工程文档编号	
版本号	V1.0
作者	樊 荣
日期	
项目编号	WLCSY-0001
表格号	
模板版本	

10G 网络测试仪数据通道寄存器说明

修订记录

版本	日期	作者	备注
V1.0	2013.10.08		

目录

1 概述说明

1.1 目的

为分便项目组成员间的交流、设计维护与调试，对网络测试仪数据通道的工作原理和跟软件相关的寄存器、配置方法等进行说明。

1.2 设计依据

系统依据以太网测试的相关标准（具体参考 RFC 相关文档）进行设计，本版依据以太网(二层)测试标准 RFC2544 设计。

1.3 参考资料

- 支持远程可重配置的网络测试仪器研究与实现（BitTester）. 硕士论文，戴硕，2012.6.
- 网络性能测试与分析.林川，施晓秋，胡波，高等教育出版社，2011.8.
- RFC2544 标准 以太网(二层)测试方案 V1.0.彭鹏.
- 10G 网络测试仪数据通道概要设计 V1.0. 樊荣.
- 10G 网络测试仪方案书 V1.0. 樊荣.
- A Packet Generator on the NetFPGA Platform. G. Adam Covington, Glen Gibb, John W. Lockwood, Nick McKeown.
- An Open-Source Hardware Module for High-Speed Network Monitoring on NetFPGA. Gianni Antichi, David J. Miller, Stefano Giordano.

