Journal of Computer and Communications, 2015, 3, 72-79
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.35009

Parallel Response Ternary Query Tree for RFID Tag Anti-Collision

Ching-Nung Yang, Song-Ruei Cai, Li-Zhe Sun

Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
Email: cnyang@mail.ndhu.edu.tw

Received February 2015

Abstract

A tag-collision (or missed reads) in RFID system (Radio Frequency Identification) system degrades the identification efficiency. The so-called tag collision is that a reader cannot identify a tag when more than one tags respond to a reader at the same time. There are some major anti-collision protocols on resolving tag collision, e.g., ALOHA-based protocol, binary tree protocol, and Query Tree (QT) protocol. Up to date, most tag anti-collision protocols are QT protocols. QT protocols are categorized into *M*-ary query tree (QT). In the previous literature, choosing *M* = 3 (*i.e.*, a ternary QT (TQT)) was proven to have the optimum performance for tag identification. Recently, Yeh *et al.* used parallel response approach to reduce the number of collisions. In this paper, we combine the partial response and TQT to propose an effective parallel response TQT (PRTQT) protocol. Simulation results reveal that our PRTQT outperforms Yeh *et al.*'s protocol and TQT protocol.

Keywords

Radio Frequency Identification (RFID), Tag Collision, Query Tree, Ternary Tree, Parallel Response

1. Introduction

Radio frequency identification (RFID) system consists of readers, tags and backend database server [1]. This technology can be applied in inventory control, distribution industry, supply chain management,⋯, and etc. However, there are two types of collision problems in RFID system, the tag collision and the reader collision [2] [3]. Tag collisions occur when multiple tags respond to a reader simultaneously and the reader cannot differentiate these tags correctly. In this paper, we deal with the tag-collision problem. When tags communicate with a reader through wireless transmission, they should be uniquely identified one by one. After the successful identification, a reader then sends the collected data from tags to a data processing system (backend database server), for the further application need.

There are two types of two tag anti-collision protocol to address tag-collision problem. One is ALOHA-based protocol [4] and the other is tree-based protocol. ALOHA protocol reduces the tag collisions since it has the starvation problem (a tag cannot be identified for a long time). Tree-based protocols can be classified to the binary tree (BT) and the query tree (QT). In BT protocols [5]-[8], a tag generates a random bit. If the bit is "0", tags transmit their Electronic Product Codes (EPCs) to the reader, and tags having "1" transmit later. By repeat-

How to cite this paper: Yang, C.-N., Cai, S.-R. and Sun, L.-Z. (2015) Parallel Response Ternary Query Tree for RFID Tag Anti-Collision. *Journal of Computer and Communications*, **3**, 72-79. http://dx.doi.org/10.4236/jcc.2015.35009

ing this process, all tags can be uniquely identified.

QT protocols [9]-[12] are sending a query string to identify the tags. If just one tag ID matches the query string, then the tag is identified. When multiple tags have the same query string, they are collided with each other. Otherwise, this situation is idle. Usually, QT protocol is implemented as binary QT (BQT) protocol that query tree is binary tree. Also, *M*-ary QT (MQT) were proposed in [10]-[12] with arbitrary number *M*. Representing binary tuples to a decimal digit, a QT can be used as MQT for M = 2, 4, 8···etc. Using large *M*-ary tree reduces the number of collisions, but increases the number of idle situations. Obviously, MQT protocol is reduced to BQT protocol for *M* = 2. Mathys *et al.* claimed the optimum performance of MQT is *M* = 3, *i.e.*, ternary QT (TQT) [13]. However, in real environment, readers and tags communicate through binary code. TQT protocol is hard to implement from binary EPC. In [14], the authors adopted a conversion of 3B2T (3 binary code to 2 ternary code), to practically implement TQT protocol.

Recently, Yeh *et al.* [15] proposed a parallel response query tree scheme that combines the frequency shift keying (FSK) modulation and Manchester coding to provide two subcarriers for tags communicating with reader in parallel. In this paper we adopt the parallel subcarriers of Yeh's protocol and the optimal performance of TQT protocol to design a parallel response ternary query tree (PRTQT) protocol. The remainder of this work is organized as follows. In Section 2, we review previous woks, Yeh *et al.*'s protocol and 3B2T conversion in TQT protocol. In Section 3, we propose the PRTQT algorithm for tag anti-collision. Performance and comparison are given in Section 4. Finally, our conclusion is drawn in Section 5.

