Logos2 系列 FPGA 专用 RAM 模块（DRM）用户指南

(UG040002, V1.1)

(2023.07.14)

深圳市紫光同创电子有限公司

文档版本修订记录

版本号	发布日期	修订记录
V0.1	2019/12/3	初始版本
V0.2	2020/4/30	修改地址和数据端口映射描述
V0.3	2021/1/26	1.修正表 1-1 中 ROM 最大位宽描述； 2.修正 GTP_FIFO36K_E1 小节表 2-6 中参数 ALMOST_FULL_OFFSET、ALMOST_EMPTY_OFFSET 的取值范围； 3.新增 GTP_FIFO18K_E1 小节； 4.读写操作小节新增注意事项：SDP 模式不可设置读写模式。 5.读写操作小节新增 36K 模式下 SP RAM（32/36/64/72 位宽）写模式描述； 6.Byte Write 模式小节新增备注 x9(x8)写数据端口模式下不支持字节使能功能。 7.新增 ECC 模式章节，文中 ECC 部分描述改到此章节，同时新增 FIFO 也支持 ECC 功能描述，更新 ECC 模式单 bit/双 bits 错误插入具体位置描述；
V1.0	2021/9/8	1.更新文档目录结构，按 DP、SDP 等模式分章节对原文档内容进行描述； 2.各模式新增应用示例章节，举例说明了各模式 DRM 的配置、级联配置、硬级联配置、ECC 模式配置； 3.更新文档内容，对原文档的部分端口名称进行统一，删改部分不易理解的描述，修正、新增部分图片，对部分内容进行补充说明； 4.更新表 1-1 的功能说明顺序，删除、更新部分说明； 5.表 1-2 新增备注，DRM 资源规模以数据手册为准； 6.新增 DRM 时序参数小节； 7.3.6 小节新增图 3-6 字节使能读写时序图（TW 模式）； 8.更新可选的输出寄存器小节名称为内部寄存器小节，新增描述及说明。 9.新增硬级联功能小节； 10.SP 模式章节读写操作小节新增 SP RAM（32/36/64/72 位宽）写模式描述； 11.FIFO 模式章节模式介绍小节新增图 7-1； 12.FIFO 模式章节读写操作小节新增第 1 小节 FIFO 时序参数； 13.FIFO 模式章节读写操作小节新增说明，本节读写时序适用于 GTP_FIFO，写入时序小节更新图 7-4 及其说明，读出时序小节更新图 7-6 及其说明，同时修正图 7-6。 14.FIFO 模式章节位宽组合小节新增了 36/18K FIFO 模式支持的位宽组合说明； 15.ECC 模式章节新增 SDP RAM 的图 8-1、图 8-2 以及图 8-3，新增 ECC 模式参数配置描述； 16.新增初始化配置参数映射小节； 17.新增 DRM 端口信号说明小节。
V1.1	2023/07/14	1.修正图 3-7 中 Latch 错误的时钟端口； 2.修正图 8-2 中 PARATY 信号错误的位宽； 3.删除地址和数据端口 Mapping 章节中内容和后续表格中描述重复的地址映射描述表格，更新统一表 9-1、表 9-2； 4.修正读写操作小节错误的读写时序描述； 5.修正文中引用的参考文档《28nm DRM RAM/FIFO IP 用户指南》名称为“DRM/FIFO IP 用户指南 UG041002”； 6.修正表 2-3 中 DIB 信号的端口位宽由[35:0]修正为[17:0]； 7.删除表 2-2、表 2-4、表 2-6、表 2-8 对 GTP 参数默认值的描述； 8.修正表 2-2 中参数 ECC_WRITE_EN、ECC_READ_EN 错误的设置值描述； 9.图 3-2、图 3-3、图 3-4、图 3-5、图 3-6、图 3-10、图 4-2、图 4-3 新增说明，图中 Mem 为对应地址存储的旧数据； 10.修正表 3-1 中错误的端口名称 ADRDB、ADRDB_HOLD 为 ADDRB、

版本号	发布日期	修订记录
		ADDRB_HOLD； 11.修正图 3-3、图 3-4、图 3-5 中最后一个 Mem(ADDR0)为 D0； 12.修正表 2-8 中参数 DATA_WIDTH 设置值范围。

术语与缩略语

Abbreviations 术语与缩略语	Full Spelling 英文全拼	Chinese Explanation 中文解释
DRM	Dedicated RAM Module	专用 RAM 模块
SP	Single Port	单口
SDP	Simple Dual Port	简单双口
DP	True Dual Port	双口
FIFO	First In First Out	先进先出存储器
GTP	Generic Technology Primitive	通用技术原语
NW	Normal-Write	普通写
RBW	Read-before-Write	先读后写
TW	Transparent-Write	透传写
OR	Output Register	输出寄存器
IR	Input Register	输入寄存器
ECC	Error Correcting Code	纠错码

目录

表目录

图目录

1 总体介绍

Logos2系列FPGA的DRM有高达36K bits的存储单元并且容量可被独立配置为2个18K bits或者1个36K bits。每个DRM都能支持DP（True Dual Port，双口）RAM模式，同时也可以被配置为SP（Single Port，单口）RAM模式，SDP（Simple Dual Port，简单双口）RAM模式，ROM模式，以及同步/异步FIFO（First In First Out）模式。DRM资源还支持输入寄存器（IR）、输出寄存器（OR），这使得DRM级联使用时拥有更加出色的性能表现。DRM的总数取决于Logos2系列器件类型。

嵌入的DP RAM、SP RAM、SDP RAM、ROM以及同步/异步FIFO模块的IP，都能便捷地通过深圳市紫光同创电子有限公司的软件Pango Design Suite内嵌的IP Compiler工具生成。

1.1 功能特性列表

如表 1-1所示，为Logos2系列FPGA的DRM功能特性列表。

表 1-1 功能特性列表

功能	说明
存储容量	1 个 36K 或拆分为 2 个 18K，可拼成最大 72bit 位宽
DP RAM 模式	最大 36 位数据宽度
SDP RAM 模式	最大 72 位数据宽度
SP RAM 模式	最大 72 位数据宽度
Rom 模式	最大 72 位数据宽度
FIFO 模式	异步及同步模式
写模式	DP 及 SP 支持 Normal-Write，Transparent-Write 及 Read-before-Write 写模式
Byte-Write 字节写使能	支持
可选的输出寄存器	支持
硬级联	两个相邻 36K 块级联支持 64Kx1（两个端口数据位宽相同）
ECC	仅在 36K(SDP/FIFO)512x72 模式下支持单 bit 纠错双 bits 检错
输出寄存器同步/异步复位	支持

注：SP 模式下 32 bit 及以上数据位宽禁止设置读写模式为 TW、RBW，需要设置为默认的 NW，详见 5.5 读写操作小节。

1.2 资源规模

表 1-2 Logos2 资源规模

器件	资源数量/36Kbits
PG2L25H	55
PG2L50H	85

器件	资源数量/36Kbits
PG2L100H	155
PG2L200H	415

注：Logos2 系列 FPGA 的 DRM 资源规模以 ***《DS04001_Logos2 系列 FPGA 器件数据手册》*** 为准。

1.3 DRM 支持的模式

每个DRM都能支持DP（True Dual Port，双口）RAM模式，同时也可以被配置为SP（Single Port，单口）RAM模式，SDP（Simple Dual Port，简单双口）RAM模式，ROM模式，以及同步/异步FIFO（First In First Out）模式。

DRM 的端口位宽支持两种类型：一种是数据位宽为2^N（包括 1/2/4/8/16/32/64bit）。另一种是数据位宽为9×2^N（包括 9/18/36/72/bit）。DP RAM 和 SDP RAM 模式还支持混合数据位宽功能，即两个端口可以配置成不同位宽。例如，一个 SDP RAM，可以在写端口配置成 16Kx1，在读端口配置成 512x32，从而节约了从 1 比特到 32 比特的串并转换逻辑。

DP RAM和SDP RAM均支持36K模式和18K模式。在DP RAM模式下，DRM有36-bit的数据最大位宽，DRM模块中A、B两个端口均可以独立进行读写操作，且均支持不同的时钟。而SDP RAM模式下，DRM数据位宽则增大至72 bits，A，B两个端口中一个端口专用于数据写入，另一个端口专用于数据读取，读写端口同样均支持不同的时钟。

36K和18K DRM模式下的DP RAM允许的位宽组合，36K和18K DRM模式下的SDP RAM允许的位宽组合见表 3-2、表 3-3，以及表 4-2、表 4-3中描述。

在 SP RAM 和 ROM 两种模式的 18K 模式下，DRM 包含两个端口，SP RAM 模式可以分别对这两个端口独立进行读写操作，而 ROM 模式下该端口只读；在两对端口共享情况下，DRM 仅包含一个端口。36K 和 18K DRM 模式下的 SP RAM 允许的位宽组合，36K 和 18K DRM 模式下的 ROM 允许的位宽组合，见表 5-2、表 5-3 以及表 6-2、表 6-3 中描述。

FIFO模式下，一个端口专用于数据写入另一个端口专用于数据读取，读写端口可以采用不同的时钟。36K模式下的FIFO模式配置，以及18K模式下FIFO模式配置见表 7-3和表 7-4中描述。

在SDP/FIFO 512x72存储器模式下，支持72bits宽度数据ECC纠错，其中有效数据位为64bit，另外的8bits为ECC校验位存储。ECC模式下支持单bit纠错、双bits检错功能，同时输出ECC_SBITERR（单bit纠错指示标志）、ECC_DBITERR（双bits检错指示标志）、ECC编码以及ECC模式下的读地址（ECC编码、读地址仅在SDP模式下支持）输出。

多个DRM可以通过级联扩展的方式组合成更大的DP RAM，SDP RAM, SP RAM，ROM或者FIFO。对此，GTP_DRM提供额外的3 bit地址扩展（CS[2:0]），常用于深度扩展的应用。

地址和数据端口Mapping详见附录9.1小节。

2 GTP 说明

GTP（Generic Technology Primitive，通用技术原语）在设计中可以直接例化，Logos2 系列 FPGA 支持的 GTP 及其使用说明详见***《UG040007_Logos2 系列产品 GTP 用户指南》***，具体各 GTP 的支持情况、端口、参数等请以 GTP 用户指南为准。

Logos2 系列 FPGA 的 DRM 相关 GTP 是 DRM 各种模式的基础，用户可通过 DRM 的 GTP 来配置实现不同模式、功能的 DRM 模块。Logos2 系列 FPGA 支持的 DRM 相关 GTP 共有四种：GTP_DRM36K_E1、GTP_DRM18K_E1 和 GTP_FIFO36K_E1、GTP_FIFO18K_E1。

2.1 GTP_DRM36K_E1

例化 GTP_DRM36K_E1 可实现 DP、SDP、SP、ROM 模式。GTP 结构框图见图 2-1。端口描述及参数描述见表 2-1，表 2-2。GTP_DRM36K_E1 例化模板见附录。

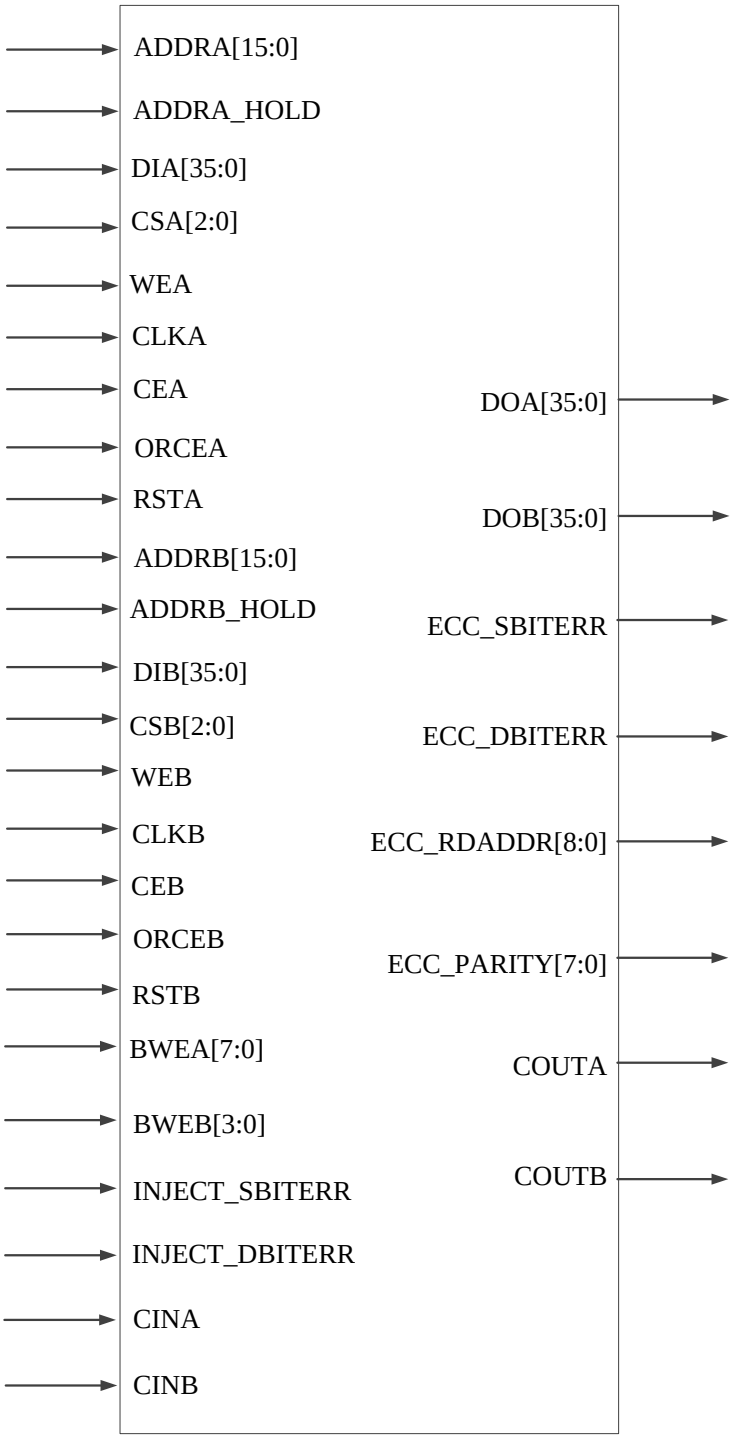

图 2-1 GTP_DRM36K_E1 结构图

表 2-1 GTP_DRM36K_E1 端口说明

端口命名	输入/输出	描述
ADDRA[15:0]	输入	A 端口输入地址
ADDRA_HOLD	输入	A 端口地址输入选择
DIA[35:0]	输入	A 端口数据输入
CSA[2:0]	输入	A 端口地址扩展
WEA	输入	A 端口写使能
BWEA[7:0]	输入	A 端口字节写使能
CLKA	输入	A 端口时钟
CEA	输入	A 端口输入寄存器时钟使能
ORCEA	输入	A 端口输出寄存器时钟使能
RSTA	输入	A 端口数据寄存器复位
DOA[35:0]	输出	A 端口数据输出
ADDRB[15:0]	输入	B 端口输入地址
ADDRB_HOLD	输入	B 端口地址输入选择
DIB[35:0]	输入	B 端口数据输入
CSB[2:0]	输入	B 端口地址扩展
WEB	输入	B 端口写使能
BWEB[3:0]	输入	B 端口字节写使能
CLKB	输入	B 端口时钟
CEB	输入	B 端口输入寄存器时钟使能
ORCEB	输入	B 端口输出寄存器时钟使能
RSTB	输入	B 端口数据寄存器复位
CINA	输入	64Kx1 模式下做深度级联用，相邻 DRM A 端口数据输出级联输入
CINB	输入	64Kx1 模式下做深度级联用，相邻 DRM B 端口数据输出级联输入
INJECT_SBITERR	输入	ECC 模式单 bit 错误插入
INJECT_DBITERR	输入	ECC 模式双 bits 错误插入
DOB[35:0]	输出	B 端口数据输出
COUTA	输出	64Kx1 模式下做深度级联用，相邻 DRM A 端口数据输出级联输出
COUTB	输出	64Kx1 模式下做深度级联用，相邻 DRM B 端口数据输出级联输出
ECC_SBITERR	输出	ECC 模式单 bit 错误标志
ECC_DBITERR	输出	ECC 模式双 bits 错误标志
ECC_PARITY[7:0]	输出	ECC 编码校验位输出
ECC_RDADDR[8:0]	输出	ECC 解码读地址输出

注：DRM 详细端口使用说明见 9.3 DRM 端口信号说明。

表 2-2 GTP_DRM36K_E1 参数说明

参数名	描述	设置值
CSA_MASK[2:0]	A 端口地址扩展控制信号	3’b000 ~ 3’b111
CSB_MASK[2:0]	B 端口地址扩展控制信号	3’b000 ~ 3’b111
DATA_WIDTH_A	A 端口数据最大位宽	1、2、4、8、16、32、9、18、36、64、72
DATA_WIDTH_B	B 端口数据最大位宽	1、2、4、8、16、32、9、18、36、64、72
WRITE_MODE_A	A 端口写模式	"NORMAL_WRITE" "TRANSPARENT_WRITE" "READ_BEFORE_WRITE"
WRITE_MODE_B	B 端口写模式	"NORMAL_WRITE" "TRANSPARENT_WRITE" "READ_BEFORE_WRITE"
DOA_REG	A 端口输出寄存	0：不使能输出寄存 1：使能输出寄存
DOB_REG	B 端口输出寄存	0：不使能输出寄存 1：使能输出寄存
RST_TYPE	复位模式选择	"SYNC"：同步复位 "ASYNC"：异步复位
RAM_MODE	RAM 模式选择	"TRUE_DUAL_PORT"：双口 RAM "SIMPLE_DUAL_PORT"：简单双口 RAM "SINGLE_PORT"：单口 RAM "ROM"：ROM
GRS_EN	全局复位使能信号（芯片内部）	"FALSE":不使能全局复位； "TRUE":使能全局复位。
DOA_REG_CLKINV	A 端口输出寄存器时钟翻转	0：时钟不翻转 1：时钟翻转
DOB_REG_CLKINV	B 端口输出寄存器时钟翻转	0：时钟不翻转 1：时钟翻转
RSTA_VAL	A 端口输出置位/复位值	36'h0_0000_0000~36'hF_FFFF_FFFF
RSTB_VAL	B 端口输出置位/复位值	36'h0_0000_0000~36'hF_FFFF_FFFF
RAM_CASCADE	64Kx1 硬级联模式	"NONE": 不进行硬级联 "UPPER": 作为硬级联数据输出模块 "LOWER": 作为硬级联附加模块
ECC_WRITE_EN	ECC 写模式使能	"FALSE": 不使能 "TRUE":使能
ECC_READ_EN	ECC 读模式使能	"FALSE": 不使能 "TRUE":使能
INIT_00 INIT_01 INIT_02 … INIT_7F	RAM 初始化配置参数，详细请参考 9.2 初始化配置参数映射	288'b0 ~2^288-1
INIT_FILE	初始化文件路径，该参数只在行为仿真中生效，综合生成网表时 RAM 的初始化数据来源于初始化配置参数 INIT_XX	"NONE"：不指定初始化文件，则初始化数据为参数 INIT_XX 设置的值； "XXX"：XXX 表示具体初始化文件路径，行为仿真中覆盖参数 INIT_XX 作为 DRM 的初始值
BLOCK_X	DRM36K 级联时数据级联坐标，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 坐标而定

参数名	描述	设置值
BLOCK_Y	DRM36K 级联时地址级联坐标，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 坐标而定
RAM_DATA_WIDTH	DRM 级联后的数据最大位宽，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 个数而定
RAM_ADDR_WIDTH	DRM 级联后的地址最大位宽，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 个数而定
INIT_FORMAT	初始化文件格式	“BIN”：二进制 “HEX”：十六进制

注：SDP 模式不支持读写模式设置，使用 GTP 配置 DRM 为 SDP 模式时，不能手动修改设置读写模式参数 WRITE_MODE_A/B。

2.2 GTP_DRM18K_E1

例化 GTP_DRM18K_E1 可实现 DP、SDP、SP、ROM 模式。GTP 结构框图见图 2-2。端口描述及参数描述见表 2-3，表 2-4。GTP_DRM18K_E1 例化模板见附录。

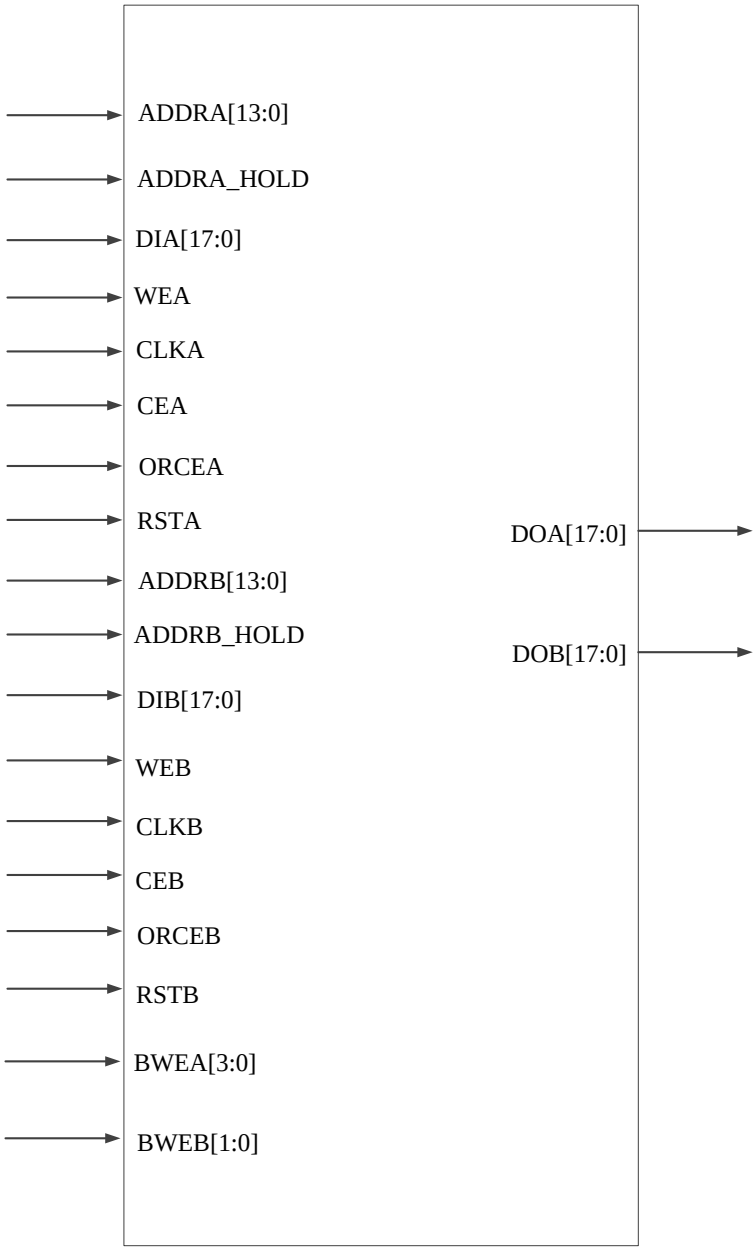

图 2-2 GTP_DRM18K_E1 结构图

表 2-3 GTP_DRM18K_E1 端口说明

端口命名	输入/输出	描述
ADDRA[13:0]	输入	A 端口输入地址
ADDRA_HOLD	输入	A 端口地址输入选择
DIA[17:0]	输入	A 端口数据输入
WEA	输入	A 端口写使能

端口命名	输入/输出	描述
BWEA[3:0]	输入	A 端口字节写使能
CLKA	输入	A 端口时钟
CEA	输入	A 端口输入寄存器时钟使能
ORCEA	输入	A 端口输出寄存器时钟使能
RSTA	输入	A 端口数据寄存器复位
DOA[17:0]	输出	A 端口数据输出
ADDRB[13:0]	输入	B 端口输入地址
ADDRB_HOLD	输入	B 端口地址输入选择
DIB[17:0]	输入	B 端口数据输入
WEB	输入	B 端口写使能
BWEB[1:0]	输入	B 端口字节写使能
CLKB	输入	B 端口时钟
CEB	输入	B 端口输入寄存器时钟使能
ORCEB	输入	B 端口输出寄存器时钟使能
RSTB	输入	B 端口数据寄存器复位
DOB[17:0]	输出	B 端口数据输出

注：DRM 详细端口使用说明见 9.3 DRM 端口信号说明。

表 2-4 GTP_DRM18K_E1 参数说明

参数名	描述	设置值
DATA_WIDTH_A	A 端口数据最大位宽	1、2、4、8、16、32、9、18、36
DATA_WIDTH_B	B 端口数据最大位宽	1、2、4、8、16、32、9、18、36
WRITE_MODE_A	A 端口写模式	"NORMAL_WRITE" "TRANSPARENT_WRITE" "READ_BEFORE_WRITE"
WRITE_MODE_B	B 端口写模式	"NORMAL_WRITE" "TRANSPARENT_WRITE" "READ_BEFORE_WRITE"
DOA_REG	A 端口输出寄存	0：不使能输出寄存 1：使能输出寄存
DOB_REG	B 端口输出寄存	0：不使能输出寄存 1：使能输出寄存
RST_TYPE	复位模式选择	"SYNC"：同步复位 "ASYNC"：异步复位
RAM_MODE	RAM 模式选择	"TRUE_DUAL_PORT"：双口 RAM "SIMPLE_DUAL_PORT"：简单双口 RAM "SINGLE_PORT"：单口 RAM "ROM"：ROM
GRS_EN	全局复位使能信号（芯片内部）	"FALSE":不使能全局复位； "TRUE":使能全局复位。
DOA_REG_CLKINV	A 端口输出寄存器时钟翻转	0：时钟不翻转 1：时钟翻转
DOB_REG_CLKINV	B 端口输出寄存器时钟翻转	0：时钟不翻转 1：时钟翻转
RSTA_VAL	A 端口输出置位/复位值	18'h0_0000~18'h3_FFFF

参数名	描述	设置值
RSTB_VAL	B 端口输出置位/复位值	18'h0_0000~18'h3_FFFF
INIT_00 INIT_01 INIT_02 … INIT_3F	RAM 初始化配置参数，详细请参考 9.2 初始化配置参数映射	288'b0 ~2^288-1
INIT_FILE	初始化文件路径，该参数只在行为仿真中生效，综合生成网表时 RAM 的初始化数据来源于初始化配置参数 INIT_XX	"NONE"：不指定初始化文件，则初始化数据为参数 INIT_XX 设置的值； "XXX"：XXX 表示具体初始化文件路径，行为仿真中覆盖参数 INIT_XX 作为 DRM 的初始值
BLOCK_X	DRM18K 级联时数据级联坐标，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 坐标而定
BLOCK_Y	DRM18K 级联时数据级联坐标，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 坐标而定
RAM_DATA_WIDTH	DRM 级联后的数据最大位宽，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 个数而定
RAM_ADDR_WIDTH	DRM 级联后的地址最大位宽，用于将同一个初始化文件映射到不同的级联 DRM 中	视级联 DRM 个数而定
INIT_FORMAT	初始化文件格式	"BIN"：二进制 "HEX"：十六进制

注：SDP 模式不支持读写模式设置，使用 GTP 配置 DRM 为 SDP 模式时，不能手动修改设置读写模式参数 WRITE_MODE_A/B。

2.3 GTP_FIFO36K_E1

例化 GTP_FIFO36K_E1 可实现 FIFO 模式。GTP 结构框图见图 2-3。端口描述及参数描述见表 2-5，表 2-6。GTP_FIFO36K_E1 例化模板见附录。