1.4 术语和缩写词

2 数据通道工作原理

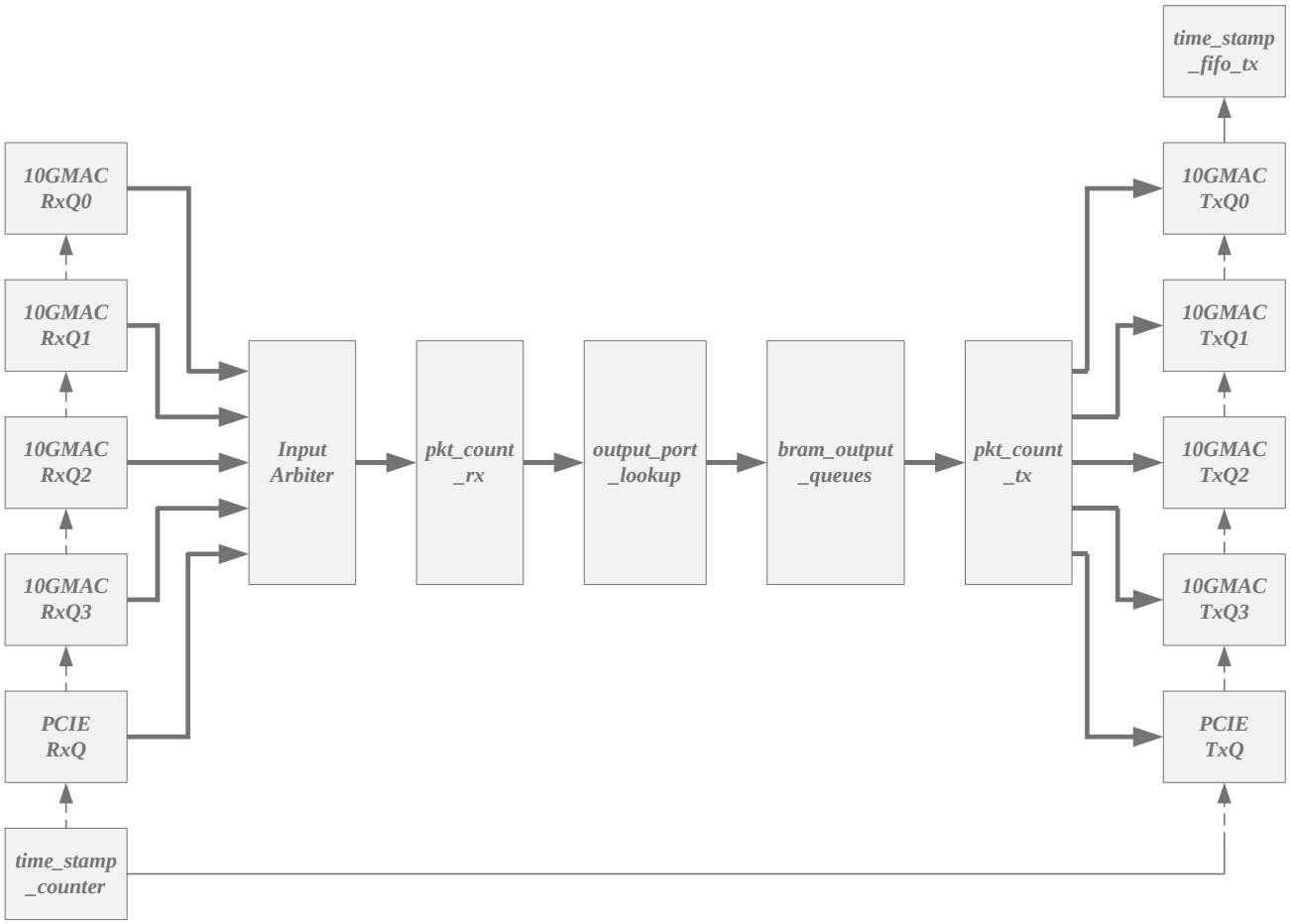

图 2.1　数据通道结构框图

如图 2.1 所示为数据通道基本结构，图中各个模块说明如下：

（1）*10GMAC RxQ* 和 *10GMAC TxQ* 封装于模块 *nf10_10g_interface*，分别表示数据接收和发送两个方向，与数据通道接口为 *AXI-Stream*，系统挂载 4 个 *nf10_10g_interface* 模块；相应的，*PCIE RxQ* 和 *PCIE TxQ* 表示 PCIE 模块的输入和输出，与数据通道接口为 *AXI-Stream*。

（2）*time_stamp_counter*为时间戳计数器，在*nf10_10g_interface*模块的输入（*10GMAC RxQ*）和输出（*10GMAC TxQ*）接口插入时间戳，即数据包接收和发送时分别插入时间戳，用于计算包延时；接收时间戳插入至数据包*AXI-Stream*接口的用户通道，传递至后级模块剥离后存储至*FIFO*，发送时间戳则直接存储于*FIFO*模块（*time_stamp_fifo_tx*），*MicroBlaze*可通过*AXI-Lite*接口读出。

（3）*Input Arbiter* 模块在 4 个 *nf10_10g_interface* 模块输入（*10GMAC RxQ*）和 PCIE 模块输入（*PCIE RxQ*）之间轮询，若有数据进来则将其拉出，发送至下一模块。

（4）*pkt_count_rx*模块对4个接收通道（*10GMAC RxQ*）的数据包进行计数，计数值存储于包计数*FIFO*，同时，依附于*AXI-Stream*接口的数据包接收时间戳也在该模块剥离，存储于时间

戳*FIFO*，*pkt_count_rx*模块通过*AXI-Lite*接口挂载于*MicroBlaze*，*MicroBlaze*通过该接口逐个读取*FIFO*中的包计数值和接收时间戳。

（5）系统分别将4个*nf10_10g_interface*和*PCIE*模块接口标记为端口0～4，每个数据包通过*AXI-Stream*接口传递时，会在TUSER位域部分标记其源端口和目的端口；*output_port_lookup*模块将接收端接收的数据包根据目的端口转发至各自输出端（*10GMAC TxQ*和*PCIE TxQ*），如从*nf10_10g_interface*0端口接收的数据可以转发至（*nf10_10g_interface*0～*nf10_10g_interface3*）中任一端口，具体由软件根据外部端口连接情况配置；此外，*output_port_lookup*模块完成数据采样功能，即对任一端口以某一采样间隔*N*（每接收N个数据包则采样一个）将接收数据包转发到*PCIE*模块，再由*PCIE*模块发送至PC；*output_port_lookup*模块通过*AXI-Lite*接口挂载于*MicroBlaze*，由*MicroBlaze*控制*output_port_lookup*模块的转发和采样。

（6）*bram_output_queues*模块缓冲接收到的数据包，将其再次转发出去；其次，*bram_output_queues*模块内部实现4个*FIFO*（一次写入循环读出），在*MicroBlaze*控制下（通过*AXI-Lite*接口）实现连续发送测试包功能，存储于*FIFO*中用于发送的特征包在系统初始化时通过*PCIE*模块写入。

（7）*pkt_count_tx*模块对发送到各个端口的数据包进行计数，计数结果分别存储于相应的*FIFO*，*MicroBlaze*可通过*AXI-Lite*接口将其读出。

