DESIGN OF LOW-COST HIGH-ACCURACY MICROCONTROLLER-BASED RESOLVER EMULATOR

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Pongpachara Limpisathian

December, 2013

DESIGN OF LOW-COST HIGH-ACCURACY MICROCONTROLLER-BASED RESOLVER EMULATOR

Pongpachara Limpisathian

Thesis

Approved:

_____	_____
Co-Advisor	Department Chair
Dr. Joan E. Carletta	Dr. Alex De Abreu-Garcia
_____	_____
Co-Advisor	Dean of the College
Dr. Kye-Shin Lee	Dr. George K. Haritos
_____	_____
Committee Member	Dean of the Graduate School
Dr. Robert Veillette	Dr. George R. Newkome
	_____
	Date

ABSTRACT

This thesis presents an architecture and analysis of a resolver emulator using a microcontroller and a multiplying digital-to-analog converter. The proposed resolver emulator architecture is designed to achieve output accuracy similar to commercially available resolver emulators but at a lower cost and complexity. A microcontroller is used to compute and store the sine and cosine representations of the digital input positions and a multiplying digital-to-analog converter is used to modulate the reference carrier with the sine and cosine representations into corresponding resolver emulator outputs. The pertinent design parameters are the number of digital input position bits, the number of multiplier bits, and the rate at which updates are communicated to the multiplying digital-to-analog converter. Simulation results show an improvement of nearly 20 arcminutes in the total error is observed as the number of multiplier bits is increased from 8 to 14 bits when the number of digital input bits is 16. The update rate should be at least 360 times larger than the rate of shaft rotation. An experimental test bench is constructed to gain further insight into the non-idealities of the resolver emulator. The results show that the implemented resolver emulator with 16 digital input position bits, 14 multiplier bits, and an update rate of 16 kHz realized an average error magnitude of approximately 32 arcminutes. The packaging and design of the analog backend can be improved to eliminate noise and allow an output accuracy of within 10 arcminutes.

ACKNOWLEDGEMENTS

I would like to thank my parents, Saritpong and Patcharee, for the supports they have given me over the years; none of this would have been possible without their supports. I also would like to thank Uea-issara Thanatwaranon for always supporting and cheering me up when things get tough. I sincerely thank my advisors and my committee member, Dr Joan Carletta, Dr Kye-Shin Lee, and Dr Robert Veillette, for their invaluable guidance and support throughout the years. I could not possibly express enough gratitude to them for everything that they have done for me. Special thanks to Mrs Gay Boden and the Department of Electrical and Computer Engineering. Lastly, I would like to thank all of my friends here at the University of Akron including Joseph Davis, Kripesh Bhattarai, Jian Liu, Purushottam Parajuli, Shivasai Bethi, Michelle Liu, Mike Willett, Shilpa Chakinala, and Sneha Bhattaram.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Motivation

Many electromechanical systems require motion control of a rotary motor. Position feedback systems involving rotating machinery typically include a feedback position sensor, a motor controller, and a motor, as shown in Figure 1.1. The most important characteristic for position feedback systems is the accuracy with which the position can be controlled. The feedback position sensor and the motor controller are regarded as the main units in the feedback position system. In many cases, a protocol converter is required to convert the signal produced by the feedback position sensor into the format expected by the motor controller. There are many types of protocol converter that are commonly used in industry; the protocol converter is necessary because feedback position sensors and motor controllers may produce and receive position information using a wide variety of different signal formats and communications protocols.

Some feedback position sensors, like synchros and resolvers, convey position via analog signals. Others convey position via digital signals communicated on a serial bus; example communications protocols used for position include Synchronous Serial Interface (SSI) [1], Bidirectional Synchronous Serial Interface (BiSS) [2], Encoder Data (EnDat) [3], and High Performance Interface (Hiperface) [4]. Similarly, the motor

controller may expect position in an analog format, or in a digital format according to any of these protocols.

A conversion from one digital protocol to another digital protocol is relatively easy. The accuracy of the digital positions can be preserved as long as the one protocol has sufficient accuracy for another protocol. However, conversion from an analog signal format to a digital format, or vice versa, is much more difficult. Accuracy can be lost during the conversion due to errors inherent in the required analog electronics, such as quantization error and offset and gain error.

This thesis looks specifically at the resolver emulator, a device that converts digital inputs into corresponding angle representations in resolver emulator sine and cosine format. The cost of a resolver emulator, or digital-to-resolver converter, is proportional to the accuracy, can run into thousands of dollars [5], making the cost of the whole position feedback system high. Therefore, low-cost high-accuracy resolver emulators are desirable for control system designers.

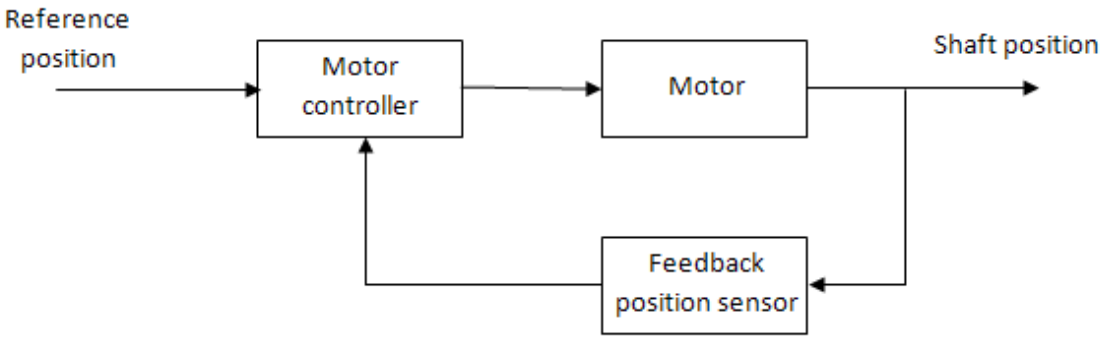


Figure 1.1: Block diagram of a position feedback system

1.2 Goal and contribution of thesis

The goal of the thesis is to explore the design of a low-cost, high-accuracy resolver emulator, and to understand how various design parameters affect the accuracy of the resulting position feedback. Using the analysis developed, good design decisions can be made based on desired accuracy. In addition, various resolver emulator architectures are evaluated at the system component level, and the advantages and disadvantages of the architectures are analyzed. Based on a predetermined set of specification requirements, the most suitable architecture is selected. These requirements include output accuracy, input resolution, complexity, and cost, and are discussed further in Chapter III.

A resolver emulator design using the selected architecture gives full consideration of how pertinent design parameters influence the accuracy of the resolver emulator's output. The selected architecture has three key parts, a microcontroller, a multiplier, and a digital-to-analog converter (DAC). As a result, the design parameters of the selected architecture include the number of bits for the input position, the number of bits for the multiplier, and the update rate of the digital-to-analog converter. Different sets of design parameter values are simulated to observe the specific effects of each component on the resolver. The accuracy of the simulated outputs of the resolver emulator for each set of design parameter values is analyzed to determine the significance of each design parameter in terms of error. Based on the analysis, the minimum requirements for the design parameters to achieve output accuracy equivalent to commercially available resolver emulators are shown. Finally, the design is implemented on an experimental test

bench after obtaining satisfactory results from the simulations. Various tests are performed, with various sets of parameter values, to compare experimental results with the simulation results. Based on the results, values for the design parameters are recommended to meet the initial specifications. The result is a complete design of the circuitry and programming for a low-cost resolver emulator with high output accuracy. The simulated results show an output error less than one arcminute can be achieved using ideal components, while the bench results show a maximum output error of approximately 32 arcminutes with non-ideal components.

1.3 Organization of thesis

The thesis is organized as follows. Chapter II introduces background information on the fundamental operating principles of feedback position sensors, with the emphasis on resolvers. It also presents prior work on resolver emulators and resolver-to-digital converters, and discusses applications for resolvers and resolver emulators. Chapter III presents and compares various architectures for the resolver emulator. Chapter IV analyzes and shows the simulations of the proposed resolver emulator architecture, with full consideration of pertinent design parameters. Chapter V presents experimental results obtained after implementing the proposed resolver emulator architecture on an experimental test bench. Finally, Chapter VI draws conclusions and makes general recommendations, as well as provides scope for future work.

CHAPTER II

RESOLVER AND RESOLVER EMULATOR BACKGROUND

There are many types of feedback position sensors, such as incremental encoders, absolute encoders, resolvers, and synchros. Each type uses a different method for sensing position. In this chapter, the operating principles of common types of feedback position sensors are described. The resolver, a position feedback sensor providing position in an analog format, is emphasized. Finally, resolver-to-digital converters and resolver emulators are presented, along with their applications.

2.1 Feedback position sensors

The primary function of a feedback position sensor is to transform physical positions into electrical signals. Feedback position sensors may provide either digital or analog outputs, in a variety of forms. Digital feedback position sensors are collectively called encoders, whereas their analog counterparts are called resolvers or synchros. There are two main categories of digital feedback position sensors in existing technology: incremental encoders and absolute encoders. The operation of incremental encoders and absolute encoders are explained in Section 2.1.1 and Section 2.1.2, respectively. Finally, the operation of resolvers and synchros is discussed in detail in Section 2.1.3 and Section 2.1.4, respectively.

2.1.1 Incremental encoders

An incremental encoder is an electromechanical feedback device that provides pulses that indicate incremental changes in position of a rotating motor shaft. As the shaft rotates, pulses are formed on three analog output signals, A, B, and Z; typical pulses are shown in Figure 2.1. The first two pulses, A and B, also called quadrature pulses, are 90° out of phase. The third pulse, Z, represents a home position of the shaft; a pulse is produced once per revolution each time the shaft passes the home position. The speed is determined by counting the number of pulses on A and B in a known amount of time. The total pulse count, since the home position was last encountered, is used to find the current position of the shaft. The direction of the shaft rotation can also be determined, by looking at the phase relationship of the A and B pulses.

An incremental encoder may operate on either optical or magnetic principles. Optical incremental encoders use light as a sensing mechanism to identify the position. Optical incremental encoders consist of three main types of component: light-emitting diodes (LEDs), photodetectors, and a code disc that lies between each diode and photodetector pair. A separate LED and photodetector may be needed for each output signal produced. Therefore, a total of three LEDs and three photodetectors may be required to produce the pulses for the incremental encoder.

The accuracy with which position can be determined using the pulses from an incremental encoder depends on the design of the code disc. Typical code discs have from 100 to 6000 sectors, and so can provide 3.6° to 0.06° of resolution [6]. Figure 2.2 shows a simplified code disc pattern for an incremental encoder; the disc is divided into

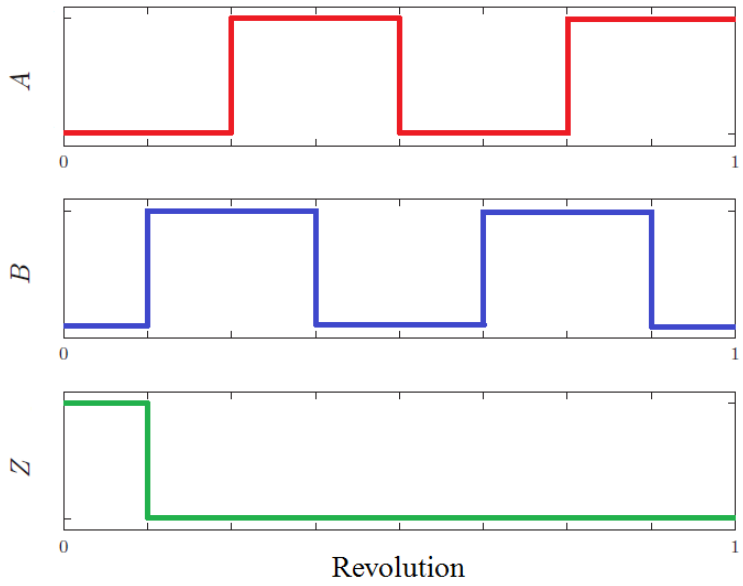
eight sectors, so that as the shaft rotates, the code disc blocks and allows passage of the light from the three LEDs simultaneously, so as to produce two A and B pulses per revolution. Using eight sectors, the position can be resolved to within 45 degrees.

A magnetic incremental encoder is similar to an optical incremental encoder, but uses magneto-resistive sensors, conditioning circuits, and a magnetically coded disc. As the shaft rotates, the code disc causes sinusoidal changes in magnetic field that are converted to electrical signals. Subsequently, the signals are conditioned using the conditioning circuit. In general, magnetic encoders are more robust to dust, moisture, temperature, and vibration than optical encoders. This makes magnetic encoders suited for rugged environments.

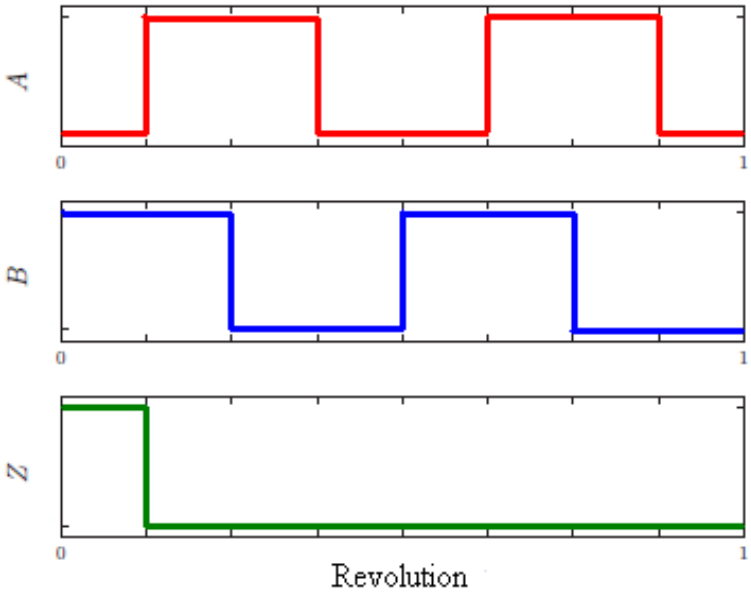
Incremental encoders are presently the most popular feedback position sensors in the industry, mainly due to their moderate robustness and the simplicity of having only three output signals [7]. There are numerous devices that accept incremental encoder inputs, ranging from motor controllers to microcontrollers and computer interface cards. The circuit and software designs for incremental encoders are relatively simple compared to those for other types of feedback position sensors.

Incremental encoders also have two distinct disadvantages. The shaft position needs to be calibrated to the home position initially upon powering up. In the event of a power outage, position cannot be determined from an incremental encoder's outputs until the home position has been encountered. Thus, incremental encoders sense the position indirectly; the history of the pulses is needed to determine the position [8]. The second

disadvantage of the incremental encoder is the potential for false counts due to electrical interference [9].



(a) Clockwise rotation of the shaft



(b) Counterclockwise rotation of the shaft

Figure 2.1: Quadrature pulses (A, B) and home pulse (Z) of an incremental encoder with eight sectors **[8]**

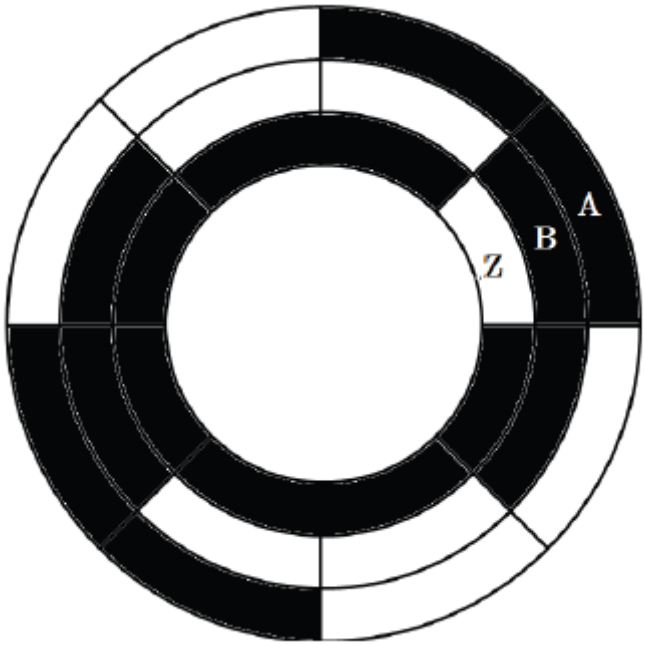


Figure 2.2: Incremental encoder disc pattern with eight sectors **[8]**

2.1.2 Absolute encoders

An absolute encoder provides a direct measurement of position. As was the case for incremental encoders, absolute encoders may operate according to either optical or magnetic principles. Optical absolute encoders are comprised of light-emitting diodes, photodetectors, and a fixed binary pattern disc with n segments and 2^n sectors [10]; Figure 2.3 shows an example code disc with ten segments and 1024 sectors. The ten segments are the concentric circles, and the sectors divide the disc radially into 1024 parts. Each sector on the disk represents an absolute position; with 1024 sectors, position can be measured to the nearest 0.35 degrees. An LED transmitter-photodetector pair is used to monitor each segment. Light is either blocked or allowed to pass through the disc's pattern to be detected by the photodetectors. Bit streams are generated from the

photodetectors, where the MSB is the output of the photodetector monitoring the innermost segment and the LSB is the output of the photodetector monitoring the outermost segment. A magnetic absolute encoder works by having an array of sensors, usually Hall-effect sensors or magneto-resistive sensors, that detect the magnetic field of the rotating magnetically coded disc with respect to the axis.

The main advantage of absolute encoders over incremental encoders is the absolute precision of the output due to the nature of construction of the disk. Absolute encoders, unlike incremental encoders, are able to provide a direct measure of the position of a rotating shaft, without needing any history. The output of an absolute encoder is a digital code indicating the current instantaneous position; this code changes directly with the physical rotation of the motor shaft. As a consequence, absolute encoders also have the ability to recover immediately following a power outage. A disadvantage of an absolute encoder is the inability to tolerate harsh environment conditions. Both optical and magnetic absolute encoders rely on the precision of the digital output from the sensors to derive the current absolute position; as a result, noise in the digital outputs due to shock and vibration can affect the accuracy of the encoder [9].

Figure 2.3: Absolute encoder disc pattern with 1024 sectors **[11]**

2.1.3 Resolvers

Resolvers are analog electromechanical devices for measuring the speed and position of a rotating shaft. Essentially, a resolver is a rotary transformer that works by exploiting the sinusoidal relationship between the shaft angle and output voltage. There are several types of resolver; the most common type is the brushless transmitter resolver.

A brushless transmitter resolver is similar to a small electrical motor in that its main components are a rotor and stators. A primary winding, often called the reference winding, is located within the rotor and rotates along with the shaft. The primary winding is excited with a high energy sinusoidal reference carrier at a fixed frequency typically between 400 Hz and 10 kHz [7]. On the stator side, two secondary windings, often called sine and cosine windings, are positioned inside the stator such that they are 90° from each

other, as shown in Figure 2.4. As a result, two voltages are induced in the two secondary windings, equal to the reference carrier modulated by the sine and cosine of the shaft angle, respectively. The reference carrier, excited between R2 and R4, is

$$r(t) = A\ \sin(\omega_1 t), \tag{2.1}$$

where A is the amplitude of the reference carrier and ω_1 is the angular frequency of the reference carrier. In order to convey the angular information, the reference carrier frequency, ω_1, must be greater than the shaft rotational frequency [9]. The two induced voltages on the sine output and cosine outputs can be represented by

$$V_s(\theta, t) = r(t)\sin(\theta) = A\ \sin(\omega_1 t)\sin(\theta), \tag{2.2}$$

and

$$V_c(\theta, t) = r(t)\cos(\theta) = A\ \sin(\omega_1 t)\cos(\theta), \tag{2.3}$$

where θ is the angle of the shaft, V_s is the differential voltage measured between S1 and S3, and V_c is the differential voltage measured between S2 and S4.

The angular position of the shaft can be derived from the sine and cosine outputs of the resolver with respect to the reference carrier. The relationship between the angle of the shaft and the resolver outputs is

$$\frac{V_S(\theta)}{V_C(\theta)} = \frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta). \tag{2.4}$$

Thus

$$\theta' = \tan^{-1}\left(\frac{V_s(\theta)}{V_c(\theta)}\right), \tag{2.5}$$

where θ' indicates the shaft angle. Note that the ratio of the resolver outputs is independent of the frequency and the amplitude of the reference carrier [12]. Because the arctangent function returns angles from −90° to 90°, while the shaft angle can be between 0 and 360 degrees, additional information is needed to resolve the quadrant of the shaft angle. The quadrant can be determined by considering the polarity of the reference carrier and the resolver outputs. Table 2.1 shows the quadrant selection process used to reconstruct the shaft angle [8].

Figure 2.5 shows how the single-ended resolver outputs change with shaft angle. The first waveform shows the reference carrier. The second and third waveforms show the sine and cosine outputs when the shaft turns at a certain speed about 1/32 the frequency of the reference carrier. The single-ended resolver outputs, V_s and V_c, contain two frequency components, the reference carrier frequency and the rotating shaft frequency. Note that the reference carrier frequency corresponds to the frequency inside the envelope and the shaft rotational frequency corresponds to the frequency of the envelope of the resolver signals.

Resolvers are more robust and durable than encoders mainly due to their construction. Unlike encoders, resolvers do not contain any optical or magnetic electronics, and so require no precision alignment; this makes them more robust in the presence of dust, oil, smoke, vibration, and shock. Resolvers can also operate at higher temperatures than encoders [7]. Therefore, resolvers are commonly used in

environmentally demanding applications. Resolvers are used in a wide range of applications including motor control systems, wind turbines, robotics, aerospace systems, and military systems [13].

