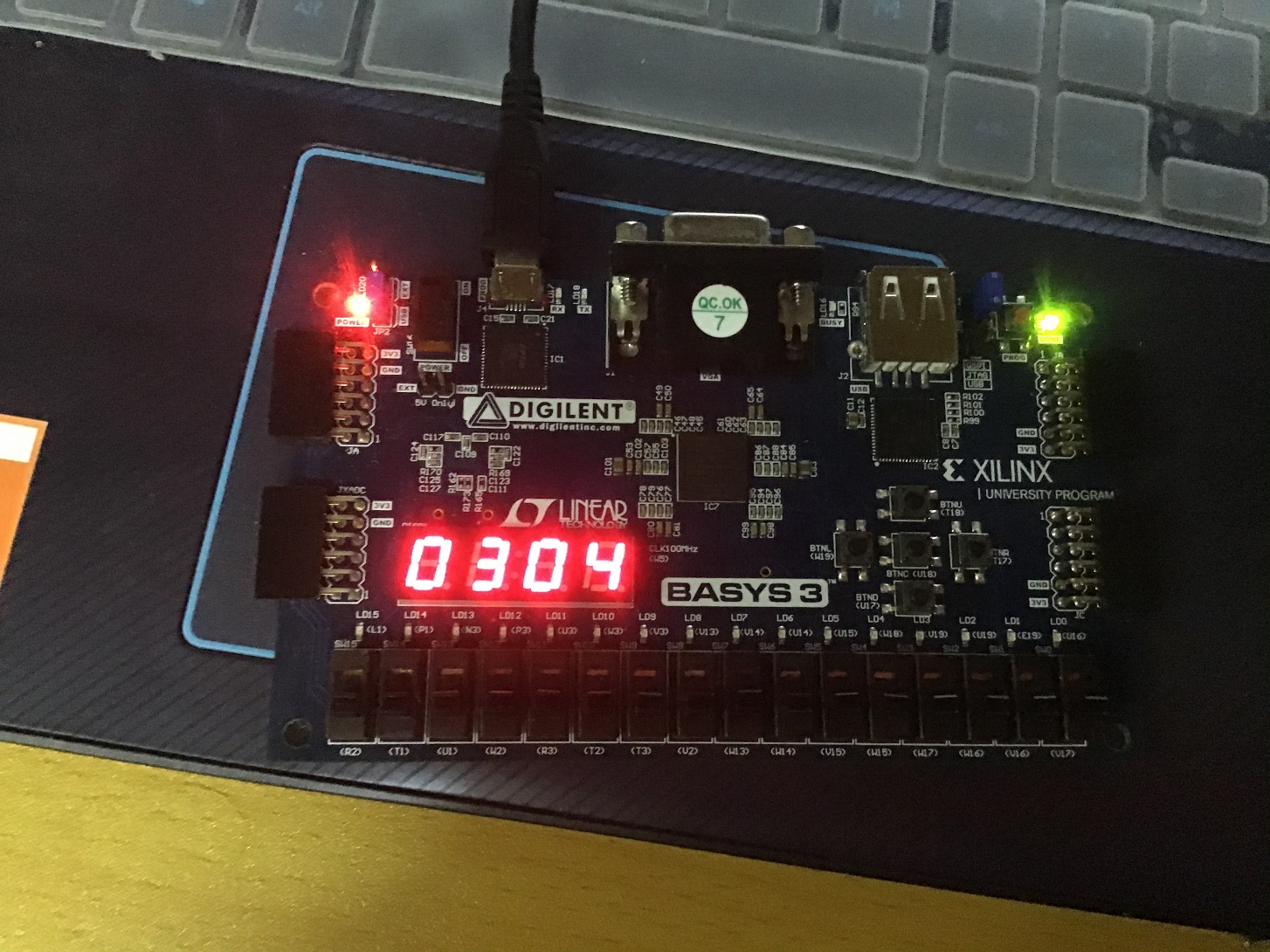
存储容量是指存储器可以容纳的二进制信息量，用存储器中存储地址数与存储字位数的乘积表示。

