# 第二届（2020年）集成电路EDA设计精英挑战赛
# 赛题指南

## 一、赛题七：图分割算法（数字集成电路设计方向）

## 二、命题单位：思尔芯（上海）信息科技有限公司

## 三、赛题背景

随着计算技术的发展，大数据时代的到来，超大规模图的分割问题越来越引起人们的关注，并被广泛应用于大数据处理的各个领域，如分布式计算问题划分、推荐策略、布局等等。典型应用有超大规模数字集成电路仿真验证中的多 FPGA 系统分割，通过不同的分组权重将图分割成若干分组，进而实现快速高效的系统验证。

## 四、赛题描述

对给定的数字集成电路门级电路的网表文件，按照分组权重约束条件 - PIO / LUT / FLIPFLOP / CONNECTION / FIX-ASSIGN，对网表进行不同模式的 Partition 算法分割，分割成若干个分组。在保证整个图拓扑结构不变和满足约束条件的情况下，确保每个分组之间互联线数目最少。

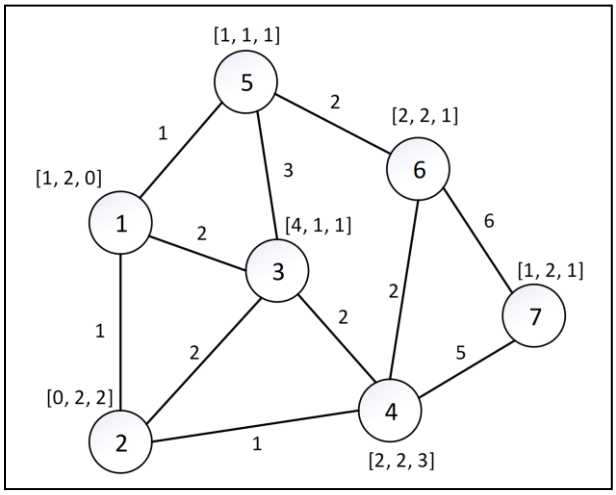
涉及知识点：

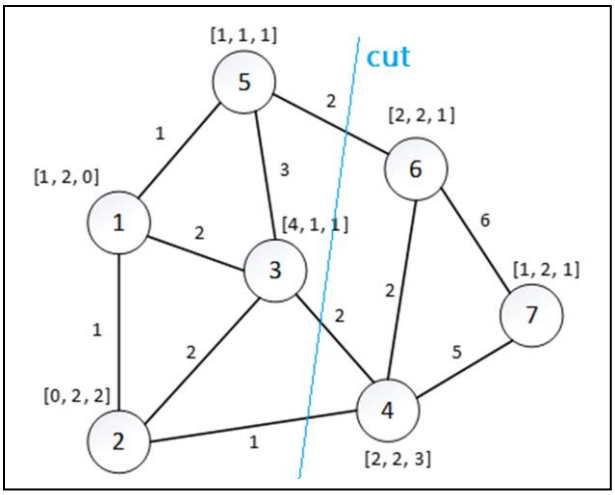
- 门级电路的图论建模
- 最小割相关 min-cut 算法

数学原理：

Partition 算法的目的是对一个带顶点权重与连接权重的图, 根据设定的分组权重约束条件，进行不同的切割尝试，最后获取到一个最小的连接权重值的切割过程。

例如：对左下图进行切割，约束条件要求分为 2 个分组，每个分组资源权重不限，一个分组包含节点 5，另一个分组包含节点 4， 则最优的切割结果为右下图， 互联线权重为 5 < 2+2+1 >。




技术评分点：

- 不同权重约束条件和不同工作模式下分割结果的正确性（主办方提供检查程序）
  - 满足资源约束
  - 网表等价（节点互连结构等价）
- 分割结果的最优化
  - Cut size（互联线权重）最小
- 满足分割结果正确的前提下，在同一平台（主办方将给出操作系统版本和编译器版本）上的运行时间，即所需的 CPU 算力
- 内存使用量
  - 使用 top 命令或工具 memtime
- 附加 Benchmark 测试（隐藏测试）的完成度

