Collision Resolution in ISO 18000-6c Passive RFID

Yuan Sun, Peter J. Hawrylak, Zhi-Hong Mao and Marlin H. Mickle

RFID Center of Excellence
Electrical and Computer Engineering Department
University of Pittsburgh

***Abstract*—** According to the ISO 18000-6c passive RFID standard (EPCglobal Gen 2), in a tag intensive environment, when multiple tags receive the reader Query command and respond simultaneously, the reader may receive multiple responses giving what is termed a collision or a collision signal. This paper reports a method for resolving the two tag collision condition in real time thus not requiring any additional input by the reader. A reader has been designed using a National Instruments set of equipment which includes an FPGA-based software defined reader. The collision signal is obtained from the data acquisition system and processed by the FPGA in real time. This is a straightforward algorithm that can be implemented in silicon or programmed in a microprocessor to replace the current FPGA.

***Index Terms*—** ISO18000-6c RFID, collision.

I. INTRODUCTION

Because passive RFID systems make use of the electromagnetic spectrum, they are relatively easy to jam using energy at the right frequency which occurs in a dense tag environment. Although this would only be an inconvenience for consumers in non critical situations (e.g. in stores with longer waits at the checkout), it could be disastrous in other environments, such as hospital emergency centers and in a military field of operation.

The collisions in UHF passive RFID communications lie in two major categories: the reader collision and the tag collision. Reader collisions occur in a reader intensive environment when the signals from two or more readers overlap in time and frequency. Tag collisions occur in a tag intensive environment when multiple tags are present in the transmitting field of the reader. In such situation, tags may respond to the reader Query command simultaneously causing the reader to fail to decode the received signal, which is the result of collision.

This paper reports the ability to resolve the tag collision in order to improve the efficiency of ISO 18000-6c RFID systems [1]. Currently, to address the tag collision problem, multiple anti-collision protocols enabling the passive RFID tags to take turns in transmitting to a reader have been developed. Generally, there are two types of anti-collision protocols in common use based on time division multiple access (TDMA): One is the dynamic framed slotted Aloha, the other is the Binary Tree scheme [2]. ISO 18000-6c RFID systems use dynamic framed slotted Aloha. The dynamic framed slotted Aloha is a probabilistic method which can decrease the probability of collision occurrences significantly, but collisions cannot be completely avoided. Therefore, in the worst case when two or more tags in an inventory round randomly select the same time slot to respond (e.g. when the reader requests all tags to respond), the reader may receive an unrecognizable (un-decodable) mixture signal as a result of a collision. Accordingly, the reader read rates will be degraded due to this collision situation in the communication process. Therefore, it is intuitive to increase the system read rate more from pure collision avoidance by recovering the original tag signals from the collision at the reader. The philosophy of this concept is to resolve the collision after it occurs rather than trying to avoid it. It is also natural to combine the use of an anti-collision mechanism and the resolution of the collision to increase the efficiency of the reader-tag communication.

In addition, ISO 18000-6c passive RFID tags rely on limited energy harvested from the interrogator carrier wave rather than an internal power supply to perform logic functions and backscatter signals. Although this feature makes passive tags simple and inexpensive to deploy and maintain compared to active tags, the passive

RFID tags are subject to more critical communication timing constraints: A reader which fully conforms to the standard, must realize the tag inventory round including a three step handshake in order to make the tag transit into the data access states, which allows for tag memory access. The effective turn-around time between each of the three steps lies in the range of from 31.25μs in the worst case when the tag back link frequency (BLF) reaches 640kHz to 0.5ms in the best case when BLF is as low as 40kHz, which implies that the interrogator or any test platform conforming to the ISO 18000-6c standard must complete the signal decoding and command assembly in real time. If at least one of the tag responses can be separated and decoded in the standard specified time constraint, one inventory round including the three step handshake can be performed. Furthermore, if more than one tag response can be resolved from the collision, inventory rounds can be performed in parallel which leads to a dramatic increase in system read rates.

II. TECHNICAL BACKGROUND

A. Link Timing of ISO 18000-6c

The ISO 18000-6c standard specifies two categories of tag states: inventory states and memory access states. In order to access the memory content in the tag, a reader shall complete an inventory round to make the tag pass all the inventory states until entering the memory access state. The challenge lies in the fact that each step of the inventory round must be completed in a soft real-time, T (i.e. the Turn-around time between the tag response and the following reader command), which requires the reader to complete the decoding of the tag response and then send out the next command in the inventory round within this turnaround time. If any transition step in the inventory round fails to complete in time, T, the tag will transit back to the Ready state (the initial state after power up).

Corresponding to each state in the inventory round, there exists a three step handshake. At the first step of the handshake, the interrogator assembles and sends out a Query command; the tag chooses a random time slot to backscatter its 16-bit random number (RN16) after receiving a Query command, and then transits from the Arbitrate state to the Reply state. In the next step, the reader decodes the tag backscattered random number and attaches it to the command header of an acknowledge command ACK, and then transmits the ACK command within the turn-around time, T. The tag receives this ACK command and responds with its ID (the tag PC and EPC number) including a 16-bit CRC code. The tag then transits into the Acknowledged state. At the last step, the interrogator receives the tag response and sends out a Req_RN command with the previous tag backscattered 16-bit random number and 16-bit CRC over the command within the same turn-around time T in Step 2 to notify the tag entering into the memory access state (Open or Secure state). The tag receives this Req_RN command and backscatters a Handle (a special 16-bit code).

B. Anti-Collision in ISO 18000-6c

To start an inventory round for the tags in the field, the reader needs to send a Query command as the first step. The anti-collision dynamic framed slotted Aloha algorithm is realized by specifying a value ranging from 0-15 to the 4-bit Q field in the Query command. Upon receiving the Query command, the matching tags pick a random number in $[0, 2^{Q}-1]$ to load into the slot counter. If a tag, in response to the Query command, loads its slot counter with zero initially or after counting down the original random number selected, it responds with a 16-bit random number (RN16). The tag collision occurs when multiple tags in the reader field load their slot counter with zero and thus respond to the Query command simultaneously. A form of collision control can be realized by specifying a large value for Q to reduce the probability of collisions, in the extreme case when Q equals 16 the probability for an N tag collision is $(1/65536)^{N-1}$.

C. ISO 18000-6c Tag Baseband Encoding

According to ISO 18000-6c standard [1], tags shall encode the backscattered data as either FM0 baseband or Miller modulation of a subcarrier at the data rate (BLF). The reader commands the encoding choice, and both FM0 and Miller are bi-phase space encoding. Fig. 1 shows the basis functions of FM0. FM0 inverts the baseband phase at each symbol boundary; a data-0 has an additional mid-symbol phase inversion. Fig. 2 shows the basis function of Miller encoding. Baseband Miller inverts its phase between two data 0's in sequence. Baseband Miller also places a phase inversion in the middle of a data 1 symbol. When employing Miller encoding, the tag modulates a square wave shaped subcarrier by the Miller baseband. The Miller sequence contains exactly two, four or eight subcarrier cycles per bit [1].

Generally, the symbol of both FM0 and Miller can be categorized into formations as shown in Fig. 3. Formation 0 features an edge transition in the middle of the symbol, while there is no edge transition in the Formation 1 symbol.

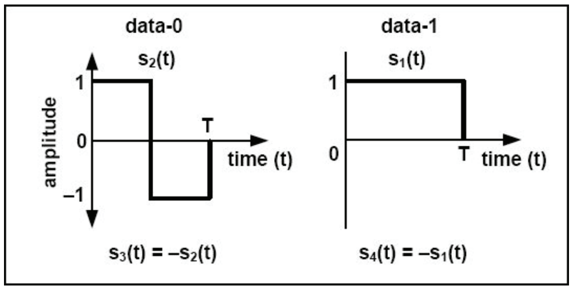


Fig. 1. FM0 baseband basis function

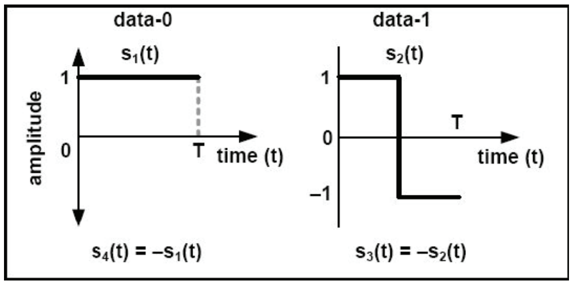


Fig. 2. Miller baseband basis function.

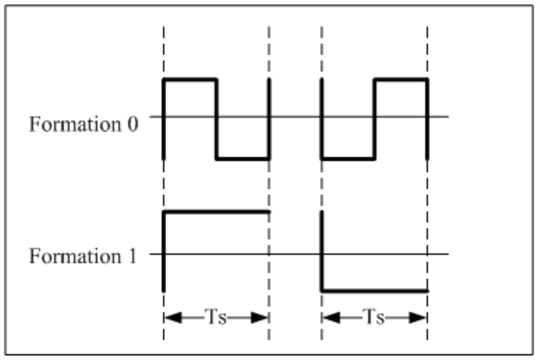


Fig. 3. Generalized tag baseband formation.

III. SOLUTION METHODOLOGY

The focus of this research is the resolution of the two tag collision situation. The collision can be forced for experimental purposes by letting the reader set the Q field in the Query command to a value of zero, and the two tags will load 0 into their slot counters and respond with their 16-bit random numbers (RN16) simultaneously. The two tag responses are linearly superimposed forming the reader received baseband collision signal after the RF and IF down converting resulting from the superposition of the two symbol formations of either FM0 or Miller encoding as discussed in the last section. Due to the attenuation of different propagation paths and the fact that the Phase Locked Loop (PLL) in the reader receiver circuitry can only lock to one of the tag backscatter carrier waves or the reader transmitting carrier wave depending on the relative strength of the three signals, the downconverted two tag responses are normally different in magnitude. In addition, due to possible analog variations in the tag chip front ends (silicon), a phase shift (delay) exists between the two tag responses even when the two tags are from the same manufacturer. (Detailed analysis of the two tag collision waveform characteristics will be introduced later) Therefore, the resulting collision baseband signal violates the standard specified FM0 or Miller encoding, which causes the functional failure in the reader decode circuitry.

Without any collision resolution, the reader can only decode tags responding in different time slots, while two tags can share a common time slot provided their separate response can be recovered and extracted from the collision signal. For two tags in the field on a reader where Q=4, the probability of two tag responses colliding is (1/16) × (1/16). The probability of three tags colliding is (1/16) × (1/16) × (1/16), which reduces the collision probability by more than an order of magnitude. Therefore, successful resolution of the two tag collision can significantly improve system data access efficiency.

As discussed in last section, the ISO 18000-6c standard mandates a real-time turnaround time specification. The collision resolution needs to be completed before this time out. Therefore, two strategies can be considered for the timing of the collision resolution as shown in Fig. 4. The first strategy starts the resolution after the complete acquisition of the collision, while the second strategy performs an on-line resolution one symbol after the other. The time available for collision resolution in the first strategy is less than the standard specified turnaround time because after the resolution, the reader needs time to decode the recovered tag response. In comparison, the time available for the resolution equals the turnaround time plus the tag signal duration, and the reader can simultaneously decode the recovered symbols one by one. The second strategy is similar to "divide and conquer", and thus saves processing time.

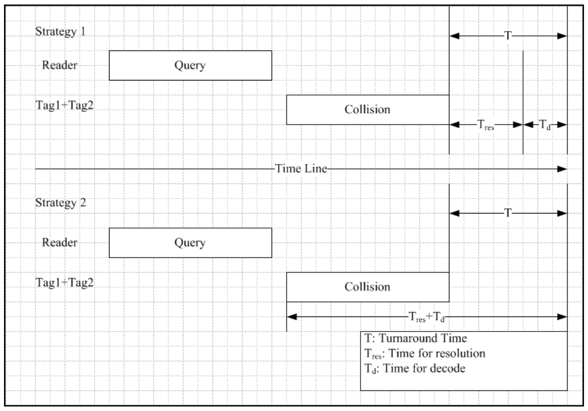


Fig. 4. Collision resolution timing strategy.

A. The Reader and Data Acquisition

An ISO 18000-6c RFID tag conformance test platform has been developed as a tool for this research. This platform is an ISO 18000-6c reader capable of, analyzing the acquired tag backscatter, and then checking the tag response baseband waveform against the standard. This platform serves as a data acquisition stage to obtain the tag response signal and the collision signal, which are the source signals in this research.

