



基于 VIVADO2022.2 版本 应用教程

摘要

远赴人间惊鸿宴，一睹人间盛世颜。最是人间留不住，朱颜辞镜花辞树

SUN

[电子邮件地址]

文档修改记录

[illegible]

目录

1	概述	3
2	软件设置	4
	Vivado 文本编辑器设置	4
	CPU 多线程 (WIN)	4
3	名词解释 (FPGA/软件)	5
	时钟	5
	仿真	6
4	名词解释 (接口相关)	8
	DDR	8
	HDMI	8
	VCU	8
	ETH	8
	PCIE	8
5	应用技巧	9
	软件及对应第三方软件版本	9
	DCP 生成步骤	9
	DCP 文件仿真	9
	EDIF 生成步骤	9
	编辑和修改官方 IP 核	9
	BIT 文件校验与回读	10
	增量式编译	10
	Vivado Store	12
	代码加密	12
	将代码文件直接添加到 BD	12
	Tcl 脚本配置 IP 方式	13
	Tcl 脚本下载程序	13
6	官方 IP 使用说明	14
	RAM 初始化	14
7	个人 IP 使用说明	15
8	硬件接口设计	15

1 概述

本文使用软件版本：Vivado 2022.2，包含 Tcl 示例工程。ISE 或者其它低版本 Vivado，不推荐参考某些工程示例或工程！

本文内容基于 Xilinx 官方文档总结或参考。

2 软件设置

Vivado 文本编辑器设置

常用 notepad++ 和 VS code

vivado->settings ->Text Editor -> Custom Editor

path/notepad++.exe [file name] -n[line number]

e.g.

D:\Program Files\Notepad++\notepad++.exe [file name] -n[line number]

D:\Programs\Microsoft VS Code\Code.exe [file name] -n[line number]

CPU 多线程 (WIN)

建立文件并命名 Vivado_init.tcl

内容: set_param general.maxThreads 8

将 tcl 文件放入...\Vivado\2022.2\scripts 文件夹

3 名词解释（FPGA/软件）

时钟

1, 时钟概述

CMT: 时钟管理块, 包含一个 MMCM 和一个 PLL

MMCM: 混合模式时钟管理器

PLL: 锁相环

BUFG: 全局时钟线, 7 系每个片子有 32 条

BUFH/BUFHCE: 水平时钟缓冲器, 允许通过水平时钟行访问单个时钟区域中的全局时钟线。它还可以用作时钟使能电路 (BUFHCE), 以独立启用或禁用跨越单个时钟区域的时钟

使用每个时钟区域中的 12 条水平时钟线, 每个时钟区域最多可支持 12 个时钟。

BUFMR: 多时钟区域缓冲器, 允许区域和 I/O 时钟跨越多达三个垂直相邻的时钟区域。

CMT 概述:

每个 7 系列 FPGA 最多有 24 个 CMT, 每个 CMT 由一个 MMCM 和一个 PLL 组成。MMCM 和 PLL 用作各种频率的频率合成器, 用作外部或内部时钟的抖动滤波器, 以及去歪斜时钟。该锁相环包含 MMCM 功能的子集。7 系列 FPGA 时钟输入连接允许多种资源为 MMCM 和 PLL 提供参考时钟。

