Double Time Slot RFID Anti-collision Algorithm based on Gray Code

Hongwei Deng[1]
School of Computer Science and Technology, Hengyang Normal University; School of Information Science and Engineering, Central South University; Hunan Provincial Key Laboratory of Intelligent Information Processing and Application
Hunan, 421002, China.
E-mail: dhwwhd@163.com

Lang Li
School of Computer Science and Technology
Hengyang Normal University, Hengyang
Hunan, 421002, China.
E-mail: lilang911@126.com

Hui Wang
School of Computer Science and Technology; Hengyang Normal University
Hunan, 421002, China.
E-mail: wanghui18188@hotmail.com

To solve the radio frequency identification problem, an anti-collision algorithm based on the Gray code (BSGC) was developed. The proposed algorithm reduces the effort required to search for tag prefixes in accordance with Gray code regulations and enhances the identification speed through efficient division and branching of the collision tags. The results of simulation and subsequent analysis of the proposed algorithm confirm that it effectively reduces the number of collisions and transmission delays, enhances the throughput rate, and increases the tag identification efficiency.

ISCC 2017
16–17 December, 2017
Guangzhou, China

[1]Speaker
This research was financially supported by the National Natural Science Foundation of China (No: 61572174) and the Science and Technology Plan Project of Hunan Province (No. 2016TP1020).

1.Introduction

The internet of things (IoT) is growing at an unprecedented rate. As such, radio frequency identification (RFID) technology is finding applications wider than ever before due to its characteristics of conveniently reading data without direct interactions between communication devices and its ability to withstand the forces and work reliably in severe environments. Multiple industry sectors rely upon RFID technology. In the real world applications, however, the technique is not without its share of flaws. In practical, multiple tags transmit data at the same time via different channels. This leads to data collision, while simultaneously affecting both the accuracy and efficiency of data transmission in the overall system. In order to address this problem of data collision, anti-collision algorithms for RFID systems have recently become a topic of considerable research interest. This paper proposes one such anti-collision algorithm based on the Gray code that simplifies the search for tag prefixes as per Gray code regulations and enhances identification speed through effective branching of collision tags.

The time division multiple access (TDMA) algorithm, used at present, is easy to implement and very popular. It can mainly be divided into two algorithms: ALOHA and the binary search tree. Pure ALOHA is easy to implement, and when the number of tags is small, it identifies tags efficiently; however, the throughput rate is low (only up to 18.4%). The frame slot ALOHA algorithm [1], dynamically determines the slot number in the frame, based on the number of tags, to maximize throughput. The binary search tree algorithm [2] is composed of query and response commands generated by a multicycle reader. The workflow of the binary search tree algorithm is similar to that of the binary tree data structure. Other correlated algorithms include the bit-by-bit tree (BBT) algorithm [3], the binary search algorithm based on a regressive binary system [4], the adaptive binary splitting (ABS) algorithm [5], the Q-tree algorithm [6], and dynamic binary search algorithms [7]. In these algorithms, tags only transfer the L-K bit data out of a command, which greatly reduces redundant data in the information being transmitted. With an improved tree structure for the RFID anti-collision algorithm [8], the binary tree is divided into several small branches ,thereby reducing the number of searches, optimizing idle time slot numbers, and improving the recognition efficiency. The algorithm, however, still seems inadequate. To overcome these inadequacies, we put forward a double time slot RFID anti-collision algorithm based on the Gray code.

2. Algorithm Principle

2.1. Introduction to the Gray Code

The Gray code, named after Frank Gray in his patent from 1953, is a binary cyclic code. To date, the code has been used in analog–digital conversion as well as in the angle conversions. The Gray code can be recognized as a reliability code and represents a coding scheme designed to minimize mistakes. Conversion from a binary code to the Gray code can be illustrated as follows.

