



東南大學
SOUTHEAST UNIVERSITY

網絡空間安全學院
School of Cyber Science and Engineering

计算机组成原理 实验讲义

2020 年 9 月

目录

实验概要	1
1 实验目标	1
2 实验环境	1
2.1 软件安装	1
2.2 新建工程	1
2.3 结果验证	3
3 DEMO_IS 指令系统	3
实验一 寄存器组的设计	5
1 实验目的	5
2 实验内容	5
3 实验原理及方案	5
3.1 D 触发器功能的测试	5
3.2 寄存器组的设计、实现及验证	7
4. 实验要求	8
实验二 ALU 的设计	9
1 实验目的	9
2. 实验内容	9
3. 实验原理及方案	9
3.1 加/减法器功能的测试	9
3.2 ALU 的设计、实现及验证	9
4. 实验要求	11
实验三 存储器设计及总线互连	12
1 实验目的	12
2 实验内容	12
3 实验原理及方案	12
3.1 RAM 功能的测试	12
3.2 存储模块的设计、实现及验证	12
3.3 部件通过总线互连的设计、实现与验证	14
4 实验要求	15
实验四 数据通路的组织	16
1 实验目的	16
2 实验内容	16
3 实验原理及方案	16
3.1 指令功能分析	16
3.2 数据通路的设计与实现	16
3.3 数据通路的验证	17
4 实验要求	18

实验概要

计算机组织与结构的课程实验是为巩固教学效果而设置的，目的是希望学生通过实验，能够加深对计算机组成及工作原理的理解，熟练数字电路芯片的使用，提高数字电路的逻辑设计能力。

计算机组织与结构的课程实验共包含 4 个实验，分别是寄存器组设计、ALU 设计、存储器设计及总线互连、数据通路组织。

所有实验都要求基于 Quartus II 进行电路实现和正确性验证，要求采用原理图方式实现电路，采用功能（或时序）仿真方式进行仿真，条件允许时基于 FPGA 芯片进行电路验证。实验讲义基于 Quartus II 撰写。

为了减少实验所花时间，实验四需要使用前三个实验的结果，因此，四个实验应使用同一个工程文件，所有文件放在同一个文件夹下。

1 实验目标

- 实现小型指令集 Demo_IS 的主机系统（CPU+内存）
- 掌握软件程序运行在硬件上的工作原理
- 掌握 EDA 工具的电路设计和仿真方法

2 实验环境

利用计算机辅助设计软件，来完成超大规模集成电路芯片的功能设计、综合、验证、物理设计（包括布局、布线、版图、设计规则检查等）等流程的设计，称为电子设计自动化（Electronic Design Automation，简称 EDA）。

本实验将采用 Altera 公司推出的 EDA 工具软件 Quartus II 完成，该软件支持原理图（Block Diagram/Schematic）和硬件描述语言（VHDL、Verilog HDL 和 AHDL 等）等多种设计输入形式，内嵌自有的综合器和仿真器，可以完成从设计输入、仿真测试以及硬件配置的完整设计过程。

2.1 软件安装

- 实验室计算机已安装 Quartus II 软件，可直接使用。
- 个人电脑安装需要安装的，可搜索网络资源，安装过程参考相关文档。

2.2 新建工程

利用 Quartus II 可以进行电路设计，下面以 $Y = A \cdot B$ 的与运算电路为例介绍一个电路设计工程的基本过程，主要包括：①建立工程文件；②编辑原理图/用硬件描述语言编写程序；③编译原理图/硬件源程序。

Quartus II 电路设计的示例操作过程如下：

（1）建立工程文件：主菜单→File→New Project Wizard，包含 5 个页面的设置

第 1 页，设置工程信息，含目录（如 E:\Workspace\Quartus\Demo）、工程名（如 Demo）；

第 2 页，工程加入已有文件，可将已有的原理图/源程序文件添加到工程中，本例中无；

第 3 页，选择元器件型号，建议采用 Cyclone III 系列的 EP3C16Q240C8 芯片；

第 4 页，添加和配置准备使用的 EDA 工具，通常选择默认，本例中直接选择 Next；

第 5 页，确认工程文件信息，如图 1 所示，无误选择 Finish 即可。

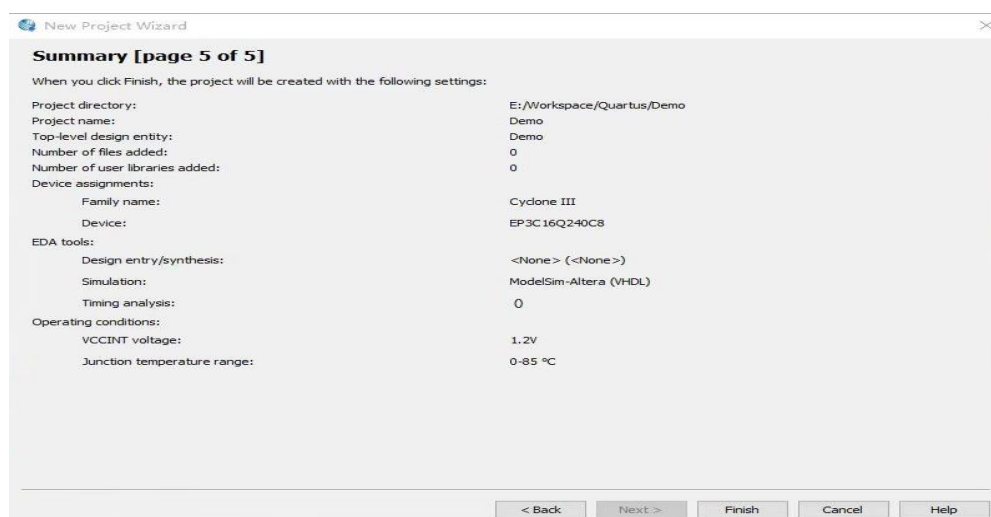


图 1 Quartus II 新建工程信息确认

(2) 编辑原理图文件

通过主菜单→File→New...→Device Design Files→Block Diagram/Schematic File（根据 Quartus 软件版本不同，路径可能为：File→New...→Design Files→Block Diagram/Schematic File）：

- ①进入到原理图编辑器，通过主菜单→File→Save 保存为原理图文件 demo_and.bdf；
- ②双击空白处出现 Symbol 窗口，展开 Libraries，选择需要门电路、引脚等，本例包括 1 个与门（libraries->primitives->logic->and2）、2 个输入（libraries->primitives->pin->input）和 1 个输出（libraries->primitives->pin->output），如图 2 所示。

