使用寄存器级读&写控制基于 PXI 平台的 FPGA

摘要：本文以基于 PXI 平台的 Marvin Test Solutions 3U FPGA 板卡 GX3500 为设计对象，通过设计实例讲解如何使用寄存器级读&写控制 FPGA，并提供 FPGA 设计相关文件(如 SVF)和软件控制例程。

简介

为了更好地理解如何访问 GX3500 FPGA 的寄存器，需要有一个使用寄存器的设计。本篇文章分为两部分内容：第一部分，以读者已经熟练使用 Altera Quartus II 设计工具为前提（参考《GX3500 User's Guide: GXFPGA Tutorial and Examples》的第五章）,概述了如何使用 GX3500 设计 128 通道的静态 I/O。此设计实例配置为 4 组 32 通道双向引脚，双缓冲结构支持同步更新 128 通道逻辑状态的读和写。

本篇文章的第二部分讲述了如何向 GX3500 FPGA 内加载设计文件，如何连接 GX3500 I/O 引脚，为了实现 FPGA 静态数字 I/O 的操作如何对寄存器进行读和写。

图 1 MTS-GX3500 FPGA 板卡

设计实例概览

地址译码(Address Decoding)

GX3500 支持对两种类型的 PCI 总线读和写操作：一类是针对寄存器，使用 PCI BAR 1；另一类是针对 RAM ,使用 PCI BAR 2。静态数字 I/O 设计实例使用寄存器控制对 I/O 引脚的读和写，所以使用 PCI BAR 1 片选信号进行地址译码 — 与 Chip Select 1 (CS[1])同意。BAR 1 信号可以访问的地址范围为 1024 byte（0x400），访问时必须以 4-byte 为准对齐。 图 2 为地址译码

逻辑单元，将 5 路地址信号（Addr[6..2]）译码后，可提供 32 路“写使能”信号（WE[31..0]）和 32 路“读使能”信号(RE[31..0]）,这些信号用于控制锁存寄存器的向 I/O 引脚写入(WE[x]) ）和从 I/O 引脚读取（RE[x]）功能。

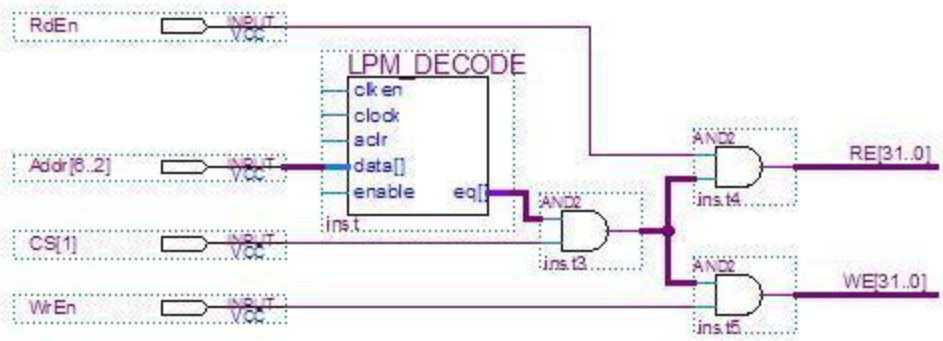

图 2：寄存器地址译码

引脚控制

本设计实例中有四组完全相同的 I/O 引脚（见图 4），每组有 32 路通道。每路通道都支持双向传输，并可以独立配置传输方向。输出寄存器为双缓冲结构，支持四组 I/O 引脚（128 路通道）同步更新。第一阶段，通过 WE_Data 控制信号，将输出数据使用写入第一级数据寄存器，通过 WE_Tristate 写入三态控制信号。以上这些信号来自于 WE[31..0]信号，并且这些信号在每组 I/O 引脚间独立。第二阶段，通过 WE_UpdatePort 控制信号，将第一级输出的数据和三态控制信号写入第二级寄存器。以上这些信号也来自于 WE[31..0]信号，但是四组 I/O 引脚共用以实现四组 I/O 引脚数据的同步更新。使用 RE_Tristatelatch，RE_DataLatch ,RE_TristatePort 和 RE_DataPort 控制信号访问输出寄存器两个阶段的数据和三态控制寄存器从而进行读操作。