**四、实验结果**

**实验1：**

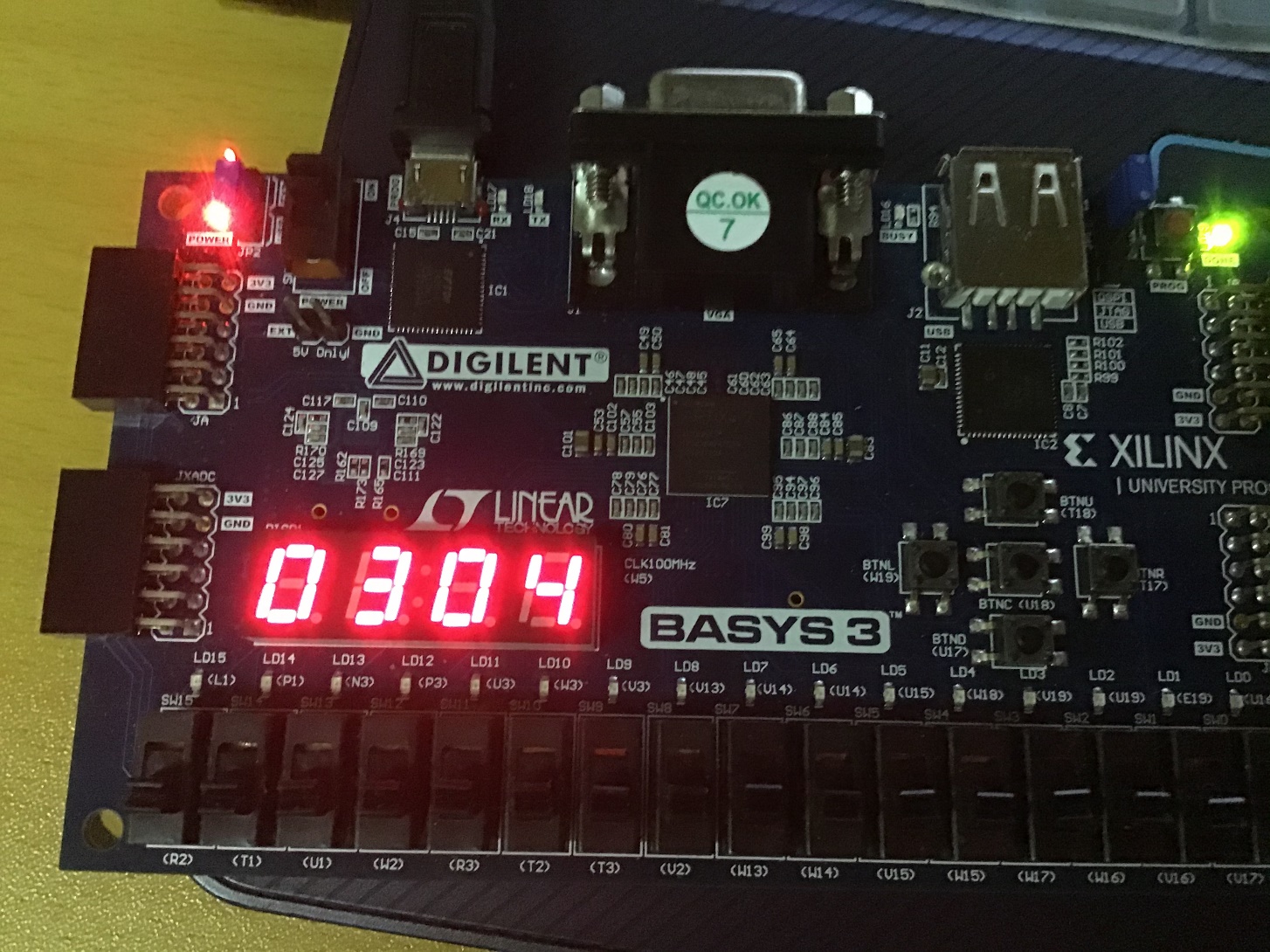


**实验2：**



对应模拟里面显示的0304

**实验3：**



01 08 03 04 05 06 09 07

是第三四个8位二进制数拼接出来的

**五、实验感想**

ip核真好用，全都是现成的……有了rom没准不久后便可以模拟出pipeline了……

**附录（流程图，注释过的代码）：、**

**实验1：**

**rom\_top：**

`timescale 1ns / 1ps

module rom\_top(Clk,Rst,data/\*,seg,sm\_wei\*/);

