XAPP918 (v1.0) 2007 年 6 月 7 日

分区实现增量设计重用

作者：Chris Zeh

提要

本应用指南讨论在增量设计流程中使用分区的方法。建议将逻辑密集型模块实例、时序关键性路径或时序关键性模块实例指定为分区。如果模块实例未经修改，则将其指定为分区会指示综合和实现工具将以前的实现结果用于该模块实例。分区的优点是可以缩短实现工具的运行时间，免于重新验证未变更的模块实例，并且帮助实现时序收敛。分区可以通过项目浏览器或 Tcl 界面操控，允许对重新使用实现结果的方式进行细粒度控制。

什么是分区?

分区用于保留以前已实现设计的未变更部分。分区可用来优化设计的实现过程。如果分区的 HDL、时序和物理约束以及实现选项未变更，则实现工具会通过 “复制粘贴” 操作保证将该分区的实现数据保留下来。

通过将实现结果保留下来，分区可以实现设计的已修改部分而不影响设计的其余部分或已保留的分区。将设计的一些部分保留下来，就不必实现整个设计，因此有助于缩短实现的运行时间。一般而言，运行时间缩短的程度与保留的逻辑量成比例：保留分区比实现整个设计更快。在逻辑模块实例上创建分区后，即可指示 Xilinx ISE™ 设计工具尽可能重新使用以前的实现结果。

分区使用的保留方法具有以下优点：

- 缩短综合与实现的运行时间 （ 因而增加每日运行次数 ）。
- 可以 100% 保留功能模块。
  - 将功能模块指定为分区，即可在实现后对其验证一次，然后将其 100% 保留下来供将来反复使用。未变更的分区不需要重新验证。
- 允许锁定设计中的功能模块。
  - 例如，如果除少数功能模块外，设计的大多数指标都稳定，则可以用分区锁定未变更的功能模块，以减少最后一刻修改设计的影响。
  - 又如，任何无变化的功能模块 （ 如以前实现的 IP 核 ） 都是指定为分区的良好对象。
- 设计中可能有按自然设计层级分解的主要功能模块 （ 如：大型设计、集体设计、EDK 设计和 DSP 设计)。一般而言，已记录功能模块边界的结果为最佳。这样可尽量发挥分区的隔离作用的优越性。
- 提高设计的时序关键部分的稳定性。
  - 设计可能存在时序问题，这些时序问题在历次实现中出现的位置不同，使得难以实现时序收敛。通过分区，可以使用分治法分析、隔离和解决时序问题。请注意，最新分区的时序等效于以前的实现，所以在保留的分区中不会再出现时序问题。
- 报告每个层级模块的资源利用统计数据。
- 将实现的变更隔离在几个 （ 最好是一个 ） 分区中，即使当变更 “很大” 时也是如此。如果设计的资源利用率很高，就可能需要将无变化分区的保留级别从布线改为布局或综合。

- 隔离不需要与设计的其余部分一起优化的核的实现。
- 当设计的少量变更局限在一个分区内时，可加快实现速度。
- 可以对设计中每个分区的保留方式进行精细级别控制。
  - 注：SmartGuide™ 可以开启或关闭，并且适用于整个设计。保留级别的控制以分区为单位。不得同时使用 SmartGuide 和分区。
- 增强不能使用 SmartGuide 的设计。

使用分区可以提高满足设计的时序目标或物理约束（AREA_GROUP 等）的能力。一旦满足了时序目标，如果将分区准确地保留下来，则分区的时序不会改变。在尝试满足时序要求时，分区可以降低时序结果的可变性。当为满足时序而修改设计的一部分时，如果设计的另一（不相关）部分不满足时序，就会发生这种情况。

为了最有效地使用分区，请遵循以下规则：

- 将驱动输出引脚的寄存器置于设计的顶层或输出缓冲器的同一分区内。
- 将由输入引脚驱动的寄存器置于设计的顶层或输入缓冲器的同一分区内。
- 将 ODDR 和 IDDR 置于设计的顶层或者 ODDR 或 IDDR 所关联引脚的同一分区内。
- 将寄存器置于分区模块的输入和输出上。
  - 当在分区边界处封装时，这样做具有最大的灵活性和可优化性。
- 为了尽量缩短运行时间，设计中的分区应包含大致等量的同步元件和逻辑。
  - 对于具有少量非时序关键性逻辑的分区，运行时间的缩短可以忽略不计。
- 将分区置于难以满足时序的模块上，这样，一旦满足了时序，即可将其保留下来。
- 要在 IP 核或 EDK 设计上设置分区，ISE 9.1 版本要求用 HDL 封装对其进行例化。然后把分区设置在 IP 核或 EDK 设计的 HDL 封装上。
- 如果实现难以为设计完全布线，请将顶层分区的 “Preserve”（保留）属性从 “Routing”（布线）改为 “Placement”（布局）。所有子分区均 “Inherit”（继承）顶层分区的保留级别。
- Synplify Pro 8.8.1 用编译点功能支持分区。
- 当使用分区执行自下而上的综合方法时，必须用顶层 HDL 封装例化较低级别的网表。然后把分区设置在网表的封装上。