3 数据通道寄存器说明

该部分对需要*MicroBlaze*通过*AXI-Lite*接口配置或读取参数的*IP*模块寄存器进行说明。数据通道*IP*模块寄存器可分为：1）需要*MicroBlaze*进行配置后启动工作的寄存器；2）保存*IP*运行结果的寄存器，需要*MicroBlaze*读取后进行分析计算。在*XPS*环境下生成的带*AXI-Lite*接口的*IP*模块，由*AXI-Lite*接口可读写的寄存器在*IP*内部用*Verilog*逻辑可表示为

reg [C_SLV_DWIDTH-1 : 0] slv_regi;（*i*为0、1、2…）

其中*C_SLV_DWIDTH*为寄存器位宽，系统所有寄存器位宽均为*32bit*；下面所说明的用户自定义的寄存器与上述寄存器的对应关系表示为：XXX——> *slv_regi*（*i*为0、1、2…），XXX即用户自定义的寄存器名称，即*MicroBlaze*对寄存器*slv_regi*（*i*为0、1、2…）进行操作，即是对XXX寄存器进行操作。

3.1 *time_stamp_fifo_tx*

该 *IP* 模块内部实现 4 个深度为 1024 的 *FIFO*（宽度为 *64bit*，时间戳值为 *64bit* 大小），存储 4 个 MAC 接口（MAC0~MAC3）的发送时间戳，即每个发送包到达 MAC 端口准备发送出去时将对应时间戳写入与该端口对应的 *FIFO*，设置 8 个只读寄存器（每个 *FIFO* 每次读操作使用两个寄存器完成），分别为：

time_stamp_mac0_l —> slv_reg0

time_stamp_mac0_h —> slv_reg1

time_stamp_mac1_l —> slv_reg2

time_stamp_mac1_h —> slv_reg3

time_stamp_mac2_l —> slv_reg4

time_stamp_mac2_h —> slv_reg5

time_stamp_mac3_l —> slv_reg6

time_stamp_mac3_h —> slv_reg7

其中，*time_stamp_macx_l（x=0、1、2、3 为相应端口号）*对应时间戳值低 32*bit*，*time_stamp_macx_h（x=0、1、2、3）*对应时间戳值高 32*bit*，即从某一 *FIFO* 读取一个时间戳值，首先读取寄存器 *slv_regx*（对应 *time_stamp_macx_l*）的值，后读取 *slv_reg(x+1)*（对应 *time_stamp_macx_h*）的值，即从 *FIFO* 读出一个完整的时间戳，依次循环可从 *FIFO* 中逐个读出所有时间戳。

3.2 *pkt_count_rx*

该模块内部实现 4 个接收包计数器（宽度为 *64bit*）和 4 个深度为 1024 的 *FIFO*（宽度为 *64bit*，时间戳值为 *64bit* 大小），用于存储接收时间戳，即接收包到达 MAC 端口时将对应的时间戳值写入相应端口时间戳 *FIFO*，设置 16 个只读寄存器，用于读取接收包计数值和接收时间戳：

pkt_count_mac0_l —> slv_reg0

pkt_count_mac0_h —> slv_reg1

pkt_count_mac1_l —> slv_reg2

pkt_count_mac1_h —> slv_reg3

pkt_count_mac2_l —> slv_reg4

pkt_count_mac2_h —> slv_reg5

pkt_count_mac3_l —> slv_reg6

pkt_count_mac3_h —> slv_reg7

time_stamp_mac0_l —> slv_reg8

time_stamp_mac0_h —> slv_reg9

time_stamp_mac1_l—> slv_reg10

time_stamp_mac1_h —> slv_reg11

time_stamp_mac2_l —> slv_reg12

time_stamp_mac2_h —> slv_reg13

time_stamp_mac3_l —> slv_reg014

time_stamp_mac3_h —> slv_reg15

其中，*pkt_count_macx_l*、*pkt_count_macx_h*（*x=0*、*1*、*2*、*3*分别表示端口号）表示计数值的低*32bit*和高*32bit*，读取两个寄存器值可得到对应端口当前接收的包数量。*time_stamp_macx_l*、*time_stamp_macx_h*（*x=0*、*1*、*2*、*3*分别表示端口号）与“3.1 *time_stamp_fifo_tx*”中所述发送时间戳寄存器类似，读取方法一致相同。

