实验一 算术逻辑单元（ALU）的设计与实现

一． 实验目的

1. 掌握全加器和行波进位加法器的结构;
2. 熟悉加减法运算及溢出的判断方法;
3. 掌握算术逻辑单元（ALU）的结构;
4. 熟练使用 SystemVerilog HDL 的行为建模和结构化建模方法对 ALU 进行描述实现。

二． 实验环境

1. 操作系统：Windows 或 Ubuntu 16.04
2. 开发环境：Xilinx Vivado 2018.3
3. 硬件平台：远程 FPGA 硬件云平台

三． 实验原理

1. 全加器

如图 1-1 所示，1 位全加器具有三个输入（A, B, C_{in}）和两个输出（S, C_{out}）。输入 A 和 B 各表示 1 位二进制数，两者相加，求和的结果通过 S 输出，进位通过 C_{out} 输出。此外，进位输入 C_{in} 和进位输出 C_{out} 在采用全加器构造多位加法器的时候会被使用。一个全加器的真值表如表 1-1 所示。

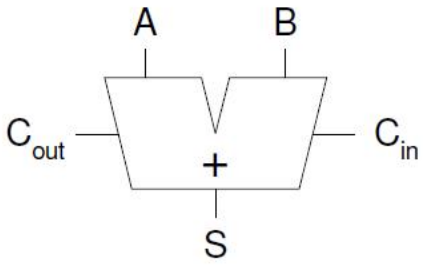


图 1-1 全加器

表 1-1 全加器的真值表

输入			输出	
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2. 行波进位加法器

行波进位加法器（ripple-carry adder）是最简单的一种进位传播多位加法器（CPA）。它就是将 N 个全加器进行串联，每级的 C_{out} 就是下一级的 C_{in}，则所有进位构成的通路称为进位链。一个 3 位行波进位加法器的结构如图 1-2 所示。

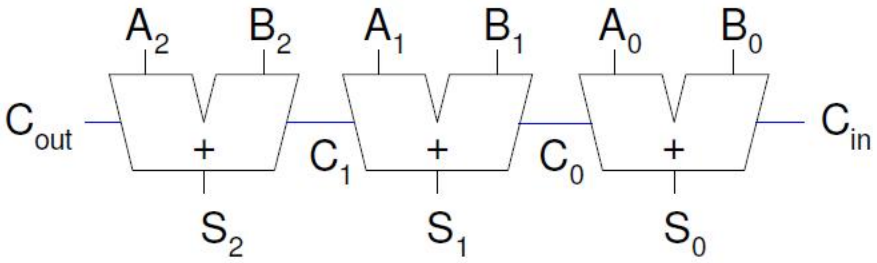


图 1-2 3 位行波进位加法器

3. 算术逻辑单元 ALU

在现代计算机中，算术逻辑单元（ALU）是专门执行算术和逻辑运算的数字电路。ALU 是计算机中处理器的重要组成部分，甚至连功能最简单的微处理器也会包含 ALU。ALU 的主要功能是进行二进制数的算术和逻辑运算，包括加法、减法、乘法（通常不含除法）、移位运算、逻辑与、逻辑或等等，一个 N 为 ALU 如

