

# RFID 折半回溯动态二进制防碰撞算法的研究

黄俊南,李展宗

(泉州经贸职业技术学院 信息系,福建 泉州 362000)

**摘要:**防碰撞算法是射频识别的关键技术之一,以动态二进制防碰撞算法为基础,提出折半回溯动态二进制防碰撞算法。根据折半搜索算法和回溯算法的综合思想,为改进后算法增加了三条规则:①仅1位碰撞可同时读取两个标签;②多位碰撞,筛选条件转变为:最高碰撞位置0,其他碰撞位置1;③每读取标签成功后将当前筛选条件转变为上一个筛选条件,如继续发生多位碰撞,执行②,否则执行③直至全部标签被识别。并通过算法验证表明,折半回溯动态二进制防碰撞算法较其他二进制算法在性能上有显著提高,且识别过程不用考虑碰撞位连续还是间隔的问题,访问效率更为迅速。

**关键词:**无线射频技术;防碰撞;折半搜索算法;回溯法

中图分类号:TP301.6

文献标识码:A

文章编号:1673-629X(2012)10-0151-06

## Research on RFID Binary Backtracking Dynamic Binary Anti-collision Algorithms

HUANG Jun-nan, LI Zhan-zong

(Department of Information, Quanzhou Economic and Trade Technology School, Quanzhou 362000, China)

**Abstract:** Anti-collision algorithm is one of the key technologies of radio frequency identification. In dynamic binary anti-collision algorithm based, proposed binary backtracking dynamic binary anti-collision algorithm. According to the binary search algorithm and backtracking algorithm's comprehensive thinking, for the improved algorithm add three rules: ① Only 1 collision bit can be read at the same time two labels; ② Many collision bit, screening conditions change; the highest collision bit position 0, position 1 in other collision bit; ③ Each label is read after the success of the current screening conditions into the last screening conditions, if continue to produce many collision bit, executive ②, otherwise executive ③ until all labels are identified. And through the algorithm test shows that, binary backtracking dynamic binary anti-collision algorithm is better than the other binary algorithm in performance is significantly improved, and the recognition process does not need to consider the collision bit is continuous or spacing issues, access efficiency more quickly.

**Key words:** RFID; anti-collision; binary search algorithm; backtracking

## 0 引言

RFID(Radio Frequency Identification)技术是一种非接触式的自动识别技术,通过射频信号自动识别目标并获取相关数据,识别工作无须人工干预,可工作于各种恶劣环境<sup>[1]</sup>。在第二次世界大战期间,英国为了识别返航的飞机发射一个询问信号,飞机上的收发器接收到这个信号后,回传一个信号给探测器,探测器根据接收到的回传信号来识别敌我。这是有记录的第一个RFID敌我识别系统,也是第一个RFID的第一次实际应用<sup>[2]</sup>。

RFID的通讯环境是一种简单的无线系统,只需由读写器和电子标签两个部件组成。其工作原理:当电

子标签进入射频感应区后,接收射频感应信号并通过感应电流获得的能量将已存储的ID发送到读写器上,再经由读写器进行信号解码处理后送至信息系统中进行数据处理。RFID系统则利用现代计算机信息技术手段对采集到的数据进一步进行加工处理,并经由通信网络实现资源共享等实际运用,提高现代企业管理的运作效率。

RFID技术的最大优点在于能同时识别多个目标。当读写器的天线作用范围内有多个电子标签时,面对读写器发出的指令,每个电子标签都会响应,如果它们同时发送信号,信号互相干扰,就会发生信道冲突问题,在RFID技术中这种现象被称为碰撞问题<sup>[3]</sup>。在RFID系统中使用某些策略及算法规避信道冲突的情况称为防碰撞。RFID防碰撞算法主要有2种:一种是基于ALOHA的不确定性算法,另一种是基于二叉树(BT, binary tree)的确定性算法<sup>[4]</sup>。文献[5~9]分别是几种确定性防碰撞算法的改进方式,各种改进均有

