### 计算机组成实验指导书

### 实验1 系统结构设计初步

### 一、实验目的

1．学习了解基于计算机系统的数字逻辑设计初步

2．掌握基本的组合和时序逻辑设计方法

3. 掌握掌握通用寄存器组的设计方法了解通用寄存器组的用途及对CPU的重要性。

### 二、实验原理

寄存器是计算机的一个重要部件，用于暂存数据、指令等。它由触发器和一些控制门组成。在寄存器中，常用的是正边沿触发D触发器和锁存器。

图1-1所示是由正沿触发的D触发器组成的4位寄存器。在CP正沿作用下，外部数据才能进入寄存器。



加法器是实现计算机算数逻辑运算的一个基本的组成成分。

1. 半加器

所谓半加器是指对两个二进制1位数相加，而不考虑低位进位的加法器。

1. 其逻辑符号如图2-1所示，其中，Ai和Bi分别为两个二进制数的i位值；Si为Ai和Bi相加之和，叫做本位和；Ci+1是本位向高位的进位。



图2-1半加器逻辑符号

1. 真值表如表2-1所示。

表2-1 半加器的真值表

|  |  |
| --- | --- |
| 输入 | 输出 |
| Ai Bi | Si Ci+1 |
| 0 0 | 0 0 |
| 0 1 | 1 0 |
| 1 0 | 1 0 |
| 1 1 | 0 1 |

（3）逻辑表达式。

由真值表可写出半加器的逻辑表达式：

Si = Bi + Ai= AiBi

Ci+1 = AiBi

（4）电路图。

由逻辑表达式可得出半加器的电路图，如图2-2所示。

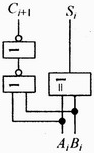


图2-2半加器的电路图

2. 全加器

能对两个1位二进制数相加并考虑低位来的进位（即相当于3个1位二进制数的相加）得到“和”及“进位”的逻辑电路，称之为全加器。其中Ai和Bi分别为两个1位二进制数的输入；Ci-1为低位来的进位输入；Si和Ci分别为相加后形成的“和”及向高位的“进位”输出。图2-3是全加器的逻辑符号图。

1. 全加器的真值表如表2-2所示：
2. 根据全加器的真值表，可得出全加器的逻辑表达式为

Si = Ci-1 + Bi+ Ai + Ai Bi Ci-1

Ci = AiBi + AiCi-1 + BiCi-1

表2-2全加器的真值表

|  |  |
| --- | --- |
| 输入 | 输出 |
| Ai BiCi | Si Ci+1 |
| 0 0 0 | 0 0 |
| 0 0 1 | 1 0 |
| 0 1 0 | 1 0 |
| 0 1 1 | 1. 1 |
| 1 0 0 | 1 0 |
| 1 0 1 | 0 1 |
| 1 1 0 | 0 1 |
| 1 1 1 | 1 1 |



图2-3全加器逻辑符号

3. 多位加法器

如果有两个n位二进制数相加，就需n位全加器，这样构成的逻辑电路称为多位并行加法器。按照进位方式的不同，并行加法器分为行波进位加法器和先行进位加法器两种。

（1）行波进位加法器的逻辑框图如图2-4所示。这种加法器的构成比较简单，只要把n位全加器串联起来，低位全加器的进位输出连到相邻的高位全加器的进位输入。

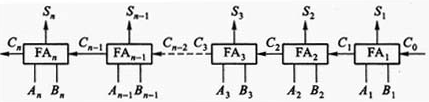


图2-14行波进位加法器

由图可见，这样构成的加法器，尽管各位相加是并行的，但其进位信号是由低位向高位逐级传递的，好像行波一样。这样，要形成高位的和，要等低位的进位形成后才能确定。正如做手算加法那样，要从低位逐级求出进位，最后才求得高位的和及进位。因此，这种加法器，由于进位是串行的，形成进位的速度很慢，加法器的速度主要受进位传递时间的限制。