图 1-3 所示。

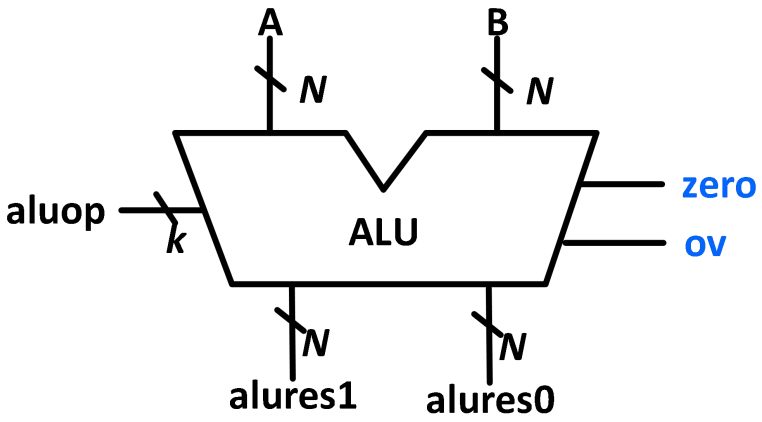


图 1-3 N 位 ALU 单元

它具有 3 个输入和 4 个输出。A 和 B 是两个 N 位的操作数，aluop 为操作类型控制信号，ALU 单元根据 aluop 信号确定对 A 和 B 进行何种操作。alures0 用于输出非乘法操作的结果或乘法运算结果的低 N 位，alures1 用于输出乘法结果的高 N 位（对于非乘法运算恒为 0）。ov 为有符号数加减法的溢出标志，zero 是表示 ALU 结果是否为 0 的零标志（对所有运算均有效）。

四． 实验内容

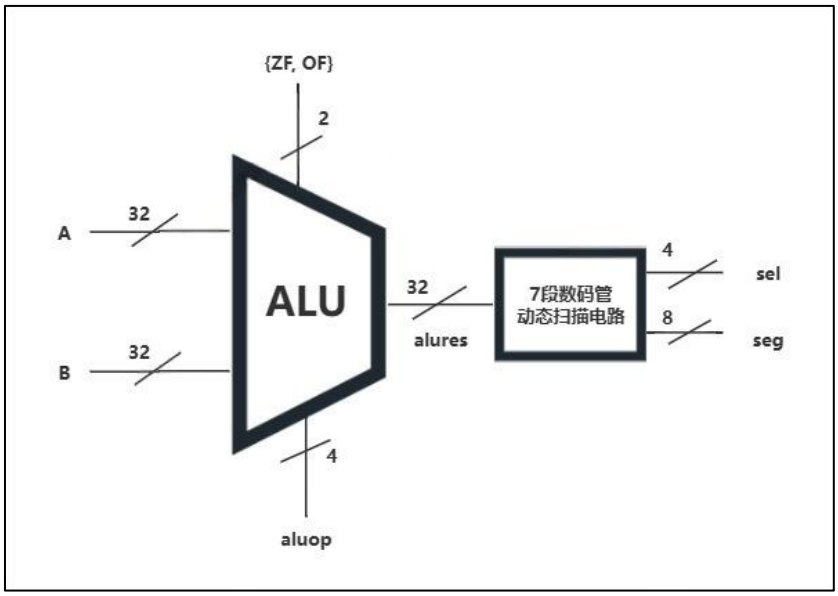


图 1-4 实验的顶层模块

基于 SystemVerilog HDL 设计并实现一个 32 位 ALU 单元。整个工程的顶层模块如图 1-4 所示，输入/输出端口如表 1-2 所示。注意，顶层模块由两个子模块组成，其中，一个是 ALU 单元，另一个是 7 段数码管动态显示扫描单元。**同学们只需要实现 ALU 单元即可**，动态显示扫描单元在工程中直接提供。

表 1-2 输入/输出端口

端口名	方向	宽度（位）	说明
A	输入	32	操作数 A。
B	输入	32	操作数 B。
aluop	输入	4	操作类型。
OF	输出	1	有符号数加/减法溢出标志。
ZF	输出	1	零标志（对所有操作均有效）。
seg	输出	8	连接七段数码管的数据输入端， 用于显示 ALU 单元的运算结果。
sel	输出	4	连接七段数码管的使能端， 用于显示 ALU 单元的运算结果。

ALU 单元所支持的运算功能如表 1-3 所示。

表 1-3 ALU 单元所支持的运算功能

aluop	助记符	功能	说明[&]
0000	AND	按位与	alures = A & B
0001	OR	按位或	alures = A | B
0010	XOR	按位异或	alures = A ⊕ B
0011	NAND	按位与非	alures = ~(A & B)
0100	NOT	逻辑非	alures = ~A
0101	SLL	逻辑左移	alures = A << B（B 取低 5 位）
0110	SRL	逻辑右移	alures = A >> B（B 取低 5 位）
0111	SRA	算术右移[&]	alures = A >> B（**算术右移**，B 取低 5 位）