收稿日期:2012-03-02;修回日期:2012-06-07

基金项目:泉州市社科重点计划项目(2011H04)

作者简介:黄俊南(1977-),男,福建泉州人,硕士,讲师,研究方向为数据库、数据仓库、系统架构、算法分析等。

利弊。文中主要探讨防碰撞二叉树算法,对二叉树算法进行分析与改进。

1 二进制防碰撞算法

二进制搜索法又名二叉树搜索法<sup>[10]</sup>,主要有二进制搜索算法、动态二进制算法及其他改进后的二进制搜索算法。文中只对二进制防碰撞搜索算法及动态二进制防碰撞搜索算法进行解析,并在动态二进制防碰撞搜索算法的基础上提出一种新的动态二进制防碰撞搜索算法改进方式。

1.1 曼彻斯特(Manchester Encoding)编码

曼彻斯特编码是一个同步时钟编码技术,编码位的值用一个位窗内电平的改变来表示。分别用上升沿和下降沿表示逻辑“0”和“1”。当数据发生碰撞时,传送的碰撞位数据的上升沿和下降沿电平会互相抵消(无电平状态),即可判断该位数据碰撞。该编码方案可以准确确定碰撞位所处位置,很好地满足于二进制搜索算法为基础的 RFID 系统,目前确定性算法主要采用该编码方案<sup>[11]</sup>。

1.2 二进制防碰撞算法原理及实例化分析

二进制防碰撞算法读写器读取标签 ID 的互动方式为:

- ①请求命令 Request (ID):读写器设置筛选条件,向标签发送请求。
- ②选择命令 Select (ID):感应区内的满足条件的标签发送自身 ID。

③读取命令 Read (ID):读写器读取标签 ID 并检测碰撞位,修改筛选条件并回到①开始执行。筛选条件修改规则为:保留左几位未碰撞位。将第一个碰撞位置 0(筛选条件 ID<= 1111111,8 位 ID),其余位置 1;或将第一个碰撞位置 1(筛选条件 ID>= 00000000,8 位 ID),其余位置 0。

④静默命令 Quiet (ID):读写器读取标签 ID 未检测到碰撞位,读取该标签 ID 并将该标签静默。然后重新从①开始对剩余标签的判断,直到全部标签静默。

实例化:对上述互动方式,以 8 位二进制作为标签 ID 号模拟该算法,取 4 个二进制数分别为:11010011、01010011、11100001、01100011。其冲突位分别为 1、3、4 和 7 位,? 号表示碰撞位,如表 1 所示。

1.3 动态二进制防碰撞算法原理及实例化分析

二进制防碰撞算法中每次发送 Request 的时候,均传输完整的标签 ID 号。动态二进制防碰撞算法的改进方式是:通过不传输左部未碰撞编码位减少数据传输量的方式,提高读写器阅读速度。因此在 Request 的时候应增加一个起始标志位,文中使用 idstart 表示(idstart 为左部第一个碰撞位)。

动态二进制防碰撞算法读写器读取标签 ID 的互动方式为(在二进制防碰撞算法上改进):

- ①请求命令 Request (ID,idstart):读写器设置筛选条件,向标签发送请求。筛选条件为 ID 小于等于由 idstart 位~0 位各位置 1。
- ②选择命令 Select (ID,idstart):感应区内的满足

