COLLISION RESOLUTION IN ISO 18000-6C PASSIVE RFID COMMUNICATION

by

Yuan Sun

B.S. in Electrical Engineering, Southeast University, China, 2004

M.S. in Electrical Engineering, Southeast University, China, 2007

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2009

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Yuan Sun

It was defended on

September 4th, 2009

and approved by

Ching-Chung Li, Professor, Electrical and Computer Engineering Dept.

J.T.Cain, Professor Emeritus, Electrical and Computer Engineering Dept.

Joseph Kabara, Assistant Professor, Information Science Dept.

Ronald G. Hoelzeman, Professor, Electrical and Computer Engineering Dept.

Zhi-Hong Mao, Assistant Professor, Electrical and Computer Engineering Dept.

Dissertation Director: Marlin H.Mickle, Professor, Electrical and Computer Engineering Dept.

COLLISION RESOLUTION IN ISO 18000-6C PASSIVE RFID COMMUNICATION

Yuan Sun, PhD

University of Pittsburgh, 2009

According to the ISO 18000-6C passive RFID standard, the tags rely on limited energy harvested from the reader carrier wave rather than an internal power supply to perform logic functions and backscatter signals. The reader receives the tag's backscattered response, and then decodes the tag signal in order to access the tag information. However, 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 signal. Because the collision signal violates the encoding as specified in the standard, the reader is not able to decode it using its built-in circuitry that is designed for non-colliding tag responses. Therefore, the reader fails to complete the inventory for tags in the field in this case, which degrades the overall performance of the passive RFID system requiring retries.

This research focuses on resolving the two-tag collision signal with extensions to more tags. A preliminary tag data acquisition system has been developed along with an ISO 18000-6C conformance test platform, which consists of an FPGA-based software defined reader. The collision signal is obtained from the data acquisition system and processed by the FPGA in real time. Two types of collision resolving algorithms based on phase and amplitude characteristics of the collision signal are developed and simulated using LabVIEW on a host PC and then

realized with a National Instruments FPGA development board NI5640R. These two algorithms deal with the two-tag collision situation with and without a distinct phase shift individually, and they can be unified. As an extension to multiple tag collision resolution, an advanced statistical signal processing method using Independent Component Analysis (ICA) is researched for the three-tag collision situation. The ICA simulation is performed using LabVIEW on the host PC, and then implemented on the target FPGA development board. Performance analysis and comparison are presented to prove the efficiency and timing conformance of the proposed methods to the standard. Finally, the collision signals acquired from moving tags are resolved using the amplitude mapping method to prove the method compatibility on dynamic tags.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

My first experience with electrical components was twenty three years ago, when I was at the age of four. My father, a senior electrical engineer, was testing a television set and that moment is what started my interest in electrical engineering. I would like to thank my father, Xiaoming Sun, for all of the generous mentoring and guidance he has provided since my childhood. On the eve of starting my American dream, three years ago, I experienced the passing away of my beloved mother, Weihua Huang. This wonderful woman gave me all of her love, hope and blessings. I built my first perceptions of the USA from this great woman. Serving as a successful manager in China Mobile for over a decade, she visited this country for several times. I will never forget her last words, "Son, go to America, to witness and study the most advanced technology in the world."

Today, on the eve of obtaining my Ph.D., I am deeply indebted to my advisor, Dr. Marlin H.Mickle. During my most confusing periods of time, Dr.Mickle's lucid explanations and patient guidance always provided me with enlightment. He would always say "Get things done, Get things done efficiently, Get things done simply" and this is something that has served me well and I've learned to live by during my time at the RFID Center of Excellence.

I would also like to thank Dr. Zhi-Hong Mao, Dr. J.T. Cain, Dr. Ronald G. Hoelzeman, Dr. Joseph Kabara and Dr. C. C. Li for serving on my committee and providing me with helpful

suggestions. I am also greatly appreciative to Dr. Peter J.Hawrylak for his detailed, technical guidance. Without the help of these advisors, I would not have been able to accomplish this work in such a short time.

I would also like to thank the dear friends that I've made during my studies here at Pitt. First I would like to thank Samuel Dickerson for greatly helping me improve my English skills during the past two years, providing me with insightful interchanges about research, and for being a good friend. I would also like to extend my thanks to Dr.Ajay Ogirala and Michael Rothfuss for all the help along the way of my Ph.D.

1.0 INTRODUCTION

Because passive RFID systems make use of the electromagnetic spectrum, they are relatively easy to jam using energy at the right frequency when they are in a dense tag environment. Although this would only be an inconvenience for consumers in non timing-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 the military in a field of operation.

The collisions in passive RFID communications lie in two major categories: the reader collision and the tag collision (as shown in Figure 1). Reader collision occurs in a reader intensive environment when the signals from two or more readers overlap in time and frequency. In such a situation, the tag is unable to respond to simultaneous queries; Tag collision occurs in a tag intensive environment when multiple tags are present in the transmitting field of the reader. In such a 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.

The scope of this dissertation is to resolve the tag collisions in order to improve the reading efficiency of ISO 18000-6C RFID systems [1]. 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 a Binary Tree scheme [2]. The ISO 18000-6C RFID systems make use of the

dynamic framed slotted Aloha. However, the dynamic framed slotted Aloha is a probabilistic method that 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 select the same time slot to respond (e.g. when the reader enforces all tags to respond), the reader may receive an unrecognizable (un-decodable) mixture of signals. 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 purely collision avoidance by recovering the original tag signals from the collision at the reader side. The philosophy of this method is to resolve the collision after it occurs rather than to avoid its occurrence. It is also possible to combine the use of an anti-collision mechanism and the resolution of the collision signal in order to increase the efficiency of the reader-tag communication.

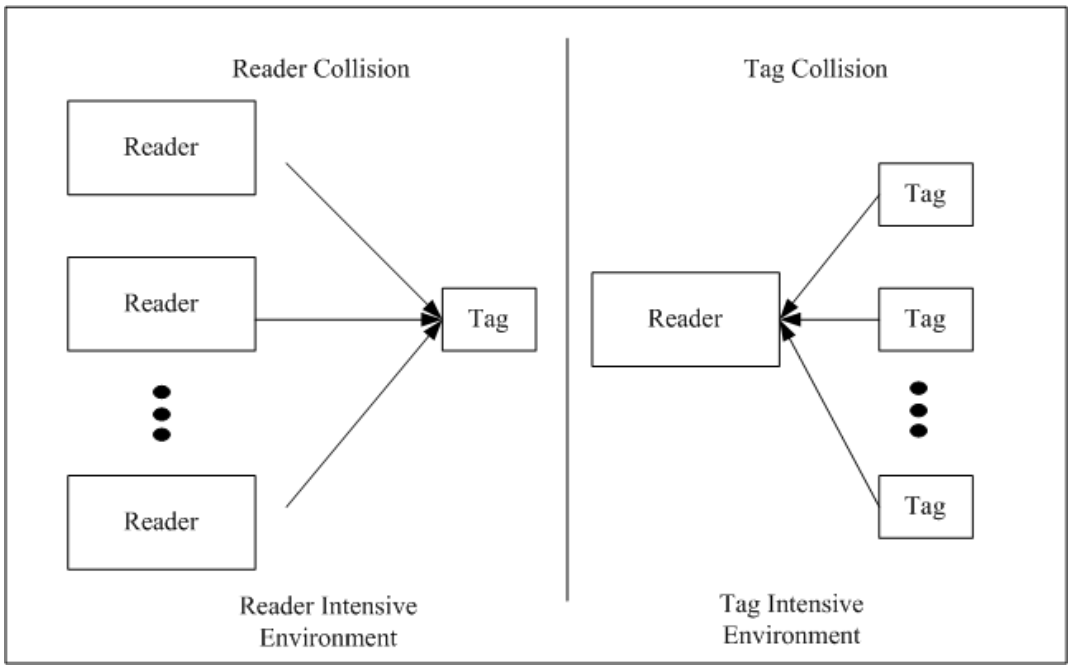

Figure 1. Passive RFID Collision Situations

In addition, the 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 RFID tags easy and inexpensive to deploy and maintain compared to active tags, the passive 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 real time 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 the test platform conforming to the ISO 18000-6C standard shall complete the signal decode and command assembly work in real time during each step of the handshake. If the reader fails to achieve this requirement, the tags will lose power and transit back to the initial state waiting for a new effective inventory round.

The timing limitation implies that the reader must resolve the tag collision in real time. If at least one of the tag responses can be resolved from the collision and decoded in the specified real time, 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 potentially can be performed in parallel which leads to a dramatic increase in system read rates.

1.1 BACKGROUND OF THE PROBLEM

1.1.1 The Link Timing of ISO 18000-6C Standard

The ISO 18000-6C standard specifies two categories of tag states: the inventory states and the memory access states as shown in Figure 2. 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 states. 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 reader command and the following tag response), which requires the reader to complete the decoding for the tag response and then sends out the next command within this turnaround time. If any transition step in the inventory round fails to complete in T, the tag will transit back to the Ready state (the initial state after power up). Table 1 lists the required turn-around time for typical tag BLF.

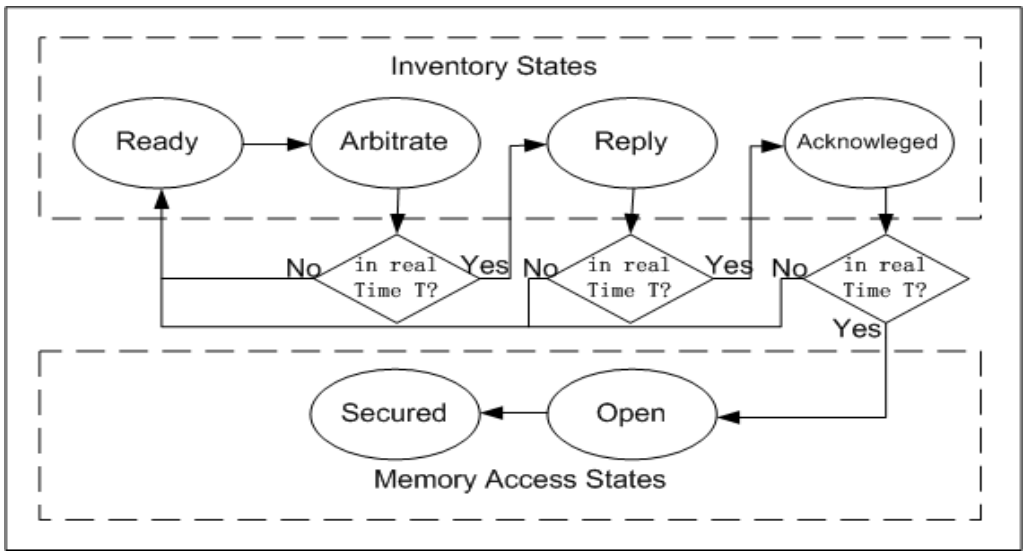

Figure 2. Tag States Transition

Table 1. Tag Turn-around Time at Typical BLF

BLF	Minimum	Maximum
64kHz	46.90μs	312.50μs
128kHz	23.45μs	156.25μs
256kHz	11.73μs	78.125μs
320kHz	9.38μs	62.50μs
640kHz	4.69μs	31.25μs

Corresponding to each state in the inventory round, there exists a three-step handshake as shown in Figure 3. At the first step (Step.1) of the handshake, the reader 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 following step (Step.2), the reader decodes the tag backscattered random number and attaches it to the command header of an acknowledge command, ACK, and then sends out 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) protected by a 16-bit CRC code. The tag then transits into the Acknowledged state. At the last step (Step.3), 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 code).

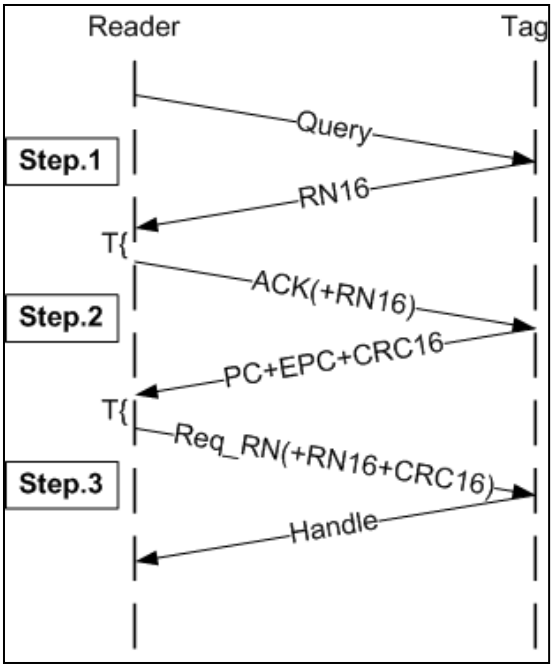

Figure 3. The Three Step Handshake

1.1.2 The Anti-Collision Mechanism in ISO 18000-6C standard

To start an inventory round for the tags in the field, the reader needs to send a Query command as the first step in the inventory round. The formation of the Query command and the tag corresponding response are shown in Figure 4. 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 the range of $[0, 2^{Q}-1]$ to load into its slot counter. If a tag, in response to the Query command, loads its slot counter with zero, 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, in the extreme case when Q equals 16 the probability for an N tag collision is $(1/65536)^{N-1}$.

***Query* command**

	Command	DR	M	TRext	Sel	Session	Target	Q	CRC-5
# of bits	4	1	2	1	2	2	1	4	5
description	1000	0: DR=8 1: DR=64/3	00: M=1 01: M=2 10: M=4 11: M=8	0: No pilot tone 1: Use pilot tone	00: All 01: All 10: ~**SL** 11: **SL**	00: S0 01: S1 10: S2 11: S3	0: *A* 1: *B*	0–15	

Tag reply to a *Query* command

	Response
# of bits	16
description	RN16

Figure 4. Query Command and Tag Response Formation

1.1.3 ISO 18000-6C Tag Baseband Encoding

According to the 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.

Figure 5 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.

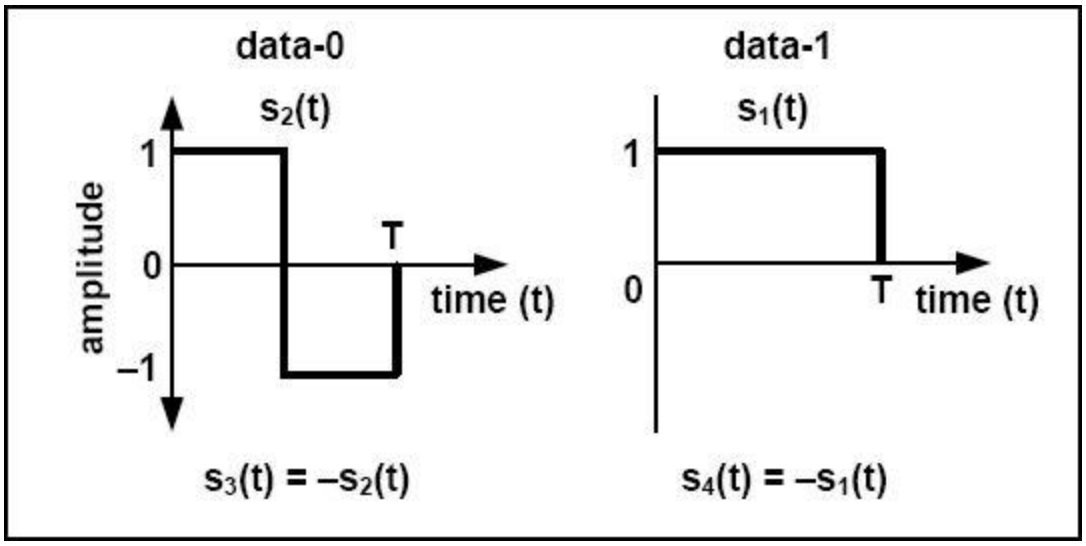

Figure 5. FM0 Baseband Basis Function

Figure 6 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].

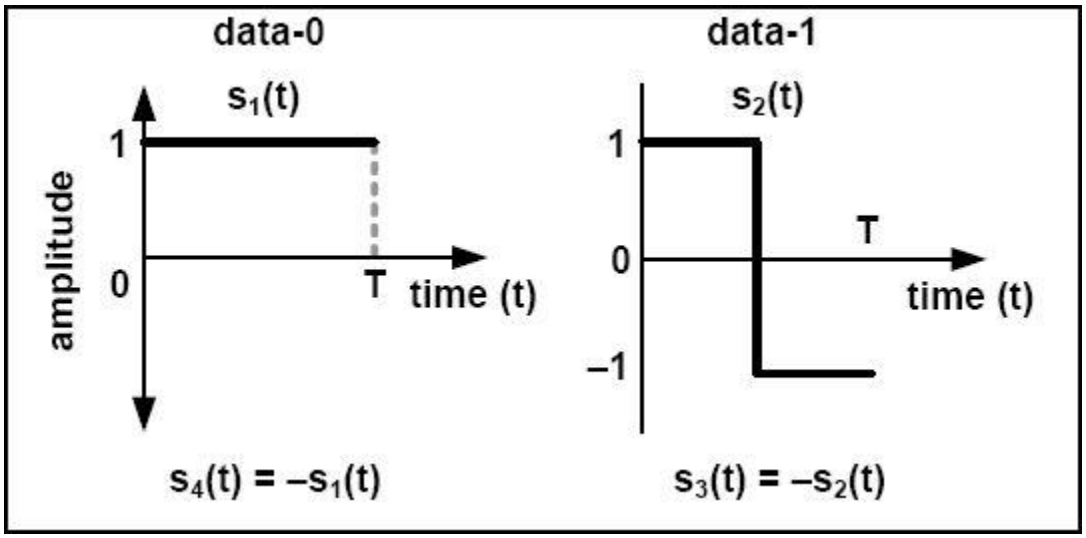

Figure 6. Miller Baseband Basis Function

Generally, the symbol of both FM0 and Miller can be categorized into formations as shown in Figure 7. Formation 0 features a edge transition in the middle of the symbol, while there is no edge transition in Formation 1 symbol.

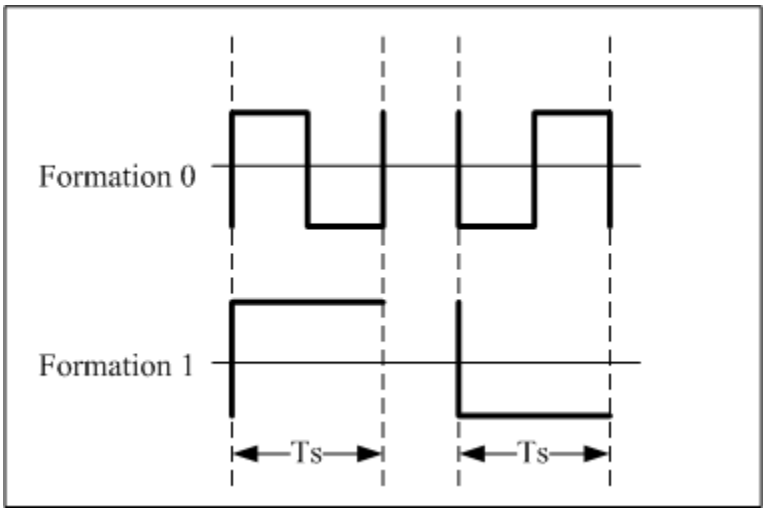

Figure 7. Generalized Tag Baseband Formation

1.2 STATEMENT OF THE PROBLEM

1.2.1 Overview

The focus of this research is to resolve the two tag collision situation with extension to three or more collision tags. The collision can be simulated by letting the reader set the Q in the Query command to a value of zero. The two tags will then both load 0 into their slot counter and respond with their 16-bit random numbers simultaneously. The two tag responses are supposed

to be linearly superimposed as will be discussed. The reader receives baseband collision signals after RF and IF down conversions and are thus the superposition of the two symbol formations of either FM0 or Miller encoding as discussed in 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 them. The downconverted two tag responses are normally different in magnitude. In addition, due to the capacitor variation in the tag chips (silicon), a phase shift (delay) may exist between the two responses even when the two tags are from the same manufacturer. (Detailed analysis of the two tag collision waveform characteristics will be introduced in Section 3.0) 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 significantly. Therefore, successful resolution of the two tag collision can significantly improve system data access efficiency. For example, the reader can selectively access one tag in a two tag collision situation as shown in Figure 8. As another example, with the reader collision resolution mechanism, the reader can simultaneously complete two independent inventory rounds for two tags as illustrated in Figure 9.

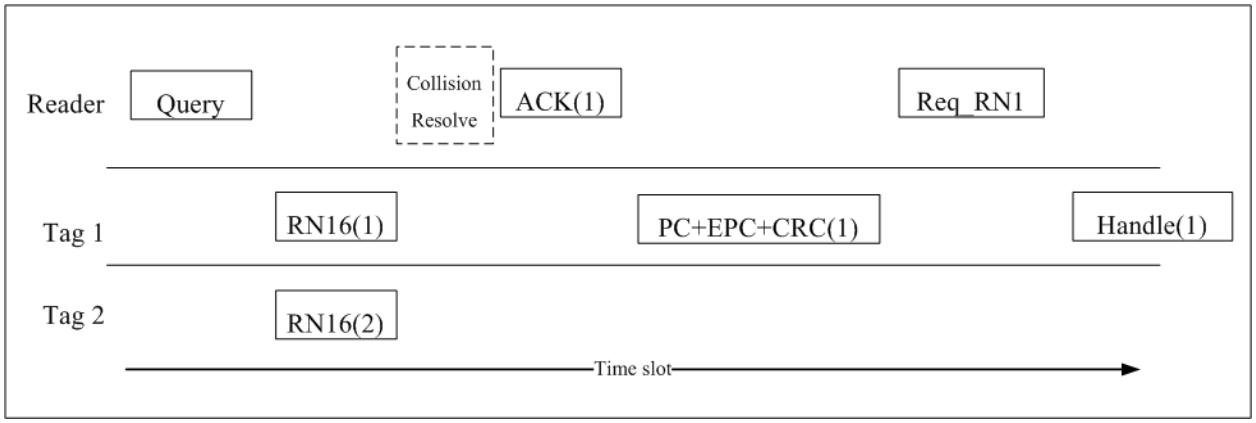

Figure 8. Selective Tag Access with Collision Resolution

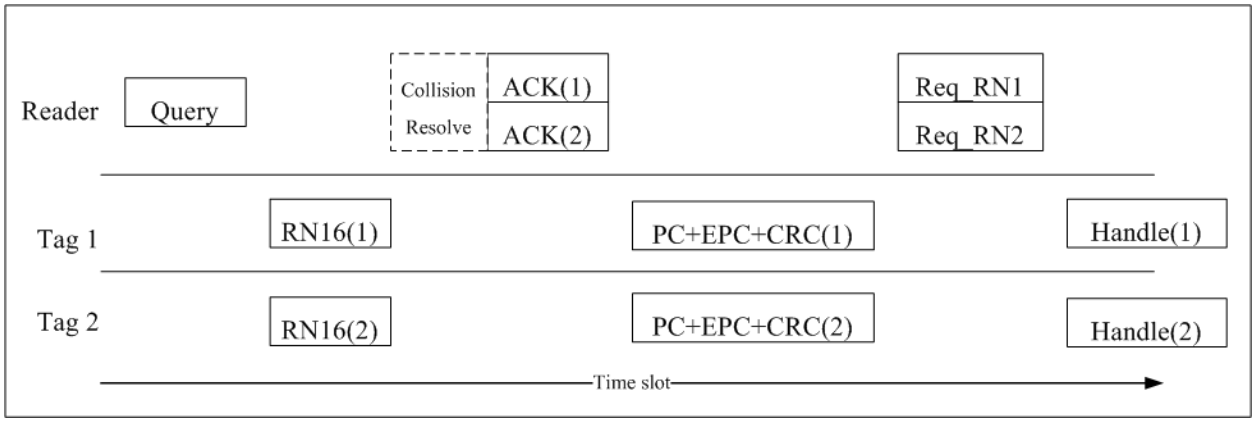

Figure 9. Simultaneous Tag Access with Collision Resolution

As discussed in the previous section, the standard mandates a real-time turnaround time specification. The collision resolution needs to be completed before the time out occurs. Therefore, two strategies can be considered for the timing of the collision resolution as shown in Figure 10. The first strategy starts the resolution after the complete acquisition of collision, while the second one 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 after it is finished. 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.

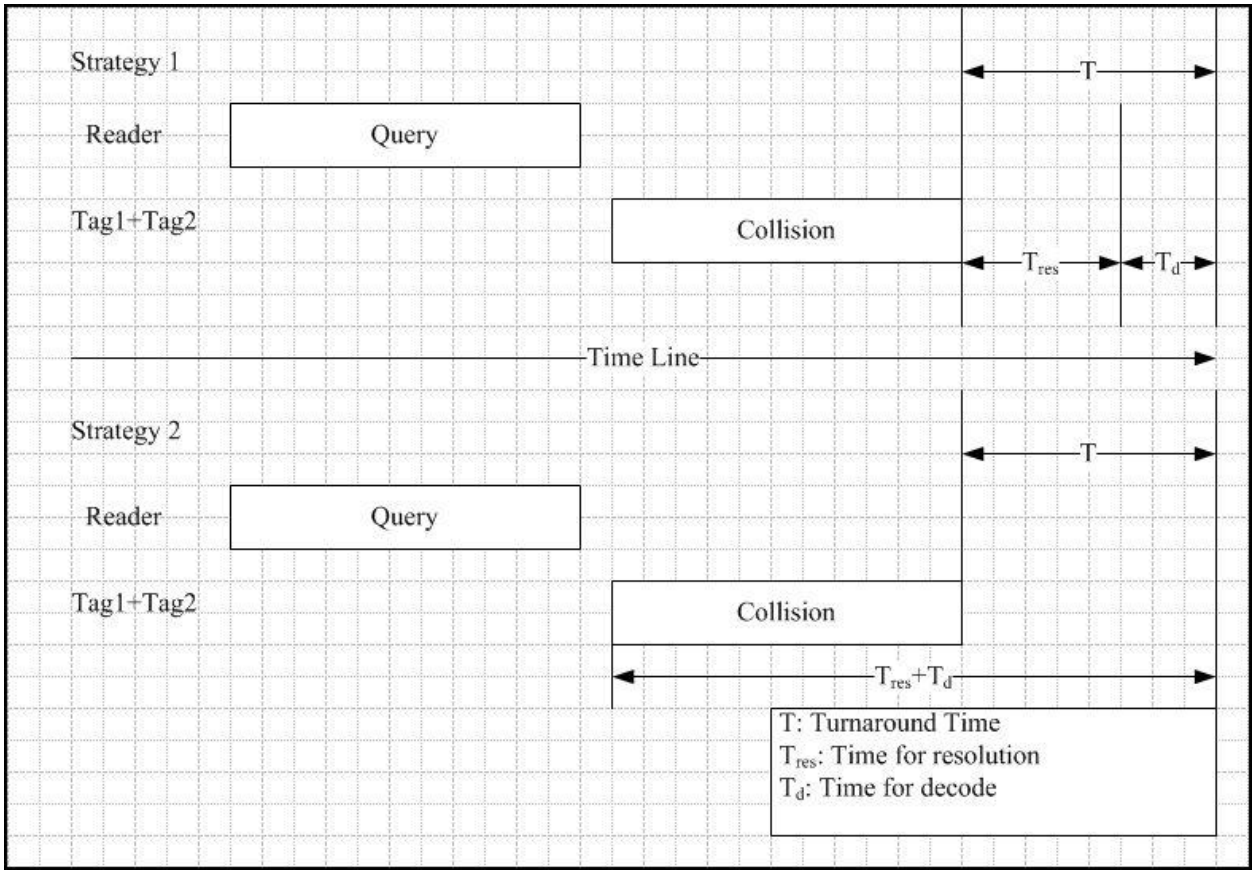

Figure 10. Collision Resolution Timing Strategy

1.2.2 Dissertation Work

This is a research to solve the tag collision resolution problem for ISO 18000-6C passive RFID communication systems.

Two straightforward online resolution methods are adapted to the two tag collision problem. The first method is the direct edge locating of the collision signal. The second one is the amplitude mapping to resolve a two-tag collision. The direct edge locating method will focus on resolving the two-tag collision with significant phase shift (larger than 15% of the symbol duration), while the amplitude mapping method will focus on resolving the collision with very

short or without phase shift. The two resolution methods can be unified by pre-processing the tag collision with a median filter in real time. As a part of the research, simulations are performed on a host PC as preliminary work to verify the functionality of the algorithms. The proposed algorithms are then optimized and implemented in one Xilinx Virtex-II Pro XC2VP30 FPGA on the NI5640R software defined transceiver board to verify their ability to resolve a two-tag collision within the real time limitation as specified in ISO 18000-6C standard.

To extend the resolution to multiple tag (greater than two) collisions, a statistic signal processing method using Independent Component Analysis (ICA) is used. The ICA method resolves two or more tag collisions without the limitation of the direct edge locating and amplitude mapping at the expense of increased hardware for additional receiving channels and higher computation load. To decrease the computation load of ICA in order to achieve the required processing speed, the ICA algorithm is optimized by working on the down sampled collision signal without losing recovery accuracy.

Finally, the collision signal acquired from moving tags with speeds up to 40 miles/hour will be resolved by the amplitude mapping method using the FPGA to prove the compatibility of the research's application on dynamic tags. All of the above methods work on commercial off the shelf passive RFID tags.

In Summary, the dissertation work includes:

- Developing a conformance test platform and a data acquisition platform ISO 18000-6C Passive RFID systems
- Study and Analysis of 2-tag and 3-tag collision signal acquired from both the stationary tags and moving tags

- Host PC Simulation and FPGA implementation of a 2-tag collision resolution algorithm based on both the amplitude and phase shift characteristics of signals
- Host PC simulation and FPGA implementation of ICA for multiple tag collision resolution

2.0 DATA ACQUISITION PLATFORM

Supported by the RFID Center of Excellence at the University of Pittsburgh, an ISO 18000-6C RFID tag conformance test platform has been developed as a preliminary to the research for this dissertation. This platform works by sending the user specified reader command to the tags under test as the test stimulus, analyzing the acquired tag backscatter, and then checking the tag response baseband waveform against the standard. This platform can thus serve as a data acquisition platform to obtain the tag response signal and the collision signal, which are the source signals in interest 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 the best device to be selected. Available commercial readers are alternative platforms, however their architectures are fixed after 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 is usually the major R&D cost if a hardware modification is required. Unlike microprocessors, FPGAs are easier to make timing accurate in their Hardware Description Language (HDL) programming nature. 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 these ends, an FPGA based software defined radio architecture is employed to implement the 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 Instrument (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.

2.1 PLATFORM ARCHITECTURE

The reconfigurable software defined radio test platform consists of two major parts:

1. The hardware, which includes the signal baseband, the intermediate frequency and the RF front end.
2. The software running on a PC controls the hardware and analyzes the backscattered signal.

2.1.1 Platform Hardware

The architecture of the platform hardware connection is shown in Figure 11. 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-based 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. An Agilent E4443A real time spectrum analyzer is employed as an auxiliary monitor of the RF communication process in the air interface. 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.

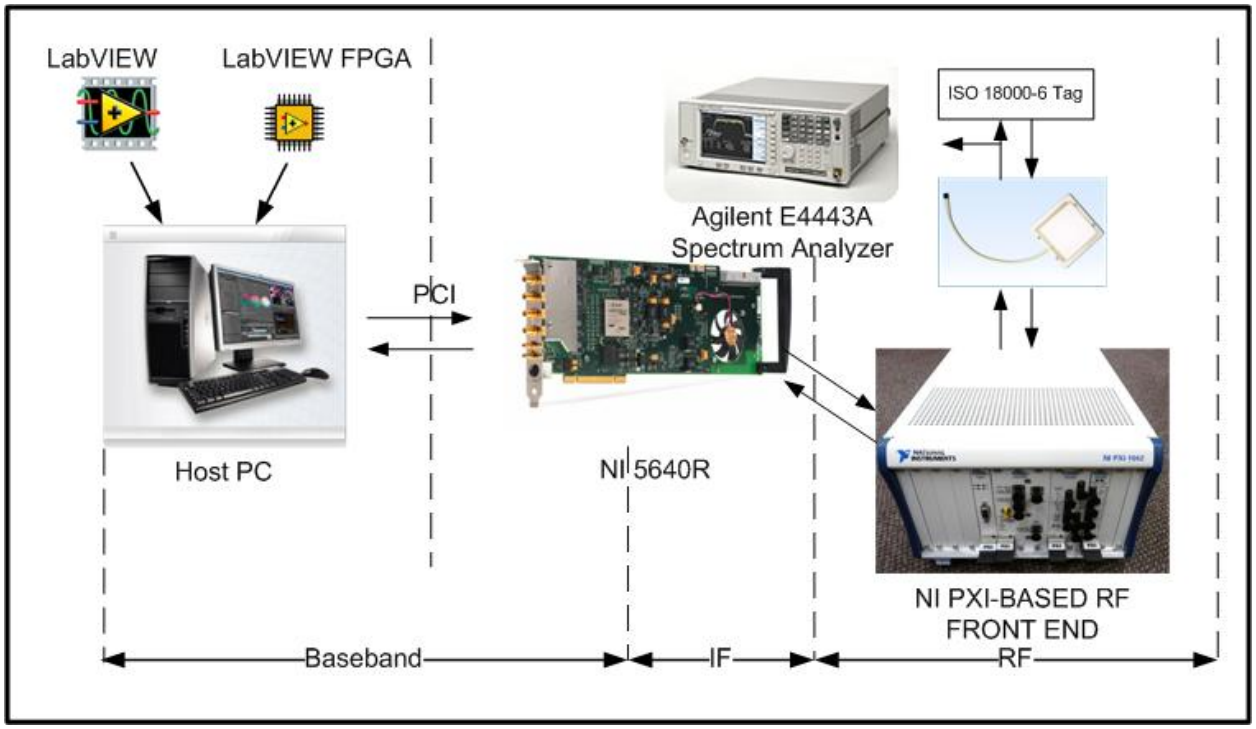

Figure 11. Platform Hardware Architecture

2.1.2 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 configured 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, the 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 remote panel interface, the host PC can be accessed by remote machines through Ethernet/Internet connection in a Client-Server relationship. 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 data acquisition and conformance test.

Figure 12 illustrates the software workflow of the host PC software.

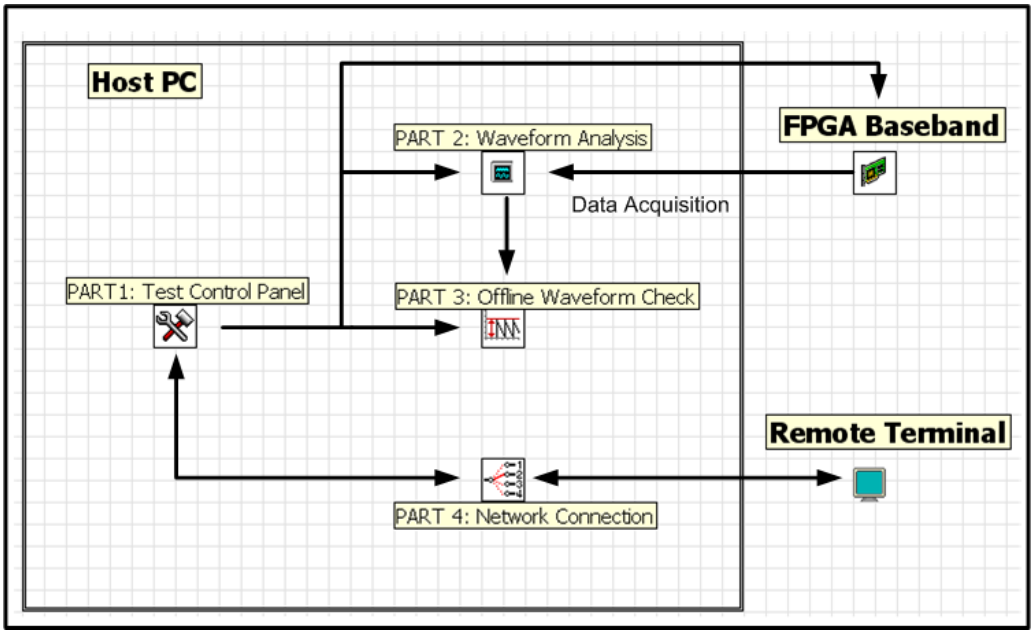

Figure 12. Platform Software Architecture

2.2 STANDARD REALIZATION AND DATA ACQUISITION

2.2.1 Standard Realization

The FPGA baseband realizes the logic function of an ISO 18000-6C conforming 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 encoding 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 Figure 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 to 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.

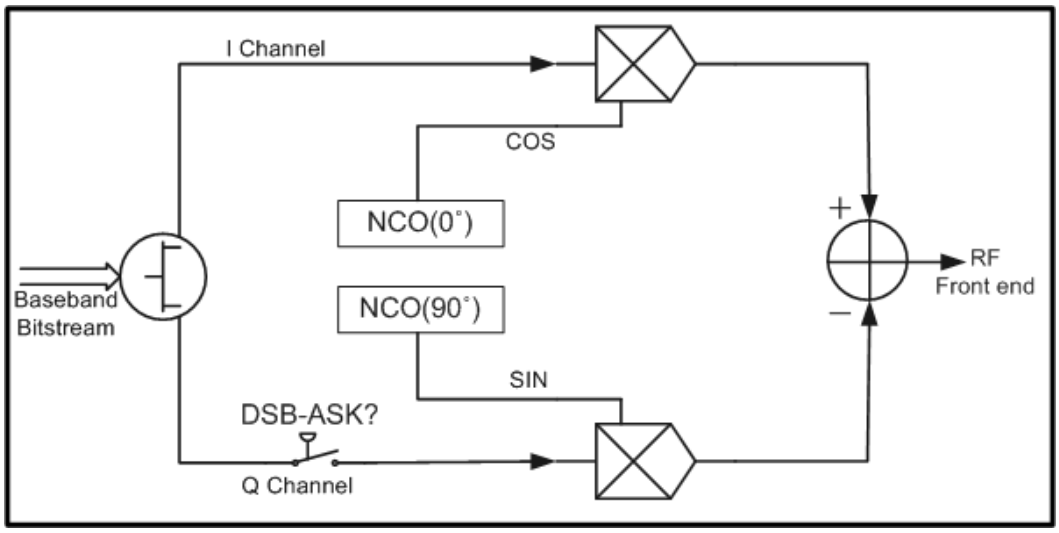

Figure 13. ASK Modulation Block

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 the ISO 18000-6C standard and make the outputting 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 outputting 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 Figure 14.

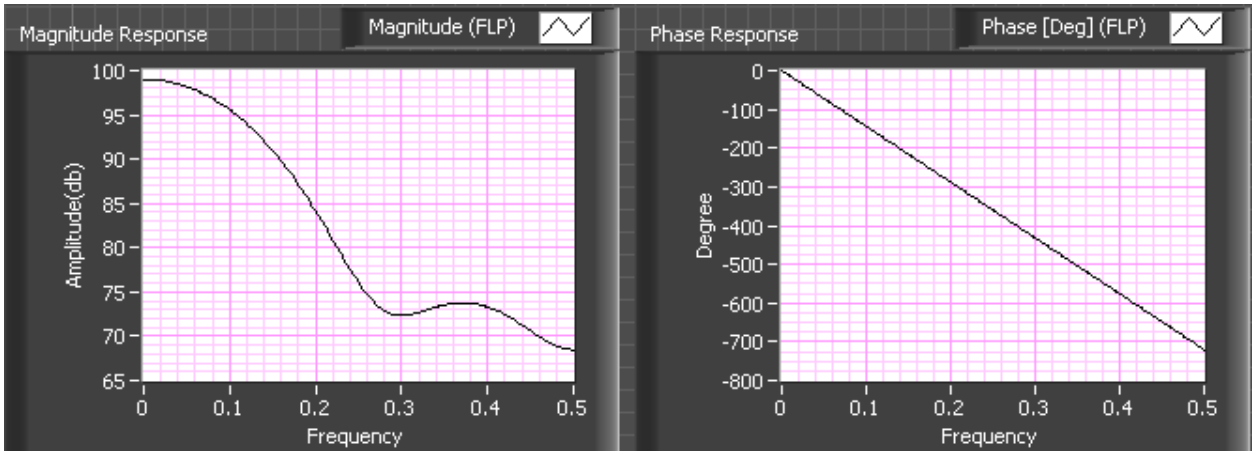

Figure 14. FIR Filter Magnitude and Phase Response

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. Figure 15 shows the realization of the filter in the LabVIEW FPGA Module and its corresponding effect. Comparing the baseband outputs, it can be seen that the under/over shoots in command signals are significantly quenched after the smoothing.

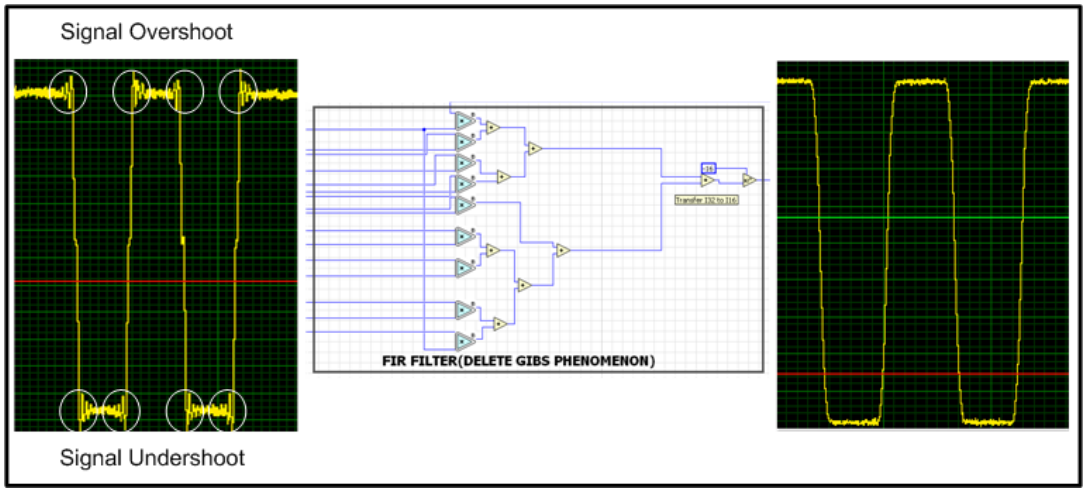

Figure 15. Signal Smoothing with 8-order FIR Filter

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 code, the decoding can be based on edge detection and pulse width measuring. Figure 16 shows the architecture of the decode module. 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 infrastructure of the handshake processing unit is shown in Figure 17. 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 a 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. Figure 18 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.