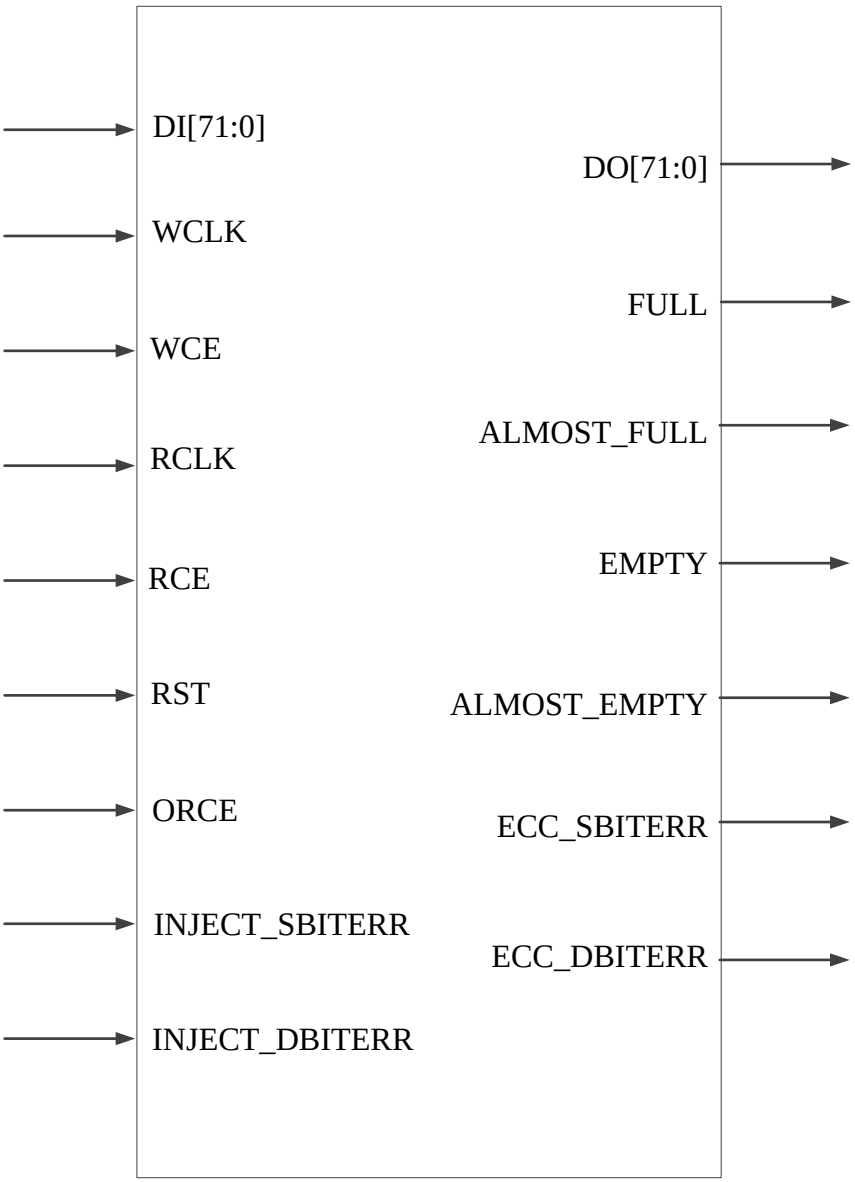

图 2-3 GTP_FIFO36K_E1 结构图

表 2-5 GTP_FIFO36K_E1 端口说明

端口命名	输入/输出	描述
DI[71:0]	输入	数据写入
WCLK	输入	写时钟信号
RCLK	输入	读时钟信号
WCE	输入	写使能信号
RCE	输入	读使能信号

端口命名	输入/输出	描述
RST	输入	复位信号
ORCE	输入	输出寄存器时钟使能信号
INJECT_SBITERR	输入	ECC 模式单 bit 错误插入
INJECT_DBITERR	输入	ECC 模式双 bits 错误插入
DO[71:0]	输出	数据读出
EMPTY	输出	读端口读空标志
FULL	输出	写端口写满标志
ALMOST_EMPTY	输出	读端口将空标志
ALMOST_FULL	输出	写端口将满标志
ECC_SBITERR	输出	ECC 模式单 bit 错误标志
ECC_DBITERR	输出	ECC 模式双 bits 错误标志

表 2-6 GTP_FIFO36K_E1 参数说明

参数名	描述	设置值
GRS_EN	全局复位使能信号（芯片内部）	"TRUE":使能全局复位； "FALSE":不使能全局复位。
DATA_WIDTH	FIFO 数据位宽	1、2、4、8、9、16、18、32、36、64、72
SYNC_FIFO	异步/同步 FIFO 选择	"TRUE":使用同步 FIFO； "FALSE":使用异步 FIFO.
ALMOST_FULL_OFFSET	当 FIFO 将满且写指针与读指针位置之差等于 ALMOST_FULL_OFFSET，产生 almost_full 标志。	DATA_WIDTH= 1bit: sync FIFO :1~32767 async FIFO :1~32764 2bits: sync FIFO :1~16383 async FIFO:1~16380 4bits: sync FIFO :1~8191 async FIFO:1~8188 8/9bits: sync FIFO :1~4095 async FIFO: 1~4092 16/18bits: sync FIFO :1~2047 async FIFO :1~2044 32/36bits: sync FIFO : 1~1023 async FIFO : 1~1020 64/72bits: sync FIFO : 1~511 async FIFO : 1~508

参数名	描述	设置值
ALMOST_EMPTY_OFFSET	当 FIFO 将空且读指针与写指针位置之差值等于 ALMOST_EMPTY_OFFSET,产生 almost_empty 标志。	DATA_WIDTH= 1bit: sync FIFO :1~32767 async FIFO :4~32767 2bits: sync FIFO :1~16383 async FIFO: 4~16383 4bits: sync FIFO :1~8191 async FIFO: 4~8191 8/9bits: sync FIFO :1~4095 async FIFO: 4~4095 16/18bits: sync FIFO :1~2047 async FIFO : 4~2047 32/36bits: sync FIFO : 1~1023 async FIFO : 4~1023 64/72bits: sync FIFO : 1~511 async FIFO : 4~511
ECC_WRITE_EN	写端口 ECC 模式使能	"TRUE":使能； "FALSE":不使能。
ECC_READ_EN	读端口 ECC 模式使能	"TRUE":使能； "FALSE":不使能。
RST_VAL	输出置位/复位值	72'h00_0000_0000_0000_0000~ 72'hFF_FFFF_FFFF_FFFF_FFFF
USE_EMPTY	使能读空标志	1:使能读空标志； 0:不使能读空标志。
USE_FULL	使能写满标志	1:使能写满标志； 0:不使能写满标志。
DO_REG	使能输出寄存器使能	1:使能； 0:不使能

2.4 GTP_FIFO18K_E1

例化 GTP_FIFO18K_E1 可实现 FIFO 模式。GTP 结构框图见图 2-4。端口描述及参数描述见表 2-7，表 2-8。GTP_FIFO18K_E1 例化模板见附录。

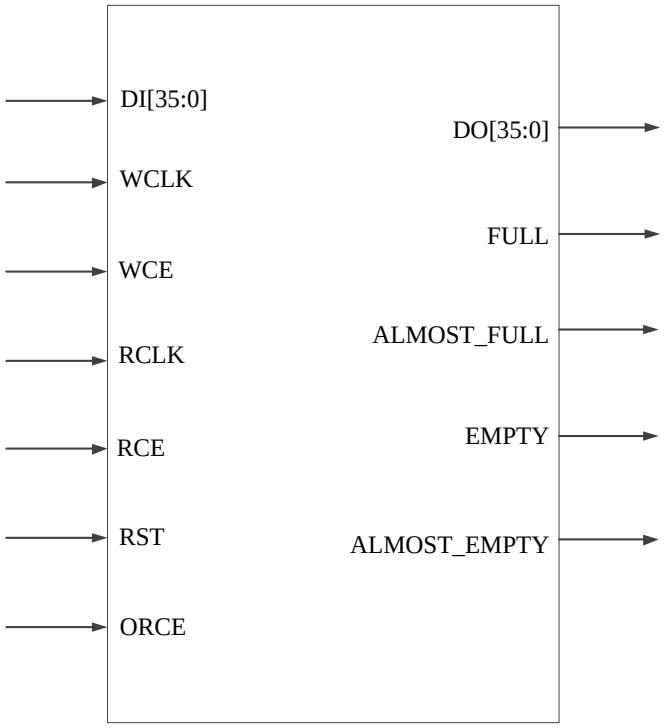

图 2-4 GTP_FIFO18K_E1 结构图

表 2-7 GTP_FIFO18K_E1 端口说明

端口命名	输入/输出	描述
DI[35:0]	输入	数据写入
WCLK	输入	写时钟信号
RCLK	输入	读时钟信号
WCE	输入	写使能信号
RCE	输入	读使能信号
RST	输入	复位信号
ORCE	输入	输出寄存器时钟使能信号
DO[35:0]	输出	数据读出
EMPTY	输出	读端口读空标志
FULL	输出	写端口写满标志
ALMOST_EMPTY	输出	读端口将空标志
ALMOST_FULL	输出	写端口将满标志

表 2-8 GTP_FIFO18K_E1 参数说明

参数名	描述	设置值
GRS_EN	全局复位使能信号（芯片内部）	"TRUE":使能全局复位； "FALSE":不使能全局复位。
DATA_WIDTH	FIFO 数据位宽	1,2,4,8,9,16,18,32,36
SYNC_FIFO	异步/同步 FIFO 选择	"TRUE":使用同步 FIFO； "FALSE":使用异步 FIFO.
ALMOST_FULL_OFFSET	当 FIFO 将满且写指针与读指针位置之差值等于 ALMOST_FULL_OFFSET，产生 almost_full 标志。	DATA_WIDTH= 1bit: sync FIFO :1~16383 async FIFO:1~16380 2bits: sync FIFO :1~8191 async FIFO:1~8188 4bits: sync FIFO :1~4095 async FIFO: 1~4092 8/9bits: sync FIFO :1~2047 async FIFO :1~2044 16/18bits: sync FIFO : 1~1023 async FIFO : 1~1020 32/36bits: sync FIFO : 1~511 async FIFO : 1~508
ALMOST_EMPTY_OFFSET	当 FIFO 将空且读指针与写指针位置之差值等于 ALMOST_EMPTY_OFFSET, 产生 almost_empty 标志。	DATA_WIDTH= 1bit: sync FIFO :1~16383 async FIFO: 4~16383 2bits: sync FIFO :1~8191 async FIFO: 4~8191 4bits: sync FIFO :1~4095 async FIFO: 4~4095 8/9bits: sync FIFO :1~2047 async FIFO : 4~2047 16/18bits: sync FIFO : 1~1023 async FIFO : 4~1023 32/36bits: sync FIFO : 1~511 async FIFO : 4~511
RST_VAL	输出置位/复位值	36’h0 – 36’hF_FFFF_FFFF
USE_EMPTY	使能读空标志	1:使能读空标志； 0:不使能读空标志。
USE_FULL	使能写满标志	1:使能写满标志； 0:不使能写满标志。
DO_REG	使能输出寄存器使能	1:使能； 0:不使能

3 DP 模式

3.1 模式介绍

RAM 的端口模式由参数 RAM_MODE 决定，当参数 RAM_MODE 的值为"TRUE_DUAL_PORT"时，RAM 进入双口模式。本文中将详细介绍 36K 的 DP RAM，18K 与 36K 结构与功能基本一致，仅位宽不同。

DP RAM 支持：

- 双口读操作
- 双口写操作
- A/B 端口读，B/A 端口写操作
- 两端口有独立的位宽设置

3.2 数据端口

DP RAM 模式下，DP RAM 拥有除了共享 DRM 内容之外完全独立的两个端口：A 端口和 B 端口，它们的结构完全对称。图 3-1 展示了 36K DP RAM 的结构图。

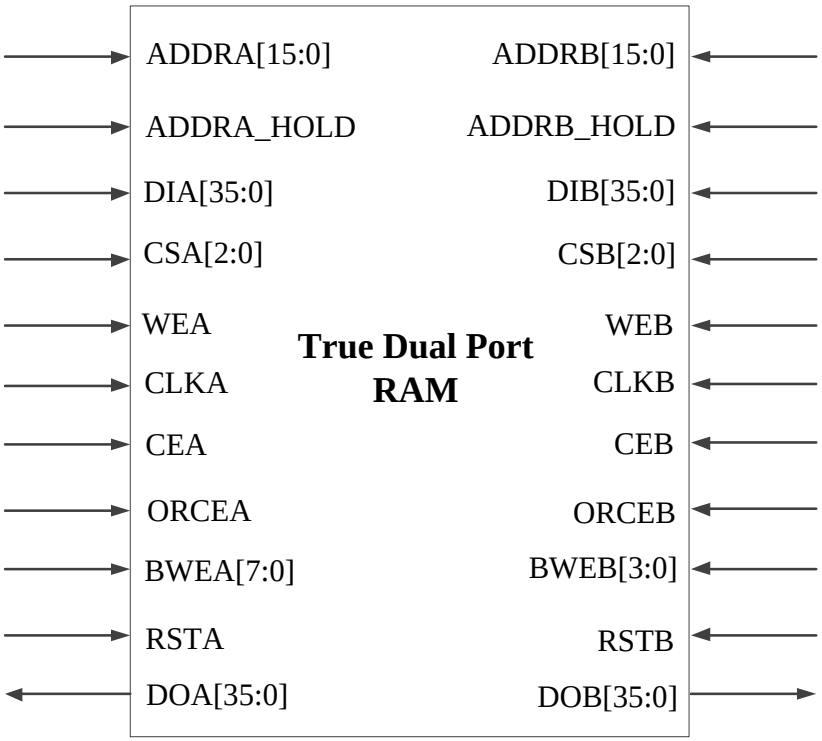

图 3-1 DP RAM 数据端口

表 3-1 罗列了 DP RAM 模式下的端口名称和端口的描述。

表 3-1 DP RAM 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
ADDRA	输入	A 端口地址输入	ADDRB	输入	B 端口地址输入
ADDRA_HOLD	输入	A 端口地址锁存信号	ADDRB_HOLD	输入	B 端口地址锁存信号
DIA	输入	A 端口数据输入	DIB	输入	B 端口数据输入
CSA	输入	A 端口地址扩展	CSB	输入	B 端口地址扩展
WEA	输入	A 端口写使能	WEB	输入	B 端口写使能
CLKA	输入	A 端口时钟	CLKB	输入	B 端口时钟
CEA	输入	A 端口输入寄存器时钟使能	CEB	输入	B 端口输入寄存器时钟使能
ORCEA	输入	A 端口输出寄存器时钟使能	ORCEB	输入	B 端口输出寄存器时钟使能
RSTA	输入	A 端口输出寄存器复位	RSTB	输入	B 端口输出寄存器复位
BWEA	输入	A 端口字节使能信号	BWEB	输入	B 端口字节使能信号
DOA	输出	A 端口数据输出	DOB	输出	B 端口数据输出

注：18K DRM 无 CSA/CSB 端口，不支持地址扩展，如需 DRM 级联进行地址深度扩展，建议使用 36K DRM。

3.3 位宽组合

RAM 的端口位宽由 GTP 中的参数 DATA_WIDTH_A/DATA_WIDTH_B 决定，例如，当参数 DATA_WIDTH_A 的值为 4 时，A 端口数据位宽被设置为 4 bit。DP 模式支持 A、B 端口设置不同数据位宽。

如表 3-2 所示为 36K DRM 模式下 True Dual Port RAM 模式允许的位宽组合。

表 3-2 36K DRM 模式下 True Dual Port RAM 模式允许的位宽组合

		B 端口								
		32Kx1	**16Kx2**	**8Kx4**	**4Kx8**	**2Kx16**	**1Kx32**	**4Kx9**	**2Kx18**	**1Kx36**
A 端口	**32Kx1**	√	√	√	√	√	√			
	16Kx2	√	√	√	√	√	√			
	8Kx4	√	√	√	√	√	√			
	4Kx8	√	√	√	√	√	√			
	2Kx16	√	√	√	√	√	√			
	1Kx32	√	√	√	√	√	√			
	4Kx9							√	√	√
	2Kx18							√	√	√
	1Kx36							√	√	√

注：√表示支持的位宽组合。

如表 3-3 所示为 18K DRM 模式下 True Dual Port RAM 模式允许的位宽组合。

表 3-3 18K DRM 模式下 True Dual Port RAM 模式允许的位宽组合

		B 端口 0/1						
		16Kx1	**8Kx2**	**4Kx4**	**2Kx8**	**1Kx16**	**2Kx9**	**1Kx18**
A 端口 0/1	**16Kx1**	√	√	√	√	√		
	8Kx2	√	√	√	√	√		
	4Kx4	√	√	√	√	√		
	2Kx8	√	√	√	√	√		
	1Kx16	√	√	√	√	√		
	2Kx9						√	√
	1Kx18						√	√

注：√表示支持的位宽组合。

3.4 时序参数

表 3-1 为 36K DRM 典型时序参数说明，后文所描述的时序参数及时序图都以 36K DRM 为例，18K DRM 的时序参数和时序图与之类似，详细的时序参数说明见***《DS04001_Logos2 系列 FPGA 器件数据手册》***。

表 3-4 DRM 典型时序参数

参数	控制信号	说明
$T_{su\_36K\_ce}$	CEA/B	时钟使能信号建立时间
$T_{hd\_36K\_ce}$		时钟使能信号保持时间
$T_{su\_36K\_we}$	WEA/B	写使能信号建立时间
$T_{hd\_36K\_we}$		写使能信号保持时间
$T_{su\_36K\_be}$	BEA/B	字节写使能信号建立时间
$T_{hd\_36K\_be}$		字节写使能信号保持时间
$T_{su\_36K\_ad}$	ADA/B	地址输入信号建立时间
$T_{hd\_36K\_ad}$		地址输入信号保持时间
$T_{su\_36K\_d}$	DA/B	数据输入信号建立时间
$T_{hd\_36K\_d}$		数据输入信号保持时间
$T_{su\_36K\_oe}$	OCEA/B	输出寄存使能信号建立时间
$T_{hd\_36K\_oe}$		输出寄存使能信号保持时间
$T_{su\_36K\_rst}$	RSTA/B	输出寄存/锁存同步复位信号建立时间
$T_{hd\_36K\_rst}$		输出寄存/锁存同步复位信号保持时间
T_{co_36K}	CLK to Q	数据输出相对时钟沿的延时（输出无寄存）
$T_{co\_36K\_reg}$		数据输出相对时钟沿的延时（输出寄存）

图 3-2 为 DRM 的时序图，图中写时序以 A 端口 TW 写模式为例，B 端口写模式与之相

同，其他写模式写时序详见 3.5 读写操作，不同写模式下，读时序是类似的。复位时序以异步复位为例，使用同步复位时，复位信号需要满足建立时间和保持时间要求。图中 Mem 为对应地址存储的旧数据。

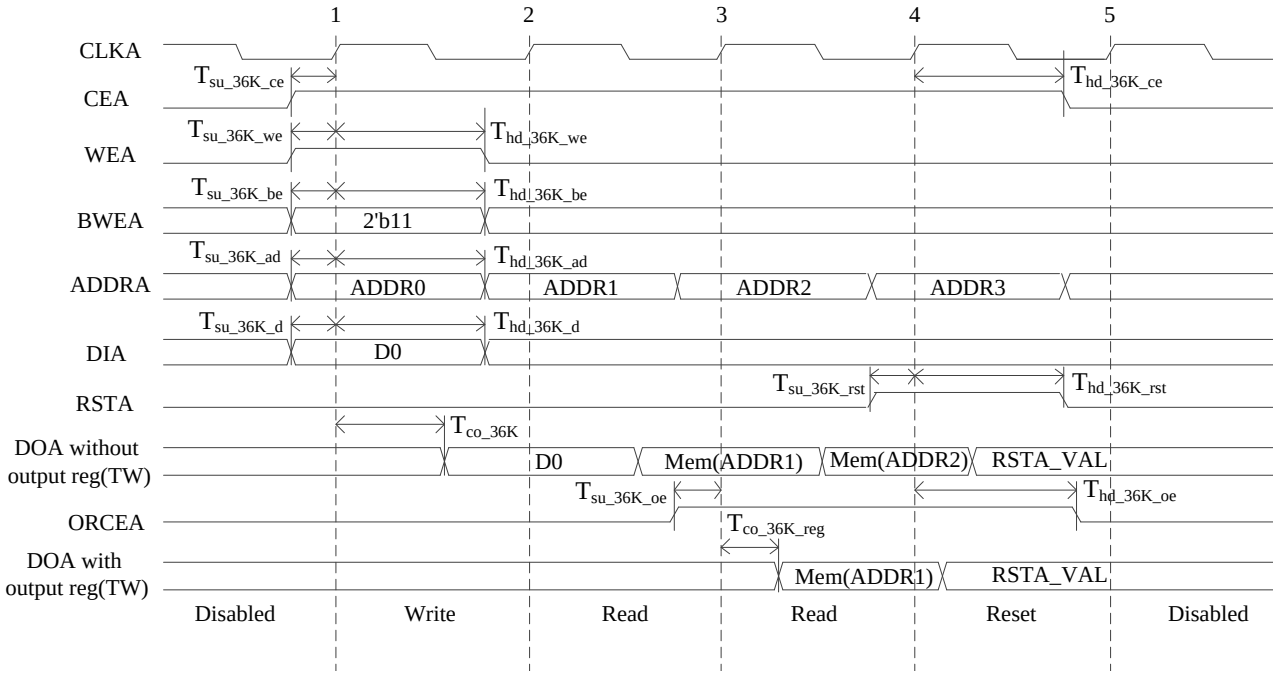

图 3-2 DRM 时序图（DP 模式 A 端口 TW 写模式）

第 1 个时钟上升沿之前，CEA、WEA 和 BWEA 拉高并满足各自建立时间要求，CSA、ADDRA、DIA 信号在达到稳定状态并满足建立时间要求；DRM 正确采样到控制信号、数据及地址信号后，将输入数据 D0 写入地址 ADDR0 中，D0 在经过输出延迟 T_{co_36K} 后，在同一周期出现在输出端口 DOA；

第 2 个时钟上升沿之前，WEA 和 BWEA 在满足保持时间要求后拉低，CEA 仍然保持高，ADDRA 达到稳定状态并满足建立时间要求；DRM 正确采样到控制信号、数据及地址信号后，将地址 ADDR1 对应的数据读出，经过输出延迟 T_{co_36K} 后，在同一周期出现在输出端口 DOA；

第 3 个时钟上升沿之前，ADDRA 达到稳定状态并满足建立时间要求；DRM 正确采样到控制信号、数据及地址信号后，将地址 ADDR2 对应的数据读出，经过输出延迟 T_{co_36K} 后，在同一周期出现在输出端口 DOA；

第 4 个时钟上升沿之前，RSTA 达到稳定状态并满足 $T_{su\_36K\_rst}$ 时间要求，输出端口 DOA 输出预设的复位值 RSTA_VAL；

第 5 个时钟上升沿之前，CEA 在满足保持时间要求后拉低，DRM 进入 disable 状态，输出端口保持不变；

若输出寄存使能 ORCEA 信号有效，DRM 输出数据将会多一个周期的延迟；而复位操作

会在当前周期完成，不会有一个周期的延迟。

3.5 读写操作

根据数据写入时同一端口输出的数据的不同，DRM 的端口写操作支持 Normal Write mode（NW），Transparent Write mode（TW），Read before Write mode（RBW）三种模式，默认的写操作模式为 NW。RAM 的写操作模式由 GTP 中的参数 WRITE_MODE_A/WRITE_MODE_B 决定，当参数 WRITE_MODE_A 的值为"NORMAL_WRITE"时，DRM 的 A 端口的写操作模式被设置为 NW。不同写模式下，读时序是类似的。

DP 模式有两个相对独立的端口，若同时通过两个端口对同一地址进行读写操作会引起冲突。DRM 禁止两端口同时向同一地址写入数据、禁止两端口同时读写同一地址，需要在实际应用中通过用户逻辑加以规避。

3.5.1 Normal Write mode

如图 3-3 Normal Write mode 读写时序图，当用户从 DRM 的一个端口写入数据时，此时该端口的输出数据不更新。图中 Mem 为对应地址存储的旧数据。

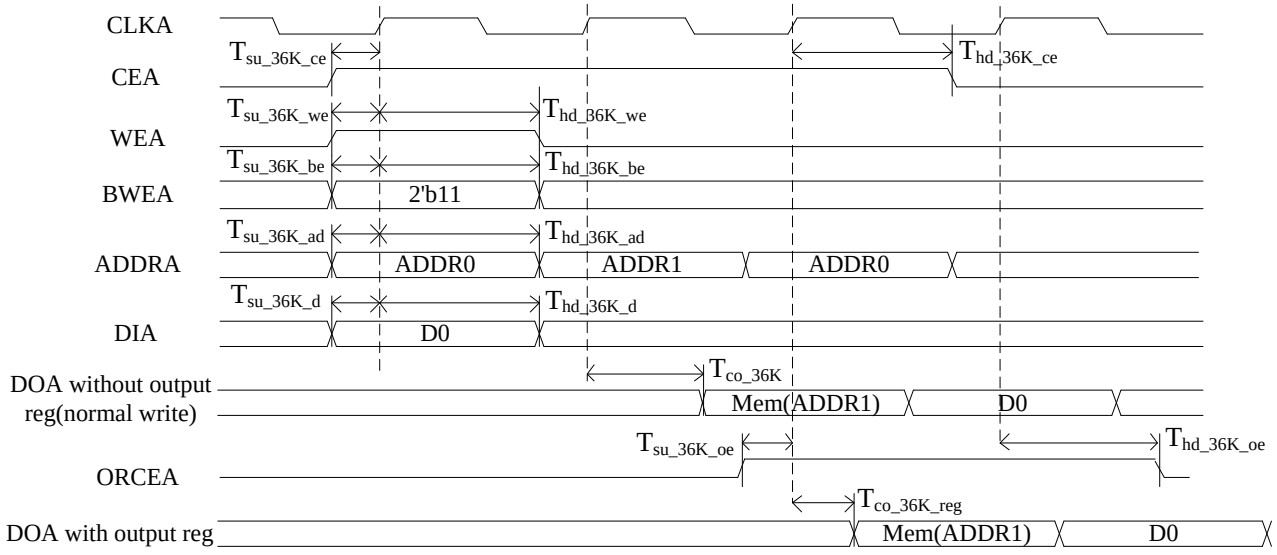

图 3-3 Normal Write mode 读写时序图

3.5.2 Transparent Write mode

如图 3-4 Transparent Write mode 读写时序图，当用户从 DRM 的一个端口写入数据时，写入数据在写入 RAM 的同时直接输出到输出端口，时钟上升沿到数据读出的输出延迟为

