sun

[电子邮件地址]

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

基于Vivado2022.2版本应用教程

**文档修改记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 版本 | 日期 | 人员 | 章节 | 修订范围 |
| 1 | 1.0 | **2023.03.19** |  | 所有 | 首次发布 |
|  |  | **2023.04.16** |  |  | 更新小节 |
|  |  | **2023.05.27** |  |  | 更新小节 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[**1** **概述** 3](#_Toc136096509)

[**2** **软件设置** 4](#_Toc136096510)

[**Vivado文本编辑器设置** 4](#_Toc136096511)

[**CPU多线程（WIN）** 4](#_Toc136096512)

[**3** **名词解释（FPGA/软件）** 5](#_Toc136096513)

[**时钟** 5](#_Toc136096514)

[**仿真** 6](#_Toc136096515)

[**4** **名词解释（接口相关）** 8](#_Toc136096516)

[**DDR** 8](#_Toc136096517)

[**HDMI** 8](#_Toc136096518)

[**VCU** 8](#_Toc136096519)

[**ETH** 8](#_Toc136096520)

[**PCIE** 9](#_Toc136096521)

[**5** **应用技巧** 10](#_Toc136096522)

[**软件及对应第三方软件版本** 10](#_Toc136096523)

[**DCP生成步骤** 10](#_Toc136096524)

[**DCP文件仿真** 10](#_Toc136096525)

[**EDIF生成步骤** 10](#_Toc136096526)

[**编辑和修改官方IP核** 10](#_Toc136096527)

[**BIT文件校验与回读** 11](#_Toc136096528)

[**增量式编译** 11](#_Toc136096529)

[**Vivado Store** 13](#_Toc136096530)

[**代码加密** 13](#_Toc136096531)

[**将代码文件直接添加到BD** 13](#_Toc136096532)

[**Tcl脚本配置IP方式** 14](#_Toc136096533)

[**Tcl脚本下载程序** 14](#_Toc136096534)

[**6** **官方IP使用说明** 15](#_Toc136096535)

[**RAM初始化** 15](#_Toc136096536)

[**MIG配置** 15](#_Toc136096537)

[**7** **个人IP使用说明** 21](#_Toc136096538)

[**8** **硬件接口设计** 21](#_Toc136096539)