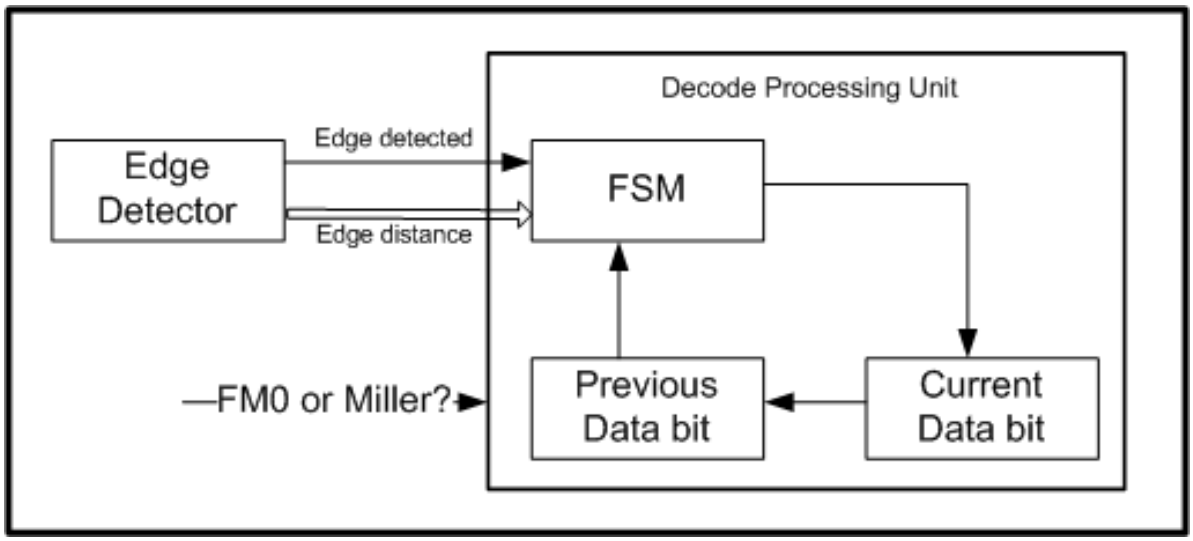

Figure 16. Decode Module Architecture

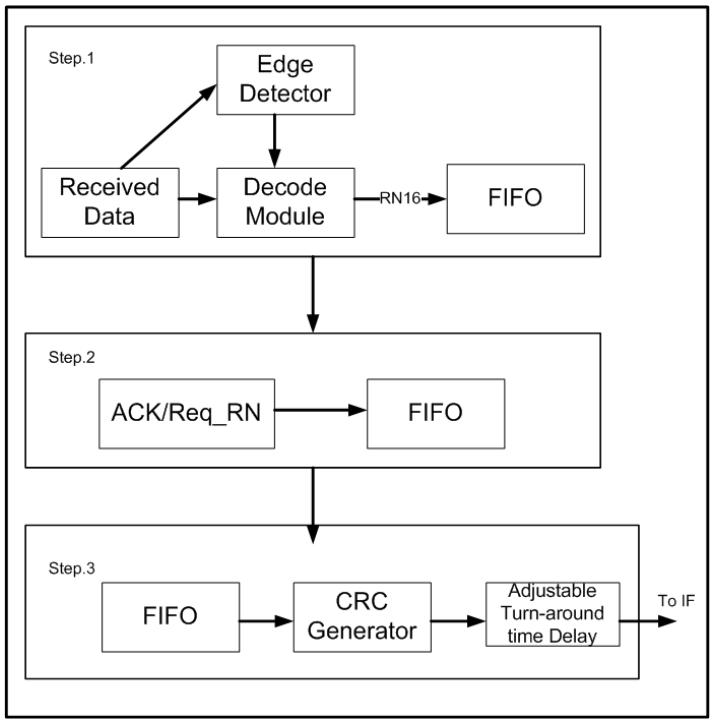

Figure 17. Handshake Processing Unit

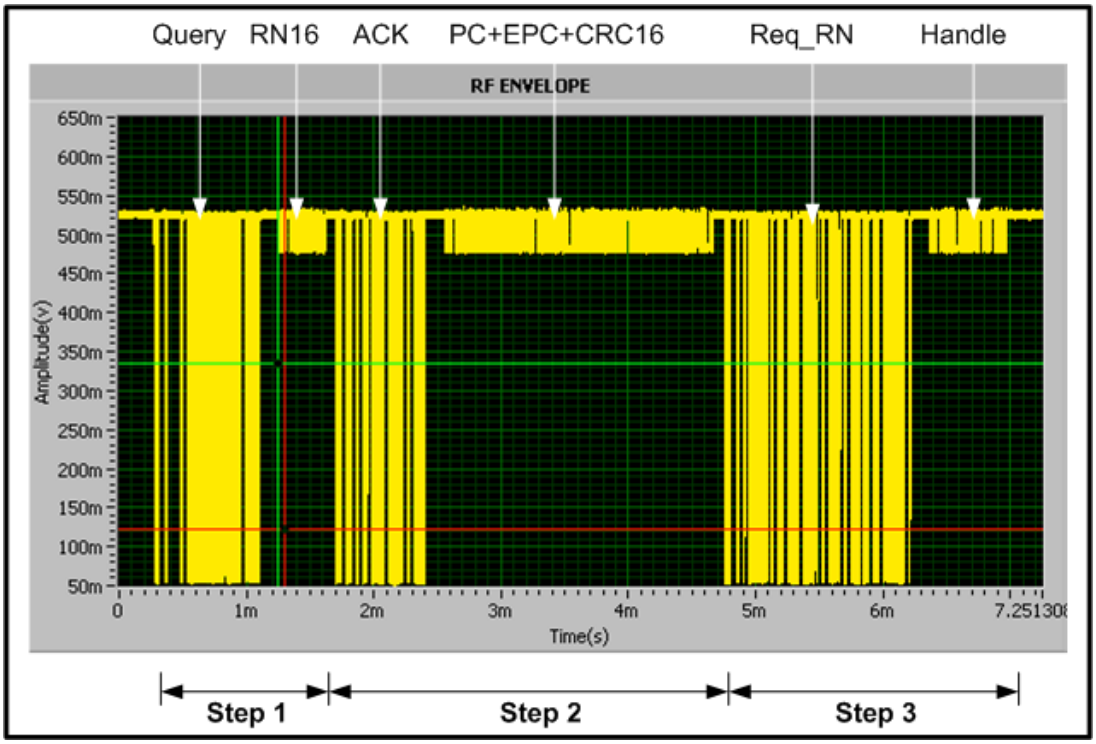

Figure 18. Acquired Inventory Round

2.2.2 Data Acquisition with the Platform

As introduced in Section 2.1.1, the test stimulus reader command as well as hardware details can be configured 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 parameters specified by the standard such as Tari value, TRcal, and RTcal can 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 distance 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. Figure 19 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. Figure 20 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.

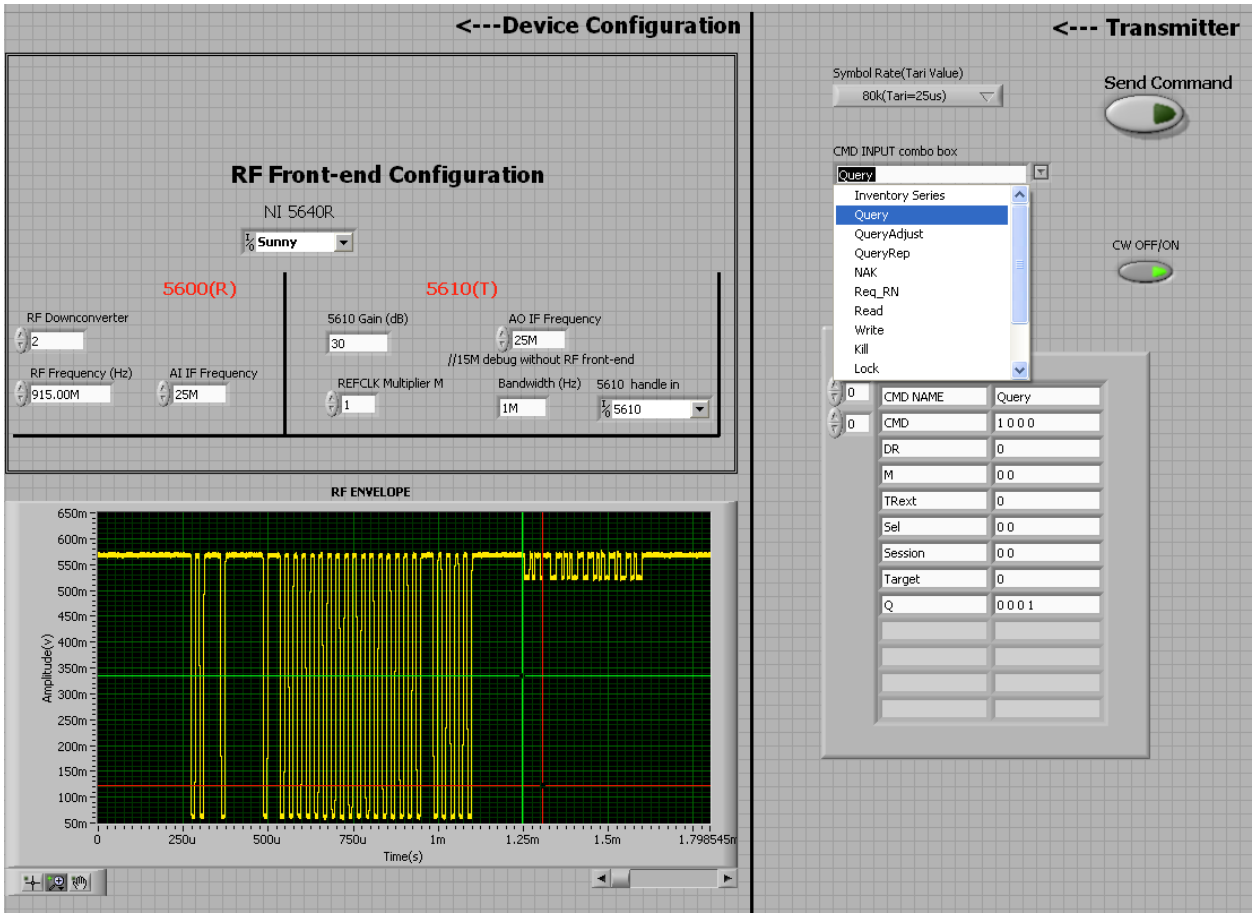

Figure 19. Test Control Panel

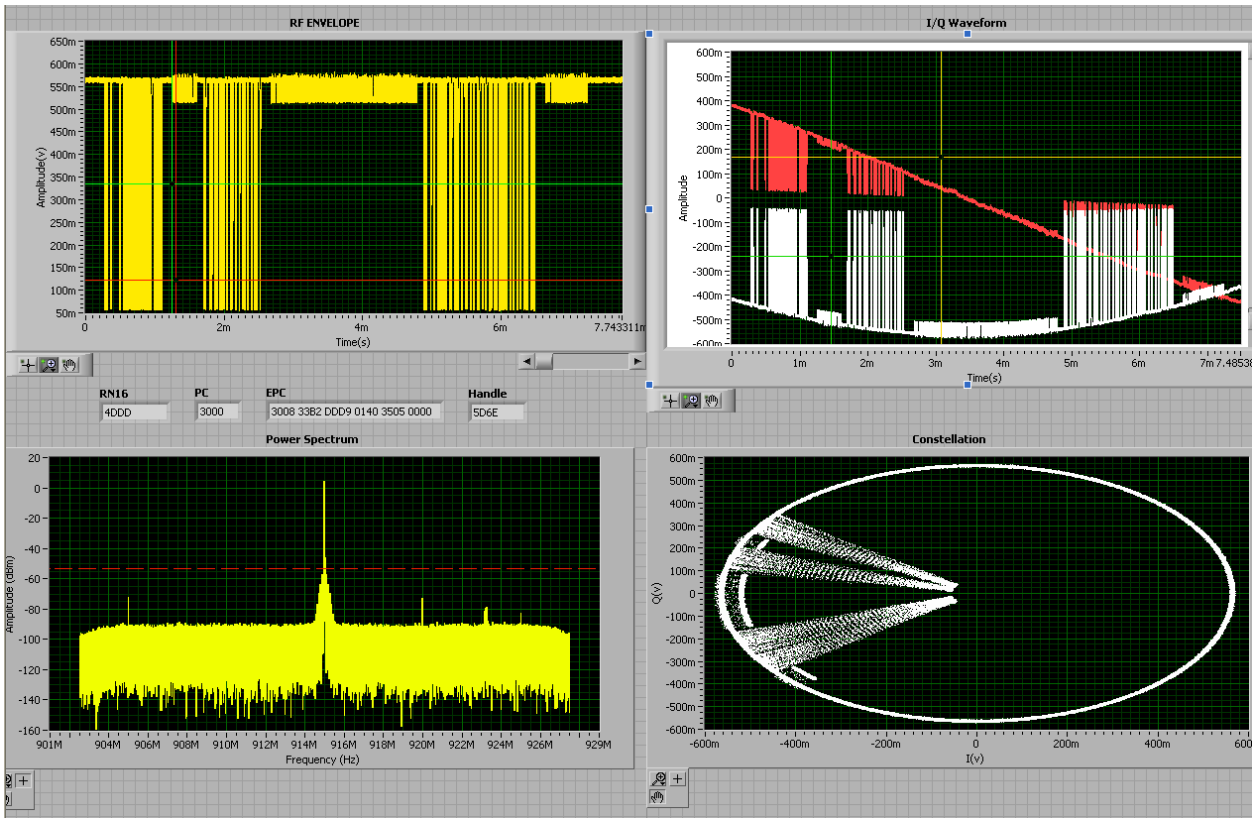

Figure 20. Waveform Analysis Module

3.0 TAG COLLISION SIGNAL CHARACTERISTICS ANALYSIS

3.1 TAG COLLISION SIGNAL CHARACTERISTICS

Because this research focuses 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 in Section 2.0 . By configuring the command parameter, the platform sends out a Query command with its Q equaling zero, and thus forces the two tags in the field to transmit their 16-bit random numbers simultaneously. By doing so, the two-tag collision situation is simulated. The RF carrier wave frequency for the air interface is set at 915MHz, and the IF frequency is set at 15MHz. At the receiver side, the collision signal is acquired by the platform, and then downconverted to the baseband signal, which is sent into the FPGA.

During the collision signal acquisition, in the vertical direction, the two tags, T1 and T2, are placed two inches from the center of the reader antenna as shown in Figure 21, while they are symmetric to the center and spaced by five inches from their individual antenna centers in horizontal direction. As shown in Figure 22, collision signal of tags from two different manufacturers are generated and captured. From the LabVIEW signal analysis module on the host PC, an acquired collision signal is visualized as shown in Figure 23.

Figure 21. Tag Positioning (Vertical Direction)

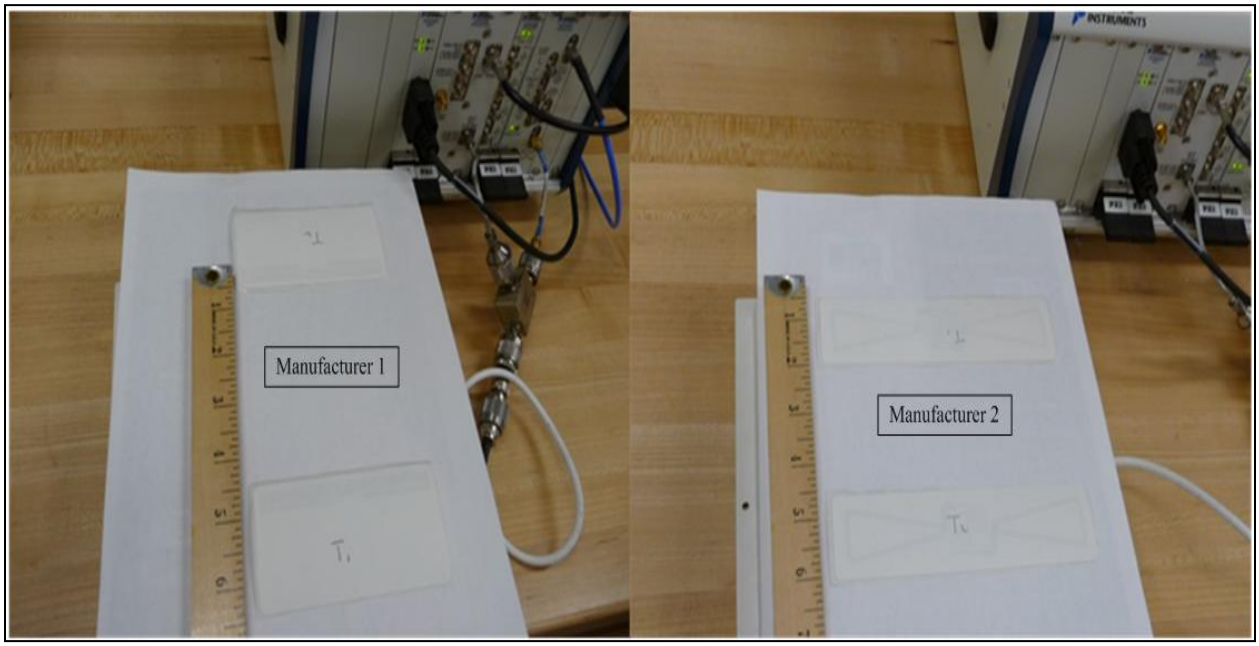

Figure 22. Tag Positioning (Horizontal Direction)

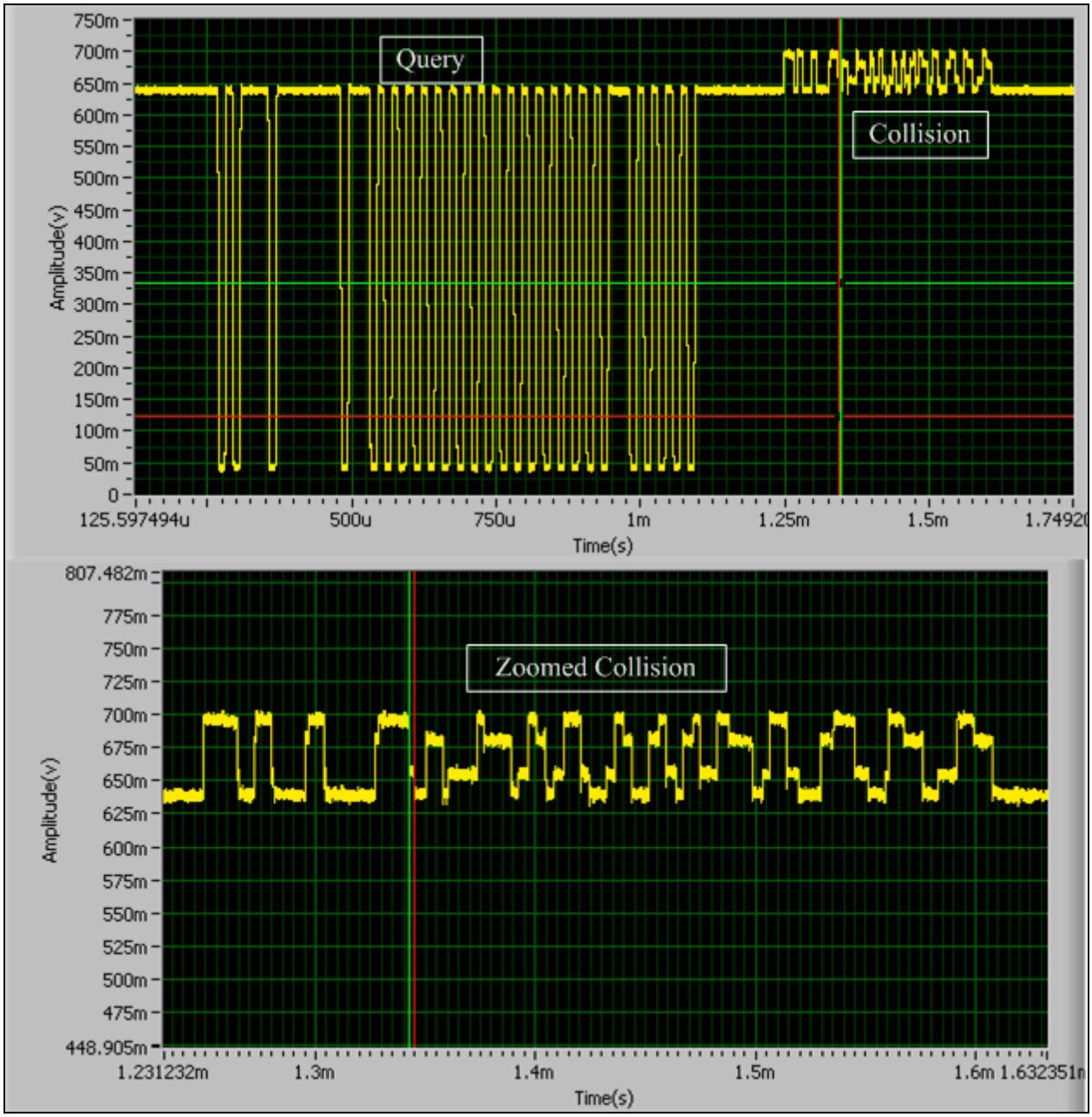

Figure 23. Acquired Collision Signal

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

1. The collision signal is the result of the linear superposition of tag responses, and the superposition follows a linear additive model as depicted in Figure 24.

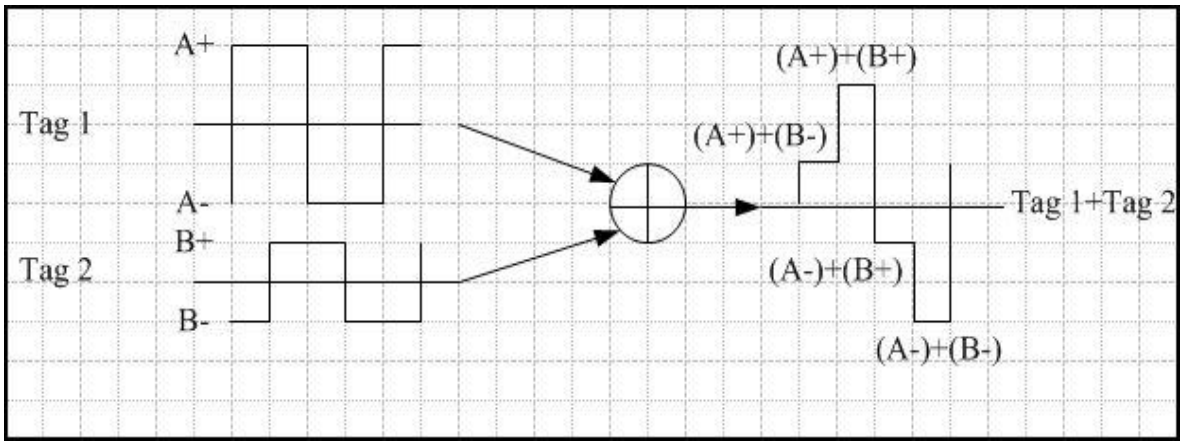

Figure 24. Linear Additive Model of Tag Responses

2. The received two tag responses are possibly different in magnitude. If the received two tag responses have the same magnitude, only three possible voltage levels can be generated when they are linearly superposed. Whereas, if they are different in magnitude, according to permutation, four possible voltage level 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+] ($A \neq B$). This assumption is proven by locating the four different voltage levels in the observed collision signal. The different received tag responses are caused by two major facts. The first one is that the two tag responses propagate to the receiver through different paths, they thus suffer from different path attenuation. The second fact is that the RF transmitting channel and the receiving channel of the data acquisition platform are combined by a circulator connected to one patch antenna as shown in Figure 25. 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 may be different in phase. Therefore, the received two tag responses suffer from different attenuation factors caused by carrier wave asynchronization with the receiver's local oscillator (LO).

Table 2. 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-)

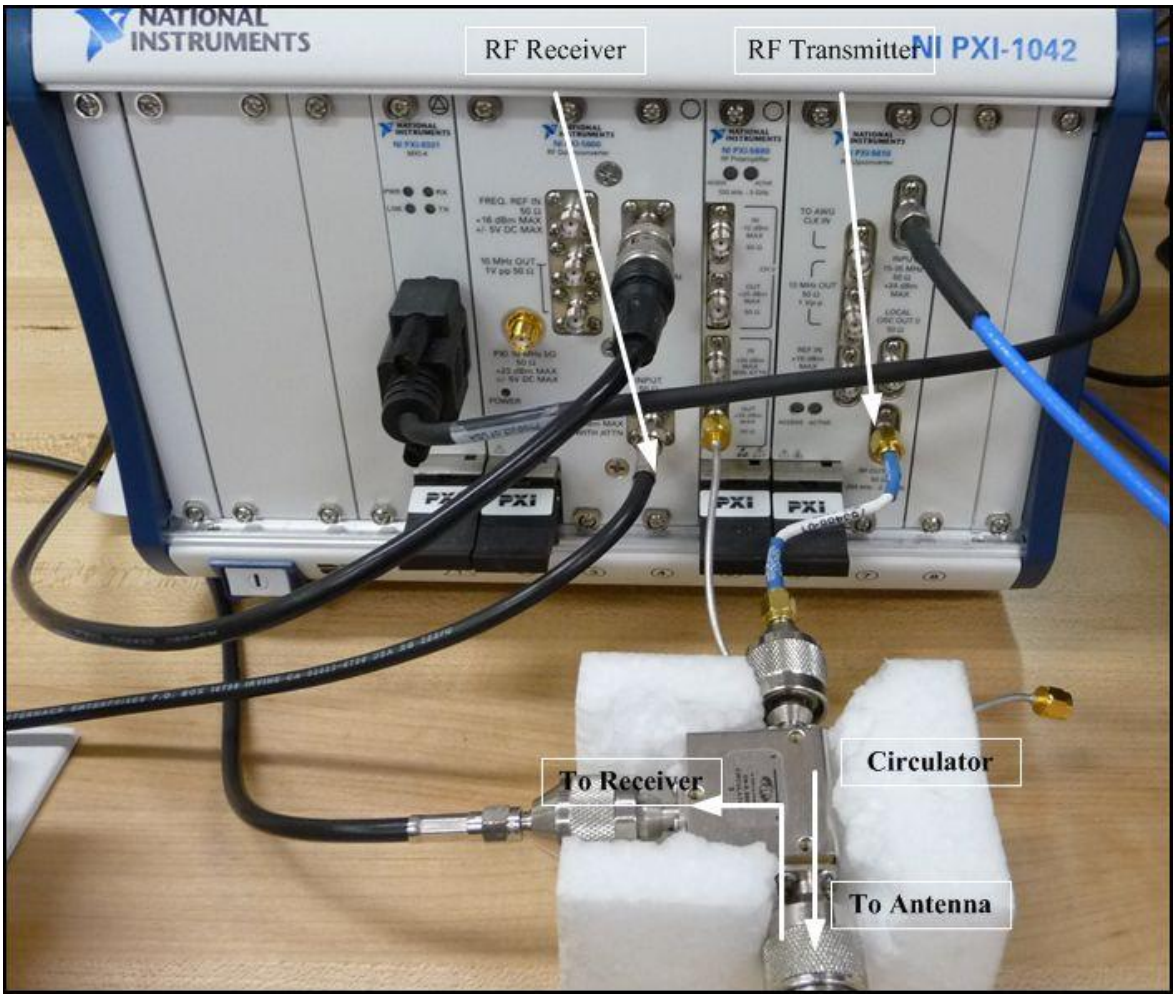

Figure 25. RF Front End Connection with Circulator

3. The two tag responses are added together with a phase shift (delay). The reason for this phase shift is due to two reasons. Due to the physical variations of the capacitors for 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 through different paths, which are 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 analysis. In summary, the observed phase shift in the collision signal is assumed to be due to the difference in the tag circuitry response speed.
4. The phase shift is not fixed during the backscatter, and it accumulates with time. Referring to the Standard, the reason of the accumulated phase shift is due to the tag response frequency deviation (BLF deviation) tolerance. As an example, this phenomenon can be observed in the zoomed tag response in Figure 26. The phase shift at the ending bit of the responses (in this case it is 2.486μs corresponding to the last white circle in Figure 26) is much larger than the initial phase shift (125ns, which implies at the beginning two tags almost respond to the reader simultaneously).

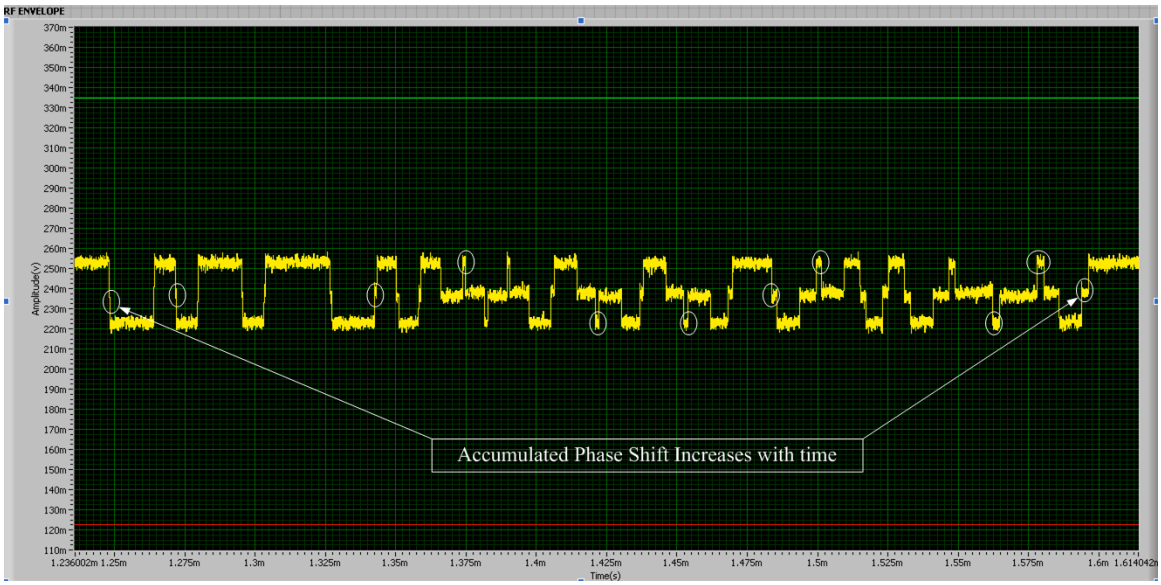

Figure 26. Accumulated Phase Shift

5. The average phase shift is inversely proportional to the tag BLF, and cannot exceed 25% 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 and Table 4. The unit of the phase shift in the tables is micro seconds. The data acquisition precision is 0.04µs (25MHz). Table 3 lists the measurement results of the phase shift among seven different tags from manufacturer 1. Table 4 lists the measurement results of the phase shift among seven different tags from manufacturer 2. For each measurement, the tags are placed together with two in a group. Figure 27 shows and compares the average phase shift of tags from each manufacturer at typical BLFs. As shown, the tags from Manufacturer 2 (red line) has a much larger phase shift than the tags from Manufacturer 1 (blue line), and thus the edge locating method is more suitable to resolve the collision of the former. Reciprocally, the amplitude method is more suitable for the latter as will be discussed in Section 4.0 and Section 5.0 . Figure 28 and Figure 29 show the percentage of average phase shift in the symbol duration for tags from each manufacturer separately. As

shown, the largest average phase shift takes 21.30% (<25%) of the symbol duration, when the BLF is 682 kHz and manufacturer number 2's tag is selected.

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, which is not fixed.

Table 3. Tag Response Statistics (Manufacturer 1)

BLF	0	1	2	3	4	5	6	Avg	Percentage(over Ts)	Multiple of Precision
64k	0.341	1.334	1.163	0.536	0.592	2.44	0.191	0.942	6%	24
128k	0.758	1.313	0.352	0.127	0.174	0.691	0.565	0.569	7.29%	15
256k	0.508	0.527	0.428	0.222	0.271	0.739	0.345	0.434	11.10%	11
341k	0.282	0.319	0.114	0.182	0.081	0.426	0.239	0.235	8%	6
682k	0.052	0.128	0.256	0.223	0.178	0.296	0.183	0.188	12.80%	4

Table 4. Tag Response Statistics (Manufacturer 2)

BLF	0	1	2	3	4	5	6	Avg	Percentage(over Ts)	Multiple of Precision
64k	4.072	1.023	3.69	1.444	2.679	2.9	1.436	2.463	15.70%	61
128k	1.303	0.72	1.409	1.328	0.959	1.429	1.3	1.207	15.50%	30
256k	0.389	1.105	0.558	0.5	0.888	1.127	0.795	0.766	19.60%	20
341k	0.704	0.461	0.265	0.54	0.476	0.338	0.491	0.468	15.90%	12
682k	0.417	0.264	0.376	0.183	0.377	0.408	0.169	0.313	21.30%	7

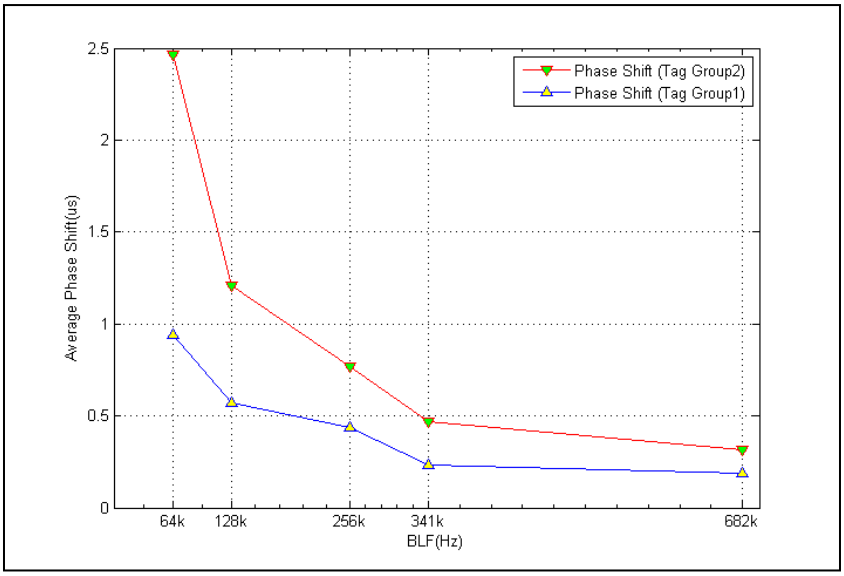

Figure 27. Tag Phase shift vs. BLF (Manufacturer 1 and Manufacturer 2)

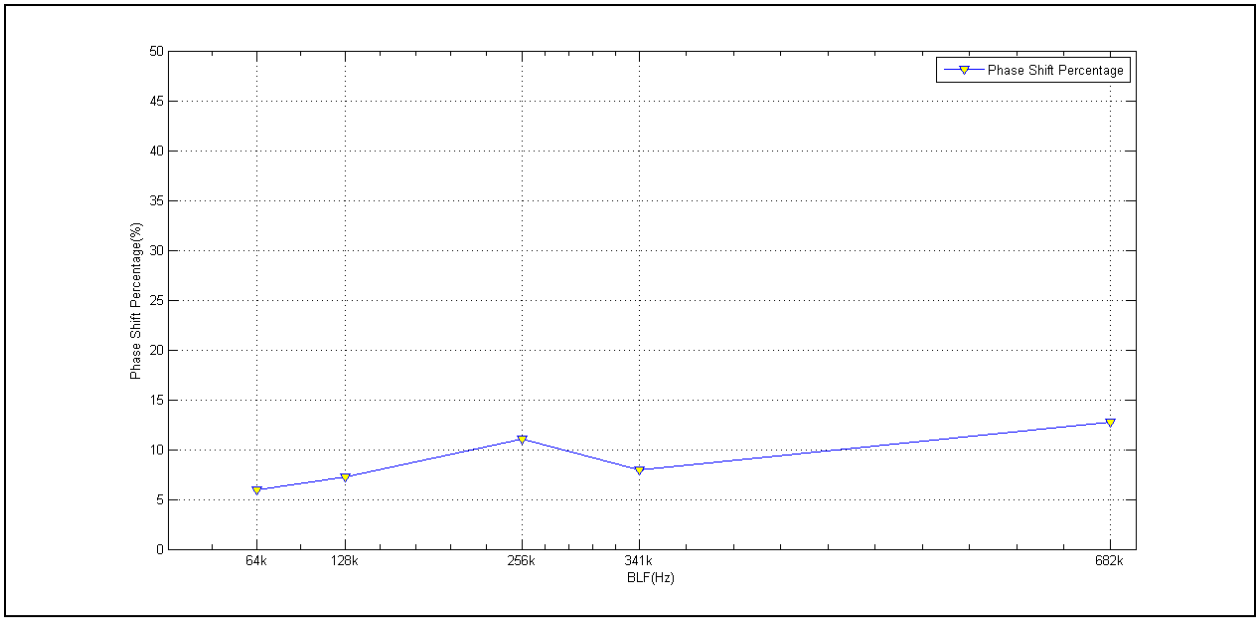

Figure 28. Tag Phase shift Percentage in Symbol duration vs. BLF (Manufacturer 1)

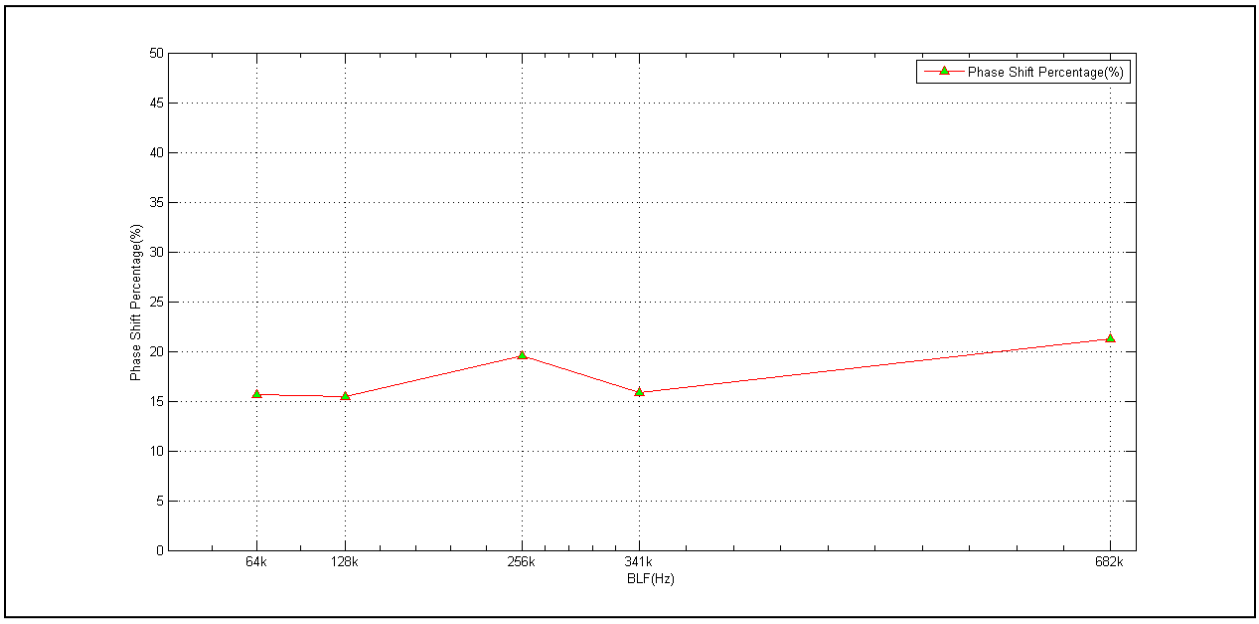

Figure 29. Tag Phase shift Percentage in Symbol duration vs. BLF (Manufacturer 2)

3.2 DATA PREPROCESSING

Before the actual collision resolution, a certain amount of data preprocessing is usually necessary to suppress the noise in the acquired collision signal, which is due to the imperfect RF receiving channel characteristics. Unlike the reader, which transmits commands with fixed baseband frequency, the tags BLF can vary in a wide range (normally from 64 kHz to 640 kHz). Therefore, improving the quality of the received baseband signal requires adapting tap coefficients for different typical BLFs if using traditional digital filters in baseband such as FIR and IIR filters. As an alternative to avoid the complicacy of designing FIR/IIR filters, a fixed point median filter using bubble sorting is realized in the FPGA baseband to preprocess the incoming collision signals. As shown in Figure 30, the SNR of the received tag response is significantly improved using a 10 order median filter.

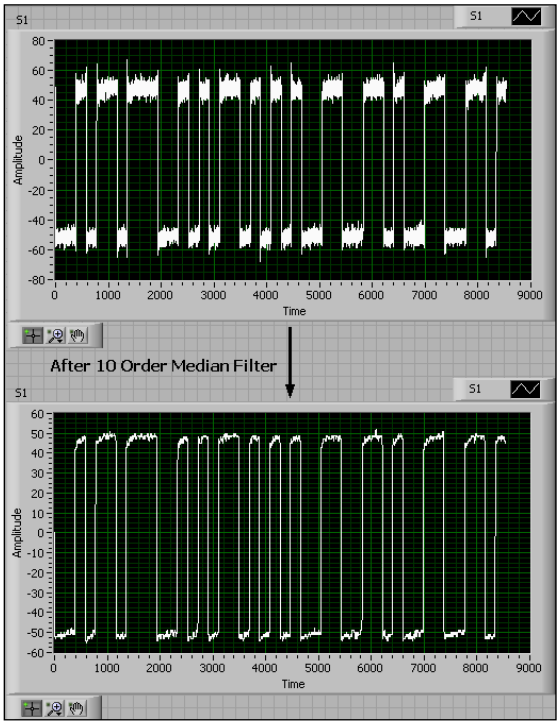

Figure 30. the effect of a 10-order median filter

A trade off can also be made between the complexity of the median filter and its effect as shown in Figure 31. Normally, a median filter with 4 orders, which guarantees a 3dB SNR improvement over original signal, is enough for the application.