T_{co_36K}。图中 Mem 为对应地址存储的旧数据。

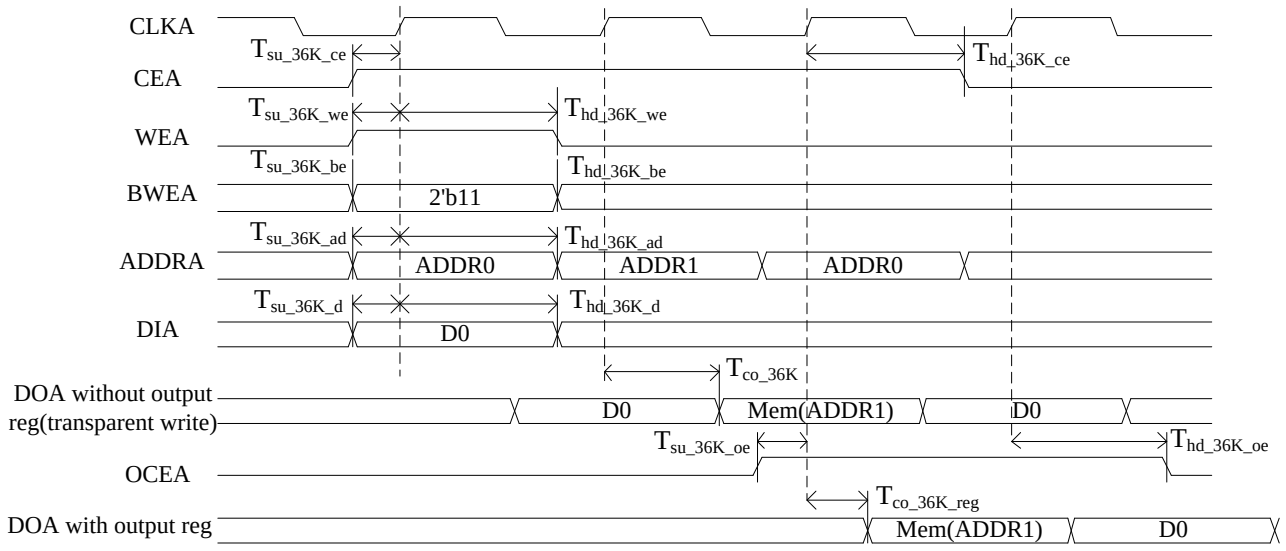

图 3-4 Transparent Write mode 读写时序图

3.5.3 Read before Write mode

如图 3-5 Read before Write mode 读写时序图，当用户从 DRM 的一个端口写入数据时，将首先读出该地址索引的原数据输出到输出端口，时钟上升沿到数据读出的输出延迟为 T_{co_36K}。图中 Mem 为对应地址存储的旧数据。

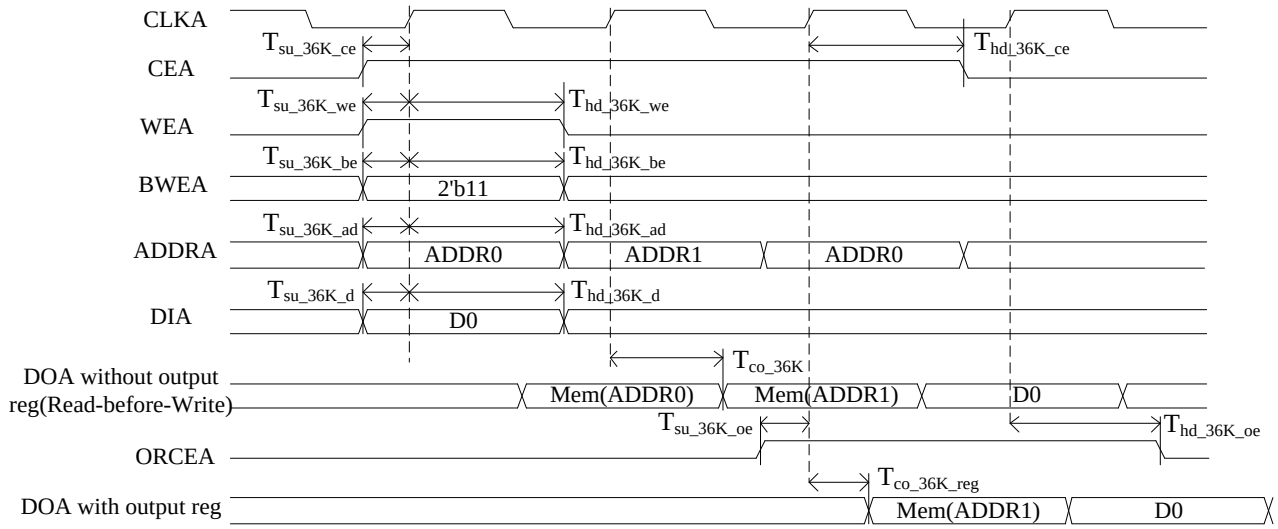

图 3-5 Read before Write mode 读写时序图

3.6 字节使能功能

DRM 支持写操作的 Byte-Write 模式，通过 BWEA/BWEB 信号（高有效）实现对选定数据字节写入，同时屏蔽对同一地址索引的其它字节的写入。该模式主要应用于在限定数据总线宽度时，只对比较窄的数据总线进行操作。比如可以在 18 位数据总线上只对 9 位宽度数据操作。

DP 模式下，当端口位宽为2^N bit，支持 16/32 bit 的写操作的字节使能，此时每个字节包含 8 bit。当端口位宽为9×2^N bit 时，支持 18/36 bit 的写操作的字节使能，此时每个字节包含 9 bit。例如，在 A 端口数据位宽为 32 bit 时，BWEA 包含四个比特，BWEA[3]控制 data[31:24]；BWEA[2]控制 data[23:16]；依次类推。当 BWEA[3:0]=4’b0001 时，写操作有效时仅 data[7:0]被写入当前地址。设计时有以下两点需要考虑的地方：

- A 端口和 B 端口分别有独立的字节使能控制。
  - A 端口：BWEA[7:0]，其中 BWEA[7:4]只在 SDP 模式、SP 模式有效，在 DP 模式无效；
  - B 端口：BWEB[3:0]。
- 字节使能信号在 x18(16) / x36(32)位数据宽度模式下总是有效，当不使用时，需要将该信号连接到高电平。

DP 模式下，字节使能写操作模式只支持 x18(16)/x36(32)写数据端口宽度，写数据端口宽度为 x9(x8)时禁止使用字节使能功能，此时字节使能功能与写使能功能重复，GTP 中字节使能信号无效。

字节使能写操作模式可以与 NW、TW 或 RBW 写操作模式共存。TW 写操作模式下字节使能模式 DRM 的读写时序图如图 3-6 所示，图中 Mem 为对应地址存储的旧数据。

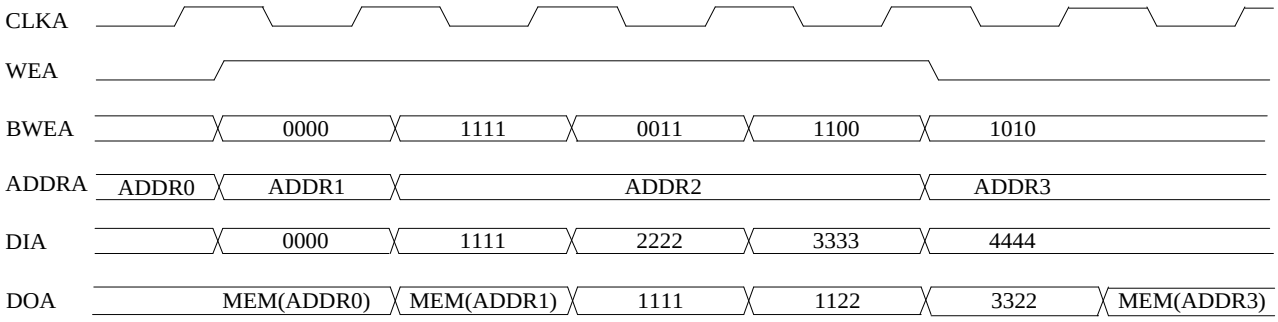

图 3-6 字节使能读写时序图（TW 模式）

3.7 内部寄存器

DRM 内部寄存器包含输入寄存器（IR）、输出寄存器（OR）和内部锁存器（Latch）。

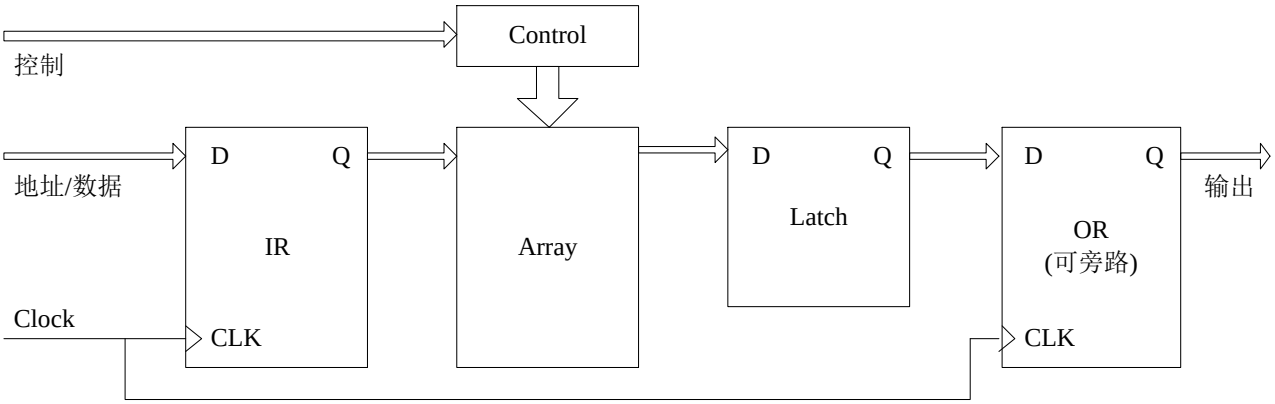

图 3-7 DRM 寄存器逻辑图

端口地址、数据、部分控制信号都有相应的输入寄存器（IR），通过 IR 可以实现同步写操作，IR 不可旁路，不可配置。在地址输入端口内置地址锁存选择通路，其逻辑实现如图 3-8：

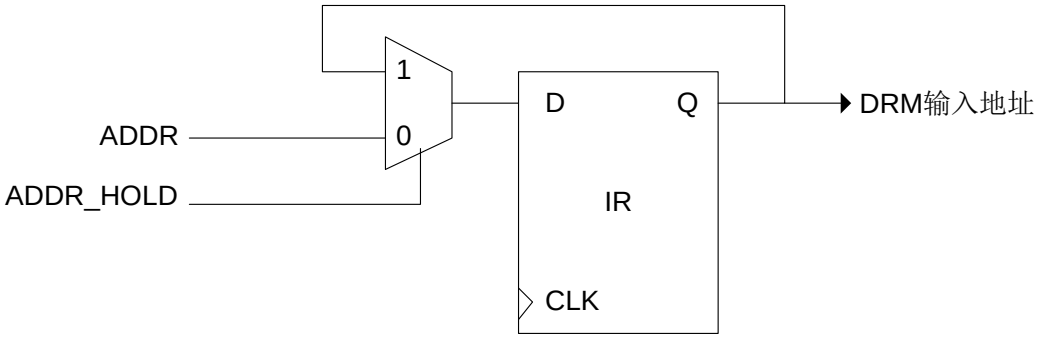

图 3-8 地址 IR 逻辑图

由地址锁存信号 ADDRA_HOLD 控制地址锁存选择通路，ADDRA_HOLD 保持高电平则输入寄存器的输出地址数据保持为当前输出不变，其时序图如图 3-9 所示：

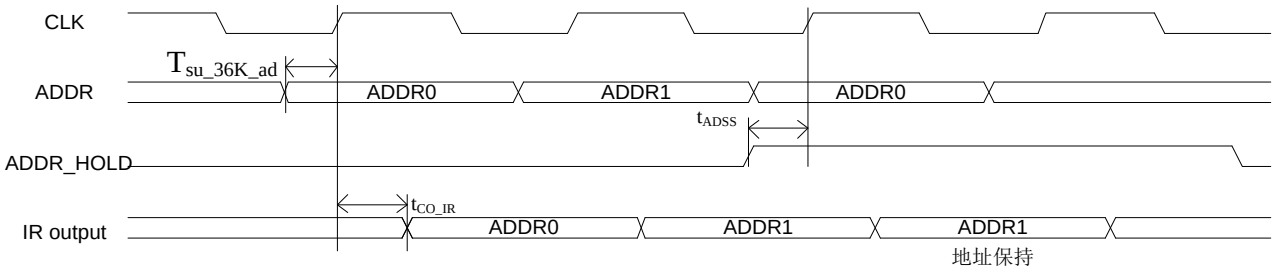

图 3-9 地址 IR 时序图

针对数据输出端口，DRM 特别提供了可选的输出寄存器（OR），以取得更好的时序性能，输出寄存器是否有效由 GTP 的参数 DOA_REG/DOB_REG 决定，例如 DOA_REG 为 1 时，A 端口输出寄存器有效，DOA_REG 为 0 时，A 端口输出寄存器被旁路。在读操作中，输出寄存器旁路时，DOA/DOB 输出为锁存器输出，在同一个读时钟周期时钟上沿输出。B 端口与之类似。

如图 3-10 所示，A 端口输出寄存器有效时可由独立的时钟使能信号 ORCEA 控制，当 ORCEA 为高电平时，输出寄存器的输出数据每个时钟周期上升沿根据输入跳变；当 ORCEA 为低电平时，输出寄存器的输出数据维持之前的数据不变；输出寄存器旁路时 ORCEA 无效。当 ORCEA 为常量 1 时，输出寄存器的使用会把 A 端口读操作的延迟从一个时钟周期增加到两个时钟周期（Case 1）；在具有流控功能的流水线设计应用中，用户也可以用逻辑灵活控制 ORCEA（Case 2）。图中 Mem 为对应地址存储的旧数据。

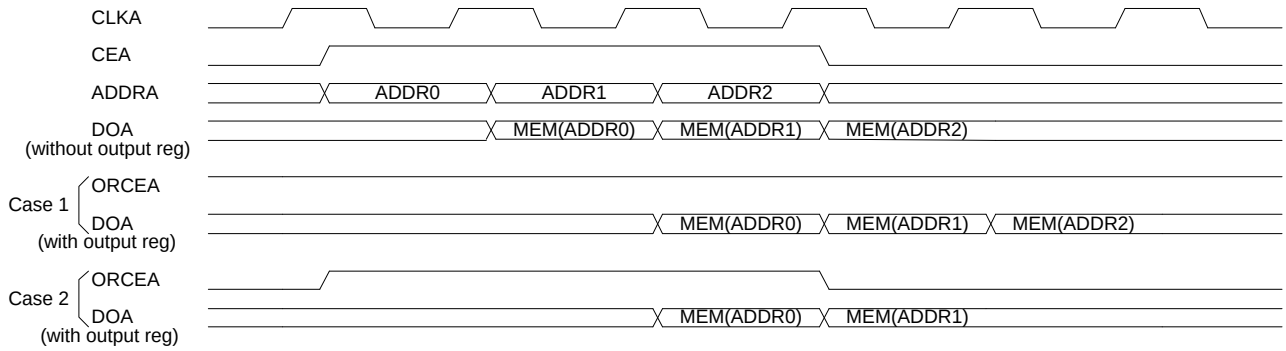

图 3-10 带输出寄存器的读时序

3.8 硬级联功能

DRM 在 36K 模式下支持相邻两个 DRM 不使用外部逻辑资源而进行级联组成 64Kx1 存储器，相邻 DRM 的 A/B 端口需要同时配置为 32Kx1 模式。顶层级联方向为自下向上，如图 3-11 所示：

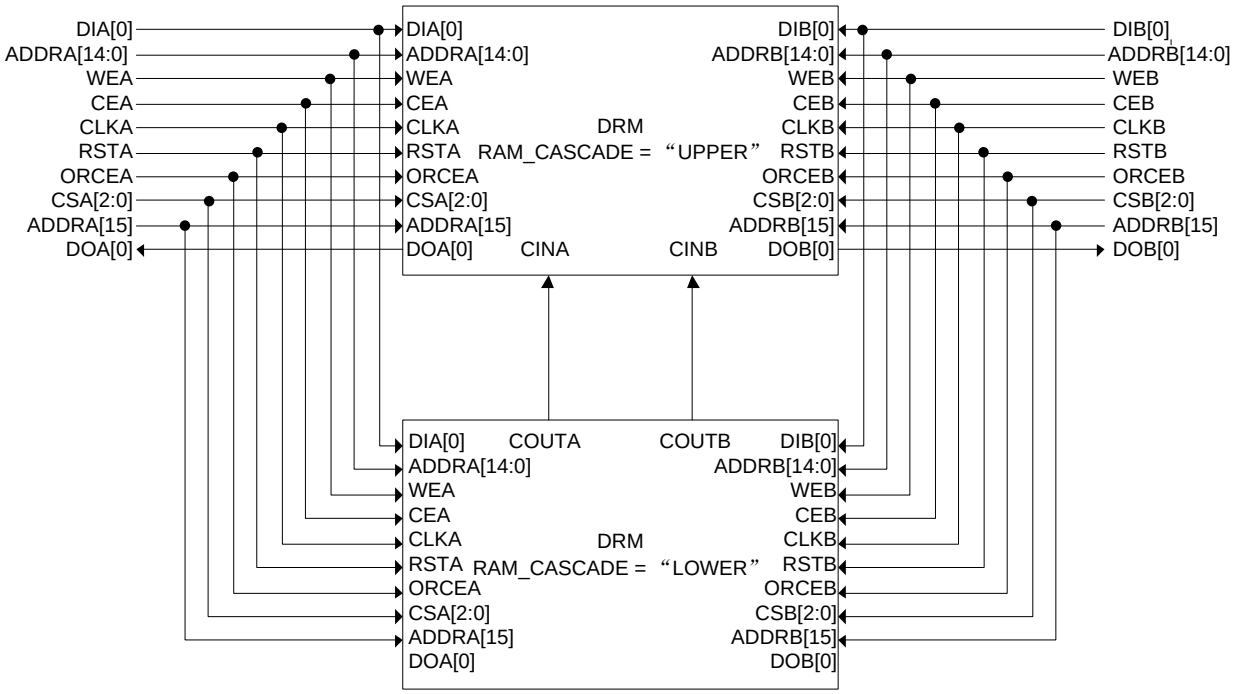

图 3-11 硬级联示意图

在硬级联模式下，两个级联的 DRM 输入控制、数据、地址需要同源连接。同时模式配置（GTP 中除 RAM_CASCADE 以外的参数）需相同，当参数 RAM_CASCADE = "UPPER" 时，与下方相邻 DRM 进行级联，下方 DRM 的配置 RAM_CASCADE = "LOWER"。级联下方 DRM 模块存储对应高 32K 地址，上方 DRM 模块存储对应低 32K 地址。非硬级联模式下，地址硬级联信号即 A/B 端口的地址输入的最高位 ADDRA[15]/ADDRB[15]需连接高电平输入。

硬级联功能将 DRM 的地址深度扩展为 64K，即地址位宽由 15bit 扩展为 16bit，但数据位宽被固定为 1。用户若需使用更大的数据位宽，可通过多个硬级联的基本单元 64Kx1 存储器进行数据级联（即数据端口拼接，其他参数、端口不变）组成 64KxN 存储器；若需使用更大地址位宽，可在硬级联的基础上再进行地址级联，详见 3.9.3 多个 36K DRM 硬级联配置。

相比普通地址级联，硬级联功能不消耗外部逻辑资源，且硬级联通过相邻两个 DRM 的级联数据输入、输出端（CINA/CINB 和 COUTA/COUTB）进行数据进位，拥有更好的性能。

3.9 应用示例

3.9.1 单个 36K DRM 配置

本小节举例说明了单个 36K DRM 的 GTP 配置步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例为混合位宽 DP 模式，A 端口配置为 4Kx8，B 端口配置为 1Kx32 且使用字节使能写模式，两端口不同时钟，使能输出寄存器，读写模式 TW。18K DRM 的配置步骤与之类似。

对单个 36K DRM 按如下步骤进行配置：

a. 按表 3-5 所述对 DRM 的参数进行配置：

表 3-5 单个 36K DRM DP 模式参数配置

参数名称	配置值	说明
DATA_WIDTH_A	8	将 A 端口配置为 4Kx8 模式
DATA_WIDTH_B	32	将 B 端口配置为 1Kx32 模式
WRITE_MODE_A	"TRANSPARENT_WRITE"	将 A 端口读写模式配置为 TW
WRITE_MODE_B		将 B 端口读写模式配置为 TW
DOA_REG	1	使能 A 端口输出寄存器
DOB_REG		使能 B 端口输出寄存器
RAM_MODE	"TRUE_DUAL_PORT"	将 DRM 配置为双口模式

b. 按表 3-6 所示对 DRM 的端口进行连接：

表 3-6 单个 36K DRM DP 模式端口连接

端口名称	连接信号	说明
ADDRA[15:0]	{1'b1,addra[11:0],3'b0}	将输入地址信号 addra[11:0]接到 ADDRA[14:3]，ADDRA[15]接高电平，ADDRA[2:0]接低电平，详细地址连接说明见 9.1 地址和数据端口 Mapping
ADDRB[15:0]	{1'b1,addrb[9:0],5'b0}	将输入地址信号 addrb[9:0]连接到 ADDRB[14:5]，ADDRB[15]接高电平，ADDRB[4:0]接低电平
ADDRA_HOLD	1'b0	不使用 A/B 端口地址锁存功能，接低电平
ADDRB_HOLD		
DIA[35:0]	dia[7:0]	将输入数据信号 dia[7:0]连接到 DIA[7:0]，未使用的 DIA 高位端口悬空，详细数据端口连接说明见 9.1 地址和数据端口 Mapping
DIB[35:0]	{1'b0, dib[31:24], 1'b0, dib[23:16], 1'b0, dib[15:8], 1'b0, dib[7:0]}	将输入数据信号 dib[31:0]连接到 GTP 端口{DIB[34:27], DIB[25:18], DIB[16:9], DIB[7:0]}，DIB[8]、DIB[17]、DIB[26]、DIB[35]为字节附加信息位，详见 9.4 字节附加信息位，数据位宽为 2^N 时悬空或接低电平
CSA[2:0]	3'b0	未使用地址扩展功能，接低电平
CSB[2:0]		
BWEA[7:0]	8'hff	A 端口未使用字节使能功能，将 BWEA 连接到高电平
BWEB[3:0]	bweb	B 端口使用字节使能功能，将字节使能信号 bweb 连接到 BWEB
DOA[35:0]	doa[7:0]	将输出数据信号 doa[7:0]连接到 GTP 端口 DOA[7:0]
DOB[35:0]	{1'bz, dob[31:24], 1'bz, dob[23:16], 1'bz, dob[15:8], 1'bz, dob[7:0]}	将输出数据信号 dob[31:0]连接到 GTP 端口{DOB[34:27], DOB[25:18], DOB[16:9], DOB[7:0]}，DOB[8]、DOB[17]、DOB[26]、DOB[35]则悬空

c. A/B 端口其余信号：将 A/B 端口的时钟、时钟使能、写使能、输出寄存器时钟使能、数据寄存器复位分别连接到相应的 GTP 端口上；

d. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

e. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.DATA_WIDTH_A   (8),
.DATA_WIDTH_B   (32),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.DOA_REG        (1),
.DOB_REG        (1),
.RAM_MODE       ("TRUE_DUAL_PORT")
```

```
) GTP_DRM36K_E1_inst (
.DOA            (doa[7:0]  ), // OUTPUT[35:0]
.DOB            (dob_gtp[35:0]), // OUTPUT[35:0]
.ADDRA          ({1'b1,addra[11:0],3'b0}), // INPUT[15:0]
.ADDRB          ({1'b1,addrb[9:0],5'b0}), // INPUT[15:0]
.BWEA           (8'hff     ), // INPUT[7:0]
.BWEB           (bweb[3:0] ), // INPUT[3:0]
.CSA            (3'b0      ), // INPUT[2:0]
.CSB            (3'b0      ), // INPUT[2:0]
.DIA            (dia[7:0]  ), // INPUT[35:0]
.DIB            ({1'b0,dib[31:24],1'b0,dib[23:16],1'b0,dib[15:8],1'b0,dib[7:0]}), // INPUT[35:0]
.ADDRA_HOLD     (1'b0      ), // INPUT
.ADDRB_HOLD     (1'b0      ), // INPUT
.CEA            (cea       ), // INPUT
.CEB            (ceb       ), // INPUT
.CLKA           (clka      ), // INPUT
.CLKB           (clkb      ), // INPUT
.ORCEA          (orcea     ), // INPUT
.ORCEB          (orceb     ), // INPUT
.RSTA           (rsta      ), // INPUT
.RSTB           (rstb      ), // INPUT
.WEA            (wea       ), // INPUT
.WEB            (web       )  // INPUT
);
assign dob[31:0] = {dob_gtp[34:27],dob_gtp[25:18],dob_gtp[16:9],dob_gtp[7:0]};
```

3.9.2 多个 36K DRM 级联配置

本文所述级联特指地址深度级联，数据级联时单个 DRM 配置相同，只需进行数据端口拼接，本文不再单独进行描述。

通过 36K DRM 进行地址深度级联，不使能硬级联功能，不增加外部逻辑对地址进行处理，DRM 地址最大可级联 3bit（CSA/CSB 位宽为 3），总地址位宽为 18bit；数据级联时最大数据位宽取决于器件 DRM 资源。

本小节举例说明了对多个 36K DRM 进行 GTP 级联配置的步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成级联的 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例级联了 4 个 A/B 端口都配置为 32Kx1 的 DP 36K DRM，总数据位宽 2bit，总地址位宽 16bit。由于 18K DRM 无地址扩展相关信号和参数，因此不推荐使用 18K DRM 进行级联。

级联的 4 个 DRM 分别命名为 DRM_1_1，DRM_1_2，DRM_2_1，DRM_2_2。以 DRM_1_2 为例，第一个数 1 为地址级联坐标，共 2 级地址级联，地址由单个 DRM 的 15bit 扩展为 16bit；第二个数 2 为数据级联坐标，共 2 级数据级联，数据由单个 DRM 的 1bit 扩展为 2bit。对多个 36K DRM 进行级联时，按如下步骤进行配置：

a. 单个 DRM 配置：按照与 3.9.1 单个 36K DRM 配置类似步骤对 4 个 36K DRM 分别配置为 32Kx1 的 DP 36K DRM；

b. 按表 3-7 所述对 DRM 的地址扩展控制参数进行配置：

表 3-7 单个 36K DRM DP 模式级联参数配置

参数名称	DRM 名称	配置值
CSA_MARK	DRM_1_1	3'b000
	DRM_1_2	3'b000
	DRM_2_1	3'b001
	DRM_2_2	3'b001
CSB_MARK	DRM_1_1	3'b000
	DRM_1_2	3'b000
	DRM_2_1	3'b001
	DRM_2_2	3'b001

c. 按表 3-8 所示对 DRM 的级联相关端口进行连接：

表 3-8 单个 36K DRM DP 模式级联端口连接

端口名称	DRM 名称	连接信号
CSA	DRM_1_1	{2'b0,addra[15]}
	DRM_1_2	
	DRM_2_1	
	DRM_2_2	
CSB	DRM_1_1	{2'b0,addrb[15]}
	DRM_1_2	
	DRM_2_1	
	DRM_2_2	
ADDRA[15:0]	DRM_1_1	{1'b1,addra[14:0]}
	DRM_1_2	
	DRM_2_1	
	DRM_2_2	
ADDRB[15:0]	DRM_1_1	{1'b1,addrb[14:0]}
	DRM_1_2	
	DRM_2_1	
	DRM_2_2	
DIA[35:0]	DRM_1_1	dia[0]
	DRM_1_2	dia[1]
	DRM_2_1	dia[0]
	DRM_2_2	dia[1]
DIB[35:0]	DRM_1_1	dib[0]

端口名称	DRM 名称	连接信号
	DRM_1_2	dib[1]
	DRM_2_1	dib[0]
	DRM_2_2	dib[1]
DOA[35:0]	DRM_1_1	doa_0[0]
	DRM_1_2	doa_0[1]
	DRM_2_1	doa_1[0]
	DRM_2_2	doa_1[1]
DOB[35:0]	DRM_1_1	dob_0[0]
	DRM_1_2	dob_0[1]
	DRM_2_1	dob_1[0]
	DRM_2_2	dob_1[1]

d. 输出数据选择：addra[15]通过寄存器延迟一拍后为 0 时 A 端口输出数据为 doa_0[1:0]，为 1 时 A 端口输出数据为 doa_1[1:0]，B 端口也与之类似。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b000),
.CSB_MASK       (3'b000),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT")
) DRM_1_1 (
.DOA            ({doa_0[0]}), // OUTPUT[35:0]
.DOB            ({dob_0[0]}), // OUTPUT[35:0]
.ADDRA          ({1'b1,addra[14:0]}), // INPUT[15:0]
.ADDRB          ({1'b1,addrb[14:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[15]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[15]}), // INPUT[2:0]
.DIA            ({dia[0]}   ), // INPUT[35:0]
.DIB            ({dib[0]}   ), // INPUT[35:0]
.ADDRA_HOLD     (addra_hold), // INPUT
.ADDRB_HOLD     (addrb_hold), // INPUT
.CEA            (cea        ), // INPUT
.CEB            (ceb        ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        )  // INPUT
```

```
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b000),
.CSB_MASK       (3'b000),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT")
) DRM_1_2 (
.DOA            ({doa_0[1]}), // OUTPUT[35:0]
.DOB            ({dob_0[1]}), // OUTPUT[35:0]
.ADDRA          ({1'b1,addra[14:0]}), // INPUT[15:0]
.ADDRB          ({1'b1,addrb[14:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[15]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[15]}), // INPUT[2:0]
.DIA            ({dia[1]}   ), // INPUT[35:0]
.DIB            ({dib[1]}   ), // INPUT[35:0]
.ADDRA_HOLD     (addra_hold), // INPUT
.ADDRB_HOLD     (addrb_hold), // INPUT
.CEA            (cea        ), // INPUT
.CEB            (ceb        ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        )  // INPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT")
) DRM_2_1 (
.DOA            ({doa_1[0]}), // OUTPUT[35:0]
.DOB            ({dob_1[0]}), // OUTPUT[35:0]
.ADDRA          ({1'b1,addra[14:0]}), // INPUT[15:0]
.ADDRB          ({1'b1,addrb[14:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[15]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[15]}), // INPUT[2:0]
.DIA            ({dia[0]}   ), // INPUT[35:0]
.DIB            ({dib[0]}   ), // INPUT[35:0]
.ADDRA_HOLD     (addra_hold), // INPUT
.ADDRB_HOLD     (addrb_hold), // INPUT
```

```
.CEA            (cea        ), // INPUT
.CEB            (ceb        ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        )  // INPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT")
) DRM_2_2 (
.DOA            ({doa_1[1]}), // OUTPUT[35:0]
.DOB            ({dob_1[1]}), // OUTPUT[35:0]
.ADDRA          ({1'b1,addra[14:0]}), // INPUT[15:0]
.ADDRB          ({1'b1,addrb[14:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[15]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[15]}), // INPUT[2:0]
.DIA            ({dia[1]}   ), // INPUT[35:0]
.DIB            ({dib[1]}   ), // INPUT[35:0]
.ADDRA_HOLD     (addra_hold), // INPUT
.ADDRB_HOLD     (addrb_hold), // INPUT
.CEA            (cea        ), // INPUT
.CEB            (ceb        ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        )  // INPUT
);
//输出数据选择
always @(posedge clka or posedge rsta)
  begin
    if (rsta)
      sel_a <= 1'b0;
    else if (~addra_hold & cea)
      sel_a <= addra[15];
    end
always @(posedge clkb or posedge rstb)
  begin
    if (rstb)
```

```
      sel_b <= 1'b0;
    else if (~addrb_hold & ceb)
      sel_b <= addrb[15];
    end
assign doa[1:0] = (sel_a)?doa_1[1:0]:doa_0[1:0];
assign dob[1:0] = (sel_b)?dob_1[1:0]:dob_0[1:0];
```