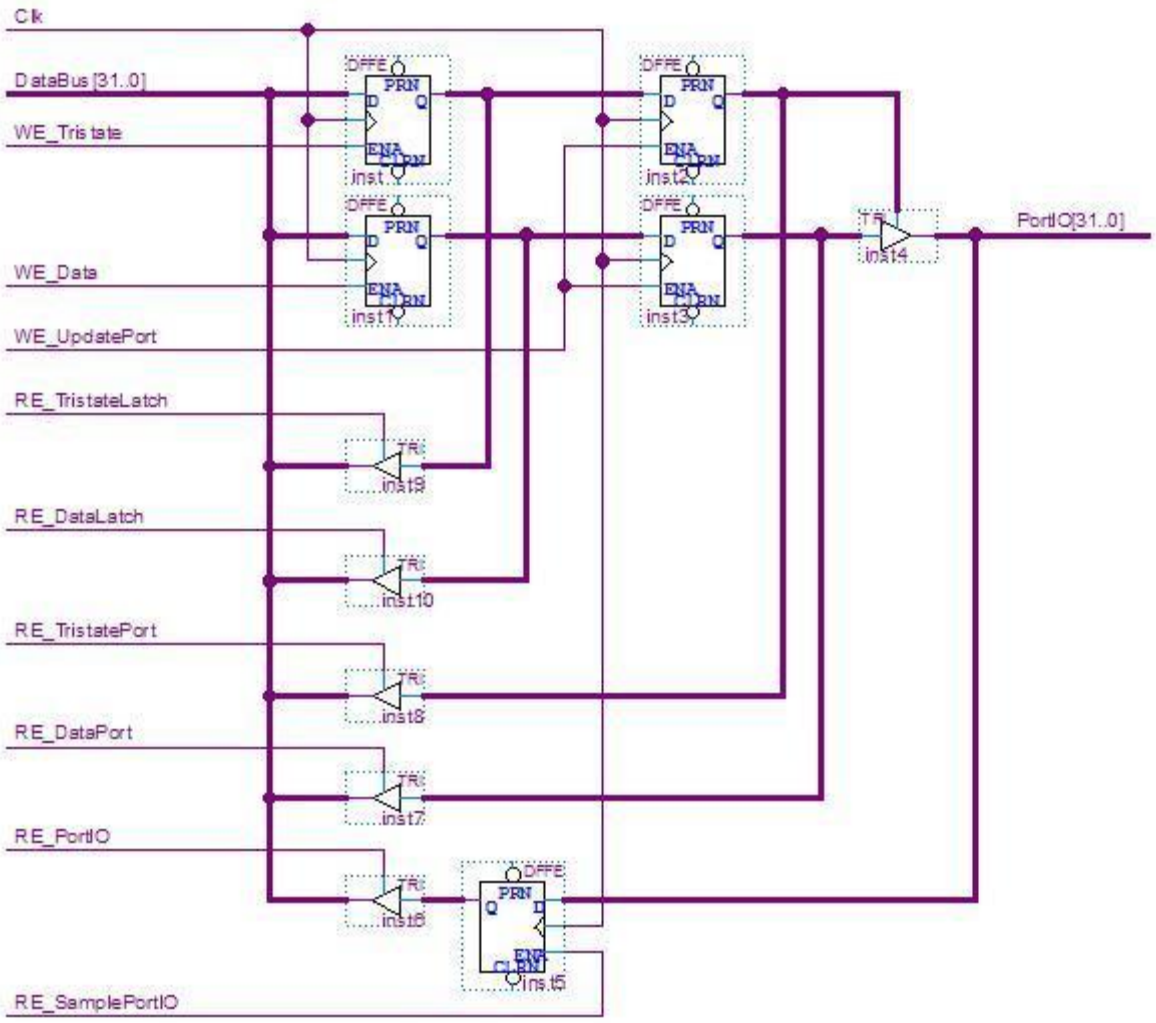

图 3：引脚控制逻辑单元（图 4 中的 Port_Control）

通过 RE_SamplePortIO 控制信号可实现四组 I/O 引脚（128 路通道）的所有通道同时被采样，采样数据被储存在锁存寄存器中以用于后续的数据检索。分别通过 RE_PortIO 控制信号实现数据检索。因为每组 I/O 引脚的三态控制信号可以被读，所以可以推断出 GX3500 或 UUT 是否处于采样输入状态（读状态）。

寄存器映象(Register Mapping)

下面是控制四组 I/O 引脚（A 组、B 组、C 组、D 组）的读和写寄存器偏移地址：

写功能：		
偏移量	**(HEX)**	**功能**
0	(0x0)	WE[0]: 向 I/O 引脚（A 组）锁存器中写入数据
4	(0x4)	WE[1]: 向 I/O 引脚（B 组）锁存器中写入数据
8	(0x8)	WE[2]: 向 I/O 引脚（C 组）锁存器中写入数据
12	(0xC)	WE[3]: 向 I/O 引脚（D 组）锁存器中写入数据
16	(0x10)	WE[4]: 向 I/O 引脚（A 组）锁存器中写入三态控制信号
20	(0x14)	WE[5]: 向 I/O 引脚（B 组）锁存器中写入三态控制信号
24	(0x18)	WE[6]: 向 I/O 引脚（C 组）锁存器中写入三态控制信号
28	(0x1C)	WE[7]: 向 I/O 引脚（D 组）锁存器中写入三态控制信号
80	(0x50)	WE[20]: 同步更新 I/O 引脚（A~D 组）
读功能：		
偏移量	**(HEX)**	**功能**
0	(0x0)	WE[0]: 从 I/O 引脚（A 组）锁存器中读取数据
4	(0x4)	WE[1]: 从 I/O 引脚（B 组）锁存器中读取数据