Let's consider a scheme of binary code represented by $B = B_{n-1}, ..., B_{i+1}, B_i, ..., B_0$. The corresponding Gray code can be written as $G = G_{n-1}, ..., G_{i+1}, G_i, ..., G_0$. The transformation formula from B to G can be expressed as follows:

$G_{n-1} = B_{n-1}$,

$$G_i = B_{i+1} \oplus B_i . \quad (n-1 \geq i \geq 0) \tag{2.1}$$

2.2. Improved RFID Anti-collision Algorithm

The proposed anti-collision algorithm utilizes the Gray code to improve instructions in the original RFID algorithm as follows:

1) An additional code M and a time slot control code N are set. When a certain tag is activated, the top three collision bits of the tag are first converted to the Gray code, the Gray code number being three. The initial value of the extra code M is set as 0000. From left to right, the positions of digits representing the value of M are named as third, second, first, and zero in that order. The extra code M have a single "1" ,and its position is determined according to the Gray code decimal value of the last two digits. If, after two tag collision bits, the Gray code value is 00, the corresponding code M takes the value 0001. Similarly, a Gray code value of 01 corresponds to M = 0010, 10 corresponds to M = 0100, 11 corresponds to M = 1000, and so on. Table 1 below lists tag identification codes along with their corresponding values for other code types. At the same time, the value of the time slot control code N corresponds to the first value taken by the Gray code. An accumulator C is set to each tag, its value corresponding to the time slot control code N in order to control the response time slot of each tag.
2) A query instruction is termed as REQUEST. Depending on the type and number of parameters it contains, the REQUEST instruction is divided into three types: single parameter (i.e., REQUEST (bit)), three parameters (i.e., REQUEST (D1, D2, D3)), and a type containing the value of the code M (REQUEST (EC)). REQUEST (bit) is the initial query instruction, to which all tags in the reader must respond; parameters D1, D2, and D3 in REQUEST (D1, D2, D3) represent collision bits of the tag; REQUEST (EC) represents a query instruction tag for the additional code M. A queue is stored in the REQUEST instruction such that when the queue is empty, the algorithm ceases to operate and the tag identification process is terminated.
3) A selection instruction is termed as SELECT. The SELECT (ID) instruction indicates that tags with the specified ID are ready to read/write data.
4) A reading instruction is termed as READ. The READ (ID) instruction reads out tags with the specified ID in the data.
5) A shield instruction is termed asUNSELECT. The UNSELECT (ID) instruction causes tags with the specified ID to enter an inactive state. As such, these tags become unresponsive when a REQUEST instruction is received.

Tag identification code (Top three collision positions)	Gray code	Time slot control code N	Additional code M
000	0 00	0	0001
001	0 01	0	0010
010	0 11	0	1000
011	0 10	0	0100
100	1 10	1	0100

101	1 11	1	1000
110	1 01	1	0010
111	1 00	1	0001

Table 1: List of Tag Codes

Table 1 explains the conversion of values from one code form to another. Each query may be responded to in one of the two response time slots as determined by the value of the time slot control code N. Tags respond in the first time slot when N = 0 and in the second time slot when N = 1. It can be seen from Table 1 where tags may have the same value for M but different values for N. During the tag identification process, the reader can identify the corresponding tag identification code (three collision bits) at different time slots by the value of the code M. The tag identification code is then added to the queue, and a new query instruction is constructed to identify the tags according to the first-in first-out (FIFO) method.

2.3. Algorithm Workflow

The algorithm operation, the flow chart for which is shown in Figure 1, which can be explained as follows:

Step 1: The reader sends an initialization query instruction REQUEST (bit).

Step 2: A response is generated by tags matching the query instruction. If a collision is detected, jump to Step 3. If only one tag responds to the reader's instruction, it is identified directly.

Step 3: In case of a single collision, the two colliding tags are identified simultaneously using the binary tree algorithm. For two collisions, the reader uses the quad-tree algorithm to identify colliding tags. In cases of more than two collisions, the algorithm checks the status of the tag time slot control code N, allocating the first time slot to the tag corresponding to N = 0. Likewise, the tag corresponding to N = 1 is allocated to the second slot response. If, however, the tag slots also collide, the reader confirms the top three identification code of the time slot, based on the value of the additional code M, and inserts relevant tag identification codes in the queue.

Step 4: Based on the queue information, the reader constructs a new query instruction. If the queue is not empty, the algorithm jumps to Step 2; if the queue is empty, the identification process is terminated.

2.4. Example of the Proposed Algorithm's Operation

Here, we present an example explaining the identification process followed by the proposed algorithm. In this example, the reader has eight tags to identify, and each tag has a six-digit binary ID number, as listed in Table 2.

