*User's Guide*

# *DLP DLPC964 Apps FPGA*


**摘要**

AMD Xilinx™ Virtex™-7 VC-707 Apps FPGA 旨在与 DLPC964 控制器和受支持的 DMD 配合使用。DLP® DLPC964 Apps FPGA 只是用于与 DLPLCRC964EVM 和 DLPLCR99EVM 连接的前端电路板的一个示例。DLPC964 Apps FPGA 用户指南将详细介绍 DLPC964 Applications FPGA (Apps FPGA) 的功能和寄存器以及所用的 VHDL 代码的组织结构。

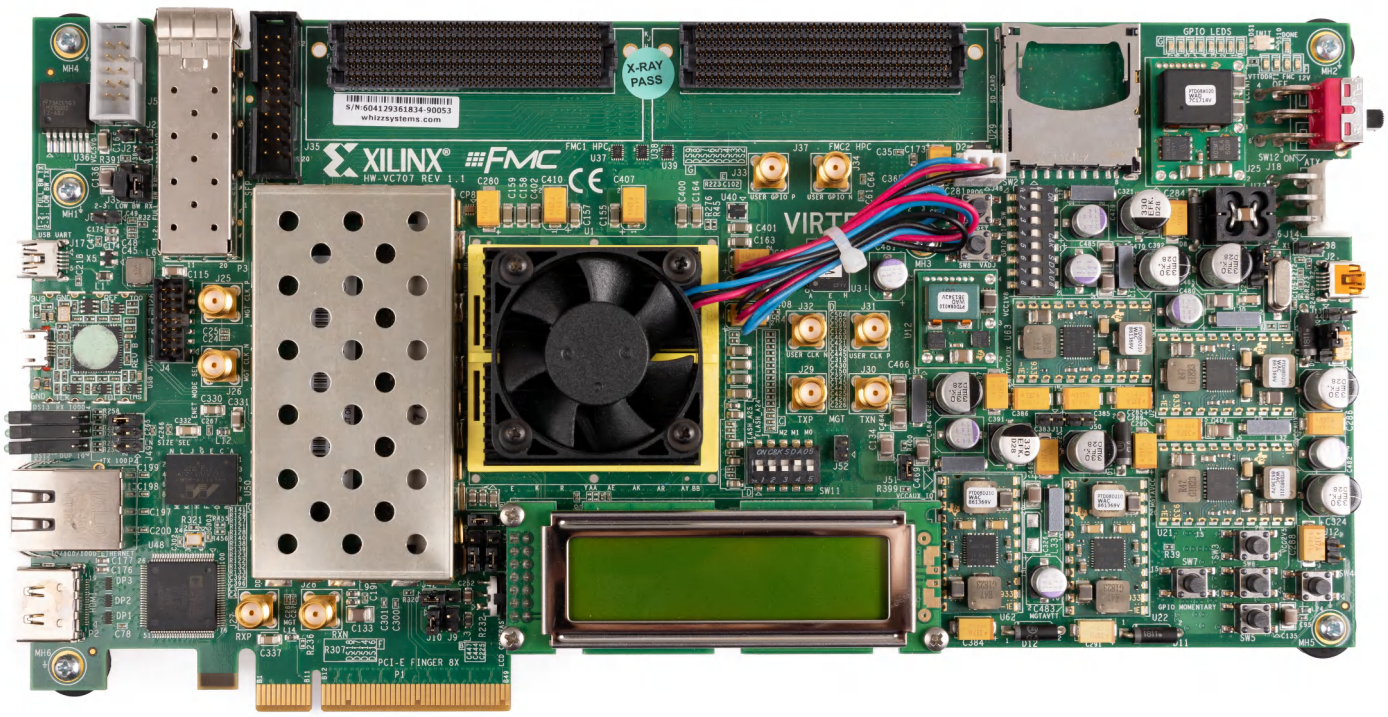




**图 1-1. AMD Xilinx Virtex-7 VC707 评估模块**

# 内容

**1 概述**....4
1.1 开始使用....4
1.2 特性....4
1.3 假设....4
1.4 Apps FPGA 硬件目标....4
**2 Apps FPGA 模块**....5
2.1 Apps FPGA 方框图....5
2.2 BPG 模块....6
2.3 BRG 模块....7
2.4 BRG_ST 模块....8
2.5 PGEN 模块....9
2.6 PGEN_MCTRL 模块....9
2.7 PGEN_SCTRL 模块....11
2.8 PGEN_PRM 模块....12
2.9 PGEN_ADDR_ROM....12
2.10 HSSTOP 模块....12
2.11 SSF 模块....13
2.12 ENC 模块....13
2.13 Xilinx IP....13
2.14 参考文档....13
2.15 DLPC964 Apps FPGA IO....14
2.16 关键定义....15
**3 功能配置**....17
3.1 启用的块数....17
3.2 图形循环启用....17
**4 附录**....25
4.1 Vivado Chipscope 捕获结果....25
4.2 DLPC964 Apps 位流加载....28
4.3 使用 Aurora 64B/66B 连接到 DLPC964 控制器....30
**5 缩略语和首字母缩写词**....48
**6 德州仪器 (TI) 相关文档**....48

# 插图清单

图 1-1. AMD Xilinx Virtex-7 VC707 评估模块....1
图 1-1. Apps FPGA 硬件目标....4
图 2-1. Apps FPGA 硬件方框图....5
图 2-2. BPG 模块硬件方框图....6
图 2-3. BRG 模块硬件方框图....7
图 2-4. BRG_ST 时序....8
图 2-5. PGEN 模块....9
图 2-6. PGEN_MCTRL FSM....10
图 2-7. PGEN_SCTRL FSM....11
图 2-8. HSSTOP 模块硬件方框图....12
图 2-9. x4 模式....15
图 2-10. 全局模式....15
图 4-1. 图形模式 0 捕获结果....25
图 4-2. 图形模式 1 捕获结果....25
图 4-3. 图形模式 2 捕获结果....26
图 4-4. 图形模式 3 捕获结果....26
图 4-5. 图形模式 4 捕获结果....26
图 4-6. 图形模式 5 捕获结果....27
图 4-7. 图形模式 6 捕获结果....27
图 4-8. 图形模式 7 捕获结果....27
图 4-9. FPGA 配置模式....28
图 4-10. GPIO DIP 开关 (VC707)....29
图 4-11. DLPC964 系统方框图....30

图 4-12. 从 IP Catalog 中选择.....31
图 4-13. 配置 Core Options.....31
图 4-14. 信道配置.....32
图 4-15. Shared Logic 选项.....32
图 4-16. 生成设计文件.....33
图 4-17. Aurora_apps_tx_x12ln.v RTL 方框图.....35
图 4-18. 块以块控制字开始的波形.....39
图 4-19. 块 DMDLOAD_REQ 置为有效结束之后的新块控制字波形.....40
图 4-20. DMDLOAD_REQ 延迟置为有效波形.....41
图 4-21. 三个 DMD 行的加载操作的 DMDLOAD_REQ 建立时间.....42
图 4-22. 块置位操作的 DMDLOAD_REQ 建立时间.....43
图 4-23. 单通道运行模式的系统方框图.....44
图 4-24. 单通道运行模式波形示例.....44
图 4-25. Aurora 数据总线到 DMD 块阵列的映射，递增方向.....46
图 4-26. Aurora 数据总线到 DMD 块阵列的映射，递减方向.....46
图 4-27. 使用 TI EVM 硬件时 Aurora Channel0 Link0 的 IBERT 眼图扫描.....47

## 表格清单

表 3-1. TPG 图形.....18
表 4-1. RTL aurora_apps_tx_x12ln.v 的信号端口列表.....36
表 4-2. RTL 包装器“aurora_apps_tx_x12ln.v”user-k 端口使用情况.....37
表 4-3. 块控制字段定义.....38
表 4-4. 用于控制 TX 收发器设置的 RTL 输入端口.....47

## 商标

# 1 概述

DLPC964 Apps FPGA 用户指南将介绍 DLPC964 Applications FPGA (Apps FPGA) 的功能和寄存器。Apps FPGA 旨在与 DLP LightCrafter DLPC964 EVM (DLPLCRC964EVM) 和配套的 DMD EVM (DLPLCR99EVM) 配合使用。此外，本指南还将概述 VHDL 代码和实现。

**备注**

DLPLCR99EVM、DLPLCRC964EVM、AMD Xilinx VC-707 评估板、电源、光学元件和光源单独出售。

## 1.1 开始使用

请访问 DLPLCRC964EVM 工具页面和 DLPLCR99EVM 工具页面了解有关每个 EVM 的更多信息，并查看 TI E2E DLP 产品论坛获取更多帮助。

## 1.2 特性

AMD Xilinx Virtex-7 VC-707 评估套件是用于 DLPLCRC964EVM 和 DLPLCR99EVM 的前端评估模块，其中包含 32 个 HSSI 输入数据信道，每个数据信道的速度高达 3.6Gb/s。

## 1.3 假设

以下文档假设用户已从 ti.com 运行 DLPC964 Apps 可执行程序。RTL、Vivado™ 工程和各种脚本位于：C:\Texas Instruments-DLP\DLPC964-Apps\。

## 1.4 Apps FPGA 硬件目标

Apps FPGA 参考代码专用于搭载在 AMD Xilinx VC-707 评估板上的 AMD Xilinx Virtex-7 FPGA。图 1-1 展示了 VC-707 评估板如何连接到德州仪器 (TI) DLPC964 评估模块 (DLPLCRC964EVM)，而后者又连接到 DLPLCR99EVM。

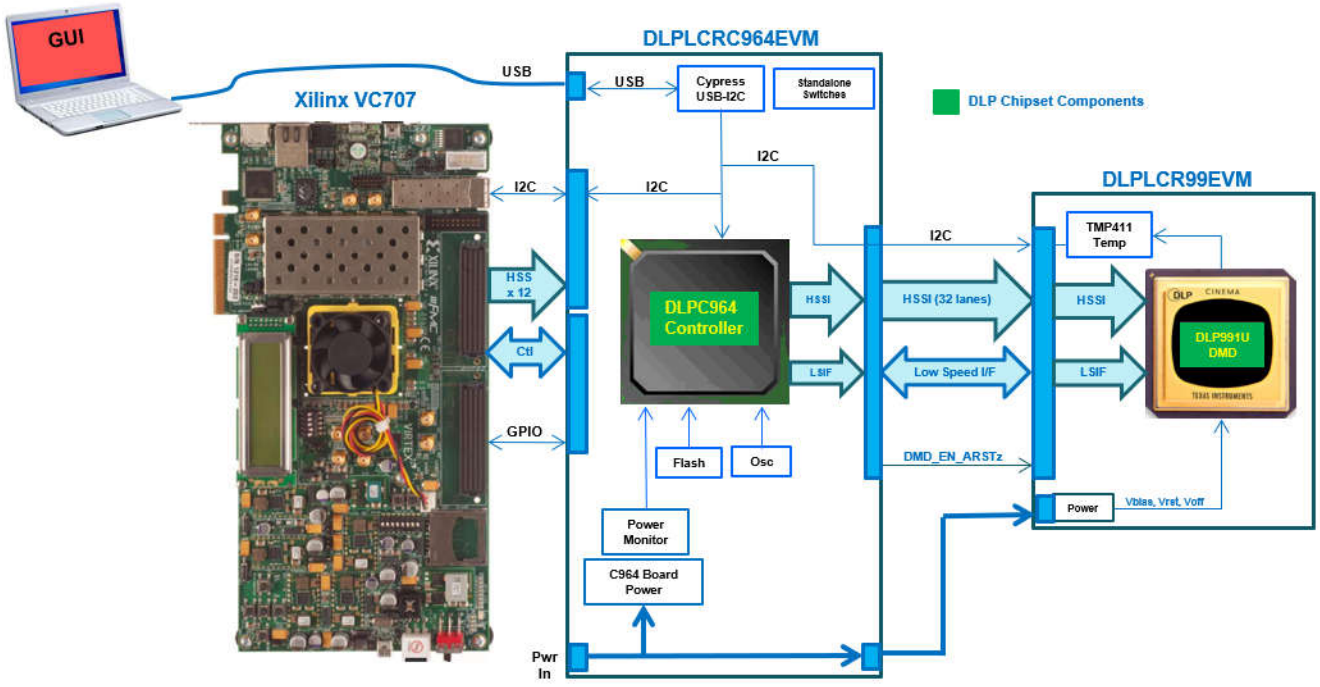



**图 1-1. Apps FPGA 硬件目标**

# 2 Apps FPGA 模块

本节详细介绍 DLPC964 Apps FPGA 中的各种模块。

## 2.1 Apps FPGA 方框图

图 2-1 展示了包含不同模块的 Apps FPGA 硬件方框图。每个模块都在将位平面数据传输至 DLPC964 控制器的过程中发挥着重要作用。DLPC964 接收来自外部前端源 (AMD Xilinx Virtex-7 VC-707) 的高速位平面数据，并对该数据进行格式转换，然后再加载到 DLPLCR99EVM 中，以便在 DLP991U DMD 上显示。

与 DLPC964 Apps FPGA 连接的主要模块是位平面图形发生器 (BPG)，有助于监测从 PGEN 加载到 DLPC964 控制器的位平面数据。块复位发生器 (BRG) 有助于在控制器不繁忙时启动发送到 DLPC964 的 PGEN 数据，繁忙状态由来自 DLPC964 控制器的 mcp_active 信号确定。

一旦准备好将数据加载到 PGEN 中，便会通过 HSSTOP 发送位平面数据，HSSTOP 是所有四个 GTX 通道 (gtx0 - gtx3) 的包装器。每个通道都有助于将位平面数据发送到 DLPC964 控制器，每个通道的速度高达 10Gbps。下面将进一步详细说明这些模块。

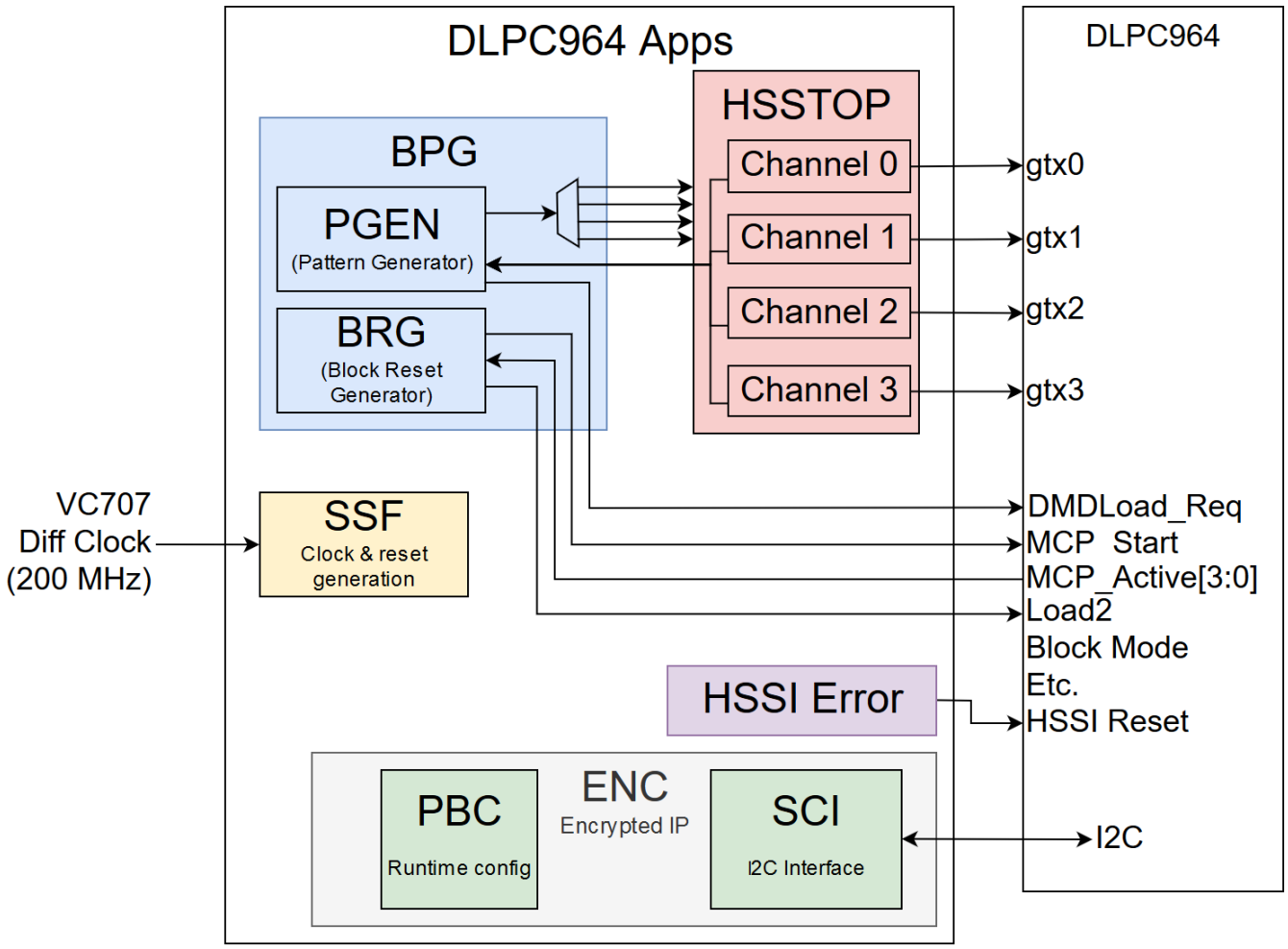



**图 2-1. Apps FPGA 硬件方框图**

## 2.2 BPG 模块

BPG（位平面图形发生器）是 DLPC964 Apps FPGA 的主要模块。此模块可用作示例来说明如何连接 DLPC964 和 Aurora 发送 IP，其中包含两个主要的块：

1. BRG（块复位发生器）
2. PGEN（图形发生器）

BPG 用作两个子块（BRG 和 PGEN）的包装器。BRG 子块负责启动 PGEN，并在 DLPC964 忙于加载数据时进行报告。

**备注**

如果正在将数据加载到 DLPC964 控制器中，BRG 会等待从 PGEN 加载更多数据，直到 mcp_active 信号变为低电平。一旦此信号变为低电平，mcp_start 信号将发送到 DLPC964 以指示可以将更多数据加载到控制器中。

PGEN 会报告向 Aurora GTX IP 发送数据的时间、要使用 mcp_start 信号从 DLPC964 复位的下一个块地址、发生的错误（超时或 DMD HSSI）和用户选择的设置。

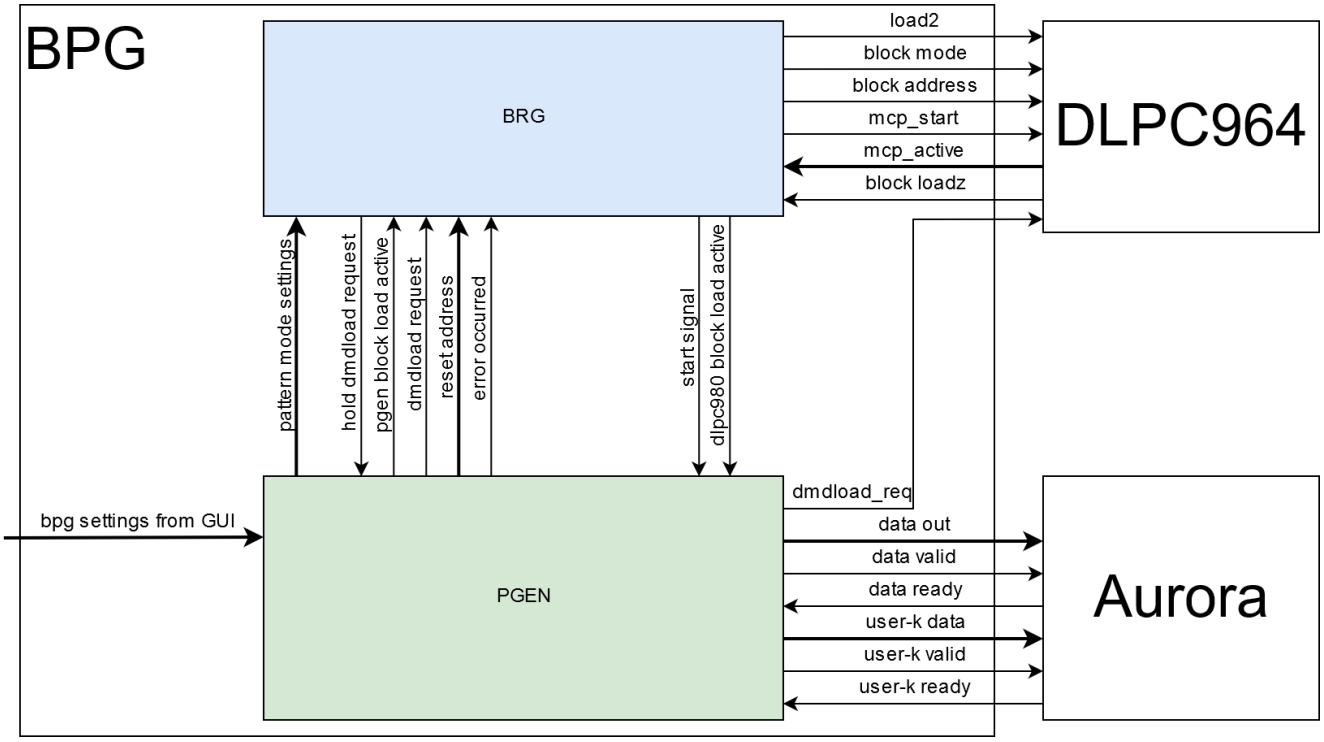



**图 2-2. BPG 模块硬件方框图**

## 2.3 BRG 模块

BRG（块复位发生器）模块是 BPG 的子模块。BRG 负责启动 PGEN（图形发生器）并与 DLPC964 进行连接。BRG 内有多个逻辑过程可帮助确定何时启动 PGEN 以及何时将另一个 MCP_Start 发送到 DLPC964 控制器。

