Hardware Based Design and Performance Evaluation of a Tree based RFID Anti-Collision Protocol

Laura Arjona, Hugo Landaluce, Asier Perallos
Deusto Institute of Technology (DeustoTech), University of Deusto
Bilbao, Spain
{laura.arjona,hlandaluce,perallos} @deusto.es

Sergio Martin
UNED University
Madrid, Spain
smartin@ieec.uned.es

***Abstract* - The growing concern in tracking, identification and localization systems has turn Radio Frequency Identification (RFID) technology into a mainstream in scientific research. In this technology, the phenomena known as the tag collision problem is becoming increasingly important, since it leads to a wastage of bandwidth, energy, and an increase in identification delay. Therefore, anti-collision protocols are required to mitigate collisions. Tree based protocols constitute one of the two main types of Radio Frequency IDentification (RFID) anti-collision protocols. Inside this group, Collision Tree (CT) protocol constitutes a representative strategy, since some other strategies are based on it. The focus of this work is to design and evaluate with VHDL the CT tag's chip design in order to extract the number of clock cycles required by the reader to identify a set of tags. A stability analysis is also performed. Simulations results show that the total number of clock cycles is linearly influenced by the tag's ID length.**

I. INTRODUCTION

Radio Frequency IDentification (RFID) is a wireless ubiquitous technology, where a spectrum of radio frequency is used to transfer the identification information between two communication devices: tags and readers. This technology is especially attractive in areas like health care, supply chain, and wireless sensor networks [1,2]. Tags are uniquely differentiated with an identification code (ID), which consists of a sequence of bits. Due to the shared nature of the wireless channel used by tags, these systems are prone to transmission collisions. A collision occurs when two or more tags transmit different information simultaneously. Collided tags must retransmit their ID until they are identified, resulting in a wastage of bandwidth, energy and an increase in identification delay and reader transmitted bits [3]. Anti-collision protocols are hence required to arbitrate these collisions.

Tree based protocols and Aloha based protocols are the two main types of anti-collision algorithms used in RFID systems [3]. Tree based protocols successively split collided tags into two or more subsets and the reader attempts to recognize each one of the subsets one by one. On the other hand, Aloha based algorithms divide time into frames and frames into slots so that tags randomly choose a slot to respond. The combination of tree based and Aloha based protocols results in hybrid protocols.

The Collision Tree protocol (CT) [4] constitutes a representative strategy inside tree based protocols, since it has been applied to other tree based strategies in order to improve the protocol performance [5]. CT arbitrates collisions by splitting colliding tags into two groups and by generating queries according to the first collided bit. The identification process consists on the following steps:

1. The reader generates the query $q_1q_2....q_m$, where $m \in [1,k]$ and k represents the length of the tags' ID. The query consists of a sequence of '0' and '1', constituting the tags' ID. These queries are taken from a stack which is initialized with the values '0' and '1'. The reader then pops a query from the stack and transmits it to the tags.

2. Considering the complete tag's ID $p_1p_2....p_k$, matching tags generate the response $p_{m+1}p_{m+2}....p_c....p_k$. It is important to note here that tags do not transmit the whole ID in every response to the reader, but only the following $k-m$ bits. This saves energy and time, since the number of bits transmitted by the tags is reduced.

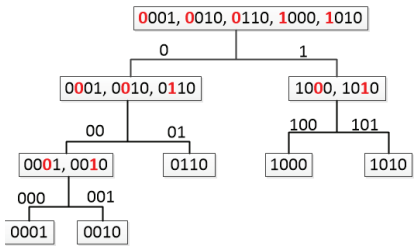

FIGURE 1 - CT IDENTIFICATION PROCESS EXAMPLE FOR FIVE TAGS WITH IDS{0001,0010,0110,1000,1010}

3. The reader generates two new queries based on the tags' responses: $q_1q_2....q_mp_{m+1}....p_{c-1}0$ and $q_1q_2....q_mp_{m+1}....p_{c-1}1$, where p_c corresponds to the first collided bit. That is, the reader concatenates the query previously sent with the tags' responses up to the first collided bit and add a '0' and a '1', thus generating two new queries. The generated queries are pushed onto the stack.

4. The reader pops a query from the stack, restarting the procedure (from step 1). This algorithm is repeated until the query stack is empty, which means that all tags have been identified.

An example of CT identification process is shown in Figure 1. Five tags are considered, with ID {0001,0010,0110,1000,1010}. The bit in bold red represents the first collided bit p_c.

