用于交换芯片地址表查找的快速并行Hash算法研究

曹晓东，石 寅，张 强

(中国科学院半导体研究所 北京 海淀区 100083)

【摘要】为了以尽量少的硬件资源实现高效能的二层交换地址表查找功能，通过分析交换芯片地址表以及循环冗余算法的特点，提出了一种硬件电路由寄存器和异或门构成的10位并行Hash算法。通过并行地将输入信号帧的48位物理地址转换为10位的地址表查询地址，可以快速准确地查询1 024存储深度的地址表，采用该地址表查询算法的二层交换芯片实现了线速交换，从而有效提高了所实现网络设备的性能。仿真显示，算法生成的Hash地址较为均匀地分布在其10位地址空间内，有效地降低Hash冲突发生的几率。采用FPGA实现的交换电路进一步验证了算法的优异性能。

关 键 词 地址查询; 循环冗余码; 哈希算法; 交换芯片; 均匀分布

中图分类号 TN4 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.02.025

Research on Fast Parallel Hash Algorithm for Switch Chip Address Lookup

CAO Xiao-dong, SHI Yin, and ZHANG Qiang

(Institute of Semiconductors, Chinese Academy of Sciences Haidian Beijing 100083)

Abstract In order to implement a high performance layer 2 switch address lookup function with minimum hardware consumption, a 10-bit Hash algorithm consisting of registers and XoR gates was presented by analyzing the characteristics of switch chip address table and cyclic redundancy check algorithm. 48-bit physical address is transferred into 10-bit lookup address in parallel and the address table of 1 024 storage depth can be quickly and accurately searched. The layer 2 switch chip using this address lookup algorithm can implement line speed exchange. The performance of the network equipments using the switch chips can be improved. The generated Hash addresses are uniformly distributed in the 10-bit address space. The performance of the algorithm was further verified by using switch circuit implemented on FPGA.

Key words address lookup; cyclic redundancy checks; hash algorithm; switch chip; uniform distribution

随着计算机和通信技术的发展，以Internet为代表的计算机网络技术得到了飞速发展，这不仅极大地丰富了信息的内容和表现形式，也为信息交流与资源共享提供了更多的途径和可能。但以太网数据的传输和交换速度制约了以太网的发展，因此需要更高性能的网络设备来提高网络速度。

交换机作为重要的网络设备，其性能直接影响着所在网络的网络速度。二层交换机中存在一个存储MAC地址、VLAN信息以及端口信息的地址表，交换机利用接收信号的源MAC地址以及VLAN信息查找到其对应的表项，从而获得其对应输出端口号。在二层交换机中交换芯片的地址表查找速度是决定交换机交换速度的重要因素，二层交换机中采用的地址表查询方法主要有二分查找法[1]、低位提取查找法[2]、基于CAM(内容可寻址存储器)的查找法[1,3-4]和基于Hash函数的查找法[1,5]。二分查找法的主要缺点是每次更新地址表时都要重新排列整个地址表，且在更新地址表时无法对地址表进行查找。低位提取查找法具有简单、成本低的特点，已在商业网络交换控制芯片中成功应用。由于低位提取查找法直接提取MAC地址的低位(LSB)用作地址表的查找，因此其地址查找的关键字对MAC地址信息利用率较低从而影响了地址表查找的命中率，进而引起过多的网络多播影响网络利用率。CAM是一种将输入数据与CAM中存储的所有数据项同时进行比较，并迅速判断输入数据是否与CAM中存储的数据项相

收稿日期：2012 – 12 – 10；修回日期：2014 – 01 – 08
基金项目：国家科技重大专项(2009ZX01031-002-002)
作者简介：曹晓东(1980 –)，男，博士，主要从事SoC、数模混合集成电路方面的研究.