为了使方框图保持简化，BRG 内的各种过程已表示为逻辑模块。每个逻辑模块如图 2-3 所示，并在下面进行了更详细的说明。

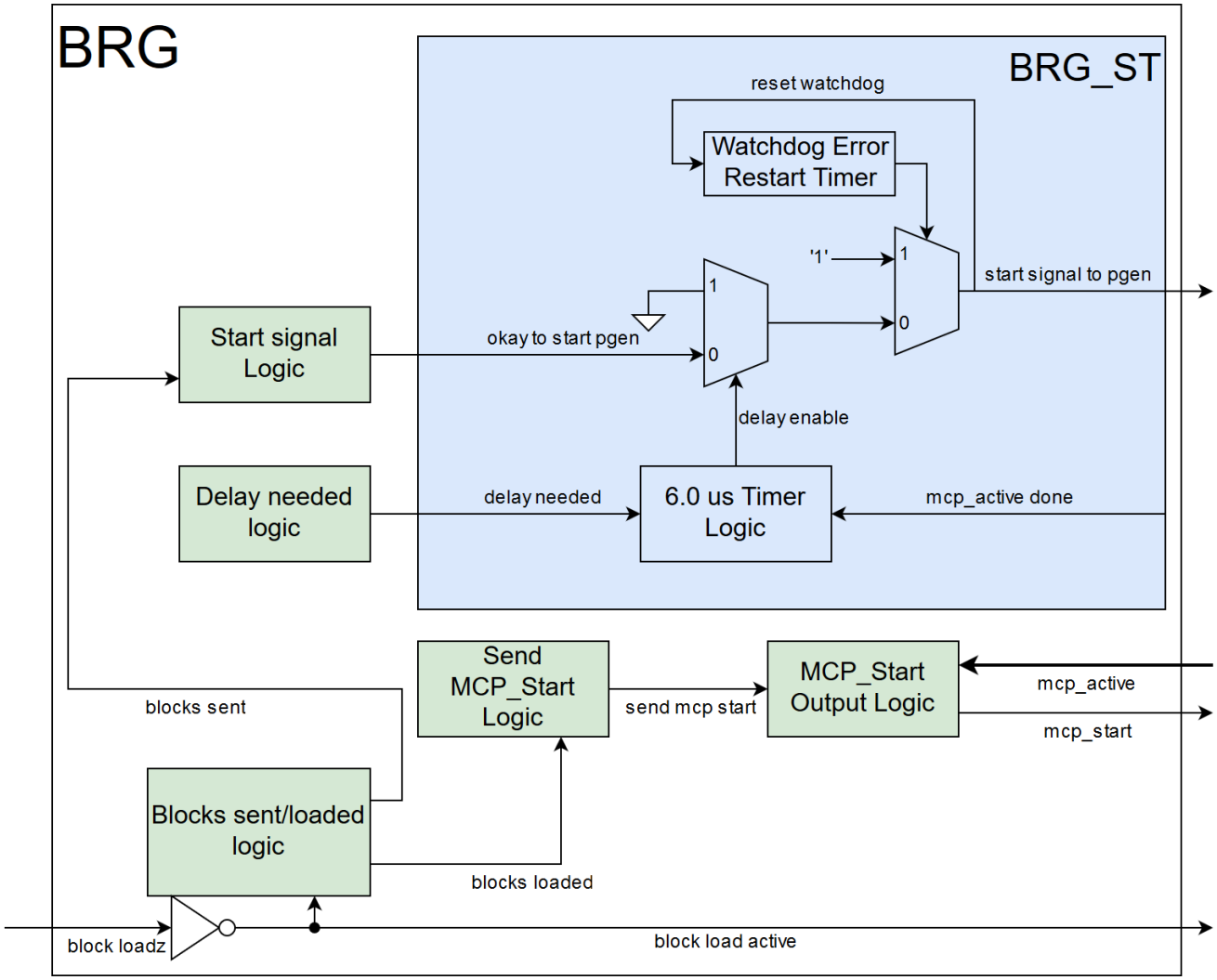



**图 2-3. BRG 模块硬件方框图**

### 2.3.1 启动信号逻辑

启动信号逻辑可处理要求 BRG 不启动 PGEN 的各种情况。

当符合以下任一情况时，不会发送另一个数据块：

- PGEN 已经在向 BRG 发送一个数据块。
- 当所有 mcp_active 信号都处于高电平时。来自 DLPC964 的这一信号使 DLPC964 Apps FPGA 知道 DLPC964 正在忙于复位 DMD 块并且需要在将另一个数据块发送到 DMD 块之前完成。
- 当 PGEN 已加载所有使能的块时。

PGEN 必须等待从 BRG 获取 mcp_start，然后用户才能将另一次数据传输发送到 DLPC964 控制器。

### 2.3.2 延时需求逻辑

延时需求逻辑负责处理可能发生的微镜稳定时间违例，并指示 BRG_ST 保持“启动信号逻辑”信号直至稳定时间到期。

微镜稳定时间适用于以下情况：

- 每当使用 mcp_start 信号对某个块进行复位时，因为更新 DMD 所需的时间会通过 mcp_active 信号传送到 DLPC964 Apps。一旦 mcp_active 信号变为低电平，该块中的微镜将设置为正确状态，但仍在趋稳到该状态。

将数据加载到正在趋稳的微镜中会导致 DMD 进入未知状态。为了避免此问题，延时逻辑中添加了微镜稳定时间，这种情况在全局模式（以及所有其他工作模式）下会遇到。这是因为所有块都加载了数据，并通过 mcp_start 信号同时全部发出。这意味着 DMD 上的所有微镜都需要时间来稳定，因此需要延迟启动信号。

**备注**

DLPC964 Apps 会处理基本的微镜稳定时间违例并相应延迟下一个负载。为了避免复杂的逻辑和大量的测试用例，只要任何模块被禁用，BRG 就会添加延时。

### 2.3.3 已发送/已加载的块数逻辑

发送到 DLPC964 的块数和 DLPC964 加载的块数，包括在启用慢速模式后跟踪发送的段数。

## 2.4 BRG_ST 模块

该模块是 BRG（块复位发生器）的子模块，负责保持 PGEN 启动信号以避免出现稳定时间问题。有关该模块的一个重要注意事项是请求所需的延时时使用的实际稳定时间延时。

启用所有块时，稳定时间延时设置为 6us。当 16 个块中的任何一个被禁用时，稳定时间延时设置为 6us。第一个稳定时间延时的用途是在全局模式下使用。由于在全局模式下所有块都会接收到复位信号，因此在微镜得到足够时间实现稳定之前，DLPC964 Apps 无法加载另一个块。第二个稳定时间用于避免许多潜在的稳定时间违例问题。

这种情况的一个示例是 DLPC964 Apps 仅启用两个块并设置为单模式的情况。时序图如下所示。

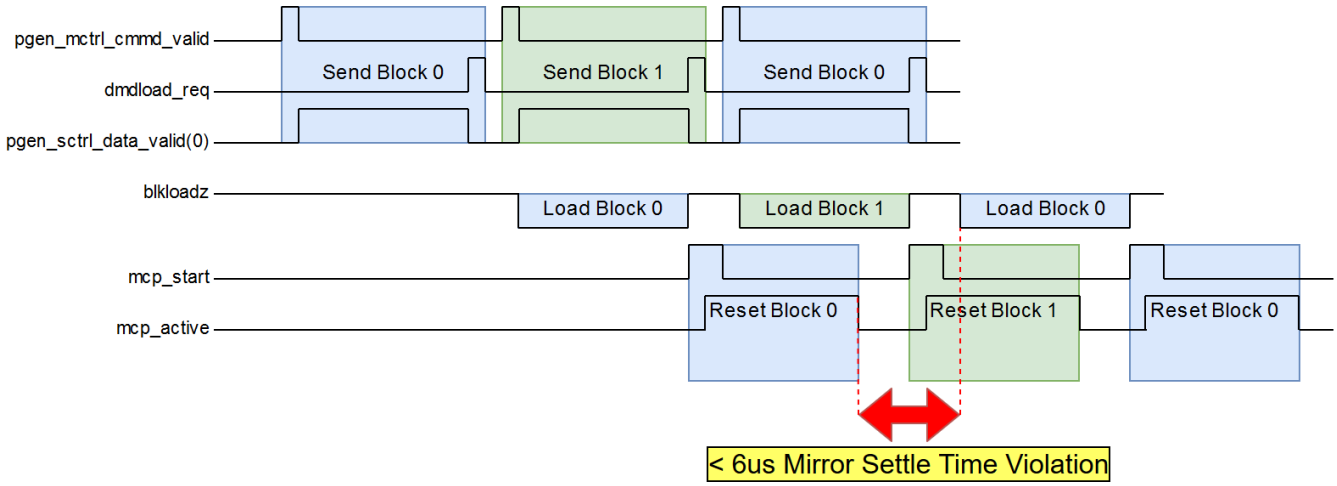



**图 2-4. BRG_ST 时序**

## 2.5 PGEN 模块

PGEN（图形发生器）模块是 BPG 的子模块。PGEN 负责报告数据发送到 Aurora GTX IP 的时间、要使用 mcp_start 信号从 DLPC964 复位的下一个块地址、发生的错误（超时或 DMD HSSI）和用户选择的设置。PGEN 模块如图 2-5 所示，下面将进行更详细的说明。

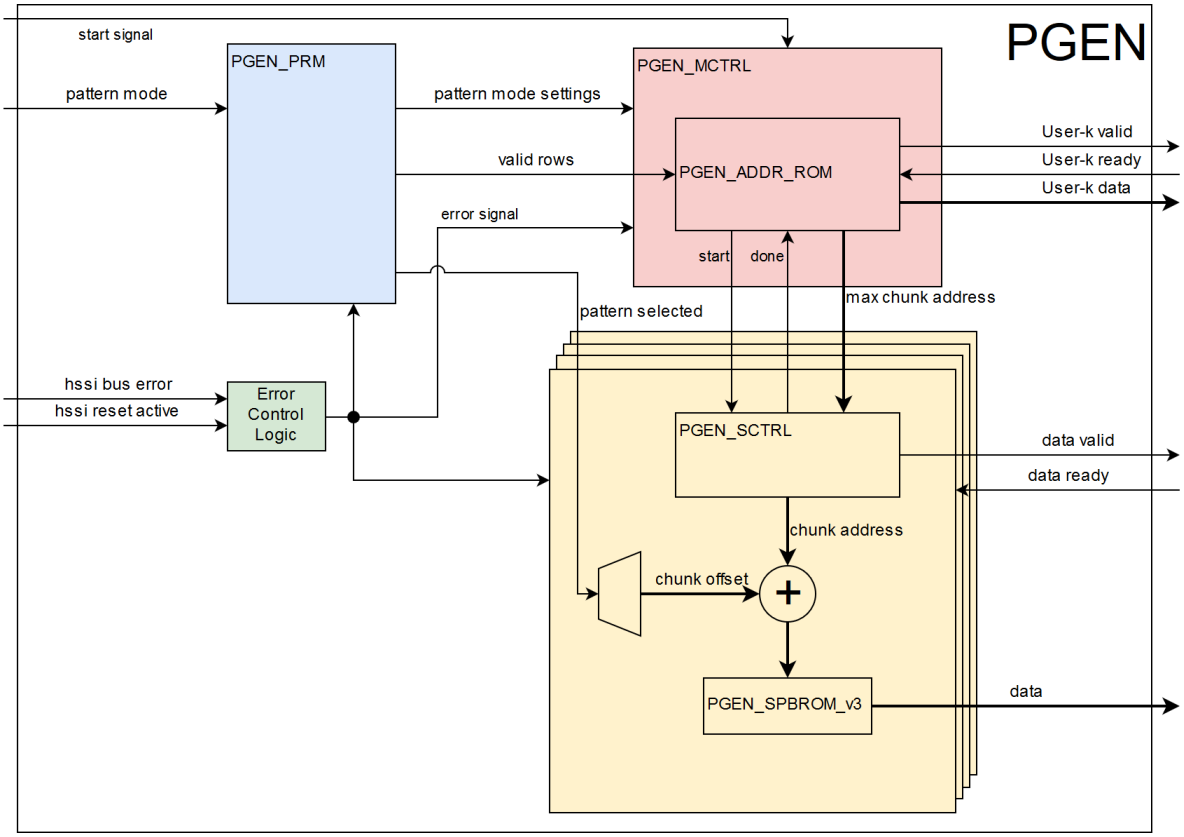



**图 2-5. PGEN 模块**

## 2.6 PGEN_MCTRL 模块

主控制模块由 BRG 的启动信号启动，此信号还控制辅助控制模块的四个副本。辅助控制模块负责 ROM 寻址并将位平面图像输出到 DLPC964 控制器。主控制模块的内核是一个通过等待 BRG 发送 mcp_start 信号来启动的有限状态机 (FSM)。一旦将信号发送到主模块，FSM 就会启动。图 2-6 展示了 PGEN_MCTRL FSM，其中每个状态机的定义如下：

- **CV_SEND_SIGNAL**：初始 FSM 状态。在接收到 BRG 启动信号时转换状态。根据所选的加载类型，FSM 进入 CV_WAIT_FOR_ACK 状态或 CV_SEND_DMDLD 状态。当块加载类型为清除 (001) 或置位 (001) 时，不需要命令有效信号，因为在这些加载类型期间不会发送数据。
- **CV_WAIT_FOR_ACK**：在 BRG 启动 FSM 之后，命令有效信号会发送到 Aurora user-k 接口。在此状态下，user-k 有效信号会保持高电平，直到 Aurora user-k 就绪信号确认 user-k 数据。一旦确认了 FSM，FSM 会将 user-k 有效信号置为无效，并进入下一个 FSM 状态。
- **CV_SEND_DMDLD**：发送命令后，DLPC964 Apps FPGA 可以开始发送位平面数据。此状态会启动并监控所有四个辅助控制模块。当所有四个辅助模块都报告自己已完成数据发送时，主控制模块就可以开始通过 Aurora user-k 接口发送 DMD 加载信号。
- **CV_HOLD_DMDLD**：主控制模块将保存 DMD 加载信号约 0.80ns，直到转换到 FSM 的开头。

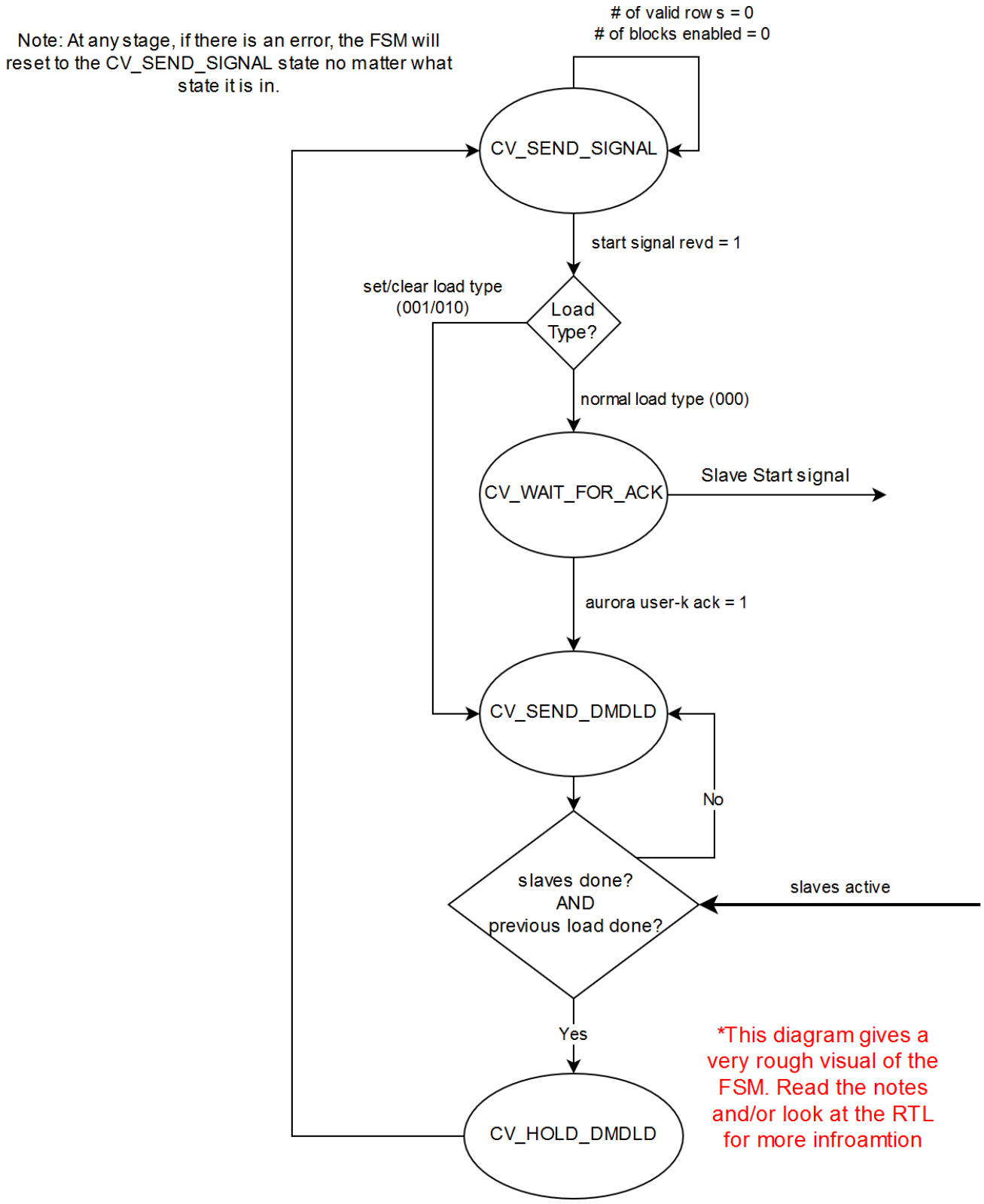



**图 2-6. PGEN_MCTRL FSM**

## 2.7 PGEN_SCTRL 模块

辅助控制模块有四个由主控制模块控制的副本。每个辅助模块负责发送正确长度的数据有效信号并使 ROM 地址递增。数据有效信号进入 Aurora 接口以将发送的数据标记为有效。Aurora 接口能够在不同的时间将就绪信号置为无效，因此辅助模块必须将这一点考虑在内，即需要保存这些值和有效信号直到就绪信号重新置为有效。主控制模块将最大 ROM 地址发送到辅助模块。辅助模块可递增计数至该值，从而允许 ROM 以用户指定的数量发出相应行数。图 2-7 展示了主要的辅助 FSM，其中每个状态机的定义如下：

- **IV_IDLE**：当不需要辅助模块时（例如，当选择的加载类型为清除或置位时），辅助模块将保持在这种空闲状态。如果需要数据（加载类型 = 正常），则 FSM 进入下一个状态。
- **IV_BEGIN**：等待主模块发送启动信号。收到之后，辅助模块进入下一状态。否则，FSM 会保持此状态，直到 FSM 接收到启动信号或加载类型发生变化。
- **IV_START**：FSM 会启动发送有效信号和 ROM 地址的过程。有效信号会延迟几个时钟周期以便使 ROM 输出与有效信号对齐。
- **IV_ACTIVE**：一旦启动辅助模块，输出 ROM 地址的过程将继续运行，直到辅助模块到达了主模块发送的 ROM 地址。一旦到达该值，FSM 就会根据指示进入起始状态。

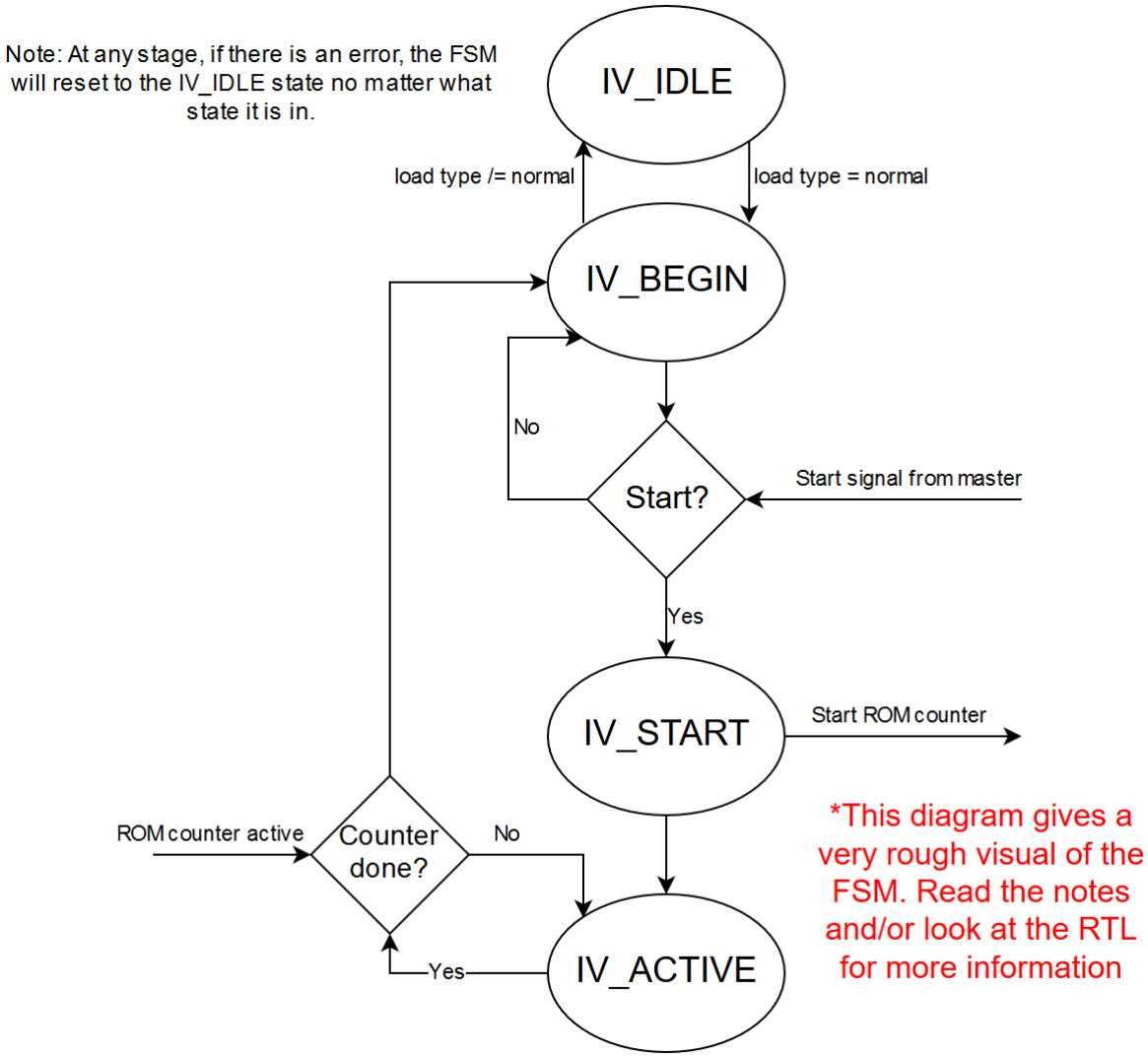



**图 2-7. PGEN_SCTRL FSM**

## 2.8 PGEN_PRM 模块

PGEN_PRM（图形发生器参数）模块负责循环遍历这些图形，发送 Aurora user-k 参数，并选择要加载/复位的下一个块。PGEN_PRM 不会测试和验证可能成千上万个测试场景，而是允许用户选择节 3.2.3 中所述的各种配置。