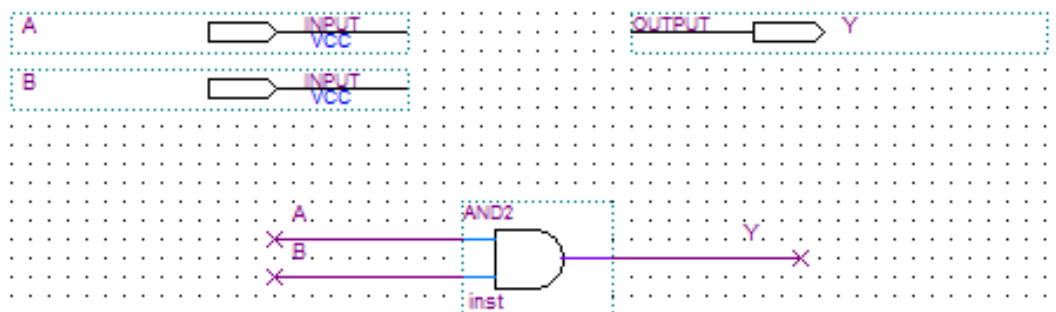


图 2 Quartus II 编辑与门电路原理图

注：门电路上的引线名与输入/输出引脚同名则表示电路相连

(3) 编译原理图

①设置顶层文件，在 Project Navigator 窗口中的 File 页面中，选择需要编译的原理图文件，点击该文件右键菜单 Set as Top-Level Entity，本例中将 demo_and.bdf 文件设置为顶层文件；

②编译顶层文件，主菜单→Processing→Start Compilation，编译时提窗口将显示编译的相关信息，包括错误和告警。若编译无误，则进入结果验证阶段。编译有错误时，修改原理图文件，重新编译；编译成功后，可进入仿真阶段。

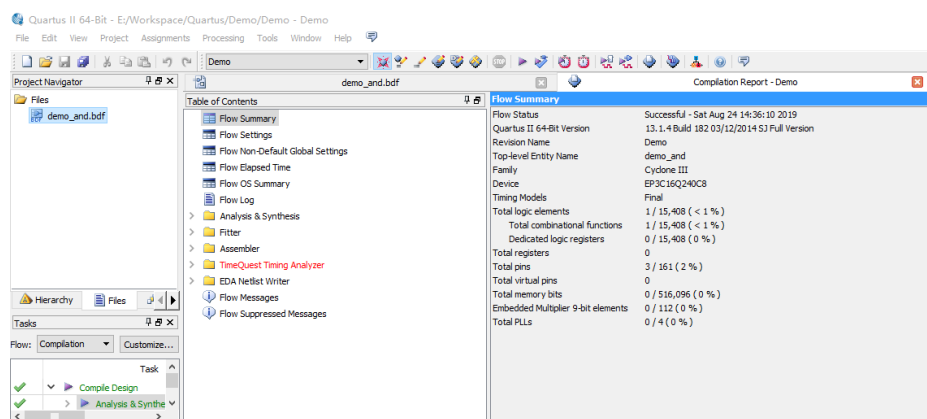


图 3 Quartus II 编译与门电路原理图结果

2.3 结果验证

利用 Quartus II 可以进行仿真验证，下面仍以 $Y = A \& B$ 的与运算电路为例介绍仿真验证的基本过程，主要包括：①新建仿真波形文件；②选择仿真方式开始仿真；③分析仿真结果。

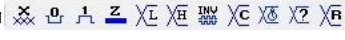
Quartus II 仿真验证的示例操作过程如下：

（1）新建仿真波形文件

通过主菜单 **File**→**New...**→**Other Files**→**Vector Waveform File**（根据 Quartus 软件版本不同，路径可能为：**File**→**New...**→**Verification/Debugging Files**→**University Program VWF**）：

①进入到**当前顶层文件**的波形文件编辑器，通过 **Edit**→**Insert**→**Insert Node or Bus...**，点击 **Node Finder**，在新窗口中点击 **List** 按钮，将引脚 A、B 和 Y 全部选中到右侧 **Selected Nodes** 框中，点击 **OK**；

②保存波形文件，通过主菜单→**File**→**Save** 保存为 **demo_and.vwf**；

③设置输入值，选中输入引脚（A 或 B），点击上方菜单中  选择需要设置的值，本例选择 **Overwrite Clock** 设置 A 和 B 的值。

注意，一个原理图文件可以对应多个波形文件，仿真时可以进行选择；波形文件中的信号组合，应覆盖电路功能表的全部功能。

（2）选择仿真方式开始仿真，仿真方式分为功能仿真（**Functional Simulation**）和时序仿真（**Time Simulation**），前者仅考虑功能性验证，后者还考虑元器件的电路时延。本例中选择功能仿真，即点击 **Simulation**→**Run Functional Simulation**

（3）分析仿真结果，对照所仿真电路的功能表，分析电路的输出结果是否正确，本例的功能仿真结果如下图所示

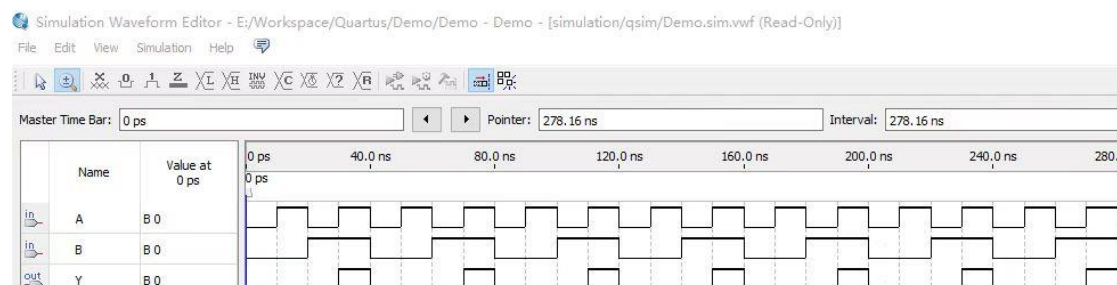


图 4 Quartus II 与门电路功能仿真结果

试试看：同样的输入，如果选择时序仿真，看看结果如何？

3 Demo_IS 指令系统

本实验将设计一个支持自定义的小型指令集（命名为 **Demo_IS**）的主机系统，分为寄存

器、运算器、存储器和数据通路四个模块，分别通过四次实验实现，最终实现完整的主机系统并运行测试程序。

Demo_IS 指令集共 8 条指令（如表 1 所示），包括 mov、ld、st、add、sub、inc、dec、jnz，分别表示赋值、读存储器、写存储器、加、减、自增、自减、结果零跳转。除 mov 和 jnz 指令各包含 1 Byte 的立即数和有效地址外，每条指令字长为 1 Byte。

表 1 本实验所实现的 Demo_IS 指令子集

Demo_IS Instructions					
Code	Byte 1			Byte 2	Specification
	op	rd	rs	immediate/address	
	7..4	3..2	1..0	7..0	
MOV	0000	rd	00	imme	$rd \leftarrow imme$
LD	0010	rd	rs	—	$rd \leftarrow M[(rs)]$
ST	0011	rd	rs	—	$M[(rs)] \leftarrow (rd)$
ADD	0100	rd	rs	—	$rd \leftarrow (rd) + (rs)$
	0101	rd	rs	—	$rd \leftarrow (rd) + M[(rs)]$
SUB	0110	rd	rs	—	$rd \leftarrow (rd) - (rs)$
INC	1000	rd	00	—	$rd \leftarrow (rd) + 1$
DEC	1001	rd	00	—	$rd \leftarrow (rd) - 1$
JNZ	1100	00	00	addr	$\text{if}(!ZF) PC \leftarrow addr$
	1101	disp		—	$\text{if}(!ZF) PC \leftarrow (PC) + disp$

