2023（第五届）集成电路EDA设计精英挑战赛

赛题指南

一、 赛题名称

统计静态时序分析算法实现

二、 命题单位

北京华大九天科技股份有限公司

三、 赛题 Chair

曹鹏（东南大学）

四、 赛题背景

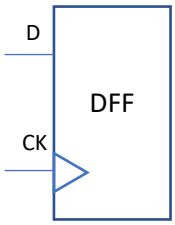
集成电路中若存在建立时间或保持时间的时序违例会导致集成电路实际工作速度不能满足设计要求，甚至会产生功能性错误而导致设计失败。作为集成电路时序违例的检查工具，静态时序分析(STA)是保障集成电路流片成功的一个重要环节。STA 工具的一个能够被广泛应用的原因是其算法复杂度与集成电路的设计复杂度成线性关系，这个特性保证了 STA 工具能够在合理的时间和内存要求下完成对集成电路的分析。

传统 STA 工具将制造工艺参数，如离子注入浓度，氧化层厚度等认为是一个固定值，因此计算得出器件和互连线的延时(delay)也是一个固定值。这是一个与实际事实有明显不符的假设。实际制造过程中由于设备等各种因素的限制和制约，哪怕是同一块芯片上面同样类型的多个逻辑门，其 delay 等参数也不可能是完全一样的。在传统 STA 工具为了引入这种不确定性就要芯片设计者根据自己的设计经验人为地引入一个悲观量，在先进制造工艺下悲观量的选取已经成为了一个越来越困难的任务。为了模拟这种不确定性，人们提出了统计 STA(SSTA)的概念。

在 SSTA 工具中，器件的延时不再是一个固定值，而是一个正态分布，时序路径的到达时间(arrival time)也是以正态分布来描述的。相比于传统 STA 算法，SSTA 虽然是一个更合理的概念，但是 SSTA 对于 EDA 企业也提出了很多的挑战。

五、 题目描述

不管是传统 STA 算法还是统计 STA 算法，STA 工具的运行一般都分为相近的几个阶段，首先是读入设计文件并建立时序图，然后进行正向传播，从时序图的根节点开始逐步计算出所有节点处的 arrival time。arrival time 为当前节点的扇入逻辑锥内所有时序路径的起点的信号传播至当前节点的 delay 累加值。arrival time 为 STA 工具中的核心数据结构，一个合理的 arrival time 应包含所有可能的时序路径起点传播至此的最长路径以及最短路径的 delay 累加值。合理设计 arrival time 数据结构以保证存储足够信息的前提下不包含冗余信息，以及对应的累加和 max 操作是实现一个 STA 工具的核心问题。正向传播结束后根据需要在指定的时序路径的终止节点(endpoint)或全部终止节点处计算时序宽裕量，即 slack 值。如需根据 slack 值找到对应的时序路径，则从对应的 endpoint 处进行反向搜索得到时序路径并输出。



对于如图所示的上升沿触发的触发器，DFF/D 处下降信号的 slack 应用如下公式计算：

reqTime = clock period + mean(DFF/CK, min, rise) – N * sigma(DFF/CK, min, rise) – setupValue(DFF/CK, DFF/D, fall)

arrTime = mean(DFF/D, max, fall) + N * sigma(DFF/D, max, fall)

slack = reqTime – arrTime

其中 N 为用户指定的可信度范围，本题中我们定为 3。design 为单一时钟，clock period 定为 10。mean(vertex, max/min, rise/fall)和 sigma(vertex, max/min, rise/fall)为在节点 vertex 上，数据跳变方向为 rise/fall，且在 max/min 环境下的 mean 和 sigma 值。setupValue(vertex1, vertex2, rise/fall)为从 vertex1 到 vertex2，且 vertex2 的跳变方向为 rise/fall 的建立时间要求。

本题目中，给定时序图中每条边的信息，包括边的源节点以及目标节点的名字，和每条边 delay 的均值(mean)和标准差(sigma)，同时另外提供时序图中所有时序路径的起点(startpoint)和终点(endpoint)列表，计算时序图中每一个节点的 arrival time，即实现正向传播过程。正向传播结束后应给出对于当前集成电路设计，即当前时序图，arrival time 的内存占用情况和数量等统计信息。给定的时序图中不会存在任何环，并不会有锁存器(latch)结构。