可以将分区放置在以下类型的任意层级上：

- HDL（Verilog 或 VHDL）源文件
- 原理图源
- EDIF 源
  - 在 HDL 封装上创建
  - 用 Synplify Pro 中的编译点创建

可以将分区嵌套在整个层级结构中。具有遍及整个层级结构的多个实例的模块可以有多个分区，或者每模块实例一个分区。当较低层实例具有已定义的分区时，设计的顶层模块自动指定为一个分区。顶层模块中的逻辑没有数量限制。设计的较低层实例用 Verilog 模块实例名称定义，而实体架构或元件实例的名称则用 VHDL 定义。

分区可以随时创建或删除。不过，创建或删除一个分区和父分区需要用综合与实现工具重新实现设计的该部分。因此，最有效的方法是在第一个实现周期之前检查设计的实例，以确定应将哪些实例指定为分区。建议在初始阶段创建分区，不要以后在设计周期中使用增量方法增加分区。

由于 XST 的限制，当使用 XST 进行综合时，请勿使用具有以下综合结构的分区：

- Generate 语句
- Included 语句

当使用分区时，应避免以下结构和优化：

- 跨越分区边界的 TBUF
- 跨越分区层级边界的异步时序关键性网络
- 全局优化设置，如 “map –global_opt”
- 在具有较高 SLICE 占用率的设计上增加分区会导致更高的 SLICE 占用率。
  - 分区在某些设计中会降低 SLICE 占用率，而在另一些设计中则会增加占用 SLICE 的数量。
- 分区完全支持 ChipScope™ CORE Generator™，但不支持 ChipScope Core Inserter。
- 当使用集成的第三方综合工具时，项目浏览器不支持分区。例如，如果使用 Synplify Pro 在 Synplify Pro 环境而非项目浏览器中综合和实现设计，则不支持分区。具有分区的 HDL 设计可以用项目浏览器和 XST 综合工具实现，也可以用 EDIF 流程中的项目浏览器和独立的 Synplify Pro 综合工具实现。
- 处于与引脚不同模块层级的 IDDR、ODDR、IFD 或 OFD 寄存器
- 较低层模块中的双向端口
- 自下而上的综合流程
- 元件的悬垂或未连接端口
- 同一源文件中的核或层级模块
- 需要跨越层级模块或跨越分区边界传播的常量。

以下架构支持分区：

- Spartan™–3
- Spartan–3E
- Spartan–3L
- Spartan–3A
- Spartan–3A DSP
- Virtex™–II
- Virtex–II Pro
- Virtex–II Pro X
- Virtex–4
- Virtex–5

分区与 SmartGuide 之间的抉择

SmartGuide 是一种保留方法，其中将以前的实现与当前实现进行比较，然后尽可能保留设计的未变更部分。如果需要满足时序目标并生成成功的实现，就会重新实现保留的设计部分。

ISE 9.1i 支持对特定设计使用分区或 SmartGuide，但不能在同一设计中支持这二者。分区与 SmartGuide 之间的抉择在很大程度上取决于设计的具体情况，因此没有可始终遵循的特定规则。SmartGuide 适用于整个设计，在 FPGA 中物理元件的最低层上操作。以下类型的设计方案适合使用分区：

- 不能使用 SmartGuide（例如，您不希望使用 map –timing）
- 设计变更导致大量修改综合结果
- 缩短综合及所有 Xilinx 实现工具的实现运行时间（如 NGDBuild、MAP 和 PAR）。SmartGuide 会缩短 MAP 和 PAR 的实现运行时间。
- 减少验证：由于实现完全相同，因而分区被保留，不需要重新验证。

- 对保留的更精细级别控制：SmartGuide 可以开启或关闭，并且适用于整个设计。分区支持对综合、布局或布线进行以分区为单位的保留控制。

以下设计变更适合使用 SmartGuide：

- 一两个模块中的少量逻辑变更（少于 10%），不影响设计的其余部分。
- 移动焊盘位置。
- 更改元件属性。
- 更改时序约束。
  - 在 ISE 9.x 中，更改时序约束会强制所有分区作废。

如何使用分区

在综合之前，需要在要求重新使用设计的层级上设置分区。综合工具不会跨分区接口进行优化。如果异步时序关键路径跨越分区边界，则不会跨越分区边界进行逻辑优化。要减轻此问题，请为分区边界处的异步信号增加一个寄存器。

分区不需要布局规划，例如重新使用实现的范围。分区支持设计顶层的任何逻辑量。项目浏览器在 “Sources”（源）标签页上显示设计层级结构。要在某层级上设置分区，请右键单击实例，并选择 “New Partition”（新建分区），如图 1 所示。