MMCM 在任一方向具有无限精细相移能力, 可用于动态相移模式。MMCM 在反馈路径或一个输出路径中也有一个小数计数器, 从而实现频率合成能力的进一步粒度

MRCC: MRCC 用于本时钟区域和相邻时钟区域

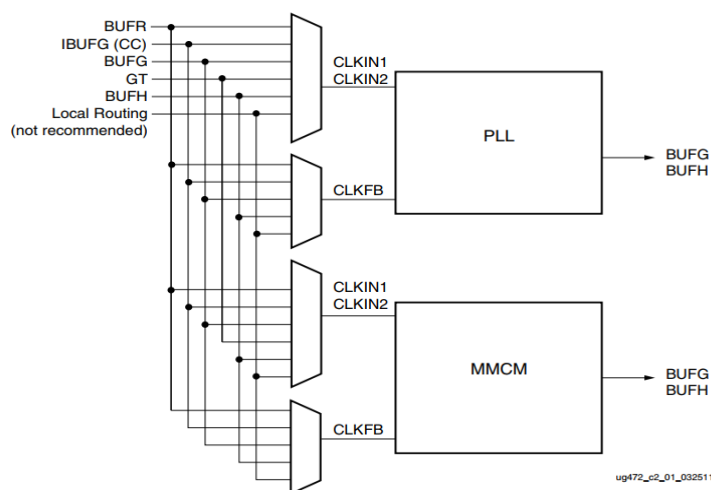
SRCC: 本区域时钟区域

两者都可以连接到全局时钟

BUFIO: IO buffer

2, 时钟管理单元

CMT



MMCM

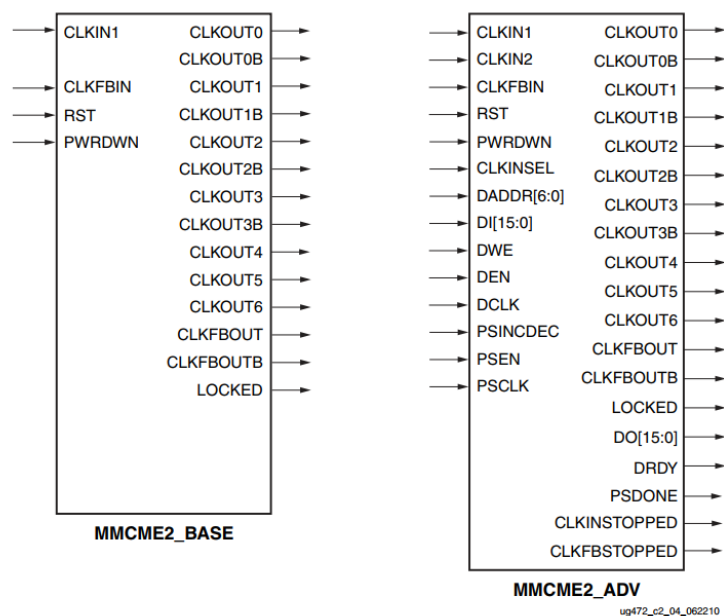


Figure 3-4: MMCM Primitives

PLL

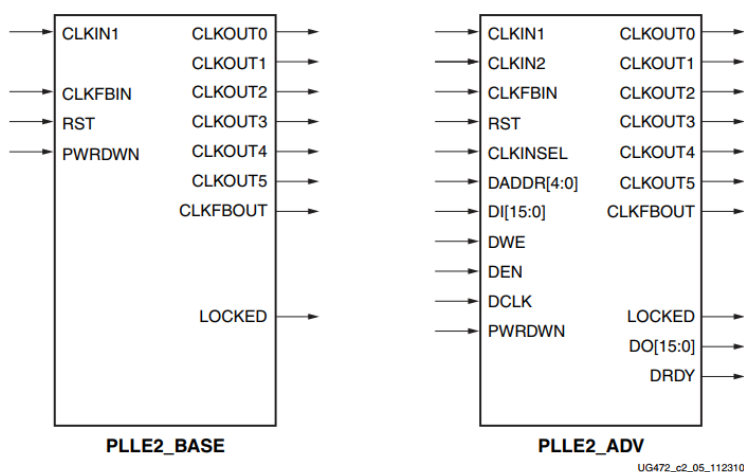


Figure 3-5: PLL Primitives

3, 时钟支持动态重配, 支持 AXI lite 和 DRP 接口
但并不常用, 参考示例 XAPP888

仿真

仿真类型

1, 仿真含三种

RTL 级

一般都是逻辑代码

Post-Synthesis Simulation

可以对网表进行仿真

Post-Implementation Simulation

实现后执行功能或时序仿真。时序仿真是最接近实际将设计下载到设备的仿真

2, 编译库时, GCC executable path 为 GCC 路径, 用于 SystemC IP。可以忽略!

