多模式匹配算法的 FPGA 实现①

孔利峰，李训根，厉海涛

(杭州电子科技大学 微电子 CAD 研究所，杭州 310018)

摘　要：针对目前模式匹配算法多采用软件实现，而软件实现效率低下的弊端，提出了一种基于硬件实现模式匹配算法的设计方案。综合 Aho-Corasick(AC)算法原理和 FPGA 硬件特点，在 FPGA 上实现 AC 算法；然后利用 Quartus II 对设计进行了验证和性能分析。实验结果表明，基于硬件实现的 Aho-Corasick(AC)算法的效率得到显著提升，有效解决了数据快速增长带来的处理速度缓慢的缺点。

关键词：模式匹配算法；现场可编程门阵列；硬件描述语言

Implementation of the Pattern Matching Technical on FPGA

KONG Li-Feng, LI Xun-Gen, LI Hai-Tao

(Institute of ICAD, Hangzhou Diangzi University, Hangzhou 310018, China)

Abstract: The pattern matching algorithm was mostly implemented based on software, but software realized of low efficiency, a design was applied to make the pattern matching algorithm implementation on the hardware. This paper combines the principle of the Aho-Corasick (AC) algorithm and the characteristics of FPGA, implements the Aho-Corasick (AC) algorithm on the FPGA. Then, it uses the Quartus II to validation and performance analysis for this design. The test results indicate that the design has high quality and is an effective solution to the fault of slow speed.

Keywords: pattern matching algorithm; FPGA; hardware description language

1　引 言

随着 Internet 数据量爆炸式的增长，基于软件实现的入侵检测系统 IDS 的检查效率已经远远赶不上数据的增长[1,2]，成为入侵检测技术发展的瓶颈。基于硬件实现的模式匹配算法，其匹配时间仅取决于待匹配的串长度和硬件的工作速度，可以达到很高的检测速度，从而满足高速网络的实时检测需求[3]。FPGA 具有可重复烧写的功能，是目前硬件实现入侵检测的主流解决方案。本文针对入侵检测系统中的多模式匹配算法-AC 算法，结合 Altera 公司的 FPGA 产品，在硬件电路上实现 AC 算法，从而大大提高系统检测效率。

2　AC算法

AC 算法是基于有穷状态自动机的，在进行匹配之前先对模式串集合 SP 进行预处理，形成树型有穷状态自动机，然后只需对文本串 T 扫描一次就可以找出与其匹配的所有模式串。

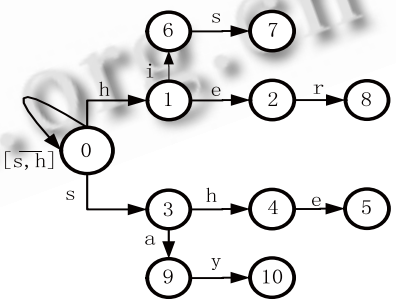

图 1　{he,she,his,her,say}的 goto 函数图

预处理生成 3 个函数：goto(转移)函数，failure(失效)函数和 output(输出)函数。设 U={0，1，2…}为状态集合，转移函数的建立过程为：逐个取出 SP 中模式串的每个字符，从状态 0 出发，如果有标注该字符的矢线存在，则将矢线所指的状态赋为当前状态；否则，

① 基金项目:浙江省重大科技专项(2007C11069)
收稿日期:2010-06-23;收到修改稿时间:2010-08-02

添加一个标号比已有状态标号大 1 的新状态，并用一条矢线从当前状态指向新加入的状态，并将新加入的状态赋为当前状态。最后，画一条从 0 状态到 0 状态的自返线。SP={ he,she,his,her,say }的 goto 函数如图 1 所示。

失效函数 f 构造方法为：所有第一层状态的失效函数为 0；对于非第一层状态 s，若其父状态为 r（即存在字符 a,使 g(r,a)=s），则 f(s)=g(f(s*),a)，其中状态 s*为追溯 s 的祖先状态得到的第一个使 g(f(s*),a)存在的状态。如 f(4)=g(f(3),h)=g(0,h)=1，f(5)=g(f(4),e)=g(1,e)=2。具体的失效函数如图 2(a)所示。

输出函数 output 的构造分两步，第一步是在构造转移函数 g 时，每处理完一个模式串，则将该模式串加入到当前状态 s 的输出函数中，如 output(2)={he},output(5)={she}。第二步是构造失效函数 f 时，若 f(s)=s^,则 output(s)=output(s)∪output(s^),如 output(5)=output(5)∪output(2)={she,he}。具体的输出函数如下图 2(b)所示:

s	f(s)
1	0
2	0
3	0
4	1
5	2
6	0
7	3
8	0
9	0
10	0

(a) 失效函数图

S	Output (S)
2	[“he”]
5	[“she,he”]
7	[“his”]
8	[“her”]
10	[“say”]

