基于 FPGA 的 AFDX 软硬件协同设计①

段晓峰, 马克杰, 周 江

(华东计算技术研究所, 上海 200233)

摘 要: 随着目前民用大型飞机的国家战略实施, 针对大飞机航电系统中骨干网络 AFDX, 提出一种基于 FPGA 的 AFDX 端点卡. 该方案基于软硬件协同设计的思想, 设计出一种新型的基于 PCIE 接口或基于 SoPC 的 AFDX 端点卡. 基于 FPGA 的 AFDX 端点卡成果可直接应用在军用大型运输机、直升机、战斗机、机场地面维护系统等平台上, 可以提升军用计算机的高安全、高可靠性能. 为各类武器平台电子设备的开发提供相关标准和规范, 可以缩短系统开发周期, 降低系统开发成本.

关键词: 实时网络; 端点卡; Virtex5 平台; AFDX 协议; 协同设计

AFDX HW/SW Co-Design Based on FPGA

DUAN Xiao-Feng, MA Ke-Jie, ZHOU Jiang

(East-China Institute of Computer Technology, Shanghai 200233, China)

Abstract: With our national strategy implementation involving large civil aircraft, AFDX endpoint card based on FPGA is proposed, which is specific to AFDX backbone network of large aircraft avionics system. Based on the thought of HW/SW co-design, the programme designs a new type of AFDX endpoint card with PCIE or SOPC interface. AFDX endpoint card based on FPGA can be directly applied in such platforms as large military transport aircraft, helicopters, fighter jets, airport ground maintenance system and so on and can enhance safety and reliability of military computer. What's more, development process will be accelerated and the cost will be lowed by offering related standards and specifications for electronic equipments of different kinds of weapon platforms.

Key words: real-time network; the endpoint card; Virtex5 platform; AFDX protoco; collaborative design

1 概述

从飞机的发展历程来看, 自 70 年代以来, 人们发现从飞机的外形、机体以及动力性能等方面提高飞机的整体性能已经变得越来越困难. 而同一时期, 由于信息工程、计算机、电子等技术的发展, 使得采用计算机作为飞机自动化的核心, 在飞机上建立指挥和控制综合系统成为可能. 现代飞机上各个电子设备或子系统(如飞控、雷达、通信导航等)都具有独立的计算机, 而这些计算机除了实现各自的功能以外, 还要进行信息交换, 从而达到功能综合的目的, 于是产生了航空数据网络(ADN, Aircraft Data Networks).

随着商业计算机工业取得的巨大成就, 将商业计算机通信模型应用于下一代航空电子系统已成为不可避免的趋势. 因此, ARINC 664(AFDX, Avionics Full Duplex Switched Ethernet 航空用全双工交换网络)应运而生, 与其他通信网络协议相比, AFDX 网络在可靠性和实时性方面提出了更严格的要求. 它在以太网的基础上充分应用商用现成技术(commercial off-the-shelf hardware, COTS)和开放式标准, 不但可以借鉴成熟技术, 获得数量庞大的第三方厂商的技术支持, 还可以缩短开发周期, 降低研发成本, 提高航空电子系统通信速率, 满足现代航电系统对信息传输和信息管理的需求.

高速、高可靠 AFDX 网络是实现综合模块化体系

① 收稿时间:2014-04-02;收到修改稿时间:2014-05-07

结构航空电子系统间大容量的高速数据交换的枢纽和核心.

但是在自主可控方面, 长期以来国外对于 AFDX 网络的核心芯片和关键技术进行了严格封锁, 所以为了能够摆脱国外的控制, 破除民机的技术壁垒, 确保国产大飞机项目的顺利开展和成功, 我们有必要投入人力、物力和精力进行航空网络技术的研究. 最终实现民机核心网络自主保障.

在技术发展趋势方面, 作为航空网络的主流协议, AFDX 端点卡目前主要使用国外的基于 PCI 接口的板卡, 因此对 PCI-E 接口的 AFDX 端点卡以及基于 SoPC 的 AFDX 端点卡的研制具有一定的现实意义.

2 总体设计思路

针对 AFDX 总线进行 FPGA 软硬件协同设计; 首先搭建 SOPC 验证环境, 技术路线为首先采用商用的 FPGA 开发板模拟实际使用板卡进行 IP 核验证. 然后把开发核移值到接口模块 PCI-E 总线接口上, 以满足军用领域特定任务要求;

FPGA 软硬件协同验证平台内部包含嵌入式 PowerPC440 硬核, 可以满足嵌入式 CPU 的需求, 运行 vxWorks 6.9. 在 FPGA 内部以 IP 核的形式集成内部总线控制器、DDR 控制器、Flash 控制器、串口(UART)、AFDX 总线接口控制器. 其结构框图如图 1 所示.