input Clk;//系统时钟

input Rst;//高电平复位

output [15:0] data;//输出数据

//output [6:0] seg;//段码

//output [3:0] sm\_wei;//哪个数码管

wire [15:0] data;

reg [7:0]addr;//rom地址

integer clk\_cnt;

reg clkin;//1.33秒一个时钟周期，以此为cpu的时钟周期 10^8/0.75\*10^8=4/3=1.33

always @(posedge Clk)

if(clk\_cnt==32'd75\_000\_000)

begin

clk\_cnt <= 1'b0;

clkin <= ~clkin;

end

else

clk\_cnt = clk\_cnt + 1'b1;

always@(posedge /\*clkin\*/Clk or negedge Rst)

if(Rst)

addr<=8'd0;

else if(addr == 8'd255)

addr<=8'd0;

else

addr<=addr+1'b1;

Ins\_Rom Ins\_Rom (

.clka(/\*clkin\*/Clk), // input wire clka

.addra(addr), // input wire [7 : 0] addra

.douta(data) // output wire [15 : 0] douta

);/\*

display dis

(.clk(Clk),

.data (data),

.sm\_duan(seg),

.sm\_wei(sm\_wei));\*/

endmodule

**prgmip16：**

memory\_initialization\_radix=16;

memory\_initialization\_vector=

0108,0304,

0506,0907,

0001,0004,

0005,0009,

0007,0005,

0000,0005,

8888,8888,

9999,9999,

Aaaa,aaaa,

Bbbb,bbbb,

1234,5678,

12ab,cdef;

**rom\_tb：**

`timescale 1ns / 1ps

`define clk\_period 10

module rom\_tb;

// Inputs

reg Clk;

reg Rst;

// Outputs

wire [15:0] dout;

rom\_top rom\_top(

.Clk(Clk),

.Rst(Rst),

.data(dout));

initial

begin

Clk = 1;

Rst=0;

end

always #(`clk\_period/2)Clk = ~Clk;

initial begin

Rst=1'b1;

#(`clk\_period\*3);

Rst=1'b0;

#(`clk\_period\*1000);

$stop;

end

endmodule

**实验2：**

**rom\_top：**

`timescale 1ns / 1ps

module rom\_top(Clk,Rst,seg,sm\_wei);

input Clk;//系统时钟

input Rst;//高电平复位

//output [15:0] data;//输出数据

output [6:0] seg;//段码

output [3:0] sm\_wei;//哪个数码管

wire [15:0] data;

reg [7:0]addr;//rom地址

integer clk\_cnt;

reg clkin;//1.33秒一个时钟周期，以此为cpu的时钟周期 10^8/0.75\*10^8=4/3=1.33

always @(posedge Clk)

if(clk\_cnt==32'd75\_000\_000)

begin

clk\_cnt <= 1'b0;

clkin <= ~clkin;

end

else

clk\_cnt = clk\_cnt + 1'b1;

always@(posedge clkin or negedge Rst)

if(Rst)

addr<=8'd0;

else if(addr == 8'd255)

addr<=8'd0;

else

addr<=addr+1'b1;

Ins\_Rom Ins\_Rom (

.clka(clkin), // input wire clka

.addra(addr), // input wire [7 : 0] addra

.douta(data) // output wire [15 : 0] douta

);

display dis

(.clk(Clk),

.data (data),

.sm\_duan(seg),

.sm\_wei(sm\_wei));

endmodule

**display：**

module display(clk,data,sm\_wei,sm\_duan);

input clk;

input [15:0] data;

output [3:0] sm\_wei;

output [6:0] sm\_duan;

//----------------------------------------------------------

//分频

integer clk\_cnt;

reg clk\_400Hz;

always @(posedge clk)

if(clk\_cnt==32'd100000)

begin clk\_cnt <= 1'b0; clk\_400Hz <= ~clk\_400Hz;

end else clk\_cnt <= clk\_cnt + 1'b1;

//位控制

reg [3:0]wei\_ctrl=4'b1110; always @(posedge clk\_400Hz)

wei\_ctrl <= {wei\_ctrl[2:0],wei\_ctrl[3]}; //段控制

reg [3:0]duan\_ctrl;

always @(wei\_ctrl)

case(wei\_ctrl)