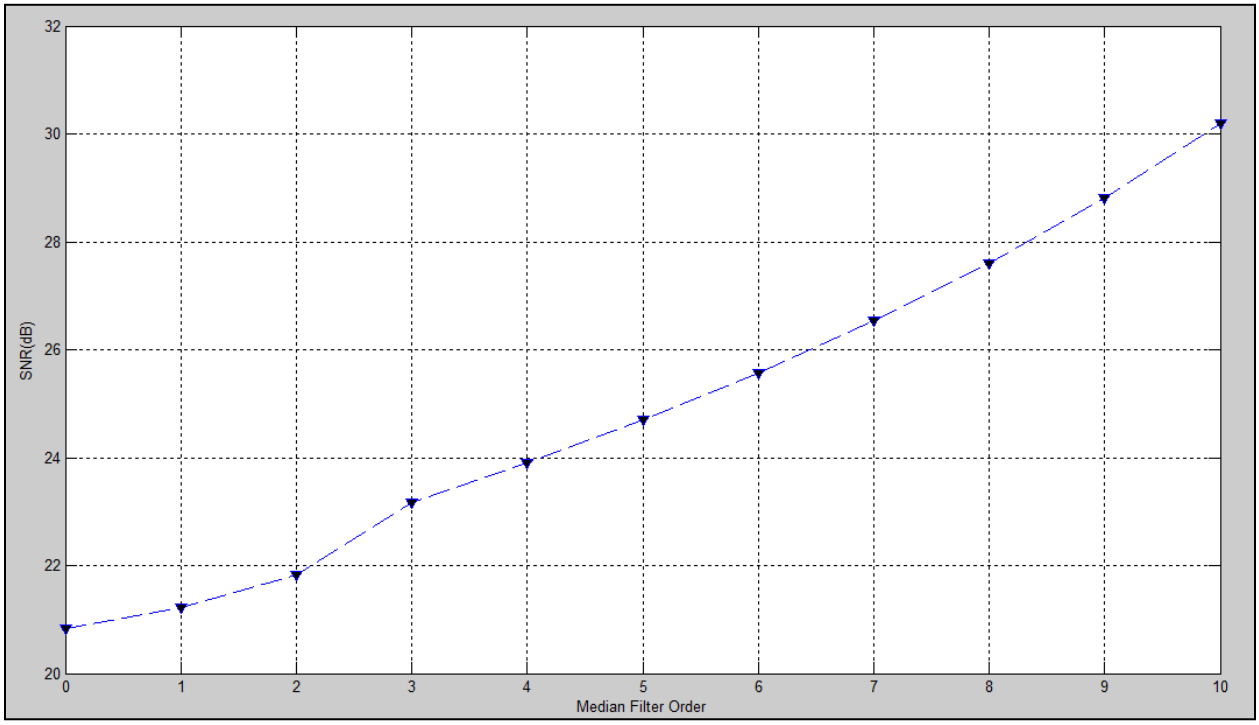

Figure 31. Median Filter order vs. Processed Signal SNR

4.0 RESOLUTION METHOD I：DIRECT EDGE LOCATING

Because a collision results in 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 because they normally propagates through different paths (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). Figure 32 illustrates the superposition of two formation-0 symbols, and the two edge transitions are maintained in the collision. As discussed in Section 3.1, the phase shift value is not fixed and can be accumulated over the collision duration due to the tag BLF deviation. Figure 33 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. Figure 34 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 responses can still be treated as being independent. It is possible to recover the information of each tag by locating the edge transition in each symbol duration from the start of each tag's response. This is based on the exact locating of each tag's start in the collision, which will be discussed in next section.

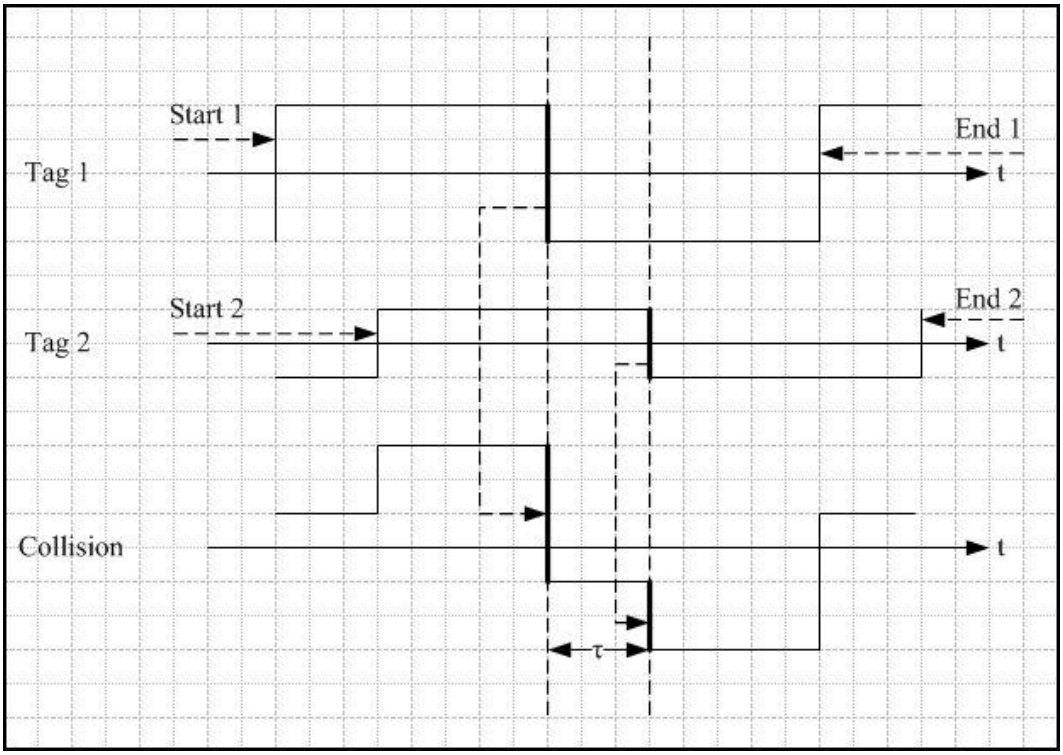

Figure 32. Superposition of Two Formation-0 Symbols

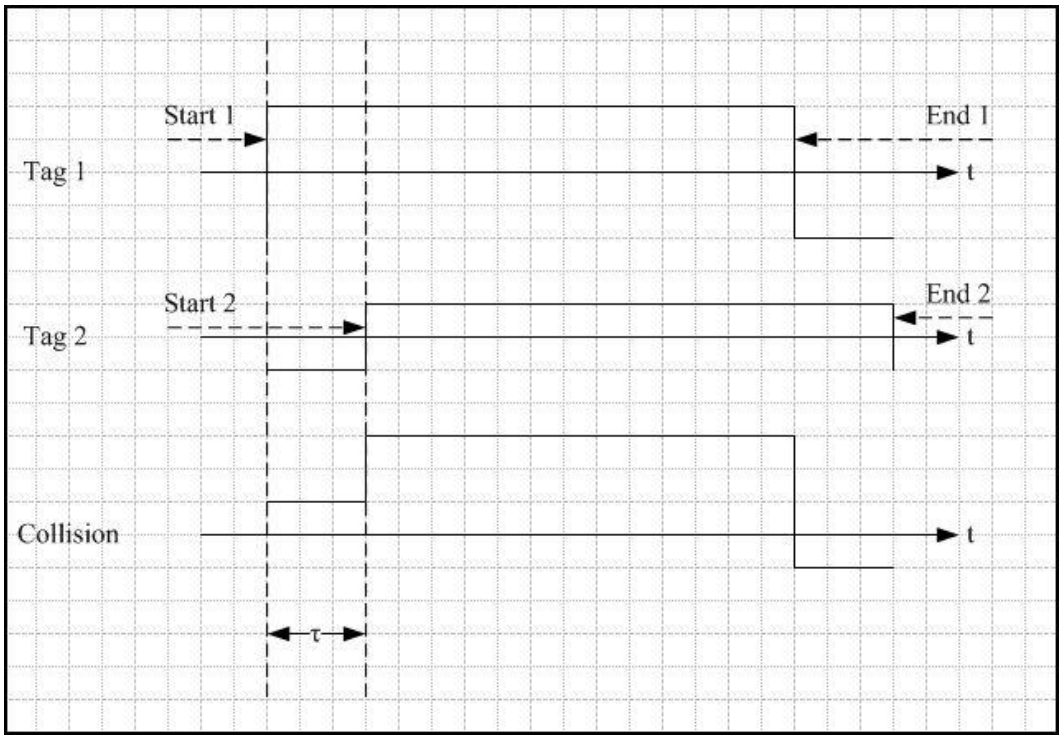

Figure 33. Superposition of Two Formation-1 Symbols

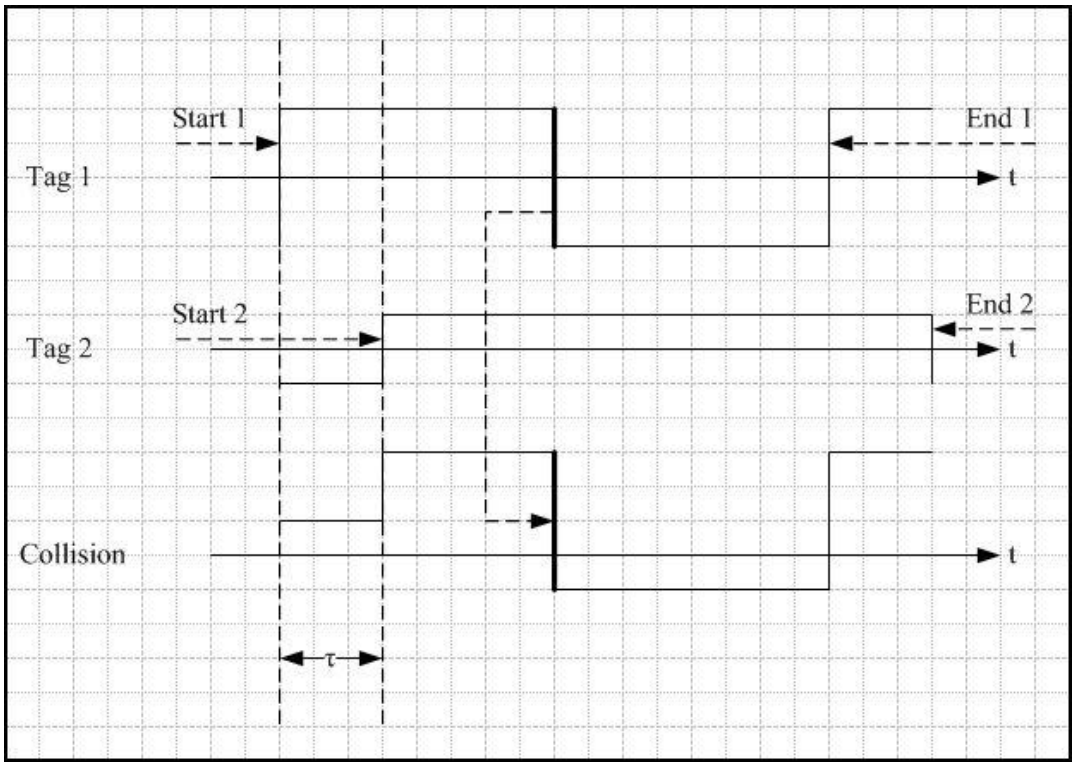

Figure 34. Superposition of Formation-0 Symbol and Formation-1 Symbol

4.1 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. Figure 35 shows the tag response frame architecture after a Query command.

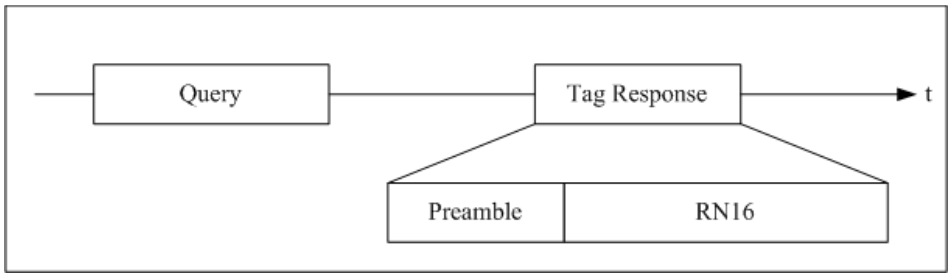

Figure 35. Tag Response Frame Architecture

Figure 36 shows the preamble of the tag response when FM0 encoding is employed. In Figure 37 the preamble of the tag response when Miller encoding is employed. The parameter TRext decides whether a 12-zero pilot tone is used in the preamble. While two tags respond to 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 starting edge of each tag's RN16. Because once the formation of the preamble is fixed, the ending 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 Figure 36, suppose no pilot tone is used (TRext=0), there are eight edges in the preamble. The ending of the preamble can thus be located by counting eight 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 $2N^{th}$ edge in the collision signal. Figure 38 shows an example of a collision signal consisting of two tag preambles in FM0 without a pilot tone. Because each tag's preamble has eight edges, the ending of the tag 1 and tag 2 responses are at the 15^{th} edge and 16^{th} edge in the collision signal. The ending location for Miller encoding is similar.

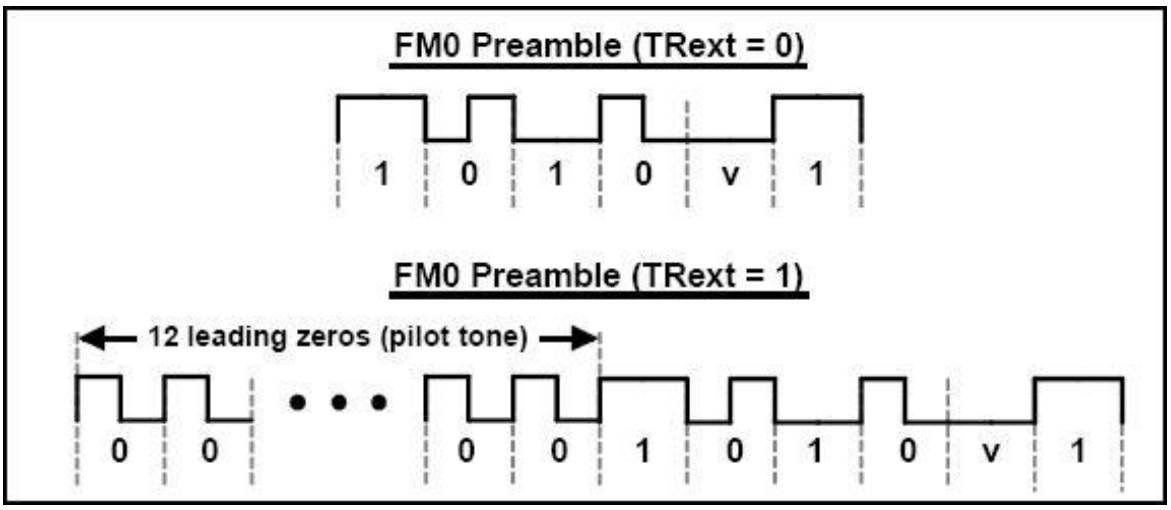

Figure 36. Tag Response Preamble in FM0

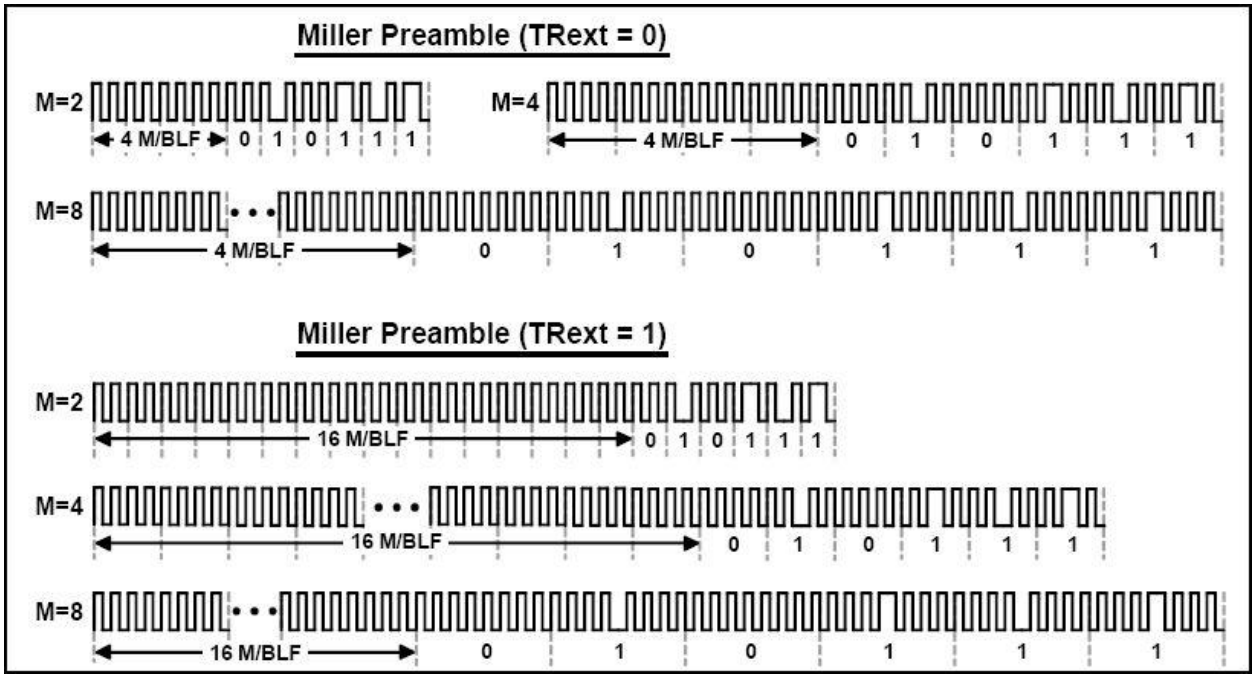

Figure 37. Tag Response Preamble in Miller Subcarrier

Figure 38. Ending Location of Preambles in Collision

4.2 ALGORITHM DESCRIPTION

After locating the end 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 5. This algorithm checks the collision signal one sample point at a time, and thus it is an online algorithm.

Table 5. Direct Edge Locating Algorithm

i.	Locating the preamble ending edge of tag 1 and tag 2 separately in the collision signal, denoted as St1 and St2.
ii.	Start from St1, check the existence of edge transition at time instant St1+$k\times T_s$ (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.
iii.	Start from St2, check the existence of edge transition at time instant St1+$k\times T_s$ (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.

4.3 ALGORITHM 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 Figure 39. 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 5. Because of the tag response BLF deviation, the exact location of the symbol middle edge transition deviates around the theoretic symbol center at $St+k\times T_s$(k= 1, 3, 5, … , 31). Therefore, if using the fixed starting point St and increasing it by the fixed symbol duration to locate the edge transitions, the probability of error increases accordingly as the time reaches later symbols in the collision. To increase the robust property of the algorithm, a calibrator is employed to align the timer and FSM to the exact starting point of the incoming symbol after successfully resolving the previous collision symbol. Figure 40 shows the state transition of the FSM control logic.

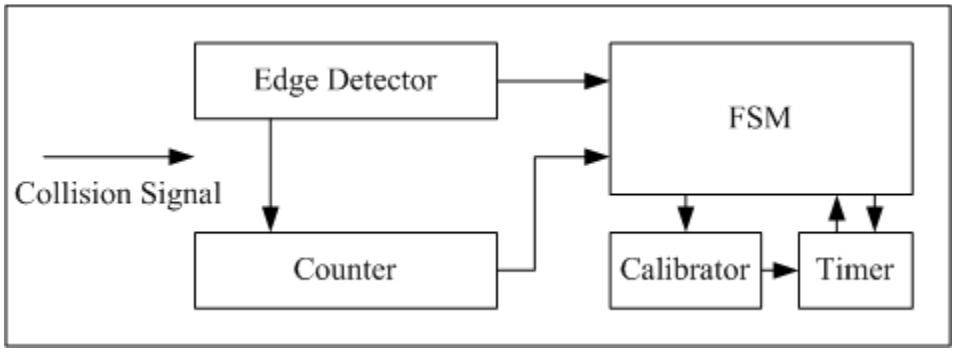

Figure 39. Implementation of the Algorithm using FSM

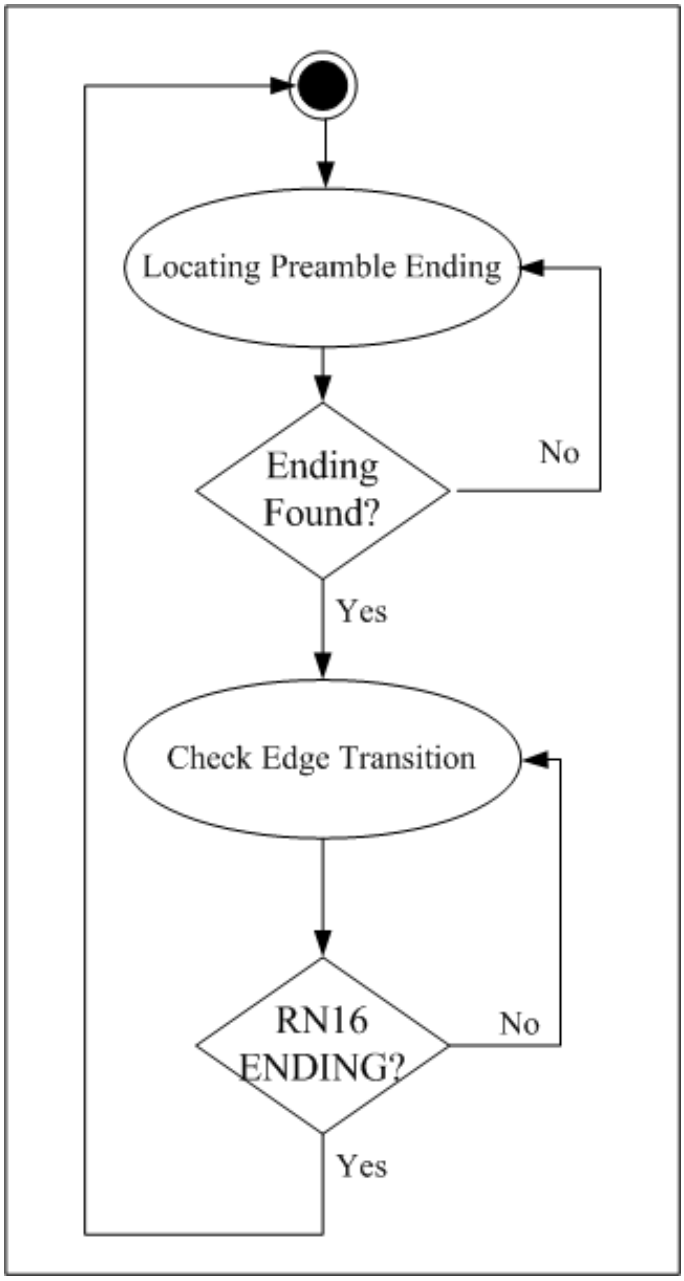

Figure 40. State Transition of the FSM

The edge detector is implemented in two ways. The first method is based on calculating the running mean of the input signal. Figure 41 shows the work flow of the edge detection using the running mean of the signal samples. 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.

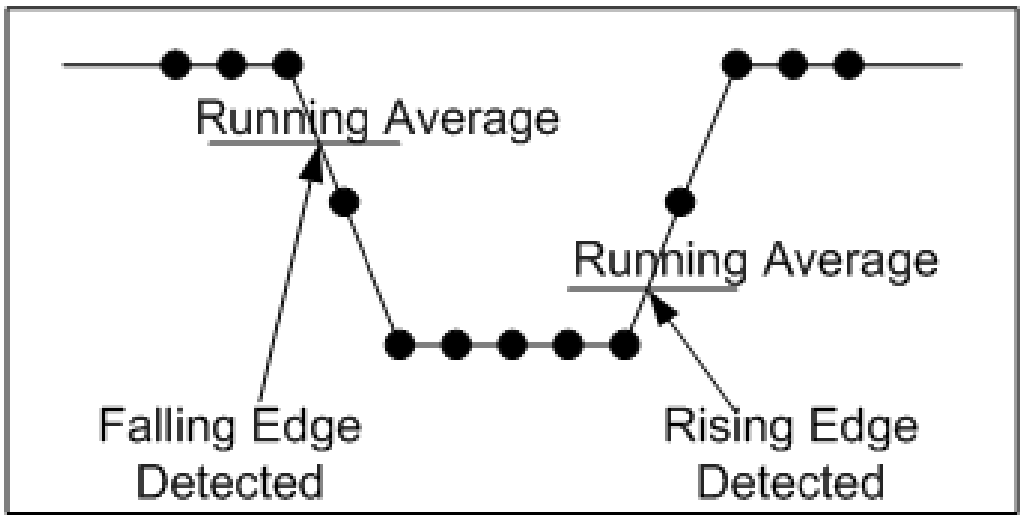

Figure 41. The Running Mean Calculation

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.4.1. In simulation, the edge transition time of the tag response normally takes no more than two sampling periods (0.08μs), the d_t is set as 1. Therefore, the differentiation actually equals the difference of the two adjacent sampling points in the signal. Figure 42 shows the preamble part of an FM0 collision and its corresponding differentiation. However, in reality it is possible that the edge transitions in the tag response can take more than one sampling periods. In this case, Eq.4.1 outputs a pulse rather than an impulse if d_t is still set to 1. To guarantee the algorithm's compatibility with the less steep edge transition scenario without dynamically changing the unpredictable d_t, the algorithm detects only the rising edge of the differentiation output as shown in the LabVIEW code segment in Figure 43.

$$\text{Differentiation} = \frac{y(t) - y(t - dt)}{dt} \qquad \text{(Eq. 4.1)}$$

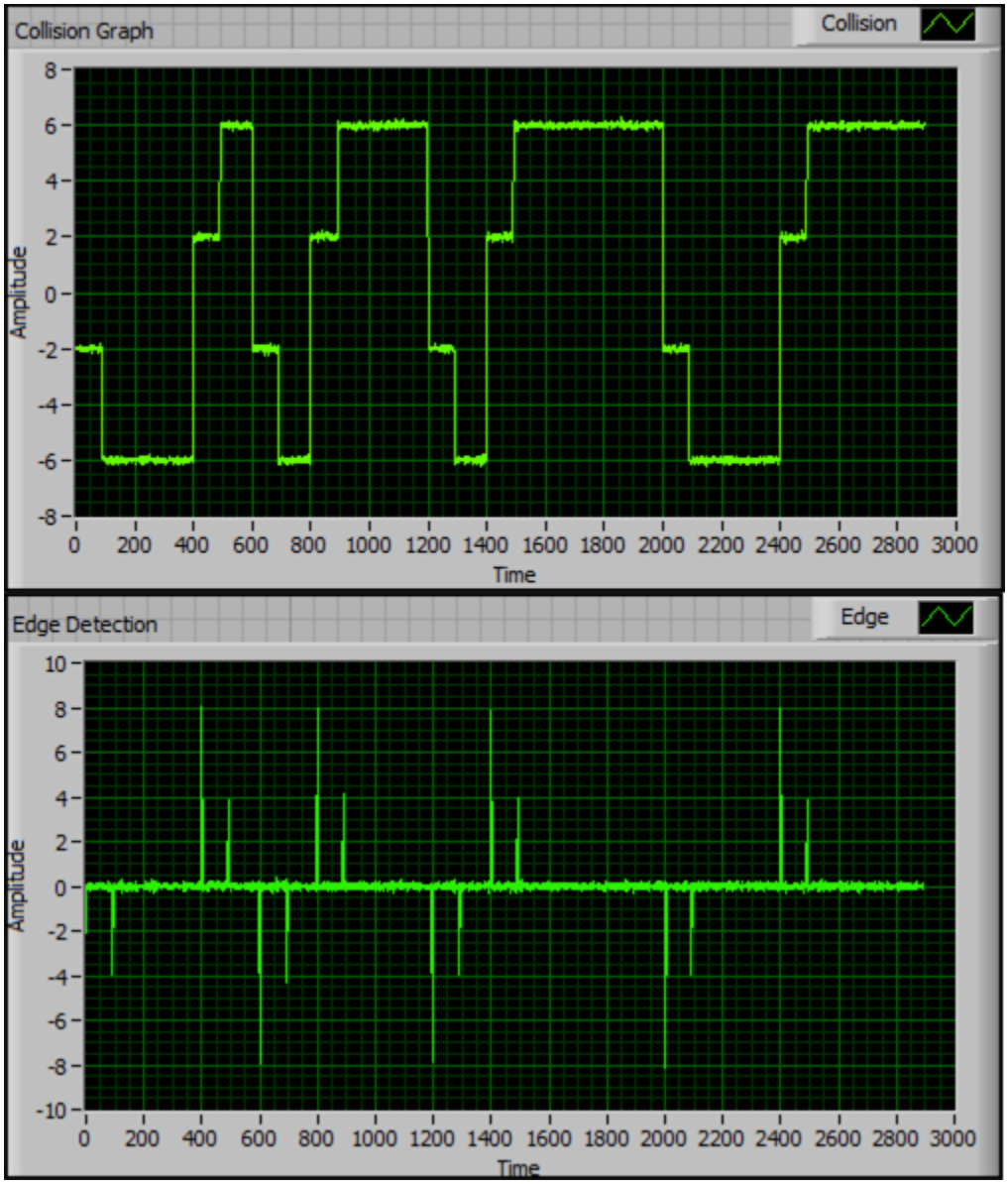

Figure 42. The Differentiation of Collision signal

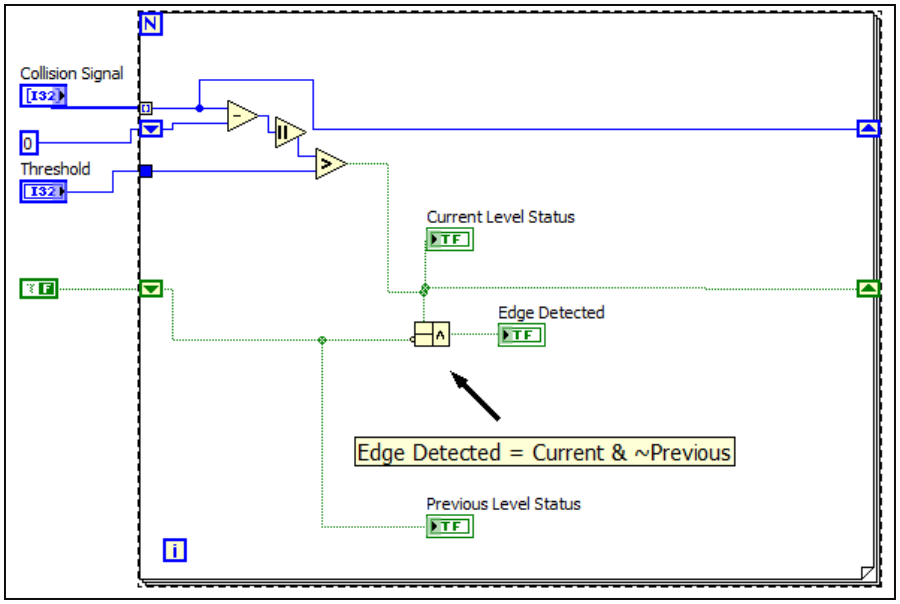

Figure 43. Collision Signal Edge Detection (LabVIEW Code Segment)

Because the variation of symbol duration exists in the practical tag response, the middle symbol edge transition can appear in a range around the supposed $St+k\times T_s$ (k= 1, 3, 5, … , 31) positions rather than exactly at the middle of each symbol. Therefore, the FSM searches 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. Figure 44 shows the searching range.

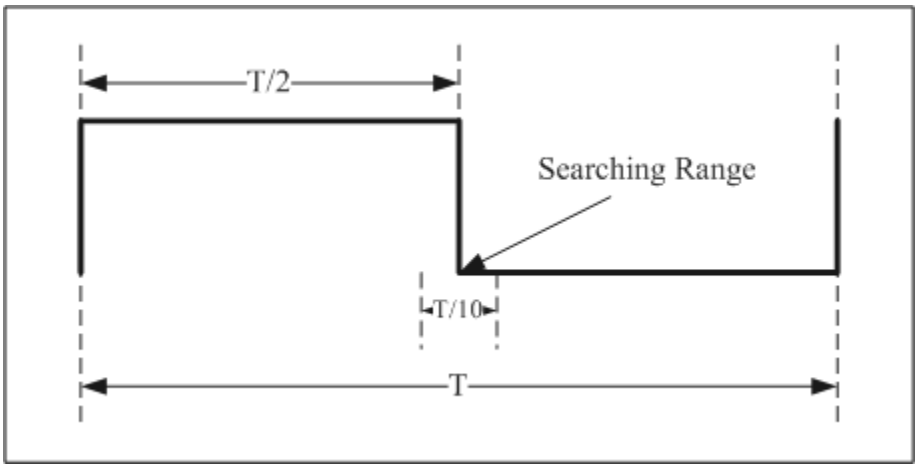

Figure 44. The Differentiation of Collision signal

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 inputted 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. Figure 45 shows the two tag responses with noise. Figure 46 shows the collision signal and its corresponding differentiation showing the edges. The LEDs in Figure 46 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 are exactly the inputted RN16s.

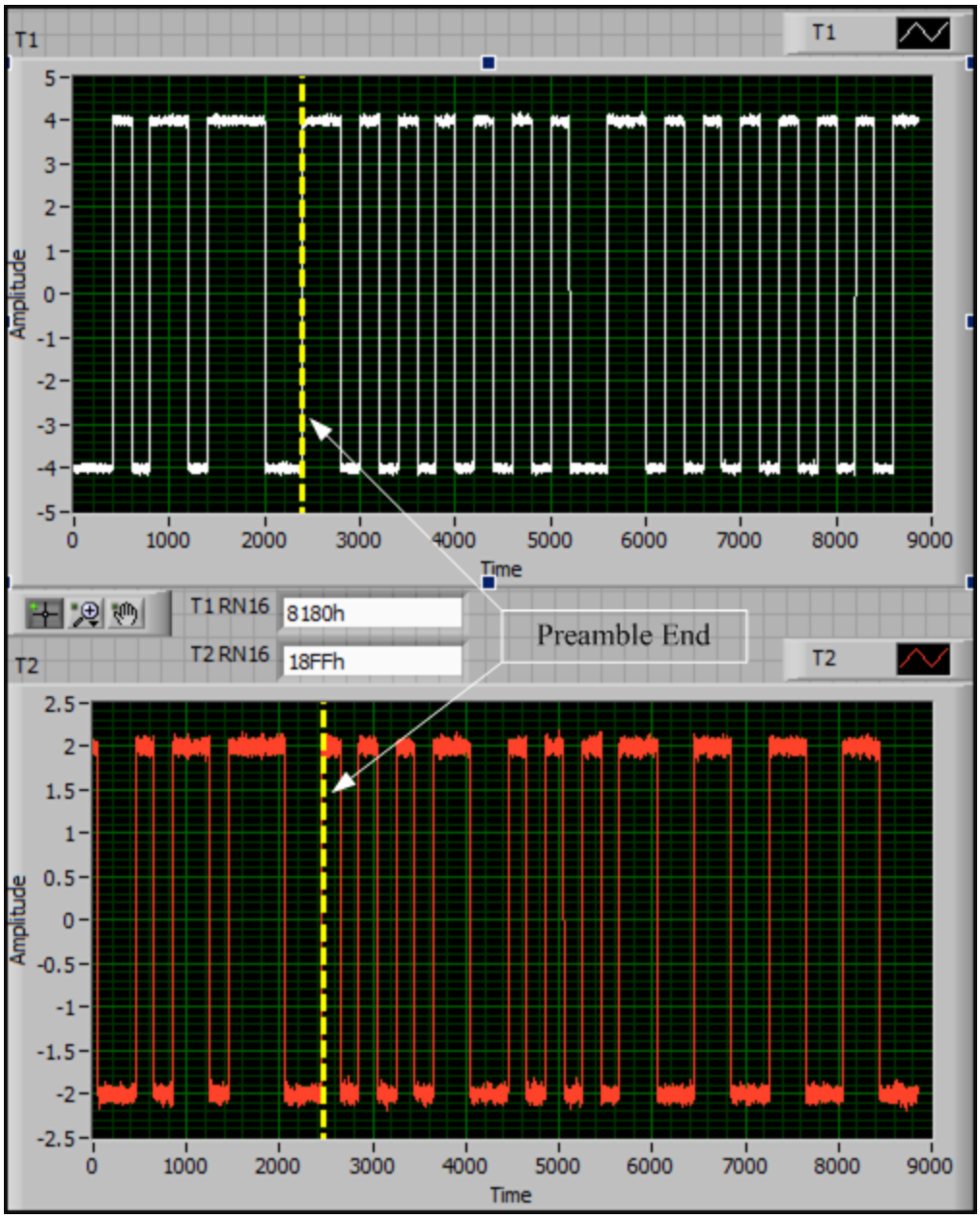

Figure 45. Two Tag Responses

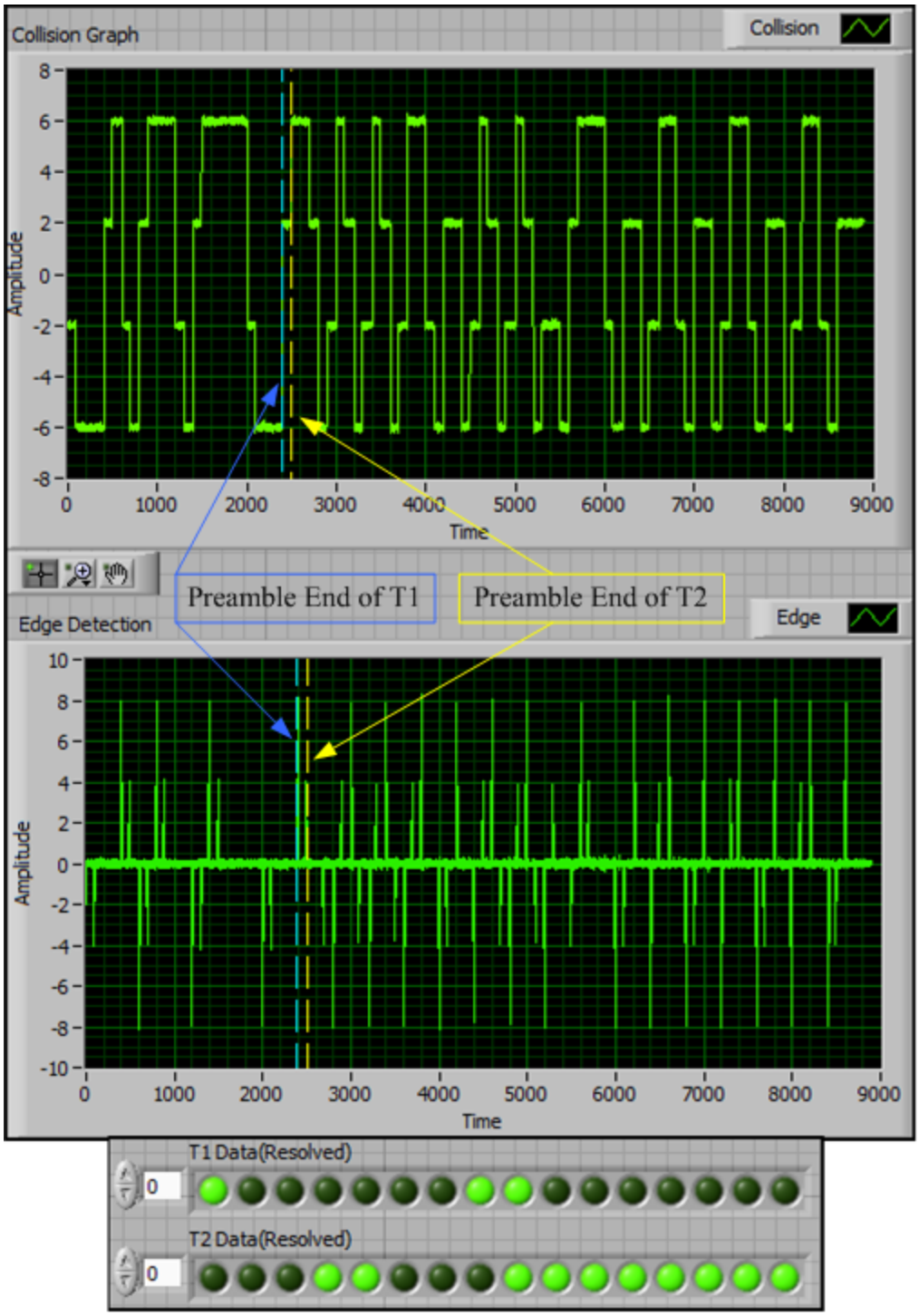

Figure 46. Collision Signal and its Differentiation

There are two limitations of the algorithm:

1. As the BLF of tag increases, the phase delay decreases as shown in Section 3, which requires more accuracy of the hardware timer to locate the searching range for each symbol. It is also possible that the searching range of the current symbol for each tag may overlap (as shown in Figure 47), which requires the shrinking of the searching range. This limitation can be removed by using the amplitude mapping which is designed for tags with minimal phase shift as will be discussed in Section 5.0 .

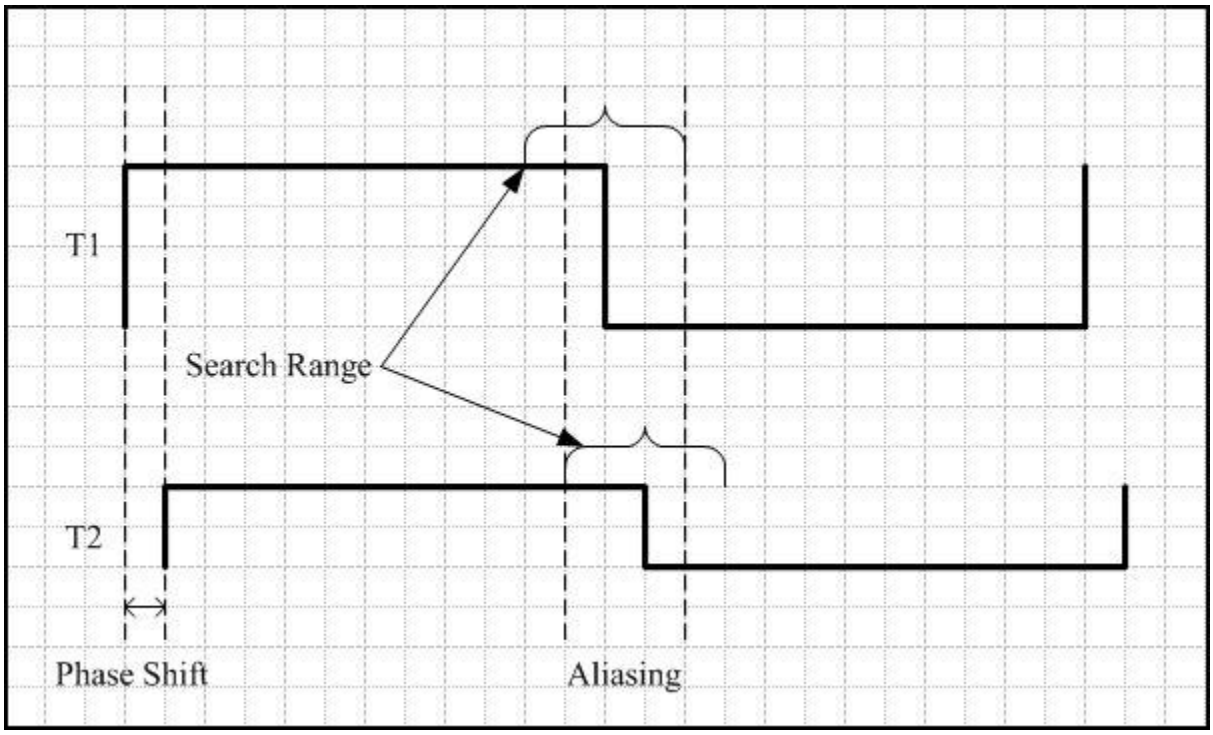

Figure 47. Searching Range Overlap

2. The phase shift cannot exceed 50% of the symbol duration, otherwise the symbol beginning of the leading tag will interfere with the middle of the lagging tag. This is illustrated in Figure 48. In Figure 48, the phase shift is exactly 50%, and the formation 1 symbol in Tag 1 is arbitrated as formation 0 due to the edge transition appeared in the

search range in the collision, which is actually is beginning edge of Tag 2's RN16. This limitation can be overcome by adjusting software parameters for the case in which the phase shift becomes larger than 50% of the symbol duration provided that it is not exactly 50%. However, based on the measurement of the practical phase shift at typical tag BLFs as shown in Figure 28 and Figure 29 of Section 3.0 , the phase shift cannot exceed 25% of the symbol duration.