若每级全加器形成进位的延时为2tpd，则在最坏情况下，从FA1的输入到产生高位进位Cn需要时间为2tpd×n。当 n增大时，完成一次加法所需时间也随之增加。

4.通用寄存器组

通用寄存器组是CPU的重要组成部分。从存储器取来的数据要放在通用寄存器中；从外部设备取来的数据除DMA方式外，要放在通用寄存器中。向存储器输出的数据也是从通用寄存器中取出的；向外部设备输出的数据除DMA方式外也是从通用寄存器中取出的。由于从通用寄存器组中取数据比从存储器或者外部设备取数据速度快得多，因此参加算术运算和逻辑运算的数据一般是从通用寄存器组中取出的，它向算术逻辑单元ALU提供了进行算术运算和逻辑运算所需要的2个操作数，同时又是运算结果的暂存地。通用寄存器组内寄存器的数目与CPU性能有关，CPU性能越高，通用寄存器组内的寄存器数目越多。由于算术逻辑运算需要2个操作数，因此通用寄存器组有2个读端口，负责提供进行算术逻辑单元需要的源操作数和目的操作数。通用寄存器组有1个写端口，负责将运算结果保存到指定的寄存器内。根据通用寄存器组的功能要求，一个只有4个16位寄存器的通用寄存器组的框图如图3-1所示。



图3-1 4个16位寄存器组成的通用寄存器组

在图3-1所示的电路中，当reset为低电平时，将4个16位寄存器R0—R3复位为0。

当寄存器的write和sel为高电平时，在时钟信号clk的上升沿将D端的输入D[15..0]写入寄存器，然后送到寄存器的输出Q[15..0]。

4个寄存器的允许写信号write和外部产生的目的寄存器写信号DRWr直接相连。每个寄存器还有另一个选择信号sel，它决定哪一个寄存器进行写操作。

4个寄存器的选择信号分别和2-4译码器产生的sel00、sel01、sel10和sel11相连。只有当1个寄存器被选中（sel为高电平）时，才允许对该寄存器进行写操作。

2-4译码器的输入sel[1..0]接DR[1..0]，2-4译码器对2位的输入信号sel[1..0]进行2-4译码，产生4个输出sel00、sel01、sel10和sel11，分别送往4个寄存器R0、R1、R2和R3的选择端sel。

4选1多路选择器1从4个寄存器R0、R1、R2和R3的输出Q[15..0]选择1路送到DR\_data[1..0],给算术逻辑单元提供目的操作数；选择信号sel[1..0]接DR[1..0]。4选1多路选择器2从4个寄存器R0、R1、R2和R3的输出Q[15..0]选择1路送到SR\_data[1..0],给算术逻辑单元提供源操作数；选择信号sel[1..0]接SR[1..0]。

通用寄存器组：

通用寄存器组中有4个16位的寄存器。

当复位信号reset = 0时，将通用寄存器组中的4个寄存器清零。

（3）通用寄存器组中有1个写入端口，当DRWr=1时，在时钟clk的上升沿将数据总线上的数写入DR[1..0]指定的寄存器。

（4）通用寄存器组中有2个读出端口，一个对应算术逻辑单元的目的操作数DR，另一个对应算术逻辑单元的源操作数SR。DR[1..0]选择目的操作数；SR[1..0]选择源操作数。

（5）设计要求层次设计。底层的设计实体有3个：16位寄存器，具有复位功能和允许写功能；一个2-4译码器，对应寄存器写选择；一个4选1多路开关，负责选择寄存器的读出。顶层设计构成一个完整的通用寄存器组。