3, 仿真库部分解释

UNISIM: 原语功能仿真

UNIMACRO: 宏功能仿真

UNIFAST: 快速仿真库

SIMPRIM: 原语时序仿真

SECUREIP: 用于功能和时序仿真库, Xilinx 器件特性仿真, 例如 PCIE, 收发器等

XPM: 原语的功能仿真

4 名词解释（接口相关）

DDR

HDMI

VCU

ETH

PCIE

SG (scatter/gather) 模式

scatter/gather 指的在多个缓冲区上实现一个简单的 I/O 操作，比如从通道中读取数据到多个缓冲区，或从多个缓冲区中写入数据到通道；

scatter（分散）：指的是从通道中读取数据分散到多个缓冲区 Buffer 的过程，该过程会将每个缓存区填满，直至通道中无数据或缓冲区没有空间；

gather（聚集）：指的是将多个缓冲区 Buffer 聚集起来写入到通道的过程，该过程类似于将多个缓冲区的内容连接起来写入通道；

5 应用技巧

软件及对应第三方软件版本

Vivado 每个版本对应的第三方工具版本是不一致的，如果不对应可能出错！
使用不同版本 Vivado，请参考官方文档为 ug973

DCP 生成步骤

disable XDC 文件

设置生成网表程序 top 层

输入以下命令

```
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top <top_module_name>
```

```
-part <part>
```

```
write_checkpoint <file_name>.dcp
```

命令解释

Command Option	Description
-mode out_of_context	Prevents I/O insertion for synthesis and downstream tools. The mode is saved in checkpoints if write_checkpoint is issued.
-flatten_hierarchy rebuilt	There are several values allowed for -flatten_hierarchy, but rebuilt is the recommended setting for HD flows.
-top	This is the module/entity name of the module being synthesized. This switch can be omitted if set_property top <top_module_name> [current_fileset] is issued prior to synth_design.
-part	This is the Xilinx part being targeted (e.g., xc7k325tffg900-3)

Figure 1

详细信息参考 ug905 page10

DCP 文件仿真

ViVado 软件直接打开 DCP 文件，或者 open_checkpoint XX.dcp

write_verilog -mode funcsim XX.v （具体参考 write_verilog 命令）

使用生成的.v 文件即可进行仿真！

EDIF 生成步骤

设置生成网表程序 top 层

在综合选项中去掉 IOBuffer，具体操作为在综合设置窗口的 Options 下面最后一项 More Options 一栏写入 -no_iobuf;

综合完成后，打开 Synthesized Design，并在 Tcl Console 中输入：write_edif path/xx.edif 例化时，要保留一个跟 eidf 同名的 hdl 文件，且文件中只保留 **module** 的接口和 parameter 参数