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。

1. **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

1. **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

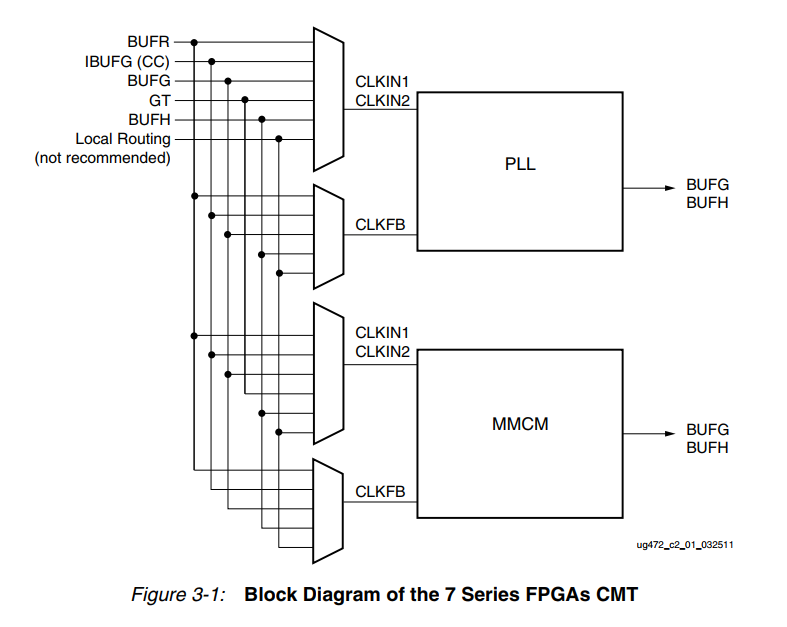
SRCC：本区域时钟区域

两者都可以连接到全局时钟

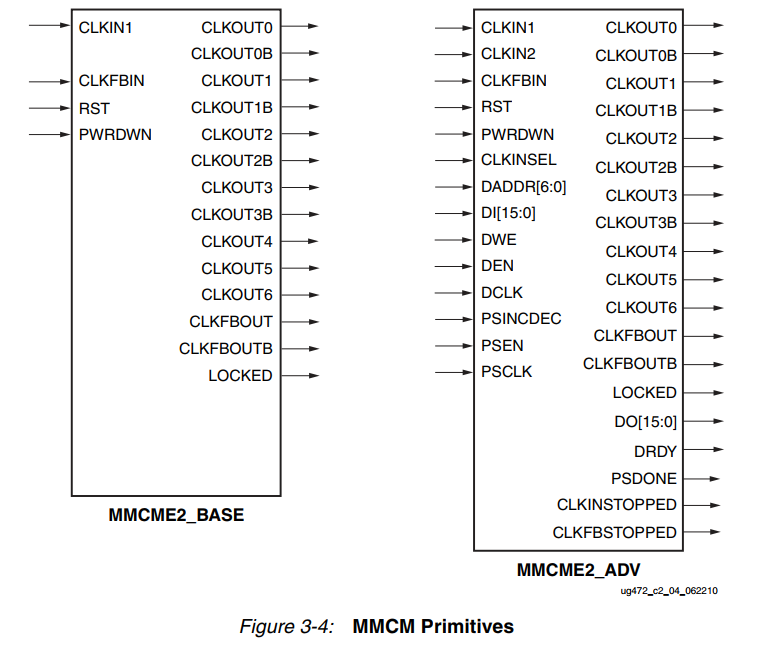
BUFIO: IO buffer

1. 时钟管理单元

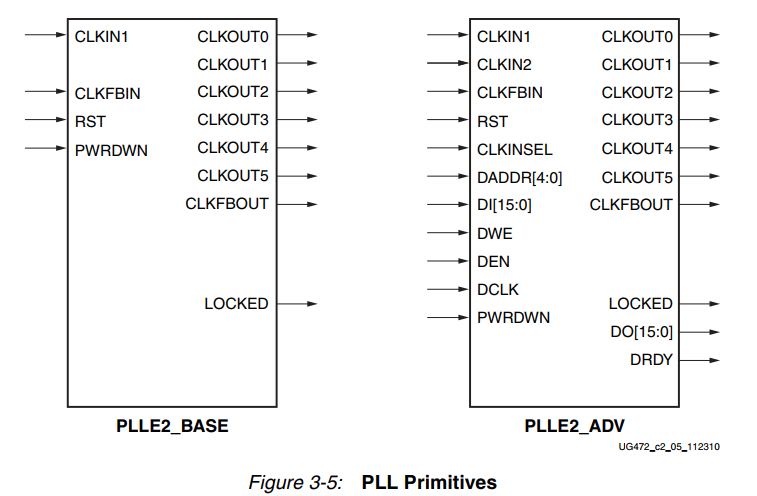
CMT



MMCM



PLL



1. 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

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

1. **名词解释（接口相关）**

**DDR**

**HDMI**

**VCU**

**ETH**

**分层概念**

[OSI模型](https://en.wikipedia.org/wiki/OSI_model)

[Internet protocol suite](https://en.wikipedia.org/wiki/Internet_protocol_suite)

[VLAN](https://en.wikipedia.org/wiki/Virtual_LAN)

[Precision\_Time\_Protocol（IEEE 1588）](https://en.wikipedia.org/wiki/Precision_Time_Protocol)

[POE](https://en.wikipedia.org/wiki/Power_over_Ethernet)

**帧概念相关**

[以太网帧](https://en.wikipedia.org/wiki/Ethernet_frame)

[巨型帧](https://en.wikipedia.org/wiki/Jumbo_frame)

IPv4/UDP/ICMP/ARP/FCS

[UDP lite](https://en.wikipedia.org/wiki/UDP-Lite)

**总结**

1,严格来说，整个帧（前导码到FCS）都属于数据链路层范围！

2,数据帧指的是IP数据，不包含首部和尾部之这18个字节。

常规帧长度64-1518，所以数据帧的长度在46-1500字节之间

3,IP头为20byte，所以IP数据区最长1480。

4,如果TCP或UDP包数据长度超过1500，，系统进行分包操作。

5, UDP 校验和计算对于 IPv4 是可选的。如果不使用校验和，则应将其设置为零值

6, TCP整个包的最大长度是由最大传输大小决定。UDP整个包的最大长度为65535

TCP

整个包的最大长度是由最大传输大小（MSS，Maxitum Segment Size）决定，MSS就是TCP数据包每次能够传输的最大数据分段

UDP

从理论上说，包含报头在内的数据报的最大长度为65535字节(64K)