引脚对应关系如下：

（1）clk对应实验台上的时钟（ST单脉冲）。

（2）reset对应实验台上的CPU复位信号CPU\_RST。

（3）SR[1..0]对应实验台开关SW7、SW6。

（4）DR[1..0]对应实验台开关SW1、SW0。

（5）DRWr对应实验台开关SW8。

（6）数据输入对应实验台开关S15-S0。

（7）目的操作数用实验台上的指示灯FR15—FR0显示，源操作数用实验台上的指示灯L15—L0显示。

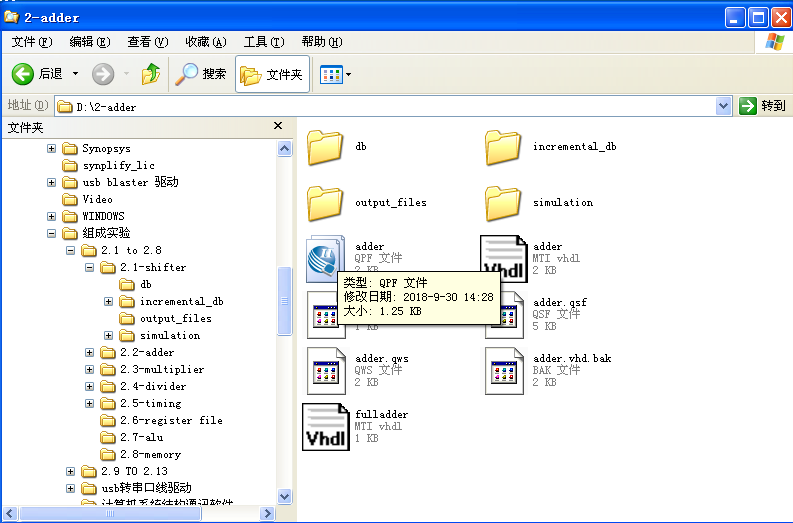
#### 三、实验步骤

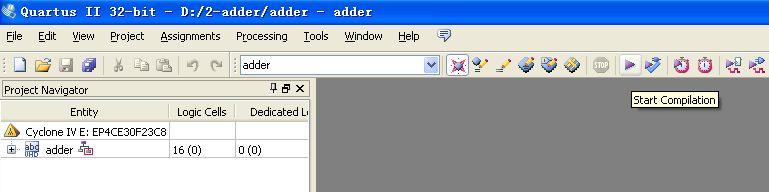
（1）组成逻辑实验 部分

1 将子板上的JTAG端口和PC机的USB口用下载电缆连接。打开实验台电源。

2 打开移位寄存器模块

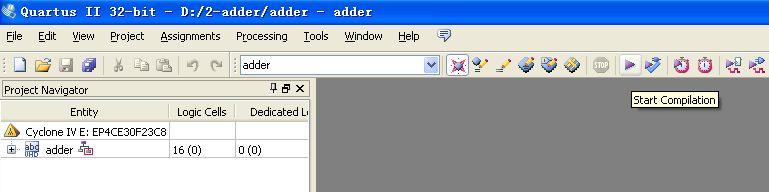