用户可以通过 I²C 接口来启用和禁用图形循环计时器。当禁用计时器，用户可以通过图形选择寄存器选择要显示的单个图形。节 3.2.3 中列出了可供用户使用的图形。每次 PGEN 发送图形之后，某些 user-k 参数（块地址和段号）都会发生变化。要加载和复位的块地址取决于用户启用的块 (pbc_bpg_blken)。

## 2.9 PGEN_ADDR_ROM

由于 DMD 中最小的可加载单位是一行，因此用户可以指定要加载的行数。然而，Aurora GTX 通道是 192 位宽，因此 PGEN_ADDR_ROM 使用以下公式对要加载到 ROM 地址的行数进行转换：

*MAX_ROM_ADDRESS = (# of lines) x 5 + CEIL(# of lines / 3)*

## 2.10 HSSTOP 模块

该模块包含的 Xilinx Aurora IP 可以将位平面数据发送至 DLPC964 控制器板。这种协议被称为 Aurora 64b/66b，如需更多信息，请参阅节 4.3。

如图 2-8 所示，HSSTOP 模块有一个用于全部四个 GTX 通道的 AURPRA_APPS_TX_X12LN 包装器。每个 Aurora GTX 通道均包含三条信道，每条信道的传输速率为 10Gbps。为帮助保持 GTX 信道同步，所有四个通道共用同一个 Aurora 时钟模块。

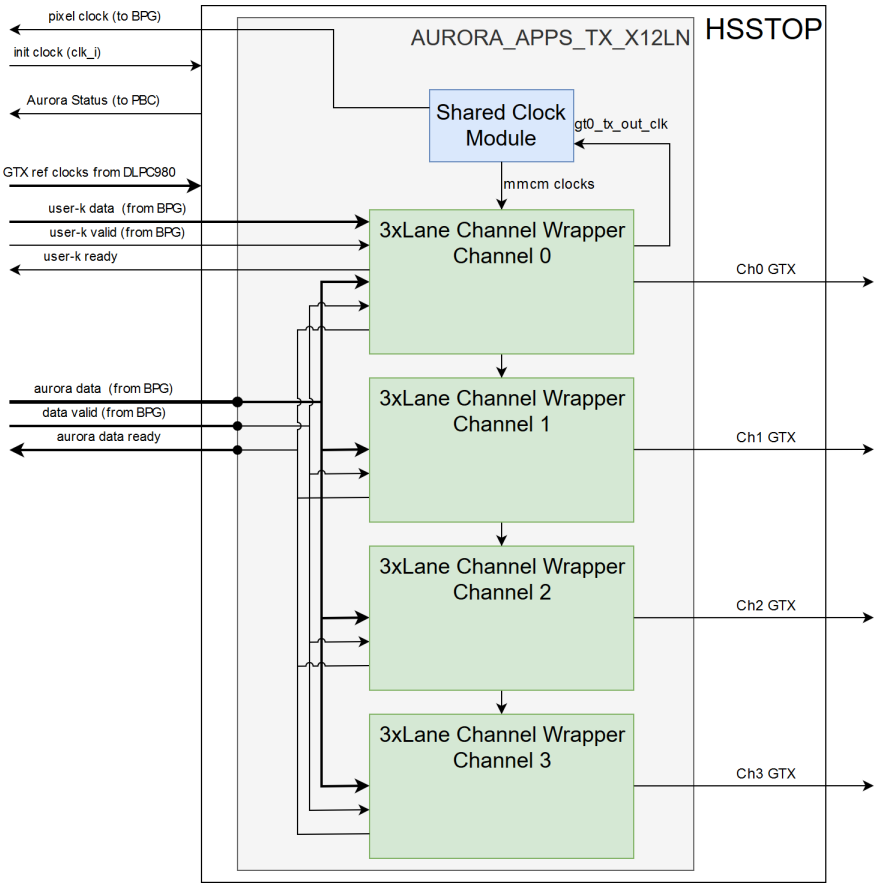



**图 2-8. HSSTOP 模块硬件方框图**

为帮助改善信号完整性，Aurora IP 允许差分信号具有预加重和后加重。DLPC964 Apps FPGA 设计中使用了以下设置。

| 信号名称 | 值 |
|---|---|
| gt_txpostcursor_in | 0.00dB (00000) |
| gt_txdiffctrl_in | 807mV (1000) |
| gt_txmaincursor_in | 0.00dB (00000) |
| gt_txprecursor_in | 0.00dB (00000) |

## 2.11 SSF 模块

SSF 负责创建 DLPC964 Apps FPGA 时钟并为整个系统进行复位。SSF 接收诸如按钮复位、DLPC964 初始化完成和 Aurora MMCM 锁定等异步信号，因此每个信号都可以与相应的时钟域同步。

## 2.12 ENC 模块

ENC（加密）模块其实就是不会向客户发布的 IP 的包装器。该模块中封装了 PBC 和 SCI 模块，旨在将所有加密的 IP 保存在相同的位置。

## 2.13 Xilinx IP

下面列出了 DLPC964 Apps FPGA 设计中使用的 Xilinx IP。

### 2.13.1 PGEN_SPBROM_v3

这是一个 192x192 ROM，用于保存要通过 Aurora 接口读取和发送的各种图形数据。可使用不同的用户模式对此 IP 进行重新编程。请参阅下方的节 3.2.5。

### 2.13.2 MAINPLL

此 PLL 生成设计中的三个主要时钟网络中的两个：一个是用于 Aurora 初始化时钟的 clk_i (100MHz) 时钟，另一个是用于配置寄存器和 $I^2C$ 逻辑的 clk_A (50MHz) 时钟。

### 2.13.3 AURORA_APPS_TX_X3LN_CLOCK_MODULE

时钟模块使用来自 GTX 通道 0 的基准输出时钟来生成用户时钟和同步时钟。生成的时钟将发送到全部四个 Aurora 通道，以确认所有四个通道都与同一个用户时钟对齐。每个通道可以稍有异相，这种情况下，BPG 架构会通过主/次架构来解决这一问题。

### 2.13.4 AURORA_APPS_TX_X3LN_CHANNEL_WRAPPER

一个通道包装器包含三条 GTX 信道和所有捆绑在一起的 Aurora IP 模块。有四个这样的副本形成四个通道。

## 2.14 参考文档

请参阅节 6 中列出的 Aurora 64B/66B v11.2 LogiCORE IP 产品指南。

## 2.15 DLPC964 Apps FPGA IO

| 信号名称 | 输入/输出 | 说明 |
|---|---|---|
| refclk_ui_p<br>refclk_ui_n | 输入 | 从 DLPC964 Apps FPGA 生成的固定 200MHz LVDS 基准时钟（参考自 VC-707：U51）。 |
| reset_ui | 输入 | 用于复位 DLPC964 Apps FPGA 的按钮（参考自 VC-707：SW7）。 |
| irqz | 输入 | 来自 DLPC964 控制器的 PBC 中断。 |
| running | 输出 | 转到 DLPC964 Apps FPGA 上的 LED0（参考自 VC-707 GPIO_LED_0）以在退出复位时发出信号。 |
| C964_init_done | 输入 | 来自 DLPC964 的输入，指示 DLPC964 Apps FPGA 退出复位。 |
| wdt_enablez | 输出 | 在运行中，看门狗计时器设置为“1” |
| rxlpmen | 输出 | 设置为 0 可实现低功耗模式均衡。更多信息，请参阅 Xilinx 应用手册。 |
| ext_hssi_rst | 输出 | 复位 DLPC964 HSSI 接口的信号。 |
| hssi_bus_err | 输入 | 从 DLPC964 指示在将最后一个块加载到 DLPC964 时存在同步错误。 |
| hssi_rst_act | 输入 | 从 DLPC964 向 Apps DLPC964 指示 HSSI 正在复位。 |
| load2 | 输出 | 在 DLPC964 初始化过程中用于在 load2 模式下设置 DMD。 |
| blkmode[1:0] | 输出 | 在 DLPC964 初始化过程中用于设置 DMD 超块模式。 |
| blkaddr[4:0] | 输出 | 发出的 mcp_start 发送到的块（或超块）地址。 |
| mcp_start | 输出 | 指示 DLPC964 加载任何发送到 DMD 的数据。 |
| mcp_active[3:0] | 输入 | 当 DMD 正在将数据加载到 DMD 时从 DLPC964 发出信号。一次只能进行 4 个加载。 |
| blkloadz | 输入 | 从 DLPC964 指示发送的块数据已加载完毕且已准备好发送至 DMD。 |
| dmdload_req | 输出 | 指示 DLPC964 将最近发送到控制器的块加载到 DMD 中。 |
| gtrx_ch0_refclk_p/n<br>gtrx_ch1_refclk_p/n<br>gtrx_ch2_refclk_p/n<br>gtrx_ch3_refclk_p/n | 输入 | DLPC964 为每个 Aurora 发送通道（GTX 通道 0 - 3）提供的基准时钟。 |
| ch0_gtx_p/n[2:0] | 输出 | Aurora 10Gbps 发送通道 0。<br>user-k 数据仅与数据一起通过通道 0 发送。<br>启用慢速模式 (pbc_bpg_normal_mode_en = 0) 时，通道 0 是唯一发送数据的通道。 |
| ch1_gtx_p/n[2:0] | 输出 | Aurora 10Gbps 发送通道 1。 |
| ch2_gtx_p/n[2:0] | 输出 | Aurora 10Gbps 发送通道 2。 |
| ch3_gtx_p/n[2:0] | 输出 | Aurora 10Gbps 发送通道 3。 |
| i2c_sda | INOUT | 与 DLPC964 共享的 $I^2C$ 数据线路。 |
| i2c_scl | INOUT | 与 DLPC964 共享的 $I^2C$ 时钟线路。 |
| fmc_gpio[6:0] | INOUT | DLPC964 Apps FPGA 和 DLPC964 之间的 GPIO。 |
| led | 输出 | 转到 DLPC964 Apps FPGA 上的 LED1（参考自 VC-707 GPIO_LED_1）以在启用 BPG 时发出信号。 |
| testmux_uo[15:0] | INOUT | 适用于 DLPC964 Apps FPGA 的调试多路复用器。 |

## 2.16 关键定义

- **块**：一个块是 DMD 的一个包含 136 行 x 4096 像素的部分。DMD 分为 16 个这样的块，因此 DMD 的总图像尺寸为 2176 行 x 4096 像素。在 DLPC964 Apps FPGA 中可以对这些块单独寻址 (0x0 - 0xF)。
- **段**：每个块有四个段（A、B、C 和 D）。每个段包含 136 行 x 1024 像素。在正常运行中，所有 4 个段同时载入。在慢速模式下，每个段单独载入。
- **组**：在双通道模式或四通道模式下，块将分组进行更新。下表列出了这些组。请注意，在双通道模式或四通道模式下，用户必须启用一个组中的所有块或禁用一个组中的所有块，这一点非常重要。

| 块载入地址 | 0x0 | 0x1 | 0x2 | 0x3 | 0x4 | 0x5 | 0x6 | 0x7 | 0x8 | 0x9 | 0xA | 0xB | 0xC | 0xD | 0xE | 0xF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 双通道模式组 | 0x0 | | 0x2 | | 0x4 | | 0x6 | | 0x8 | | 0xA | | 0xC | | 0xE | |
| 四通道模式组 | 0x0 | | | | 0x4 | | | | 0x8 | | | | 0xC | | | |

下面是在 x4 模式和全局模式下发送块的简单时序示例。

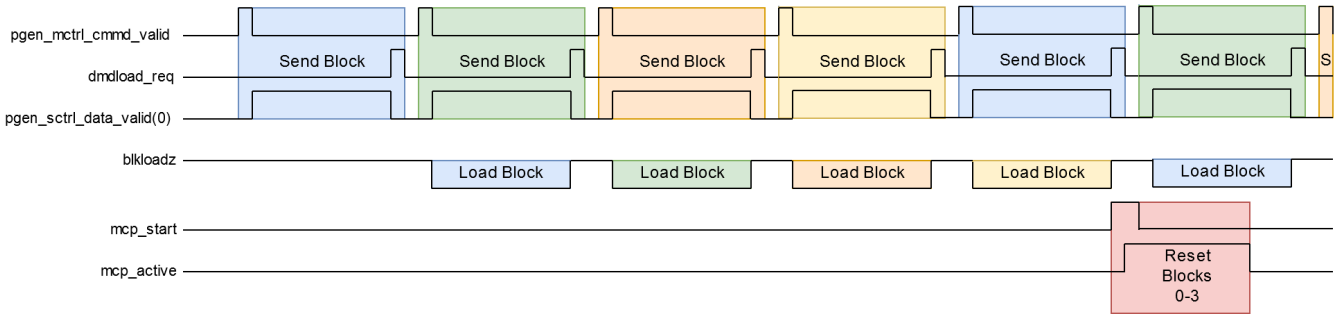



**图 2-9. x4 模式**

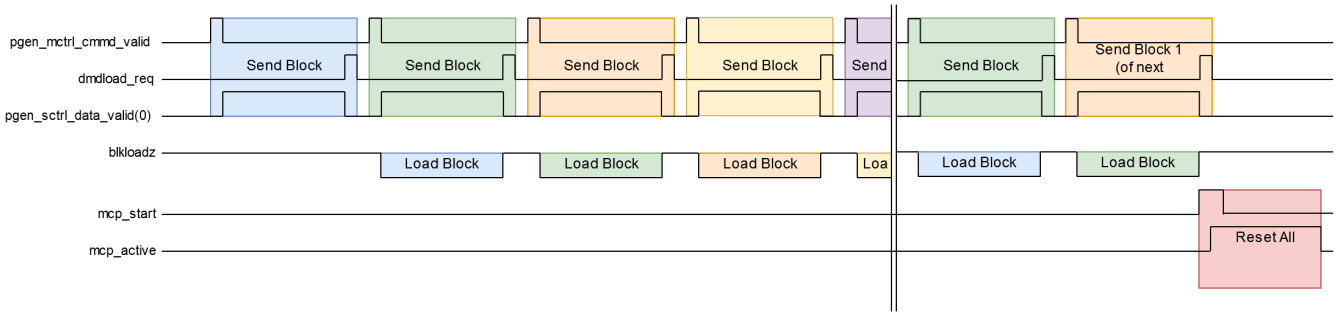



**图 2-10. 全局模式**

- **DLPC964 Apps**：载入了 Apps FPGA 位流的 Xilinx VC707。
- **DLPC964**：与 DMD 连接的 DLPC964 控制器。
- **模式**：有 4 种模式：单通道 (0x0) 模式、双通道或 x2 (0x1) 模式、四通道或 x4 (0x2) 模式和全局 (0x3) 模式。下面和节 3 中详细介绍了这些模式。
- **单通道 (0x0) 模式**：在单通道模式下，每个块中加载数据，一旦 DLPC964 完成将单个块加载到 DMD 的过程，DMD 就会通过 MCP_Start 信号进行更新。由于每个块可单独更新，因此有效的块模式地址为 0x0 - 0xF。
- **双通道 (0x1) 模式**：双通道模式意味着，一旦 DLPC964 在一个组的 2 个块中加载了数据，DMD 就会使用单个 MCP_Start 信号更新这两个块。由于一次更新 2 个块，因此有效的块模式地址为 0x0、0x2、0x4、0x6、0x8、0xA、0xC 和 0xE。

**备注**

请注意，在双通道模式中启用和禁用块时，必须启用或禁用组中的所有块。

- **四通道 (0x2) 模式**：四通道模式意味着，一旦 DLPC964 在一个组的 4 个块中加载了数据，DMD 就会使用单个 MCP_Start 信号更新所有这 4 个块。由于一次更新 4 个块，因此有效的块模式地址为 0x0、0x4、0x8 和 0xC。
- **全局 (0x3) 模式**：这是默认启动模式。在全局模式下，所有已启用的块都会加载，一旦 DLPC964 完成，MCP_Start 信号就会一次性更新所有块。由于已经一次性更新所有块，因此下一个负载操作需要等待，直至微镜达到稳定状态。

**备注**

这一时间称为微镜稳定时间，必须约为 8us。

- **块加载类型**：使用 Aurora GTX 接口中的 user-k 在数据之前先发送块加载类型。DMD 支持 3 种不同类型的加载：正常 (0x0)、清除 (0x1) 和置位 (0x2)。
- **正常 (0x0)**：这是默认的块加载类型。正常加载类型指示 DLPC964 在 DMD 中加载出现在 user-k 数据之后的任何数据。
- **清除 (0x1)**：清除加载类型不发送任何数据。这是因为当 DLPC964 接收到清除加载类型时，DLPC964 会将指定块中的微镜设置为关闭状态 (0)。
- **置位 (0x2)**：置位加载类型不发送任何数据。这是因为当 DLPC964 接收到置位加载类型时，DLPC964 会将指定块中的镜像设置为开启状态 (1)。
- **MCP_Start**：MCP_Start ( 微镜时钟脉冲启动 ) 指示 DMD 以发送的任何数据更新微镜。DLPC964 根据所选模式和块模式地址确定要更新的块。
- **像素行/行**：像素行是指 4096 个像素的水平行。这可以看作是 DMD 上的 y 位置。
- **像素列/列**：像素列是指 2176 个像素的垂直列。这可以看作是 DMD 上的 x 位置。
- **快速/慢速模式**：默认情况下启用快速模式。快速模式在 4 个 Aurora GTX 通道上并行发送一个块的所有 4 个段。慢速模式仅使用第一个 GTX 通道并按顺序发送段。更多有关段排序的详细信息，请参阅“图形模式”部分。
- **Load2 模式**：默认情况下禁用 Load2 模式。在 Load2 模式下，DLPC964 Apps 仅发送指定行数的一半 ( 请求 136 行时，仅发送 68 行 )。这是因为在该模式下，DMD 每 2 行加载一次相同的数据。

# 3 功能配置

启动时，DLPC964 Apps FPGA 处于全局复位模式，并会更改每 2 秒发送至 DLP DMD EVM 的图形。如果用户想要更改图形模式，则应按照下文节 3.2.4 中列出的步骤操作。

## 3.1 启用的块数

DLPC964 Apps FPGA 在默认启动时会启用所有 16 个 DMD 块。用于禁用/启用 16 个块的寄存器是一个 16 位可配置寄存器。为了避免代码复杂性 DMD 时序违例问题，如果禁用单个块，则后续的每次 DMD 加载都将延迟。TI 建议在修改启用的块数之前禁用 BPG。

**备注**

在双通道或四通道复位模式下，复位组中的所有块都会启用或全部禁用。例如，在四通道复位模式下，仅启用块 0-3 和 8-11 是有效的，但启用块 1-4 和 9-12 是无效的。

## 3.2 图形循环启用

DLPC964 Apps FPGA 在默认启动时大约每 2 秒会在前八个预定义图形之间循环。用户可切换该设置。禁用此设置时会将单个选定的图形发送到 DLPC964。节 3.2.2 说明了可供用户在 DLP DMD EVM 上显示的图形。