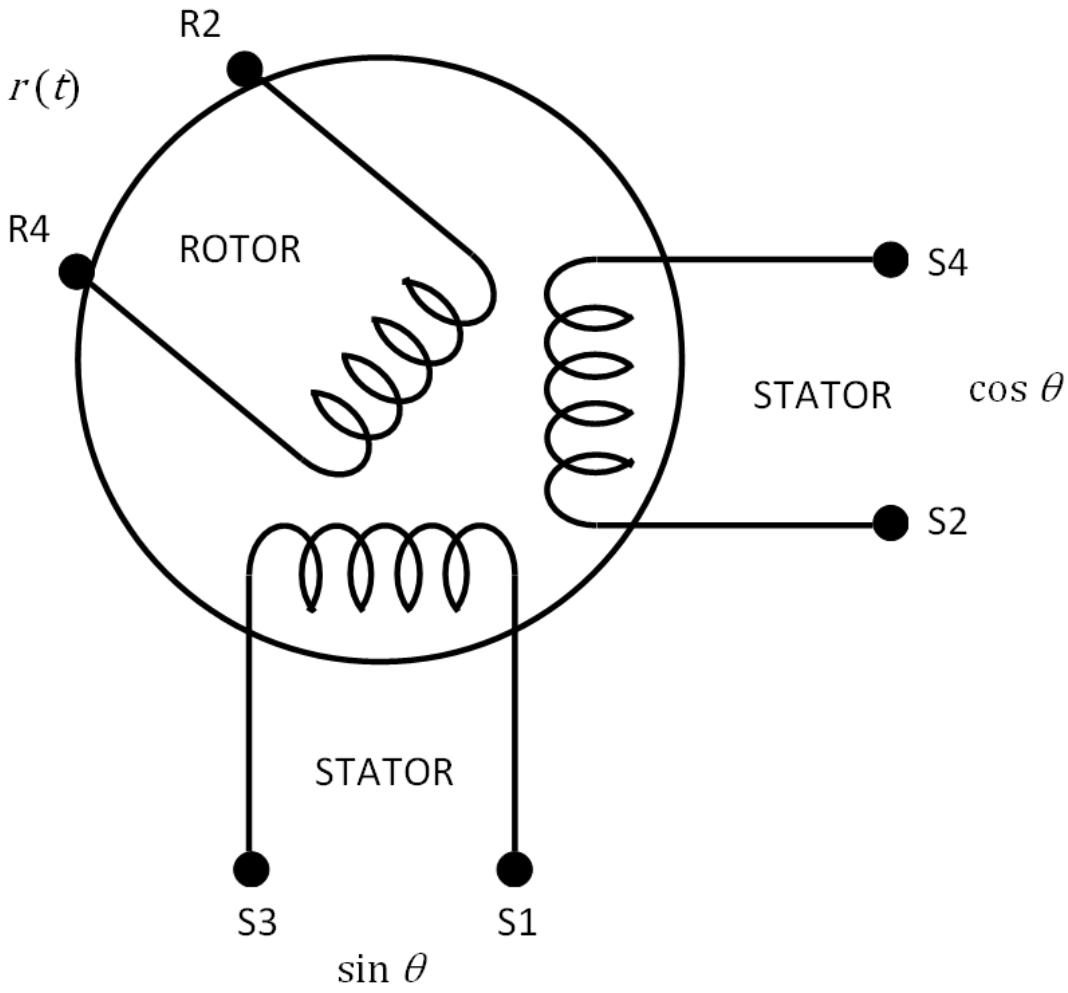


Figure 2.4: Rotor and stators placement of a resolver

Table 2.1: Quadrant selector of resolver outputs

Quadrant	If the sign of the reference carrier and the resolver sine output are:	And the sign of the reference carrier and the resolver cosine output are:	Then
1st	The same	The same	θ = θ'
2nd	The same	Different	θ = θ' + 180
3rd	Different	Different	θ = θ' + 180
4th	Different	The same	θ = θ' + 360

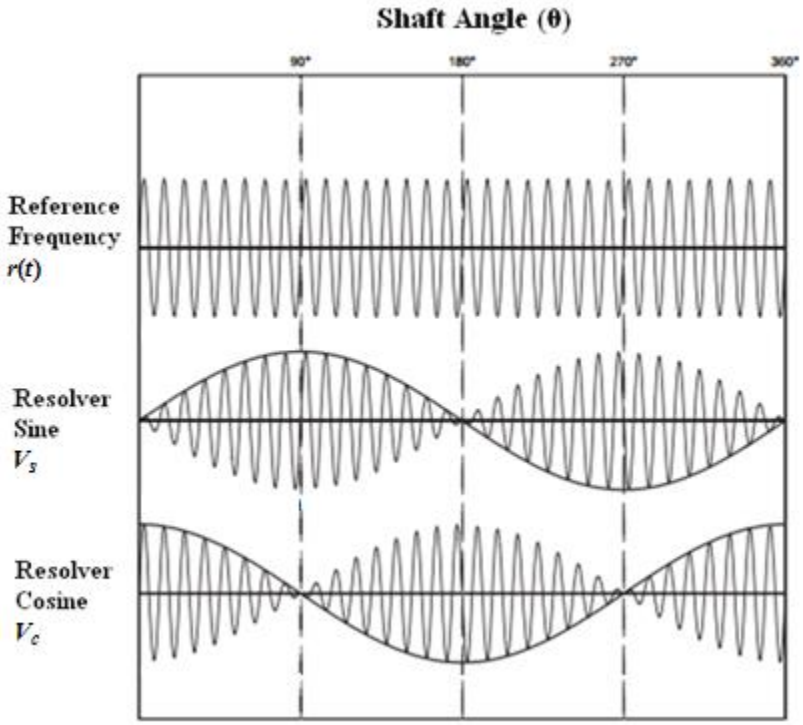


Figure 2.5: Single-ended outputs of resolver as a function of the shaft angle **[13]**

2.1.4 Synchros

A synchro operates according to the same basic principle as a resolver. However, instead of two secondary windings, a synchro has three secondary windings that are positioned 120° apart from each other, as shown in Figure 2.6. The rotor is excited by a reference carrier typically at 60 Hz or 400 Hz [14] to produce induced voltages at the stator windings. The voltage induced on each winding is directly proportional to the cosine of the shaft angle between the rotor coil axis and the coil axis of that winding.

Operation of the synchro is illustrated with an example. Assuming that the reference carrier is again as given in (2.1), the induced voltages across the stator terminals S1 to S3, S3 to S2, and S2 to S1 are, respectively, represented by

$$V_x(\theta, t) = r(t)\sin(\theta) = A\ \sin(\omega_1 t)\sin(\theta), \tag{2.6}$$

$$V_y(\theta, t) = r(t)\sin(\theta + 120) = A\ \sin(\omega_1 t)\sin(\theta + 120), \tag{2.7}$$

and

$$V_z(\theta, t) = r(t)\sin(\theta + 240) = A\ \sin(\omega_1 t)\sin(\theta + 240), \tag{2.8}$$

where θ is the angle of the shaft, and ω_1 is the angular frequency of the reference carrier. The relationships between the shaft angle and synchro outputs are

$$\frac{V_x(\theta)}{V_y(\theta)} = \frac{\sin(\theta)}{\sin(\theta+120)}, \tag{2.9}$$

$$\frac{V_y(\theta)}{V_z(\theta)} = \frac{\sin(\theta+120)}{\sin(\theta+240)}, \tag{2.10}$$

and

$$\frac{V_z(\theta)}{V_x(\theta)} = \frac{\sin(\theta+240)}{\sin(\theta)}. \tag{2.11}$$

Synchros share many of the same qualities as resolvers when it comes to robustness and sensitivity. Synchros are able to withstand shock, vibration, and extreme temperatures due to their construction. Because synchros do not contain any optical or magnetic electronics, no precision alignment is required. Synchros are used for many of the same applications as resolvers; these include motor control systems, wind turbines, robotics, aerospace systems, and military systems [13].

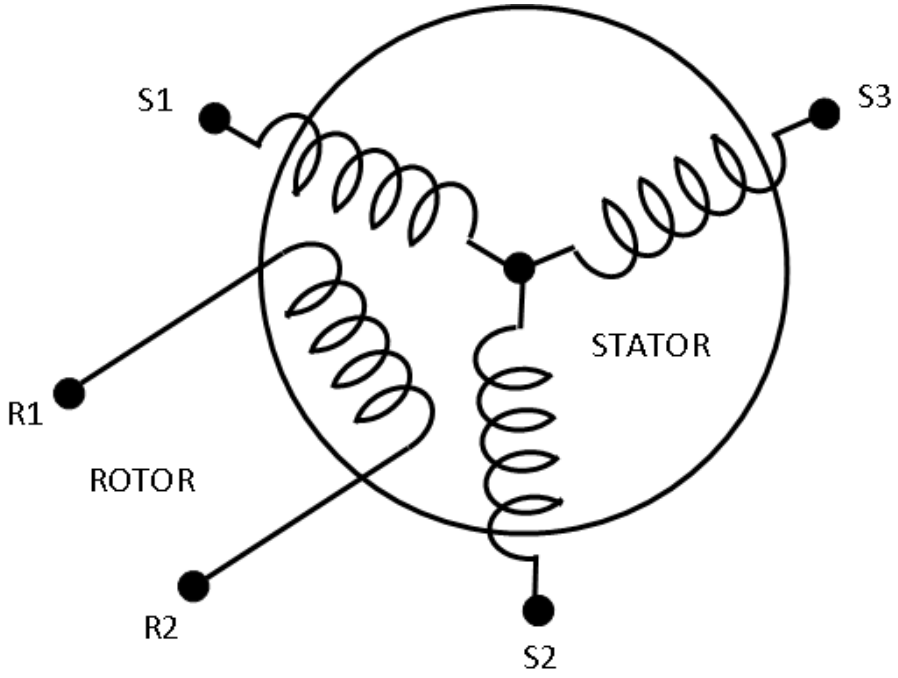


Figure 2.6: Rotor and stator placement of a synchro

2.2 Resolver conversions

Many position feedback systems utilize a resolver and a digital motor controller, while others may utilize an encoder and an analog motor controller. In such systems, conversions from a resolver signal format to a digital format or vice versa are needed; these conversions are handled by resolver-to-digital converters and resolver emulators, respectively, as shown in Figure 2.7. Resolver-to-digital converters are discussed in Section 2.2.1 and resolver emulators are discussed in Section 2.2.2.

2.2.1 Resolver-to-digital converters

There are distinct advantages of implementing control systems digitally; digital controllers are more robust, more flexible, and less prone to noise than their analog counterparts. On the other hand, resolvers and synchros, which produce position feedback in an analog form, have advantages over position feedback sensors that produce digital outputs; resolvers and synchros are more appropriate than encoders for use in harsh environments. Consequently, accurate conversion of the analog signals produced by a resolver to a digital format becomes a key factor to achieve precision control in some position feedback systems. In such systems, a resolver-to-digital converter is employed to interpret the resolver outputs to produce a digital position. Resolver-to-digital converters, sometimes referred to as tracking resolver converters, are used to convert resolver sine and cosine outputs into digital format. The digital signals are in the form of binary word usually ranging from 10 to 18 bits [14].

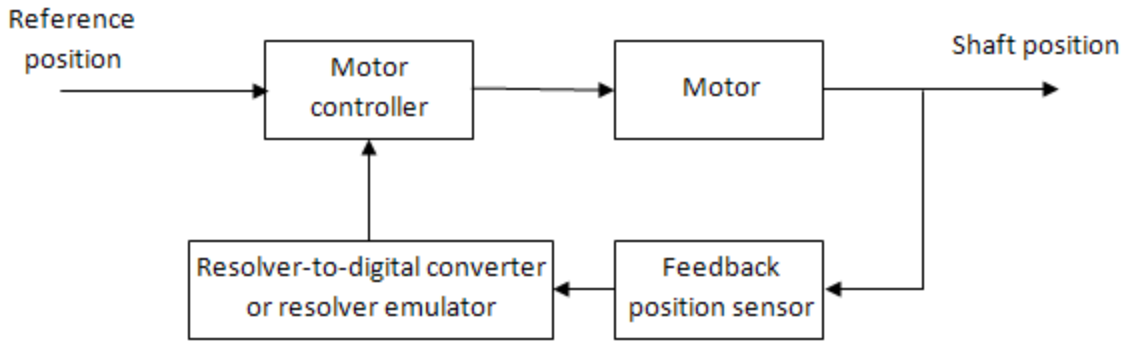


Figure 2.7: Block diagram of a position feedback system with a resolver-to-digital converter or a resolver emulator

There are many conversion methods to convert a resolver's output to a digital position representation. The most common methods are the single or double RC phase-shift methods [15]. A single or double RC phase-shift circuit compares the zero-crossing times between the differential sine and the cosine resolver outputs. A phase tracking circuit output signal is toggled whenever the difference between the differential sine and the differential cosine changes sign.

Figure 2.8 shows a block diagram of a single RC phase-shift resolver-to-digital converter. This resolver-to-digital conversion method consists of four main components; input transformer, sine and cosine multipliers, phase tracking circuit, and up-down counter [14]. At a given point in time, the up-down counter provides the current position of the shaft in a digital format; feedback is used so that when the shaft position changes, the up-down counter's value is incremented or decremented until its value once again corresponds to the position.

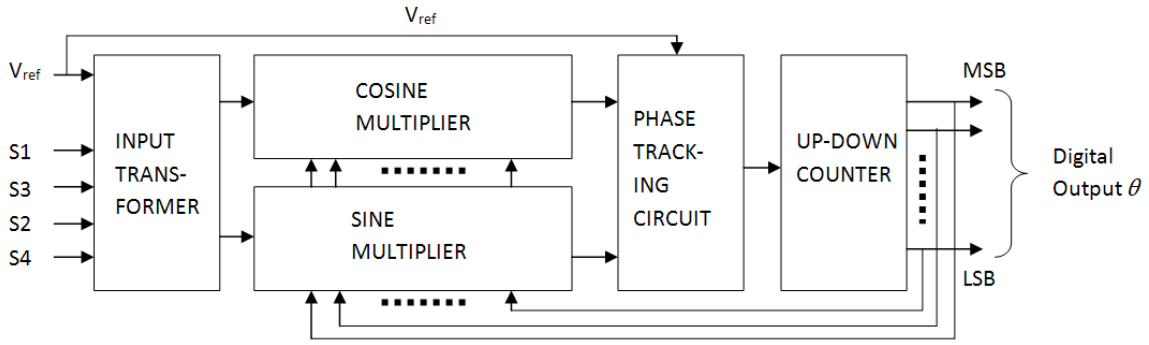


Figure 2.8: Block diagram of a single RC phase-shift resolver-to-digital converter

The input transformer is used to produce single-ended resolver sine and cosine voltages from the differential sine and cosine signals produced by the resolver. Recall from previous discussion that the differential resolver outputs are

$$V_s(\theta, t) = A\ \sin(\omega_1 t)\sin(\theta) \tag{2.2}$$

and

$$V_c(\theta, t) = A\ \sin(\omega_1 t)\cos(\theta), \tag{2.3}$$

where A is the amplitude, ω_1 is the external frequency of the reference carrier, and θ is the angle of the shaft. Assume the current content of the up-down counter corresponds to a shaft angle φ. The cosine and sine multipliers multiply the angle φ by V_s and V_c respectively to produce

$$V_{sm} = A\ \sin(\omega_1 t)\sin(\theta)\cos(\varphi), \tag{2.12}$$

and

$$V_{cm} = A\ \sin(\omega_1 t)\cos(\theta)\sin(\varphi). \tag{2.13}$$

The phase tracking circuit is used to generate a signal to increment or decrement the up-down counter so that it tracks the angle indicated by the resolver outputs. An error amplifier finds the difference between the two voltages V_{sm} and V_{cm} to get

$$V_e = A \sin \omega_1 t \, (\sin(\theta) \cos(\varphi) - \cos(\theta) \sin(\varphi)), \tag{2.14}$$

thus

$$V_e = A \sin \omega_1 t \sin(\theta - \varphi). \tag{2.15}$$

A phase sensitive detector then demodulates the error signal with the reference carrier to obtain a DC error signal proportional to $\sin(\theta - \varphi)$.

The rest of the phase tracking circuit derives pulses from the DC error signal to increment or decrement the up-down counter until the digital angle reported by the up-down counter matches the actual digital angle. Once the output digital position is properly tracking the shaft angle, the output of the phase tracking circuit is null when the angle is not changing; the up-down counter is updated only when the actual shaft angle changes by more than the angle that corresponds to the least significant bit.

2.2.2 Resolver emulators

Resolver emulators, also called digital-to-resolver converters (DRC), are devices that convert digital input position into corresponding angle representations in resolver emulator sine and cosine format. Resolver emulators are found in systems that utilize a digital encoder as a feedback position sensor [12], while requiring resolver signals to communicate with the rest of the system interface. This setup is generally found in older

position feedback systems in which a resolver has been replaced with a digital feedback position sensor. Replacing just the feedback position sensors is easier and more cost effective than redesigning the control system interface.

Resolver emulators are commonly used to drive devices such as control transformers, torque receivers, control differential transmitters, and torque differential transmitters. Many of these devices are electromechanical loads, and thus a power amplification stage is usually required at the output of the resolver emulator [14]. Some of the applications of resolver emulator include resolver simulators, flight trainers, flight instrumentation, fire control systems, IR, radar and navigation systems, motor control, and robotic systems [5].

A number of architectures can be used to implement the resolver emulator. Some of the architectures use microcontrollers and digital-to-analog converters; others rely purely on analog circuits. Regardless of the architecture used for the resolver emulator, the goal is to produce resolver emulator outputs, (2.2) and (2.3), from a digital input position code that represents the shaft angle. The advantages and disadvantages of the architectures are discussed in detail in Chapter 3.

2.3 Summary

The purpose of this chapter is to familiarize readers with feedback position sensors. Different types of feedback position sensors are presented, including incremental encoders, absolute encoders, resolvers, and synchros. The fundamental operating principles and applications of resolvers, digital-to-resolver converters, and resolver

emulators are discussed in detail. The next chapter aims to explore the architectures and implementations of the resolver emulator. The rest of the thesis focuses solely on the architecture and implementation of the resolver emulator.

CHAPTER III

ANALYSIS OF RESOLVER EMULATOR ARCHITECTURES

A resolver emulator can be implemented using several different architectures. In this chapter, important specifications of a resolver emulator are presented. Different resolver emulator architectures are explored and evaluated based on cost, complexity, and required accuracy of the components. Finally, one resolver emulator architecture is chosen for further study and implementation, based on its potential for low cost and high accuracy.

3.1 Specifications of the resolver emulator

A resolver emulator is similar to a digital-to-analog converter (DAC) in that it converts information from a digital format to an analog format, and it shares many of the specifications of a DAC; these specifications include input resolution, output accuracy, output settling time, and maximum output current [13]. Additionally, there are also specifications that are unique to the resolver emulator; these specifications include range of reference carrier frequency and amplitude that the resolver emulator is designed to accept, and the phase shift of the resolver emulator outputs with respect to the reference carrier [16]. Each of these specifications is now described in turn.

A resolver emulator takes as input a position in the form of digital bits, received by either serial or parallel communication. The resolution of the digital input position is determined by the number of bits representing the position. The number of bits representing the position for commercial resolver emulators typically ranges from eight to 16. An *n*-bit resolver emulator has 2^n unique angle representations within a single revolution, and so expresses position with a resolution of $360/2^n$ degrees; for example, a 16-bit digital input position provides 65,536 unique angle representations, and therefore the input position has a resolution of 0.0055° within a single revolution.

The range of reference carrier frequency and reference carrier amplitude accepted are another two specifications of a resolver emulator. Reference carrier frequency and reference carrier amplitude vary depending on the system interfaces for the resolver emulator. Most commercial resolver emulators accept reference carriers ranging in frequency from 400 Hz to 10 kHz and ranging in peak-to-peak amplitude from 3.5 V to 26 V [13].

The output accuracy of the resolver emulator measures how closely the resolver emulator's sine and cosine outputs represent the digital input position [12]. Output accuracy is the most important characteristic of a resolver emulator. The output accuracy is measured by comparing the angular position of the resolver emulator outputs to the actual position of the shaft. Resolver emulator errors are then calculated from these differences, and typically measured in arcminutes or arcseconds.

Phase shift is the phase difference between the reference carrier and the resolver emulator outputs. An ideal resolver would have a phase shift of zero; in practice resolvers

may have significant phase shift. Similarly, resolver emulators may produce resolver emulator signals with a slight phase shift because of delays in the circuitry. Most resolver interfacing systems can tolerate phase shift of up to 45° [13]. Because resolvers generally introduce a noticeable phase shift, and resolver emulators are designed to behave similarly, some commercial resolver emulators provide a programmable means of introducing an intentional desired phase shift.

Output settling time is the time that is required for the resolver emulator outputs to change and enter a new steady-state condition after a change in the digital input position. Depending on the architecture of the resolver emulator, the output settling time may include the time required for the resolver emulator output to respond to the input change and the time required for the resolver emulator outputs to slew and settle to the final value. Typically, the output settling time for commercial resolver emulators ranges from 1 μs to 20 μs [17].

The maximum output current specifies the ability of the resolver emulator to drive the load presented by devices with which it interfaces. A maximum output current of 2 mA rms is sufficient to drive common system interfaces, including resolver-to-digital converters, solid-state control transformers, display modules, and motor controller interface cards [5]. If additional driving capability is required, a power amplifier may be used with the resolver emulator to increase its ability to provide a current to drive a load.

Table 3.1: Specifications for the resolver emulator

Parameter	Specification	Note
Resolution of digital input position	8-bit	Minimum
Reference carrier frequency	2 – 10 kHz	
Reference carrier amplitude	2.5 – 5 V	
Output accuracy	±10 arcminutes	Within the actual shaft position
Phase shift	1°	Maximum phase shift introduced by the hardware
Output settling time	20 μs	Maximum
Maximum output current	2 mA	Root mean square

3.2 Architectures considered for the resolver emulator

There are several architectures for implementing a resolver emulator. Both analog and digital circuitry can be used to generate resolver emulator outputs. Each architecture has its own advantages and disadvantages and which architecture is most appropriate largely depends on the specifications, which are derived from the application for which the resolver emulator will be used. In this thesis, design specifications similar to commercial resolver emulators [5] [17] [18] are assumed; these specifications are shown in Table 3.1. The resolver emulator is assumed to operate in a common drive system such as a motor drive, which typically requires an output current of no more than 2 mA. Several architectures are considered based on these requirements for the resolver emulator. Finally, a particular resolver emulator architecture is proposed for further study.

3.2.1 Weighted resistor-ratio network with amplitude modulation

The first architecture for a resolver emulator is based on amplitude modulation of the reference carrier. This architecture makes use of weighted resistor-ratio networks to create analog voltages that correspond to the sine and cosine of the angle indicated by the digital input position. An example of a four-bit weighted resistor-ratio network is shown in Figure 3.1. The weighted resistor-ratio network works as a resistive divider in such a way that the analog output voltage produced is proportional to the sine or cosine of the digital input position.

A resolver emulator architecture using weighted resistor-ratio networks is shown in Figure 3.2. When the digital input position has a large number of bits, it is impractical to use the input position as direct input to a weighted resistor-ratio network, as that network would be large. Instead, the bits of the digital input position are divided into coarse and fine bits, called the *i*-bits and the *j*-bits, respectively. The coarse bits contain the most significant half of the digital input position bits, while the fine bits contain the least significant half of the bits. The corresponding angles for the *i*-bits and the *j*-bits are represented by θ_i and θ_j, respectively, and the shaft angle θ is the sum of θ_i and θ_j.

Essentially, the sine and cosine outputs of the resolver emulator are generated by composing signals corresponding to sines and cosines of the angles θ_i and θ_j, using the trigonometric identities

$$V_s = r(t)\sin(\theta_i + \theta_j) = r(t)[\sin\theta_i \cos\theta_j + \cos\theta_i \sin\theta_j], \tag{3.1}$$

and

$$V_c = r(t)\cos(\theta_i + \theta_j) = r(t)[\cos\theta_i \cos\theta_j - \sin\theta_i \sin\theta_j], \quad (3.2)$$

where $r(t)$ is the reference carrier. The four analog voltages coming from the weighted resistor-ratio networks are cross-multiplied and summed using discrete components to form analog voltages corresponding to the sine and cosine of the angle represented by the full digital input position. These sine and cosine voltages are then passed through single-ended to differential converters and modulated with the reference carrier to produce the desired resolver emulator outputs.

An advantage of this architecture is that the multiplying and summing operations can be realized easily with a four-quadrant analog multiplier such as the Analog Devices AD633 [19]. Thus, only readily available discrete components are used in this architecture; resistors, an analog multiplier, a differential operational amplifier, and multipliers are needed.