7, ICMP属于网络层协议

**帧组成示例**

以TCP和UDP（IPV4）为例，，帧最少64字节，最高1518字节

我们知道数据帧长度46 – 1500。即

46 = 64 – 18（首部+FCS）

1500= 1518 – 18（首部 +FCS）

以帧最低长度64为例

TCP

首部（14byte）+ IP头（20byte）+ TCP头（20byte）+payload（6byte） + FCS（4byte）

UDP

首部（14byte）+ IP头（20byte）+ UDP头（8byte）+payload（18byte） + FCS（4byte）

这就出现一个问题，假如TCP或UDP传输少于payload最低字节数，就需要填充0补充到最低字节

**关于wireshark**

PC接收数据：

物理层的网卡会去掉前导码和帧开始符，然后对帧进行CRC校验，如果帧校验错误，丢弃此帧！如果校验和正确，就判断帧的目的硬件地址是否符合自己接收条件（目的地址是自己的物理硬件地址、广播地址、可接收的多播硬件地址等），如果符合，就将帧交“设备驱动程序”做进一步处理。这时我们的抓包软件才能抓到数据**，因此，抓包软件抓到的是去掉前导同步码、帧开始分界符、FCS之外的数据**

PC发送数据：

Wireshark是可以抓到FCS的。

为什么会出现接收和发送，wireshark有无FCS的区别呢？

Wireshark，IP头校验和错误

这是由于网卡属性->高级->硬件校验和的关系。默认RXTX开启，由网卡计算。

**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；

1. **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

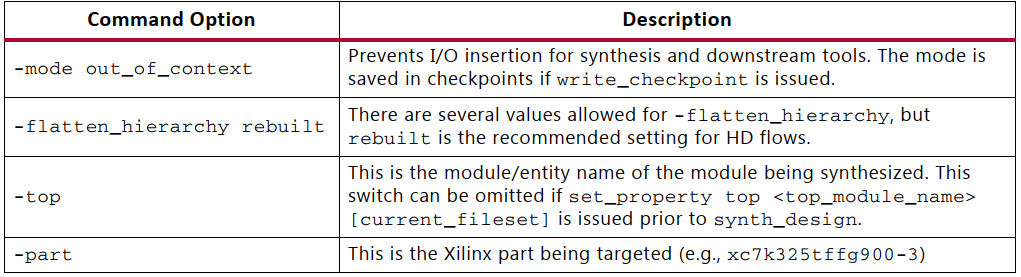
设置生成网表程序top层

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



Figure

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](https://so.csdn.net/so/search?q=module&spm=1001.2101.3001.7020) 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

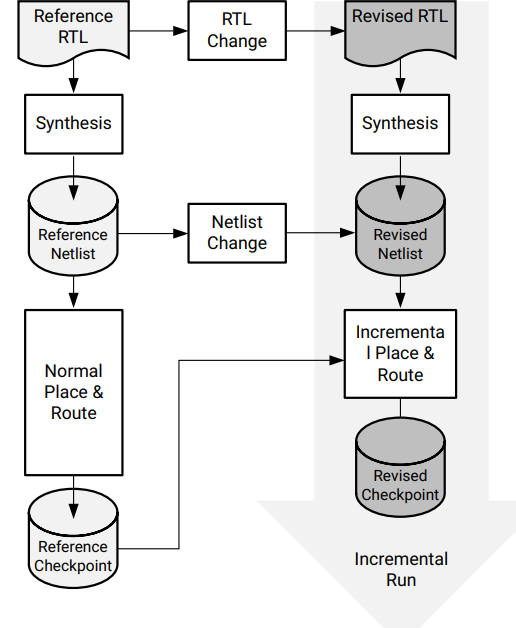
校验的前提是生成mask file

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

***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，运行时间将显着减少。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着影响布局和布线，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

1. **官方IP使用说明**

**RAM初始化**

IP

memory\_initialization\_radix = 2;

memory\_initialization\_vector =

00000000000010110000000100000000

00000000000010110000001000000001

00000000000000010000110100001100

00000000000000010000111000001101；

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

**MIG配置**

使用和配置MIG前，，需要了解DDR芯片或者内存条的容量，组成方式，数据宽度等基本信息!

