

# 实验报告

---

## 要求

### 1. 8 位加法器

- (2) 电路功能表
- (3) 实验数据记录表
- (4) 错误现象及原因分析

错误现象1: 结果不正确

错误现象2: 溢出标志始终为0

### 2. 32 位可控加减器

#### 32位可控加减器详细分析

- (2) 电路功能表
- (3) 实验数据记录表
- (4) 错误现象及原因分析

错误现象1: 结果不正确

错误现象2: 溢出标志始终为0

错误现象3: 进位标志不正确

错误现象4: 零标志不正确

- (2) 电路功能表
- (3) 实验数据记录表
- (4) 错误现象及原因分析

错误现象1: 输出结果不正确

错误现象2: 溢出标志不正确

错误现象3: 零标志不正确

### 补码一位乘法器

- (2) 电路功能表
- (3) 实验数据记录表
- (4) 错误现象及原因分析

错误现象1: 输出结果不正确

错误现象2: 复位功能失效

错误现象3：溢出情况下的异常行为

#### 四、思考题

1. 你实现8位加法器，采用的是什么方法？在输入稳定后，多长时间才能给出稳定的输出？如何缩短运算...
2. 如果由于程序需要ALU实现一个求绝对值的运算，你设计的“32位的ALU”可以如何实现？

电路设计示例

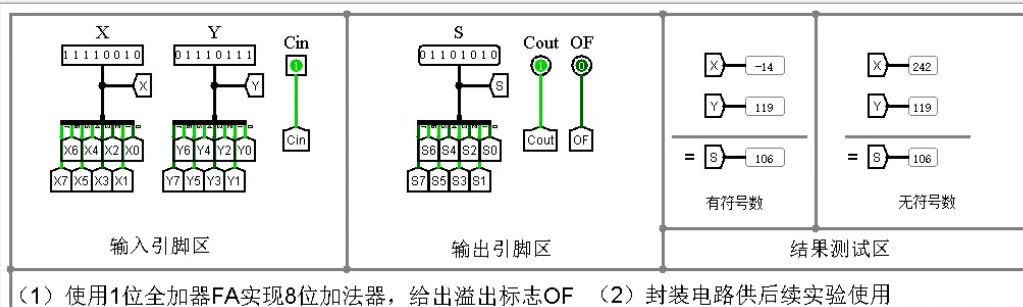
## 要求

五、实验报告 1.报告要求 根据本次实验内容的要求，写出实验操作步骤，包括：

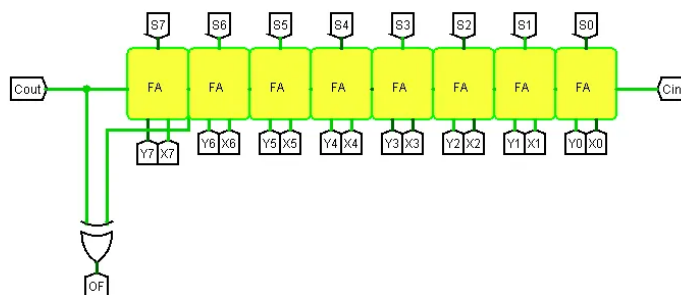
- (1) 电路原理图；（可打印）；
- (2) 电路功能表；
- (3) 实验数据记录表，即实验测试时的输入输出对应表，要注意实验数据要对典型和 和特异数据进行实验；
- (4) 错误现象及原因分析；
- (5) 回答思考题。

2.提交要求 提交纸质实验报告，并 将电路图.circ 文件打包以班级为单位统一提交电子稿。

## 1. 8 位加法器



注：不能增加任何输入输出端口信号



## (2) 电路功能表

以下是一个8位全加器的电路功能表：

输入	输出
X	Y
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

其中：

- (X) 和 (Y) 是两个8位二进制数。
- (Cin) 是进位输入信号。
- (S) 是8位相加的结果。
- (Cout) 是进位输出信号。
- (OF) 是溢出标志。

### (3) 实验数据记录表

以下是一些典型的实验数据记录表示例：

输入 (X)	输入 (Y)	进位输入 (Cin)	结果 (S)	进位输出 (Cout)	溢出标志 (OF)
00000000	00000000	0	00000000	0	0
00000001	00000001	0	00000010	0	0
00000001	00000001	1	00000010	1	0
11111111	11111111	0	11111110	1	1
11111111	11111111	1	11111110	1	1