Three of the disadvantages of this architecture are the need for many resistors with a wide range of values to produce the sine and cosine signals, the low precision of the resolver emulator outputs due to device mismatches of discrete resistors, and high settling time, especially when a large resistor network is used. Therefore, this architecture is only suited for applications with a low input resolution. Another disadvantage of this architecture is the high cost of components; the architecture requires eight analog multipliers, and each analog multiplier costs around ten dollars [19].

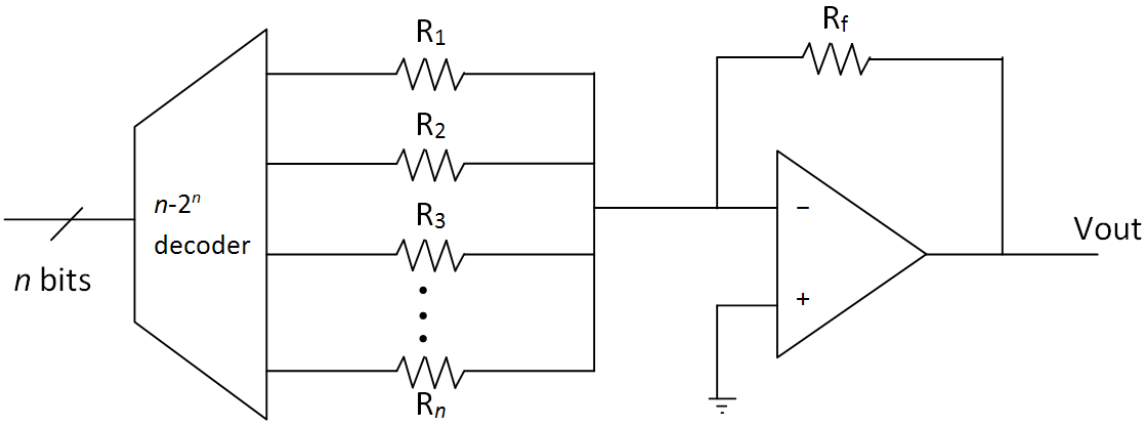


Figure 3.1: An example of an *n*-bit weighted resistor-ratio network **[20]**

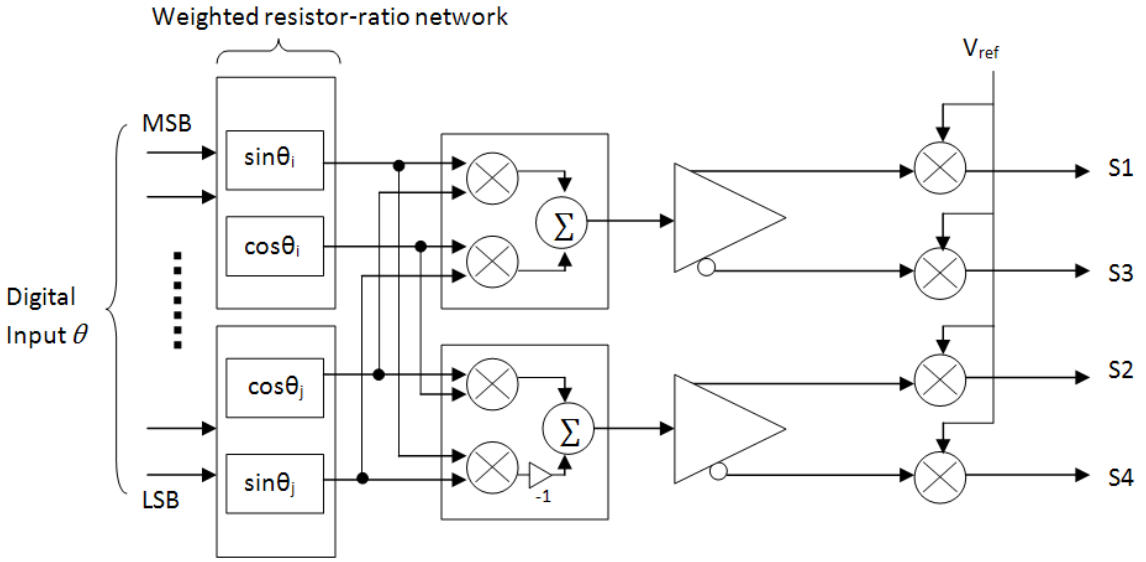


Figure 3.2: Architecture using weighted resistor-ratio networks with amplitude modulation

3.2.2 Microcontroller with pulse-width modulation, low-pass filters and multipliers

Another way of implementing a resolver emulator is to generate differential sine and cosine voltage signals from the digital input position using a microcontroller and then to modulate those signals with the reference carrier. There are several ways to generate the required differential signals using a microcontroller. One particular method uses the resolver emulator architecture shown in Figure 3.3. It makes use of the pulse-width modulation (PWM) drivers available on many microcontrollers. Each of the four outputs of the resolver emulator requires its own PWM driver.

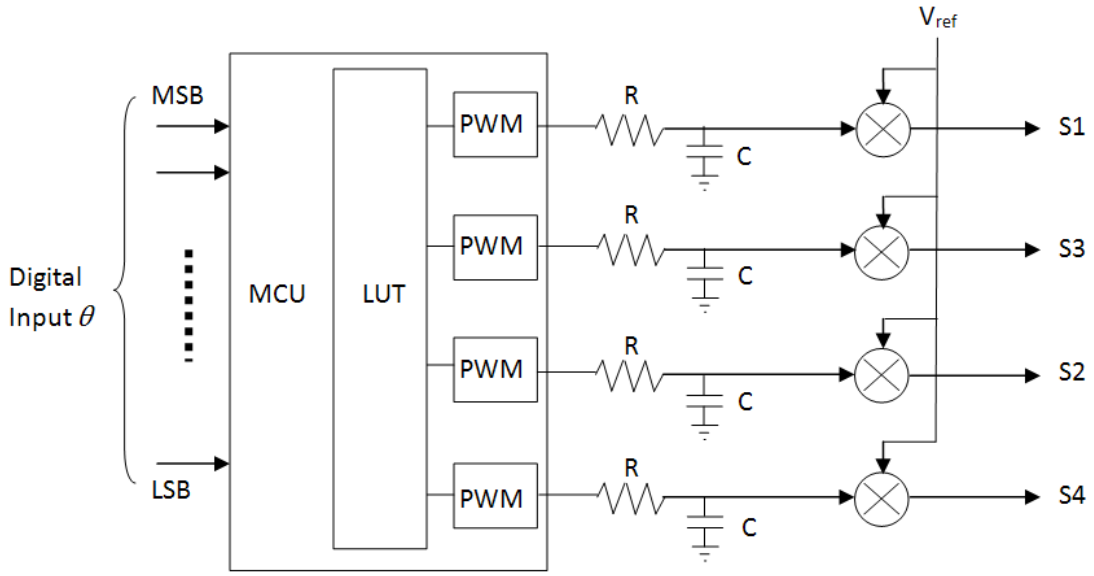


Figure 3.3: Architecture using microcontroller with pulse-width modulation, low-pass filters and multipliers

A microcontroller is used to convert the digital input position into corresponding pulse-width modulation duty cycles for the four resolver emulator output signals; the duty cycles can be stored in a lookup table (LUT) in the memory. Passive low-pass filters are used to remove the high PWM base frequency; the filtered voltages are directly proportional to the average time that a PWM signal is high, and the duty cycle for each PWM signal is set so that the average corresponds to the required sine or cosine value. Finally, the reference carrier is modulated with the sine and cosine signals to produce the resolver emulator outputs.

Many microcontrollers, including the PIC24 used later in this thesis, have a dedicated hardware peripheral for the generation of PWM signals, and matched pairs of PWM values with opposite polarity can be easily generated using the peripheral. The PWM approach can provide high resolution. No calculation is required, since the values stored in the LUT are pre-calculated, and so the microprocessor can also be used to implement other functions. Some microcontrollers do not have PWM drivers, and for these microcontrollers software implementation of PWM is also possible. Generally, software-based PWM is limited to applications for which the desired base frequency of the PWM is less than 1 kHz [20].

When designing the passive low-pass filter, both the PWM base frequency and the frequency of the shaft rotation must be carefully considered. The cut-off frequency should be low enough to effectively remove the PWM base frequency, but not so low that the filter attenuates signal at the frequency of the rotation. The PWM base frequency

must be significantly higher than the frequency of the shaft rotation in order for this to be possible.

A primary advantage of this architecture is its low cost; the cost of this architecture is moderate compared to other architectures. The discrete resistors and capacitors are generally inexpensive. Microcontrollers are also inexpensive. Analog multipliers account for the majority of the cost.

This architecture has several disadvantages. Because each of the four resolver emulator outputs requires its own unique PWM duty cycle for each possible digital input position, the memory required to store the LUT is relatively large. The architecture generally results in low precision resolver emulator outputs due to unavoidable device mismatches in the discrete components that produce the resolver emulator sine output and resolver emulator cosine output.

3.2.3 Microcontroller with digital-to-analog converters and multipliers

An alternative method of generating differential sine and cosine signals from digital input position is by using a combination of DACs and a microcontroller. This architecture creates sine and cosine representations of the digital input position directly, rather than indirectly using PWM.

Figure 3.4 shows the components of the architecture of microcontroller with digital-to-analog converters and multipliers. The first component of the architecture is the microcontroller, which uses a look-up table (LUT) with stored sine and cosine values to produce digital signals corresponding to the sine and cosine of the angle represented by

the digital input position. Two digital-to-analog converters with differential outputs are used to convert the digital sine and cosine representations into analog voltages in differential format. Finally, four analog multipliers are used to modulate the differential sine and cosine voltages with the reference carrier.

One of the advantages of this architecture is that a relatively simple microcontroller is sufficient. No PWM peripheral is needed. Further, the symmetry of the sine and cosine functions can be exploited when the look-up tables are constructed; only enough memory to store one-quarter of a cycle of sine data is needed. The reduction in LUT size is very important especially when the amount of memory in the microcontroller is limited and a high-resolution digital input position is desired.

The main disadvantage of this architecture is the relatively high cost of the four analog multipliers and two DACs. Device mismatch can also occur because two different DACs are used to produce the resolver emulator sine and cosine outputs.

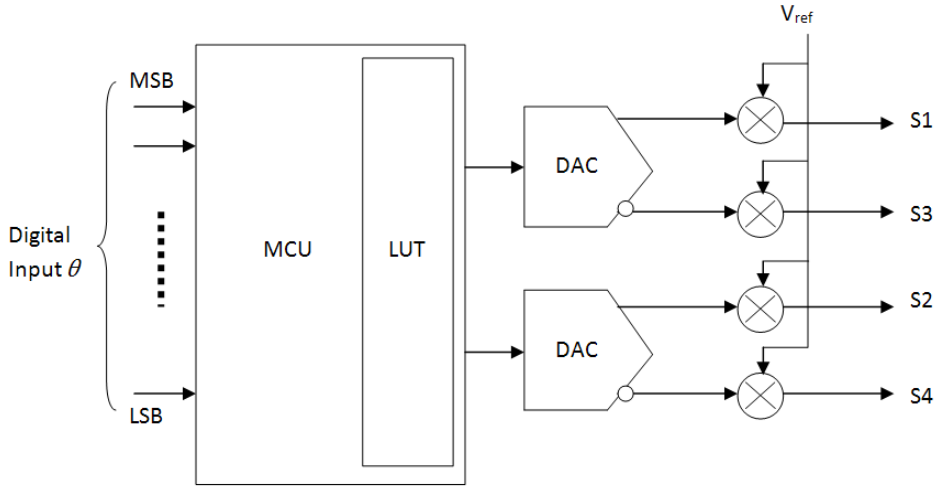


Figure 3.4: Architecture using microcontroller with digital-to-analog converters and multipliers

3.3 Microcontroller with a multiplying digital-to-analog converter

The final architecture presented in this chapter uses a microcontroller with a multiplying digital-to-analog converter, as shown in Figure 3.5. It is like the microcontroller with digital-to-analog converter and multipliers architecture, except that the two single-ended to differential DACs and four analog multipliers are replaced by a single four-quadrant multiplying digital-to-analog converter (mDAC). The mDAC is a two-channel device to convert multiple digital signals simultaneously to analog signals while modulating these signals with an external analog signal. The digital sine and cosine representing the digital input position are given as input to the mDAC, along with the reference carrier. The mDAC converts the single-ended digital sine and cosine representations from the microcontroller into four differential analog outputs that correspond to the input angle before modulating the analog signal with the reference carrier to produce the resolver emulator sine and cosine.

The architecture has several advantages. One is its simplicity; it requires only two components, the microcontroller and the external mDAC. Because problems of variation and mismatch are eliminated, the output accuracy is high. The absence of analog multipliers significantly drives down the component cost for this architecture. The cost for an mDAC is approximately $18 [19]. Multiplying DACs are available with a wide range of resolutions, up to 16 bits, and typically provide an output accuracy within a quarter of an LSB, so that the resolver emulator outputs have very low errors due to the mDAC.

Table 3.2 shows a qualitative comparison of the resolver emulator architectures in this chapter in terms of cost, number of external components, complexity, and output accuracy of the architectures. The architecture using a microcontroller with an mDAC is the most promising for implementing a resolver emulator with low cost and high accuracy. For this reason, it is chosen for further study and implementation.

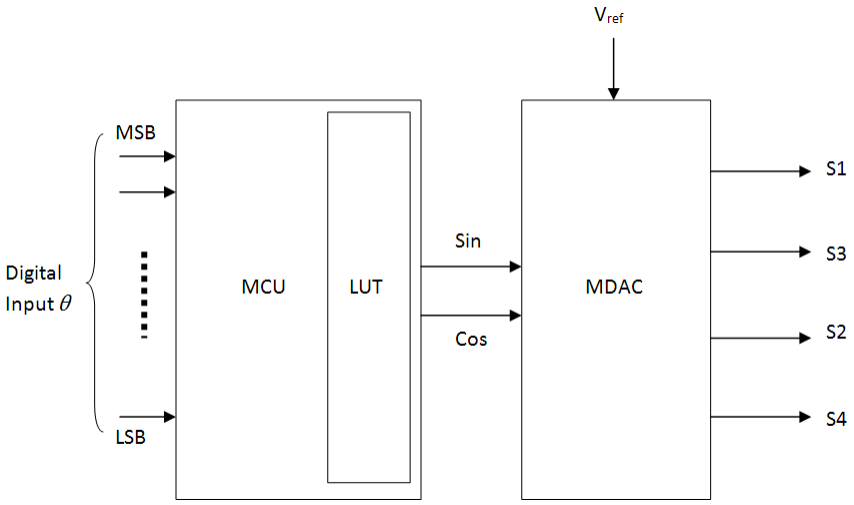


Figure 3.5: Architecture using microcontroller with mDAC

Table 3.2: Comparison of resolver emulator architectures

	Architectures			
	Weighted resistor-ratio network with AM	Microcontroller with PWM, LPFs, and multiplier	Microcontroller with DACs and multipliers	Microcontroller with mDAC
Cost[1]	High	Moderate	Moderate	Low
Number of components	13+	13	7	2
Complexity	High	Medium	Medium	Low
Output accuracy	Low	Average	High	Very high

Note 1: Cost based on digikey.com obtained on March 15, 2013

3.4 Summary

The important specifications of resolver emulators are considered in this chapter. Different resolver emulator architectures are described, and their advantages and disadvantages are explored. One particular architecture, the microcontroller with multiplying digital-to-analog converter, is chosen for further study based on cost, number of components, complexity, and output accuracy. Chapter IV will carefully study sources of error affecting the output accuracy of resolver emulators that use this architecture.

CHAPTER IV

SIMULATION OF PROPOSED ARCHITECTURE FOR RESOLVER EMULATOR

The previous chapter discussed several possible architectures for a resolver emulator, ending with the architecture proposed for implementation and further study, based on a microcontroller and a multiplying digital-to-analog converter. This architecture has the advantages of low cost, simplicity, and high output accuracy. In this chapter, a detailed analysis of the source of errors affecting the accuracy of the resolver emulator's outputs for the proposed architecture is conducted. Supporting simulations are conducted for each source of error, and results from analysis and simulation are compared.

4.1 Analysis of sources of error

The sources of error of the chosen architecture for the resolver emulator are shown in Figure 4.1. There are four main sources of error; they are quantization error in the digital input position, e_q, quantization errors in the sine and cosine computations, $e_{\sin}$ and $e_{\cos}$, and error due to the finite rate of update of the mDAC converter, e_u. The sources of error are treated as independent, and the system as linear, so that the effects of the four sources of error may be superimposed to find the total error in the outputs of the resolver emulator.

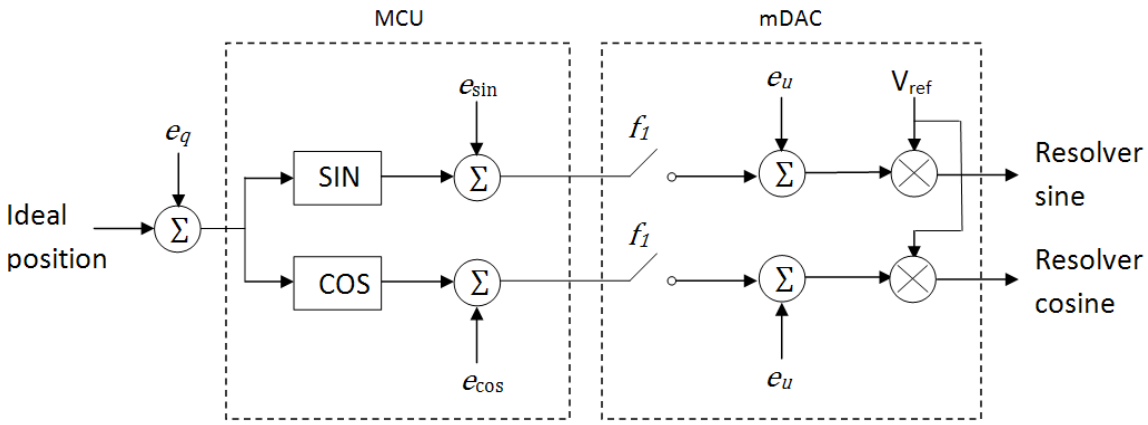


Figure 4.1: Sources of error in the proposed resolver emulator architecture

4.1.1 Quantization error in the digital input position

The actual position of a shaft can take any value between 0° and 360°; however, the input of the resolver emulator comes in the form of a digital position code with a limited number of bits, usually between eight and sixteen. Therefore, the digital input position is a quantized version of the actual position, and already has some quantization error in it. The quantized position can take only values that are a multiple of the quantization step size, where the quantization step size is the difference in degrees between two consecutive digital position codes. The relationship between the quantization step size, Δ_q, and the number of bits, b, in the digital position code is

$$\Delta_q = \frac{360^\circ}{2^b}, \tag{4.1}$$

where Δ_q has the units of degrees. The more bits in the digital input position code, the more accurately it can represent the actual shaft angle.

Because the digital input position of the resolver emulator is quantized, the resolver emulator outputs can represent the actual shaft angle no more accurately than the quantized input. Therefore, the number of bits in the digital input position code must be large enough to satisfy the output accuracy requirement. The quantized position, $(\theta)_q$, is defined as

$$(\theta)_q = Q_R(\theta_a, b), \tag{4.2}$$

where θ_a is the actual shaft position, b is the number of digital input bits, and Q_R is the quantization method that is used to quantize the actual position. The q-subscript is used as a notation for a value that has been quantized. The quantization error in the digital input position, e_q , in degrees is defined as

$$e_q = \theta_a - (\theta)_q. \tag{4.3}$$

Depending on the method that was used to quantize the actual position when producing the digital input position code for the resolver emulator, the maximum quantization error can vary. If the actual angle is represented by the closest corresponding digital input position code, the quantization error may have a magnitude no larger than half of the unit of the least significant bit; that is,

$$\left|e_q\right| \leq \frac{1}{2} \times \Delta_q. \tag{4.4}$$

4.1.2 Quantization error in the sine and cosine computation

The first step of the resolver emulator computation is to calculate the sine and cosine of shaft angle represented by the digital input position code. This calculation is

done by accessing a look-up table, stored in the microcontroller's memory, of sine and cosine values. Quantization error is introduced during this step because the memory has a finite capacity; as a result, both the number of words in the look-up table and the number of bits in a word are limited. The sine and cosine values are provided in digital form to the mDAC, which accepts values with only a limited number of bits, and it does not make sense to store more bits in the look-up table than can be used by the mDAC. For this reason, the analysis assumes that the number of bits in each sine and cosine value is equal to the number of bits accepted by the mDAC; this is referred to as the number of multiplier bits. Further, it is assumed that there is a unique sine or cosine value stored in the look-up table for each digital input position code.

The look-up table is structured to exploit the symmetry in the sine and cosine functions; one look-up table is used for both the sine and cosine functions. Only the first quadrant of the sine function is stored on the look-up table. The other quadrants are reflected versions of the first quadrant. The sine values of one quadrant range from 0 to 1. The sine values are stored in the look-up table using n bits, and the range from 0 to 1 is represented with 2^{n-1} discrete levels; the actual sine values are rounded as the look-up table of sine values is created, as part of the process of programming the microcontroller. The quantized sine $(\sin\ \theta_q)_q$ and quantized cosine $(\cos\ \theta_q)_q$ representations are denoted by

$$\left(\sin\theta_q\right)_q = \mathrm{Q}_R\left(\sin\theta_q, n\right), \tag{4.5}$$

and

$$\left(\cos\theta_q\right)_q = \mathrm{Q}_R\left(\cos\theta_q, n\right), \tag{4.6}$$

where θ_q is a quantized shaft angle corresponding to a digital input position code, n is the number of multiplier bits, and $\mathrm{Q_R}$ is the function that rounds a value to the nearest 2^{-n}. The quantization errors, $e_{\sin}$ and $e_{\cos}$, for the multiplier bits are

$$e_{\sin} = \sin\theta_q - \left(\sin\theta_q\right)_q, \tag{4.7}$$

and

$$e_{\cos} = \cos\theta_q - \left(\cos\theta_q\right)_q, \tag{4.8}$$

respectively. The maximum magnitude of the quantization error is half of the unit of the least significant bit, that is,

$$|e_{\sin}|, |e_{\cos}| \leq \frac{1}{2} \times \frac{1}{2^{n-1}} = 2^{-n}. \tag{4.9}$$

4.1.3 Error due to finite rate of update of the multiplying digital-to-analog converter

The third source of error is caused by the finite rate of update of the mDAC. The mDAC receives the quantized sine and cosine representations from the microcontroller via a serial communication link; these values are then converted into analog voltages before being used as multiplication factors in modulating the reference carrier to produce the resolver emulator outputs. Because new multiplier factors can be communicated only at discrete intervals in time, the multiplication factors being used by the mDAC become out-of-date between communications packets. The rate at which new multiplication factors are communicated to the mDAC is referred to as the update rate.

Several factors limit the update rate for the mDAC. One is the physical limits of the serial communication link itself. The serial communications between the microcontroller and the mDAC can transmit only one bit of data per clock cycle. If there are 16 bits of data per sine or cosine representation, it takes a total of 32 clock cycles to transmit both the sine and cosine representations. The clock frequency is itself limited; the maximum clock frequency is specified on the mDAC's datasheet. In addition, it does not make sense to communicate new multiplication factors to the mDAC faster than they can be converted to analog voltages by the mDAC.

