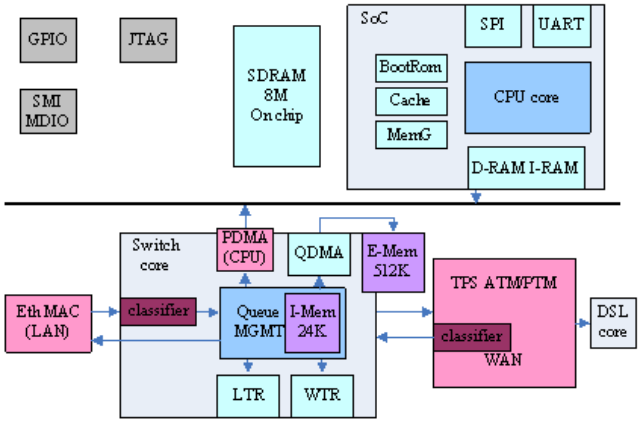
网络嵌入式设备

2013-05-26 15:57 by zmkeil, 777 阅读, 1 评论, 收藏, 编辑

这不是什么新鲜东西，无线路由器很早前就开始使用了，不过最近才慢慢理解其原理。现在网络嵌入式设备的功能越来越强大了，各芯片厂商的解决方法支持着这种复杂性。最近公司实习，做一个家庭网关的项目。下面的内容应该不算泄密吧，写得随意一点。

1.一个网络芯片架构



大家应该看得出来，这是一个DSL芯片。其网络部分由一个switch core构成核心，对外有3个交换口（粉红色的）：左侧为以太网MAC，可外接以太网芯片MAC或PHY，称为LAN端；右侧为DSL的TPS子层，按照DSL标准传输ATM/PTM数据，成为WAN端；上面的是一个PDMA片级总线，与片上系统SoC通信。

另外还有几个额外的交换口，主要是提供额外功能的：一个QDMA用于扩展core外Mem；LTR和WTR是两个转换引擎，当LAN或WAN端的数据包需要一些特殊处理时（如VLAN、PPPOE头等），会被分别交换到这两个口进行转换后，再发回Queue中。再另外，LAN、WAN端各有一个classifier，是预分类器，在数据包进入switch core前，先进行一个粗略的划分，决定发往哪个端口。

以上的这些功能都市switch core独立完成的，SoC系统只需对其进行简单的配置、控制即可。所以SoC的新能并不需要太高（CPU大概是130MHz的）。注意，该switch core和一般以太网switch的区别，首先其端口类型就不同，所提供的功能也跟为复杂。

这里给出一个简单的以太网switch芯片的架构，如下图所示，其核心就是一个register集，对它们进行配置（内部EEPROM、或外部MDIO总线），可以实现port-VLAN、二层filter等功能。其结构相对简单，因为其所有端口都是以太网口。

About

昵称：zmkeil
园龄：3年3个月
粉丝：37
关注：0
+加关注

最新评论

Re:Luci实现框架
您好，想请教一个问题，我想将Luci的admin-full下面的syslog显示功能移植到admin-mini，请问怎么实现？ -- zyzferrari

日历

<	2013年5月					>
日	一	二	三	四	五	六
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

随笔档案

- 2016年5月(2)
- 2016年2月(1)
- 2015年11月(1)
- 2015年2月(1)
- 2015年1月(1)
- 2013年8月(3)
- 2013年5月(9)
- 2013年4月(13)

随笔分类

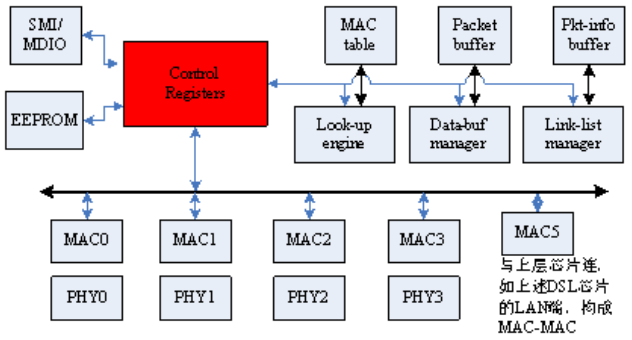
- Linux开发杂记(4)
- 编程语言C/C++/JAVA(5)
- 操作系统(4)
- 计算机架构(1)
- 算法(2)
- 网络相关(15)
- 信号处理DSP(2)
- 有感而发(4)