2. Previous Works

2.1. Partial Response in Query Tree

Yeh *et al.* [15] used frequency shift keying (FSK) modulation technique combining Manchester code to provide two subcarriers for tags to communicate with reader in parallel. In [16], two subcarrier tones in f_0 = 2.2 MHz and f_1 = 3.3 MHz based on the baseband carrier in 900 MHz is provided for a reader could receive two separate signals at the same time. The responded bits of tags are encoded as Manchester code, in which a low-to-high transition stands for 0, and a high-to-low transition stands for 1, as shown in **Figure 1(a)** and **Figure 1(b)**. By this parallel two subcarrier responding and partial parallel prefix matching, Yeh *et al.*'s protocol performs like 4-ary QT protocol. However, Yeh *et al.*'s enhance the performance of 4-ary QT protocol since using two subcarriers.

A simple example with 8 tags with IDs {(0000100), (0010100), (0011010), (0011101), (1010110), (1011000), (1100111), (1101110)}. When reader send a query string *S* = (00), tags with prefix (00) and (11) will respond.

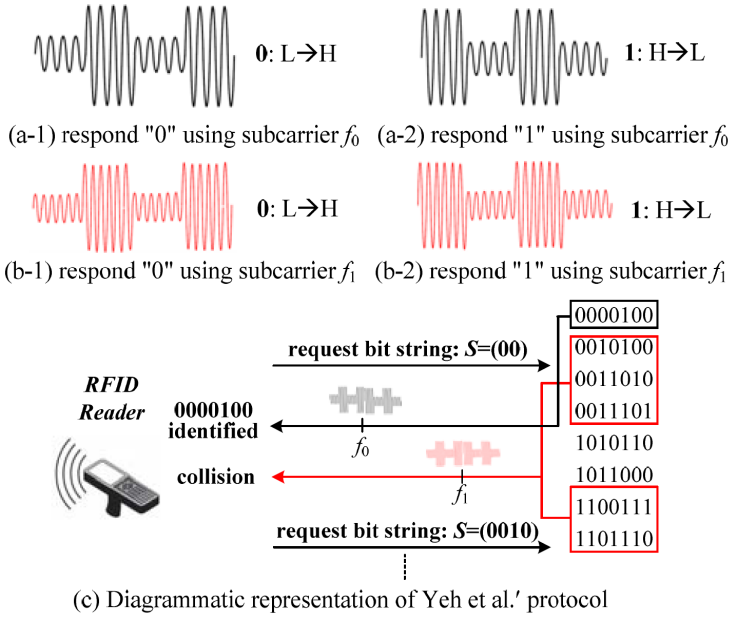

Figure 1. Tag's responses of Yeh *et al.*'s protocol with two subcarriers.

Also, the MSB + R_1 = 0 (respectively, 1) will use subcarrier f_0 (respectively, f_1) to respond, where MSB is the first bit of ID and R_1 is the first bit of the tag remainder exclusive S. In **Figure 1(c)**, since MSB + R_1 = 0 for the tag (0000100), and MSB + R_1 = 1 for other tags of (0010100), (0011010), (0011101), (1100111), and (1101110), thus the tag (0000100) respond using subcarrier f_0 to sender and uniquely identified. Other five tags respond using f_0 and collide with each other. By the same argument, reader continuously sends the query string S = (0010). This procedure is repeated until all tags are successfully identified.

2.2. 3B2T Conversion in TQT

A ternary tree is impossible directly implemented in QT protocol to identify a binary EPC. So using TQT delivers a problem how to efficiently convert 96-bit $(EPC)_2$ to a ternary $(EPC)_3$.