When selecting an appropriate update rate, the frequency of the shaft rotation should be considered, because when the position is changing more quickly, more multiplier values become more out-of-date over the course of a fixed update period. The faster the update rate, the smaller the error due to the finite rate of update becomes. Therefore, choosing an update rate is one of the keys to design of an accurate resolver emulator. The error due to the update rate is derived from the number of degrees that the shaft has rotated since the multiplication factors were last updated; this error is bounded by

$$|e_u| \leq f_2 \times 360° \times \frac{1}{f_1} \tag{4.10}$$

where f_1 is the update rate in updates per second, and f_2 is the shaft rotational frequency in revolutions per second. To keep the error due to the update rate below one degree, the update rate should be at least 360 times faster than the shaft rotational frequency.

4.2 Resolver emulator simulated

Simulations were performed using various combinations of values for the three important parameters to achieving high output accuracy for the resolver emulator: the number of digital input bits (b), the number of multiplier bits (n), and the update rate (f_1). Simulations are done with 12 and 16 bits in the digital input position code; the corresponding codes range from 0 to 4095 and 0 to 65535 for the two choices, respectively. Using (4.4), the quantization error for the digital input position are 0.044° or 2.63 arcminutes for 12 digital input bits and 0.0027° or 0.16 arcminutes for 16 digital input bits.

Simulations are done with the number of multiplier bits set to 8, 12, and 14; this parameter sets the number of bits used to communicate a sine or cosine value from the microcontroller to the mDAC, and also determines the number of bits used to store the sine and cosine values in the microcontroller's look-up table. Using (4.9), the maximum magnitudes of the quantization error for the sine and cosine representations are 0.0039, 0.000245, and 0.00006 for 8, 12, and 14 bits of multiplier, respectively.

Two update rates are used for the simulations, 8 kHz and 16 kHz. These two rates, which are much slower than the maximum possible update rate for the mDAC, were chosen after considering the rate at which the microcontroller is able to operate. Assuming the shaft rotates at a rate of 1 Hz, using (4.10), the maximum errors due to the update rate are 2.7 arcminutes and 1.35 arcminutes for the 8 kHz and 16 kHz update rates, respectively. All combinations of these parameter values were simulated; thus, a total of twelve distinct resolver emulators were simulated.

4.3 Simulation method

Two types of simulation are conducted in this chapter. The first type of simulation is a forward ramp simulation. The forward ramp simulates the shaft position starting at 0° and making one complete revolution in the forward direction to $360° - \Delta_q$ at a constant speed of one revolution per second. The other type of simulation is a reverse ramp simulation. Instead of starting the shaft position at 0°, the shaft position is set to $360° - \Delta_q$ and makes one complete revolution in the reverse direction to 0°. The frequency of the reference carrier is set to 2 kHz.

The simulation itself is a discrete time simulation, implemented in MATLAB. The step size of the simulation is 1/131072 of a second; in other words, there are 131072 simulation steps as the shaft makes one complete revolution. For the simulations with a 16-bit digital input position code, there are two simulation steps for every digital input position code. Similarly, for the simulations with a 12-bit digital input position code, there are 32 simulation steps for every digital input position code.

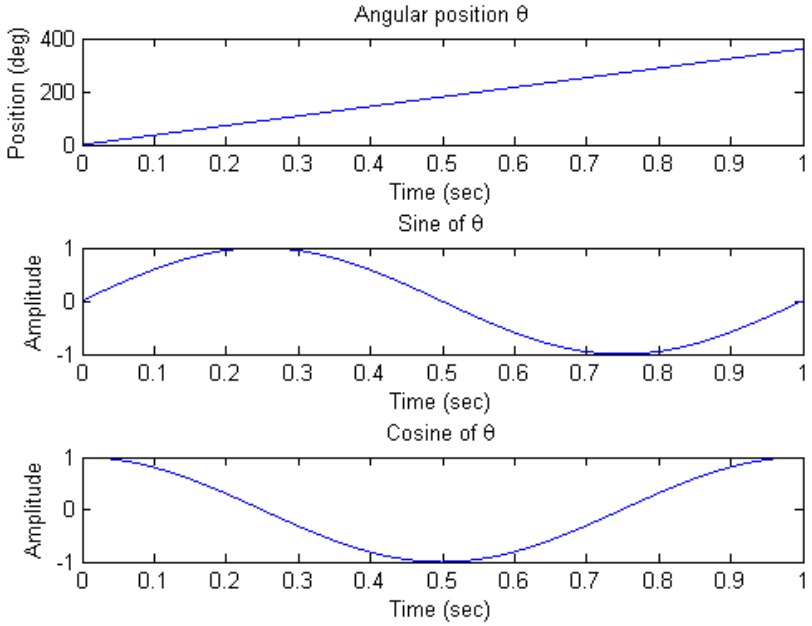
In each discrete time step, the quantized position is obtained by rounding the digital representation of the actual shaft angle to fit into the number of bits in the digital input position code. The digital input position is then converted into the sine and cosine representations. The quantized sine and cosine representations are obtained by rounding the sine and cosine representations to fit into the number of bits in the multiplier. Finally, for every t_1 seconds, where t_1 is the time between the updates, new quantized sine and cosine representations are obtained to be modulated with the reference carrier; thus there are f_1 updates per second, where f_1 is the update rate.

Figure 4.2 illustrates typical simulation results for a forward ramp simulation. It shows an example of the sine and cosine representation signals and the resolver emulator outputs for the forward ramp simulation with a reference carrier frequency of 20 Hz and a rotation speed of one revolution per second. Note that a slow reference carrier frequency is used for the plots in this figure for the purposes of illustration only; the simulations themselves use a frequency of 2 kHz. Figure 4.2(a) shows the shaft angles as function of time for a forward rotating shaft, and the corresponding sine and cosine of the shaft angle. Figure 4.2(b) shows the reference carrier signal and the resolver emulator sine and cosine outputs of shaft position. Note that the sine and cosine representation signals from Figure 4.2(a) become the envelopes of the resolver emulator outputs.

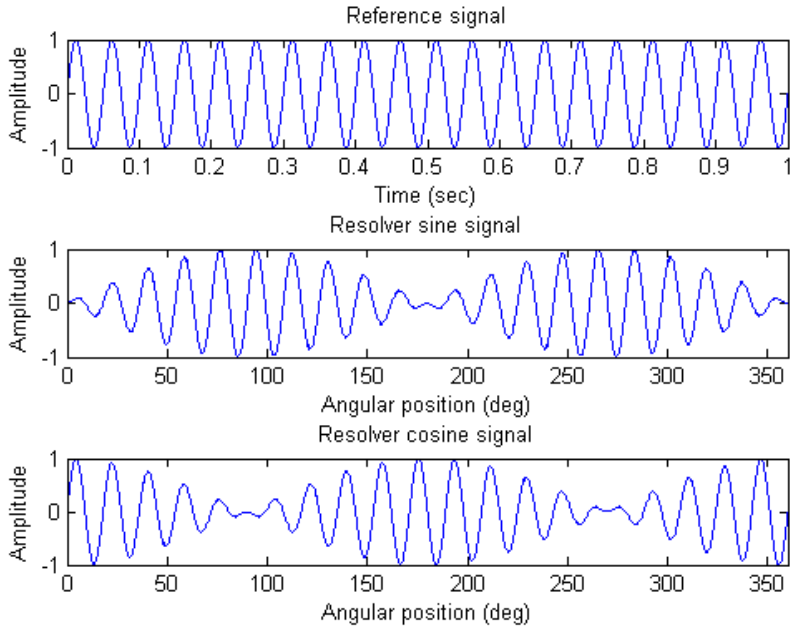
After the simulated resolver emulator outputs are produced, the shaft angle is calculated from these outputs. While this step does not simulate the resolver emulator itself, it is necessary for evaluating the accuracy of the resolver emulator's output. The algorithm for determining the shaft angle from the resolver emulator outputs is

$$\theta' = \tan^{-1}\left(\frac{V_s(\theta)}{V_c(\theta)}\right), \tag{2.5}$$

where V_s is the resolver emulator sine, V_c is the resolver emulator cosine, and θ'indicates the shaft angle with undetermined quadrant. The quadrant for the shaft angle can be determined by looking at the phases of the sine and cosine outputs with respect to the reference carrier, following the procedure in Table 2.1.



(a) Sine and cosine as a function of time for a forward rotating shaft



(b) Resolver emulator outputs as a function of shaft angle for the forward rotating shaft

Figure 4.2: Sine and cosine representations and ideal resolver emulator outputs for forward ramp simulation with a reference carrier frequency of 20 Hz and a rotation speed of 1 Hz

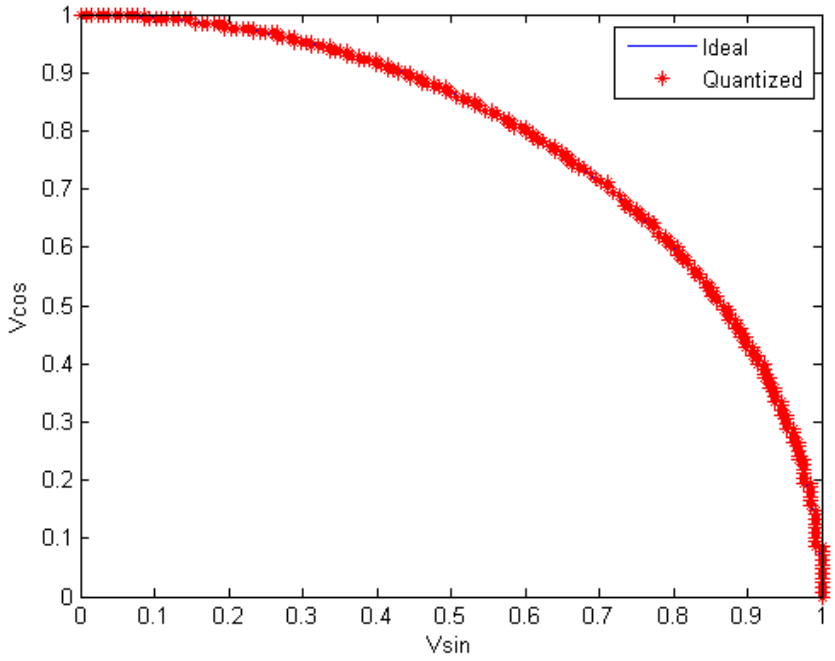
To obtain the total error of the resolver emulator, the calculated shaft angle is compared with the actual shaft angle at each discrete time step, except when the resolver cosine output is zero. Because the calculated shaft angle is undefined when the resolver cosine is zero, the error calculations cannot be done at these time steps. Errors are reported in several forms. The maximum positive and negative errors seen in the simulation are reported. The simulation is intentionally designed so that the worst case seen in practice for each digital input position code is also seen in simulation. As a result, the maximum positive and negative errors seen in the simulation correspond to the theoretical worst cases. An average error is also reported; because some errors are positive and others are negative, the average may be small because of cancellation. For this reason, an average magnitude of the error is also reported.

4.4 Illustration of the effect of quantization on the sine and cosine of the shaft angle

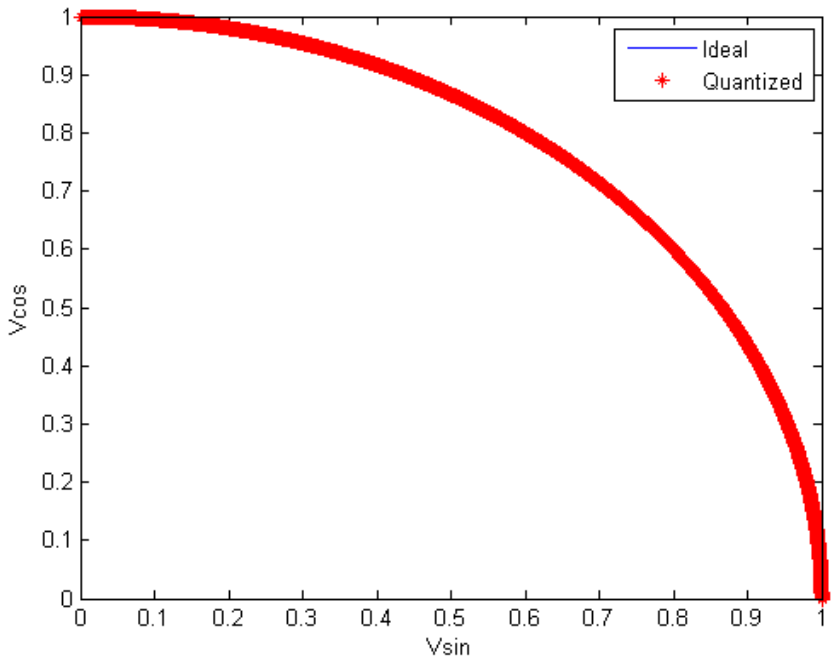
The purpose of the simulations is to understand how the choices for the main design parameters affect the output accuracy with which the sine and cosine outputs of the resolver emulator reflect the actual shaft position. Preliminary to a full simulation, the effects of the qualtization on the sine and cosine of the shaft angle are addressed. Ideally, if a plot of the sine of a shaft angle is plotted versus the cosine of the same shaft angle as the shaft makes one complete forward revolution, the points will lie along the unit circle; in the presence of quantization, the points may no longer be perfectly aligned on the circle. Two versions of the sine versus the cosine for the first quarter rotation are plotted in Figure 4.3 (a) for a resolver emulator with a digital input position code of 12 bits and multiplier of 8 bits; the ideal and the quantized. The first plot, perfectly circular, is the

ideal case. The second, in red, shows the relationship between sine and cosine after the values have been quantized to eight bits to fit in the look-up table. Figure 4.3(b) shows the same two plots, but for a resolver emulator with a digital input position code of 16 bits and multiplier of 14 bits.

Figure 4.4(a) and (b) show close-up views of part of the same plots that are in Figure 4.3 so that the effect of the quantization on the relationship between sine and cosine is more easily seen. The plot for the ideal sine and cosine representations is a unit circle. The quantized version, however, shows that the implemented curve is a series of steps; for the scale shown, the sine versus cosine plot for the resolver emulator using a digital input position code of 12 bits and a multiplier of 8 bits looks like a series of steps, but the plot for the resolver emulator using a digital input position code of 16 bits and a multiplier of 14 bits still looks like a smooth circle. An even more zoomed-in version, in Figure 4.5(a) and (b), shows that if the scales are fine enough, the plot for the resolver emulator with a digital input position of 16 bits and a multiplier of 14 bits also becomes a series of steps. Recall from (4.10) that the sizes of the errors from the quantization of sine and cosine representations are inversely proportional to 2^n, where n is the number of multiplier bits. This is demonstrated on the figures, where the higher the number of multiplier bits, the smaller the step size becomes.

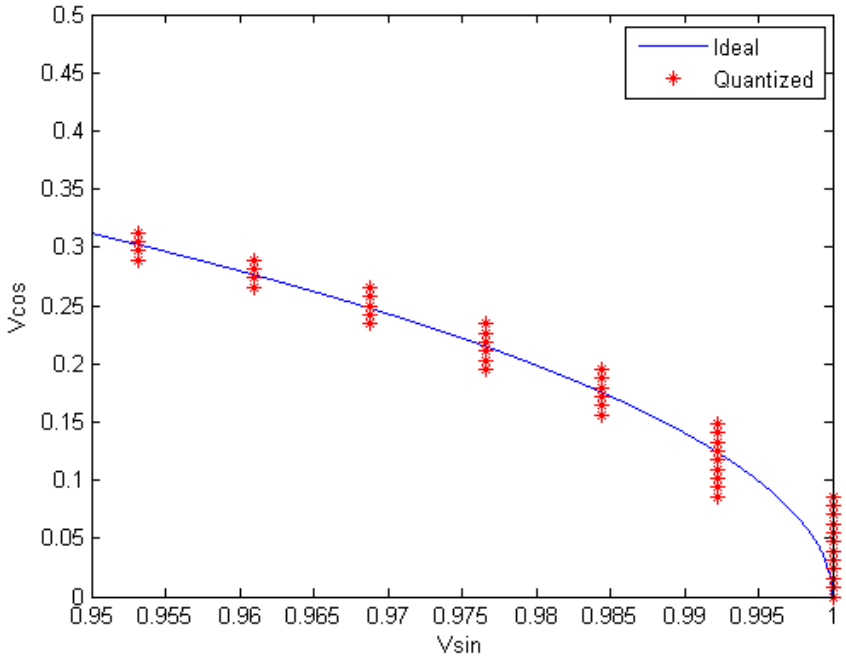


(a) With 12 digital input bits and 8 multiplier bits

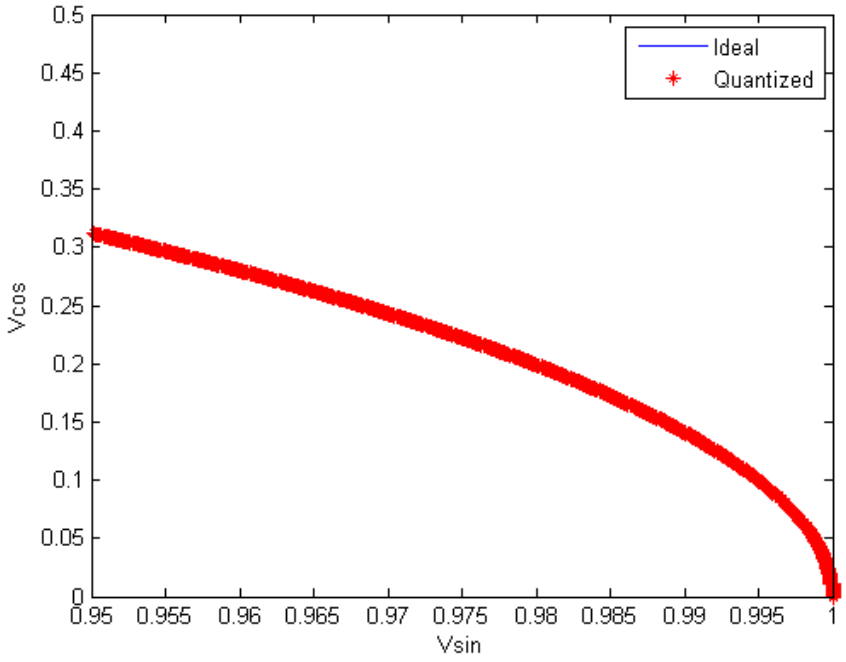


(b) With 16 digital input bits and 14 multiplier bits

Figure 4.3: Plots of the sine versus cosine for the forward ramp simulation

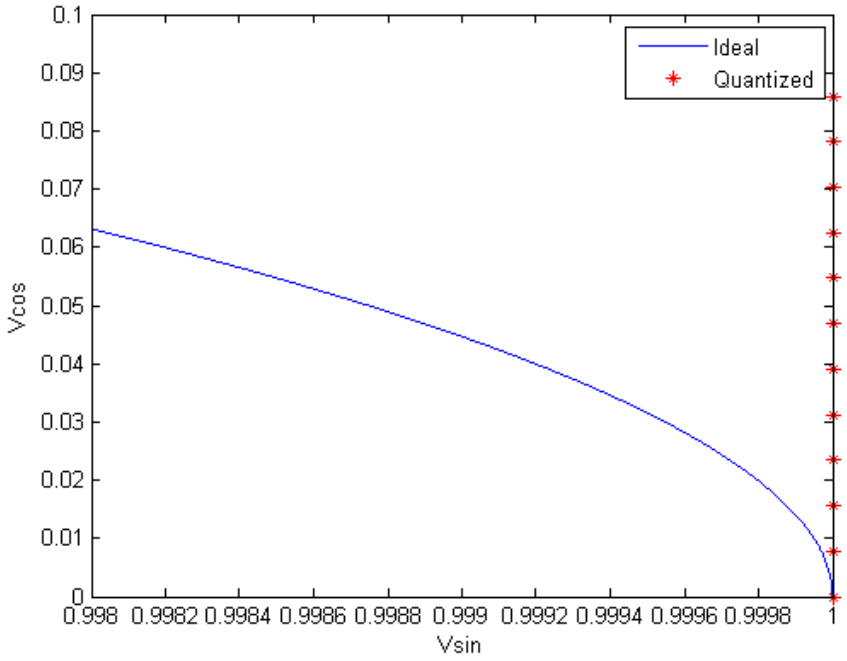


(a) With 12 digital input bits and 8 multiplier bits

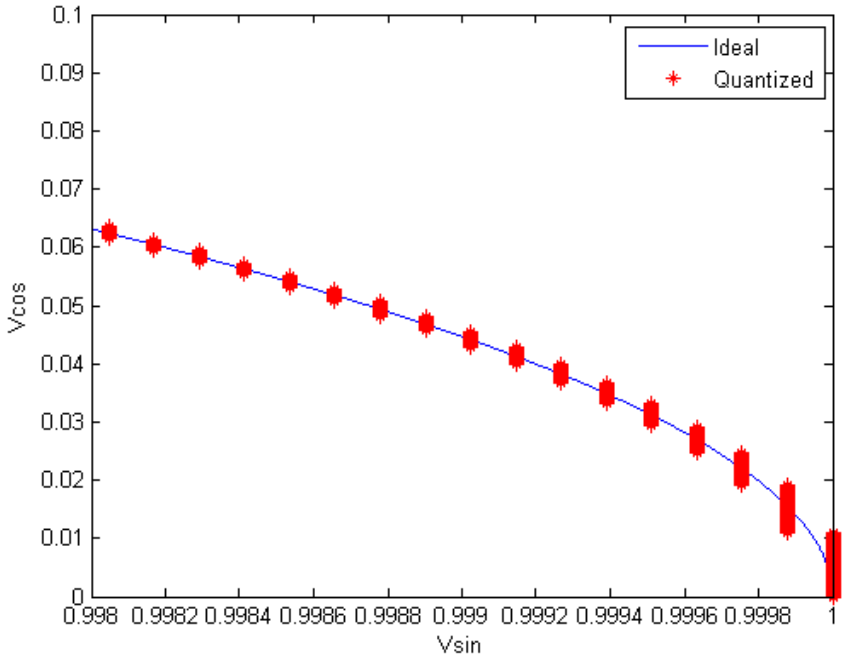


(b) With 16 digital input bits and 14 multiplier bits

Figure 4.4: First zoomed-in version of plots of the sine versus cosine for the forward ramp simulation



(a) With 12 digital input bits and 8 multiplier bits



(b) With 16 digital input bits and 14 multiplier bits

Figure 4.5: Second zoomed-in version of plots of the sine versus cosine for the forward ramp simulation

4.5 Effect of the quantized sine and cosine on the forward and reverse ramp simulations

In this section, the effect of the quantized sine and cosine on the full simulation of the forward and reverse ramps is described. The forward ramp simulation is for the shaft rotating in a clockwise direction at a constant speed of one revolution per second with a starting position of 0° (digital input position code 0) and an ending position of $360° - \Delta_q$ (digital input position code $2^b - 1$). As before, the reference carrier frequency is 2 kHz, and the simulation step size is 1/131072 of a second. In the description that follows, one source of error at a time is added to the simulation; at the end, the combined effect of all the sources is considered.