3.9.3 多个 36K DRM 硬级联配置

通过 36K DRM 进行硬级联，每两个 36K DRM 硬级联后组成 64Kx1 存储器，共 16bit 地址位宽，再进行 3bit（CSA/CSB 位宽为 3）地址深度级联，不增加外部逻辑对地址进行处理，级联后最大地址位宽为 19bit。

本小节举例说明了对多个 36K DRM 进行 GTP 硬级联配置的步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具勾选资源类型为 DRM64K 直接生成硬级联 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

硬级联时 36K DRM 被配置为固定的 32Kx1 模式，两个 36K DRM 硬级联在一起组成硬级联的基本单元 64Kx1 存储器，示例级联了 8 个 A/B 端口都配置为 32Kx1 的 DP 36K DRM，总数据位宽 2bit，总地址位宽 17bit。18K DRM 不支持硬级联。

8 个 DRM 分别命名为 DRM_1_1_upper、DRM_1_1_lower、DRM_1_2_upper、DRM_1_2_lower、DRM_2_1_upper、DRM_2_1_lower、DRM_2_2_upper、DRM_2_2_lower。以 DRM_1_2_upper 为例，第一个数 1 为地址级联坐标，共 2 级地址级联，地址由单个硬级联的基本单元的 16bit 扩展为 17bit；第二个数 2 为数据级联坐标，共 2 级数据级联，数据由单个 DRM 的 1bit 扩展为 2bit；第三个单词 upper 代表硬级联上方的 36K DRM。

对多个 36K DRM 进行硬级联时，按如下步骤进行配置：

a. 单个 DRM 配置：按照与 3.9.1 单个 36K DRM 配置类似步骤对 8 个 36K DRM 分别配置为 32Kx1 的 DP 36K DRM；

b. 按表 3-9 所述对 DRM 的地址扩展控制参数进行配置：

表 3-9 单个 36K DRM DP 模式硬级联参数配置

参数名称	DRM 名称	配置值
CSA_MARK	DRM_1_1_upper	3'b000
	DRM_1_1_lower	3'b000
	DRM_1_2_upper	3'b000
	DRM_1_2_lower	3'b000
	DRM_2_1_upper	3'b001
	DRM_2_1_lower	3'b001
	DRM_2_2_upper	3'b001
	DRM_2_2_lower	3'b001

参数名称	DRM 名称	配置值
CSB_MARK	DRM_1_1_upper	3'b000
	DRM_1_1_lower	3'b000
	DRM_1_2_upper	3'b000
	DRM_1_2_lower	3'b000
	DRM_2_1_upper	3'b001
	DRM_2_1_lower	3'b001
	DRM_2_2_upper	3'b001
	DRM_2_2_lower	3'b001

c. 按表 3-10 所示对 DRM 的级联相关端口进行连接：

表 3-10 单个 36K DRM DP 模式硬级联端口连接

端口名称	DRM 名称	连接信号
CSA	DRM_1_1_upper	{2'b0,addra[16]}
	DRM_1_1_lower	
	DRM_1_2_upper	
	DRM_1_2_lower	
	DRM_2_1_upper	
	DRM_2_1_lower	
	DRM_2_2_upper	
	DRM_2_2_lower	
CSB	DRM_1_1_upper	{2'b0,addrb[16]}
	DRM_1_1_lower	
	DRM_1_2_upper	
	DRM_1_2_lower	
	DRM_2_1_upper	
	DRM_2_1_lower	
	DRM_2_2_upper	
	DRM_2_2_lower	
ADDRA[15:0]	DRM_1_1_upper	{addra[15:0]}
	DRM_1_1_lower	
	DRM_1_2_upper	
	DRM_1_2_lower	
	DRM_2_1_upper	
	DRM_2_1_lower	
	DRM_2_2_upper	
	DRM_2_2_lower	
ADDRB[15:0]	DRM_1_1_upper	{addrb[15:0]}
	DRM_1_1_lower	
	DRM_1_2_upper	

端口名称	DRM 名称	连接信号
	DRM_1_2_lower	
	DRM_2_1_upper	
	DRM_2_1_lower	
	DRM_2_2_upper	
	DRM_2_2_lower	
DIA[35:0]	DRM_1_1_upper	dia[0]
	DRM_1_1_lower	dia[0]
	DRM_1_2_upper	dia[1]
	DRM_1_2_lower	dia[1]
	DRM_2_1_upper	dia[0]
	DRM_2_1_lower	dia[0]
	DRM_2_2_upper	dia[1]
	DRM_2_2_lower	dia[1]
DIB[35:0]	DRM_1_1_upper	dib[0]
	DRM_1_1_lower	dib[0]
	DRM_1_2_upper	dib[1]
	DRM_1_2_lower	dib[1]
	DRM_2_1_upper	dib[0]
	DRM_2_1_lower	dib[0]
	DRM_2_2_upper	dib[1]
	DRM_2_2_lower	dib[1]
DOA[35:0]	DRM_1_1_upper	doa_0[0]
	DRM_1_2_upper	doa_0[1]
	DRM_2_1_upper	doa_1[0]
	DRM_2_2_upper	doa_1[1]
DOB[35:0]	DRM_1_1_upper	dob_0[0]
	DRM_1_2_upper	dob_0[1]
	DRM_2_1_upper	dob_1[0]
	DRM_2_2_upper	dob_1[1]

d. 输出数据选择：addra[16]通过寄存器延迟一拍后为 0 时 A 端口输出数据为 doa_0[1:0]，为 1 时 A 端口输出数据为 doa_0[1:0]，B 端口也与之类似；

e. 按 3.8 硬级联功能所述对每个硬级联的基本单元对应的两个 DRM 分别按 upper 和 lower 进行硬级联参数配置、硬级联端口连接。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b000),
.CSB_MASK       (3'b000),
```

```
.DATA_WIDTH_A  (1),
.DATA_WIDTH_B  (1),
.WRITE_MODE_A  ("TRANSPARENT_WRITE"),
.WRITE_MODE_B  ("TRANSPARENT_WRITE"),
.RAM_MODE      ("TRUE_DUAL_PORT"),
.RAM_CASCADE   ("LOWER")
) DRM_0_0_lower (
.DOA           (          ), // OUTPUT[35:0]
.DOB           (          ), // OUTPUT[35:0]
.ADDRA         ({addra[15:0]}), // INPUT[15:0]
.ADDRB         ({addrb[15:0]}), // INPUT[15:0]
.BWEA          (8'hff     ), // INPUT[7:0]
.BWEB          (4'hf      ), // INPUT[3:0]
.CSA           ({2'b0,addra[16]}), // INPUT[2:0]
.CSB           ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA           ({dia[0]}  ), // INPUT[35:0]
.DIB           ({dib[0]}  ), // INPUT[35:0]
.ADDRA_HOLD    (1'b0      ), // INPUT
.ADDRB_HOLD    (1'b0      ), // INPUT
.CEA           (1'b1      ), // INPUT
.CEB           (1'b1      ), // INPUT
.CLKA          (clka      ), // INPUT
.CLKB          (clkb      ), // INPUT
.ORCEA         (1'b0      ), // INPUT
.ORCEB         (1'b0      ), // INPUT
.RSTA          (rsta      ), // INPUT
.RSTB          (rstb      ), // INPUT
.WEA           (wea       ), // INPUT
.WEB           (web       ), // INPUT
.CINA          (          ), // INPUT
.CINB          (          ), // INPUT
.COUTA         (couta[0]  ), // OUTPUT
.COUTB         (coutb[0]  )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK      (3'b000),
.CSB_MASK      (3'b000),
.DATA_WIDTH_A  (1),
.DATA_WIDTH_B  (1),
.WRITE_MODE_A  ("TRANSPARENT_WRITE"),
.WRITE_MODE_B  ("TRANSPARENT_WRITE"),
.RAM_MODE      ("TRUE_DUAL_PORT"),
.RAM_CASCADE   ("UPPER")
) DRM_0_0_upper (
.DOA           ({doa_0[0]}), // OUTPUT[35:0]
.DOB           ({dob_0[0]}), // OUTPUT[35:0]
.ADDRA         ({addra[15:0]}), // INPUT[15:0]
.ADDRB         ({addrb[15:0]}), // INPUT[15:0]
.BWEA          (8'hff     ), // INPUT[7:0]
.BWEB          (4'hf      ), // INPUT[3:0]
.CSA           ({2'b0,addra[16]}), // INPUT[2:0]
.CSB           ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA           ({dia[0]}  ), // INPUT[35:0]
.DIB           ({dib[0]}  ), // INPUT[35:0]
```

```
.ADDRA_HOLD     (1'b0       ), // INPUT
.ADDRB_HOLD     (1'b0       ), // INPUT
.CEA            (1'b1       ), // INPUT
.CEB            (1'b1       ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        ), // INPUT
.CINA           (couta[0]   ), // INPUT
.CINB           (coutb[0]   ), // INPUT
.COUTA          (           ), // OUTPUT
.COUTB          (           )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b000),
.CSB_MASK       (3'b000),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("LOWER")
) DRM_0_1_lower (
.DOA            (           ), // OUTPUT[35:0]
.DOB            (           ), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[1]}   ), // INPUT[35:0]
.DIB            ({dib[1]}   ), // INPUT[35:0]
.ADDRA_HOLD     (1'b0       ), // INPUT
.ADDRB_HOLD     (1'b0       ), // INPUT
.CEA            (1'b1       ), // INPUT
.CEB            (1'b1       ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        ), // INPUT
.CINA           (           ), // INPUT
.CINB           (           ), // INPUT
.COUTA          (couta[1]   ), // OUTPUT
.COUTB          (coutb[1]   )  // OUTPUT
);
```

```
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b000),
.CSB_MASK       (3'b000),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("UPPER")
) DRM_0_1_upper (
.DOA            ({doa_0[1]}), // OUTPUT[35:0]
.DOB            ({dob_0[1]}), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[1]}   ), // INPUT[35:0]
.DIB            ({dib[1]}   ), // INPUT[35:0]
.ADDRA_HOLD     (1'b0       ), // INPUT
.ADDRB_HOLD     (1'b0       ), // INPUT
.CEA            (1'b1       ), // INPUT
.CEB            (1'b1       ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        ), // INPUT
.CINA           (couta[1]   ), // INPUT
.CINB           (coutb[1]   ), // INPUT
.COUTA          (           ), // OUTPUT
.COUTB          (           )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("LOWER")
) DRM_1_0_lower (
.DOA            (           ), // OUTPUT[35:0]
.DOB            (           ), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
```

```
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[0]}  ), // INPUT[35:0]
.DIB            ({dib[0]}  ), // INPUT[35:0]
.ADDRA_HOLD     (1'b0      ), // INPUT
.ADDRB_HOLD     (1'b0      ), // INPUT
.CEA            (1'b1      ), // INPUT
.CEB            (1'b1      ), // INPUT
.CLKA           (clka      ), // INPUT
.CLKB           (clkb      ), // INPUT
.ORCEA          (1'b0      ), // INPUT
.ORCEB          (1'b0      ), // INPUT
.RSTA           (rsta      ), // INPUT
.RSTB           (rstb      ), // INPUT
.WEA            (wea       ), // INPUT
.WEB            (web       ), // INPUT
.CINA           (          ), // INPUT
.CINB           (          ), // INPUT
.COUTA          (couta[2]  ), // OUTPUT
.COUTB          (coutb[2]  )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("UPPER")
) DRM_1_0_upper (
.DOA            ({doa_1[0]}), // OUTPUT[35:0]
.DOB            ({dob_1[0]}), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
.BWEA           (8'hff     ), // INPUT[7:0]
.BWEB           (4'hf      ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[0]}  ), // INPUT[35:0]
.DIB            ({dib[0]}  ), // INPUT[35:0]
.ADDRA_HOLD     (1'b0      ), // INPUT
.ADDRB_HOLD     (1'b0      ), // INPUT
.CEA            (1'b1      ), // INPUT
.CEB            (1'b1      ), // INPUT
.CLKA           (clka      ), // INPUT
.CLKB           (clkb      ), // INPUT
.ORCEA          (1'b0      ), // INPUT
.ORCEB          (1'b0      ), // INPUT
.RSTA           (rsta      ), // INPUT
.RSTB           (rstb      ), // INPUT
.WEA            (wea       ), // INPUT
.WEB            (web       ), // INPUT
.CINA           (couta[2]  ), // INPUT
.CINB           (coutb[2]  ), // INPUT
```

```
.COUTA          (           ), // OUTPUT
.COUTB          (           )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("LOWER")
) DRM_1_1_lower (
.DOA            (           ), // OUTPUT[35:0]
.DOB            (           ), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[1]}), // INPUT[35:0]
.DIB            ({dib[1]}), // INPUT[35:0]
.ADDRA_HOLD     (1'b0       ), // INPUT
.ADDRB_HOLD     (1'b0       ), // INPUT
.CEA            (1'b1       ), // INPUT
.CEB            (1'b1       ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        ), // INPUT
.CINA           (           ), // INPUT
.CINB           (           ), // INPUT
.COUTA          (couta[3]   ), // OUTPUT
.COUTB          (coutb[3]   )  // OUTPUT
);
GTP_DRM36K_E1 #(
.CSA_MASK       (3'b001),
.CSB_MASK       (3'b001),
.DATA_WIDTH_A   (1),
.DATA_WIDTH_B   (1),
.WRITE_MODE_A   ("TRANSPARENT_WRITE"),
.WRITE_MODE_B   ("TRANSPARENT_WRITE"),
.RAM_MODE       ("TRUE_DUAL_PORT"),
.RAM_CASCADE    ("UPPER")
) DRM_1_1_upper (
.DOA            ({doa_1[1]}), // OUTPUT[35:0]
.DOB            ({dob_1[1]}), // OUTPUT[35:0]
.ADDRA          ({addra[15:0]}), // INPUT[15:0]
.ADDRB          ({addrb[15:0]}), // INPUT[15:0]
```