8	(0x8)	WE[2]: 从 I/O 引脚（C 组）锁存器中读取数据
12	(0xC)	WE[3]: 从 I/O 引脚（D 组）锁存器中读取数据
16	(0x10)	WE[4]: 从 I/O 引脚（A 组）锁存器中读取三态控制信号
20	(0x14)	WE[5]: 从 I/O 引脚（B 组）锁存器中读取三态控制信号
24	(0x18)	WE[6]: 从 I/O 引脚（C 组）锁存器中读取三态控制信号
28	(0x1C)	WE[7]: 从 I/O 引脚（D 组）锁存器中读取三态控制信号
32	(0x20)	WE[8]: 读取 I/O 引脚（A 组）的输出数据
36	(0x24)	WE[9]: 读取 I/O 引脚（B 组）的输出数据
40	(0x28)	WE[10]: 读取 I/O 引脚（C 组）的输出数据

44	(0x2C)	WE[11]: 读取 I/O 引脚（D 组）的输出数据
48	(0x30)	WE[12]: 读取 I/O 引脚（A 组）的三态控制信号
52	(0x34)	WE[13]: 读取 I/O 引脚（B 组）的三态控制信号
56	(0x38)	WE[14]: 读取 I/O 引脚（C 组）的三态控制信号
60	(0x3C)	WE[15]: 读取 I/O 引脚（D 组）的三态控制信号
64	(0x40)	WE[16]: 从 I/O 引脚（A 组）输入锁存器中读取采样数据
68	(0x44)	WE[17]: 从 I/O 引脚（B 组）输入锁存器中读取采样数据
72	(0x48)	WE[18]: 从 I/O 引脚（C 组）输入锁存器中读取采样数据
76	(0x4C)	WE[19]: 从 I/O 引脚（D 组）输入锁存器中读取采样数据
80	(0x50)	WE[20]: 对 I/O 引脚（A~D 组）同步采样到输入锁存器