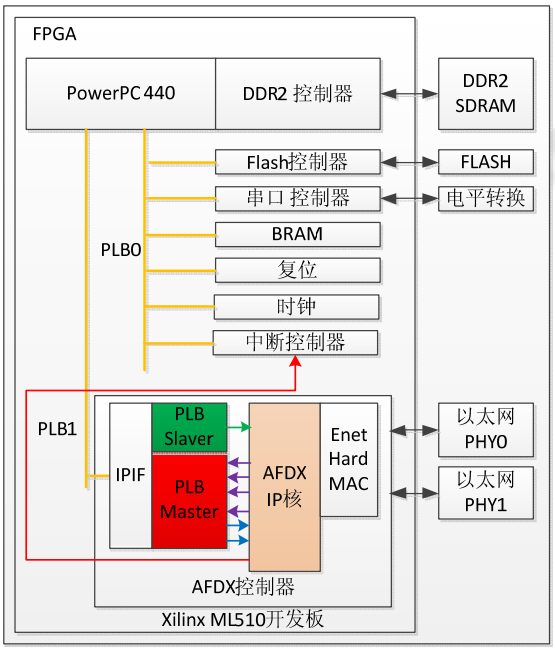

图 1 SoPC 软硬件协同设计模块图

其中主要以 AFDX 控制器逻辑设计和驱动设计为重点. 该 AFDX 控制器包括 Xilinx 的 IPIF 控制器用于连接 PLB; PLB Slaver 模块用于将处理器的读写寄存器变为内部寄存器配置 WISHBONE 总线; PLB Master 模块为 FPGA 中的 DMA 申请内存数据或更改内存数据提供通道; AFDX IP 核完成 AFDX 相关协议; Enet Hard MAC 为 AFDX 提供以太网 MAC 功能.

基于 PCI-E 总线接口, 如图 2 所示. 该控制器主要包括上述 AFDX IP 核, 完成 AFDX 相关协议; PCIE Slaver 模块, 用于将处理器的读写寄存器变为内部寄存器配置 WISHBONE 总线; PCIE Master 模块, 为 FPGA 中的 DMA 申请内存数据或更改内存数据提供通道; PCI E Interrupt 模块, 主要提供对一般中断与 PCIE 中断的适配.

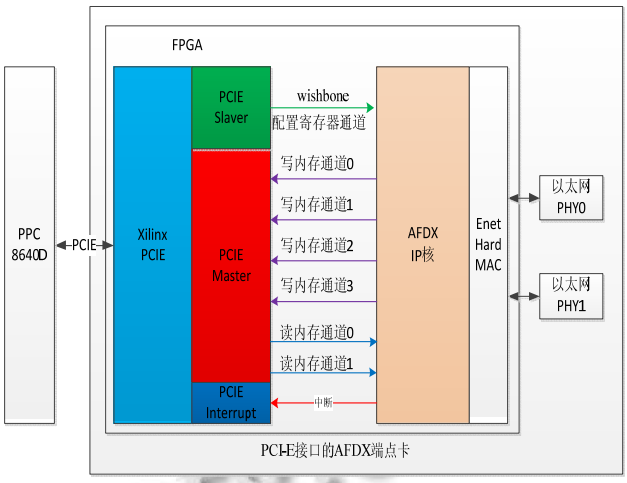

图 2 PCI E 总线接口模块图

通道 3 主要用于将 PHY1 收到的 AFDX 头信息写入内存; 写内存信号波形如图 3 所示. 读内存通道 0 主要用于读取 AFDX 头信息结构体通道; 读内存通道 1 主要用于读取 AFDX 数据通道. 其读数据波形如下图 4、图 5 所示.