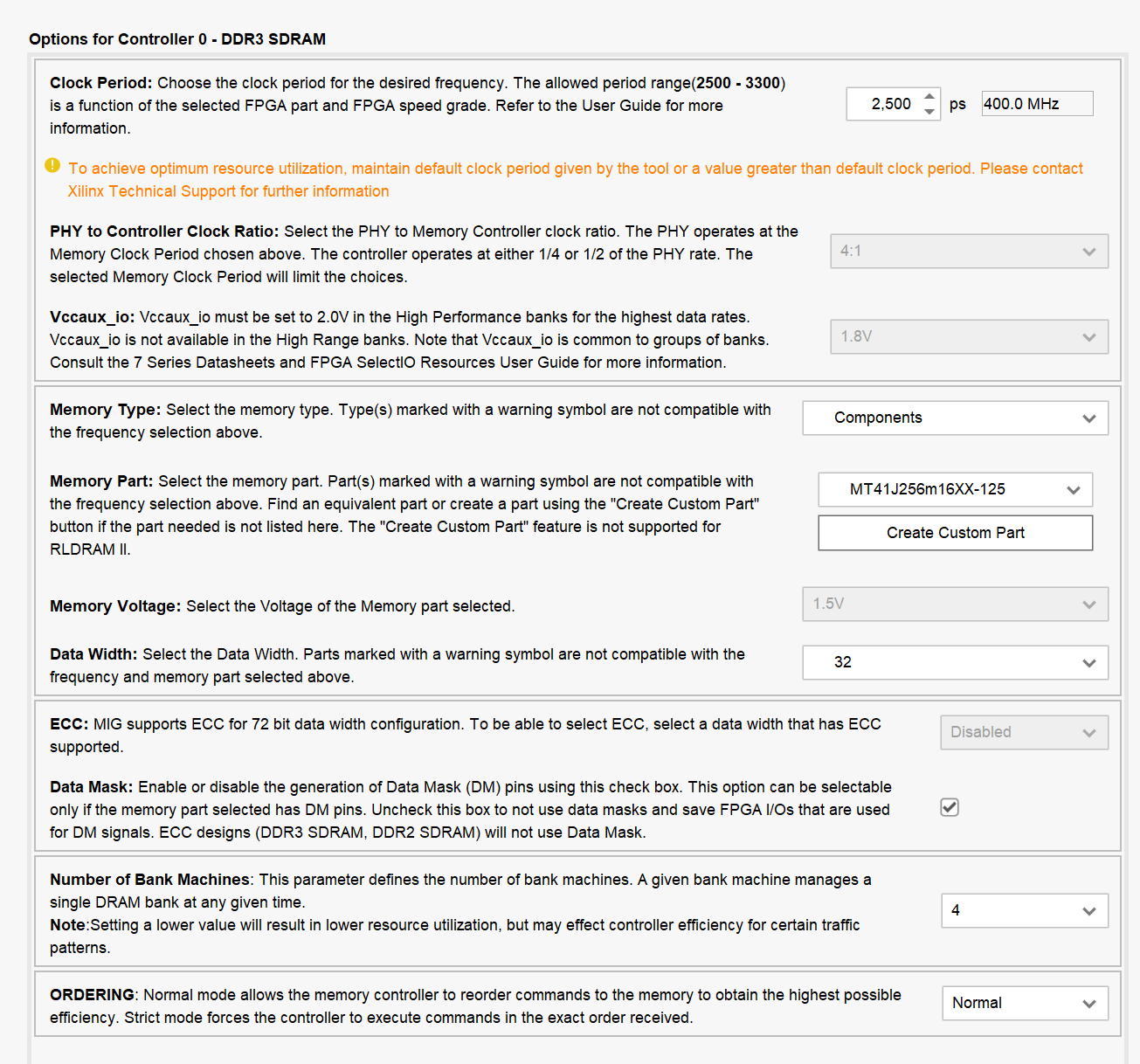
MIG 不同芯片选择的IP核些许不同！但方式原理大体相似！

下面简要介绍IP配置关键参数及含义！

逻辑模式：MIG可以选择逻辑接口或者AXI

BD 模式：只能AXI模式

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Clock Period：

控制器的工作频率，该频率和FPGA芯片以及DDR芯片有关（DDR工作频率）

PHY to Controller Clock Ratio：（一般可选2:1 或 4:1）

以AXI为例，即DDR3频率与外部逻辑频率之比。

假设DDR频率为400MHZ，选择2:1，外部逻辑频率为200MHZ，该数据影响后续配置的位宽

Memory Type

Components：一般DDR芯片选择此项

RDIMM：带寄存器的双线内存模块（用于工作站服务器内存条）

UDIMM：无缓冲双信道内存模块 （一般台式机内存条）

SODIMM：小型双列直插式内存模块（笔记本内存条）

Memory Part :