Tags	Manchester coding
Tag1	1 0 0 1 0 1
Tag2	1 0 0 1 1 1
Tag3	0 0 0 0 1 1
Tag4	0 0 0 1 0 0
Tag5	0 0 1 1 0 0
Tag6	0 0 1 1 1 0
Tag7	1 0 1 0 0 0
Tag8	1 0 1 0 1 0

Table 2: Manchester coding

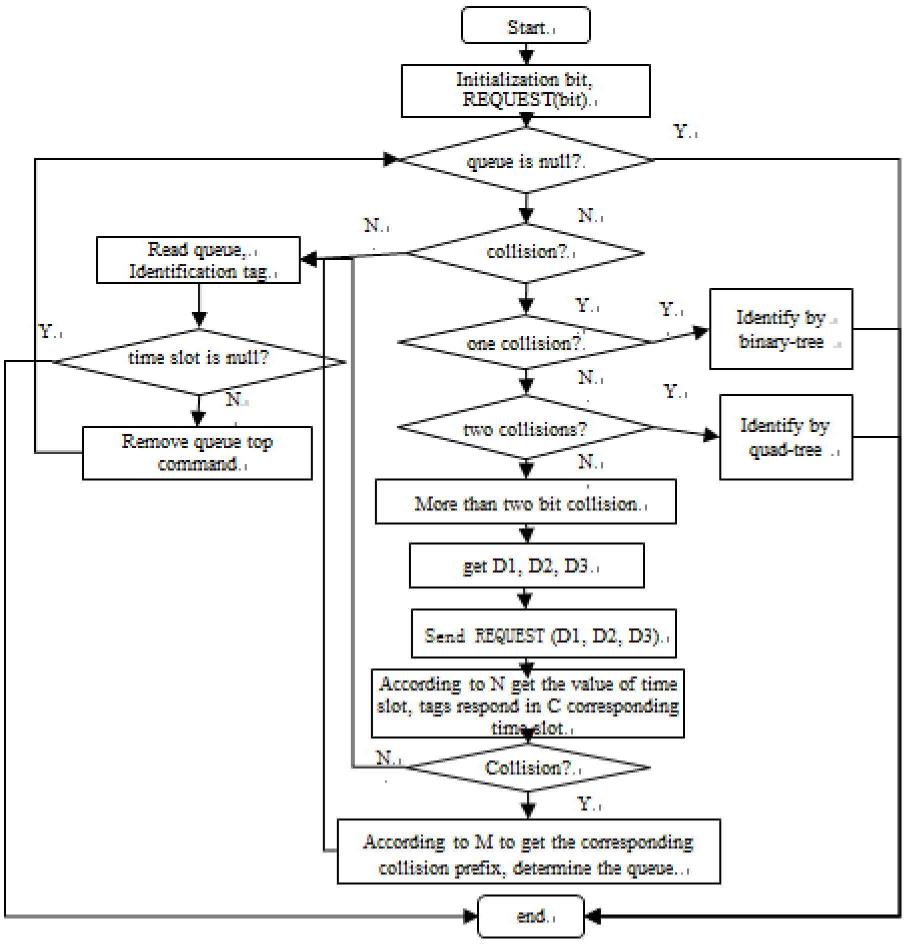

Start.
Initialization bit, REQUEST(bit).
queue is null?
Y.
N.
collision?.
N.
Read queue, Identification tag.
Y.
one collision?.
Y.
Identify by binary-tree
time slot is null?
Y.
N.
N.
two collisions?
Y.
Identify by quad-tree
Remove queue top command.
N.
More than two bit collision.
get D1, D2, D3.
Send REQUEST (D1, D2, D3).
According to N get the value of time slot, tags respond in C corresponding time slot.
N.
Collision?.
Y.
According to M to get the corresponding collision prefix, determine the queue.
end.

Figure 1: Algorithm Flowchart.

The stepwise identification process followed by the algorithm is as follows.

(1) The reader initializes a bit 111111.

(2) A query instruction REQUEST (bit) is generated, and tags of interest to the reader respond to the request. The decoding information can be easily recognized as X0XXXX based on the principle of Manchester encoding, as listed in Table 3. As seen in Table 3, positions 1, 3, and 4 can be identified as corresponding to the top three collisions. The reader then generates a new query instruction, REQUEST (D1, D3, D4), and sends it to the tags. At the same time, the tags convert their top three positions of collision into a Gray code and a value is assigned to the time slot control code N corresponding to the first bit of the Gray code. An accumulator C is then set to each tag, and N values corresponding to each tag are stored in its own accumulator. Based on the values of C, a response to the reader is generated in the corresponding time slot. Table 4 lists the sequence of events.