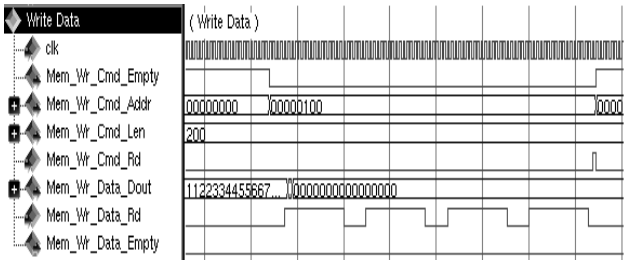

图 3 写内存通道时序图

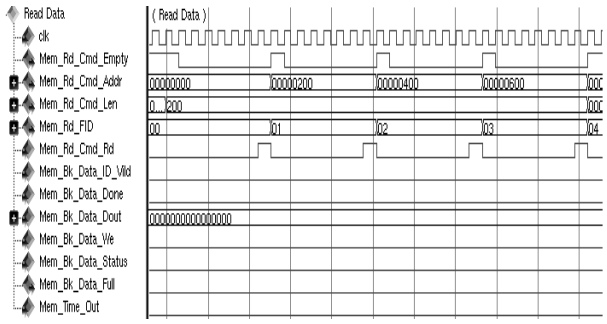

图 4　写内存命令通道时序图

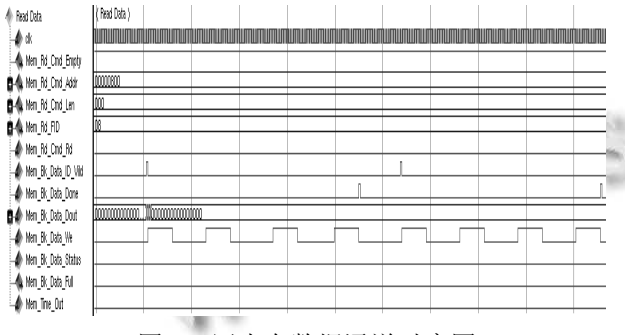

图 5　写内存数据通道时序图

3　软硬件协同设计方法

AFDX IP 核首先根据具体需求功能给软硬件合理划分. 在软硬件合理划分以后, 可以分别进行软件设计和硬件设计. AFDX 软硬件协同设计工具使用 EDK 设计实现. 用户可以通过集成在 EDK 中的 XPS(Xilinx Platform Studio)在 Windows 的图形界面下, 方便地调用各种工具, 完成整个 FPGA 系统的开发, 并且软硬件的开发可以同时进行. 正是由于在 EDK 环境下集成了丰富而完善的开发工具, 大量的 IP 核资源, 以及符合 Windows 标准的图形化界面, 使它已成为了目前性能比较优异的嵌入式微处理器开发工具. 但由于仿真功能不是很强大, 因此选用 modesim 软件进行仿真.

硬件部分设计采用 VHDL/ VerilogHDL 语言描述, 硬件描述经编译生成中间结构数据, 同时通过 EDK 转换算法将代码经过编译、链接, 生成可执行的硬件模拟器. 软件部分设计采用 C 语言描述, 软件描述与 RTOS 共同交叉编译、链接、标准目标代码处理, 形成能直接运行于目标处理器的指令集合, 由指令集模拟器解释执行. 通过 EDK 工具, 软硬件设计可以同时进行.

软硬件协同设计完成后把 EDK 工具生成的网表导入 Modesim 软件中进行仿真, 可通过仿真验证设计.

4　基于软硬件协同设计方法的AFDX IP核设计

依照 AFDX 协议技术指标, 首先按照上层协议栈(SNMP 和 ARINC615A)由软件实现; 底层协议(支持队列、采样、SAP 类型端口)由软硬件共同完成; 底层功能(支持 128 个 VL、支持完整性检测和余度管理功能、双冗余 10M/100M AFDX 端口、支持 PLB 总线接口)由硬件实现.

AFDX 功能模块可分为底层功能模块、底层协议模块、上层协议栈模块.

4.1 底层功能模块

按照 AFDX 协议, FPGA 中硬件主要处理调度算法和底层功能模块实现; 和软件一起实现底层协议.

按照 AFDX 协议划分为如图 6 所示.

1)发送模块

Slave: 外部总线转 Ram 总线, 用于访问内部双端口 Ram 与寄存器.

Cap_Ram: 用于存放 AFDX 256 个 VL 描述符, 包括 bag、VL 描述符地址等.

VL_Scan_Ctrl: 用于扫描 Cap_Ram 选出需要发送的 VL; 将需要发送的 VL 描述符地址写入 Desc_Fifo.

Desc_Fifo: 用于队列存放将要发送的 VL 描述符地址.

Desc Read: 使用 Desc_Fifo 中 VL 描述符地址向内存读取 VL 描述符请求.

Desc_Head_Ctrl: 用于处理 AFDX 数据包头信息.

Ans_Desc_fifo、Ans Data Fifo、AnsLenfifo 用于存放 VL 描述符 VL 信息、VL 描述符数据、VL 描述符长度.

DescAddr_RAM: 存放当前数据结构地址(指针).

Desc_Head_Ctrl: 根据 NDescAddr_RAM 存放下一个数据结构地址(指针), 载入下一个存放数据结构的地址.

Len_Fifo/data_fifo/info_fifo/ Desc Head Fifo(A/B): 存放带发送的网络报的数据. 跨过总线时钟和网路接口时钟的两个时钟域.

Afdx_Send_Ctrl: 将 Len_Fifo/data_fifo/info_fifo/ Desc Head Fifo(A/B)中的数据打成网络包.

2)接收模块

接收模块说明

Afdx_Recv_Ctrl 接收网络数据包, 将网络包头映

射成软硬件接口的数据结构, 数据包的头部存入 Desc Head Fifo. 网络包的数据部分直接存入 data_fifo. 并且计算产生长度数据存入到 Len_Fifo.