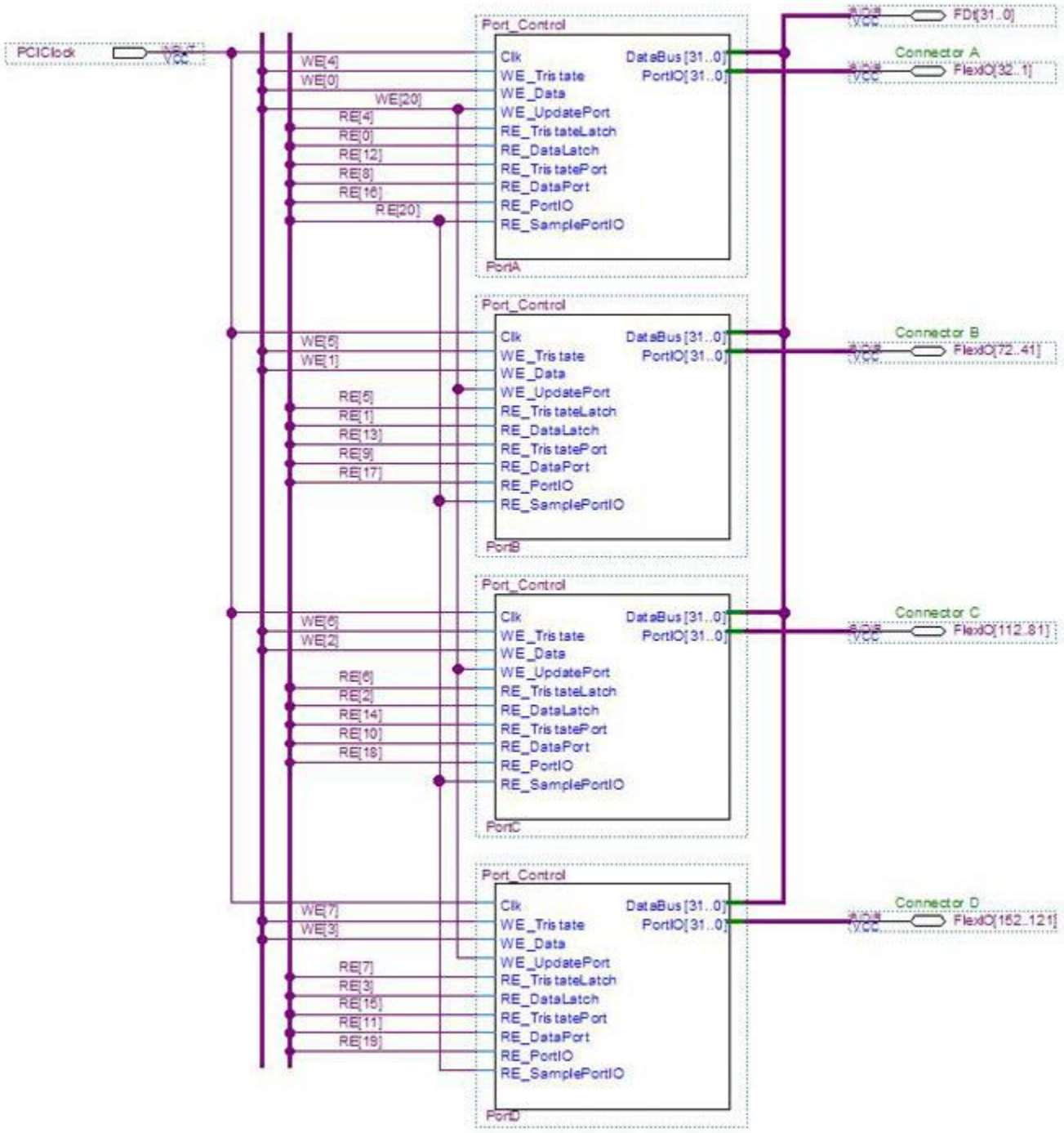

图 4：I/O 引脚（A~D 组）的读和写控制

GX3500 静态 I/O 设计文件和 SVF 文件下载：here.

