



计算机体系设计实践报告

2022 学年秋季学期

序号	学号	姓名	专业	班级	成绩
	20211120138	薛凡豪			
	20211120148	罗安乔			

指导教师：

软件学院 计算机体系设计实践 王道

二〇二一年九月

计算机体系设计实践

实验目的

实践课设计4个实验,基本涵盖了计算机体系结构与组成课程的重点内容,实验电路的设计与教材保持一致,实验电路在FPGA实验板上实现,并使用调试软件JULAB完成实验的调试与分析。

实验要求

实验课前复习相关原理,按实验指导书的要求认真预习。实验时独立思考,掌握实验设备或软件的构造和操作方法,按实验指导书要求设计或验证实验内容,测试有关数据,分析相应的问题。

实验课结束时需要整理提交 fpga 工具箱,并检查工具箱附件是否缺失。

由于fpga工具箱数量有限,实验以小组形式完成,2人1个小组,提交1份实验报告。

实验资料环境和资料

Win10+FPGA设计软件Quartus II、实验调试软件Julab、实验相关设计文件(工程模板、电路设计源文件、虚拟构图文件)等。请找实验指导老师索取。

实验地点

计算机硬件实验室:软件学院大楼1428室

实验2 高速缓冲存储器

实验操作

1. 下载实验资源

将通用文件“DE2-115_proj”解压缩到E 盘或F 盘。得到DE2-115 工程文件夹。将高速 缓冲存储器中文件解压到DE2-115 工程文件夹中；

实验电路顶层文件*Lab_Top.v*、地址译码*Decoder.v*、地址寄存器*R.v*、多路器*MUX.v*、VALID 模块*ram_valid.v*。

lab5.vpl 和 *lab5.bmp* 是留给实验调试软件使用的虚拟面板构图文件。

init_mm.mif 是主存MM 内容的初始化文件。

Q12 文件夹里是使用FPGA 内部的RAM 资源设计的主存MM 模块*ram_mm.v*、CACHE 模块*ram_cache.v*、标志存储器TAG 模块*ram_tag.v* 以及它们各自的IP 核文件。

2. 实验电路设计与下载

在工程文件夹DE2-115 中双击工程文件DE2_115_Lab.qpf 打开实验电路的QuartusII 工程。

点击工具栏中分析与综合（Start Analysis & Synthesis）按钮，检查语法错误，参阅实验 指导书第五章5.1.1 设计流程的“分析综合”。

分析综合通过后，直接点击工具栏中的全编译（Start Compilation）按钮，自动完成分析 综合、布局布线、生成编程文件等整个过程，全编译完成后，点击工具栏中的编程按钮（Programmer），将生成的实验电路文件DE2_115_Lab.sof 下载到实验板。

3. 实验电路功能验证

打开实验调试软件JULAB3，选择逻辑部件实验类型，在“虚拟实验板”菜单的面板构 图选项下，浏览选择工程文件夹中的*lab5.vpl* 文件，打开本实验的虚拟面板，根据实验原理， 控制虚拟面板的开关、按键，观察对应的指示灯，在实验报告册中填写实验结果记录和分析。

本实验验证时需要使用 QuartusII 软件的在系统存储器数据编辑器（In-System Memory Content Editor）， 实时查看和修改标志存储器TAG、高速缓存存储器CACHE 和主存MM 的内容，In-System Memory Content Editor 的更多使用方法，参阅实验指导书第五章 5.1.3 在系统存储器数据编辑器。

实验记录

1. 主存地址格式各部分的位数。

AR	TAG	BLOCK	WORD
----	-----	-------	------

2. 初始状态






使用Quartus II 的In-System Memory Content Editor 查看TAG、CACHE 和MM 的内容，并对后面用到的主存 50H~53H、64H~67H、84H~87H 单元输入一些已知的内容，记录在下表中。

地址	50H	51H	52H	53H	64H	65H	66H	67H	84H	85H	86H	87H
内容	1	2	3	4	5	6	7	8	A	B	C	D

观察8 个VALID 单元的状态应都为0，如果不是，按RESET 键清零。（实验原理图上没有画出RESET 按键与VALID 模块的连接）

3. 不命中情况下CACHE 内容的装入

因为是直接映像，映射关系已经固定，主存中的某一块只能存入cache 的指定位置，所以不需要考虑替换算法，不命中时直接装入即可。装入时需要依次装入4 个字，由OFFSET 选择写入哪一个字，WR_CACHE 给出读MM 和写CACHE 的时钟。

	AB	CLK	OFFSET	WR_CACHE	MM_DATA	WR0	WR1	WR2	WR3	HIT
①	52H		—	0	—	0	0	0	0	0
②	52H	0	00		1	1	0	0	0	0
③	52H	0	01		2	0	1	0	0	0
④	52H	0	10		3	0	0	1	0	0
⑤	52H	0	11		4	0	0	0	1	1