1000	MULU	无符号数乘法	alures = $(A * B)_{[31:0]}$，**高 32 位舍弃**
1001	MUL	有符号数乘法	alures = $(A * B)_{[31:0]}$，**高 32 位舍弃**
1010	ADD	有符号数加法	alures = A + B，**需要设置 OF 位**
1011	ADDU	无符号数加法	alures = A + B
1100	SUB	有符号数减法	alures = A - B，**需要设置 OF 位**
1101	SUBU	无符号数减法	alures = A - B
1110	SLT	有符号数比较	alures = (A < B)? 1 : 0
1111	SLTU	无符号数比较	alures = (A < B)? 1 : 0

$：所有运算均需要设置 ZF 位；不需要设置 OF 位的运算，OF 位默认为 0。

&：算术右移高位补符号位的前提是对有符号数进行右移，需要特别注意这一点。

完成上述 32 位 ALU 单元的设计，必须满足如下几点要求：

1. ALU 单元的输入 A 和 B 均是补码形式。
2. 实现加法和减法时，不能使用“+”和“-”两种运算符，且只能通过一个行波进位加法器和其它必要的逻辑电路实现。
3. 可以使用“*”运算符实现乘法，但该运算符只适用无符号数的乘法，有符号数的乘法需要同学们考虑如何处理。
4. 在实现算术右移时，可以使用“>>>”符号，但需要对被移动参数附加 $signed() 限制，即规定其为有符号数，例如 $signed(A)。

五． 实验步骤

1. 解压缩 ALU_32bits.zip，打开教师提供的工程文件 **ALU_32bits.xpr**，如图 1-5 所示。目前工程中已经提供了顶层文件（**ALU_32bits_top.v**）、连接 32 位 ALU 与 7 段数码管的文件（**ALU_32bits.sv**）、32 位 ALU 实现文件（**alu.sv**）以及涉及 7 段数码管的文件。同学们只需设计实现图 1-5 中的 ALU 单元，即**只需**

要修改 alu.sv 文件，其余文件不需要有任何改动!