若使用的DDR选项里没有，可以选择参数类似的，，或者可以先点击Creat Custom Part 进行创新建器件

（配置时需熟悉DDR芯片参数概念，诸如，row/col/bank/等）

Memory Voltage：

DDR电压，，参考使用DDR

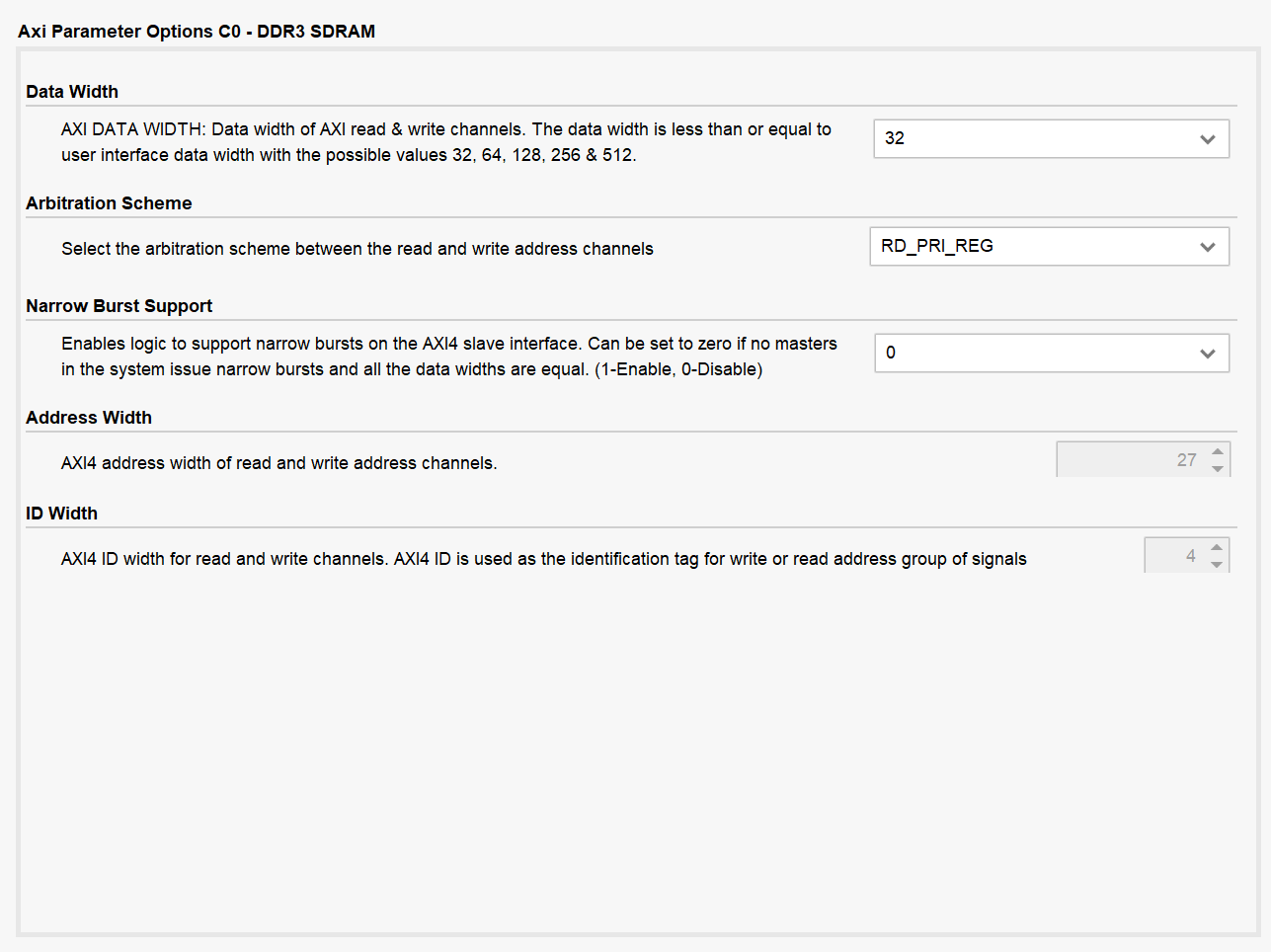
Data Width：

DDR 数据宽度，，如果单片16位，，两片并联则为32位