表 1 二进制搜索算法流程表

|                         |             |                  |                  |                  |                  |
|-------------------------|-------------|------------------|------------------|------------------|------------------|
| Request (ID)            | Read (ID)   | 标签 1<br>11010011 | 标签 2<br>01010011 | 标签 3<br>11100001 | 标签 4<br>01100011 |
| Request (ID<= 11111111) |             | 11010011         | 01010011         | 11100001         | 01100011         |
|                         | ? 1?? 00? 1 |                  |                  |                  |                  |
| Request (ID<= 01111111) |             |                  | 01010011         |                  | 01100011         |
|                         | 01?? 0011   |                  |                  |                  |                  |
| Request (ID<= 01011111) |             |                  | 01010011         |                  |                  |
| Select (标签 2)           |             |                  | Quiet            |                  |                  |
| Request (ID<= 11111111) |             | 11010011         |                  | 11100001         | 01100011         |
|                         | ? 1?? 00? 1 |                  |                  |                  |                  |
| Request (ID<= 01111111) |             |                  |                  |                  | 01100011         |
| Select (标签 4)           |             |                  |                  |                  | Quiet            |
| Request (ID<= 11111111) |             | 11010011         |                  | 11100001         |                  |
|                         | 11?? 00? 1  |                  |                  |                  |                  |
| Request (ID<= 11011111) |             | 11010011         |                  |                  |                  |
| Select (标签 1)           |             | Quiet            |                  |                  |                  |
| Request (ID<= 11111111) |             |                  |                  | 11100001         |                  |
| Select (标签 3)           |             |                  |                  | Quiet            |                  |

条件的标签发送自身 ID。

③读取命令 Read(ID,idstart):读写器读取标签从 idstart 位到 0 位的 ID 值并检测剩余二进制数据的碰撞位,修改筛选条件并回到①开始执行。

④静默命令 Quiet(ID):读写器读取标签 ID 未检测到碰撞位,读取该标签 ID 并将该标签静默。然后重新从①开始对剩余标签的判断,直到全部标签静默。

实例化:引用与 1.1 中同样的数据进行算法演示,? 号表示碰撞位,如表 2 所示。

对表 1 和表 2 进行分析,可见动态二进制防碰撞算法和二进制防碰撞算法发送 Request 命令次数一样,但动态二进制防碰撞算法在一定程度上降低数据传输量。当标签 ID 达到 10 个字节以上,且当连续碰撞位更多的时候明显地降低了数据传输量,以此减少传输冗余数据以提高标签识别率。

## 2 动态二进制防碰撞算法改进——折半动态二进制防碰撞算法

通过对以上的两种二进制防碰撞算法分析,在初始筛选条件为  $ID \leq 11 \cdots 11$  (共  $n$  个 1, $n$  代表二进制位数) 的情况下,算法呈现如下两种特征:特征一:标签 ID 读取顺序为从小到大;特征二:每读取一个标签成功,筛选条件就恢复为初始筛选条件。以这种状况读取下一个标签往往要发送 Request 命令至少  $N-i$  次( $N$  代表感应区内的标签数, $i$  代表已经读取的标签数)。

### 2.1 折半查找算法搜索二进制数

折半查找<sup>[12]</sup>的算法思想:将数列按有序化(递增或递减)排列,查找过程中采用跳跃式方式查找,即先以有序数列的中点位置为比较对象,如果要找的元素值小于该中点元素,则将待查序列缩小为左半部分,否则为右半部分。假设有 6 个标签 ID,二进制编码集合  $B = \{00100110, 01101110, 01111110, 10110110, 11101110, 11111110\}$ 。以折半查找算法读取标签 ID,查找过程流程如图 1 所示:

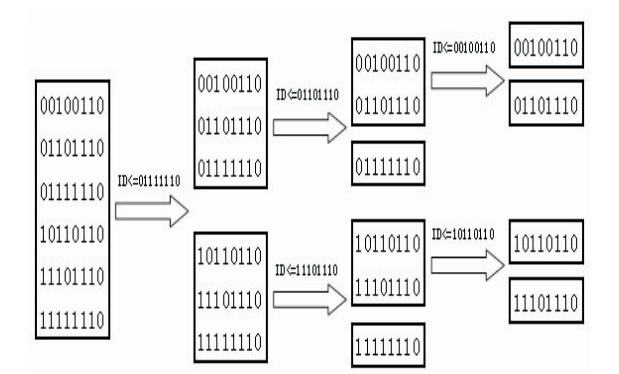


图 1 折半查找读取标签 ID 图

根据特征一,集合  $B$  读取数据过程由小到大应为:  $00100110 \rightarrow 01101110 \rightarrow 01111110 \rightarrow 10110110 \rightarrow 11101110 \rightarrow 11111110$ , 初始筛选的条件应为  $ID \leq 11111111$ 。