六、 题目要求及考查重点：

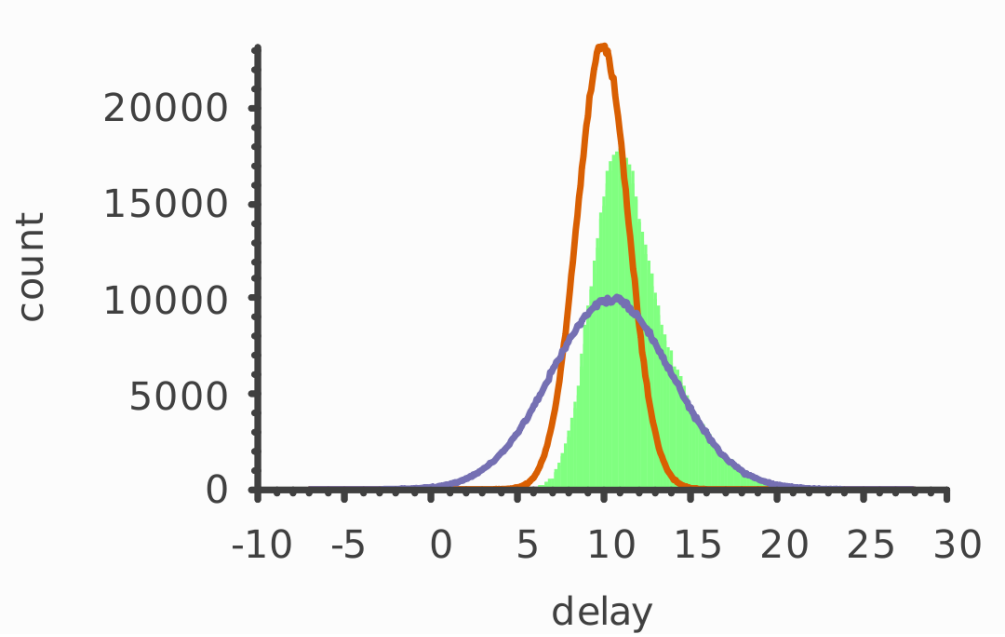
a) 实现考虑统计信息的累加和 max 操作。题目后续会给出一个简单例子在不考虑 sigma 的情况下的详细计算过程，参赛队伍可以以此为参考先理解累加和 max 操作的用途，再考虑实现。

b) 计算所有在 endpoints.list 文件中列出的时序路径终点处的 slack。

c) 加分项:

i. 计算整个时序图中所有顶点的 global slack。对于任一个节点，它的 global slack 为经过它的所有可能的时序路径的最差(即最小)的 slack。global slack 计算算法时间复杂度应与图时序图中节点数量近似成线性关系。使用穷举算法则本项不得分。

ii. arrival time 中 skewness 的考虑。两个正态分布取 max 操作会引入非线性成分，从而得到一个大多数情况下为正偏斜(positively skewed)的分布，如下图：



图中红色和蓝色为两个正态分布的概率密度函数曲线，红色为 $N(10, 1.5^2)$，蓝色为 $N(10.5, 3.5^2)$，绿色阴影部分为对两个正态分布的 max 操作进行蒙特卡罗模拟得到的结果。我们可以很明显地看到得出的概率密度分布是一个正偏斜的分布，不符合正态分布的对称特性。

此种分布应如何模拟，并实现其对应的累加和 max 操作？

若参赛队伍选择考虑 skewness，则 global slack 中 reqTime 和 arrTime 以及详细时序报告中的 total delay 计算方法会和本赛体描述不同。具体计算公式由参赛队伍自行提出并在提交的报告中说明。

d) 本题不考虑 CPPR(clock path pessimism removal)，即在 slack 计算过程中不用考虑消去 CPPR 的影响。

e) 工具实现使用 C++，GCC 版本为 9.4.0，不允许使用除 STL 外其他库文件。若涉及到多线程的实现，需要提供参数或命令设置最大线程数。评测期间以最大线程数为 16 线程为准。

f) 本赛题不强制要求公开源代码，但是要求在提交的设计报告中详细描述算法和参赛队伍认为的设计亮点的细节。

七、 输入输出文件与格式说明

a) 输入文件

i. 时序图各个边的信息 edge_data.csv：文件格式为标准的逗号分割值(CSV)文件。格式说明如下：

1. 文件每列数据内容分别为：

a) from vertex: 时序图中边的源节点名称

b) to vertex: 目的节点名称

c) sense: 边的 unateness 和功能性描述。可能的值为 pos_unate, neg_unate, rising_edge, falling_edge。具体每种可能值的意义解释如下：