3.3 *output_port_lookup*

该模块实现不同端口间的数据的相互转发，如来自源端口MAC0的数据包可转发至其他任一端口发送出去，转发端口地址由寄存器控制，模块内部设置10个只写寄存器，分别对每个源端口设置一个目的转发地址：

dst_port_mac0 —> slv_reg0

dst_port_mac1 —> slv_reg1

dst_port_mac2 —> slv_reg2

dst_port_mac3 —> slv_reg3

dst_port_oped0 —> slv_reg4

dst_port_oped1 —> slv_reg5

dst_port_oped2 —> slv_reg6

dst_port_oped3 —> slv_reg7

pkt_samp_intel —> slv_reg8

dst_port_ctrl —> slv_reg9

其中，*PCIE*模块内部模拟4个端口，所以有*dst_port_oped0*～*dst_port_oped3*四个端口地址设置；*dst_port_macx(x=0*、*1*、*2*、*3)*表示来自源端口为*macx*的数据包的转发地址，*dst_port_opedx(x=0*、*1*、*2*、*3)*表示来自源端口为*oped*的数据包的转发地址，*dst_port_ctrl*控制数据采样（*dst_port_ctrl[7:0]*—设置采样源端口*(mac0*～*mac3)*、*dst_port_ctrl[15:8]*—设置采样目的端口*(oped0~oped3)*、*dst_port_ctrl[16]* —*使能采样*，该位设置为1时使能数据包采样），*pkt_samp_intel*设置采样间隔*N*。上述8个端口地址采用独热编码：

PORT_OPED0 = 8'b00000010

PORT_OPED1 = 8'b00001000

PORT_OPED2 = 8'b00100000

PORT_OPED3 = 8'b10000000

PORT_MAC0 = 8'b00000001

PORT_MAC1 = 8'b00000100

PORT_MAC2 = 8'b00010000

PORT_MAC3 = 8'b01000000

3.4 *bram_output_queues*

该模块实现两个功能：1）存储接收到的数据包（目前不使用该功能）；2）实现数据包的连续发送和控制。采样至*PC*的数据包也由该模块转发到*PCIE*接口进行发送。因此，模块内部设置4个*FIFO*（特征*FIFO*）实现测试包的一次写入循环读出，写入操作在测试开始前在*output_port_lookup*模块的控制下由*PC*发送至该*FIFO*。模块内部设置两个寄存器：

pkt_wr_ctrl —> slv_reg0

pkt_wr_status —> slv_reg1

其中，*pkt_wr_ctrl*为特征*FIFO*发送控制寄存器（该寄存器只写），*pkt_wr_status*为特征*FIFO*状态寄存器，通过该寄存器可获知特征*FIFO*是否已写入数据包（该寄存器只读）。*pkt_wr_ctrl[i]=1，i=0、1、2、3*，则对应*maci*的特征*FIFO*开始发送测试数据包，设置*pkt_wr_ctrl[i]=0*则停止发送。*pkt_wr_status[i]=1，i=0、1、2、3*，则指示由*PC*转发来的测试数据包已写入对应的特征*FIFO*，*MicroBlaze*通过读取该寄存器向PC发送应答信息。

注：在测试版程序中，数据包已例化在V代码中，不需要*PC*写入，则*MicroBlaze*通过控制寄存器*pkt_wr_ctrl*即可控制数据包的发送。

3.5 *pkt_count_tx*

该模块对*bram_output_queues*模块发送到各个端口（*maci，i=0*、*1*、*2*、*3*）的数据包进行计数，模块内部实现*4*个接收包计数器（宽度为*64bit*），分别对应4个端口。共设置*8*个寄存器对各个端口的包计数结果进行读取（每个计数值需要两个寄存器分别读取高32*bit*和低32*bit*）：

pkt_count_mac0_l —> slv_reg0

pkt_count_mac0_h —> slv_reg1

pkt_count_mac1_l —> slv_reg2

pkt_count_mac1_h —> slv_reg3

pkt_count_mac2_l —> slv_reg4

pkt_count_mac2_h —> slv_reg5

pkt_count_mac3_l —> slv_reg6

pkt_count_mac3_h —> slv_reg7

其中，*pkt_count_macx_l*、*pkt_count_macx_h（x=0*、*1*、*2*、*3*分别表示端口号）表示计数值的低*32bit*和高*32bit*，读取两个寄存器值可得到对应端口当前接收的包数量。

4 数据通道控制说明

在该测试版的数据通道中（缺少PCIE部分功能），对数据通道的控制只有两步：1）启动数据包的发送，2）读取模块计数结果（数据包计数和时间戳）。启动数据包发送即对*pkt_wr_ctrl*对应*bit*位写入*1*，如3.4节所述。可在数据通道工作过程中从各个模块读取运行结果，或在数据通道结束运行后读取（对*pkt_wr_ctrl*对应*bit*位写入*0*）。