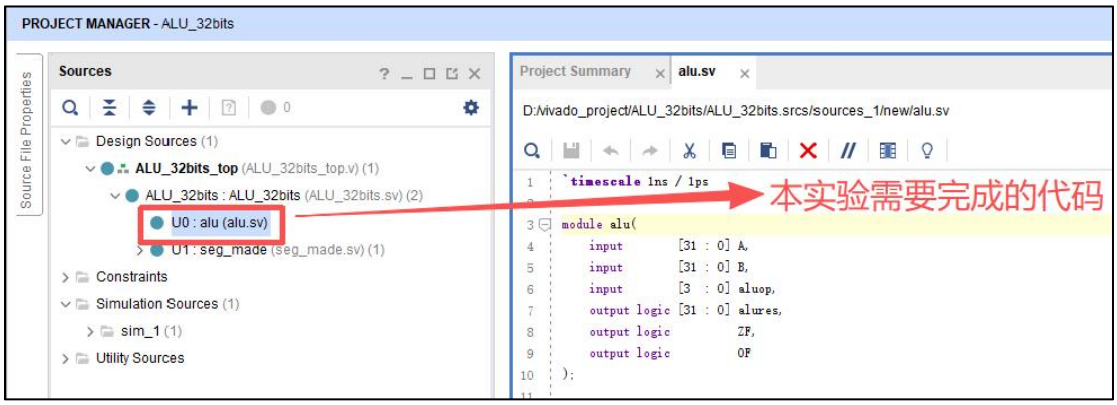


图 1-5 工程 ALU_32bits

2. 在工程中，添加文件 **fulladder.sv**，实现全加器。

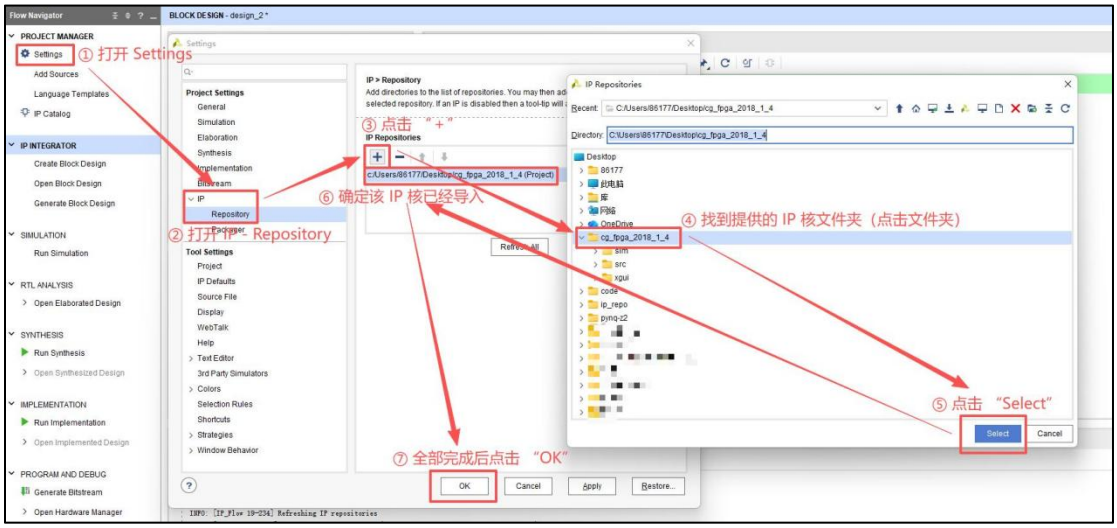
3. 在工程中，添加文件 **rca.sv**，调用全加器，实现 32 位行波进位加法器。

4. 在文件 **alu.sv** 中，通过调用行波进位加法器和添加必要的逻辑电路实现表 1-3 所示的所有运算操作。**注意不要修改 alu.sv 文件的模块定义!**

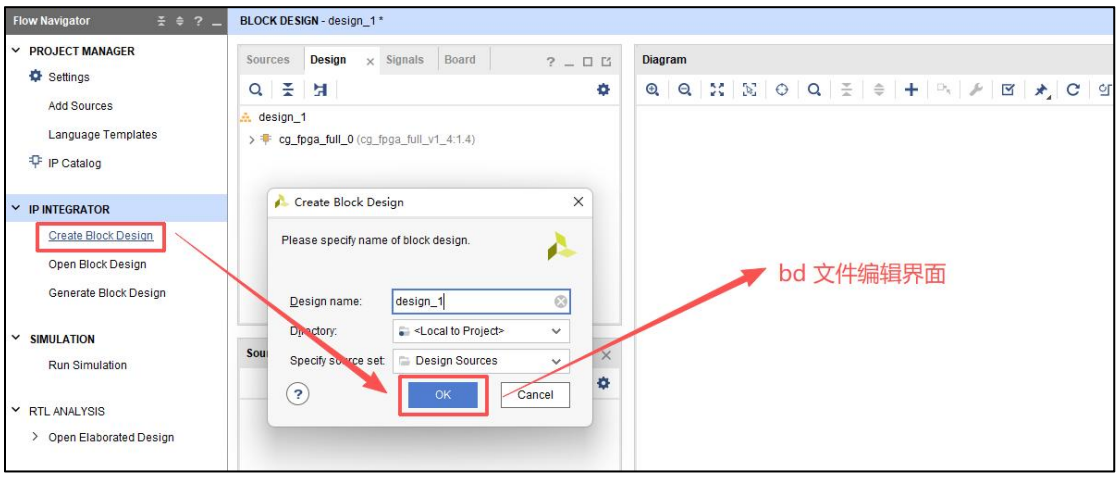
5. 添加测试文件 **ALU_32bits_tb.sv**，对所实现的 32 位 ALU 单元进行行为仿真（**注意：仅对 alu.sv 文件进行仿真即可，不要仿真整个系统**）。仿真过程中必须采用**自动化测试**的方法（注：读取文件时，如果不使用绝对路径，请将存放测试向量的文本文件拷贝到“工程路径/ALU_32bits.sim/sim_1/ behave/xsim”目录下，否则读取文件将失败）。