When selecting the hardware device for this platform, general purpose microprocessors and available commercial readers are both considered as candidates for ISO 18000-6c realization. General purpose microprocessors could be utilized to perform the communication, but they usually take multiple cycles to finish one instruction, and the total number of machine language instructions varies with the efficiency of user programming algorithms and the compilers. Therefore, they are difficult to time accurately and sometimes inefficient in communication timing control and thus not necessarily be the best device to be selected. Available commercial readers are alternative platforms, however their architectures are fixed at manufacture and most of those devices require users to familiarize themselves with the vendor specified command format in order to manipulate the reader which burdens the user and degrades the RFID data access efficiency. In addition, there is no evidence that the commercial readers can provide convenient forward compatibility while the passive tags advance in technology. Upgrading those readers to support the new features of tags usually leads R&D costs if a hardware modification is required. Unlike microprocessors, FPGAs are timing accurate in their Hardware Description Language (HDL) programming. Compared to the fixed structure of commercial interrogators, a reconfigurable platform can be easily customized to perform in-depth functions. Therefore, in light of the defects of the possible solutions, it is intuitive to utilize the re-programmability of an FPGA baseband processor to build a flexible interrogator for ISO 18000-6c passive RFID tags to evaluate the proposed collision resolution algorithms. In addition, a fixed point on-line digital signal processing algorithm has been performed by the FPGA with satisfactory speed to provide signals of high quality and address the communication conflicts in a dense tag environment.

To these ends, an FPGA based software defined radio architecture is employed to implement ISO 18000-6c standard for the data conformance test platform. It features: 1) a real-time FPGA baseband, 2) a software controllable IF and RF front-end, and 3) a host PC based GUI control panel. The development tool set includes National Instruments (NI) LabVIEW 8.5 (for software programming and test front panel control), and LabVIEW FPGA module 8.5 (for FPGA baseband hardware programming). LabVIEW is distinctly suited for FPGA programming because it clearly represents parallelism and data flow. With the LabVIEW FPGA Module, custom measurement and control hardware requiring high-speed hardware reliability and tight determinism can be simulated and synthesized without low-level hardware description languages or board-level design. The LabVIEW compiler automatically translates the LabVIEW graphic code into low-level HDL code.

B. Platform Architecture

The reconfigurable software defined radio test platform consists of two major parts: The hardware, which includes the signal baseband, the intermediate frequency and the RF front end. The software is running on a PC, which controls the hardware and analyzes the backscattered signal.

C. Platform Hardware

The architecture of the platform hardware connection is shown in Fig. 5. The 2-way signal flow includes the transmitter side and the receiver side. On the transmitter side, the software on the host PC selects and sends out the user specified command to the FPGA baseband. The baseband assembles the received binary command using PIE encoding according to the standard and then passes the data into the intermediate frequency band (IF). The IF stage consists of a DA9857 14-bit quadrature digital upconverter and a DA6654 14-bit downconverter. With the built-in Numerical Controlled Oscillator (NCO), the IF upconverter modulates the baseband data in DSB-ASK at the IF center frequency of 25MHz. The signal baseband (Xilinx Virtex-II Pro XC2VP30 FPGA) and the IF stage are together in the NI PCI-5640R Software-Defined Radio transceiver board. The 25MHz IF ASK signal is sent into the RF front end tuned to 915MHz for RF stage modulation. The RF front end consists of an NI 5610 RF upconverter and the NI5600 RF downconverter, which are connected by the NI PXI bus. On the receiver side, the tag backscattered signal passes through the 2-stage ASK demodulation in the RF and IF bands, and then enters into the baseband. For comparison purposes, an Agilent E4443A real time spectrum analyzer is employed as an auxiliary monitor of the RF communication process. The FPGA decodes the signal using FM0 or Miller encoding and sends the received signal as well as the decoded binary data back to the host PC for software offline analysis.

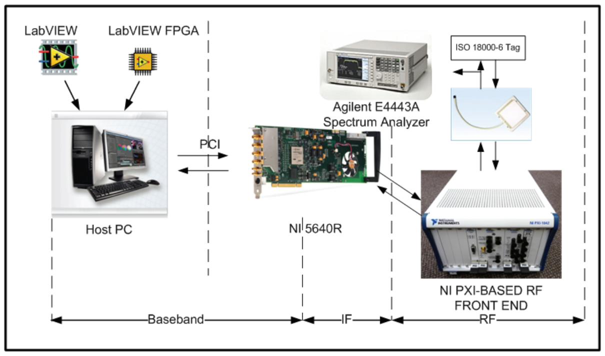


Fig. 5. Platform hardware architecture.

D. Platform Software

The platform software running on the host PC handles four major functions:

1. The test control panel: Using this panel, the user can select the command to send and set the command parameters. A set of the physical layer (PHY) features such as modulation depth, the length of Tari value, RTcal, and TRcal can be controlled by inputting values in corresponding text fields on the front panel. In addition, the IF and RF stage hardware are both conFig.d by specifying the expected center frequency, the output power level, and the sampling length/accuracy.

2. The signal analysis: The host PC software receives the demodulated baseband I/Q signals, and displays the RF envelope, I vs. Q waveform, the constellation diagram, and also calculates the spectrum.

3. The offline conformance check: The received RF envelope and calculated spectrum can be stored by the user optionally for offline conformance checking in a separate offline analysis module. The pulse width of the data stream and the integration of power in the specified bandwidth are checked against the standard.

4. The interface for remote control: By using the LabVIEW supported VISA interface, the host PC can be accessed by remote machines through the RS-232 (serial) connection or Ethernet connection in a Clint-Server manner. This allows a remote terminal to send a command to the local host PC, which connects to the baseband and RF front end to perform the test.

IV. STANDARD REALIZATION AND DATA ACQUISITION

A. Standard Realization

The FPGA baseband realizes the logic function of an ISO 18000-6c conformed reader. The FPGA baseband consists of four parts to realize the ISO 18000-6c standard: the ASK modulator, the real-time DSP unit, the signal decode module for both the FM0 and the Miller encoded signal, and the processing unit for the previously mentioned three step real-time handshake.

On the transmitter side, the FPGA assembles the reader commands. It receives the binary command from the host PC control panel, and then modulates the data stream using ASK after performing the encoding. The modulation of the ASK signal is shown in Fig. 13. The encoded data are sent into two separate quadrature channels − I and Q in the IF stage. The ASK modulation is realized by setting the magnitude multiplier in each channel as 1 when the incoming data bit is "1" in binary, and setting the multiplier as zero when the data bit is "0". The DSB-ASK and SSB-ASK modulation manner can be selected by controlling the strobe of the Q channel.

However, the square wave shaped command bit stream generated by the FPGA contains a theoretically unlimited bandwidth. If it is passed into the IF upconverter without bandwidth limiting, the so called Gibbs phenomenon [7] occurs, resulting in significant over/under shoot at the data edges which degrades the output signal. The over/under shoots of the output command usually exceed the maximum tolerance specified in ISO 18000-6c standard and make the output command signal invalid for the tag in test. Therefore, real-time DSP work is necessary for guarantying the quality of generated interrogator signals. To eliminate the Gibbs phenomenon, a low pass filter is placed between the FPGA baseband and the IF stage in order to limit the bandwidth of the output baseband signals below the cut off frequency of the interpolation filter in IF upconverter. With the LabVIEW built-in Digital Filter Design (DFD) toolkit, the tap coefficients for an 8-order Bessel FIR low pass filter are generated and the magnitude/phase responses are displayed in Fig. 6.

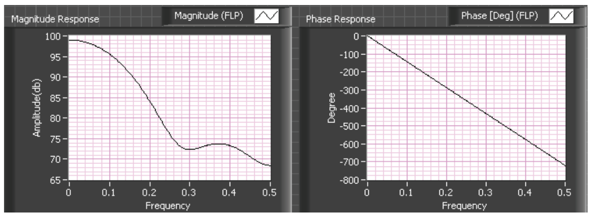


Fig. 6. FIR filter magnitude & phase responses.

As shown, the bandwidth of the baseband signal is limited to 30% of the sampling frequency, which is 10% below the cut-off frequency of the interpretation filter in the IF stage. The coefficients are selected to be symmetric to the center tap in order to ensure a linear phase response. Fig. 7 shows the realization of the filter in the LabVIEW FPGA Module and the corresponding effect. Comparing the baseband outputs, it can be shown that the under/over shoots in command signals are significantly quenched after the smoothing.

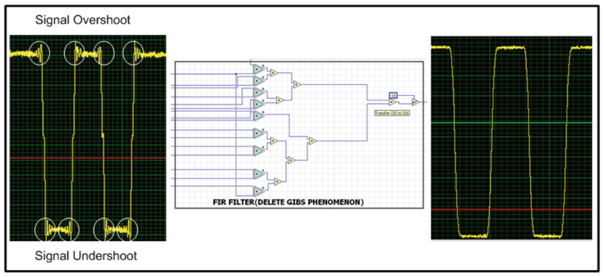


Fig. 7. Smoothing with an 8-order FIR filter.

On the receiver side, the FPGA baseband receives the IF demodulated tag backscattered signals and recovers the binary information in this FM0 or Miller encoded bit stream using a decode module. In the decode process, the clock recovery method is critical to accuracy. In light of the fact that the FM0 and Miller encoding are both self-clocking bi-phase codes, the decoding can be based on edge detection and pulse width measuring. In the decode module, the edge detector scans the data series, and sends out a notification signal as well as the distance between current and previous edges (i.e. the pulse widths) to the decode processor. The decode processor includes a 2-state Finite State Machine (FSM), which arbitrates the corresponding logic value for the data bit based on the inputs from the edge detector and the previous data bit.

The architecture of the handshake processing unit is shown in Fig. 8. In Step 1 of the three step handshake, the decode module notifies the decode module every time it detects an edge in the tag response. The decode module decodes the incoming tag response and sends out a handshake signal to the processing unit once the decode finishes for the tag backscattered random number. In Step 2, the decoded 16-bit binary random number is stored in the block memory in the FPGA and passed into an FIFO for command assembly. The ACK command header is sent into the same FIFO, and by doing this the random number gets attached to the header. The FIFO is connected to an optional delay module which allows for turn-around time adjustment. In Step 3, the assembly of the Req_RN command follows the same procedure as in Step 2 except that the FIFO is also connected to a CRC16 generator for the CRC attachment. Handshake signals are also sent after the command assembly finishes in each step to notify the ASK modulator to modulate the assembled command for transmission. Fig. 9 shows the acquired sampling of the three step handshake.

The synthesis result of the platform provided by Xilinx XST shows that the design utilizes 86% of the total slices, 73% of the Block RAMs and 403 user I/Os among the total 556 I/O blocks of the Virtex-II Pro XC2VP30 FPGA. Consequently, the design takes reasonable advantage of the device resources.

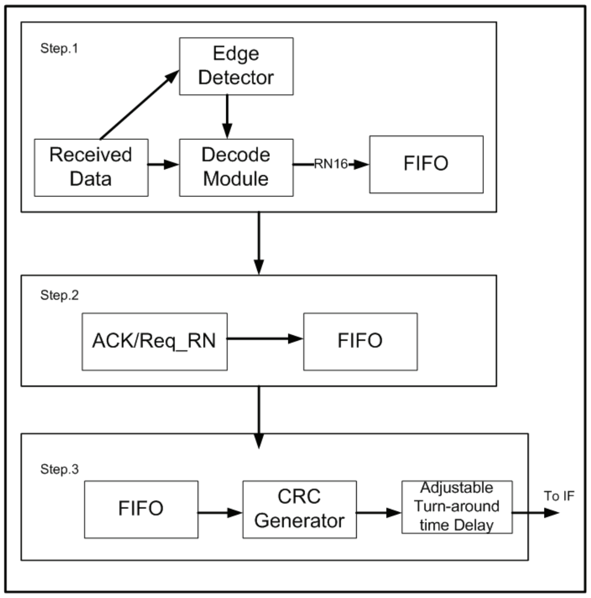


Fig. 8. Handshake processing unit.

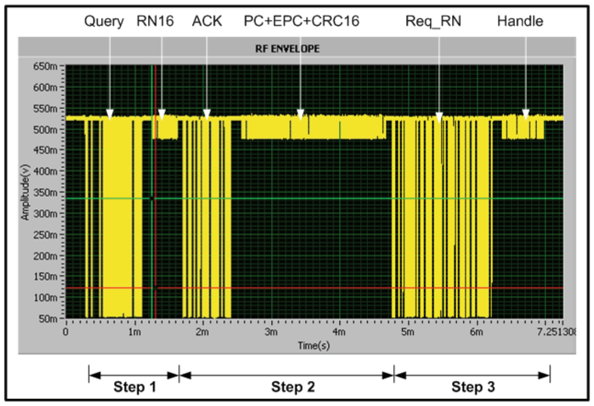


Fig. 9. Acquired inventory round.