匹配的特殊存储阵列，可用作交换机的地址表存储器。在相同存储容量的情况下，CAM的芯片面积为普通RAM面积的4～5倍，因此CAM价格昂贵而且存储容量有限。Hash函数是一种数学函数，可以将大地址表映射为较小的地址表[6]。在二层以太网交换芯片中，Hash函数通过运算将48位的MAC地址转换为10～12位的数据[7]，该数据将作为2^{48}存储深度地址表子集的地址指针。以太网交换芯片输入信号48位的MAC地址无法唯一地映射到10～12位地址数据，多个地址会共用同一个指针，这会引起地址查询过程中的Hash冲突，从而导致不必要的网络广播。采用溢出缓冲器存储引起Hash冲突的地址信息可以降低Hash冲突对网络交换性能的影响，但实际交换芯片中很少采用这种结构[7]。Hash算法已被广泛应用于网络设备的地址表查找，但不同Hash算法在Hash冲突与实现复杂度方面存在差异，因此选择一个合适的Hash算法可以有效地降低交换芯片的实现复杂度，并减少交换芯片工作过程中的Hash冲突。

本文通过分析交换芯片地址表以及Hash算法的特点，提出并设计了一种硬件电路由寄存器和异或门构成的10位并行循环冗余算法。该算法可以并行地将输入信号帧的48位物理地址转换为10位的地址表查询地址，从而快速准确地查询1 024存储深度的地址表，采用该地址表查询算法的二层交换芯片可以实现线速交换。

1 Hash算法分析

Hash函数是一种可以将长地址映射为短地址的数学函数。一个Hash函数的输出与输入之间必须具有固定对应关系且输出相对输入是随机的。在二层以太网交换芯片中，利用Hash函数的这些特点可以实现48位MAC地址映射为10～12位的地址，极大地减少二层以太网交换芯片地址表的尺寸并提高地址表的查找速度。很多性能优异的Hash函数被用作计算机网络的地址查找。

1.1 基于部分地址位的Hash函数

基于部分地址位的Hash函数直接提取输入地址信号的部分地址位作为Hash函数的输出，因此基于部分地址位的Hash函数是最简单的Hash函数。在以太网LAN中任意一个48位MAC地址的网卡都具有唯一的标识，MAC地址的高24位是由IEEE分配的厂商地址，低24位由网卡厂商分配，因此作为以太网交换芯片的地址查找关键字MAC地址的高24位具有较少的信息，而第5个字节具有最多的信息。在二层以太网交换芯片中，以MAC地址的第5个字节为中心向两边的相邻字节分别取1～2位以构成具有较高信息量的10～12位的地址表查找关键字。该算法的缺点是没有完全应用整个输入地址信号的信息来生成地址查找关键字，因此降低了其作为Hash函数进行地址表查找的能力。

1.2 基于异或门的Hash函数

基于异或门的Hash函数是一种简单而且快速的Hash函数，其输出的每个索引位都是由输入地址数据位的子集通过异或运算得到的[9]。假设Hash函数的输入地址数据A可以用二进制表示为$a_{n-1}a_{n-2}\cdots a_1a_0$,输出$m$位的地址索引$H$可以用二进制表示为$h_{m-1}h_{m-2}\cdots h_1h_0$，将输入地址数据$A$的$n$个二进制位要分成$m$组，各组的所有二进制位分别异或运算即可得到$m$位Hash函数输出。

1.3 基于Fletcher校验的Hash函数

Fletcher校验是一种计算位置无关校验和的算法[8]。Fletcher校验算法易于软件实现，因此主要用于ISO/OSI传输。假设输入n个字节的数据分别为$B[0],B[1],\cdots,B[n-1]$且16位Fletcher校验的输出分别为C_0和C_1，那么输入的8位数据首先与C_0相加得到新的C_0值，然后新的C_0值与C_1值相加得到新的C_1值，在输入数据结束时就可以计算出Fletcher校验值。

$$C_0 = C_{0\text{prev}} + B[i] \tag{1}$$

$$C_1 = C_{1\text{prev}} + C_0 \tag{2}$$

式中，$B[i]$为输入的第i个字节的数据，i=0,1,$\cdots$,n–1；C_0和C_1的初值均为0。

$$\text{Checksum} = -(C_0 + C_1) \tag{3}$$

将对应$B[n-1]$的C_0和C_1代入式(3)即可计算出Checksum的低字节Checksum_0。将Checksum_0代入式(1)$B[i]$位置计算出C_0和C_1，将C_0和C_1代入式(3)即可计算出Checksum的高字节Checksum_1。