The first source of error introduced when designing a resolver emulator architecture comes from the fact that the digital input position code uses a finite number of bits, so that the actual position is quantized at the input of the resolver emulator. This effect is illustrated in simulation. Figure 4.6 shows the actual and quantized digital input positions of the forward ramp simulation for the first 20 time steps in the simulations using a16-bit digital input position code. Note that the step size of the quantized digital input position is 0.0055° or 0.33 arcminutes, as expected from (4.4).

The simulation is set up such that there are two discrete time steps for each possible quantized digital input position code. The first step corresponds to the instant in time at which the actual position corresponds exactly to that code, and the second time step corresponds to the instant at which the actual position is halfway between that code and the next position. Note that the second discrete time step for each position corresponds to a worst case error in the quantized digital input position; this error is 0.162

arcminutes. This matches the calculations in Section 4.2, where the worst case error is half of the quantization step size for the number of bits, b, in the digital position code. Because of the simulation set-up, the maximum error seen in the simulation is the maximum error over all actual positions, and the average error of the discrete times simulated corresponds to the average error over all possible actual positions.

The second source of error comes from the fact that the sine values, stored in a look-up table in the microcontroller use only a finite number of bits. This effect is also illustrated in simulation. Figure 4.7 shows the actual and quantized sine representations of the first 20 discrete time steps of the forward ramp simulation for a resolver emulator that uses a 14-bit multiplier and quantized digital input position code with 16 bits. Note that, as given in (4.5) the sine representations have a fixed step size, based on the number of bits used to represent each sine value in the look-up table. However the error in the sine representation at a given step in the simulation depends not only on the shaft angle, but also on the quantization of the digital input position; as a result, it is not a linear function of shaft angle. Using (4.9), the maximum quantization error of the sine and cosine representations is 0.000061 over the entire 360 degrees for a resolver emulator with 14 multiplier bits and a 16-bit digital input position code. The maximum error seen in the simulation matches this theoretical maximum.

The third source of error stems from the fact that the sine and cosine values communicated to the mDAC, and used to modulate the reference carrier frequency, can be updated at only a finite rate. The final step is to consider the total error in the resolver emulator's sine and cosine outputs results from the combined effects of all three sources

of errors; the quantization error in the digital input position, the quantization error in the sine and cosine computation, and the error due to finite rate of update of the multiplying digital-to-analog converter. The end-to-end simulation is described shortly in Section 4.6.

The effects of quantization of the sine and cosine values on the forward ramp simulation have now been discussed; the same steps are now taken for the reverse ramp simulation. Similar to the forward ramp, a reverse ramp simulation imitates the shaft rotating at a constant speed. However, the shaft rotates in a counterclockwise direction with a starting position of $360° - \Delta_q$ (digital input position code $2^b - 1$) and an ending position of 0° (digital input position code 0). The rotation speed is also 1 revolution per second or 1 Hz.

Figure 4.8 shows the actual and quantized digital input positions of the reverse ramp simulation for the last 20 time steps in the simulations using a 16-bit digital input position code. The step size of the quantized digital input position is 0.0055° or 0.33 arcminutes. Like the forward ramp, the simulation is set up such that there are two discrete time steps for each possible quantized digital input position code. The first step corresponds to the instant in time at which the actual position corresponds exactly to that code, and the second time step corresponds to the instant at which the actual position is halfway between that code and the next position. Note that the second discrete time step for each position corresponds to a worst case error in the quantized digital input position; this error is 0.162 arcminutes, which matches the theoretical worst case error in Section 4.2. In addition, note that the error has an opposite sign from the forward ramp.

The sine and cosine representations of the digital input position of the reverse ramp are also similar to the forward ramp; the difference is the opposite sign of the actual and quantized sine representations, as shown in Figure 4.9. The maximum quantization error of the sine and cosine representations is 0.000061 over the entire 360 degrees for a resolver emulator with 14 multiplier bits and a 16-bit digital input position code, which matches the theoretical maximum. As for the forward ramp, the combined effects of all three sources of errors are considered in an end-to-end simulation in the next section.

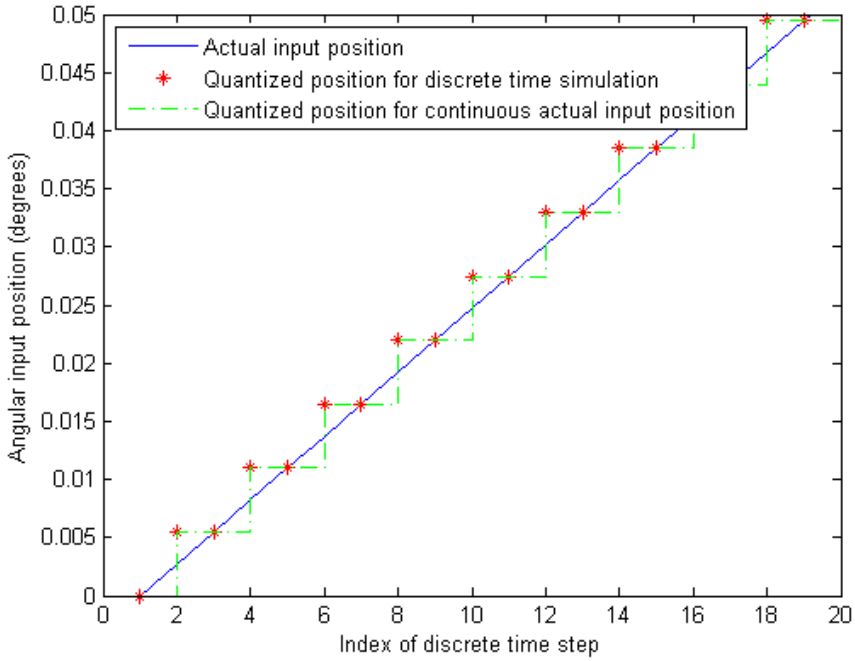


Figure 4.6: Actual and quantized input positions of the first 20 discrete time steps of the forward ramp simulation for 16-bit digital input position

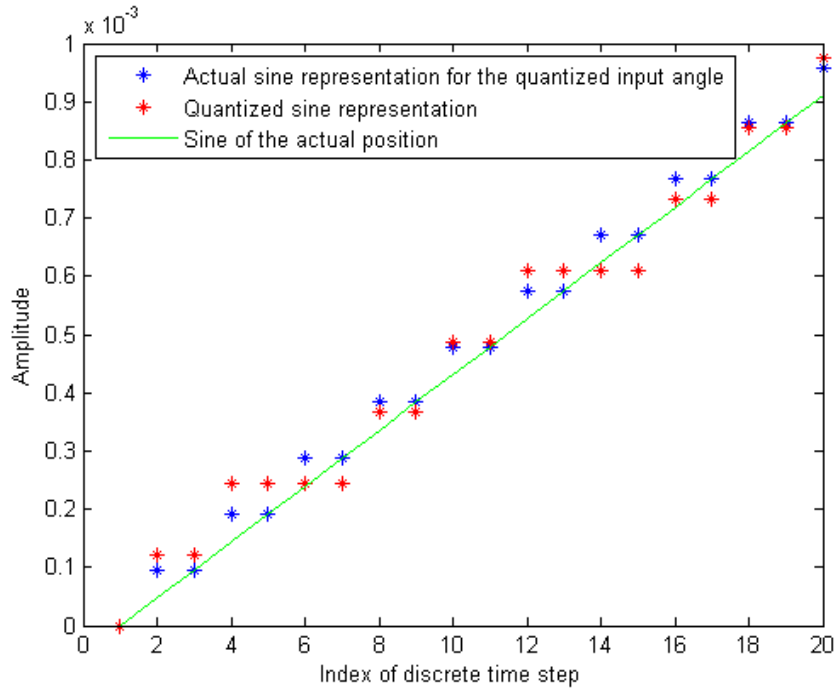


Figure 4.7: Actual and quantized sine representations of the first 20 discrete time steps of the forward ramp simulation for 14-bit multiplier and 16-bit digital input position

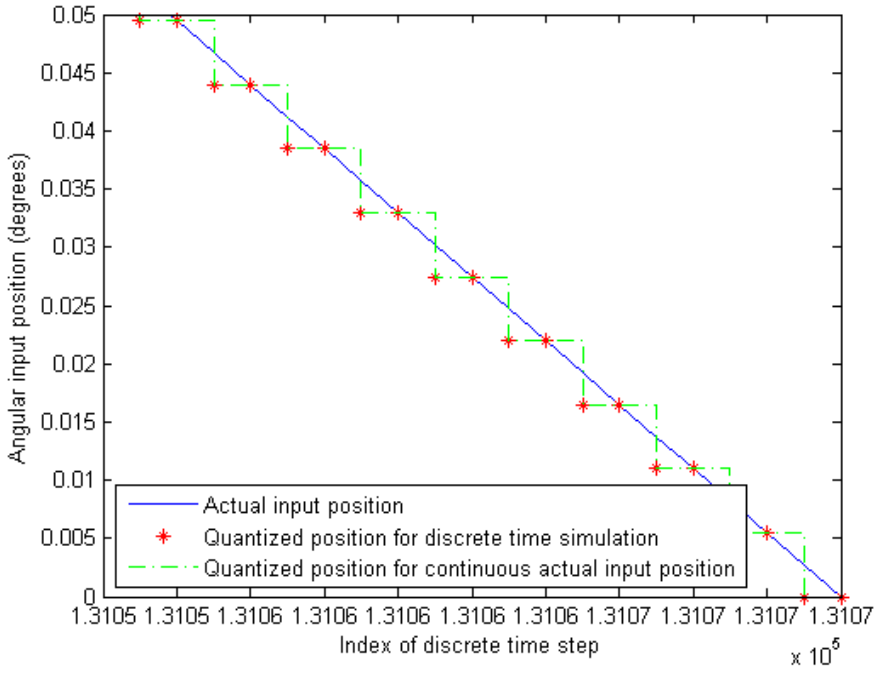


Figure 4.8: Actual and quantized input positions of the last 20 discrete time steps of the reverse ramp simulation for 16-bit digital input position

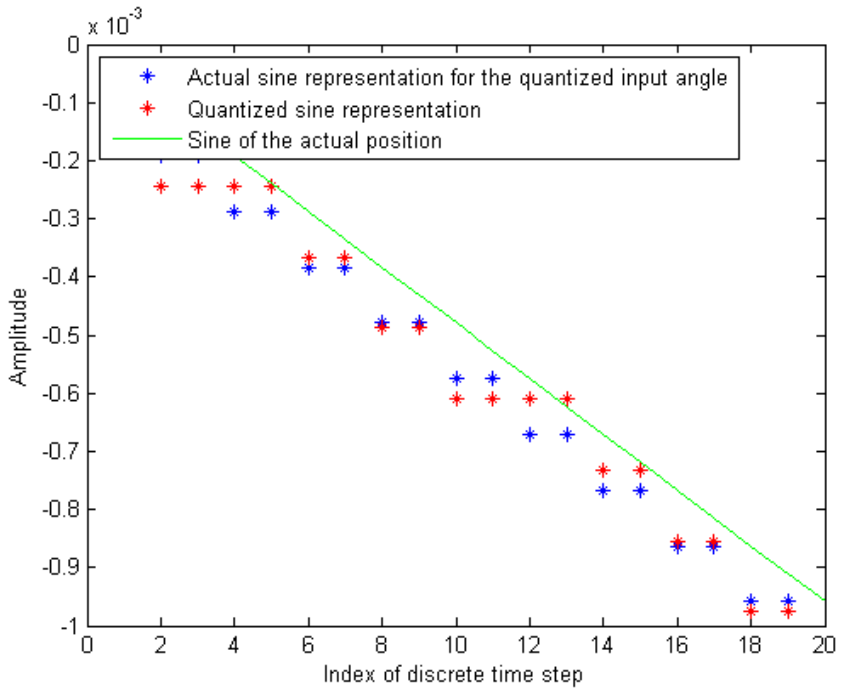


Figure 4.9: Actual and quantized sine representations of the first 20 discrete time steps of the reverse ramp simulation for 14-bit multiplier and 16-bit digital input position

4.6 Analysis of the total error of the forward ramp and the reverse ramp simulations

Table 4.1 and Table 4.2 summarize the total error from the simulations of the forward ramp and reverse ramp, respectively, for the twelve variations of the resolver emulator architecture, using various combinations of the design parameters. Maximum positive and negative errors are reported, as are the average error and the average error magnitude. Average error is the obtained by averaging the errors at the individual discrete time steps. Average error magnitude, on the other hands, is obtained by averaging the absolute values of the errors at the individual discrete time steps.

The total error results are illustrated by showing detailed plots for two of the twelve resolver emulator variations as examples, for both the forward and reverse ramp

simulations. The first example is the resolver emulator having a digital input position of 12 bits, a multiplier of 8 bits and an update rate of 8 kHz. The second example is the resolver emulator having a digital input position of 16 bits, a multiplier of 14 bits and an update rate of 16 kHz. Figure 4.10 shows the position indicated by the resolver emulator outputs at each discrete time step in the forward ramp simulation, for the full one second time frame. Note that it is difficult to see any difference between the two example resolver emulators when the results are plotted on this scale. In order to analyze the differences, close-up views of the same simulation are shown in Figure 4.11. The two close-up views are plotted on different scales; Figure 4.11(a), for the resolver emulator having a digital input position of 12 bits, a multiplier of 8 bits, and an update rate of 8 kHz, has a scale of 0 to 0.01 seconds on the x-axis and Figure 4.11(b), for the resolver emulator having a digital input position of 16 bits, a multiplier of 14 bits, and an update rate of 16 kHz, has a scale of 0 to 0.5 milliseconds on the x-axis. This was done intentionally to make the total error clearly visible in each case. To see the difference of errors between the two resolver emulators from Figure 4.11, these errors are plotted in Figure 4.12, which shows the absolute errors in the angle indicated by the resolver emulator's sine and cosine outputs as a function of time for the forward ramp simulation.

Similar to the forward ramp examples, Figure 4.13 shows the position indicated by the resolver emulator outputs at each discrete time step in the reverse ramp simulation, for the full one second time frame. Close-up views of the same simulation are shown in Figure 4.14. The two close-up views are plotted on different scales; Figure 4.14(a), for the resolver emulator having a digital input position of 12 bits, a multiplier of 8 bits, and an update rate of 8 kHz, has a scale of 0.99 to 1 second on the x-axis and Figure 4.14(b),

for a resolver emulator having a digital input position of 16 bits, a multiplier of 14 bits, and an update rate of 16 kHz, has a scale of 999.5 to 1000 milliseconds on the x-axis. Figure 4.15 shows the absolute errors in the angle indicated by the resolver emulator's sine and cosine outputs as a function of time for the reverse ramp simulation.

The main difference between the forward ramp and the reverse ramp is the sign of the average total error for a given resolver emulator. The average total error for the forward ramp is always positive, as shown in Figure 4.11, because the resolver emulator output angle becomes outdated while waiting for the next multiplier update to come; at the same time, because the ramp is in the forward direction, the outdated angle is smaller than the actual angle, and the error is negative. This can be seen on Figure 4.12(b), which shows the absolute error for the reverse ramp simulation the resolver emulator having a 16 digital input bits, a 14 multiplier bits, and an update rate of 16 kHz.

The opposite can be seen for the reverse ramp simulation. The average total error for the reverse ramp is always negative, as shown in Figure 4.14. The resolver emulator output angle become outdated while waiting for the next multiplier update to come; so the outdated angle is larger than the actual angle, thus the error is positive, as seen in Figure 4.15(b).

Table 4.1: Summary of total errors of forward ramp simulation

Digital input bit	Multiplier bit	Update rate (Hz)	Range of error (arcmin)	Average error (arcmin)	Average error magnitude (arcmin)
12-bit	8-bit	8k	(-20.55, 20.39)	13.10	19.91
12-bit	8-bit	16k	(-20.55, 20.39)	13.10	19.91
12-bit	12-bit	8k	(-3.75, 3.59)	2.55	4.01
12-bit	12-bit	16k	(-3.75, 3.59)	2.55	4.01
12-bit	14-bit	8k	(-2.91, 2.75)	2.55	3.95
12-bit	14-bit	16k	(-2.91, 2.75)	2.55	3.95
16-bit	8-bit	8k	(-18.47, 20.94)	14.41	19.85
16-bit	8-bit	16k	(-18.74, 19.89)	13.76	19.79
16-bit	12-bit	8k	(-1.11, 3.59)	1.23	1.29
16-bit	12-bit	16k	(-1.11, 2.37)	0.57	0.69
16-bit	14-bit	8k	(-0.28, 2.75)	1.23	1.24
16-bit	14-bit	16k	(-0.28, 2.37)	0.57	0.59

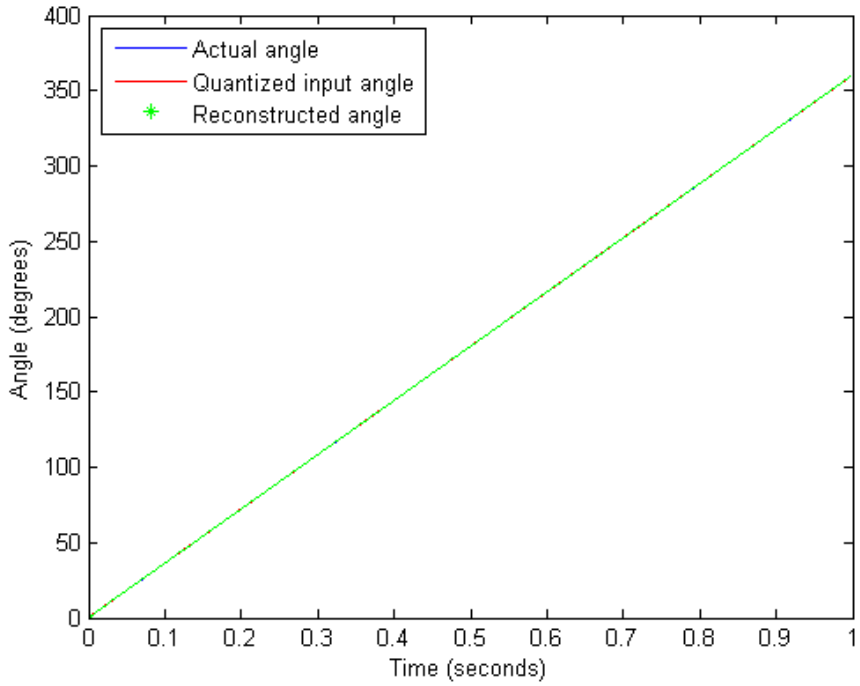
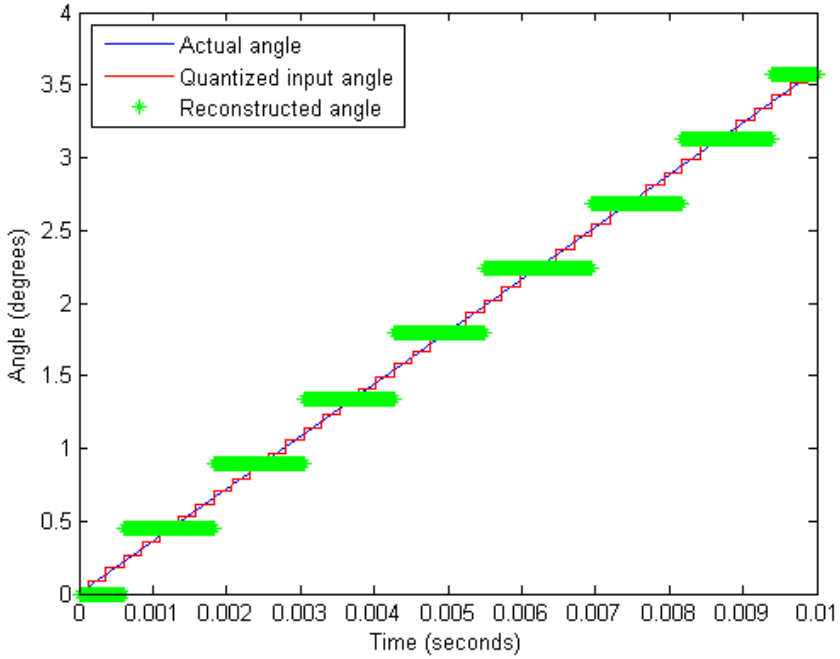
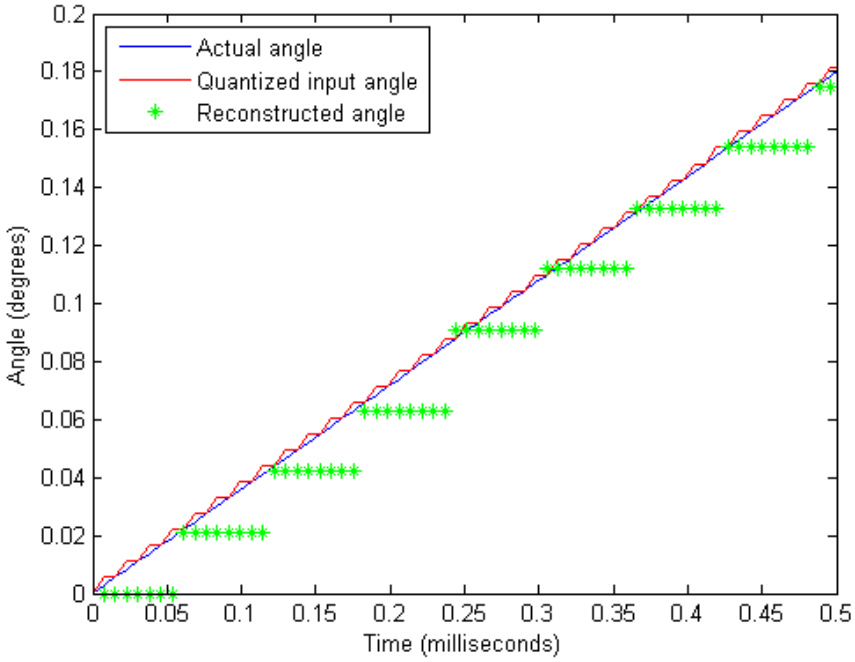


Figure 4.10: Shaft angle of the resolver emulator for the forward ramp simulation

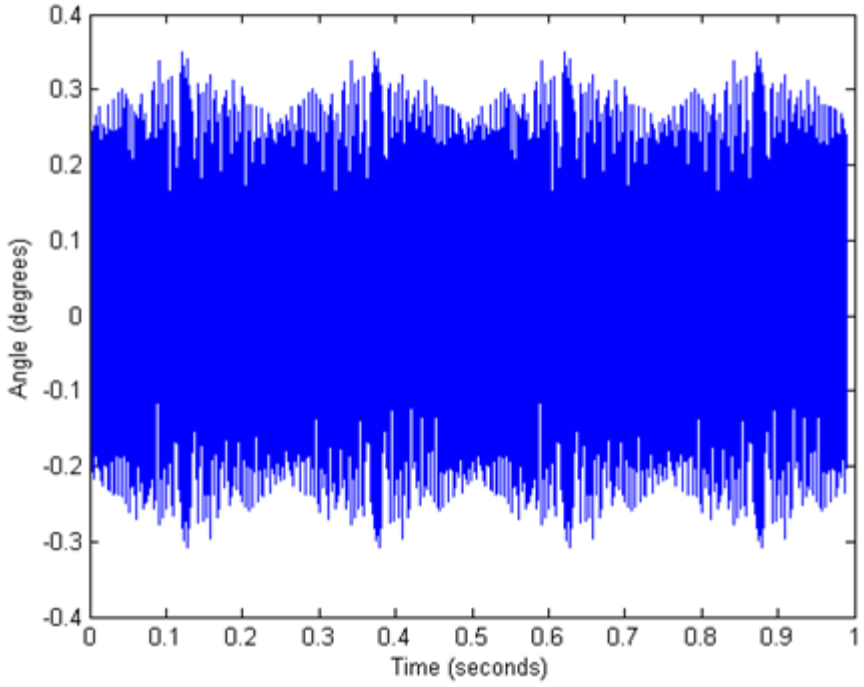


(a) With 12 digital input position bits, 8 multiplier bits, and 8 kHz update rate