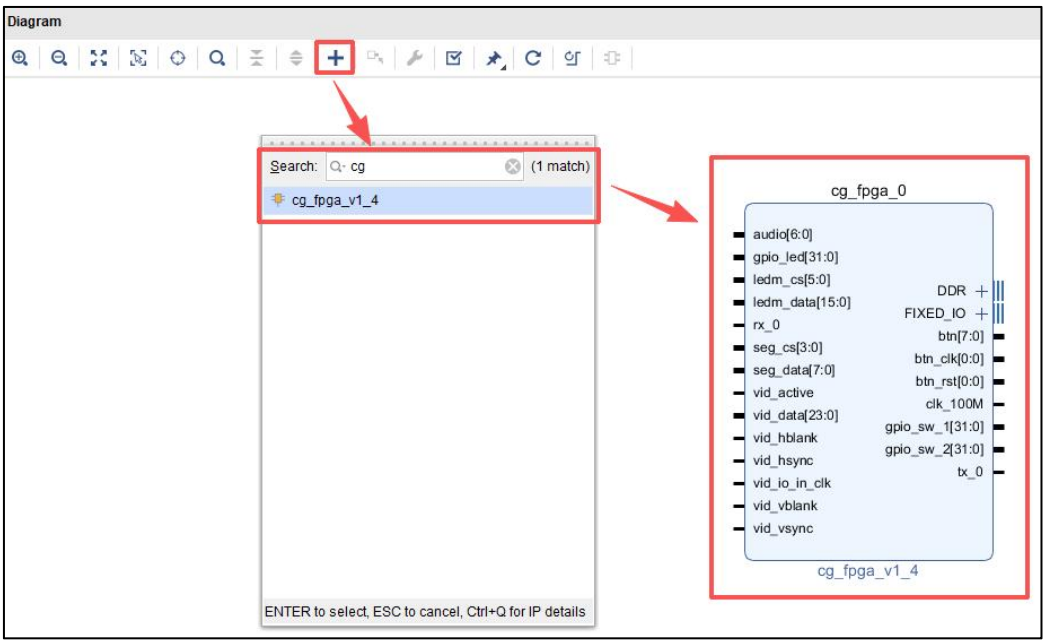
6. 如果第 5 步行为仿真通过，则按以下步骤对工程进行综合与实现：

① 打开页面左侧 “Settings”，点击 IP 目录下的 Repository，点击“+”导入提供的 IP 核文件夹：

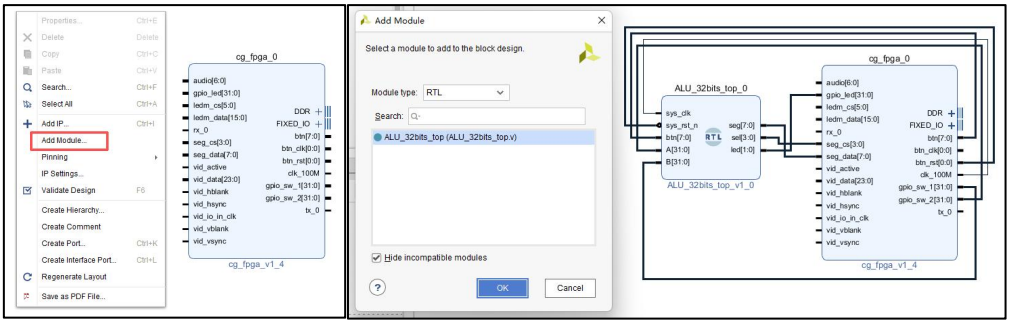


② 点击页面左侧“Create Block Design”，新建 bd 项目，点击“+”搜索“cg_fpga_v1_4”添加，如下图所示：

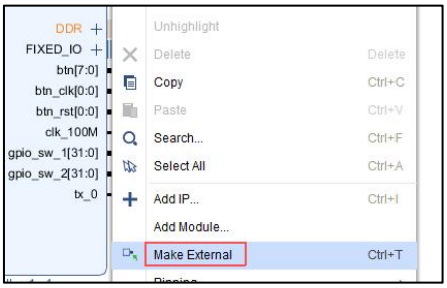


