数字电路与数字系统实验

EX07:存储器

191220029 傅小龙
周一 5-6 节班
1830970417@qq.com
2020 年 10 月 14 日

目录

一、实验内容

1.1 实验要求

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
```

初始化数值详见 2.1.3 节.

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。

1.2 实验工具

软件环境：

设计、编译、仿真：Quartus Prime Version 17.1.0 Build 590 10/25/2017 SJ Lite Edition

DE10_Standard_SystemBuilder

硬件环境：DE-10 Standard 开发平台

FPGA 芯片：Cyclone V 5CSXFC6D6F31C6

二、实验过程

2.1 RAM1

2.1.1 模型概述

该存储器拥有 16*8 大小的内存，可进行读写操作. 当写使能信号有效时，在时钟信号的上升沿写入数据；当写使能信号无效时，在时钟信号的上升沿输出数据. 输出端有输出缓存. 时钟信号采用 1Hz 分频器的输出信号.

2.1.2 数字抽象

I) 输入:

时钟信号 clk： 与时钟信号连接.
写使能信号 we: 控制存储器进行读/写操作.
读地址[3:0]inaddr: 进行读操作时要输出的数据的地址.
写地址[3:0]outaddr: 进行写操作时要写入的数据的地址.
数据输入[7:0]din: 8 位二进制码的输入.

II) 输出:

数据输出[7:0]dout: 8 位二进制码的输出

下表\图给出了以上输入输出信号在 DE10 平台对应的信号：

	信号名称	DE-10 平台信号
输入	clk	LEDR[9]①
	we	KEY[0]
	[3:0]inaddr	SW[9:6]
	[3:0]outaddr	SW[5:2]
	[7:0]din	SW[1:0]②
输出	[7:0]dout	[7:0]dout0③

①:LEDR[9]为分频器产生的 1Hz 时钟信号输出

②:由于 FPGA 开发板的输入端数量有限，写入数据宽度约束为 2 位数据.

③:dout0 在顶层文件中为 wire[7:0]型变量，用以接收存储器的数据输出，并连接至七段数码管以在开发板上显示结果.

表 2-1-1:RAM1 输入输出信号与 DE10 平台信号对应关系

2.1.3 建立模型

实现思路：参照 exp07.pdf 中的表 7-2 中的部分 Verilog 代码和 exp07.pdf 中的 7.2.2 节给出的使用外部文件给存储器初始化的介绍，RAM1 的实现如下：

```
1   module RAM1(clk, we, inaddr, outaddr, din, dout);
2       input clk;
3       input we;
4       input [3:0]inaddr, outaddr;
5       input [7:0]din;
```

```
6       output reg [7:0]dout;
7
8       reg [7:0] ram [15:0];
9       reg [3:0] out;
10
11      initial begin
12          $readmemh(".\\init\\mem1.txt", ram, 0, 15);
13          out = 0;
14      end
15
16      always @(posedge clk) begin
17          if(we)
18              ram[inaddr] <= din;
19          out <= ram[outaddr];
20          dout <= out;     //输出缓存
21      end
22  endmodule
```

2.2 RAM2

2.2.1 模型概述

通过 IP 核设计一个双口存储器. 用.mif 文件进行初始化.

2.2.2 数字抽象

由 IP 核得到的双口存储器的数据输入输出如下所示：

