部分可重构实现流程

（基于 PR_AddSub 工程）

一、 编写 RM 的代码

1. 编写 2 个 RM_AddSub.v 文件（分别实现加法和减法运算），分别存放在 /Sources/RM_Add/和/Sources/RM_Sub/路径下。

二、 生成 RM 的.dcp 文件

1. 搭建整个系统，分别使用 2 个 RM_AddSub.v 文件进行综合，将生成的 design_1_RM_AddSub_0_0.dcp 文件分别复制到/Synth/RM_AddSub/add/和 /Synth/RM_AddSub/sub/路径下；
2. 可以进行后续的布线等操作，进行功能验证。

三、 Draw Pblock（利用加法的 RM_AddSub 模块）

1. 设置 RM 的属性 HD.RECONFIGURABLE 为 true；
2. 进行 Draw Pblock 操作，选择合适的位置和大小；
3. 进行 DRC 检查，Rules 选择 Partial Reconfiguration，若没有错误和警告，则 Draw Pblock 操作完成，否则需要修改 Pblock；
4. 保存约束文件。

四、 生成不同配置下的.dcp 文件

1. 进行布局布线（利用加法的 RM_AddSub 模块），

   opt_design

   place_design

   route_design

   保存 top_route_design.dcp 文件到/Implement/Config_add/路径下，

   write_checkpoint -force Implement/Config_add/top_route_design.dcp

   生成 bit 文件，更新软核，关闭工程；

   close_project
2. 打开/Implement/Config_add/top_route_design.dcp 文件，

   open_checkpoint Implement/Config_add/top_route_design.dcp

   将 RM_AddSub 模块设置为 black_box，即删除原有的 RM_AddSub 模块，

   update_design -cell design_1_i/RM_AddSub -black_box

执行 lock_design 命令，锁住静态部分的布局布线，

```
lock_design -level routing
```

保存 top_route_design.dcp 文件到/Implement/Config_static/路径下，

```
write_checkpoint -force Implement/Config_static/top_route_design.dcp
```

关闭工程；

```
close_project
```

3. 打开/Implement/Config_static/top_route_design.dcp 文件，

```
open_checkpoint Implement/Config_static/top_route_design.dcp
```

读入/Synth/RM_AddSub/sub/design_1_RM_AddSub_0_0.dcp 文件，

```
read_checkpoint -cell design_1_i/RM_AddSub
Synth/RM_AddSub/sub/design_1_RM_AddSub_0_0.dcp
```

进行布局布线，

```
opt_design
place_design
route_design
```

保存 top_route_design.dcp 文件到/Implement/Config_sub/路径下，

```
write_checkpoint -force Implement/Config_sub/top_route_design.dcp
```

关闭工程；

```
close_project
```

4. 打开/Implement/Config_static/top_route_design.dcp 文件，

```
open_checkpoint Implement/Config_static/top_route_design.dcp
```

设置与 RM_AddSub 模块相关的端口接地（或接 VCC），

```
update_design -buffer_ports -cell design_1_i/RM_AddSub
```

进行布局布线，

```
place_design
route_design
```

保存 top_route_design.dcp 文件到/Implement/Config_blank/路径下，

```
write_checkpoint -force Implement/Config_blank/top_route_design.dcp
```

关闭工程。

```
close_project
```

五、 Run PR_Verify

1. 执行以下命令：

```
pr_verify -initial Implement/Config_add/top_route_design.dcp -additional
{Implement/Config_sub/top_route_design.dcp
Implement/Config_blank/top_route_design.dcp}
```

没有报错则关闭工程。

六、 生成 bit 文件

1. 打开/Implement/Config_add/top_route_design.dcp 文件，

```
open_checkpoint Implement/Config_add/top_route_design.dcp
```

生成 bit 文件（实际会生成 2 个 bit 文件），

write_bitstream -force -file Bitstreams/Config_add.bit

关闭工程；

2. 打开/Implement/Config_sub/top_route_design.dcp 文件，

open_checkpoint Implement/Config_sub/top_route_design.dcp

生成 bit 文件，

write_bitstream -force -file Bitstreams/Config_sub.bit

关闭工程；

3. 打开/Implement/Config_blank/top_route_design.dcp 文件，

open_checkpoint Implement/Config_blank/top_route_design.dcp

生成 bit 文件，

write_bitstream -force -file Bitstreams/blanking.bit

关闭工程。

七、 测试

通过上述步骤，最终得到的 bit 文件如图 1 所示。

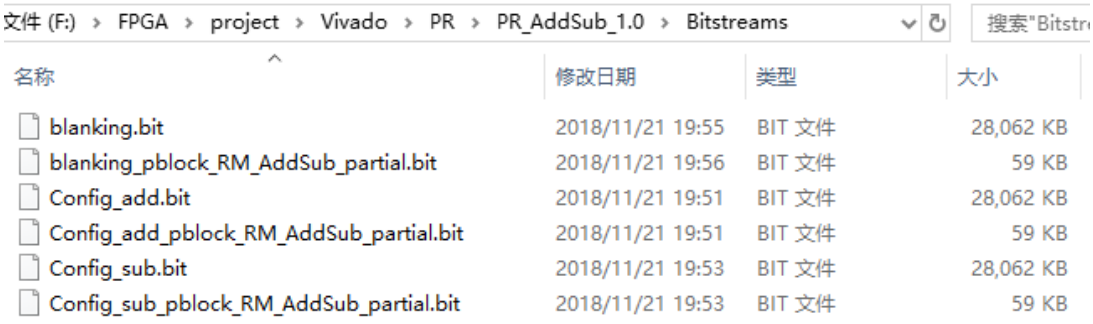

图 1 Bitstreams 文件夹截图

测试步骤如下：

1. 打开原始工程和 SDK，打开 SDK Terminal；
2. 下载 blanking.bit 文件和.elf 文件，SDK Terminal 中的输入输出如下：
   输入：1 2
   输出：1 + 2 = 0
3. 下载 Config_add_pblock_RM_AddSub_partial.bit，输入输出如下：
   输入：1 2
   输出：1 + 2 = 3
4. 下载 Config_sub_pblock_RM_AddSub_partial.bit，输入输出如下：
   输入：1 2
   输出：1 - 2 = -1
5. 下载 blacking_pblock_RM_AddSub_partial.bit，输入输出如下：
   输入：1 2
   输出：1 + 2 = 0
6. 下载 Config_add.bit 文件和.elf 文件，SDK Terminal 中的输入输出如下：
   输入：1 2

输出：1 + 2 = 3

7. 重复步骤 3~5；
8. 下载 Config_sub.bit 文件和.elf 文件，SDK Terminal 中的输入输出如下：

   输入：1 2

   输出：1 - 2 = -1
9. 重复步骤 3~5。

若所有结果与上面描述的一致，则测试成功。

注意：下载 blanking.bit/Config_add.bit/Config_sub.bit 文件后，需要重新下载.elf 文件，下载其他 bit 文件则不需要。