双击adder.qpf

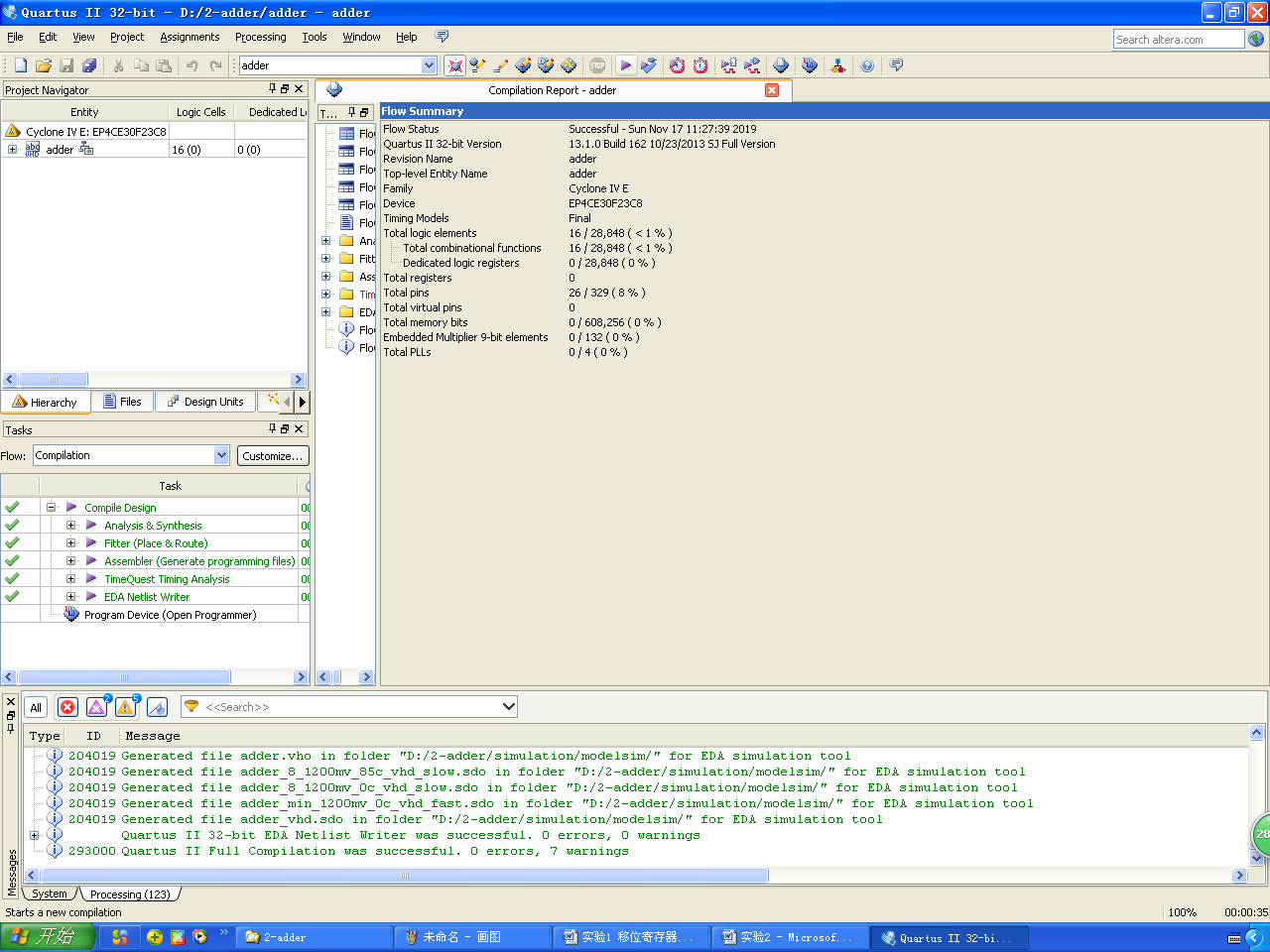




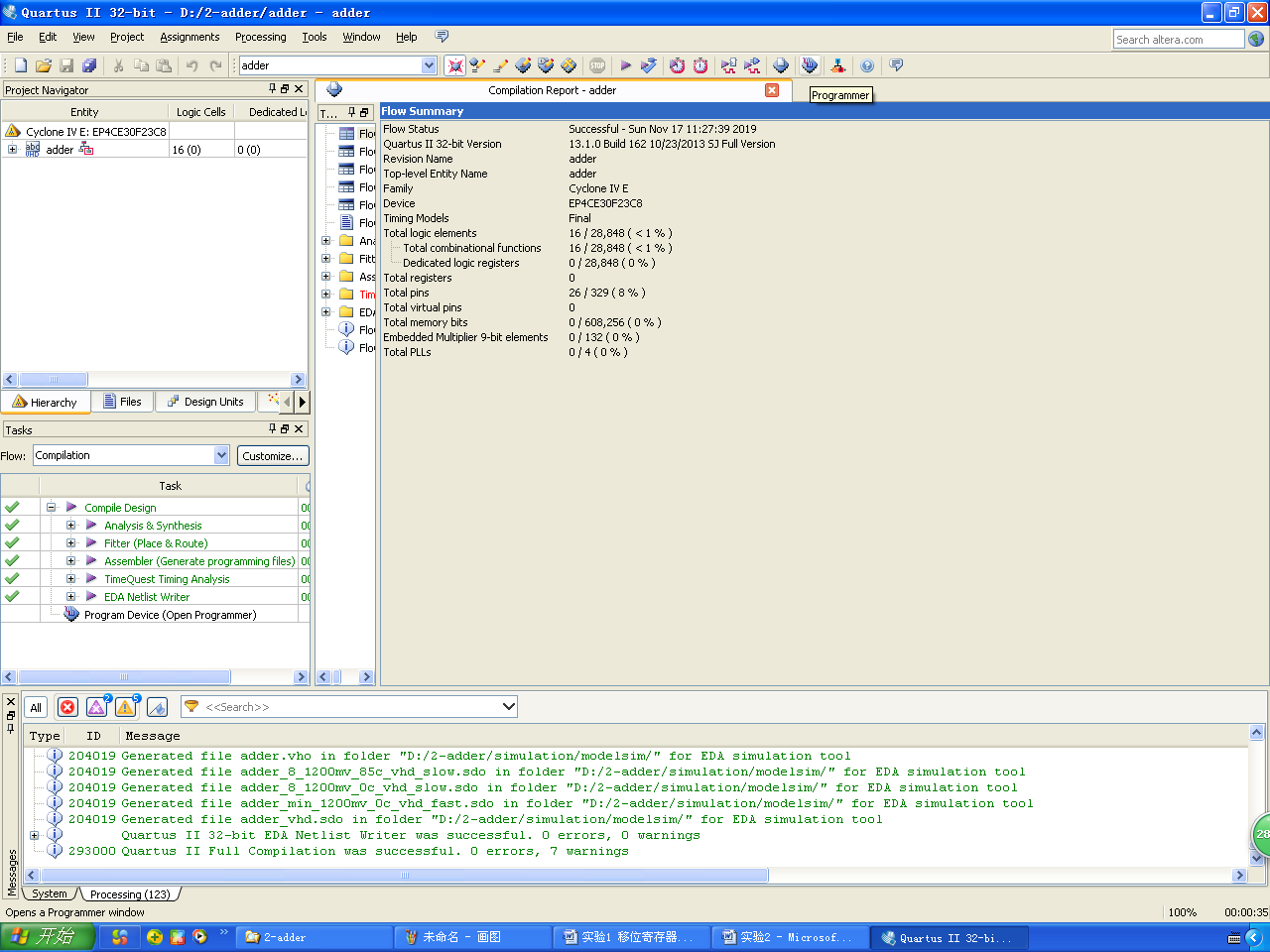
3执行Tools→Programmer命令，将adder.sof下载到FPGA中。注意在执行Programmer命令中，应在program/configure下的方框中打勾，然后下载。