Sn 模块根据接收到的 Sn 标志, 判断当前 Sn 标志是否有效, 是否存在 Sn 标志的错误, 主要判断依据是根据 Afdx 的协议.如果连续接收到的 Sn 号是连续的则接收以太网数据帧, 如果出现 Sn 号不连续, 则置表示 Sn_vld 标志删除该以太网帧.

Desc Head Fifo 一旦接收到数据, 向 Afdx_Del_Ctrl 模块写入数据, Afdx_Del_Ctrl 主要判断接收数据的表示 Sn_vld 标志是否有效判断是否删除该帧. 进行删除操作, 如果该帧有效, 就向下一个模块 Select_H 和 Select_D 排队请求, 该模块将数据排队后, 向 Master 请求内存写.

当 Interrupt Fifo 中 VL 接收数据帧个数>6(暂定), 或者第一帧接收数据超过一定时间后, 发起中断, 由中断处理程序通知处理接收到的数据包.

3)中断模块

中断模块: 发送数据包完成以后. 中断产生模块将发送数据包的 Vl 信息存入模块的接收中断 fifo. 接收数据包完成以后. 中断产生模块将接收到的数据包的 Vl 信息存入模块的发送中断 fifo. 中断产生模块发现中断 FIFO(深度为 512, 最小 BRAM 使用即可)中存在数据即产生相应的中断给 Pcie 的中断控制器.

上位机中断响应函数发现接收中断产生, 处理数据并且根据配置告知 FPGA 下一个数据包的节点地址. 每次写节点地址的操作就将产生一次对于中断 FIFO 的读取操作. 如果 FIFO 中还有中断数据, 那么中断信号还会有效, 向 PCIE 设备发送中断请求. 驱动程序发现发送中断产生, 则指向一个发送的网络数据包的地址. 由 FPGA 完成接收和发送操作.