软件控制

为了控制本文中的 GX3500 设计实例，首先要初始化仪器驱动，然后加载 SVF 文件，然后写和读本设计实例中使用的寄存器地址。GX3500 API，包含用于访问内存的函数、使能或旁路扩展板卡继电器的函数和其他函数。参考《GX3500 User's Guide》获取 API 函数详细列表和调用语法。

本文提供的控制 GX3500 FPGA 静态 I/O 的代码，支持三种不同的编程环境：ATEasy，"C"和 LabView。这些例程假设 GX3500 安装在 PXI 机箱的第 12 槽中、使用 Altera Quartus II 设计软件生成 SVF 文件"Static_IO.svf"、SVF 文件在应用程序可读入的目录下。

ATEasy 例程 (使用 ATEasy GX3500 驱动)

```
dwData:DWord[4] ! Contains output state for 32-bit ports A- D
dwTristate:DWord[4] ! Contains tristate control for 32-bit ports A-D
dwInput:DWord[4] ! Contains data read from four latches A-D
i:Long ! Index counter

Driver Initialize (12) ! Initialize driver for instrument in slot #12
FPGA Load (".\\Static_IO.svf",TARGET_VOLATILE,MODE_SYNC) ! Load SVF file to volatile FPGA memory
FPGA Set ExpansionBoardBypass(0b1111) ! Set the expansion bypass – signals route from the FPGA to the connectors
For i=0 to 3 ! Repeat for port A - D
   FPGA Write Register(i*4,4,dwData[i]) ! Write to Data Latch WE[i]
   FPGA Write Register((i+4)*4,4,dwTristate[i]) ! Write to Tristate Latch WE[i+4]
Next
FPGA Write Register(80,4,0) ! Simultaneous Update all 128 I/O pins (tristate and data)
FPGA Read Register(80,4,dwData) ! Simultaneous Sample all 128 I/O pins
For i=0 to 3 ! Repeat for port A-D
   FPGA Read Register(i*4,4,dwInput[i]) ! Read sampled state from Latch RE[i]
Next
```

“C” 例程

```
int    nHandle, nStatus, i;
DWord    dwData[4], dwTristate[4], dwInput[4];

GxFpgaInitialize (12, nHandle, nStatus);  \\ Initialize driver for instrument in slot #12
GxFpgaLoad (nHandle, 0, "Static_IO.svf" ,0,, pnStatus);  \\ Load SVF file to volatile FPGA memory
GxFpgaSetExpansionBoardBypass (nHandle , 0xF, pnStatus);  \\ Set the expansion bypass
for(i=0;i<4;i++){  \\ Repeat for port A - D
   GxFpgaWriteRegister (nHandle ,i*4, dwData[i], 4, nStatus);  \\ Write to Data Latch WE[i]
   GxFpgaWriteRegister (nHandle ,(i+4)*4, dwTristate[i], 4, nStatus);  \\ Write to Tristate Latch WE[i+4]
}
GxFpgaWriteRegister (nHandle ,80, 0, 4, nStatus);  \\ Simultaneous Update all 128 I/O pins (tristate and data)
GxFpgaReadRegister (nHandle, 80, dwInput[0], 4, nStatus);  \\ Simultaneous Sample all 128 I/O pins

for(i=0;i<4;i++){  \\ Repeat for port A - D
   GxFpgaReadRegister (nHandle, i*4, dwInput[i], 4, nStatus);  \\ Read sampled state from Latch RE[i]
}
```

