**Verilog期末设计实验报告 —七路裁判打分电路**

邢清画 2211999 网络空间安全学院

1. **实验名称：七路裁判打分电路**

实验描述：7个裁判各自在不同时刻打分，满分15分，输出平均整数得分，从第三个裁判给出分数开始，计算平均分时要去掉一个最高分和一个最低分。

1. **实验目的**

本实验旨在通过设计和实现一个数字电路模块，深化对Verilog编程语言的理解，特别是其数据类型、模块结构和控制流语句。此外，实验强调时序逻辑设计的重要性，目的是理解时钟信号在同步数字电路中的作用，并应用这些知识来设计有效的时序电路。

实验还包括对数据处理和算法实现的实践，设计电路以处理和计算7个裁判的打分数据，这涉及数据存储、排序、去除极值和计算平均值等功能，从而增强对硬件中复杂算法实现的理解。

熟悉数字电路设计工具Vivado的使用，包括从编写代码到仿真验证再到电路综合的整个流程，以提高对FPGA编程和设计流程的理解。

最后，实验旨在培养解决电路设计中遇到的问题的能力，例如处理复位逻辑、解决时序问题和优化资源使用等，并将理论知识与实际操作相结合，以加深对数字电路设计理论的理解和应用。

1. **设计思路**

实现一个 Verilog 模块 ScoreProcessor，用于计算七位裁判逐个打分的实时平均分。该模块能够在每次收到新的分数时更新平均分，当收到三个及以上分数时，去除一个最高分和一个最低分后计算平均分。该模块还通过一个清零信号(rst)重置，以便开始处理新一轮的分数。

#### 3.1 输入输出定义

**输入**:

clk: 时钟信号，用于同步分数处理。rst: 异步清零信号，用于重置模块状态。score: 当前裁判的分数。score\_valid: 标识当前分数是否有效。

**输出**:

averageScore: 计算出的平均分数。

#### 3.2 分数处理逻辑

**分数存储**：使用一个寄存器数组 scores[6:0] 存储每位裁判的分数。

**有效分数计数**：使用 num\_scores 记录已经打分的裁判数量。

**分数更新**：在收到有效的分数（score\_valid 为高）时，更新 scores 数组，并增加 num\_scores。

**平均分计算**：

当 num\_scores 少于3时，直接计算所有分数的平均值。

当 num\_scores 大于等于3时，首先找出最高分和最低分，然后计算去除这两个分数后剩余分数的平均值。

#### 3.3 清零和重置逻辑

在 rst 信号为高时，重置 num\_scores、averageScore 以及 scores 数组，以便模块能够开始处理新一轮的分数。

#### 3.4 Testbench 设计

**时钟信号生成**：生成一个测试时钟信号，模拟实际硬件中的时钟。

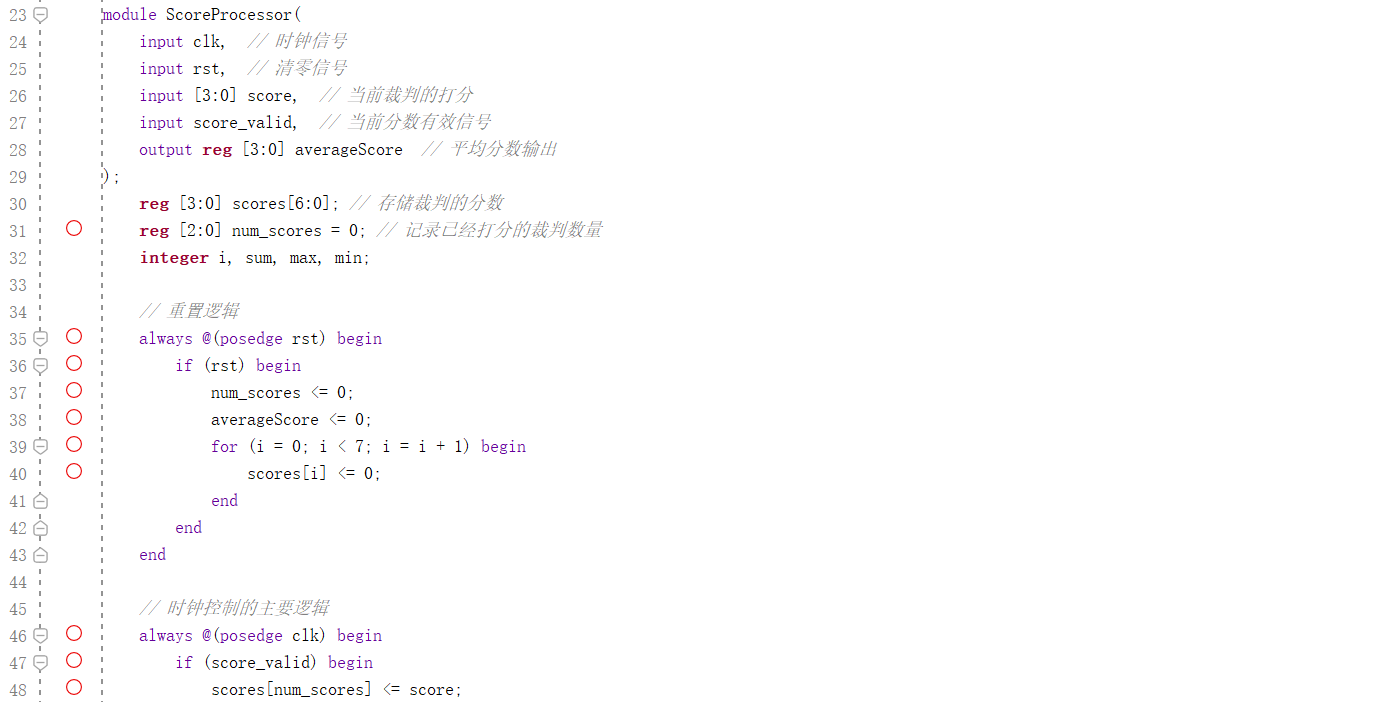
**多轮打分模拟**：模拟多轮裁判打分过程，每轮包括7个分数的输入，每输入一个分数后观察平均分的变化。