```
.BWEA           (8'hff      ), // INPUT[7:0]
.BWEB           (4'hf       ), // INPUT[3:0]
.CSA            ({2'b0,addra[16]}), // INPUT[2:0]
.CSB            ({2'b0,addrb[16]}), // INPUT[2:0]
.DIA            ({dia[1]}), // INPUT[35:0]
.DIB            ({dib[1]}), // INPUT[35:0]
.ADDRA_HOLD     (1'b0       ), // INPUT
.ADDRB_HOLD     (1'b0       ), // INPUT
.CEA            (1'b1       ), // INPUT
.CEB            (1'b1       ), // INPUT
.CLKA           (clka       ), // INPUT
.CLKB           (clkb       ), // INPUT
.ORCEA          (1'b0       ), // INPUT
.ORCEB          (1'b0       ), // INPUT
.RSTA           (rsta       ), // INPUT
.RSTB           (rstb       ), // INPUT
.WEA            (wea        ), // INPUT
.WEB            (web        ), // INPUT
.CINA           (couta[3]   ), // INPUT
.CINB           (coutb[3]   ), // INPUT
.COUTA          (           ), // OUTPUT
.COUTB          (           )  // OUTPUT
);
//输出数据选择
always @(posedge clka or posedge rsta)
  begin
    if (rsta)
      sel_a <= 1'b0;
    else
      sel_a <= addra[16];
    end
always @(posedge clkb or posedge rstb)
  begin
    if (rstb)
      sel_b <= 1'b0;
    else
      sel_b <= addrb[16];
    end
assign doa[1:0] = (sel_a)?doa_1[1:0]:doa_0[1:0];
assign dob[1:0] = (sel_b)?dob_1[1:0]:dob_0[1:0];
```

4 SDP 模式

4.1 模式介绍

RAM 的端口模式由参数 RAM_MODE 决定，当参数 RAM_MODE 的值为"SIMPLE_DUAL_PORT"时，RAM 进入简单双口模式。本文中将详细介绍 36K 的 SDP RAM，18K 与 36K 结构与功能基本一致，仅位宽不同。

SDP RAM 模式支持：

- 双端口读和写，其中一个端口作为读端口，另一个端口作为写端口。
- 任意一个端口不能同时做读或者写操作。
- 两端口有独立的位宽设置
- 18K 模式下 x32/x36 位宽以及 36K 模式下 x64/x72 位宽 A 端口固定为写入控制端口（WEA 为高时有效），B 端口固定为读出控制端口（WEB 为低时有效）。

4.2 数据端口

图 4-1 展示了 36K SDP RAM 的结构图。

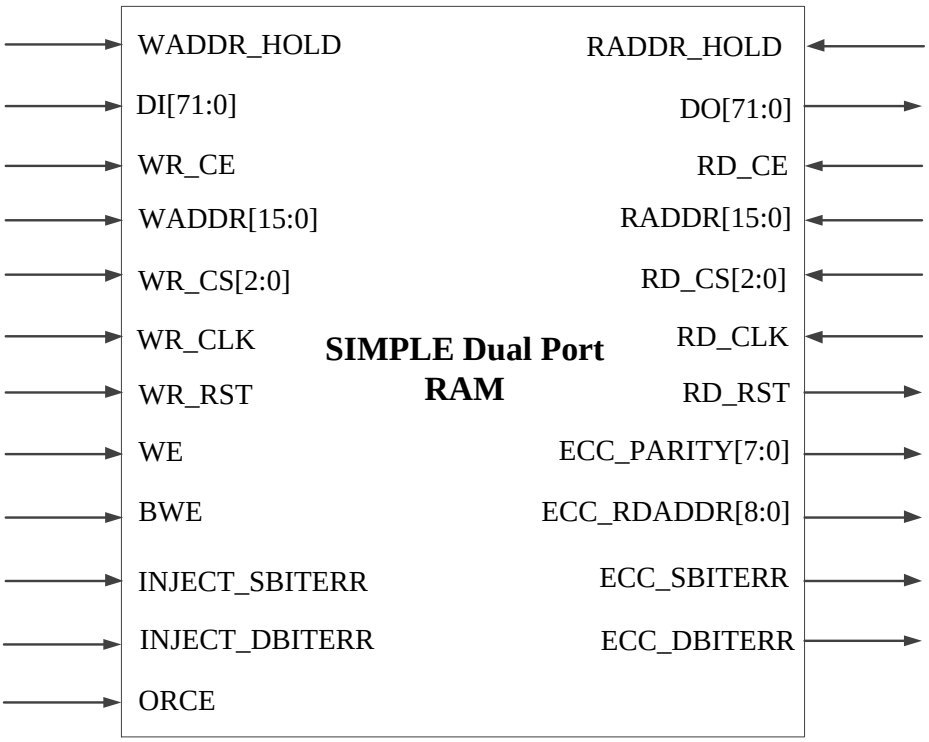

图 4-1 SDP RAM 数据端口

每个 18K、36K 的 DRM 也能被配置为 SDP（简单双口）RAM。在这个模式下，DRM

端口数据位宽增大至 72 bits，SDP RAM 模式包含两对端口（每对有 A，B 两个不同的端口），A、B 两个端口中一个端口专用于数据写入，另一个端口专用于数据读取，读写端口同样均支持不同的时钟。与 DP RAM 相同的是，若同时通过两个端口对同一地址进行读写操作会引起冲突。

表 4-1 罗列了 SDP RAM 模式下的端口名称和端口的描述。

表 4-1 SDP RAM 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
WADDR_HOLD	输入	写端口地址锁存信号	RADDR_HOLD	输入	读端口地址锁存信号
DI	输入	写端口数据输入	DO	输出	读端口数据输出
WR_CE	输入	输入寄存器时钟使能	RD_CE	输入	输入寄存器时钟使能
WADDR	输入	写端口地址输入	RADRD	输入	读端口地址输入
WR_CS	输入	写端口地址扩展	RD_CS	输入	读端口地址扩展
WR_CLK	输入	写端口时钟	RD_CLK	输入	读端口时钟
WR_RST	输入	写端口输出寄存器复位	RD_RSTB	输入	读端口输出寄存器复位
WE	输入	写使能	ECC_PARITY[7:0]	输出	ECC 模式校验位输出
BWE	输入	字节使能信号	ECC_RDADDR[8:0]	输出	ECC 模式读地址输出
INJECT_SBITERR	输入	ECC 模式单比特错误注入	ECC_SBITERR	输出	ECC 模式单比特错误标志
INJECT_DBITERR	输入	ECC 模式双比特错误注入	ECC_DBITERR	输出	ECC 模式双比特错误标志
ORCE	输入	输出寄存器时钟使能			

注：1.SDP 模式的 x64/x72 位宽下，DI 由 A/B 端口的 DIA/DIB 拼接而成，DO 同理，详见 9.1 地址和数据端口 Mapping；
2.18K DRM 无 CSA/CSB 端口，不支持地址扩展，如需 DRM 级联进行地址深度扩展，建议使用 36K DRM。

4.3 位宽组合

RAM 的端口位宽由 GTP 中的参数 DATA_WIDTH_A/DATA_WIDTH_B 决定，例如，当参数 DATA_WIDTH_A 的值为 4 时，A 端口数据位宽被设置为 4 bit。SDP 模式支持 A、B 端口设置不同数据位宽。

如表 4-2 所示为 36K DRM 模式下 Simple Dual Port RAM 模式允许的位宽组合。

表 4-2 36K DRM 模式下 Simple Dual Port RAM 模式允许的位宽组合

		写端口										
		32Kx1	**16Kx2**	**8Kx4**	**4Kx8**	**2Kx16**	**1Kx32**	**512x64**	**4Kx9**	**2Kx18**	**1Kx36**	**512x72**
读端口	**32Kx1**	√	√	√	√	√	√	√				
	16Kx2	√	√	√	√	√	√	√				
	8Kx4	√	√	√	√	√	√	√				
	4Kx8	√	√	√	√	√	√	√				
	2Kx16	√	√	√	√	√	√	√				

		写端口										
		32Kx1	16Kx2	8Kx4	4Kx8	2Kx16	1Kx32	512x64	4Kx9	2Kx18	1Kx36	512x72
读端口	1Kx32	√	√	√	√	√	√	√				
	512x64	√	√	√	√	√	√	√				
	4Kx9								√	√	√	√
	2Kx18								√	√	√	√
	1Kx36								√	√	√	√
	512x72								√	√	√	√

注：√表示支持的位宽组合。

如表 4-3 所示为 18K DRM 模式下 Simple Dual Port RAM 模式允许的位宽组合。

表 4-3 18K DRM 模式下 Simple Dual Port RAM 模式允许的位宽组合

		写端口								
		16Kx1	8Kx2	4Kx4	2Kx8	1Kx16	512x32	2Kx9	1Kx18	512x36
读端口 0/1	16Kx1	√	√	√	√	√	√			
	8Kx2	√	√	√	√	√	√			
	4Kx4	√	√	√	√	√	√			
	2Kx8	√	√	√	√	√	√			
	1Kx16	√	√	√	√	√	√			
	512x32	√	√	√	√	√	√			
	2Kx9							√	√	√
	1Kx18							√	√	√
	512x36							√	√	√

注：√表示支持的位宽组合。

4.4 时序参数

SDP 模式下，DRM 的时序参数与 DP 模式相同，详见 3.4 时序参数。

4.5 读写操作

SDP 模式不支持读写模式设置，在使用时禁止设置成 TW 或 RBW 模式，只能设置为默认的 NW 模式，即 GTP 中参数 WRITE_MODE_A/WRITE_MODE_B 的值必须为默认的 "NORMAL_WRITE"，否则 DRM 将进入非正常的写操作状态。

SDP 模式有两个相对独立的端口，若同时通过两个端口对同一地址进行读写操作会引起冲突。DRM 禁止两端口同时向同一地址写入数据、禁止两端口同时读写同一地址，需要在实际应用中通过用户逻辑加以规避。

如图 4-2 SDP 模式读写时序图，用户从 DRM 的 A 端口写入数据，从 DRM 的 B 端口的读出数据。图中 Mem 为对应地址存储的旧数据。

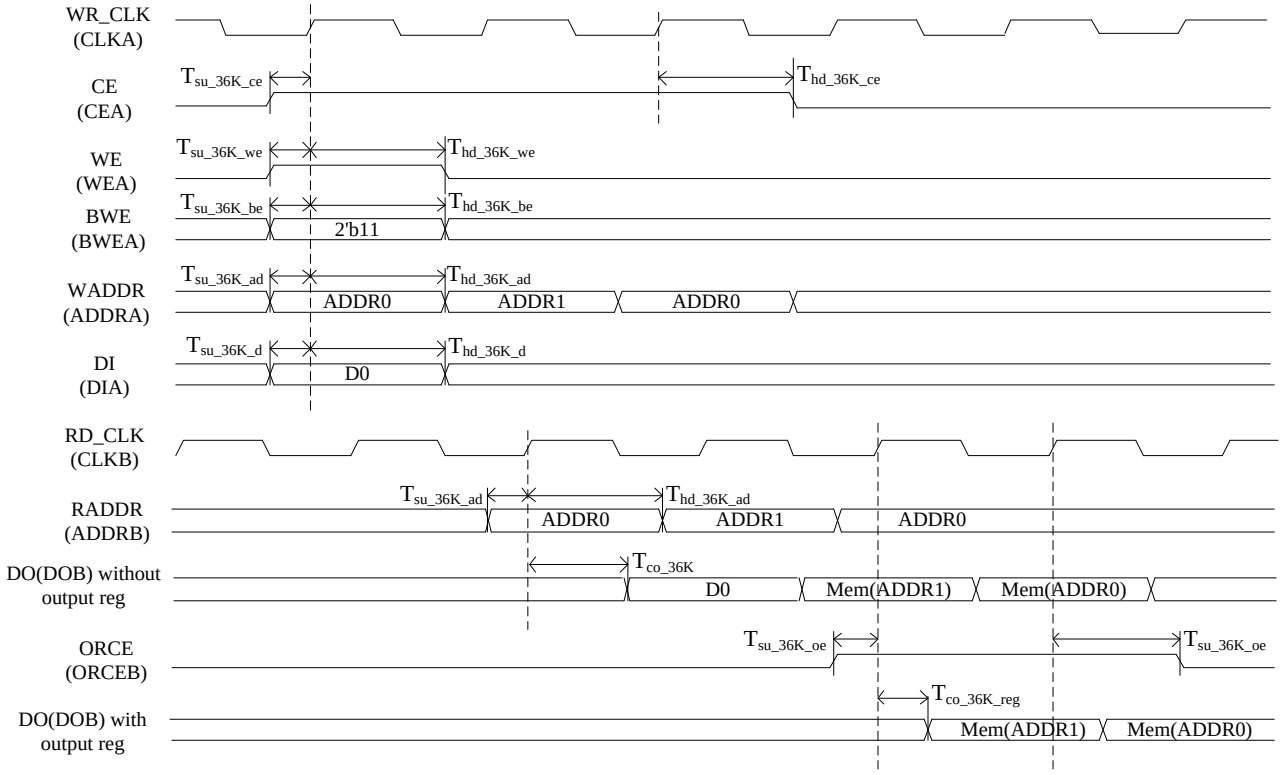

图 4-2 SDP 模式读写时序图

4.6 字节使能功能

SDP 模式的字节使能功能与 DP 模式类似，详见 3.6 字节使能功能。SDP 模式的字节使能与 DP 模式主要差异为以下两点：

- SDP 模式一个端口固定为写端口，一个端口固定为读端口，读端口字节使能信号需接低电平；
- SDP 模式字节使能信号还支持 x72(64)位数据宽度模式，此时，A 端口固定为写端口，字节使能信号 WEA[7:4]有效；其余位宽下，A 端口为写端口时 WEA[7:4]接高电平，A 端口为读端口时 WEA[7:4]接低电平。

SDP 模式下字节使能模式 DRM 的读写时序图如图 4-3 所示，图中 Mem 为对应地址存储的旧数据。

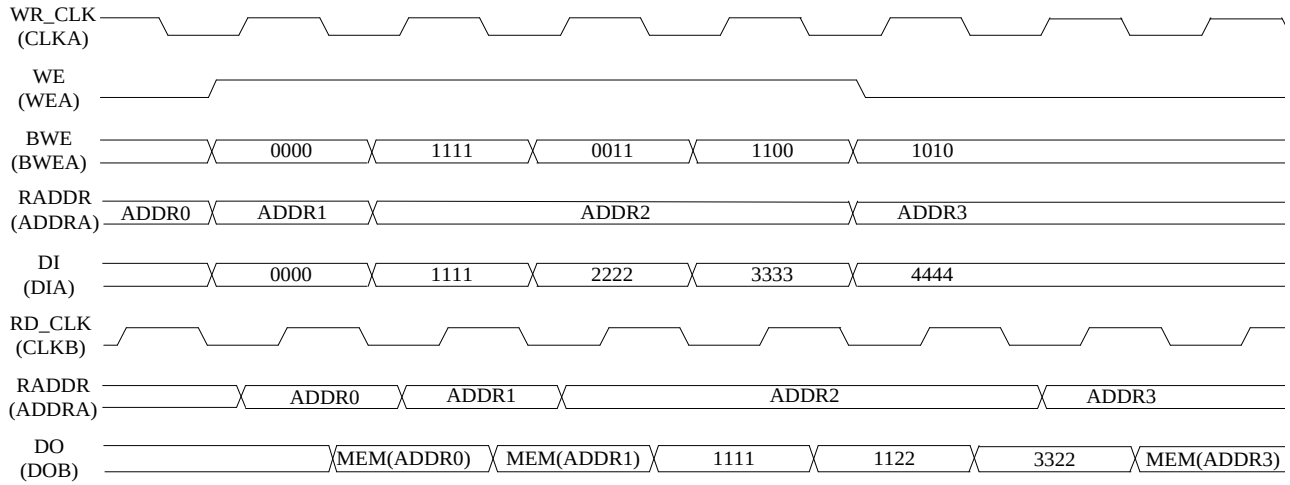

图 4-3 字节使能读写时序图（SDP 模式）

4.7 内部寄存器

SDP 模式的内部寄存器和 DP 模式相同，详见 3.7 内部寄存器。

4.8 硬级联功能

SDP 模式的硬级联功能和 DP 模式相同，详见 3.8 硬级联功能。

4.9 应用示例

4.9.1 单个 36K DRM 配置

本小节举例说明了单个 36K DRM 的 GTP 配置步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例为混合位宽 SDP 模式，读端口 A 配置为 4Kx9，写端口 B 端口配置为 512x72（由 A/B 端口 DOA/DOB 拼接实现），两端口不同时钟，使能输出寄存器。18K DRM 的配置步骤与之类似。

对单个 36K DRM 按如下步骤进行配置：

a. 按表 4-4 所述对 DRM 的参数进行配置：

表 4-4 单个 36K DRM SDP 模式参数配置

参数名称	配置值	说明
DATA_WIDTH_A	9	将 A 端口配置为 4Kx9 模式
DATA_WIDTH_B	72	将 B 端口配置为 512x72 模式
WRITE_MODE_A	"NORMAL_WRITE"	将 A 端口读写模式配置为 NW

参数名称	配置值	说明
WRITE_MODE_B		将 B 端口读写模式配置为 NW
DOA_REG	1	使能 A 端口输出寄存器
DOB_REG		使能 B 端口输出寄存器
RAM_MODE	"SIMPLE_DUAL_PORT"	将 DRM 配置为简单双口模式

b. 按表 4-5 所示对 DRM 的端口进行连接：

表 4-5 单个 36K DRM SDP 模式端口连接

端口名称	连接信号	说明
ADDRA[15:0]	{1'b1,waddr[11:0],3'b0}	A 端口做写端口,将写地址信号 waddr[11:0]连接到 ADDRA[14:3]，ADDRA[15]接高电平，ADDRA[2:0]接低电平，详细地址连接说明见 9.1 地址和数据端口 Mapping
ADDRB[15:0]	{1'b1,raddr[8:0],6'b0}	B 端口做读端口，将读地址信号 raddr[8:0]连接到 ADDRB[14:6]，ADDRB[15]接高电平，ADDRB[5:0]接低电平
ADDRA_HOLD	1'b0	不使用 A/B 端口地址锁存功能，接低电平
ADDRB_HOLD		
DIA[35:0]	di[8:0]	将写数据信号 di[8:0]连接到 DIA[8:0]，未使用的 DIA 高位端口悬空，详细数据端口连接说明见 9.1 地址和数据端口 Mapping
DIB[35:0]	悬空或接低电平	B 端口为读端口且未进行写入数据拼接，GTP 端口 DIB 悬空或接低电平
CSA[2:0]	3'b0	未使用地址扩展功能，接低电平
CSB[2:0]		
BWEA[7:0]	8'hff	A 端口为写端口，未使用字节使能功能，将 BWEA 连接到高电平
BWEB[3:0]	4'b0	B 端口为读端口，禁止写操作，将 BWEB 连接到低电平
DOA[35:0]	do[35:0]	A/B 端口拼接实现 72bit 数据位宽，DOA 输出低 36bit 读数据
DOB[35:0]	do[71:36]	A/B 端口拼接实现 72bit 数据位宽，DOB 输出高 36bit 读数据
WEA	we	将写使能信号 we 连接到写端口 A 的 WEA 上
WEB	1'b0	读端口 B 的写使能 WEB 接低电平

c. 读写端口其余信号：将读写端口的时钟、时钟使能、输出寄存器时钟使能、数据寄存器复位分别连接到相应的 GTP 相应 A/B 端口上；

d. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

e. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.DATA_WIDTH_A  (9),
.DATA_WIDTH_B  (72),
.WRITE_MODE_A  ("NORMAL_WRITE"),
.WRITE_MODE_B  ("NORMAL_WRITE"),
.DOA_REG       (1),
.DOB_REG       (1),
.RAM_MODE      ("SIMPLE_DUAL_PORT")
) GTP_DRM36K_E1_inst (
.DOA           (do[35:0]  ), // OUTPUT[35:0]
.DOB           (do[71:36] ), // OUTPUT[35:0]
.ADDRA         ({1'b1,waddr[11:0],3'b0}), // INPUT[15:0]
.ADDRB         ({1'b1,raddr[8:0],6'b0}), // INPUT[15:0]
.BWEA          (8'hff     ), // INPUT[7:0]
.BWEB          (4'b0      ), // INPUT[3:0]
.CSA           (3'b0      ), // INPUT[2:0]
.CSB           (3'b0      ), // INPUT[2:0]
.DIA           (di[8:0]   ), // INPUT[35:0]
.DIB           (          ), // INPUT[35:0]
.ADDRA_HOLD    (1'b0      ), // INPUT
.ADDRB_HOLD    (1'b0      ), // INPUT
.CEA           (wr_ce     ), // INPUT
.CEB           (rd_ce     ), // INPUT
.CLKA          (wr_clk    ), // INPUT
.CLKB          (rd_clk    ), // INPUT
.ORCEA         (orce      ), // INPUT
.ORCEB         (orce      ), // INPUT
.RSTA          (wr_rst    ), // INPUT
.RSTB          (rd_rst    ), // INPUT
.WEA           (we        ), // INPUT
.WEB           (1'b0      )  // INPUT
);
```

4.9.2 多个 36K DRM 级联配置

SDP 模式多个 36K DRM 级联与 DP 模式相同，详见 3.9.2 多个 36K DRM 级联配置。

4.9.3 多个 36K DRM 硬级联配置

SDP 模式多个 36K DRM 硬级联与 DP 模式相同，详见 3.9.3 多个 36K DRM 硬级联配置。

5 SP 模式

5.1 模式介绍

RAM 的端口模式由参数 RAM_MODE 决定，当参数 RAM_MODE 的值为"SINGLE_PORT"时，RAM 进入单口模式。本文中将详细介绍 36K 的 SP RAM，18K 与 36K 结构与功能基本一致，仅位宽不同。

SP RAM模式支持：

- 单端口读操作
- 单端口写操作
- 18K 模式下 x32/x36 位宽以及 36K 模式下 x64/x72 位宽 A 端口固定为写入控制端口（WEA 为高时有效），B 端口固定为读出控制端口（WEB 为低时有效）。

5.2 数据端口

每个 18K、36K 的 DRM 也能被配置为 SP（单口）RAM。在这个模式下，DRM 端口数据位宽增大至 72 bits，SP RAM 模式在两个端口共享情况下，DRM 仅包含一个端口，SP RAM 模式可以对这个端口进行读写操作。图 5-1 展示了 36K SP RAM 的结构图。

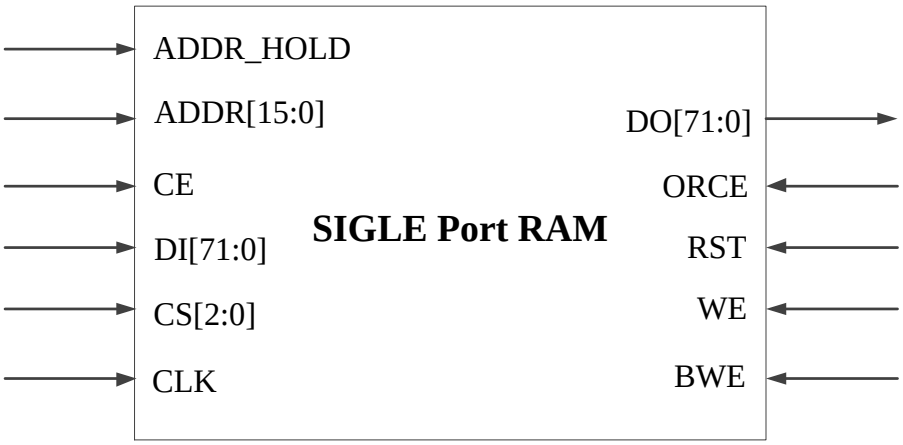

图 5-1 SP RAM 数据端口

表 5-1 罗列了 SP RAM 模式下的端口名称和端口的描述。

表 5-1 SP RAM 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
ADDR_HOLD	输入	地址锁存信号	DO	输出	读数据输出
ADDR	输入	地址输入	ORCE	输入	输出寄存器时钟使能

端口名称	方向	描述	端口名称	方向	描述
CE	输入	输入寄存器时钟使能	RST	输入	输出寄存器复位
DI	输入	写数据输入	WE	输入	写使能
CS	输入	地址扩展	BWE	输入	字节使能信号
CLK	输入	端口时钟			

注：1.SP 模式的 x64/x72 位宽下，DI 由 A/B 端口的 DIA/DIB 拼接而成，DO 同理，详见 9.1 地址和数据端口 Mapping；
2.18K DRM 无 CSA/CSB 端口，不支持地址扩展，如需 DRM 级联进行地址深度扩展，建议使用 36K DRM。

5.3 位宽组合

RAM 的端口位宽由 GTP 中的参数 DATA_WIDTH_A/DATA_WIDTH_B 决定，例如，当参数 DATA_WIDTH_A 的值为 4 时，A 端口数据位宽被设置为 4 bit。SP 模式的 A、B 端口需设置为相同数据位宽。

如表 5-2 所示为 36K DRM 模式下 Single Port RAM 模式允许的位宽。

表 5-2 36K DRM 模式 Single Port RAM 模式允许的位宽

模式	32Kx1	16Kx2	8Kx4	4Kx8	2Kx16	1Kx32	512x64	4Kx9	2Kx18	1Kx36	512x72
SP RAM	√	√	√	√	√	√	√	√	√	√	√

注：1.√表示支持的位宽组合；
2.36K DRM 模式下 Single Port RAM 模式，1Kx32、1Kx36 模式禁止直接设置 TW、RBW 读写模式，需要额外配置；512x64、512x72 模式禁止设置 TW、RBW 读写模式，详见 5.5 读写操作。

如表 5-3 所示为 18K DRM 模式下 Simple Port RAM 模式允许的位宽。

表 5-3 18K DRM 模式 Single Port RAM 模式允许的位宽

模式	16Kx1	8Kx2	4Kx4	2Kx8	1Kx16	512x32	2Kx9	1Kx18	512x36
SP RAM	√	√	√	√	√	√	√	√	√

注：1.√表示支持的位宽组合。
2.18K DRM 模式下 Single Port RAM 模式，512x32、512x36 模式禁止直接设置 TW、RBW 读写模式，需要额外配置，详见 5.5 读写操作。

5.4 时序参数

SP 模式下，DRM 的时序参数与 DP 模式相同，详见 3.4 时序参数。

5.5 读写操作

SP 模式的各读写模式的读写操作与 DP 模式的单端口类似，详见 3.5 读写操作。

DRM 在 SP 模式下各位宽都支持设置读写模式为 NW（Normal Write）模式，SP 模式下 18 bit 及以下数据端口位宽读写模式还可直接设置为 TW（Transparent Write）或 RBW（Read before Write）模式，但 SP 模式下 32 bit 及以上端口数据位宽禁止直接设置读写模式为 TW 和 RBW 模式，需要设置为默认的 NW 模式，具体支持情况如表 5-4 所示：

表 5-4 SP 模式读写模式设置支持情况

数据端口位宽	是否支持	读写模式实现方式
1（18K/36K）	支持	GTP 中可直接设置模式参数为 SP 模式，18K 及 36K 的 18 bit 及以下位宽下读写模式可直接设置为 NW、TW、RBW
2（18K/36K）	支持	
4（18K/36K）	支持	
8（18K/36K）	支持	
9（18K/36K）	支持	
16（18K/36K）	支持	
18（18K/36K）	支持	
32（18K/36K）	不支持	GTP 中 SP 模式的 32/36 bit 位宽禁止直接设置读写模式，需要使用默认的 NW；TW、RBW 需要额外配置，具体方法见下一段
36（18K/36K）	不支持	
64（36K）	不支持	GTP 中 SP 模式的 64/72 bit 位宽禁止设置读写模式，需要使用默认的 NW。如需在该场景下设置读写模式，需例化两个 32/36 bit 的 36K SP RAM 来拼接实现 64/72 bit，但部分情况下可能会造成所需 DRM 资源的增加
72（36K）	不支持	

DRM 禁止通过直接配置实现 32/36 bit SP 模式下的 TW 和 RBW 模式，需要额外的配置，具体实现方式如下：

- 18K SP RAM 模式的 32/36 bit：
  - Port A 和 Port B 同时配置为 16/18 bit DP 模式（分别针对 32/36 bit SP）；
  - Port A 和 Port B 配置为相同的写模式；其他配置位也需要相同；
  - Port A 和 Port B 的 CE，CS，WE，ORCE，RST，CLK 等控制信号分别并联起来；
  - DIA、DIB 并联作为数据输入；DOA、DOB 并联作为数据输出；
  - Port A 和 Port B 地址输入端口需同时接地址输入；ADDRA[4]和 ADDRB[4]需分别接 0 或者 1（相反），如图 5-2 所示：

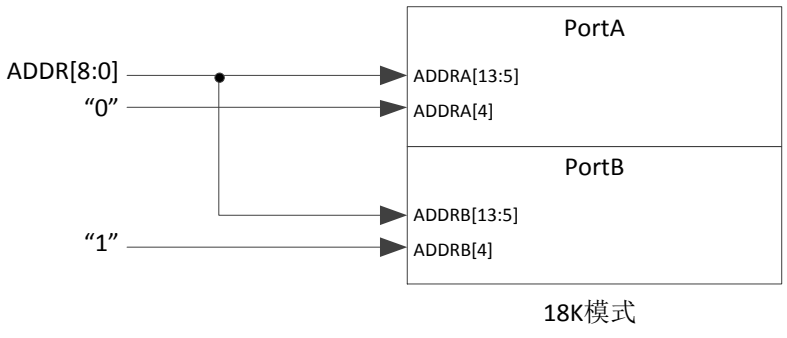

图 5-2 18K SP 模式的 32/36 bit 模式下 TW 和 RBW 模式地址线配置

  - BWEA、BWEB 信号的连接需要分别参考两个端口 DP 16/18 bit 模式下的连接.
- 36K SP RAM 模式的 32/36 bit：
  - A、B 两端口配置为 32/36 bit DP 模式（分别针对 32/36 bit SP）；
  - 只使用其中一个端口进行读写，另一端口写使能接 0，读数据端口悬空，如图 5-3

所示：

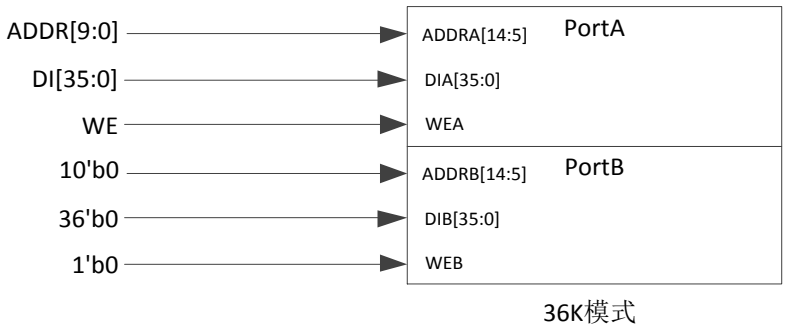

图 5-3 36K SP 模式的 36 bit 模式下 TW 和 RBW 模式端口信号配置

- 字节使能信号的连接需要参考进行读写的端口 DP 32/36 bit 模式下的连接。

➢ 36K SP RAM 模式的 64/72 bit：

36K 模式下的 64/72 bit 禁止配置读写模式为 TW 或 RBW，如在该场景下需要使用 TW 或 RBW 模式，需要例化两个 32/36 bit 的 36K SP RAM（按上文进行配置）来进行数据端口拼接实现 64/72 bit，如图 5-4 所示：

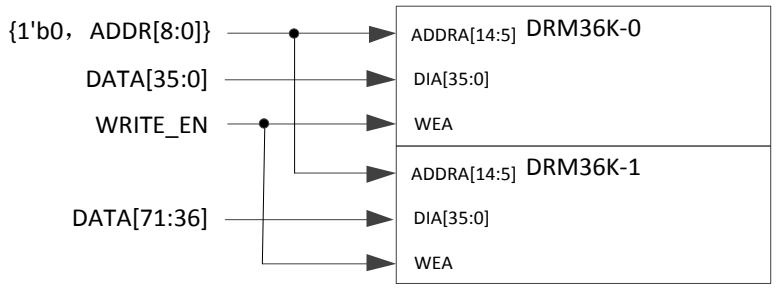

图 5-4 36K SP 模式的 72 bit 模式下 TW 和 RBW 模式端口信号配置

5.6 字节使能功能

SP 模式的字节使能功能与 DP 模式类似，详见 3.6 字节使能功能。

5.7 内部寄存器

SP 模式的内部寄存器和 DP 模式相同，详见 3.7 内部寄存器。

5.8 硬级联功能

SP 模式的硬级联功能和 DP 模式相同，详见 3.8 硬级联功能。

5.9 应用示例

5.9.1 单个 36K DRM 配置

本小节举例说明了单个 36K DRM 的 GTP 配置步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例为 SP 模式，读写端口配置为 512x64，使能输出寄存器，读写模式 NW。18K DRM 的配置步骤与之类似。

对单个 36K DRM 按如下步骤进行配置：

a. 按表 5-5 所述对 DRM 的参数进行配置：

表 5-5 单个 36K DRM SP 模式参数配置

参数名称	配置值	说明
DATA_WIDTH_A	64	将 A 端口配置为 512x64 模式
DATA_WIDTH_B		将 B 端口配置为 512x64 模式
WRITE_MODE_A	"NORMAL_WRITE"	将 A 端口读写模式配置为 NW
WRITE_MODE_B		将 B 端口读写模式配置为 NW
DOA_REG	1	使能 A 端口输出寄存器
DOB_REG		使能 B 端口输出寄存器
RAM_MODE	"SINGLE_PORT"	将 DRM 配置为单口模式

b. 按表 5-6 所示对 DRM 的端口进行连接：

表 5-6 单个 36K DRM SP 模式端口连接

端口名称	连接信号	说明
ADDRA[15:0]	{1'b1,addr[8:0],6'b0}	A/B 端口地址接相同地址信号，将输入地址信号 addr[8:0]连接到 ADDRA[14:6]，ADDRA[15]、接高电平，ADDRA[5:0]接低电平，ADDRB 连接相同，详细地址连接说明见 9.1 地址和数据端口 Mapping
ADDRB[15:0]		
ADDRA_HOLD	1'b0	不使用 A/B 端口地址锁存功能，接低电平，如果使用地址锁存功能 A/B 端口需接相同信号
ADDRB_HOLD		
DIA[35:0]	{1'b0, di[31:24], 1'b0, di[23:16], 1'b0, di[15:8], 1'b0, di[7:0]}	A/B 端口拼接实现 64bit 写数据端口，DIA 输入低 32bit 写数据 di[31:0]，详细数据端口连接说明见 9.1 地址和数据端口 Mapping
DIB[35:0]	{1'b0, di[63:56], 1'b0, di[55:48], 1'b0, di[47:40], 1'b0, di[39:32]}	A/B 端口拼接实现 72bit 写数据端口，DIB 输入高 32bit 写数据 di[63:32]，DIB[8]、DIB[17]、DIB[26]、DIB[35]为字节附加信息位，详见 9.4 字节附加信息位
CSA[2:0]	3'b0	未使用地址扩展功能，接低电平，如果使用地址扩展功能 A/B 端口需接相同信号
CSB[2:0]		
BWEA[7:0]	8'hff	未使用字节使能功能，将 BWEA 连接到高电平
BWEB[3:0]	4'h0	BWEB 未作字节使能信号，连接到低电平