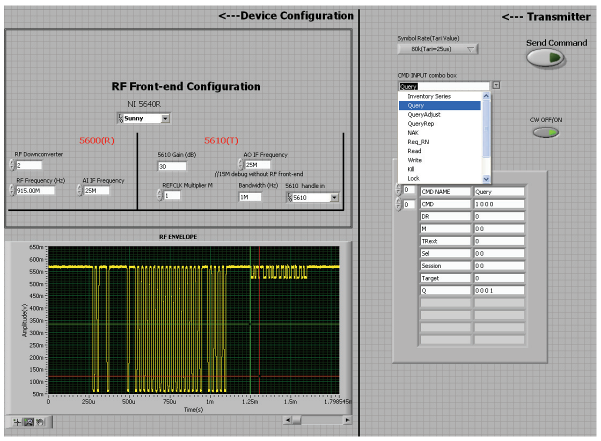
B. Data Acquisition with the Platform

As introduced previously, the test stimulus reader command as well as hardware details can be conFig.d by the LabVIEW Graphic User Interface (GUI) test control panel and the acquired tag response can be displayed by the signal analysis module.

In the test control panel, the user gets full control of the communication PHY and Media Access Control (MAC) configuration. The output power level and the center frequency of the RF front-end can be changed and reset by inputting the expected value in the corresponding number fields.

In addition, the center frequency of the IF stage and the ASK modulation depth can also be configured. The command Tari, TRcal, and RTcal value scan be input by the user. In addition, the accuracy and length of the FPGA acquisition can be customized. The acquisition sampling rate varies from 2MHz to 25MHz (by default), which allows for a measurement resolution up to 0.04μs (25MHz sampling rate) in spacing between adjacent sampling points. The user can easily select the command type to send, and the corresponding pre-stored default command parameters are loaded into an editable combo box automatically. The command parameter in the command combo box can be changed if necessary to send out a customized command. Fig. 10 shows the test control panel. As shown, the RF front-end and the IF stage are tuned at 915MHz and 25MHz, separately; The Tari value is set as 25μs, the modulation depth is 90% and the sampling frequency is 25MHz; A Query command has been selected from the command menu, and its Q field is changed from “0000” to “0001”. After clicking the “Send” button, the corresponding RF envelope of the reader command and tag response are captured by the platform hardware and displayed in the virtual oscilloscope. As shown, all the other ISO 18000-6c commands can be selected by the user from the combo box.

The waveform analysis module accepts the captured RF envelope and calculates the spectrum. The I vs. Q waveform as well as the constellation diagram are also generated from the acquired sampling data. The decoded message of the tag response is simultaneously displayed. Fig. 11 shows an excerpt of the front panel of the waveform analysis module. Accompanying the RF envelope of the three step handshake are the decoded binary message for the RN16 after Query command, the PC+EPC+CRC after the ACK command and the Handle+CRC16 after the Req_RN command.

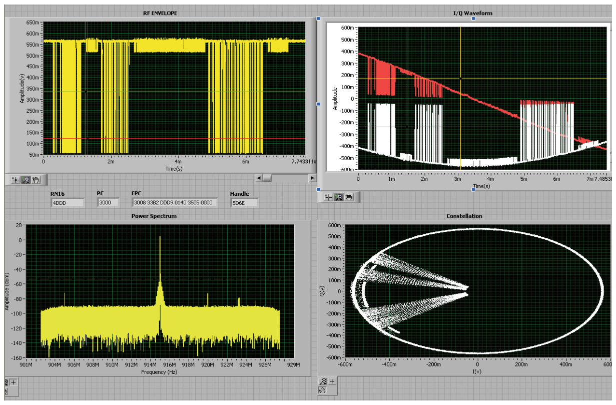


Sunny

RF Downconverter: 2
RF Frequency (Hz): 915.00M
AI IF Frequency: 25M
5610 Gain (dB): 30
AO IF Frequency: 25M
//15M debug without RF front-end
REFCLK Multiplier M: 1
Bandwidth (Hz): 1M
5610 handle in: 5610
Symbol Rate(Tari Value): 80k(Tari=25us)

CMD NAME	Query
CMD	1 0 0 0
DR	0
M	0 0
TRext	0
Sel	0 0
Session	0 0
Target	0
Q	0 0 0 1

Fig. 10. Test control panel.



RN16: 4DDD
PC: 3000
EPC: 3008 33B2 DDD9 0140 3505 0000
Handle: 5D6E

Fig. 11. Waveform analysis module.

V. TAG COLLISION ANALYSIS

Because this research focuses primarily on the resolution of two-tag collisions, the collision signal is generated from two tags from the same manufacturer in the transmitting field of the data acquisition platform as introduced previously. By configuring the command parameter, the platform sends out a Query command with its Q filed equaling zero, and thus forces the two tags in the field to respond their 16-bit random number simultaneously. The RF carrier wave frequency for the air interface is set at 915MHz, and the IF frequency of the interrogator/reader is set at 15MHz. At the receiver side, the collision signal is captured by the platform and then downconverted to the baseband signal, which is sent into the FPGA. From the LabVIEW signal analysis module on the host PC, an acquired collision signal can be visualized as shown in Fig. 12.

Based on observation, the features of the collision signal can be summarized as follows:

The collision signal is the result of the linear superposition of tag responses, and the superposition follows a linear additive model as depicted in Fig. 13.A phase shift can also be observed in the collision signal, and the phase shift value is not fixed due to the tag BLF deviation. As can be seen, the initial phase shift is minimal, but it accumulates as the time increases.

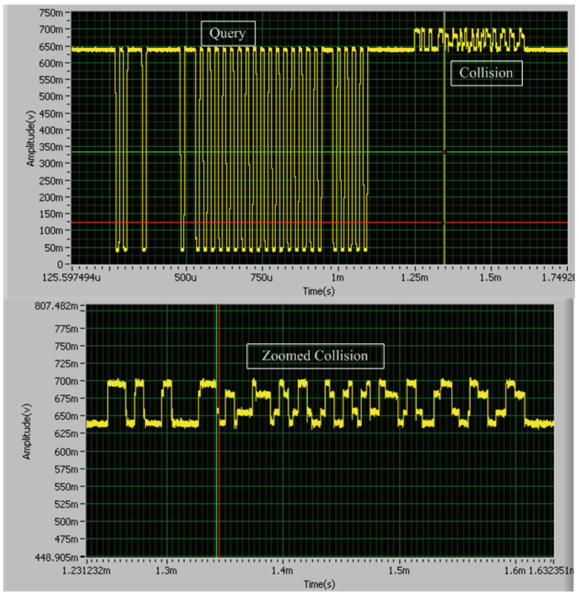


Fig. 12. Acquired collision signal.

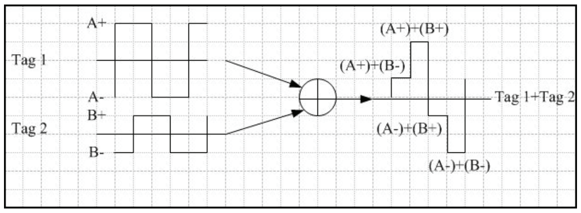


Fig. 13. Linear additive model of tag response.

The two tag responses received are likely different in magnitude. If the received two tag responses are with the same magnitude, only three possible voltage levels can be generated when they are linearly superimposed. Whereas, if they are different in magnitude, according to permutation, four possible voltage levels can be obtained from their linear superposition. The assumption is summarized in Table 2, in the left sub table, the response of Tag 1 and Tag 2 are with the same magnitude of [A-, A+], while in the right sub table, Tag 1 is with a magnitude of [A-, A+], and Tag 2 is with a magnitude [B-, B+] (A≠B). This assumption is proven by locating the four different voltage levels in the observed collision signal. The amplitude difference in received tag responses are caused by two major facts. The first one is that the two tag responses propagate through different paths; they thus suffer from different path attenuations. The second fact is that the transmitting channel and the receiving channel of the data acquisition platform are combined by a circulator connected to one patch antenna as shown in Fig. 14. Because the transmitting power of the reader carrier wave is significantly larger than the power of the received tag response, the PLL in the RF receiving circuitry locks onto the phase of the transmitting carrier wave. In addition, due to the capacitor variation in each tag, the tags carrier waves are different in phase. Therefore, the received two tag responses suffer from different attenuation factors caused by the carrier wave being out of synchronization with the receiver's local oscillator (LO).

Table 1. Possible voltage levels in collision.

Tag1 / Tag2	A+	A-		Tag1 / Tag2	A+	A-
A+	(A+)+(A+)	0		B+	(A+)+(B+)	(A-)+(B+)
A-	0	(A-)+(A-)		B-	(A+)+(B-)	(A-)+(B-)

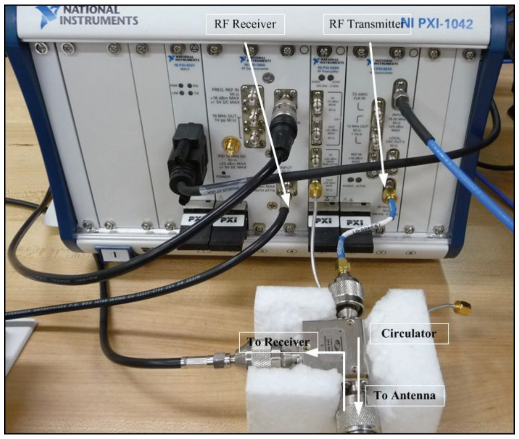


Fig. 14. RF front end connection with circulator.

The two tag responses are likely added together in the air with a phase shift (delay). The reason for this phase shift is due to two factors. First, because of physical variations of certain capacitors and antenna connections in forming the backscatter signal in each tag, the two tags respond to the reader with different speeds after receiving the Query command. In addition, the signal from each tag propagates along different paths, which are likely different in distance. The phase shift caused by the propagation path difference equals the quotient of path length difference divided by the speed of light. Because the range of passive RFID communication cannot exceed 10m normally, it is thus minimal and can be neglected in the current analysis. In sum, the observed phase shift (on average) in the collision signal is assumed to be due to the difference in the tag circuitry response characteristics.

The average phase shift is inversely proportional to the tag BLF, and cannot exceed 20% of the symbol duration. This conclusion is obtained by commanding the tags to respond at typical BLF and measuring the length of phase shift as shown in Table 3. The measurement is carried out when the two tags are placed 2.5 inches from the antenna. Table 3 lists the measurement results of the phase shift among 8 different tags from the same manufacturer. For each measurement, the tags are placed together with two in a group. The mean of the phase shift at each BLF as well as the phase shift percentage compared to the symbol duration are listed in Table 4. Figs. 15 and 16 show the data listed in Table 2 and Table 3.

In summary of the features of the collision signal, four voltage levels and a phase shift (observed as short edge transition) exist in the received collision baseband.

Table 2. Phase shift at typical tag BLF.

Measurement Number	1	2	3	4	5	6	7	8
BLF=64kHz	1.118μs	1.076μs	1.604μs	0.904μs	1.179μs	0.965μs	1.026μs	1.143μs
BLF=128kHz	0.719μs	0.573μs	0.813μs	0.53μs	0.844μs	0.639μs	0.524μs	0.862μs
BLF=256kHz	0.626μs	0.371μs	0.473μs	0.657μs	0.695μs	0.587μs	0.526μs	0.64μs
BLF=341kHz	0.206μs	0.243μs	0.301μs	0.586μs	0.527μs	0.498μs	0.276μs	0.507μs
BLF=682kHz	0.101μs	0.078μs	0.336μs	0.144μs	0.156μs	0.17μs	0.064μs	0.052μs

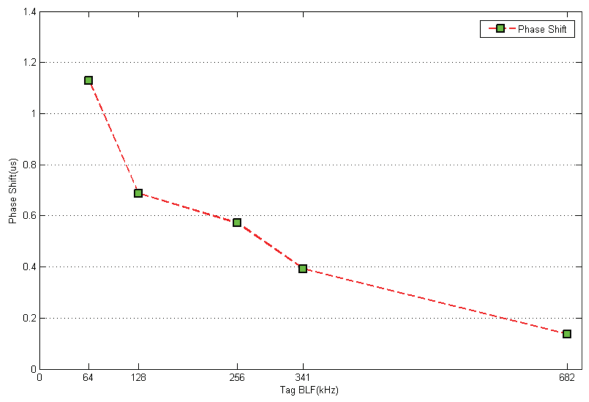


Fig. 15. Tag phase shift vs. BLF.