In tags identification, a slot is considered idle when no tag responds to the query; if two or more tags respond, the slot is considered collided; and if there is only one response, the slot is defined as success. All internal nodes in the collision tree of CT correspond to collision slots, and all leaf nodes correspond to success slots. Therefore, there are at least two tags in an internal node and exactly only one tag in a leaf node. As a result, there is no empty nodes in the collision tree and no idle slots in the tags identification process using CT.

A hardware based design and performance evaluation of CT is presented in this paper, where the total number of clock cycles to identify a set of tags is extracted. Moreover, the effect of k over the number of clock cycles has been analyzed. To the best of our knowledge, this relationship has not been studied before.

Subsequently, the remainder of this paper is organized as follows: Section II presents the hardware design of CT; Section III extracts the total number of clock cycles and performs a stability analysis of CT; Section IV shows the simulation results and studies the effect of k over the number of clock cycles; and finally, Section V concludes the paper.

II. HARDWARE BASED DESIGN OF CT TAG'S CHIP

This section presents a hardware based design of a tag's chip which implements the CT protocol. The proposed design is based on a blocks diagram, where the state diagram block represents the main structure.

978-3-200-03735-9/15/$31.00 ©2015 IEEE

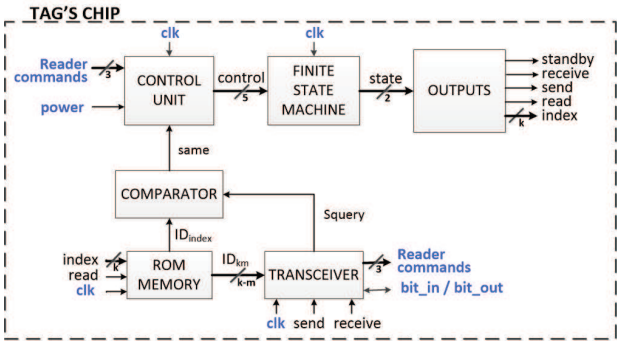

FIGURE 2 - CT PROTOCOL HARDWARE DESIGN BASED ON A BLOCKS DIAGRAM

2.1 Blocks diagram design

A global hardware blocks diagram of the tag side of the CT protocol is shown in Figure 2. It is composed of six main blocks. The main function of each input (I), output (O) and internal (INT) signal is presented in Table 1. All signals are active when taking the high value. The length in bits of each signal is indicated in Figure 2. Next, the main function of each block is briefly described. All synchronous blocks are activated with the positive edge of the clock signal.

- **Control unit**: Synchronous block. It is the brain of the system. It generates the *control* signal of the state machine. So it ultimately determines the output of the system. The input signals are the voltage power and the reader commands.

- **Finite state machine (FSM)**: Synchronous block that defines the tag's transitions from one state to another. The states represent the actions which the tags must develop in each clock cycle. The actions will be implemented in the Outputs block. The FSM will be explained in detail in Section 2.2.

- **Comparator**: Asynchronous block which compares the two input bit values (ID_{index} and $Squery$). It compares the bit from the query received by the reader with the corresponding bit of the tag ID.

- **ROM memory**: Synchronous block, where the ID of the tag is stored. In this system, only the ID is stored, although in real applications, much more information can be stored. The input signal *index* of this block indicates the bit which must be read from the tag's ID. It is a k-bits length signal, where only one bit is active at a time. The position of the active bit corresponds to the position of the ID bit which must be read. This memory provides serial and parallel access.

- **Transceiver**: Asynchronous block which receives the reader's commands and sends back the tag's response, depending on which signal is active (*send* or *receive*). It outputs the reader commands to the control unit and the query received to the comparator.

- **Outputs**: Synchronous block. Regarding the current state of the finite state machine, the Outputs blocks represents the action which the tag must develop at any point during the identification process.

A special attention must be paid to the internal signal *control*, since it plays a key role in the VHDL simulation performed later. This signal is the input of the finite state machine, and it ultimately determines the transitions between the different states. It is composed of five bits, and each bit, when active, involves the execution of an action. Provided that the implemented state machine is a Moore type, the output *state* is determined exclusively by the input *control*. This is in contrast to a Mealy machine, whose output values are determined both by its current state and by the values of its inputs. The *control* bits are active when taking the high value. The interpretation of each bit is provided next, where bit 0 represents the MSB:

TABLE 1 - I/O/INT SIGNALS OF THE HARDWARE DESIGN OF CT

Signal	Function (when active)
clk (I)	Clock signal. Activates the synchronous processes
power (I)	Power signal to activate passive tags
reader commands (I)	Identification commands transmitted by the reader {start(100), begin(010), end(001)}
bit_in (I)	Each one of the bits received corresponding to the reader's query
bit_out (O)	Bit from ID to transmit to the reader
standby (INT)	Low power mode. Wait for voltage power
receive (INT)	Receive and interpret data from reader
send (INT)	Transmit ID (partial or complete) to reader
read (INT)	Read ID (partial or complete) from ROM memory
index (INT)	Pointer to particular bit of the tag's ID
control (INT)	Controls the state machine transitions
same (INT)	Comparator's output. High value involves inputs match bit by bit
state (INT)	Current state of the finite state machine
ID_{index} (INT)	bit read from the position $index$ of the ID
ID_{km} (INT)	$k-m$ bits of the ID read from the ROM (from *index+1* to k)
Squery (INT)	bit received corresponding to the reader's query

- bit 0 (*start*): the reader initiates the identification process. Tags become powered

- bit 1 (*begin*): start receiving the reader's query

- bit 2 (*end*): finish receiving the reader's query

- bit 3 (*match*): output of the comparator block. When active, it indicates that the received bit from the reader's query and the corresponding bit of the tag's ID match.

- bit 4 (*finish*): finish transmitting the remaining bits of the tag's ID to the reader

2.2 Hardware design and simulation

Once the hardware block diagram is designed, the next step is to proceed to the hardware simulation of the proposed design. For this purpose, Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) has been selected for the design given its current relevance and considering a possible future implementation on a FPGA. In this section, a behavioral level is considered, representing the highest level of abstraction. It describes the system in terms of what it does (or how it behaves) rather than in terms of its components and interconnections between them.

The main block of the hardware design is the finite state machine, so it is studied in detail next. Figure 3 shows the implementation of the state machine.

Table 2 summarizes the main function of each state. The signal *state* codifies the four possible states with two bits. The value associated to each state is indicated in parentheses in the table.

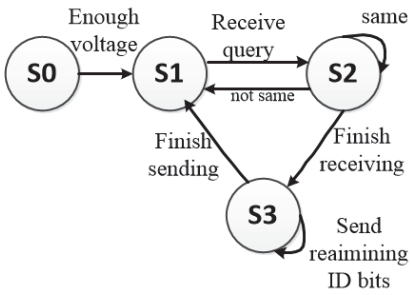

FIGURE 3 - CT PROTOCOL FINITE STATE MACHINE

TABLE 2 - CT TAG STATES FUNCTION

State	Function
S0 (00)	Low power mode
S1 (01)	Waiting for query
S2 (10)	Receiving query
S3 (11)	Sending ID

S0 constitutes the initial state. Initially, all tags in the system are set to this state. It is important to note that if the tag becomes underpowered at any point during the identification process, it jumps back to this initial state, waiting to be powered again. Once the tag is powered, it transits to state S1. The tags remains in this state until it receives the reader's query. Once received, it goes to S2. Here, the tag compares the query received with the corresponding bit of its ID. If both bits do not match, the tag goes to S1. Otherwise, the tag keeps receiving the remaining bits of the query. Once all bits of the query have been received, it transits to S3. In the final state, as CT indicates, the tag must send the remaining bits of the ID (those which have not been compared with the reader's query). Once all bits have been transmitted, the tag returns to S1 and the procedure repeats. If only one tag reaches state S1 from S3, it means that there is only one tag which completely matches the reader's query. Therefore, that tag is successfully identified.

The remaining blocks shown in Figure 2 have been implemented using behavioral-based statements. Next, the proposed design is compiled. For this purpose, the software Quartus II Web Edition has been applied. The protocol hardware design is validated for one example tag. A simulation profile is defined in order to manually introduce the reader's commands and the clock signal of both reader and tags. It is assumed that the reader's and tag's clock are perfectly synchronized. Besides, all signals are checked with the rising edge. In order to facilitate the visualization of the results, it is assumed that one symbol, data-0 or data-1, is transmitted per clock cycle.

A simulation scenario with one reader and two tags is presented: tag A (ID=0001) and tag B (ID=0010). Simulation results extracted from the Quartus simulation correspond to tag A. Results are shown in Figure 4. According to Table 1, only the input and outputs signals are showed. Additionally, the signal *state* is represented, given its high influence over the reading process. The signal reset is also implemented to restart the process.

III. EXTRACTION OF CLOCK CYCLES

In this section the total number of clock cycles required to identify a whole set of tags is extracted. The reader implements three main commands to arbitrate the reading process: $start$, $begin$ and end . Theses signals are grouped together and named $reader\ commands$. Their value is specified in parentheses in Table I.