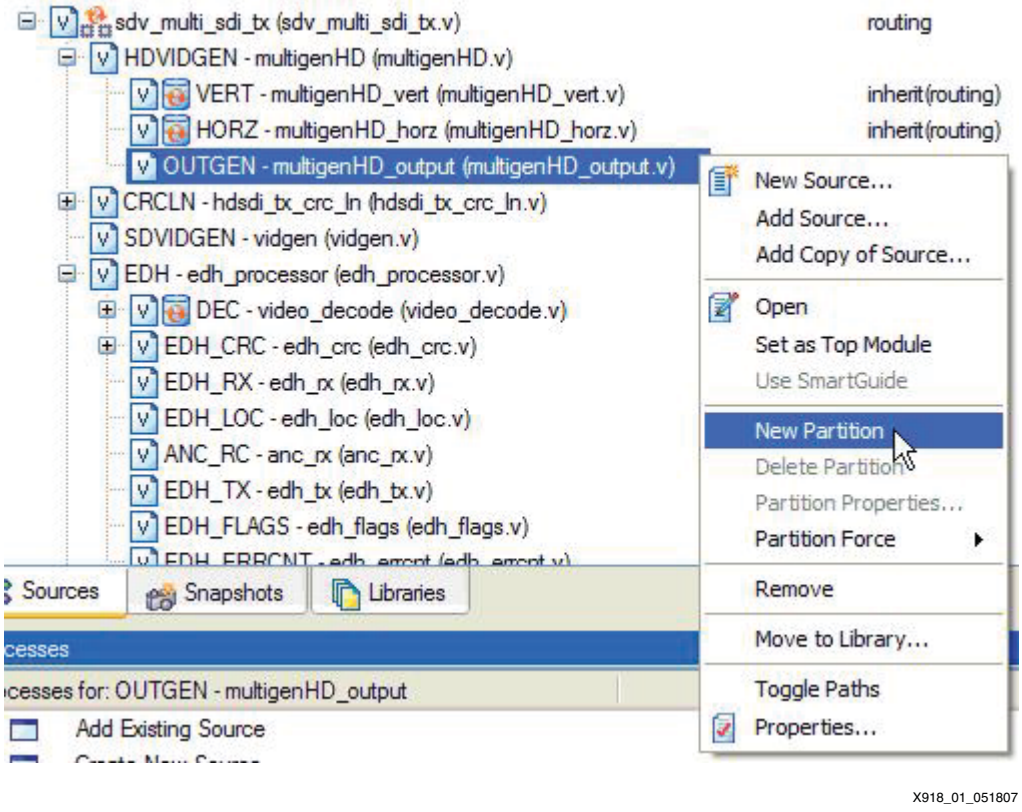


图 1: **在层级上设置分区**

有几个表示分区状态的图标。这些图标提供有关分区状态和各分区所需操作的信息。有两组分区图标：

- 一组顶层分区图标
- 一组实例级分区图标

表示顶层分区状态的图标如下：


设计中没有来自以前实现周期的分区。


下一实现周期将保留分区数据。


下一实现周期将实现分区。


下一实现周期可能保留整个分区或分区的一部分，但取决于将重新实现的其他分区及保留级别。

表示实例级分区状态的图标如下：


下一实现周期将保留分区数据。


下一实现周期将实现分区。


下一实现周期可能保留整个分区或分区的一部分，但取决于将重新实现的其他分区及保留级别。

在 ISE 9.1i 中，分区不支持以下实现选项：

- 多次多起点的布局布线 (MPPR) 选项（–s、–n 等）
- 逻辑优化 (–logic_opt)
- 全局优化 (–global_opt)
- SmartGuide (–smartguide)

当资源利用率过高时，请将父分区划分为较小的子分区。要缩短工具运行时间，请将大分区划分为几个较小的分区。如果不能将大分区划小，则将剩余分区的保留级别从 “Routing”改为 “Placement”有助于缩短运行时间。

大约在 15–20 个实现周期之后，Xilinx 建议实现时将保留级别设置为 “Placement”或 “Synthesis”（综合）以优化整个设计。这样可以针对新逻辑或修改的逻辑优化布局和布线。这样做有可能改善总体实现结果。

用 Synplify 8.8.1 使用分区

可以用编译点约束为 Synplify Pro 和 Synplify Premier 工具赋予创建分区的功能。define_compile_point 命令用于在顶层约束文件 (SDC) 中定义编译点。要定义分区，请对每个编译点或模块使用一条 define_compile_point 命令。您可以用 Scope 工具修改 SDC 文件，以此在 Synplify Pro 中创建此命令。SDC 文件显示为一个附加标签页，其中有若干子标签页。子标签页之一是 “Compile Points”（编译点），可以选择要定义为编译点的模块。选择模块后，必须将其类型指定为 “locked, partition”（锁定，分区）。这样即可将以下约束添加到 SDC 文件：

- define_compile_point {v:work.multigenHD_vert} –type {locked, partition} –cpfile {}
- define_compile_point {v:work.multigenHD_horz} –type {locked, partition} –cpfile {}