LabView 例程

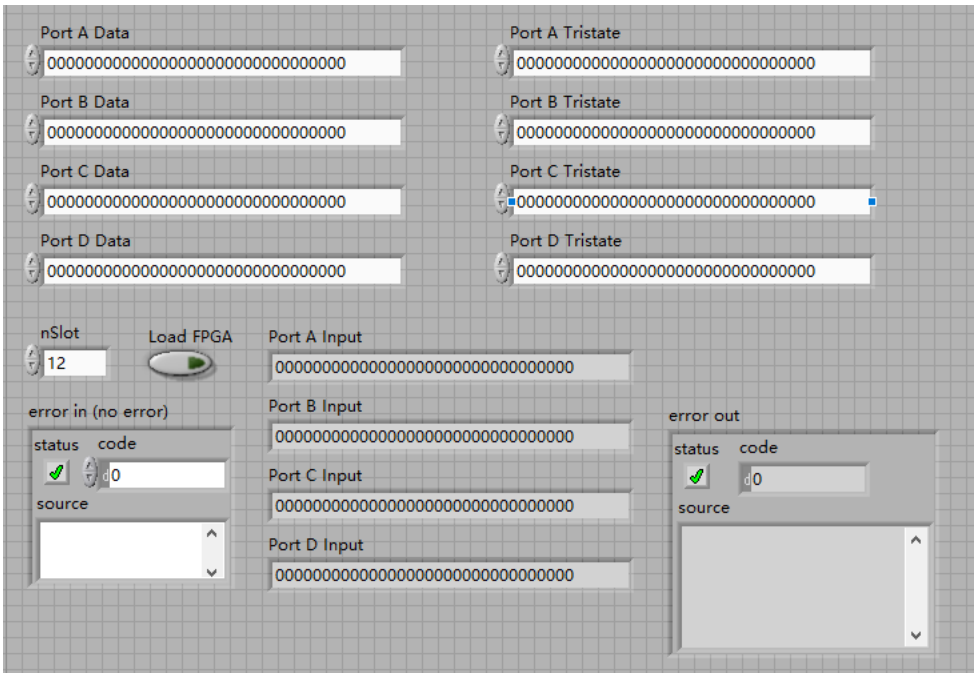

图 5： LabView 例程用户控制界面

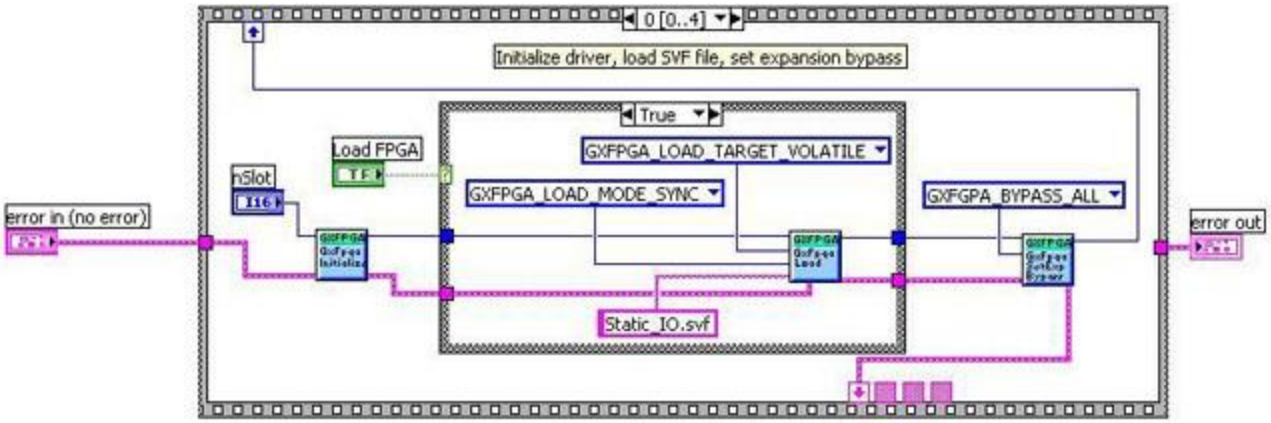

图 6： LabView 例程中层叠顺序结构的帧 0