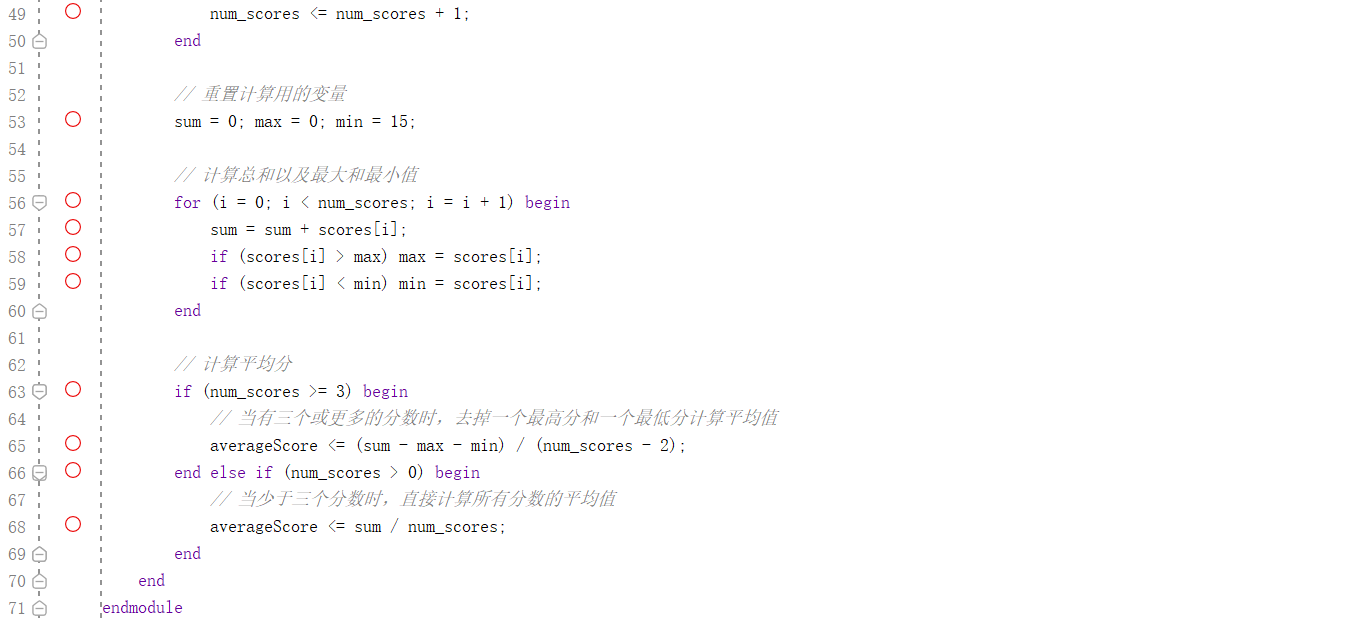
**清零信号模拟**：在每轮打分结束后发出清零信号，重置 ScoreProcessor，然后开始下一轮打分。