```
1   RAM2 r2(
2   .clock(LEDR[9]),
3   .data(SW[1:0]),
4   .rdaddress(SW[5:2]),
5       .wraddress(SW[9:6]),
6       .wren(KEY[0]),
7   .q(dout1)  //①
8   );
```

①:dout1 在顶层文件中为 wire[7:0]型变量，用以接收存储器的数据输出，并连接至七段数管解码器以在开发板上显示结果.

①:LEDR[9]为分频器产生的 1Hz 时钟信号输出

2.2.3 建立模型

参照 exp07-7.3.1 节内容生成所需的 RAM，RAM2 的实现步骤如下：

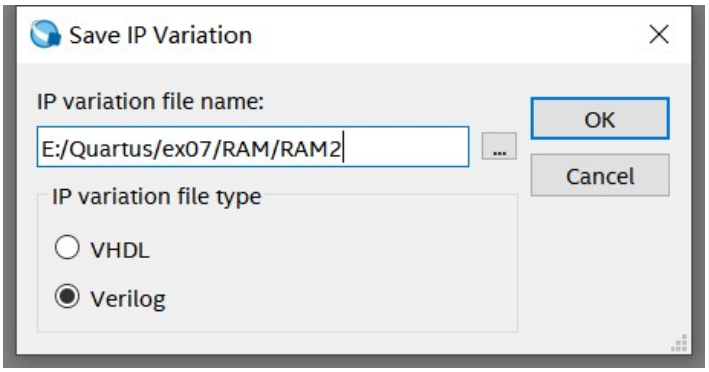


图 2-2-1：选择目标文件名

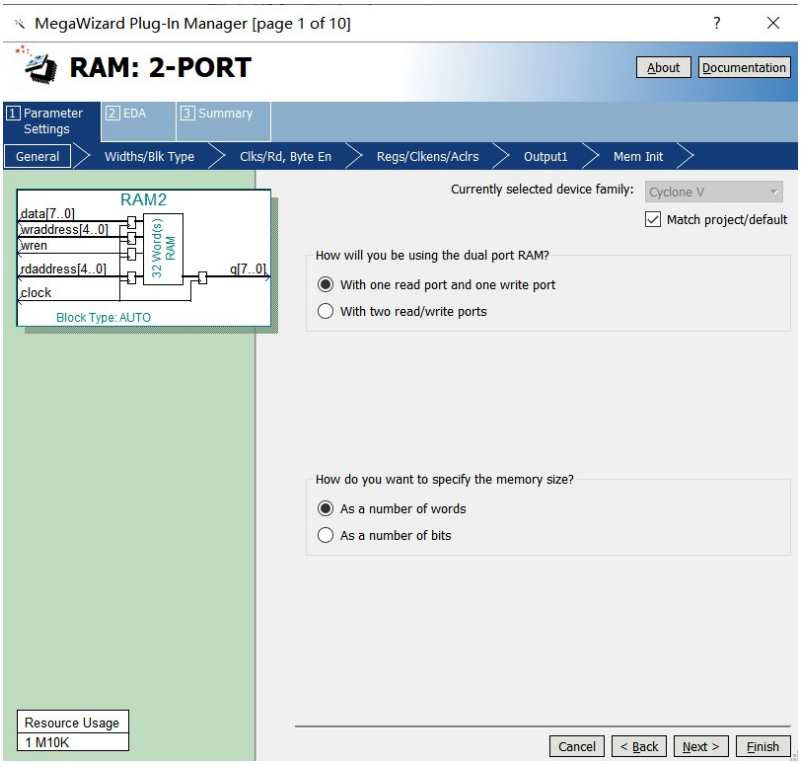


图 2-2-2：选择 RAM 格式

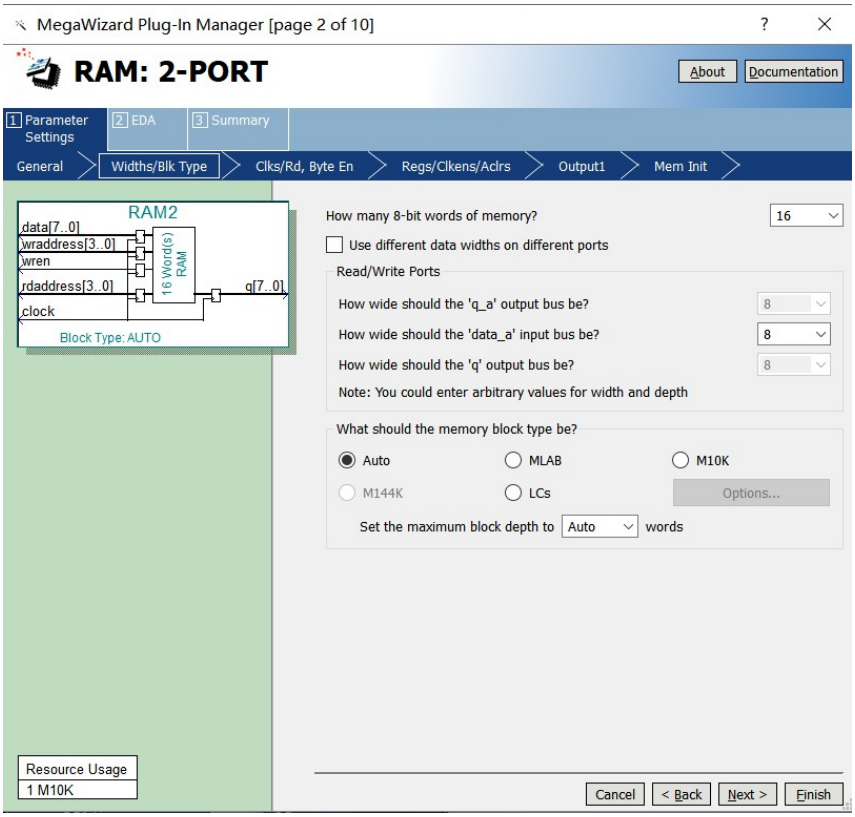


图 2-2-3：选择 RAM 规模

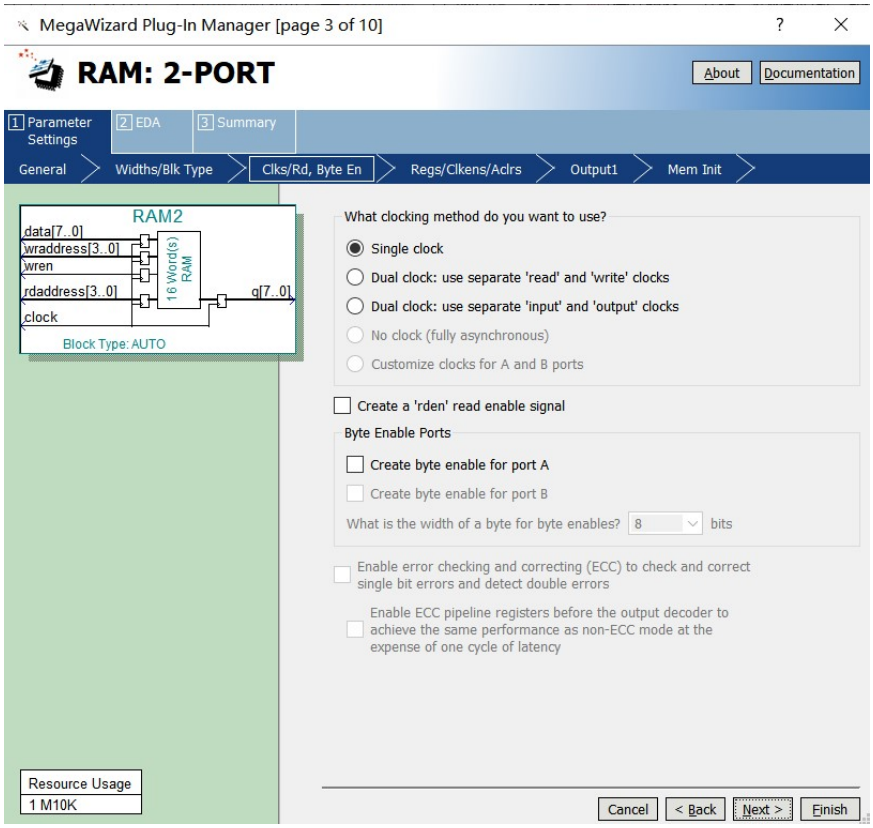


图 2-2-4：时钟信号配置

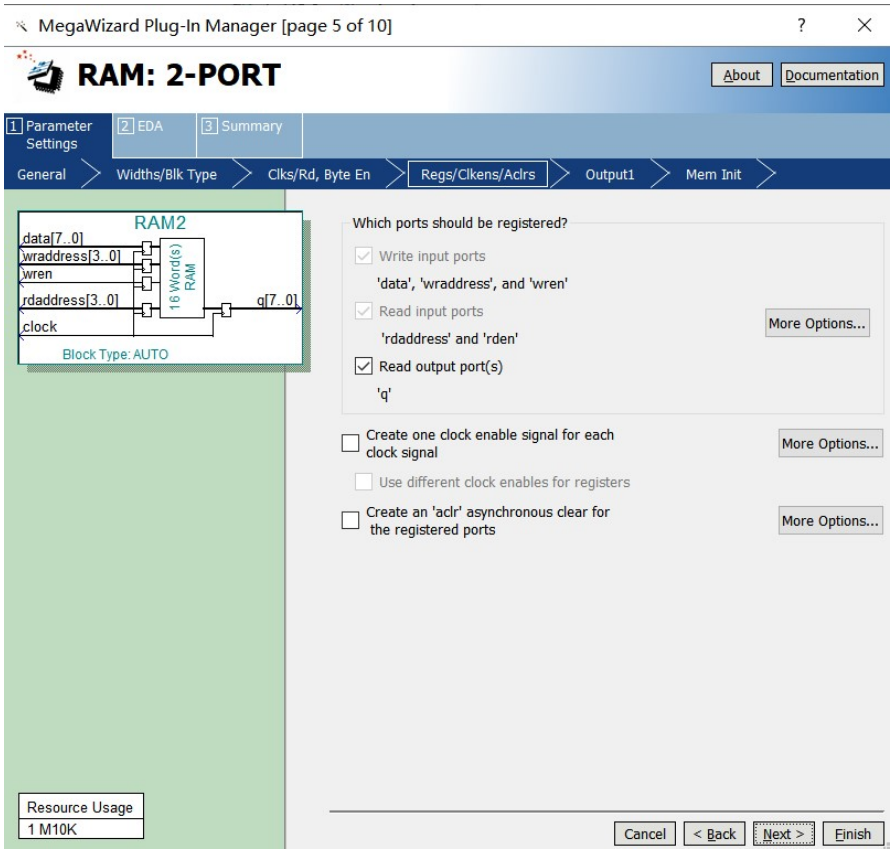


图 2-2-5：输出缓存选择

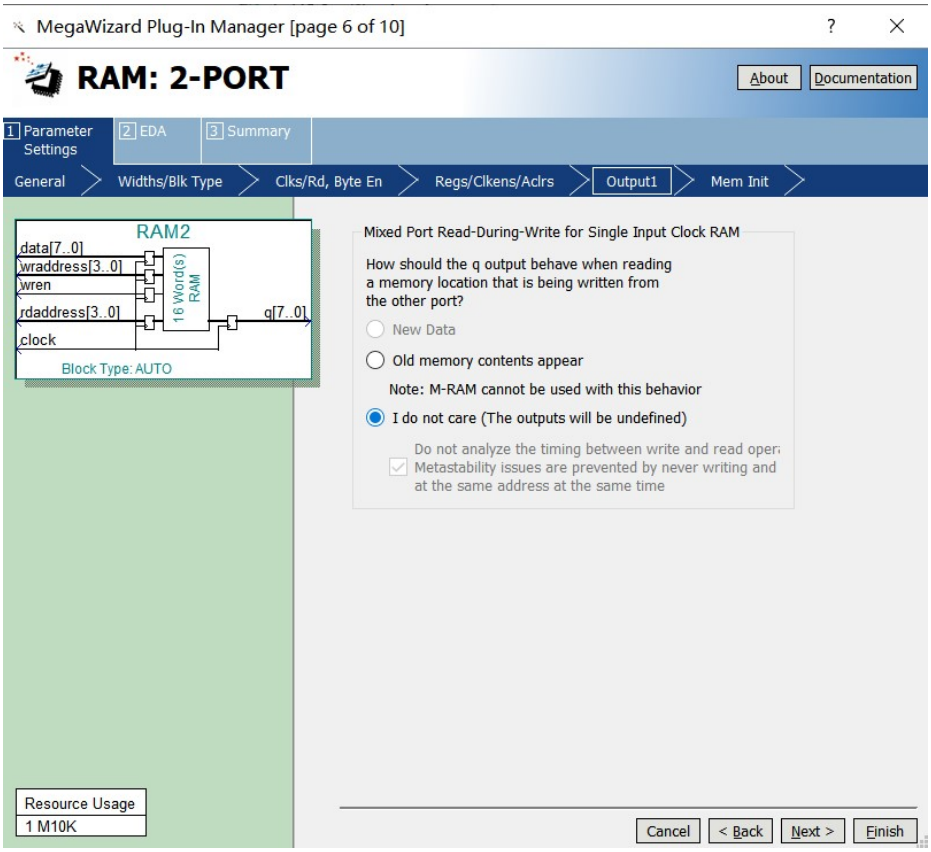


图 2-2-6：读写冲突解决

采用.mif 文件来对该存储器进行初始化.

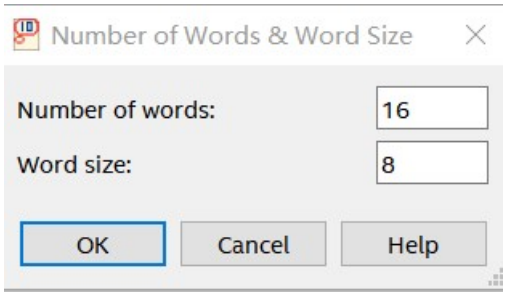


图 2-2-6：初始化文件大小选择

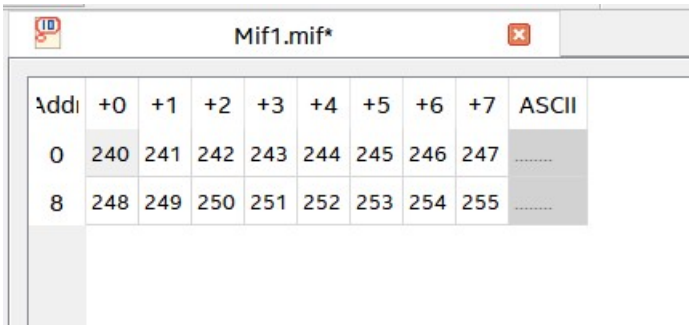


图 2-2-7：编辑初始化文件

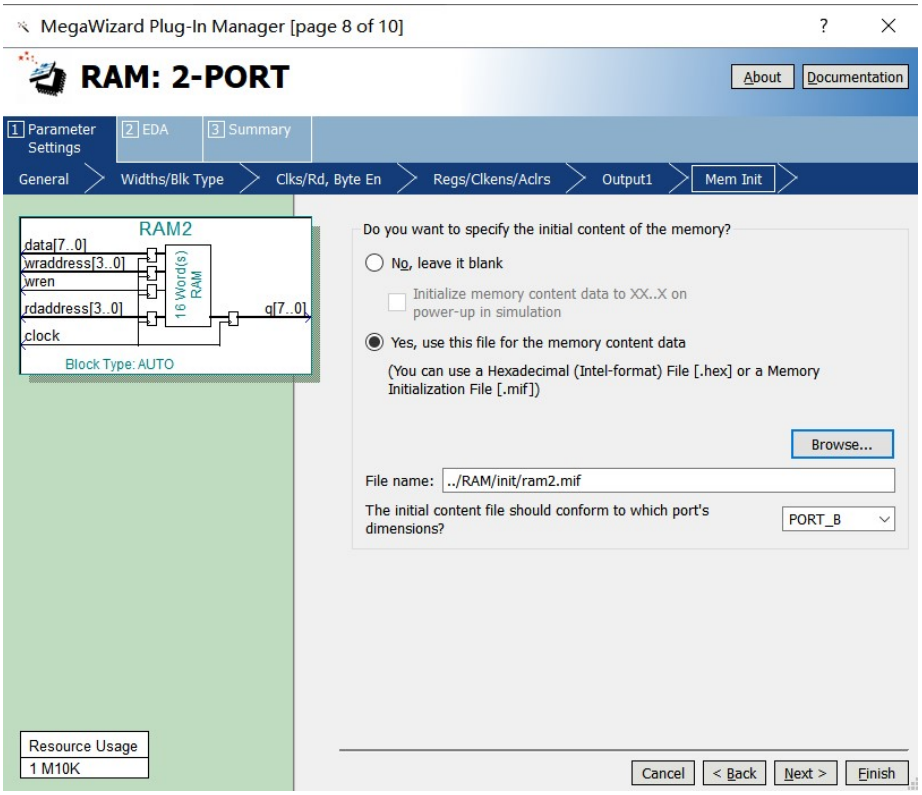


图 2-2-8：初始化文件选择

2.3 分析/综合

分析/综合实验成功，如下图所示：

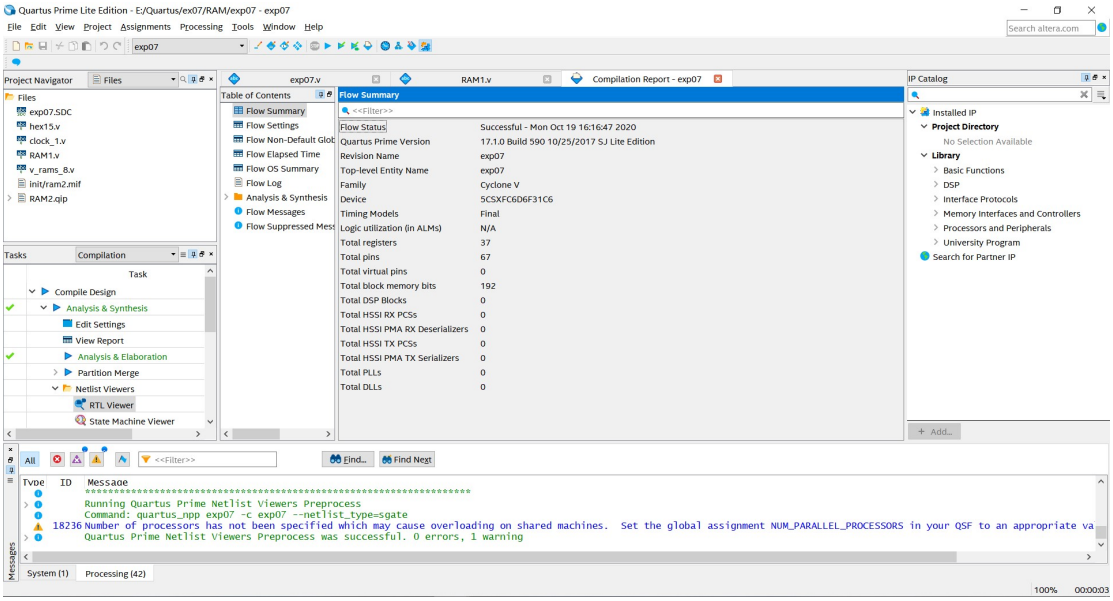
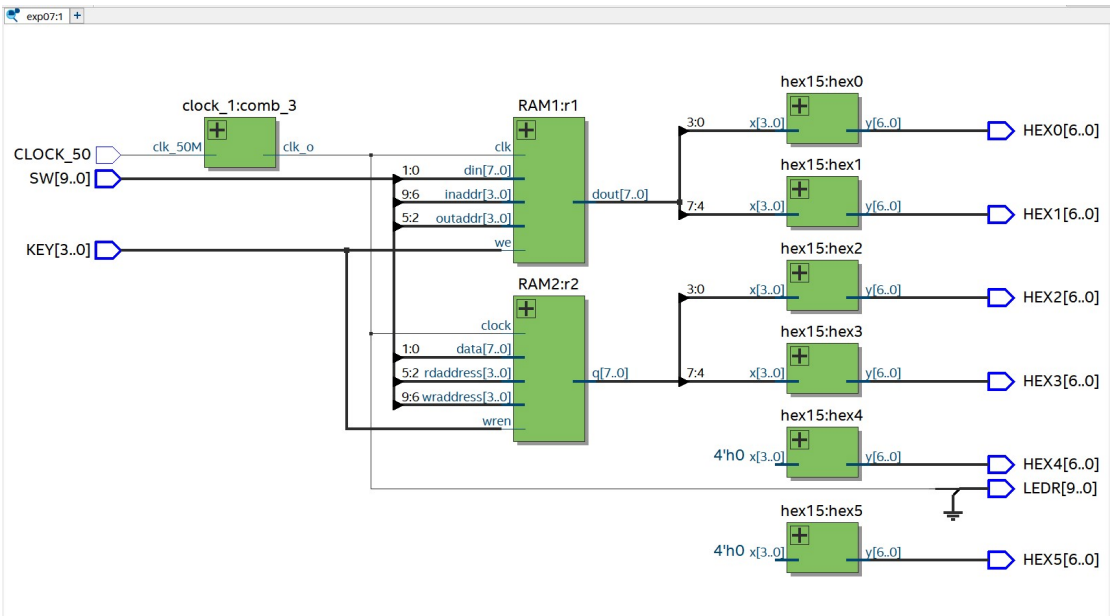


图 2-3-1：分析/综合成功



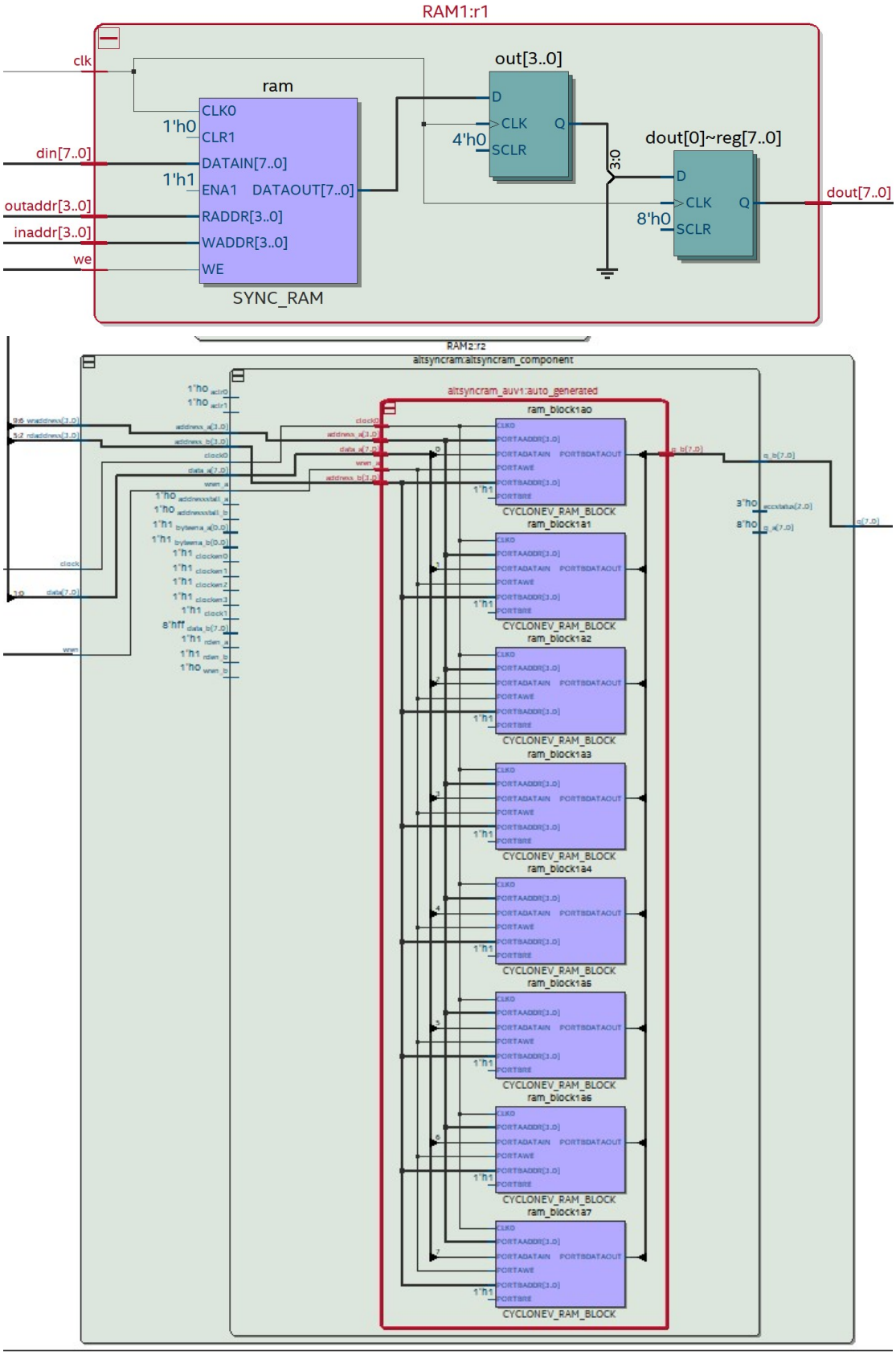
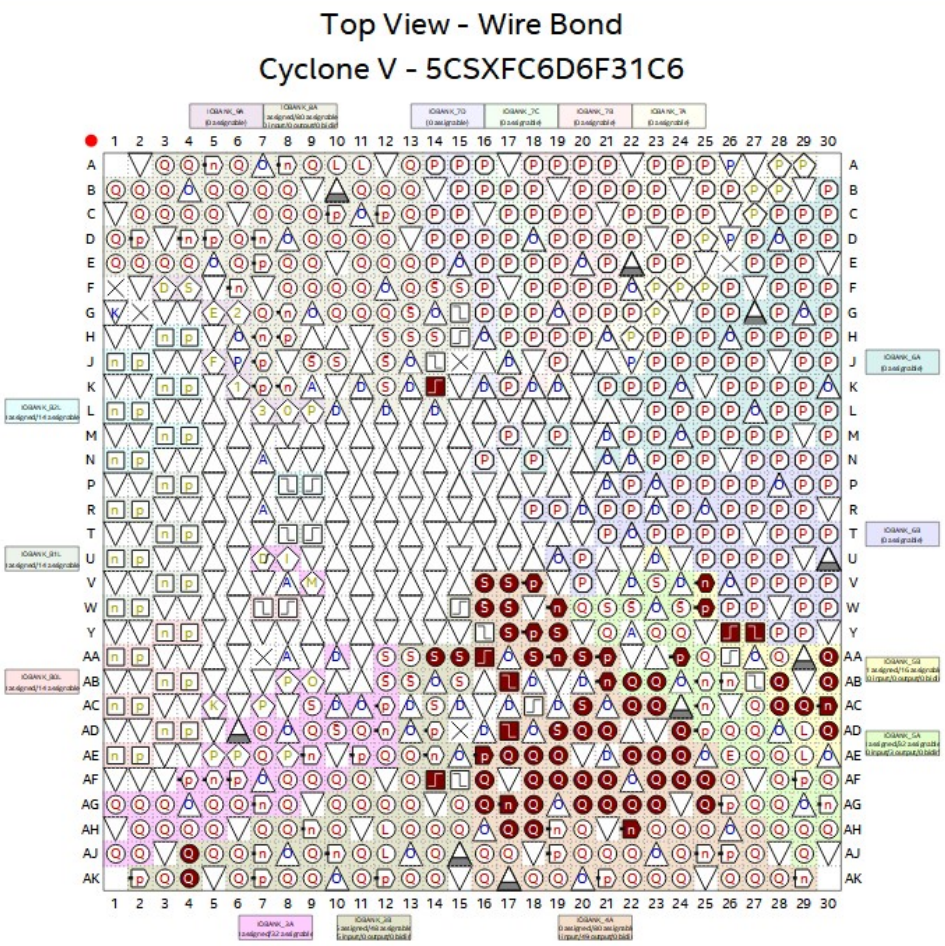

RAM1:r1
clk
din[7..0]
outaddr[3..0]
inaddr[3..0]
we
ram
CLK0
CLR1
DATAIN[7..0]
ENA1
DATAOUT[7..0]