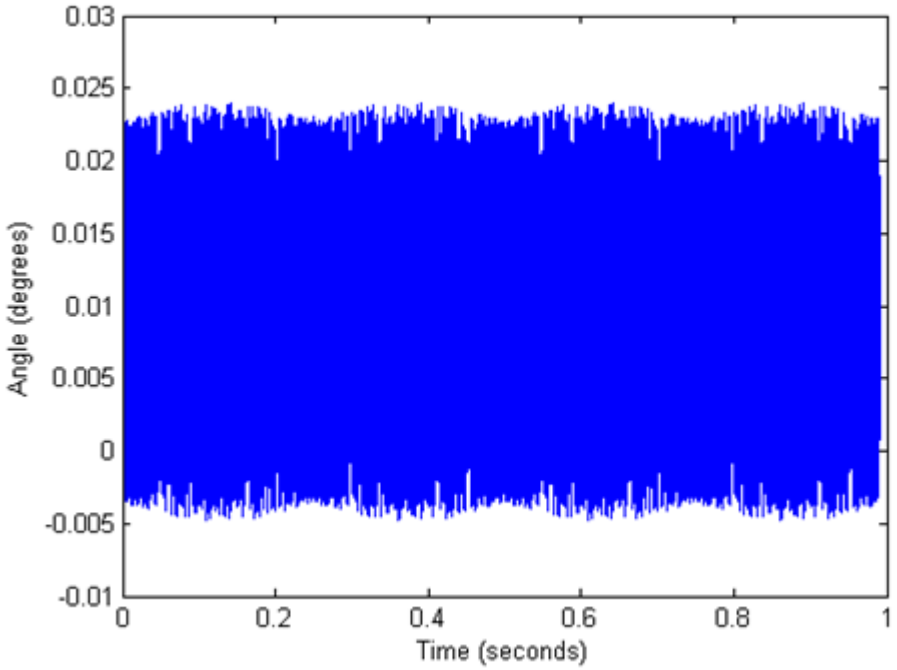


(b) With 16 digital input position bits, 14 multiplier bits, and 16 kHz update rate

Figure 4.11: Zoomed-in version of resolver emulator output for the forward ramp simulation



(a) With 12 digital input position bits, 8 multiplier bits, and 8 kHz update rate



(b) With 16 digital input position bits, 14 multiplier bits, and 16 kHz update rate

Figure 4.12: Absolute errors of resolver emulator output for the forward ramp simulation

Table 4.2: Summary of total errors of reverse ramp simulation

Digital input bit	Multiplier bit	Update rate (Hz)	Range of error (arcmin)	Average error (arcmin)	Average error magnitude (arcmin)
12-bit	8-bit	8k	(-20.55, 20.39)	12.93	19.74
12-bit	8-bit	16k	(-20.55, 20.39)	12.93	19.74
12-bit	12-bit	8k	(-3.75, 3.59)	2.38	3.84
12-bit	12-bit	16k	(-3.75, 3.59)	2.38	3.84
12-bit	14-bit	8k	(-2.91, 2.75)	2.38	3.79
12-bit	14-bit	16k	(-2.91, 2.75)	2.38	3.79
16-bit	8-bit	8k	(-21.11, 18.31)	14.25	22.34
16-bit	8-bit	16k	(-20.05, 18.57)	13.59	20.95
16-bit	12-bit	8k	(-3.75, 2.21)	1.07	3.91
16-bit	12-bit	16k	(-2.43, 2.21)	0.41	1.96
16-bit	14-bit	8k	(-2.91, 2.21)	1.07	3.87
16-bit	14-bit	16k	(-1.61, 2.21)	0.41	1.89

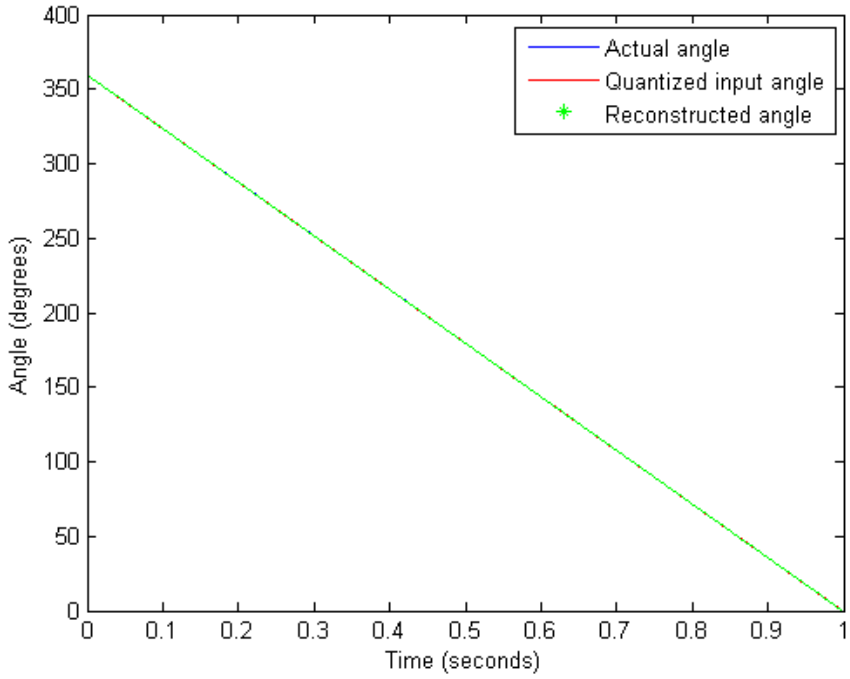
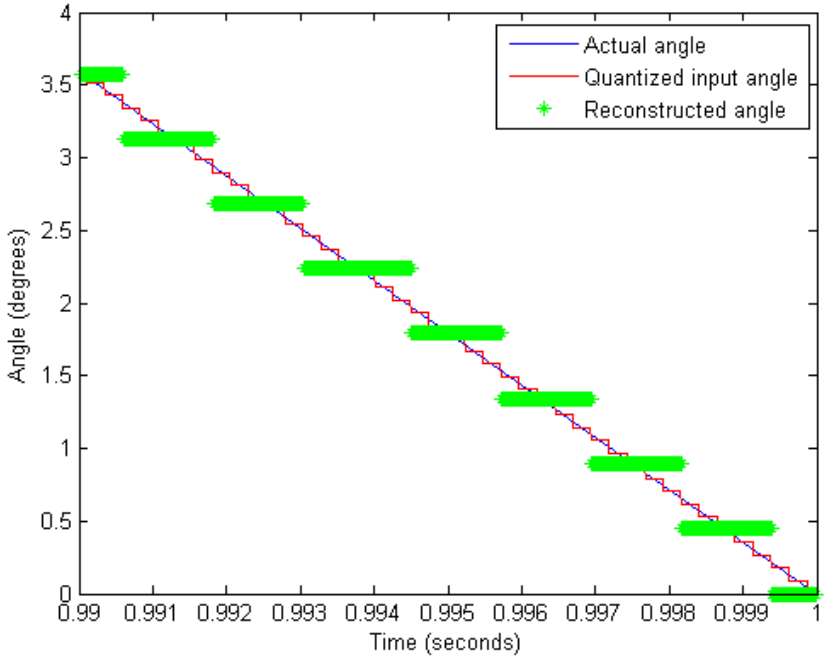
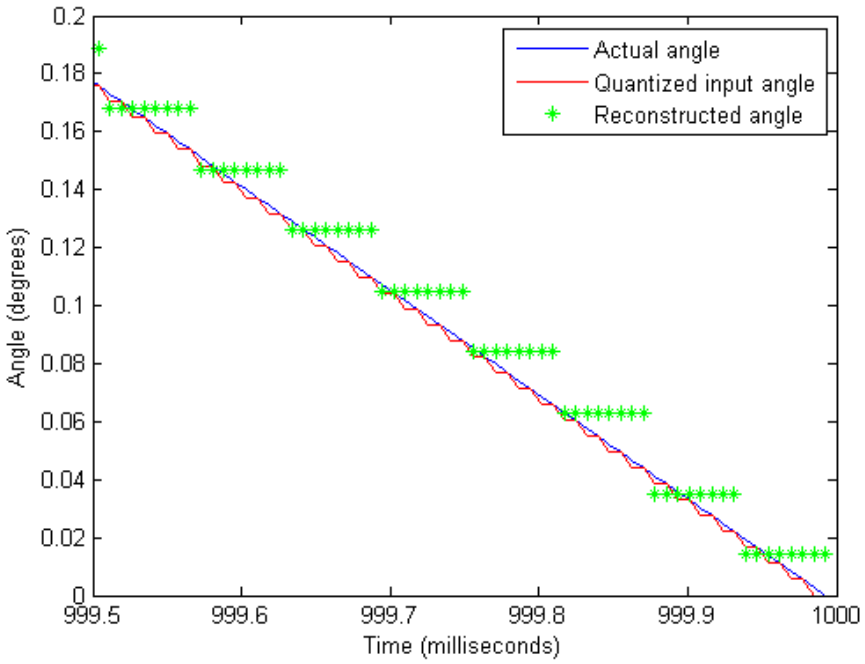


Figure 4.13: Shaft angle of the resolver emulator output for the reverse ramp simulation

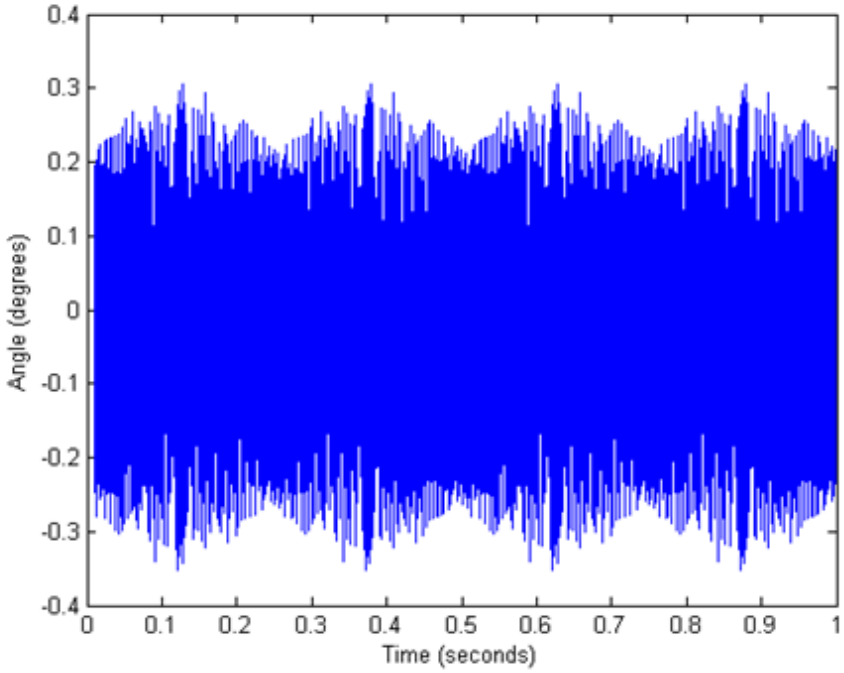


(a) With 12 digital input position bits, 8 multiplier bits, and 8 kHz update rate

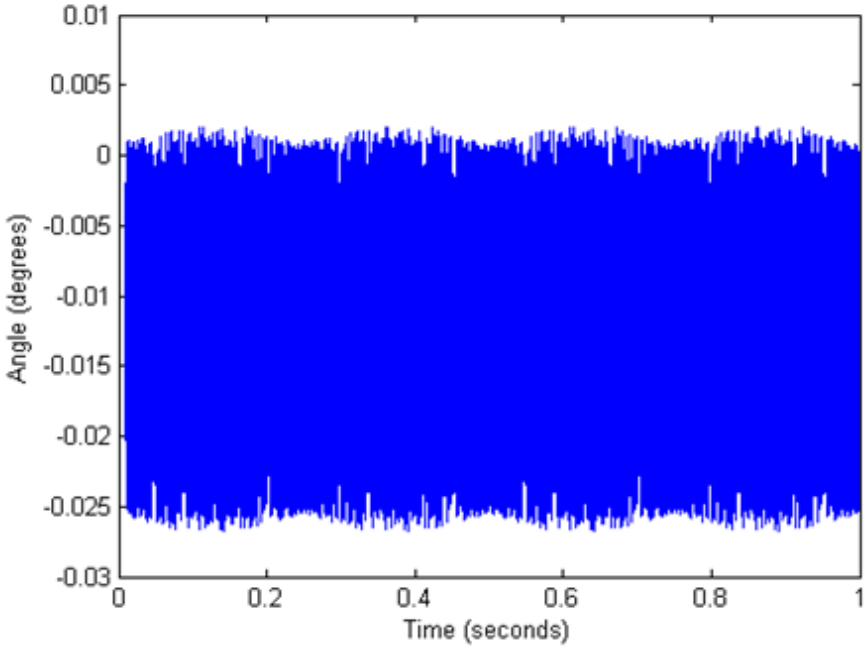


(b) With 16 digital input position bits, 14 multiplier bits, and 16 kHz update rate

Figure 4.14: Zoomed-in version of resolver emulator output for the reverse ramp simulation



(a) With 12 digital input position bits, 8 multiplier bits, and 8 kHz update rate



(b) With 16 digital input position bits, 14 multiplier bits, and 16 kHz update rate

Figure 4.15: Absolute errors of resolver emulator output for the reverse ramp simulation

The results in Tables 4.1 and 4.2 are plotted in a number of ways to give insight into the effects of the design parameters on the accuracy with which the resolver emulator outputs indicate the actual shaft position. Figure 4.16 shows the average and range of total error for the forward ramp simulation, and Figure 4.17 shows the average magnitude of total error for the forward ramp simulation. This allows the average errors and average error magnitudes of each resolver emulator in Table 4.1 to be visualized in two plots. Figure 4.18 and Figure 4.19 show the same kind of results, but for the reverse ramp simulation in Table 4.2.

Figures 4.16 to 4.19 give insight into how the number of bits of digital input position should be chosen. The total errors are observed in terms of the average error, as shown in Figures 4.16 and 4.18, and the average error magnitude, as shown in Figures 4.17 and 4.19. Observe the difference between the subplots within the figures in particular, when increasing the number of digital input position bits from 12 to 16 for both the forward and reverse ramp simulations. With the same number of multiplier bits and the same update rate, the increase in the number of the digital input position bits slightly improves the average errors and average error magnitudes of the resolver emulator. This improvement becomes more noticeable as the number of multiplier bits is increased; this is because the errors caused by a low number of multiplier bits dominate the total errors in the resolver emulator. By selecting the number of multiplier bits as 14, the errors caused by the number of multiplier bits no longer dominate the total errors in the resolver emulator. Note that the maximum and minimum ranges also become narrower as the number of digital input position bits increases from 12 to 16.

Next, the total errors are observed with the focus on the number of multiplier bits to see how the number of multiplier bits should be chosen. As shown in Figures 4.16 and 4.18, with the same number of digital input bits and the same update rate, an increase in the number of the multiplier bits significantly reduces the average and range of errors. Figure 4.17 and Figure 4.19 show a sharp drop in average error magnitude especially when the number of multiplier bits increases from 8 bits to 12 bits. The average error and the average error magnitude include accumulated effects from the number of digital input position bits; the results show that an increase of the number of multiplier bits is most beneficial in reducing both the average error and the average error magnitude when the number of digital input bits is high. In fact, as the number of multiplier bits is increased from 8 bits to 14 bits, the average error magnitudes are reduced approximately 500% when the number of digital input bits is 12, and 1500% when the number of digital input position bits is 16. Similarly, the average errors are reduced sharply by 500% and 1200% when the number of digital input position bits are 12 and 16, respectively.

Finally, the total errors are observed with the focus on the update rate to give insight on how the update rate should be chosen. The contribution of the errors from the finite update rate on the total errors depends on a couple of factors. The first factor is the accumulated errors from the quantization error in digital input position and the quantization error in the sine and cosine representations. The other factor is the rate at which the shaft rotates.

Using the same number of digital input position bits and the same number of multiplier bits, the effect of the update rate can be seen only when the number of digital

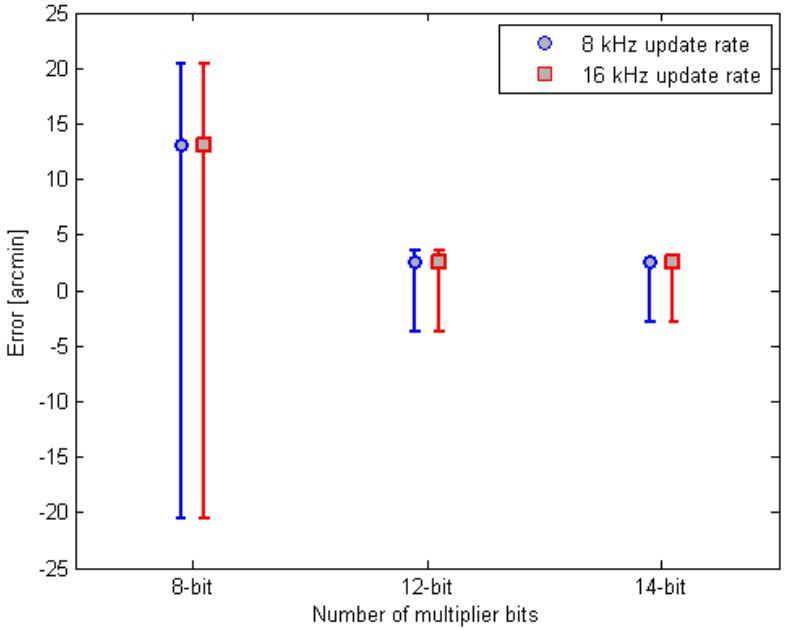
input position bits and multiplier bits are sufficiently large, as seen on the average error magnitude plots in Figures 4.17 and 4.19. The average and range of errors also become smaller as shown in Figures 4.16 and 4.18. The total errors for 8k updates per second are shown in blue, and the total errors for 16k updates per second are shown in red in these figures. To further examine the total error, consider the resolver emulator with 12 bits of digital input position and eight multiplier bits. Note that both the average error and the average error magnitude are relatively large, and do not vary with the update rate. This is due to the fact that the error caused by the low number of multiplier bits is large compared to the error caused by the update rate. Therefore, the update error is masked. Now consider the resolver emulator with 16 bits of digital input position and 14 multiplier bits. The errors accumulated from the quantizations no longer dominate the total errors. In fact, with 16 bits of digital input position, as the number of multiplier bits is increased from 8 to 14 bits, the update rate becomes more significant. Increasing the update rate from 8k updates per second to 16k updates per second, the average errors and the average error magnitudes are reduced by approximately 1%, 187%, and 215% for a number of multiplier bits of 8, 12, and 14 bits, respectively. Thus, in resolver emulator with 16 bits or more of digital input position and 12 bits or more of multiplier the update rate plays a significant role in the resolver emulator's output accuracy.

In conclusion, an increase in the number of digital input position bits, the number of multiplier bits and the update rate results in the reduction of total error, which leads to higher accuracy of the resolver emulator outputs. The three design parameters should be upgraded together, and commensurately. The resolver emulator with 16 digital input bits position, 14 multiplier bits, and an update rate of 16k updates per second produces the

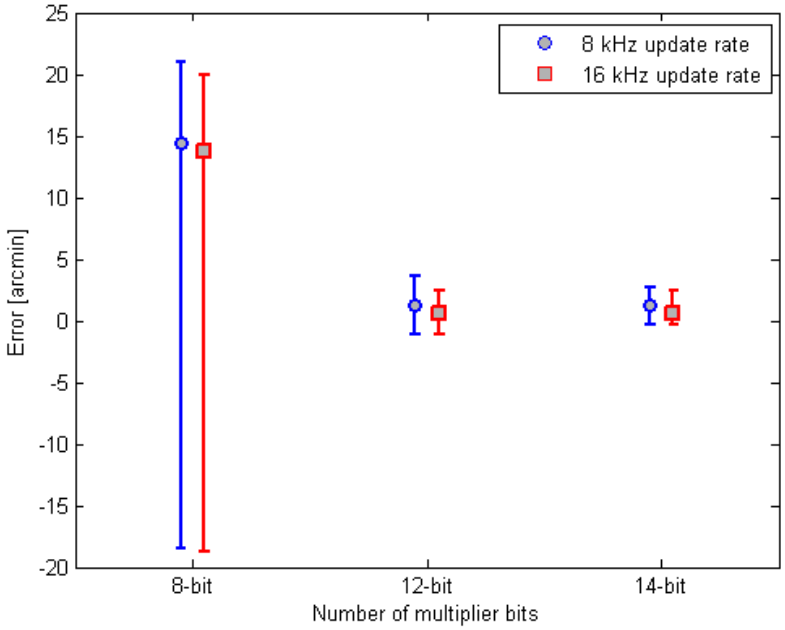
smallest total errors; for this design, the average error is 0.59 arcminutes or less, and the average error magnitude is 1.89 arcminutes or less for both forward and reverse ramp simulations. These two error measures indicate the overall output accuracy that can be expected when using the resolver emulator with this combination of design parameters. The absolute range of worst case error is from 2.37 to −1.61 arcminutes for this combination of design parameters.

4.7 Summary

Three main design parameters and the sources of error of the resolver emulator are discussed in this chapter. Forward ramp and reverse ramp simulations are conducted to study the effects of each parameter on the output accuracy. In Chapter V, the resolver emulator is implemented on an experimental test bench. The same design parameter values from this chapter are used. A comparison between the simulations and the bench tests is made; the bench test results differ from the simulation results in that non-idealities such as environmental noise are introduced.