Tcl 命令：

不含 Xilinx IP

```
write_edif F:/FPGA/abc.edf
```

包含 Xilinx IP

```
write_edif -security_mode all F:/FPGA/abc.edf
```

编辑和修改官方 IP 核

在某些情况下，需要修改官方 IP 核。由于 vivado 版本不同，方法步骤也不尽相同，，具体参考软件使用版本对应文档。

本章节不具体介绍方法步骤！

具体参考文档为 UG896

Editing or Overriding IP Sources 章节

BIT 文件校验与回读

回读

使用 Vivado tcl 命令可以回读程序，可以生成 ASCII 和二进制两种格式！

ASCII (rbd)

```
readback_hw_device -readback_file a.rbd
```

二进制 (bin)

```
readback_hw_device -bin_file a.bin
```

校验

校验的前提是生成 mask file

另外：软件可以永久禁止回读，通过配置寄存器禁止 JTAG 访问

增量式编译

增量包含两部分，增量综合和增量布局布线！

过程如下：

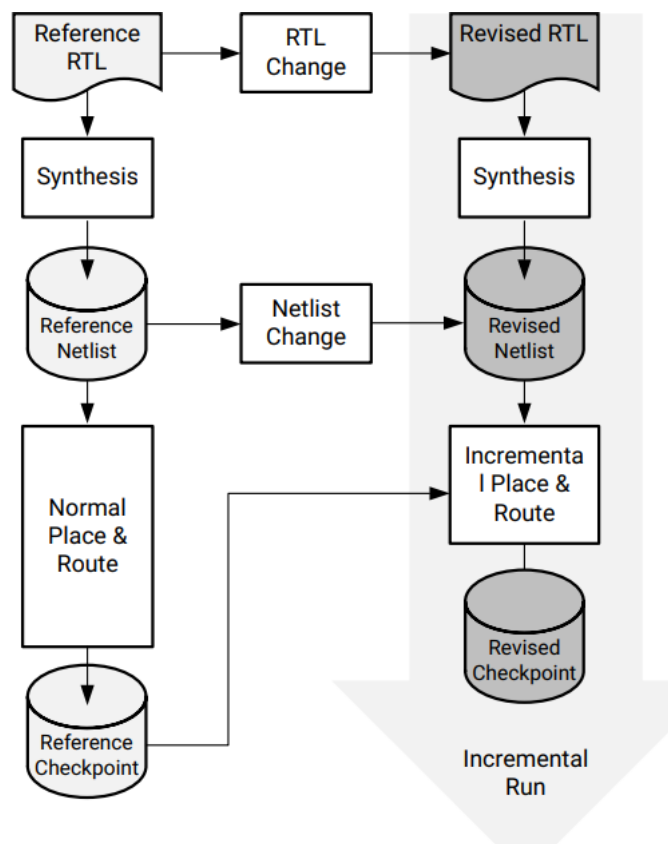


Figure 2

增量综合

在此过程中，该工具会将增量综合信息放入生成的 DCP 文件中，以便在以后的运行中引用。它将检测设计何时发生更改，然后仅对已更改的设计部分重新运行综合。此流程的主要优势在于，对于更改很小的设计，运行时间将显著减少。此外，在 RTL 中插入小的更改时，设计的 QoR 波动会更小。

Settings -> Synthesis -> Incremental Synthesis 进行设置

Incremental Synthesis: 可以选择使用已知 dcp，最后创建 dcp（默认），关闭增量综合

incremental_mode: 描述跨分区优化，可以选择 Quick, default, aggressive, off

增量布局布线:

发生在增量编译设计流程的实现阶段

通过重复使用参考设计中先前的布局和布线，保持 QoR 的可预测性。

加快布局和布线运行时间或尝试最后一英里时序收敛。

增量设计包括 RTL 更改、网表更改或两者。

允许更改约束，但一般收紧约束将显著影响布局和布线，通常最好在增量流程之外添加

Incremental Implementation 和 Incremental Synthesis 是有区别的，前者默认状态是没用设置的，需要在软件上进行设置

Incremental Directives: 包含以下三种方式

RuntimeOptimized

指令尝试尽可能多地重用来自参考运行的布局和布线信息。计时目标将与参考运行相同。如果参考运行的 WNS 为 -0.050 ，那么增量运行将不会尝试关闭此设计的时序，而是以 -0.050 为目标。这仅影响设置时间。这是未指定指令时的默认行为

TimingClosure

指令将重用参考中的布局和布线，但它会破坏不符合时序的路径并尝试关闭它们。运行一些运行时间密集型算法以获得尽可能多的时序改进，但由于很大程度上放弃了布局，因此前端增益有限。这种技术对参考 WNS > -0.250 ns 的设计非常有效。

Quick

是一种特殊的模式，在布局布线过程中不调用定时器，而是以相关逻辑的布局为指导。这是最快的模式，但不适用于大多数设计

设计需要 WNS > 1.000 ns 才能有效。这些通常是 ASIC 仿真或原型设计

增量实现包含三种模式：自动，高复用，低重用！

自动

需满足以下条件

94% cell 匹配

90% net 匹配

WNS > -0.250

高复用

Cell 重复率高于 75%，运行

低复用

用户可以针对单元类型、分层单元、时钟区域和 SLR 进行重用

目标 WNS 始终为 0

增量指令被忽略，而使用默认布局布线算法中的指令

低重用模式对于在特定区域对布局和布线提出挑战的设计最为有效！

备注：增量式编译，用户一般只进行实现部分，综合部分软件默认即可！这是一个较为复杂的过程，此文仅为简单介绍！详细参考相关官方文档！另外部分内容不同软件版本有稍许不同！