In CT, the depth of the tree is defined as $d = log_2(n)$ [6], where n represents the total number of tags in the set to be read. To start the identification process, the reader transmits a $start$ command. In order

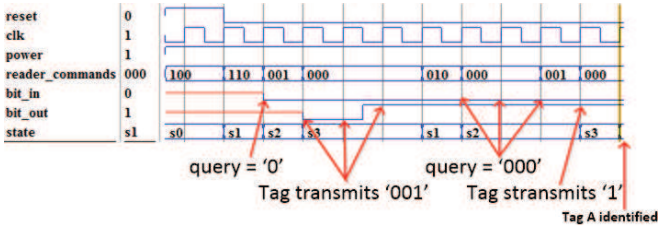

FIGURE 4 - SIGNAL DIAGRAM FROM SIMULATION EXAMPLE OF CT. TAG'S ID = 0001

to inform the tags of the start and end of the received data, the data are enclosed with a *begin* for indicating the start of transmission and an *end* to indicate the end. Each arrival at a node requires the tag to experience $[k(\text{either receiving or sending data}) + 2(begin) + 2(end)]$ clock cycles. The arrival at a leaf node implies that a tag is successfully identified. Thus, the total number of clock cycles required is given by:

$$\begin{aligned} clk_{t\_CT} &= 2 + (k+4)(2^0 + 2^1 + ... + 2^{d-1} + 2^d) = \\ &= 2 + (k+4)(-1+2n) = \\ &= 2 + n(2k+8) - (k+4) = \\ &= n(2k+8) - (k+2) \end{aligned} \tag{1}$$

The term $(-1+2n)$ is obtained as the sum of the first $d+1$ terms of the geometric progression defined by $a_1 = 2^d = n$, $a_n = 1$ and $r = \frac{1}{2}$. For k constant, a linear relationship between the total number of clock cycles to identify a set of tags and the number of tags in the set is obtained.

3.1 Stability

An anti-collision protocol is stable if its performance is only dependent on the number of tags to be identified (condition 1) and if the average performance for one-tag identification converges to a constant (condition 2) [4]. This section examines the stability of the protocol regarding the total number of clock cycles required to identify a set of tags. Typically, the tag ID length is fixed for a particular RFID system. Therefore, a scenario with k constant is considered.

Condition 1.

Examining (1), it is clear that if k is assumed constant, the total number of clock cycles to identify the whole set of tags is only dependent on the number of tags.

Condition 2.

The average number of clock cycles to identify one tag is examined in (2):

$$clk_{1\_CT} = clk_{t\_CT}/n = \frac{n(2k+8) - (k+2)}{n} \tag{2}$$

Further

$$lim_{n\to\infty} clk_{1\_CT} = 2k + 8 \tag{3}$$

It follows that the total number of clock cycles to identify one tag converges to the constant $2k + 8$.

Since both conditions are satisfied, the total number of clock cycles of CT is stable.

IV. PERFORMANCE EVALUATION

This section presents the results of the simulation experiments using Matlab R2013a software. The performance of CT is been evaluated regarding the total number of clock cycles. The proposed simulation defines a scenario with one reader and a varying number of tags from 100 to 1000 tags with a step size of 100. It is assumed that the transmission channel is ideal [4, 5]. The simulation responses are averaged over 1000 iterations for accuracy in the results. It is also assumed that the identification procedure ends when the query stack is empty, which

means that all tags have been identified In order to extract the number of clock cycles for both presented protocols, it is assumed that one symbol, data-0 or data-1, is transmitted per clock cycle. This assumption must be carefully taken into account. Moreover, it is considered that one symbol corresponds to one bit. A value of $k = 64$ bits is evaluated.

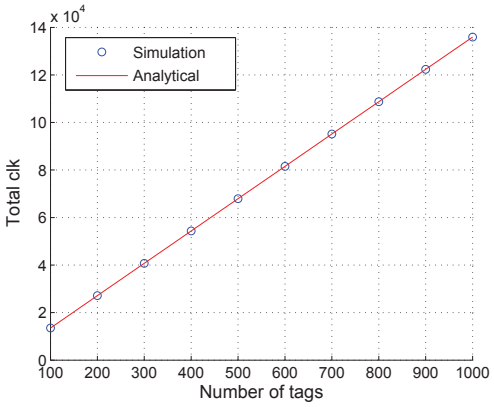

FIGURE 5 - TOTAL NUMBER OF CLOCK CYCLES FOR CT AS A FUNCTION OF THE TAG POPULATION. $k = 64$ BITS

Simulations results are shown in Figure 5. Analytical results correspond to (1). Clearly, simulation and analytical results perfectly match. An increase of the total number of clock cycles with the tag population is observed, with a linear behavior. This result was expected by analyzing (1).