### 3.2.1 南/北翻转

启用南/北翻转可让 DLPC964 垂直翻转发送的图像。

### 3.2.2 TPG 图形

启用图形循环后，图形 1-8 会通过 DLPC964 控制器进行循环。图形 9-14 不会循环，但可以由客户进行选择。

**表 3-1. TPG 图形**

| 图形编号 | 值 | 名称 | 说明 |
|---|---|---|---|
| 1 | 0x0 | 全开 | 全白背景，DMD 上的所有微镜均处于打开位置。 |
| 2 | 0x1 | 全关 | 全黑背景，DMD 上的所有微镜均处于关闭位置。 |
| 3 | 0x2 | 棋盘格 | 黑白色棋盘格图形，其中的方格为 64 像素（长）x 68 行（高）。图形高度和宽度选择为可重复的 136 x 1024 图像。 |
| 4 | 0x3 | 带边框的单像素网格 | 每个 136 x 1024 区域周围都有一个单像素边框，每个区域内都刻有网格图形。垂直线的间隔为 32 像素，水平线的间隔为 34 行。 |
| 5 | 0x4 | 自西向东的对角线 | 对角线自西向东跨越每个段。 |
| 6 | 0x5 | 自东向西的对角线 | 对角线自东向西跨越每个段。 |
| 7 | 0x6 | 水平线 | 16 行宽的水平线。 |
| 8 | 0x7 | 垂直线 | 16 像素宽的垂直线。 |
| 9 | 0x8 | Load2 棋盘格 | **调试图形**<br>黑白色棋盘格图形，其中的方格为 32 像素（长）x 34 行（高）。该图形在 0-67 行之间延续。68-135 行全部为黑色。这是为了方便说明 load2 操作是如何进行的。 |
| 10 | 0x9 | 10 点 x 10 点 | **调试图形**<br>客户请求的图形，白色单像素在 X 和 Y 方向均匀间隔 8 像素。 |
| 11 | 0xA | 反转棋盘格 | **调试图形**<br>这是棋盘格图形 (0x2) 的反转版本。当用户选择该图形 (0xA) 时，图形计时器寄存器会使 BPG 在该图形和原始棋盘格图形 (0x2) 之间翻转。这有助于解决铰链记忆问题，因此必须在光源关闭时使用。 |
| 12 | 0xB | 随机噪声图形 | **调试图形**<br>用于客户倾斜角测试的随机噪声图形。 |
| 13 | 0xC | 1x1 水平线 | **调试图形**<br>每隔一行黑/白交替，可用于检查行加载是否有问题。 |
| 14 | 0xD | 1x1 垂直线 | **调试图形**<br>每隔一列黑/白交替，可用于检查数据总线是否有问题。 |
| 15 | 0xE | 全开/全关 | **调试图形**<br>选择此图形会使 BPG 根据图形计时器值在全开 (0x0) 和全关 (0x1) 图形之间切换。 |

### 3.2.3 图形模式

**备注**

更改图形模式时，请执行下面节 3.2.4 中的步骤

图形模式寄存器允许用户尝试各种 DLPC964 运行模式。下表介绍了所有可用的图形模式：

| 模式编号 | 值 | 名称 | 设置 | 注意 |
|---|---|---|---|---|
| 1 | 0x0 | 全局模式 | • 全局复位模式 (0x3)<br>• 正常加载类型 (0x0)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 在全局复位模式下，所有启用的块按顺序加载数据。加载所有块后，MCP_Start 信号会同时复位所有块。 |
| 2 | 0x1 | 四通道模式 | • 四通道复位模式 (0x2)<br>• 正常加载类型 (0x0)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 在四通道复位模式下，4 个块按顺序加载。加载一个组中的 4 个块后，MCP_Start 信号会同时对该组中的 4 个块发出复位命令。<br>**备注**<br>在四通道复位模式下有 4 个块“组”。块 0-3、4-7、8-11 和 12-15。一个组中的所有块必须同时启用或禁用。 |
| 3 | 0x2 | 双通道模式 | • 双通道复位模式 (0x1)<br>• 正常加载类型 (0x0)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 在双通道复位模式下，2 个块按顺序加载。加载一个组中的 2 个块后，MCP_Start 信号会同时对该组中的 2 个块发出复位命令。<br>**备注**<br>在双通道复位模式下有 8 个块“组”。块 0-1、2-3、4-5、6-7、8-9、10-11、12-13、14-15。一个组中的所有块必须同时启用或禁用。 |
| 4 | 0x3 | 单通道模式 | • 单通道复位模式 (0x0)<br>• 正常加载类型 (0x0)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 在单通道复位模式下，每次加载一个块，一旦 DLPC964 将发送的数据加载到 DMD 中，MCP_Start 信号就会复位这一个块。 |
| 5 | 0x4 | 全局清除模式 | • 全局复位模式 (0x3)<br>• 清除加载类型 (0x1)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 此模式显示 DLPC964 系统中如何使用清除块加载类型。<br>清除加载类型不需要任何数据，因为相应的块会将所有微镜置于关闭状态 (0)。由于清除加载类型后续不会发送任何数据，因此无需发送命令有效信号，而仅发送 DMD 加载信号。<br>MCP_Start 信号遵循与全局模式相同的图形。 |
| 6 | 0x5 | 全局置位模式 | • 全局复位模式 (0x3)<br>• 置位加载类型 (0x2)<br>• 禁用 Load2 (0x0)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 此模式显示 DLPC964 系统中如何使用置位块加载类型。<br>置位加载类型与清除加载类型的作用刚好相反，也不需要任何数据。置位加载类型会将所有微镜设置为开启状态 (1)。与清除加载类型一样，无需命令有效信号，只需 DMD 加载信号。<br>MCP_Start 信号遵循与全局模式相同的图形。 |

| 模式编号 | 值 | 名称 | 设置 | 注意 |
|---|---|---|---|---|
| 7 | 0x6 | 全局 Load2 模式 | • 全局复位模式 (0x3)<br>• 正常加载类型 (0x0)<br>• 启用 Load2 (0x1)<br>• 启用快速模式 (0x1)<br>• 加载的总行数 (136 = 0x88) | 启用 Load2 操作会指示 DMD 将接收到的 1 行数据加载到 DMD 的 2 行中。<br>DLPC964 Apps FPGA 在 Load2 操作期间的作用是确保通过 Aurora HSS 通道发送最多 68 行，并确保在 user-k 控制参数中启用的行数也减半。 |
| 8 | 0x7 | 单通道慢速模式 | • 单通道复位模式 (0x0)<br>• 正常加载类型 (0x0)<br>• 禁用 Load2 (0x0)<br>• 启用慢速模式 (0x0)<br>• 加载的总行数 (136 = 0x88) | 慢速模式（或禁用快速模式）会使 DLPC964 Apps FPGA 仅通过单个通道（4 个 10Gbps 信道，而不是 12 个）发送数据。<br>为此，必须在 1 个通道上按顺序发送块的每个段，而不是并行发送。这些段必须按以下顺序发送：D (0x3) → C (0x2) → B (0x1) → A (0x0)。<br>一旦发送了全部 4 个段，即可发出 MCP_Start 信号。<br>MCP_Start 信号的行为与单通道模式下相同。 |

### 3.2.4 切换模式

按照以下说明了解在图形模式之间切换的正确方法（例如，从全局复位模式切换到双复位模式）。

1. 关闭 BPG。
2. 将 DLPC964 Apps FPGA 更改为所需的图形模式。
3. 复位 DLPC964 并等待 DLPC964 退出复位。
4. 开启 BPG。

**备注**

按照这些步骤操作可以验证硬件是否进入未知状态。

### 3.2.5 更改 BPG 图形

用户可根据以下说明更改 ROM 中用于 TPG 选择的默认图形。

1. 转至目录 C:\Texas Instruments\DLPC964-Apps\docs\patterns，验证用户是否安装了 Python 2.6 或更高版本。
2. 打开 binary_to_coe.py 文件并通读顶部注释信息。DLPC964 Apps 中使用的图形是使用此脚本生成的。查看靠近脚本顶部的 `bit_fnames` 列表。

```
#       RTL-defined pattern (full-on)           # Pattern 1   (0x0)
#       RTL-defined pattern (full-off)          # Pattern 2   (0x1)
bit_fnames = ["chkrbrd_136x1024.txt"          , # Pattern 3   (0x2)
              "grid_136x1024.txt"             , # Pattern 4   (0x3)
              "diag_e2w_136x1024.txt"         , # Pattern 5   (0x4)
              "diag_w2e_136x1024.txt"         , # Pattern 6   (0x5)
              "horiz_136x1024.txt"            , # Pattern 7   (0x6)
#       RTL-defined pattern (veritcal lines)    # Pattern 8   (0x7)
              "load2chkrbrd_136x1024.txt"     , # Pattern 9   (0x8)
              "invchkrbrd_136x1024.txt"       , # Pattern 10  (0x9)
              "dots8by8_136x1024.txt"         , # Pattern 11  (0xA)
              "rand_136x1024.txt"             , # Pattern 12  (0xB)
              "horiz1x1_136x1024.txt"         , # Pattern 13  (0xC)
              "vert1x1_136x1024.txt"          ] # Pattern 14  (0xD)
#       TBD                                     # Pattern 15  (0xE)
#       Full-on/off toggle                      # Pattern 16  (0xF)
```

3. 用户可以创建一个新的 .txt 文件，并替换 `bit_fnames` 列表的其中一个名称。

**备注**

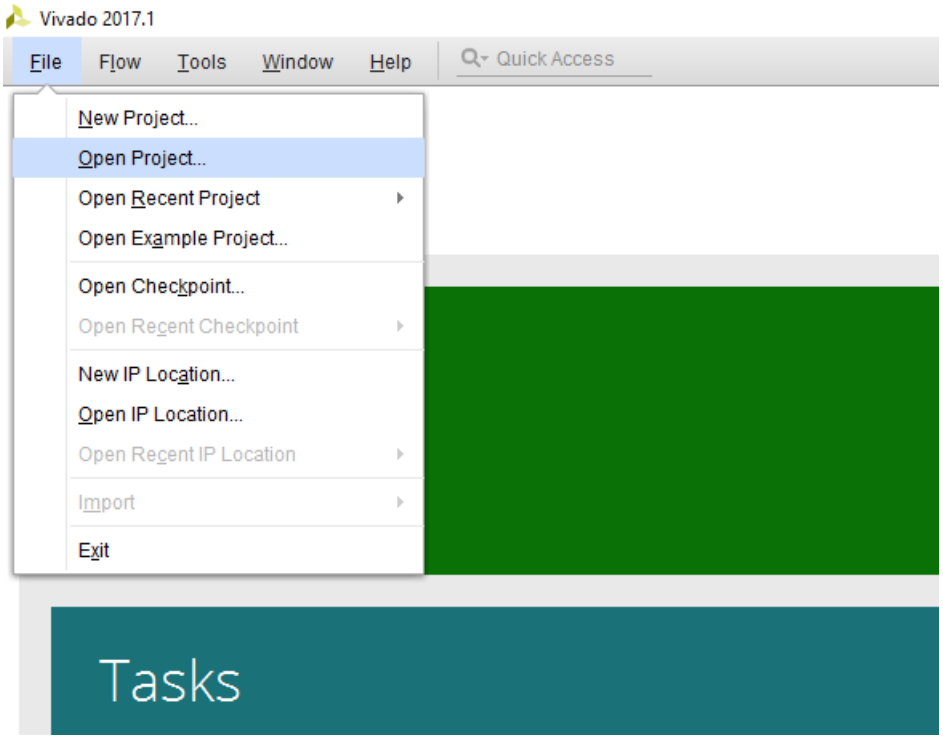
如果有任何图形被指定为 RTL 定义的图形，则无法将这些图形更改为其他图形，因为这些图形不是从 ROM 中读取的。

   a. 有关为 python 脚本创建 .txt 文件的说明：
      i. 此文本文件必须有 1024 列和 136 行。
      ii. 此文本文件中的每个字符必须为“1”或“0”。
      iii. 确保此文本文件与 python 脚本位于同一目录中。
4. 以新的文本文件名更新 `bit_fnames` 后，运行 python 脚本。此时会创建一个名为 `bpg_patterns.coe` 的文件。

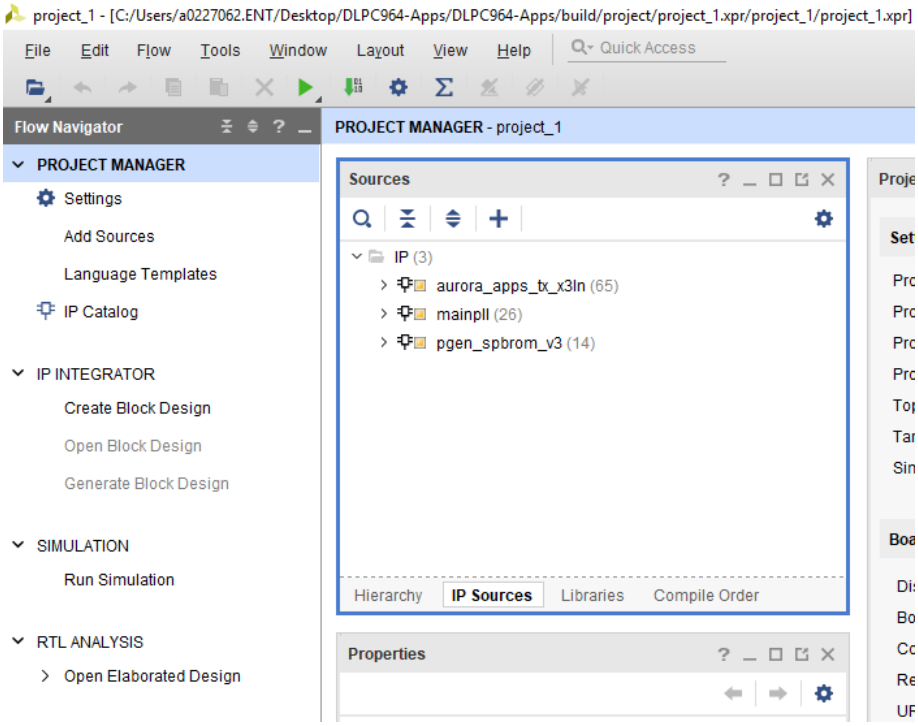
5. 打开 Vivado 工程（方法是将已存档的工程解压缩到 build\project 目录中，或运行 run.tcl 脚本）。

**备注**

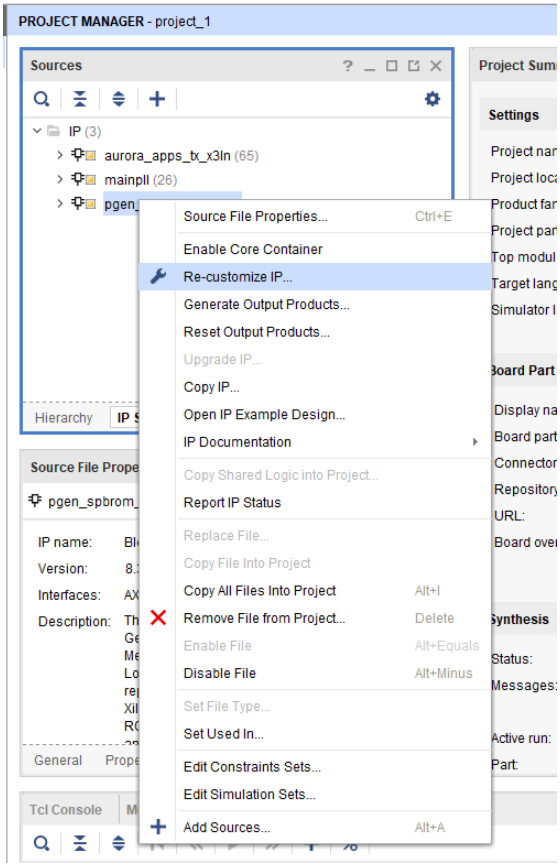
解压缩工程的速度更快，但如果需要有关运行方法的说明，可以查看 run.tcl 脚本中提供的这些说明。



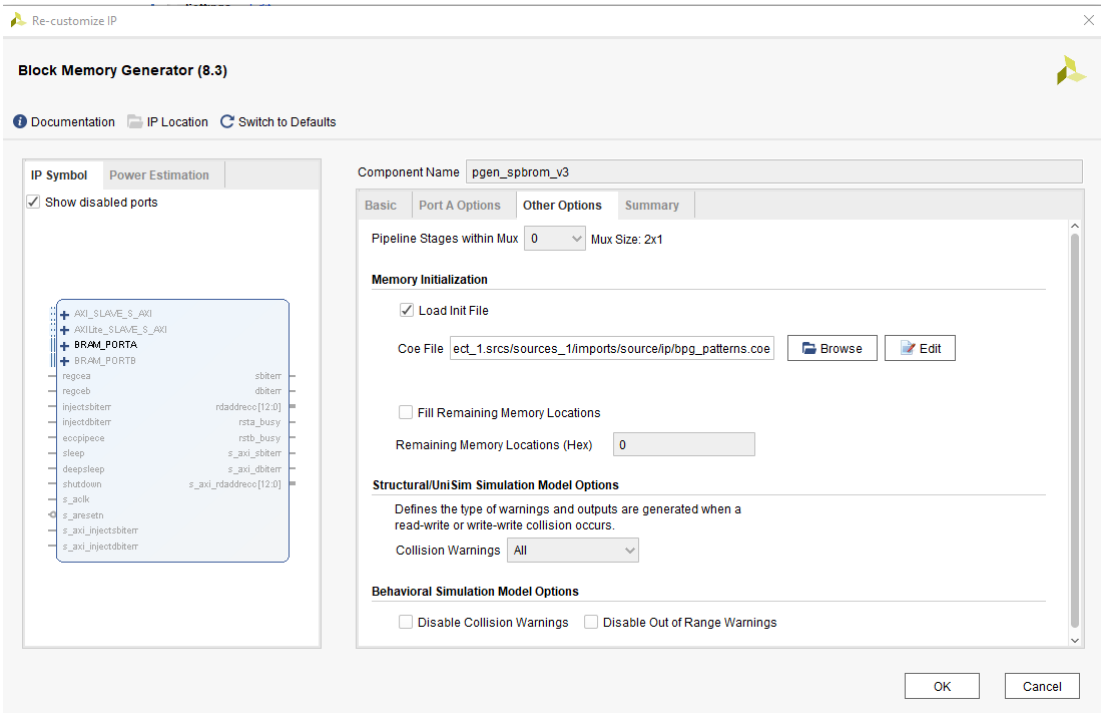

6. 打开工程后，在“Project Manager”窗口中找到标记为 IP 源的选项卡并进行点击。



7. 右键点击 pgen_spbrom_v3 并选择“Re-customize IP”。

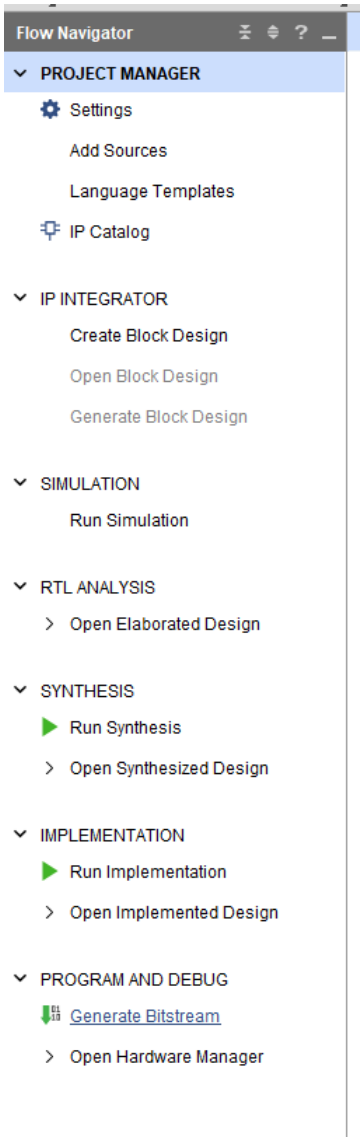



8. IP 配置工具打开后，转到“Options”选项卡，用户将看到 *Memory Initialization* 部分。




9. 点击 *Browse*，然后导航到步骤 4 中通过 python 脚本创建的 bpg_patterns.coe 文件的位置。假设没有错误，点击 *OK*。在下一个窗口中，点击 *Generate*。
10. 用户现在已对 DLPC964 Apps FPGA 中的 ROM 进行重新编程。现在，重新构建工程。

11. Xilinx 完全生成输出产品后，点击 Flow Navigator 左侧的 *Generate Bitstream*。在出现任何提示时点击 *OK*，一旦 Vivado 完成，就可以在 project_1\project_1.runs\impl_1\ 目录中找到位流。