Table 3. Statistics of the phase shift.

	Mean(μs)	Symbol Duration(μs)	Percentage
BLF=64kHz	1.1296	15.625	7.23%
BLF=128kHz	0.6880	7.8125	8.81%
BLF=256kHz	0.5719	3.9100	14.63%
BLF=341kHz	0.3930	2.9300	13.41%
BLF=682kHz	0.1376	1.4600	9.42%

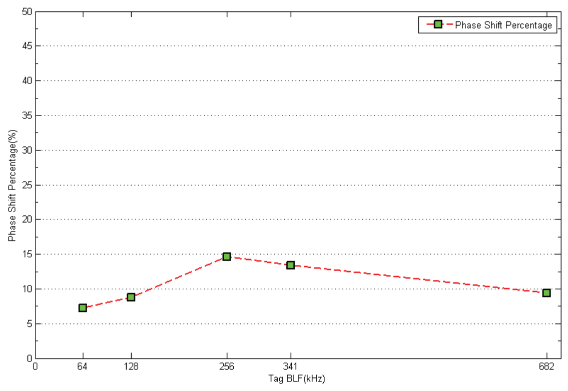


Fig. 16. Tag phase shift percentage in symbol duration vs. BLF.

VI. PROPOSED SOLUTIONS

A. The First Proposed Solution - Direct Edge Locating

Because a collision is the result of the linear superposition of each tag's response, the formation of each tag's response is invariant. In addition, because each tag's response potentially arrives at the receiver side at different instants (i.e. the phase shift observed in the collision), the edge transition in each tag's response is kept and not overlapped with each other (spaced by the phase shift). Fig. 17 illustrates the superposition of two formation-0 symbols, and the two edge transitions are maintained in the collision. Fig. 18 illustrates the superposition of two formation-1 symbols, there is no edge transition in the collision because of the absence of edge transition in each individual tag response. Fig. 19 illustrates the superposition of one formation-0 symbol and one formation-1 symbol, only one edge transition can be observed in the collision. Therefore, although superimposed, the two tag's responses can still be treated as being independent. Therefore, it is possible to recover the information of each tag by locating the edge transition in each symbol duration from the start of individual tag's response. The prerequisite of this resolution method is to locate the exact position of each tag's start in the collision signal, which will be discussed later.

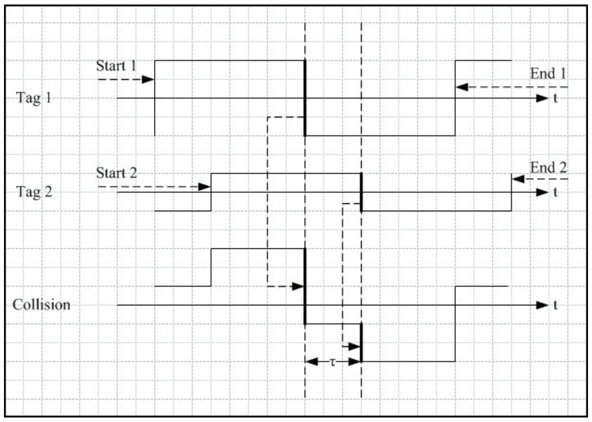


Fig. 17. Superposition of two formation-0 symbols.

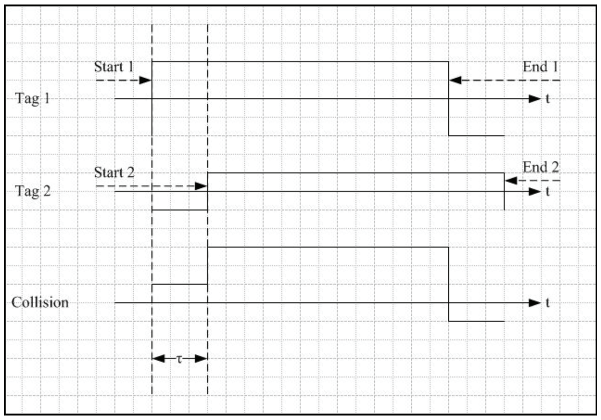


Fig. 18.Superposition of two formation-1 symbols.

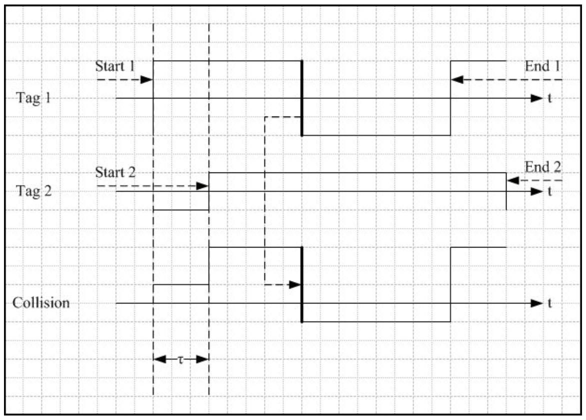


Fig. 19. Superposition of symbols.

1. Tag Response Frame Architecture

According to the ISO 18000-6c standard, when the tag responds to the reader, it shall precede its data with a preamble. Fig. 20 shows the preamble of the tag response when FM0 encoding is employed. In Fig. 21 the preamble of the tag response when FM0 encoding is employed. The parameter TRext decides whether a 12-zero pilot tone is used in the preamble. While two Tags respond the reader's Query command with their individual RN16, the two RN16 numbers shall be preceded by the same preamble. This information can be utilized to locate the start edge of each tag's RN16. Because once the formation of the preamble is fixed, the end of the preamble (i.e. the start of the RN16) can be located by counting to the edge number in the preamble from the start of the preamble. For example, in the FM0 encoding preamble as shown in Fig. 20, suppose no pilot tone is employed (TRext=0), there are eight edges in the preamble. The ending of the preamble can thus be located by counting 8 edges from the start edge of the preamble. In the collision signal, because the two tag responses are linearly superimposed with a phase shift, the ending of each preamble can be located by counting to the edge number in the preamble from the start of each preamble in the collision. Suppose the preamble contains N edges, the ending of the individual tag response are at the position of the (2N-1)th and the 2Nth edge in the collision signal. Fig. 22 shows an example of a collision signal consisting of two tag preambles in FM0 without pilot tone. Because each tag's preamble has 8 edges, the ending of tag 1 and tag 2's responses are at the 15th edge and 16th edge in the collision signal. The ending location for Miller encoding is similar.

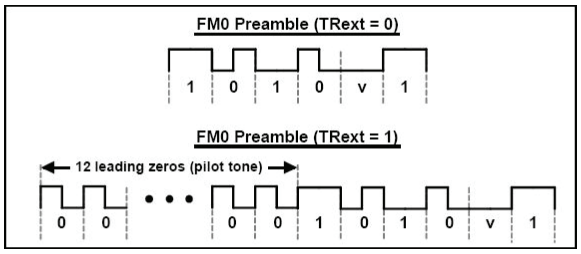


Fig. 20. Tag response preamble in FM0.

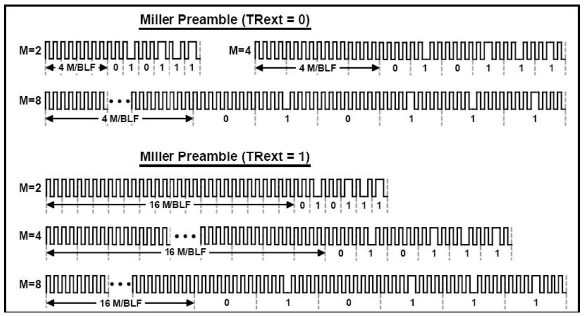


Fig. 21. Tag response preamble in Miller Subcarrier.

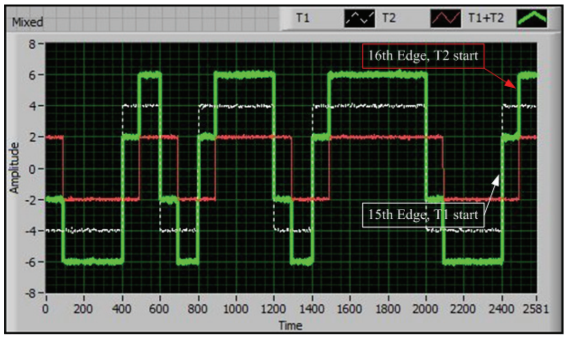


Fig. 22. Ending location of preambles in collision.

2. Algorithm Description

After locating the ending of the preamble (i.e. the starting of the RN16 in the tag responses), each bit in the RN16 of individual tag corresponding to one symbol duration can be recovered. The direct edge locating algorithm for two-tag RN16 collision resolution is listed in Table 4. This algorithm checks the collision signal one sample point at a time, and thus it is an online algorithm.

3. LabVIEW Simulation

The direct edge locating algorithm is simulated using LabVIEW on the Host PC. It is implemented by using a finite state machine (FSM) as shown in Fig. 23. The edge detector detects each edge transition in the collision signal, and sends out handshake signals to the counter and the FSM once an edge is found. The counter records the number of the edges detected. A timer counts the elapsed time since the start of the RN16, and notifies the FSM once it reaches the middle of a symbol as specified in Table 4. The edge detector is implemented in two ways. The first method is based on calculating the running mean of the input signal as shown in Fig. 24. The edge detector keeps recording the running mean of the magnitude of the latest 5 signal sample points, and updates the average of the positive peak and the negative peak online. If the previous signal point is below the average while the current signal point is above the average, a rising edge is detected; the falling edges are found in a similar manner. The second method is to calculate the differentiation of the input signal. Because the edge is a singular point in the signal, the corresponding differentiation appears as a spike. The differentiation is calculated using Eq. 1. Because the edge transition time of the tag response normally takes no more than two sampling periods (0.08µs, 25MHz), the d_t is set as 1. Therefore, the differentiation actually equals the difference of the two adjacent sampling points in the signal. In the case when the tag edge transition takes more than two sampling periods, the edge in the collision signal corresponds to each rising edge of the obtained pulses in the differentiation Fig. 25 shows the preamble part of an FM0 collision and its corresponding differentiation.

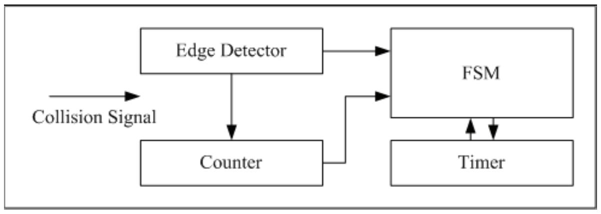


Fig. 23. FSM implementation of the algorithm.

$$\text{Differentiation} = \frac{y(t) - y(t-dt)}{dt} \quad (1)$$

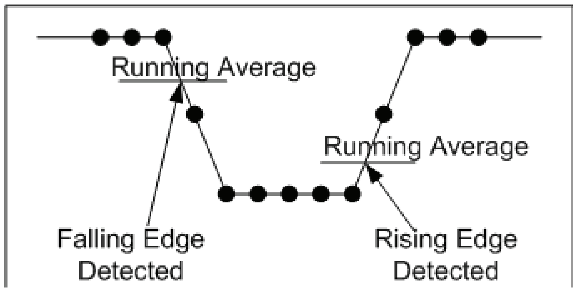


Fig. 24.The running mean calculation.

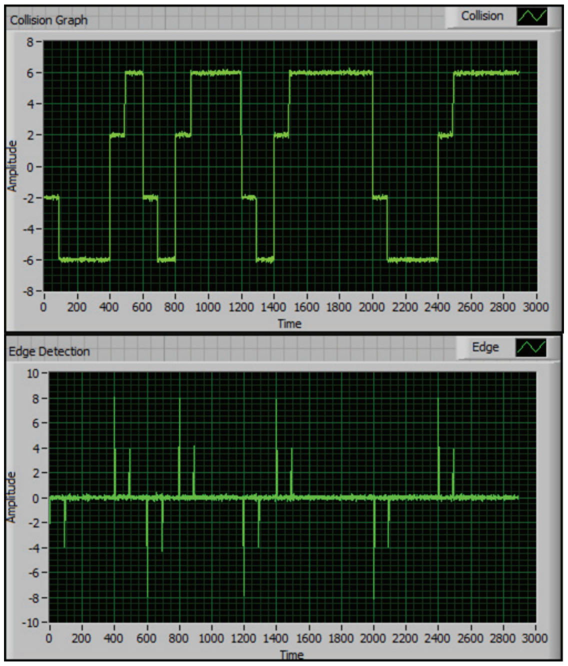


Fig. 25. The differentiation of collision signal.