RADDR[3..0]
WADDR[3..0]
WE
1'h0
1'h1
SYNC_RAM
out[3..0]
D
CLK
Q
SCLR
4'h0
3:0
dout[0]~reg[7..0]
8'h0
dout[7..0]
altsyncram:altsyncram_component
altsyncram_auv1:auto_generated
ram_block1a0
ram_block1a1
ram_block1a2
ram_block1a3
ram_block1a4
ram_block1a5
ram_block1a6
ram_block1a7
CYCLONEV_RAM_BLOCK

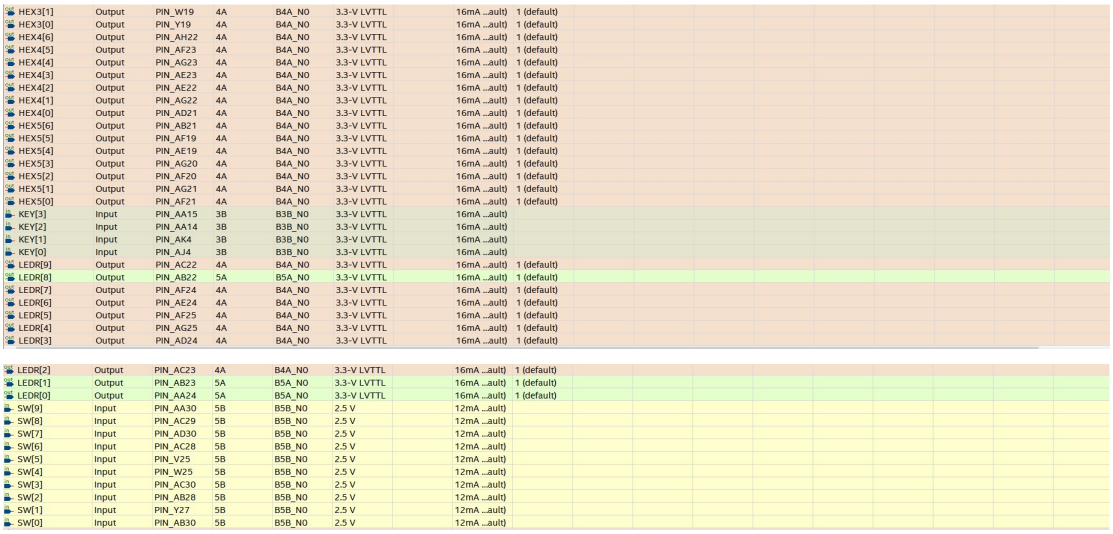
图 2-3-2:RTL 视图

2.4 分配引脚

引脚分配使用 DE10_Standard_SystemBuilder 生成。



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	rrent Streng	Slew Rate	fferential Pai	Analog Settir	(B/VCCT_GXE	er I/O Pin Tei	icated Refclk	mmon Mode	tter Slew Rate	Differential Ou	er Common I
CLOCK_50	Input	PIN_AF14	3B	B3B_N0	3.3-V LVTTL		16mA ...ault)										
HEX0[6]	Output	PIN_AH18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[5]	Output	PIN_AG18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[4]	Output	PIN_AH17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[3]	Output	PIN_AG16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[2]	Output	PIN_AG17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[1]	Output	PIN_V18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX0[0]	Output	PIN_W17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[6]	