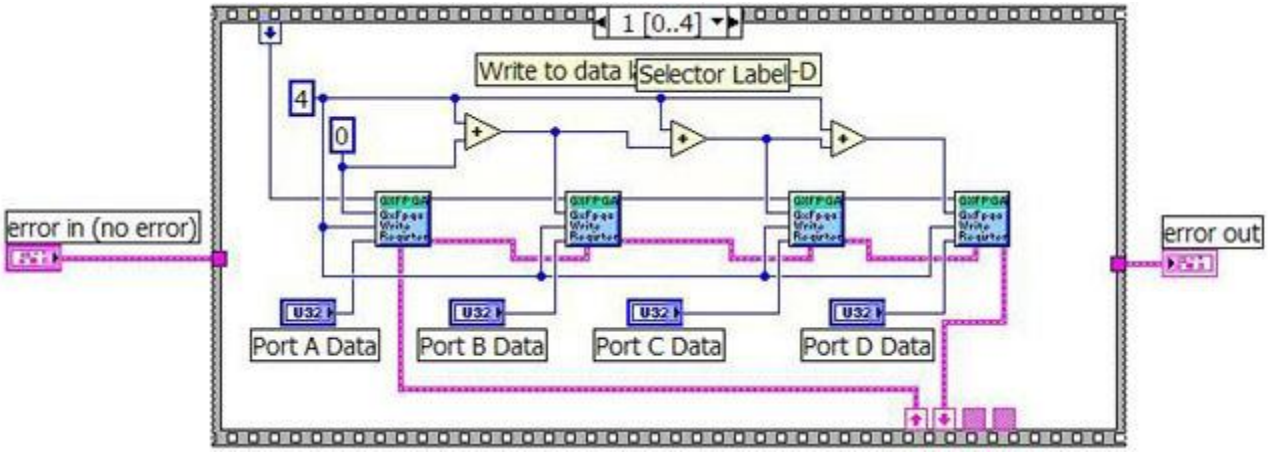

图 7： LabView 例程中层叠顺序结构的帧 1

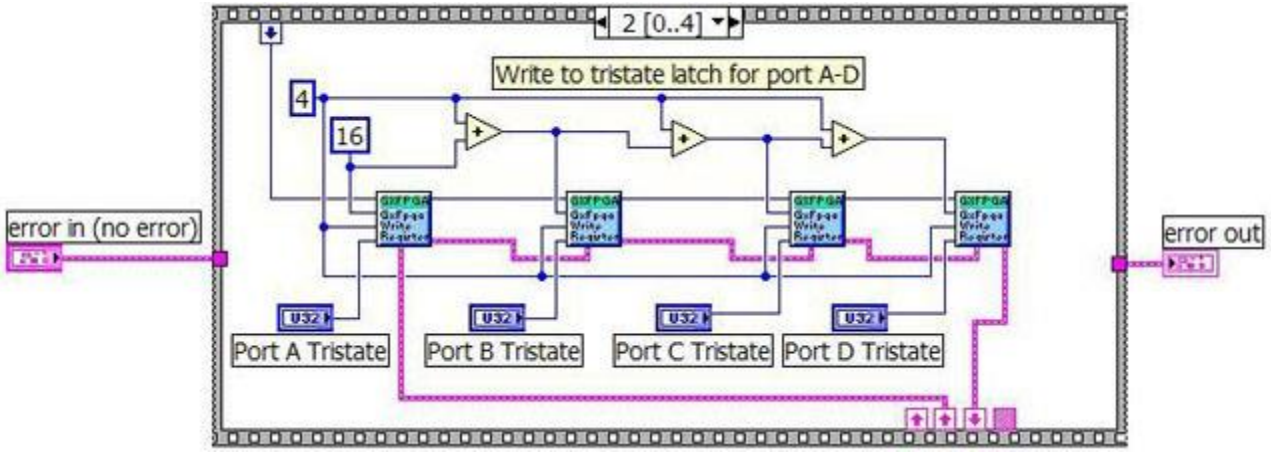

图 8： LabView 例程中层叠顺序结构的帧 2

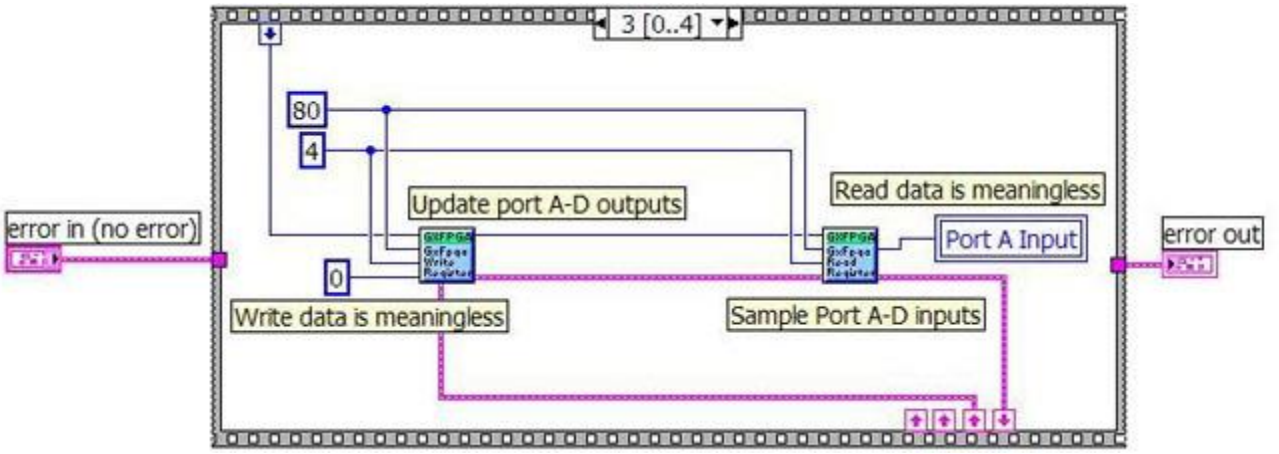