推荐排行榜

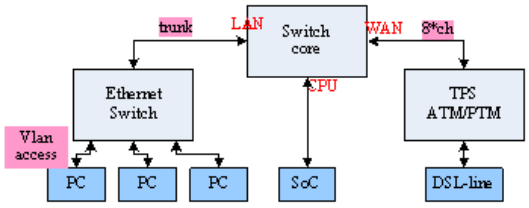
1. Linux下的虚拟Bridge实现(4)
2. 网络嵌入式设备(2)
3. 关于uC/OS的简单学习(2)
4. Luci实现框架(2)
5. uhttpd的实现框架(2)

阅读排行榜

1. Luci实现框架(12752)
2. uhttpd的实现框架(4069)



用该芯片接在上述DSL芯片的LAN端，构成网络系统如下图所示：



以太网switch可以自主实现port-VLAN功能，并通过一个trunk口与switch core相连。而switch core有很好的vlan_tag classifier功能。另外WAN端实现了8个硬件通道PVCs，且switch core也能对它进行很好的classifier。综上，就可以实现所谓的port-mapping功能，只要一跟DSL接入线，就可在家庭里实现IPTV、Internet、可视电话等业务的分离。

注意，所有这些功能都是switch core自动完成的，网络数据包不需要进入SoC的协议栈，这和后面讲的一般的路由器是不同的。

2.嵌入式片上系统

2.1与外界的交互方式

这里的SoC系统主要功能有：与外界用户的交互，解析用户指令，配置系统。

解析指令对软件系统来说很简单，配置系统，前面也说了，主要是读写一些register，也很简单。关键就在于与外界用户交互。

很容易想到的一种方式是网络，SoC也连在switch core上，有自己的IP、MAC。当然它和switch core是片上bus相连的，通信时并不需要MAC，这里只是把自己伪装成一个通用的以太网设备，可以被switch core和外界PC识别。

要通信，当然就需要协议栈了，不过，这里的SoC系统不需要处理额外（正常通信）的数据，所以协议栈也选择简单的LWIP，如之前的博文所述。最常用的网络通信方式就http了，另外还有telent等。

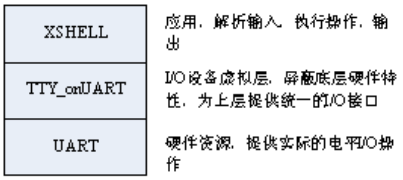
呵呵，连接192.168.1.1实际就是去连接其内部的SoC，一般的交换机、路由器都是这样的，而不是什么端口。当我还是一个超级菜鸟时，这个问题困扰了好久，纠结。

另外，嵌入式系统中，还有一个最常用的交互方式是串口UART。UART是一个非常简单的I/O设备，它通过直接读写管脚的电平信号（串行的）来实现输入输出，没有任何额外的中断、控制等机制。虽然简单，不能用以实现复

3. Linux下的虚拟Bridge实现(3963)
4. OpenWRT平台搭建及简单应用(3162)
5. Linux下VLAN功能的实现(1967)

杂、可靠的功能，但用作嵌入式系统的调试方法却非常有效。

UART就像是嵌入式设备的键盘/显示器。它是一种非常简单的硬件资源，在它之上可以构建通用的I/O设备tty，在tty之上，就可以实现各种应用，如shell等。



硬件资源UART、虚拟层设备tty都是系统的资源，在Uc/OS中，一般作为全局量，在其上的应用则通过task来完成。如XSHELL_TASK中，就是通过一个while(1)循环，不停地通过tty_get_line()读取命令行。注意，该函数已经不是裸的硬件操作了，而是加上了一个上层操作，即识别\r\n来作为结束符，也是通过一个while(1)循环来作的。读到电平为空，则忽略，因为UART太简单了，没有中断、缓存机制等（没有详细区考究，只是粗略地浏览了一下代码，好像是这样的吧！）。

最后，外界用户读写电平，当然不同用示波器了。呵呵，PC上装个串口驱动，那么PC的键盘/显示器就为嵌入式板子所用啦。

2.2bootloader

这就像一个心结，你一天不理解它，就一天不能安心地开发嵌入式系统，尽管你可以把软件写得很出色。

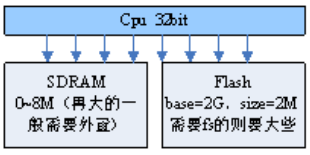
传统的PC机上电后，cpu核的指令指针（如cs:ip）会指向系统内某段固化的代码，如BIOS，这些代码被烧录在rom存储器中，断电也不会丢失。它们会调用我们开发的代码（如操作系统软件、或一些简单的前后台程序）。