端口名称	连接信号	说明
DOA[35:0]	{1'bz, do[31:24], 1'bz, do[23:16], 1'bz, do[15:8], 1'bz, do[7:0]}	A/B 端口拼接实现 64bit 读数据端口，DOA 输出低 32bit 读数据 do[31:0]，DOA[8]、DOA[17]、DOA[26]、DOA[35]则悬空
DOB[35:0]	{1'bz, do[63:56], 1'bz, do[55:48], 1'bz, do[47:40], 1'bz, do[39:32]}	A/B 端口拼接实现 64bit 读数据端口，DOB 输出高 32bit 读数据 do[63:32]，DOB[8]、DOB[17]、DOB[26]、DOB[35]则悬空

c. 读写端口其余信号：将读写端口的时钟、时钟使能、输出寄存器时钟使能、数据寄存器复位、写使能分别连接到相应的 GTP 相应 A/B 端口上，上述信号 A/B 端口都需相同连接；

d. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

e. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.DATA_WIDTH_A  (64),
.DATA_WIDTH_B  (64),
.WRITE_MODE_A  ("NORMAL_WRITE"),
.WRITE_MODE_B  ("NORMAL_WRITE"),
.DOA_REG       (1),
.DOB_REG       (1),
.RAM_MODE      ("SINGLE_PORT")
) GTP_DRM36K_E1_inst (
.DOA           (doa[35:0] ), // OUTPUT[35:0]
.DOB           (dob[35:0] ), // OUTPUT[35:0]
.ADDRA         ({1'b1,addr[8:0],6'b0}), // INPUT[15:0]
.ADDRB         ({1'b1,addr[8:0],6'b0}), // INPUT[15:0]
.BWEA          (8'hff     ), // INPUT[7:0]
.BWEB          (4'h0      ), // INPUT[3:0]
.CSA           (3'b0      ), // INPUT[2:0]
.CSB           (3'b0      ), // INPUT[2:0]
.DIA           ({1'b0,di[31:24],1'b0,di[23:16], 1'b0, di[15:8],1'b0,di[7:0]}), // INPUT[35:0]
.DIB           ({1'b0,di[63:56],1'b0,di[55:48],1'b0,di[47:40],1'b0,di[39:32]}), // INPUT[35:0]
.ADDRA_HOLD    (1'b0      ), // INPUT
.ADDRB_HOLD    (1'b0      ), // INPUT
.CEA           (ce        ), // INPUT
.CEB           (ce        ), // INPUT
.CLKA          (clk       ), // INPUT
.CLKB          (clk       ), // INPUT
.ORCEA         (orce      ), // INPUT
.ORCEB         (orce      ), // INPUT
.RSTA          (rst       ), // INPUT
.RSTB          (rst       ), // INPUT
.WEA           (we        ), // INPUT
.WEB           (we        )  // INPUT
);
assign do[63:0] = {dob[34:27],dob[25:18],dob[16:9],dob[7:0],doa[34:27],doa[25:18],doa[16:9],doa
[7:0]};
```

5.9.2 多个 36K DRM 级联配置

SP 模式多个 36K DRM 级联与 DP 模式相同，详见 3.9.2 多个 36K DRM 级联配置。

5.9.3 多个 36K DRM 硬级联配置

SP 模式多个 36K DRM 硬级联与 DP 模式相同，详见 3.9.3 多个 36K DRM 硬级联配置。

6 ROM 模式

6.1 模式介绍

RAM 的端口模式由参数 RAM_MODE 决定，当参数 RAM_MODE 的值为"ROM"时，RAM 进入 ROM 模式。本文中将详细介绍 36K 的 ROM，18K 与 36K 结构与功能基本一致，仅位宽不同。

ROM模式支持只支持数据读出，不支持数据写入。

6.2 数据端口

DRM 可以配置为 ROM，通过配置接口可以初始化 ROM 内容。ROM 模式对 18K 及 36K 模式的端口只读。如图 6-1 所示为 ROM 模式的结构图。

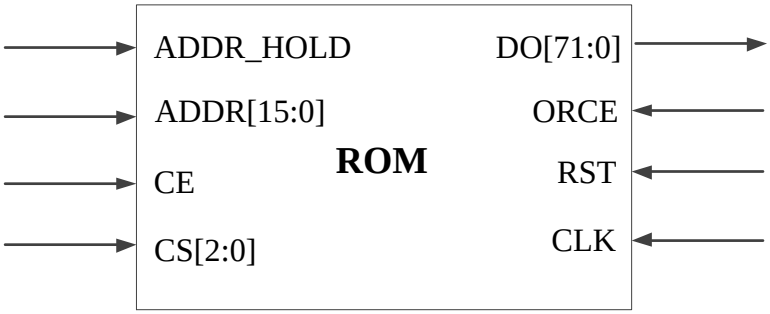

图 6-1 ROM 模式数据端口

表 6-1 罗列了 ROM 模式下的端口名称和端口的描述。

表 6-1 ROM 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
ADDR_HOLD	输入	地址锁存信号	DO	输出	读数据输出
ADDR	输入	地址输入	ORCE	输入	输出寄存器时钟使能
CE	输入	输入寄存器时钟使能	RST	输入	输出寄存器复位
CS	输入	地址扩展	CLK	输入	端口时钟

注：1.ROM 模式的 x64/x72 位宽下，DI 由 A/B 端口的 DIA/DIB 拼接而成，DO 同理，详见 9.1 地址和数据端口 Mapping；
2.18K DRM 无 CSA/CSB 端口，不支持地址扩展，如需 DRM 级联进行地址深度扩展，建议使用 36K DRM。

6.3 位宽组合

RAM 的端口位宽由 GTP 中的参数 DATA_WIDTH_A/DATA_WIDTH_B 决定，例如，当参数 DATA_WIDTH_A 的值为 4 时，A 端口数据位宽被设置为 4 bit。ROM 模式的 A、B 端

口需设置为相同数据位宽。

如表 6-2 所示为 36K DRM 模式下 ROM 模式允许的位宽。

表 6-2 36K DRM 模式 ROM 模式允许的位宽

模式	32Kx1	16Kx2	8Kx4	4Kx8	2Kx16	1Kx32	512x64	4Kx9	2Kx18	1Kx36	512x72
ROM	√	√	√	√	√	√	√	√	√	√	√

注：√表示支持的位宽组合；

如表 6-3 所示为 18K DRM 模式下 ROM 模式允许的位宽。

表 6-3 18K DRM 模式 ROM 模式允许的位宽

模式	16Kx1	8Kx2	4Kx4	2Kx8	1Kx16	512x32	2Kx9	1Kx18	512x36
ROM	√	√	√	√	√	√	√	√	√

注：√表示支持的位宽组合。

6.4 时序参数

ROM模式下，DRM不支持写操作，读操作的时序参数及时序与DP模式相同，详见3.4时序参数。

6.5 内部寄存器

ROM 模式的内部寄存器和 DP 模式相同，详见 3.7 内部寄存器。

6.6 硬级联功能

ROM 模式的硬级联功能和 DP 模式相同，详见 3.8 硬级联功能。

6.7 应用示例

6.7.1 单个 36K DRM 配置

本小节举例说明了单个 36K DRM 的 GTP 配置步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成 DRM 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例为 ROM 模式，读写端口配置为 2Kx18，使能输出寄存器。18K DRM 的配置步骤与之类似。

对单个 36K DRM 按如下步骤进行配置：

a. 按表 6-4 所述对 DRM 的参数进行配置：

表 6-4 单个 36K DRM ROM 模式参数配置

参数名称	配置值	说明
DATA_WIDTH_A	18	将 A 端口配置为 2Kx18 模式
DATA_WIDTH_B		将 B 端口配置为 2Kx18 模式
DOA_REG	1	使能 A 端口输出寄存器
DOB_REG		使能 B 端口输出寄存器
RAM_MODE	"ROM"	将 DRM 配置为 ROM 模式
INIT_00	{18'h0000f, ……. 18'h00003, 18'h00002, 18'h00001, 18'h00000}	将 ROM 的初始值参数设置为对应的地址值,即 INIT_00 的 1-18bit 对应地址 0，设置为 0；19-36bit 对应地址 1 则设置为 1，以此类推，具体初始化参数映射规则详见 9.2 初始化配置参数映射；
…..	……	
INIT_7F	{18'h007ff, ……. 18'h007f3, 18'h007f2, 18'h007f1, 18'h007f0}	
RSTA_VAL	{18'h01234}	设置输出数据复位值为 18'h01234
RSTB_VAL		

b. 按表 6-5 所示对 DRM 的端口进行连接：

表 6-5 单个 36K DRM ROM 模式端口连接

端口名称	连接信号	说明
ADDRA[15:0]	{1'b1,addr[10:0],4'b0}	A/B 端口地址接相同地址信号，将输入地址信号 addr[10:0] 连接到 ADDRA[14:4]，ADDRA[15]接高电平，ADDRA[3:0] 接低电平，ADDRB 连接相同，详细地址连接说明见 9.1 地址和数据端口 Mapping
ADDRB[15:0]		
ADDRA_HOLD	1'b0	不使用 A/B 端口地址锁存功能，接低电平，如果使用地址锁存功能 A/B 端口需要接相同信号
ADDRB_HOLD		
DIA[35:0]	悬空	ROM 不支持写操作，端口 DIA/DIB 悬空
DIB[35:0]		
CSA[2:0]	3'b0	未使用地址扩展功能，接低电平，如果使用地址控制功能 A/B 端口需要接相同信号
CSB[2:0]		
BWEA[7:0]	8'h0	ROM 不支持写操作，将 BWEA/BWEB 连接到低电平
BWEB[3:0]	4'h0	
DOA[35:0]	do[17:0]	将输出数据信号 do[17:0]连接到 DOA[17:0]
DOB[35:0]	悬空	未进行数据拼接，DOB 悬空
WEA	1'b0	ROM 不支持写操作，将 WEA/WEB 连接到低电平
WEB		

c. 读写端口其余信号：将读写端口的时钟、时钟使能、输出寄存器时钟使能、数据寄存器复位分别连接到相应的 GTP 相应 A/B 端口上，上述信号 A/B 端口都需相同连接；

d. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

e. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见在上文

描述）。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.DATA_WIDTH_A  (18),
.DATA_WIDTH_B  (18),
.DOA_REG       (1),
.DOB_REG       (1),
.RAM_MODE      ("ROM"),
.RSTA_VAL      ({18'h01234}),
.RSTB_VAL      ({18'h01234}),
.INIT_00       ({18'h0000f, 18'h0000e, 18'h0000d, 18'h0000c, 18'h0000b, 18'h0000a, 18'h00009,
18'h00008, 18'h00007, 18'h00006, 18'h00005, 18'h00004, 18'h00003, 18'h00002, 18'h00001, 18'h00000}),
…
…
.INIT_7F       ({18'h007ff, 18'h007fe, 18'h007fd, 18'h007fc, 18'h007fb, 18'h007fa, 18'h007f9,
18'h007f8, 18'h007f7, 18'h007f6, 18'h007f5, 18'h007f4, 18'h007f3, 18'h007f2, 18'h007f1, 18'h007f0})
) GTP_DRM36K_E1_inst (
.DOA           (do[17:0]  ), // OUTPUT[35:0]
.DOB           (          ), // OUTPUT[35:0]
.ADDRA         ({1'b1,addr[10:0],4'b0}), // INPUT[15:0]
.ADDRB         ({1'b1,addr[10:0],4'b0}), // INPUT[15:0]
.BWEA          (8'b0      ), // INPUT[7:0]
.BWEB          (4'b0      ), // INPUT[3:0]
.CSA           (3'b0      ), // INPUT[2:0]
.CSB           (3'b0      ), // INPUT[2:0]
.DIA           (          ), // INPUT[35:0]
.DIB           (          ), // INPUT[35:0]
.ADDRA_HOLD    (1'b0      ), // INPUT
.ADDRB_HOLD    (1'b0      ), // INPUT
.CEA           (ce        ), // INPUT
.CEB           (ce        ), // INPUT
.CLKA          (clk       ), // INPUT
.CLKB          (clk       ), // INPUT
.ORCEA         (orce      ), // INPUT
.ORCEB         (orce      ), // INPUT
.RSTA          (rst       ), // INPUT
.RSTB          (rst       ), // INPUT
.WEA           (1'b0      ), // INPUT
.WEB           (1'b0      )  // INPUT
);
```

6.7.2 多个 36K DRM 级联配置

ROM 模式多个 36K DRM 级联与 DP 模式相同，详见 3.9.2 多个 36K DRM 级联配置。

6.7.3 多个 36K DRM 硬级联配置

ROM 模式多个 36K DRM 硬级联与 DP 模式相同，详见 3.9.3 多个 36K DRM 硬级联配置。

7 FIFO 模式

7.1 模式介绍

DRM_FIFO 由 DRM 和 DRM_FIFO_CTRL 模块组成，DTM_FIFO_CTRL 模块包括：读写指针生成、状态标志生成逻辑。FIFO 的顶层结构框图如图 7-1 所示：

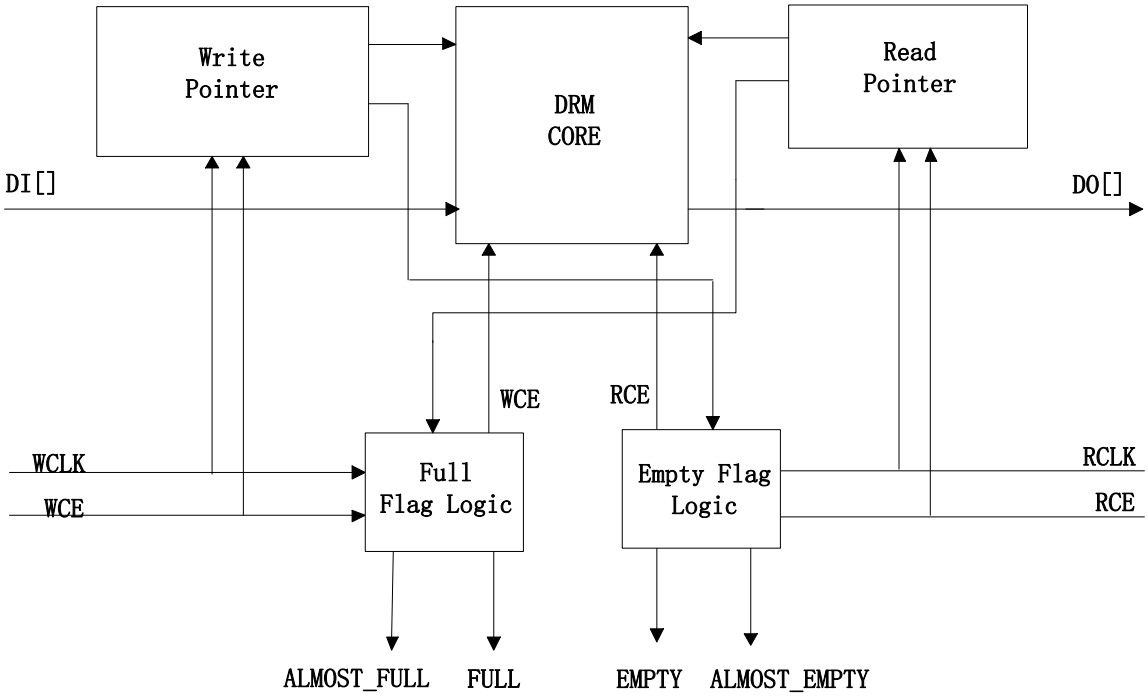

图 7-1 FIFO 顶层结构框图

用户可通过例化 GTP_DRM 配置为 SDP 模式再通过用户逻辑实现 DRM_FIFO_CTRL 模块或直接例化 GTP_FIFO 来实现 DRM_FIFO。本文将详细介绍 36K 下的 GTP_FIFO，18K 与 36K 结构与功能基本一致，仅位宽不同。

GTP_FIFO 分为同步和异步 FIFO，由参数 SYNC_FIFO 决定，当参数 SYNC_FIFO 的值为"TRUE"时，进入同步 FIFO 模式；当参数 SYNC_FIFO 的值为"FALSE"时，进入异步 FIFO 模式。

异步 FIFO 模式如图 7-2 所示，在该模式下，当写使能有效，且 FIFO 不为满时，在写时钟的上升沿向 FIFO 写入数据；当读使能有效，且 FIFO 不为空时，在读时钟的上升沿从 FIFO 读出数据。当 FIFO 写满时产生满信号，满信号同步于写时钟域。当 FIFO 读空时产生空信号，空信号同步于读时钟域。

同步 FIFO 模式如图 7-3 所示，在该模式下，读写时钟为同一个时钟。当写使能有效，且 FIFO 不为满时，在时钟的上升沿向 FIFO 写入数据；当读使能有效，且 FIFO 不为空时，

在时钟的上升沿从 FIFO 读出数据。当 FIFO 写满时产生满信号，当 FIFO 读空时产生空信号。

7.2 数据端口

异步 FIFO 模式下结构图如图 7-2 所示：

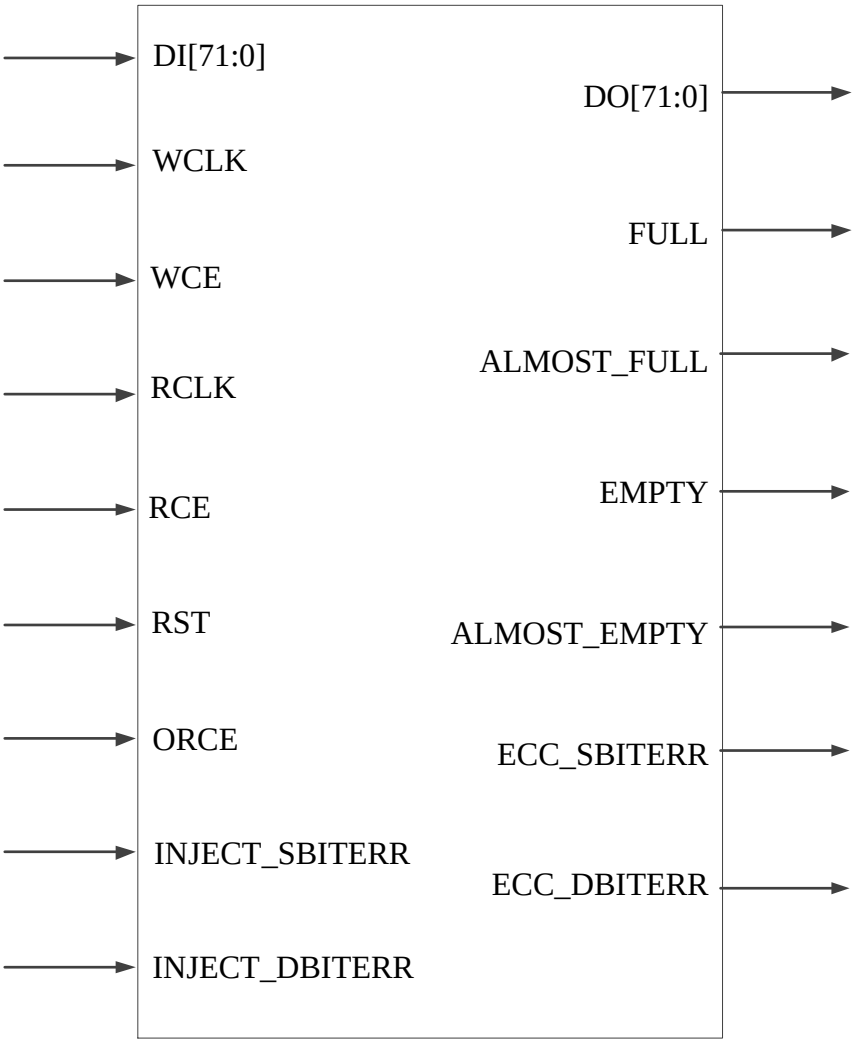

图 7-2 异步 FIFO 模式

表 7-1 罗列了异步 FIFO 模式下的端口名称和端口的描述。

表 7-1 异步 FIFO 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
DI	输入	写数据输入	DO	输出	读数据输出
WCLK	输入	写端口时钟	RCLK	输入	读端口时钟
WCE	输入	写端口时钟使能	RCE	输入	读端口时钟使能
RST	输入	输出寄存器复位	ORCE	输入	输出寄存器时钟使能
FULL	输出	满标志信号	EMPTY	输出	空标志信号
ALMOST_FULL	输出	将满标志信号	ALMOST_EMPTY	输出	将空标志信号
INJECT_SBITERR	输入	ECC 模式单比特错误注入	ECC_SBITERR	输出	ECC 模式单比特错误标志

端口名称	方向	描述	端口名称	方向	描述
INJECT_DBITERR	输入	ECC 模式双比特错误注入	ECC_DBITERR	输出	ECC 模式双比特错误标志

同步 FIFO 模式下结构图如图 7-3 所示：

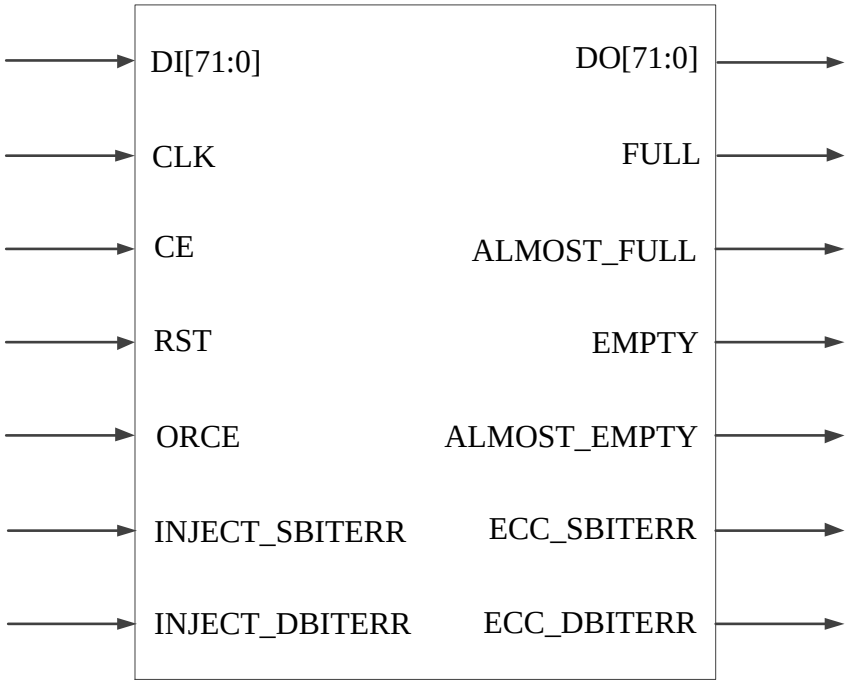

图 7-3 同步 FIFO 模式

表 7-2 罗列了同步 FIFO 模式下的端口名称和端口的描述。

表 7-2 同步 FIFO 端口名称及描述

端口名称	方向	描述	端口名称	方向	描述
DI	输入	写数据输入	DO	输出	读数据输出
CLK	输入	端口时钟	CE	输入	端口时钟使能
RST	输入	输出寄存器复位	ORCE	输入	输出寄存器时钟使能
FULL	输出	满标志信号	EMPTY	输出	空标志信号
ALMOST_FULL	输出	将满标志信号	ALMOST_EMPTY	输出	将空标志信号
INJECT_SBITERR	输入	ECC 模式单比特错误注入	ECC_SBITERR	输出	ECC 模式单比特错误标志
INJECT_DBITERR	输入	ECC 模式双比特错误注入	ECC_DBITERR	输出	ECC 模式双比特错误标志

7.3 位宽组合

GTP_FIFO 的端口位宽由 GTP 中的参数 DATA_WIDTH 决定，例如，当参数 DATA_WIDTH 的值为 4 时，端口数据位宽被设置为 4 bit。GTP_FIFO 不支持混合位宽。

如表 7-3 所示为 36K GTP_FIFO 允许的位宽组合。

表 7-3 36K GTP_FIFO 允许的位宽

模式	32Kx1	16Kx2	8Kx4	4Kx8	2Kx16	1Kx32	512x64	4Kx9	2Kx18	1Kx36	512x72
FIFO	√	√	√	√	√	√	√	√	√	√	√

注：√表示支持的位宽组合；

如表 7-4 所示为 18K GTP_FIFO 允许的位宽组合。

表 7-4 18K GTP_FIFO 允许的位宽

模式	16Kx1	8Kx2	4Kx4	2Kx8	1Kx16	512x32	2Kx9	1Kx18	512x36
FIFO	√	√	√	√	√	√	√	√	√

注：√表示支持的位宽组合。

7.4 读写操作

本节读写操作说明仅适用于 GTP_FIFO，即 GTP_FIFO36K_E1 和 GTP_FIFO18K_E1。

7.4.1 FIFO 时序参数

表 7-5 为 36K GTP_FIFO 的典型时序参数及说明，后文所描述的时序参数及时序图都以 36K GTP_FIFO 为例，18K GTP_FIFO 的时序参数和时序图与之类似，详细的时序参数说明见《*DS04001_Logos2 系列 FPGA 器件数据手册*》。

表 7-5 GTP_FIFO 典型时序参数

参数	控制信号	说明
$T_{su\_fifo\_wctl}$	WR_EN	写使能信号建立时间
$T_{hd\_fifo\_wctl}$		写使能信号保持时间
$T_{su\_fifo\_rctl}$	RD_EN	读使能信号建立时间
$T_{hd\_fifo\_rctl}$		读使能信号保持时间
$T_{su\_36K\_d}$	DI	输入数据建立时间
$T_{hd\_36K\_d}$		输入数据保持时间
T_{co_36K}	RCLK to Q	数据输出相对时钟沿的延时（输出无寄存）
$T_{co\_36K\_reg}$		数据输出相对时钟沿的延时（输出寄存）
$T_{co\_flag\_full}$	WCLK to ALMOSTFULL/FULL	各标志信号相对时钟沿的延时
$T_{co\_flag\_empty}$	RCLK to ALMOSTEMPTY/EMPTY	

7.4.2 写入时序

EMPTY 信号指示 FIFO 为空，当 WCE 有效并且有效数据写入后，同步 FIFO 时，EMPTY 信号在 1 个 CLK 时钟周期后置”0”；写入异步 FIFO 时，有效数据写入（图 7-4 中上升沿①）后 EMPTY 信号在第 3 个 RCLK 时钟上升沿置零（图 7-4 中上升沿②）。当 WCE 继续有效，ALMOST_EMPTY 信号根据 ALMOST_FULL_OFFSET 参数的配置延迟置零（图 7-4 中上升

沿④）。若 WCLK 上升沿与 RCLK 上升沿接近，则 EMPTY、ALMOST_EMPTY 信号可能会延迟一个 RCLK 时钟周期置零。

写入空异步 FIFO 时序图如图 7-4 所示：

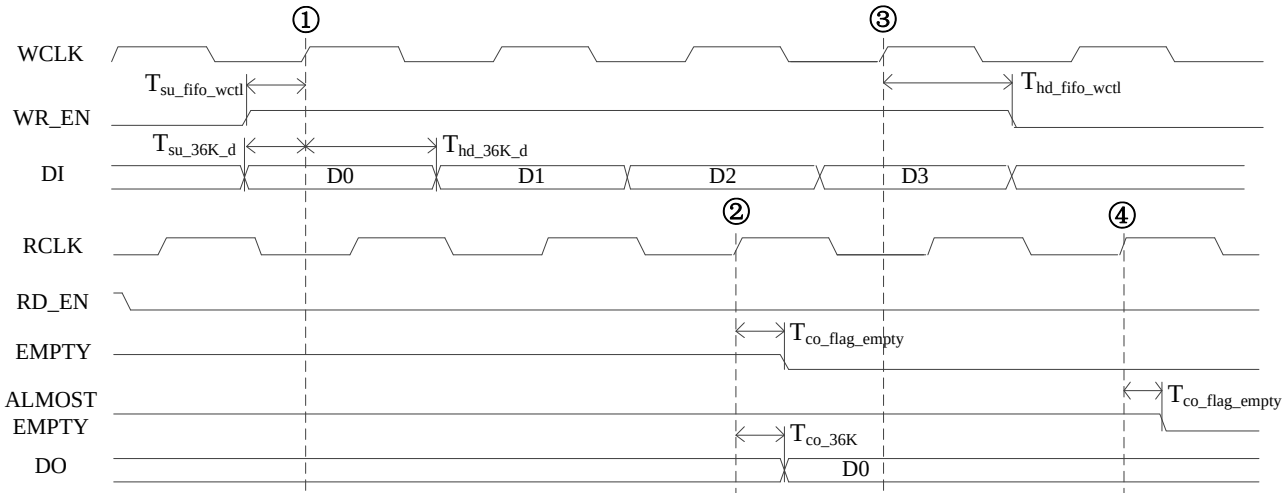

图 7-4 写入空异步 FIFO 时序图

FIFO 将满时，根据 ALMOST_FULL_OFFSET 参数的配置，写指针与读指针位置之差值等于设置值时，ALMOST_FULL 标志将在 WCLK 的上升沿置 1（图 7-5 中上升沿②）。当 FIFO 写满时（图 7-5 中上升沿③），写指针不再增加，不再进行写入操作，FULL 信号立刻置 1。

写入将满异步 FIFO 时序图如图 7-5 所示：

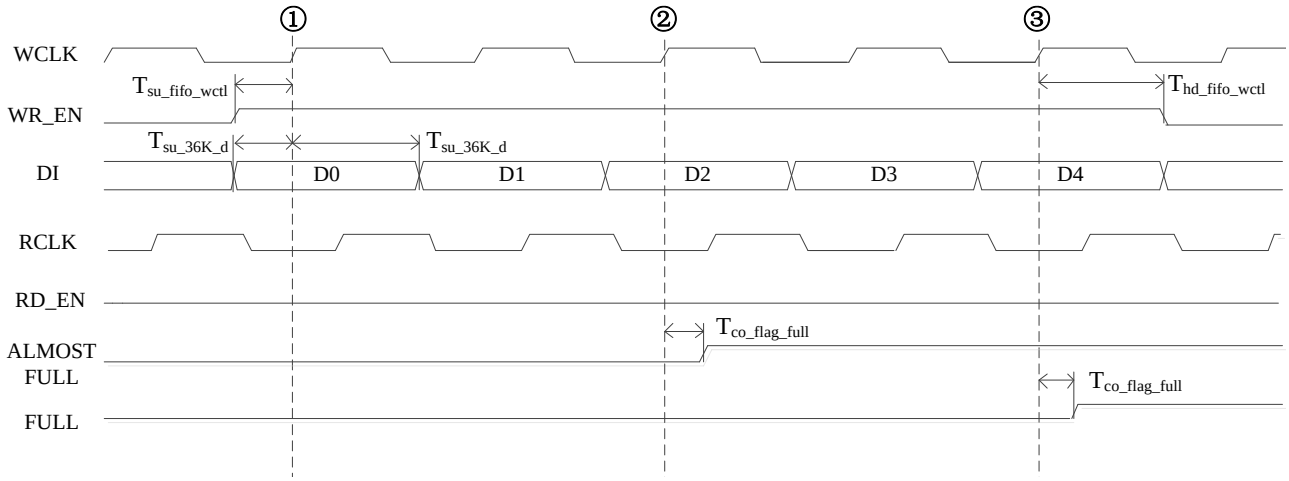

图 7-5 写入将满异步 FIFO 时序图

7.4.3 读出时序

FULL 信号指示 FIFO 为满，当 RCE 有效读出数据后，同步 FIFO 时，FULL 信号在 1 个 CLK 时钟周期后置”0”；异步 FIFO 时，有效数据读出（图 7-6 中上升沿①）后 FULL 信号在第 3 个 WCLK 时钟上升沿置零（图 7-6 中上升沿②）。当 RCE 继续有效，ALMOST_FULL