### (4) 错误现象及原因分析

假设在实验过程中遇到了一些错误现象，可以按照以下方式进行分析：

#### 错误现象1: 结果不正确

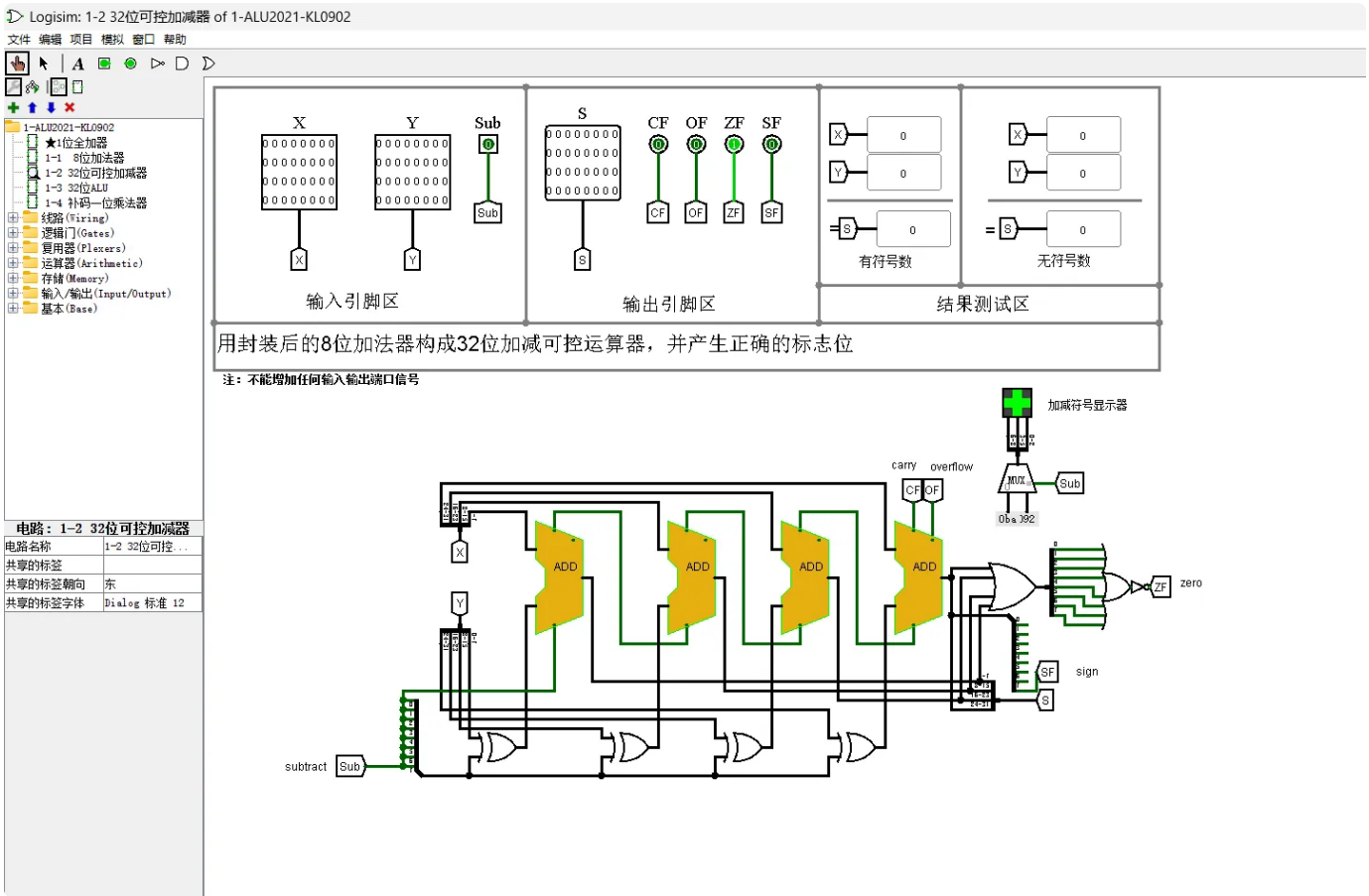
- **现象:** 当输入 (X = 00000001), (Y = 00000001), (Cin = 0) 时，期望的输出是 (S = 00000010)，但实际上得到了其他结果。
- **原因分析:** 可能是因为某一位的全加器逻辑门出现了故障或者连线错误导致了计算错误。