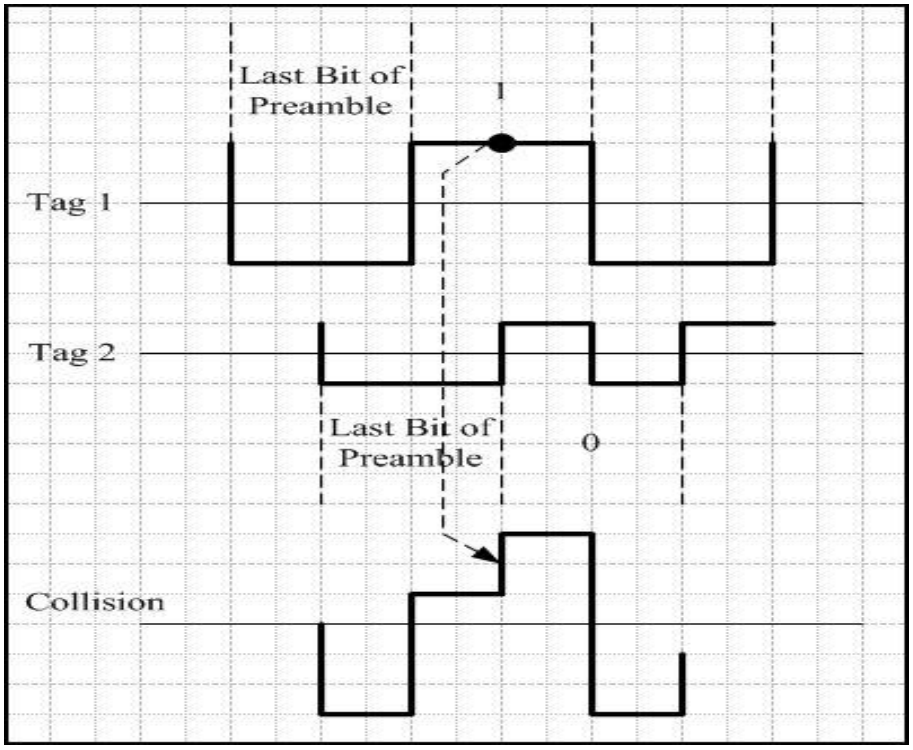

Figure 48. Miss Arbitration

4.4 ALGORITHM IMPLEMENTATION

When simulated in LabVIEW on the host PC, the direct edge locating algorithm is developed as an online function for convenient FPGA transplant: The program reads in the collision signal and

performs point-by-point processing on it. LabVIEW provides its FPGA compilation environment in its 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 computation involved in the algorithm is Boolean for the arbitration logic and countering for timers, the NI5640R FPGA baseband used in the data acquisition platform featuring a Xilinx Virtex-II Pro XC2VP30 FPGA is capable. The implementation verification flow is shown in Figure 49. 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. 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 the 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.

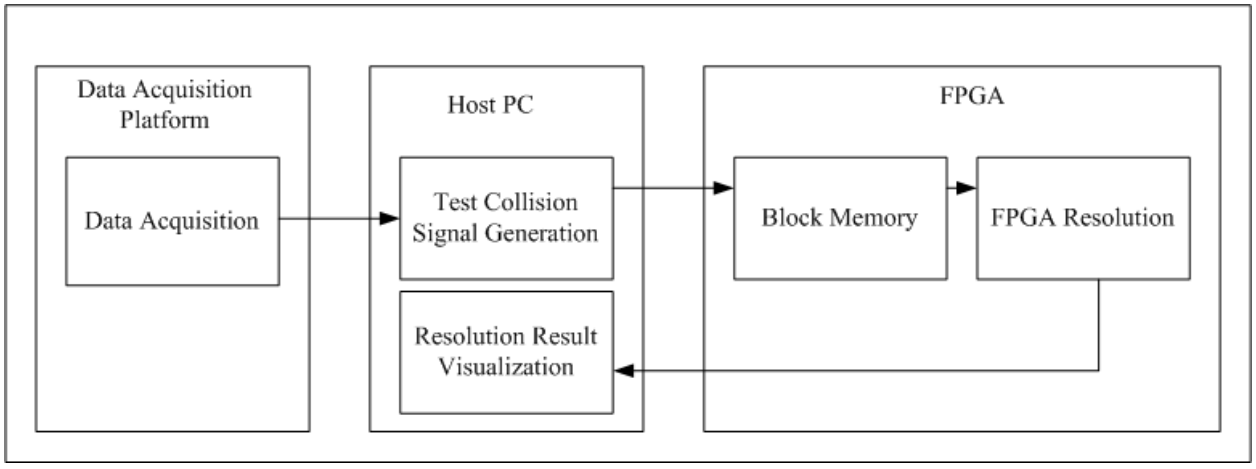

Figure 49. Verification Flow of Algorithm Implementation

Figure 50,Figure 51,Figure 52,Figure 53,Figure 54 show the resolution results on actual tag collision signal at each typical tag BLF.

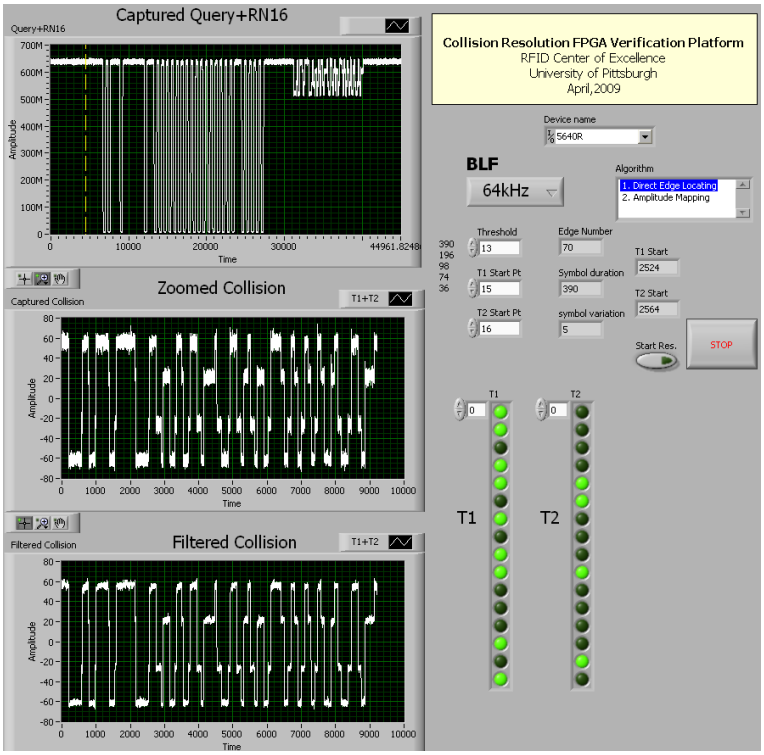

Figure 50. Resolution Result When BLF=64 kHz

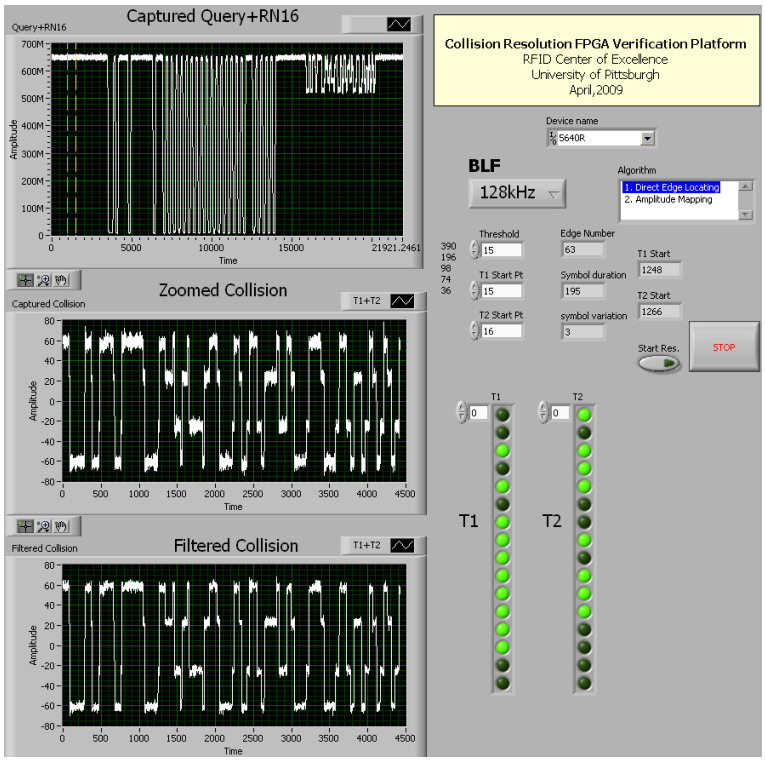

Figure 51. Resolution Result When BLF=128 kHz

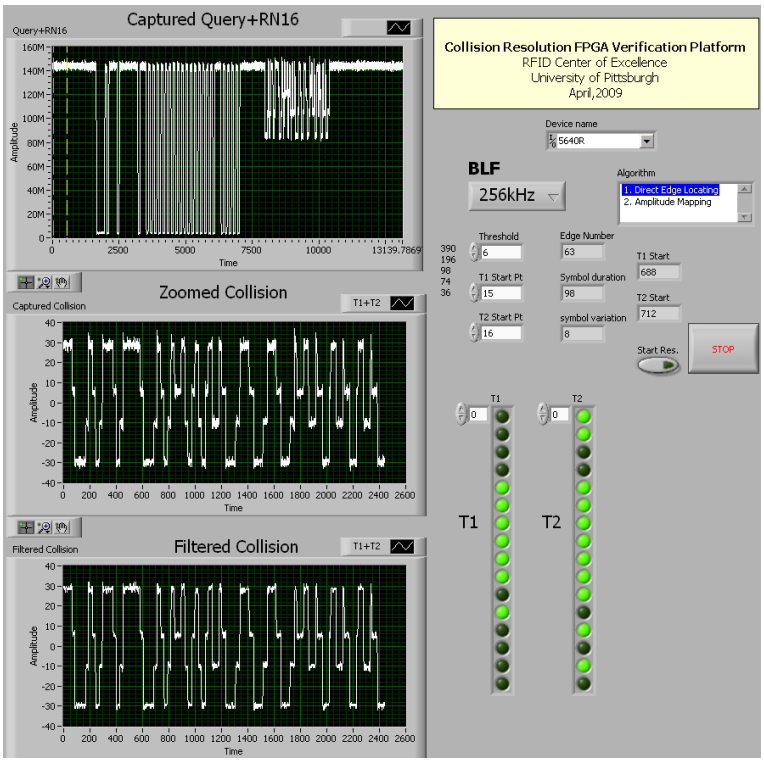

Figure 52. Resolution Result When BLF=256 kHz

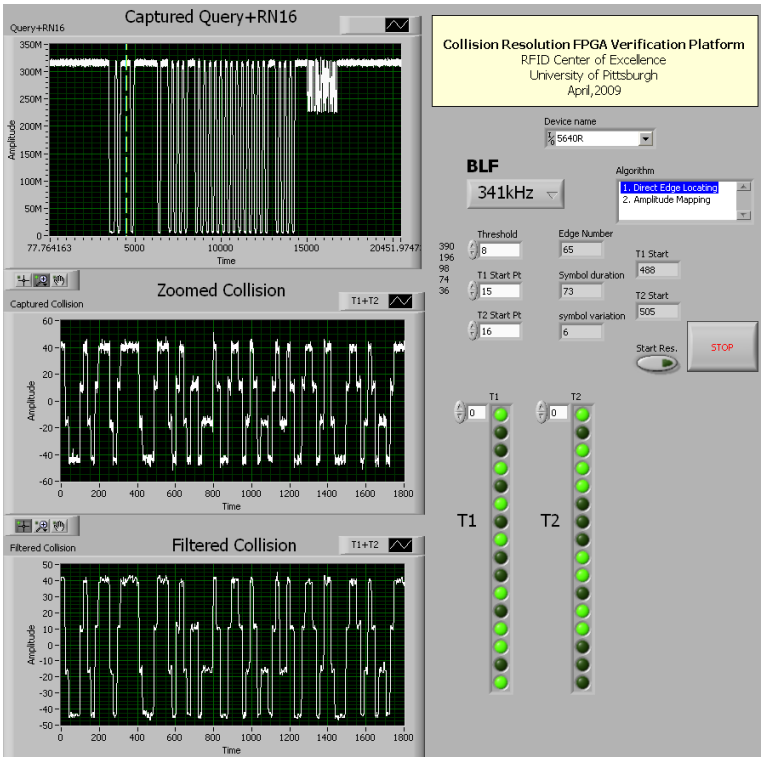

Figure 53. Resolution Result When BLF=341 kHz

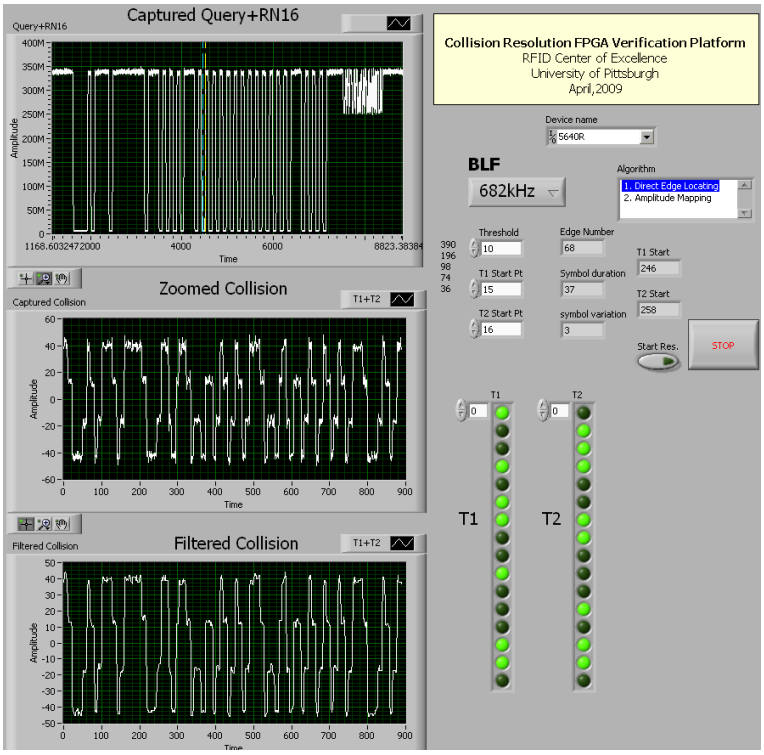

Figure 54. Resolution Result When BLF=682 kHz

The FPGA resource utilization reported by Xilinx XST FPGA compiler is listed in Table 6. As shown, the design only utilizes 20% of the total FPGA logic slices.

Table 6. FPGA Resources Utilization

```
Compilation Summary
-------------------
Device Utilization Summary:
  Number of BUFGMUXs                    7 out of 16     43%
  Number of External IOBs             403 out of 556    72%
  Number of LOCed IOBs                403 out of 403   100%
  Number of MULT18X18s                  2 out of 136     1%
  Number of RAMB16s                    45 out of 136    33%
  Number of SLICEs                   2849 out of 13696  20%

Clock Rates: (Requested rates are adjusted for jitter and accuracy)
  Base clock: Configuration_Clk
     Requested Rate:      20.000000MHz
     Theoretical Maximum: 69.754464MHz
  Base clock: RTSI_Ref_Clk
     Not used
  Base clock: ADC_0_Port_A_Clk
     Requested Rate:      25.001250MHz
     Theoretical Maximum: 34.907669MHz
  Base clock: DAC_0_IQ_Clk
     Not used
```

Because the algorithm is in online real time, the resolution completes half a symbol duration before the end of the collision. The processing time corresponding to each BLF is compared to the standard specified turn-around time as shown in Figure 55. The required time in the figure is calculated as the sum of the collision signal length and the standard specified turn-around time.

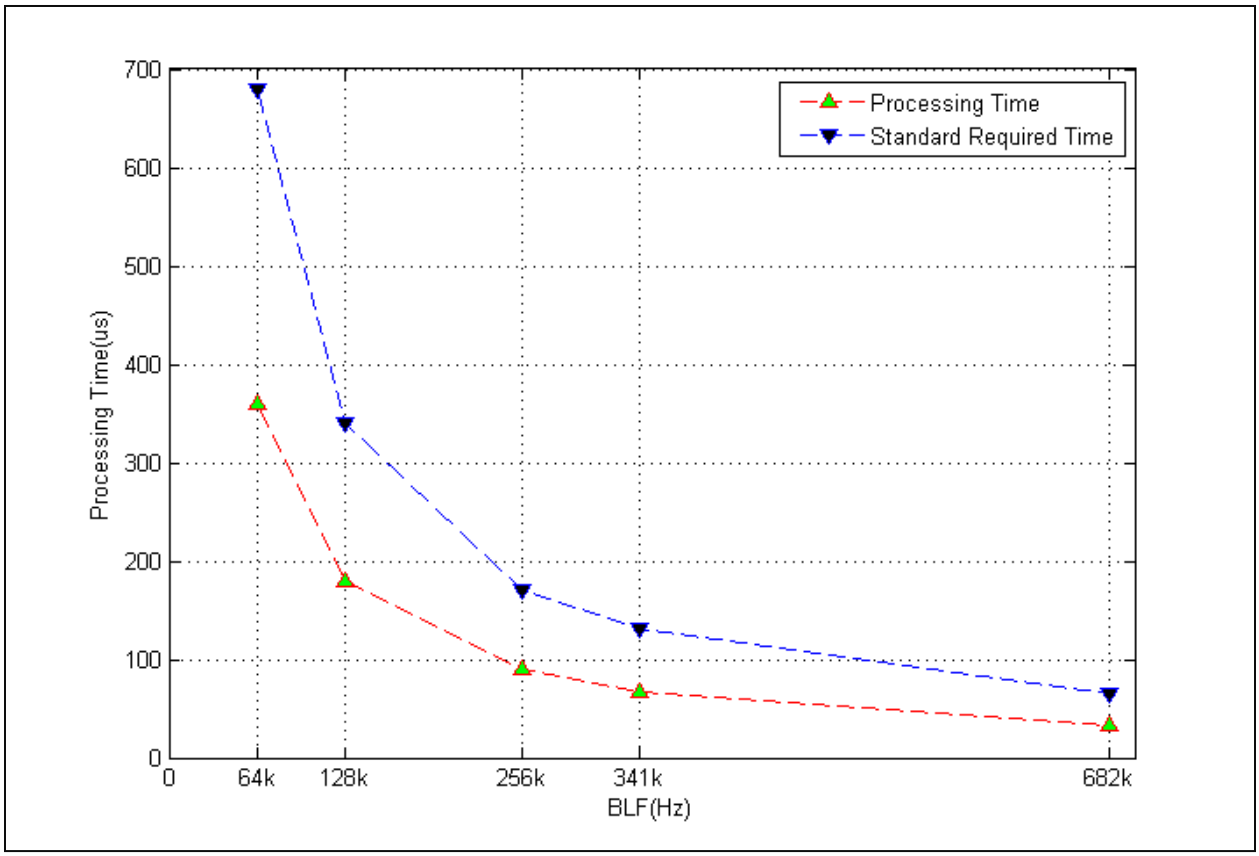

Figure 55. Processing Time vs. Standard Required Time

5.0 RESOLUTION METHOD II：AMPLITUDE MAPPING

As discussed in Section 3.0 , 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 in Section 4.0 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 Figure 56 makes the direct edge locating method fail entirely when at least one tag has a formation 0 symbol appear. In Figure 56, 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 decided. 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 Figure 56, 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.

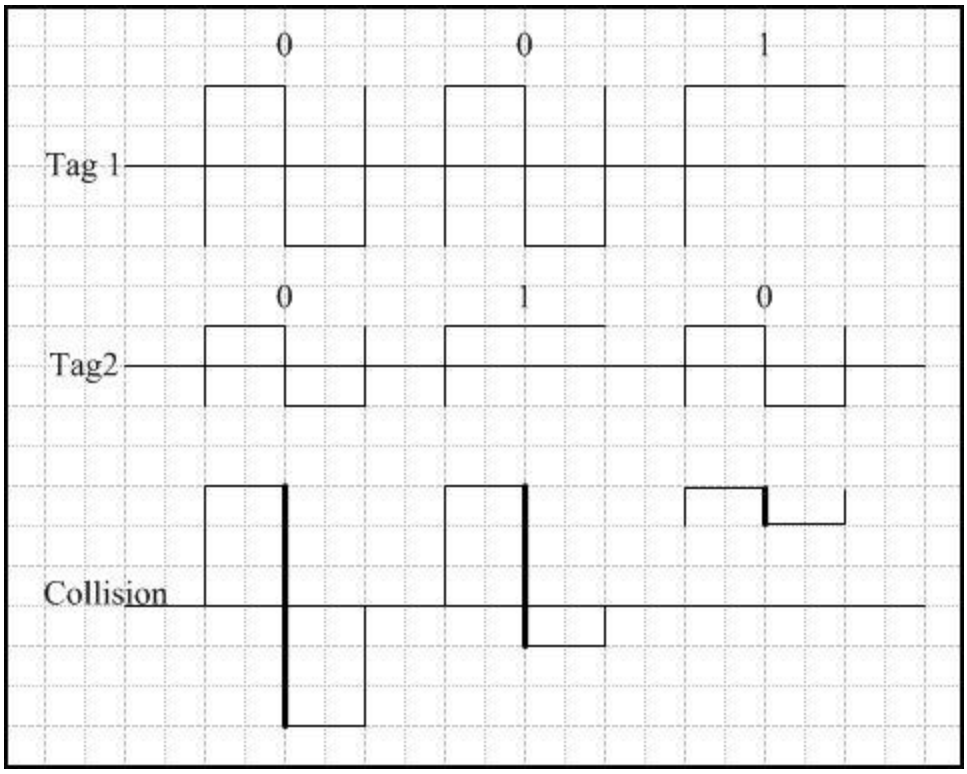

Figure 56. Edge Ambiguity

Because there are two possible formation 0 symbols and two possible formation 1 symbols, each symbol in the tag responses can have four possible shapes as shown in Figure 7 in Section 1.1. Therefore, when the two tag responses are superimposed, there can be 16 possible shapes of the collision symbol as shown in Figure 57. 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 Figure 57, the amplitude of Tag 1's response is supposed to be twice 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 information, while the bottom two portions (area 3 and area 4) correspond to the cases when both of the tag responses are different in formation.

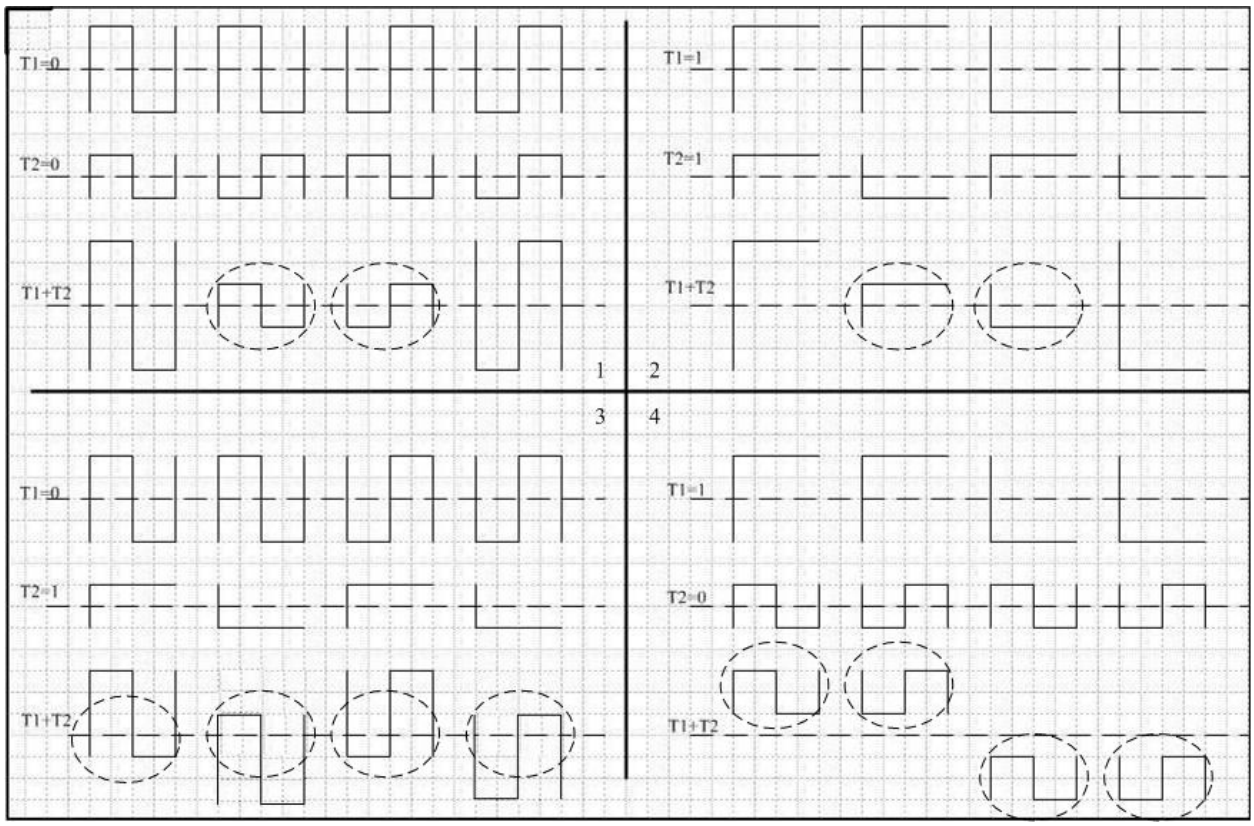

Figure 57. Collision Patterns

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

1. 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 level distribute at different sides of the average line.
2. 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 level distribute at the same side of the average line.

3. 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 level distribute at different sides of the average line.
4. 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 level distribute at the same side of the average line.

As discussed early, 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 appears 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 individual tag response with opposite polarization are superimposed as indicated by the circled cases in Figure 57. Figure 58 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.

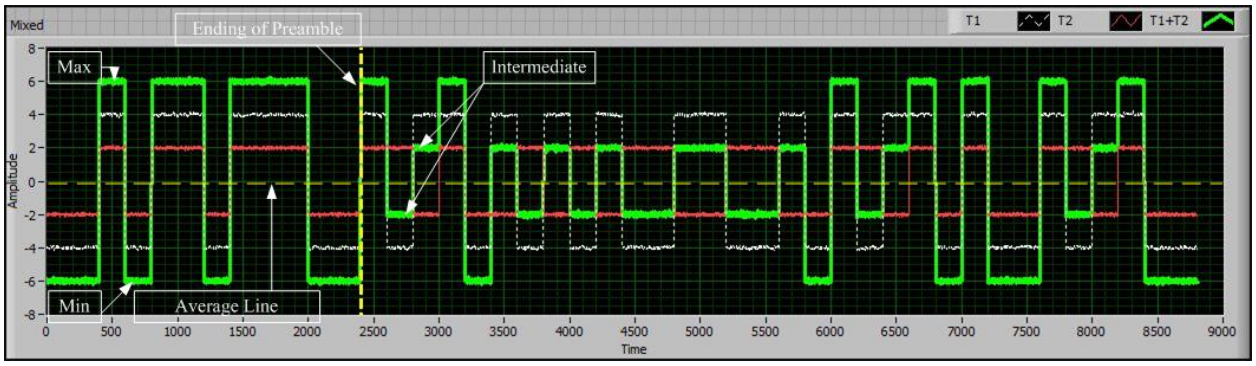

Figure 58. Example Superposition of Tag Responses without Phase shift

5.1 ALGORITHM DESCRIPTION

Based on the characteristics of the collision symbol, the individual tag symbol can resolved by scanning through the collision and mapping the collision signal to the cases as listed in Figure 57. The algorithm is therefore named Amplitude Mapping and listed in Table 7. The mapping process is described in the pseudo code in Table 8.

Table 7. Amplitude Mapping Algorithm

1. Scan the duration of the preamble, 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 collision symbol to the cases as shown in Figure 43 to decide the individual tag data.

Table 8. Pseudo code for Mapping Process

```
collision_avg= (collision_Max+collision_Min)/2;
symbol_avg= (fst_value+scnd_value)/2;

if (!edge_transition) //case 2
   T1=1;
   T2=1;
 else  //case 1,3,4
    if (symbol_avg==collision_avg) //case 1
      T1=0;
      T2=0;
    else //case 3,4
       if ((fst_value>collision_avg) XOR (fst_value>collision_avg)) //case 3
          T1=0;
          T2=1;
         else
          T1=1;
          T2=0;
         end
      end
 end
```

5.2 ALGORITHM SIMULATION

The amplitude mapping algorithm is simulated using LabVIEW on the Host PC. It is implemented by using a FSM as shown in Figure 59. 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 Figure 60. 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 to the four cases according to the result of the comparator. Figure 61 shows the state transition of the FSM control logic.

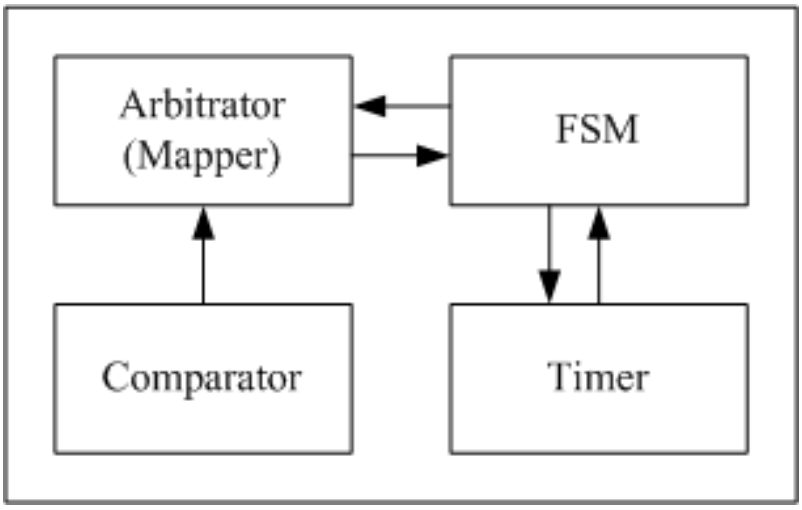

Figure 59. Implementation of the Algorithm using FSM

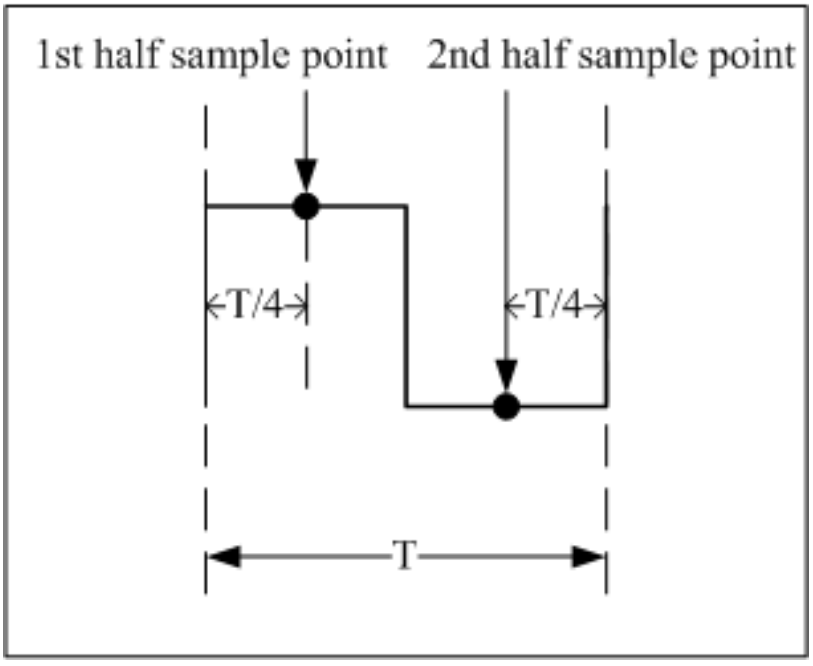

Figure 60. Symbol Sampling Point

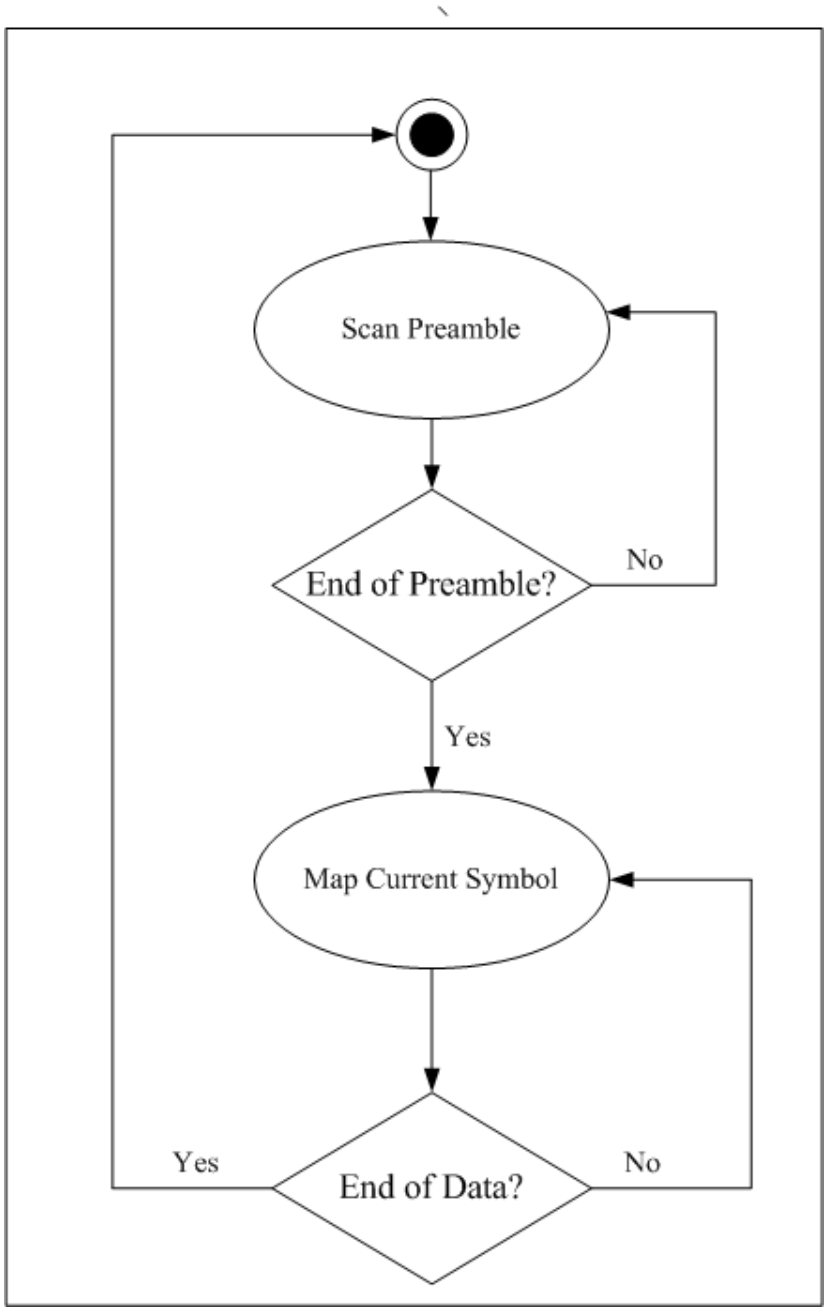

Figure 61. State Transition of FSM

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. Figure 62 shows the two tag responses with noise. Figure 63 shows the collision signal and recovered data. The LEDs in Figure 63 show the recovered data from the collision, and the LED turned on corresponds to a 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 inputted RN16s.

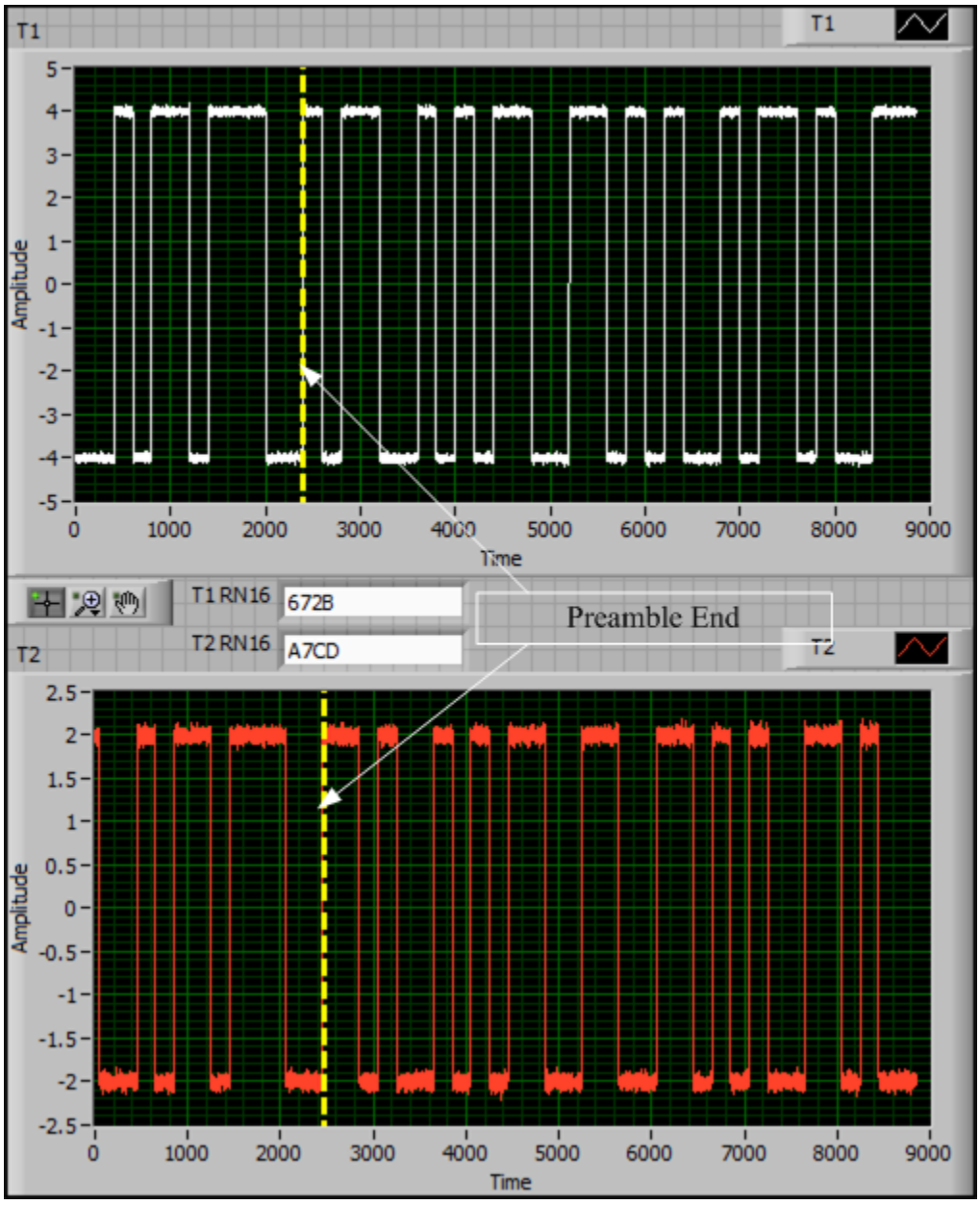

Figure 62. Two Tag Responses

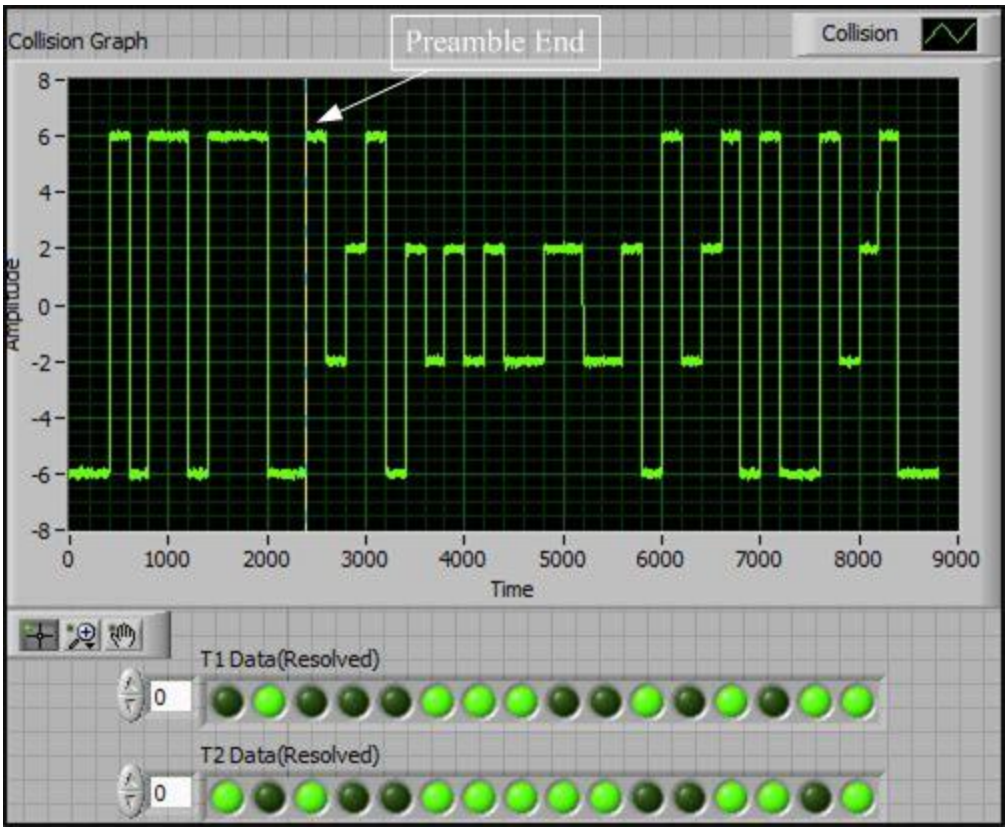

Figure 63. Collision and Resolved data (Simulation)

5.3 ALGORITHM IMPLEMENTATION

The amplitude mapping algorithm is implemented using the same FPGA device as the direct edge locating method discussed in Section 4. The implementation and verification scheme is similar to the architecture as illustrated in Figure 49. Figure 64,Figure 65,Figure 66, Figure 67,and Figure 68 show the resolution results on the actual tag collision signal at typical BLFs.