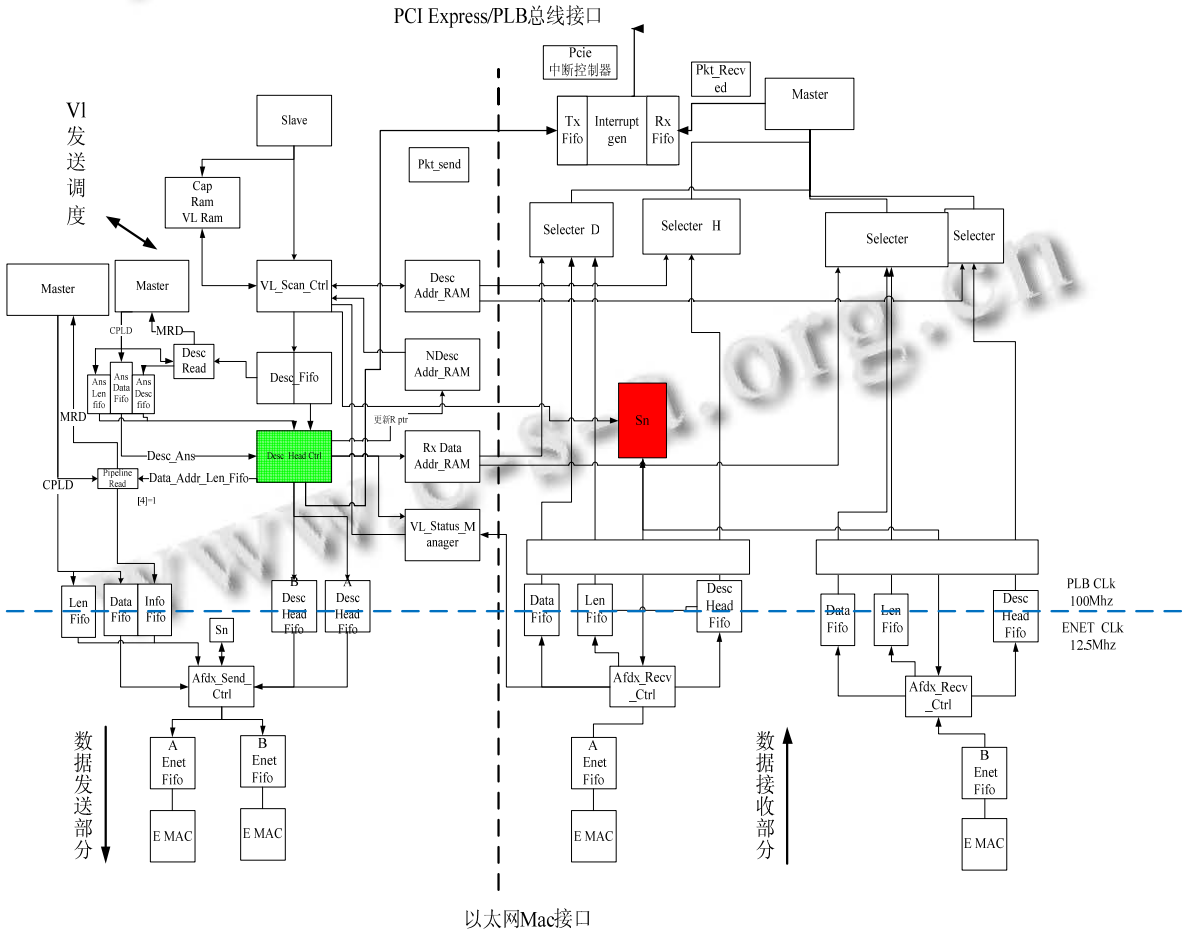

图 6 AFDX 协议的功能模块

4.2 底层协议模块

底层协议模块即软硬件交互模块为 Cap_Ram、Desc_Fifo 、 Desc Read 、 DescAddr_RAM 、Desc_Head_Ctrl、Desc_Head_Fifo、data_fifo、len_fifo 模块. 基本思想是在 RAM 中开一个链表队列, 硬件控制写指针, 软件控制读指针. 通过软硬件协同设计来完成 AFDX 的底层机制.

1)发送机制原理如图 7 所示;

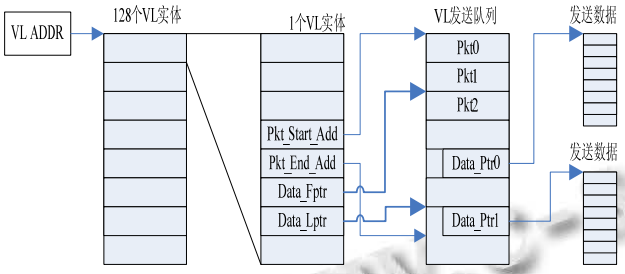

图 7　AFDX 软硬件协同发送机制

软件在 FPGA RAM 中建立 VL 数据结构, 大致如下图所示. 并对 128 个 VL(eciport)实体初始化.

a) VL Scanner 读取一个 VL 实体 VLn.

b) VL Scanner 判断该 VLn 在当前时隙可否发送, 若不为发送, Current_Bag++, 写入 Back Fifo, 由 VL Backer 写回相应内存(实现遍历与更改). 必须同时满足如下条件:

要用于判断起始时刻, 将不同 vl 的发送时刻错开.

发送对列不空.

c) VL Scanner 将需要发送的 afdx_desc_entity 地址写入 Fifo 对列.

d) Desc_Head_Ctrl 发现 Fifo 不空, 读取 afdx_desc_entity 结构体地址, 并根据 afdx_desc_entity 地址向 MASTER 请求数据.

e)当 afdx_desc_entity 结构体数据返回后, 组成以太网头信息, 写入 A、BDesc Head Fifo, 数据地址*addr 与长度 len 写入 Data Head & Len Fifo, * next 写入 NDescAddr_RAM 用于 VL_Scanner 的更新.

f) Data Head & Len Fifo 不空时, 向 MASTER 请求地址长度和 len 数据, 写入 Fifo.

g) Afdx_Send_Ctrl 发现 Len Fifo 不空, 则读取数据信息, 组成以太网包, 通过 E MAC 发送出去.

Sending Machine 通过总线 Master 读取内存数据, 并添加 VL、Port 等信息写入 Channel Data Fifo

Channel Mac Tx 发送数据包. 接收机制

a) 软件配置 VL Info 模块, Bag、Sn 信息, VL Info 模块将对 Bag、Sn 进行遍历管理, 记录当前该收的 SN.

b) 软件将每个 VL 接收地址写入查找表中, 用于存放 VL 包数据.

c) 当 Channel MAC Rx 接收到一帧后, 经过 Fifo, 输入到 Pkt Checker 模块.

e) Pkt Checker 检查: 发现为 VL 帧, 并与 VL Info 比较 SN 是否匹配. 若不匹配则删除 VL 帧.

f)与 VL Info 中该 VL 的 SN 是否匹配. 若不匹配则读 Fifo、删除 VL 帧.

g) Pkt Checker 读取 VL Addr 中的值, 并通过 Master 写入内存当中. 完成接收动作

h) SN++回写到 VL Info 模块中.

软件配置 VL Info 模块, Bag、Sn 信息, VL Info 模块将对 Bag、Sn 进行遍历管理, 记录当前该收的 SN.

软件将每个 VL 接收地址写入 VL_Addr 中, 用于存放 VL 包数据. 当 Channel MAC Rx 接收到一帧后, 经过 Channel Data Fifo, 输入到 Pkt Checker 模块.

Pkt Checker 发现为 VL 帧, 并与 VL Info 中该 VL 的 SN 是否匹配. 若不匹配则读 Data Fifo、删除 VL 帧.