根据图 1 所示,读取标签 ID 为 00100110 的二进制数,筛选条件的变化为  $ID \leq 01111110 \rightarrow ID \leq 01101110 \rightarrow ID \leq 00100110$ 。折半算法中是以有序

表 2 动态二进制搜索算法流程表

| Request(ID)                   | Read(ID)    | 标签 1     | 标签 2     | 标签 3     | 标签 4     |
|-------------------------------|-------------|----------|----------|----------|----------|
|                               |             | 11010011 | 01010011 | 11100001 | 01100011 |
| Request(NULL, ID<=11111111,8) |             | 11010011 | 01010011 | 11100001 | 01100011 |
|                               | ? 1?? 00? 1 |          |          |          |          |
| Request(0, ID<=1111111,7)     |             |          | 1010011  |          | 1100011  |
|                               | ?? 0011     |          |          |          |          |
| Request(010, ID<=11111,5)     |             |          | 10011    |          |          |
| Select(标签 2)                  |             |          | Quiet    |          |          |
| Request(NULL, ID<=11111111,8) |             | 11010011 |          | 11100001 | 01100011 |
|                               | ? 1?? 00? 1 |          |          |          |          |
| Request(0, ID<=1111111,7)     |             |          |          |          | 1100011  |
| Select(标签 4)                  |             |          |          |          | Quiet    |
| Request(ID<=NULL,11111111,8)  |             | 11010011 |          | 11100001 |          |
|                               | 11?? 00? 1  |          |          |          |          |
| Request(110, ID<=11111,5)     |             | 10011    |          |          |          |
| Select(标签 1)                  |             | Quiet    |          |          |          |
| Request(NULL, ID<=11111111,8) |             |          |          | 11100001 |          |
| Select(标签 3)                  |             |          |          | Quiet    |          |

数列的中点位置作为比较对象,易于获取,而在 RFID 读写过程中数据上传碰撞位会导致数据位缺失,因此无法直接确定比较二进制数对象。

确定比较二进制数对象:由于标签 ID 为二进制数,特性是每个位非 0 即 1,如:8 位二进制数可模拟数值范围为 00000000 ~ 11111111 (0 ~ 255),01111111 为其中点值,假设碰撞位在第 7 位,直接将第 7 位置 0,后几位位置 1,这种状况很好地体现了折半思想。将这种特性运用在最高碰撞位上,沿用规则为:将最高碰撞位置 0,最高碰撞位后各位置 1。分析读取标签 00100110,筛选条件原始变化情况为:ID <= 01111110 → ID <= 01101110 → ID <= 00100110。以此分组数据(下表仅以独立分组为依据,并非完整搜索过程具体情况),对原有筛选条件进行转变的方式为:ID <= 01111111 → ID <= 00111111 → ID <= 00111111。

## 2.2 回溯法改进动态二进制防碰撞算法

根据特征二,动态二进制防碰撞算法每读取一个标签 ID 后,筛选条件恢复为初始筛选条件。这种搜索方式 Request 次数过多,导致识别速度缓慢。因此,当成功读取一个标签 ID 后,运用合理的算法转变筛选条件能让读写器直接读取下一区域的二进制数据。根据 2.1 中的分组筛选条件转变方式可见数据在每次筛选后分组数据量逐渐减少。文中使用回溯法<sup>[12]</sup>提高标签搜索速度,流程如下:

根据 2.1 中规则:“最高碰撞位置 0,最高碰撞位后各位置 1”。对集合 B 进行搜索,读取 00100110 的过程为:设置初始筛选条件 ID <= 11111111,全部 6 个标签响应请求,上传信息为?? 1?? 110;筛选条件转变为 ID <= 01111110,3 个标签响应请求分别为 00100110、01101110 和 01111110,上传信息为 0? 1?? 110;筛选条件转变为 ID <= 00111110,标签 00100110 响应请求,上传其 ID。筛选条件转变为 ID <= 11111111 → ID <= 01111110 → ID <= 00111110。