Because the variation of symbol duration (BLF deviation) exists in the practical tag response, the middle symbol edge transition can appear in a range around the supposed St+k×T_s (k= 1, 3, 5, … , 31) positions rather than exactly the middle of each symbol. Therefore, the FSM searches in a range around the middle of each symbol for the edge transition rather than at the exact middle position. The search range is of a length of 5% of the symbol duration centered at the middle of each symbol. Fig. 26 illustrates the searching range.

Table 4. Direct edge locating algorithm.

```
1. Locating the preamble ending edge of tag 1 and tag 2 separately in the collision signal, denoted as St1 and St2.
2. Start form St1, check the existence of edge transition at time instant St1+k×T_s (k= 1, 3, 5,…, 31), which corresponds to the middle of each symbol in tag 1's RN16 ( where T_s is the symbol duration). If an edge transition is found at the specified position, the corresponding data bit in the RN16 is in formation 0. Otherwise, it is in formation 1.
3. Start from St2, check the existence of edge transition at time instant St1+k×T_s (k= 1, 3, 5,…, 31), which corresponds to the middle of each symbol in tag 2's RN16( where T_s is the symbol duration). If an edge transition is found at the specified position, the corresponding data bit in the RN16 is in formation 0. Otherwise, it is in formation 1.
```

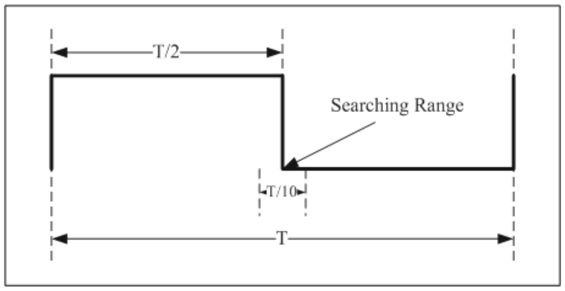


Fig. 26. Differentiation of a collision signal.

In the simulation, the symbol duration is set as 400 sample points, and the search range is thus 40 data points centered at the middle of each symbol. Two RN16 (8180h and 18FFh) encoded with FM0 are input to simulate the two tag responses separately. A Gaussian white noise with 5% of the signal strength is added to each tag response to simulate the channel noise. Fig. 27 shows the two tag responses with noise. Fig. 28 shows the collision signal and its corresponding differentiation showing the edges. The LEDs in Fig. 28 show the recovered data from the collision, the LED turned on corresponds to the formation 1 symbol while the LED turned off corresponds to the formation 0 symbol. As a comparison, the recovered data is exactly the input RN16s.

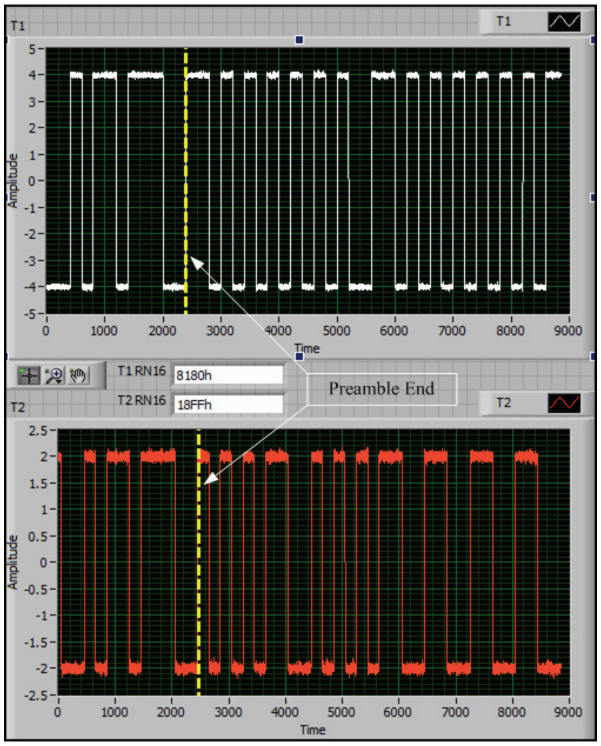


Fig. 27. Two simulated tag responses.

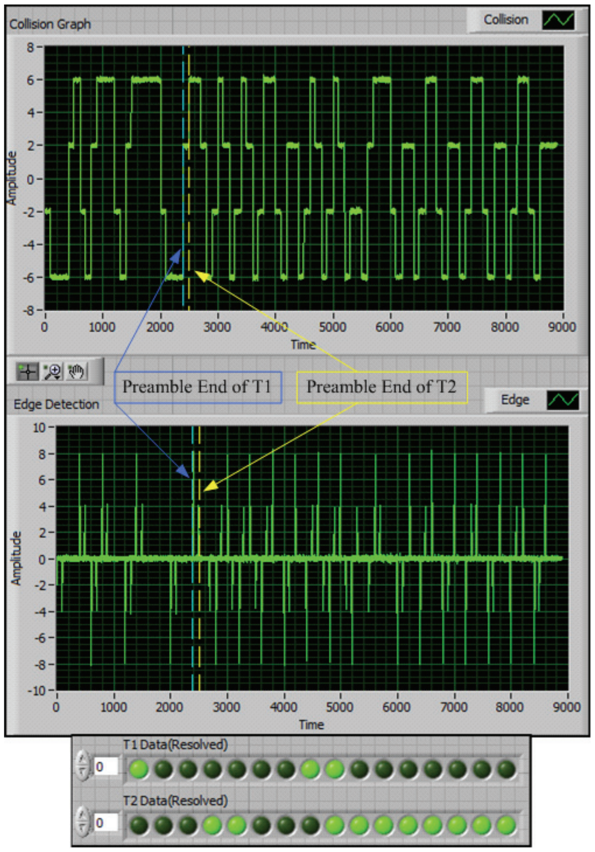


Fig. 28. Collision signal and differentiation.

There are two limitations of the algorithm:

First, as the BLF of tag increases, the phase delay decreases as shown previously, which requires the hardware timer to locate the searching range for each symbol more accurately. It is also possible that the searching range of the current symbol for each tag may overlap (as shown in Fig. 29), which requires the shrinking of the searching range.

Second, because of the variable phase shift caused by the BLF deviation, a self calibration function is required which searches for the ending edge of each data bit in the collision, and calibrates the algorithm FSM to the exact start point of the successive data bit symbol.

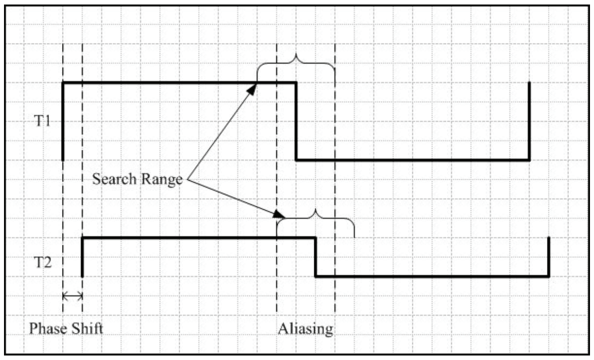


Fig. 29. Searching range overlap.

B. The Second Solution - Amplitude Mapping

As discussed previously, when the tag BLF increases, the phase shift between the two tag responses decreases. When the phase shift decreases to within the search range length, the direct edge locating method for collision resolution as described previously, does not apply because of the interference of the adjacent edge transition in each tag symbol. In the extreme case, when the phase shift shrinks to zero, an ambiguity as shown in Fig. 30 makes the direct edge locating method fail entirely when at least one tag has a formation 0 symbol appear. In Fig. 30, two tag symbols are linearly superimposed without phase shift. When one tag is responding a symbol in formation 0, while another is responding a symbol in formation 1, or when both of the tags are responding with formation 0 symbols, the observed edge transition in the middle of the symbol leads to the ambiguity when symbol arbitration. Because there is no phase shift and the edge transitions in each tag response are completely overlapped, although an edge transition can be detected in the middle of the symbol, whether Tag 1 or Tag 2 is in transmitting formation 0 cannot be determined. However, in the case when both of the tags are transmitting formation 1 symbol, the original information can still be resolved from the collision because there is no edge transition in the middle of the symbol. Observed from Fig. 30, although all the ambiguity cases feature an edge transition in the middle of the symbol, they are different in the amplitude level positions. Therefore, it is intuitive to resolve the collision based on the relative position of the voltage levels.

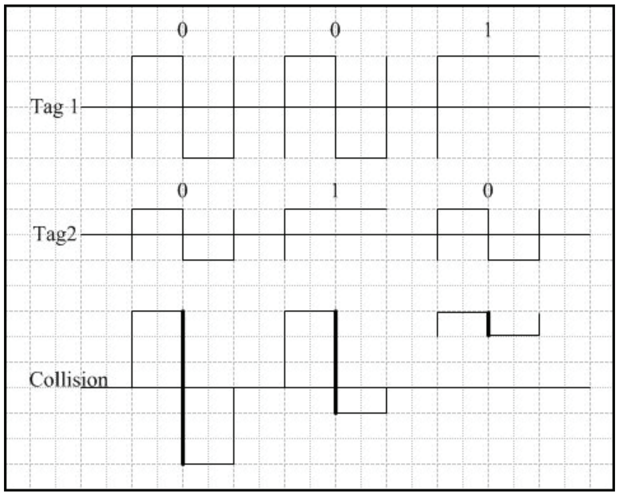


Fig. 30. Edge ambiguity.

Because there are two possible formation 0 symbols and two possible formation 1 symbols, each symbol in the tag response can have four possible shapes as shown in Fig. 31. Therefore, when the two tag responses are superimposed, there can be 16 possible shapes of the collision symbol as shown in Fig. 31. It is supposed that the amplitude of Tag 1's response is twice as the amplitude of Tag 2's response for illustration purpose. In Fig. 31, the amplitude of Tag 1's response is supposed to be twice as the amplitude of Tag 2's response for illustration purposes. The upper two portions (area 1 and area 2) correspond to the cases when both of the tag responses are same in formation, while the bottom two portions (area 3 and area 4) correspond to the cases when both of the tag responses are different in formation.

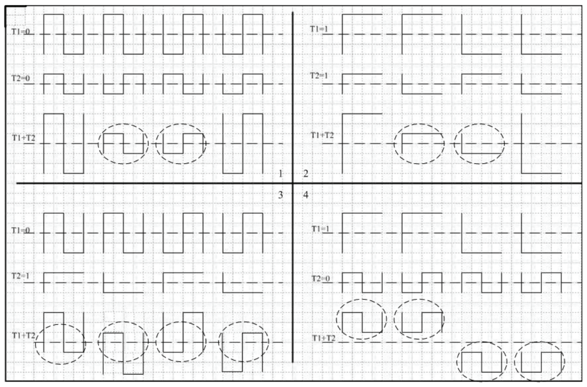


Fig. 31. Collision patterns.

The characteristics of the collision pattern can be summarized as following:

For the collision in area 1 (T1=T2=0): The collision signal is symmetric to the average line, and the first half symbol level and the second half symbol levels distribute at different sides of the average line.

For the collision in area 2 (T1=T2=1): There is no edge transition in the middle of the symbol, and the first half symbol level and the second half symbol levels distribute at the same side of the average line.

For the collision in area 3 (T1=0, T2=1): The collision signal is not symmetric to the average line, and the first half symbol level and the second half symbol levels distribute at different sides of the average line.

For the collision in area 4 (T1=1, T2=0): There is an edge transition in the middle of the symbol, and the first half symbol level and the second half symbol levels distribute at the same side of the average line.

As discussed earlier, there are four possible voltage levels in the collision signal. Because the tag responses are preceded by a common preamble, the maximum and the minimum voltage levels appear in the preamble part of the collision. The average line is calculated over the maximum and minimum voltage level. Except for the maximum and minimum voltage levels, the other two possible intermediate voltage levels, which are symmetrically distributed at the opposite side of the average line, appear when the voltage levels in the individual tag response with opposite polarization are superimposed as indicated by the circled cases in Fig. 31.

Figure 32 shows an example collision signal consisting of two tag responses in FM0 without a phase shift. The RN16 in Tag 1 is 672Bh, and the RN16 in Tag 2 is A7CDh. To arbitrate the symmetry of the collision symbol to the average line, the average of the current symbol is compared to the average line of the collision. Two samples are taken at the first half and the second half of each collision symbol, and the symbol average is calculated as the difference of the two sample values divided by two. If the average of the current symbol voltage levels equals the collision average line, the symbol is symmetric to the average line.

