

基于二进制搜索算法的 RFID 系统防碰撞算法

邓 洁, 程良伦

(广东工业大学 自动化学院, 广东 广州 510006)

摘要: 为了有效解决二进制搜索算法标签识别延时过长的问题, 提出了二进制搜索算法的一种改进算法, 通过设置堆栈, 变量 Count 引入了反向搜索的机制. 仿真试验结果表明该改进算法显著地减少了读写器向标签发送的请求次数, 解决了二进制搜索算法延时过长的问題, 提高了标签识别的效率.

关键词: 射频识别; 防碰撞算法; 二进制搜索算法; 堆栈; 反向搜索

中图分类号: TN92

文献标识码: A

文章编号: 1007-7162(2009)03-0072-05

射频识别(Radio Frequency Identification, RFID)是一种非接触式的自动识别技术, 它通过射频信号自动识别目标对象并获取相关数据. 典型的 RFID 系统主要包括两部分: 读写器(Reader)和标签(Tag).

射频识别系统工作时, 在读写器的作用范围内, 可能会有多个应答器(标签)存在, 这些应答器的数据同时传送到读写器时出现碰撞即数据碰撞, 导致读写器无法读出数据, 这就是射频识别系统中的多路存取的碰撞问题. 现阶段的 RFID 系统应用中, 应用最为广泛的是时分多路(TDMA)法. 基于时分多路法(TDMA)^[2]的防碰撞算法又分为: 基于比特的二进制搜索算法和基于时隙的 ALOHA 算法^[3].

如何快速准确地识别标签是 RFID 防碰撞算法要解决的主要问题. 影响二进制搜索算法识别标签时间的因素主要有两个^[4]: 一个是读写器向标签发送请求的总次数, 另一个是标签识别过程中, 标签和读写器之间所传送的数据量. 针对基于比特的二进制搜索算法识别所有标签的延时较长的问题, 本文提出一种改进算法, 主要是通过减少读写器向标签发送的请求次数来减少延时, 从而达到提高标签识别效率的目的. 设置堆栈和一个变量 Count 引入了反向搜索的机制, 变量 Count 有两个取值 0 和 1, 当 Count=0 时, 作为开启反向搜索的标志. 堆栈的作用是将每次发生碰撞的比特的最高位入栈, 当反向搜索开启时, 再通过出栈, 重新设定标签筛选条件, 减

少符合筛选条件的标签数目, 从而减少读写器向标签发送请求的次数, 解决二进制搜索算法延时过长的问題.

1 二进制搜索防碰撞算法

实现“二进制搜索”算法的必要前提是能够辨认出读写器中数据碰撞位的准确位置. 为此选择曼彻斯特码作为位的编码法^[5].

二进制搜索法又名二叉树搜索法^[6]. 所有用二进制唯一标识的电子标签的 ID 号可以构成一棵完全二叉树. 在读写器作用范围内同步向读写器发送信号的标签的 ID 号也构成一棵二叉树^[7]. 读写器根据信号碰撞的情况反复对完全二叉树的分枝进行筛选, 最终找出这棵二叉树. 在寻找的过程当中逐一确定了作用区域内响应的标签, 同时也完成了它们与读写器之间的信息交换. 接下来将通过一个实例对二进制搜索算法具体实现过程进行详细的说明. 采用 ID 为 8 位的 4 个标签, 其 ID 分别为 10110010, 10100011, 10110011, 11100011.

1.1 二进制搜索算法的标签识别流程

1) 读写器设置筛选条件 ($ID < 11111111$), 向标签发送请求.

2) 阅读区域内的所有标签均符合筛选条件, 响应读写器的请求, 发送各自的 ID.

3) 读写器检测到第 2、4、8 位发生碰撞, 即 1X1X001X. 读写器将碰撞的最高位置 Q 其余低位

收稿日期: 2009-01-07

基金项目: 国家自然科学基金资助项目(60673132); 广东省自然科学基金重点资助项目(07117421); 广东省科技计划重点资助项目(2005B10101065); 广东省自然科学基金重点资助项目(8251009001000002)

作者简介: 邓 洁(1983-), 女, 硕士研究生, 主要研究方向为射频识别技术、传感器网络、工业控制网络.

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

置 1, 重新设定筛选条件为 ($ID < 10111111$), 向标签发送请求.

4) 标签 10110010, 10100011, 10110011 响应读写器请求, 发送各自的 ID.

5) 读写器检测到第 4、8 位发生碰撞, 即 101X001X, 读写器将碰撞最高位置 Q 其余低位置 1, 重新设定筛选条件为 ($ID < 10101111$), 向标签发送请求.