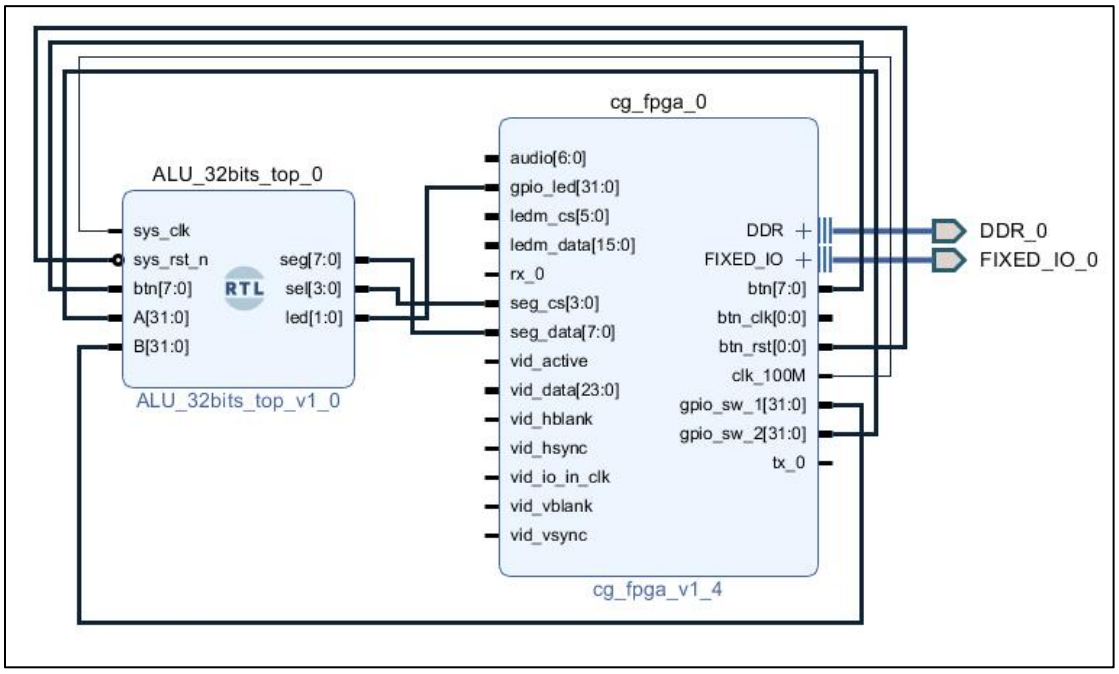


③ 点击鼠标右键，单击“Add Module”，选择“ALU_32bits_top”模块进行添加，并按照下图所示进行连线：

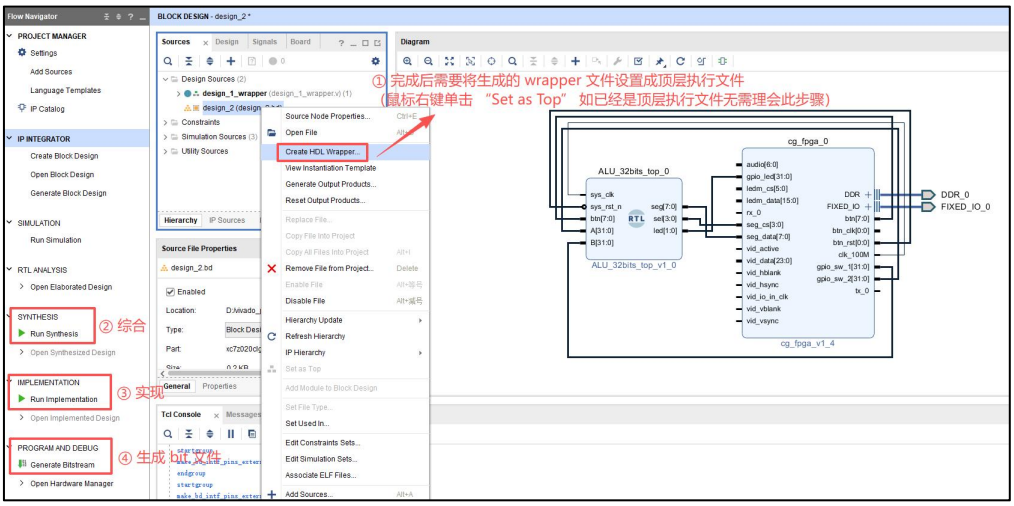


④ 针对“FIXED_IO”和“DDR”两个信号，点击鼠标右键选择“Make External”，最终如下图所示：





⑤ 鼠标右键点击 Sources 目录下的新建 bd 文件，选择“Create HDL Wrapper”生成 Wrapper 文件，并根据下图所示依次进行该文件的综合、实现、bit 文件生成过程（该过程时间较长）：