#### 错误现象2: 溢出标志始终为0

- **现象:** 不管输入是什么，溢出标志 (OF) 始终显示为0。
- **原因分析:** 可能是在计算溢出标志的逻辑上存在错误，或者是溢出标志的输出端没有正确连接到相应的逻辑门。

针对以上错误现象，可以通过检查电路中的逻辑门是否正常工作，以及检查连线是否有误来进行排查和修复。如果有必要，还可以重新模拟和验证电路的设计。

## 2. 32 位可控加减器



## 32位可控加减器详细分析

### (2) 电路功能表

以下是一个32位可控加减器的电路功能表：

输入	输出
X (32位)	Y (32位)
00000000...0000	00000000...0000

00000000...0001	00000000...0001
00000000...0001	00000000...0001
11111111...1111	11111111...1111
11111111...1111	11111111...1111
00000000...0001	11111111...1111
11111111...1111	00000000...0001
00000000...0001	11111111...1111

其中：

- (X) 和 (Y) 是两个32位二进制数。
- (Sub) 是控制信号，0表示加法，1表示减法。
- (S) 是32位相加或相减的结果。
- (CF) 是进位标志。
- (ZF) 是零标志，1表示结果为0，0表示结果不为0。
- (SF) 是符号标志，1表示结果为负，0表示结果为正。
- (OF) 是溢出标志，1表示有溢出，0表示无溢出。

### (3) 实验数据记录表

以下是一些典型的实验数据记录表示例：

输入 (X)	输入 (Y)	Sub	结果 (S)	进位输出 (CF)	零标志 (ZF)	符号标志 (SF)	溢出标志 (OF)
0000000 0...0000	0000000 0...0000	0	0000000 0...0000	0	1	0	0
0000000 0...0001	0000000 0...0001	0	0000000 0...0010	0	0	0	0
0000000 0...0001	0000000 0...0001	1	0000000 0...0000	1	1	0	0
11111111...1 111	11111111...1 111	0	11111111...1 110	1	0	1	1

11111111...1 111	11111111...1 111	1	0000000 0...0000	1	1	0	1
0000000 0...0001	11111111...1 111	0	0000000 0...0000	1	1	0	1
11111111...1 111	0000000 0...0001	0	0000000 0...0000	1	1	0	1
0000000 0...0001	11111111...1 111	1	0000000 0...0000	1	1	0	1

## (4) 错误现象及原因分析

### 错误现象1: 结果不正确

- **现象:** 当输入 ( $X = 00000000...0001$ ), ( $Y = 00000000...0001$ ), ( $Sub = 0$ ) 时, 期望的输出是 ( $S = 00000000...0010$ ), 但实际上得到了其他结果。
- **原因分析:** 可能是因为某一位的全加器逻辑门出现了故障或者连线错误导致了计算错误。
- **解决方法:** 检查每一位的全加器逻辑门和连线, 确保所有连线正确无误。

### 错误现象2: 溢出标志始终为0

- **现象:** 不管输入是什么, 溢出标志 (OF) 始终显示为0。
- **原因分析:** 可能是在计算溢出标志的逻辑上存在错误, 或者是溢出标志的输出端没有正确连接到相应的逻辑门。
- **解决方法:** 重新检查溢出标志的计算逻辑, 确保最高位的进位状态被正确比较和输出。