- 向 SDC 文件添加编译点之后，Synplify 会将 “PARTITION”字符串和时间戳添加到每个层级节点的 EDIF 文件。创建 ISE 项目后，将 EDIF 作为设计源加入。ISE 在解析 EDIF 文件时会用 “PARTITION”字符串为 EDIF 中的每个层级节点创建一个分区。

Synplify Pro 创建的 EDIF 将用于在 ISE 项目浏览器中创建一个项目，以确保在实现期间能识别和保留分区。

用 Tcl 使用分区

也可以通过 Xilinx Tcl 界面创建分区。《开发系统参考手册》的 Tcl 章节中讲述了 Tcl 界面。Xilinx “partition”Tcl 命令包含一个用于分区的子命令集。以下示例脚本显示如何在设计中创建和管理分区。

创建分区的样本 Tcl 脚本：

```
puts "Create new project dev_ccir_top.ise\n"
project new dve_ccir_top.ise

puts "Set the device\n"
project set family Virtex4
project set device xc4vlx15
project set speed -11
project set package sf363

puts "Add the HDL source files\n"
xfile add Hdl/dve_ccir_aps.v
xfile add Hdl/dve_ccir_dds.v
xfile add Hdl/dve_ccir_dph.v
xfile add Hdl/dve_ccir_fir.v
xfile add Hdl/dve_ccir_lut.v
xfile add Hdl/dve_ccir_mlt8x9.v
xfile add Hdl/dve_ccir_top.v
xfile add dve_ccir_top.ucf
xfile add Hdl/dve_ccir_vtg.v

puts "Define the partitions\n"
partition new /dve_ccir_top/DATAPATH
partition new /dve_ccir_top/DATAPATH/CHROMA_FIR
partition new /dve_ccir_top/GENERATOR

puts "set the implementation tool options\n"
# 设置批处理应用选项：
#XST options
# 1. 将综合优化目标设置为 speed
project set "Optimization Goal" Speed
#translate options
# 2. 忽略 NGDBuild 中的所有 LOC
project set "Use LOC Constraints" FALSE
#map options
# 3. 执行时序驱动封装
project set "Perform Timing-Driven Packing and Placement" TRUE
#par options
# 4. 使用最高级 par 努力程度
project set "Place & Route Effort Level (Overall)" High
# 5. 设置特级 par 努力程度
project set "Extra Effort (Highest (PAR level only)"  "Continue on
Impossible"
# 6. 设置 verbose report type
project set "Report Type" Verbose
# 7. 将 “-instyle xflow” 送至 par 命令行
project set "Other Place & Route Command Line Options" "-intsyle xflow"
# 8. 从 trce 生成详细报告
project set "Report Type" "Verbose Report"
# 9. 在 bitgen 过程中创建 IEEE 1532 文件
project set "Create IEEE 1532 Configuration File" TRUE

puts "Run implementation tools\n"
if {[catch {process run "Implement Design"}]}{
   puts "Caught an Error executing process or time commands"
   exit 1
}
process "Generate Post-Place & Route"

puts "Close project\n"
project close
```

取出和修改分区属性的样本 Tcl 脚本：

```
#open project file
puts "open project file"
project open dve_ccir_top.ise

puts "Getting Partition Properties\n"
# Returns the preservation level for this Partition
partition get /dve_ccir_top/DATAPATH/CHROMA_FIR preserve
# Returns status of Implementation results of this Partition (true or false)
partition get /dve_ccir_top/DATAPATH/CHROMA_FIR up_to_date_implementation

puts "Modifying the Partition Properties\n"
# Forces the Partition to rerun Synthesis
partition rerun /dve_ccir_top/DATAPATH/CHROMA_FIR synthesis
# Sets the preservation level of this Partition to Placement
partitions set /dve_ccir_top/DATAPATH/CHROMA_FIR preserve placement

puts "Run implementation tools\n"
if {[catch {process run "Implement Design"}]}{
   puts "Caught an Error executing process or time commands"
   exit 1
}
process "Generate Post-Place & Route"
# close the project
puts "close the project"
project close
```

分区的保留级别

保留级别指示实现工具如何对每个复制的分区进行操作。可以为设计中的每个分区设置保留级别。默认情况下，将顶层模块的保留级别设置为 “Routing”，而将较低级分区的保留级别设置为 “Inherit”。默认的保留级别将保留设计的从综合到布线的所有实现数据。设置各分区保留级别的方法可以是：在源视图中右键单击 Partition，然后选择 “Partition Properties”（分区属性），如图 2 所示。