信号根据 ALMOST_FULL_OFFSET 参数的配置延迟置零（图 7-6 中上升沿③）。若 WCLK 上升沿与 RCLK 上升沿接近，则 FULL、ALMOST_FULL 信号可能会延迟一个 WCLK 时钟周期置零。

满异步 FIFO 读出时序图如图 7-6 所示：

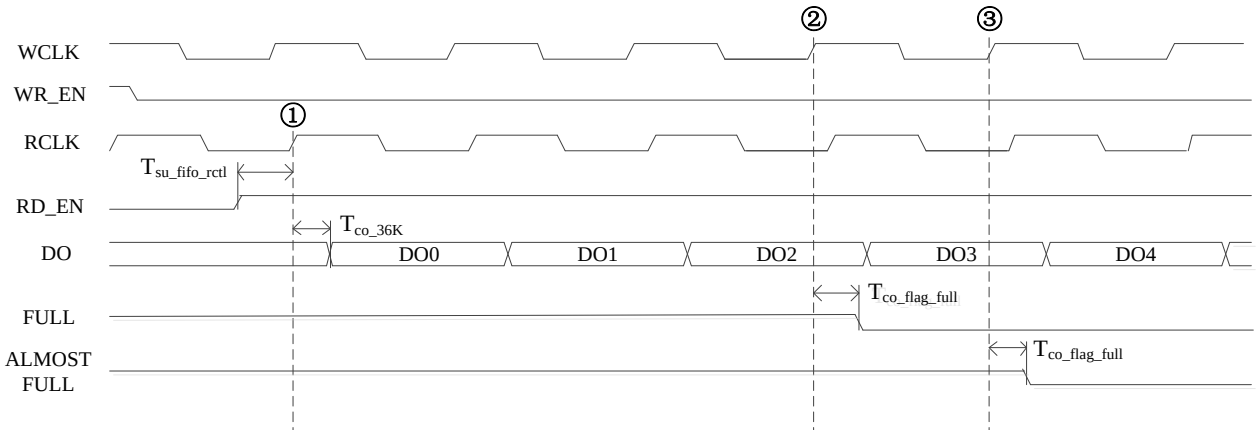

图 7-6 满异步 FIFO 读出时序图

FIFO 将空时，根据 ALMOST_FULL_OFFSET 参数的配置，写指针与读指针位置之差值等于设置值时 ALMOST_EMPTY 提前置 1（图 7-7 中上升沿②）。当 FIFO 读空时（图 7-7 中上升沿③），读指针不再增加，EMPTY 信号立刻置 1。

将空异步 FIFO 读出时序图如图 7-7 所示：

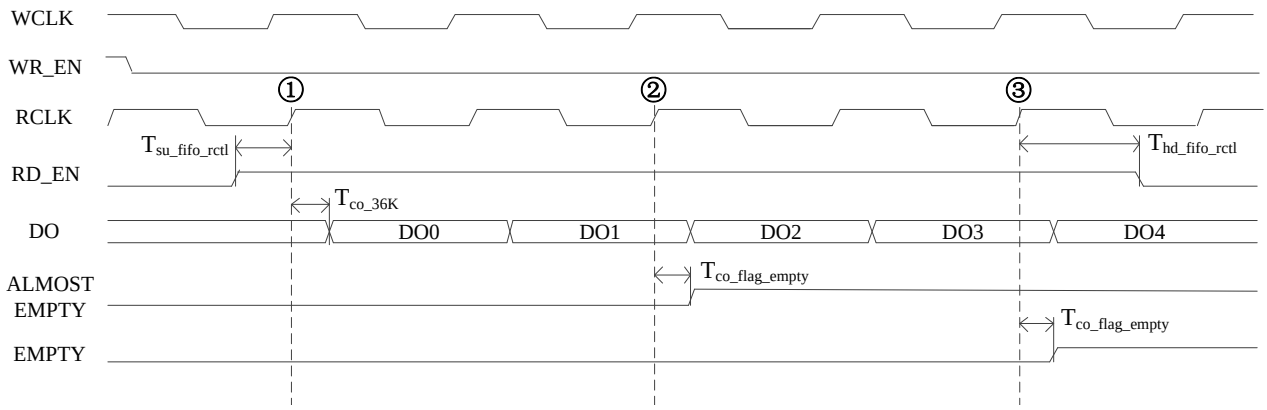

图 7-7 将空异步 FIFO 读出时序图

7.4.4 标志信号复位时序

复位信号 RST 是用来重置所有标志信号的异步复位信号，在其置 1 后，标志信号 EMPTY、ALMOST_EMPTY 置 1，FULL 和 ALMOST_FULL 置 0。RST 置 1 复位后应保持至少 5 个读和写时钟周期，以保证 FIFO 所有内部状态和标志信号复位到正确的值，所有标志信号的复位时序图如图 7-8 所示：

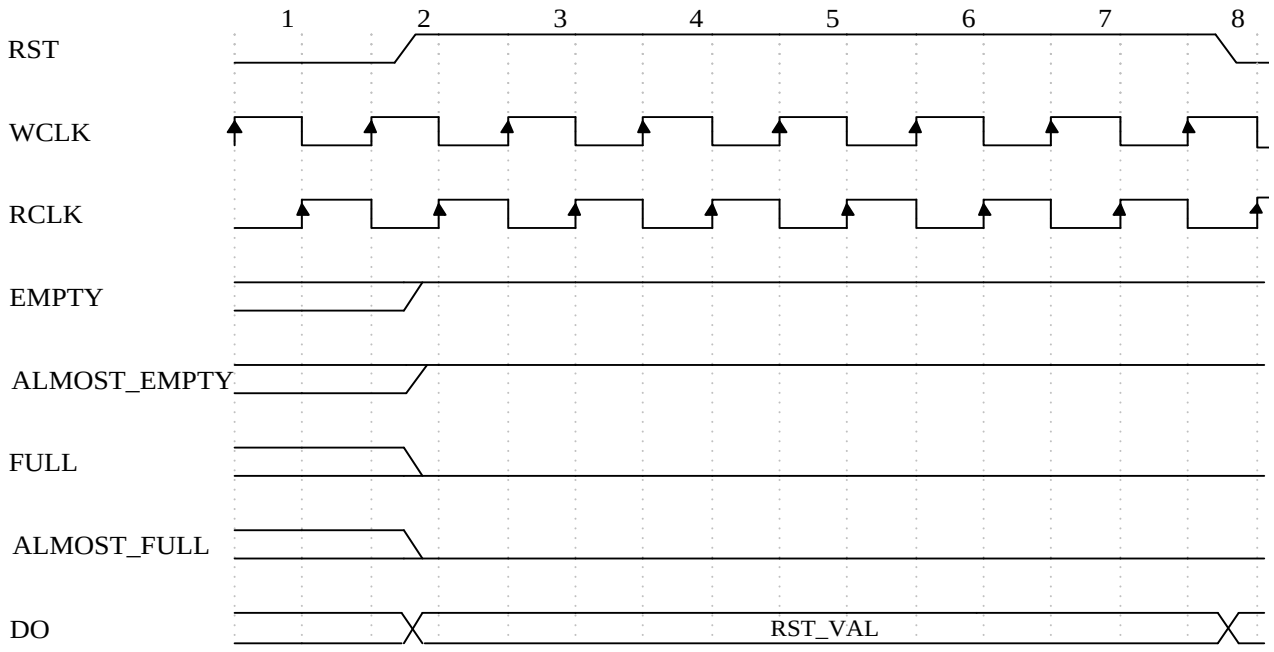

图 7-8 FIFO 标志信号复位时序图

7.5 内部寄存器

FIFO 模式的内部寄存器和 DP 模式相同，详见 3.7 内部寄存器。

7.6 应用示例

本小节举例说明了单个 36K GTP_FIFO 的配置步骤，用户还可以通过 Pango Design Suite 软件内嵌的 IP Compiler 工具直接生成基于 GTP_DRM 的 FIFO 的 IP，详见 IP Compiler 工具内的 DRM/FIFO IP 用户指南 UG041002。

示例为异步 FIFO 模式，端口数据位宽配置为 1Kx36，使能输出寄存器。18K FIFO 的配置步骤与之类似。

对单个 36K FIFO 按如下步骤进行配置：

a. 按表 7-6 所述对 FIFO 的参数进行配置：

表 7-6 单个 36K FIFO 参数配置

参数名称	配置值	说明
DATA_WIDTH	36	将端口配置为 1Kx36 模式
DO_REG	1	使能端口输出寄存器
SYNC_FIFO	"FALSE"	异步 FIFO 模式
USE_EMPTY	1	空标志使能

参数名称	配置值	说明
USE_FULL	1	满标志使能
ALMOST_EMPTY_OFFSET	'd4	将将空信号设置为 FIFO 内部储存数据个数小于等于 4 个时置 1
ALMOST_FULL_OFFSET	'd1020	将将满信号设置为 FIFO 内部储存数据个数大于等于 1020 个时置 1

b. 按表 7-7 所示对 FIFO 的端口进行连接：

表 7-7 单个 36K FIFO 端口连接

端口名称	连接信号	说明
DI[71:0]	di[35:0]	将输入数据信号 di[35:0]连接到 DI[35:0]，高位 DI 端口悬空或接低电平
DO[71:0]	do[35:0]	将输入数据信号 do[35:0]连接到 DO[35:0]，高位 DO 端口悬空

c. 读写端口其余信号：将读写端口的时钟、读写时钟使能、输出寄存器时钟使能、数据寄存器复位、标志信号分别连接到相应的 GTP 相应读写端口上；

d. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

e. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_FIFO36K_E1 #(
.DATA_WIDTH           (36),
.DO_REG               (1),
.ALMOST_FULL_OFFSET   ('d1020),
.ALMOST_EMPTY_OFFSET  ('d4),
.USE_EMPTY            (1),
.USE_FULL             (1),
.SYNC_FIFO            ("FALSE")
) GTP_FIFO36K_E1_inst (
.DO                   (do[35:0]    ), // OUTPUT[71:0]
.DI                   (di[35:0]    ), // INPUT[71:0]
.ALMOST_EMPTY         (almost_empty), // OUTPUT
.ALMOST_FULL          (almost_full ), // OUTPUT
.EMPTY                (empty       ), // OUTPUT
.FULL                 (full        ), // OUTPUT
.ORCE                 (orce        ), // INPUT
.RCE                  (rce         ), // INPUT
.RCLK                 (rclk        ), // INPUT
.RST                  (rst         ), // INPUT
.WCE                  (wce         ), // INPUT
.WCLK                 (wclk        )  // INPUT
);
```

8 ECC 模式

8.1 模式介绍

仅当 DRM 在单 36K 模式下，配置为 SDP/FIFO 512x72 存储器模式下，支持 72bits 宽度数据 ECC 单 bit 纠错双 bits 检错，其中有效数据位为 64bit，另外的 8bits 为 ECC 校验位存储，同时输出 ECC_SBITERR（单 bit 纠错指示标志）、ECC_DBITERR（双 bits 检错指示标志）、ECC 编码以及 ECC 模式下的读地址（ECC 编码、读地址仅在 SDP 模式下支持）输出。

配置 ECC 模式需要将 GTP_DRM36K_E1（配置为 SDP 模式）或 GTP_FIFO36K_E1 的端口读写数据位宽设置为 72，然后设置参数 ECC_WRITE_EN、ECC_READ_EN 为”TRUE”。

- 当 ECC 编码和 ECC 解码使能时：
  - 64 位数据读出时存在单 bit 错误时，可纠正错误并读出正确数据，同时 ECC_SBITERR 状态标志置位。
  - 64 位数据读出时存在双 bits 错误时，不能纠正错误读出数据，ECC_DBITERR 状态标志置位。
- ECC 编码模块支持用户插入单 bit 或者双 bits 错误。
  - 单 bit 错误插入到 WD[30]。
  - 双 bits 错误插入到 WD[62]和 WD[30]。

8.2 数据端口

SDP RAM 的 ECC 模式结构图如图 8-1 所示：

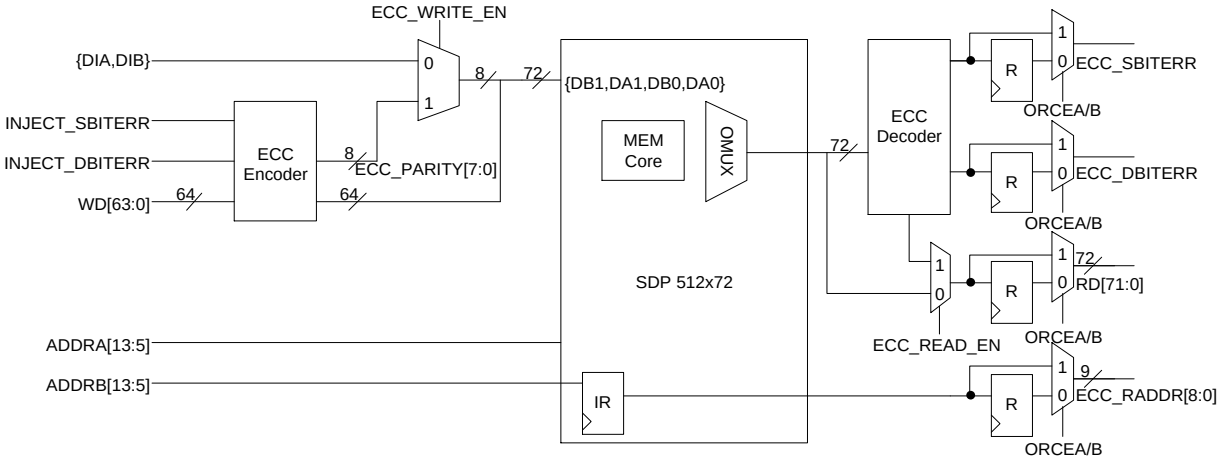

图 8-1 SDP RAM 的 ECC 模式结构图

WD[63:0]={DIB[34:27, 25:18, 16:9, 7:0],DIA[34:27, 25:18, 16:9, 7:0]}为 ECC 模式 64 位写

入有效数据；RD[71:0]={DOB, DOA}为 ECC 模式 72 位读出数据。其中 ECC_PARITY={DOB[35, 26, 17, 8], DOA[35, 26, 17, 8]}为 8 位 ECC 校验码输出，dout[63:0]={DOB[34:27, 25:18, 16:9, 7:0], DOA[34:27, 25:18, 16:9, 7:0]}为该模式下 64 位有效读出数据。

8.3 读写操作

SDP RAM 的 ECC 模式读写时序图如图 8-2 和图 8-3 所示，A'=A，注入的单比特错误被纠正，单 bit 错误标志信号置 1；C''[63, 61:31, 29:0]=C[63, 61:31, 29:0]，C''[62, 30]≠C[62, 30]，注入的双比特错误未被纠正，但被检测出来，双 bits 错误标志信号置 1。FIFO 的 ECC 模式读写时序与之类似。

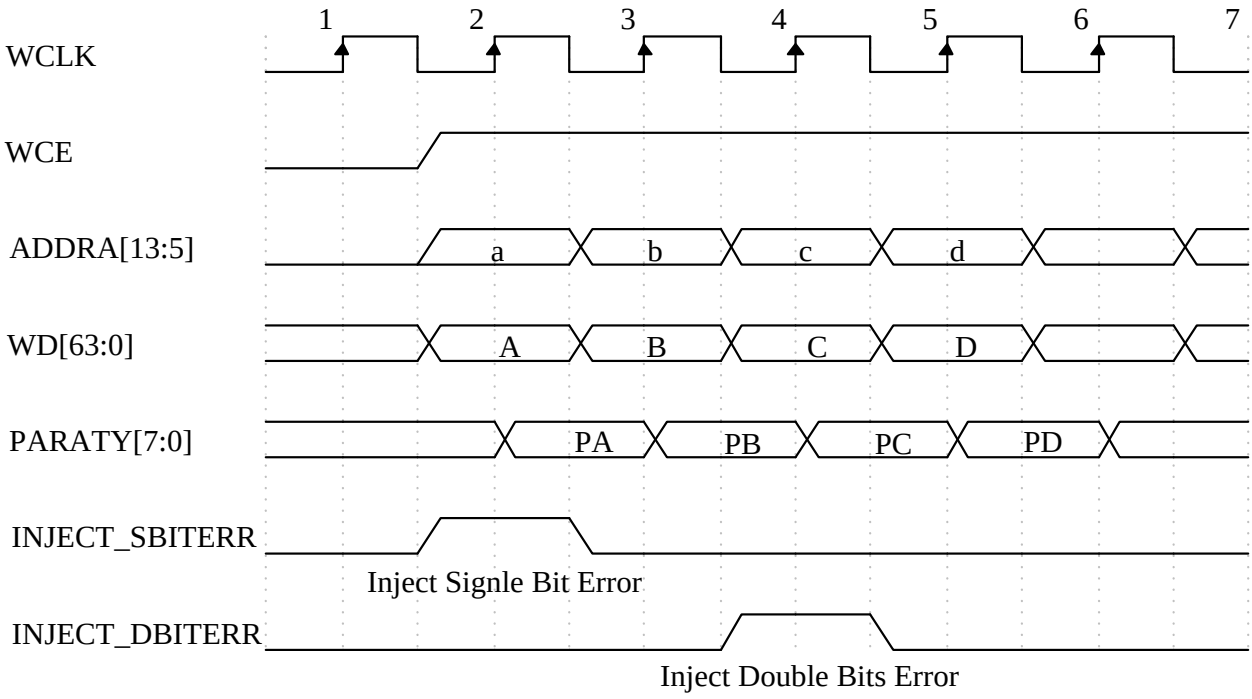

图 8-2 SDP RAM 的 ECC 模式写时序图

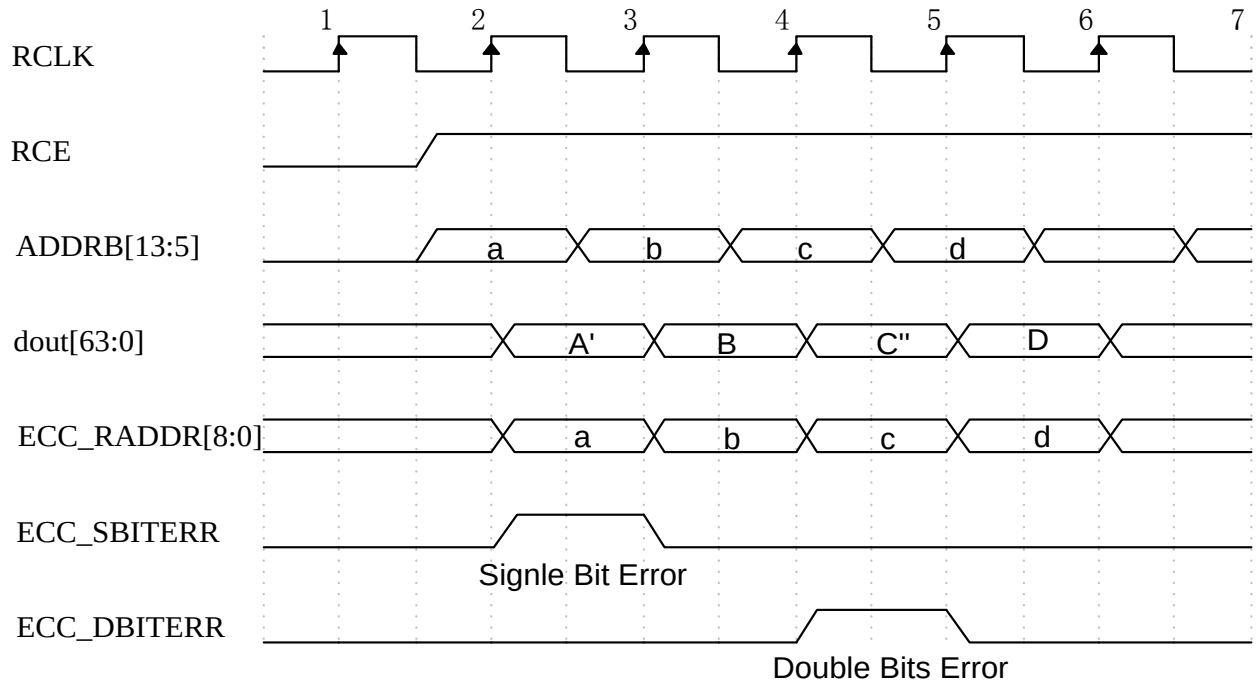

图 8-3 SDP RAM 的 ECC 模式读时序图

8.4 应用示例

8.4.1 SDP RAM 的 ECC 模式配置

本小节举例说明了单个 36K DRM 的 ECC 模式配置步骤，示例端口位宽为 512x72（由 A/B 端口 DOA/DOB 拼接实现），两端口不同时钟，使能输出寄存器。

对单个 36K DRM 按如下步骤进行配置：

a. 对单个 DRM 配置为 SDP RAM 模式 512x72 模式，可参考 4.9.1 单个 36K DRM 配置；

b. 按表 8-1 所述对 DRM 的 ECC 相关参数进行配置：

表 8-1 SDP RAM 的 ECC 模式参数配置

参数名称	配置值	说明
ECC_WRITE_EN	"TRUE"	使能 ECC 读写功能
ECC_READ_EN		

c. 按表 8-2 所示对 DRM 的端口进行连接：

表 8-2 SDP RAM 的 ECC 模式端口连接

端口名称	连接信号	说明
DIA[35:0]	{1'b0, di[31:24], 1'b0, di[23:16], 1'b0, di[15:8], 1'b0, di[7:0]}	A/B 端口拼接实现 72bit 写数据端口，DIA 输入低 32 位 ECC 模式有效写数据 di[31:0]，DIA[8]、DIA[17]、DIA[26]、DIA[35]（字节附加信息位，详见 9.4 字节附加信息位）为 ECC 校验位存储，悬空或接低电平

端口名称	连接信号	说明
DIB[35:0]	{1'b0, di[63:56], 1'b0, di[55:48], 1'b0, di[47:40], 1'b0, di[39:32]}	A/B 端口拼接实现 72bit 写数据端口，DIB 输出高 32 位 ECC 模式有效写数据 di[63:32]，DIB[8]、DIB[17]、DIB[26]、DIB[35]为 ECC 校验位存储，悬空或接低电平
DOA[35:0]	{1'bz, do[31:24], 1'bz, do[23:16], 1'bz, do[15:8], 1'bz, do[7:0]}	A/B 端口拼接实现 72bit 读数据端口，DOA 输出低 32 位 ECC 模式有效读数据 do[31:0]，DOA[8]、DOA[17]、DOA[26]、DOA[35]为 ECC 校验位存储，悬空
DOB[35:0]	{1'bz, do[63:56], 1'bz, do[55:48], 1'bz, do[47:40], 1'bz, do[39:32]}	A/B 端口拼接实现 72bit 读数据端口，DOB 输出高 32 位 ECC 模式有效读数据 do[63:32]，DOB[8]、DOB[17]、DOB[26]、DOB[35]为 ECC 校验位存储，悬空

d. 读写端口其余信号：将读写端口的时钟、时钟使能、输出寄存器时钟使能、数据寄存器复位、ECC 相关信号分别连接到相应的 GTP 相应 A/B 端口上；

e. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

f. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_DRM36K_E1 #(
.DATA_WIDTH_A    (72),
.DATA_WIDTH_B    (72),
.WRITE_MODE_A    ("NORMAL_WRITE"),
.WRITE_MODE_B    ("NORMAL_WRITE"),
.DOA_REG         (1),
.DOB_REG         (1),
.RAM_MODE        ("SIMPLE_DUAL_PORT"),
.ECC_READ_EN     ("TRUE"),
.ECC_WRITE_EN    ("TRUE")
) GTP_DRM36K_E1_inst (
.DOA             (doa           ), // OUTPUT[35:0]
.DOB             (dob           ), // OUTPUT[35:0]
.ADDRA           ({1'b1,addr[8:0],6'b0}), // INPUT[15:0]
.ADDRB           ({1'b1,addr[8:0],6'b0}), // INPUT[15:0]
.BWEA            (8'hff         ), // INPUT[7:0]
.BWEB            (4'b0          ), // INPUT[3:0]
.CSA             (3'b0          ), // INPUT[2:0]
.CSB             (3'b0          ), // INPUT[2:0]
.DIA             ({1'b0,di[31:24],1'b0,di[23:16],1'b0,di[15:8],1'b0,di[7:0]}), // INPUT[35:0]
.DIB             ({1'b0,di[63:56],1'b0,di[55:48],1'b0,di[47:40],1'b0,di[39:32]}), // INPUT[35:0]
.ADDRA_HOLD      (1'b0          ), // INPUT
.ADDRB_HOLD      (1'b0          ), // INPUT
.CEA             (ce            ), // INPUT
.CEB             (ce            ), // INPUT
.CLKA            (wr_clk        ), // INPUT
.CLKB            (rd_clk        ), // INPUT
.ORCEA           (orce          ), // INPUT
.ORCEB           (orce          ), // INPUT
.RSTA            (wr_rst        ), // INPUT
.RSTB            (rd_rst        ), // INPUT
```

```
.WEA              (we             ), // INPUT
.WEB              (1'b0           ), // INPUT
.ECC_DBITERR      (ecc_dbiterr    ), // OUTPUT
.ECC_SBITERR      (ecc_sbiterr    ), // OUTPUT
.INJECT_DBITERR   (inject_dbiterr), // INPUT
.INJECT_SBITERR   (inject_sbiterr)  // INPUT
);
assign do[63:0] = {dob[34:27],dob[25:18],dob[16:9],dob[7:0],doa[34:27],doa[25:18],doa[16:9],
doa[7:0]};
```