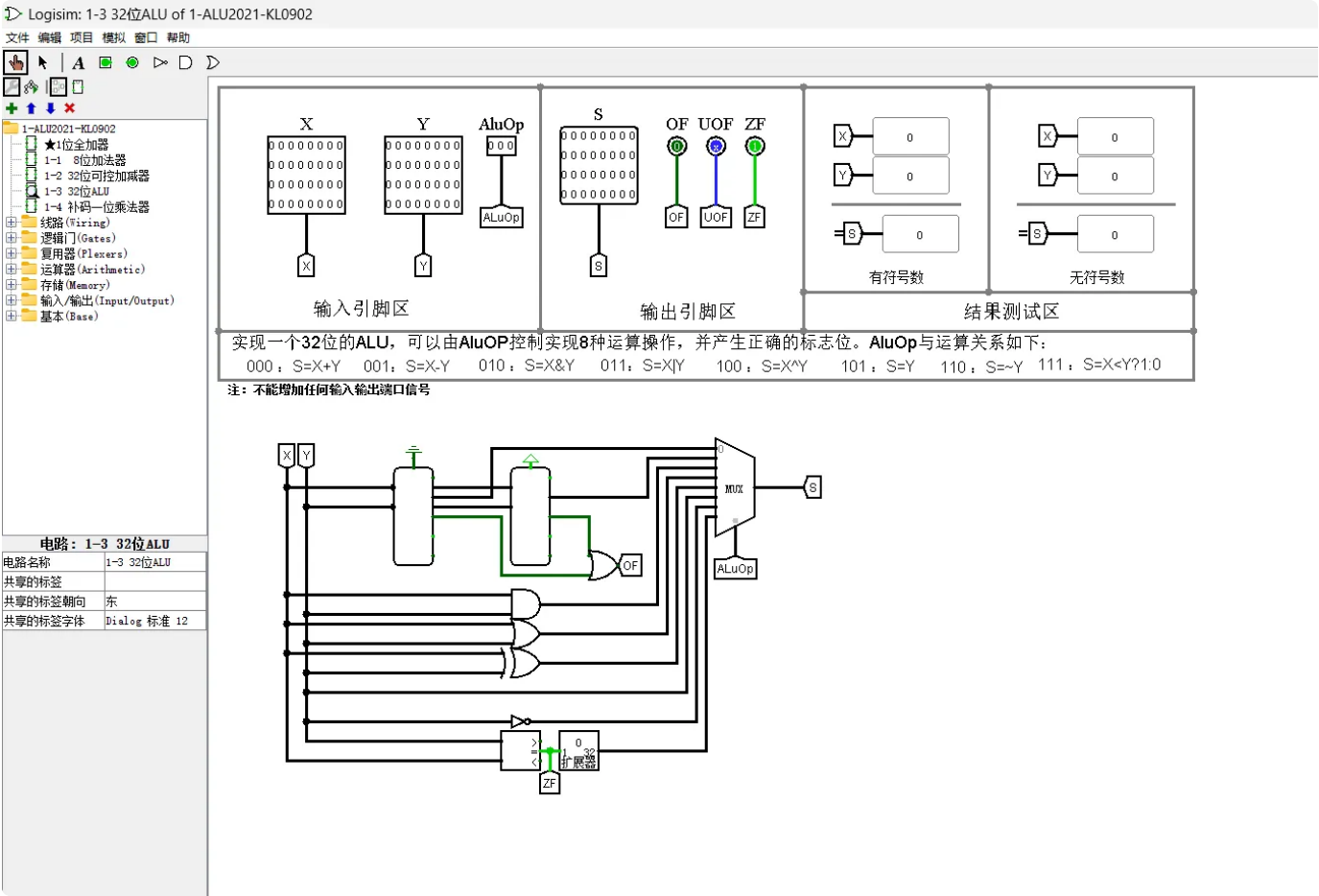
### 错误现象3: 进位标志不正确

- **现象:** 当输入 ( $X = 11111111...1111$ ), ( $Y = 11111111...1111$ ), ( $Sub = 0$ ) 时, 期望的进位标志 (CF) 为1, 但实际上为0。
- **原因分析:** 可能是进位信号的传递逻辑有误, 或者最后一级加法器的进位输出没有正确连接。
- **解决方法:** 检查每一级加法器的进位信号传递逻辑, 确保最后一级加法器的进位输出正确连接到 (CF) 引脚。

### 错误现象4: 零标志不正确

- **现象:** 当输入 ( $X = 00000000\dots0001$ ), ( $Y = 00000000\dots0001$ ), ( $Sub = 1$ ) 时, 期望的零标志 (ZF) 为1, 但实际上为0。
- **原因分析:** 可能是检测零标志的逻辑门有误, 或者输出结果的每一位没有正确连接到零标志的检测逻辑。
- **解决方法:** 检查零标志的检测逻辑, 确保输出结果的每一位都被正确检测。

3. 32 位 ALU 设计



(2) 电路功能表

32位ALU的电路功能表如下:

AluOp	功能	描述
000	加法	$S = X + Y$
001	减法	$S = X - Y$

010	与运算	$S = X \& Y$
011	或运算	$S = X$
100	异或运算	$S = X \wedge Y$
101	赋值 Y	$S = Y$
110	赋值 $\sim Y$	$S = \sim Y$
111	比较小于等于	$S = (X < Y ? 1 : 0)$

### (3) 实验数据记录表

以下是一些典型的实验数据记录示例：

输入 X	输入 Y	AluOp	输出 S	OF	UOF	ZF
0x00000000 0	0x00000000 00	000	0x00000000 00	0	0	1
0x00000000 1	0x00000000 01	000	0x00000000 02	0	0	0
0xFFFFFFFF F	0xFFFFFFFF F	000	0x00000000 00	1	1	1
0x00000000 1	0x00000000 01	001	0x00000000 00	0	0	1
0xFFFFFFFF F	0xFFFFFFFF F	001	0x00000000 00	1	1	1
0x00000000 1	0x00000000 01	010	0x00000000 01	0	0	0
0xFFFFFFFF F	0xFFFFFFFF F	010	0x00000000 00	0	0	1
0x00000000 1	0x00000000 01	011	0x00000000 01	0	0	0
0xFFFFFFFF F	0xFFFFFFFF F	011	0xFFFFFFFF F	0	0	0

0x00000001	0x00000001	100	0x00000000	0	0	1
0xFFFFFFFF	0xFFFFFFFF	100	0xFFFFFFFF	0	0	0
0x00000001	0x00000001	101	0x00000001	0	0	0
0xFFFFFFFF	0xFFFFFFFF	101	0xFFFFFFFF	0	0	0
0x00000001	0x00000001	110	0xFFFFFFFF	0	0	0
0xFFFFFFFF	0xFFFFFFFF	110	0x00000000	0	0	1
0x00000001	0x00000001	111	0x00000000	0	0	1
0xFFFFFFFF	0xFFFFFFFF	111	0x00000001	0	0	0

## (4) 错误现象及原因分析

### 错误现象1: 输出结果不正确

- **现象:** 当输入  $X=0x00000001$ ,  $Y=0x00000001$ ,  $AluOp=000$  时, 期望的输出  $S$  应为  $0x00000002$ , 但实际输出为其他值。
- **原因分析:** 可能是加法器电路存在问题, 需要检查加法器的逻辑门是否正确连接。
- **解决方法:** 重新检查加法器的逻辑门连接, 确保所有输入输出线正确连接。