Output	PIN_V17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[5]	Output	PIN_AE17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[4]	Output	PIN_AE18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[3]	Output	PIN_AD17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[2]	Output	PIN_AE16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[1]	Output	PIN_V16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX1[0]	Output	PIN_AF16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[6]	Output	PIN_W16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[5]	Output	PIN_AF18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[4]	Output	PIN_Y18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[3]	Output	PIN_Y17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[2]	Output	PIN_AA18	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[1]	Output	PIN_AB17	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX2[0]	Output	PIN_AA21	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX3[6]	Output	PIN_AD20	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX3[5]	Output	PIN_AA19	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX3[4]	Output	PIN_AC20	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX3[3]	Output	PIN_AA20	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
HEX3[2]	Output	PIN_AD19	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									



HEX3[1]	Output	PIN_W19	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX3[0]	Output	PIN_Y19	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[6]	Output	PIN_AH22	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[5]	Output	PIN_AF23	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[4]	Output	PIN_AG23	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[3]	Output	PIN_AE23	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[2]	Output	PIN_AE22	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[1]	Output	PIN_AG22	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX4[0]	Output	PIN_AD21	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[6]	Output	PIN_AB21	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[5]	Output	PIN_AF19	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[4]	Output	PIN_AE19	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[3]	Output	PIN_AG20	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[2]	Output	PIN_AF20	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[1]	Output	PIN_AG21	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