快速读取其他标签的方式:在读取标签 00100110 后,如果将筛选条件转变为 ID <= 01111110,可直接识别 00100110 和 01101110 这两个标签。此时,上传数据 0? 10? 110,根据 2.1 中规则将筛选条件转变为 ID <= 00101110,识别 00100110 标签,后再将筛选条件转变为 ID <= 01111110,识别 01111110。这种识别模式符合回溯法的运算模式,将其运用在动态二进制搜索中能更好地提高识别速度。

## 2.3 折半回溯动态二进制防碰撞算法

将上述算法的改进机制总结为:改进算法仍然以动态二进制搜索算法原理为根基,同样满足原有的 4 种命令:Request, Select, Read, Quiet。主要通过减少发送 Request 命令的次数来提升算法速度。假设 Request

命令为:Request (Idchar, IDcondition, idstart), Idchar 表示不读取的 id 二进制位, IDcondition 表示筛选条件, idstart 表示最高碰撞位。

折半回溯动态二进制防碰撞算法在原有的动态二进制防碰撞算法的基础上增加如下三条规则:

规则 1:由于二进制位上的取值具有互相排斥特性(非 1 即 0),所以假如只有 1 位二进制位发生碰撞,则读写器不需要发送请求命令而能够识别出这两个标签<sup>[1]</sup>。

规则 2:当多位发生碰撞时,筛选条件规则为:未碰撞位保留原始数据,将碰撞位最高位置 0,其余碰撞位置 1,发送 Request 命令,按动态二进制规则排除标签至读取标签。

规则 3:读取标签后使用回溯法,将筛选条件转变为上一个筛选条件。如多位发生碰撞,执行规则 2,否则执行规则 3,直至全部标签被识别。

## 2.4 算法模拟演示

实例化:引用与 1.1 中同样的数据进行算法演示,? 号表示碰撞位,如表 3 所示。

从表 3 中可见改进的算法发送 Request 命令 7 次;数据总传输量为 143 位数据。由此可见改进后的算法比动态二进制搜索算法减少了 1 次 Request 命令并更好地降低了数据传输量,数据传输量明显降低 14.88%。以上,更进一步提升了标签的识别率。

## 2.5 算法性能分析

根据 2.3 中新增三个规则,将读写器读取标签 ID 时发送 Request 请求次数设为  $R(N)$ ,  $N$  表示标签数量。可得:

$$N - 1 \leq R(N) \leq 2N - 1 \quad (1)$$

证明(1):

1) 当  $N = 1$  时,读写器工作区域仅一个标签,可直接读取标签 ID,因为  $R(1) = 1$ ,满足(1)的情况。

2) 当  $N = 2$  时,读写器工作区域有两个标签,根据标签 ID 的碰撞位情况可分为如下两种进行考虑:

a) 第一种情况:标签仅一个碰撞位,根据 2.3 规则一,该碰撞位非 0 即 1,以 ID <= 11111111 位初始筛选条件可以直接读取这两个标签,故:  $R(2) = 1$ ,满足(1)的情况。

b) 第二种情况:标签有多个碰撞位,首先设置 ID <= 11111111 为初始筛选条件,2 个标签上传数据发生位碰撞,根据 2.3 规则二,高碰撞位置 0,低碰撞位置 1,转变筛选条件;读取 2 个标签中最小值;然后根据 2.3 规则三,改变筛选条件位 ID <= 11111111 读取另一个标签 ID,总请求数 3 次。故:  $R(2) = 3$ ,满足(1)情况。

3) 验证  $R(N) \leq 2N - 1 (N \geq 3)$ 。当  $R(N) = 2N - 1$  时表示读写器读取标签发送的 Request 次数最多,