The tradition method of convert binary digits to ternary digits is difficult to implement, and it would waste more time. The 3B2T conversion divides 96 bits EPC to 32 parts of three binary digits. And each three digits can map to two ternary digits by conversion table, this is shown in **Figure 2**. Actually, 96 binary bits just needs 61 ternary digits for conversion. Using 96 × (2/3) = 64 ternary digits for conversion is not the optimal choice, but this way could finish conversion faster than the traditional conversion. By 3B2T conversion divides all bits to be 32 parts, and each part is independent with others. Thus, we do not need to turn covert all EPC at once, and just convert the part we use.

3. The Proposed PRTQT Protocol

As we know, in [13], the authors showed that TQT (M = 3) has the optimum performance (*i.e.*, the less identification time). The proposed PRTQT is motivated by the parallel subcarriers of Yeh's protocol and the optimal performance of TQT protocol. We use a more complex reader that could distinguish three responded signals with different frequencies from tags, and meantime process each response in parallel. Notations and their descriptions used in the proposed PRTQT protocol are defined below.

The proposed PRTQT protocol combines the parallel subcarrier response method and TQT protocol. In the proposed PRTQT, we have to convert binary IDs to ternary IDs for all tags by using 3B2T conversion. A tag with ternary digit $t = (t_1, t_2, \cdots, t_{64})$, where $t_i \in \{0, 1, 2\}$ and $1 \le i \le 64$. Readers first push $(Q, 0)$ into a NULL queue Q. When the queue Q is not null, readers pop a ternary string q ($|q| = x$) from Q to broadcast. If $t_i = q_i$, for $1 \le I \le x - 1$. Then the tag responds by using subcarrier c, where c is t_x. There are three outcomes may occur in subcarrier. When more than one tags respond with the same subcarrier, the collision occurs. Then, readers would push $(Q, q'||c||0)$ into the queue Q, where $q' = (q_1, q_2, \cdots, q_{x-1})$. On the other hand, if only one tag responds with the subcarrier, the tag would be uniquely identified. After the successful identification, tag sends its ID to the reader. This procedure is repeated until the queue Q is empty. In the proposed PRTQT, all binary digits of ID are not necessarily converted to ternary digits at first. We just need 3B2T conversion when the query strings need to be used, so that the performance can be improved. The proposed PRTQT algorithm is shown in **Figure 3**.

Example 1. We use eight tags with 12-bit ID: (000101010010), (000111010111), (001101110000), (001110010110), (010110100001), (101010110001), (110000100010), and (110001110011), to test the pro-

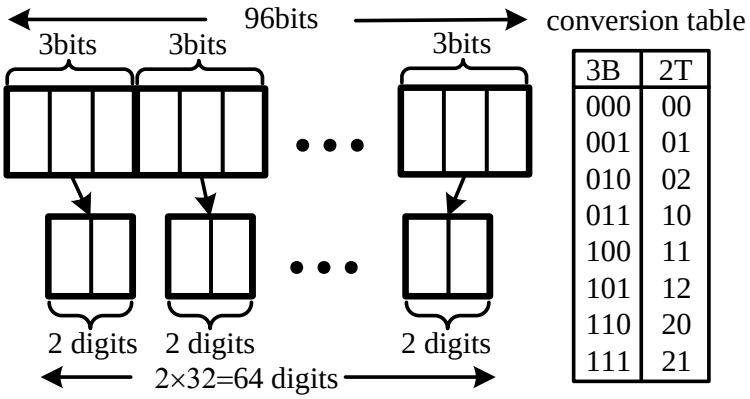

Figure 2. 3B2Tconversion.

```
Algorithm : PRTQT
/* initialize Q */
if (Q = NULL)
  Push(Q, 0);
end
begin
  While (Q!=NULL)
    q=Pop(Q);
    let q' =(q1, q2, ⋯, qx−1);
    reader broadcasts q to tags; /* reader queries tags */
    tag compares its ID with the query q' ;
    /* the (ID)2 is converted to gain the required ti by 3B2T */
    if (V(q' , t)=1)
      tag responds in subcarrier c; /* c is tx */
      for (c is from 0 to 2)
        if (collision in subcarrier c) /* two or more tags respond */
          Push(Q, q' ||c||0);
        else if (only one tag responds in subcarrier c)
          tag transmit its (ID)2 to reader;
        end
      end
    end
  end while
end
```

Figure 3. The Proposed PRTQT.

posed PRTQR protocol.