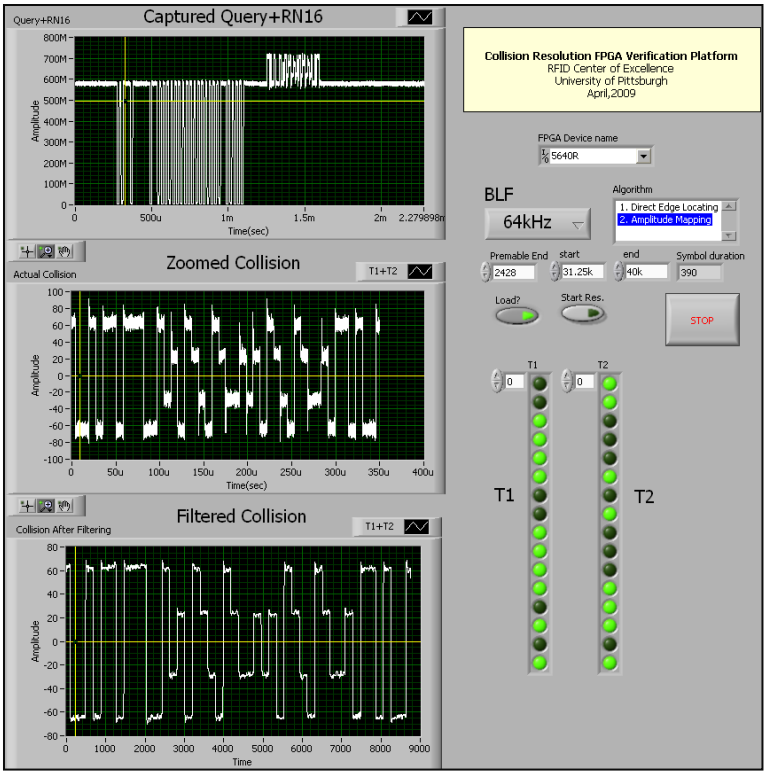

Figure 64. Collision and Resolved data (BLF=64 kHz)

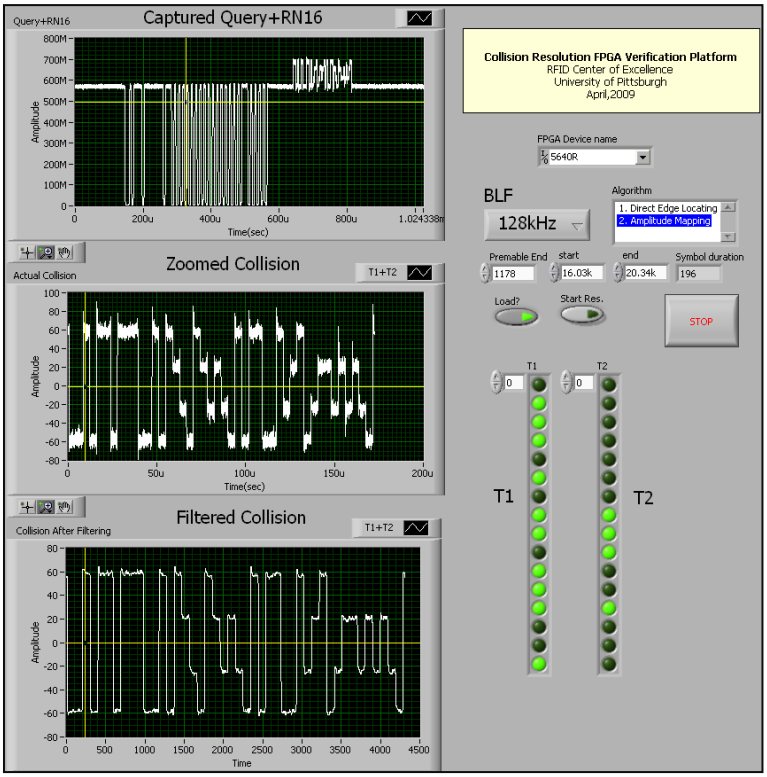

Figure 65. Collision and Resolved data (BLF=128 kHz)

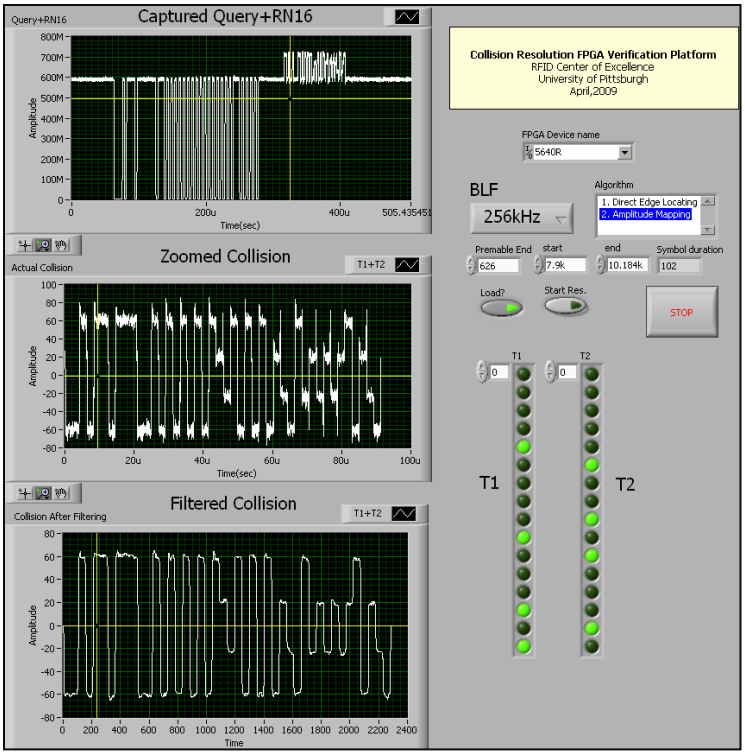

Figure 66. Collision and Resolved data (BLF=256 kHz)

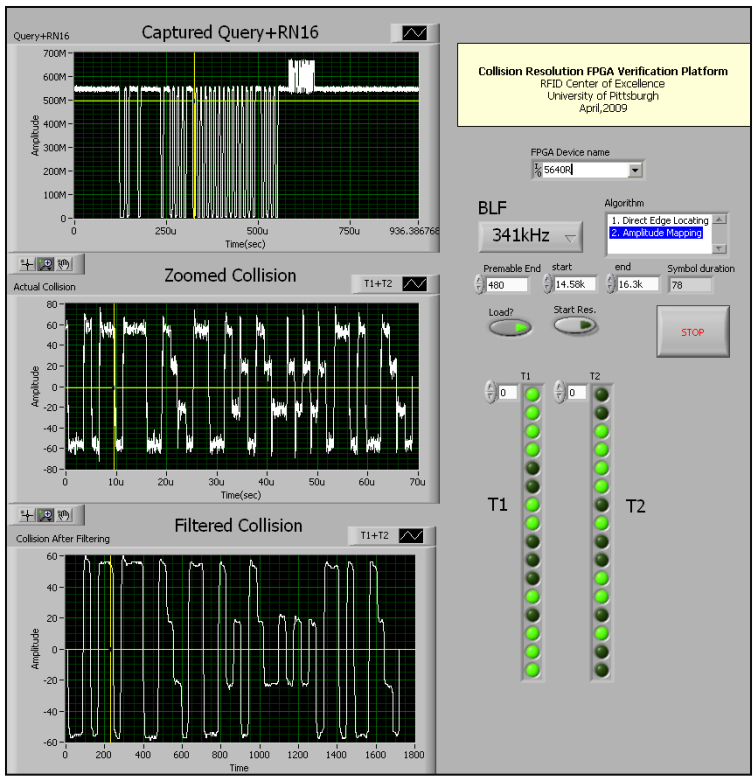

Figure 67. Collision and Resolved data (BLF=341 kHz)

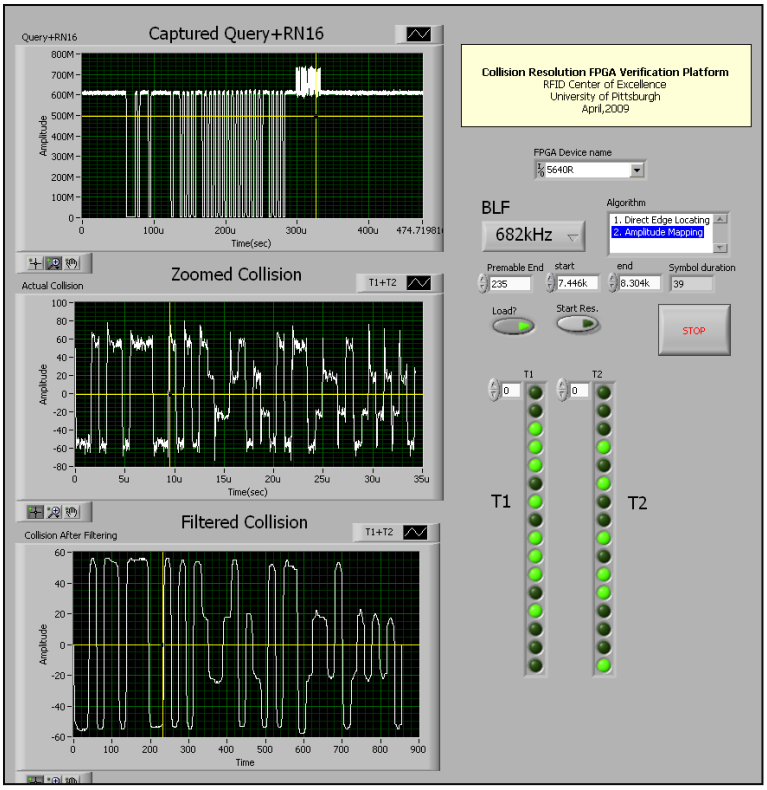

Figure 68. Collision and Resolved data (BLF=682 kHz)

The FPGA resources utilization reported by Xilinx XST FPGA compiler is listed in Table 9. As shown, the design only utilizes 24% of the total FPGA logic slices.

Table 9. FPGA Resources Utilization

```
Compilation Summary
--------------------
Device Utilization Summary:
  Number of BUFGMUXs                    7 out of 16     43%
  Number of External IOBs             403 out of 556    72%
  Number of LOCed IOBs                403 out of 403   100%
  Number of MULT18X18s                  2 out of 136     1%
  Number of RAMB16s                    45 out of 136    33%
  Number of SLICEs                   3329 out of 13696  24%

Clock Rates: (Requested rates are adjusted for jitter and accuracy)
 Base clock: Configuration_Clk
    Requested Rate:      20.000000MHz
    Theoretical Maximum: 68.780521MHz
 Base clock: RTSI_Ref_Clk
    Not used
 Base clock: ADC_0_Port_A_Clk
    Requested Rate:      25.001250MHz
    Theoretical Maximum: 26.281209MHz
 Base clock: DAC_0_IQ_Clk
    Not used
```

Because the algorithm is in online real time, the resolution completes half a symbol duration before the end of the collision. The processing time corresponding to each BLF is compared to the standard specified turn-around time as shown in Figure 69. The required time in the figure is calculated as the sum of the collision signal length and the standard specified turn-around time.

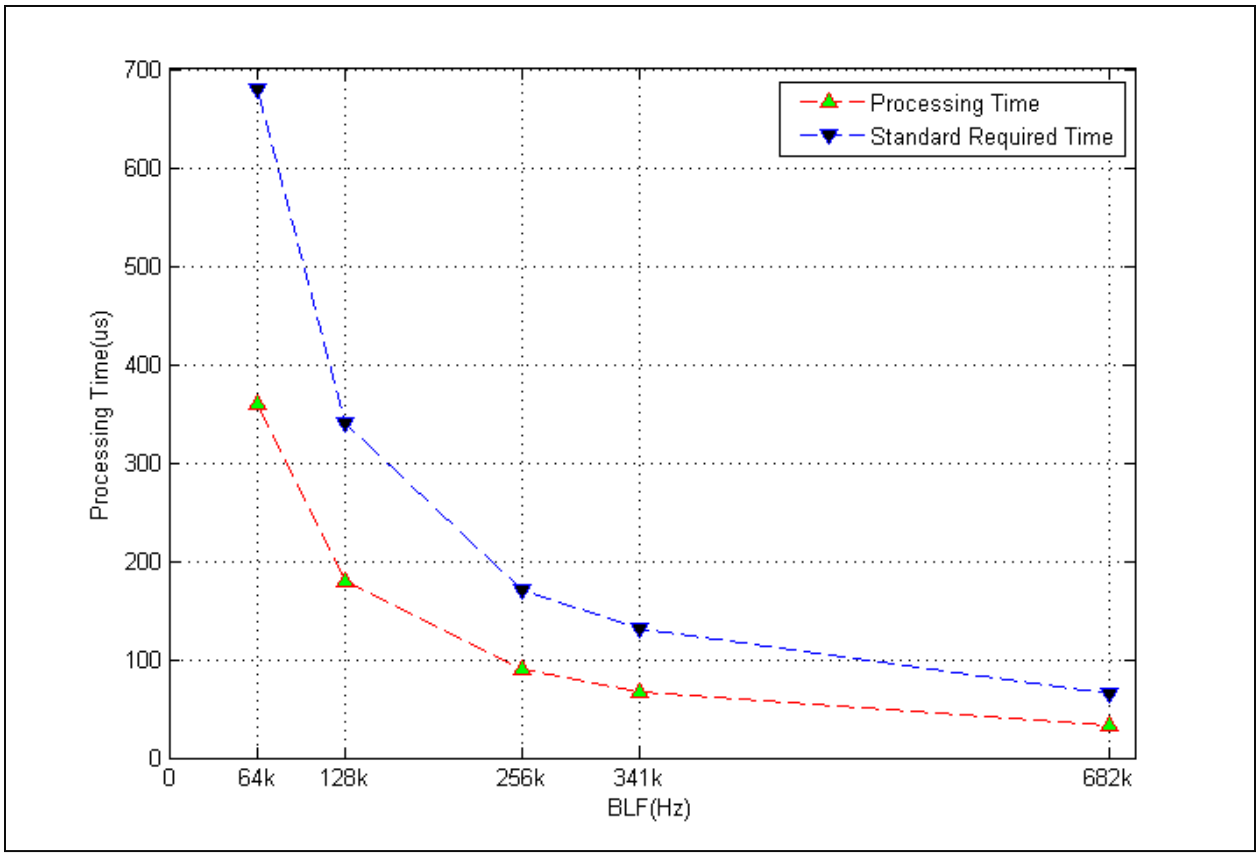

Figure 69. Processing Time vs. Standard Required Time

5.4 SOLUTION UNIFICATION

The solution to resolving the collision with phase shift and without phase shift can be unified by passing the collision signal with phase shift into a median filter. The median filter takes in N data points each time, sorts the data value 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 shall be equal to or greater than the phase shift in order to filter out the phase shift caused impulse. In Figure 70, a collision with phase shift of 5% of 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.

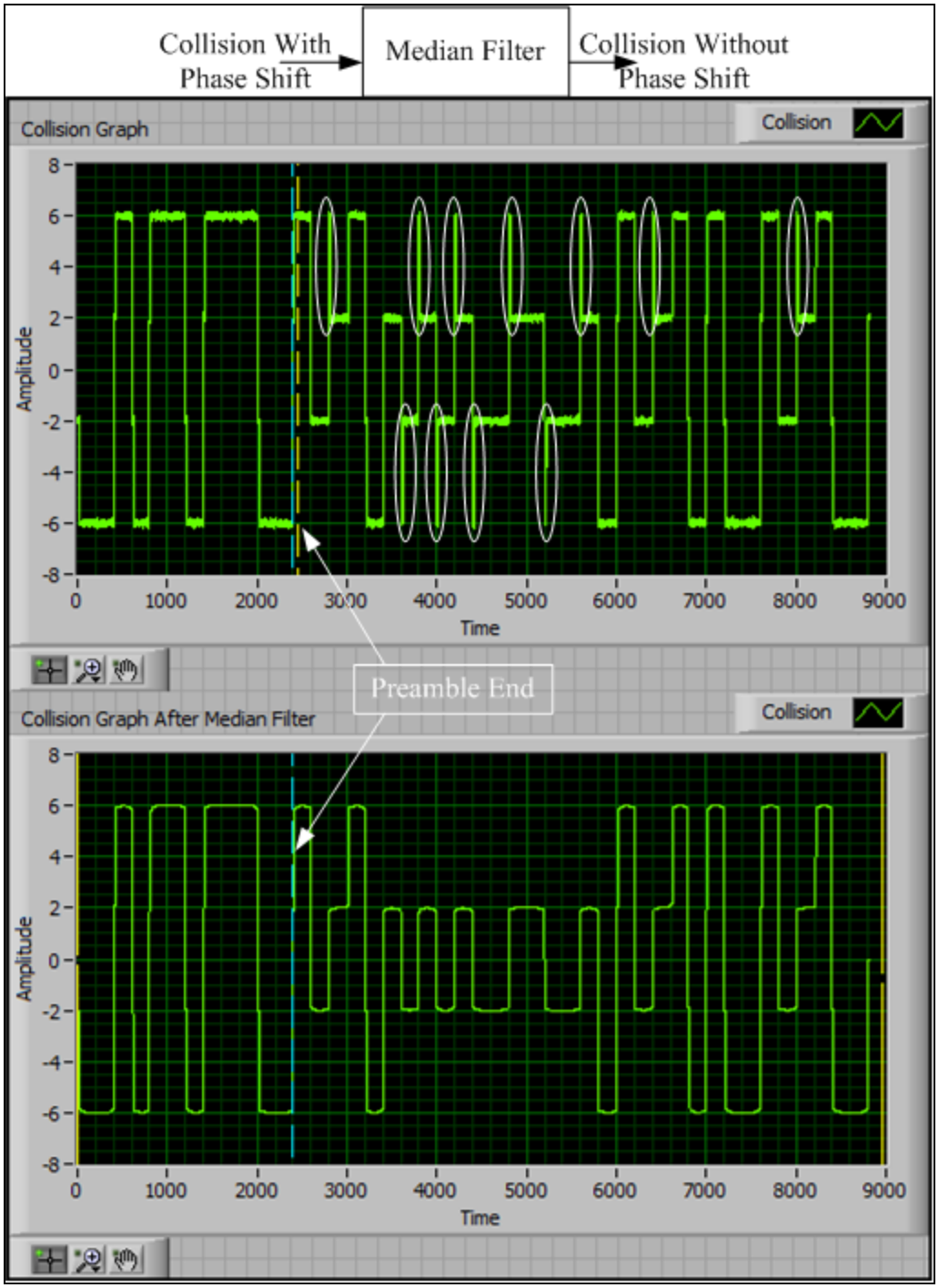

Figure 70. The Effect of Median Filter

One side benefit of the median filter is that it filters the noise in the collision signal and improves the 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 to 400 data points in length, and the phase shift is 20 data points (5% of T_s). The collision includes two RN16 (672Bh and A7CDh). Figure 71 shows the result. The resolution results of the two algorithms are the same.

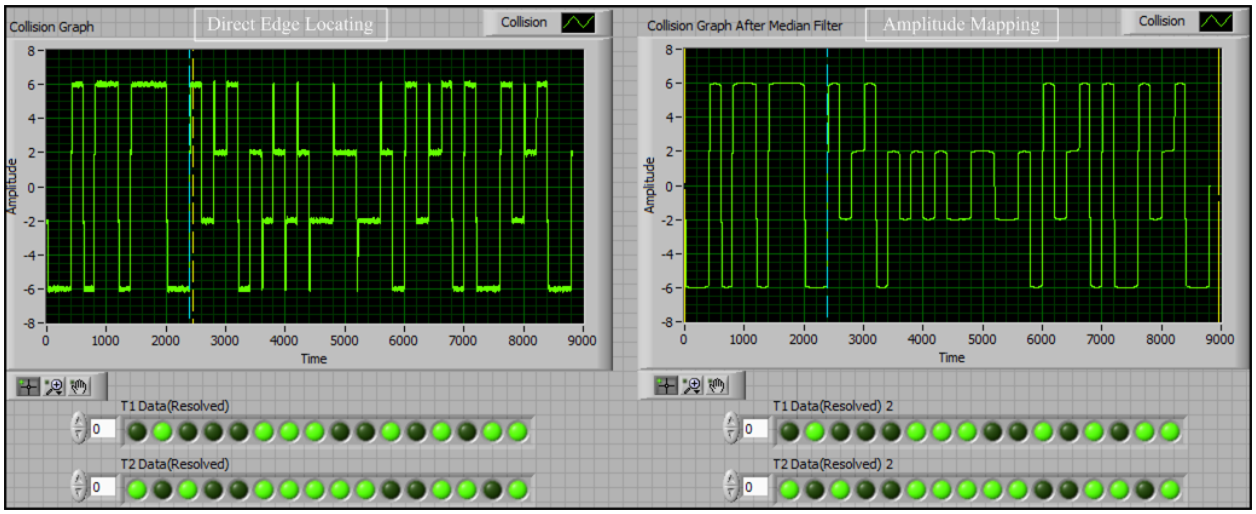

Figure 71. Algorithm Comparison

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. Figure 72 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 in Section 3.0 , the phase shift cannot exceed 25% of the symbol duration at typical tag BLFs.

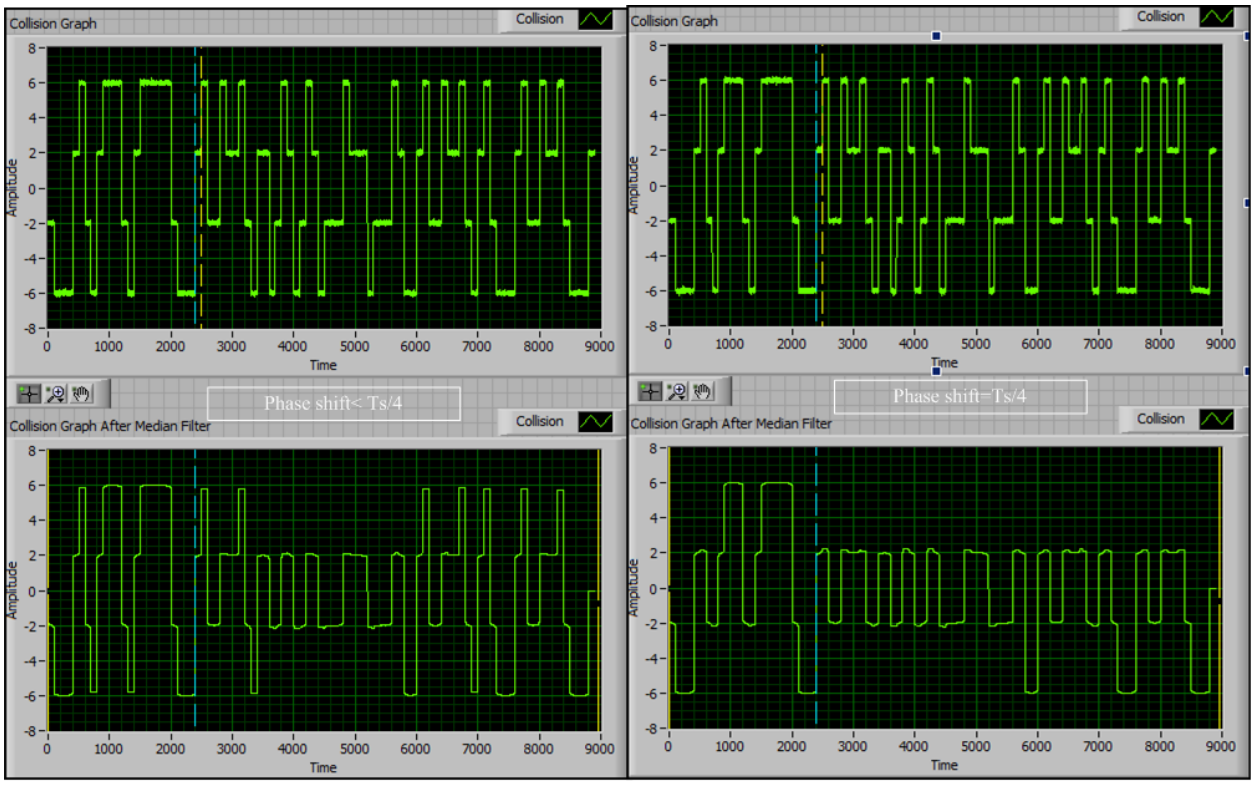

Figure 72. Effect of the Limitation of Phase Shift

6.0 TAG DYNAMICS

In Section 4.0 and Section 5.0 , the collision signal is acquired from stationary tags placed in the reader transmitting field. In the real-world, however, the tags are normally moving in certain application scenarios such as supply chain dynamic tracking. To study the effects of moving speed on tag signal acquisition and the algorithm's ability to resolve the collision from two dynamic tags, an experiment platform has been built to capture the collision signal from two moving tags as shown in Figure 73. Figure 74 shows the platform picture.

In the diagram, Tag 1 which is attached to a wooden bullet or shuttle (as shown in Figure 75) propelled by a CO_2 cartridge moves along the track, while Tag 2 is fixed at the illustrated position. When Tag 1 moves to the position where Tag 2 is located, the wooden bullet shields the light emitting from the photo sensor which causes the sensor sending out a positive pulse trigger signal to both the reader and the oscilloscope. After receiving the trigger signal, the reader sends out a Query command with Q=0 to the two tags, and begins to acquire the collision signal. The trigger signal is also captured by the oscilloscope, and the speed of the moving tag can be calculated by dividing the length of wooden bullet (10 cm) with the trigger pulse width.

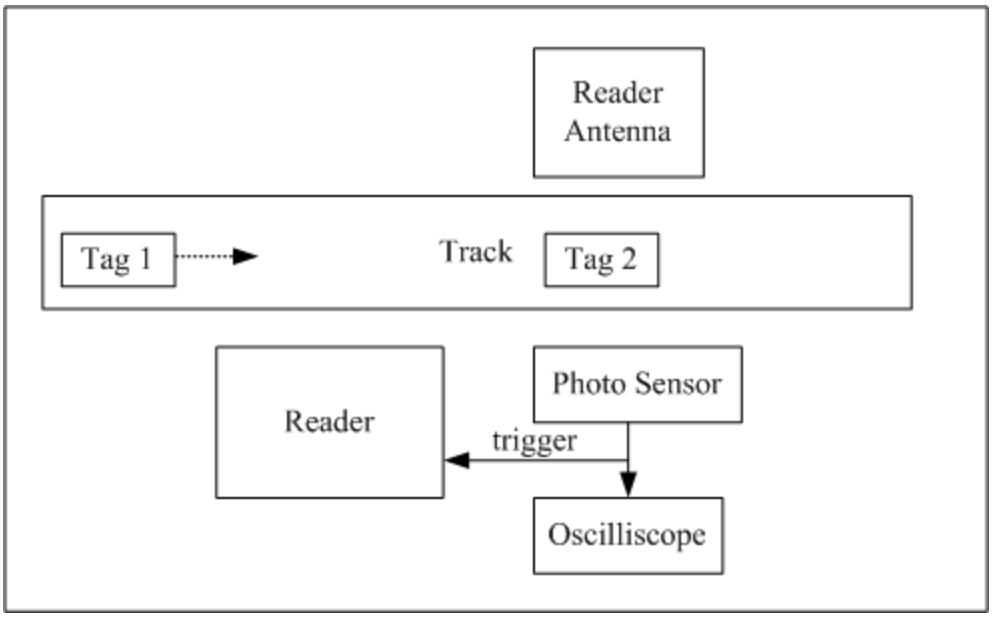

Figure 73. Moving Tag Data Acquisition Platform

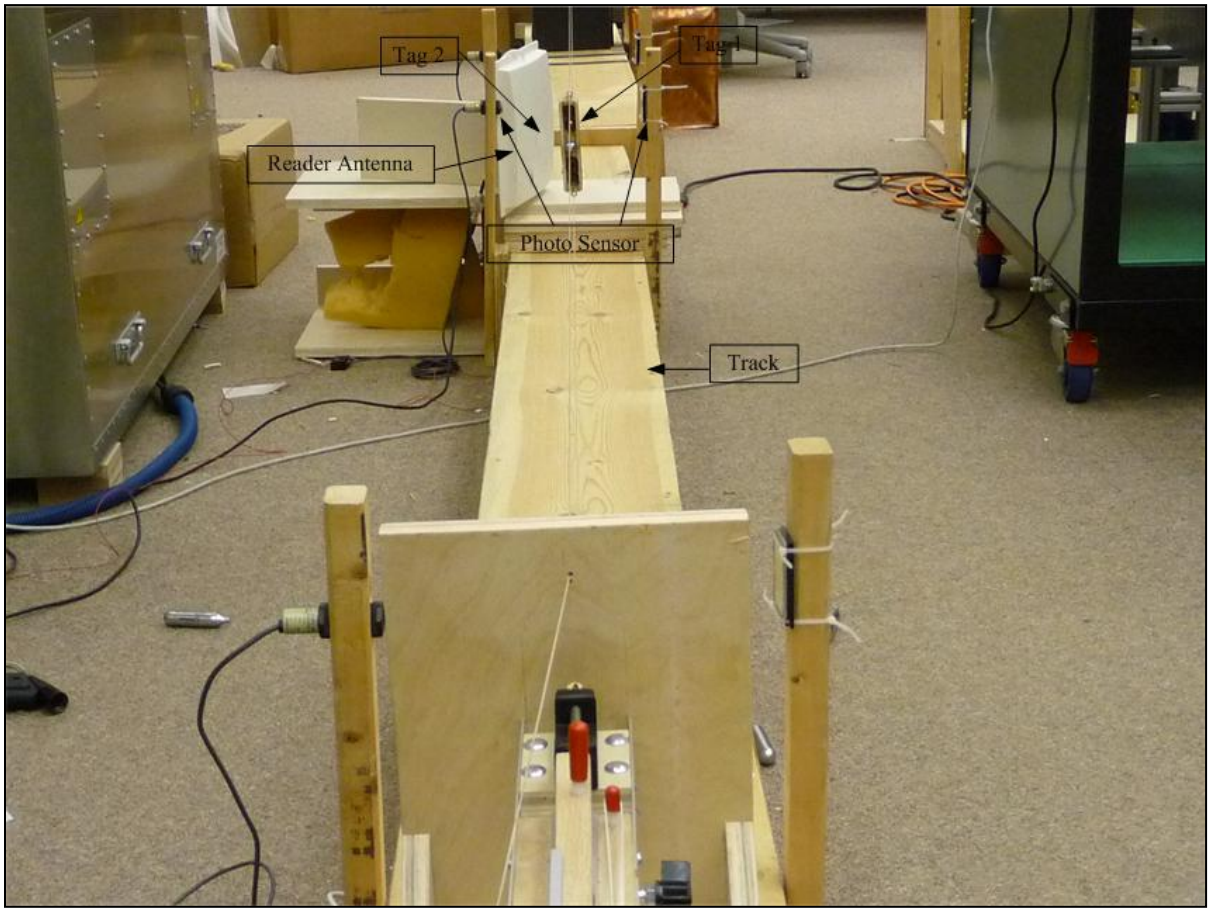

Figure 74. Moving Tag Data Acquisition Platform (Picture)

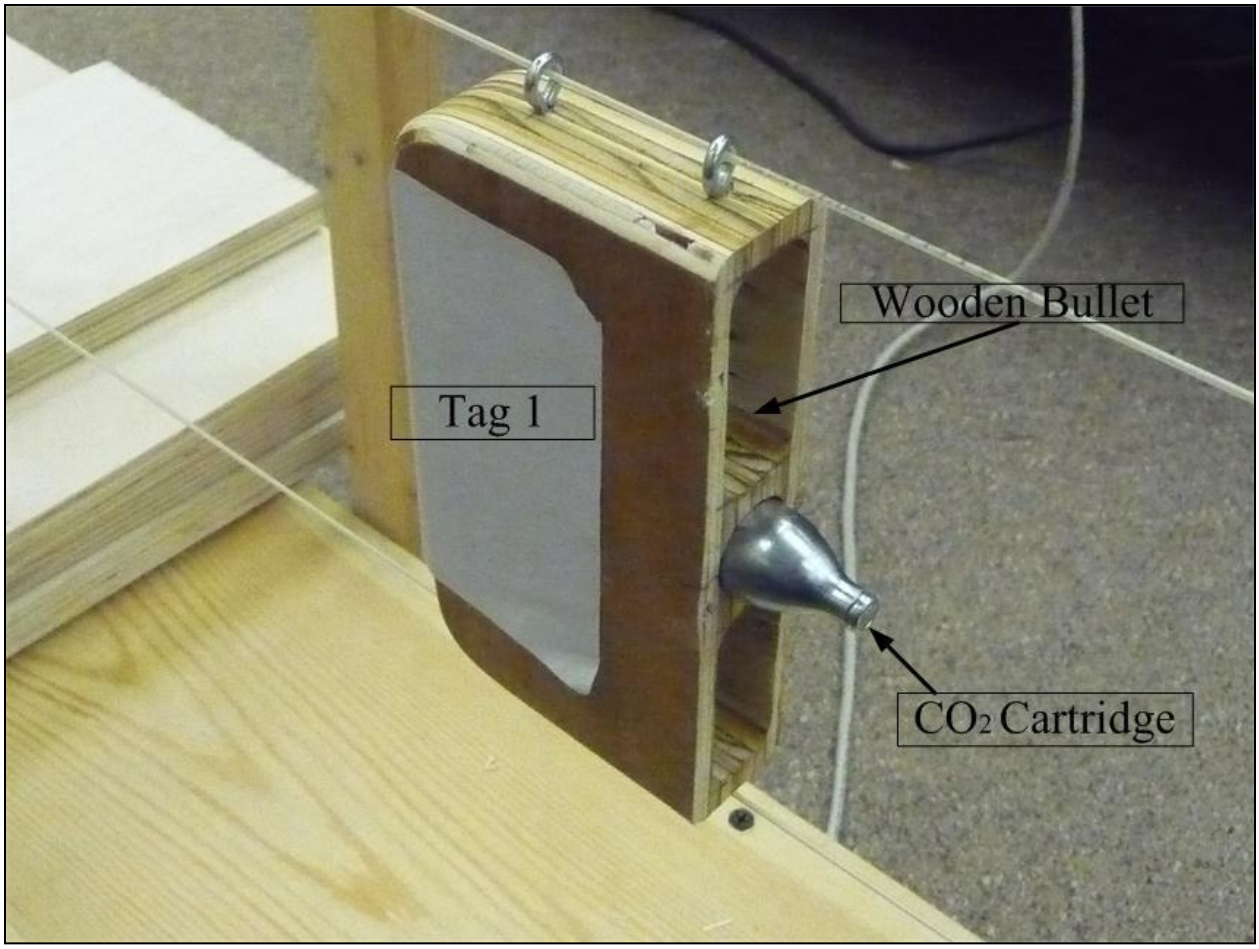

Figure 75. Wooden Bullet

When propelling the wooden bullet, the CO_2 cartridge is penetrated by a nail which is struck by a catapult as shown in Figure 76. The speed of the bullet can be controlled by adjusting the penetration extent of the nail into the cartridge, which is achieved by moving the position of the nail back and forth along the catapult trigger track as illustrated double edge arrow in Figure 76. The larger the hole size at the bottom of the cartridge after penetration, the greater amount of CO_2 emitted in the unit time, which leads to higher moving speed accordingly.

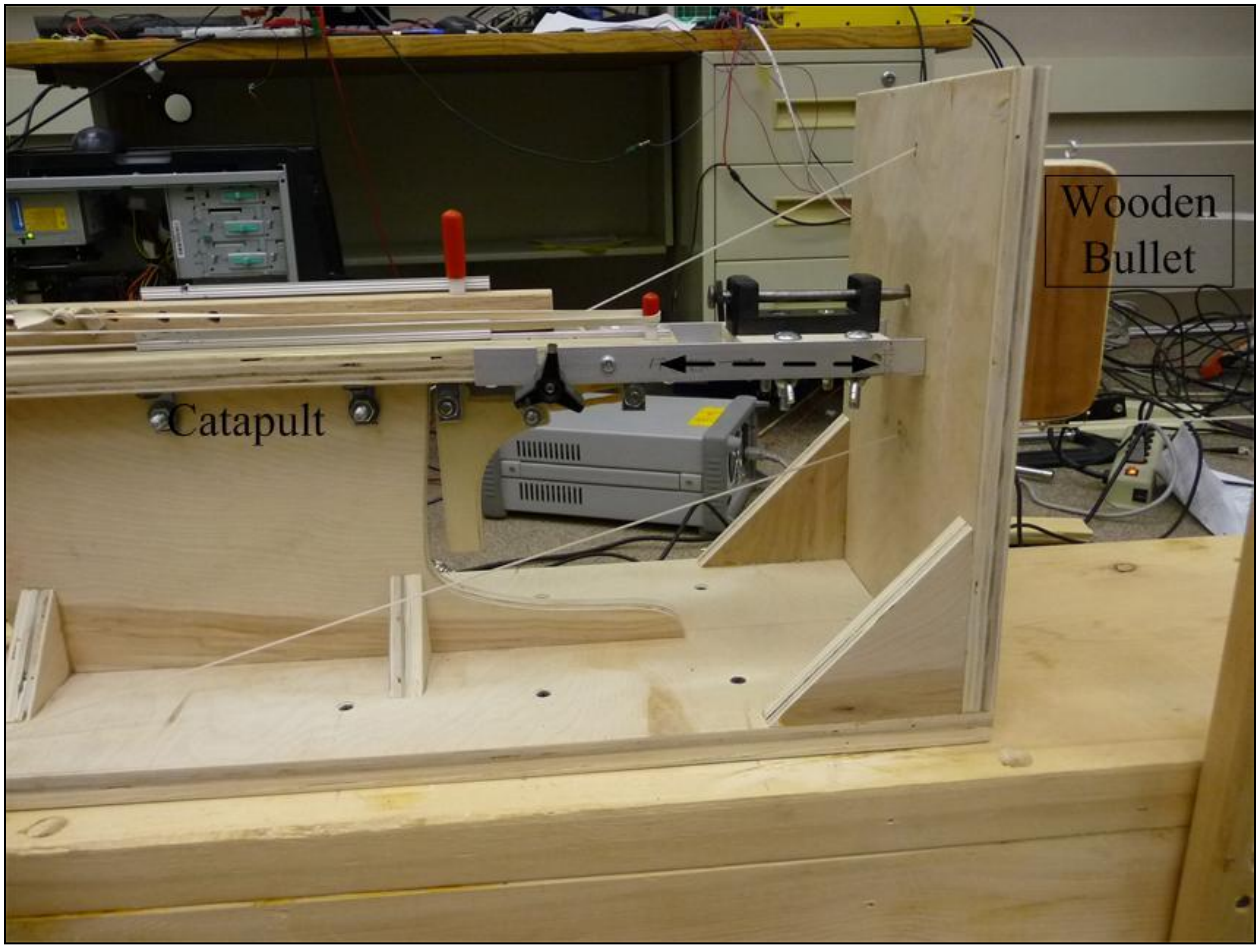

Figure 76. Wooden Bullet Triggering System

The type of the photo sensor is Omron E3F2-RC4CF, which outputs a 12V voltage when there is no object blocking its light. When the bullet moves to the position as shown in Figure 77, where the light beam emitted from the sensor is blocked, the sensor's output is 0 volt. However, the NI5640R FPGA board in the reader requires a TTL compatible trigger voltage level. Therefore, an interface circuit is required to transfer the 12V to 0V falling edge triggering signal to a 0V to 5V rising edge signal when connecting the sensor with the reader. A Maxim Max232 IC [7] is used in this case as shown in Figure 78. The input port 13 (RS232 Input) and the output port 12 (TTL/CMOS Output) with a built-in inverter is used to connect the sensor signal and the reader trigger input as circled in Figure 78.

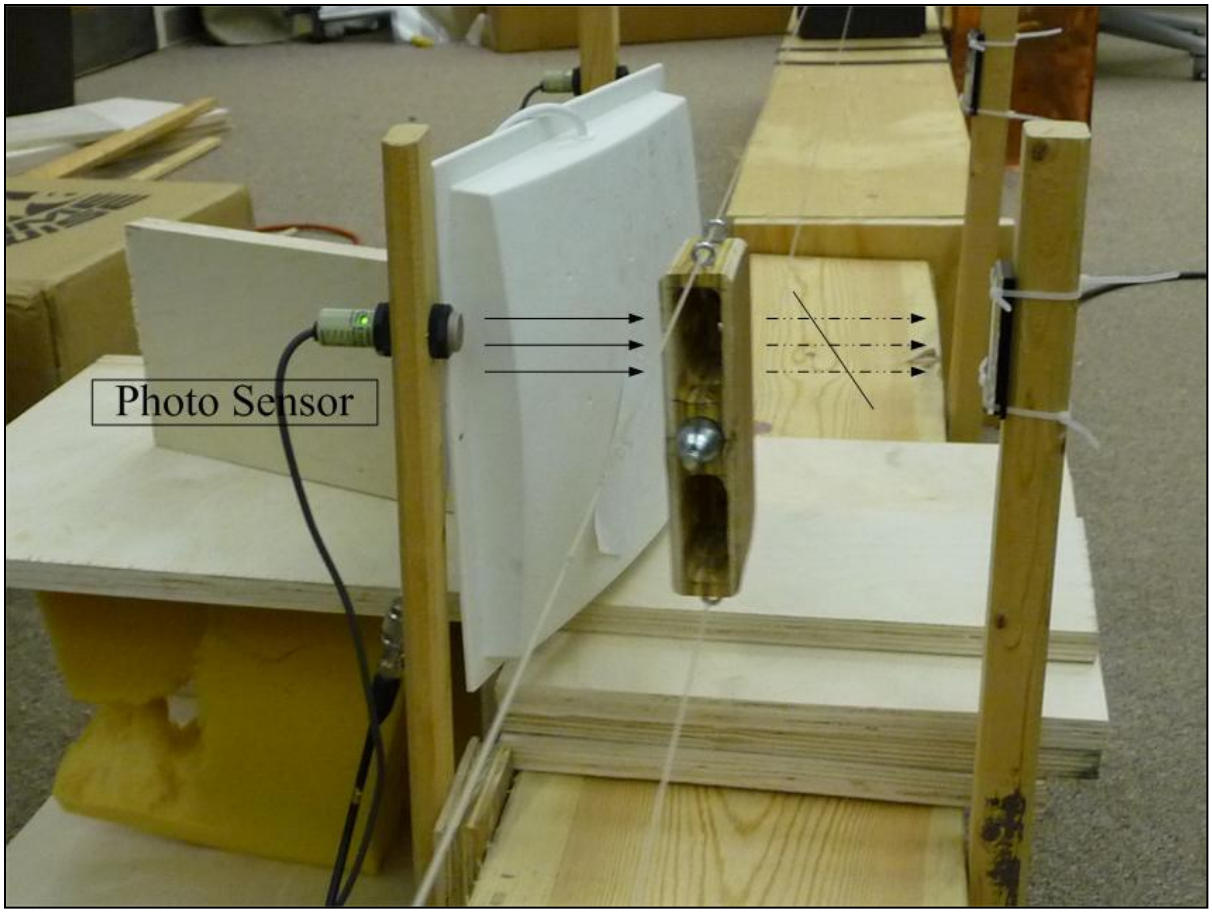

Figure 77. Wooden Bullet Blocking the Sensor Light

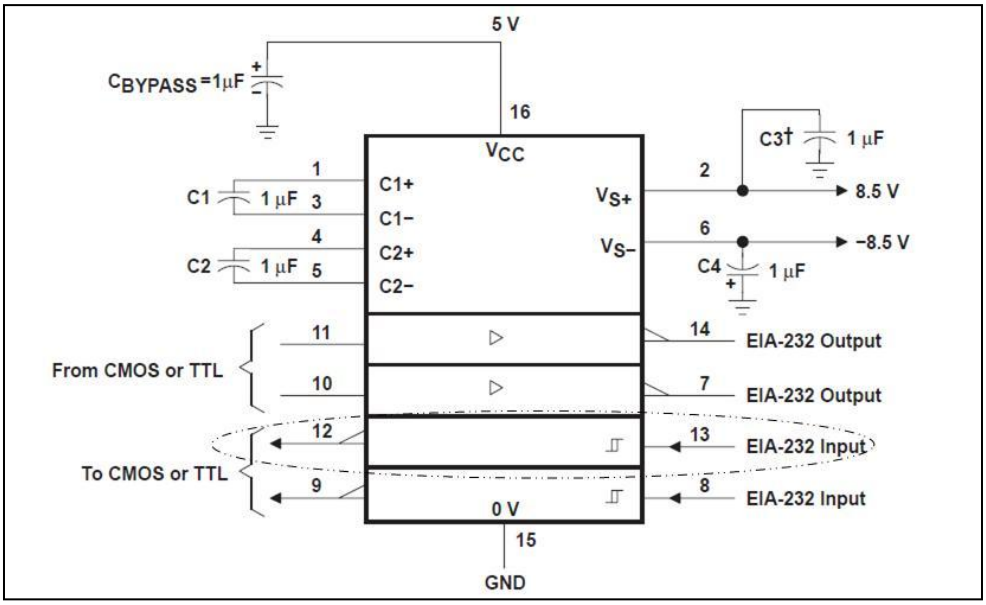

Figure 78. Max232 Port Connections

6.1 MOVING TAG COLLISION SIGNAL ANALYSIS

The tag collision signals are acquired from two distinct moving speed levels at 12 miles/hour (pulse width= 20ms) and 25miles/hour (pulse width= 9ms) separately as the trigger signal length displayed on the oscilloscope in Figure 79 and Figure 80.