表 3 折半回溯动态二进制防碰撞算法标签识别流程表

| Request (ID)                      | Read (ID)   | 标签 1<br>11010011 | 标签 2<br>01010011 | 标签 3<br>11100001 | 标签 4<br>01100011 |
|-----------------------------------|-------------|------------------|------------------|------------------|------------------|
| Request ( NULL, ID<= 11111111, 8) |             | 11010011         | 01010011         | 11100001         | 01100011         |
| 碰撞位为: 7、5、4、1                     | ? 1?? 00? 1 |                  |                  |                  |                  |
| Request ( 0, ID<= 1110011, 7)     |             |                  | 1010011          |                  | 1100011          |
| 碰撞位为: 5、4                         | 1?? 0011    |                  |                  |                  |                  |
| Request ( 010, ID<= 10011, 5)     |             |                  | 10011            |                  |                  |
| Select ( 标签 2)                    |             |                  | Quiet            |                  |                  |
| Request ( 0, ID<= 1110011, 7)     |             |                  |                  |                  | 1100011          |
| Select ( 标签 4)                    |             |                  |                  |                  | Quiet            |
| Request ( NULL, ID<= 11111111, 8) |             | 11010011         |                  | 11100001         |                  |
| 碰撞位为: 5、4、1                       | 11?? 00? 1  |                  |                  |                  |                  |
| Request ( 110, ID<= 010011, 5)    |             | 10011            |                  |                  |                  |
| Select ( 标签 1)                    |             | Quiet            |                  |                  |                  |
| Request ( NULL, ID<= 11111111, 8) |             |                  |                  | 11100001         |                  |
| Select ( 标签 3)                    |             |                  |                  | Quiet            |                  |

这种情况的实现应满足以下 2 种条件:

a) 标签中不能存在 2 个标签仅一位二进制位碰撞的情况(即不遵循 2.3 规则一)。

b) 有  $N$  个标签,且可使用折半算法(筛选条件)将标签均分为 2 半,直至最后各组标签均为 2 个,与满二叉树状况相似。由此可见: $2^C = N$ ,  $C$  表示筛选条件深度(满二叉树叶子女数状况,其余节点为筛选条件)。

根据上述描述情况,以  $C = 1$  到  $C = 4$  为例,假设有标签  $A_1, A_2, A_3, \dots, A_{16}$  (由小至大)分别模拟  $R(2^1) \sim R(2^4)$  (即:  $R(2) \sim R(16)$ ) 标签的识别次数情况,如表 4 所示。

根据表 4,可见当  $C = 1$  到  $C = 4$  的满二叉树情况时,识别次数的情况为:

$R(2) = R(2^1) = 2 + 1 = 3; R(4) = R(2^2) = (3 + 1) + (2 + 1) = 7; R(8) = R(2^3) = ((4 + 1) + (2 + 1)) + ((3 + 1) + (2 + 1)) = 15; R(16) = R(2^4) = (((5 + 1) + (2 + 1)) + ((3 + 1) + (2 + 1))) + (((4 + 1) +$

$(2 + 1)) + ((3 + 1) + (2 + 1))) = 31$ 。据此推导,可得:

$R(2^C) = 2R(2^{C-1}) + 1$  (2)

$R(N) = 2N - 1$  (3)

将  $A_{N+1}$  标签添加到  $A_1 \cdots A_N$  中,原有识别情况与增加标签后的识别情况如图 2 所示。

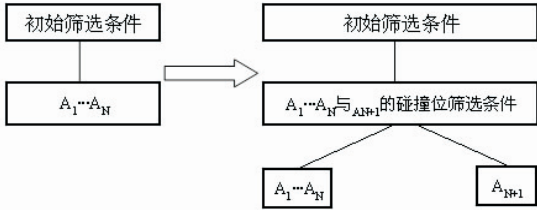


图 2  $A_{N+1}$  标签添加到  $A_1 \cdots A_N$  识别情况改变过程图

根据图 2 所示,可见  $R(N + 1) = R(N) + 2$  (每增加 1 个标签,Request 次数增 2),将(3)代入可得  $R(N + 1) = 2N - 1 + 2 = 2N + 2 - 1 = 2(N + 1) - 1$ ,依此继