由表 1 可以看出：

- (1) 指令操作类型均是通过操作码 op 来区分。
- (2) 地址码部分最多包含两个寄存器地址，即 rd（目的寄存器）和 rs（寄存器）；寄存器地址长度最大为 2 bits，即支持 4 个寄存器。
- (3) 本指令系统为变长指令系统，其中，mov 指令中的 imme 表示立即数，jnz 指令中 addr 表示有效地址。

实验一 寄存器组的设计

1 实验目的

- (1) 熟悉 D 触发器的功能及使用方法。
- (2) 掌握寄存器组的组成原理。

2 实验内容

- (1) 测试 D 触发器的功能。
- (2) 设计具有 1 个读端口、1 个写端口的 4×8 位寄存器组，并验证设计正确性。

3 实验原理及方案

3.1 D 触发器功能的测试

触发器有边沿触发、电位触发两种方式，通常，前者称为触发器，后者称为锁存器。触发器的状态在时钟脉冲信号 CP 上升沿时发生变化（触发）、在其余时间保持不变；锁存器的状态在控制信号 E 为高电平时跟随输入端发生变化、为低电平时保持不变（锁存）。

Quartus II 提供了多种类型的触发器，如 D 触发器、T 触发器等。每种类型的触发器有引脚固定、引脚可变（又称参数化）两类，如 74173、74273、lpm_ff、lpm_dff、lpm_tff 等，参数化触发器的参数不同、型号不同，如 lpm_dff0、lpm_dff1 等，其中序号是自动生成的。

- (1) D 触发器的功能与引脚

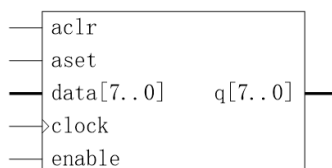


图 5 lpm_dff 引脚示例

图 5 为 8 位 D 触发器 lpm_dff 的 I/O 引脚示例。其中，data[7..0]、q[7..0] 分别为数据输入、数据输出引脚，aclr、aset 分别为异步的清零、置位引脚（高电平有效），同步的清零、置位引脚为 sclr、sset，clock 为时钟脉冲引脚，enable 为写使能（即 clock 使能）引脚（高电平有效）。lpm_dff 的状态（所存储信息）由 q[7..0] 直接输出，无需信号控制。

在 Quartus II 中，通过双击原理图空白处，出现 Symbol 窗口，展开 Libraries→megafuncions→storage，可找到 lpm_dff，如下图（图 6）所示：

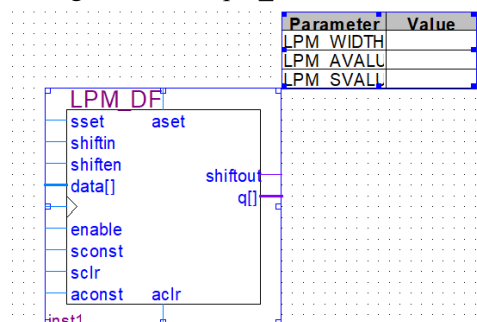


图 6 Quartus II 中的 lpm_dff 器件

对 `lpm_dff` 的操作有清零、置位、写入三种。异步清零 (`aclr=1`、`aset=0`) 时, `q` 立即变为全 0; 异步置位 (`aclr=0`、`aset=1`) 时, `q` 立即变为全 1。写操作 (`enable=1`、`aclr=0`、`aset=0`) 时, `data` 上信号在 `clock` 上升沿时写入触发器。

实际应用中,通常会省略不使用的引脚,如 `aset`、`sclr`、`sset` 等引脚,缺省引脚的信号在芯片内部都处于无效状态。不建议省略 `enable` 引脚,以避免 `q` 端产生毛刺。

省略不使用的引脚、设置寄存器位数: 选中该器件并右击,在弹出的窗口中选择底部的“Properties”,在弹出的 Symbol Properties 窗口中,在 Ports 栏,将不需要的引脚置为 Unused; 在 Parameter 栏中,修改 `LPM_WIDTH` 的值来设置寄存器的位数 (8 位),如下图 (图 7) 所示。

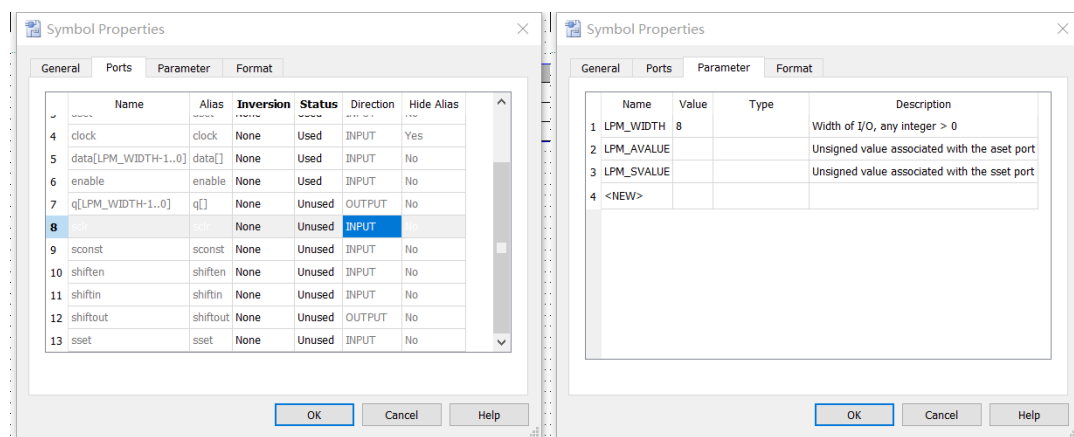


图 7 `lpm_dff` 的 Properties 窗口

(2) D 触发器功能的测试

触发器功能的测试包括电路实现、电路仿真及结果分析几个步骤,仿真时先采用功能仿真方式得到结果并分析,再采用时序仿真方式查看器件操作的时延特征。

电路实现需要编辑原理图文件 (如 `test_dff.bdf`), 电路由一个 `lpm_dff`、若干输入引脚及输出引脚组成,如图 8 所示。注意,信号线的连接有多种实现方法,如连线、信号线同名等;信号线经过器件后即可重新命名,器件 `wire` 不进行任何操作 (可用于重命名);总线信号线中部分信号线的使用方法与 `verilog` 语言相同,如 `DIn[5]`、`DIn[2..0]`。