4.1 Effect of ID length over the number of clock cycles

This section examines the effect of the ID length over the total number of clock cycles. For this purpose, it is considered an ID length of k bits and a length of ak bits, where a is a positive integer. The relationship between these two scenarios is the following:

$$\frac{clk_{t\_CT}(ak)}{clk_{t\_CT}(k)} = \frac{n(2k+8)-(k+2)}{n(2ak+8)-(ak+2)} = \\ = \frac{k(2n-1)+(8n-2)}{ak(2n-1)+(8n-2)} \quad (4)$$

In the current Standard in RFID (EPCglobal) [7], k is typically 64, 96 or 128 bits. Since the number of tags to be read is frequently much higher than k, it can be assumed that $n >> k$. Therefore, it can be assumed that $k(2n-1) >> 8n-2$. This leads to the following expression:

$$\frac{clk_{t\_CT}(ak)}{clk_{t\_CT}(k)} \approx \frac{ak(2n-1)}{k(2n-1)} = a \quad (5)$$

$$clk_{t\_CT}(ak) \approx a \times clk_{t\_CT}(k) \quad (6)$$

Next, the protocol is evaluated for five different k values (64,96,128,160,192). Simulation results are shown in Figure 6. Taking 64 bits as the reference length, the increase produced in the total number of clock cycles for the different scenarios is compared in Table 3.

TABLE 3 - INCREASE IN THE NUMBER OF CLOCK CYCLES WITH RESPECT TO CT WITH $k = 64$ BITS

	ID length (k bits)			
	96 (a=1.5)	128 (a=2)	160 (a=2.5)	192 (a=3)
Increase in clk	1.49	1.97	2.46	2.94

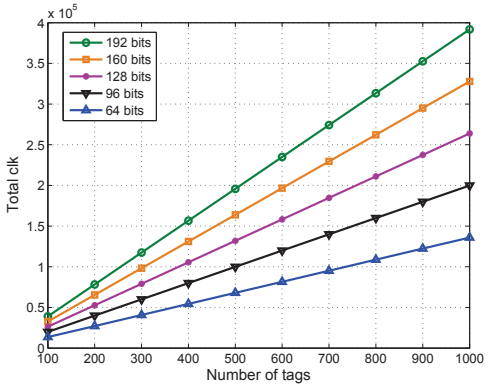

FIGURE 6 - EFFECT OF k OVER THE TOTAL NUMBER OF CLOCK CYCLES FOR CT AS A FUNCTION OF THE TAG POPULATION

From Table 3, it can be appreciated that the total number of clock cycles to identity a set of tags is directly proportional to its ID length. The approximation made in (6) is therefore justified.

V. CONCLUSION

This work has presented a hardware based design of a tag's chip implementing the CT protocol. The total number of clock cycles to identify all tags in a set has been extracted and evaluated. It has been demonstrated that the total number of clock cycles of CT is stable. Furthermore, this factor has been shown to be directly proportional to the tag ID length.

As a concluding remark, future work could lead to the design and implementation of the reader's chip. To complement this work, the physical implementation of the protocol on a real FPGA would be of great relevance, providing more realistic results.

REFERENCES

[1] A.A.N. Shirehjini, A. Yassine, and S. Shirmohammadi. Equipment Location in Hospitals Using RFID-Based Positioning System. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1058–1069, Nov 2012.

[2] Z. Xiong, Z.Y. Song, A. Scalera, F. Sottile, R. Tomasi, and M.A. Spirito. Enhancing WSN-Based Indoor Positioning and Tracking through RFID Technology. In *Fourth International EURASIP Workshop on RFID Technology (EURASIP RFID)*, pages 107–114, September 2012.

[3] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, Inc., New York, NY, USA, 2 edition, 2003.

[4] X. Jia, Q. Feng, and L. Yu. Stability Analysis of an Efficient Anti-Collision Protocol for RFID Tag Identification. *IEEE Transactions on Communications*, 60(8):2285–2294, August 2012.

[5] Y.C. Lai, L.Y. Hsiao, H.J. Chen, C.N. Lai, and J.W. Lin. A Novel Query Tree Protocol with Bit Tracking in RFID Tag Identification. *IEEE Transactions on Mobile Computing*, 12(10):2063–2075, Oct 2013.

[6] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min. Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design, 2004. ISLPED '04.*, pages 357–362, Aug 2004.

[7] Radio Frecuency Identity Protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz, November 2013.