6) 标签 10101111 响应读写器的请求, 发送 ID 号.

7) 读写器检测到没有碰撞发生, 成功识别出标签 10100011, 选择标签 10100011, 完成对标签的读写.

8) 读写器重新设定筛选条件为 ($ID < 11111111$), 重复识别过程.

二进制搜索算法的标签识别示意图如图 1 所示.

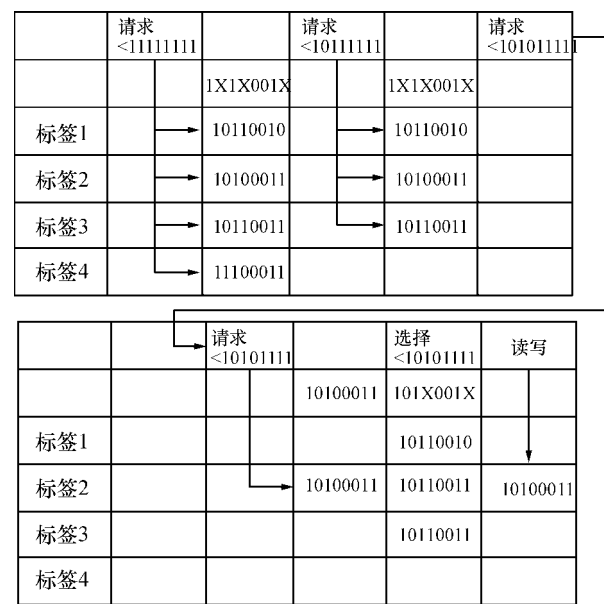


图 1 二进制搜索算法的标签识别示意图

1.2 算法分析

由图 1 可知, 读写器成功识别第一个标签需要发送 3 次请求. 当第一个标签成功识别之后, 又重新设置筛选条件为 $ID < 11111111$, 开始第二个标签的识别过程. 由以上的分析可知, 要实现对所有标签的识别, 读写器总共需要向标签发送的请求次数为 9 次. 显然, 二进制搜索算法并没有采取有效的机制减少符合筛选条件的标签的数目, 只是简单地重复识别第一个标签的过程, 因而也就没有降低标签发生碰撞的可能性, 从而无法减少读写器向标签发送的请求次数.

2 改进算法

传统的二进制搜索算法在每一次成功识别一只标签后, 都会从头开始, 重复识别过程, 并没有采取有效的措施来减少读写器向标签发送的请求次数. 本文提出的算法, 主要是通过设置堆栈和 Count 标志位, 使读写器具有记忆能力, 缩小了符合筛选条件的标签范围, 减少了标签碰撞的可能性, 从而有效地减少了读写器向标签发送请求的次数, 进而减少了读写器识别所有标签的时间.

为了与二进制搜索算法作对比, 这一部分采用和二进制搜索算法相同的标签设置. 仍然采用 ID 为 8 位的 4 个标签, 其 ID 分别为 10110010, 10100011, 10110011, 11100011.

2.1 改进算法的标签识别流程

首先将标志位 Count 置 1, 并设置一个空的堆栈.

- 1)、2) 与传统的二进制搜索法相同.
- 3) 读写器检测到第 2、4、8 位发生碰撞, 即 1X1X001X. 保持 Count=1, 将碰撞的最高位 2 入栈. 按照与传统二进制搜索算法相同的步骤重新设置筛选条件 ($ID < 10111111$), 向标签发送请求.
- 4) 与传统的二进制搜索法相同.
- 5) 读写器检测到第 4、8 位发生碰撞, 即 101X001X, 读写器保持 Count=1, 将碰撞的最高位入栈, 但要注意在 4 入栈前, 应先将栈顶 2 复制一次, 然后再将碰撞最高位 4 入栈. 此时栈的内容为 2、2、4. 按照与传统二进制搜索算法相同的步骤重新设置筛选条件 ($ID < 10101111$), 向标签发送请求.
- 6) 标签 10101111 响应读写器的请求, 发送 ID 号.
- 7) 读写器检测到没有碰撞发生, 成功识别出标签 10100011, 选择标签 10100011, 完成对标签的读写. 将 Count 置 0 然后实现连续两次出栈. 将先出栈的位置 1, 其余低位置 Q 作为筛选条件的下边界, 后出栈的位置 Q 其余低位置 1 作为筛选条件的下边界, 从新设定筛选条件为 ($10110000 ID < 10111111$), 向标签发送请求. 读写器检测到栈不为空, 所以再次将标志位 Count 置 1. 此时栈内容为 2.
- 8) 标签 10110010, 10110011 响应读写器请求, 发送各自的 ID.
- 9) 读写器检测到第 8 位发生碰撞, 同时判断碰