i. pos_unate：目的节点输出信号方向与源节点相同。多见于 buffer, 与门，或门等逻辑。

ii. neg_unate: 目的节点输出信号方向与源节点想法，即源节点处的上升信号会在目的节点处产生一个下降信号，反之亦然。多见于反相器（非门）逻辑。

iii. rising_edge: 源节点的上升沿会在目的节点同时产生上升沿信号和下降沿信号，多见于上升沿触发的 D 触发器。

iv. falling_edge: 源节点的下降沿会在目的节点同时产生上升信号和下降信号，多见于下降沿有效的 D 触发器。

d) max rise delay mean: max 条件下，to vertex 输出为上升信号时的 delay mean 值。

e) max rise delay sigma: max 条件下，to vertex 输出为上升信号时的 delay sigma 值。

f) max fall delay mean: max 条件下，to vertex 输出为下降信号时的 delay mean 值。

g) max fall delay sigma: max 条件下，to vertex 输出为下降信号时的 delay sigma 值。

h) min rise delay mean: min 条件下，to vertex 输出为上升信号时的 delay mean 值。

i) min rise delay sigma: min 条件下，to vertex 输出为上升信号时的 delay sigma 值。

j) min fall delay mean: min 条件下，to vertex 输出为下降信号时的 delay mean 值。

k) min fall delay sigma: min 条件下，to vertex 输出为下降信号时的 delay sigma 值。

2. 第一行为表头，即每列数据名称，该行不包含任何设计信息，仅仅为了方便对应各列数据。

3. 从第二行开始为实际的设计数据。

ii. 建立时间检查列表 setup_check.csv: CSV 文件，记录了所有检查约束信息。

1. 文件每列数据分别为：

a) from vertex: 建立时间检查的源节点，一般为 D 触发器的 clock pin。

b) to vertex:建立时间检查的目标节点，一般为 D 触发器的 data pin。

c) sense:建立时间检查类型，该值可以为 setup_rising 和 setup_falling。

i. setup_rising: 源节点的上升沿触发一个建立时间的检查。后面的 rise constraint 和 fall constraint 指的是检查的目标节点的上升沿和下降沿两种不同情况下的建立时间检查要求。多见于上升沿有效的 D 触发器。另外本题目不考虑建立时间的 sigma，后面给出的 rise constraint 和 fall constraint 的值均为均值。

ii. setup_falling: 同 setup_rising，只是出发建立时间检查的边沿为下降沿。多见于下降沿有效的 D 触发器。

d) rise constraint: 目标节点为下降沿时的约束时间。

e) fall constraint: 目标节点为上升沿时的约束时间。

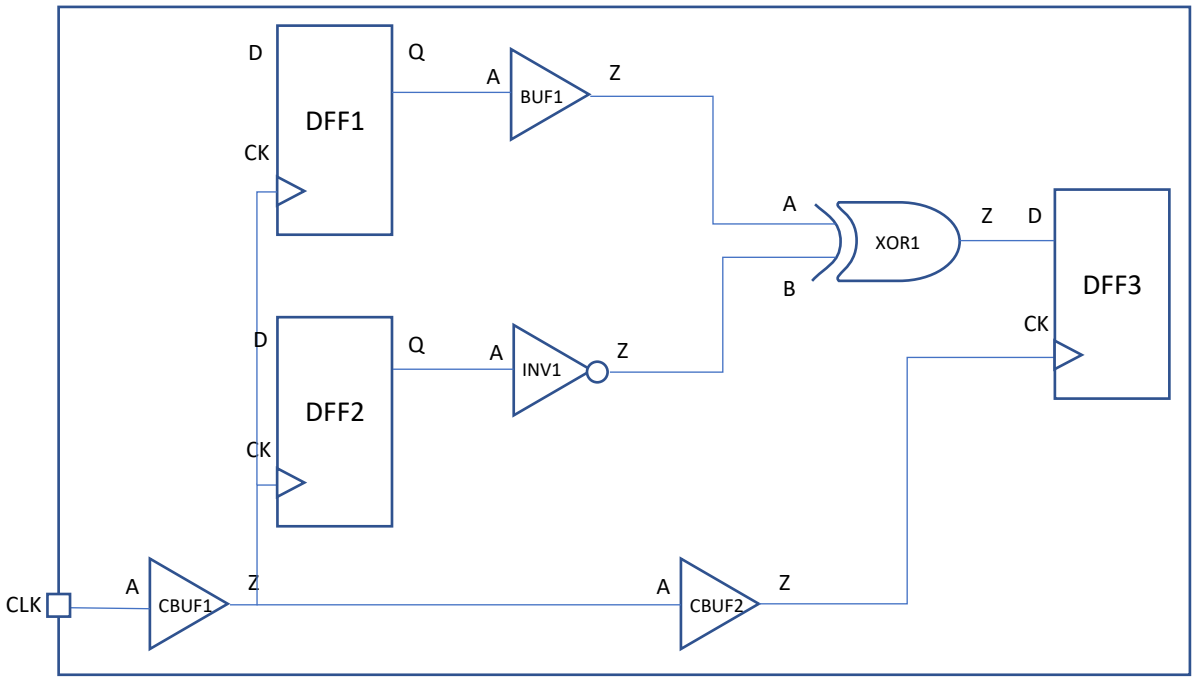
iii. 所有时序路径起点列表 startpoints.list: 文本文件，每行为一个节点的名字。