注：参赛队伍提供技术报告（含时间空间复杂度分析）、可执行程序与源代码

## 五、评分标准：

技术评分，共 100 分。

1. 基础任务 - 60 分 。（Benchmark 测试的完成度占比 70%，结果正确性、最优化、运行速度、内存使用量指标占比 30%）

| 测试项 | 测试描述 |
|---|---|
| testdata-0 | 简单的 2-FPGA 切割，较少的节点(20）和连线（40） |
| testdata-1 | 小型的 2-FPGA 切割，少量的节点(1000）和连线（1000） |
| testdata-1-e | 针对 testdata-1 的错误权重约束的检测 |
| testdata-2 | 大型的 2-FPGA 切割，大量的节点(12000）和连线（14000） |
| testdata-3 | 大型的 2-FPGA 切割，大量的节点(12000）和连线（33000） |
| testdata-4 | 大型的 4-FPGA 切割，大量的节点(12000）和连线（14000） |
| testdata-5 | 大型的 8-FPGA 切割，大量的节点(12000）和连线（14000） |

2. 高性能任务 – 40 分。（Benchmark 测试的完成度占比 70%，结果正确性、最优化、运行速度、内存使用量指标占比 30%）

| 测试项 | 测试描述 |
|---|---|
| testdata-4-cnn | 大型的 4-FPGA 切割，基于可变的 FPGA 之间的互联关系，大量的节点(12000）和连线（14000） |
| testdata-4-mean | 大型的 4-FPGA 平均值切割，基于可变的 FPGA 之间的互联关系 |
| testdata-5-ffd | 大型的 8-FPGA 切割，基于节点之间的 FlipFlop 驱动规则，大量的节点(12000）和连线（14000） |
| testdata-5-clk | 大型的 8-FPGA 切割，基于节点之间的 CLK 关系，大量的节点(12000）和连线（14000） |
| testdata-6 | 200,000 节点范例的切割 |

说明：相关 FAQ 解答指南请联系 eda_ec@s2cinc.com。

附件：赛题接口文件说明

## 5.1 输入文件

输入文件包含当前图(Graph)里面所有节点(Node)，连线(Net)，权重(Weight)和 分组(Group)的资源信息，要求参赛者的分割工具根据这些信息对图中的节点进行切割，根据实现算法的最优结果，生成输出若干个小图.

- **Node definition file <节点定义文件 design.are>**

每个节点名称以字母 g 和一个不重复的数字组成，
每行表示一个节点的资源信息，包含 10 种资源，每种资源权重以 10 进制数值表示。
*<node> PIO INT FF LUT BUFG TBUF DCM BRAM DSP PPC [timing-property-list]*

各个资源的定义如下表。

| *Resource* | *Description* | *Comment* |
|---|---|---|
| 资源 1 | PIO | PIO |
| 资源 2 | INT | INT |
| 资源 3 | FF | Flipflop, Latch |
| 资源 4 | LUT | LUT2, LUT4, LUT6, LUT8 |
| 资源 5 | BUFG | BUFG |
| 资源 6 | TBUF | BUFR, BUFT |
| 资源 7 | DCM | PLL, DLL |
| 资源 8 | BRAM | RAMB36, RAMB128 |
| 资源 9 | DSP | DSP, ARM |
| 资源 10 | PPC | PPC |

例如，
常规节点文件，
*g0 0 1 200 0 0 0 0 0 0 0*
*g1 0 1 200 0 0 0 0 0 0 0*
*g2 0 1 200 0 0 0 0 0 0 0*
*g3 0 1 200 0 0 0 0 0 0 0*
*g4 0 1 200 0 0 0 0 0 0 0*
*g5 0 1 200 0 0 0 0 0 0 0*

带 timing 属性的节点文件，
*g0 0 1 200 0 0 0 0 0 0 0 {ff}*
*g1 0 1 200 0 0 0 0 0 0 0 {c0}*
*g2 0 1 200 0 0 0 0 0 0 0 {c2}*
*g3 0 1 200 0 0 0 0 0 0 0*
*g4 0 1 200 0 0 0 0 0 0 0 {ff c2}*
*g5 0 1 200 0 0 0 0 0 0 0 {ff c2c3}*

其中， *[timing-property-list]* 为可选性，里面属性定义如下，
*ff 节点为 ffd 类型*
*c0 节点含有名为 c0 的 clk 属性*
*c2c3 节点含有名为 c2 和 c3 2 个 clk 属性*