At first, the first three bits of all tags are converted into two ternary digits: (00), (00), (01), (01), (02), (12), (20), and (20) by 3B2T conversion. Readers then push (0) into a NULL queue Q for initialization. Readers pop a ternary string q = 0 from queue Q and broadcast the query string to all tags. Five tags with ternary prefix (00), (01) and (02), where t_1 = 0, respond to this query using subcarrier 0, and collide with each other since they respond with the same subcarrier 0. However, two tags with ternary prefix (20), where t_1 = 2, respond with subcarrier 2, and collide with each other since they respond with the same subcarrier 2. The tag with ternary prefix (12), where t_1 = 1, responds with subcarrier 1, and is uniquely identified. Thus, the tag (101010110001) with ternary prefix (12) is identified successfully. Readers then push (Q, 00) and (Q, 20) into the queue Q. When tags respond with the same subcarrier c, readers push string (q'||c||0) into the queue Q. After the first query cycle, the remainder of queue Q is {00, 20}. Hence, in the second query cycle, readers pop the string q = (q_1, q_2) = (00) and broadcast to all tags. **Table 1** lists the detail of identifying procedure for all eight tags, and **Figure 4** is its corresponding tree plot. Since we use three subcarriers to send the ternary digits, and thus there are no idle situations in the proposed PRTQT protocol. Finally, there are total 7 interrogation cycles.

4. Performance Evaluation

In this section, we conduct two experiments to evaluate the performance of BQT, TQT and the proposed PRTQT. Suppose tag's ID is using EPC (96-bits). And we have n tags need to be identified, where n = 100, 200, 300, 400, 500, 1000 and 2000. In Experiment A, n tags are randomly chosen that is for real situation. Experiment B is special case for test the performance with similar EPC. In [17], we consider the RFID warehouse distribution. It is reasonable to assume that the EPC data of most items from the same warehouse will be very similar since the items are manufactured by the same company, and are stacked together in a large warehouse. So, tags have very similar EPCs for this case. Both experiments are showing the number of collision cycles N_C, idle cycles N_I and total cycles N_T.

Experiment A. Three protocols are tested: BQT, TQT, Yeh *et al.*'s protocol (denoted as PRQT), and the proposed PRTQT. Also, 96-bit EPCs of test tags are randomly generated.

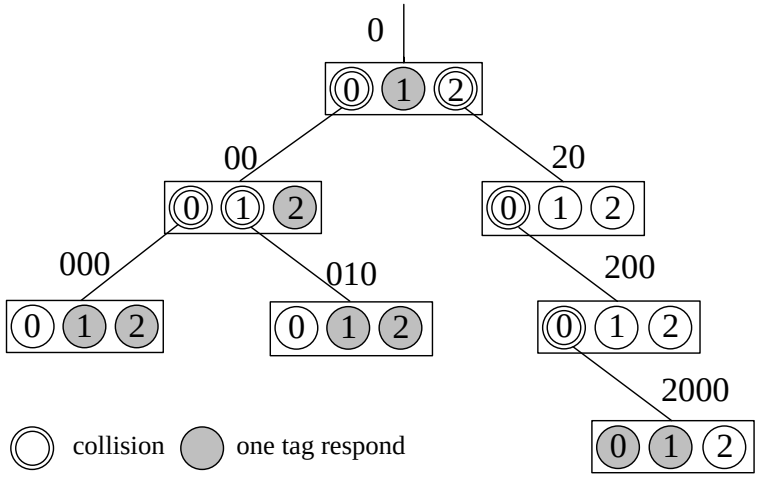

Figure 4. The query tree of PRTQT for Example 1.

Table 1. Identification of nine tags in Example 1 using the proposed PRTQT.