图 9： LabView 例程中层叠顺序结构的帧 3

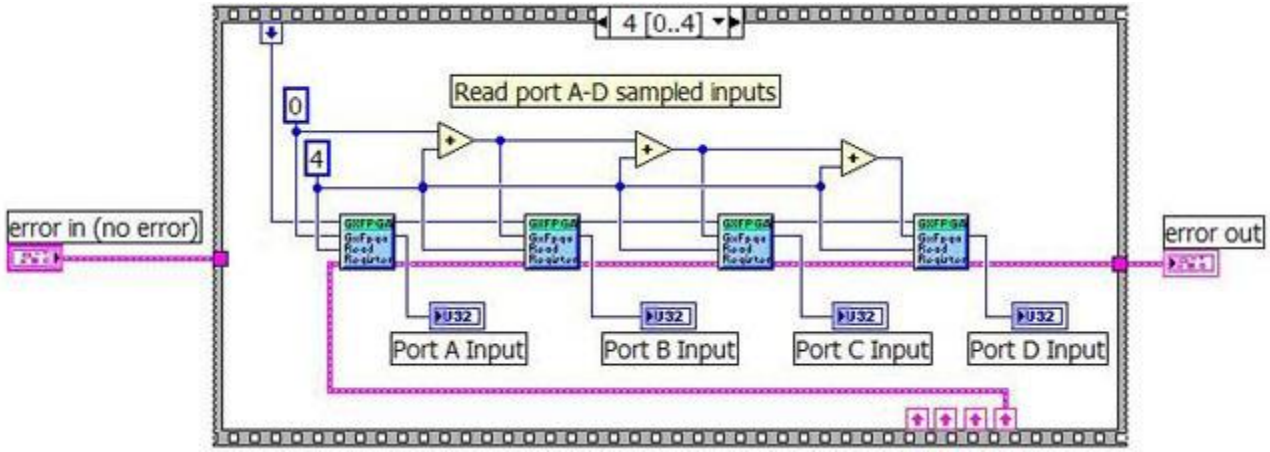

图 10： LabView 例程中层叠顺序结构的帧 4

关键字：GX3500，FPGA，Altera，Quartus，FPGA Design tools，Static I/O，PCI，PXI