- **Net definition file <连线定义文件 design.net>**

每个连线信息由 2 个或更多节点组成，一个为驱动节点(driver)，其他为负载节点(load),

每行表示一个连线的部分信息，格式如下，

*<node> <s/l> [weight]*

*s 含有驱动(driver)节点的连线部分*
*l 含有负载(load)节点的连线部分，一个连线可能含有一个或多个负载部分*
*weight 连线的权重值，可选*

例如,
*g1 s 1*
*g0 l*
*g0 s 1*
*g2 l*
*g1 l*

- **The FPGA group resource list file <FPGA 资源文件定义 design.info>**

Fpga 分组的资源定义与节点资源一致，每一行表示一个分组的资源最大权重值信息和分组的序号。
格式如下，
*<FPGA> PIO INT FF LUT BUFG TBUF DCM BRAM DSP PPC [int-list]*

*其中，[int-list] 为可选性，*
采用 list 列表表示，每个分组的序号(从 1 开始)，对应列表中相应的位置元素(从第 1 个元素开始)。列表里面的值表示当前分组与其他分组的互连线约束权重值。
每个分组的内部互联权重值为 0，不用考虑。
第 2 列 INT 的值是最后 list 的和，如下 800 = 0+200+200+400

分割算法必须根据分组资源信息对图中节点进行分割，输出的结果不能够超出每个分组的资源权重值。

例如 2 个分组，
*FPGA 576 890 427200 427200 24 2048 24 55562240 0 0*
*FPGA 576 890 427200 427200 24 2048 24 55562240 0 0*

4 个分组，
*FPGA 576 800 427200 427200 24 2048 24 55562240 0 0 { 0 200 200 400 }*
*FPGA 576 800 427200 427200 24 2048 24 55562240 0 0 { 200 0 100 500 }*
*FPGA 576 900 427200 427200 24 2048 24 55562240 0 0 { 200 100 0 600 }*
*FPGA 576 1500 427200 427200 24 2048 24 55562240 0 0 { 400 500 600 0 }*

- **The fix or preassigned node list file <预先分配的节点定义文件 design.fix>**

预先定义的节点分配文件，一行一个或多个节点在指定分组里面的分配信息。
分割算法必须根据节点预分配信息对图中节点进行分割，输出的结果满足所有预分配的节点分组信息。

**注： 这个是优先级最高的规则，分割分组结果必须满足此文件预定义的分组**

格式如下,
*<FPGA TYPE mm>: <node> <node> ...*

mm 分组编号是从 1 开始的正整数，不一定是连续的数字定义.