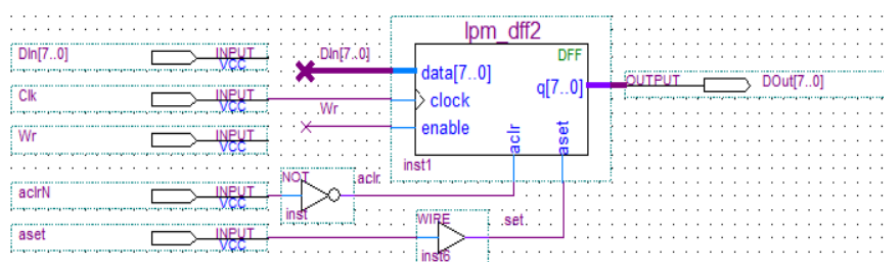


图 8 触发器测试电路的组成示例

电路仿真需要先建立仿真波形文件 (如 `test_dff.vwf`), 再设置各输入信号在不同时刻的取值, 输入/输出信号应包含电路的所有引脚, 输入信号的组合应能够反映电路功能表的所有功能。注意, 时序逻辑电路中, 输入信号的时长应是时钟周期的倍数, 初始化信号除外; 时钟周期的宽度应大于所有操作的时延, 如 `lpm_dff` 时延约 5ns, 时钟周期可设置为 20ns。

实际应用中, 时钟周期的开始都用时钟脉冲信号的上升沿来标志, 输入信号的长度为时钟周期的倍数、状态变化滞后于时钟脉冲信号, 因为许多信号通过时钟脉冲信号产生。为了

使功能仿真方式、时序仿真方式的结果相同，要求时钟脉冲信号的上升沿略早于输入信号的改变，如图 9 所示。实现方法是，设置时钟脉冲信号时，修改 Time period 参数的 offset 值。

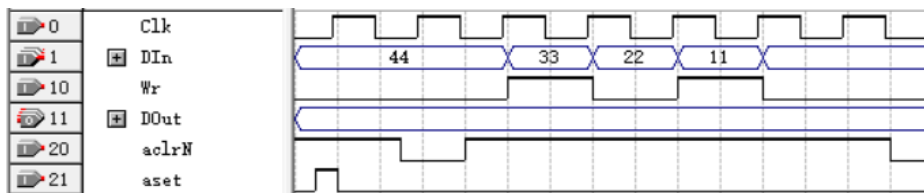


图 9 触发器测试电路的信号波形示例

分析仿真结果时，应以时钟周期为单位进行分析，查看每个输出信号的变化与输入信号的关联，分析是否满足电路功能表的要求。

3.2 寄存器组的设计、实现及验证

寄存器组将多个寄存器组织在一起，按地址进行操作，以简化控制。本实验要求的寄存器组包含 4 个 8 位寄存器，具有 1 个读端口、1 个写端口。

(1) 寄存器组的设计

由于寄存器组包含 4 个 8 位寄存器，故操作时的地址引脚为 $\log_2 4 = 2$ 位、数据引脚为 8 位。由于要求读/写端口分离，寄存器所存信息的输出无需信号控制，故寄存器组可同时进行读、写操作，读操作相关引脚为地址引脚 `raddr[1..0]`、数据输出引脚 `q[7..0]`，写操作相关引脚为地址引脚 `waddr[1..0]`、数据输入引脚 `data[7..0]`、写使能引脚 `wen`、时钟脉冲引脚 `Clk`，清零操作相关引脚为清零引脚 `Clr`。

寄存器组中，写操作通过 `waddr[1..0]` 指定目标寄存器，通过 `wen`、`Clk` 实现写入控制（边沿触发），寄存器的选择可使用译码器来实现；读操作通过 `raddr[1..0]` 选择目标寄存器，寄存器输出无需信号控制，数据输出的选择可使用选择器来实现；清零操作通过 `Clr` 来控制。寄存器组的内部组成如图 10 所示，假设 `wen`、`Clr` 都是高电平有效。

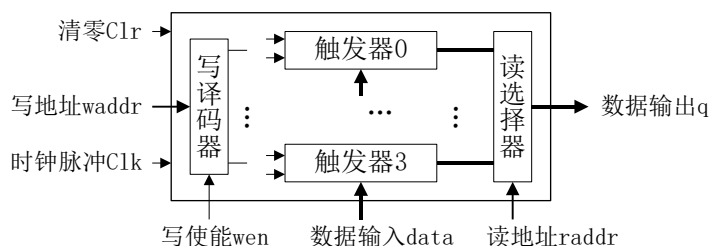


图 10 寄存器组的组成

(2) 寄存器组的实现与验证

寄存器组的实现需要编辑原理图文件（如 `GPRs.bdf`），电路有多种实现方法，译码器、选择器可使用 Quartus II 提供的参数化模块 `lpm_decode`、`lpm_mux`。注意，由字符及数字组成的多个相似信号线命名时，数字位数应相同（如 `wren01` 与 `wren21`）；添加 `lpm` 器件时，应从本工程文件的工具条 `symbol Tool` 中选择，或 `.bdf` 文件中拷贝，不能从其他工程文件的 `.bdf` 文件中拷贝，否则编译时会产生错误（`lpm` 器件未定义或定义冲突），错误修改方法是手工选择的 `lpm` 器件替换 `.bdf` 中所有的 `lpm` 器件。

寄存器组的验证包括电路仿真、结果分析两个步骤，仿真采用功能仿真方式进行，可以再采用时序仿真查看器件操作的时延特征。

电路仿真需要先建立仿真波形文件（如 `GPRs.vwf`），再设置 `waddr`、`raddr`、`data` 及各控

制信号在不同时刻的取值，输入信号的组合需能够反映电路的所有功能特性。注意，读、写操作的测试次数应 ≥ 2 ，以避免错误现象被隐藏；所有输入信号的时长都应与时钟周期为单位，输入信号的改变应滞后于时钟脉冲信号 Clk 的上升沿。

分析仿真结果时，应以时钟周期为单位进行分析，查看每个输出信号的状态及时序是否满足电路功能表的要求。

4. 实验要求

(1) 做好实验预习。了解触发器的功能特性、寄存器组的组成方法，基于 Quartus II 提供的元器件，画出电路图，标明引脚名，写出仿真时的操作序列（最好含信号取值）。

(2) 完成实验内容。实现所设计电路，验证电路正确性，保存仿真波形文件。

(3) 撰写实验报告。按所给模板撰写。

(4) 保存实验电路图。由于第四个实验需要使用前三个实验所设计的电路，因此，每个实验的电路图都必须保存起来。由于每个工程文件的同类 lpm_ 器件，都从 0 开始编号，因此，为了实现已有电路的重用，所有实验应该使用同一个工程文件，所有文件放在同一个文件夹下；否则，编译时 lpm_ 器件易产生错误，需要重新修改电路。

实验二 ALU 的设计

1 实验目的

- (1) 熟悉加/减法器的功能及使用方法。
- (2) 掌握 ALU 的组成原理。

2. 实验内容

- (1) 测试加/减法器的功能。
- (2) 设计具有加法、减法、逻辑与、逻辑非功能的 8 位 ALU，ALU 可产生结果状态标志 ZF、CF、OF、SF，并验证设计正确性。