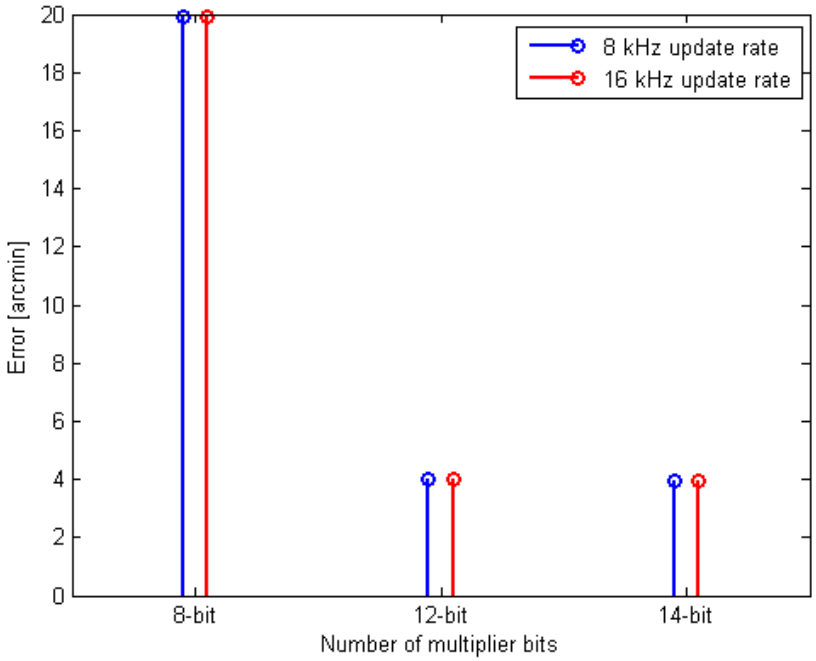


(a) Resolver emulators using a 12-bit digital input position

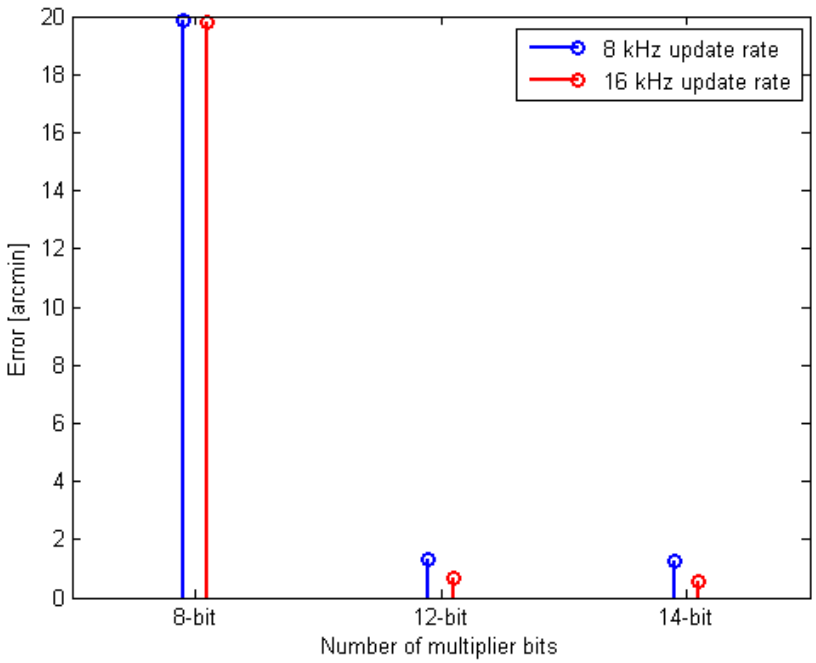


(b) Resolver emulators using a 16-bit digital input position

Figure 4.16: Average and range of total error for the forward ramp simulation

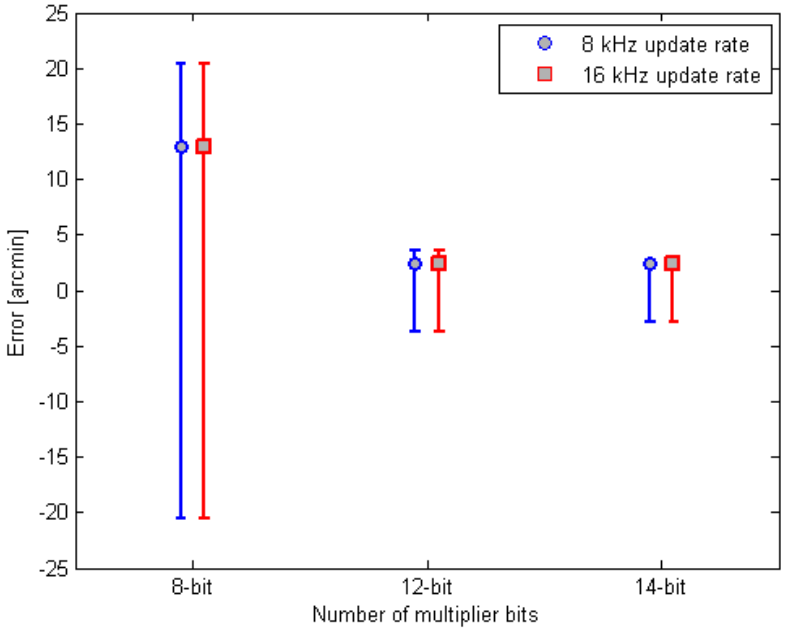


(a) Resolver emulators using a 12-bit digital input position

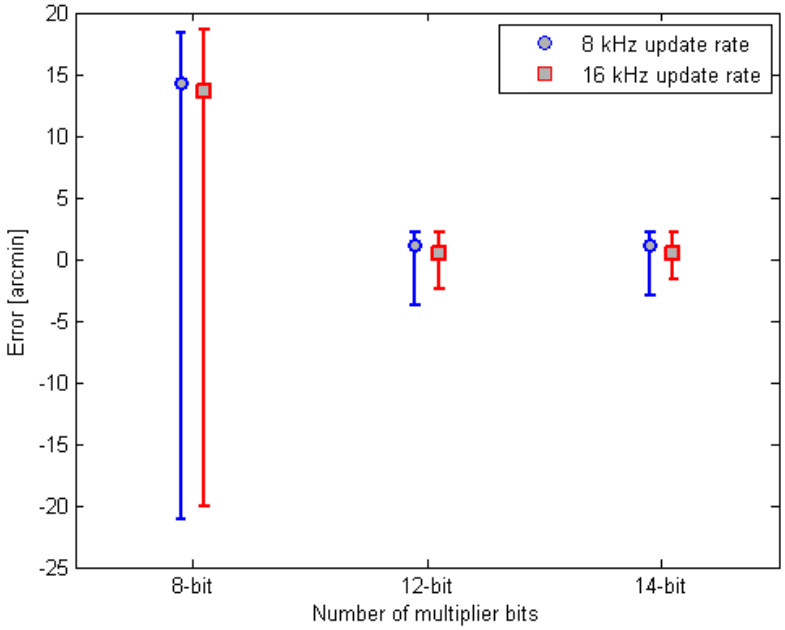


(b) Resolver emulators using a 16-bit digital input position

Figure 4.17: Average magnitude of total error for the forward ramp simulation

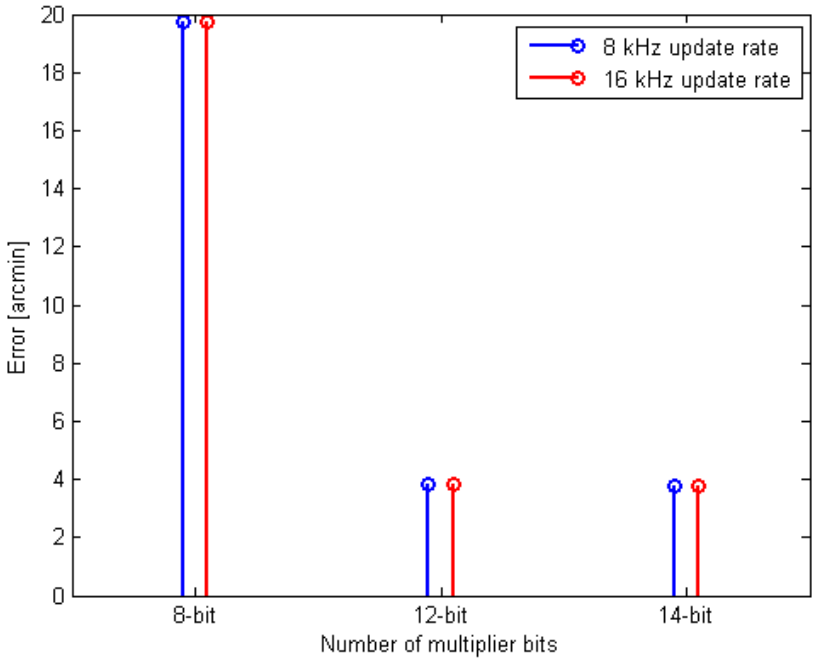


(a) Resolver emulators using a 12-bit digital input position

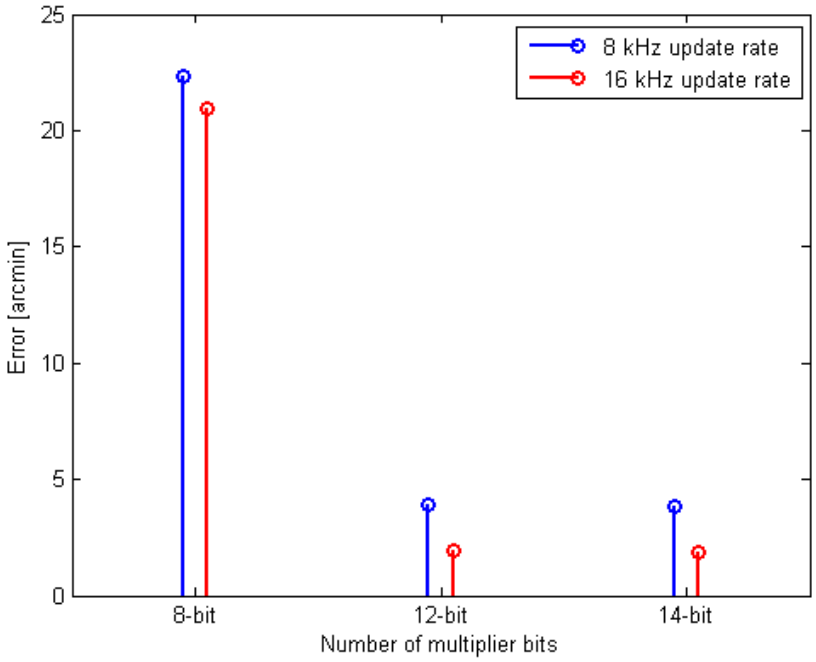


(b) Resolver emulators using a 16-bit digital input position

Figure 4.18: Average and range of total error for the reverse ramp simulation



(a) Resolver emulators using a 12-bit digital input position



(b) Resolver emulators using a 16-bit digital input position

Figure 4.19: Average magnitude of total error for the reverse ramp simulation

CHAPTER V

TEST RESULTS FOR THE IMPLEMENTED RESOLVER EMULATOR

The previous chapter used simulations of the proposed resolver emulator architecture to explore the impact of various design parameters on the accuracy of the output. It also developed an analysis of the effects of the design parameters and the sources of error. All components were assumed to be ideal; noise and imperfections that arise in the actual physical implementation were not considered. This chapter focuses on the implementation of the resolver emulator on an experimental test bench. The test bench setup and testing procedures are presented. An analysis considers how component non-idealities affect the implemented resolver emulator's output accuracy; the output settling time for the resolver emulator conversion is also measured. Factors such as limitations on memory size and clock frequency, mismatch error, offset error, and amplification error affect the performance. Finally, a comparison of the simulation results and the experimental test bench results is made to evaluate the differences between the two.

5.1 The implemented resolver emulator

Figure 5.1 shows a block diagram of the resolver emulator as implemented on the test bench for experimental evaluation. The architecture described and analyzed in Chapter IV is implemented using a PIC24FJ128GA010 microcontroller from Microchip

[21] to look up sine and cosine values corresponding to the digital input position code. These values are communicated digitally at a fixed update rate and with a fixed number of bits of precision to an AD5545 multiplying digital-to-analog converter (mDAC) from Analog Devices [22]; the mDAC converts the digital sine and cosine values to analog voltages, and multiplies them by the reference carrier. Several components not simulated in Chapter IV but needed in a practical implementation are used to process the single-ended currents that are produced by the mDAC. The two output channels of the mDAC, one for the sine-modulated carrier and one for the cosine-modulated carrier, produce their outputs in the form of current. Thus, each channel needs a current-to-voltage converter to obtain the resolver emulator signals. The current-to-voltage converters are realized with operational amplifiers; the OPA4277 from Texas Instruments [23] was chosen. The results of the conversion are two single-ended resolver emulator signals corresponding to the sine and cosine outputs. Passive low-pass filters with a cutoff frequency of 16 kHz are used to filter out high frequency noise components. Finally, the single-ended resolver emulator signals are passed through single-ended to differential converters to obtain the differential sine and cosine resolver emulator signals, S1, S3, S2, and S4; the converters chosen were the LME49724 from National Semiconductor [24], which have the ability to provide 2 mA rms of current.

A timing diagram of the communications between the microcontroller and the mDAC is shown in Figure 5.2. The four digital signals are the clock (CLK), the serial data in (SDI), the active-low chip select (CSBAR), and the active-low dual channel simultaneous update (LDACBAR). The operations for the communication between the microcontroller and the mDAC are as follows. Communicating a digital multiplication

value for one of the two channels in the mDAC takes a total of 18 clock cycles; the microprocessor initiates communications by bringing the CSBAR line low and providing the clock signal. In each clock cycle, the microcontroller provides one bit of data on the SDI line, the first two bits are an address indicating the channel, and the next 16 bits are the multiplication value for that channel, presented most significant bit first. Following the 18 clocks, the microcontroller activates the LDACBAR for a period of around 20 ns to complete the communication. The mDAC can receive new communications packets, each one updating the multiplier for one channel, at a rate of 2M updates per second [22].

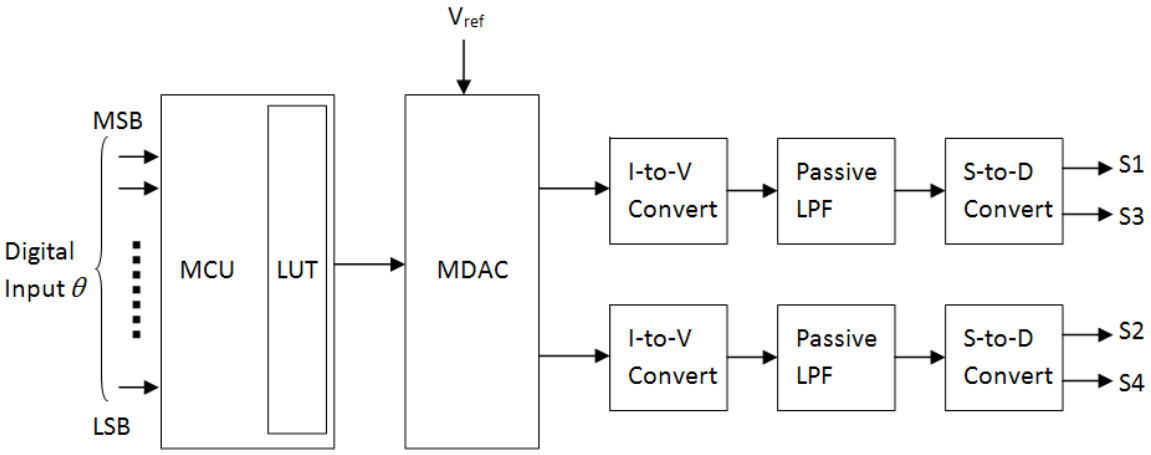


Figure 5.1: The implemented resolver emulator

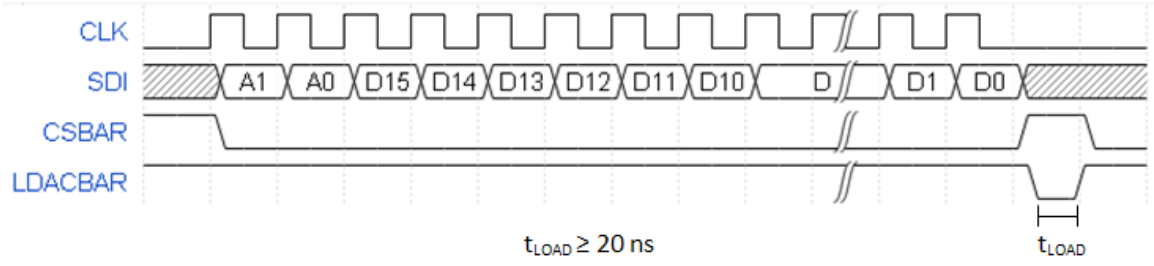


Figure 5.2: Timing diagram of the communications between the microcontroller and the multiplying digital-to-analog converter

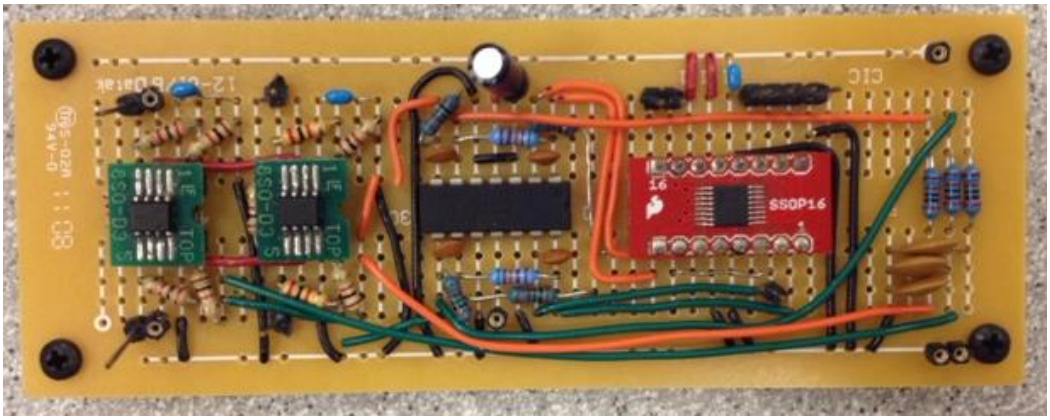


Figure 5.3: Implemented resolver emulator

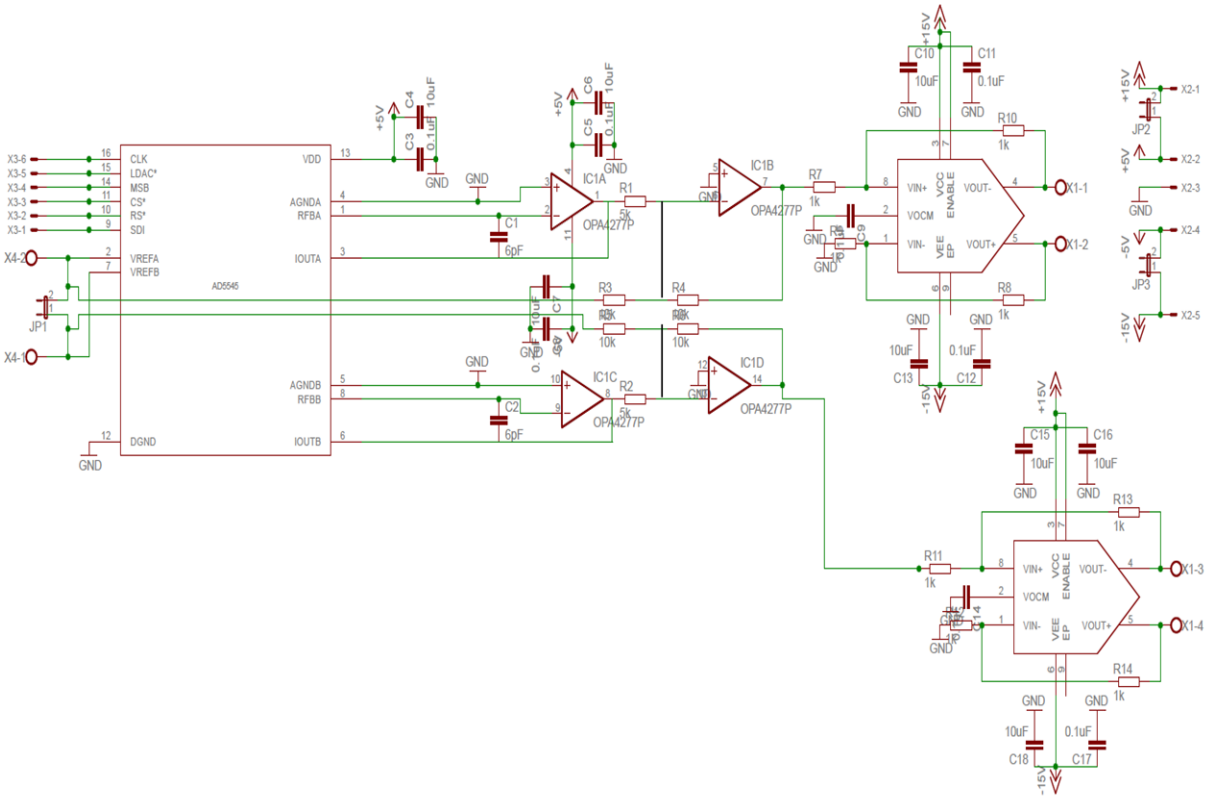


Figure 5.4: Resolver emulator circuit diagram

Figure 5.3 shows a photograph of the resolver emulator as implemented on a soldered protoboard and Figure 5.4 shows the circuit diagram for the resolver emulator. There are five main supply voltages for the components in the resolver emulator: +3.3 V, +5 V, −5 V, +15 V, and −15 V. The microcontroller requires a 3.3 V supply. The mDAC requires a 5 V supply. The operational amplifiers used for current-to-voltage conversion require both +5 V and −5 V. Finally, the single-ended to differential converters also require +6 V and −6 V. Note that the supply voltages of the single-ended to differential converter can be increased to ±18 V to allow for higher peak-to-peak output voltages. All of the voltages are produced using external switching power supplies that give particular dc voltages.

Unlike the simulated resolver emulator, the resolver emulator on the experimental test bench is prone to environmental noise. Sources of environment noise include motor noise, background RF noise, thermal noise, power supply noise, transient noise, and crosstalk. Many noise reduction techniques are used to mitigate the effects of noise on the resolver emulator outputs. The analog and digital ground planes were separated to avoid ground loops and digital ground bounce, and passive RC low-pass filters were put on the single-ended resolver outputs to reduce the high frequency ripple on the supply voltage produced by the digital power supply used in the experiments. Wires used were as direct and short as possible. The passive RC low-pass filters have a resistor value of 2 kΩ, a capacitor value of 5 nF, and a cut-off frequency of 16 kHz. Furthermore, a 0.1 μF ceramic capacitor and a 10 μF tantalum capacitor were added as bypass capacitors between each supply voltage and analog ground to minimize any transient disturbance and ripple.

An additional source of error in the implemented resolver emulator comes from the inclusion of circuitry on the backend of the design, needed to interface the resolver emulator to a motor controller. It is not possible to build current-to-voltage converters and single-ended to differential converters for the sine and cosine channels that are exactly the same; mismatch in the discrete components used for the two channels is inevitable. In constructing the circuitry for the implemented resolver emulator, resistors with a precision of ±5% and capacitors with a precision of ±5% were used. The result of the mismatch is additional error in the shaft angle indicated by the resolver emulator outputs.