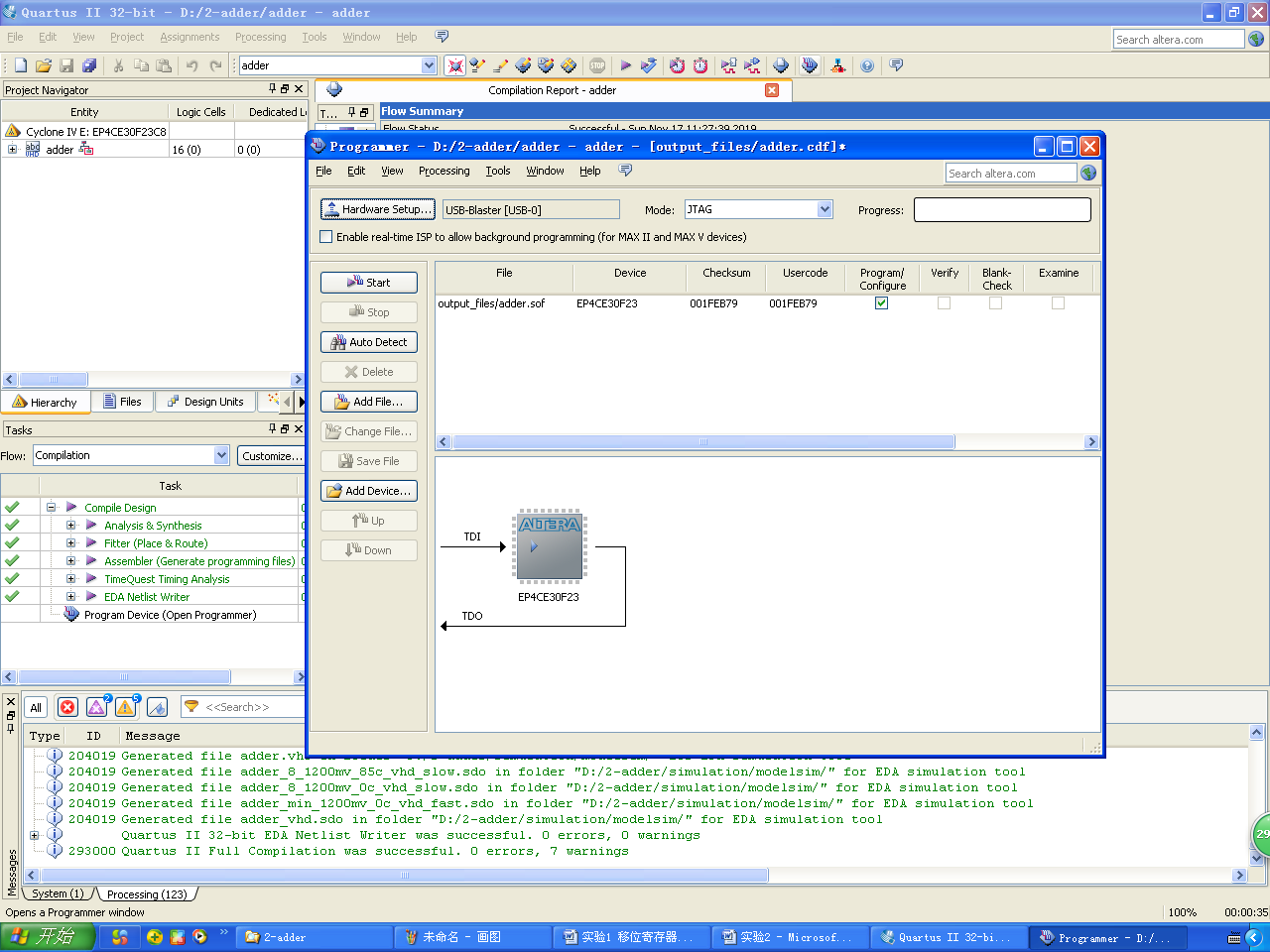
单击



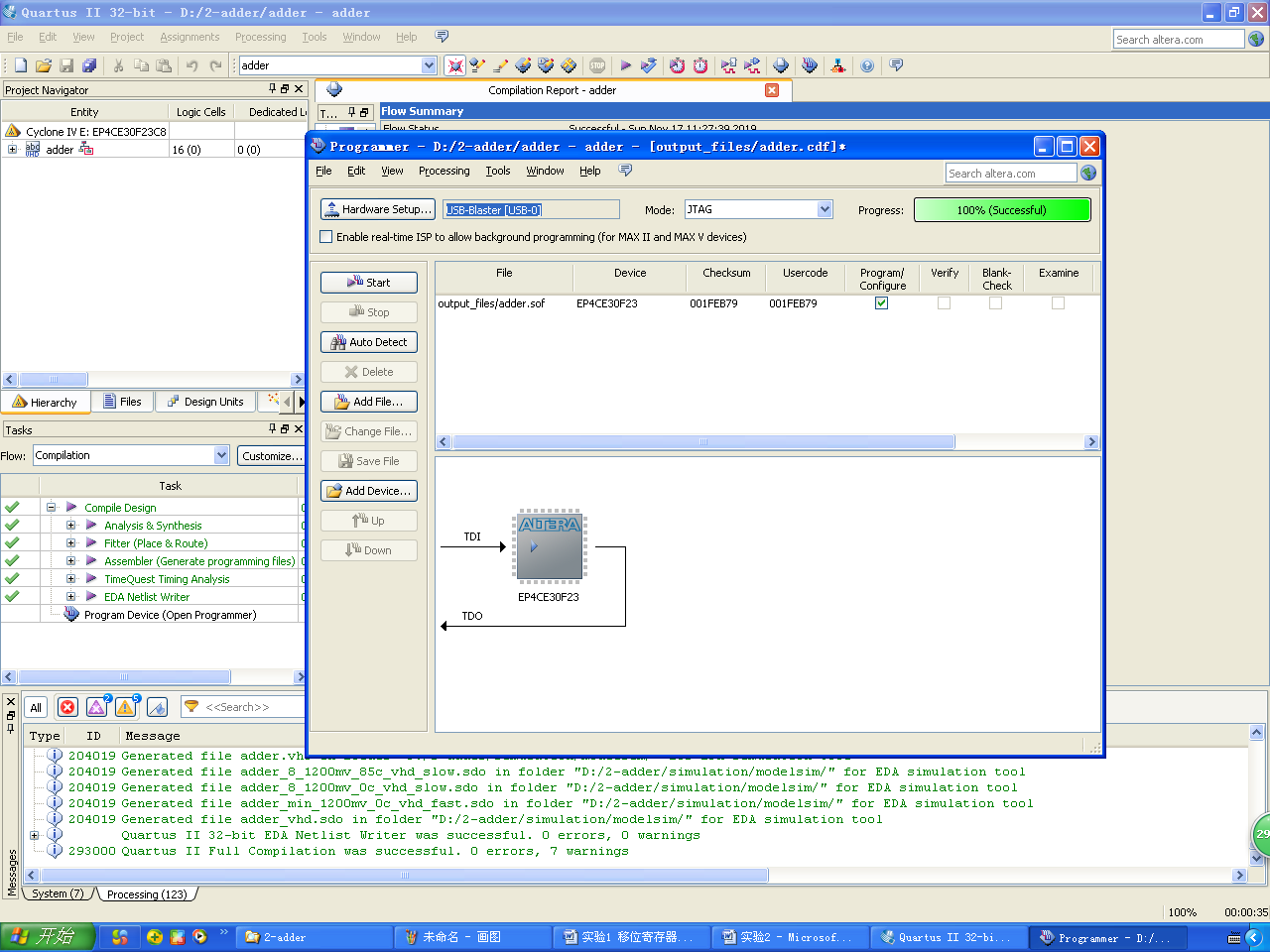


单击





单击



4将STEP开关置成1。

5本实验实现8位数相加A+B = sum(CarryOut)。

输入输出规则对应如下：

（1）输入的8位操作数A7～A0对应开关SW7～SW0。

（2）输入的8位操作数B7～B0对应开关SW15～SW8。

（3）最低位进位cin对应开关SW16。

（4）和sum7～sum0对应灯L7～L0，最高位进位CarryOut对应灯L8。

如（00000000）+（11111111）=（11111111），CarryOut = 0。

（1）拨动开关SW7～SW0输入8位操作数Ⅰ（A7～A0）00000000、SW15～SW8输入8位操作数Ⅱ（B7～B0）11111111，SW16输入cin，

（2）观察和sum（灯L7～L0），CarryOut（灯L8），填入表中，并检查结果是否正确。

8位加法器实验

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作数Ⅰ | 操作数Ⅱ | Cin | Sum | CarryOut |
| 1011 1111 | 0100 1111 | 0 |  |  |
| 1010 1010 | 1010 1010 | 1 |  |  |
| 1111 0000 | 0000 1111 | 0 |  |  |
| 1111 1111 | 1111 1111 | 1 |  |  |
| 1010 1010 | 01010101 | 1 |  |  |
| 1111 1111 | 00000001 | 0 |  |  |

（2）时序逻辑部分

1 将子板上的JTAG端口和PC机的USB口用下载电缆连接。打开实验台电源。

2 打开移位寄存器模块

双击regfile.qpf