**结果监视**：监视并打印每次分数输入和平均分的变化。

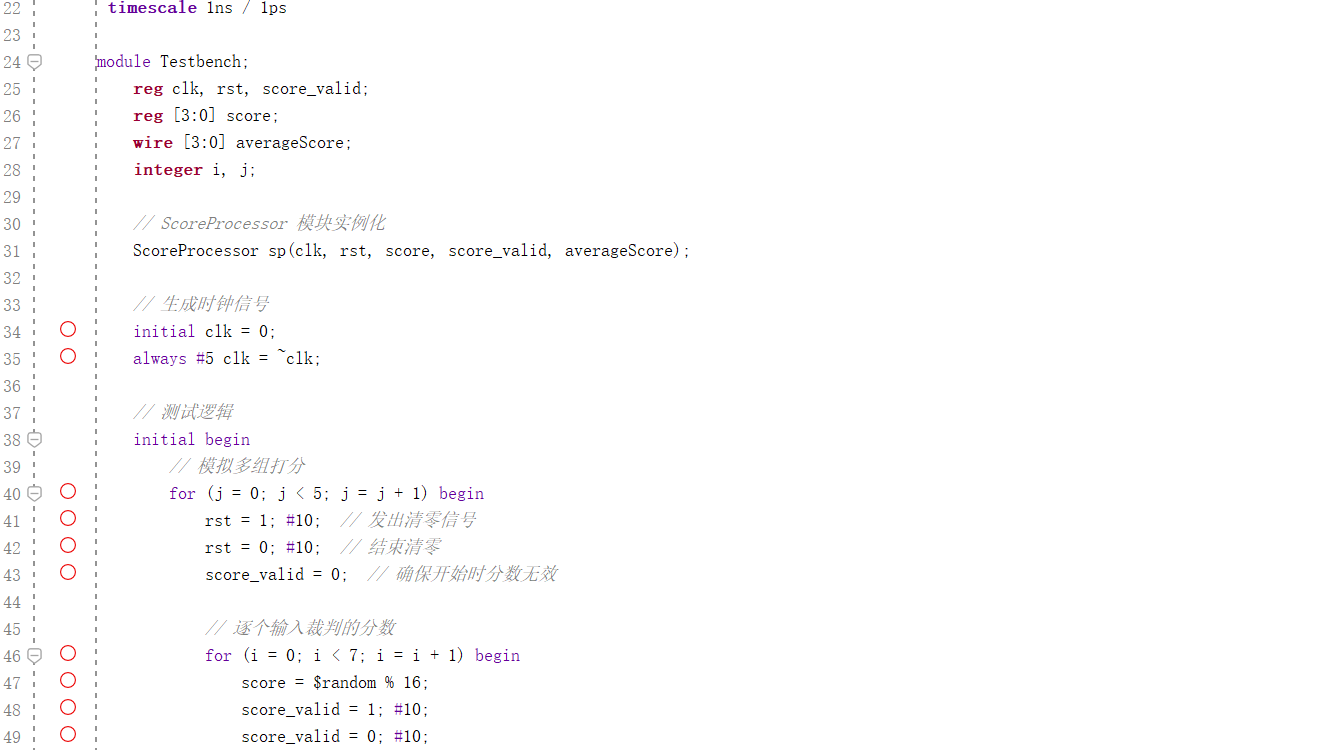
1. **实验代码**

**ScoreProcessor 模块代码**





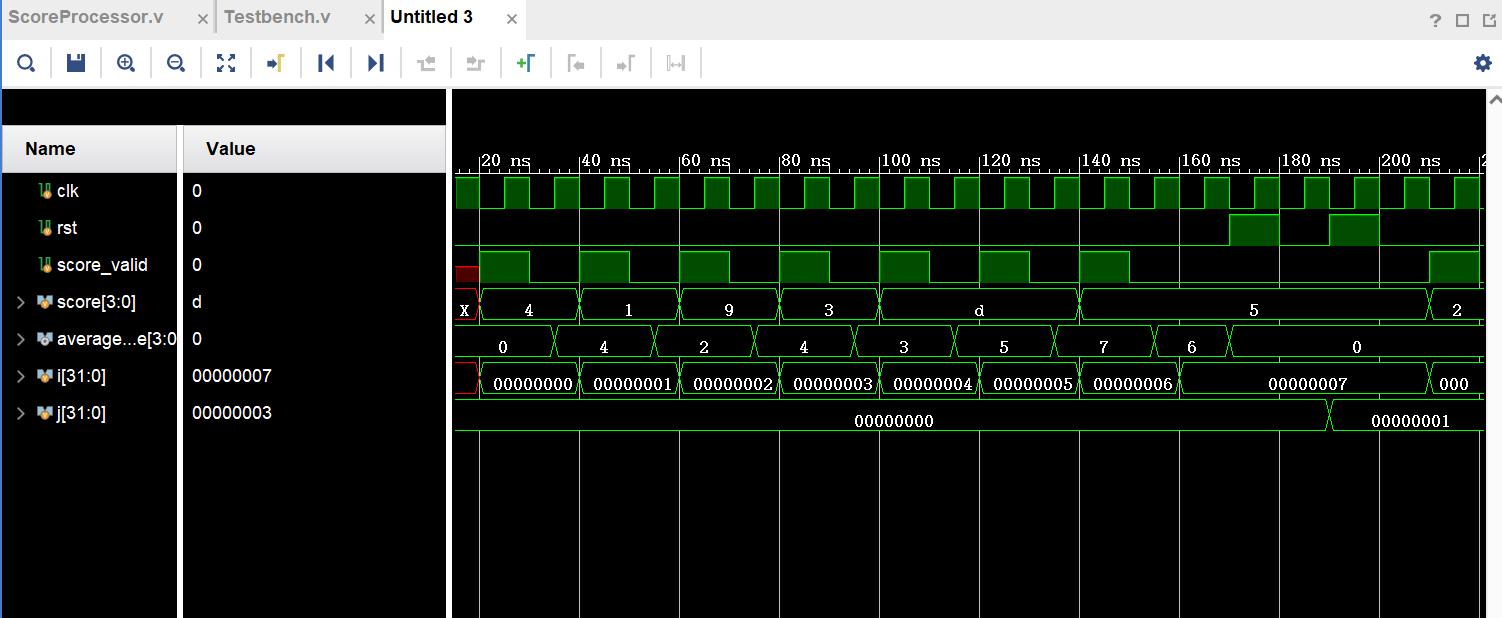
**Testbench 代码**





1. **实验结果**

**波形图**展示：



如图所示，第一至第七位裁判打分依次为：4,1,9,3,13,13,5

第一位裁判打完分后只有一个分数，所以平均分为4；

第二位裁判打完分之后，平均分为（4+1）/2=2.5，向下取整为2；

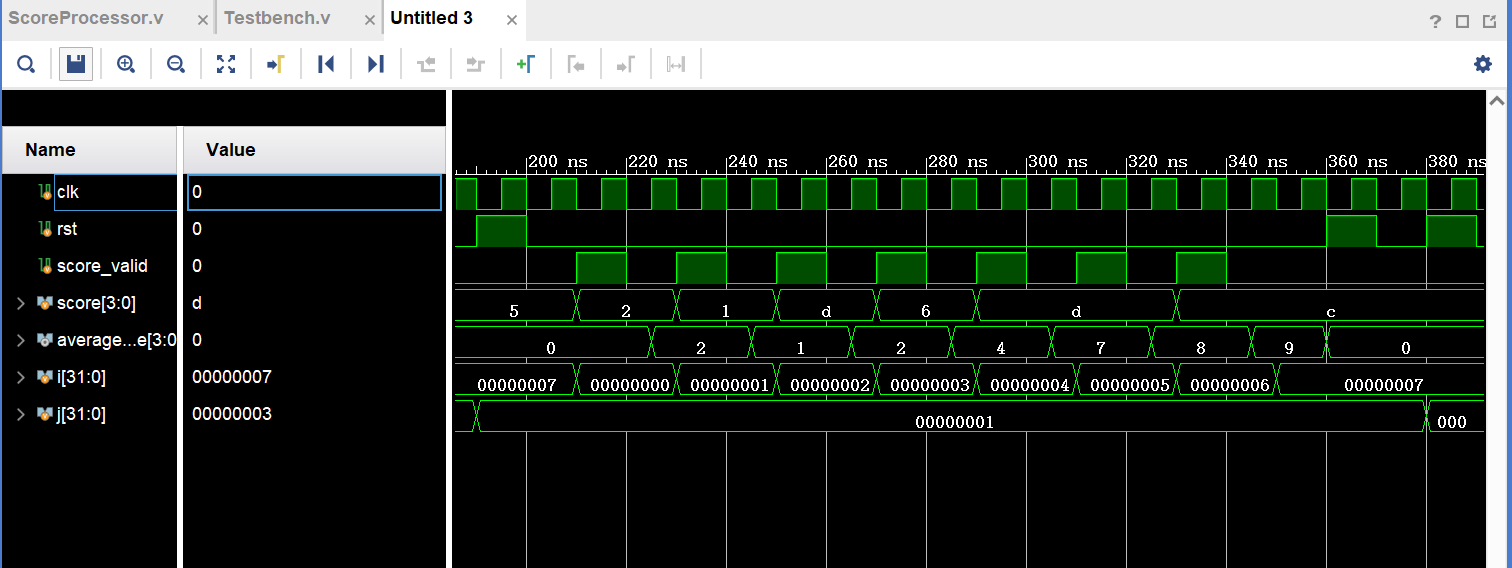
第三位裁判打分后，去掉目前最高分9和最低分1，平均分为4；

第四位裁判打分后，去掉目前最高分9和最低分1，平均分为（4+3）/2=3.5 向下取整为3；

第五位裁判打分后，去掉目前最高分13和最低分1，平均分为（4+3+9）/3=（向下取整）5；

第六位裁判打分后，去掉目前最高分13和最低分1，平均分为（4+3+9+13）/4=（向下取整）7；

第七位裁判打分后，去掉目前最高分13和最低分1，平均分为（4+3+9+13+5）/4=（向下取整）6 ；经检验，答案正确。



如图所示，第一至第七位裁判打分依次为：2,1,13,6,13,13,12

第一位裁判打完分后只有一个分数，所以平均分为2；

第二位裁判打完分之后，平均分为（2+1）/2=1.5，向下取整为1；

第三位裁判打分后，去掉目前最高分13和最低分1，平均分为2；

第四位裁判打分后，去掉目前最高分13和最低分1，平均分为（2+6）/2= 4；

第五位裁判打分后，去掉目前最高分13和最低分1，平均分为（2+6+13）/3=（向下取整）7；