iv. 所有时序路径终点列表 endpoints.list: 文本文件，每行为一个时序路径终点的名字。

b) 输出文件：

i. global_slack.csv: global slack 报告。格式为三列的 CSV 文件，每行为一个时序节点的 global slack 信息，每列分别为 vertex 名称，rise slack，fall slack。

c) 举例：假设我们有以下设计：



对于此设计，edge_data.csv 应包含如下内容：

```
from vertex, to vertex, sense, max rise delay mean, max rise delay sigma, max fall delay mean, max fall delay sigma, min rise delay mean, min rise delay sigma, min fall delay mean, min fall delay sigma

CLK, CBUF1/A, pos_unate, …(delay 数值省略，下同)

CBUF1/A, CBUF1/Z, pos_unate, …

CBUF1/Z, DFF1/CK, pos_unate, …

DFF1/CK, DFF1/Q, rising_edge, …

DFF1/Q, BUF1/A, pos_unate, …

BUF1/A, BUF1/Z, pos_unate, …

BUF1/Z, XOR1/A, pos_unate, …

XOR1/A, XOR1/Z, pos_unate, …

XOR1/A, XOR1/Z, neg_unate, …

XOR1/Z, DFF3/D, pos_unate, …

CBUF1/Z, DFF2/CK, pos_unate, …

DFF2/CK, DFF2/Q, rising_edge, …
```

```
DFF2/Q, INV1/A, pos_unate, …

INV1/A, INV1/Z, neg_unate, …

INV1/Z, XOR1/B, pos_unate, …

XOR1/B, XOR1/Z, pos_unate, …

XOR1/B, XOR1/Z, neg_unate, …

CBUF1/Z, CBUF2/A, pos_unate, …

CBUF2/A, CBUF2/Z, pos_unate, …

CBUF2/Z, DFF3/CK, pos_unate, …
```

注意 XOR1/A 和 XOR1/B 到 XOR1/Z 共有 4 条边，原因为异或门为逻辑在布尔代数中为典型的 non-unate 或 bi-nate，此种逻辑在本题中均拆开成为两条边，一条为 pos_unate，一条为 neg_unate。

setup_check.list 应为如下内容：

```
from vertex, to vertex, sense, rise constraint, fall constraint

DFF3/CK, DFF3/D, setup_rising, …
```

startpoints.list 应为如下内容：

```
DFF1/CK

DFF2/CK

DFF3/CK
```

endpoints.list 应为如下内容：

```
DFF3/D
```

八、 计算详解及示例：

时序图中起点的 max rise arrival，max fall arrival， min rise arrival，min fall arrival 均为 0.

当 arrival time 经过 sense 为 pos_unate 的 edge 时，目标节点处 max rise arrival = 源节点处 max rise arrival + N(max rise delay mean, max rise delay sigma^2)。其中 N(max rise delay mean, max rise delay sigma^2)指以 max rise delay mean 为均值，max rise delay sigma 为标准差的正态分布，下同；目标节点处 max fall arrival = 源节点处 max fall arrival + N(max fall delay mean, max fall delay sigma^2)。