3. 实验原理及方案

ALU 的核心是加/减法器，加/减法器的基础为加法器，产生的结果状态标志可以用来实现关系运算功能。

3.1 加/减法器功能的测试

Quartus II 提供的 `lpm_add_sub` 模块可实现加法、减法运算，可输出溢出标志 `overflow`、最高位进位 `cout`。注意，控制引脚 `add_sub=1、0` 时分别实现加法、减法（与常见约定相反）；`cout` 是加法器最高位进位，不是 CF；有些 Quartus 版本中，`lpm_add_sub` 可以选择加/减运算是无符号运算还是有符号运算，选择无符号运算时 `overflow` 结果不正确。

加/减法器功能的测试包括电路实现、电路仿真及结果分析几个步骤，仿真时先采用功能仿真方式得到结果并分析，再采用时序仿真方式查看器件操作的时延特征。

功能测试时，电路实现、电路仿真、结果分析的方法同触发器的功能测试。

注意，`lpm_add_sub` 中 `cin` 引脚的功能是实现带进位加法或带借位减法，本实验不需要设置该引脚；仿真波形中数据信号的选择应能够产生各种输出结果，如加法运算测试应包含 5 组数据（2 组++、2 组--、1 组+-），枚举 `dataA` 最高位为 0 及 1 时 `overflow`、`cout` 的可能组合，减法运算测试同样需要 5 组数据。

结果分析时，应以操作为单位进行分析，查看每个输出信号是否正确。最后，还应分析出无符号加/减运算的结果溢出条件，即 CF 的有效逻辑。

3.2 ALU 的设计、实现及验证

表 2 ALU 功能表

功能选择		实现功能			
SEL[1]	SEL[0]	操作	助记符	功能函数	影响的状态标志
0	0	加法	ADD	$F = A + B$	ZF、CF、OF、SF
0	1	减法	SUB	$F = A - B$	ZF、CF、OF、SF
1	0	逻辑与	AND	$F = A \cdot B$	ZF
1	1	逻辑非	NOT	$F = \overline{A}$	ZF

ALU 能够实现多种算术运算、逻辑运算功能，其功能由指令系统决定。本实验要求 ALU

的数据宽度为 8 位，具有 4 种算术及逻辑运算功能，可产生结果状态标志 ZF、CF、OF、SF。ALU 的功能如表 2 所示，其中，A、B 为数据入端，F 为数据出端，SEL 为功能选择（操作控制）信号。可见，SEL 为 2 位（记为 SEL[1..0]），F 的位数与 A 及 B 相同。

(1) ALU 的设计

由表 2 可知，ALU 的数据引脚 A、B 及 F 都为 8 位，控制引脚 SEL 为 2 位，还包含 4 根状态标志引脚。

ALU 中，加法、减法运算可用加/减法器来实现，逻辑与、逻辑非运算可用与门、非门来实现，当前操作的结果输出可用选择器来实现，ALU 的内部组成如图 11 ALU 的组成所示。

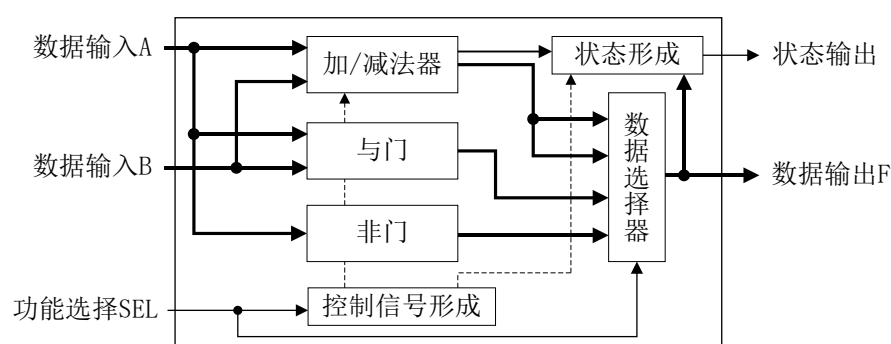


图 11 ALU 的组成

其中，状态形成电路负责产生 4 个结果状态标志，控制信号形成电路负责产生各模块所需的操作控制信号，本实验中只有加/减法器、状态形成电路需要使用控制信号。注意，结果状态标志 ZF 是由输出数据 F 形成的。

这里详细说明加减运算对 CF 的影响。在二进制加法运算中，CF 是指参与运算的两个数的最高位在运算时向更高位产生的进位；在做减法运算时，CF 是指参与运算的两个数的最高位在运算时向更高位产生的借位。在加法器 lpm_add_sub 中，进行加法运算时，cout 即是 CF。但是在进行减法运算时，加法器将减法运算转换成了加法，cout 实际是转换后的加法的进位，而不是减法运算的借位，即此时加法器的输出 cout 不是 CF。因此，减法运算的 CF 需要分析出相应的逻辑，画出逻辑电路。

(2) ALU 的实现与验证

ALU 实现需要编辑原理图文件（如 ALU.bdf），电路有多种实现方法，加/减法器、与门、非门可分别使用 Quartus II 提供的参数化模块 lpm_add_sub、lpm_and、lpm_inv。

ALU 的验证包括电路仿真、结果分析两个步骤，仿真采用功能仿真方式进行。

电路仿真需要先建立仿真波形文件（如 ALU.vwf），再设置所有输入信号在不同时刻的取值，输入信号的组合需能够反映电路的所有功能特性。例如，加法、减法功能验证时，CF 测试需包含 2×2 组数据，OF 无需测试（加/减法器测试中已进行过），SF、ZF 还需若干数据（可利用已有数据）。

分析仿真结果时，应以操作为单位进行分析，查看每个输入信号对应的输出信号是否正确。最后，还应分析出有/无符号关系运算的结果表示方法。

4. 实验要求

- (1) 做好实验预习。了解加/减法器的功能特性、ALU 的组成原理，基于 Quartus II 提供的元器件，画出电路图，标明引脚名，写出仿真时的操作序列及输出结果。
- (2) 完成实验内容。实现所设计电路，验证电路正确性，保存仿真波形文件。
- (3) 撰写实验报告。按所给模板撰写。
- (4) 保存实验电路图。原因同实验一。

实验三 存储器设计及总线互连

1 实验目的

- (1) 熟悉 RAM 的功能及使用方法。
- (2) 掌握存储器的设计方法。
- (3) 掌握基于总线的部件互连及操作方法。

2 实验内容

- (1) 测试 RAM 的功能。
- (2) 设计一个读/写端口分离的 128×8 位存储模块, 前 64B 为只读空间, 并验证设计正确性。
- (3) 将所设计存储模块、输入部件、输出部件连接到地址/数据复用的 8 位总线上, 通过输入部件对存储器进行读/写操作、通过输出部件查看结果, 并验证设计正确性。