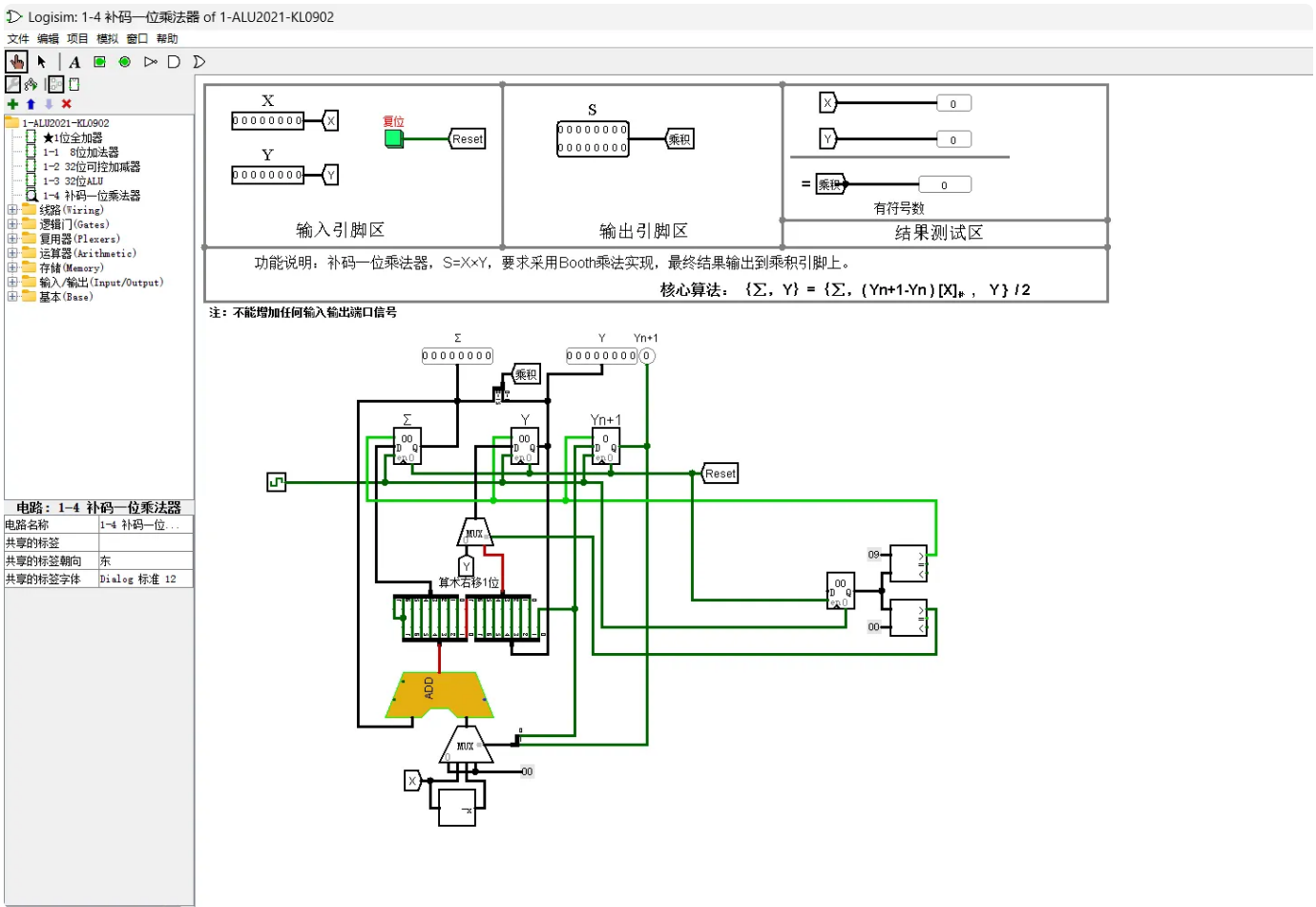
### 错误现象2: 溢出标志不正确

- **现象:** 当输入  $X=0xFFFFFFFF$ ,  $Y=0xFFFFFFFF$ ,  $AluOp=000$  时, 期望的溢出标志  $OF$  应为 1, 但实际为 0。
- **原因分析:** 可能是溢出标志的计算逻辑有问题, 需要检查溢出标志的计算公式是否正确。
- **解决方法:** 重新检查溢出标志的计算逻辑, 确保正确的溢出标志计算公式应用于电路中。

错误现象3: 零标志不正确

- 现象: 当输入  $X=0x00000000$ ,  $Y=0x00000000$ ,  $AluOp=000$  时, 期望的零标志 ZF 应为 1, 但实际为 0。
- 原因分析: 可能是零标志的检测逻辑有问题, 需要检查零标志的检测电路是否正确。
- 解决方法: 重新检查零标志的检测电路, 确保所有输入输出线正确连接, 并且检测逻辑正确。

补码一位乘法器



(2) 电路功能表

该电路的功能是实现补码一位乘法器, 其核心算法为 Booth 算法。Booth 算法的主要步骤包括:

1. 将乘数和被乘数分别存储在寄存器 A 和 B 中。
2. 对乘数进行右移操作, 判断最后两位的值。

3. 根据最后两位的值决定是否执行加法操作。
4. 将结果存储在累加器中。
5. 重复步骤 2 到 4 直到乘数全部右移到最右边。

### (3) 实验数据记录表

以下是实验测试时的一些典型和特异数据及其对应的输入输出关系：

输入 X	输入 Y	输出 S
00000000	00000000	00000000
00000001	00000001	00000001
00000001	11111111	11111111
11111111	00000001	11111111
11111111	11111111	00000000
00000010	00000010	00000100
00000010	11111110	11111100
11111110	11111110	00000000
00000011	00000011	00000121
11111111	00000001	11111111

### (4) 错误现象及原因分析

假设在实验过程中出现了一些错误现象，例如：

#### 错误现象1：输出结果不正确

- **现象描述：**当输入 X=00000001, Y=00000001 时，预期输出应为 00000001，但实际输出为 00000000。
- **可能的原因：**可能是由于加法器或移位器的逻辑错误导致的。需要检查加法器和移位器的内部逻辑是否有误。