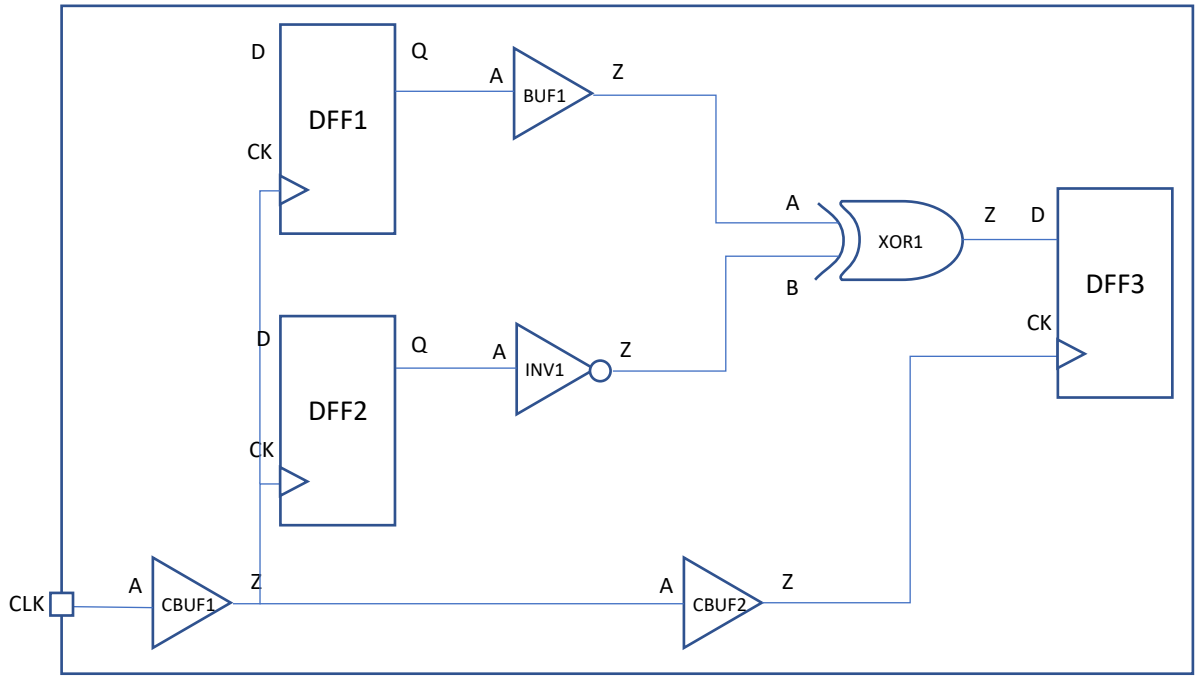
当 arrival time 经过 sense 为 neg_unate 的 edge 时，目标节点处 max rise arrival = 源节点处 max fall arrival + N(max rise delay mean, max rise delay sigma^2)；目标节点处 max fall arrival = 源节点处 max rise arrival + N(max fall delay mean, max fall delay sigma^2)。

当 arrival time 经过 sense 为 rising_edge 的 edge 时，目标节点处 max rise arrival = 源节点处 max rise arrival + N(max rise delay mean, max rise delay sigma^2)；目标节点处 max fall arrival = 源节点处 max rise arrival + N(max fall delay mean, max fall delay sigma^2)。

当 arrival time 经过 sense 为 falling_edge 的 edge 时，目标节点处 max rise arrival = 源节点处 max fall arrival + N(max rise delay mean, max rise delay sigma^2)；目标节点处 max fall arrival = 源节点处 max fall arrival + N(max fall delay mean, max fall delay sigma^2)。

目标节点处 min rise arrival 和 min fall arrival 计算方式与 max 情况相同，只是 delay 选取相应的 min delay 即可。

仍以下图简单设计为例，当不考虑时序图中各条边的 sigma 值，同时为了计算简单，我们假设所有边的 delay 值 max rise mean 值为 1.1， max fall mean 值为 1.2， min rise mean 值为 0.9， min fall mean 值为 0.8， 对于 setup check 边来说，rise constraint 值为 0.5， fall constraint 值为 0.6。为了简化，我们记 MaxRiseMean(X)为时序图顶点 X 出的 max rise mean 值，MaxRiseDelay(X->Y)指 X 到 Y 这条边的 max rise delay 的 mean 值，其余类同。则下图例子中各个顶点的 arrival time 计算如下：



MaxRiseMean(CLK) = MaxFallMean(CLK) = MinRiseMean(CLK) = MinFallMean(CLK) = 0，因为 CLK 为时序图起点。

CLK->CBUF1/A 边的 sense 为 pos_unate，则

MaxRiseMean(CBUF1/A) = MaxRiseMean(CLK) + MaxRiseDelay(CLK->CBUF1/A) = 0 + 1.1 = 1.1.

MaxFallMean(CBUF1/A) = MaxFallMean(CLK) + MaxFallDelay(CLK->CBUF1/A) = 0 + 1.2 = 1.2.

MinRiseMean(CBUF1/A) = MinRiseMean(CLK) + MinRiseDelay(CLK->CBUF1/A) = 0 + 0.9 = 0.9.

MinFallMean(CBUF1/A) = MinFallMean(CLK) + MinFallDelay(CLK->CBUF1/A) = 0 + 0.8 = 0.8.

同理我们可以得出

MaxRiseMean(CBUF1/Z) = 2.2,

MaxFallMean(CBUF1/Z) = 2.4,

MinRiseMean(CBUF1/Z) = 1.8,

MinFallMean(CBUF1/Z) = 1.6,

MaxRiseMean(DFF1/CK) = MaxRiseMean(DFF2/CK) = 3.3,

MaxFallMean(DFF1/CK) = MaxFallMean(DFF2/CK) = 3.6,

MinRiseMean(DFF1/CK) = MinRiseMean(DFF2/CK) = 2.7,

MinFallMean(DFF1/CK) = MinFallMean(DFF2/CK) = 2.4.