# 4 附录

## 4.1 Vivado Chipscope 捕获结果

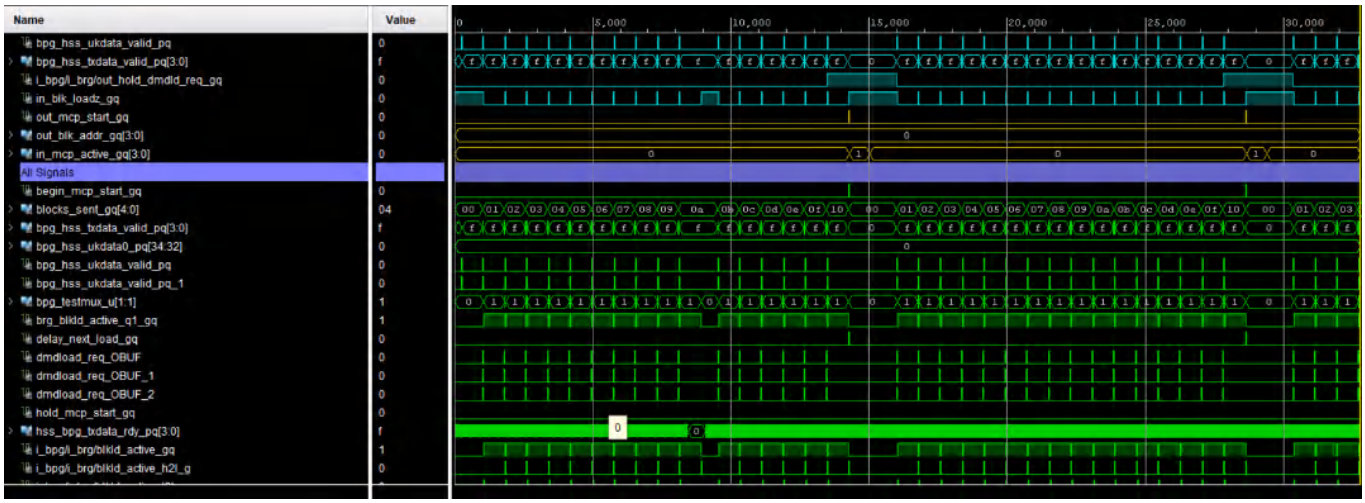



图 4-1. 图形模式 0 捕获结果

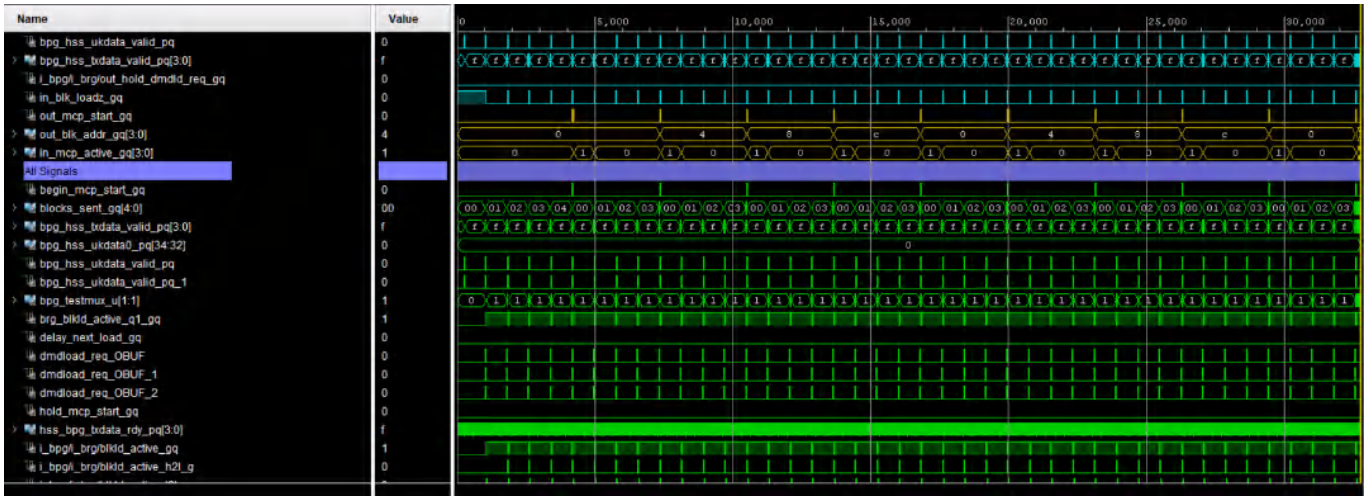



图 4-2. 图形模式 1 捕获结果

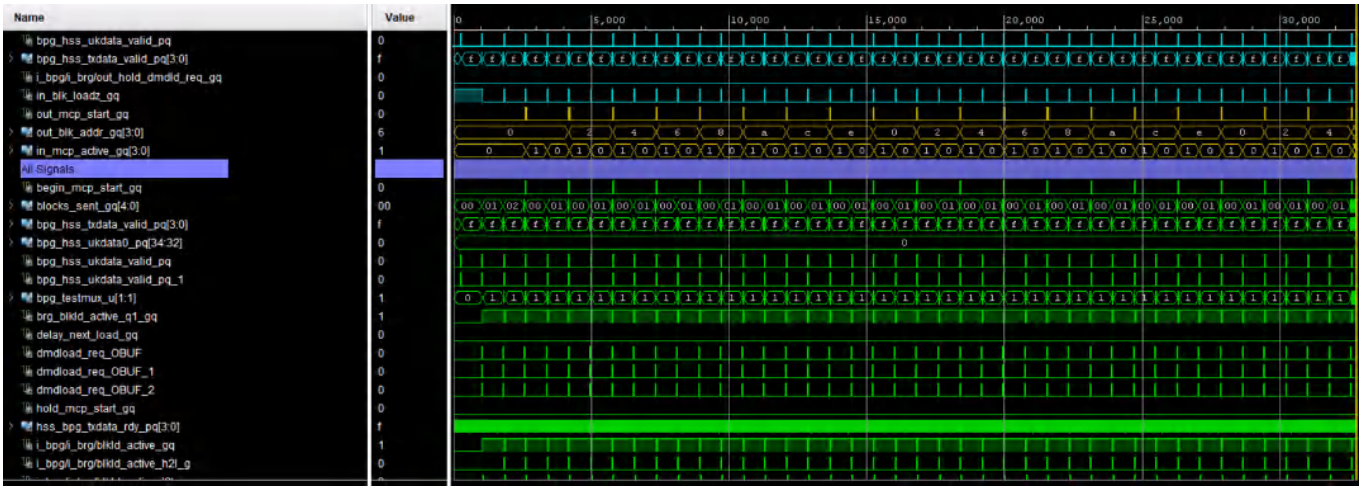



图 4-3. 图形模式 2 捕获结果

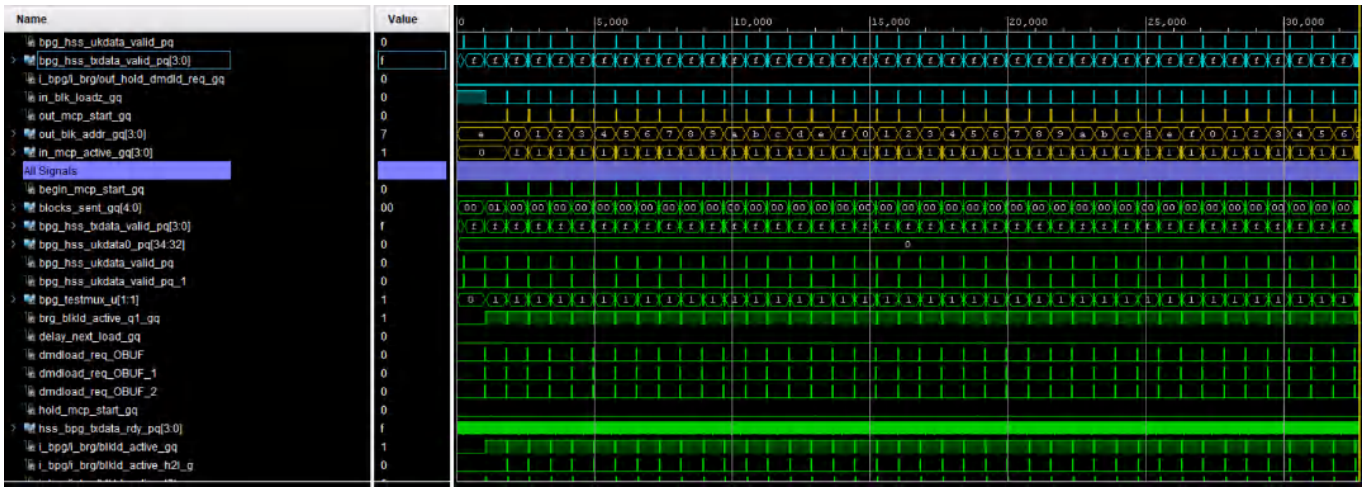



图 4-4. 图形模式 3 捕获结果


图 4-5. 图形模式 4 捕获结果


图 4-6. 图形模式 5 捕获结果


图 4-7. 图形模式 6 捕获结果

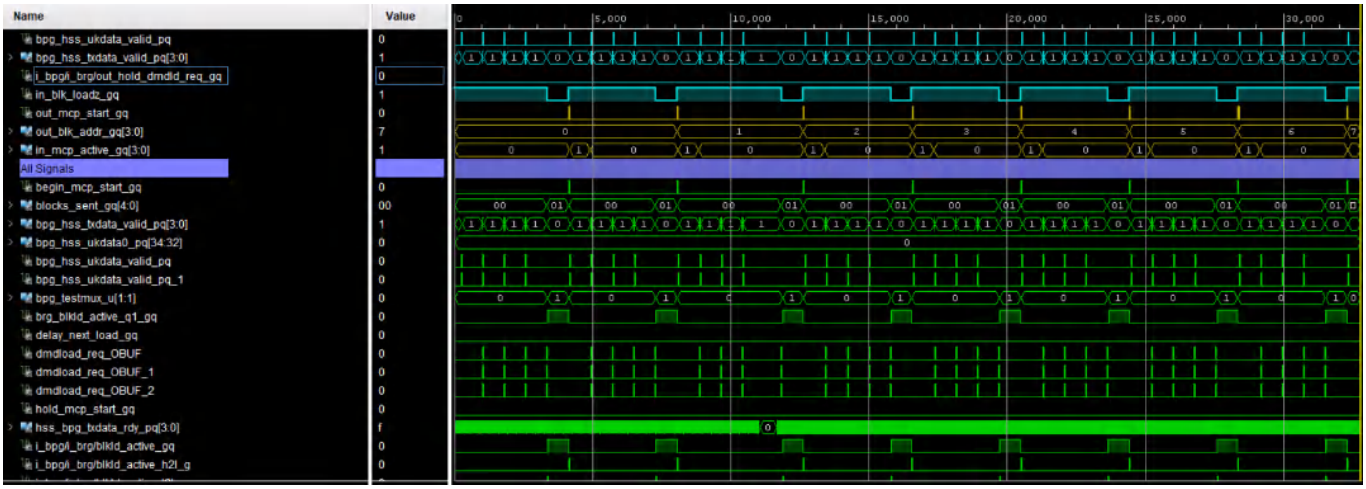



图 4-8. 图形模式 7 捕获结果

## 4.2 DLPC964 Apps 位流加载

### 4.2.1 将位流加载到 FPGA 中

按照以下说明使用 Vivado Lab Solutions 2018.2 通过位流将 DLPC964 Apps 二进制文件加载到 FPGA 中。

**备注**

点击上面的链接，下载 Vivado Lab Solutions 2018.2。网页加载完毕后，找到已存档的 2018.2 文件夹，然后导航到 Vivado Lab Solutions 2018.2 可下载链接并下载安装文件。

**备注**

每次断电/断开 AMD EVM 电源时，都需要重新加载 FPGA。

1. 将 Micro-B USB 电缆的一端插入 VC707，将另一端插入运行 Vivado 的计算机。
2. 在计算机上启动 Vivado Lab Solutions 2018.2。
3. 从主窗口中选择 *Open Hardware Manager*。
4. 点击位于硬件管理器左上角的 *open target*，然后点击 *Auto Connect*。
   a. 如果 AMD EVM 是唯一插入计算机的 FPGA，则 Vivado 会自动连接到 AMD EVM。
5. 右键点击 *FPGA*，然后选择 *Program Device*。
6. 导航到 appstop.mcs 文件并选择 *Program*。

### 4.2.2 将位流加载到闪存中

按照以下说明使用 Vivado Lab Solutions 2018.2 通过位流将 DLPC964 Apps 二进制文件加载到闪存中。

**备注**

点击上面的链接，下载 Vivado Lab Solutions 2018.2。网页加载完毕后，找到已存档的 2018.2 文件夹，然后导航到 Vivado Lab Solutions 2018.2 可下载链接并下载安装文件。

**备注**

该位流始终在 AMD EVM 上电时加载到 FPGA 中。

1. 将 Micro USB 的一端插入 AMD EVM，将另一端插入运行 Vivado 的计算机。
2. 确保将 SW11 设置为 00010（1 = 打开，位置 1 → 位置 5，从左到右）。

**图 4-9. FPGA 配置模式**

3. 将 SW2 设置为 00000000（1 = 打开，位置 1 → 位置 8，从左到右）。


**图 4-10. GPIO DIP 开关 (VC707)**

4. 在计算机上启动 Vivado Lab Studios 2018.2。
5. 从主窗口中选择 *Open Hardware Manager*。
6. 点击位于硬件管理器左上角的 *open target*，然后点击 *Auto Connect*。
   a. 如果 AMD EVM 是唯一插入计算机的 FPGA，则 Vivado 会自动连接到 AMD EVM。否则，此过程会稍微复杂一些。
7. 右键点击“FPGA”，然后选择 *Add Configuration Memory Device*。
8. 找到名为 mt28gu01gaax1e-bpi-x16 的闪存，然后点击 *OK*。
9. 再次选择 *OK*，然后选择配置文件 (appstop.mcs)。
   a. 确保所有其他设置都匹配。
10. 设置完成后，点击 *OK*。编程可能需要几分钟时间。
11. 完成后，对 AMD EVM 下电上电，DLPC964 Apps 位流会自动加载到 AMD EVM 上。

## 4.3 使用 Aurora 64B/66B 连接到 DLPC964 控制器

### 4.3.1 工作原理

DLPC964 控制器的数据处理以 DMD 块为基础。DLP991U 总共有 16 个 DMD 块，每个块为 4096 列 x 136 行。如图 4-11 所示，单行的 DMD 列进一步细分为四段（每段 1024 列），并独立映射到四个 Aurora 串行通道。因此，每个 Aurora 内核都是一个 1024 列 x 136 行的完整 DMD 块阵列。

### 4.3.2 概述

VC-707 Apps FPGA 和 DLPC964 控制器之间的数据传输通过 12 条 10Gbps 串行链路执行，如下面的图 4-11 所示。链路层协议是 Xilinx Aurora 64b/66b 串行接口。

Aurora 64b/66b 串行接口涵盖以内容：

- 使用 Xilinx Vivado IP Catalog 生成 Aurora 64b/66b TX 内核。
- RTL 包装器 aurora_apps_tx_x12ln.v 和 Apps 用户逻辑之间的接口信号。
- 使用 Aurora 64b/66b 传输 DMD 数据块的工作原理。
- 使用 Xilinx IBERT 工具集验证 10Gbps 通道链路的眼图张开度。

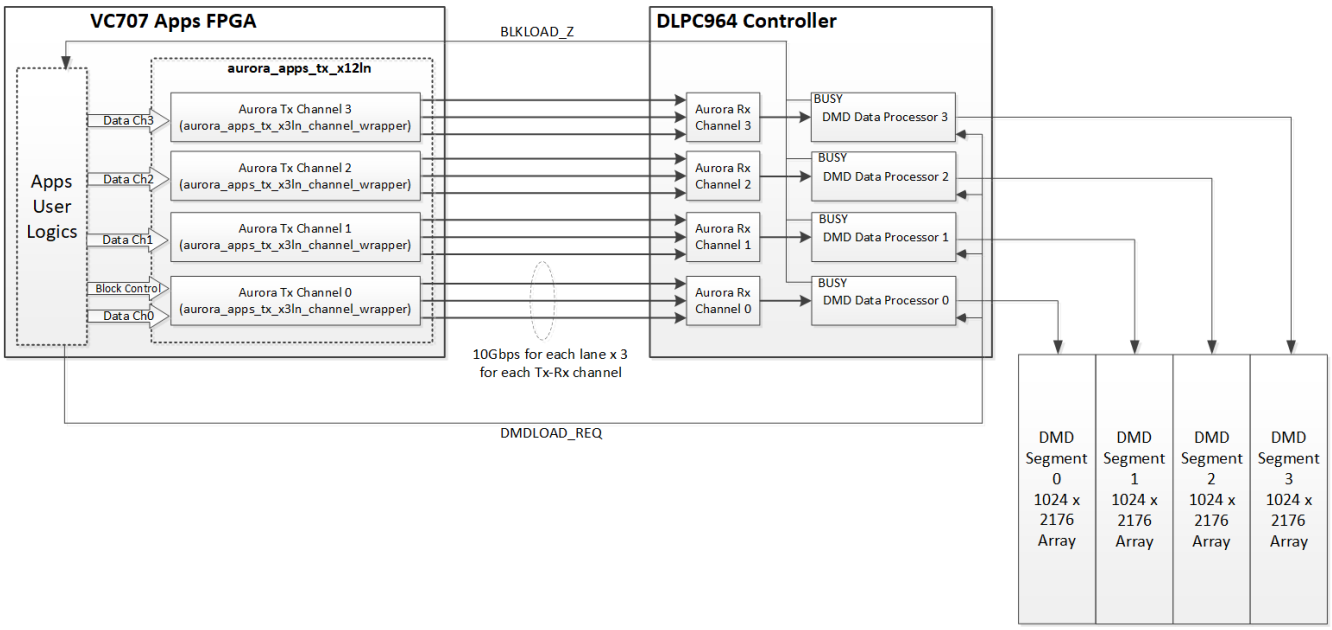



**图 4-11. DLPC964 系统方框图**

### 4.3.3 Aurora 64B/66B TX 内核和 RTL 生成

TI 针对 VC-707 Apps FPGA 参考工程发布了一个名为 aurora_apps_tx_x12ln.v 的 RTL 模块。此模块包含四个单独的 Aurora 64B/66B x3 信道内核 (aurora_apps_tx_x3ln_channel_wrapper.v) 以用于管理每个发送通道的流量。本节介绍使用 Xilinx Vivado 11.2 IP Catalog 生成 Aurora 内核的步骤。

### 4.3.3.1 从 IP Catalog 中选择 Aurora 64B66B

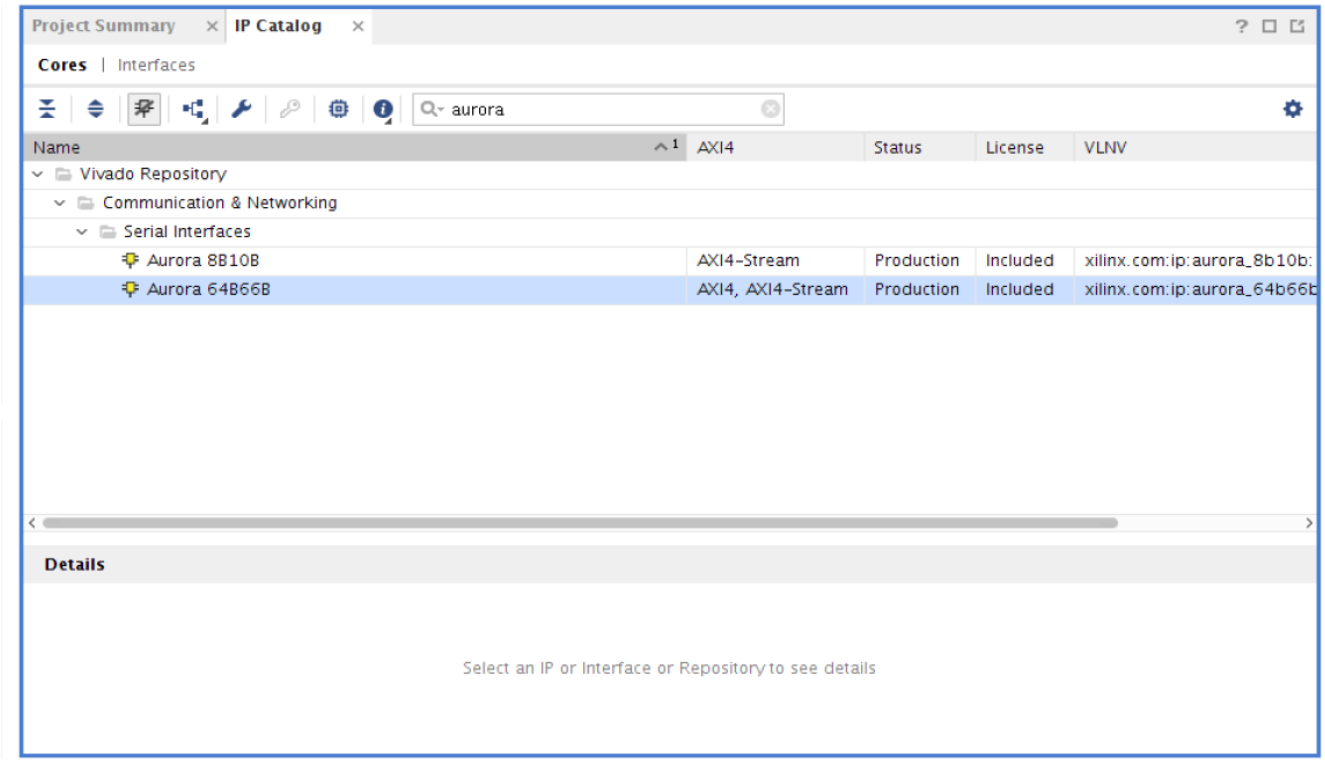



图 4-12. 从 IP Catalog 中选择

### 4.3.3.2 配置 Core Options

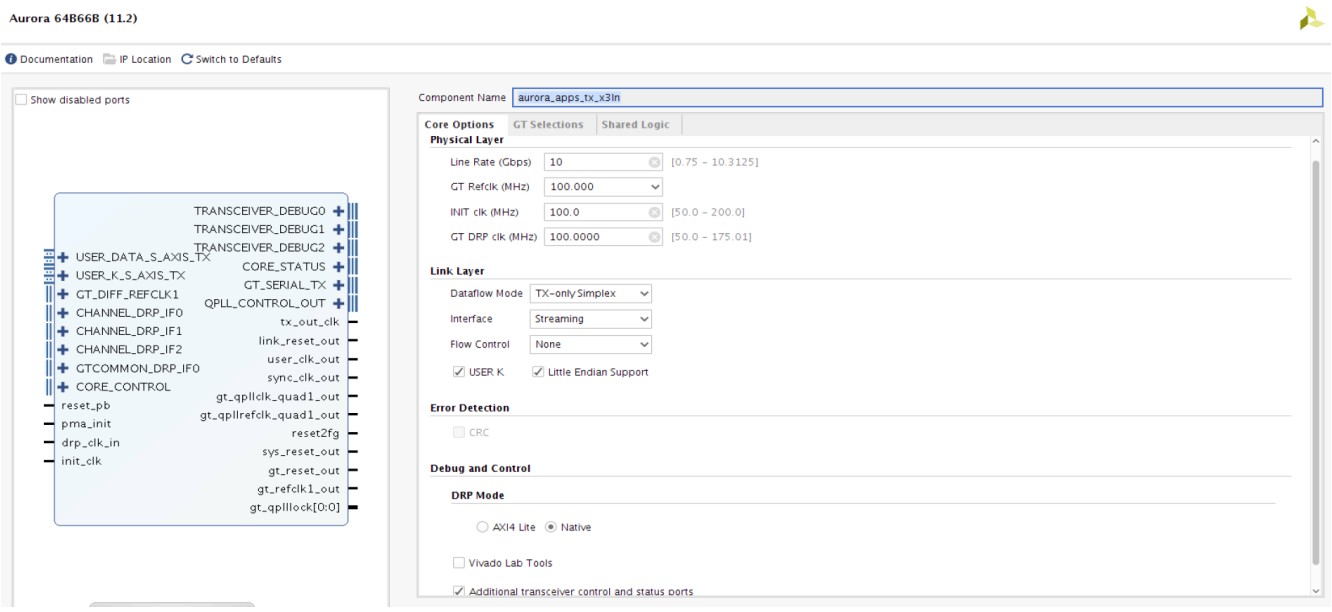



图 4-13. 配置 Core Options

#### 4.3.3.3 信道配置

在 *GT Selections* 下选择 3 条信道配置。

**备注**

尽管在此步骤中选择了四通道 GTXQ0，但内核的 RTL 没有任何位置锁定。因此，可以在顶层 x12 信道模块 (aurora_apps_tx_x12ln.v) 中实例化四次。