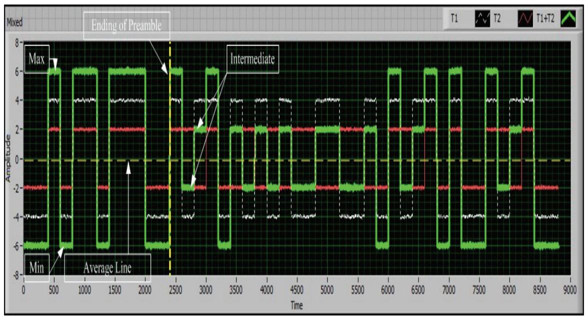


Fig. 32. Superposition of tag responses without phase shift.

1. Algorithm Description

Based on the characteristics of the collision symbol, the individual tag symbol can be resolved by scanning through the collision and mapping the collision signal to the cases as listed in Fig. 31. The algorithm is therefore named Amplitude Mapping and as listed in Table 5.

Table 5. Amplitude mapping algorithm.

1. Scan the duration of the preamble, and find the maximum/minimum value and the average line of the collision.
2. Beginning from the end of the preamble, check each symbol duration of the collided data. Calculate the average of each collide symbol.
3. Map the collided symbol to the cases as shown in Fig. 31 to decide the individual tag data.

2. LabVIEW Simulation

The amplitude mapping algorithm is simulated using LabVIEW on the Host PC. It is implemented by using a FSM as shown in Fig. 33. In each collision symbol, two sample points at 25% and

75% of the symbol duration are taken for the first half level and the second half voltage level separately as shown in Fig. 34. The timer counts the elapsed time since the start of the RN16, and notifies the FSM once it reaches the sampling points in each symbol. The comparator compares the voltage level of the sampling points in each symbol with the collision average line. A threshold is introduced in the comparator to get rid of the interference of the noise. The Arbitrator maps the collision symbols to the four cases according to the result of the comparator. In the simulation, the symbol duration is set as 400 data points, and the range for the comparator to neglect the interference of noise is set as 10% of the maximum amplitude of the collision signal. Two RN16 (672Bh and A7CDh) encoded with FM0 are input to simulate the two tag responses separately. A Gaussian white noise with 5% of the signal strength is added to each tag response to simulate the channel noise

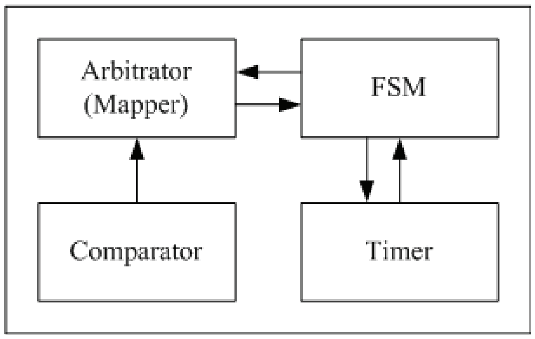


Fig. 33. FSM implementation of the algorithm.

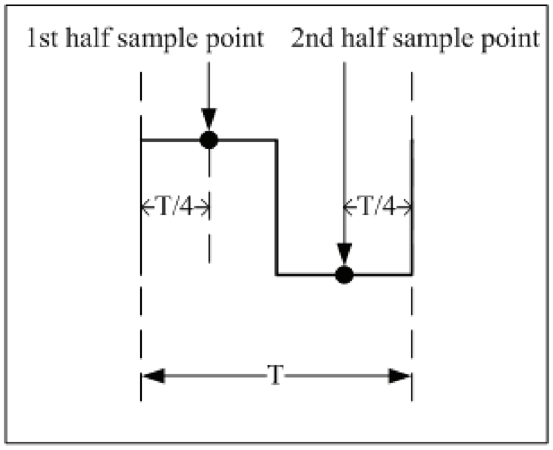


Fig. 34. Symbol sampling point.

Figure 35 shows the two tag responses with noise. Figure 36 shows the collision signal and recovered data. The LEDs in Fig. 36 shows the recovered data from the collision, and the LED turned on corresponds to formation 1 symbol while the LED turned off corresponds to the formation 0 symbol. As a basis for comparison, the recovered data are exactly the input RN16s.

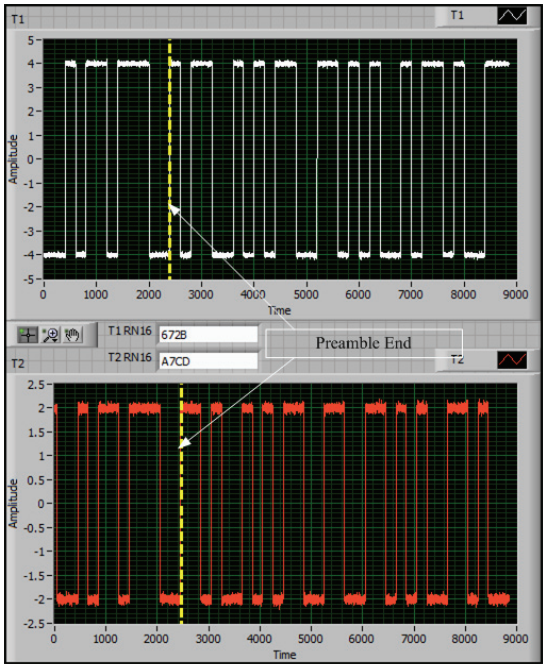


Fig. 35. Two simulated tag responses.

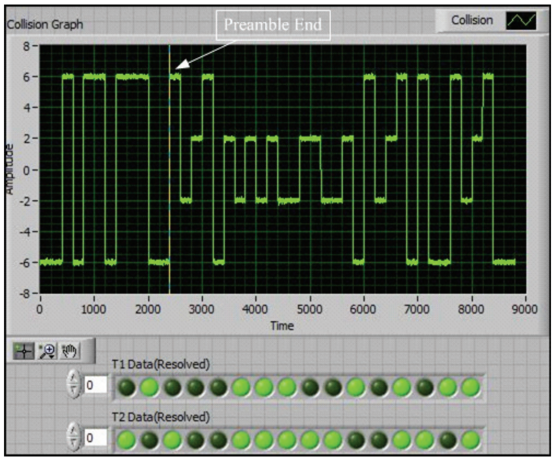


Fig. 36. Collision and resolved data.

VII. SOLUTION UNIFICATION

The solution to resolving the collision with phase shift and without shift can be unified by passing the collision signal with phase shift into a median filter. The median filter takes in N data points and outputs the median of the N data points. (N is the length of the median filter). The median filter thus filters out the impulse in the collision signal caused by the phase shift and therefore transfers the collision with phase shift to the collision without phase shift, which can be resolved using amplitude mapping. The length of the median filter must be equal to or greater than the phase shift in order to filter out the phase shift caused impulse. In Fig. 37, a collision with phase shift of 5% symbol duration is filtered by a median filter with a length of 10% symbol duration. The impulses caused by the phase shift are circled, and in the filtered waveform they are removed.

One side benefit of the median filter is that it filters out some of the noise in the collision signal and improves the signal-to-noise ratio (SNR). As an illustration, the results of the two algorithms working on the same collision with phase shift are compared. In the simulation, the symbol duration is set 400 data points in length, and the phase shift is 20 data points (5% of T_s). The collision includes two RN16s (672Bh and A7CDh). Fig. 38 shows the resolution result. The resolution results of the two algorithms working on the same collision signal are the same.

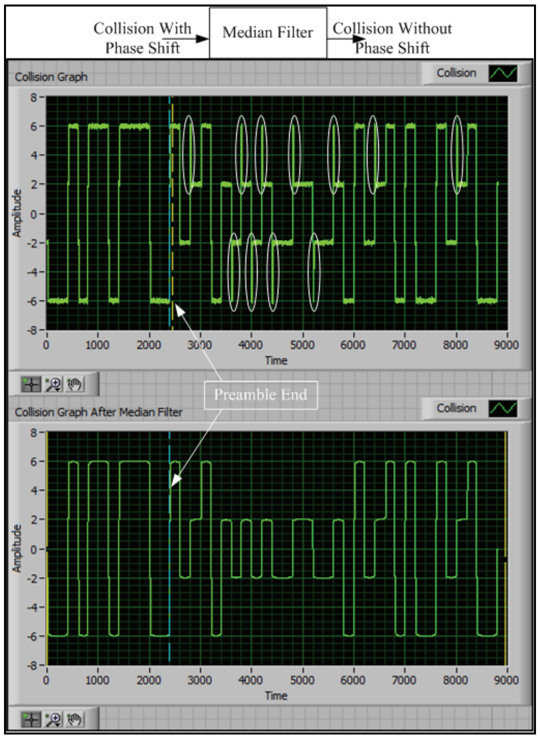

Collision With Phase Shift
Median Filter
Collision Without Phase Shift
Collision Graph
Collision
Amplitude
Time
Preamble End
Collision Graph After Median Filter

Fig. 37. The effect of median filter.

There is one limitation of the unification: the phase shift cannot exceed 25% of the symbol duration. This is because when the phase shift exceeds 25% of the symbol duration, the impulse caused by the phase shift is greater in length than the voltage level used for amplitude mapping. Because the length of the median filter shall always be greater than the phase shift, the median filter filters out both the phase shift and part of the useful voltage level which happens to be in the shape of an impulse. Fig. 39 shows this limitation, as shown the filtered collision with a phase shift less than 25% of symbol duration(left) differentiate with the filtered collision with a phase shift of 25% of symbol duration(right). Fortunately, due to the feature of the collision signal as discussed previously, the phase shift cannot exceed 20% of the symbol duration at typical tag BLFs.

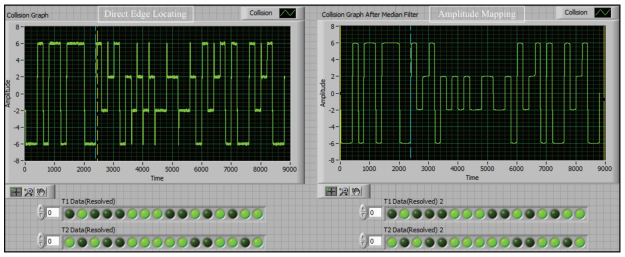


Fig. 38. Algorithm comparison.

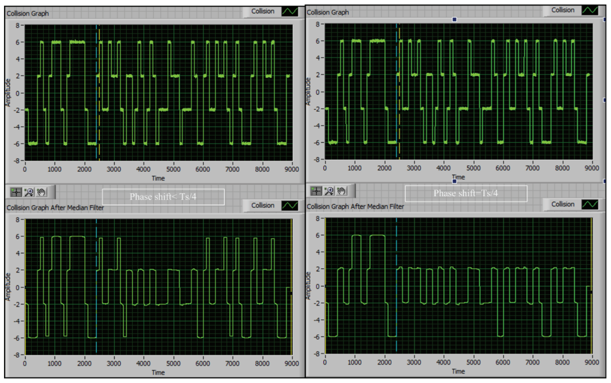


Fig. 39. The limitation of phase shift.

VIII. ALGORITHM HARDWARE IMPLEMENTATION

When simulated in LabVIEW on the host PC, both the direct edge locating algorithm and the amplitude mapping algorithm are developed and function on-line. The program reads in the collision signal and performs point-by-point processing on it. The benefit of this programming style is for a simple FPGA implementation. LabVIEW provides its FPGA compilation environment in a LabVIEW FPGA module, which allows for direct translation of LabVIEW code into low-level HDL code. The floating-point numbers used in the host PC LabVIEW need to be transferred to fixed point numbers in LabVIEW FPGA module.

Because most of the computations involved in both algorithms are Boolean for the arbitration logic and counting for timer, the NI5640R FPGA baseband used in the data acquisition platform featuring a Xilinx Virtex-II Pro XC2VP30 FPGA is quite adequate. Two individual tag responses of typical BLF are acquired by the data acquisition platform and mixed to generate the collision signal as the test bench on Host PC. The typical BLFs are 64 kHz, 128 kHz, 256 kHz, 341 kHz and 682 kHz. Fig. 40 shows the implementation verification flow. The test bench signal is then streamed into the FPGA, and stored in the FPGA block memory for play back. The resolution result generated by the FPGA is streamed back to host PC for visualization and comparison with the original tag responses. The FPGA also generates a time stamp once it finishes the resolution, which shows the processing time. The processing time corresponding to each BLF is compared to the standard specified turnaround time to check the real time conformance of each algorithm as listed in Table 6. As shown, the implementation performs collision resolution within the required real time. Fig. 41 shows the front panel of the FPGA verification platform in LabVIEW.