DFF1/CK->DFF1/Q 边的 sense 为 rising_edge，则

MaxRiseMean(DFF1/Q) = MaxRiseMean(DFF1/CK) + MaxRiseDelay(DFF1/CK -> DFF1/Q) = 3.3 + 1.1 = 4.4.

MaxFallMean(DFF1/Q) = MaxRiseMean(DFF1/CK) + MaxFallDelay(DFF1/CK -> DFF1/Q) = 3.3 + 1.2 = 4.5.

MinRiseMean(DFF1/Q) = MinRiseMean(DFF1/CK) + MinRiseDelay(DFF1/CK -> DFF1/Q) = 2.7 + 0.9 = 3.6.

MinFallMean(DFF1/Q) = MinFallMean(DFF1/CK) + MinFallDelay(DFF1/CK -> DFF1/Q) = 2.7 + 0.8 = 3.5.

DFF2/Q 处各个 Arrival time 值与 DFF1/Q 处相同。

INV1/A 处 arrival time 为 DFF2/Q arrival time 加一个 pos_unate 边的 delay 之后的值：

MaxRiseMean(INV1/A) = 4.4 + 1.1 = 5.5

MaxFallMean(INV1/A) = 4.5 + 1.2 = 5.7

MinRiseMean(INV1/A) = 3.6 + 0.9 = 4.5

MinFallMean(INV1/A) = 3.5 + 0.8 = 4.3

INV1/A->INV1/Z 边的 sense 为 neg_unate,则

MaxRiseMean(INV1/Z) = MaxFallMean(INV1/A) +
MaxRiseDelay(INV1/A->INV1/Y) = 5.7 + 1.1 = 6.8

MaxFallMean(INV1/Z) = MaxRiseMean(INV1/A) +
MaxFallDelay(INV1/A->INV1/Z) = 5.5 + 1.2 = 6.7

MinRiseMean(INV1/Z) = MinFallMean(INV1/A) +
MinRiseDelay(INV1/A->INV1/Z) = 4.3 + 0.9 = 5.2

MinFallMean(INV1/Z) = MinRiseMean(INV1/A) +
MinFallDelay(INV1/A->INV1/Z) = 4.5 + 0.8 = 5.3

XOR1/B 处 arrival time 为 INV1/Z 处 arrival time 加一个 pos_unate 边 delay，即

MaxRiseMean(XOR1/B) = 6.8 + 1.1 = 7.9

MaxFallMean(XOR1/B) = 6.7 + 1.2 = 7.9

MinRiseMean(XOR1/B) = 5.2 + 0.9 = 6.1

MinFallMean(XOR1/B) = 5.3 + 0.8 = 6.1

XOR1/A 处 arrival time 应为 DFF1/Q 处 arrival time 经过 3 个 pos_unate 边之后的值,具体计算过程不再赘述：

MaxRiseMean(XOR1/A) = 4.4 + 1.1 * 3 = 7.7

MaxFallMean(XOR1/A) = 4.5 + 1.2 * 3 = 8.1

MinRiseMean(XOR1/A) = 3.6 + 0.9 * 3 = 6.3

MinFallMean(XOR1/A) = 3.5 + 0.8 * 3 = 5.9

对于 XOR1/Z 处的 arrival time，由于在时序图中有 4 条边可以到达此顶点，因此要分别计算经过 4 条边的 arrival time，然后进行合并操作，保留最差结果，具体计算过程如下：

对于 XOR1/A->XOR1/Z，sense 为 pos_unate 的边来说，

MaxRiseMean(XOR1/Z) = MaxRiseMean(XOR1/A) +
MaxRiseDelay(XOR1/A->XOR1/Z) = 7.7 + 1.1 = 8.8

MaxFallMean(XOR1/Z) = MaxFallMean(XOR1/A) +
MaxFallDelay(XOR1/A->XOR1/Z) = 8.1 + 1.2 = 9.3

MinRiseMean(XOR1/Z) = MinRiseMean(XOR1/A) +
MinRiseDelay(XOR1/A->XOR1/Z) = 6.3 + 0.9 = 7.2

MinFallMean(XOR1/Z) = MinFallMean(XOR1/A) +
MinFallDelay(XOR1/A->XOR1/Z) = 5.9 + 0.8 = 6.7

对于 XOR1/A->XOR1/Z，sense 为 neg_unate 的边来说，

MaxRiseMean(XOR1/Z) = MaxFallMean(XOR1/A) +
MaxRiseDelay(XOR1/A->XOR1/Z) = 8.1 + 1.1 = 9.2