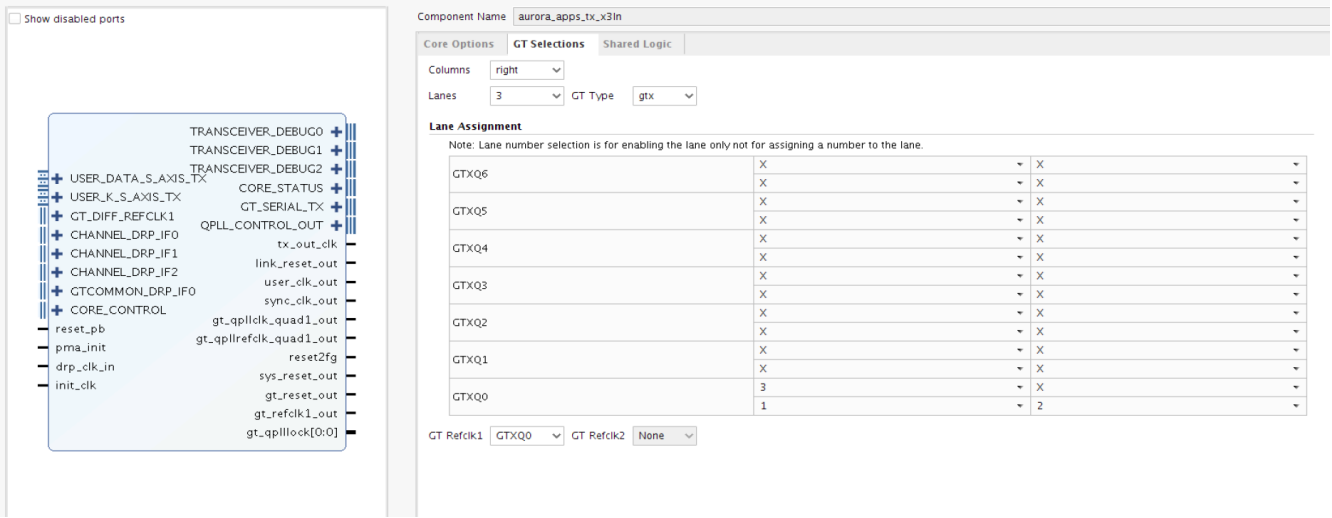



**图 4-14. 信道配置**

#### 4.3.3.4 Shared Logic 选项

选择 *Shared Logic* 下的 *Include Shared Logic in core* 选项。

继续生成 Aurora 内核。

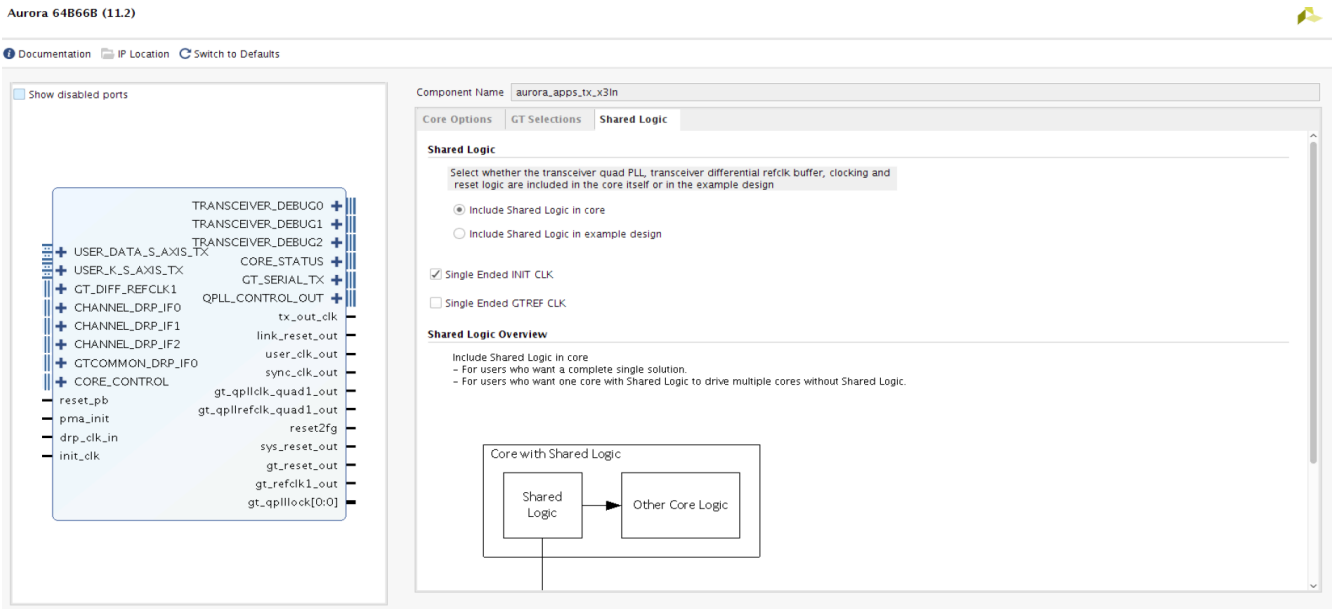



**图 4-15. Shared Logic 选项**

#### 4.3.3.5 生成示例设计文件

用户需要在工程的 *Sources* 部分中找到生成的内核。右键点击，然后选择 *Open IP Example Design* 选项，让 Vivado 生成示例设计。

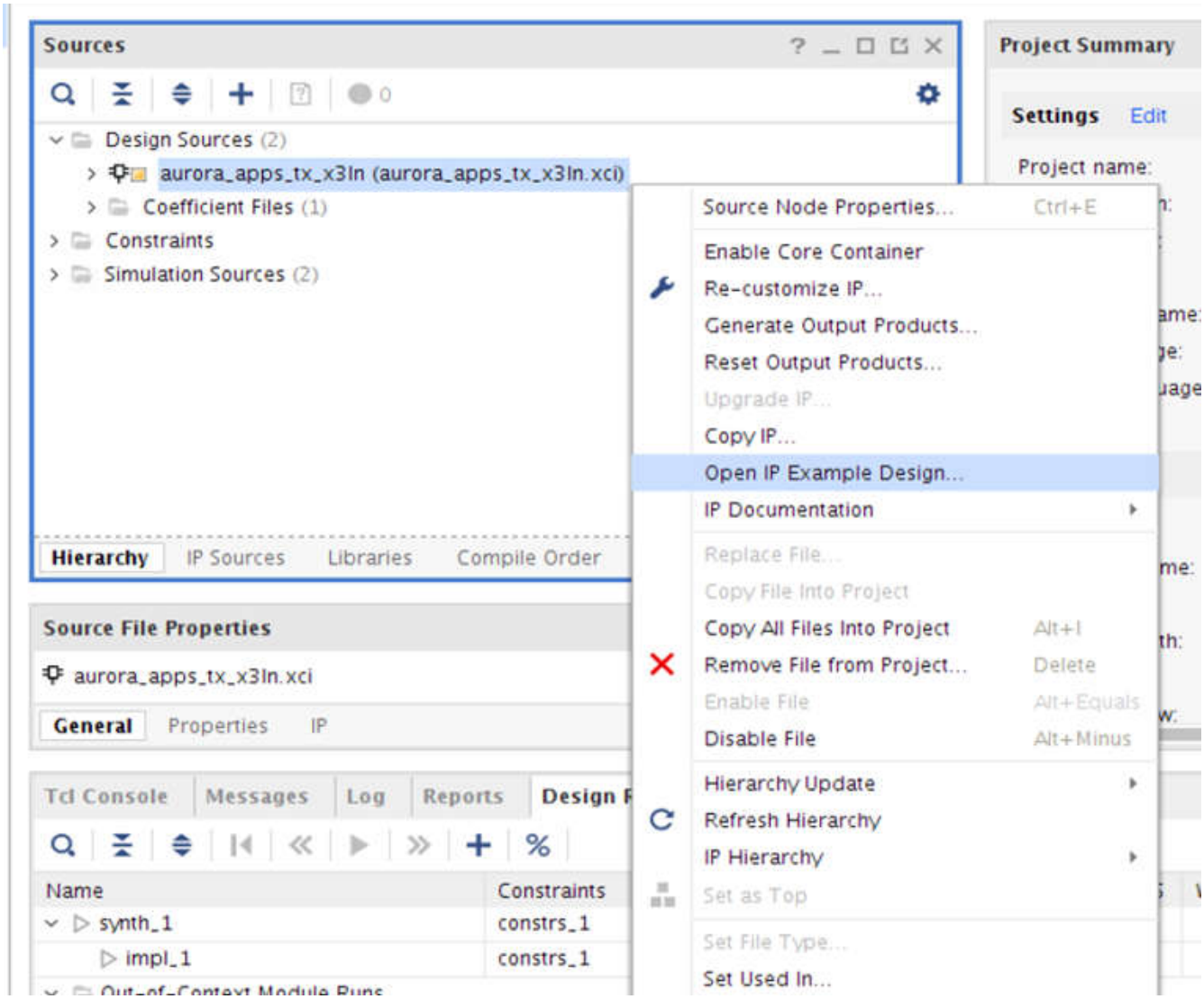



**图 4-16. 生成设计文件**

#### 4.3.3.6 RTL 文件列表

Vivado IP Catalog 生成的 RTL 文件列表如下所示。

**备注**

PROJECT_DIRECTORY 是用户的 Vivado 工程的主目录

RTL 示例设计文件位于以下目录：

**PROJECT_DIRECTORY/ip/aurora_apps_tx_x3ln/aurora_apps_tx_x3ln/example_design/gt**：

- aurora_apps_tx_x3ln_gtx.v
- aurora_apps_tx_x3ln_multi_wrapper.v
- aurora_apps_tx_x3ln_wrapper.v

RTL 设计文件的所有主要源文件都位于以下目录：

**PROJECT_DIRECTORY/ip/aurora_apps_tx_x3ln/aurora_apps_tx_x3ln/src/**：

- aurora_apps_tx_x3ln_64b66b_scrambler.v
- aurora_apps_tx_x3ln_axi_to_ll.v
- aurora_apps_tx_x3ln_cdc_sync.v
- aurora_apps_tx_x3ln_clock_module.v
- aurora_apps_tx_x3ln_gt_common_wrapper.v
- aurora_apps_tx_x3ln_ll_to_axi.v
- aurora_apps_tx_x3ln_reset_logic.v
- aurora_apps_tx_x3ln_standard_cc_module.v
- aurora_apps_tx_x3ln_support_reset_logic.v
- aurora_apps_tx_x3ln_sym_gen.v
- aurora_apps_tx_x3ln_tx_aurora_lane_simplex.v
- aurora_apps_tx_x3ln_tx_ch_bond_code_gen_simplex.v
- aurora_apps_tx_x3ln_tx_channel_err_detect_simplex.v
- aurora_apps_tx_x3ln_tx_channel_init_sm_simplex.v
- aurora_apps_tx_x3ln_tx_err_detect_simplex.v
- aurora_apps_tx_x3ln_tx_global_logic_simplex.v
- aurora_apps_tx_x3ln_tx_lane_init_sm_simplex.v
- aurora_apps_tx_x3ln_tx_startup_fsm.v
- aurora_apps_tx_x3ln_tx_stream_control_sm_simplex.v
- aurora_apps_tx_x3ln_tx_stream_datapath_simplex.v
- aurora_apps_tx_x3ln_tx_stream_simplex.v
- aurora_apps_tx_x3ln_support.v

#### 4.3.3.7 单通道 3 信道 Aurora 内核 RTL 包装器

3 通道 Aurora 包装器 aurora_apps_tx_x3ln_channel_wrapper.v 是 aurora_apps_tx_x3ln_support.v 经过简化和清理的 RTL，其中包含以下更改：

- 连接了所有 DRP 端口。
- 连接了未使用的 GTX 控制端口。
- 将所有三个通道的 DIFFCTRL、后标、前标和主标控制功能合并到一个通道中。
- 移除 aurora_apps_tx_x3ln_clock_module ( 时钟模块移动到 aurora_apps_tx_x12ln.v，因此所有四个通道可以共享同一个用户时钟 )。有关详细信息，请参阅节 4.3.3.8。

清理 RTL 的目的是为了简化 I/O 端口列表，以便于进行下一节中所述的元件实例化。

#### 4.3.3.8 四通道 12 信道顶层 RTL 包装器

图 4-17 所示的 aurora_apps_tx_x12ln.v 具有模块 aurora_apps_tx_x3ln_channel_wrapper.v 的四个实例，旨在形成四个 Aurora TX 通道实体。

通道 0 的 tx_out_clk 会馈送到 aurora_apps_tx_x3ln_clock_module.v 中以生成用于驱动 Apps FPGA 和 Aurora 用户逻辑接口的 clk_user。请参阅 Xilinx 应用手册的*第 2 章表 2-7 : Aurora 64B/66B 内核时钟端口* 和*第 3 章图 3-1 Aurora 64B/66B 时钟架构*，了解有关 tx_out_clk 和 clk_user 的信息。

**备注**

如果使用 64B/66B 接口时的链路速度为 10Gbps，则 clk_user 频率 = 10GHz / 64 = 156.25MHz。

复位逻辑会为四个 Aurora TX 通道生成复位信号 *reset_pb* 和 *pma_init*。如需了解有关生成 reset_pb 和 pma_init 的规格，请参阅 Xilinx 应用手册的*第 3 章图 3-5 Aurora 64B/66B 单工正常运行复位序列*。

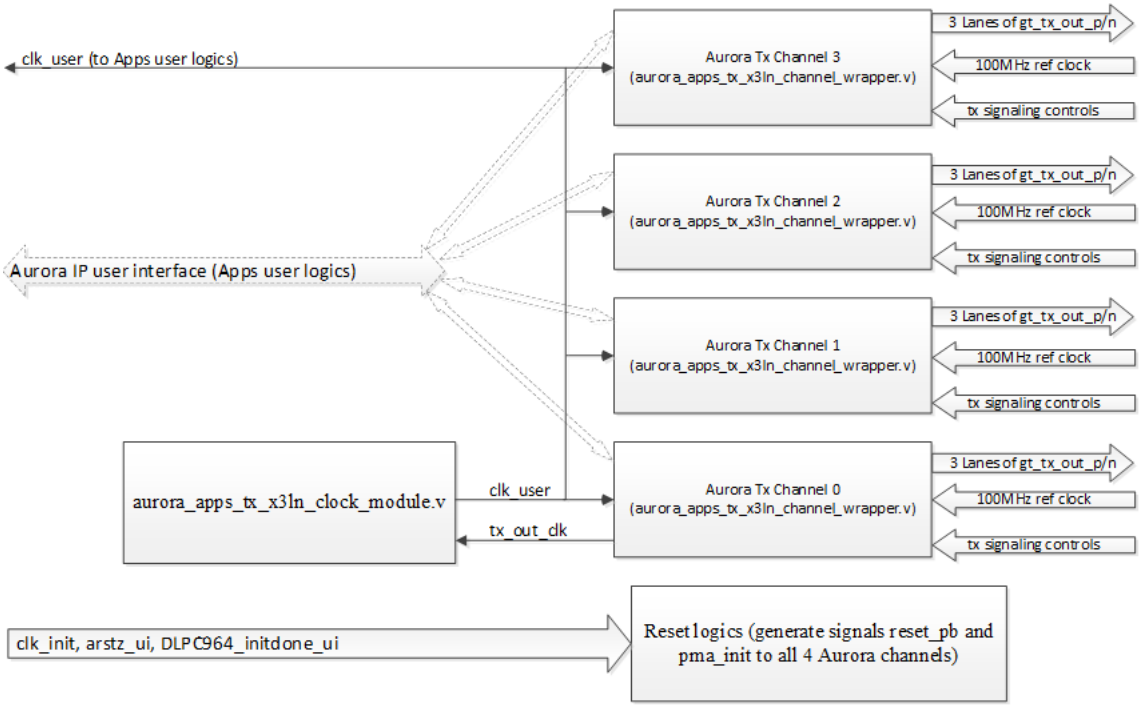



**图 4-17. Aurora_apps_tx_x12ln.v RTL 方框图**

表 4-1. RTL aurora_apps_tx_x12ln.v 的信号端口列表

| 名称 | 方向 | 时钟域 | 说明 |
|---|---|---|---|
| clk_init | 输入 | clk_init | 用于为 Aurora 内核生成复位信号的 100MHz 自由运行时钟。 |
| arstz_ui | 输入 | async | 低电平有效复位输入。输入低电平将触发 Aurora 内核的复位操作。 |
| dlpc964_initdone_ui | 输入 | async | DLPC964 控制器 INIT_DONE 状态信号。低电平使 Aurora 内核保持复位状态。 |
| gt0_txout_n[0:2] | 输出 | async | 连接到 DLPC964 控制器的通道 0 10Gbps 差分输出信道 0、1 和 2。 |
| gt1_txout_p[0:2] | 输出 | async | 连接到 DLPC964 控制器的通道 1 10Gbps 差分输出信道 0、1 和 2。 |
| gt2_txout_p[0:2] | 输出 | async | 连接到 DLPC964 控制器的通道 2 10Gbps 差分输出信道 0、1 和 2。 |
| gt3_txout_p[0:2] | 输出 | async | 连接到 DLPC964 控制器的通道 3 10Gbps 差分输出信道 0、1 和 2。 |
| gt0_refclkin_p | 输入 | async | 来自低抖动振荡器的通道 0 100MHz 差分收发器外部基准时钟。 |
| gt1_refclkin_p | 输入 | async | 来自低抖动振荡器的通道 1 100MHz 差分收发器外部基准时钟。 |
| gt2_refclkin_p | 输入 | async | 来自低抖动振荡器的通道 2 100MHz 差分收发器外部基准时钟。 |
| gt3_refclkin_p | 输入 | async | 来自低抖动振荡器的通道 3 100MHz 差分收发器外部基准时钟。 |
| gt_txpostcursor_in[4:0] | 输入 | async | 收发器后标 TX 预加重控制，对于 TI EVM 硬件，设置为“00000”。<br>客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txdiffctrl_in[3:0] | 输入 | async | 收发器 TX 驱动器摆幅控制，对于 TI EVM 硬件，设置为“1000”（807mV 差分峰峰值摆幅）。<br>客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txmaincursor_in[6:0] | 输入 | async | 收发器主标 TX 控制，对于 TI EVM 硬件，设置为“0000000”。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txprecursor_in[4:0] | 输入 | async | 收发器前标 TX 预加重控制，对于 TI EVM 硬件，设置为“00000”。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| clk_user | 输出 | clk_user | 连接到 Aurora 内核的用户接口的 156.25MHz 时钟。 |
| clk_user_not_locked_uo | 输出 | async | 处于高电平时，表示 clk_user 未被锁定，如果 clk_user 在上电或系统复位条件下锁定松动，则可用于将用户逻辑保持在复位状态。 |
| gt(0,1,2,3)_s_axi_tx_tdata[191:0] | 输入 | clk_user | 要通过 Aurora 链路发送的 DMD 像素数据。 |
| gt(0,1,2,3)_s_axi_tx_tvalid | 输入 | clk_user | 用户逻辑将此信号置为高电平有效以向 Aurora 内核指示 DMD 像素数据有效，因此可进行发送。Aurora 内核在 tvalid 为低电平时会忽略数据。请参阅 Xilinx 应用手册，了解 AXI4-stream *tready* 信号行为。 |
| gt(0,1,2,3)_s_axi_tx_tready | 输出 | clk_user | 当接受 DMD 像素数据时，Aurora 内核将此信号置为高电平有效。当忽略像素数据时（例如内核未准备好接受数据），置为无效。请参阅 Xilinx 应用手册，了解 AXI4-stream *tready* 信号行为。 |
| gt(0,1,2,3)_s_axi_user_k_tx_tdata[191:0] | 输入 | clk_user | 要通过 Aurora 链路发送的 user-k 控制字数据。 |
| gt(0,1,2,3)_s_axi_user_k_tx_tvalid | 输入 | clk_user | 用户逻辑将此信号置为高电平有效以向 Aurora 内核指示 user-k 控制字数据有效，因此可进行发送。Aurora 内核在 tvalid 为低电平时会忽略数据。 |
| gt(0,1,2,3)_s_axi_user_k_tx_tready | 输出 | clk_user | 当接受 user-k 控制字数据时，Aurora 内核将此信号置为高电平有效。当忽略数据时（例如内核未准备好接受数据），置为无效。 |
| gt(0,1,2,3)_hard_err | 输出 | clk_user | Aurora 内核检测到硬错误时，置为高电平有效。如需了解硬错误定义，请参阅 Xilinx 应用手册中的*表 2-13*。 |
| gt(0,1,2,3)_soft_err | 输出 | clk_user | Aurora 内核检测到软错误时，置为高电平有效。如需了解软错误定义，请参阅 Xilinx 应用手册中的*表 2-13*。 |
| gt(0,1,2,3)_channel_up | 输出 | clk_user | Aurora 内核完成通道初始化序列后，置为高电平有效。 |
| gt(0,1,2,3)_lane_up[2:0] | 输出 | clk_user | 成功进行信道初始化后使每个信道置为高电平有效（每个位代表一个信道）。 |
| tp_gt0_pll_lock | 输出 | async | Aurora 通道 0 tx_out_clk 处于稳定状态时，置为高电平有效。如前一节所述，通道 0 tx_out_clk 用于生成 clk_user。<br>tx_out_clk 是来自 Aurora 收发器的 312.5MHz，除以 2 形成 156.25MHz 的 clk_user。 |