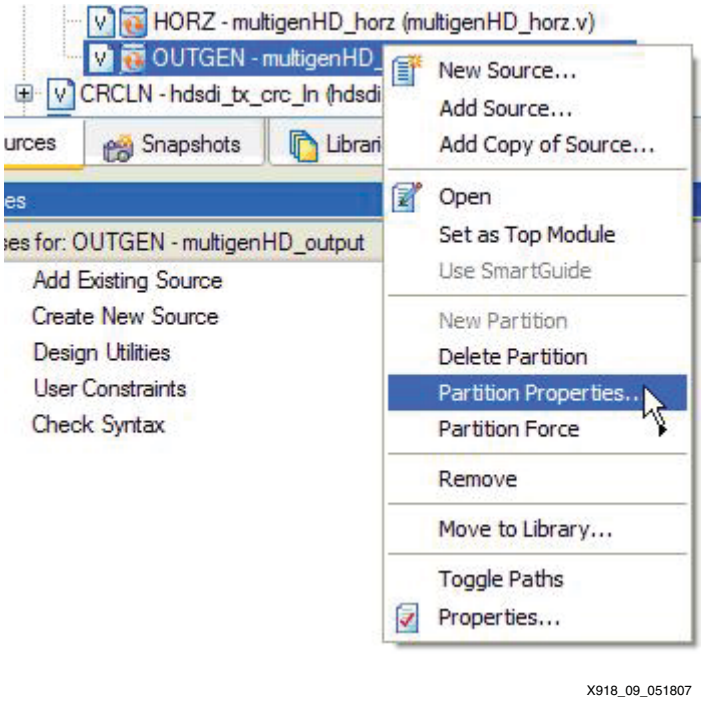


图 2: **设置各分区的保留级别**

可以将各分区的默认保留级别 “Routing” 改成其他级别。保留的设计数据量随着保留级别从 “Routing” 变为 “Placement” 再变为 “Synthesis” 而减少。如果设计不满足时序约束目标，或者无法为设计布线或适合设计，请放宽或降低分区的保留级别。可以将保留设置为以下级别之一：

- 布线
- 布局
- 综合
- 继承

布线保留级别通过布线保留分区的数据。此保留级别给予实现工具满足时序或实现目标的灵活性最小，但提供的保留程度最高。当 “Preserve” 设置为 “Routing” 时，会保留以下实现数据：

- 综合的网表和信息
- 布局信息
- 布线信息

布局保留级别通过布局保留分区的数据。如果重新实现的分区需要另一分区的布线资源，则 “Preserve” 设置为 “Placement” 的已保留分区的布线资源可能被重新实现的分区修改。分区始终会尝试保留布线。因此，即使将 “Preserve” 设置为 “Placement”，布线也会被保留，除非为了便于成功实现另一分区而必须修改布线。PAR 报告文件 (design.par) 中的分区摘要告诉用户是否已经修改分区的布线。当 “Preserve” 设置为 “Placement” 时，始终会保留以下实现数据：

- 综合的网表和信息
- 布局信息
- 某些布线信息，可能所有布线信息

综合保留级别只保留设计的综合网表。如果重新实现的分区需要另一分区的布局和布线资源，则可以在实现过程中修改 “Preserve” 设置为 “Synthesis” 的已保留分区的资源。如果 “Preserve” 设置为 “Synthesis” 的分区的资源没有修改，则会准确地保留分区。当 “Preserve” 设置为 “Synthesis” 时，会保留以下实现数据：

- 综合的网表和信息
- 某些布局信息，可能是所有布局信息
- 某些布线信息，可能是所有布线信息。

Inherit 保留级别将 “Preserve” 属性设置为与父分区相同的级别。所有子分区的默认设置均为继承。顶层分区的默认设置是布线。所以，所有设计的初始状态均为全部子分区的 “Preserve” 都设置为 “Inherit”，其值解算为顶层分区的 “Routing”。

分区保留

未修改的分区在后续实现周期中重新使用。修改 HDL 源代码或改变物理约束会导致重新实现分区。如果输入源文件已经改变，分区会自动检测到，从而强制重新实现该分区。某些类型的源改变会强制重新实现所有分区。这些类型的改变包括：

- 时序约束的改变
- 过程属性（MAP/PAR 的努力程度等优化属性）的改变
- 项目属性（器件、速度级别等）的改变
- 删除实现文件，例如从项目目录中删除 NGD 或 NCD

可以使用 “Force”（强制）命令手动强制重新实现分区。这与 HDL 源文件的修改或保留级别无关。强制重新实现分区只影响该分区，不会强制重新实现整个设计。要强制改变分区的状态，请右键单击该分区，然后选择 “Partition Force –> Force Synthesis Out–of–date”（分区强制 – 强制综合作废）或 “Partition Force –> Force Implementation Design Out–of–date”（分区强制 – 强制实现设计作废），如图 3 所示。