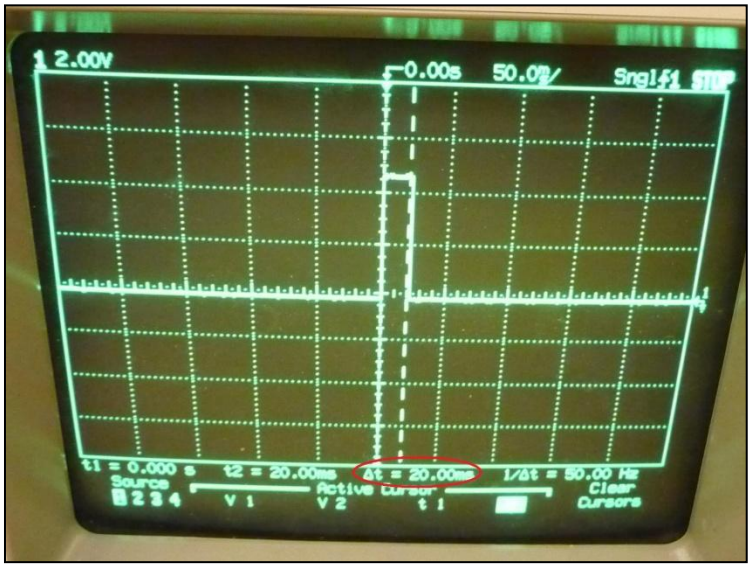

Figure 79. Trigger Pulse Width Corresponding to 12 miles/hour

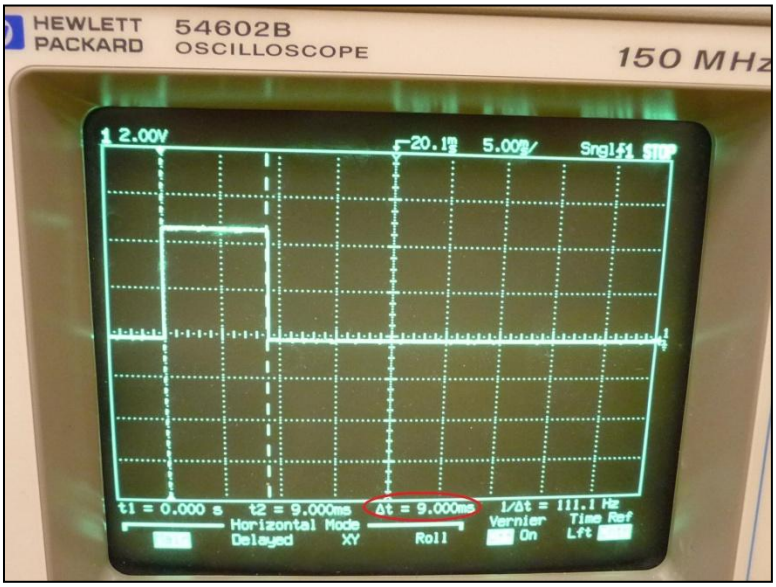

Figure 80. Pulse Width corresponding to 25 miles/hour

Figure 81~85 show the acquired 2-tag collision signals when the moving speed is 12 miles/hour at typical BLFs, and Figure 86~90 show the acquired 2-tag collision signals when the moving speed is 25 miles/hour at typical BLFs.

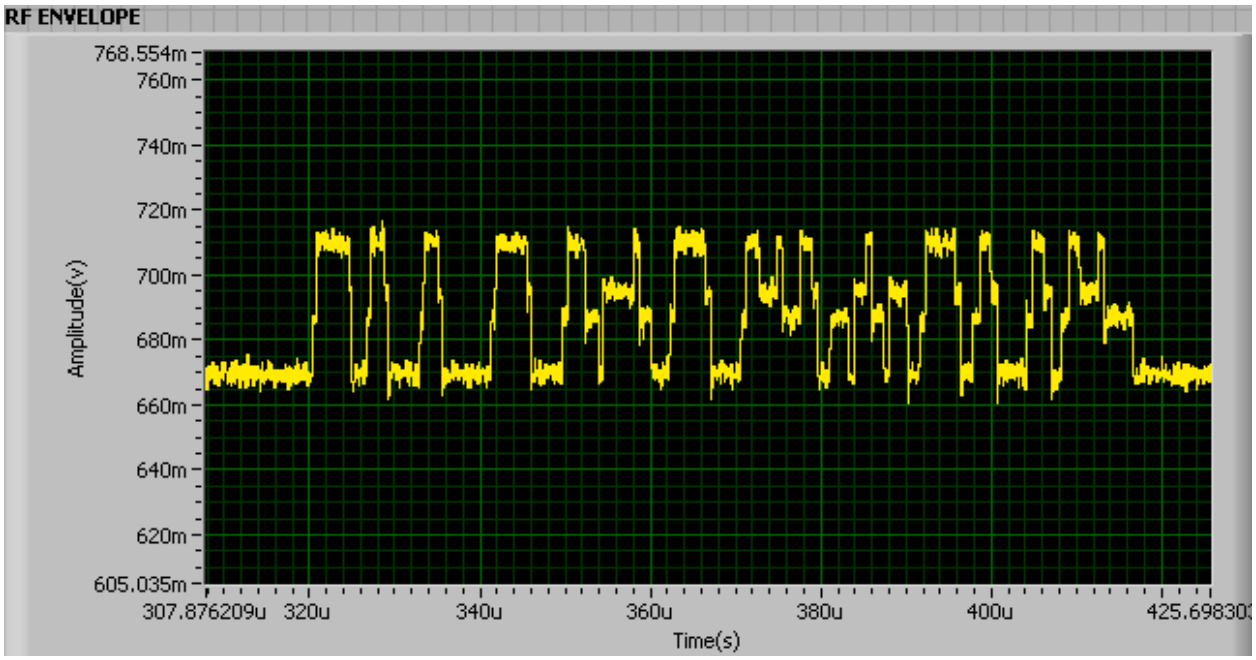

Figure 81. 2-tag Collision Signal (Speed=12miles/hour, BLF=64 kHz)

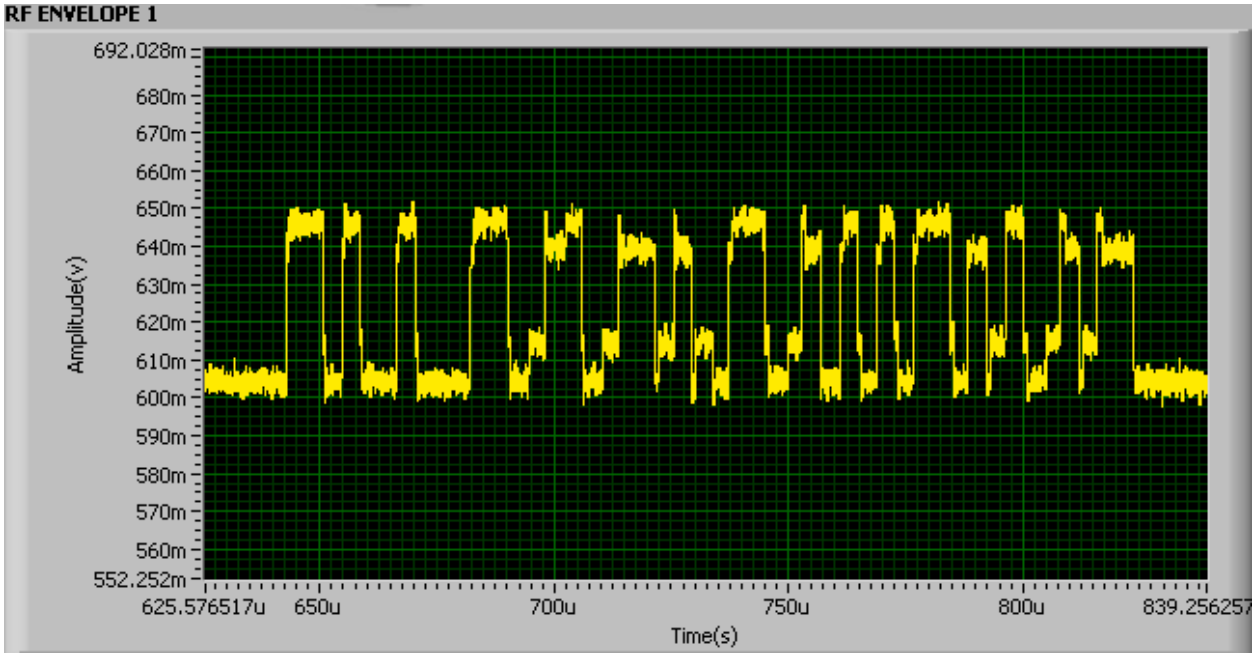

Figure 82. 2-tag Collision Signal (Speed=12miles/hour, BLF=128 kHz)

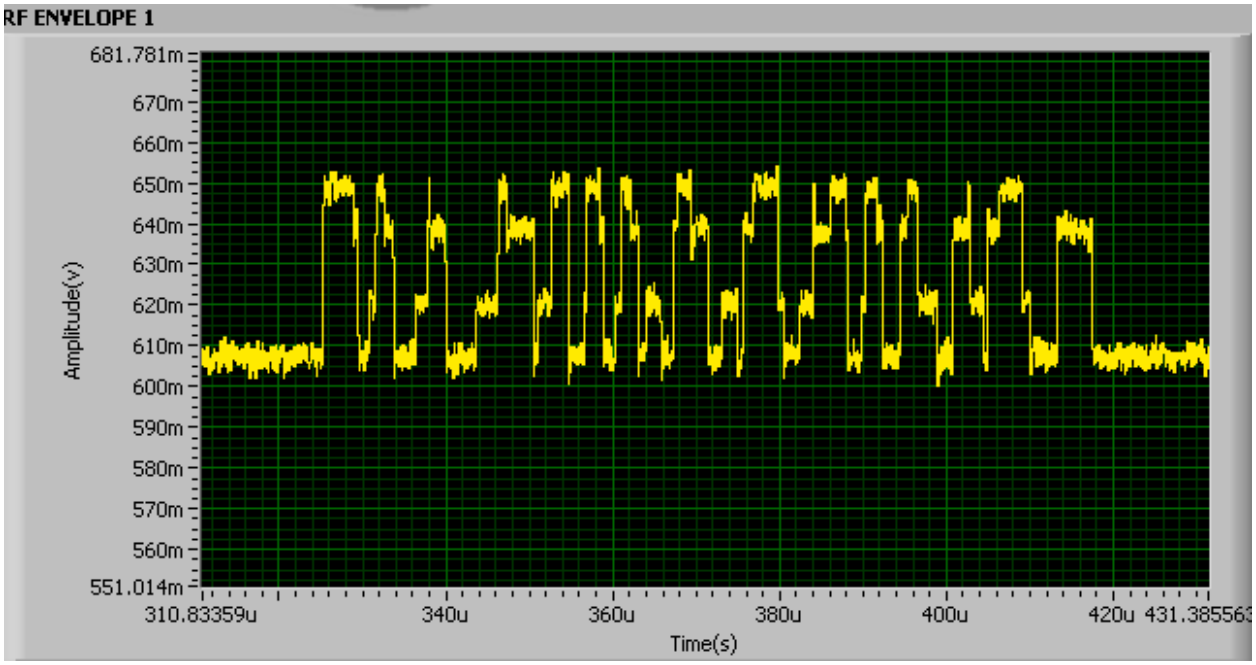

Figure 83. 2-tag Collision Signal (Speed=12miles/hour, BLF=256 kHz)

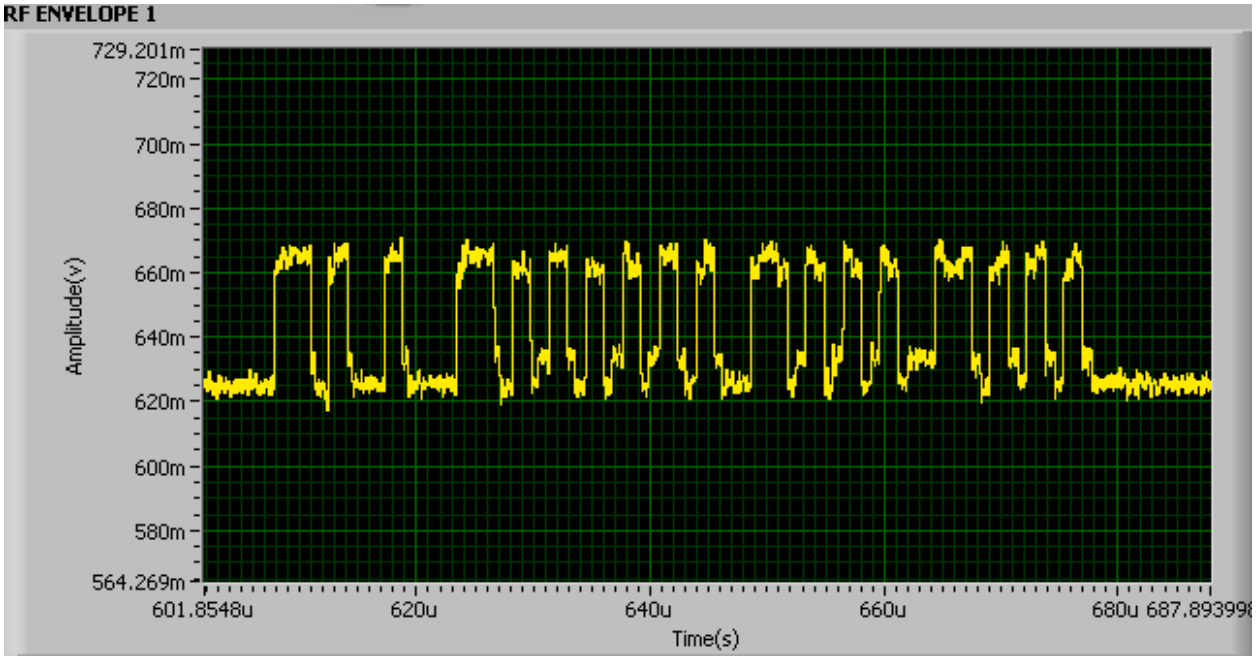

Figure 84. 2-tag Collision Signal (Speed=12miles/hour, BLF=341 kHz)

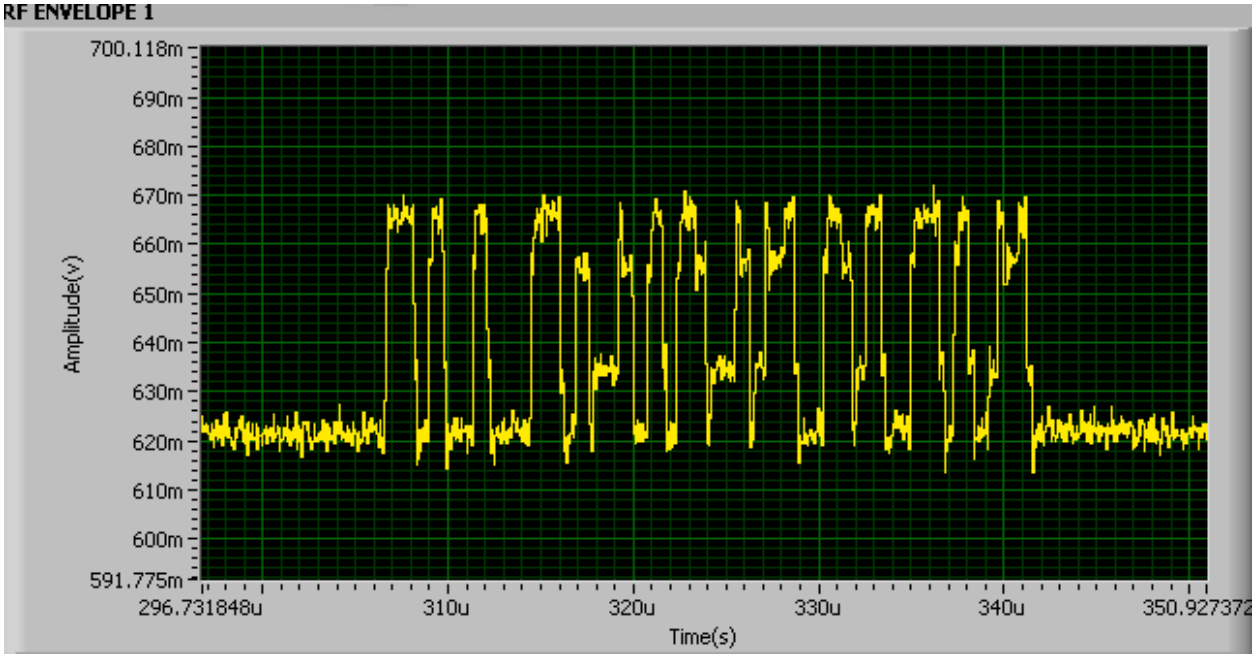

Figure 85. 2-tag Collision Signal (Speed=12miles/hour, BLF=682 kHz)

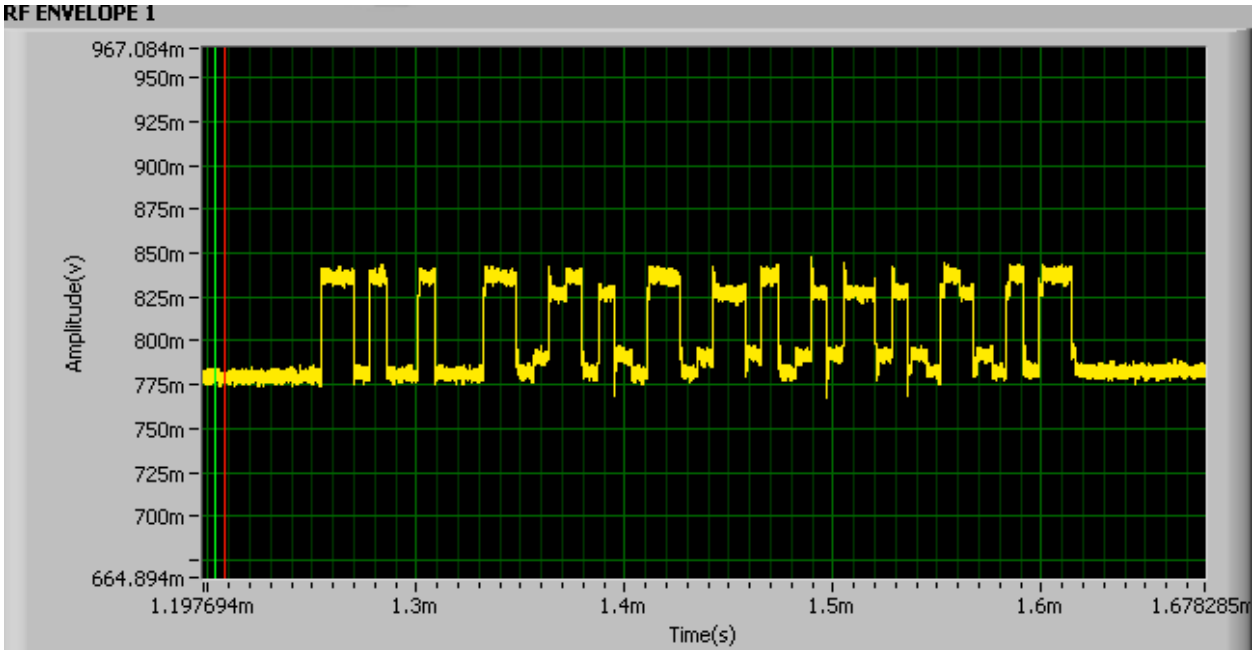

Figure 86. 2-tag Collision Signal (Speed=25miles/hour, BLF=64 kHz)

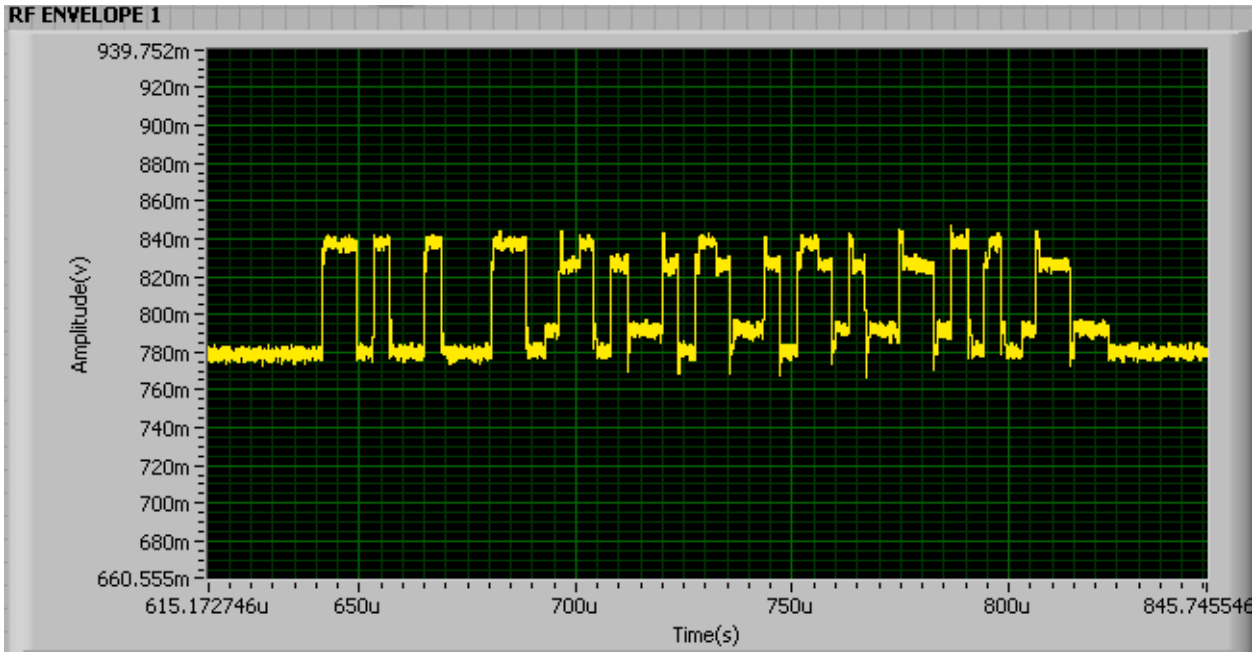

Figure 87. 2-tag Collision Signal (Speed=25miles/hour, BLF=128 kHz)

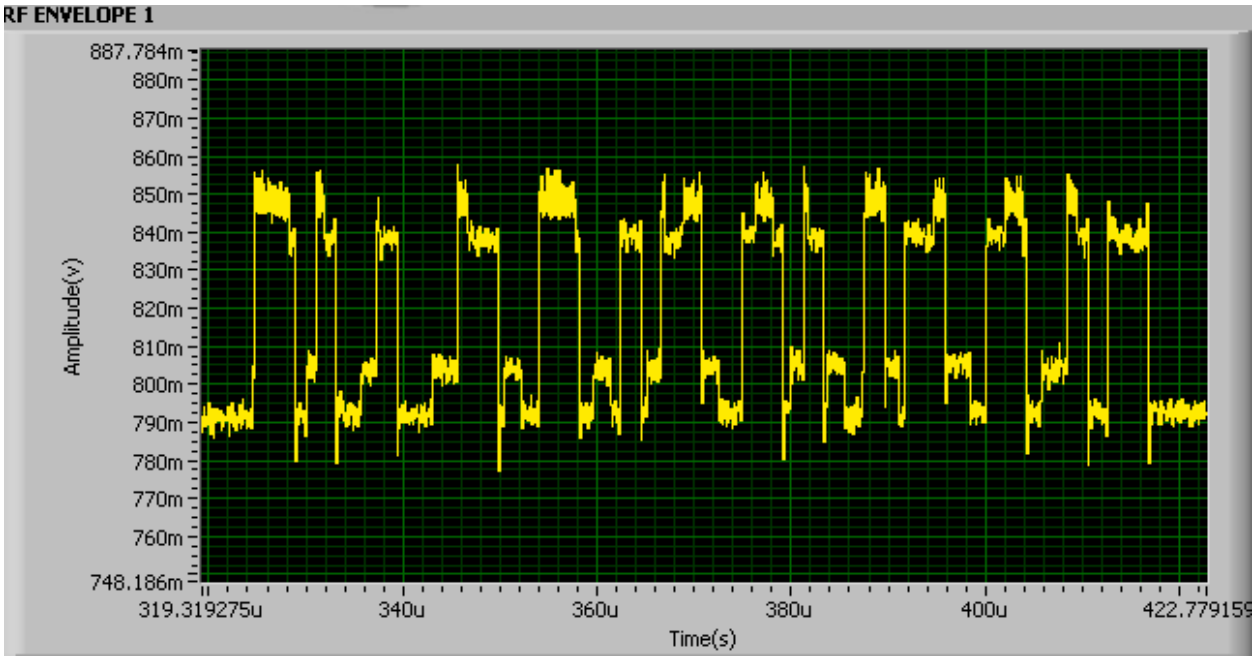

Figure 88. 2-tag Collision Signal (Speed=25miles/hour, BLF=256 kHz)

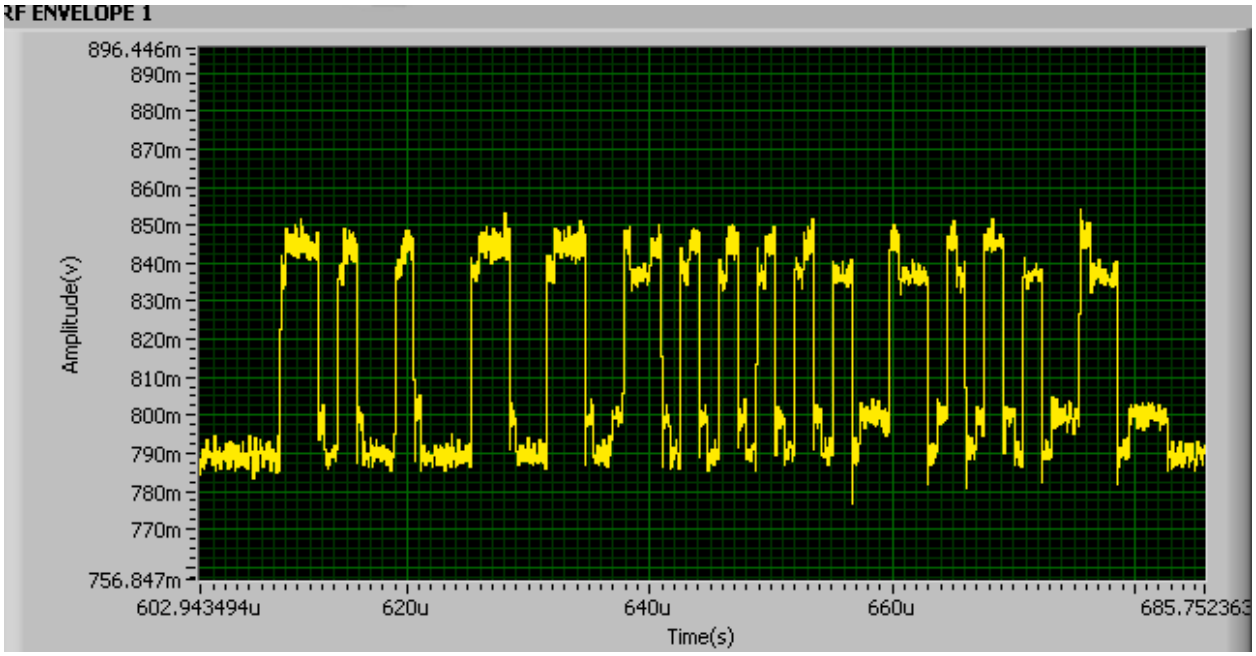

Figure 89. 2-tag Collision Signal (Speed=25miles/hour, BLF=341 kHz)

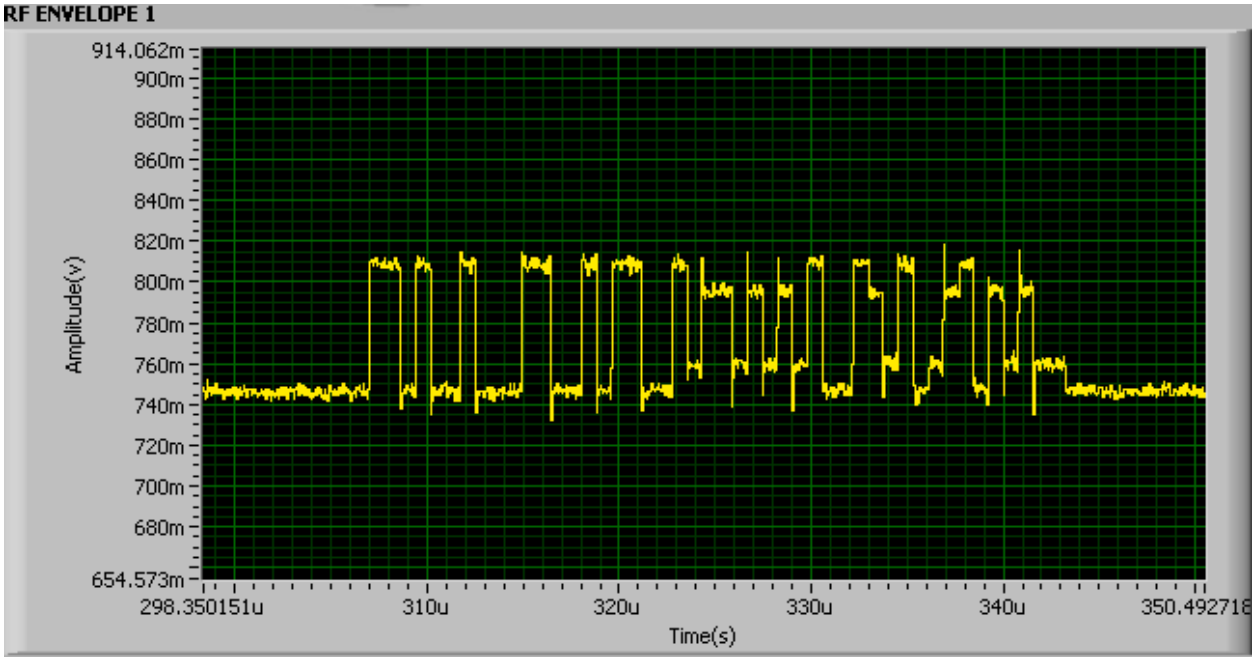

Figure 90. 2-tag Collision Signal (Speed=25miles/hour, BLF=682 kHz)

Figure 91 shows the comparison of average SNR of the acquired collision signal at different speed. As shown, the SNR is relatively stable which implies the collision signal quality

is irrelevant of the tag moving speed. Because the collision signals from moving tags do not differentiate from stationary tags, they are successfully resolved by amplitude mapping method after the median filter preprocessing. Therefore, there is no bound of the resolution method's application to the moving tag cases provided the reader firmware can acquire the collision signal fast enough.

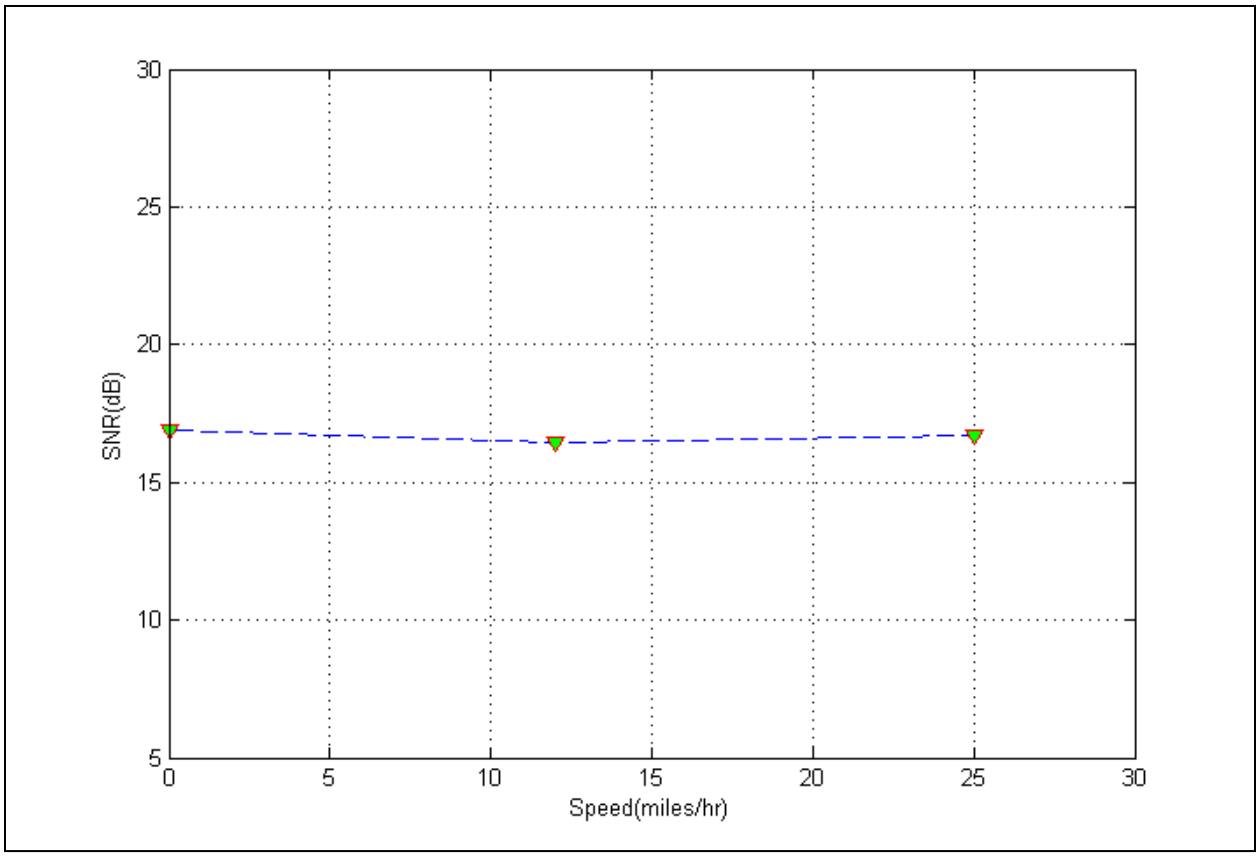

Figure 91. Average Signal-to-Noise Ratio Comparison

7.0 ICA ALGORITHM ANALYSIS AND SIMULATION

As discussed in Section 4 and Section 5, direct edge locating can resolve two-tag collision 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 unifying the edge mapping method with the direct edge locating method, the phase shift cannot be greater than 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 similar 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 previously proposed solutions?

7.1 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 of the popular methods 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 mutually independent, and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by 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 include biomedical signal processing, audio signal separation, telecommunications, fault diagnosis, feature extraction, financial time series analysis, and data mining began to appear [9]. Because each tag’s response with effective reader command is independent, and the statistical

characteristics of the response signal is fixed and can be pre-known by checking large samples 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 conflicting tags are all 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 consist 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.7.1, 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.7.1) is M, and the number of receiving channels (the size of the X vector in Eq.7.1) 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.7.1 square (because only a square matrix has an inverse). The ICA algorithm is then performed to find the inverse of A in an iterative manner until the algorithm converges.

$$X = AS \qquad \text{(Eq.7.1)}$$

$$X = (x_1, x_2, \ldots, x_m)'; S = (s_1, s_2, \ldots s_n)';$$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

7.2 ALGORITHM DESCRIPTION

Various practical methods for employing the ICA model can be employed for the current application. Reference [9] 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 of minimizing the mutual information. According to [9], 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. 7.2) is a measure of non-Gaussianity. (Where, $P_y(y)$ is the pdf function of signal y.)

$$\mathrm{H(y)} = -\int P_y(y) \log P_y(y)\, dy \quad \text{(Eq. 7.2)}$$

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 [11]. According to the ICA model in Eq.7.1, the mixture signal X (multiple tag collision signal) is the linear superposition of the source signal (the response of each tag), this implies that the distribution of the mixture signal approaches the 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 compared to the collision signal for minimum entropy. To obtain a measure of non-Gaussianity, which is zero for a Gaussian variable and always nonnegative, one often uses a normalized version of differential entropy, called negentropy J as defined in Eq.7.3. (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.

$$J(y) = H(y_{gauss}) - H(y) \quad (\text{Eq. } 7.3)$$

Although Entropy and Negentropy can be used to measure the Gaussianity, the integral in the calculation hinders the computation efficiency. As an effective approximation, the Negentropy of signal can be calculated without involving the integral as shown in Eq.7.4. (Where G is a nonquadratic function)

$$J(y) \propto [E\{G(y)\} - E\{G(v)\}]^2 \quad (\text{Eq. } 7.4)$$

A. Hyvärinen and E. Oja proposed a computation efficient algorithm FastICA [10] based on a fixed-point iteration scheme for finding a maximum of the non-Gaussianity as measured in (Eq.7.4). The algorithm performs a Gaussian-Newton optimization when maximizing the Negentropy. Following Eq.7.1, the source can be recovered from the collision signal X by left-

multiplying the collision with a separating matrix W (as shown in Eq.7.5), which is the inverse of the mixing matrix X. Table 10 lists the FastICA algorithm using Negentropy maximization.

$$S = WX; \qquad (Eq.\ 7.5)$$

$$X = (x_1, x_2, \dots, x_n)'; \quad S = (s_1, s_2, \dots s_n)';$$

$$W = (w_1', w_2', \dots w_n');$$

w_n' is the nth row in the separating matrix.

Table 10. FastICA algorithm

1. Preprocessing the collision signal X by removing the mean
2. Whiten the data to Z, ($Z=(z_1,z_2,\dots,z_n)$')
3. Initialize the w' vectors (each row) in the separating matrix W
4. Update w = E{Z*g*(w'Z)}- E{*g*'(w'Z)}w ,where g is the first derivative of the G function in Eq.7.4.
5. Normalize w= w/||w||
6. If not converged, go back to step 4.

7.3 ALGORITHM OPTIMIZATION

7.3.1 Host PC Floating Point Algorithm with Batch Training

The FastICA algorithm is simulated using LabVIEW on the Host PC. The G function in Eq. 7.4 is selected according to Eq. 7.6. Therefore, its derivation, *g(u)*, is a cubic function.

$$G(u) = \frac{1}{4}u^4 \qquad Eq.\ 7.6$$

To verify the functionality, the FastICA algorithm as shown in Table 9 is realized in LabVIEW using double precision floating point on the Host PC. The algorithm reads all the data points of the collision signal and performs the computation based on the expected value of the data. This is similar to batch training rather than point-by point online training.

Figure 92 shows the two tag collision resolution simulation result using the FastICA. Two RN16 numbers D2CCh (S1) and 3D74h (S2) are generated in FM0 encoding and mixed using a mixing matrix as shown in Figure 92 to simulate the two collisions (X_1 and X_2) captured by the two independent receiving channels.

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 the recovered S1) and 3D74h (for the recovered S2) 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., the mixing matrix A, and then comparing the products with the mixtures X (the upper-most in Figure 93). The algorithm converges in just two iterations due to the relatively uncomplicated formation of RFID signal which is similar to a simple bipolar square wave.

Compare the mixing matrix and the separating matrix, the first row in the mixing matrix is vector [1, 0.5], which corresponds to the direction of the second column in the separating matrix ([0.732978, -0.491724]); the second row in the mixing matrix is vector [2, 6], which corresponds to the direction of the first column in the separating matrix ([-1.46256, 4.69849]). This correspondence relationship between the mixing matrix and the separating matrix reveals that the separation is successful.