Fletcher校验算法的缺点是无法区分全0或者全1的数据，0x00与0xFF的16位Fletcher校验和相同，因此降低了其作为Hash函数进行地址表查找的能力。

1.4 基于循环冗余校验的Hash函数

基于循环冗余校验的Hash函数作为数据检错算法被广泛地用于数据通信、数据存储、数据压缩等领域[10]，其作为地址表查找关键字产生算法在以太网LAN的地址表查找中也得到了广泛应用。输入数据的循环冗余校验码由输入数据与生成多项式模2除运算得到。输入数据D可用一个多项式$D(x)$表示[11]。

$$D(x) = d_{l-1}x^{l-1} + d_{l-2}x^{l-2} + \cdots + d_1x^1 + d_0 \tag{4}$$

式中，$d_{l-1}d_{l-2}\cdots d_1d_0$为用二进制表示的输入数据$D$，$l$为输入数据$D$的二进制位数。

CRC生成多项式$G(x)$决定了CRC算法的检错能力、实现复杂度以及地址表查找的应用性能。

$$G(x)=g_{i-1}x^{i-1}+g_{i-2}x^{i-2}+\cdots+g_1x^1+g_0 \qquad (5)$$

式中，$g_{i-1},g_{i-2},\cdots,g_1,g_0$分别为0或者1，$i$为CRC生成多项式的长度。

输入数据多项式$D(x)$与x^i相乘后，再模2除CRC生成多项式$G(x)$，得到的余数即为CRC的值。

$$\mathrm{CRC}(D)=D(x)\times x^i \bmod G(x) \qquad (6)$$

一个可以生成优良循环冗余校验码的CRC生成多项式可以作为一个性能优越的Hash函数[4]。CRC是一种功能强大且易于实现的算法，许多CRC生成多项式在以太网数据校验、USB数据校验、ATM机数据校验等应用中得到了成功应用[12]。

2　快速并行Hash算法

通过前面的分析可知，基于CRC的Hash函数是一种优越的Hash函数。基于线性反馈移位寄存器的CRC生成电路以串行方式输入数据，因此每个时钟周期内只能计算一个二进制数据位。利用寄存器和异或门实现的基于线性反馈移位寄存器的CRC-10硬件电路图如图1所示。

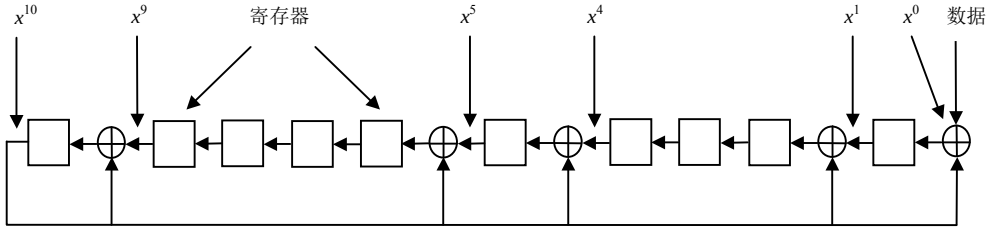

图1　基于线性反馈移位寄存器的CRC-10硬件电路图

在以太网交换芯片的地址表查找中，需要Hash电路能够并行地将输入信号帧的48位物理地址转换为10～12位的地址表查找关键字，这要求将基于CRC的串行LFSR电路转换为并行10～12位输出的电路，从而使其能够在一个时钟周期内并行地输出10～12位的数据。在具有1 024存储深度的地址表的以太网交换芯片中，可以采用CRC-10生成多项式来构建从48位物理地址转换为10位的地址表查询地址的Hash函数。

$$G(x)=x^{10}+x^9+x^5+x^4+x^1+1 \qquad (7)$$

式(7)为CRC-10生成多项式，其主要应用在ATM通信中。

许多不同的并行CRC算法被提出并得到应用[10]，这些算法或者注重工作速度或者注重消耗的逻辑资源，但其中多数算法仅仅是理论上的研究，难以采用硬件来实现。本文利用CRC算法的线性特性设计一种10位数据宽度的并行CRC算法，采用该算法的二层交换芯片并行地将输入信号帧的48位物理地址转换为10位的地址表查询地址，从而快速准确地查询1024存储深度的地址表，以实现线速交换。该算法采用式(7)的CRC-10多项式，以如下步骤实现48位位宽数据输入和10位位宽数据输出的并行CRC算法。