3 实验原理及方案

3.1 RAM 功能的测试

Quartus II 提供了多种类型的 RAM 模块, 如 `lpm_ram_dq`、`lpm_ram_dp`、`lpm_rom` 等, 这些模块都为同步存储器, 即读/写操作都在时钟脉冲信号的上升沿开始; 读操作都可以选择是否需要信号控制; 读操作的数据输出都可以选择是否带输出锁存功能。

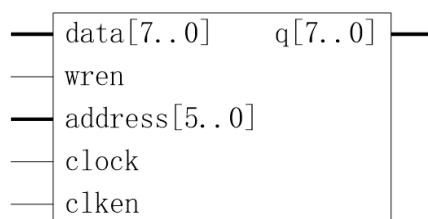


图 12 `lpm_ram_dq` 引脚示例

以 64×8 位的读/写端口分离的 `lpm_ram_dq` 模块为例, 其 I/O 引脚示例如图 12 所示。其中, `data[7..0]`、`q[7..0]` 分别为数据输入引脚、数据输出引脚, `wren` 为写使能引脚 (高电平有效), `clock` 为时钟脉冲引脚, `clken` 为时钟脉冲使能引脚 (高电平有效)。`clken` 用于允许、禁止 `clock` 信号进入芯片内部, 因此, `clken` 可以用作芯片的片选信号。

对 `lpm_ram_dq` 的操作有读、写两种。写操作时 (`wren=1`、`clken=1`), 地址及数据在 `clock` 上升沿被锁存, 然后数据被写入到指定存储单元中。读操作时 (`wren=0`、`clken=1`), 地址在 `clock` 上升沿被锁存, 然后指定存储单元的数据经过内部端口 `q`, 立即被送到引脚 `q` (端口 `q` 不带锁存功能时), 或下个时钟周期上升沿被输出到引脚 `q` (端口 `q` 带锁存功能时)。注意, 设置有读使能引脚 `rden` 时, 读操作的有效逻辑是 `rden=1`、`clken=1`。

存储器功能的测试放在存储模块设计的验证中进行 (存储模块设计太简单)。

3.2 存储模块的设计、实现及验证

存储模块可对存储器芯片进行容量扩展来实现, 容量扩展的方法有位扩展、字扩展、字位扩展 3 种。本实验要求设计一个读/写端口分离的 128×8 位存储模块 MEM, 前 64B 为只

读空间。

(1) 存储模块的设计

由设计要求可知，存储模块 MEM 可对 1 片 64×8 位 ROM、1 片 64×8 位 RAM 进行字扩展来实现。感兴趣的同学，可用位扩展方法实现 64×8 位 ROM 或 64×8 位 RAM。

读/写端口分离的 64×8 位 ROM 的引脚为：6 位地址、8 位数据输出、时钟脉冲 clock、片选 $\overline{c}lken$ ， 64×8 位的 RAM 的引脚还有 8 位数据输入、写使能 \overline{wren} 。

假设存储模块 MEM 的地址引脚为 $A[6..0]$ 、数据输入引脚为 $D[7..0]$ 、数据输出引脚为 $Q[7..0]$ 、片选引脚为 CS（高电平有效），则 ROM、RAM 连接时，地址引脚连接 $A[5..0]$ ，数据输入、clock、 \overline{wren} 引脚直接连接对应引脚，数据输出引脚通过 MUX 连接到 Q、用 $A[6]$ 进行选择，ROM 的 $\overline{c}lken = CS \cdot \overline{A[6]}$ 、RAM 的 $\overline{c}lken = CS \cdot A[6]$ 。

(2) 存储模块的实现及验证

存储模块的实现需要编辑原理图文件（如 Mem.bdf），电路有多种实现方法，ROM、RAM、MUX 可使用 Quartus II 提供的参数化模块 lpm_rom、lpm_ram_dq、lpm_mux，lpm_rom 需要预先写入各存储单元内容（可通过设置其初始化文件[如 rom.mif]来实现）。lpm_rom 初始化文件的建立，可通过主菜单 File→New→Other Files→Memory Initialization File（根据 Quartus 软件版本不同，路径可能为：File→New→Memory Files→Memory Initialization File）进入。注意，存储模块 MEM 应设置片选引脚 CS；应使 lpm_rom 及 lpm_ram_dp 都不带输出锁存功能，以便于立即输出读操作结果；应使 lpm_rom 中拟测试的存储单元的内容各不相同，以防止错误现象被隐藏。所建立的 ROM 初始化文件（名为 rom.mif）的文件名要赋给 lpm_rom 的 LPM_FILE 参数（注意要加上双引号），如图 13 所示。

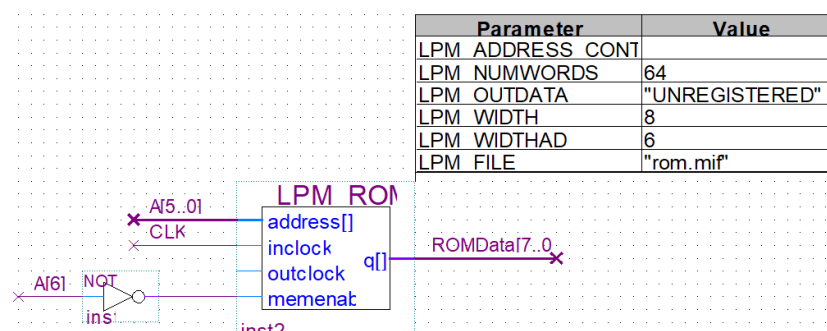


图 13 Quartus II 中 lpm_rom 的参数设置

存储模块的验证包括存储器功能测试、存储模块功能验证两个环节，可通过分别访问不同存储器芯片方法来实现。验证有电路仿真、结果分析两个步骤，仿真时先采用功能仿真方式得到结果，再采用时序仿真方式查看器件操作的时延特征。

电路仿真需要先建立仿真波形文件（如 MEM.vwf），在设置所有输入信号在不同时刻的取值，输入信号的组合需能够反映电路的所有功能特性。波形文件中，先使 $0 \leq A[6..0] < 63$ ，通过 2 组数据测试 ROM 的功能；再使 $64 \leq A[6..0] < 128$ ，通过 3 组数据（写单元 a、写单元 b、读单元 a）测试 RAM 的功能。注意，所有输入信号的时长都应以时钟周期为单位，输入信号的改变应滞后于时钟脉冲信号 clock 的上升沿。

分析仿真结果时，应以时钟周期为单位进行分析，查看每个输出信号的状态及时序是否

正确。

3.3 部件通过总线互连的设计、实现与验证

由于总线上同时只能有一个部件发送数据，因此，每个部件的输出端必须通过三态门连接到总线上。当部件有多个输入端时，为了防止不同输入端之间的信号干扰，输入端连接到总线之前需要设置锁存器；为了防止输入端与输出端之间的信号反馈（仅针对组合逻辑部件），输出端连接到总线之前需要设置锁存器，任何部件只有一个输入端或输出端可以直接连接到总线上。