与 VL Info 中该 VL 的 SN 是否匹配. 若不匹配则读 Data Fifo、删除 VL 帧.

Pkt Checker 读取 VL Addr 中的值, 并通过总线 Master 写入内存当中.

SN++回写到 VL Info 模块中.

向 Interrupt Fifo 写入中断信息.

当 Interrupt Fifo 中 VL 接收数据帧个数>6, 或者第一帧接收数据超过一定时间后, 发起中断, 由中断处理程序通知接收到数据包.

4.3 上层协议栈模块

总体来说, AFDX 的上层协议栈模块包含如下两个方面: 网络管理软件、数据加载软件. 如图 8 所示;

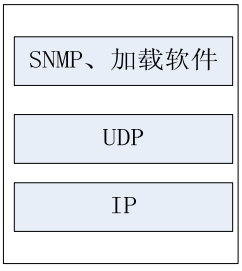

图 8　AFDX 协议栈

1)网络管理软件

AFDX 交换机的 SNMP 协议软件, 用于对 AFDX 网络上的各个交换机进行网络管理和健康维护, 并且能得到交换机上的各种 AFDX 异常情况. 从 SNMP 协议的规定看, 该软件需要实现 SNMP 体系结构中 Agent 节点的功能: 接收来自管理端发送的 SNMP 协议的 Get、Next、Bulk、Set 和 Config 消息, 并根据消息的类型和内容查询或设置交换机节点 MIB 库中的内容.

2)数据加载软件

ARINC615A 是 AFDX 网络中专用的数据加载协议, 它在 TFTP 的传输基础上针对 AFDX 定义了一些特殊的文件结构和操作方法. 数据加载器软件的总体结构如图 9 所示:

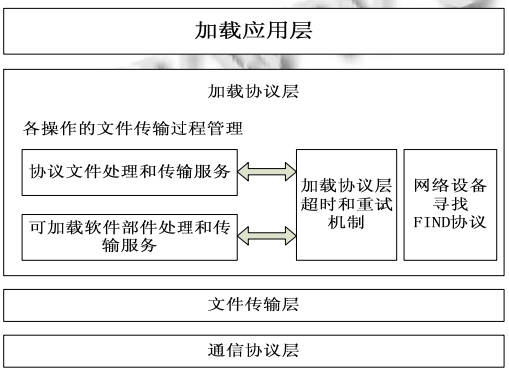

图 9 ARINC615A 数据加载协议图

如图所示, 数据加载软件分为四层:

文件传输层: 直至 TFTP 协议, 根据 ARINC615A 对 TFTP 协议做出了部分适应性修改, 对原有的错误包进行了扩展, 引入了“ABORT”消息. 该消息用于及时告知被加载设备终止此次操作, 使得被加载设备回到加载初始状态, 避免了被加载设备长期驻留某个操作的加载过程中;

数据加载协议层: 定义了符合 ARINC615A 协议操作过程, 包括信息操作、上传操作、下载操作和注册操作;

用户加载应用层: 组织或使用符合 ARINC615A 协议的数据. 对于交换机而言, 需要以从特定文件类型中提取出所需要的配置参数.

基本操作:

信息操作: 通过获得信息操作列表文件得到被加载设备及被加载设备上可加载软件部件的信息, 如: 被加载设备的名称, 被加载设备的硬件序列号, 被加载设备上的软件名称、部件等.

上传操作: 上传操作从数据加载器上传数据文件到被加载设备. 上传操作根据 ARINC665 标准定义, 应加载软件部件中的加载头文件, 对需要加载的数据文件进行索引加载. 根据对加载头文件中索引的数据文件的加载方式, 上传操作可分为全加载和增量加载.

下载操作: 下载操作将目标机上的软件或数据用文件的形式下载到目标机上. 下载操作有两种模式: 媒介定义下载模式和操作员定义下载模式.

5 验证测试

主要采用商用测试仪如 AIM、TechSAT 公司的 AFDX 网络协议测试仪对 AFDX 网络端系统进行测试. 如图 10 所示.