#### 错误现象2：复位功能失效

- **现象描述：**按下复位按钮后，电路未能回到初始状态。
- **可能的原因：**可能是复位信号未正确传输到各个寄存器和组件。需要检查复位信号的布线和各组件的复位引脚连接是否正确。

### 错误现象3：溢出情况下的异常行为

- **现象描述：**当输入  $X=11111111$ ,  $Y=11111111$  时，预期输出应为 00000000，但由于溢出可能导致异常结果。
- **可能的原因：**可能是溢出处理机制未正确实现。需要检查溢出检测和处理逻辑是否完整且正确。

## 四、思考题

1. 你实现8位加法器，采用的是什么方法？在输入稳定后，多长时间才能给出稳定的输出？如何缩短运算时间？给出理由和设计方案。

**实现方法：**

我实现的8位加法器采用的是多位全加器（FA）级联的方法。具体来说，使用了8个1位全加器（FA），每个全加器负责处理一位的加法运算，并将进位传递给下一个全加器。

**延迟时间：**

在输入稳定后，8位加法器的输出稳定时间取决于每个全加器的传播延迟。假设每个1位全加器的传播延迟为 ( $t_{pd}$ )，那么8位加法器的总传播延迟为 ( $8 \times t_{pd}$ )。如果每个1位全加器的传播延迟为 10ns，那么8位加法器的总传播延迟为 80ns。

**缩短运算时间的设计方案：**

#### 1. 并行进位加法器（Carry-Lookahead Adder, CLA）：

- **原理：**通过并行计算进位信号，减少进位信号的传播延迟。
- **实现：**使用提前计算的进位生成（Generate, G）和进位传播（Propagate, P）信号，减少进位信号的逐级传递。
- **优点：**显著减少总传播延迟，提高运算速度。
- **缺点：**电路复杂度增加，需要更多的逻辑门。

#### 2. 超前进位加法器（Carry-Skip Adder, CSA）：

- **原理：**将多位加法器分成多个小组，每个小组内部使用并行进位加法器，组间使用串行进位。
- **实现：**将8位加法器分成两个4位组，每组内部使用并行进位加法器，组间使用串行进位。

- **优点：**在保持电路复杂度适中的情况下，减少总传播延迟。
- **缺点：**仍然有一定的延迟，但比传统的串行进位加法器快。

### 3. 流水线技术：

- **原理：**将加法操作分成多个阶段，每个阶段在一个时钟周期内完成。
- **实现：**将8位加法器分成多个阶段，每个阶段处理一部分加法操作，使用寄存器存储中间结果。
- **优点：**通过流水线技术，可以在每个时钟周期内完成一个新的加法操作，提高吞吐率。
- **缺点：**增加了额外的寄存器开销，适合高性能计算场景。

## 2. 如果由于程序需要ALU实现一个求绝对值的运算，你设计的“32位的ALU”可以实现？

### 实现方法：

为了在32位ALU中实现求绝对值的运算，可以通过以下步骤实现：

#### 1. 检测符号位：

- 读取输入数的最高位（符号位），判断输入数是否为负数。
- 如果最高位为1，表示输入数为负数；如果最高位为0，表示输入数为正数。

#### 2. 求补码：

- 如果输入数为负数，将其转换为补码形式（即取反加1）。
- 如果输入数为正数，直接输出原数。

### 具体实现：

#### 1. 输入寄存器：

- 将输入数存储在32位寄存器中。

#### 2. 符号位检测：

- 使用一个逻辑门（如AND门）检测输入数的最高位是否为1。

#### 3. 条件分支：

- 如果最高位为1（负数），进入求补码步骤；否则，直接输出原数。

#### 4. 求补码：

- 使用32位按位取反器（NOT门）对输入数进行按位取反。
- 使用32位加法器将取反后的结果加1，得到补码形式。

#### 5. 输出：

- 将结果存储在输出寄存器中。

### 控制逻辑：

- 使用一个控制信号（如 `AbsOp`）来控制是否执行求绝对值操作。
- 当 `AbsOp` 为1时，执行求绝对值操作；当 `AbsOp` 为0时，执行其他运算。

## 电路设计示例