Cycle	*q*	Response		Queue *Q*
0	NULL	Initialization		**0**
		subcarrier 0	collision	
1	0	subcarrier 1	**1 (101010110001*)**	**00**,**20**
		subcarrier 2	collision	
		subcarrier 0	collision	
2	00	subcarrier 1	collision	20,**000**, **010**
		subcarrier 2	**02 (010110100001*)**	
		subcarrier 0	collision	
3	20	subcarrier 1	–	000,010, **200**
		subcarrier 2	–	
		subcarrier 0	–	
4	000	subcarrier 1	**001 (000101010010*)**	010,200
		subcarrier 2	**002 (000111010111*)**	
		subcarrier 0		
5	010	subcarrier 1	**011 (001101110000*)**	200
		subcarrier 2	**012 (001110010110*)**	
		subcarrier 0	collision	
6	200	subcarrier 1	–	**2000**
		subcarrier 2	–	
		subcarrier 0	**2000 (110000100010*)**	
7	2000	subcarrier 1	**2001 (110001110011*)**	NULL
		subcarrier 2	–	

*successfully identified

In BQT and TQT, the leaf in a query tree just can be one of identification, collision, or idle node, where identification node is that one tag is uniquely identified. Hence $N_T = n + N_C + N_I$. For example, when $n = 100$, the BQT has 140 collisions and 42 idle times. The number of total interrogation cycles is $N_T = 282$ (= 100 + 140 + 42). However, in the proposed PRTQT, when $n = 100$, we have 61 collisions and no idle times. The number of total interrogation cycles is $N_T = 95$ (≠100 + 0 + 61). In fact, the PRTQT protocol can identify more than one tag in a cycle. Since three subcarriers are used in the proposed PRTQT, at most three tags can be simultaneously identified within one cycle. Thus, for PRTQT, $N_T \approx [n/3] + N_C + N_I = [n/3] + N_C$ (∵ no idle cycle). For example, the PRTQT has $N_T = 1836$ interrogation cycles, while $[n/3] + N_C = 667 + 1164 = 1831$. Both values are almost the same. For the case $n = 2000$, there 5534 and 5282 interrogation cycles for BQT protocol and TQT protocol, respectively. Our PRTQT uses parallel response method and ternary query tree, and there are total 1836 interrogation cycles, and enhance the $N_T = 2729$ in PRQT. **Table 2** shows all the values of N_C, N_I, and N_T, and **Figure 5** illustrates the number of total collisions for $100 \le n \le 2000$. Obviously, our PRTQT protocol has the less interrogation cycles among all protocols.

Experiment B. Redo Experiment A, but use test tags with the very similar EPC data.

As we know, EPC embraces four sections: header (H: 8 bits), GMN (G: 28 bits), object class (O: 24 bits), and serial number (S: 36 bits). In fact, l_f and l_s are the first half and the second half in an EPC. Thus, in the so-called very similar EPC, we use the same first $l_f = 60$ bits (H + G + O) cascaded with the random $l_s = 36$ bits (S) for all tags. Experimental data are shown in **Table 3**. Compared with Experiment A using random EPC, the numbers of N_T for BQT, TQT, PRQT, and PRTQT increase when using similar EPC. **Figure 6** illustrates the number of total

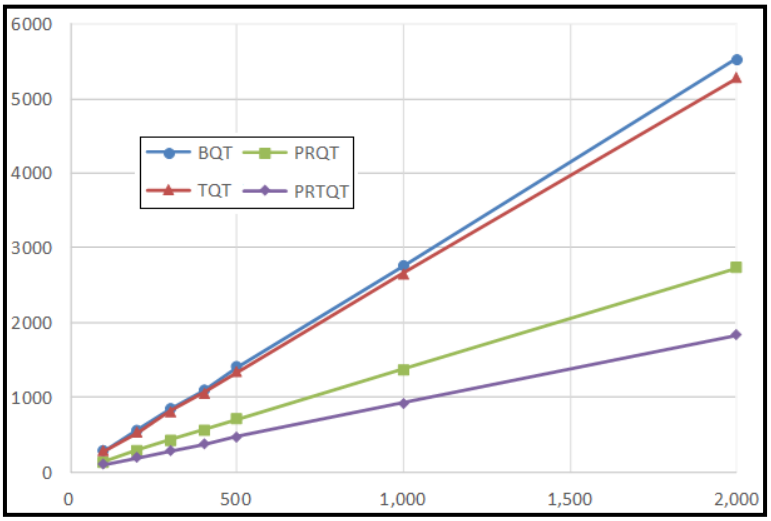

Figure 5. Total cycles of each protocol for Experiment A.