撞位已经是 **D** 的最低位, 则只将 8 入栈, 不用实现第 (5) 中的二次入栈. 同时因为碰撞位是 **D** 的最低位, 所以就将筛选条件的不等号设置为等号, 即 (**D** = 10110010), 向标签发送请求. 此时栈内容为 2 8

10) 读写器检测到没有碰撞发生, 成功识别出标签 10110010 将 **Count**置 **Q** 读写器判断栈顶 8 已经是 **D** 最低位, 所以只进行一次出栈, 将出栈位置 1, 不等号设置为等号, 即 (**D** = 10110011), 向标签发送请求. 读写器检测到栈不为空, 所以再次将标志位 **Count**置 1. 此时栈内容为 2

11) 读写器检测到没有碰撞发生, 成功识别出标签 10110011 将 **Count**置 **Q** 重复 6) 中的出栈过程, 由于此时栈内只有 2 所以将 2 出栈, 设置筛选左边界为 (**D** > 10111111), 向标签发送请求. 读写器检测到栈为空, 所以保持 **Count**= 0

12) 读写器检测到没有碰撞发生, 成功识别出标签 111100011. 读写器检测到栈为空, **Count**= 0 所以成功识别出所有标签.

改进算法的标签识别示意图如图 2 所示. 为了方便与二进制搜索算法对比, 主要给出了改进部分的流程图.

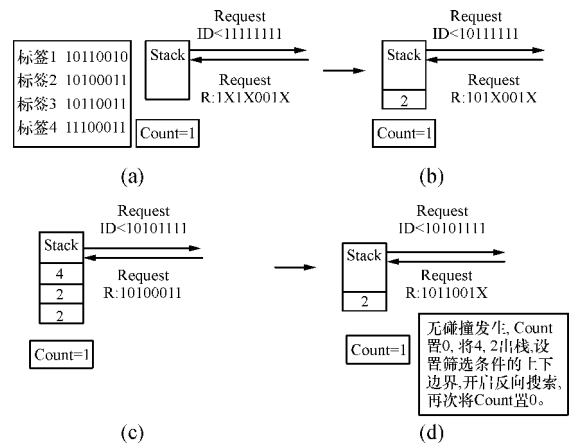


图 2 改进算法标签识别示意图

2 2 算法分析

通过以上分析可知, 通过该算法, 读写器总共需要向标签发生 6 次请求就能完成对阅读区域内所有标签的识别. 相对于二进制搜索算法的改进之处主要由 (c)、(d) 两步体现出来. 也就是在每一次成功识别一个标签后, 并没有像二进制搜索算法一样, 将筛选条件重新设置为 **D** < 11111111 (在这种情况下, 有 3 个标签响应阅读器), 而是将标志位 **Count**置 **Q** 通过连续两次出栈的机制, 设置双边的筛选条件 (符合筛选条件的标签数目为 2), 开启反向搜索.

减少了符合筛选条件的标签数目, 也就减少了标签的碰撞次数, 从而达到了减少读写器向标签发送的请求次数的目的. 通过以上分析可知, 该改进算法有效地减少了读写器向标签发送的请求次数.

2 3 改进算法的规则设定

- 1) 每当成功识别一个标签, 即将 **Count**置 **Q** 开启反向搜索.
- 2) 读写器每次入栈时, 检测栈内容是否为空, 当栈不为空, 并且碰撞位不是 **D** 最低位时, 进行二次入栈.
- 3) 当栈内容不为 **D** 最低位时, 均进行二次出栈.
- 4) 当出栈的内容为 **D** 最低位时, 将筛选条件的不等号改为等号.
- 5) 在进行二次出栈时, 先出栈的位置 1, 其余低位位置 **Q** 设置筛选的下边界, 后出栈的置 **Q** 其余低位位置 1 设置筛选的上边界.

2 4 算法流程图

改进算法的算法流程图如图 3 所示.