HEX5[0]	Output	PIN_AF21	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
KEY[3]	Input	PIN_AA15	3B	B3B_N0	3.3-V LVTTL	16mA ...ault)	
KEY[2]	Input	PIN_AA14	3B	B3B_N0	3.3-V LVTTL	16mA ...ault)	
KEY[1]	Input	PIN_AK4	3B	B3B_N0	3.3-V LVTTL	16mA ...ault)	
KEY[0]	Input	PIN_AJ4	3B	B3B_N0	3.3-V LVTTL	16mA ...ault)	
LEDR[9]	Output	PIN_AC22	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[8]	Output	PIN_AB22	5A	B5A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[7]	Output	PIN_AF24	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[6]	Output	PIN_AE24	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[5]	Output	PIN_AF25	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[4]	Output	PIN_AG25	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[3]	Output	PIN_AD24	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[2]	Output	PIN_AC23	4A	B4A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[1]	Output	PIN_AB23	5A	B5A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
LEDR[0]	Output	PIN_AA24	5A	B5A_N0	3.3-V LVTTL	16mA ...ault)	1 (default)
SW[9]	Input	PIN_AA30	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[8]	Input	PIN_AC29	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[7]	Input	PIN_AD30	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[6]	Input	PIN_AC28	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[5]	Input	PIN_V25	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[4]	Input	PIN_W25	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[3]	Input	PIN_AC30	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[2]	Input	PIN_AB28	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[1]	Input	PIN_Y27	5B	B5B_N0	2.5 V	12mA ...ault)	
SW[0]	Input	PIN_AB30	5B	B5B_N0	2.5 V	12mA ...ault)	

图 2-5 脚分配图

2.5 全编译

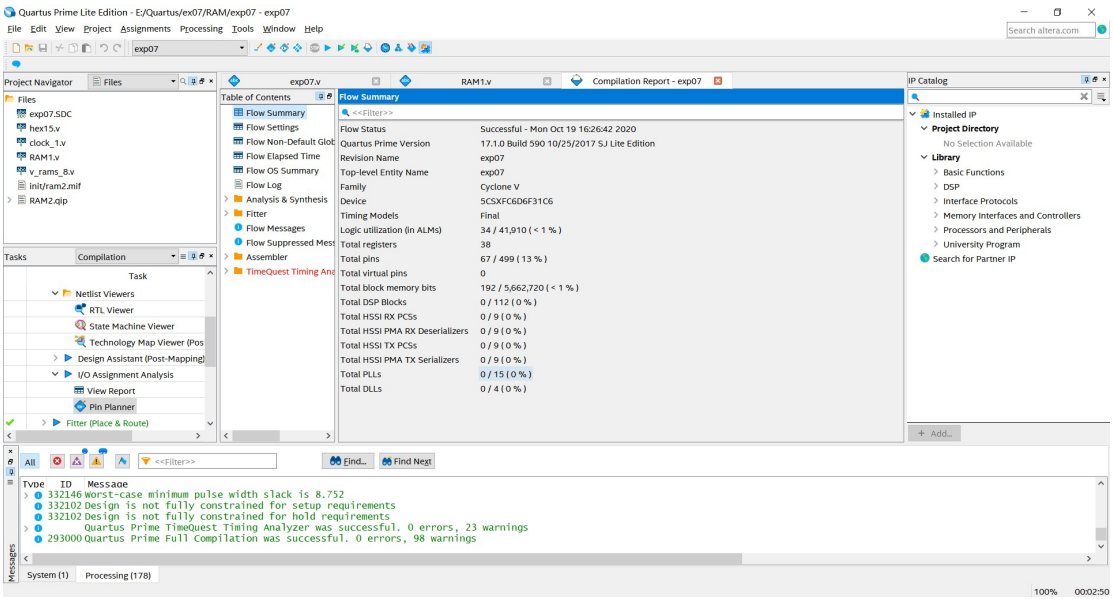


图 2-6 全编译成功

三、实验总结

本次实验主要学习实践了存储器的设计、应用.还学习了不同的初始化方式——代码内初始化、外部 txt 文件初始化和.mif 文件初始化. 通过对 always 语句和 if 语句的组合可以设计出不同读写模式的存储器.

本次实验还学习实践了使用 IP 核设计双口存储器,相比使用 Verlilog 语言来实现，该方法更加快捷、便利.

四、附

4.1 关于思考题

如果将表 7 2 中存储器实现部分改为

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

该存储器的行为是否会发生变化？

答：

该存储器的行为会发生改变.

修改部分对应 exp07.pdf 表 7-2 中的部分 Verilog 代码为：

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

assign dout = ram[outaddr];
```