表 4  $R(2^1) \sim R(2^4)$  标签的识别次数对照表

| 标签数量          | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 |
|---------------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|
| 标签数量 $N = 2$  | A1 | A2 |    |    |    |    |    |    |    |     |     |     |     |     |     |     |
| Request 次数    | 2  | 1  |    |    |    |    |    |    |    |     |     |     |     |     |     |     |
| 标签数量 $N = 4$  | A1 | A2 | A3 | A4 |    |    |    |    |    |     |     |     |     |     |     |     |
| Request 次数    | 3  | 1  | 2  | 1  |    |    |    |    |    |     |     |     |     |     |     |     |
| 标签数量 $N = 8$  | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |    |     |     |     |     |     |     |     |
| Request 次数    | 4  | 1  | 2  | 1  | 3  | 1  | 2  | 1  |    |     |     |     |     |     |     |     |
| 标签数量 $N = 16$ | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 |
| Request 次数    | 5  | 1  | 2  | 1  | 3  | 1  | 2  | 1  | 4  | 1   | 2   | 1   | 3   | 1   | 2   | 1   |

续推断可得  $R(N) = 2N - 1$  成立。引入 2.3 规则一: 仅 1 位碰撞可识别 2 个标签。当存在满二叉树叶子节点标签, 且两两标签均只有 1 位碰撞的情况下, 很容易判断出  $N$  个标签的识别将减少  $N$  次回溯改变筛选条件状况, 并且这个状况识别次数最低, 即  $R(N) = 2N - 1 - N = N - 1$ 。

综上所述, 证明  $N - 1 \leq R(N) \leq 2N - 1$  成立, 其 Request 请求命令时间复杂度为:  $O(N)$ 。同时本算法识别标签过程中并不考虑碰撞位连续还是间隔问题。

### 3 结束语

本算法通过研究了现有的二进制防碰撞算法, 并在此基础上为原有算法融进了折半搜索算法和回溯算法的综合思想。并为该算法提出三条规则:

①仅 1 位碰撞可同时读取两个标签;

②多位碰撞, 筛选条件转变为: 最高碰撞位置 0, 其他碰撞位置 1;

③每读取标签成功后将当前筛选条件转变为上一个筛选条件, 如继续发生多位碰撞, 执行②, 否则执行③直至全部标签被识别。

除此规则外本算法识别标签过程中并不用考虑碰撞位连续还是间隔的问题。

现有的二进制防碰撞算法读写器寻呼次数为  $R(N) = 2N - 1$  (锁位后退算法 BLBO<sup>[4]</sup>) 已优于其他二叉搜索树算法等二叉树算法。本算法则更进一步提高了识别效率, 且当标签数量越多则  $R(N)$  越趋近于  $N - 1$  次寻呼的状况。通过算法验证  $N - 1 \leq R(N) \leq 2N - 1$  ( $N$  表示标签数,  $R(N)$  表示寻呼次数)。

(上接第 150 页)

中的应用[D]. 海口: 海南师范大学, 2010.

[16] 王 晶. 基于 ARM 指纹识别预处理的研究与实现[D]. 太原: 中北大学, 2011.

[17] 刘卫刚, 胡锡梅, 王福明, 等. 一种基于频域滤波的指纹图像增强算法[J]. 机器视觉, 2011(9): 72-73.

[18] 李 昊, 傅 曦. 精通 Visual C++ 指纹模式识别系统算法及实现[M]. 北京: 人民邮电出版社, 2008.

[19] 李 夏, 武君胜. 一种面向指纹识别的鲁棒脊线跟踪算法[J]. 计算机仿真, 2009, 26(1): 205-208.

[20] 谢 睿. 指纹识别系统中图像质量评估与匹配算法的研究及实现[D]. 成都: 电子科技大学, 2011.

[21] 张新森. 基于 Gabor 滤波的指纹图像识别研究与实现[J]. 计算机与现代化, 2011(7): 24-25.