1) 选定数据宽度N和CRC多项式的宽度。在以太网交换芯片的MAC地址查找中N选择48位，CRC的宽度选择10位。

2) 根据选择的CRC多项式实现一个串行CRC算法$\mathrm{CRC}_{\mathrm{serial}}$，然后通过$N$次调用$\mathrm{CRC}_{\mathrm{serial}}$实现一个并行的CRC算法$\mathrm{CRC}_{\mathrm{parallel}}(D_i,G_i)$。

3) 构造一个矩阵$\boldsymbol{H}_1$，该矩阵表示G_i为0即仅有输入数据D_i决定的CRC的下一个状态$\mathrm{CRC}_{\mathrm{next}}$的值。输入独热码数据，并计算矩阵值。

4) 构造一个矩阵$\boldsymbol{H}_2$，该矩阵表示D_i为0即仅有输入数据G_i决定的CRC的下一个状态$\mathrm{CRC}_{\mathrm{next}}$的值。输入独热码数据，并计算矩阵值。

5) 将对应不同$\mathrm{CRC}_{\mathrm{next}}$位的矩阵$\boldsymbol{H}_1$和矩阵$\boldsymbol{H}_2$的对应高电平位进行异或运算，最终得到对应CRC-10多项式的并行CRC算法。

表1　并行CRC的$\boldsymbol{H}_2$矩阵

D_i=0	GO [10]	GO [9]	GO [8]	GO [7]	GO [6]	GO [5]	GO [4]	GO [3]	GO [2]	GO [1]
G_i[1]	0	1	1	0	0	0	1	1	1	0
Gi[2]	1	1	0	0	0	1	1	1	0	0
Gi[3]	0	0	0	1	0	1	1	0	1	1
G_i[4]	0	0	1	0	1	1	0	1	1	0
G_i[5]	0	1	0	1	1	0	1	1	0	0
G_i[6]	1	0	1	1	0	1	1	0	0	0
G_i[7]	1	1	1	1	0	1	0	0	1	1
G_i[8]	0	1	1	1	0	0	0	1	0	1
G_i[9]	1	1	1	0	0	0	1	0	1	0
G_i[10]	0	1	0	1	1	1	0	1	1	1

表1为计算得到的并行CRC的$\boldsymbol{H}_2$矩阵。采用上述步骤计算得到的用于二层交换芯片的Hash地址算法

如下：

GO[10]=DI[48]^DI[47]^DI[45]^DI[44]^DI[43]
^DI[42]^DI[41]^DI[39]^DI[38]^DI[35]^DI[34]
^DI[30]^DI[28]^DI[27]^DI[26]^DI[20]^DI[15]
^DI[14]^DI[13]^DI[12]^DI[11]^DI[1]^GI[9]^GI[7]
^GI[6]^GI[2]

GO[9]=DI[47]^DI[46]^DI[41]^DI[40]^DI[38]^DI[36]
^DI[34]^DI[31]^DI[30]^DI[29]^DI[26]^DI[21]
^DI[20]^DI[16]^DI[2]^GI[10]^GI[9]^GI[8]^GI[7]
^GI[5]^GI[2]^GI[1]

GO[8]=DI[48]^DI[47]^DI[42]^DI[41]^DI[39]^DI[37]
^DI[35]^DI[32]^DI[31]^DI[30]^DI[27]^DI[22]
^DI[21]^DI[17]^DI[12]^DI[3]^GI[9]^GI[8]^GI[7]
^GI[6]^GI[4]^GI[1]

GO[7]=DI[48]^DI[43]^DI[42]^DI[40]^DI[38]^DI[36]
^DI[33]^DI[32]^DI[31]^DI[28]^DI[23]^DI[22]
^DI[18]^DI[13]^DI[4]^GI[10]^GI[8]^GI[7]^GI[6]
^GI[5]^GI[3]