MaxFallMean(XOR1/Z) = MaxRiseMean(XOR1/A) +
MaxFallDelay(XOR1/A->XOR1/Z) = 7.7 + 1.2 = 8.9

MinRiseMean(XOR1/Z) = MinFallMean(XOR1/A) +
MinRiseDelay(XOR1/A->XOR1/Z) = 5.9 + 0.9 = 6.8

MinFallMean(XOR1/Z) = MinFallMean(XOR1/A) +
MinFallDelay(XOR1/A->XOR1/Z) = 6.3 + 0.8 = 7.5

对于 XOR1/B->XOR1/Z，sense 为 pos_unate 的边来说，

MaxRiseMean(XOR1/Z) = MaxRiseMean(XOR1/B) +
MaxRiseDelay(XOR1/B->XOR1/Z) = 7.9 + 1.1 = 9.0

MaxFallMean(XOR1/Z) = MaxFallMean(XOR1/B) +
MaxFallDelay(XOR1/B->XOR1/Z) = 7.9 + 1.2 = 9.1

MinRiseMean(XOR1/Z) = MinRiseMean(XOR1/B) +
MinRiseDelay(XOR1/B->XOR1/Z) = 6.1 + 0.9 = 7.0

MinFallMean(XOR1/Z) = MinFallMean(XOR1/B) +
MinFallDelay(XOR1/B->XOR1/Z) = 6.1 + 0.8 = 6.9

对于 XOR1/B->XOR1/Z，sense 为 neg_unate 的边来说，

MaxRiseMean(XOR1/Z) = MaxFallMean(XOR1/B) +
MaxRiseDelay(XOR1/B->XOR1/Z) = 7.9 + 1.1 = 9.0

MaxFallMean(XOR1/Z) = MaxRiseMean(XOR1/B) +
MaxFallDelay(XOR1/B->XOR1/Z) = 7.9 + 1.2 = 9.1

MinRiseMean(XOR1/Z) = MinFallMean(XOR1/B) +
MinRiseDelay(XOR1/B->XOR1/Z) = 6.1 + 0.9 = 7.0

MinFallMean(XOR1/Z) = MinFallMean(XOR1/B) +
MinFallDelay(XOR1/B->XOR1/Z) = 6.1 + 0.8 = 6.9

XOR1/Z 上的最终 arrival time 为所有扇入的边计算得来的 arrival time 经过对 Max 数据取 max 操作，对 Min 数据取 min 操作得来：

MaxRiseMean(XOR1/Z) = max(8.8, 9.2, 9.0, 9.0) = 9.2

MaxFallMean(XOR1/Z) = max(9.3, 8.9, 9.1, 9.1) = 9.3

MinRiseMean(XOR1/Z) = min(7.2, 6.8, 7.0, 7.0) = 6.8

MinFallMean(XOR1/Z) = min(6.7, 7.5, 6.9, 6.9) = 6.7

按照上述方法继续计算，我们最终可以得到 DFF3/D 处以及 DFF3/CK 处 arrival time：

MaxRiseMean(DFF3/D) = 10.3

MaxFallMean(DFF3/D) = 10.5

MinRiseMean(DFF3/D) = 7.5

MinFallMean(DFF3/D) = 7.5

MaxRiseMean(DFF3/CK) = 5.5

MaxFallMean(DFF3/CK) = 6.0

MinRiseMean(DFF3/CK) = 4.5

MinFallMean(DFF3/CK) = 4.0

之后我们即可计算 DFF3/D 这个 endpoint 上的 slack，此 slack 即为所有在 DFF3/D 处结束的 path 中最差的 slack，即 global slack：

SlackRise(DFF3/D) = ClockPeriod + MinRiseMean(DFF3/CK) –
MaxRiseMean(DFF3/D) – RiseConstraint(DFF3/CK->DFF3/D) = 10 + 4.5 –
10.3 – 0.5 = 3.7

SlackFall(DFF3/D) = ClockPeriod + MinRiseMean(DFF3/CK) –
MaxFallMean(DFF3/D) – FallConstraint(DFF3/CK->DFF3/D) = 10 + 4.5 –
10.5 – 0.6 = 3.4

对于 sense 为 setup_falling 的边，处理方式与 DFF3/D 处类似，但是要使用 MinFallMean(CK)的 arrival time。