(b) 输出函数图

图 2 失效函数图与输出函数图

AC 算法实现如下：从状态 0 出发，每取出文本串中的一个字符，利用 g 和 f 函数进入下一状态。当某个状态的 output 函数不为空时输出其值，表示在文本串中找到该模式串。

AC 算法模式匹配的时间复杂度是 O(n)，与模式集中模式串的个数和模式串的长度无关。

3 硬件实现

根据 AC 算法的原理，利用 verilog hdl 语言构建 FSA，在硬件上设计和实现 AC 算法的功能。采用可编程逻辑器件 FPGA 来实现算法，系统可靠性高，可维护性好，使用灵活。

3.1 设计思路

AC 算法的核心就是 FSM(有限状态机)的建立，所以硬件设计的核心问题就是如何把状态机固化到 FPGA 的片内 RAM 当中。由于片内 RAM 是可寻址的，也就可以把每个状态都划分到片内 RAM 中特定的地址上。一个状态占据了片内 RAM 中 256 个单元的地址，例如状态 0，它的地址就占据了标号为 0 到 255 的片内 ram 空间。在每个地址中都可以存储相应的数据，包括下一状态的地址(对应 goto 函数)，是否有匹配项成功(对应 output 函数)，如果失败要转到哪个状态(对应 failure 函数)等。通过控制逻辑来产生相应的地址和数据的匹配，不占用 CPU 的处理时间，大大的提高了数据的处理速度[4-6]。

数据匹配的问题，也是应用 RAM 来实现的，每个 FPGA 芯片，都可以搭载外部 RAM。因为 AC 算法是字符串的匹配方案，所以所有的字符都可以转化为 ASCII 码值来进行表示。片内 RAM 中每个状态的地址都是有编号的，每个状态都占据了 256 个单元。基于这一点，可以对所要处理的字符串先进行转化，转变成 0，1 表示的字符串。对于 0 状态而言，只需要根据输入字符的 ASCII 码值来对应 ram 中地址上的数据，并进行匹配；对于不是 0 状态的状态，只需要在输入数据的 ASCII 码值所对应的地址值加上对应的状态数×256 就可以进行寻址了。对于具体的控制，需要编写控制逻辑来实现。

最后实现的原理图，如图 3 所示:

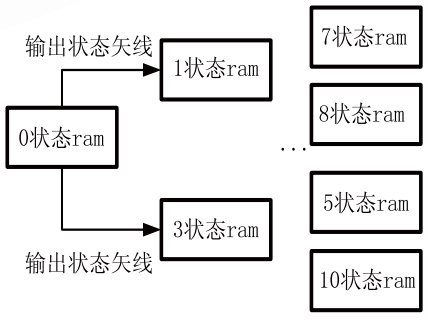

图 3 设计原理图

3.2 硬件实现步骤

通过对 AC 算法的深入研究，并根据 FPGA 的特点，提出了原理框图，具体如图 4 所示。

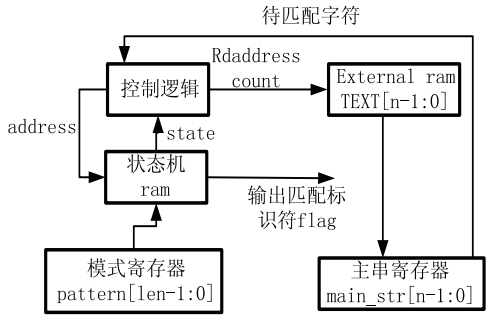

图 4　原理框图

由图 4 可以看出，整个电路结构主要由 3 部分构成：状态数目和跳转状态的计算，文本匹配和控制逻辑。进行文本匹配时，主串可以先放在寄存器中，或者外部 RAM 中，然后由控制逻辑来读取主串中的字符。模式串由 FPGA 来控制生成状态机，用于模式的识别。对于控制逻辑，要实现的功能有两点：一是控制地址计数器，供主串读取数据；二是控制设计中的有限状态机的计数器和状态转换，保证有限状态机能够正确工作。

3.3 设计实例分析

根据上述设计思想，采用 verilog hdl 语言设计实现图 4 中的模式匹配电路，开发环境为 Altera 公司的 Quartus II8.1，目标板为 Altera 的 cyclone 系列 EP1C6Q240C8 型 FPGA，由 Quartus II8.1 对设计实例进行了前仿真、逻辑综合、功能验证和时序验证，并在目标板上进行了功能性验证，证明了方法的可行性。

(1) 用 Quartus II 自带的波形仿真器进行波形仿真，输入的 Pattern(模式)为{“BOY”,“GIRAFFE”}，待匹配的主串为{BBBOYGIRLBOY}，从主串中寻找 Pattern 可能出现的情况。大约 6 个时钟周期以后得到输出，输出标示符为 flag，包括初始化的将近 2 个时钟周期。当 flag 输出不为 0 时，表明匹配成功。抓取前仿真的波形图如图 5 所示。