本实验要求将所设计存储模块 MEM（128×8 位）、输入部件、输出部件连接到地址/数据复用的 8 位总线上，通过输入部件对 MEM 进行读/写操作、通过输出部件查看结果。

根据 MEM 的原理图文件（.bdf）生成相应的符号文件（.bsf）（通过 File→Create/Update→Create Symbol Files for Current File 实现），然后就可以像使用 lpm_dff 等器件一样，使用所设计电路了。

(1) 总线互连的设计

由于 MEM 是读/写端口分离的，其数据输出引脚 Q 需通过三态门连接到总线；由于总线是地址/数据复用总线，MEM 的数据输入引脚 D 直接连接到总线时，其地址引脚 A 需通过锁存器连接到总线；由于 MEM 是时序逻辑部件，数据输出信号仅与内部保存的状态有关、与数据输入信号无关，MEM 的数据输出引脚连接总线时无需设置锁存器。

同样，输入部件也需要通过三态门连接到总线，3 个部件通过地址/数据复用总线互连的原理图如图 14 所示，图中还标出了所需的操作控制信号，电路 C 为 MEM 的片选信号 CS 的连接电路（MEM 地址引脚 A 为 7 位、总线宽度为 8 位）。

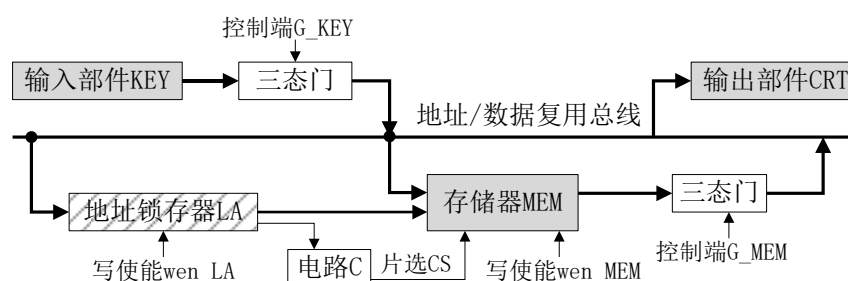


图 14 部件通过总线互连的连接电路

(2) 总线互连的实现与验证

总线互连的实现需要编辑原理图文件（如 Bus.bdf），电路有多种实现方法，存储模块 MEM 需使用实验三设计的电路（MEM.bsf），地址锁存器 LA、三态门可使用 Quartus II 提供的参数化模块 lpm_dff、lpm_bustri，KEY、CRT 可直接使用引脚 input、output。注意，LA 应该用触发器实现，以便于基于 clock 进行操作控制（同 MEM 控制方式）。

总线互连的验证包括电路仿真、结果分析两个步骤，仿真采用功能仿真方式进行。

电路仿真需要先建立仿真波形文件（如 Bus.vwf），再在文件中进行写 RAM 单元 b1、读 ROM 单元 c、写 RAM 单元 b2、读 RAM 单元 b1 操作即可。注意，每个操作需要 2 个时钟周期实现（写 LA、访存）；所有输入信号的时长都应以时钟周期为单位，输入信号的改变

应滞后于时钟脉冲信号 clock 的上升沿。

分析仿真结果时，应以时钟周期为单位进行分析，查看每个输出信号的状态及时序是否正确。

4 实验要求

- (1) 做好实验预习。了解 RAM 的组成原理、单总线的部件互连方法，基于 Quartus II 提供的元器件，画出电路图，标明引脚名，写出仿真时的操作序列（含信号取值）。
- (2) 完成实验内容。实现所设计电路，验证电路正确性，保存仿真波形文件。
- (3) 撰写实验报告。按所给模板撰写。
- (4) 保存实验电路图。原因同实验一。

实验四 数据通路的组织

1 实验目的

- (1) 了解数据通路的组织方法。
- (2) 掌握指令执行过程的控制原理。

2 实验内容

- (1) 设计一个单总线结构的数据通路，支持 Demo_IS 指令系统（见[实验概要第3节](#)）的取数(LD)、减法(SUB)、双字长分支(JNZ)指令。
- (2) 将测试程序存入主存，根据程序执行过程的 μ OPCmd（微操作命令）序列，控制所设计数据通路，来验证数据通路的正确性。

3 实验原理及方案

3.1 指令功能分析

本实验要求支持 Demo_IS 指令系统的取数(LD)、减法(SUB)、双字长分支(JNZ)指令，其功能分别为： $RD \leftarrow M[(RS)]$ 、 $RD \leftarrow (RD) - (RS)$ 、 $ZF=0$ 时 $PC \leftarrow \text{addr}$ 。其中，RD、RS 表示寄存器编号， (Rx) 、 $M[(Rx)]$ 表示寄存器、存储单元的内容，addr 为直接寻址方式的地址码，ZF 为上一条指令所产生的结果状态标志。

由 Demo_IS 的指令格式及上述 3 条指令功能约定，可得到如下分析结果：

- 1) 数据类型只有 8 位整数一种，采用定点格式（补码编码）表示；
- 2) 数据操作只有 8 位的加法、减法两种，需产生状态标志 ZF；
- 3) 数据寻址有寄存器寻址、寄存器间接寻址两种方式，地址无需计算；
- 4) 指令寻址有直接寻址、隐含寻址两种方式，地址计算方法为 8 位加法；
- 5) 寄存器有 4 个，长度为 8 位，每条指令最多 2 次读、1 次写操作；
- 6) 存储器按字节编址、地址空间为 8 位，每条指令最多有 1 次读/写操作。

3.2 数据通路的设计与实现

数据通路由通路部件、部件互连两部分组成，互连结构有总线结构、点点结构两种类型。本实验的数据通路要求采用单总线结构。

(1) 功能部件设计

为了满足 Demo_IS 中 3 条指定指令的要求，数据通路的功能部件应包括 ALU、寄存器组 GPRs、状态寄存器 PSR、存储器 MEM、程序计数器 PC、指令寄存器 IR，以及地址寄存器 MAR、数据寄存器 MDR。

由 Demo_IS 指令系统分析结果可见，ALU 应具有加法、减法功能，需产生状态标志 ZF，可使用实验二设计的 ALU 来实现；GPRs 应包含 4 个 8 位寄存器，具有 1 个读端口和 1 个写端口，可使用实验一设计的寄存器组来实现；MEM 的容量应 $\leq 256 \times 8$ 位，可使用实验三设计的存储模块来实现。为了简化控制，假设 PC 具有计数功能，可用 Quartus II 提供的 lpm_counter 模块来实现。

想使用实验一至实验三的设计结果，就需要先根据原理图文件（.bdf）生成相应的符号文件（.bsf）（通过 File→Create/Update→Create Symbol Files for Current File 实现），然后就可以像使用 lpm_dff 等器件一样，使用所设计电路了。

(2) 部件互连设计

单总线结构的数据通路中，所有部件的数据入端、数据出端都连接在同一个总线上。为了保证数据传送的正确性，部件的出端需通过三态门连接到总线；部件的入端及出端中，只有 1 个可以直接连接总线，其余都需通过锁存器连接到总线。