第六位裁判打分后，去掉目前最高分13和最低分1，平均分为（2+6+13+13）/4=（向下取整）8；

第七位裁判打分后，去掉目前最高分13和最低分1，平均分为（2+6+13+13+12）/4=（向下取整）9 ；经检验，答案正确。

1. **思考与总结**

**6.1其他因素考虑**

**6.1.2 有裁判弃票**

在模拟实际投票过程中，可能存在裁判弃票的情况，这时需要对原有的 ScoreProcessor 模块和测试逻辑进行修改。这里的关键是要区分有效的打分和弃票（例如，可以用一个特定的分数表示弃票）。假设使用 score 值为 16 来表示弃票（因为正常分数的范围是 0 到 15），修改代码来忽略这些弃票。

1. **跳过弃票的处理**：在收到新分数时，检查分数是否为弃票标志，如果是，则不将其计入 scores 数组。
2. **更新计算逻辑**：在计算平均分时，只考虑有效的分数。

修改后的模块的关键部分如下：

// 时钟控制的主要逻辑

always @(posedge clk) begin

if (score\_valid && score != 16) begin // 检查分数是否为弃票标志

scores[num\_scores] <= score;

num\_scores <= num\_scores + 1;

end

// 重置计算用的变量

sum = 0; max = 0; min = 15;

integer valid\_scores = 0; // 有效分数的数量

// 计算总和以及最大和最小值

for (i = 0; i < num\_scores; i = i + 1) begin

if (scores[i] != 16) begin // 忽略弃票