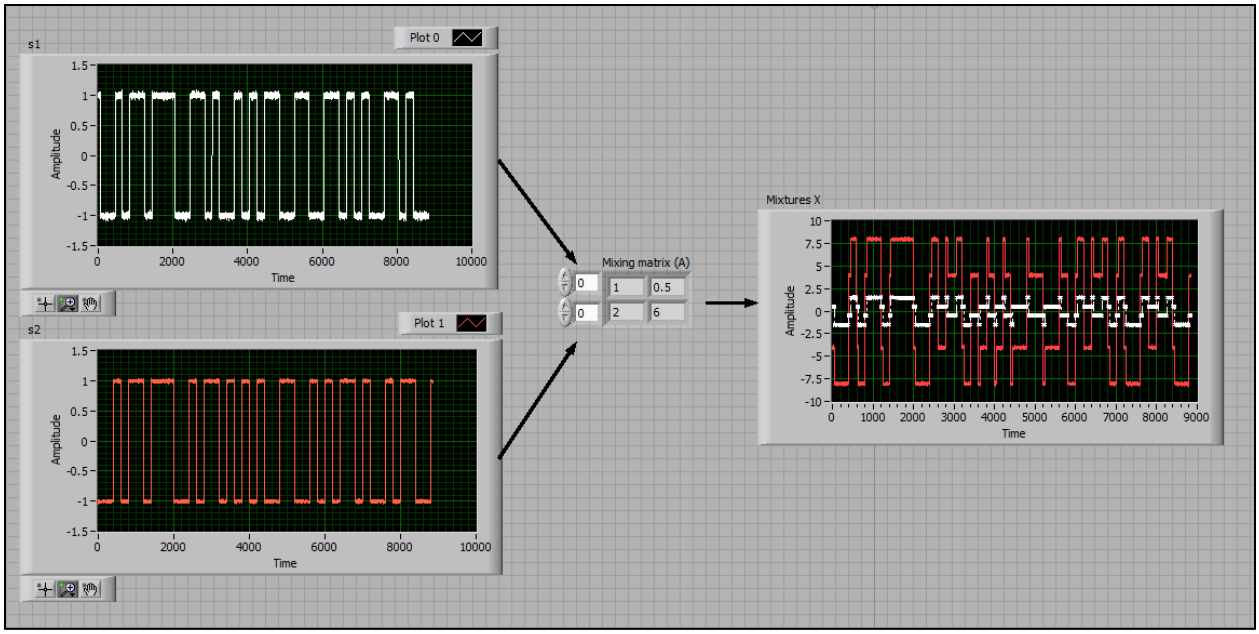

Figure 92. Simulated Tag Signal Mixing

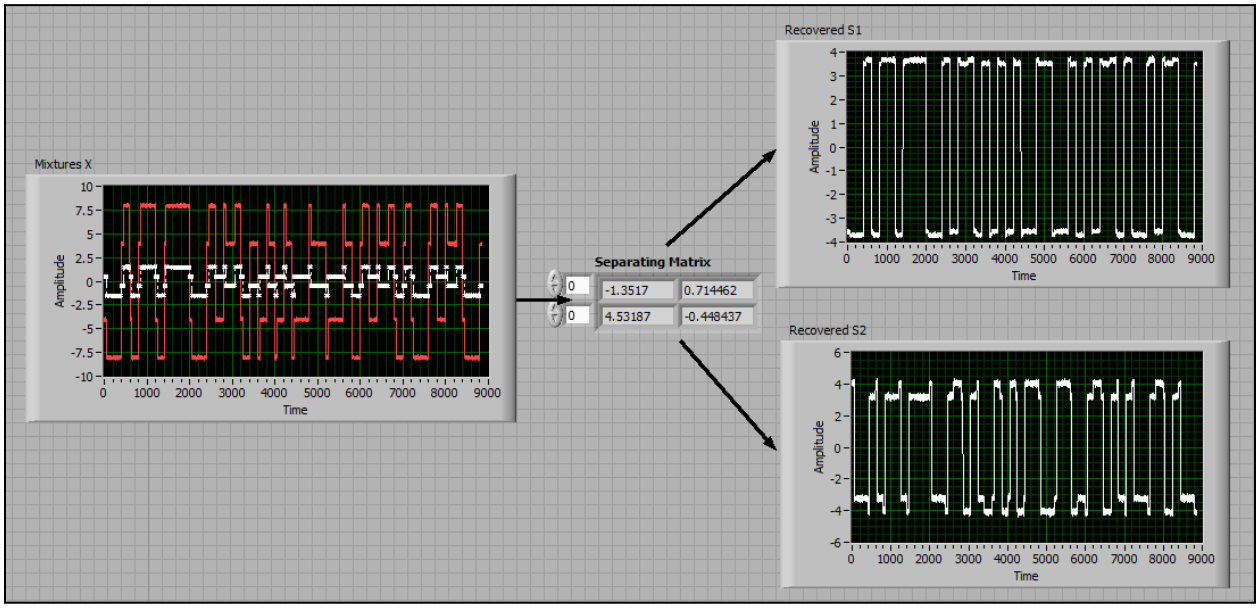

Figure 93. Resolution Result

7.3.2 Design of FPGA based Fixed-Point Algorithm

The computation load of the FastICA linearly increases with the number of collided tag responses. In one extreme case, when the tag BLF is 64 kHz, the length of the collision signal can reach more than 8000 double precision floating-point data elements if the acquisition sampling frequency is 25MHz. To process this large amount of data points using the FastICA including the data whitening and the matrix weight updating in the FPGA device in the standard specified real time is unrealistic.

There is a choice between on-line and batch algorithms. An on-line version of the algorithm can be obtained by substituting the expectation value in the algorithm with the instant data value. However, the on-line algorithm trades the accuracy for processing speed, and the algorithm may not converge as well as the batch training algorithm. This problem can be alleviated by combinations of the two as shown in Figure 94. Rather than performing the FastICA after completely receiving the collision signal, the "divide and conquer" architecture in Figure 94 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 separating matrix W is successively updated: W1 is updated based on W0, W2 is updated based on W1, …etc, until the change between successive W becomes minimal. Once the W converges at a certain stage, the ICA does not need to 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 end of the collision.

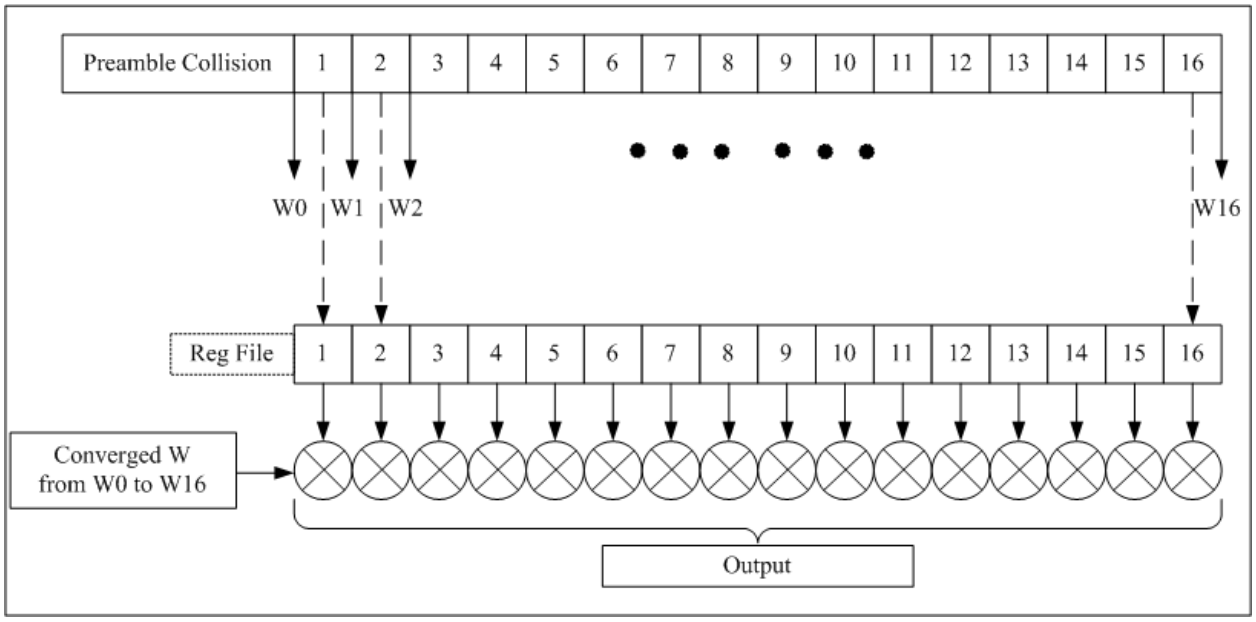

Figure 94. Combination of Batch Training and Online Training

In addition, the following measures also decrease the resolution computation load significantly:

1. In fact, the only useful feature of the collision for ICA to train its parameter is the way in which the two tag signals S1 and S2 are mixed, i.e., the components in the mixing matrix. Although the collision signal includes both the preamble part and the RN16 part, not all of them are needed to extract the mixing information for resolution, i.e., there are redundant information in the data). Therefore, one or two bit length data in the collision signal may be sufficient to represent the mixing characteristics of the entire collision signal, and the ICA can be trained only using one portion of the collision signal as shown in Figure 95.

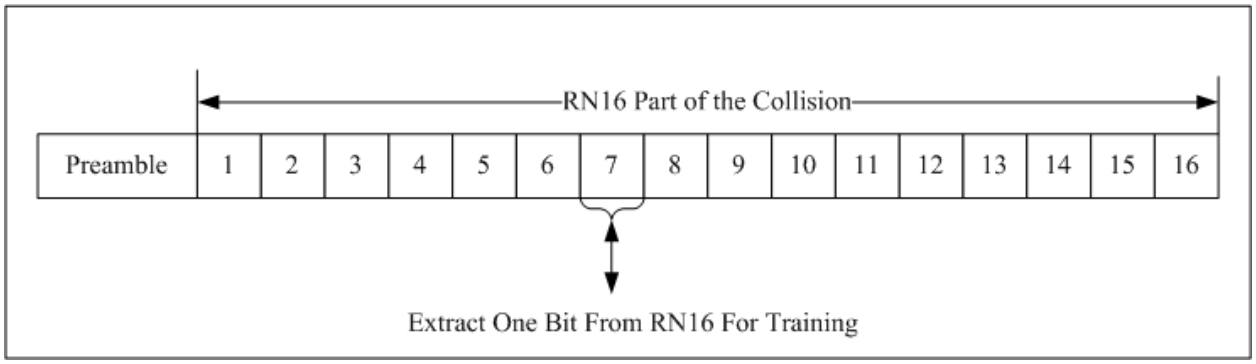

Figure 95. Perform ICA on Portions of the Entire signal

2. Once the separating matrix (the inverse of the mixing matrix) is obtained by the ICA, not all of the data points in collision signal X_1 and X_2 need to be multiplied with it to retrieve the original source data T1 and T2. Because in each data bit of the RN16 there are at most two possible voltage levels (when there is an edge transition in the middle), only two data points from the first half, and the second half of each bit duration need to be multiplied with the inverse of the mixing matrix. Therefore, only 32 fixed point dot product calculations need to be performed by the FPGA to recover data for one tag.

3. The FPGA acquisition sampling frequency is 25MHz, but the maximum tag BLF is 682 KHz. Therefore, multi-rate DSP techniques can be employed to reduce the computation load of ICA. The two collision signals X1 and X2 from two receiving antennas are down-sampled by a factor of M (M=8, 16 or 32) before being sent into ICA module as shown in Figure 96. In case of M=16, the sampling rate after down sampling is 25MHz/16=1.5625MHz, it still satisfies with Nyquist sampling theorem for the fastest BLF.

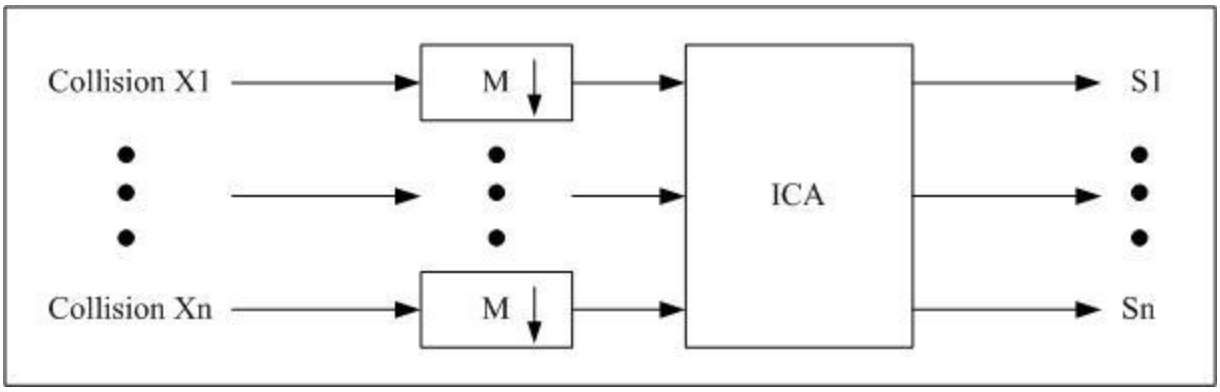

Figure 96. Collision Signal down Sampling

Synthesizing the above optimization options, the FPGA implementation for 2-tag collision resolution performs the algorithm on a collision signal subset of 512 data points and the down sampling factor M is selected as a value of 32 (acquisition resolution is 25 MHz); The FPGA implementation for 3-tag collision resolution performs the algorithm on a collision signal subset of 256 data points and the down sampling factor M is selected as a value of 8 (acquisition resolution is 12.5 MHz). The Fixed-Point math Library provided by LabVIEW FPGA module is used to realize the number format conversion and fixed point calculations. Therefore, only 512/32=16 fixed point data are used to train the weight in the ICA separating matrix in 2-tag collision situation, and 256/8=32 fixed point data are used in 3-tag collision situation. A major reason that 16 and 32 are selected is to avoid fixed-point divisions during the expectation calculation because a number divided by 16 and 32 can be replaced by right shifting.

7.4 ALGORITHM IMPLEMENTATION

7.4.1 2-Tag Collision Resolution

To resolve 2-tag collision, two independent receiving channels are required to acquire the collision signal. Figure 97 shows the experiment setup. Two RF front-ends including both the NI 5610 upconverter and NI 5600 downconverter are connected to one NI 5640R FPGA based transceiver through separate Analog I/O ports. During the data acquisition, the reader antennas are placed at a 45 degree angle to each other, and the two tags are held together as shown in Figure 98.

Due to the resource capability of the FPGA hardware (Xilinx Virtex-II Pro XC2VP30) in the NI 5640R, the FastICA algorithm is scaled to recover the first tag data, i.e., only trained for the first column of the separating matrix. The second tag data can be recovered by running the FPGA program twice (in sequential order). This is reasonable because the data of both tags can be recovered from the collision simultaneously if the corresponding processing units/circuits can be placed in parallel in future FPGA hardware with sufficient logic slices, e.g., NI 5641 with the Xilinx Virtex 5 Chip. In addition, when future design requires resolving collision signal from more than three tags, a higher volume FPGA device can handle the additional receiving channels in parallel.

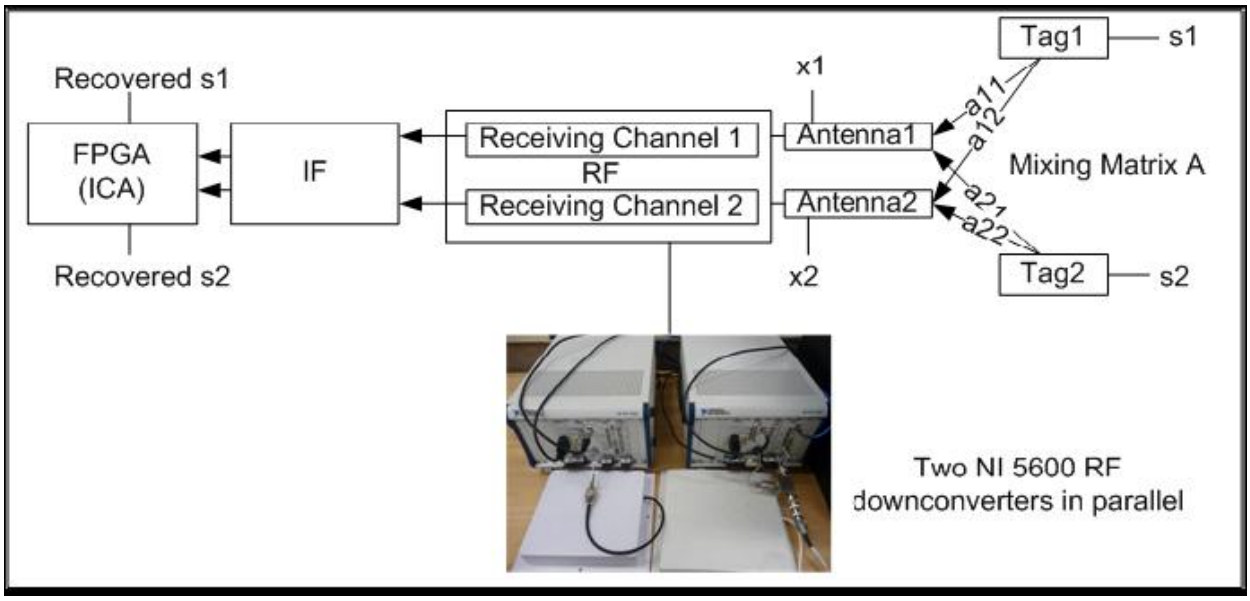

Figure 97. 2-Tag Collision Signal Acquisition Setup for ICA

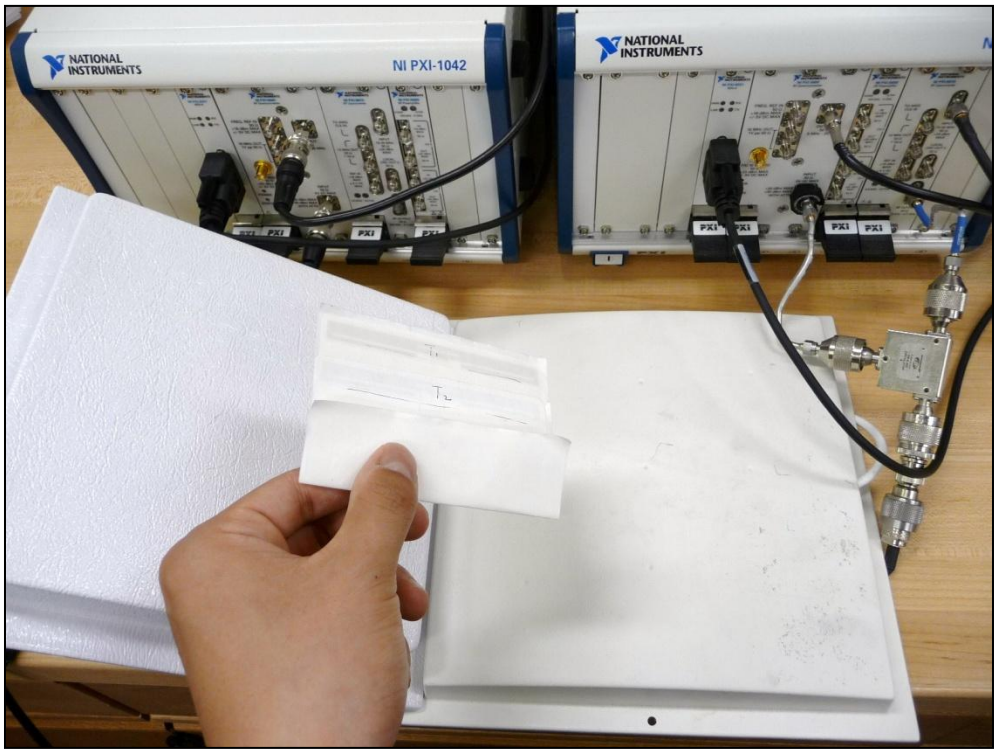

Figure 98. Tag and Antenna Positioning

Figure 99~103 show the resolution result on real tag collision signal at typical BLFs.

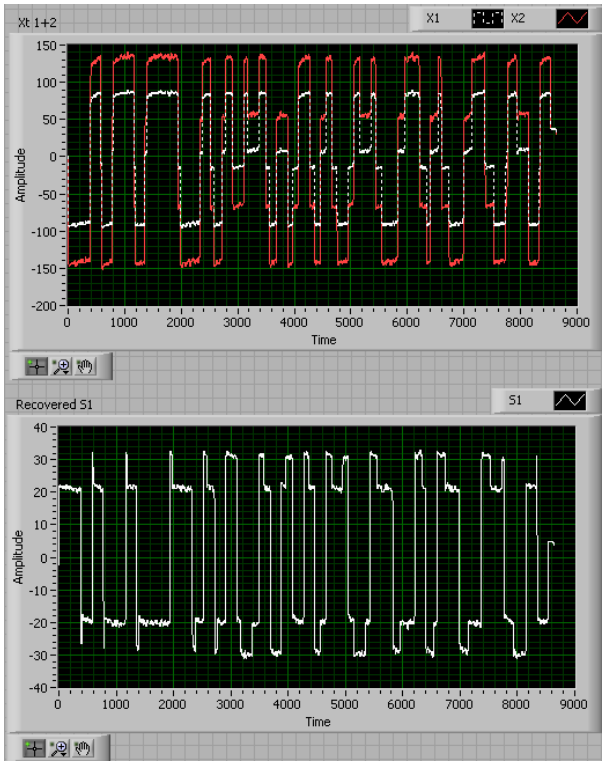

Figure 99. 2-tag Collision Resolution Result using FastICA in FPGA (BLF=64 kHz)

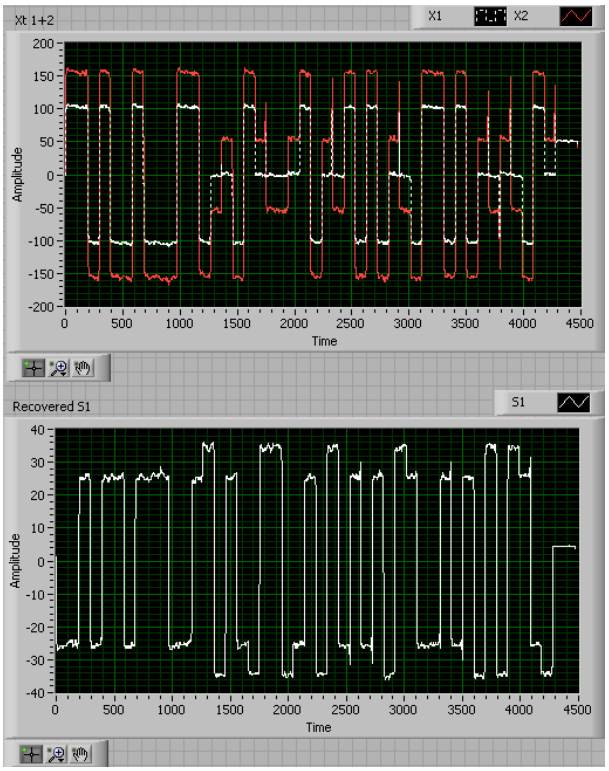

Figure 100. 2-tag Collision Resolution Result using FastICA in FPGA (BLF=128 kHz)

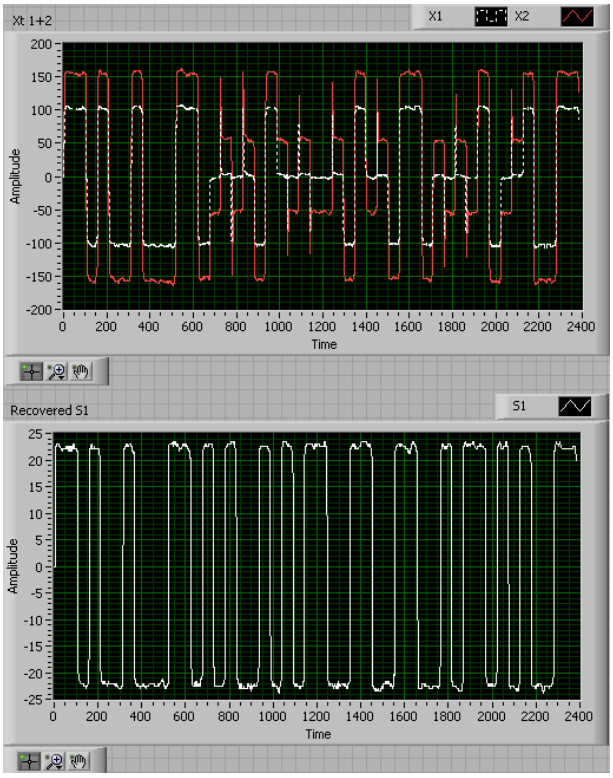

Figure 101. 2-tag Collision Resolution Result using FastICA in FPGA (BLF=256 kHz)

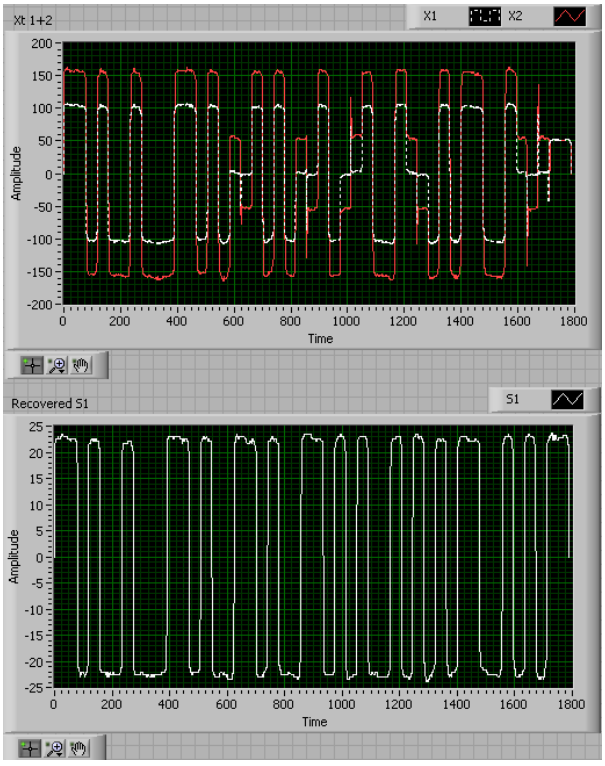

Figure 102. 2-tag Collision Resolution Result using FastICA in FPGA (BLF=341 kHz)

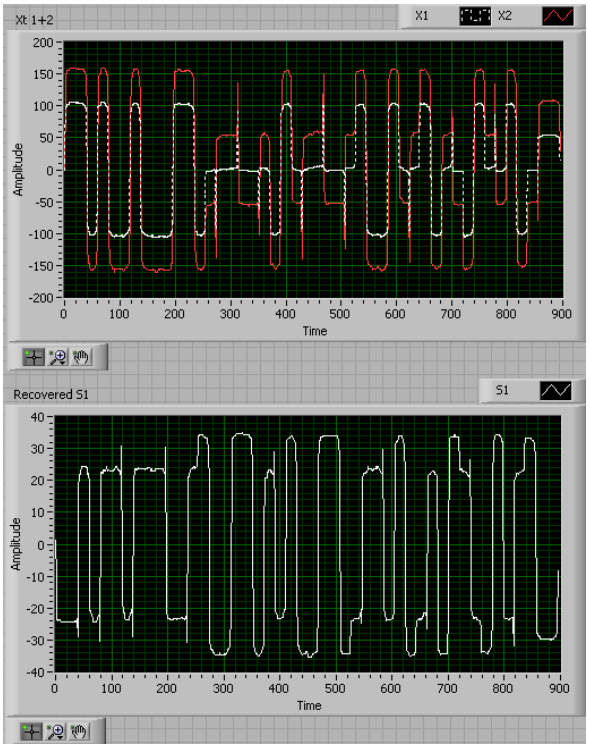

Figure 103. 2-tag Collision Resolution Result using FastICA in FPGA (BLF=682 kHz)

In the FPGA, two timers are activated to record the calculation processing speed as shown in Figure 104 (one for the starting instant, another for the ending instant). The ICA processing time is calculated by deducting the value of the starting timer from the ending timer.

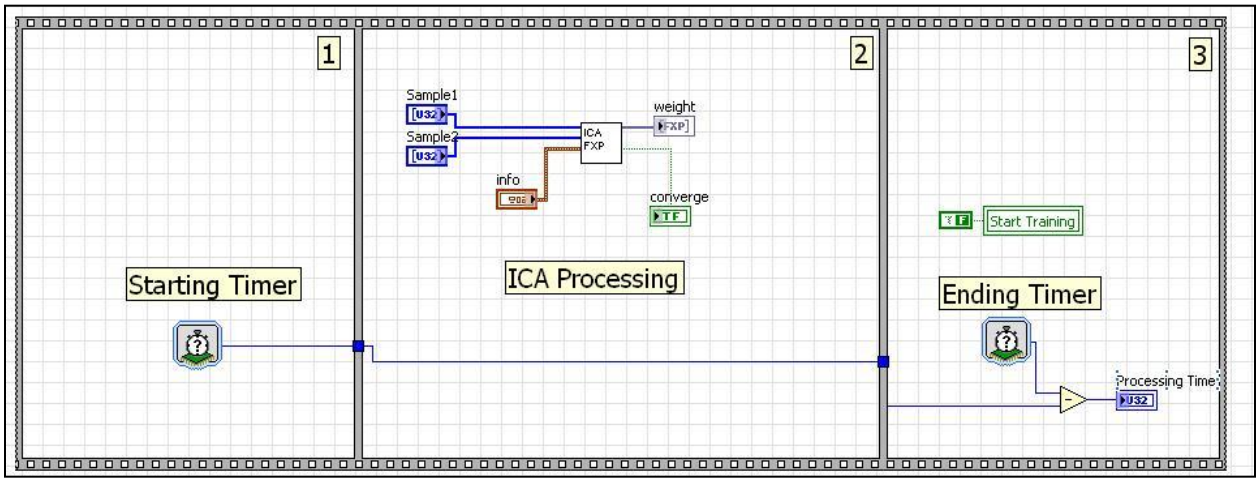

Figure 104. Processing Time Recording (LabVIEW FPGA code Segment)

The algorithm converges in 2 iterations for all BLF situations, which corresponds to 22μs real time reported from the FPGA timer. After the ICA algorithm converges to the separating matrix, 32 fixed point 2-point dot product calculations (2 for each data bit in the 16-bit RN16, with one for the first half bit and the other for the second half) need to be completed to decode the recovered data ("0" features the difference of the two halves, while "1" includes the same halves.). The 32 multiplications are realized by 32 parallel on-chip multiplication and accumulation units, and consume 1μs as reported by the FPGA timer. The FPGA on chip resource utilization summary for 2-tag ICA resolution showing that the design takes 53% of the total logic slices is listed in **Table 11**. The processing speed for collisions at typical BLFs is compared with the standard specification as shown in Figure 105. Because the 22μs training time is online real time along with the collision signal, the actual data recovery only requires the 1μs dot production calculation after receiving the second half point in the last bit of the collision signal. As shown, the FPGA resolves the collision successfully within the real time specification.

Table 11. FPGA Resources Utilization

```
Compilation Summary
-------------------
Device Utilization Summary:
  Number of BUFGMUXs                    7 out of 16     43%
  Number of External IOBs             403 out of 556    72%
  Number of LOCed IOBs                403 out of 403   100%
  Number of MULT18X18s                 48 out of 136    35%
  Number of RAMB16s                    24 out of 136    17%
  Number of SLICEs                   7262 out of 13696  53%

Clock Rates: (Requested rates are adjusted for jitter and accuracy)
  Base clock: Configuration_Clk
    Requested Rate:      20.000000MHz
    Theoretical Maximum: 53.433075MHz
  Base clock: RTSI_Ref_Clk
    Not used
  Base clock: ADC_0_Port_A_Clk
    Requested Rate:      25.001250MHz
    Theoretical Maximum: 43.884671MHz
  Base clock: DAC_0_IQ_Clk
    Not used
```

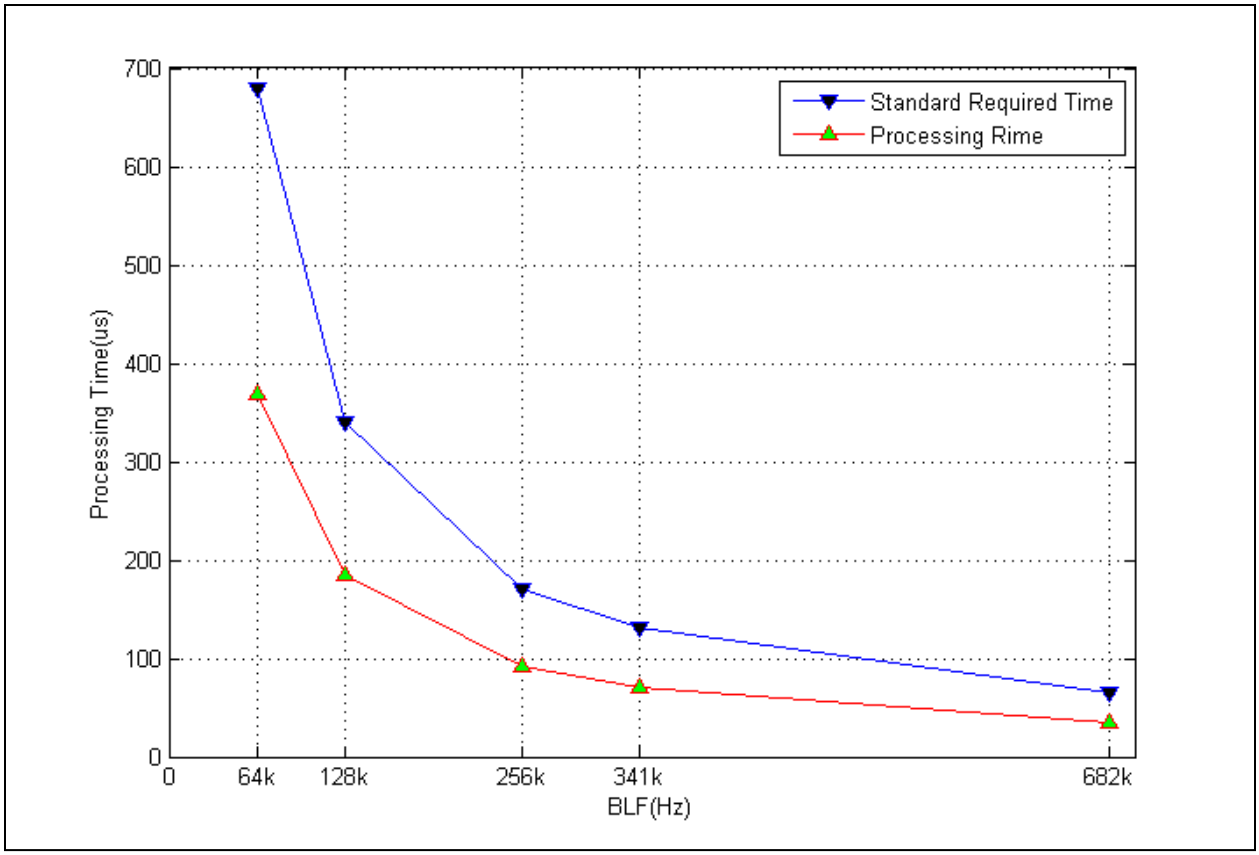

Figure 105. 2-tag Collision Resolution Processing Time vs. Standard Specification

7.4.2 3-Tag Collision Resolution

As an extension to multiple tag (more than 2) collision resolution, the 3-tag collision signal resolution in FPGA is implemented as well. The three collision signals from each of the receiving channels are artificially synthesized by mixing a real 2-tag collision and a real individual tag response. Figure 106~110 show the resolution result on real tag collision signal at typical BLFs.

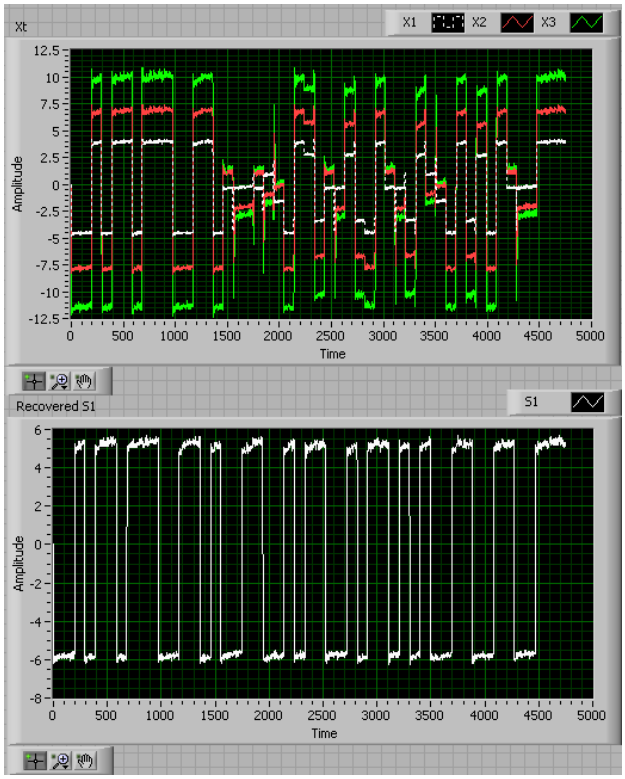

Figure 106. 3-tag Collision Resolution Result using FastICA in FPGA (BLF=64 kHz)

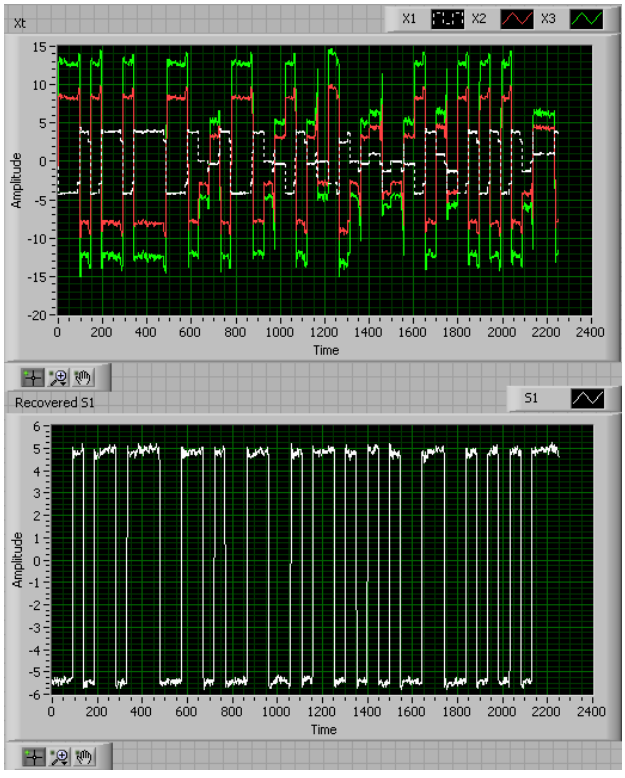

Figure 107. 3-tag Collision Resolution Result using FastICA in FPGA (BLF=128 kHz)

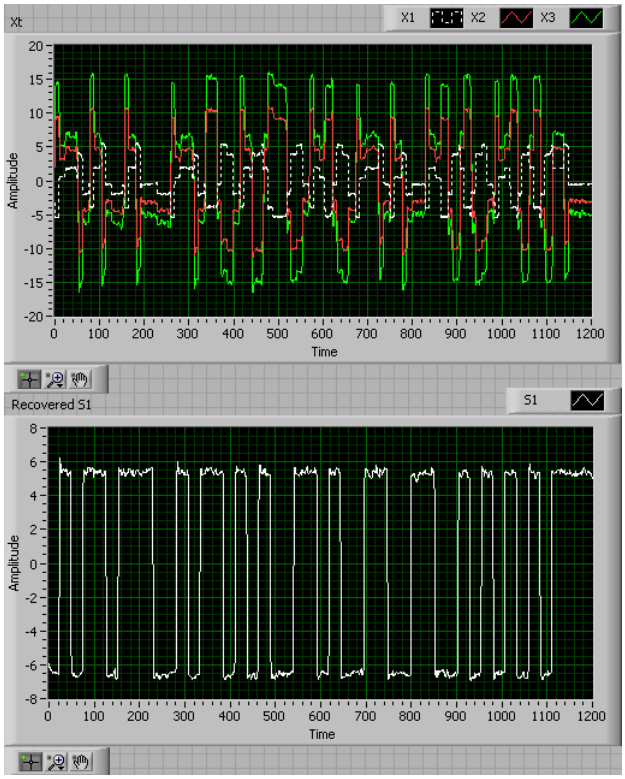

Figure 108. 3-tag Collision Resolution Result using FastICA in FPGA (BLF=256 kHz)

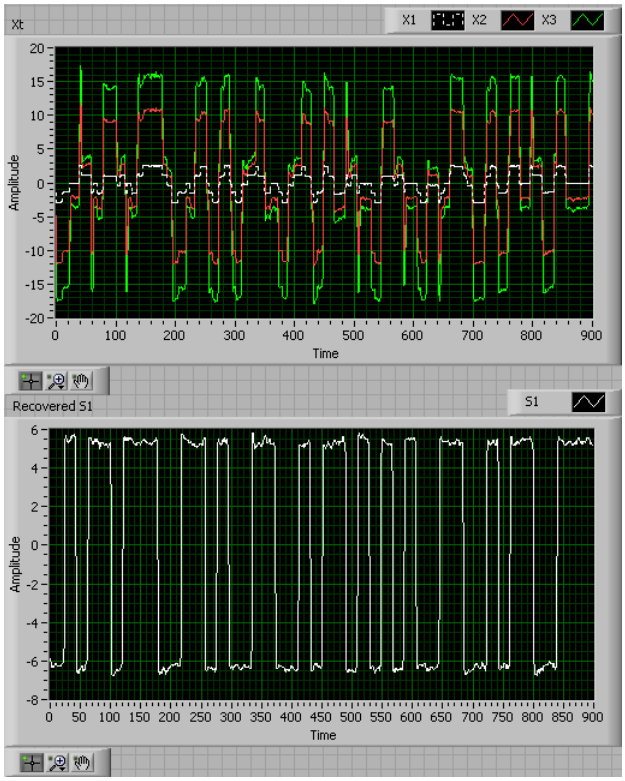

Figure 109. 3-tag Collision Resolution Result using FastICA in FPGA (BLF=341 kHz)

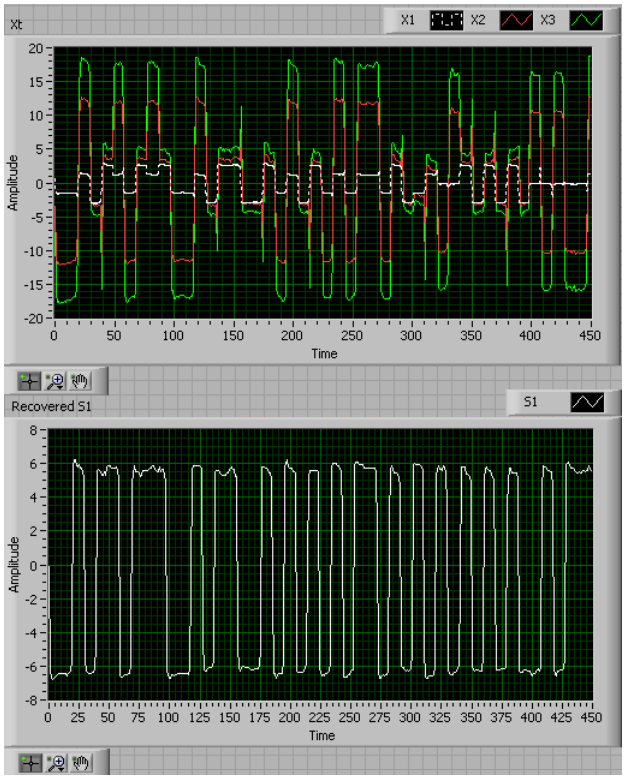

Figure 110. 3-tag Collision Resolution Result using FastICA in FPGA (BLF=682 kHz)

The FPGA on chip resources utilization summary is listed in Table 12 for 3-tag ICA resolution. The total FPGA slice usage significantly increases from 53% for 2-tag design to 89% in the 3-tag design.