Vivado Store

Vivado Store 将 tcl 应用程序，pcb 和可配置的示例设计整合到一个位置。所有的项目都显示在 GUI 中，并且可以安装，删除或升级单个项目。

TCL 应用程序：主要为与 Vivado 设计套件一起使用而设计的 Tcl 代码的开源存储库。

Boards：Xilinx 和第三方托管板文件的 GitHub 存储库。将电路板文件与 Vivado 结合使用可以通过将电路板级资源集成到设计环境中来简化设计创建。

示例设计：由 Xilinx 和第三方可配置示例设计组成的 GitHub 存储库
可以添加和扩展 tcl 命令，也可以进行修改。可以自定义部分 tcl 命令！
还有一些示例工程链接，有一定参考价值！

代码加密

Xilinx 软件 Vivado 可以对 verilog 或 VHDL 代码进行加密

加密方式采用 RSA 加密方式！

加密注意事项

- 1, 每个版本加密文件只能用于此版本，不支持其他版本混用
- 2, 加密文件为.vp 文件
- 3, Xilinx 每个软件版本均提供公版 RSA 密钥，不能混用。
- 4, Verilog 和 VHDL 加密命令不同
- 5, 可以对整个代码加密，也可以加密代码一部分。
整个代码加密无端口号，key 文件要进行部分修改。

以 Verilog 文件加密为例

a), 代码和密钥文件独立

```
encrypt -lang verilog -ext .vp -key keyfile.txt user.v
```

b), 将密钥写入代码（全加密/部分加密）

```
encrypt -lang verilog -ext .vp user.v
```

将代码文件直接添加到 BD

选中代码文件

- 1, set_property source_mgmt_mode All [current_project]
 - 2, create_bd_cell -type module -reference <module_name> <module_name_inst>
- 或者 右键 add module to block design

如若顶层内含有一般 IP，转成 xci 格式！

```
set_property generate_synth_checkpoint 0 [get_files <ip_name>.xci]
```

此方式类似一般打包 IP，但不能包含 dcp 文件。

Tcl 脚本配置 IP 方式

我们在分析了解某些官方示例工程中，，有时会出现一些不同情况。

比如，某些 IP 有些端口我在 GUI 配置时无法隐藏，或者不用，但却无法进行配置。

Tcl 脚本下载程序

Download.bat

内容如下

```
////////////////////////////////////
call C:\Xilinx\Vivado\2022.2\bin\vivado.bat -mode batch -source download.tcl

if exist *isWriteableTest*.tmp del /F *isWriteableTest*.tmp
if exist vivado_*.backup.jou del /F vivado_*.backup.jou
if exist vivado_*.backup.log del /F vivado_*.backup.log
if exist vivado_*.str del /F vivado_*.str
if exist *isWriteableTest*.tmp del /F *isWriteableTest*.tmp
pause
////////////////////////////////////

Bit_download.tcl
内容如下
////////////////////////////////////
open_hw
connect_hw_server -url localhost:3121
current_hw_target [get_hw_targets */xilinx_tcf/Digilent/*]
set_property PARAM.FREQUENCY 15000000 [get_hw_targets */xilinx_tcf/Digilent/*]
open_hw_target
set_property PROGRAM.FILE {example_top.bit} [lindex [get_hw_devices] 0]
current_hw_device [lindex [get_hw_devices] 0]
refresh_hw_device -quiet [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 0]
refresh_hw_device -quiet [lindex [get_hw_devices] 0]
close_hw
```

6 官方 IP 使用说明

RAM 初始化

IP

```
memory_initialization_radix = 2;
memory_initialization_vector =
000000000000010110000000100000000
000000000000010110000001000000001
000000000000000010000110100001100
000000000000000010000111000001101;
```

FIR

```
Radix = 16;
CoefData= 346, EDA, 0D6, F91, F91, 0D6, EDA, 346;
注意对应 IP 参数配置
```

逻辑代码方式

```
reg [DATA_WIDTH-1:0] ram [DEPTH-1:0];
integer i;
initial for (i=0; i<DEPTH; i=i+1) ram[i] = 0;
end
```

```
ASCII 文本 reg [31:0] ram [0:63];
initial begin
$readmemb("rams_20c.data", ram, 0, 63);
end
```

7 个人 IP 使用说明

8 硬件接口设计