sum = sum + scores[i];

if (scores[i] > max) max = scores[i];

if (scores[i] < min) min = scores[i];

valid\_scores = valid\_scores + 1;

end

end

// 计算平均分

if (valid\_scores >= 3) begin

averageScore <= (sum - max - min) / (valid\_scores - 2);

end else if (valid\_scores > 0) begin

averageScore <= sum / valid\_scores;

end

end

### 3.修改测试逻辑：在测试逻辑中需要确保在某些情况下生成弃票的分数。例如，可以随机决定是否为弃票：

// 逐个输入裁判的分数

for (i = 0; i < 7; i = i + 1) begin

if ($random % 5 == 0) begin // 随机决定是否弃票

score = 15; // 弃票标志

end else begin

score = $random % 15; // 正常分数

end

score\_valid = 1; #10;

score\_valid = 0; #10;

end

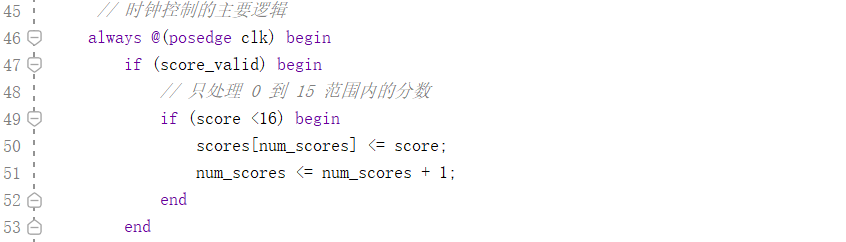
在实际测试中，需要根据实际情况调整弃票的概率和逻辑。

**6.1.2重复或无效的分数输入**

确保模块能够正确处理重复输入的分数或无效的分数信号。例如，如果同一裁判不小心输入了两次分数，或者分数输入信号不稳定，模块应该能够识别并处理这种情况。能够识别并处理异常或不合理的分数，例如超出预定范围的分数。