Table 2. N_C, N_I AND N_T for tags with random EPC.

	BQT			TQT			PRQT			PRTQT		
n	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T
100	140	42	282	94	79	273	50	20	134	61	0	95
200	280	82	562	183	146	530	106	42	284	117	0	184
300	417	119	836	278	224	802	156	63	424	177	0	279
400	545	147	1093	366	286	1052	208	74	560	234	0	367
500	703	205	1408	465	374	1339	261	105	703	296	0	466
1000	1378	380	2750	921	731	2652	505	192	1372	579	0	922
2000	2766	768	5534	1835	1447	5282	1000	354	2729	1164	0	1836

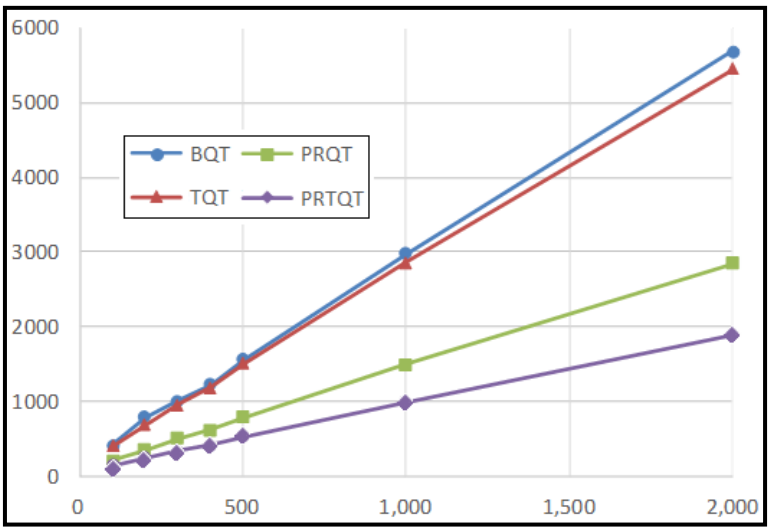

Figure 6. Total cycles of each protocol for Experiment B.

Table 3. N_C, N_I AND N_T FOR TAGS WITH SIMILAR EPC.

	BQT			TQT			PRQT			PRTQT		
n	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T
100	205	107	413	135	158	393	84	54	205	104	0	136
200	351	153	784	236	245	681	141	78	350	170	0	237
300	498	200	998	332	324	957	193	99	498	228	0	333
400	609	211	1220	410	373	1183	233	105	615	276	0	411
500	776	278	1555	526	485	1511	301	134	778	355	0	527
1000	1487	489	2976	990	864	2855	559	253	1493	655	0	991
2000	2852	854	5706	1898	1556	5454	1060	417	2852	1217	0	1899

collisions for $100 \leq n \leq 2000$. It is observed that our PRTQT also has the better performance. No matter what types of tags (random EPC or similar EPC) are tested, our protocols has the less N_T. For example, for $n = 2000$, our PRTQT has $N_T = 1899$, less than $N_T = 5706$, 5454, and 2852 in BQT, TQT, and PRQT, respectively.

5. Conclusion

In this paper, we proposed a PRTQT protocol to address the tag collision in RFID system. We adopt the parallel-subcarrier approach into TQT, so that tags can be identified in parallel subcarriers. Experiments reveal that the proposed PRTQT has better performance than BQT protocol, TQT protocol, and PRQT protocol.

Acknowledgements

This work was supported in part by Ministry of Science and Technology, under Grant 102-2221-E-259-009-MY2 and 103-2221-E-259-015.

References

[1] Finkenzeller, K. (2000) RFID Handbook: Radio Frequency Identification Fundamentals and Applications. John Wiley & Sons.

[2] Yang, C.N., He, J.Y. and Kun, Y.C. (2012) RFID Tag Anti-Collision Protocols. Chapter 7 in the book Advanced RFID Systems, Security, and Applications, IGI Global.