#### *4.3.3.9 块以块控制字开始*

Aurora 是没有 DMD 块概念的通用数据传输链路。要定义 DMD 块的开始，Apps 用户逻辑必须在数据传输之前通过通道 0 Aurora user-k 端口发送块控制字数据包。

如需有关 Aurora user-k 接口端口的详细信息，请参阅 Xilinx 应用手册第 2 章的表 2-10。总之，user-k 接口端口用于实现特定于应用的控制功能，独立于数据接口，且优先级高于数据接口。如表 4-2 所示，RTL 包装器 aurora_apps_tx_x12ln.v 有四个 user-k 端口接口通道向 Apps FPGA 用户逻辑公开。

**备注**

只有通道 0 用于发送块控制字。DLPC964 控制器不会使用通过通道 1、2 和 3 的 user-k 端口发送的控制字数据包，而是将这些数据包忽略。

**表 4-2. RTL 包装器“aurora_apps_tx_x12ln.v”user-k 端口使用情况**

| 信号名称 | 信号方向 | DLPC964 应用使用情况 |
|---|---|---|
| gt0_s_axi_user_k_tx_tdata[191:0] | Aurora 通道 0 的输入 | 要发送的 192 位块控制字数据包 |
| gt0_s_axi_user_k_tx_tvalid | Aurora 通道 0 的输入 | 用户逻辑将此信号置为高电平有效以向 Aurora 内核指示块控制字有效，因此可进行发送。<br>Aurora 内核在 tvalid 为低电平时会忽略字。 |
| gt0_s_axi_user_k_tx_tready | Aurora 通道 0 的输出 | 当接受块控制字时，Aurora 内核将此信号置为高电平有效。<br>当忽略字时（例如内核未准备好接受输入字），置为无效。 |
| gt1_s_axi_user_k_tx_tdata[191:0] | Aurora 通道 1 的输入 | 未使用 |
| gt1_s_axi_user_k_tx_tvalid | Aurora 通道 1 的输入 | 未使用 |
| gt1_s_axi_user_k_tx_tready | Aurora 通道 1 的输出 | 未使用 |
| gt2_s_axi_user_k_tx_tdata[191:0] | Aurora 通道 2 的输入 | 未使用 |
| gt2_s_axi_user_k_tx_tvalid | Aurora 通道 2 的输入 | 未使用 |
| gt2_s_axi_user_k_tx_tready | Aurora 通道 2 的输出 | 未使用 |
| gt3_s_axi_user_k_tx_tdata[191:0] | Aurora 通道 3 的输入 | 未使用 |
| gt3_s_axi_user_k_tx_tvalid | Aurora 通道 3 的输入 | 未使用 |
| gt3_s_axi_user_k_tx_tready | Aurora 通道 3 的输出 | 未使用 |

表 4-3 介绍了 192 位块控制字中的各种字段。块控制字不仅定义了 DMD 块的开始，还包含关于 DLPC964 控制器如何处理 DMD 块数据接收的指导说明和信息。

**表 4-3. 块控制字段定义**

| 字段位置 | 字段类型 | 字段说明 |
|---|---|---|
| gt0_s_axi_user_k_tx_tdata[7:0] | USER_BLOCK_NUMBER | 必须设置为零 (0x00)。0x00 以外的值均无效，如果此字段不为零，DLPC964 控制器会忽略整个 192 位控制字。 |
| gt0_s_axi_user_k_tx_tdata[11:8] | BLOCK_ADDRESS | 指示 DLPC964 将操作应用到的 DMD 块地址。<br>0000：DMD 块 0<br>0001：DMD 块 1<br>0010：DMD 块 2<br>….….<br>1111：DMD 块 15 |
| gt0_s_axi_user_k_tx_tdata[15:7] | | 保留，未使用。 |
| gt0_s_axi_user_k_tx_tdata[24:16] | ROW_LENGTH | DLPC964 加载用户数据的行数。DLP991U DMD 在每个块中有 136 行，因此有效范围为 1-136。包括 0 在内的所有其他值均无效。设置为 136 表示执行全块的操作。值 1 至 135 表示非全块的操作。<br>**备注**<br>在 LOAD_TYPE 为 000 时使用此字段。 |
| gt0_s_axi_user_k_tx_tdata[34:32] | LOAD_TYPE | 000：块加载。DLPC964 将用户数据加载到由 BLOCK_ADDRESS 和 ROW_LENGTH 定义的 DMD 阵列区域中。<br>001：块清除。DLPC964 将 DMD 阵列中由 BLOCK_ADDRESS 定义的整个块清零。<br>010：块置位。DLPC964 将 DMD 阵列中由 BLOCK_ADDRESS 定义的整个块设置为 1。其他值：保留，请勿使用。<br>**备注**<br>在 001（块清除）或 010（块置位）操作中，ROW_LENGTH 和 NORTH_SOUTH_FLIP 字段会被忽略，因为清除和置位操作会影响整个 DMD 块阵列。换句话说，块置位/清除操作不支持非全块的操作。 |
| gt0_s_axi_user_k_tx_tdata[36] | NORTH_SOUTH_FLIP | 控制 DMD 块内的数据加载方向。<br>0：DLPC964 从第 1 行开始加载数据并向上计数。<br>1：DLPC964 从第 136 行开始加载数据并向下计数。 |
| gt0_s_axi_user_k_tx_tdata[29:28] | DMD_SEGMENT | 当 SINGLE_CHANNEL_MODE =“1”时，选择 DLC964 将操作应用到的 DMD 段。<br>如果 SINGLE_CHANNEL_MODE =“0”，DLPC964 控制器会忽略此字段。 |
| gt0_s_axi_user_k_tx_tdata[30] | SINGLE_CHANNEL_MODE | 1：单通道运行模式。仅使用 Aurora 通道 0 运行 DMD 阵列。<br>0：正常运行模式。使用所有四个 Aurora 通道运行 DMD 阵列。 |
| gt0_s_axi_user_k_tx_tdata[191:31] | | 保留，未使用。 |

图 4-18 显示了 Aurora 块发送开始时通过通道 0 user-k 端口发送 192 位块控制字（本例中加载 DMD 块 1 的全部 136 行）。

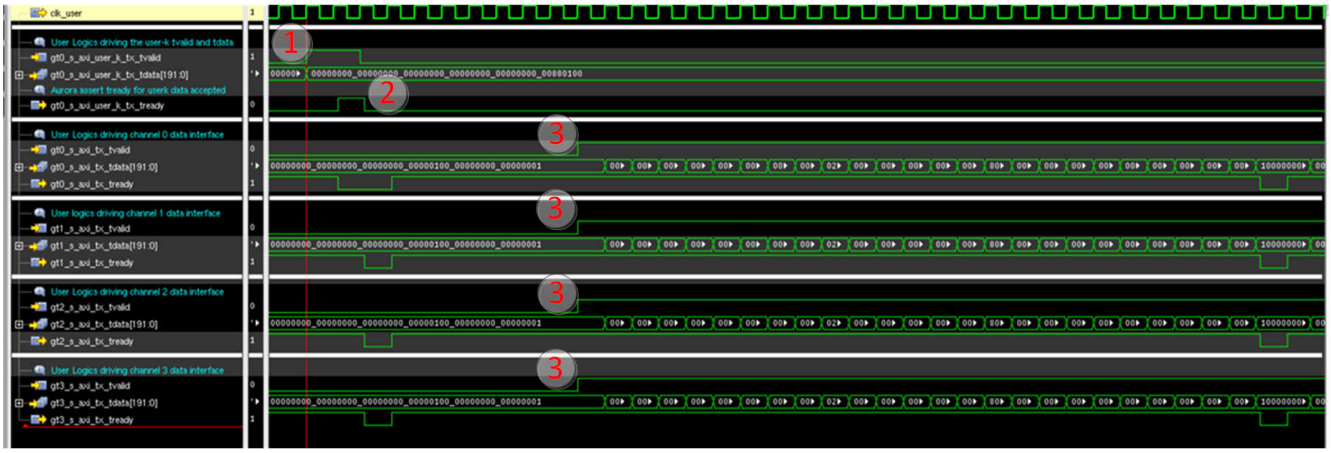



**图 4-18. 块以块控制字开始的波形**

1. 通过在 gt0_s_axi_user_k_tx_tdata[191:0] 总线上使用正确的块控制字，Apps FPGA 用户逻辑将 TVALID 标志 (gt0_s_axi_user_k_tx_tvalid) 置为有效，并等待 Aurora 内核的响应。
2. Aurora 将 TREADY 标志 gt0_s_axi_user_k_tx_tready 置为有效，指示内核已接受 192 位 user-k 数据。
3. 发送块控制字后，Apps FPGA 用户逻辑开始在全部四个数据接口上进行 Aurora 块传输。

#### *4.3.3.10 块以 DMDLOAD_REQ 完成*

请参阅图 4-11：DLPC964 系统方框图

DMDLOAD_REQ 是从 Apps FPGA 发送到 DLPC964 控制器的信号。

一旦 Aurora 块数据传输完成，Apps FPGA 用户逻辑必须将 DMDMDLOAD_REQ 置为有效以便向 DLPC964 指示这是 DMD 块的结束位置并触发执行块控制字中编码的操作。

将 DMDLOAD_REQ 信号置为有效和发送块控制字的指南：

- Apps FPGA 用户逻辑必须等待块传输在全部四个 Aurora 数据通道上完成，然后再将 DMDLOAD_REQ 置为有效。
  - Apps FPGA 必须考虑到四个 Aurora 数据通道接口彼此不完全同步，因此不会在完全相同的时钟周期完成数据传输。所以，在将 DMDLOAD_REQ 置为有效前，Apps FPGA 必须监控并验证 Aurora 块数据传输在所有四个通道上均已完成。

**备注**

在 Aurora 块传输完成前将 DMDLOAD_REQ 置为有效可能导致数据不能正确加载到 DMD。

- 在完成一个 Aurora 块传输后，只要满足 300ns 的 DMDLOAD_REQ 建立时间，即可将 DMDLOAD_REQ 立即置为有效（更多信息，请参阅节 4.3.3.11）。
- Apps FPGA 用户逻辑必须在启动下一个新 DMD 块的发送之前将当前块的 DMDLOAD_REQ 置为有效。每个块必须以一个块控制字数据包开头并以 DMDLOAD_REQ 置位结束。
- 对于不涉及块数据传输的操作（例如块清除/置位操作），仍然需要 DMDLOAD_REQ，并且仍然必须满足 300ns 的建立时间（更多信息，请参阅节 4.3.3.11）。
- 请参阅图 4-19，了解在 Apps 用户逻辑完成当前块的传输后的情况。用户可以发现 DLPC964 仍在将上一个块加载到 DMD 中（例如，BLKLOADZ 为低电平）。在 BLKLOADZ 为低电平时，Apps FPGA 仍可以将 DMDLOAD_REQ 置为有效，因为 DLPC964 可以检测并存储该 DMDLOAD_REQ 请求。在完成当前 DMD 块的 Aurora 数据传输并将 DMDLOAD_REQ 信号置为有效后，Apps FPGA 必须等待 DLPC964 将 BLKLOADZ 置为无效（例如，BLKLOADZ 从低电平转换为高电平），然后再启动下一个块的传输。BLKLOADZ 置为无效意味着 DLPC964 已完成了前一个块的 DMD 数据加载操作，并且释放了数据缓冲器以便从 Aurora 接口接受新的数据块。

**备注**

DLPC964 具有两个数据块缓冲器：一个用于接收传入的 Aurora 数据块，另一个用于保存前一个数据块以将数据流式输出到 DMD。如果 Apps FPGA 没有将 Aurora 块传输与 BLKLOADZ 的“置为无效”信号同步，则缓冲器可能会溢出，数据也无法正确加载到 DMD。

- 请参阅图 4-20，了解 Apps FPGA 选择向 DLPC964 发送 DMD 数据块但延迟 DMDLOAD_REQ 置为有效的情况。


**图 4-19. 块 DMDLOAD_REQ 置为有效结束之后的新块控制字波形**

1. 在全部四个 Aurora 数据接口上的当前块数据传输完成后，Apps FPGA 用户逻辑立即将 DMDLOAD_REQ 置为有效。
2. DLPC964 将 BLKLOADZ 置为无效以指示前一个 DMD 块的数据加载操作完成。
3. APPS FPGA 用户逻辑会检测 BLKLOADZ 置为无效的情况，并在 Aurora 通道 0 user-k 端口上为下一个块发送新的块控制字。
4. Apps FPGA 用户逻辑发送下一个块的数据。
5. BLKLOADZ 由 DLPC964 置为低电平有效，指示当前块的数据加载操作由 DMDLOAD_REQ 触发。

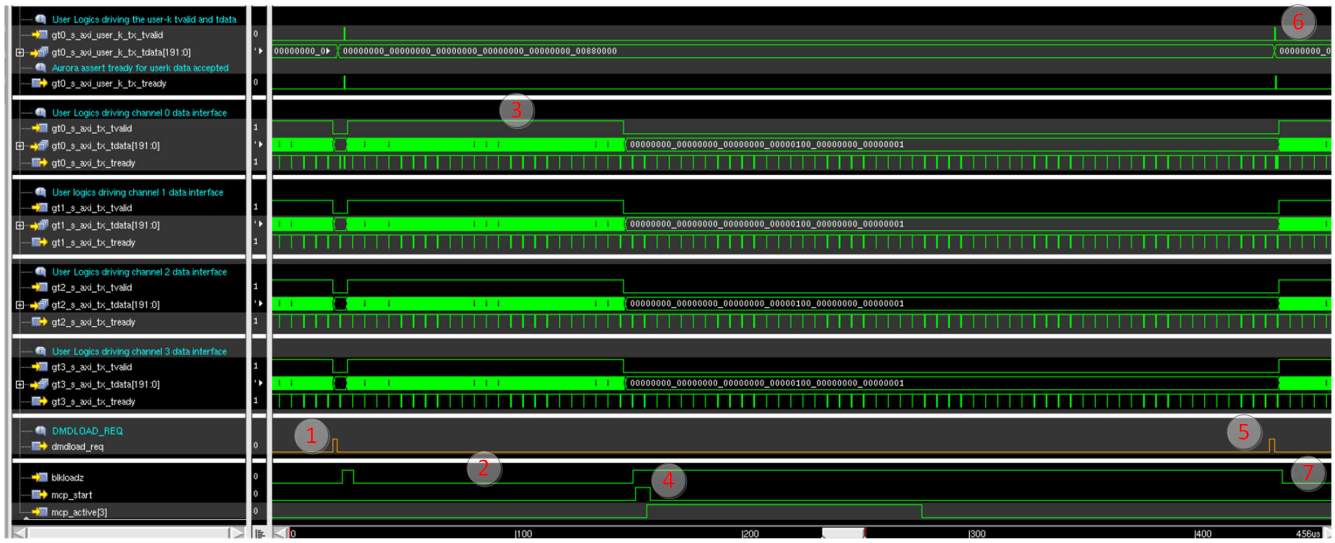



**图 4-20. DMDLOAD_REQ 延迟置为有效波形**

1. Apps FPGA 完成当前图形的最后一个块（块 15）的数据发送，并将 DMDLOAD_REQ 置为有效以指示 DLPC964 执行数据加载操作。
2. 从块 1 到块 15，DLPC964 均由 DMDLOAD_REQ 触发加载数据。
3. 当 DLPC964 加载当前图形的块 15 时，Apps FPGA 通过 Aurora 数据接口发送下一个图形第一个块（块 0）的数据。
4. 在当前图形的块 15 的数据加载完成后，DLPC964 会将 BLKLOADZ 置为无效。Apps FPGA 检测到针对当前图形最后一个块的 BLKLOADZ 已置为无效并且这个块已经加载到 DMD 上，然后发出 MCP_START 以进行全局块复位操作。
5. 由于需要满足微镜稳定时间的要求，Apps FPGA 延迟将下一个图形中块 0 的 DMDLOAD_REQ 置为有效。
6. 将下一个图形的块 0 的 DMDLOAD_REQ 置为有效后，发送块 1 的块控制字。
7. DLPC964 将 BLKLOADZ 置为有效以指示 DMD 数据加载操作是由第 5 部分的 DMDLOAD_REQ 触发的。

#### 4.3.3.11 DMDLOAD_REQ 建立时间要求

在传输完一个 Aurora 块后，只要满足在发送该块的第一个数据包后至少经过了 300ns，Apps FPGA 用户逻辑即可将 DMDLOAD_REQ 信号置为有效。之所以需要这样的建立时间，是因为 Aurora TX/RX 通道路径有 300ns 的发送延时，旨在验证 DLPC964 在 Aurora 块数据到达后收到了 DMDLOAD_REQ 标志。

在大多数情况下，由于数据块足够大，因此可以确认从发送数据块的第一个有效数据包到最后一个数据包的经历时间超过 300ns，此时 Apps 能够将 DMDLOAD_REQ 信号置为有效，所以这 300ns 的建立时间要求将自然得到满足。Apps FPGA 尝试发送一个小的不完整 DMD 块时，这种情况下 300ns 的建立时间窗口变得至关重要，如图 4-21 中的示例所示，Apps FPGA 将不完整 DMD 块的总共 3 行（表 4-2，ROW_LENGTH = 3）发送到 DLPC964：

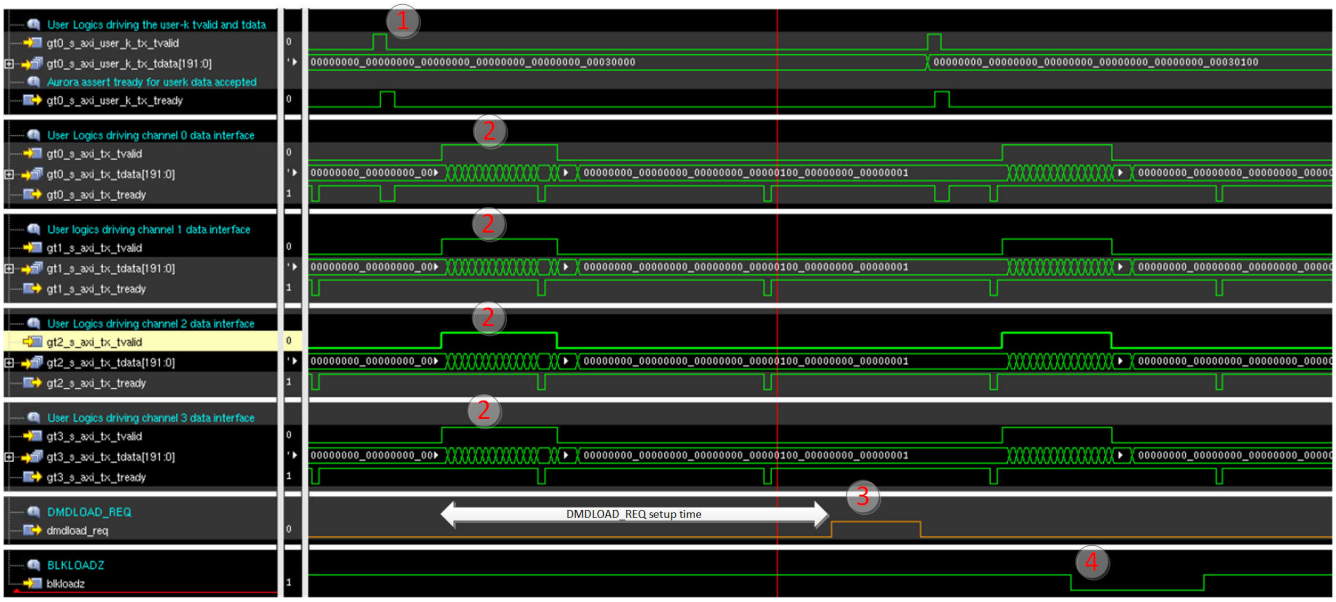



**图 4-21. 三个 DMD 行的加载操作的 DMDLOAD_REQ 建立时间**

1. APPS FPGA 发送一个块控制字以指示 Aurora 块传输开始。
2. 通过四个 Aurora 数据接口通道发送三个数据行后，Apps FPGA 等待 300ns 的建立时间到期，然后再发出 DMDLOAD_REQ。

**备注**

300ns 是从数据接口上的第一个 TVALID 开始进行测量的。

3. 一旦满足建立时间，Apps FPGA 就会将 DMDLOAD_REQ 置为有效。
4. BLKLOADZ 由 DLPC964 置为有效以指示 DMD 数据加载操作正在运行。

对于不需要数据包的操作，例如块清除（表 3，LOAD_TYPE = 001）和块置位（表 3，LOAD_TYPE = 010），这 300ns 的建立时间 DMDLOAD_REQ 仍然是必需的，并从块控制字数据包测得。图 4-22 是一个块置位操作的示例。