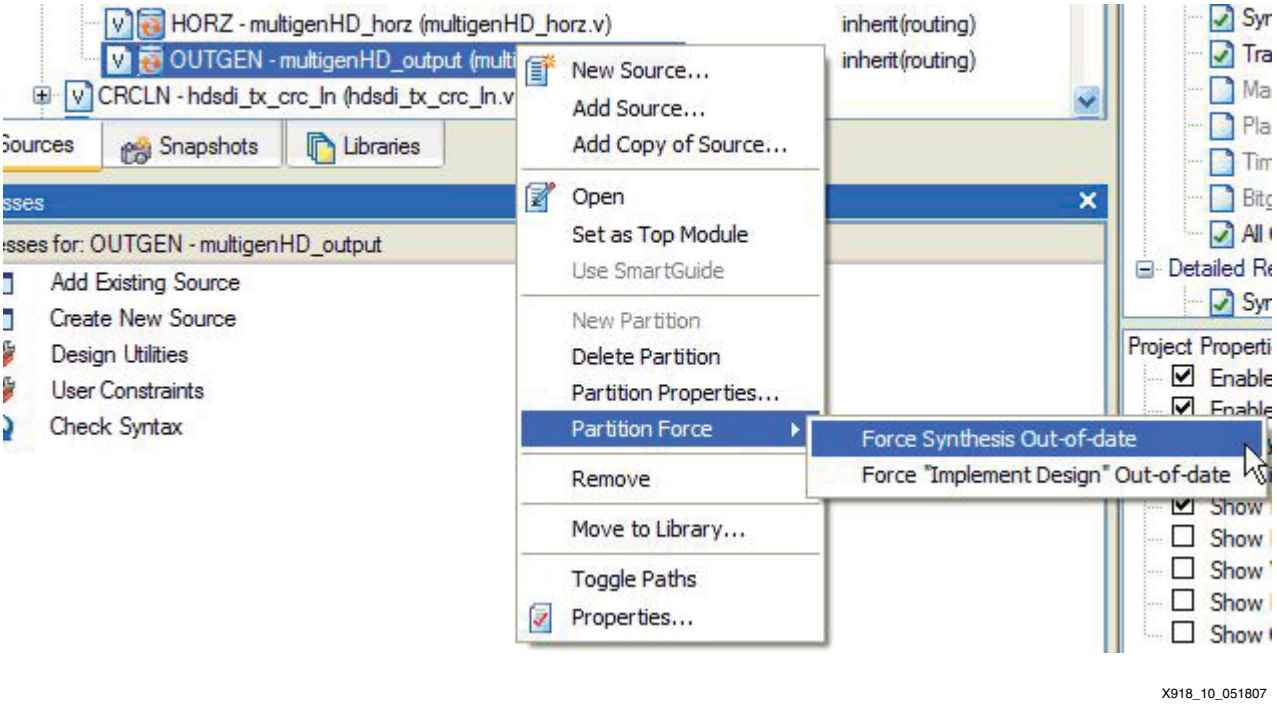


图 3: **强制改变分区的状态**

在 Partition Force 菜单中，可以使用以下选项：

- Force Synthesis Out–of–date – 致使重新综合并重新实现分区
- Force Implement Design Out–of–date – 致使重新实现分区，保留已综合的网表

一旦使用 Partition Force 命令，则无论以前的实现数据如何都会重新实现分区。在下一实现周期之后，将根据分区的保留级别保留分区。

对分区进行布局规划

分区不需要区域组约束或范围等布局约束。可以用分区所关联逻辑的 Floorplan 选项解决时序收敛问题。Xilinx 建议不要对分区所关联逻辑进行布局规划，除非分区难以满足时序目标。解决时序收敛问题的方法最好是合理编制一整套时序约束。不能用布局规划取代时序约束。经过布局规划的分区可以将分区所关联逻辑约束在器件的特定区域，以此提高设计的性能。area_group 范围约束并非设置在分区上，而是设置在与分区关联的逻辑或实例上。

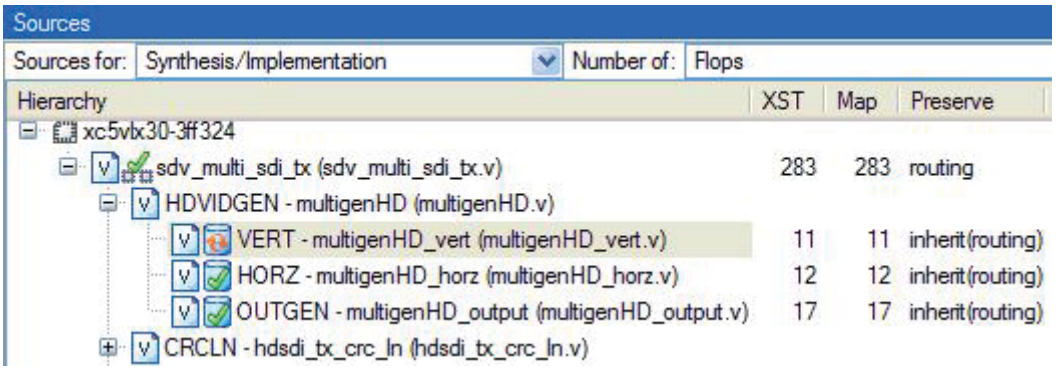
要检查设计单元的布局，请查看有效位置；要创建 area_group 约束，请使用布局规划编辑器和布局规划器。对分区逻辑进行布局规划时，在用户约束文件 (UCF) 中为关联的实例给定一个 SLICE 约束范围，类似于 “SLICE_X46Y73:SLICE_X20Y100”。area_group 约束的范围定义在器件中放置分区中逻辑的位置。层级节点中的所有逻辑元件都会包括在位置约束内。