(3) When C = 0, that is, when N = 0, tags 3, 4, 5, and 6 responds in the time slot 0. At this time, the reader sends the REQUEST (EC) instruction to query the value of the additional code M inside the tags and obtains the collision information of M as 0XXX, as listed in Table 5. According to Table 4, there may arise one of four different situations for M in time slot 0 (0001, 0010, 1000, 0100). At the same time, the collision information for M (0XXX) only has three possible scenarios (0001, 0010, 0100), and their corresponding tag collision bit information is (000, 001, 011). Once these three prefix REQUEST instructions are in the "queue," instructions REQUEST (000) and REQUEST (001) can identify Tag3 and Tag4, respectively. When the

reader sends the instruction REQUEST (011), collision information is received, as listed in Table 6. As it is a case of a single collision, Tag5 and Tag6 can be directly identified using the query tree algorithm. At this point, all tags in time slot 0 have been identified.

Position	1 2 3 4 5 6
Information	X0XXXX
Tag1	1 0 0 1 0 1
Tag2	1 0 0 1 1 1
Tag3	0 0 0 0 1 1
Tag4	0 0 0 1 0 0
Tag5	0 0 1 1 0 0
Tag6	0 0 1 1 1 0
Tag7	1 0 1 0 0 0
Tag8	1 0 1 0 1 0

Table 3: Information of Colliding bits

Time slot	Tag identification code (top three collision)	Corresponding Gray code	N	M	C	In response to the tag
a0	000	000	0	0001	0	Tag3
	001	001		0010		Tag4
	010	011		1000		
	011	010		0100		Tag5, Tag6
a1	100	110	1	0100	1	
	101	111		1000		Tag1, Tag2
	110	101		0010		Tag7, Tag8
	111	100		0001		

Table 4: Tag Response Time Slot

Position	1 2 3 4 5 6	M
Information	0 0 XXXX	0 XXX
Tag3	0 0 0 0 1 1	0 0 0 1
Tag4	0 0 0 1 0 0	0 0 1 0
Tag5	0 0 1 1 0 0	0 1 0 0
Tag6	0 0 1 1 1 0	0 1 0 0

Table 5: Code M collision Information (First Round)

Position	1 2 3 4 5 6	M
Information	0 0 1 1 X 0	0 1 0 0
Tag5	0 0 1 1 0 0	0 1 0 0
Tag6	0 0 1 1 1 0	0 1 0 0

Table 6: Code M Collision Information (Second Round)

(4) Next, the accumulator C takes a value equal to 1 and inspects the occurrence of collisions in time slot 1; the steps follow the same sequence as mentioned above.

(5) Once all tags have been identified by the reader and the queue is empty, the identification process is terminated.

3.Performance Analysis

3.1. Query-command Time Analysis of the Reader

The various parameters used in the analysis can be defined as follows:

K: Tag length

L: Top number of bits of the tag collision (L< K)

N: Total number of query tags

P(c): In N tags, the probability of collision

P(s): In N tags, the probability of recognition

P(i): In N tags, the probability of free and idle prefixes

n(c): Number of collision prefixes

n(s): Number of successful recognition prefixes

n(i): Number of idle prefixes

Q: Total number of query instructions that the reader needs to transmit.

In practical applications, the total number of tags to be identified is massive; that is, N is very large. In the definition stage of tag prefix, the reader transmits a command, REQUEST (bit), to which all tags respond. From these responses, the reader obtains information regarding tag collisions. Let us suppose there are K bit collisions. For the top L collision bits, there may be 2^L prefixes. The variables defined above are interrelated in Equations (3.1)–(3.6).

$$P(i) = (1 - 1/2^L)^N \tag{3.1}$$

$$P(s) = N/2^L (1 - 1/2^L)^{N-1} \tag{3.2}$$

$$P(c) = 1 - P(i) - P(s) = 1 - (1 - 1/2^L)^N - N/2^L (1 - 1/2^L)^{N-1} \tag{3.3}$$

$$n(i) = 2^L P(i) = 2^L (1 - 1/2^L)^N \tag{3.4}$$

$$n(s) = 2^L P(s) = 2^L N/2^L (1 - 1/2^L)^{N-1} \tag{3.5}$$

$$n(c) = 2^L P(c) = 2^L (1 - (1 - 1/2^L)^N - N/2^L (1 - 1/2^L)^{N-1}) \tag{3.6}$$