[3] Sarma, S., Weis, S. and Engels, D. (2003) RFID Systems and Security and Privacy Implications. *LNCS*, **2523**, 454-470. http://dx.doi.org/10.1007/3-540-36400-5_33

[4] Lee, S.R., Joo, S.D. and Lee, C.W. (2005) An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification. *Proceedings of MobiQuitous* 2005, Jul., 166-172.

[5] Zhou, F., Jin, D., Huang, C. and Hao, M. (2003) Optimize the Power Consumption of Passive Electronic Tags for Anti-Collision Schemes. *Proceedings of the* 5*th International Conference on ASIC* (*ASIC* 03), Oct., 1213-1217.

[6] Feng, B., Li, J.T., Guo, J.B. and Ding, Z.H. (2006) Id-Binary Tree Stack Anti-Collision Algorithm for RFID. *Proceedings of* 11*th IEEE Symposium on Computers and Communications* (*ISCC* 06), Jun., 207-212. http://dx.doi.org/10.1109/ISCC.2006.86

[7] Myung, J. and Lee, W. (2005) Adaptive Binary Splitting: A RFID Tag Collision Arbitration Protocol for Tag Identification. *Proceedings of* 2*nd International Conference on Broadband Networks* (*BroadNets* 05), Oct., 347-355.

[8] Cui, Y. and Zhao, Y. (2008) Mathematical Analysis for Binary Tree Algorithm in RFID. *Proceedings of* 67*th IEEE Vehicular Technology Conference* (*VTC* 08), May, 2725-2729. http://dx.doi.org/10.1109/VETECS.2008.596

[9] Hsu, C.H., Yu, C.H. and Huang, Y.P. (2008) An Enhanced Query Tree (EQT) Protocol for Memoryless Tag Anti-Collision in RFID System. *Proceedings of* 2*nd International Conference on Future Generation Communication and Networking* (*FGCN* 08), Dec., 427-432. http://dx.doi.org/10.1109/FGCN.2008.224

[10] Pupunwiwat, P. and Stantic, B. (2009) Unified q-ary Tree for RFID Tag Anti-Collision Resolution. *Proceedings of* 20*th Australasian Database Conference* (*ADC* 09), Mar., 49-58.

[11] Yang, C.N. and He, J.Y. (2011) An Effective 16-bit Random Number Aided Query Tree Algorithm for RFID Tag Anti-Collision. *IEEE Communications Letters*, **15**, 539-541. http://dx.doi.org/10.1109/LCOMM.2011.031411.110213

[12] Yang, C.N., Hu, L.J. and Lai, J.B. (2012) Query Tree Algorithm for RFID Tag with Binary-Coded Decimal EPC. *IEEE Communications Letters*, **16**, 1616-1619. http://dx.doi.org/10.1109/LCOMM.2012.090312.121213

[13] Mathys, P. and Flajolet, P. (1985) Q-ary Collision Resolution Algorithms in Random-access Systems with Free or Blocked Channel Access. *IEEE Transactions on Information Theory*, **31**, 217-243. http://dx.doi.org/10.1109/TIT.1985.1057013

[14] Yang, C.N., He, Y.C. and Wu, C.C. (2010) A Practical Implementation of Ternary Query Tree for RFID Tag Anti-Collision. *Proceedings of* 2010 *IEEE International Conference on Information Theory and Information Security*, Dec., 283-286. http://dx.doi.org/10.1109/ICITIS.2010.5689460

[15] Yeh, M.K., Jiang, J.R. and Huang, S.T. (2011) Parallel Response Query Tree Splitting for RFID Tag Anti-Collision. *Proceedings of* 40*th International Conference on Parallel Processing Workshops*, Sep., 6-15.

[16] Draft Protocol Specification for a 900 MHz Class 0 Radio Frequency Identification Tag, MIT Auto-ID Center, Feb. 2003.

[17] Pupunwiwat, P. and Stantic, B. (2009) Unified q-ary Tree for RFID Tag Anticollision Resolution. *Proceedings of Australasian Database Conference*, 49-58.