Table 12. FPGA Resources Utilization

```
Compilation Summary
--------------------
Device Utilization Summary:
  Number of BUFGMUXs                    7 out of 16     43%
  Number of External IOBs             403 out of 556    72%
  Number of LOCed IOBs                403 out of 403   100%
  Number of MULT18X18s                 56 out of 136    41%
  Number of RAMB16s                    32 out of 136    23%
  Number of SLICEs                  12211 out of 13696  89%

Clock Rates: (Requested rates are adjusted for jitter and accuracy)
  Base clock: Configuration_Clk
     Requested Rate:      20.000000MHz
     Theoretical Maximum: 44.698731MHz
  Base clock: RTSI_Ref_Clk
     Not used
  Base clock: ADC_0_Port_A_Clk
     Requested Rate:      25.001250MHz
     Theoretical Maximum: 37.707391MHz
  Base clock: DAC_0_IQ_Clk
     Not used
```

The algorithm converges in 7 iterations corresponding to 126μs real time. The data bit recovery takes 32 fixed point 3-point dot product calculations consuming 1.28μs. The Processing speed for collisions at typical BLFs is compared with the standard specification as shown in Figure 111.

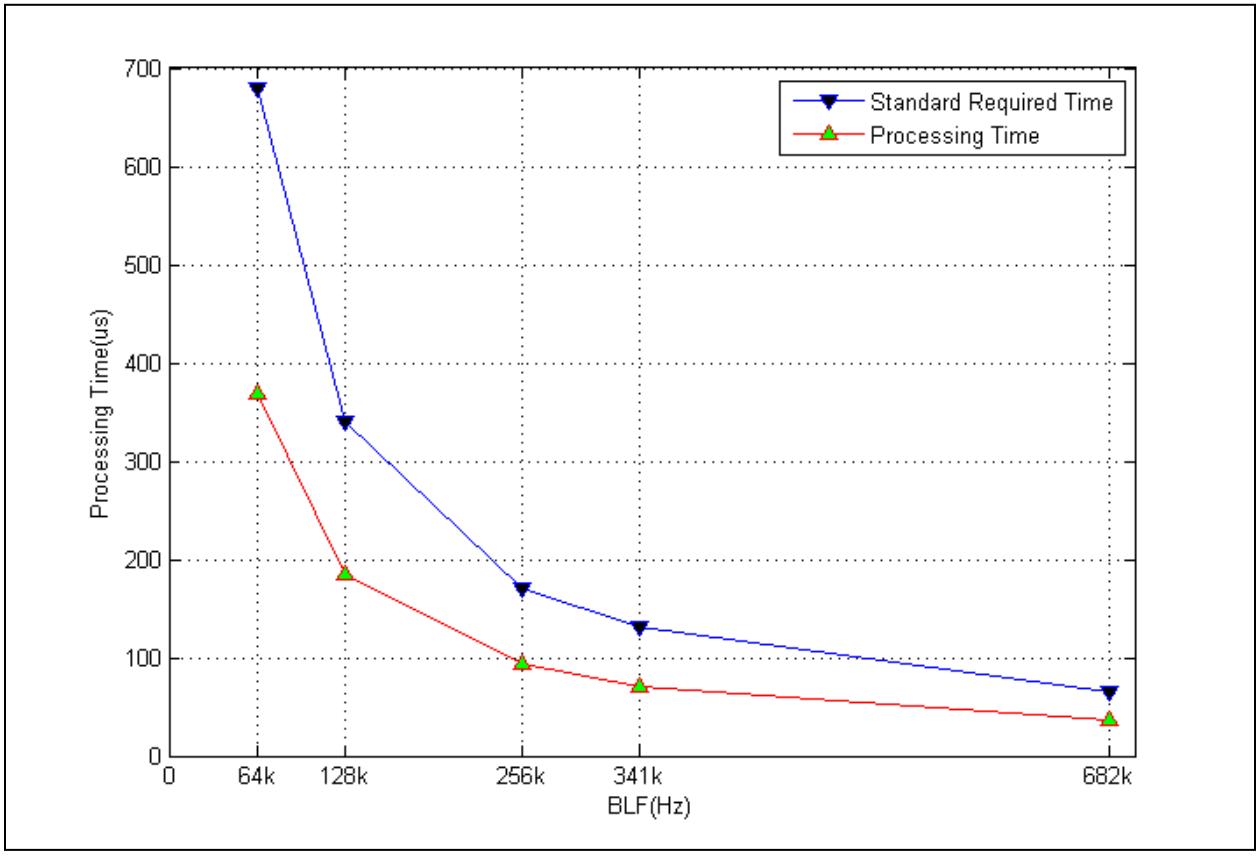

Figure 111. 3-tag Collision Resolution Processing Time vs. Standard Specification

8.0 DATA WHITENING

As a preprocessing step in the FastICA algorithm as described in Table 10 (Section 7.0), the whitening of the data samples can significantly simplify the ICA computation and thus accelerate the convergence [9].This is because after whitening, the original mixing matrix is transformed to a new mixing matrix, which is orthogonal. Therefore, this restricts the ICA algorithm's search for the separating matrix to only orthogonal matrix. The whitening process is as follows:

1. Calculate the covariance matrix of the training sample.
2. Calculate the eigenvalues and eigenvectors of the covariance matrix.
3. Form the whitening matrix as $V=D^{-1/2}E^{T}$, where D is a diagonal matrix with all the eigenvalues of the covariance matrix on the diagonal, and E is the matrix consisting of all the eigenvectors corresponding to the eigenvalues in D.
4. Multiply the training data with the whitening matrix.

The proof of the whitening process is in Appendix A. Obviously, the whitening requires the Eigen decomposition of the covariance matrix of the training sample (Step 1 and Step 2). According to the experimental results, the 2-tag collision resolution effect is satisfactory even without data whitening because of the simplicity of the RFID signal formation. However, that is not true for 3-tag collision situation. The FastICA algorithm cannot converge after large number of iterations without whitening preprocessing on the training sample.

In the current design, this problem is circumvented by using a universal whitening matrix instead of calculating the instant whitening matrix at every reading. The eigenvalues and eigenvectors of the covariance matrix of the collision signal represent the characteristics of the mixing matrix, thus if the mixing matrix is approximately fixed each time and the tags are from the same manufacturer, the covariance matrix of the acquired collision signals from receiving channels are statistically stable. Therefore, the experiment can be carried out by fixing the position of the reader and the three tags each time, and measuring a group of covariance matrices of the data in order to calculate the expectation. This expectation matrix is then decomposed for the eigenvalues and eigenvectors, and used thereafter to calculate the whitening matrix as a universal whitening matrix. The resulting universal whitening matrix can be used to whiten the following collision signals from target tags at the fixed positions. This process is equivalent to a calibration before ICA processing. This strategy applies to the application scenario in which the relative position of the tags and the reader are fixed, and the tags are all from the same manufacturer as shown in Figure 112. As shown, the reading process consists of two steps:

1. Select a batch of 3-tag groups to calibrate for the universal whitening matrix
2. Whiten successive actual tags placed at the fixed slots using the universal whitening matrix

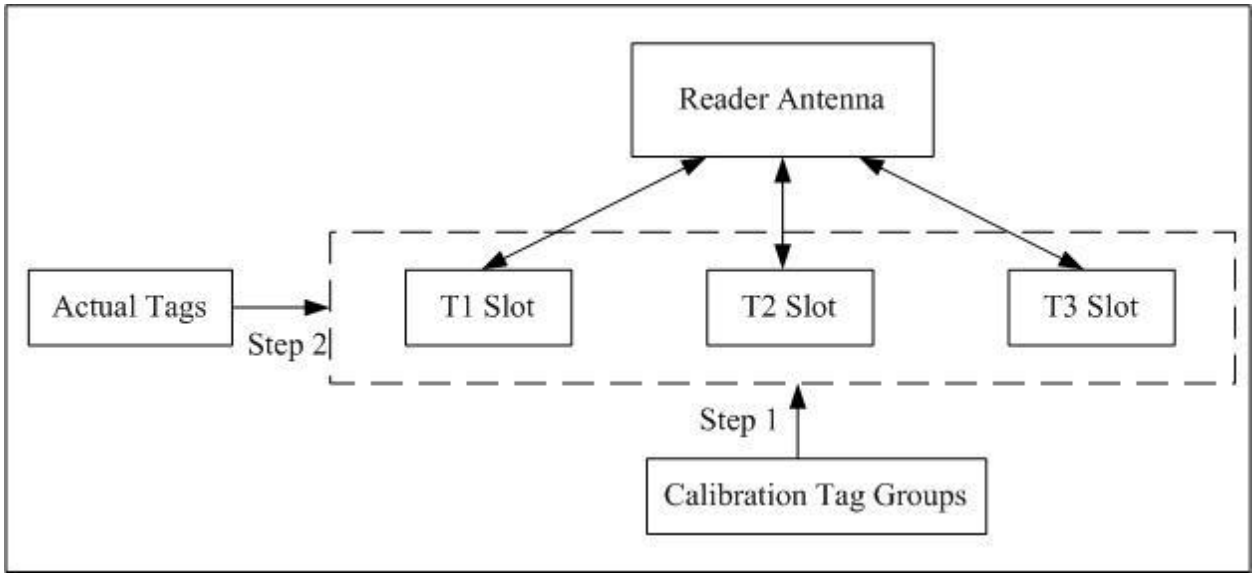

Figure 112. Calibration for Universal Whitening Matrix

To illustrate the effectiveness of this approach, 8 groups of three tags are placed within the range of the reader, and each time the mixing matrix is fixed as shown in Figure 113 corresponding to the fixed position of the placement of the tags relative to the reader. For each group, the whitening matrix is calculated by the PC, and the expectation of the 8 obtained whitening matrices is used for the universal whitening matrix as shown in Figure 114.

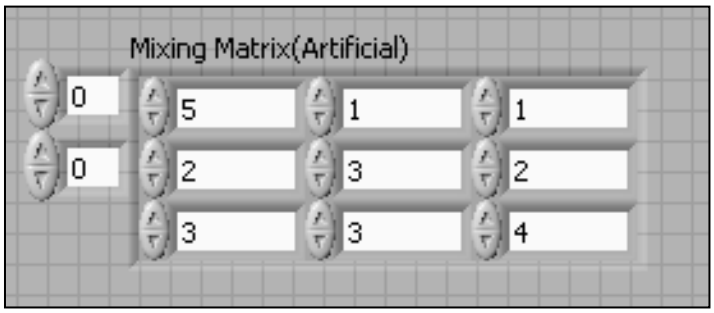

Figure 113. Artificial Mixing Matrix

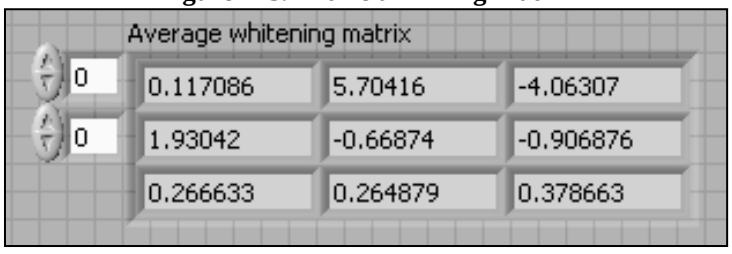

Figure 114. Estimation of the Universal Whitening Matrix

The ICA resolution results for each individual group of the 3-tag collision based on the whitened data generated by the universal whitening matrix is as following:

Group 1: It can be seen, the ICA resolution result, as shown in Figure 115, for this group is not satisfactory after 7 iterations. This is because the non-diagonal components in the whitened signal covariance matrix, as shown in Figure 116, are still significant, which means the correlation between the three collision signals is not effectively removed.

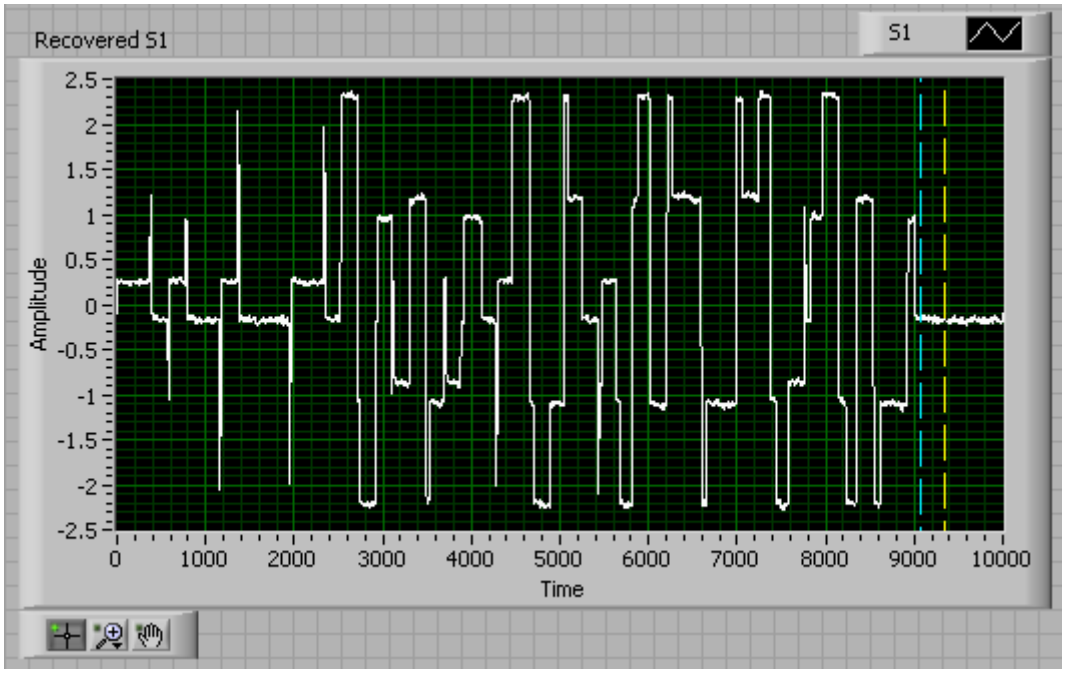

Figure 115. Resolution Result Based on Universal Whitening Matrix (Group 1)

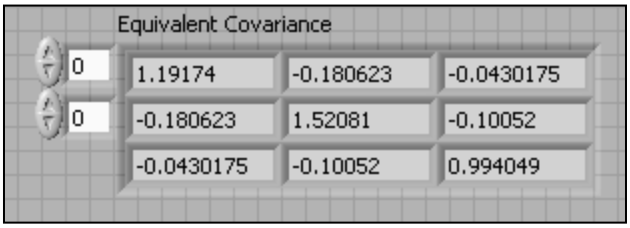

Figure 116. Covariance Matrix of the Whitened Collision Signal (Group 1)

Group 2: The ICA resolution result, as shown in Figure 117, for this group is satisfactory after 7 iterations. This is because the non-diagonal components in the whitened signal covariance matrix,

as shown in Figure 118, are an order of magnitude less than the diagonal components, which means the decorrelation of the universal matrix on this group of data is effective.

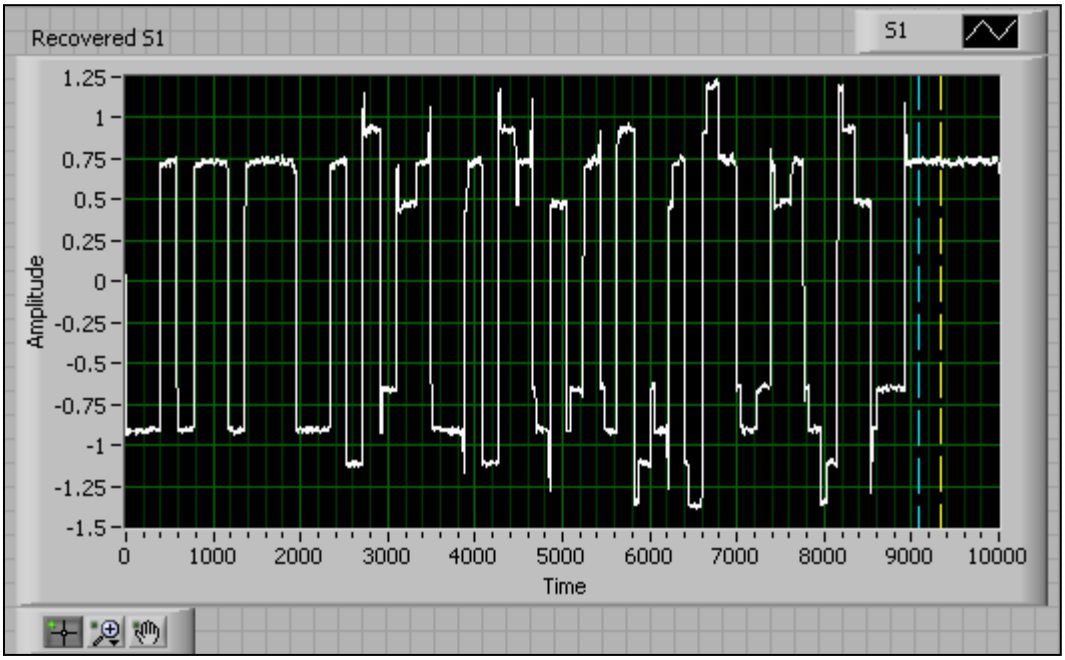

Figure 117. Resolution Result Based on Universal Whitening Matrix (Group 2)

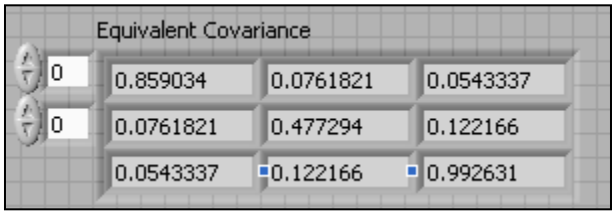

Figure 118. Covariance Matrix of the Whitened Collision Signal (Group 2)

Similarly, the resolution results (Figure 119,Figure 121,Figure 123,Figure 125,Figure 127,Figure 129) for Groups 3~8 and the corresponding whitened covariance matrix of the training samples (Figure 120,Figure 122,Figure 124,Figure 126,Figure 128) are shown as follows:

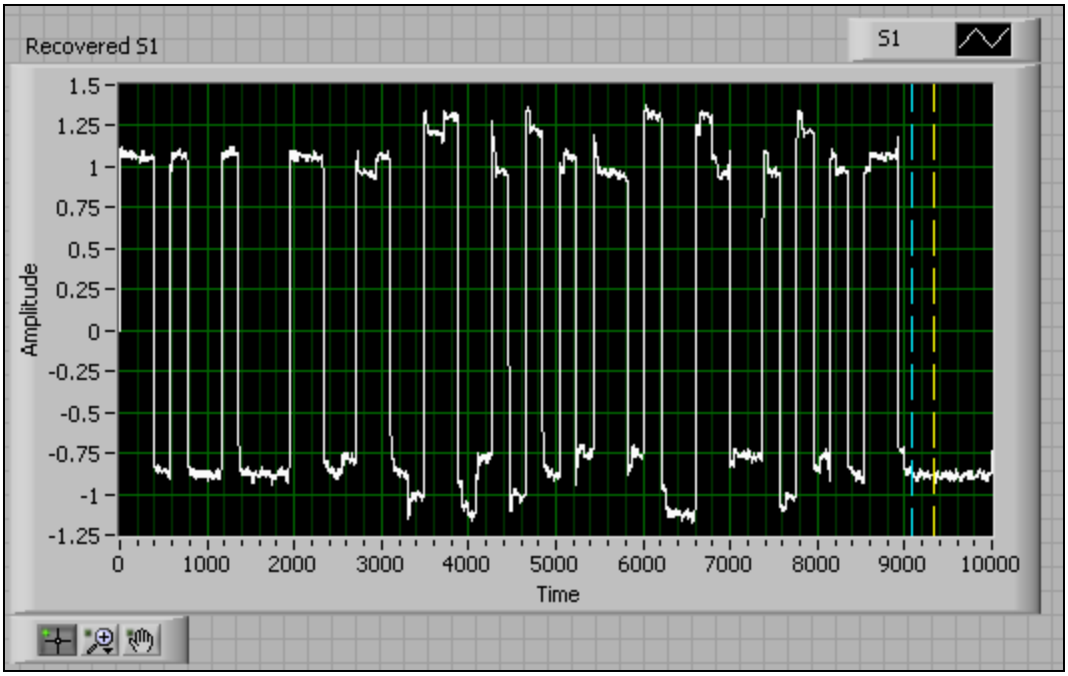

Figure 119. Resolution Result Based on Universal Whitening Matrix (Group 3)

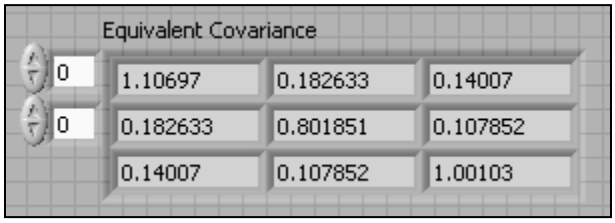

Figure 120. Covariance Matrix of the Whitened Collision Signal (Group 3)

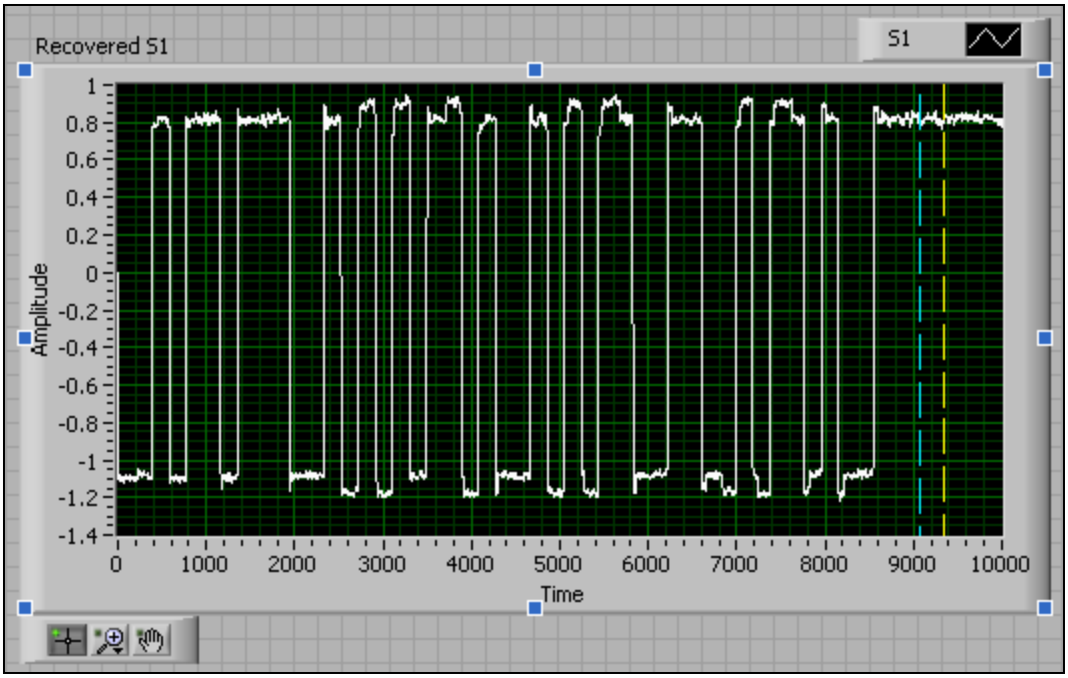

Figure 121. Resolution Result Based on Universal Whitening Matrix (Group 4)

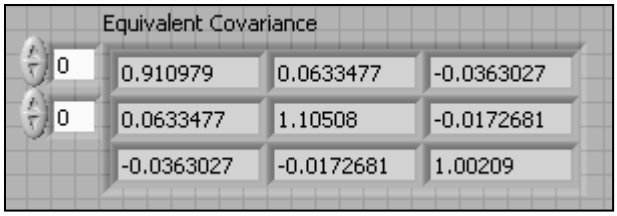

Figure 122. Covariance Matrix of the Whitened Collision Signal (Group 4)

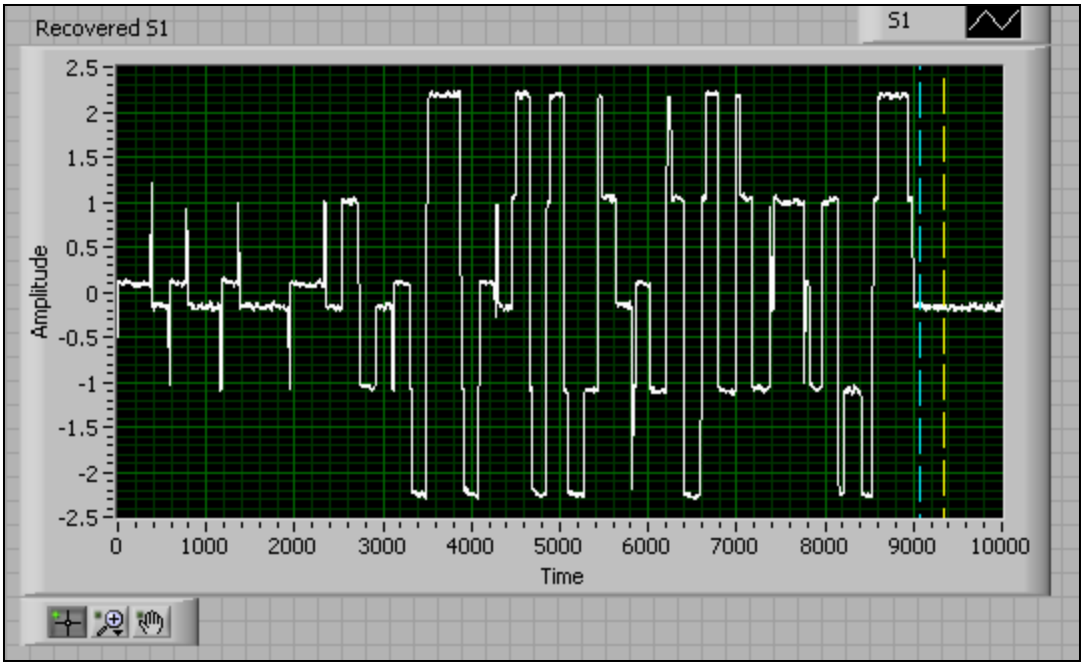

Figure 123. Resolution Result Based on Universal Whitening Matrix (Group 5)

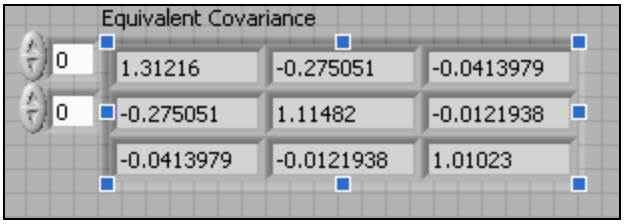

Figure 124. Covariance Matrix of the Whitened Collision Signal (Group 5)

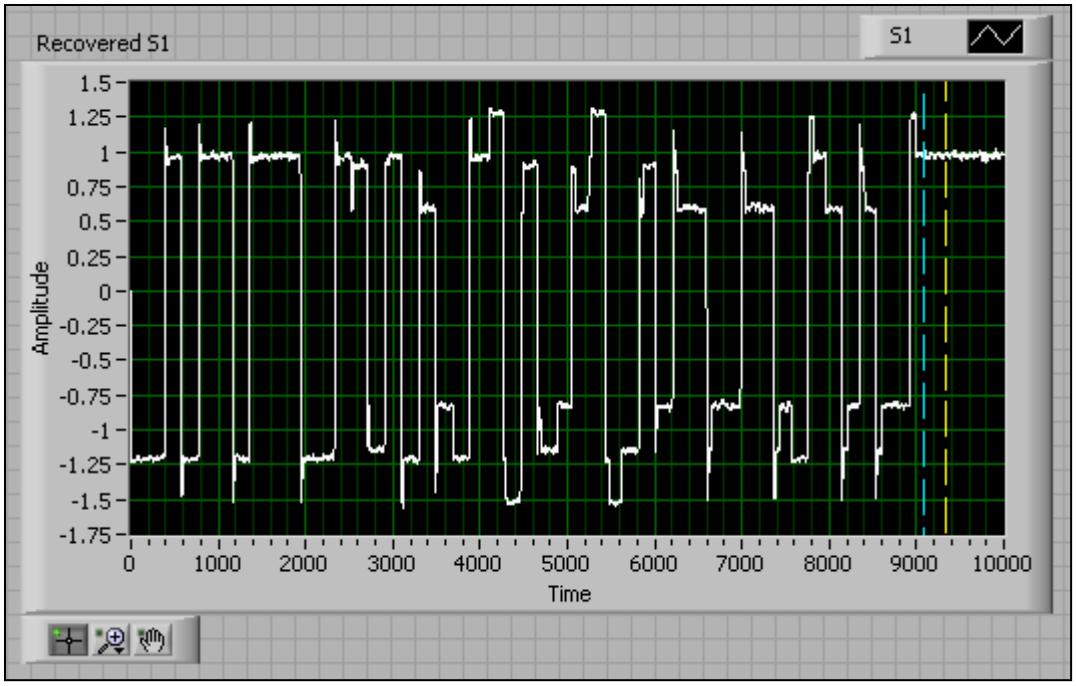

Figure 125. Resolution Result Based on Universal Whitening Matrix (Group 6)

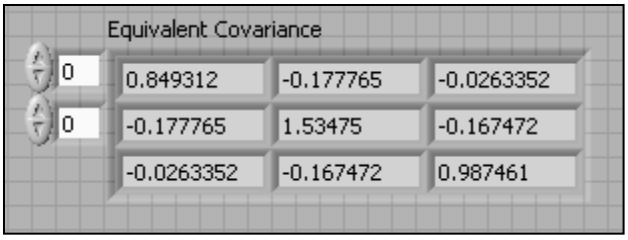

Figure 126. Covariance Matrix of the Whitened Collision Signal (Group 6)

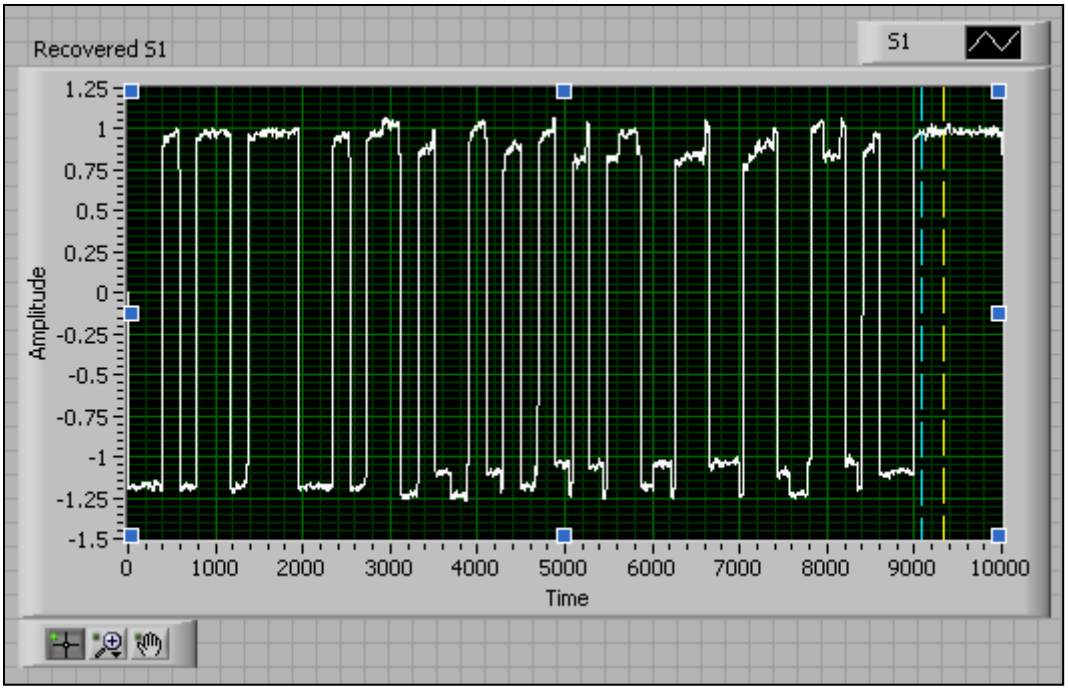

Figure 127. Resolution Result Based on Universal Whitening Matrix (Group 7)

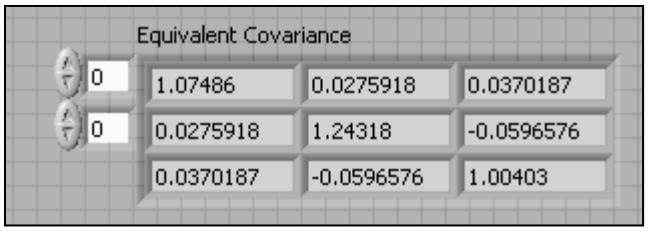

Figure 128. Covariance Matrix of the Whitened Collision Signal (Group 7)

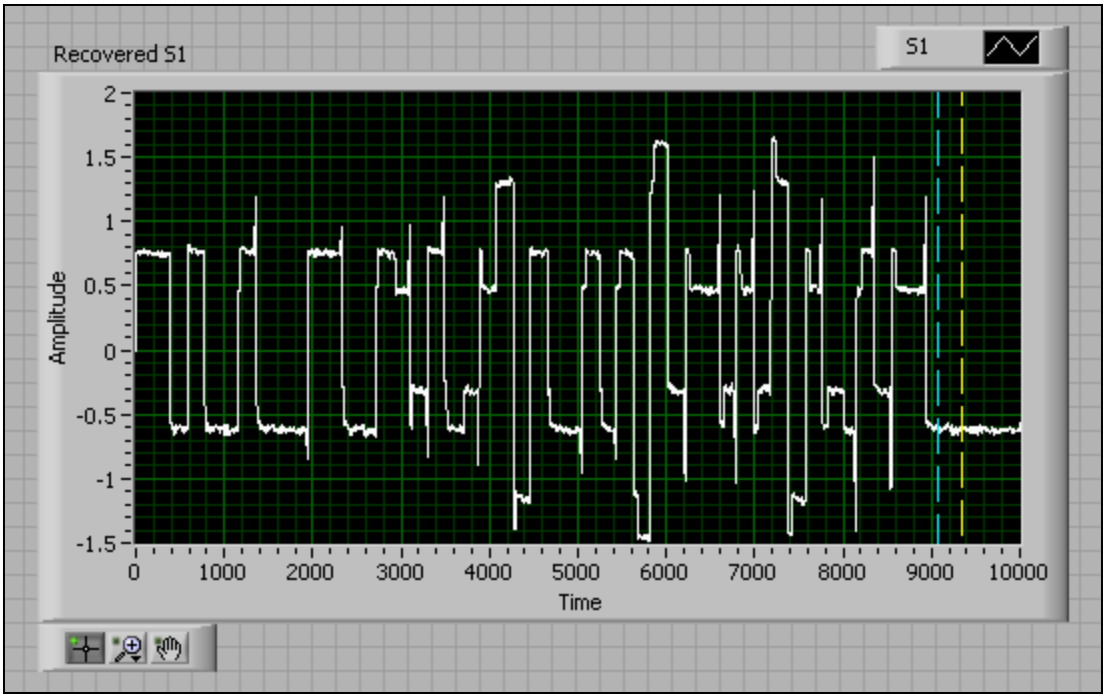

Figure 129. Resolution Result Based on Universal Whitening Matrix (Group 8)

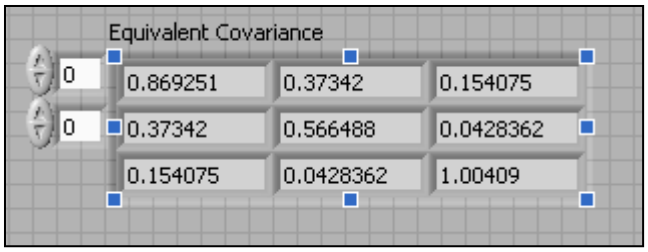

Figure 130. Covariance Matrix of the Whitened Collision Signal (Group 8)

As mentioned above, the universal whitening matrix approach only applies to the scenario when the tag positions are fixed relatively to the reader. To resolve 3-tag collisions in cases when this fixed position prerequisite is not available, the instant whitening matrix of the three acquired collision signals needs to be calculated at each reading in real time. This requires realizing high efficiency matrix decomposition algorithm/scheme such as Singular Value Decomposition (SVD) and L-U factorization for the signal covariance matrix in hardware. The

algorithm proposed in [12] and [13] can be implemented and integrated in a more advanced FPGA device as future works.

9.0 CONCLUSION

In this research, multiple collision resolution methods for ISO 18000-6C passive RFID system are developed. The characteristics of the collision signal acquired from both stationary and moving tags are analyzed, and collision situations including both 2 tags and 3 tags are studied. Two straightforward methods: the direct edge locating and the amplitude mapping are simulated and implemented using National Instruments software and hardware with NI LabVIEW and LabVIEW FPGA module. The implementation takes proper utilization of the FPGA device and successfully resolves a 2-tag collision with and without significant phase shift between tag responses within the standard specified real time. However, these two methods do not apply to multiple tag (more than 3) collision situations due to their intrinsic limitations such as amplitude ambiguity. To extend the result, a more advanced digital signal processing approach, ICA, based on the statistical characteristics of passive RFID signal is employed. As one branch of ICA, a speed optimized algorithm FastICA focused on extracting the tag information through minimizing signal entropy is adopted. A trade-off between the resolution accuracy and computation load is made to guarantee the FPGA realization efficiency and processing speed within specification. In addition, an experiment on data whitening is carried out to show its affect as a useful preprocessing step in ICA for algorithm convergence speed up. Future work can be completed to resolve each colliding tag signal in parallel and integrate instant data whitening in the design using a more advanced FPGA device.

In summary, the contributions of this research include:

- Developed a software defined radio ISO 18000-6C compatible reader using NI software and hardware for conformance test and data acquisition
- Simulated, implemented and verified two straightforward methods for 2-tag collision resolution based on the phase shift and amplitude characteristics of the collision signal
- Simulated, implemented and verified an ICA based method for 3-tag collision resolution based on the statistical characteristics of the collision signal
- Developed an experiment platform to efficiently capture data from moving tags
- Studied the characteristics of the collision signal acquired from moving tags with comparison to the stationary tag collision signal

10.0 FUTURE WORKS

While this research has covered the collision situation with 2 tags and 3 tags, future works can be carried out in the following areas including:

For 2-tag collision resolution with the edge locating and the amplitude mapping:

1. Although these two methods work for their individual application scenario(tag response with and without significant phase shift), the reader does not have this information beforehand. So problem occurs when a reader with edge locating method deployed happens to encounter a collision signal without significant phase shift. To handle this problem, the edge locating method and the amplitude mapping method can be deployed in parallel in one hardware device in parallel.

For ICA part:

1. Using a more advanced FPGA device with higher volume of logic slices (e.g. Xilinx Virtex 5 series) to perform ICA in parallel on each colliding tag response to separate the source simultaneously.

2. Deploying effective matrix decomposition algorithms in hardware to calculate the instant whitening matrix for the collision signal, and then incorporation the calculation unit in the current design with an FPGA of a higher volume.

APPENDIX A

PROOF OF DATA WHITENING PROCESS

Each component in vector X represents one collision signal acquired from the corresponding receiving channels, and X is defined as a column vector and of zero mean due to the FPGA preprocessing. The covariance matrix of X is defined as in Eq.A.1:

$$C = E\{XX^T\} \quad Eq.A.1$$

Whiteness of a zero-mean random vector (e.g. Y), means that its components are uncorrelated and their variances equal unity. The covariance matrix of Y equals the identity matrix as shown in Eq.A.2.

$$E\{YY^T\} = I \quad Eq.A.2$$

Accordingly, whitening means performing a linear transform to the collision signal X by multiplying it with the whitening matrix V to make the transformed signal Z white as shown in Eq.A.3.

$$Z = VX \quad Eq.A.3$$

The covariance matrix of X can be eigenvalues decomposed (EVD) as shown in Eq.A.4. Where E is the orthogonal matrix of eigenvectors of X's covariance matrix as shown in Eq.A.1 and D is the diagonal matrix of its eigenvalues, $D = diag(d_1, d_2, \ldots d_n)$.

$$C = E\{XX^T\} = EDE^T \quad Eq.A.4$$

In this design, the whitening matrix V is selected as shown in Eq.A.5. Where $D^{-1/2}$= $diag(d_1^{-1/2}, d_2^{-1/2}, \ldots d_n^{-1/2})$.

$$V = D^{-1/2}E^T \quad Eq.A.5$$

Therefore, the whiteness of the preprocessed signal is guaranteed as shown in Eq.A.6.

$$E\{VX(VX)^T\} = VE\{XX^T\}X^T = D^{-1/2}E^T EDE^T ED^{-1/2} = I \quad Eq.A.6$$

BIBLIOGRAPHY

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

[2] Joon Goo Lee, Seok Joong Hwang, Seon Wook Kim, "Performance Study of Anti-collision Algorithms for EPC-C1 Gen2 RFID Protocol", ICOIN2007 pp.523-532,2007

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

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

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

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

[7] "+5V-Powered, Multichannel RS-232 Drivers/receivers" , Maxim, Sunnyvale, CA, 2006. http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf

[8] Leslie Balmer, 1997, "Signals and Systems: An Introduction (2nd Edition)", by Prentice Hall

[9] A. Hyvärinen, J. Karhunen and E. Oja.,2001,"Independent Component Analysis", by John Wiley & Sons, Inc.

[10] A. Hyvärinen and E. Oja.1997, "A fast fixed-point algorithm for independent component analysis". Neural Computation, 9(7):1483-1492.

[11] John A. Rice, 1995, "Mathematical Statistics and Data Analysis (Second edition)", by Duxrury Press

[12] Joann M. Paul, M.H.Mickle. "Three-Dimensional Computational Pipelining with Minimal Latency and Maximum Throughput for L-U Factorization", IEEE Transactions on Circuits and Systems, Vol. 45, No. 11, November 1998, pp. 1465-1475.

[13] Christophe Bobda, Klaus Danne, André Linarth, “Efficient Implementation of the SingularValue Decomposition on a Reconfigurable System”, 13th International Conference on Field Programmable Logic and Applications 2003: pp.1123-1126