When $N >> 2^L$, $(1 - 1/2^L)^N \rightarrow 0$ and $(1 - 1/2^L)^{N-1} \rightarrow 0$; thus, $n(i) \rightarrow 0$, $n(s) \rightarrow 0$, and $n(c) \rightarrow 2^L$. Thus, it can be shown that when the number of tags is very large, there are only a negligible number of free and successful identification prefixes, and almost all prefixes collide.

The BSGC algorithm is divided into two stages—tag prefix determination and time slot query.

(1) In the prefix determination stage, readers need to broadcast one query instruction, REQUEST (bit), to obtain tag information. This is to be followed up by two commands of the type REQUEST (EC) to obtain collision bit information.

(2) In the time slot query stage, time slots are assigned to colliding tags and are queued up in respective accumulators. Each of the two time slots (0 and 1) has 2^L branches, and each branch has $(2^{K-L} \times 2^{L-1})$ sub-branches. In one time slot, the reader sends a query instruction number (Q_{one}) for each branch, defined as follows:

$$Q_{one} = 2^{K-L} - 1 \tag{3.7}$$

To query a time slot, the number of query instructions (Q_0) required can be expressed as follows:

$$Q_0 = 2^{L-1} Q_{one} = 2^{L-1} (2^{K-L} - 1) \tag{3.8}$$

Similarly, the same query instruction number (Q_0) is required for another time slot. The total number of query instructions (Q) that reader needs to generate is derived as follows:

$$Q = 3 + 2Q_0 = 3 + 2^K - 2^L \ (1 < 2^L \leq K) \tag{3.9}$$

3.2. Transmission Delay Analysis

In the prefix determination stage, the reader broadcasts a query instruction of length R to which the tag responds with K-bit collision information. The reader then determines the prefix of tag length L and continues to broadcast a query instruction of length L. Upon receipt of this query instruction, the tag responds with the information of the additional code M collisions having a length 2^{L-1}. Therefore, the total length (L_1) of the transmission channel can be expressed as

$L_1 = 2\,(R + L + 2^{L-1})$ (3.10)

In the time slot identification stage, the total length of the query instruction sent by the reader and responded to by the tag is equal to K. Therefore, the length of data transmitted in each branch (L_d) is equal to K times the number of query instructions sent by the reader. Here, L_d can be derived as follows:

$L_d = K\,Q_{one} = K\,(2^{K-L} - 1)$ (3.11)

Thus, we get 2^L search branches, and the total number of bits (L_2) in the transmission channel is given by

$L_2 = 2^L\,L_d = 2^L \times K \times (2^{K-L} - 1)$ (3.12)

From Eqs. (3.11) and (3.12), the total number of bits (L) in the transmission channel of the proposed algorithm can be expressed as follows:

$L = L_1 + L_2 = 2\,(R + L + 2^{L-1}) + 2^L \times K \times (2^{K-L} - 1)$ (3.13)

Assuming that the data transmission rate in the channel is V bit/s, the transmission delay of the BSGC algorithm can be expressed as

$T = L / V = (2\,(R+L+2^{L-1}) + 2^L K\,(2^{K-L} - 1)) / V$ (3.14)

3.3. Analysis of the Throughput Rate

If the number of the tags is N and the number of query instructions generated by the reader is Q, the system of throughput (S) can be defined as

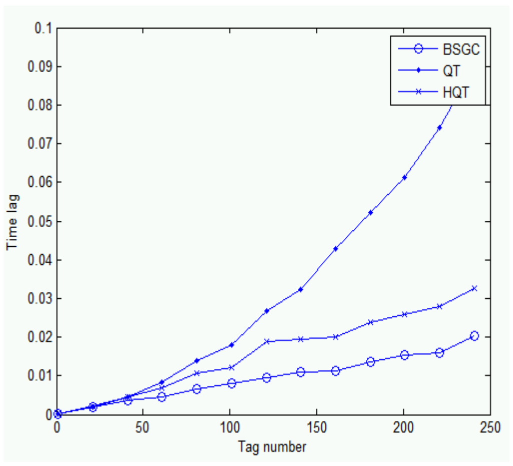
$S = (N / Q) \times 100\% = (N / (3 + 2^K - 2^L) \times 100\%$ (3.15)

The throughput rate is determined by the number of searches. Therefore, when the length of the tag prefixes approaches a value closer to the tag length, the throughput increases accordingly.