例如,
*FPGA TYPE 1: g0*
*FPGA TYPE 2: g1*

*例如，编号不连续的预分配定义，*

FPGA TYPE 1: g0
FPGA TYPE 2: g1
FPGA TYPE 3: g3013
FPGA TYPE 4: g6026
FPGA TYPE 7: g30
FPGA TYPE 8: g40

- **The cut mode <分割模式>**

目前定义的分割算法的工作模式，

***--fix-mincut*** 默认分割模式，所有的资源权重都是固定不变的，要求分割算法能够在不超出分组资源权重的前提下算出最优的结果，且所有分组的对外互联数目总和最小。

***--int-mincut*** 同 *--fix-mincut*, 此外，额外要求分割算法满足每个分组之间的互联权重约束，得出最优解，且所有分组的对外互联数目总和最小。

***--ffd-mincut*** 同 *--fix-mincut*, 此外，额外要求分割算法满足每个分组 flipflop 驱动规则的情况下，得出最优解，且所有分组的对外互联数目总和最小。

*Flipflop 驱动规则如下:*
*1. 每个可分割的节点必须为带有 ffd 属性的节点，或者其他节点*

*仅当它所有的驱动连线都为带 ffd 属性的节点。*

2. *每个分组的节点必须由其他分组的 ffd 节点驱动.*

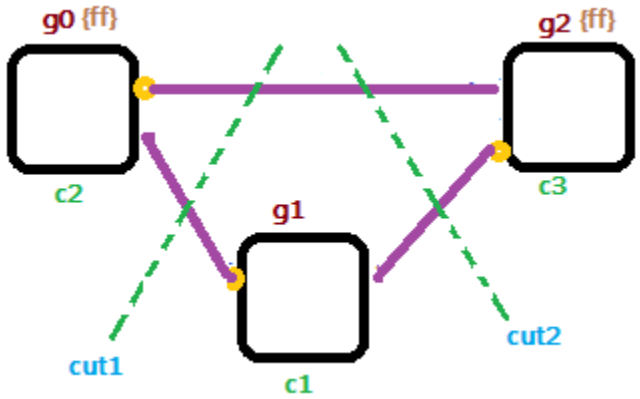



Figure 1 正确的 FFD 切割示意图

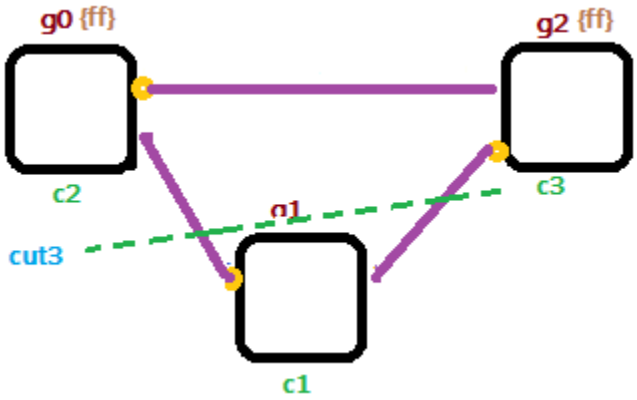



Figure 2 错误的 FFD 切割示意图

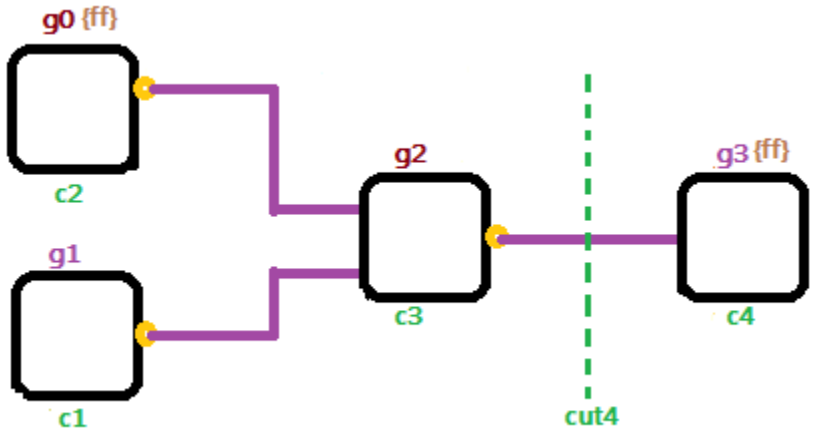



Figure 3 错误的 FFD 切割示意图

***--clk-mincut***
***--num <n>***
***--clks***
***<candidate_clk_list>***

同 *--fix-mincut*，此外，额外要求分割算法满足每个分组间出现的节点关联的 clk 最大数目或（和）合适名称的条件下，得出最优解，且所有分组的对外互联数目总和最小。
*--num/--clks* 为可选项。没有指定时要求每个分组间出现的节点关联的 clk 最大数目最小。
*clk 规则如下:*
*每个不带 clk 属性的节点可以继承所有驱动它的最近的带 clk 属性节点的 clk 属性值。*