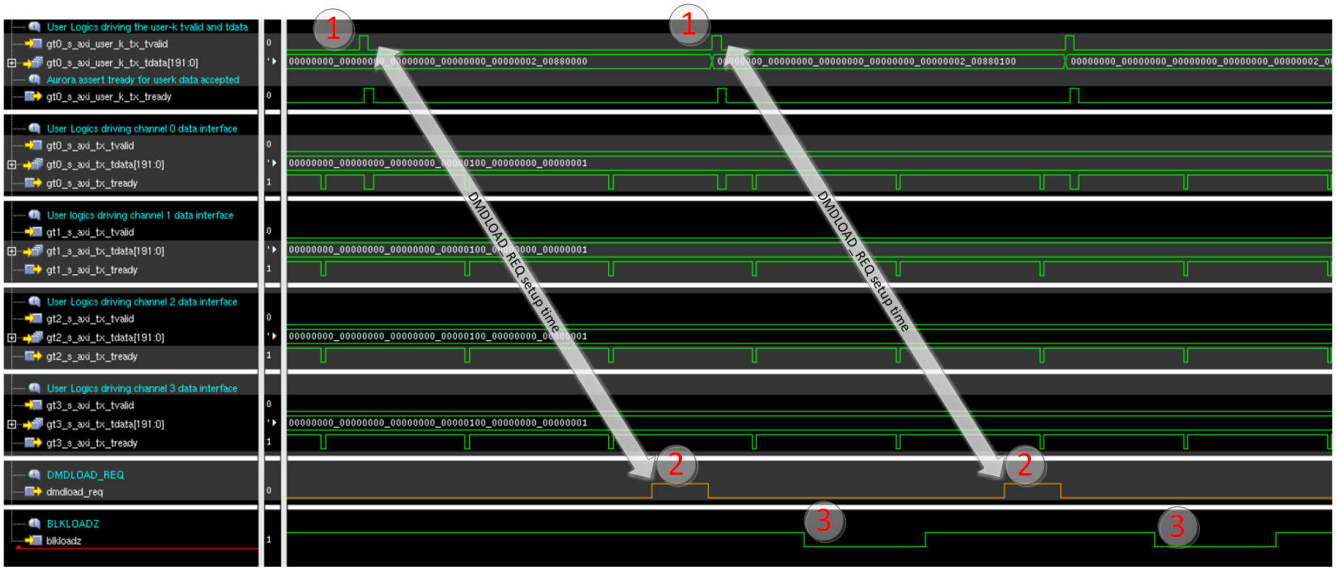



**图 4-22. 块置位操作的 DMDLOAD_REQ 建立时间**

1. Apps FPGA 发送一个块控制字数据包以启动块置位操作。请注意，该操作不需要任何块数据，因为四个数据接口保持空闲状态 (gtX_s_axi_tx_tvalid = ‘0’)。
2. Apps FPGA 在 300ns 的建立时间后将 DMDLOAD_REQ 置为有效。由于块置位操作不需要 Aurora 数据传输，因此这 300ns 的建立时间从块控制字测得。
3. DLPC964 将 BLKLOADZ 置为有效以指示正在进行块置位操作。

#### 4.3.3.12 单通道传输模式

对于非关键图形速率应用，DLPC964 支持仅在 Aurora 通道 0 中运行。在此运行模式下，仅使用通道 0 的三条 10Gbps 串行链路，并且必须是通道 0。

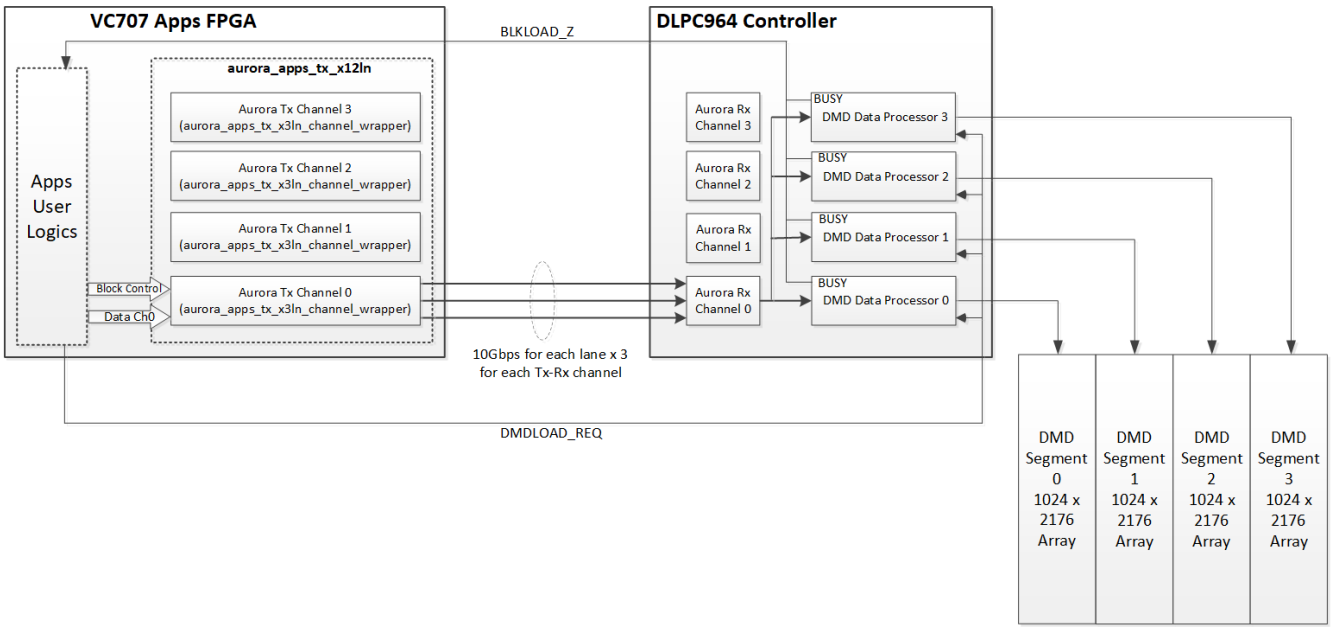



**图 4-23. 单通道运行模式的系统方框图**

为了实现在该模式下运行，需要设置块控制字字段、SINGLE_CHANNEL = “1”（表 4-2）并按 3（第一个）、2、1、0（最后一个）的顺序传输 DMD 段（表 4-2 中的 DMD_SEGMENT 字段）。换句话说，为了控制一个特定的 DMD 块，Apps FPGA 必须首先操作这个块的段 3，之后是段 2、段 1 和段 0（最后一个传输段）。

节 4.3.3.9、节 4.3.3.10 和节 4.3.3.11 中所述的指导原则仍然适用于单通道运行模式，在此模式下，Aurora 传输的每个块和段仍必须以块控制字开始，以 DMDLOAD_REQ 结束，并需要 300ns 的建立时间。但是，此模式下与 Apps FPGA/DLPC964 握手相关的一个主要区别是，仅由 DMDLOAD_REQ 触发的实际 DMD 操作对应于段 0；例如，BLKLOADZ 不会针对段 3、2 和 1 置为无效。（如需了解详细信息，请参阅图 4-24）。

所选块的所有四个段都必须按段 3（第一个）、2、1 和 0（最后一个）的顺序操作，否则 DLPC964 不会对该块执行正确的 DMD 操作。

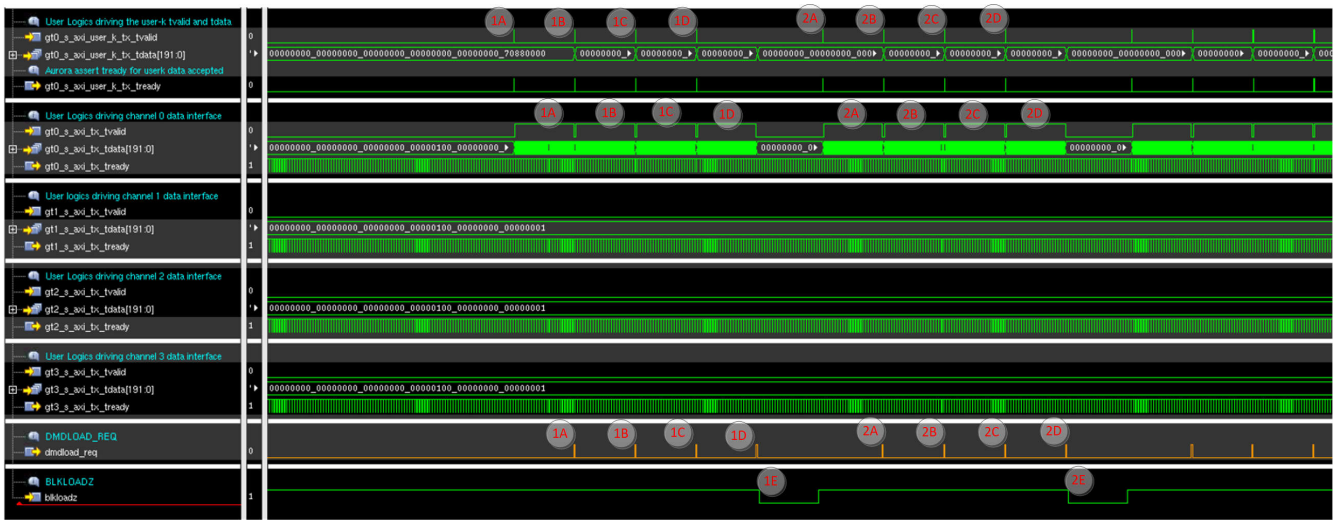



**图 4-24. 单通道运行模式波形示例**

1. 单通道模式下 DMD 块 0 的 Apps FPGA Aurora 数据传输。
   a. DMD 块 0 段 3 的块控制字、DMD 数据和 DMDLOAD_REQ。
   b. DMD 块 0 段 2 的块控制字、DMD 数据和 DMDLOAD_REQ。
   c. DMD 块 0 段 1 的块控制字、DMD 数据和 DMDLOAD_REQ。
   d. DMD 块 0 段 0 的块控制字、DMD 数据和 DMDLOAD_REQ。
   e. 段 0 的 DMDLOAD_REQ 触发 DLPC964 以开始块 0 的数据加载，使 BLKLOADZ 置为有效以指示正在进行操作。
2. 单通道模式下 DMD 块 1 的 Apps FPGA Aurora 数据传输。
   a. DMD 块 1 段 3 的块控制字、DMD 数据和 DMDLOAD_REQ。
   b. DMD 块 1 段 2 的块控制字、DMD 数据和 DMDLOAD_REQ。
   c. DMD 块 1 段 1 的块控制字、DMD 数据和 DMDLOAD_REQ。
   d. DMD 块 1 段 0 的块控制字、DMD 数据和 DMDLOAD_REQ。
   e. 段 0 的 DMDLOAD_REQ 触发 DLPC964 以开始块 1 的数据加载，使 BLKLOADZ 置为有效以指示正在进行操作。

请注意，在 GT 通道 1、2 和 3 上不会发生数据传输。在单通道模式下只会运行通道 0。

### 4.3.3.13 DMD 块阵列数据映射

对于每个 Aurora 内核，一个完整的 DMD 块阵列为 1024 列 x 136 行。表 4-25 显示了以递增方向将 192 位 Aurora 数据总线映射到一个完整 DMD 块（第一个 Aurora 数据包从第 0 行开始）。一个完整 DMD 块需要 726 个 Aurora 数据包才能发送一个完整的块。对于最后一个数据包，仅需要位 0-63，DLPC964 会忽略位 64-191。

表 4-26 下一页显示了以递减方向进行的数据映射，第一个 Aurora 数据包从第 135 行开始。

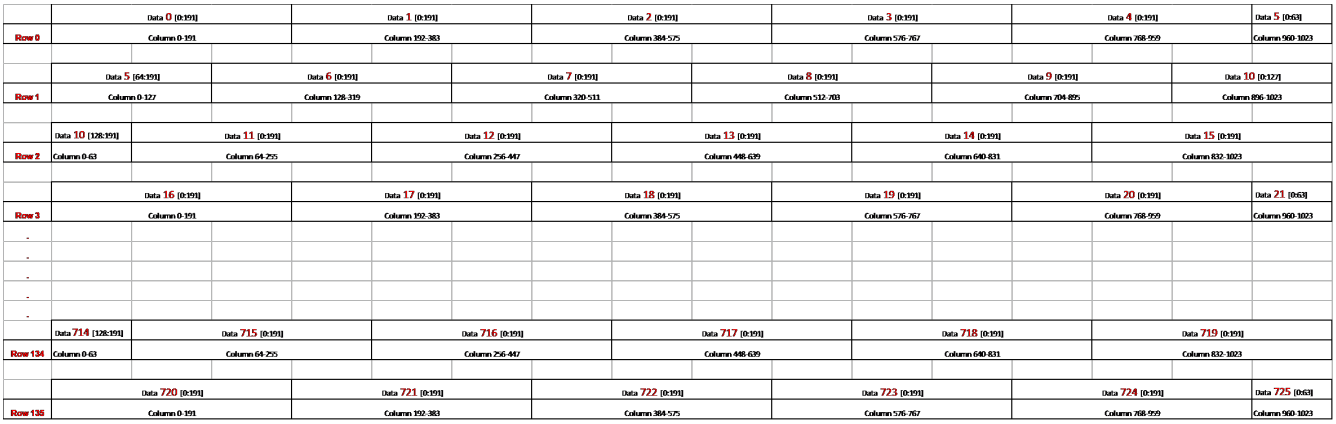



**图 4-25. Aurora 数据总线到 DMD 块阵列的映射，递增方向**

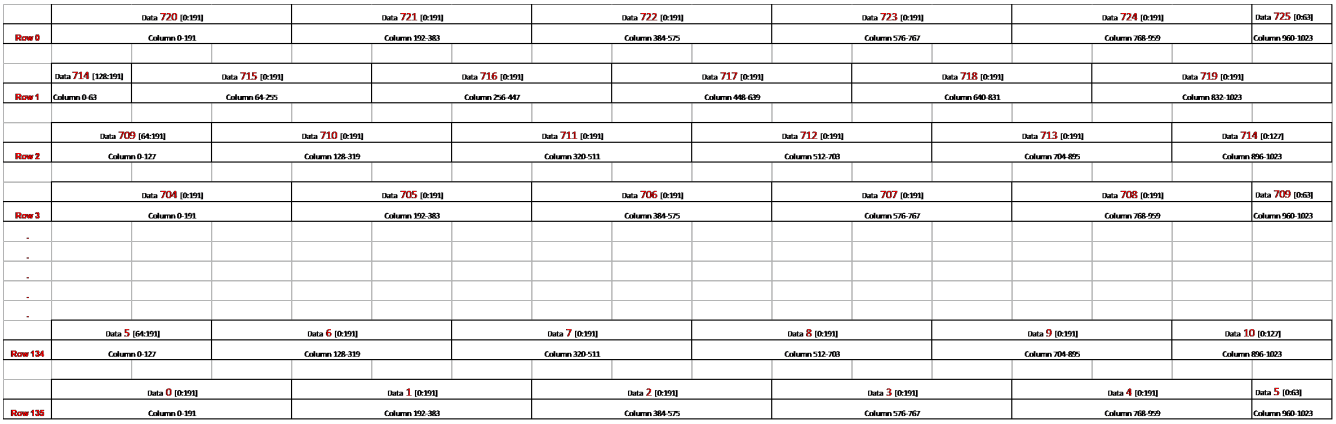



**图 4-26. Aurora 数据总线到 DMD 块阵列的映射，递减方向**

#### 4.3.3.14 Xilinx IBERT

使用 Xilinx IBERT（集成误码率测试仪）工具集可以验证 10Gbps 链路的信号完整性。如需了解 IBERT 工具的详细信息，请参阅 Xilinx 用户指南（节 6）。

如表 4-1 所示，RTL 有 4 个输入端口用于控制 TX 收发器设置。TI EVM 硬件配置如下。

**表 4-4. 用于控制 TX 收发器设置的 RTL 输入端口**

| 信号名称 | I/O 方向 | 时钟域 | 说明 |
|---|---|---|---|
| gt_txpostcursor_in[4:0] | 输入 | 异步 | 收发器后标 TX 预加重控制。对于 TI EVM 硬件，设置为“00000”。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txdiffctrl_in[3:0] | 输入 | 异步 | 收发器 TX 驱动器摆幅控制。对于 TI EVM 硬件，设置为“1000”（807mV 差分峰峰值摆幅）。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txmaincursor_in[6:0] | 输入 | 异步 | 收发器主标 TX 控制。对于 TI EVM 硬件，设置为“0000000”。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |
| gt_txprecursor_in[4:0] | 输入 | 异步 | 收发器前标 TX 预加重控制。对于 TI EVM 硬件，设置为“00000”。客户必须执行 IBERT 眼图扫描以确定硬件的最佳设置。 |

此外，DLPC964 有一个输入引脚 RXLPEN 可为 DLPC964 Xilinx GT 单元收发器选择低功耗模式（‘0') 或 DFE（‘1') 均衡。对于 TI EVM，RXLPEN 设置为 0 以实现低功耗模式均衡。如需有关 RXLPEN 的信息，请参阅 Xilinx 应用手册。

在 IBERT GUI 中选择并启用上述 RX/TX 收发器设置（TX 后标、主标、前标、TX diffctrl 和 RXLPEN）后，12 个高速链路之一的 IBERT 扫描结果如图 4-27 所示，其中眼图张开度 200+ 垂直码，水平 0.6UI - BER 1e-12（紫色区域）。

**备注**

为了模拟 64b/66B 编码特征的流量模式，用户需要在 IBERT 工具中选择占空比最高的二进制序列选项 PRBS31。

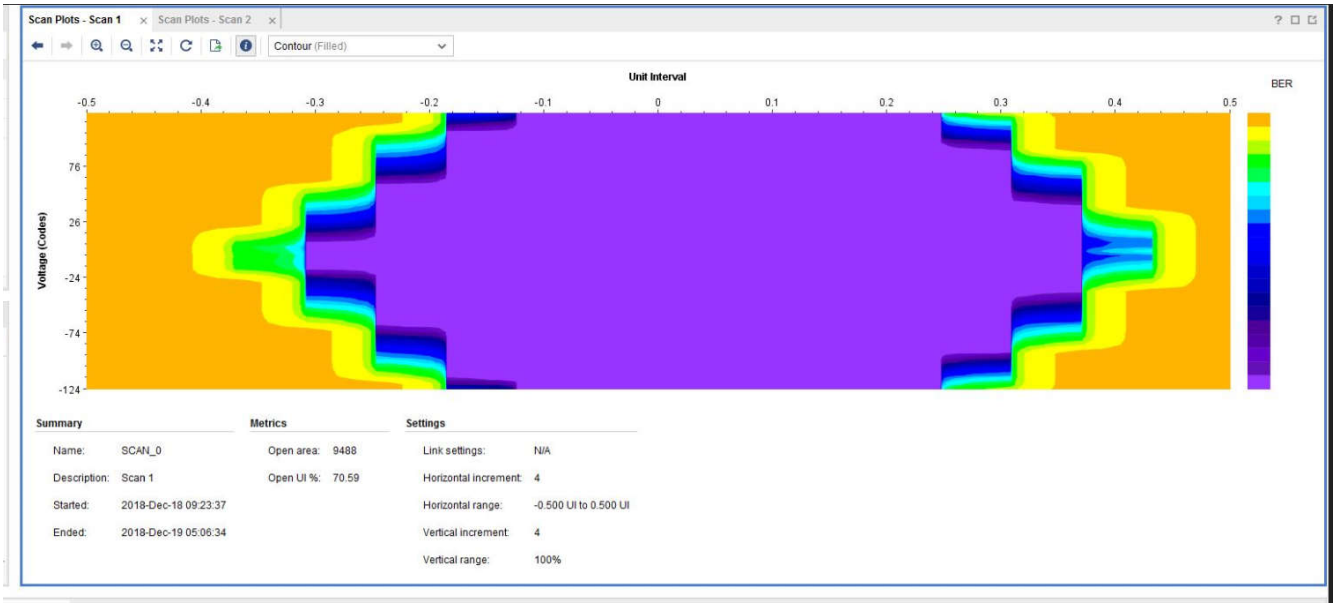



**图 4-27. 使用 TI EVM 硬件时 Aurora Channel0 Link0 的 IBERT 眼图扫描**

# 5 缩略语和首字母缩写词

下面列出了本手册中使用的缩略语和首字母缩写词：

| | |
|---|---|
| **Apps FPGA** | VC-707 EVM 或类似电路板上面向客户应用的 AMD Xilinx Virtex 7 FPGA |
| **BRG** | 块复位发生器 |
| **DMD** | 数字微镜器件 |
| **EVM** | 评估模块（电路板） |
| **FCC** | 联邦通信委员会 |
| **FPGA** | 现场可编程门阵列 |
| **FW** | 固件 |
| **GPIF** | 通用接口 |
| **GPIO** | 通用输入输出 |
| **GUI** | 图形用户界面 |
| **HSSI** | 高速串行接口 |
| **HW** | 硬件 |
| **IBERT** | 集成误码率测试仪 |
| **$I^2C$** | 内部集成电路 |
| **IP** | 互联网协议 |
| **JTAG** | 联合测试行动组 |
| **MCP** | 微镜时钟脉冲 |
| **PBC** | 处理器总线控制 |
| **PCB** | 印刷电路板 |
| **PGEN** | 图形发生器 |
| **PLL** | 锁相环 |
| **SDK** | 软件开发套件 |
| **SPI** | 串行外设接口 |
| **SW** | 开关 |
| **USB** | 通用串行总线 |
| **VHDL** | 验证和硬件描述语言 |

# 6 德州仪器 (TI) 相关文档

可以在下述链接中查看元件数据表、技术文档、设计文档和订购信息：

DLPC964 数字控制器产品文件夹

DLP LightCrafter DLPC964 EVM 工具文件夹

DLP991UFLV DMD 产品文件夹

DLP LightCrafter DLP991UFLV DMD EVM 产品文件夹

DLPC964 EVM 用户指南

Aurora 64B/66B v11.2 LogiCORE IP 产品指南

集成误码率测试仪 7 系列 GTH 收发器 v3.0

# 重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265