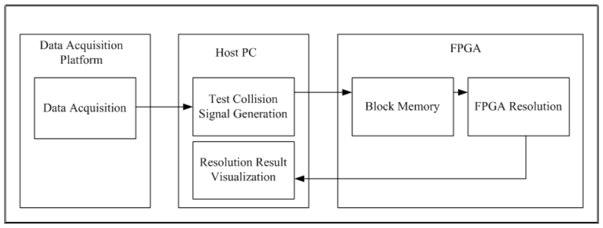
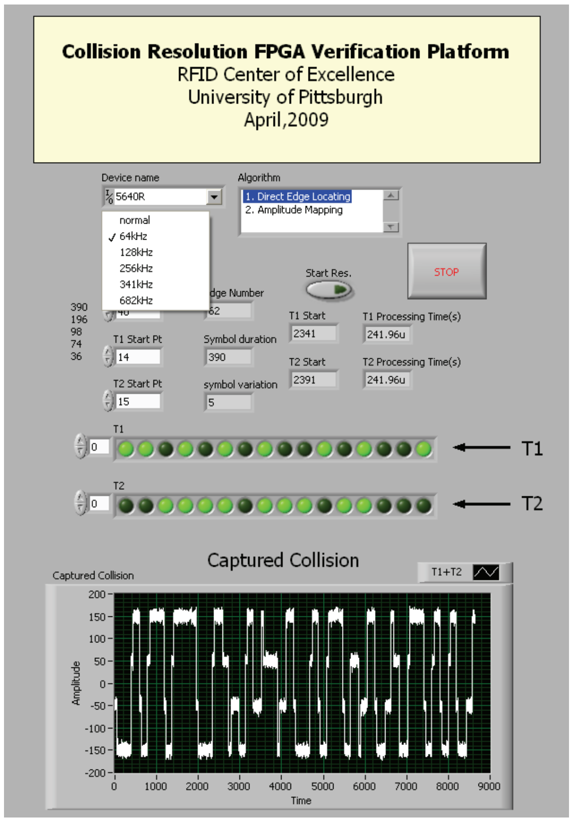

Data Acquisition Platform
Data Acquisition
Host PC
Test Collision Signal Generation
Resolution Result Visualization
FPGA
Block Memory
FPGA Resolution
Fig. 40. Verification flow of the implementation.

Table 6. FPGA processing time of algorithms.

BLF(Hz)	Collision Length N	Processing Time(us)	Required Time(us)
64000	6256	241.96	261.4+312.5
128000	3120	121.68	124.8+156.25
256000	1568	60.92	62.72+78.125
341000	1168	46.04	46.72+62.5
682000	592	22.40	23.68+31.25


Collision Resolution FPGA Verification Platform
RFID Center of Excellence
University of Pittsburgh
April,2009

Device name: 5640R
Algorithm: 1. Direct Edge Locating (selected); 2. Amplitude Mapping
normal
✓ 64kHz
128kHz
256kHz
341kHz
682kHz
Edge Number: 62
T1 Start Pt: 14
T2 Start Pt: 15
Symbol duration: 390
symbol variation: 5
T1 Start: 2341
T2 Start: 2391
T1 Processing Time(s): 241.96u
T2 Processing Time(s): 241.96u

Fig. 41. Verification platform front panel.

IX. COLLISION OF MORE THAN 2 TAGS

A. Background

As discussed previously, direct edge locating can resolve two-tag collisions with phase shift, while amplitude mapping can deal with the collision without phase shift and the two methods can be unified by incorporating a median filter. However, using the direct edge locating method, as the phase shift decrease in length, the searching range of each symbol may alias and cause error; while with the unifying edge mapping method with the direct edge locating method, the phase shift cannot exceed or get close to 25% of the symbol duration, otherwise the useful edge information will be removed by the median filter. In addition, both methods require that the two tag responses are different in amplitude, which increases the probability of arbitration error when they are close in amplitude. Finally, both methods work only for two-tag collision resolution. When the number of tag collisions increases, more edge transitions may occur in one symbol duration, the probability of aliasing in the searching range increases. The difficulty of direct edge locating will thus increase. Similarly, N^2 possible voltage levels can appear in one symbol duration for an N tag collision, which increases the logic for arbitration if using amplitude mapping. Therefore, it is intuitive to ask: is it possible to resolve multiple tag collision without the limitation of the two pre-proposed solutions?

B. Introduction to ICA

The multiple tag collision problem is similar to the "cocktail party problem", where a number of people are talking simultaneously in a room (like at a cocktail party), and one person is trying to follow one of the discussions. To resolve the collision, a Blind Source Separation (BSS) is required. BSS problems in digital signal processing are those in which several signals have been mixed together, and the objective is to find out what the original signals were. Independent Component Analysis (ICA) is one popular method for BSS. ICA is a statistical and computational technique for revealing hidden factors that underlie sets of random variables, measurements, or signals. ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear or nonlinear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed non-Gaussian and statistically independent as they are called the independent components of the observed data. These independent components, also called sources or factors, can be found using the ICA method. Highly successful new algorithms in ICA were introduced by several research groups, together with impressive demonstrations on problems like the cocktail-party effect, where the individual speech waveforms are found from their mixtures. ICA became one of the exciting new topics, both in the field of neural networks, especially unsupervised learning, and more generally in advanced statistics and signal processing. Reported real-world applications of ICA on biomedical signal processing, audio signal separation, telecommunications, fault diagnosis, feature extraction, financial time series analysis, and data mining are discussed in [8]. Because each tag's response with a reader command is independent, and the statistical characteristics of the response signal are fixed and can be determined *a priori* large sample of responses, ICA can be a candidate for resolving multiple tag collisions.

ICA is a method for finding underlying factors or components from multivariate statistical data, and it looks for components that are both statistically independent, and with non-Gaussian distribution. Because the responses from each conflicting tag is a 16-bit random numbers and all the responses are driven by an independent clock signal on each tag, the reader actually receives conflicting tag responses which are statistically independent of each other. Because the baseband binary tag responses consists of only two separate logic values 0 and 1 (or +1 and -1), they are intrinsically super-Gaussian distributed characterized by two distinct peaks located near the two logic values in the probability density function (pdf) curve. Therefore, the conflicting signals satisfy both of the prerequisites of ICA. As in Eq. 2, the mixture can be represented as a vector X, where each vector variable corresponds to a received mixture; the source can be represented as a vector S, and where each vector variable corresponds to a source signal. The mixing matrix is represented as A. Assume the number of the conflicting tags (the size of the S vector in Eq. 2) is M, and the number of receiving channels (the size of the X vector in Eq. 2) is N, ICA requires that n shall be at least equal to M (i.e. There shall be no less captured/observed mixtures than the mixing sources). Therefore, in order to resolve M tag collisions, at least M receiving channels are required. If the number of the receiving channels is more than the number of conflicting tags, a preprocessing Principle Component Analysis (PCA) will be performed to extract N most significant components from the mixtures in order to make the mixing matrix A in Eq. 2 square (because only a square matrix has inverse). ICA is then performed to find the inverse of A (an N by M matrix) in an iterative manner until the algorithm converges.

$$X = AS \qquad (2)$$

where $X = (x_1,x_2, ..., x_m)'$, $S = (s_1, s_2, ... , s_n)'$.

X. ALGORITHM DESCRIPTION

A. General

Various practical methods for employing the ICA model can be employed for the current application. Reference [8] lists several candidate methods including (1) the approach based on finding the maxima of non-Gaussianity, (2) the classic maximum likelihood estimation method, and (3) the method on minimizing the mutual information. According to [8], among the available methods, one approach based on minimizing Entropy of the collision signal in category (1) features medium computation load with reasonable separation quality. It is thus selected to be used in the ICA model for multiple tag collision resolution, which requires real time signal processing.

In information theory, Entropy H of a signal y (as described in Eq. 3) is a measure of non-Gaussianity. (Where, Py(y) is the pdf function of signal y.)

$$H(y) = -\int P_y(y) \log P_y(y)\, dy \quad . \qquad (3)$$

According to information theory, a Gaussian signal has the largest Entropy among all random signals of equal variance. In probability theory, the central limit theorem (CLT) states conditions under which the sum of a sufficiently large number of independent random variables, each with finite mean and variance, will be approximately normally distributed [10]. According to the ICA model in Eq. 2, the mixture signal X (multiple tag collision signal) is the linear superposition of the source signal (the response of each tag), which implies that the distribution of the mixture signal approaches to normal distribution (more Gaussian) more than the source signals. Therefore, the ICA algorithm resolves the collision signal by minimizing its Entropy (Gaussianity), because once the source signals are recovered, they are with minimum Entropy compared to the collision signal. To obtain a measure of non-Gaussianity that is zero for a Gaussian variable and always nonnegative, a normalized version of differential entropy, called negentropy J as defined in Eq. 4 is used. (where y_{gauss} is a Gaussian random vector.) Negentropy is always nonnegative, and it is zero if and only if signal y has a Gaussian distribution.

$$J(y) = H(y_{gauss}) - H(y) \; . \qquad (4)$$

Although Entropy and Negentropy can be used to measure the Gaussianity, the integral in the calculation hinders computation efficiency. As an effective approximation, the Negentropy of a signal can be calculated without involving the integral as shown in Eq. 5. (Where G is a nonquadratic function):

$$J(y) \propto [E\{G(y)\} - E\{G(v)\}]^2 \; . \qquad (5)$$

Hyvärinen and Oja proposed a computation efficient algorithm FastICA [9] based on a fixed-point iteration scheme for finding a maximum of the non-Gaussianity of as measured in Eq. 5. The algorithm performs a Gaussian-Newton optimization when maximizing the Negentropy. Following Eq. 2, the source can be recovered from the collision signal X by left-multiplying the collision with a separating matrix W as shown in Eq. 6, which is the inverse of the mixing matrix X. Table 7 lists the FastICA algorithm using Negentropy maximization.

$$S = WX; \qquad (6)$$

where $X = (x_1, x_2, \ldots, x_n)'$, $S = (s_1, s_2, \ldots, s_n)'$.

B. ICA simulation

The FastICA algorithm is simulated using LabVIEW on the Host PC. The G is selected according to Eq. 7.

$$G(u) = (1/3)u^3 \; . \qquad (7)$$

Figure 42 shows the two tag collision resolution simulation result using FastICA. Two RN16 numbers D2CCh (S_1) and 3D74h (S_2) are generated in FM0 encoding and mixed to simulate the two collisions (X_1 and X_2) captured by the two receiving channels as the upper-most waveform shows. The separated source signals from the collision are displayed in the middle and at the bottom. As shown, the recovered signals are clear enough compared to the original mixture for decoding, and they correspond to D2CCh (for recovered S_1) and 3D74h (for recovered S_2) exactly. The separating matrix W is also shown, and the result can be verified by multiplying the recovered S vector with the inverse of W, i.e.

mixing matrix A, and then comparing the products with the mixtures X (the top in Fig. 42). Because the formation of the tag response is similar to a simple bipolar square wave, the algorithm converges after only one iteration.

The FastICA as shown in Table 7 reads all the collision data points and performs the computation based on the expectation of data. This is similar to batch training rather than point-by point online training. There is the choice between on-line and batch algorithms. An on-line version of the algorithm can be obtained by substituting the expected value in the algorithm with instant data value. However, the on-line algorithm trades the accuracy for processing speed, and the algorithm may not converge well as the batch training algorithm. This problem can be alleviated by combinations of the two as shown in Fig. 43. Rather than performing the FastICA after completely receiving the collision signal, the "divide and conquer" architecture in Fig. 43 divides the collision signal into the preamble and each data bit in RN16 and then performs ICA on each section. The data portions of the collision are successively stored into a register file, and the W is successively updated: W_1 is updated based on W_0, W_2 is updated based on W_1, … , etc, until the change between adjacent W becomes minimal. Once the W converges at a certain stage, ICA need not be performed thereafter, and the collisions before the converge stage are resolved by multiplying the converged W with the pre-stored data. Therefore, the computation load can be further decreased, and the resolution can be finished at the very end of the collision provided W converges before the collision end.