Number of Bank Machines：

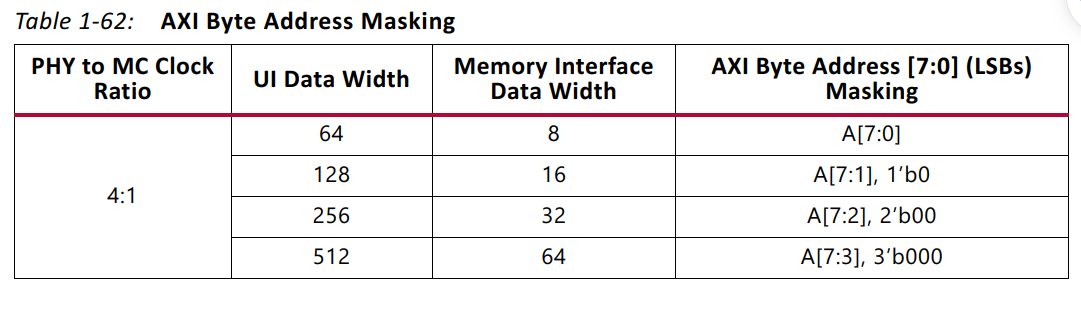
和设备大小有关和资源利用有关，默认4（无休修改）

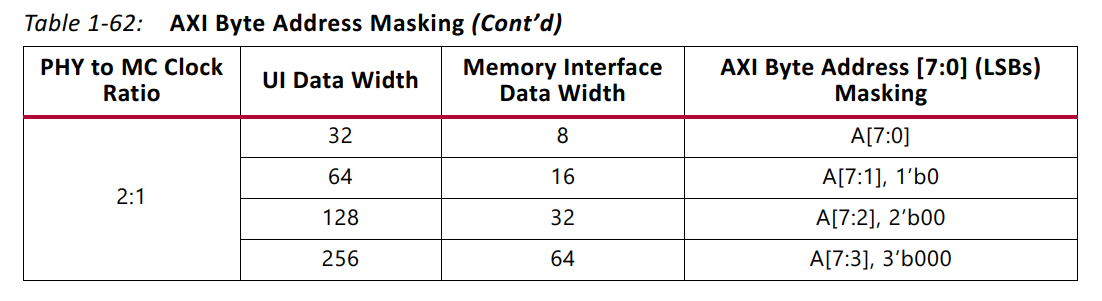
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



(参考page164)