**1.异常分数处理**

增加了一个条件检查，只有当 score 小于16时，才会将其计入 scores 数组，并增加 num\_scores 的值。这样可以确保只有有效的分数才会被处理。

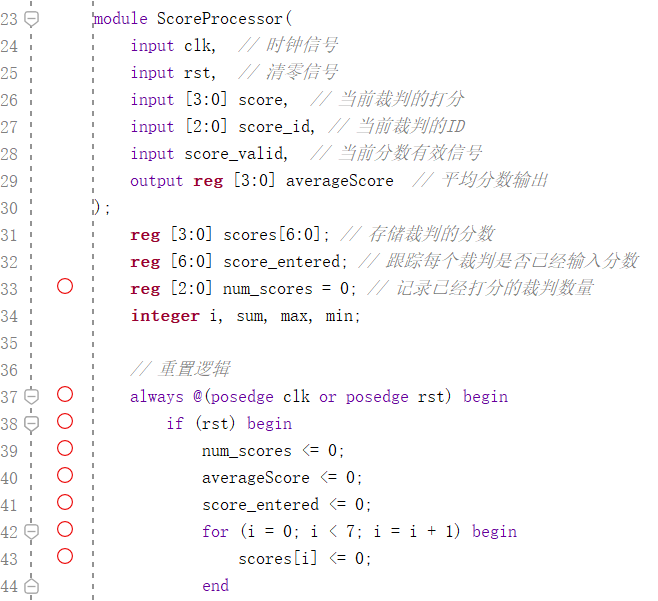
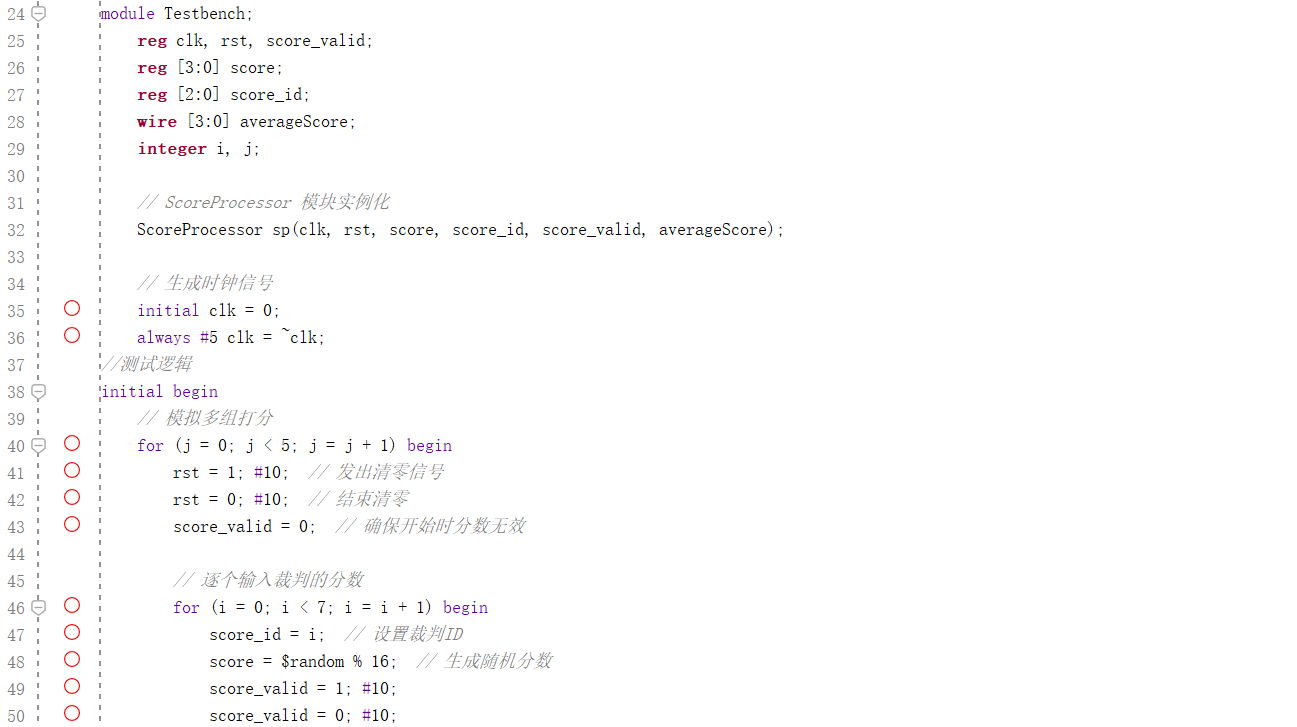