Table 7. FastICA algorithm.

```
1. Preprocessing the collision signal X by removing the mean
2. Whiten the data to Z, ( $Z=(z_1,z_2,\ldots,z_n)'$ )
3. Initialize the w' vectors (each row) in the separating matrix W
4. w = E{Zg(w'Z)} - E{g'(w'Z)}w ,where g is the first derivative of the G function in Eq.5.
5. Let w= w/||w||
6. If not converged, go back to step 4.
```

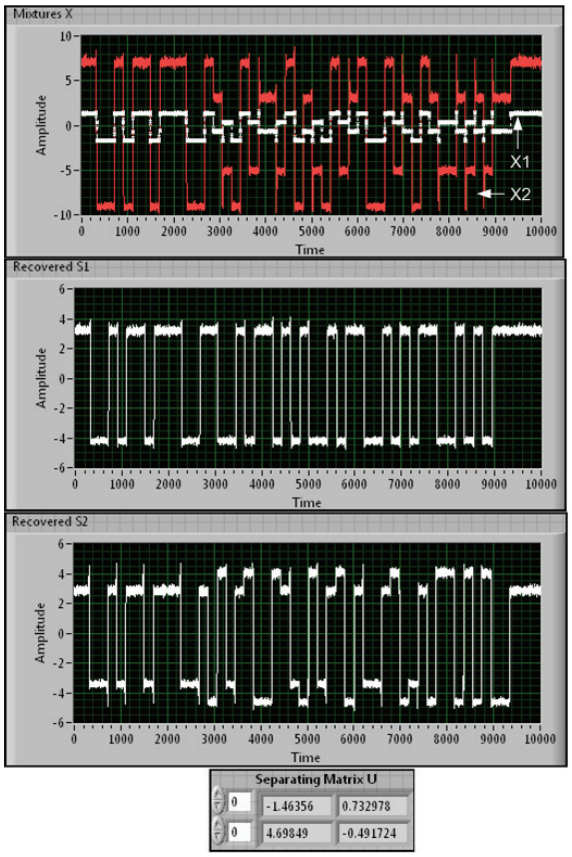


Fig. 42. ICA collision resolution result.

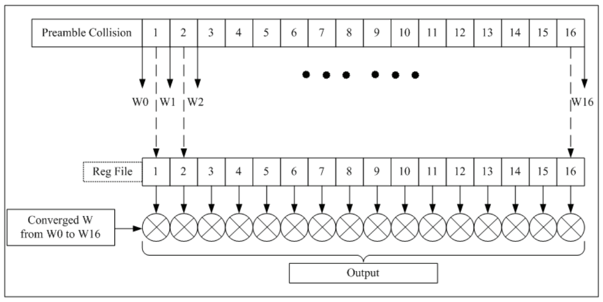


Fig. 43. Combination of batch training and online training of ICA.

XI. ICA IMPLEMENTATION

ICA treats each tag response as an independent component, and resolves the collision based on the statistical characteristics of signals rather than the phase shift information as in direct edge locating and amplitude mapping discussed earlier. Therefore, ICA performs the separation without the limitation of the previous solutions at the expense of higher computation load. The higher computation load tends to slow down of the resolution speed, which is critical in ISO 18000-6c applications. This can be alleviated by incorporating any trade-off that can be made between the resolution accuracy and processing speed as discussed previously.

The computation load of ICA also increases linearly with the number of collided tag responses. Because the scope of this paper is limited to two-tag collision resolution, and the algorithm can be easily extended for multiple tag collision, ICA will be implemented for resolving two-tag collision for prototyping and function verification purpose.

A. Implementation Device Selection

The computation load of FastICA relates to two major parts: the update of the weight vector and the separation. Suppose the number of collided tags is two and using batch training, the computation to update weight is as shown in Eq. 8. Each function and parameter in Eq. 8 has been defined previously. Suppose the collision signal contains N data points, to calculate the first expectation in Eq. 8, it requires approximately 2N additions and 2N multiplications. To calculate the second item, it requires also approximately 2N additions and 2N multiplications. The resolution of one tag signal is shown in Eq. 9, which requires 2N multiplications and N additions. Therefore the total computation load for separate one tag signal from the collision of N data points is 6N multiplications and 5N additions, and it can be alleviated by using online training.

$$w = E\{Zg(w'Z)\} - E\{g'(w'Z)\}w \qquad (8)$$

$$S_1 = w_1'Z \quad \text{and } S_2 = w_2'Z\,. \qquad (9)$$

Based on the analysis of the computation load, the Xilinx Virtex-5 ML506 Evaluation Platform has been selected. The development board features a XC5VSX50TFFG1136 FPGA, which is in 65nm technology and optimized for DSP and memory-intensive applications with low-power serial connectivity. It combines enhanced DSP blocks (DSP48E) for parallel processing, highest memory-to-logic ratio, and low-power serial transceivers for high I/O bandwidth. According to the Xilinx DSP48E specification [12], the multiplication speed of the target FPGA can reach 345MHz while the addition speed can reach 500MHz. Table 8 lists the expected ICA resolution speed for two tag collision resolution using the target FPGA. Fig. 44 shows the corresponding speed at typical tag BLFs. As shown, the target FPGA can satisfy the standard specified real time limitation.

Table 8. Expected ICA speed using FPGA vs. maximum time allowed.

BLF(Hz)	Collision Length N	Processing Time(us)	Required Time(us)
64000	6256	171.36	312.5
128000	3120	85.46	156.25
256000	1568	42.95	78.125
341000	1168	31.99	62.5
682000	592	16.22	31.25

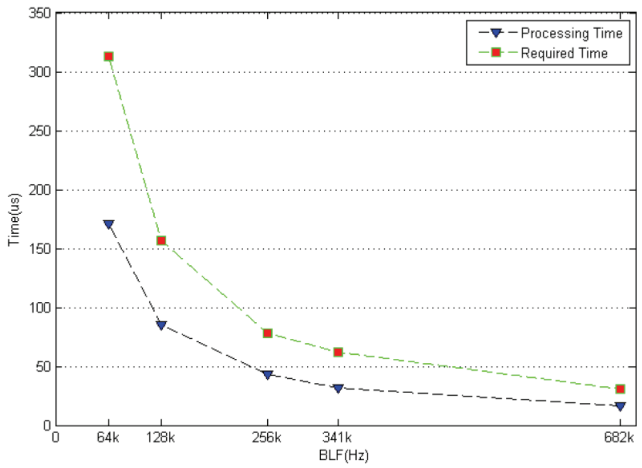


Fig. 44. Expected ICA speed using FPGA vs. required speed.

Matlab/Simulink +Xilinx System generator has been selected as the development tool chain. The FPGA design using the Xilinx System Generator is different from the typical HDL approach. Using the system generator, the FPGA is designed by first developing algorithm in Matlab/Simulink, and then the code is automatically compiled into HDL. This method can significantly reduce the hardware verification workload.

B. Experiment Setup

To resolve two-tag collision, two receiving channels are required. The experiment setup is shown in Fig. 45. Two National Instruments NI5600 RF downconverters are used as the two independent receiving channels. Collision responses of typical BLF will be acquired by the data acquisition platform. Batch training ICA and on-line training ICA will be implemented separately, and comparison on accuracy and processing speed will be reported.

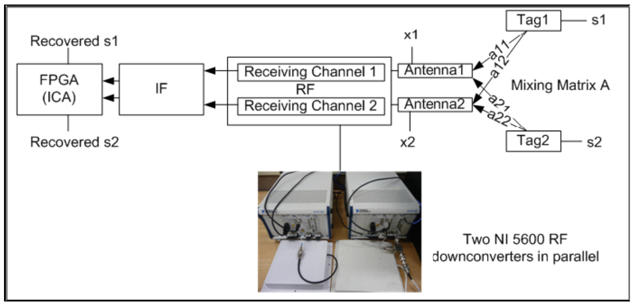


Fig. 45. ICA for two-tag collision resolution experiment configuration.

XII. SUMMARY

In this paper, the tag collision resolution into distinct readable information for ISO 18000-6c passive RFID communication has been accomplished. Two online resolution methods: direct edge locating and amplitude mapping for resolve two-tag collision has been presented. The edge locating method resolves the two-tag collision with phase shift, while the amplitude mapping method deals with the collision with very short or without phase shift. The two resolution methods are then unified by pre-processing the tag collision with a median filter. Corresponding simulations using LabVIEW on host PC were performed as preliminary work to verify the functionality of proposed algorithms. In addition, the limitation of each method is discussed separately.

To extend the resolution to multiple tag collisions, a statistic signal processing method using ICA has been introduced. The ICA method resolves two or more tag collision without the limitation of the direct edge locating and amplitude mapping at the expense of hardware cost for additional receiving channels and higher computation load. The ICA algorithm has been shown to work with both batch training and online training thus verifying the functionality.

REFERENCES

[1] EPCglobal, "EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Conformance Requirements V.1.0.2", http://www.epcglobalinc.org, 2005.

[2] J. G. Lee, S.J. Hwang, and S. W. Kim, "Performance Study of Anti-collision Algorithms for EPC-C1 Gen2 RFID Protocol", *Proceeding of the International Conference on Information Networking*, pp. 523-532, Jan. 2007.

[3] "NI 5640R IF Transceiver User Guide, " National Instruments, Austin, TX, http://www.ni.com/pdf/manuals/374603a.pdf, April 2007.

[4] "NI PCI-5640R Specifications" National Instruments, Austin, TX, http://www.ni.com/pdf/manuals/371620b.pdf , April 2007.

[5] "NI PXI-5670 RF Vector Signal Generator Hardware User Manual", National Instruments, Austin, TX, http://www.ni.com/pdf/manuals/rfsg_um .pdf, March 2006.

[6] "NI 5660 RF Vector Signal Analyzer User Manual", National Instruments, Austin, TX, Aug.2004.http://www.ni.com/pdf/manuals/371237d.pdf.

[7] L. Balmer, *Signals and Systems: An Introduction (2nd Edition),* Prentice Hall, 1997.

[8] A. Hyvärinen, J. Karhunen and E. Oja. *Independent Component Analysis,* John Wiley & Sons, 2001.

[9] A. Hyvärinen and E. Oja., "A fast fixed-point algorithm for independent component analysis". *Neural Computation*, Vol. 9, No. 7, pp. 1483-1492, Oct. 1997.

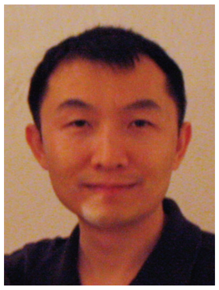
[10]J.A. Rice, *Mathematical Statistics and Data Analysis (Second edition),* Duxbury Press, 1995.

[11]P. J. Hawrylak, A. Ogirala, J. T. Cain, and M. H. Mickle. "Automated Test System for ISO 18000-7-Active RFID", *IEEE 2008 International conference on RFID,* pp. 9-18, 2008.

[12]"Virtex-5 FPGA XtremeDSP Design Considerations", Xilinx,www.xilinx.com/support/documentation/user_guides/ug193.pdf, 2009.

Yuan Sun received his B.S. and M.S. in Electrical Engineering at Southeast University, Nanjing, China in 2004 and 2007, respectively. Currently, He is a Ph.D. candidate in Electrical Engineering and research assistant with the RFID Center of Excellence at the University of Pittsburgh, USA. His research interests include ISO18000-6c UHF Passive RFID, high performance FPGA based digital design, embedded systems design, and network scheduling.

Peter J. Hawrylak received his B.S., in computer engineering, M.S. of electrical engineering, and Ph.D. of electrical engineering at the University of Pittsburgh, Pittsburgh, PA in 2002, 2004, and 2006 respectively. He is currently a research associate with the RFID Center of Excellence at the University of Pittsburgh, PA. His current research interests are in the areas of sensor networks, RFID, embedded system design, and system on a chip (SOC) design. Dr. Hawrylak is a member of the IEEE and IEEE Computer Society.



Zhi-Hong Mao received the dual Bachelor's degrees in automatic control and mathematics and the M.Eng. degree in intelligent control and pattern recognition from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, the M.S. degree in aeronautics and astronautics from Massachusetts Institute of Technology (MIT) in 2000, and the Ph.D. degree in electrical and medical engineering from the Harvard-MIT Division of Health Sciences and Technology, Cambridge, MA, in 2005. He has been an Assistant Professor in the Department of Electrical and Computer Engineering and the Department of Bioengineering, University of Pittsburgh, Pittsburgh, PA, since 2005.

Marlin H. Mickle is Nickolas A. DeCecco Professor in Electrical and Computer Engineering (Primary), Professor of Computer Engineering, Telecommunications, and Industrial Engineering at the University of Pittsburgh. He is the Director of the RFID Center of Excellence. He received the B.S.E.E., M.S.E.E., and the Ph.D. University of Pittsburgh in 1961, 1963, and 1967. Marlin received the Carnegie Science Center Award for Excellence in Corporate Innovation - 2005; he has 21 patents and received the Pitt Innovator Award 2005, 2006, 2007 and 2008; 1988 Recipient of the Systems Research and Cybernetics Award of the IIASSRC, a member of the AIDC100, and he is a Life Fellow of the IEEE.