GO[6]=DI[48]^DI[47]^DI[45]^DI[42]^DI[38]^DI[37]
^DI[35]^DI[33]^DI[32]^DI[30]^DI[29]^DI[28]
^DI[27]^DI[26]^DI[24]^DI[23]^DI[20]^DI[19]
^DI[15]^DI[13]^DI[12]^DI[11]^DI[5]^GI[10]^GI[5]
^GI[4]

GO[5]=DI[47]^DI[46]^DI[45]^DI[44]^DI[42]^DI[41]
^DI[36]^DI[35]^DI[33]^DI[31]^DI[29]^DI[26]
^DI[25]^DI[24]^DI[21]^DI[16]^DI[15]^DI[11]^DI[6]
^GI[10]^GI[7]^GI[6]^GI[4]^GI[3]^GI[2]

GO[4]=DI[48]^DI[47]^DI[46]^DI[45]^DI[43]^DI[42]
^DI[37]^DI[36]^DI[34]^DI[32]^DI[30]^DI[27]
^DI[26]^DI[25]^DI[22]^DI[17]^DI[16]^DI[12]^DI[7]
^GI[9]^GI[6]^GI[5]^GI[3]^GI[2]^GI[1]

GO[3]=DI[48]^DI[47]^DI[46]^DI[44]^DI[43]^DI[38]
^DI[37]^DI[35]^DI[33]^DI[31]^DI[28]^DI[27]
^DI[26]^DI[23]^DI[18]^DI[17]^DI[13]^DI[8]^GI[10]
^GI[8]^GI[5]^GI[4]^GI[2]^GI[1]

GO[2]=DI[48]^DI[47]^DI[45]^DI[44]^DI[39]^DI[38]
^DI[36]^DI[34]^DI[32]^DI[29]^DI[28]^DI[27]
^DI[24]^DI[19]^DI[18]^DI[14]^DI[9]^GI[10]^GI[9]
^GI[7]^GI[4]^GI[3]^GI[1]

GO[1]=DI[47]^DI[46]^DI[44]^DI[43]^DI[42]^DI[41]
^DI[40]^DI[38]^DI[37]^DI[34]^DI[33]^DI[29]
^DI[27]^DI[26]^DI[25]^DI[19]^DI[14]^DI[13]
^DI[12]^DI[11]^DI[10]^GI[10]^GI[8]^GI[7]^GI[3]

3 快速并行Hash算法性能分析

本文中的并行Hash算法是一种48位物理地址关键字集合映射到10位Hash表地址集合的算法。Hash表地址集合是物理地址关键字集合的压缩映射，这就不可避免地会产生冲突。均匀Hash函数(uniform Hash)是一种可以将关键字集合中的任何一个关键字等概率地映射到地址空间内任何一个地址位的Hash函数。均匀Hash函数通过将Hash地址均匀分布在整个地址空间内，可以有效地减少冲突，因此可以将均匀Hash函数作为评价一个Hash函数性能的标准[13]。通过分析和优化Hash函数使得映射得到的Hash地址在其整个地址空间内均匀分布，可以得到性能优异的Hash函数。设K为关键字集合中的关键字个数，B为Hash表的深度，M为关键字集合通过Hash函数映射得到的地址个数，那么均匀Hash函数的冲突率E_M可以表示为[13]：

$$E_M = C/K \tag{8}$$

式中，$C = K - M$，$M = B - B(1-1/B)^K$。由此可见，K越大于B，冲突就越明显。

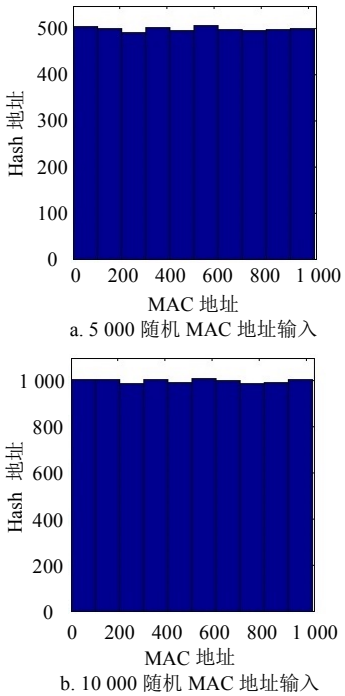

图2 快速并行Hash算法生成Hash地址的柱状分布图

如图2所示为随机输入48位物理地址关键字集合中的5 000个关键字和10 000个关键字时，快速并行Hash算法生成Hash地址的柱状分布图。通过该柱状分布图可以看出，该快速并行Hash算法生成的Hash地址较为均匀地分布在其10位地址空间内，这