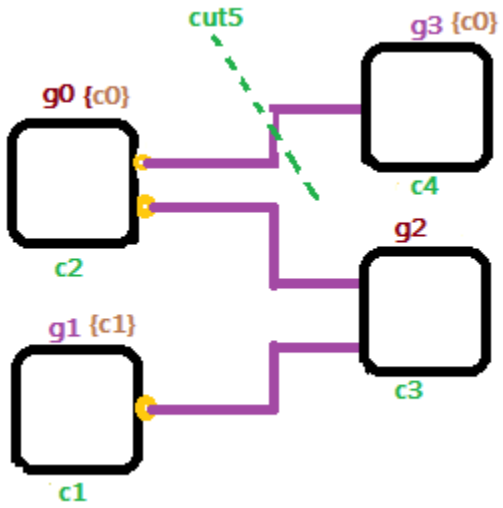



Figure 4 正确的 CLK 分割示意图

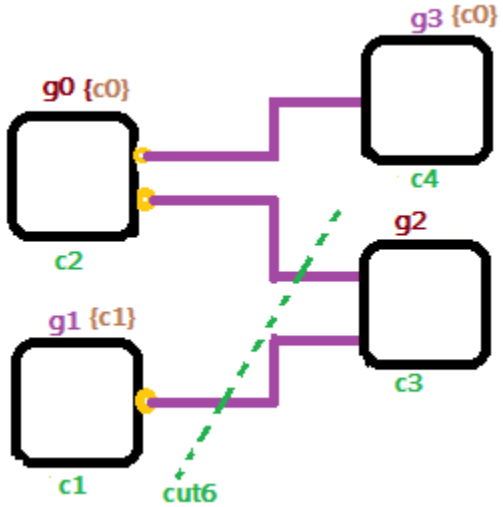



Figure 5 错误的 CLK 分割示意图

| | |
|---|---|
| ***--mean-mincut*** | 同 *--fix-mincut*, 此外，额外要求分割算法满足每个分组间互联线与资源权重值的比例在给定的均值误差的范围内，得出最优解，且所有分组的对外互联数目总和最小。 |
| ***--percentage*** | *--percentage* 为可选项，没有指定时默认值为 20. |

## 5.2 输出文件

输出文件包含对图中所有节点进行分割后的最优分割结果和报告信息。

- **The output partition result file <分割结果文件 design.output>**

  分割结果文件，由分割算法根据当前的分割策略和模式运算得出的最优结果。
  每一行包含一个分组信息里面包含的节点列表，以 'FPGA+序号 分组+mm' 字串开始，每个节点之间以空格隔开，每行建议最大包含 20 个节点，可以多行表示。

  格式如下,
  *<FPGAnn TYPE nn>: <node-list>*

  例如,
  *FPGA1 TYPE 1 :g0 g1 ..... g19*
  *g20 g21 gp0*
  *FPGA2 TYPE 2 :g100 gp1 gp2*

- **The output partition report file <分割结果报告文件 design.rpt>**

  分割结果报告文件，包含分组的实际资源权重使用值，以及每 2 个分组之间的互联信息。
  只有 2 分组的情况下，{int-list} 不需要。
  格式如下,
  *<FPGAnn TYPE nn>: <resource-list> [int-list]*
  {int-list} 采用 list 列表表示，每个分组的序号(从 1 开始)，对应列表中相应的位置元素(从第 1 个元素开始)。列表里面的值表示当前分组与其他分组的互连线约束权重值。

  每个分组的内部互联权重值为 0，不用考虑。

  例如,
  2 个分组的报告文件,
  *FPGA1 TYPE 1: 1 401 109 1814 0 0 0 0 0 0*
  *FPGA2 TYPE 3: 2 401 0 4000 0 0 0 0 0 0*

  4 个分组的报告文件件，
  *FPGA1 TYPE 1: 1 400 109 1814 0 0 0 0 0 0 { 0 200 50 150 }*
  *FPGA2 TYPE 2: 2 350 0 4000 0 0 0 0 0 0 { 200 0 50 100 }*
  *FPGA3 TYPE 3: 1 200 109 1814 0 0 0 0 0 0 { 50 50 0 100 }*

*FPGA4 TYPE 4: 2 350 0 4000 0 0 0 0 0 0 { 150 100 100 0 }*

- 关于 **FPGA 编号和 TYPE 编号**

  FPGA 后面的编号 n 仅仅表示一个处理序号，不是分组号。一般情况下代码处理时 fpga 后编号和 Type 后分组号一致。比如 design.output， design.rpt 文件格式。如果不一致，以 TYPE 后编号为准（分组号）
  如果 fpga 后无编号，以 TYPE 后编号为准，作为分组号。比如 design.fix 格式。

  FPGA TYPE 1: g0
  FPGA TYPE 2: g1
  FPGA TYPE 3: g3013
  FPGA TYPE 4: g6026
  FPGA TYPE 7: g30
  FPGA TYPE 8: g40