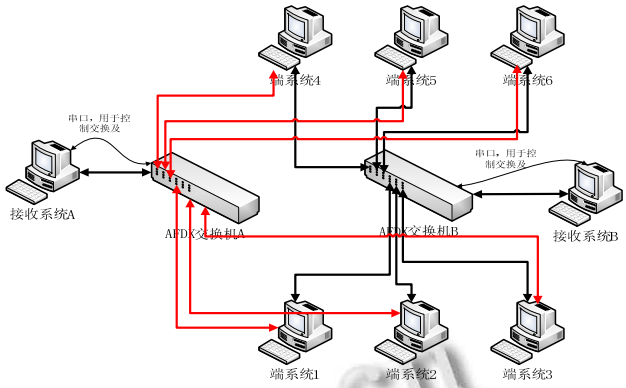

图 10 AFDX 端点卡验证系统

5.1 测试结果

在软硬件协同设计完成后把 EDK 工具生成的网表导入 Modesim 软件中进行仿真, 可通过仿真验证设计. 仿真主要针对发送模块、接收模块、调度模块仿真进行, 仿真结果如图 11、图 12、图 13 所示;

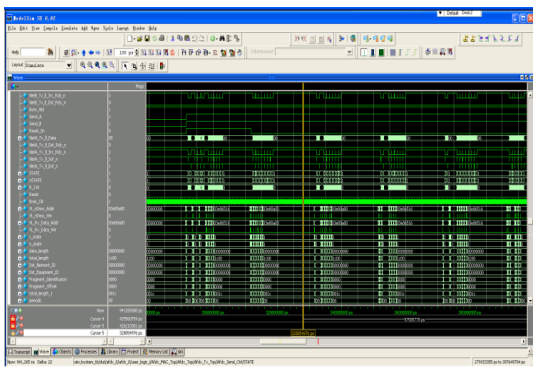

图 11 发送模块仿真图

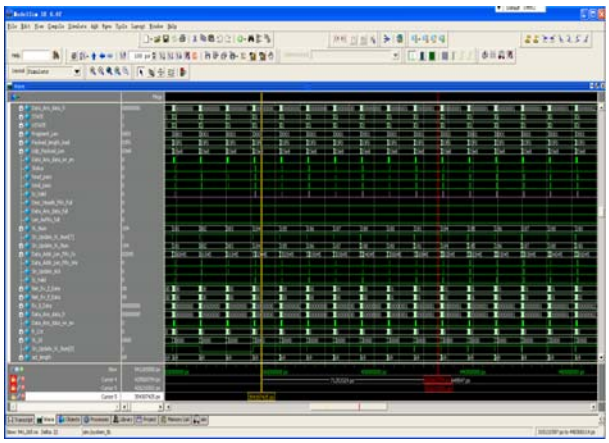

图 12　接收模块仿真图

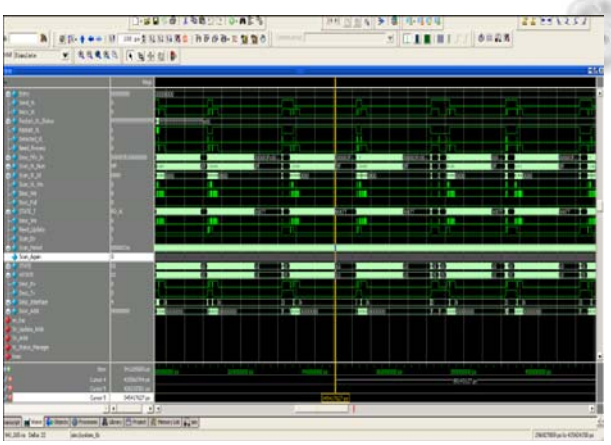

图 13　调度模块仿真图

仿真后 AFDX 接口模块通过验证平台检测, 功能和性能符合设计的要求.

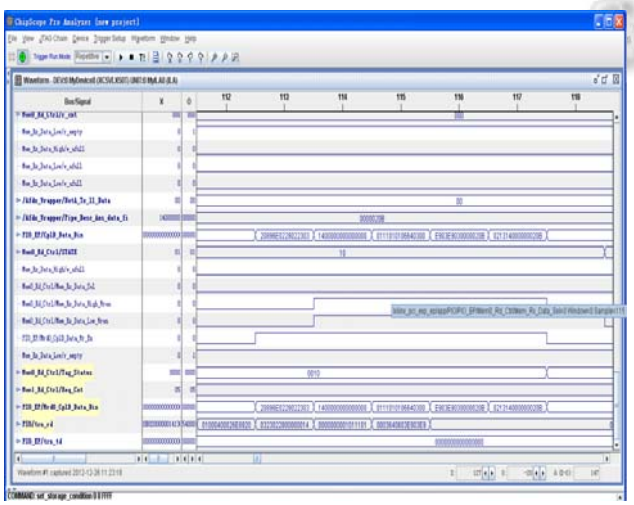

图 14　发送模块 chipscope 的波形图

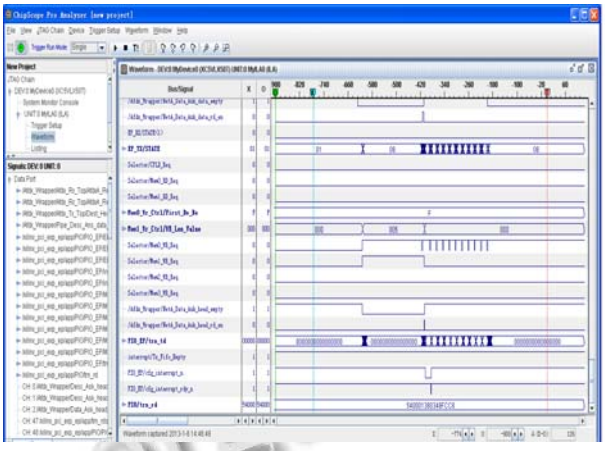

图 15　接收模块 chipscope 的波形图

基于 PCIE 接口的 AFDX 卡 FPGA 资源如下: Slice 和查找表, 以及 LUT 的 Ram 如表 1 所示;