Date WIdth：AXI 数据宽度（AXI时钟可以由ui\_clk）





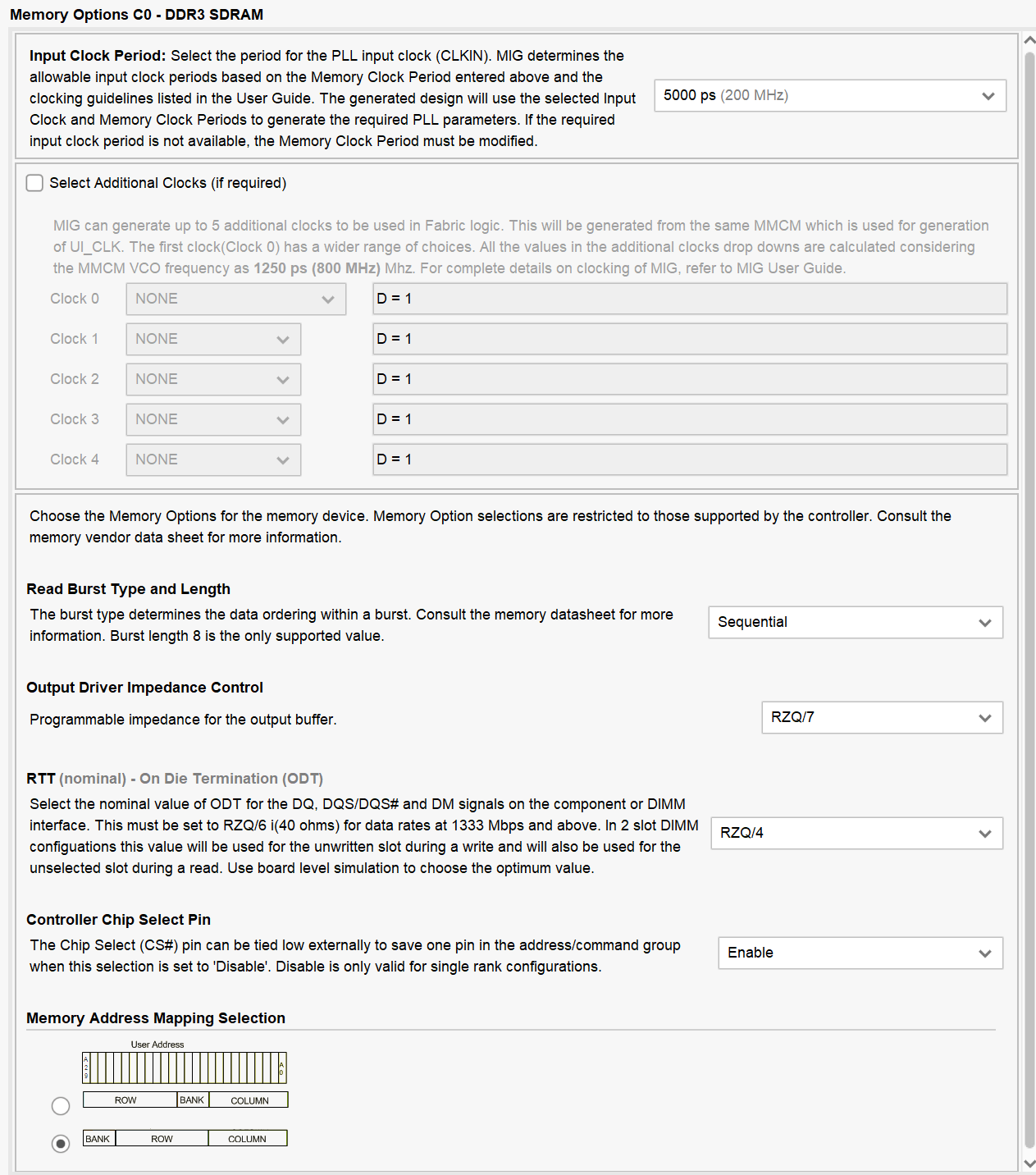
Arbitration Scheme：（仲裁方案）

在读写地址之间选择仲裁方案

Narrow Burst Suppot：（窄突发支持）

1-Enable ，0-Disable

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



input clock period:

与连接晶振输出时钟保持一致

Read Burst Type and Length：

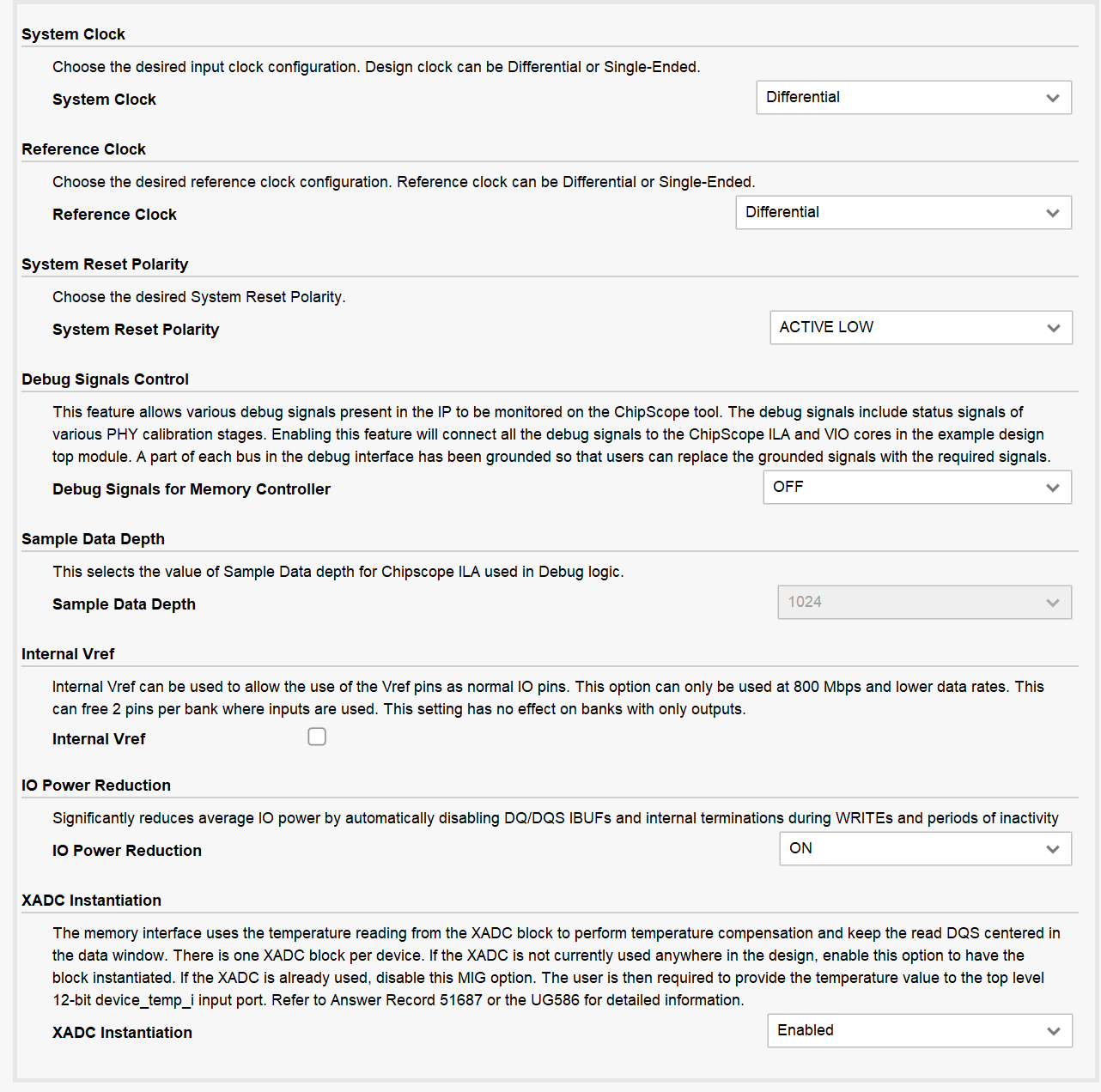
默认

Output Driver Impedance Control：（输出阻抗，RZQ是240欧，RZQ/6就是40欧）

查看DDR

RTT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



System Clock

Differential / Single-Ended 一般连接输入时钟

No Buffer 一般使用PLL输出时钟

Reference Clock

当输入频率在 199 和 201 MHz 之间（即输入时钟周期在 5,025ps (199 MHz) 和 4,975 ps (201 MHz) 之间）时，会出现Use System Clock选项。参考时钟频率基于数据速率请注意，添加MMCM以创建适当的REF\_CLK频率以上1,333 MB / s。当没有选择“无缓冲器”选项时，在RTL代码中不实例化IBUF原语，不为参考时钟分配引脚

如果从 MIG 为No Buffer选项生成的设计在未执行更改的情况下实施，则设计可能会由于未针对 ref\_clk\_i 信号实例化的 IBUF 而导致实施失败。因此对于No Buffer场景，需要将 ref\_clk\_i 信号连接到内部时钟

System Reset Polarity

选择高低电平复位（默认低电平复位）

1. **个人IP使用说明**
2. **硬件接口设计**