一个嵌入式怎么启动，其实大意和PC（所有这种代码机器）差不多。当然不同厂商的芯片，也有各自的方式特点，以公司的这款芯片来说（注意，这里说的是芯片上集成的SoC子系统），它有好几种方式启动。

首先，其芯片内集成了一个BootRom，它里面的代码（也就是二进制的门电路）是在芯片的一部分，即芯片生产出来就有的。芯片对外有个引脚（boot-mode_pin），把它接低，则芯片上电后的ip指向该BootRom，执行里面的代码。这些代码很简单，一般会实现BOOTP、tffp等功能，从网络上下载OS的内核到内存中来运行。这就是所谓的网络无盘系统的工作方式。

不过现在好像这种方式用的少了，存储器便宜啊。一般都会把boot-mode_pin拉高，这样上电后ip指向外部flash。很容易想到flash和PC上的硬盘类似，是差不多，有点区别。PC上电后先执行BIOS，由BIOS装载硬盘的bootloader扇区。而嵌入式系统一般不这么麻烦，它直接就在flash中运行这些代码。

现在的cpu-core一般都是32位的，即有4G物理寻址空间，而嵌入式的SDRAM并不要那么大，所有可以把外部flash和SDRAM一起编址：

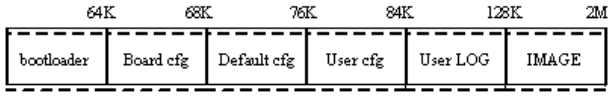


系统上电后，ip指向2G处，则可以直接在flash运行初始的代码，只是速率比较慢，所以开始的代码往往是把后面的一个image下载到SDRAM中去，然后在SDRAM中运行。

至于flash的基址为什么能是2G，这是由CPU的地址总线和flash的SPI总线的特殊的电路连接方式决定的，呵呵，电子出身的应该不难理解。而且，有些芯片还提高一些外部引脚pin，来为地址线加上一个offset（比如1M），那么两个处理芯片就可以使用同一个flash的不同的部分了（0~1M，1~2M），而其内部只觉得都是从2G地址开始的。

下面一个关键问题就是，flash中是什么，怎么来的，能改变吗？

一般flash中的东西，是有具体应用来决定的，开头一般都是bootloader代码，后面有一个image文件，公司的网络系统，需要一个启动配置文件，也简单的放在flash中：



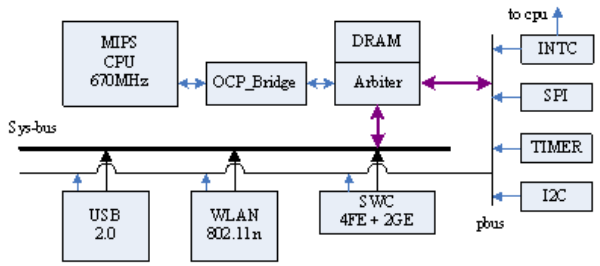
Bootloaer一般放在开头，便于执行。后面的则也可以直接这样按照物理空间分配，复杂一点，也可以做成文件系统fs，如嵌入式linux。主要包括一些配置文件，log信息文件。最重要的是IMAGE文件，一般要被加载到SDRAM中去运行，这也是我们开发系统功能应用的关键。

那这些内容怎么来的呢？一般初始时，会用特定的硬件工具厂商提供的一个初始的文件（格式就是上图所示）烧到flash中，就像当年我们烧8051单片机一样，再把flash焊在板子上。如果运行过程中，代码被改死了，系统再也起不来了，那对不起，只能重新焊一个新的flash了。而一个不就的方法是在flash中准备两个image，板子上预留一个特殊的案件，按下后启动新的image（前面讲的flash-offset方法）。还有一种方法，是通过内部bootrom启动，无盘启动。实现当然需要一定的硬件支持，如前面所述的那个boot-mode_pin不能焊死，用一个跳冒，或者通过一些特殊电路接在以太网上，以太网事先有数据传输时，则使用内启动，如组播升级。方法各不一啦。

最后，flash里面的内容怎么变？很简单，它既然也在cpu的变址内，直接用cpu把数据写到flash中不就行了。一般用http服务，client会post一个image文件上来，server端检查没问题，就写到flash中，这就是手动升级。