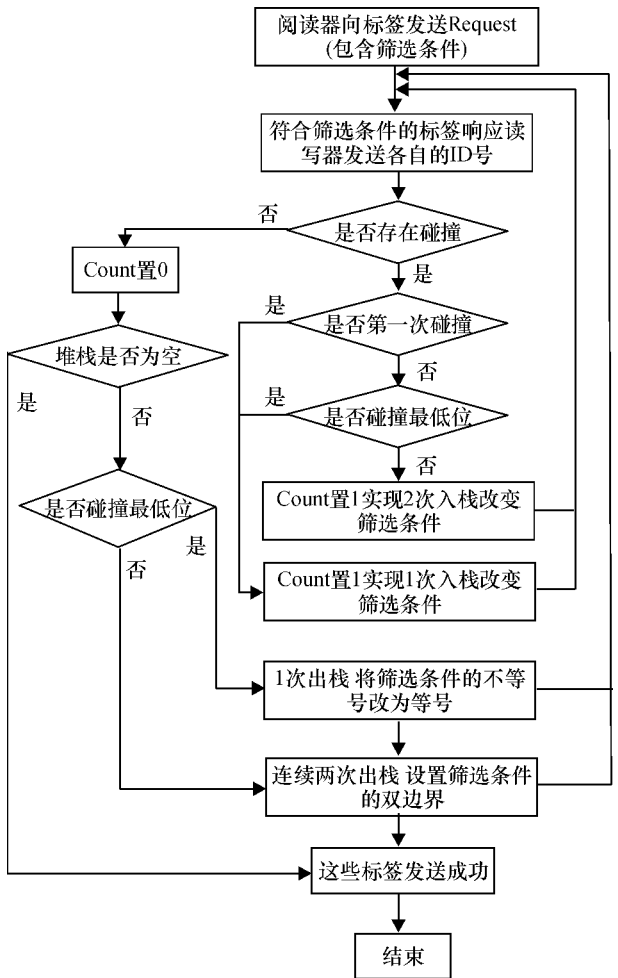


图 3 改进算法流程图

3 仿真结果与分析

通过第 2 部分的实例分析, 可知本文提出的算法可以通过减少读写器向标签发送的请求次数, 从而实现缩短读写器识别阅读区域内所有标签时间的目的. 为了更进一步考察该算法在不同 ID 号长度, 不同标签数目情况下的性能, 本文进行了两组仿真试验. 第 1 组是 ID 号长度等于 8 时, 标签数目从 4 增加到 64 的情况下, 二进制算法和改进算法发送请求次数的对比情况; 第 2 组是 ID 号长度等于 16 时, 标签数目从 4 增加到 64 的情况下, 二进制算法和改进算法发送请求次数的对比情况. 其中图 4 图 6 是随机选取 ID 号的仿真结果, 图 5 图 7 是顺序选取 ID 号的仿真结果. 其中标签数目为 P , 读写器向标签发送的请求次数为 T .

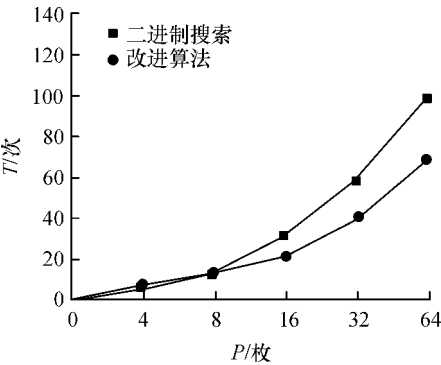


图 4 ID 长度为 8 随机选择 ID 的仿真结果

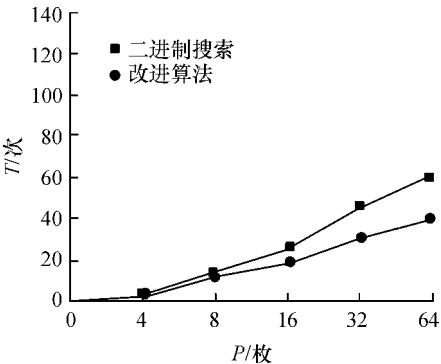


图 5 ID 长度为 8 顺序选择 ID 的仿真结果

由图 4 图 5 可以看出, 在 ID 长度为 8 时, 随着标签数目的增加, 与二进制算法相比, 改进算法显著地减少了读写器向标签发送的请求次数. 以图 4 为例, 当标签数目 $P = 64$ 时, 二进制算法的请求次数为 98 次, 改进算法的请求次数为 70 次, 降低了约 20%.