由于 Verilog 使用无符号数，因此不需要专门处理负数，任何负数值在硬件中都会被解释为一个大于或等于0的数。

**2.裁判输入分数数量不唯一**

当一个裁判输入多个分数时，可以采用取第一次有效值作为最终分数。由于要判断每个裁判输入的分数是否有效，以及每个裁判输入的分数是否唯一，**这要求需要知道哪个分数对应于哪个裁判，需要额外的输入来表示当前输入的分数是哪个裁判的（score\_id），**下面是解决这类问题的原始代码**：**

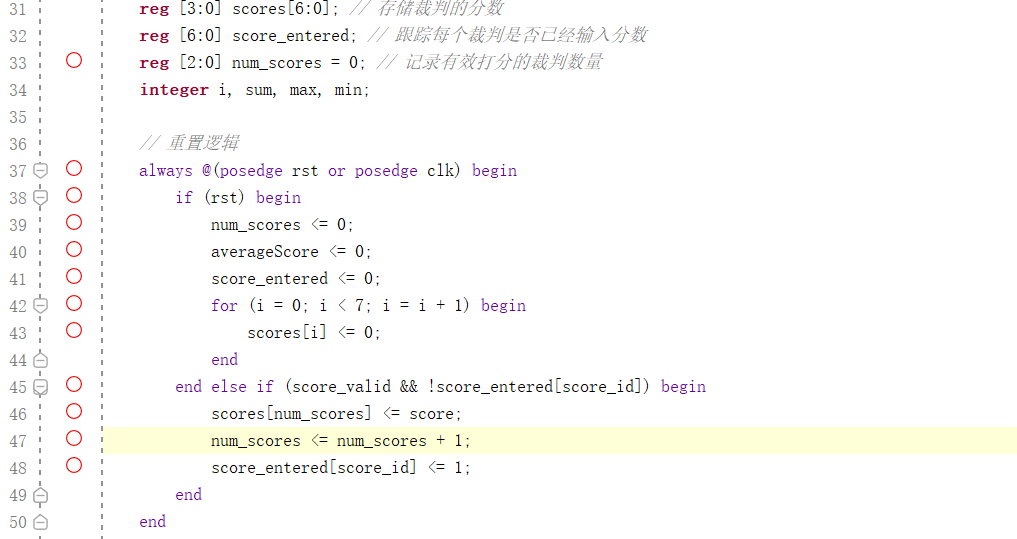
**ScoreProcessor**

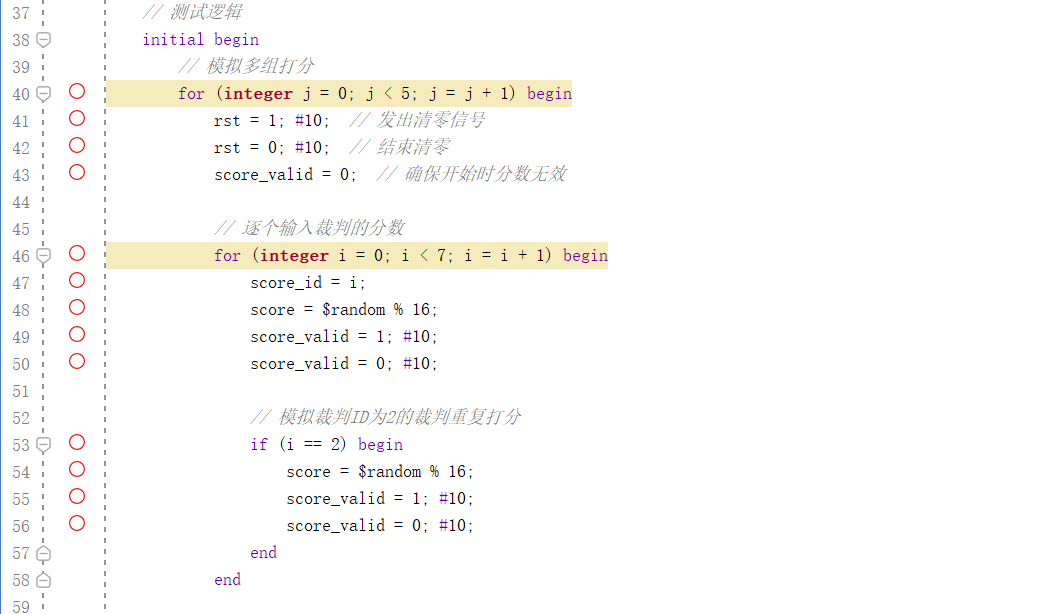
 Testbench

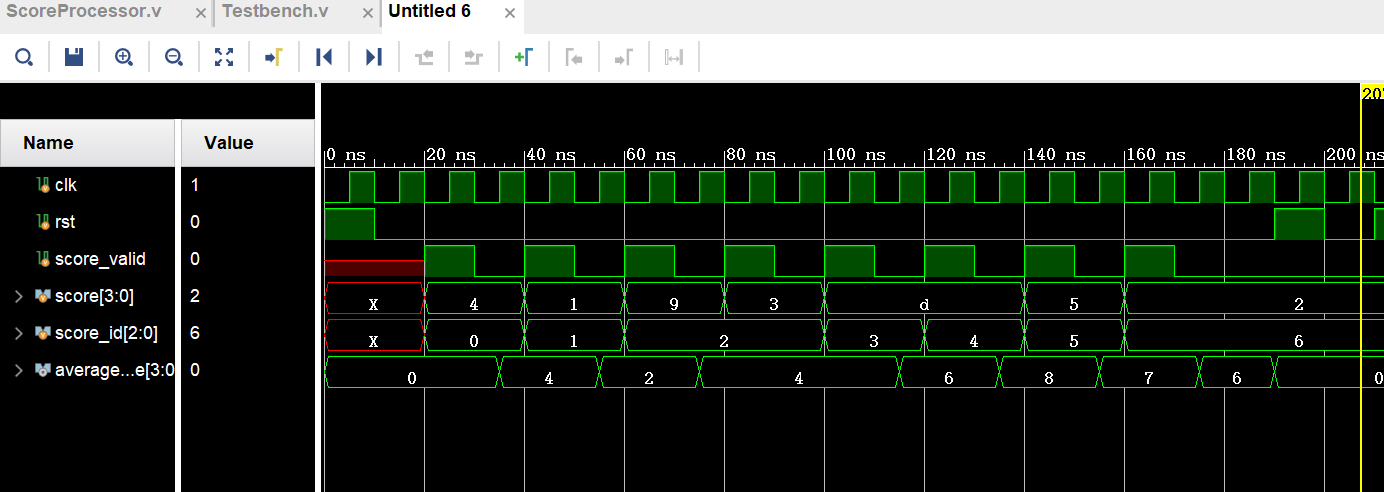


**处理办法1：同一裁判的多次打分只取第一次的分数**

在这个版本中使用了两个 always 块：一个用于处理分数的输入和重置逻辑，另一个用于计算平均分。当一个新的分数输入时，如果是首次输入，它将被记录并用于后续的平均分计算。这个设计确保了即使一个裁判尝试多次输入分数，也只会记录他们的第一次输入。







**可以看到第三位裁判打分两次，记录了第一次的9分作为后续计算的值。**

当然也可以由裁判自行决定取哪一次的分数，只需模拟裁判进行选择即可

**6.2总结**

这个实验通过设计和实现一个数字电路模块，深化了对 Verilog 编程语言的理解，尤其在数据类型、模块结构和控制流语句方面。突出了时序逻辑设计的重要性，特别是在理解时钟信号在同步数字电路中的作用，并运用这些知识来设计有效的时序电路。实验还包括了对数据处理和算法实现的实践，设计电路以处理和计算7个裁判的打分数据，这涉及数据存储、排序、去除极值和计算平均值等功能，加深了对硬件中复杂算法实现的理解。通过熟悉数字电路设计工具Vivado的使用，从编写代码到仿真验证再到电路综合的整个流程，实验有助于提高对FPGA编程和设计流程的理解。培养了解决电路设计中遇到的问题的能力，例如处理复位逻辑、解决时序问题和优化资源使用等，并将理论知识与实际操作相结合，加深了对数字电路设计理论的理解和应用。

在统计多路裁判打分取平均分的影响因素时，可能有更多的情况需要考虑，或者有更多的解决方案。