7. 登录远程 FPGA 硬件云平台，打开本实验对应的实验作业所提供的开发板，如图 1-6 所示。

图 1-6 实验作业开发板打开流程

8. 利用提供的开发板进行实验验证，**操作说明如图 1-7 所示**。

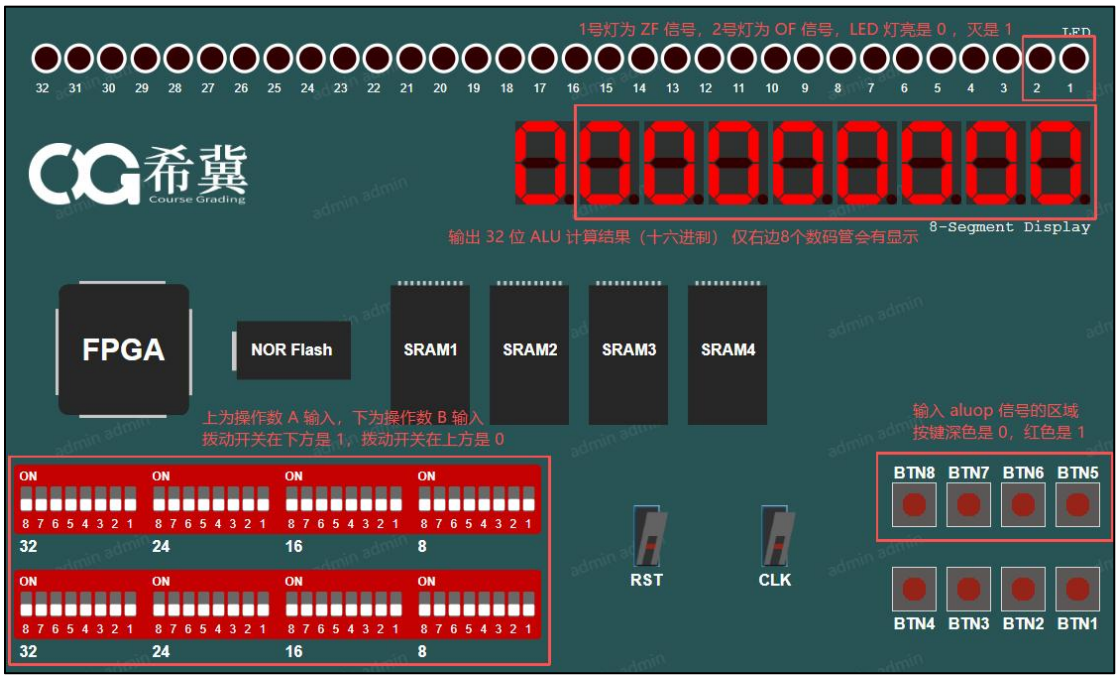


图 1-7 操作说明

9. **按图 1-8 说明**对第 6 步生成的 bit 文件进行烧写，烧写完成后进行实验操作，根据操作结果对实验代码进行修改与完善。

注意：该平台 bit 文件烧写一段时间后会失效，需要重新烧写！

图 1-8 bit 文件烧写说明

10. 在 bit 文件烧写完成，云平台处于 "working" 状态时，可按图 1-9 说明对该实验进行**自动评测**，自动评测**满分 100 分**，通过即算完成实验。

图 1-9 自动评测操作说明

自动评测等待界面如图 1-10 左侧，等待时间可能较长，最终测试完成展示图 1-10 右侧界面，如未达成 100 分可根据输出内容修改代码并重新测试（需要重新生成 bit 文件），**自动评测输出内容说明**见图 1-11：

图 1-10 自动评测界面

图 1-11 自动评测输出内容说明

六． 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

七． 参考内容

教材内容和课件

八． 实验报告

1. 画出实现加/减法运算的逻辑电路原理图（可简化），并说明为什么加/减法可以只使用一个加法器进行实现？
2. 给出有符号数加/减法溢出的判断规则？
3. 给出 32 位 ALU 单元的 SystemVerilog HDL 代码。
4. 给出具有自动化测试功能的仿真程序和对应的波形图截图，并说明为什么选取这些测试向量？