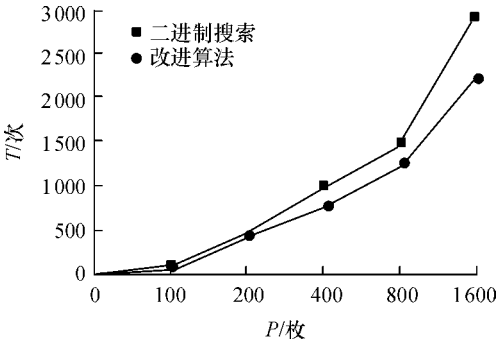


图 6 ID 长度为 16 随机选择 ID 的仿真结果

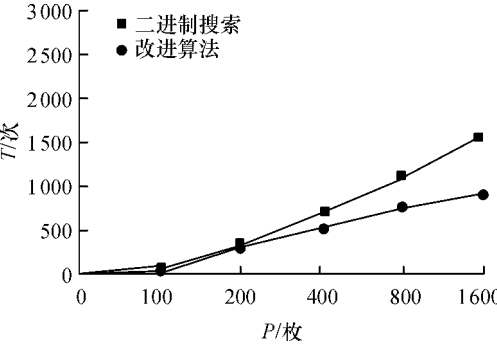


图 7 ID 长度为 16 顺序选择 ID 的仿真结果

由图 6 图 7 可以看出, 在 ID 长度为 16 时, 随着标签数目的增加, 与二进制算法相比, 改进算法显著地减少了读写器向标签发送的请求次数. 以图 7 为例, 当标签数 $P = 1\ 600$ 时, 二进制算法的请求次数为 1 500 次, 改进算法的请求次数为 800 次, 降低了大约 47%.

但是对比图 4 图 5 图 6 图 7 可以看出, 当标签数目较少时, 两种算法的区别并不明显. 原因是, 当标签数目较少时, 读写器并不需要或只需要很少次数的反向搜索就能将所有标签识别出, 因而并没有充分体现本算法所提出的反向搜索算法的优越性.

通过对两组仿真试验结果的分析可知, 改进算法显著地减少了读写器向标签发送的请求次数, 从而有效地解决了二进制搜索算法标签识别延时过长的问

4 结论

如何快速准确地识别阅读区域内的所有标签, 是 RFID 系统的关键问题之一. 本文针对二进制搜索算法延时过长的问

件的标签数目,从而有效地减少了读写器向标签发送请求的次数,进而减少了读写器范围内所有标签的识别时间.仿真结果也表明,该改进算法在各种情况下都比二进制搜索算法性能更为优越.

参考文献:

- [1] Klaus Finkenzeller. RFID Handbook Fundamentals and Applications in Contactless Smart Card and Identification [M]. Second Edition. Germany: John Wiley & Sons, 2002: 132-150.
- [2] 游战清, 李苏剑, 刘克胜, 等. 无线射频识别技术 (RFID) 理论与应用 [M]. 北京: 电子工业出版社, 2004.
- [3] 马剑波. RFID 系统防碰撞算法研究 [D]. 北京: 北京交通大学硕士学位论文, 2006.
- [4] Quan C. Design and Performance Evaluation of High Performance Anti-collision Algorithm in the RFID System [D].

Korea Daegu Univ, 2004.

- [5] Finkenzeller K. RFID Handbook Fundamentals and Applications in Contactless Smart Card and Identification [M]. Second Edition. Germany: John Wiley & Sons Ltd, 2003: 195-219.
- [6] Jimmying Lee W. Adaptive binary splitting A RFID tag collision arbitration protocol for tag identification [J]. ACM / Springer Mobile Networks and Applications, 2005, 2(1): 127-137.
- [7] Liu L, Xie Z, Xi J et al. An Improved Anticollision Algorithm in RFID System, Mobile Technology, Applications and Systems, 2nd International [C]. Beijing: Communications Letter IEEE, 2005.
- [8] Myung J, Lee W. Adaptive binary splitting A RFID tag collision arbitration protocol for tag identification, ACM / Springer Mobile Networks and Applications [J]. ACM MO-NET, 2006, 11(5): 711-722.

An Improved Anti-collision Algorithm Based on Binary Search Algorithm in the RFID System

Deng Jie, Cheng Liang-lin

(Faculty of Automation, Guangdong University of Technology, Guangzhou 510006, China)

Abstract How to quickly and accurately identify the tags in the region of the reader is a key issue of the RFID anti-collision algorithm. In order to efficiently solve the problem that it takes so long a time for the binary search algorithm to identify all the tags in the region of the reader, it proposes an improved anti-collision algorithm, based on the binary search algorithm. By setting a stack and a variable count, it introduces the converse search. Simulation results show that the improved algorithm significantly reduces the number of requests which the reader sends to the tags, that the improved algorithm has solved the long delay problem of the binary algorithm, and that it improves the efficiency of the tag's identification.

Key words Radio Frequency Identification (RFID); anti-collision algorithm; binary search algorithm; stack; converse search