[22] 陈燕玲. 一种基于滤波特征和不变矩的混合指纹识别算法[J]. 桂林电子科技大学学报, 2010, 30(1): 63-65.

[23] 李艳华. 序数特征在指纹识别及可撤销模板中的应用

### 参考文献:

- [1] 陈 冲, 徐 志, 何明华. 一种新的 RFID 防碰撞算法的研究[J]. 福州大学学报(自然科学版), 2009, 35(3): 367-371.
- [2] 郎为民. 射频识别(RFID)技术原理与应用[M]. 北京: 机械工业出版社, 2006: 58-91.
- [3] 芬肯策勒. 无线电感应的应答器和非接触 ic 卡的原理与应用[M]. 陈大才译. 第 2 版. 北京: 电子工业出版社, 2005.
- [4] 王 雪, 钱志鸿, 胡正超, 等. 基于二叉树的 RFID 防碰撞算法的研究[J]. 通信学报, 2010, 31(6): 49-57.
- [5] Shih D H, Sun P L, Yen D C, et al. Taxonomy and survey of RFID anti-collision protocols[J]. Computer Communication, 2006, 29: 2150-2166.
- [6] Kim S S, Kim Y H, Lee S J, et al. An improved anti-collision algorithm using parity bit in RFID system[C]//The 7th IEEE International Symposium on Network Computing and Applications. [s. l.]: [s. n.], 2008: 224-227.
- [7] 李世煜, 冯全源, 鲁 飞. 基于 BIBD(4, 2, 1) 的 RFID 防碰撞算法[J]. 计算机工程, 2009, 35(3): 279-281.
- [8] 孙文胜, 陈安辉. 高效的 RFID 混合询问树防碰撞算法[J]. 计算机应用研究, 2011, 28(10): 3717-3719.
- [9] 伍继雄, 江 岸, 黄生叶, 等. RFID 系统中二叉树防碰撞算法性能的提升[J]. 湖南大学学报(自然科学版), 2010, 37(12): 82-86.
- [10] Myung J, Lee W. Adaptive binary splitting: a RFID tag collision arbitration protocol for tag identification[J]. Mobile Networks and Applications, 2006, 11(5): 711-722.
- [11] 侯胜宇, 冯 锋. 一种改进的二叉树型 RFID 防碰撞算法[J/OL]. 2012-01-16. <http://www.cnki.net/kcms/detail/11.2127.tp.20120116.0926.017.html>.
- [12] 严蔚敏. 数据结构[M]. 北京: 清华大学出版社, 1997.
- [13] [D]. 西安: 西安电子科技大学, 2011.
- [24] Galbally J, Alonso-Fernandez F, Fierrez J, et al. A high performance fingerprint liveness detection method based on quality related features[J]. Future Generation Computer Systems, 2012, 28(1): 311-321.
- [25] Tan B, Schuckers S. Comparison of ridge and intensity based perspiration liveness detection methods in fingerprint scanners[J]. Biometric Technology for Human Identification III, 2006, 6202: A2020-A2020.
- [26] Wei Z, Qiu X, Sun Z, et al. Counterfeit IRIS detection based on texture analysis[C]//ICPR. [s. l.]: [s. n.], 2008: 1340-1343.
- [27] Lapsley P, Lee J, Pare D, et al. Anti-fraud biometric scanner that accurately detects blood flow[P]. US: 5737439, 1998-04-07.
- [28] Baldisserra D, Franco A, Maio D, et al. Fake fingerprint detection by odor analysis[J]. LNCS, 2005, 3832: 265-272.

# RFID折半回溯动态二进制防碰撞算法的研究

作者: [黄俊南, 李展宗](#)  
作者单位: [泉州经贸职业技术学院 信息系, 福建 泉州 362000](#)  
刊名: [计算机技术与发展](#)  
英文刊名: [Computer Technology and Development](#)  
年, 卷(期): 2012(10)

本文链接: [http://d.g.wanfangdata.com.cn/Periodical\\_wjfz201210040.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjfz201210040.aspx)