表 1　Slice 和查找表, 以及 LUT 的 Ram

Slice Logic Utilization:	
Number of Slice Registers:	9,947 out of 28,800 34%
Number used as Flip Flops:	9,941
Number used as Latches:	4
Number used as Latch-thrus:	2
Number of Slice LUTs:	11,866 out of 28,800 41%
Number used as logic:	11,467 out of 28,800 39%
Number using O6 output only:	8,996
Number using O5 output only:	881
Number using O5 and O6:	1,590
Number used as Memory:	304 out of 7,680 3%
Number used as Dual Port RAM:	132
Number using O6 output only:	16
Number using O5 output only:	1
Number using O5 and O6:	115
Number used as Single Port RAM:	8
Number using O6 output only:	8
Number used as Shift Register:	164
Number using O6 output only:	164
Number used as exclusive route-thru:	95
Number of route-thrus:	1,016
Number using O6 output only:	971
Number using O5 output only:	41
Number using O5 and O6:	4

2)BlockRam 和 GTP 资源如表 2 所示:

表 2 BlockRam 和 GTP 资源

Specific Feature Utilization:		
Number of BlockRAM/FIFO:	53 out of 60	88%
Number using BlockRAM only:	53	
Total primitives used:		
Number of 36k BlockRAM used:	38	
Number of 18k BlockRAM used:	27	
Total Memory used (KB):	1,854 out of 2,160	85%
Number of BUFG/BUFGCTRLs:	8 out of 32	25%
Number used as BUFGs:	8	
Number of BUFDSs:	2 out of 6	33%
Number of BUFRs:	2 out of 24	8%
Number of DCM_ADVs:	1 out of 12	8%
Number of GTP_DUALs:	3 out of 6	50%
Number of LOCed GTP_DUALs:	3 out of 3	100%
Number of PCIEs:	1 out of 1	100%
Number of PLL_ADVs:	1 out of 6	16%
Number of TEMACs:	1 out of 2	50%

参考文献

1 Airlines Electronic Engineering Committee. Air-craft data network part 7avionics full duplex switched Ethernet (AFDX) network. ARINC 664. 2005.

2 Actel. Developing AFDX Solutions Mountain View: 2005.

3 Scharbarg JL, Ridouard F, Fraboul C. A probabilistic analysis of end-to-end delays on an AFDX avionic network. IEEE Trans. on Industrial Informatics, 2009.

4 Bisson K, Troshynski T. Switched ethernet testing for avionics applications. Autotestcon 2003. IEEE Systems Readiness Technology Conference.

5 Monticelli A, Murari CAF, Wu FF. A hybrid state estimator: solving normal equations by orthogonal transformations. IEEE Trans. on Power Apparatus and Systems, 1985.

6 Clements KA. Observability methods and optimal meter placement. International Journal of Electrical Power and Energy Systems, 1990.

7 Castillo E, Conejo AJ, et al. State estimation observability based on the null space of the measurement jacobian matrix. IEEE Trans. on Power Systems, 2005.

8 Bei G. Jacobian matrix-based observability analysis for state estimation. IEEE Trans. on Power Systems, 2006.

9 Vempati N, Shoults RR. Sequential bad data analysis in state estimation using orthogonal transformation. IEEE Trans. on Power Systems, 1991.

10 Costa AS, Loureco EM, Clements KA. Power system topological observability analysis including switching branches. IEEE Trans. on Power System, 2002.

11 Brajou F, Ricco P. The Airbus A380-an AFDX-based flight test computer concept. AUTOTESTCON 2004, Proceedings, 2004.

12 Afdx Software Network Stack Implementation-Practical Lessons Learned. Digital Avionics Systems Conference, 2009. DASC' 09.

13 Ding L, Song D, Zeng XX. The research of AFDX system simulation model. Multimedia Technology (ICMT), 2010 International Conference.

14 Yao M, Qiu Z, Kwak K. Leaky Bucket Algorithms in AFDX. Electronics Letters. 2009.

15 Anand M, Vestal S, Dajani-Bronwn S, Lee I. Formal modeling and analysis of the AFDX frame management design. Proc. of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing. 2006.