本实验中，数据通路的宽度为 8 位，数据通路的组成如图 15 所示，其中的 TS0~TS3 为三态门。为了便于调试，图中增加了输出信号 PC、IR 及 CRT，可以根据需要，自行增加输出信号，要求至少需输出 PC、IR、CRT、RS、RD。

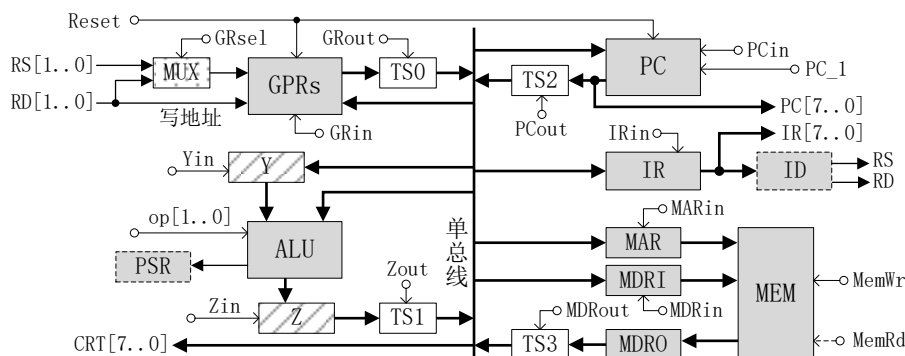


图 15 单总线结构的数据通路组成

图中的 GPRs 的读地址引脚前增加了选择器 MUX，以满足 SUB 指令周期中 2 次读 GPRs 的需求，当前地址（RS 或 RD）用控制信号 GRsel 来选择。图中的 MEM 需要存放测试程序，故要求使用实验三设计的存储模块（前 64B 为只读空间）来实现。由于实验三的存储模块未设置读操作控制信号，图 15 中的控制信号 MemRd 可以省略（读命令用 $\overline{\text{MemWr}}$ 表示）。

PSR 用于存放结果状态标志，供 CU 使用，由于本实验的 μOPCmd 由手工给出，因此，判断 $\text{ZF}=0$ 也是由人工完成的，故 PSR 可以省略（用虚线表示）。

(3) 数据通路的实现

数据通路的实现需要编辑原理图文件（如 DataPath.bdf），电路有多种实现方法，ALU、GPRs、MEM、PC 都按照功能部件设计的方案来实现，IR、MAR、MDR、Y、Z 应使用 lpm_dff 实现，所有的部件控制信号都需设置为输入引脚。

为了正确地进行数据传送， μOP （微操作）控制应采用电位-脉冲制，即发送部件在时钟周期开始时就输出数据（如打开三态门），接收部件在时钟周期结束前接收数据并写入。由于 μOPCmd 由 CU 在时钟周期开始时（clock 上升沿）产生（电路会有一定延迟），因此， μOPCmd 在整个时钟周期内都有效， μOPCmd 有效时，clock 上升沿已经错过了， μOP 只能在当前时钟周期结束时（即下个 clock 上升沿时）写入数据，效果如图 9 所示。

3.3 数据通路的验证

本实验要求将测试程序存入主存，根据程序执行过程的 μOPCmd 序列，控制所设计数据通路，来验证数据通路的正确性。

测试程序需要自行编写，只需包含 3 种指令即可。测试程序示例如下：① $R1 \leftarrow M[(R0)]$ 、② $R2 \leftarrow M[(R1)]$ 、③ $R2 \leftarrow (R2) - (R1)$ 、④JNZ 22H，其中，前 3 条指令为单字长指令，第 4 条指令为双字长指令，程序共占 5 个存储单元。

计算机启动时都会进行硬件初始化，假设图 15 初始化后，GPRs 及 PC 的内容均为 0。计算机中执行一个程序都有准备、执行两个环节。

(1) 程序执行的准备

程序执行的准备工作是将程序调入主存、将程序首地址写入 PC。该工作原本由操作系统完成，本实验通过将程序预先存入 MEM 的初始化文件中来实现。

MEM 的前 64B 空间由 lpm_rom 实现，其初始化文件（如 rom1.mif）中，可存放测试程序及预存数据。由于初始化后 $(PC)=0$ ，故测试程序从地址为 0 的单元开始存放。

为了有效进行测试程序中指令③的测试，应该使 $(R2) \neq (R1)$ 。由于初始化后 GPRs 的内容都为 0，故 $(R1)=M[0]$ ，而 $M[0]$ 的内容为测试程序中指令①的内容（值为 24H）。欲使 $(R2) \neq (R1)$ ，在 $M[24H]$ 单元中存入不同的值（如 35H）即可。

(2) 程序执行的实现

程序执行的任务是自动、逐条按(PC)取出指令并执行，本实验通过手工输入（代替 CU 自动产生）各条指令执行过程所需的 μOPCmd 序列，来实现程序执行的控制。

每条指令的 μOPCmd 序列，可以根据指令功能、图 15 的数据通路给出。指令执行过程由取指、译码、执行三个阶段组成，译码阶段有一定延、但没有 μOP ，本实验假设，译码安排在取指阶段的最后一个 μOP 中实现，每个 μOP 时延为 1 个时钟周期。注意，不同指令的 μOPCmd 序列之间不要留空档，尽量模仿真实的程序执行过程。

(3) 数据通路的验证

数据通路的验证包括电路仿真、结果分析两个步骤，仿真采用功能仿真方式进行。

电路仿真需要先建立仿真波形文件（如 DataPath.vwf），再在文件中初始化硬件（Reset 有效后无效），然后给出与测试程序相对应的每一条指令周期所需的 μOPCmd 序列。注意，所有输入信号的时长都应与时钟周期为单位，输入信号的改变应滞后于时钟脉冲信号 clock 的上升沿

分析仿真结果时，应以 μOP （1 个时钟周期）为单位进行分析，查看每个输出信号的状态及时序是否正确（与预期结果一致）。若不一致，则数据通路或 μOPCmd 序列有错误，分析原因、找出错误点、修改电路或 μOPCmd 序列，直到结果一致为止。注意，一定要预先写好完整的 μOPCmd 序列，及每个 μOP 的预期结果，否则根本不知道对与错。

4 实验要求

(1) 做好实验预习。了解数据通路的组成原理及指令执行过程的组织方法，基于 Quartus II 提供的元器件，画出电路图，标明引脚名，编写测试程序，写出相应的 μOPCmd 序列。

(2) 完成实验内容。实现所设计电路，验证电路正确性，保存仿真波形文件。

(3) 撰写实验报告。按所给模板撰写。

(4) 上交实验结果及实验报告。本实验做完的下一周，将 4 个实验所共用的工程文件内容、实验报告（包含 4 个实验）的电子稿交上来。每组学生交一份，文件夹名称为该组学生的学号及姓名（如 301XXX302YYY），实验报告就放在这个文件夹下，工程文件的内容放在下一级文件夹中。班长收齐电子稿后，一起交给老师。