4'b1110:duan\_ctrl=data[3:0];

4'b1101:duan\_ctrl=data[7:4];

4'b1011:duan\_ctrl=data[11:8];

4'b0111:duan\_ctrl=data[15:12];

default:duan\_ctrl=4'hf;

endcase

//----------------------------------------------------------

//解码模块

reg [6:0]duan;

always @(duan\_ctrl)

case(duan\_ctrl)

4'h0:duan=7'b100\_0000;//0

4'h1:duan=7'b111\_1001;//1

4'h2:duan=7'b010\_0100;//2

4'h3:duan=7'b011\_0000;//3

4'h4:duan=7'b001\_1001;//4

4'h5:duan=7'b001\_0010;//5

4'h6:duan=7'b000\_0010;//6

4'h7:duan=7'b111\_1000;//7

4'h8:duan=7'b000\_0000;//8

4'h9:duan=7'b001\_0000;//9

4'ha:duan=7'b000\_1000;//a

4'hb:duan=7'b000\_0011;//b

4'hc:duan=7'b100\_0110;//c

4'hd:duan=7'b010\_0001;//d

4'he:duan=7'b000\_0111;//e

4'hf:duan=7'b000\_1110;//f

// 4'hf:duan=7'b111\_1111;//不显示

default : duan = 7'b100\_0000;//0

endcase

//----------------------------------------------------------

assign sm\_wei = wei\_ctrl;

assign sm\_duan = duan;

endmodule

**constraint：**

set\_property PACKAGE\_PIN U7 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN U5 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN U8 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN W6 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN W7 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN U2 [get\_ports {sm\_wei[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {sm\_wei[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {sm\_wei[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {sm\_wei[3]}]

set\_property PACKAGE\_PIN W5 [get\_ports Clk]

set\_property PACKAGE\_PIN U18 [get\_ports Rst]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports Clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports Rst]

**实验3：**

**rom\_top：**

`timescale 1ns / 1ps

module rom\_top(Clk,Rst/\*,data\*/,seg,sm\_wei);

input Clk;//系统时钟

input Rst;//高电平复位

//output [15:0] data;//输出数据

output [6:0] seg;//段码

output [3:0] sm\_wei;//哪个数码管

wire [7:0] data1;

wire [7:0] data2;

wire [15:0] data;

reg [7:0]addr;//rom地址

integer clk\_cnt;

reg clkin;//1.33秒一个时钟周期，以此为cpu的时钟周期 10^8/0.75\*10^8=4/3=1.33

always @(posedge Clk)

if(clk\_cnt==32'd75\_000\_000)

begin

clk\_cnt <= 1'b0;

clkin <= ~clkin;

end

else

clk\_cnt = clk\_cnt + 1'b1;

always@(posedge clkin/\*Clk\*/ or negedge Rst)

if(Rst)

addr<=8'd0;

else if(addr == 8'd255)

addr<=8'd0;

else

//addr<=addr+1'b1;

addr<=addr+2'b10;

/\*

Ins\_Rom Ins\_Rom (

.clka(clkin/\*Clk\*//\*), // input wire clka

.addra(addr), // input wire [7 : 0] addra

.douta(data) // output wire [15 : 0] douta

);\*/

Ins\_Rom\_2 Ins\_Rom\_1 (

.clka(clkin/\*Clk\*/), // input wire clka

.ena(1'b1),

.addra(addr), // input wire [7 : 0] addra

.douta(data1) // output wire [15 : 8] douta

);

Ins\_Rom\_2 Ins\_Rom\_2 (

.clka(clkin/\*Clk\*/), // input wire clka

.ena(1'b1),

.addra(addr+1'b1), // input wire [7 : 0] addra

.douta(data2) // output wire [7 : 0] douta

);

assign data[15:8] = data1[7:0];

assign data[7:0] = data2[7:0];

display dis

(.clk(Clk),

.data (data),

.sm\_duan(seg),

.sm\_wei(sm\_wei));

endmodule

**prgmip8：**

memory\_initialization\_radix=16;

memory\_initialization\_vector=01 08 03 04 05 06 09 07;