4. Algorithm Emulation

Through MATLAB emulation, the performance of the proposed algorithm (hereafter the BSGC algorithm) was compared to the query tree (QT) and hybrid query number (HQT) algorithms. Fig. 2 shows that as the tag number assumes larger values, the QT and HQT algorithms required more time to complete the identification process, and that they were less efficient compared to the proposed BSGC algorithm.

From Fig. 3, it can be seen that the QT and HQT algorithms account for much longer time compared to the proposed BSGC algorithm. Given the same number of tags, the BSGC algorithm is far superior to the QT and HQT algorithms.



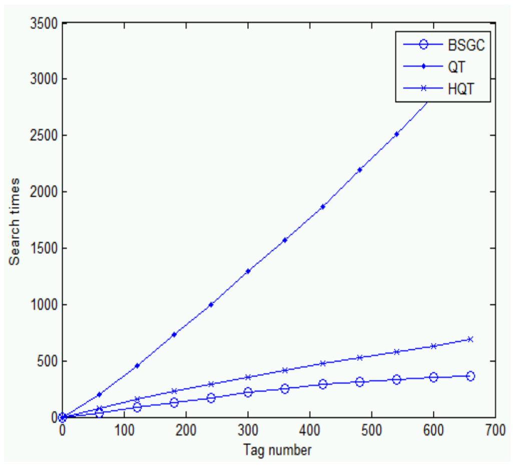


Figure 2:Time lag of the proposed BSGC algorithm compared to the QT and HQT algorithms.

Figure 3:Search time of the proposed BSGC algorithm compared to the QT and HQT algorithms.

5. Conclusion

On the basis of the above analyses and the comparative algorithm emulation, it is clear that the proposed algorithm outperforms the popular query tree and hybrid query number algorithms in terms of both transmission delay and average search time. As such, the proposed algorithm for RFID systems will help to optimize the overall running time, while considerably enhancing identification efficiency.

References

[1] C. W. Qing, Z. M. Xin, et al. *Enhanced dynamic frame slotted ALOHA algorithm for anti-collision in RFID systems* [J]. Journal of Huazhong University of Science and Technology: Humanities and Social Science Edition. 35(6): 14–16 (2007).

[2] K. Finkenzeller. *RFID Handbook: Radio-Frequency Identification Fundamentals and Applications* [M]. 2nd edition, John Wiley (2003).

[3] D. Connors, B. Ryu, G. J. Pottie, S. Dao. *A medium access control protocol for real time video over high latency satellite channels* [J]. Mobile Networks and Applications. 7(1): 9–20 (2002).

[4] L. B. J. Z. L. Ye. *A RFID Anti-collision searching algorithm based on regressive style binary system* [J]. Computer Application and Software. 26(12): 96–98 (2009).

[5] C. Law, K. Lee, K. Y. Siu. *Efficient memoryless protocol for tag identification* [A]. Proceedings of the 4th International Workshop on DIALM. Boston. 75–84 (2000).

[6] H. Landaluce, A. Perallos, I. G. Zuazola. *A fast RFID identification protocol with low tag complexity* [J]. IEEE Communications Letters. 17(9): 1704–1706 (2013).

[7] Y. S. Sen, Z. Y. Ju, P. W. Dong. *An anti-collision algorithm based on binary-tree searching of regressive index and its practice* [J]. Application and engineering of Single Chip processor, 40(16): 26–28 (2004).

[8] L. Xiaohui, Q. Zhihong, Z. Yanhang, G. Yuqi. *An adaptive tag anti-collision protocol in RFID wireless systems* [J]. Communications China. 11(7): 117–127 (2014).

[9] W. Xue, Q. Zhihong, L. Xiaohui, C. Chao. *Improved tree structure anti-collision algorithm of RFID* [J]. Journal on Communications. 36(7): 129–137 (2015).