在布局规划编辑器中，可以通过拖放操作将与分区关联的实例或逻辑置入器件的某个区域。矩形区域的大小经过自动估算，以适合实例中的逻辑量。将实例置入矩形区域后，根据需要修改该区域的形状和大小。布局规划编辑器的设计规则检查 (DRC) 功能可确保所放置实例的矩形区域符合实现工具的规则。area_group 范围约束告诉实现工具必须将相应逻辑包容在该矩形区域的边界之内。

分区报告

用分区实现设计后，请用 “设计摘要”和各种实现报告检查各分区占用的资源。如果分区过大或具有高难度时序约束，则该分区的实现可能需要比预期更长的时间。如果可能，应将这些分区划分为较小的分区。

Synplify Pro、XST、NGDBuild、MAP 和 PAR 生成的报告包含有关分区的信息，如保留状态和实现期间使用的资源。引导报告文件 (GRF) 提供重新实现的元件、新元件以及设计中与 SmartGuide 关联的网状连线的有关细节。XST 综合报告和 MAP 报告中记述的各分区资源在项目浏览器的 Sources 标签页中相应分区旁边 （如图 4 所示）和 “设计摘要”（如图 5 所示）中反映。



X918_11_051807

图 4: **XST 和 MAP 中的各分区资源估算**

LAB1 Partition Summary			
Partition Name	**Synthesis Status**	**Placement Status**	**Routing Status**
/sdv_multi_sdi_tx	Preserved	Preserved	Preserved
/sdv_multi_sdi_tx/EDH/DEC	Preserved	Preserved	Preserved
/sdv_multi_sdi_tx/HDVIDGEN/HORZ	Preserved	Preserved	Preserved
/sdv_multi_sdi_tx/HDVIDGEN/OUTGEN	Preserved	Preserved	Preserved
/sdv_multi_sdi_tx/HDVIDGEN/VERT	Implemented	Implemented	Implemented

X918_12_051807

图 5: **各分区的状态摘要**

Synplify Pro 综合报告包含以下针对分区的子标题：

- 编译点摘要 – 列出保留的分区

```
Name                  Status        Reason
-------------------------------------------------
multigenHD_vert       Remapped      Design changed
multigenHD_horz       Remapped      Design changed
sdv_multi_sdi_tx      Unchanged     -
```

XST 综合报告在 “分区实现状态”项下包含以下针对分区的子标题：

- 保留的分区 – 列出保留的分区
  - 示例：Partition "/sdv_multi_sdi_tx/EDH/DEC"
- 实现的分区 – 列出实现的分区以及实现这些分区的原因
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/VERT":
    - HDL source file(s) were modified.
- 分区 NGC 文件 – 列出所有分区和关联的 NGC 文件
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/HORZ":
    - NGC File: HDVIDGEN_HORZ#multigenHD_horz.ngc

NGDBuild 报告在 “分区实现状态”项下包含以下针对设计中分区的子标题：

- 保留的分区 – 列出保留的分区
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/HORZ"
    - Implemented Partitions - List of implemented Partitions and reason why they were implemented
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/VERT":
    - Synthesis modified the Partition.

MAP 报告在第 9 节 “区域组与分区摘要” -> “分区实现状态”项下包含以下针对设计中分区的子标题：

- 保留的分区 – 列出保留的分区
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/HORZ"
- 实现的分区 – 列出实现的分区以及实现这些分区的原因
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/VERT":
    - An upstream application reimplementation of the Partition.
- 分区资源摘要 – 列出使用的所有分区和资源
  - 示例：Partition "/sdv_multi_sdi_tx/EDH/DEC":
    - Slice Logic Utilization:
    - Number of Slice Registers:221
    - Number of Slice LUTs:263
    - Number used as logic:263
  - Slice Logic Distribution:
    - Number of occupied Slices:136
    - Number of LUT Flip Flop pairs used:357
    - Number with an unused Flip Flop:136 out of 357 38%
    - Number with an unused LUT:93 out of 357 26%

```
    - Number of fully used LUT-FF pairs: 128 out of 357
      35%
```