实现的功能为，若写使能(we)有效(值为 1),在时钟上升沿将 din 写入存储器的写地址(inaddr)对应内存单元中. dout 的值在时钟的任何时刻都是读地址(outaddr)对应内存单元中的值.

修改后的代码实现的功能是：写使能有效时在时钟上升沿将 din 写入存储器的 inaddr 对应的地址,这一点和修改前一致，但 dout 的值在写使能无效且时钟上升沿到来时才改变为 outaddr 对应存储器地址中的值,这是和原来代码实现功能的不同之处.

4.2 关于存储器实例分析

exp07.pdf-7.2.1 节中表 7-3 给出的存储器实例对应的 RTL 图如下：

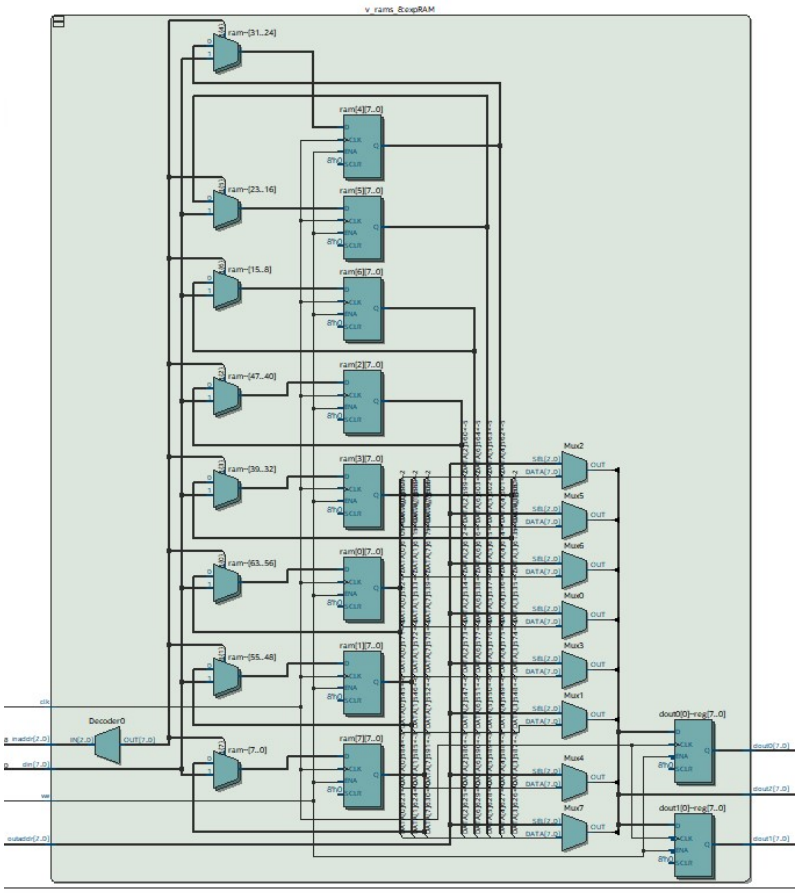


图 4-2-1 实例分析-RTL 视图

通过分析 xp07.pdf-表 7-3 中的 dout0, dout1, dout2 的赋值方式和 RTL 视图中对 dout0, dout1, dout2 输出的处理可以发现，dout0 输出只有在写使能无效且时钟上升沿到来时才会改变为读地址输入(outaddr)对应内存地址中的数据;dout1 输出只有在写使能无效且时钟下降沿到来时才会改变为读地址输入(outaddr)对应内存地址中的数据;dout2 的赋值方式为阻塞式赋值，其变化与时钟变化无关，能直接得到读地址输入(outaddr)对应内存地址中的数据. 数据的输入则是在写使能信号有效且时钟上升沿到来时才会

该实例通过引脚约束后再开发板上验证得到的输入/输出时序与上面所述相同.

4.3 1Hz 分频器的 Verilog 实现

参照 exp05.pdf 表 5-2 秒时钟生成代码，本实验所用到的 1Hz 分频器的实现如下：

```
1   module clock_1(clk_50M, clk_o);
2       input clk_50M;
3       output reg clk_o;
4       integer cnt;
5
```

```
6       always @(posedge clk_50M) begin
7          if(cnt == 25000000) begin
8                cnt <= 0;
9                clk_o <= ~clk_o;
10            end
11            else cnt <= cnt + 1;
12      end
13   endmodule
```

4.4 七段数码管编码器的 Verilog 实现

```
1   module hex15(x, y);
2      input [3:0]x;
3      output reg [6:0]y;
4
5      always @(*) begin
6         case(x)
7            4'b0000:y = 7'b1000000;
8            4'b0001:y = 7'b1111001;
9            4'b0010:y = 7'b0100100;
10           4'b0011:y = 7'b0110000;
11           4'b0100:y = 7'b0011001;
12           4'b0101:y = 7'b0010010;
13           4'b0110:y = 7'b0000010;
14           4'b0111:y = 7'b1111000;
15           4'b1000:y = 7'b0000000;
16           4'b1001:y = 7'b0010000;
17           4'b1010:y = 7'b0001000;
19           4'b1011:y = 7'b0000011;
20           4'b1100:y = 7'b1000110;
21           4'b1101:y = 7'b0100001;
22           4'b1110:y = 7'b0000110;
23           4'b1111:y = 7'b0001110;
24           default: y = 7'b1111111;
25        endcase
26     end
```