8.4.2 FIFO 的 ECC 模式配置

本小节举例说明了单个 36K FIFO 的 ECC 模式配置步骤，示例端口位宽为为 512x72，两端口不同时钟，使能输出寄存器。

对单个 36K FIFO 按如下步骤进行配置：

a. 配置单个 FIFO，可参考 7.6 应用示例；

b. 按表 8-3 所述对 FIFO 的 ECC 相关参数进行配置：

表 8-3 FIFO 的 ECC 模式参数配置

参数名称	配置值	说明
ECC_WRITE_EN	"TRUE"	使能 ECC 读写功能
ECC_READ_EN		

c. 按表 8-4 所示对 FIFO 的端口进行连接：

表 8-4 FIFO 的 ECC 模式端口连接

端口名称	连接信号	说明
DI[72:0]	{1'b0, di[63:56], 1'b0, di[55:48], 1'b0, di[47:40], 1'b0, di[39:32], 1'b0, di[31:24], 1'b0, di[23:16], 1'b0, di[15:8], 1'b0, di[7:0]}	DI 输入 64 位 ECC 模式有效写数据 di[63:0]，DI[8]、DI[17]、DI[26]、DI[35]、DI[44]、DI[53]、DI[62]、DI[71]（字节附加信息位，详见 9.4 字节附加信息位）为 ECC 校验位存储，悬空或接低电平
DOB[35:0]	{1'bz, do[63:56], 1'bz, do[55:48], 1'bz, do[47:40], 1'bz, do[39:32], 1'bz, do[31:24], 1'bz, do[23:16], 1'bz, do[15:8], 1'bz, do[7:0]}	DO 输出 64 位 ECC 模式有效读数据 do[63:0]，DO[8]、DO[17]、DO[26]、DO[35]、DO[44]、DO[53]、DO[62]、DO[71]为 ECC 校验位存储，悬空

d. 读写端口其余信号：将读写端口的时钟、时钟使能、输出寄存器时钟使能、数据寄存器复位、ECC 相关信号分别连接到相应的 GTP 相应读写端口上；

e. 其余不使用的参数：其余不使用的参数不进行设置，使用默认值；

f. 其余不使用的端口：其余不使用的端口悬空（不使用也必须进行连接的端口见上文描述）。

配置后的 GTP 如下所示：

```
GTP_FIFO36K_E1 #(
.DATA_WIDTH       (72),
```

```
.DO_REG          (1),
.ECC_READ_EN     ("TRUE"),
.ECC_WRITE_EN    ("TRUE"),
.USE_EMPTY       (1),
.USE_FULL        (1),
.SYNC_FIFO       ("FALSE")
) GTP_FIFO36K_E1_inst (
.DO              (do_gtp        ), // OUTPUT[71:0]
.DI              ({1'b0, di[63:56], 1'b0, di[55:48], 1'b0, di[47:40], 1'b0, di[39:32], 1'b0,
di[31:24], 1'b0, di[23:16], 1'b0, di[15:8], 1'b0, di[7:0]}), // INPUT[71:0]
.EMPTY           (empty         ), // OUTPUT
.FULL            (full          ), // OUTPUT
.ORCE            (orce          ), // INPUT
.RCE             (rce           ), // INPUT
.RCLK            (rclk          ), // INPUT
.RST             (rst           ), // INPUT
.WCE             (wce           ), // INPUT
.WCLK            (wclk          ), // INPUT
.ECC_DBITERR     (ecc_dbiterr   ), // OUTPUT
.ECC_SBITERR     (ecc_sbiterr   ), // OUTPUT
.INJECT_DBITERR  (inject_dbiterr), // INPUT
.INJECT_SBITERR  (inject_sbiterr)  // INPUT
);
assign do[63:0] = {do_gtp[70:63], do_gtp[61:54], do_gtp[52:45], do_gtp[43:36], do_gtp[34:27],
do_gtp[25:18], do_gtp[16:9], do_gtp[7:0]};
```

9 附录 A

9.1 地址和数据端口 Mapping

表 9-1、表 9-2 为 36K、18K DRM 对应的地址和数据端口 Mapping。Logos2 系列 DRM 具有单个 36Kbits 或者两个 18Kbits 存储空间。

表 9-1 36K DRM 模式下地址和数据端口映射表

DRM 端口模式	地址端口映射		数据端口映射			
	A 端口地址	B 端口地址	A 端口数据输入	B 端口数据输入	A 端口数据输出	B 端口数据输出
32K*1	ADDRA[14:0]	ADDRB[14:0]	DIA[0]	DIB[0]	DOA[X],X=16~9,7~0 中任意一位	DOB[X],X=16~9,7~0 中任意一位
16K*2	ADDRA[14:1]	ADDRB[14:1]	DIA[1:0]	DIB[1:0]	DOA[X+1:X],X=0,2,4,9,11,12,15	DOB[X+1:X],X=0,2,4,9,11,12,15
8K*4	ADDRA[14:2]	ADDRB[14:2]	DIA[3:0]	DIB[3:0]	DOA[X+3:X],X=0,4,9,12	DOB[X+3:X],X=0,4,9,12
4K*8	ADDRA[14:3]	ADDRB[14:3]	DIA[7:0]	DIB[7:0]	DOA[7:0]或者 DOA[16:9]	DOB[7:0]或者 DOB[16:9]
4K*9	ADDRA[14:3]	ADDRB[14:3]	DIA[8:0]	DIB[8:0]	DOA[8:0]或者 DOA[17:9]	DOB[8:0]或者 DOB[17:9]
2K*16	ADDRA[14:4]	ADDRB[14:4]	DIA[16:9,7:0]	DIB[16:9,7:0]	DOA[16:9,7:0]	DOB[16:9,7:0]
2K*18	ADDRA[14:4]	ADDRB[14:4]	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]
1K*32 (DP)	ADDRA[14:5]	ADDRB[14:5]	DIA[34:27,25:18,16:9,7:0]	DIB[34:27,25:18,16:9,7:0]	DOA[34:27,25:18,16:9,7:0]	DOB[34:27,25:18,16:9,7:0]
1K*36 (DP)	ADDRA[14:5]	ADDRB[14:5]	DIA[35:0]	DIB[35:0]	DOA[35:0]	DOB[35:0]
1K*32 (SP/SDP)	ADDRA[14:5]	ADDRB[14:5]	DIA[34:27,25:18,16:9,7:0]	N/A	N/A	DOB[34:27,25:18,16:9,7:0]
1K*36 (SP/SDP)	ADDRA[14:5]	ADDRB[14:5]	DIA[35:0]	N/A	N/A	DOB[35:0]
512*64	ADDRA[14:6]	ADDRB[14:6]	{DIB[34:27,25:18,16:9,7:0],DIA[34:27,25:18,16:9,7:0]}	N/A	N/A	{DOB[34:27,25:18,16:9,7:0],DOA[34:27,25:18,16:9,7:0]}
512*72	ADDRA[14:6]	ADDRB[14:6]	{DIB,DIA}	N/A	N/A	{DOB,DOA}

表 9-2 18K DRM 模式下地址和数据端口映射表

DRM 端口模式	地址端口映射		数据端口映射			
	A 端口地址	B 端口地址	A 端口数据输入	B 端口数据输入	A 端口数据输出	B 端口数据输出
16K*1	ADDRA[13:0]	ADDRB[13:0]	DIA[0]	DIB[0]	DOA[X],X=16~9,7~0 中任意一位	DOB[X],X=16~9,7~0 中任意一位
8K*2	ADDRA[13:1]	ADDRB[13:1]	DIA[1:0]	DIB[1:0]	DOA[X+1:X],X=0,2,4,9,11,12,15	DOB[X+1:X],X=0,2,4,9,11,12,15
4K*4	ADDRA[13:2]	ADDRB[13:2]	DIA[3:0]	DIB[3:0]	DOA[X+3:X],X=0,4,9,12	DOB[X+3:X],X=0,4,9,12

DRM 端口模式	地址端口映射		数据端口映射			
	A 端口地址	B 端口地址	A 端口数据输入	B 端口数据输入	A 端口数据输出	B 端口数据输出
2K*8	ADDRA[13:3]	ADDRB[13:3]	DIA[7:0]	DIB[7:0]	DOA[7:0]或者 DOA[16:9]	DOB[7:0]或者 DOB[16:9]
2K*9	ADDRA[13:3]	ADDRB[13:3]	DIA[8:0]	DIB[8:0]	DOA[8:0]或者 DOA[17:9]	DOB[8:0]或者 DOB[17:9]
1K*16	ADDRA[13:4]	ADDRB[13:4]	DIA[16:9,7:0]	DIB[16:9,7:0]	DOA[16:9,7:0]	DOB[16:9,7:0]
1K*18	ADDRA[13:4]	ADDRB[13:4]	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]
512*32	ADDRA[13:5]	ADDRB[13:5]	DIB[16:9,7:0], DIA[16:9,7:0]}	N/A	N/A	DOB[16:9,7:0], DOB[16:9,7:0]}
512*36	ADDRA[13:5]	ADDRB[13:5]	{DIB[17:0], DIA[17:0]}	N/A	N/A	{ DOB[17:0], DOA[17:0]}

如表 9-3、表 9-4 所示分别为 x1，x2，x4，x8，x16，x32 数据位宽和 x9，x18，x36 数据位宽的数据地址 Mapping。

表 9-3 不同位宽数据地址 Mapping（x1，x2，x4，x8，x16，x32 数据宽度）

数据位宽	最低位端口地址 (与最大数据宽度模式最低位端口地址比较)																											
32	0																											
16	1																0											
8	3				2								1								0							
4	7				6				5				4				3				2				1			
2	15		14		13		12		11		10		9		8		7		6		5		4		3		2	
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4

表 9-4 不同位宽数据地址 Mapping（x9，x18，x36 数据宽度）

数据位宽	最低位端口地址 (与最大数据宽度模式最低位端口地址比较)																															
36	0																															
18	1																	0														
9	3								2									1									0						
Index	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

9.2 初始化配置参数映射

INIT_XX 是 DRM 的初始化配置参数，它们对 DRM 进行了初始化，决定了内存的初始值，默认状态下，DRM 的初始值全为 0。GTP_DRM18K_E1 有包含从 INIT_00 到 INIT_3F 的 64 个初始化配置参数，GTP_DRM36K_E1 有包含从 INIT_00 到 INIT_7F 的 128 个初始化配置参数，每个配置参数是 288 位数，配置了 DRM 对应地址的 288 bits 内存。

下面的公式和表格用于确认每个 INIT_XX 映射的内存 bit 的位置（式中 YY 为十六进制

数 XX 转换后的十进制数）：

最高位 = [(YY+1)*288)]-1

最低位 = (YY)*288

表 9-5 INIT_XX 参数映射

参数	最高位	最低位
INIT_00	287	0
INIT_01	575	288
INIT_02	863	576
……	……	……
INIT_20	9503	9216
……	……	……
INIT_3F	18431	18144
……	……	……
INIT_7F	36575	36288

数据位宽为2^N即 1/2/4/8/16/32/64bit 时，上表中 INIT_XX 每 9 位数据只映射低 8 位到内存；数据位宽为9×2^N即 9/18/36/72bit 时，INIT_XX 数据全部映射到内存。例如，当 DRM 配置为 8x4K 时，INIT_00[7:0]对应地址 addr 为 0 时读出数据的初始值，INIT_00[16:9]对应地址 addr 为 1 时读出数据的初始值；当 DRM 配置为 9x4K 时，INIT_00[8:0]对应地址 addr 为 0 时读出数据的初始值，INIT_00[17:9]对应地址 addr 为 1 时读出数据的初始值。

初始化配置参数 INIT_XX 的应用示例详见 6.7.1 单个 36K DRM 配置。

9.3 DRM 端口信号说明

1.地址总线（ADDRA[15:0]、ADDRB[15:0]）：A/B 端口读写地址，其中，ADDRA[15]、ADDRB[15]为硬级联地址扩展信号，在不使用硬级联模式时，需要接到高电平；剩下的地址总线的有效位数与读写数据位宽有关，例如 DRM36K 配置为 32 位读写数据位宽，则地址位宽为 10 位，有效地址输入为 ADDRA[14:5]、ADDRB[14:5]，剩下的 ADDRA[4:0]和 ADDRB[4:0]则接低电平。DRM18K 不支持硬级联，无硬级联地址扩展信号，DRM18K 的 A/B 端口读写地址为 ADDRA[13:0]、ADDRB[13:0]，例如 DRM18K 配置为 32 位读写数据位宽，则地址位宽为 9 位，有效地址输入为 ADDRA[13:4]、ADDRB[13:4]，剩下的 ADDRA[3:0]和 ADDRB[3:0]则需要接低电平。详细地址映射关系可参考 9.1 地址和数据端口 Mapping。

2.地址保持信号（ADDRA_HOLD、ADDRB_HOLD）：该端口置高电平时对应端口地址输入保持之前的值不变，不随输入地址的改变而改变；该端口置低电平时对应端口地址输入

才为地址总线端口输入的地址。在不使用地址保持功能时需要将该端口接至低电平。

3.数据总线（DIA[35:0]、DIB[35:0]、DOA[35:0]、DOB[35:0]）：A、B 端口的读写数据端口，详细数据端口映射关系可参考附录 9.1 地址和数据端口 Mapping。

4.地址扩展信号（CSA[2:0]、CSB[2:0]）：多个 DRM 可以通过级联扩展的方式组合成更大的 DP RAM，SDP RAM，SP RAM，ROM 或者 FIFO。对此，DRM 的 A、B 端口各提供额外的 3 bit 地址扩展，常用于深度扩展的应用，详见 3.9.3 多个 36K DRM 硬级联配置。

5.写使能信号（WEA、WEB）：DRM 的写操作控制信号，具体写操作时序可参考 3.5 读写操作。

6.字节使能信号（BWEA[7:0]、BWEB[3:0]）：DRM 的字节使能写操作控制信号，每一位控制一个字节的数据在写使能拉高时是否写入对应地址，特别的，SDP 模式下，字节使能写操作只由 BWEA[7:0]控制。在不使用字节使能模式时应将字节使能信号接到高电平。

7.时钟相关信号（CLKA、CLKB、CEA、CEB）：DRM 的 A、B 端口时钟及其使能信号，SDP 模式下，CLKA、CEA 为写时钟及其使能信号，CLKB、CEB 为读时钟及其使能信号。

8.输出寄存器相关信号（ORCEA、ORCEB、RSTA、RSTB）：A、B 端口输出寄存器使能信号及复位信号。

9.级联信号（CINA、CINB、COUTA、COUTB）：64Kx1 模式下做深度级联用，相邻 DRM 的 A、B 端口数据输出级联输入/输出。

10.ECC 模式相关信号（INJECT_SBITERR、INJECT_DBITERR、ECC_SBITERR、ECC_DBITERR、ECC_PARITY[7:0]、ECC_RDADDR[8:0]）：上述信号分别为单 bit/双 bits 错误注入信号，单 bit/双 bits 错误标志信号，ECC 编码校验位输出，ECC 解码读地址输出。

9.4 字节附加信息位

在 18K 的数据宽度为 x9、x18 模式以及 36K 的数据宽度为 x9、x18、x36、x72 模式下，对应每字节有一位附加信息存储位，如表 9-6 所示：

表 9-6 字节附加信息位列表

端口	模式	数据输入		数据输入	
		字节	附加信息位	字节	附加信息位
A	18K、36K	DIA[7:0]	DIA[8]	dia[7:0]	dia[8]
	18K、36K	DIA[16:9]	DIA[17]	dia[16:9]	dia[17]
	36K	DIA[25:18]	DIA[26]	dia[25:18]	dia[26]
	36K	DIA[34:27]	DIA[35]	dia[34:27]	dia[35]
B	18K、36K	DIB[7:0]	DIB[8]	dib[7:0]	dib[8]

端口	模式	数据输入		数据输入	
		字节	附加信息位	字节	附加信息位
	18K、36K	DIB[16:9]	DIB[17]	dib[16:9]	dib[17]
	36K	DIB[25:18]	DIB[26]	dib[25:18]	dib[26]
	36K	DIB[34:27]	DIB[35]	dib[34:27]	dib[35]

上述附加信息位在 x1、x2、x4、x8、x16、x32、x64 模式下需要悬空或接低电平。

9.5 GTP_DRM36K_E1 例化模板

```
GTP_DRM36K_E1 #(
.GRS_EN("TRUE"),
.CSA_MASK(3'b000),
.CSB_MASK(3'b000),
.DATA_WIDTH_A(18),
.DATA_WIDTH_B(18),
.WRITE_MODE_A("NORMAL_WRITE"),
.WRITE_MODE_B("NORMAL_WRITE"),
.DOA_REG(0),
.DOB_REG(0),
.DOA_REG_CLKINV(0),
.DOB_REG_CLKINV(0),
.RSTA_VAL(36'b0),
.RSTB_VAL(36'b0),
.RST_TYPE("SYNC"),
.RAM_MODE("TRUE_DUAL_PORT"),
.RAM_CASCADE("NONE"),
.ECC_READ_EN("FALSE"),
.ECC_WRITE_EN("FALSE"),
.INIT_00(288'b0),
.INIT_01(288'b0),
.INIT_02(288'b0),
.INIT_03(288'b0),
.INIT_04(288'b0),
.INIT_05(288'b0),
.INIT_06(288'b0),
.INIT_07(288'b0),
.INIT_08(288'b0),
.INIT_09(288'b0),
.INIT_0A(288'b0),
.INIT_0B(288'b0),
.INIT_0C(288'b0),
.INIT_0D(288'b0),
.INIT_0E(288'b0),
.INIT_0F(288'b0),
.INIT_10(288'b0),
.INIT_11(288'b0),
.INIT_12(288'b0),
.INIT_13(288'b0),
.INIT_14(288'b0),
.INIT_15(288'b0),
```

```
.INIT_16(288'b0),
.INIT_17(288'b0),
.INIT_18(288'b0),
.INIT_19(288'b0),
.INIT_1A(288'b0),
.INIT_1B(288'b0),
.INIT_1C(288'b0),
.INIT_1D(288'b0),
.INIT_1E(288'b0),
.INIT_1F(288'b0),
.INIT_20(288'b0),
.INIT_21(288'b0),
.INIT_22(288'b0),
.INIT_23(288'b0),
.INIT_24(288'b0),
.INIT_25(288'b0),
.INIT_26(288'b0),
.INIT_27(288'b0),
.INIT_28(288'b0),
.INIT_29(288'b0),
.INIT_2A(288'b0),
.INIT_2B(288'b0),
.INIT_2C(288'b0),
.INIT_2D(288'b0),
.INIT_2E(288'b0),
.INIT_2F(288'b0),
.INIT_30(288'b0),
.INIT_31(288'b0),
.INIT_32(288'b0),
.INIT_33(288'b0),
.INIT_34(288'b0),
.INIT_35(288'b0),
.INIT_36(288'b0),
.INIT_37(288'b0),
.INIT_38(288'b0),
.INIT_39(288'b0),
.INIT_3A(288'b0),
.INIT_3B(288'b0),
.INIT_3C(288'b0),
.INIT_3D(288'b0),
.INIT_3E(288'b0),
.INIT_3F(288'b0),
.INIT_40(288'b0),
.INIT_41(288'b0),
.INIT_42(288'b0),
.INIT_43(288'b0),
.INIT_44(288'b0),
.INIT_45(288'b0),
.INIT_46(288'b0),
.INIT_47(288'b0),
.INIT_48(288'b0),
.INIT_49(288'b0),
.INIT_4A(288'b0),
.INIT_4B(288'b0),
```

```
.INIT_4C(288'b0),
.INIT_4D(288'b0),
.INIT_4E(288'b0),
.INIT_4F(288'b0),
.INIT_50(288'b0),
.INIT_51(288'b0),
.INIT_52(288'b0),
.INIT_53(288'b0),
.INIT_54(288'b0),
.INIT_55(288'b0),
.INIT_56(288'b0),
.INIT_57(288'b0),
.INIT_58(288'b0),
.INIT_59(288'b0),
.INIT_5A(288'b0),
.INIT_5B(288'b0),
.INIT_5C(288'b0),
.INIT_5D(288'b0),
.INIT_5E(288'b0),
.INIT_5F(288'b0),
.INIT_60(288'b0),
.INIT_61(288'b0),
.INIT_62(288'b0),
.INIT_63(288'b0),
.INIT_64(288'b0),
.INIT_65(288'b0),
.INIT_66(288'b0),
.INIT_67(288'b0),
.INIT_68(288'b0),
.INIT_69(288'b0),
.INIT_6A(288'b0),
.INIT_6B(288'b0),
.INIT_6C(288'b0),
.INIT_6D(288'b0),
.INIT_6E(288'b0),
.INIT_6F(288'b0),
.INIT_70(288'b0),
.INIT_71(288'b0),
.INIT_72(288'b0),
.INIT_73(288'b0),
.INIT_74(288'b0),
.INIT_75(288'b0),
.INIT_76(288'b0),
.INIT_77(288'b0),
.INIT_78(288'b0),
.INIT_79(288'b0),
.INIT_7A(288'b0),
.INIT_7B(288'b0),
.INIT_7C(288'b0),
.INIT_7D(288'b0),
.INIT_7E(288'b0),
.INIT_7F(288'b0),
.INIT_FILE("NONE"),
.BLOCK_X(0),
```

```
.BLOCK_Y(0),
.RAM_DATA_WIDTH(9),
.RAM_ADDR_WIDTH(12),
.INIT_FORMAT("BIN")
) GTP_DRM36K_E1_inst (
.DOA             (doa           ),
.DOB             (dob           ),
.ECC_PARITY      (ecc_parity    ),
.ECC_RDADDR      (ecc_rdaddr    ),
.ADDRA           (addra         ),
.ADDRB           (addrb         ),
.BWEA            (bwea          ),
.BWEB            (bweb          ),
.CSA             (csa           ),
.CSB             (csb           ),
.DIA             (dia           ),
.DIB             (dib           ),
.COUTA           (couta         ),
.COUTB           (coutb         ),
.ECC_DBITERR     (ecc_dbiterr   ),
.ECC_SBITERR     (ecc_sbiterr   ),
.ADDRA_HOLD      (addra_hold    ),
.ADDRB_HOLD      (addrb_hold    ),
.CEA             (cea           ),
.CEB             (ceb           ),
.CINA            (cina          ),
.CINB            (cinb          ),
.CLKA            (clka          ),
.CLKB            (clkb          ),
.INJECT_DBITERR (inject_dbiterr),
.INJECT_SBITERR (inject_sbiterr),
.ORCEA           (orcea         ),
.ORCEB           (orceb         ),
.RSTA            (rsta          ),
.RSTB            (rstb          ),
.WEA             (wea           ),
.WEB             (web           )
);
```

9.6 GTP_DRM18K_E1 例化模板

```
GTP_DRM18K_E1 #(
.GRS_EN("TRUE"),
.DATA_WIDTH_A(18),
.DATA_WIDTH_B(18),
.DOA_REG(0),
.DOB_REG(0),
.DOA_REG_CLKINV(0),
.DOB_REG_CLKINV(0),
.RSTA_VAL(18'b0),
.RSTB_VAL(18'b0),
.RST_TYPE("SYNC"),
.RAM_MODE("TRUE_DUAL_PORT"),
.WRITE_MODE_A("NORMAL_WRITE"),
```

```
.WRITE_MODE_B("NORMAL_WRITE"),
.INIT_00(288'b0),
.INIT_01(288'b0),
.INIT_02(288'b0),
.INIT_03(288'b0),
.INIT_04(288'b0),
.INIT_05(288'b0),
.INIT_06(288'b0),
.INIT_07(288'b0),
.INIT_08(288'b0),
.INIT_09(288'b0),
.INIT_0A(288'b0),
.INIT_0B(288'b0),
.INIT_0C(288'b0),
.INIT_0D(288'b0),
.INIT_0E(288'b0),
.INIT_0F(288'b0),
.INIT_10(288'b0),
.INIT_11(288'b0),
.INIT_12(288'b0),
.INIT_13(288'b0),
.INIT_14(288'b0),
.INIT_15(288'b0),
.INIT_16(288'b0),
.INIT_17(288'b0),
.INIT_18(288'b0),
.INIT_19(288'b0),
.INIT_1A(288'b0),
.INIT_1B(288'b0),
.INIT_1C(288'b0),
.INIT_1D(288'b0),
.INIT_1E(288'b0),
.INIT_1F(288'b0),
.INIT_20(288'b0),
.INIT_21(288'b0),
.INIT_22(288'b0),
.INIT_23(288'b0),
.INIT_24(288'b0),
.INIT_25(288'b0),
.INIT_26(288'b0),
.INIT_27(288'b0),
.INIT_28(288'b0),
.INIT_29(288'b0),
.INIT_2A(288'b0),
.INIT_2B(288'b0),
.INIT_2C(288'b0),
.INIT_2D(288'b0),
.INIT_2E(288'b0),
.INIT_2F(288'b0),
.INIT_30(288'b0),
.INIT_31(288'b0),
.INIT_32(288'b0),
.INIT_33(288'b0),
.INIT_34(288'b0),
```

```
.INIT_35(288'b0),
.INIT_36(288'b0),
.INIT_37(288'b0),
.INIT_38(288'b0),
.INIT_39(288'b0),
.INIT_3A(288'b0),
.INIT_3B(288'b0),
.INIT_3C(288'b0),
.INIT_3D(288'b0),
.INIT_3E(288'b0),
.INIT_3F(288'b0),
.INIT_FILE("NONE"),
.BLOCK_X(0),
.BLOCK_Y(0),
.RAM_DATA_WIDTH(9),
.RAM_ADDR_WIDTH(11),
.INIT_FORMAT("BIN")
) GTP_DRM18K_E1_inst (
.DOA         (doa       ),
.DOB         (dob       ),
.ADDRA       (addra     ),
.ADDRB       (addrb     ),
.BWEA        (bwea      ),
.BWEB        (bweb      ),
.DIA         (dia       ),
.DIB         (dib       ),
.ADDRA_HOLD (addra_hold),
.ADDRB_HOLD (addrb_hold),
.CEA         (cea       ),
.CEB         (ceb       ),
.CLKA        (clka      ),
.CLKB        (clkb      ),
.ORCEA       (orcea     ),
.ORCEB       (orceb     ),
.RSTA        (rsta      ),
.RSTB        (rstb      ),
.WEA         (wea       ),
.WEB         (web       )
);
```

9.7 GTP_FIFO36K_E1 例化模板

```
GTP_FIFO36K_E1 #(
.GRS_EN("TRUE"),
.DATA_WIDTH(18),
.DO_REG(0),
.ECC_READ_EN("FALSE"),
.ECC_WRITE_EN("FALSE"),
.ALMOST_FULL_OFFSET('b0),
.ALMOST_EMPTY_OFFSET('b0),
.RST_VAL('b0),
.USE_EMPTY(0),
.USE_FULL(0),
.SYNC_FIFO("FALSE")
```

```
) GTP_FIFO36K_E1_inst (
.DO              (do            ),
.DI              (di            ),
.ALMOST_EMPTY    (almost_empty  ),
.ALMOST_FULL     (almost_full   ),
.ECC_DBITERR     (ecc_dbiterr   ),
.ECC_SBITERR     (ecc_sbiterr   ),
.EMPTY           (empty         ),
.FULL            (full          ),
.INJECT_DBITERR (inject_dbiterr),
.INJECT_SBITERR (inject_sbiterr),
.ORCE            (orce          ),
.RCE             (rce           ),
.RCLK            (rclk          ),
.RST             (rst           ),
.WCE             (wce           ),
.WCLK            (wclk          )
);
```

9.8 GTP_FIFO18K_E1 例化模板

```
GTP_FIFO18K_E1 #(
.GRS_EN("TRUE"),
.DATA_WIDTH(18),
.DO_REG(0),
.ALMOST_FULL_OFFSET('b0),
.ALMOST_EMPTY_OFFSET('b0),
.RST_VAL('b0),
.USE_EMPTY(0),
.USE_FULL(0),
.SYNC_FIFO("FALSE")
) GTP_FIFO18K_E1_inst (
.DO              (do          ),
.DI              (di          ),
.ALMOST_EMPTY    (almost_empty),
.ALMOST_FULL     (almost_full ),
.EMPTY           (empty       ),
.FULL            (full        ),
.ORCE            (orce        ),
.RCE             (rce         ),
.RCLK            (rclk        ),
.RST             (rst         ),
.WCE             (wce         ),
.WCLK            (wclk        )
);
```

免责声明

版权声明

免责声明

1、本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因本文档使用不当造成的直接或间接损失，本公司不承担任何法律责任。

2、本文档按现状提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

3、公司保留任何时候在不事先声明的情况下对公司系列产品相关文档的修改权利。