- 区域组信息 – 列出 SLICE 的位置和使用形式
  - 示例：Area Group "AG_HDVIDGEN/HORZ"
    - No COMPRESSION specified for Area Group "AG_HDVIDGEN/HORZ"
  - RANGE: SLICE_X7Y49:SLICE_X2Y46
  - Slice Logic Utilization:
    - Number of Slice Registers:12 out of _12%
    - Number of Slice LUTs:33 out of _34%
    - Number used as logic:33
  - Slice Logic Distribution:
    - Number of occupied Slices:11 out of 24
      45%
    - Number of LUT Flip Flop pairs used:33
    - Number with an unused Flip Flop:21 out of 33
      63%
    - Number with an unused LUT:0 out of 33
      0%
    - Number of fully used LUT-FF pairs:12 out of 33
      36%
      - Note: Percentages are based upon an AREA GROUP for this Partition.
  - Number of Block RAM/FIFOs:1

PAR 报告在 “分区实现状态”项下包含以下针对设计中分区的子标题：

- 保留的分区 – 列出保留的分区
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/HORZ"
- 实现的分区 – 列出实现的分区以及实现这些分区的原因
  - 示例：Partition "/sdv_multi_sdi_tx/HDVIDGEN/VERT":
    - An upstream application reimplementation of the Partition.

删除分区

如果使用分区达不到缩短运行时间或促进设计保留的目的，就可能有必要删除一个或几个分区。这时，与所删除分区关联的逻辑就会成为其父分区（可能是顶层分区）的一部分。该分区的命令和属性将不再适用于该分区中原有的逻辑。要从设计的分区层级结构中删除分区，请在项目浏览器中右键单击该分区，然后选择 “Delete Partition”（删除分区）（如图 6 所示）。

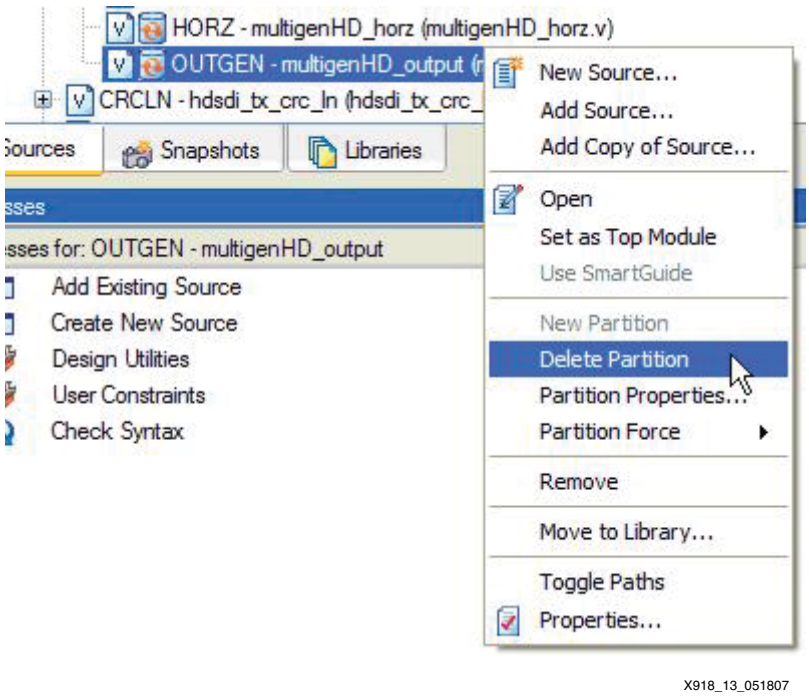


图 6: **从设计的分区层级结构中删除分区**

分区和旧增量设计法

ISE 8.1i 及以前版本不支持用分区法保留设计。ISE 的这些版本使用增量引导和增量综合来保留设计。以下增量设计功能与使用分区的设计不兼容：

- XCF 文件中的 XST 增量综合约束
- MAP 引导模式 “–gm incremental”
- PAR 引导模式 “–gm incremental”
- Area_group 范围（分区不需要此功能，但布局规划可能使用此功能。）

建议不要在 ISE 8.2i 及以后版本中使用增量引导和增量综合法。应使用分区进行设计保留。如果将使用增量设计法的较旧设计移植到 ISE 8.2i 或以后版本，应删除这些约束并以分区代之。

结论

建议在主要或时序关键性模块上放置分区。这样，综合与实现工具就可以重新使用和综合旧版设计以及实现已变更的分区。这样做有助于缩短实现工具的运行时间。可以通过项目浏览器或 Tcl 界面创建分区。《开发系统参考手册》的 Tcl 章节中讲述了 Tcl 界面。

建议使用 “map –timing” 帮助改善设计的封装并提高其密度。如果设计具有较高资源利用率，则建议使用 80–90% 的 “封装系数” 创建空间，以避免布局布线问题。

在大多数设计中，保留设计数据比重新实现整个设计更快。可以通过使用分区或 SmartGuide 来缩短实现时间。在一个用来测试分区和 SmartGuide 的大型设计套件中，平均运行速度大约提高到了原实现的 2.5 倍。在最好情况下，使用分区的设计交互作用的运行速度提高到了不使用分区的同等设计变更的 6 倍。

修订历史

下表说明此技术文档的修订历史。

日期	版本	修订
2007 年 6 月 7 日	1.0	最初版本