上述操作完成后，用In-System Memory Content Editor 查看TAG 和CACHE 中变化的内容记录在下表中。

行号	TAG	DATA0	DATA1	DATA2	DATA3
4	00002000	00001000	00002000	00003000	00004000

该行的V = 1 (0/1)。

实验分析：

(1) 52H 的地址访问的是CACHE 的第 4 行第 10 个字。


(2) 当所访问的地址不命中时，需将访问地址所指向的主存块的 一个单元 (一个单元 / 所有单元) 装入CACHE。

(3) 在向 CACHE 存储器中写入第 3 (0 / 1 / 2 / 3) 个字的时候，TAG 存储器、VALID 存储器也同时写入。

4. 命中情况下CACHE 的读出

访问 50H, 51H, 52H, 53H 地址，这4 个地址对应着同一个主存块中的4 个单元，在 上一步操作中，访问52H 地址不命中后，访问地址所指向的主存块已经整个装入了CACHE 块，所以访问该主存块中的任意单元，应该都是命中的，直接从CACHE 读出。

	AB	CLK	WR_CACHE	AR- 区号	TAG	HIT	AR - 字地址	CACHE_WORD
①	50H		0	0	010	1	00	1
②	51H		0	0	010	1	01	2
















③	52H	—	0	0	010	1	10	3
④	53H		0	0	010	1	11	4

访问某一主存单元时，根据地址寄存器 AR 的 块号（区号 / 块号 / 字地址）找到 CACHE、TAG 和 VALID 的行；如果该行的 TAG 与地址寄存器 AR 的 区号（区号 / 块号 / 字地址）相同，并且 VALID = 1（0/1），则要访问的主存地址命中，判断是否命中的代 码在 **错误!未找到引用源。**的第 67 行；命中时根据地址寄存器 AR 的 字地址（区号 / 块号 / 字地址）由多路器 MUX 选择读出 CACHE 行中的哪一个字，多路器 MUX 的实例化在 **错误!未找到引用源。**的第 63 行。

5. 抖动现象

直接映像方式下，每个主存块都只有固定的一个 cache 位置可以存放，当主存地址的区 内块号相同的时候，由于对应同一个 cache 块，即便其他 cache 块都是空闲，也无法使用。

当某段时间内恰巧要访问主存不同区号但相同区内块号的两块数据时，例如下面第一个表格中 1~8 行的地址 84H~87H，与 9~16 行的地址 64H~67H，分别属于主存的第 100 区和第 011 区，区号不同，但它们的区内块号相同，都是 001，如果 CPU 交替访问这两块数据，就会出现这两块主存数据交替调入调出 CACHE 的现象，这种现象称为抖动。

	AB	CLK	OFFSET	WR_CACHE	MM_DATA	AR- 区号	AR - 块号	AR - 字地址	HIT	CACHE_WORD
1	84H		—	0	—	100	001	00	0	—
2	84H	0	00		A	（同上）	（同上）	（同上）	0	A
3	84H	0	01		B	（同上）	（同上）	（同上）	0	—
4	84H	0	10		C	（同上）	（同上）	（同上）	0	—
5	84H	0	11		D	（同上）	（同上）	（同上）	1	—
6	85H		—		—	100	001	01	1	B
7	86H		—	0	—	100	001	10	1	C
8	87H		—	0	—	100	001	11	1	D
9	64H		—	0	—	011	001	00	0	—
10	64H	0	00		5	（同上）	（同上）	（同上）	0	
11	64H	0	01		6	（同上）	（同上）	（同上）	0	—
12	64H	0	10		7	（同上）	（同上）	（同上）	0	—
13	64H	0	11		8	（同上）	（同上）	（同上）	1	—
14	65H		—		—	011	001	01	1	6
15	66H		—	0	—	011	001	10	1	7
16	67H		—	0	—	011	001	11	1	8
17	84H		—	0	—	100	001	00	0	—

	AB	CLK	OFFSET	WR_CACHE	MM_DATA	AR- 区号	AR - 块号	AR - 字地址	HIT	CACHE_WORD

实验小结及实验分工

一、出现的问题

1. 在查看 TAG、CACHE、MM 内容是一开始在 In-system Memory Content Editor 打开 “在系统存储器数据编辑” 时并没有出现任何内容，后来在硬件设置项选择 USB-Blaster 后解决。
2. 选择需要观察的存储器，右键菜单中选择 “Read data from in-system memory” 后看不到预设的初始数据值，后来双击 ram_mm 模块，在插件管理器中打开主存模块设置，点击 Browse...按钮，找到工程路径下的 init_mm.mif 文件修改并重新编译工程、下载后解决问题。

二、收获和心得

1. CACHE 共 32 个字，直接映像，分为 8 个 BLOCK（块），每块 4 个 WORD（字）。
2. 主存地址空间 256 个字，按 CACHE 大小区分，即每区 32 字，共分为 8 个区。
3. 对于不命中情况下 CACHE 内容的装入、命中情况下 CACHE 内容的读出、抖动现象以及每个灯表示的意义都有了详细的了解和掌握。