3.一个无线路由器的架构分析

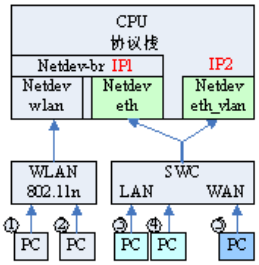
前面提到的公司芯片解决方案，SoC子系统的功能很有限，通信数据包一般不会到SoC的协议栈。而目前市场上的无线路由器系统，系统中的CPU功能一般很强大，RAM配置也很强大，一般都能运行大型系统软件，如Linux。



这是一块单独的芯片，而其几乎包含了一个完整的PC机主板上的所有内容，CPU的功能还是很强劲了，670MHz的MIPS核，通过桥片连接数据总线。这里的sys_bus差不多相当于PC中的PCI总线，诸如USB控制器等都挂在

其下。这里主要看一下网络设备，主要有两个，一个802.11n标准的WLAN收发器，一个4FE+2GE的以太网交换机。

和之前介绍的DSL芯片不同，他的两个网络接口都是接在系统总线上，而没有通过一个switch-core交互。因此，CPU内必须维护一个功能完整的协议栈，而且SWC往往也被虚拟化为wan和lan口，参见下面的结构图：



首先，先明确一个概念，在统计网络设备的端口数，不要忘了还有一个端口连接CPU的。比如，SWC对外有4FE+2GE，而实际上它还有第七个端口连接到CPU。然后就能理解有些数据传输只在这些只能得网络设备中完成了，而有些则需要进入CPU的协议栈。更准确的概念应该是，片内CPU系统也相当于一个PC，共同接在交换设备上。

这里比较特殊的是这个SWC设备，照理它的所有端口应该是等价的，所处网段也相同，但通过VLAN技术，把3、4、CPU端口化为一个VLAN，而5、CPU化为另一个VLAN，这是SWC硬件支持的port-mapping，它能只允许同一个VLAN下的机器通信。这里就有一个特殊的CPU端口，它同属于两个VLAN，称为trunk，SWC硬件在trunk口下收发数据时，必须带有vlan_tag（802.1q）。在CPU系统内，因为物理通路只有一个，只能通过软件的方式来实现VLAN的划分（如Linux下的VLAN）。另外，一般把wlan挂在LAN内，所以在Linux中可以用一个虚拟Bridge设备来连接这两个设备。下面来看几种数据交换的途径：

1. 3、4间通信，在PC机发起的ARP协议阶段，SWC也学习并记录了MAC表，SWC可以直接交换。
2. 1、2间通信，也像ethernet交换机那样，直接在wlan设备中交换吗？没研究过wlan协议啊，姑且这样认为吧。
3. 3、1间通信，是同一网段的，DMAC为PC1的MAC地址，在SWC中会将它从CPU端口发出去，当然会加上lan_vlan-tag，因此会被netdev_eth设备接收到，而该设备已经变成了bridge设备的一个端口，因此bridge设备会接管该pkt，并根据MAC表，从netdev_wlan设备（已经是端口啦）发出，wlan设备受到pkt后，根据802.11n协议，发给对应的PC。
4. 5和其它通信，不管是3、4还是1、2，都是不同网段，因此在PC的路由系统中，会将数据发往gw-IP2，即pkt的DMAC为片上CPU的MAC，那么在SWC中会将pkt从CPU端口发出去，当然会打上wan_vlan-tag，因此会被netdev_eth-vlan设备接收，并进入片上CPU系统的协议栈（注意，之前的3种情况都不会进的）。此时片上CPU系统就充当了路由功能，选择bridge设备下发，bridge设备中根据MAC选择对应端口下发。
5. 其它向5通信，和上面一样，只是方向反过来。
6. 片上CPU系统自身也有MAC、IP（192.168.1.1），外部PC发送数据，通过wlan、swc设备到达CPU系统的interface，interface发现是发给自己的，则会传递给上层协议栈。

当然这里只是最基本的情况，现在的路由器设备已经集成了非常多的功能，如DHCP、NAT、DNS等等。

好文要顶 关注我 收藏该文

zmkeil
关注 - 0
粉丝 - 37

2 0

+加关注

« 上一篇：三皇五帝
» 下一篇：一个VLAN配置的实际例子

分类：网络相关

#1楼 wantongtang
2015-01-15 04:58

ADD YOUR COMMENT

我是您的粉丝啊，大神一样的人物。。。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。