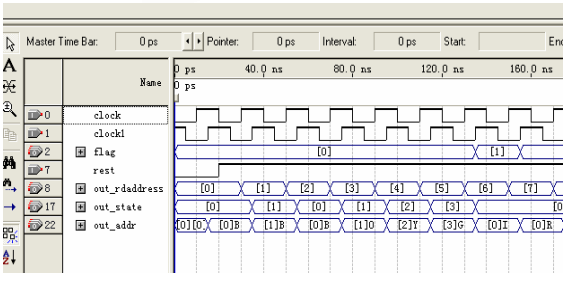

图 5　Quartus II 前仿真的波形图

(2) 经 Quartus II 布局布线和综合仿真后，进行时序验证，也得到正确的结果，如图 6 所示。

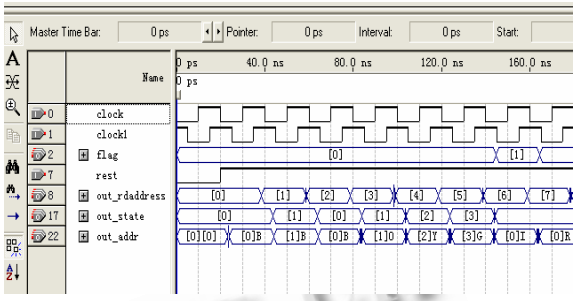

图 6　Quartus II 后仿真的波形图

3.4 性能总结

由于 AC 算法的时间复杂度是 O(n)，而且与模式集中模式串的个数和每个模式串的长度无关。即无论模式串 SP 是否出现在 T 中，T 中的每个字符都必须输入状态机中，所以无论是最好情况还是最坏情况，AC 算法模式匹配的时间复杂度都是 O(n)。这样，对于模式串超多的模式匹配过程，有很好的检测效率，也就很好的解决了规则集数目呈指数形式发展所带来的影响。由于 AC 算法出色的性能，在短时期内不会处在淘汰的边缘。

有以下性能分析，在相同时钟频率，相同模式串长度下，随着模式串长度和个数的增加，比较该模块匹配的时间有何变化。经过时序仿真验证，得出的数据如表 1 所示。

表 1　不同主串长度下的性能分析

主串/bit	模式串/bit	时钟周期/ns	时间/ms
32’hFFFF	8’hFFFF	20	1.28
32’hFFFF	16’hFFFF	20	1.28
32’hEE8939	16’hFFFF	20	312

由表 1 看出，在相同的主串长度下，AC 算法的匹配时间并不随着模式串的改变而改变。

与 BM 算法和 KMP 算法进行比较，选取 2.46M 的英文文本，在其中查找多个模式串，找出所有出现的位置．从文本中选取不同长度的模式串，测试在相同模式集和模式长度下算法的性能．测试机使用 P4 电脑，CPU 为 1.8 GHz，内存 1．O G，操作系统为 windows XP，系统工作频率为 50HZ，实验结果如表 2 所示：

表 2 不同算法的比较结果

模式串长度	AC/s	BM/s	KMP/s
80	0.123	2.1	9.84
800	0.123	6.9	98.4
8000	0.123	35.5	984

从表 2 中可以看出，硬件化的 AC 算法于其他算法有着非常明显的优势，并且随着模式串长度的增加，优势更加明显。

4 结束语

本文设计了一个模式匹配的硬件模块，主要应用在网络数据当中的字符串匹配问题上。从性能上看，在 75MHz 时钟频率下仿真，一个时钟就可以匹配一个字符，输出结果完全由主串当中出现的 Pattern 位置所决定。对于后续的工作，可以再进行代码的优化和硬件产品的升级来提高匹配性能。如把本文中应用的 cyclone 系列的 FPGA 换成 cyclone III 系列的 FPGA，再进行布局布线的优化，估计可以很大程度的提高系统性能。

参考文献

1 邰晓英.信息检索技术导论.北京:科学出版社,2006.40－60.
2 晓妍,戴冠中,杨黎斌.改进的多模式字符串匹配算法.计算机应用,2007,27(6):1415－1417.
3 黄栋,余综.模式匹配算法在 FPGA 芯片上的设计与实现.计算机工程与设计,2006,27(17):3273－3276.
4 夏宇闻.Verilog 数字系统设计教程.北京:北京航空航天大学出版社,2003:89－95.
5 Yu F, Chen Z. Fast and memory efficient regular expression matching for deep packet inspection. Tech Rep: UCB/EECS -2006-76. University of California Berkeley, 2006:93－102.
6 郭军,笹尾勤.入侵检测中模式匹配算法的 FPGA 实现.系统仿真学报,2007,19(14):3215－3229.