将有效地降低Hash冲突发生的几率，提高Hash查找表的性能，因此该快速并行Hash算法是一种性能优异的地址表查询算法。

4 结　论

本文通过分析交换芯片地址表以及Hash算法的特点设计了一种紧凑且易于实现的10位并行循环冗余Hash算法，该算法可以并行地将输入信号帧的48位物理地址转换为10位的地址表查询地址，从而快速准确地查询1 024存储深度的地址表。通过对数据分析可知，该快速并行Hash算法生成的Hash地址均匀地分布在其地址空间内，这有效地降低了Hash冲突发生的几率，提高了Hash查找表的性能，进而保证了整个交换芯片的性能。实验证明，采用该快速并行Hash算法设计实现的二层交换芯片实现了线速交换。

参 考 文 献

[1] SEIFERT R, EDWARDS J. The all-new switch book: the complete guide to lan switching technology[M]. Indianapolis: Wiley Publishing, 2008.

[2] PAPAEFSTATHIOU V, PAPAEFSTATHIOU I. A hardware-engine for layer-2 classification in low-storage, ultrahigh bandwidth environments[C]//Proceedings of the Conference on Design, Automation and Test. Belgium: European Design and Automation Association, 2006: 112-117.

[3] KUMAR S, NOOR A, KAUSHIK B K, et al. Design of ternary content addressable memory (TCAM) with 180 nm [C]//International Conference on Devices and Communications. Mesra: IEEE, 2011: 1-5.

[4] 万成威, 邬江兴, 李玉峰, 等. CAM辅助的哈希表查找性能分析[J]. 电子信息学报, 2011, 33(2): 272-277.
WAN Cheng-wei, WU Jiang-xing, LI Yu-feng, et al. Lookup of CAM Aided Hash table[J]. Journal of Electronics & Information Technology, 2011, 33(2): 272-277.

[5] JAIN R. A comparison of hashing schemes for address lookup in computer networks[J]. IEEE Transactions on Communications, 1992, 40(10): 1570-1573.

[6] MARTINEZ C, LIN W M, PATEL P. Optimal XoR hashing for a linearly distributed address lookup in computer networks[C]//Symposium on Architecture for Networking and Communications Systems. Princeton: ACM, 2005: 203-210.

[7] HUNTLEY C, ANTONOVA G , GUINAND P. Effect of hash collisions on the performance of lan switching devices and networks[C]//IEEE Conference on Local Computer Networks. Tampa: IEEE, 2006: 280-284.

[8] VANDIERENDONCK H, BOSSCHERE K D. XoR-based Hash functions[J]. IEEE Transactions on Computers, 2005, 54(7): 800-812.

[9] FLETCHER J. An arithmetic checksum for serial transmissions[J]. IEEE Transactions on Communications, 1982, 30(1): 247-252.

[10] MATHUKIYA H, PATEL N. A novel approach for parallel crc generation for high speed application[C]//IEEE Conference on Communication Systems and Network Technologies. Gujarat: IEEE, 2012: 581-585.

[11] AKAGI′C A, AMANO H. Performance evaluation of multiple lookup tables algorithms for generating crc on an FPGA[C]//IEEE International Symposium on Access Spaces. Yokohama: [s.n.], 2011: 164-169.

[12] KOOPMAN P, CHAKRAVARTY T. Cyclic redundancy code (CRC) polynomial selection for embedded networks [C]//The International Conference on Dependable Systems and Networks. Florence: IEEE, 2004:1-10.

[13] 李之棠, 程鹏. 一种评价Hash编码策略性能的方法[J]. 计算机应用研究, 1996, 13(5): 31-32.
LI Zhi-tang, CHENG Peng. An evaluation method of Hash coding strategy performance[J]. Application Research of Computers, 1996, 13(5): 31-32.

编　辑　张　俊