3 执行Tools→Programmer命令，将adder.sof下载到FPGA中。注意在执行Programmer命令中，应在program/configure下的方框中打勾，然后下载。

4 将STEP开关置成1。

5将下列数据存入寄存器

R0 0x1111

R1 0x5555

R2 0xAAAA

R3 0xFFFF

具体操作步骤如下（以写寄存器R0为例，写其他寄存器操作步骤除DR，SR开关和输入数据设置不同，其他一致）：

将DRWr对应的实验台开关SW8置为1；

设置目的寄存器DR[1..0]对应的实验台开关SW1、SW0为00；

设置源寄存器SR[1..0]对应实验台开关SW7、SW6为00；

将数据输入对应的实验台开关S15-S0设置为0001 0001 0001 0001；

按启动按钮，观察目的操作数对应的指示灯FR15—FR0，源操作数对应的指示灯L15—L0并填表。

(4) 在实验台指示灯FR15—FR0和L15—L0上观察各寄存器的值，并填表中，并检查结果是否正确。

表3通用寄存器组实验

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 步骤 | FR15—FR0 | | | | L15—L0 | | | |
| R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 |
| （1）reset后 |  |  |  |  |  |  |  |  |
| （2）R0写入1111后 |  |  |  |  |  |  |  |  |
| （3）R1写入5555后 |  |  |  |  |  |  |  |  |
| （4）R2写入AAAA后 |  |  |  |  |  |  |  |  |
| （5）R3写入FFFF后 |  |  |  |  |  |  |  |  |
|  |  | | | |  | | | |

计算机系统结构实验报告

级 班 姓名 学号

（组合逻辑部分）加法器

8位加法器实验