九、 评分标准：

a) Endpoint slack 准确性（分值：35）：

i. 举办方提供评测脚本和标准结果，对参赛队伍提交的结果和标准结果进行对比并得出一个最终的准确度分数，按照准确度分数对参赛队伍的结果进行排序和评分。

ii. 对于单个 endpoint， 如式(1)计算提交结果的误差，再如式(2)计算其得分。当单个 endpoint 的误差超过 20%时记为无效结果，该 endpoint 得分计零。

iii. 对于所有 endpoint，如式(3)计算得分，满分为 35。

$$err_{single} = \mathrm{abs}\left(\frac{提交结果 - 标准结果}{标准结果}\right) \tag{1}$$

$$score_{single} = \begin{cases} 1 - err_{single} & , err_{single} \leq 20\% \\ 0 & , err_{single} > 20\% \end{cases} \tag{2}$$

$$score_{total} = 35 \times \frac{\sum_{所有结果} score_{single}}{所有结果数量} \tag{3}$$

b) Arrival time 计算及 endpoint slack 计算运行时间及峰值内存占用（分值：35）：

i. 使用 GNU time 对参赛队伍提交的可执行文件的计算 Arrival time 和 endpoint slack 的过程进行监督，取其输出结果中“Elapsed (wall clock) time”作为运行时间，“Maximum resident set size”作为峰值内存占用大小进行评比。

ii. 提交作品的每只参赛队伍的运行时间和峰值内存占用大小分别排序得到运行时间名次和内存占用名次，然后按照下式得出本项目分数：

$$PerfScore = 20 \times \left(1 - \frac{内存占用名次 - 1}{提交作品队伍数量}\right) + 15 \times \left(1 - \frac{运行时间名次 - 1}{提交作品队伍数量}\right)$$

iii. 当 endpoint slack 准确度评测中累计无效结果（单个 endpoint 的误差超过 20%）数量超过 endpoint 总数 30%时，本项不得分。

c) 全图顶点 global slack 准确性（分值：10）：

i. 举办方提供评测脚本和标准结果，对参赛队伍提交的结果和标准结果进行对比并得出一个最终的准确度分数，按照准确度分数对参赛队伍的结果进行排序和评分。具体的准确度评判标准与 endpoint slack 相同。

d) 全图顶点 global slack 计算运行时间及峰值内存占用（分值：10）：

i. 使用 GNU time 对参赛队伍提交的可执行文件的计算 global slack 过程进行监督，取其输出结果中“Elapsed (wall clock) time”作为运行时间，“Maximum resident set size”作为峰值内存占用大小进行评比。

ii. 提交作品的每只参赛队伍的运行时间和峰值内存占用大小分别排序得到运行时间名次和内存占用名次，然后按照下式得出本项目分数：

$$PerfScore = 5 \times \left(1 - \frac{内存占用名次 - 1}{提交作品队伍数量}\right) + 5 \times \left(1 - \frac{运行时间名次 - 1}{提交作品队伍数量}\right)$$

iii. 同样地，当累计无效结果数量超过所有给出的全图顶点数量30%时，本项不得分。

e) Arrival time skewness 影响（分值：10）：

i. 参赛队伍在最终提交的设计报告中详细描述对 skewness 的建模和计算方法。若考虑此项则得分，否则不得分。注意此项目会对输出的报告精度有正面影响。

ii. 若参赛队伍选择此项，则在 endpoint slack 准确性评判标准中使用考虑了 skewness 的标准结果作为对比，且此项总分调整为 45 分。同时为了鼓励参赛队伍探索，避免出现由于考虑不周等问题导致 slack 结果太差，评测中会同时以原分数 35 分为总分计算与不考虑 skewness 的结果对比分数。同时 arrival time 计算和 endpoint slack 计算的运行时间和内存占用情况评测也会分别对于考虑 skewness 和不考虑 skewness 的情况进行评比，取两种情况下精度和性能两项评分总分最大为最终得分。因此要求在最终提交的应用程序中提供选项关闭 skewness 的影响，以免影响不考虑 skewness 的结果评判。

f) 评测测试用例分为公开用例和隐藏用例，最终评分按照所有测试用例使用相同权重，将每个测试用例的分数累加，得出参赛队伍最终得分。

十、 参考文献：

a) D. Blaauw, K. Chopra, A. Srivastava and L. Scheffer, "Statistical Timing Analysis: From Basic Principles to State of the Art," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 4, pp. 589-607, April 2008, doi: 10.1109/TCAD.2007.907047.

b) Charles E. Clark, (1961) The Greatest of a Finite Set of Random Variables. Operations Research 9(2):145-162. https://doi.org/10.1287/opre.9.2.145

c) C. Visweswariah et al., "First-Order Incremental Block-Based Statistical Timing Analysis," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 10, pp. 2170-2180, Oct. 2006, doi: 10.1109/TCAD.2005.862751.