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作数Ⅰ | 操作数Ⅱ | Cin | Sum | CarryOut |
| 1011 1111 | 0100 1111 | 0 |  |  |
| 1010 1010 | 1010 1010 | 1 |  |  |
| 1111 0000 | 0000 1111 | 0 |  |  |
| 1111 1111 | 1111 1111 | 1 |  |  |
| 1010 1010 | 01010101 | 1 |  |  |
| 1111 1111 | 00000001 | 0 |  |  |

回答问题

1. 下图是本实验用到的8位全加器的器件引脚图，请结合实验补全引脚，并说明每个引脚的功能。

OUPUT

8位全加器器件

INPUT

？

？

8位全加器器件图

（时序逻辑部分）通用寄存器组

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 步骤 | FR15—FR0 | | | | L15—L0 | | | |
| R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 |
| （1）reset后 |  |  |  |  |  |  |  |  |
| （2）R0写入1111后 |  |  |  |  |  |  |  |  |
| （3）R1写入5555后 |  |  |  |  |  |  |  |  |
| （4）R2写入AAAA后 |  |  |  |  |  |  |  |  |
| （5）R3写入FFFF后 |  |  |  |  |  |  |  |  |
|  |  | | | |  | | | |

1. 下图是本实验用到的通用寄存器组的器件引脚图，请结合实验补全引脚，并说明每个引脚的功能。

OUPUT

通用寄存器器件

INPUT

？

？

通用寄存器器件图