5.2 Experimental test bench setup

In order to determine the output accuracy of the resolver emulator, the resolver emulator is put under test. Figure 5.5 shows the test set up for the resolver emulator. A data acquisition system (DAQ), the USB-6361 from National Instruments [25], is used to generate input signals for the resolver emulator, as well as to capture the output signals from the resolver emulator. The DAQ is controlled using LabVIEW Virtual Instrument software. The reference carrier, which has a frequency of 2 kHz and an amplitude of 5 V peak-to-peak, is generated by the DAQ. The DAQ is also used to emulate a position encoder; it provides digital positions to the resolver emulator via a parallel connection using a set of digital output pins of 16-bit resolution at a rate of 1M updates per second to emulate the digital position from an encoder. On the other end, the single-ended resolver emulator signals are sampled at a rate of 200k samples per second by the DAQ; the process includes some noise filtering. The singled-ended resolver emulator outputs are

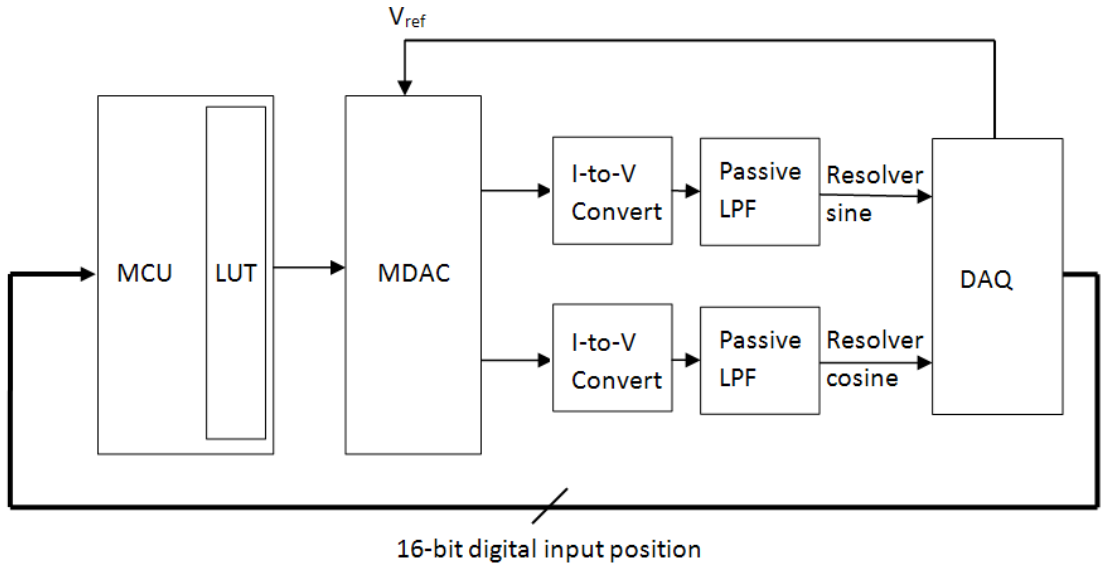


Figure 5.5: Experimental test bench setup for the implemented resolver emulator circuit

then recorded by the DAQ in a file containing comma-separated values. Instead of sampling the differential resolver emulator signals, the DAQ on the test bench samples the single- ended resolver emulator signals because the differential resolver emulator was not part of the analysis for the resolver emulator. Furthermore, the differential resolver emulator signals are intended to be used for a medium to long distance communication of over a meter [26]. The test bench is setup in such a way that the DAQ is adjacent to the resolver emulator.

The sampled resolver emulator outputs are processed in software, written as a LabVIEW Virtual Instrument, to compute the corresponding shaft angle. The algorithm for deriving the angle represented by the sine and cosine outputs from the sampled versions of the sine and cosine signals that are collected by the experimental test bench is as follows. The resolver emulator sine and resolver cosine signal, as well as the reference

carrier, are sampled at a discrete time interval of 5 microseconds. From these samples, the peak samples are identified and used for reconstructing the shaft angle. Because the signals are at a 2 kHz frequency, there are at least 4000 peak samples per second. Sampling each signal near its peak helps to ensure accuracy in the computations used to reconstruct the shaft angle. The peak samples of the reference carrier, the resolver emulator sine, and the resolver emulator cosine from the same time instant are used to reconstruct the shaft angle. Recall from Chapter II, the algorithm for determining the shaft position from the resolver emulator outputs is

$$\theta' = \tan^{-1}\left(\frac{V_s(\theta)}{V_c(\theta)}\right), \tag{2.5}$$

where V_s is the resolver emulator sine, V_c is the resolver emulator cosine, and θ' indicates the shaft angle with undetermined quadrant. Table 2.1 is used to determine the correct quadrant of the shaft angle. The reconstructed shaft angles are then plotted and stored in file of comma-separated values along with the sampled resolver emulator outputs. Finally, the difference between the actual shaft positions and the shaft position reconstructed from the resolver emulator outputs is calculated to obtain the total errors for the resolver emulator.

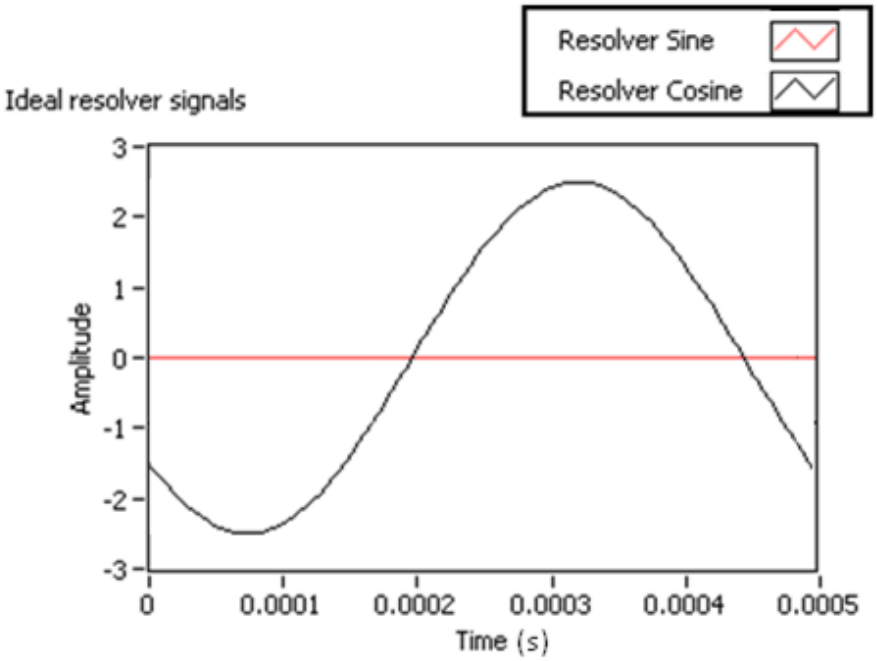
The tests are done emulating a stationary shaft. Like the simulations in Chapter IV, two types of errors are measured, the average error and the average error magnitude. Use of a stationary shaft helps in assessing the effects of noise on the bench, as noise is then the only factor that will cause fluctuation in the reconstructed shaft angle. The tests are performed multiple times for each stationary shaft position in order to ensure the

correctness of the results. Each time, the experimental test bench is setup to run over 1000 cycles in a LabVIEW Virtual Instrument, where each cycle captured over 4000 peak samples. Furthermore, the DAQ is set to sample at a rate of 100 times faster than the reference carrier frequency used for the bench test. The averages are taken from a large number of samples. Therefore, the average errors and the average error magnitudes measured on the bench at the discrete time intervals used by the DAQ are close to the true values in continuous time.

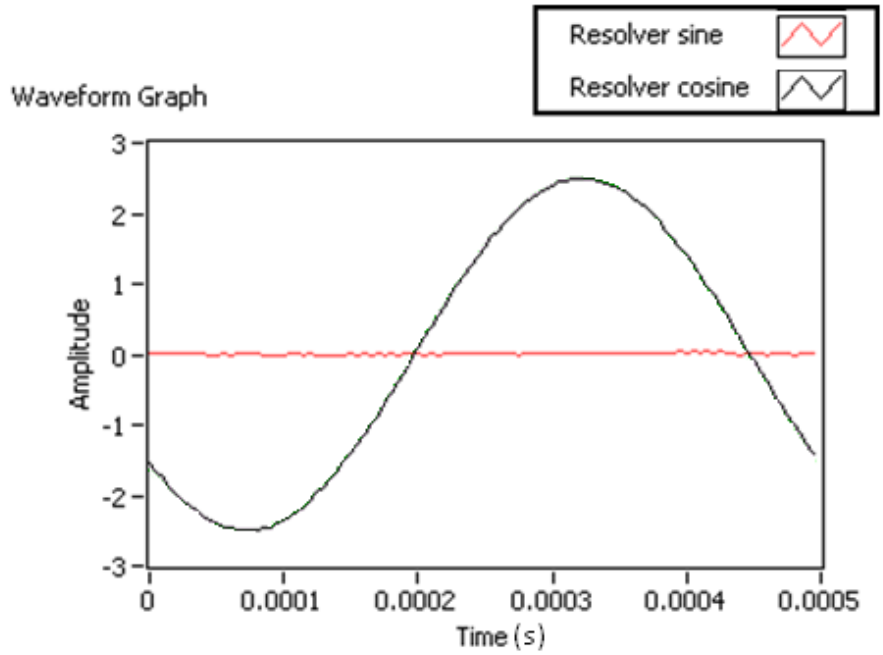
5.3 Test to determine baseline noise and dc offset

The first test on the bench was designed to establish the baseline level of the noise and the dc offset in the implemented resolver emulator. This was done by emulating a stationary position of zero degrees. Figure 5.6 shows the ideal and experimental outputs of the resolver emulator for a stationary shaft position of zero degrees. At zero degrees shaft position, the resolver emulator cosine is ideally equal to the reference carrier, and the resolver emulator sine is ideally zero. The ideal resolver emulator outputs, shown in Figure 5.6(a), are smooth. However, the actual resolver emulator outputs, shown in Figure 5.6(b), are corrupted with a small amount of noise and dc offset. The unevenness and a small amount of positive dc offset can be seen on the resolver emulator sine signal.

Although inclusion of the passive low-pass filters with a cut-off frequency of 16 kHz helped in mitigating high frequency noise, the resolver emulator outputs still contain noise. In particular, the predominant noise had a center frequency of 12 MHz, a maximum peak-to-peak voltage of approximately 150 mV, and a zero average. This environment noise is present in the surroundings and corrupts the resolver emulator



(a) Ideal resolver emulator outputs



(b) Actual resolver emulator outputs

Figure 5.6: The resolver emulator outputs for a shaft position of 0°

signals; presumably, the noise enters the system after the low-pass filter and prior to entering the DAQ. The maximum peak-to-peak value of this noise is 3% of the maximum peak-to-peak value of the resolver emulator outputs. The predominant noise is most likely RF noise based on its frequency characteristic. No special shielding was used in the prototype resolver emulator circuit built for the experimental test bench; this noise could be mitigated with appropriate packaging. A small amount of dc offset of approximately 100 mV is measured in this test. The sources of the dc offset are the mismatch and inaccuracy in the discrete components used in the current-to-voltage converter and the single-ended to differential converter.

It is important to note that noise is in addition to the errors that stemmed from the design that were analyzed in Chapter IV. In the worst case, the errors due to noise and the errors stemming from the design act in the same direction, and the total error in the resolver emulator outputs is the sum of the two. Therefore, the resolver emulator is expected to have lower output accuracy in bench testing than it did in the simulation results of Chapter IV.

5.4 Tests of output accuracy

The resolver emulator was tested on the bench for steady-state situations, that is, ones in which the digital input position code does not change. These tests give insight into the output accuracy of the resolver emulator outputs compared to the known actual shaft position. Note that because the shaft position does not change from one digital input position code to another, the update rate does not affect the output accuracy of the resolver emulator for this test. The DAQ is used to generate the digital input position

code that emulates a non-moving shaft that is set at a particular angle. The arbitrarily chosen positions for the tests are 125° and 310°.

The first shaft position tested is 125°, which lies in the second quadrant. This shaft position cannot be exactly represented by the digital input position code. The closest digital input position code is 1422 when the number of bits in the digital input position code is 12 bits and 22756 when the number of bits in the digital input position code is 16 bits. The corresponding quantization errors for the shaft position are −1.17 arcminutes and 0.146 arcminutes for the 12-bit and 16-bit digital input position codes, respectively. The precision with which the design can control the amplitudes of the resolver emulator outputs will be displayed in the test. Also, any noise that affects the amplitudes of the resolver emulator outputs on the bench has the potential to play a significant role in determining the final output accuracy of the resolver emulator.

Table 5.1 summarizes the bench results for a digital input position code corresponding to a 125° shaft position for the resolver emulator. The bench test was conducted for the twelve different variations on the resolver emulator architecture, each using a different combination of the design parameters (number of bits of digital input position code, number of multiplier bits, and update rate). The microcontroller is manually reprogrammed with the values for the design parameters to be tested.

In order to analyze the difference between the simulation results and the bench test results, the sources of noise that cause error in the resolver emulator outputs are considered, for the resolver emulator variations with 16 digital input position bits, and 14 multiplier bits. As mentioned earlier, one of the noises that affects the resolver emulator

outputs is a noise with a center frequency of 12 MHz, a maximum peak-to-peak value of 150 mV, and a zero average. This 12 MHz noise affects the range of error seen but not the average error. To see the effect of this noise on the resolver emulator outputs when the shaft position is 125°, the amplitudes of the resolver emulator sine and cosine outputs can be calculated using (2.2) and (2.3). For the reference carrier used on the bench, which has a frequency of 2 kHz and an amplitude of 2.5 V, the resolver emulator sine is expected to have an amplitude of 2.63 V and be 180° out of phase with the reference carrier, and the resolver emulator cosine is expected to have an amplitude of 1.84 V and be in phase with the reference carrier for the shaft position of 125°. Theoretically, if a noise of 75 mV amplitude is applied to both the resolver emulator sine and cosine signals after the low-pass filter in Figure 5.4, that noise can cause a range of errors in the shaft angle indicated by the resolver emulator outputs. The corrupted resolver emulator sine and cosine amplitudes are 2.56 V and 1.92 V when a positive noise voltage of 75 mV is applied, and 2.71 V and 1.77 V when a negative noise voltage of 75 mV is applied. As a result, the reconstructed shaft angle, found using equation (2.5) and Table 2.1, can range from 126.87° to 123.15°. Therefore, the error caused by the noise may be as big as ±1.86° or ±111 arcminutes. This error is consistent with the range of error seen experimentally as recorded in Table 5.1; the table shows errors of approximately 40 arcminutes above and below the average error.

The average error is not affected by the 12 MHz noise and can be compared directly to the simulation results. Because the test conducted in the chapter assumes the shaft does not move, the update rate does not affect the total error seen experimentally for the resolver emulator. After taking away the part of the total error that is due to the

Table 5.1: Summary of error from bench testing results for a shaft angle of 125°

Digital input bits	Multiplier bits	Update rate (updates/ sec)	Range of error (arcmin)	Average error (arcmin)	Average error magnitude (arcmin)
12-bit	8-bit	8k	(-154.39, -75.68)	-113.46	113.46
12-bit	8-bit	16k	(-158.62, -75.66)	-118.00	118.00
12-bit	12-bit	8k	(-83.43, 0.77)	-44.81	44.83
12-bit	12-bit	16k	(-80.24, 1.59)	-44.13	44.13
12-bit	14-bit	8k	(-81.87, -0.67)	-36.09	36.09
12-bit	14-bit	16k	(-79.62, 2.61)	-33.49	33.52
16-bit	8-bit	8k	(-158.26, -66.87)	-110.69	110.69
16-bit	8-bit	16k	(-155.61, -77.68)	-115.38	115.38
16-bit	12-bit	8k	(-80.56, 1.63)	-49.16	49.23
16-bit	12-bit	16k	(-82.74, 0.95)	-40.19	40.21
16-bit	14-bit	8k	(-81.26, -2.01)	-40.90	36.90
16-bit	14-bit	16k	(-80.63, -2.85)	-40.29	32.32

update rate from the simulation, simulations show an average error of 10.4 arcminutes and an average error magnitude of 17.21 arcminutes. The bench test result shows that for the stationary shaft at 125°, the average error magnitude is approximately 20 arcminutes larger than the simulation results. Mismatch and inaccuracy of discrete components in the current-to-voltage converter and the single-ended to differential converter are most likely the cause of these differences.

Next, a comparison is made by looking at the overall pattern of average errors and average error magnitudes when the design parameters are changed. The simulation results and the experimental test bench results show similar patterns in terms of average errors and average error magnitudes when the design parameters change. Recall the simulation results from Table 4.1, for which the average errors and the average error magnitudes show an improvement of nearly 500% when the number of multiplier bits increases from

8 to 12 bits. This significant improvement can also be observed in the bench test; a 350% improvement is seen for the same design parameter change, as shown in Figure 5.9 and Figure 5.11.

The second tested shaft position is 310°, which lies in the fourth quadrant. This shaft position is also not divisible by the quantization step sizes of the 12-bit or 16-bit digital input position; thus it cannot be exactly represented by the digital input position code. The closest digital input position code is 3527 for a 12-bit and 56434 for a 16-bit digital input position. The corresponding quantization errors for the shaft position are −0.59 arcminutes and −0.073 arcminutes for 12-bit and 16-bit digital input positions, respectively. Like the first tested shaft position, the precision with which the design can control the amplitudes and the noise that affect the outputs of the resolver emulator are the two major factors that affect the output accuracy in this test.

Table 5.2 summarizes the bench results for the 310° shaft position. Using (2.2), (2.3), and reference carrier used on the bench, which has a frequency of 2 kHz and an amplitude of 2.5 V, the ideal resolver emulator sine is expected to have an amplitude of 3.32 V and be in phase with the reference carrier, and the resolver emulator cosine is expected to have an amplitude of 2.78 V and be 180° out of phase with the reference carrier for the shaft position of 310°. In the presence of a high frequency noise with 75 mV amplitude that affects the sine and cosine equally, the corrupted resolver emulator sine and cosine amplitudes are expected to be 3.39 V and 2.71 V when positive 75 mV is applied, and 3.24 V and 2.86 V when negative 75 mV is applied. Accordingly, the reconstructed shaft angle is expected to vary between 308.6° and 311.4° in the presence

Table 5.2: Summary of error from bench testing results for a shaft angle of 310°

Digital input bits	Multiplier bits	Update rate (updates/ sec)	Range of error (arcmin)	Average error (arcmin)	Average error magnitude (arcmin)
12-bit	8-bit	8k	(-80.66, 3.80)	-40.43	40.79
12-bit	8-bit	16k	(-81.70, 7.41)	-31.54	39.92
12-bit	12-bit	8k	(-42.96, 46.36)	1.64	35.91
12-bit	12-bit	16k	(-42.37, 47.26)	6.60	35.80
12-bit	14-bit	8k	(-42.64, 45.09)	6.54	35.78
12-bit	14-bit	16k	(-42.74, 47.27)	5.70	35.66
16-bit	8-bit	8k	(-79.76, 7.77)	-40.01	41.13
16-bit	8-bit	16k	(-79.58, 6.35)	-28.94	40.86
16-bit	12-bit	8k	(-42.06, 44.42)	2.90	35.01
16-bit	12-bit	16k	(-40.13, 46.15)	5.30	35.13
16-bit	14-bit	8k	(-39.66, 44.94)	3.44	35.24
16-bit	14-bit	16k	(-45.73, 45.10)	1.11	34.86

of the noise. Therefore, the error caused by the noise may be as big as ±1.4° or ±83.87 arcminutes. This error is consistent with the range of error seen experimentally as recorded in Table 5.2; the table shows errors of approximately 40 arcminutes above and below the average error.

Again, the average error is compared directly to the simulation results. After taking away the part of the total error that is due to the update rate from the simulation, the simulations show an average error and an average error magnitude of 10.4 arcminutes and 17.21 arcminutes. For the stationary shaft at 310°, the average error magnitude is approximately 20 arcminutes larger than the simulation results due to mismatch and inaccuracy of discrete components in the current-to-voltage converter and the single-ended to differential converter.

The overall pattern of average errors and average error magnitudes when the design parameters are changed for the stationary shaft at 310° is also similar to the simulation with one exception. The average errors for the 8-bit multiplier are about 500% larger than 12-bit multiplier, as shown in Figure 5.8 and Figure 5.10. However, the average error magnitudes are approximately 30 to 40 arcminutes overall. This is due to higher peak-to-peak voltages of the resolver emulator outputs for the shaft position of 310° compared to the peak-to-peak voltages of the resolver emulator outputs for the shaft position of 125°. Thus, the resolver emulator outputs are relatively less sensitive to noise and mismatch.

5.5 Output settling time

A test was conducted to approximate the output settling time for the resolver emulator when there is a change in the digital input position code. Recalling the definition from Chapter III, the output settling time is the time that is required for the resolver emulator outputs to enter and remain in steady-state after a change in the digital input position. It is important to note that the output settling time largely depends on the time it takes for the microcontroller to look-up values, as well as the slew rate of the mDAC. The test measures the output settling time when the digital input position code changes by only a few subsequent codes. This is the kind of situation that will occur in practice, when the shaft is rotating slowly relative to the rate at which the position encoder provides updates of the digital input position code.

To approximate the output settling time of the resolver emulator, a starting position and a final position have to be selected. Unlike the outputs of a digital-to-analog

converter that settle to a final static analog voltage, the outputs of the resolver emulator instead settle to sinusoidal analog voltages with specific amplitude. The most straightforward way to approximate the output settling time of the resolver emulator is to select a final position for which the resolver emulator sine output amplitude is zero and a starting shaft position that is only a few degrees away from the final shaft position; then, the settling time can be measured as the time it takes for the resolver emulator sine output amplitude settle to with ten percent of its steady-state value of zero. The test uses the DAQ to emulate a step change in the digital input position corresponding to a step change in the shaft position from five degrees of zero degrees. Figure 5.7 shows the resolver emulator sine output and the reference carrier when the shaft angle changes from five degrees to zero degrees. Unlike the previous sections in this chapter where the resolver emulator outputs are captured using the DAQ, the resolver emulator outputs in this plot are captured using an oscilloscope, which does not have an internal low-pass filter. Thus the noise seen on this plot is larger than the noise analyzed previously.

The output settling time is determined by approximating the time from the instant at which the digital input position bits change and the time at which the resolver emulator sine outputs settle to zero. The output settling time includes both response time and slew time. For this resolver emulator architecture, the response time includes the time the microcontroller takes to read the digital input position, the time to look up the sine and cosine representations, and the time to send out the multiplier bits via serial communication to the mDAC. Further, the response time includes the time it takes for the mDAC to modulate the sine and cosine representations with the reference carrier. The response time is measured by observing the time it takes for the resolver emulator sine

output to start changing in response to the digital input position change; the response time could only be measured approximately because of the noise. Slew time is the time for the resolver emulator outputs to slew to their final analog voltages. The slew time typically depends on the settling time of the mDAC. The slew time is measured by observing the point at which the resolver emulator sine output settle to zero; again, this could only be measured approximately. Table 5.3 shows the measured times of the resolver emulator with 16 bits of digital input position, and 14 bits of multiplier, and an update rate of 16k updates per second.

5.6 Summary

This chapter presented the bench implementation of the resolver emulator. The setup and the testing procedure are presented. The input parameters for the bench result are identical to the input parameters of the simulation. The results of the experimental test bench are summarized in tables and plots; this is followed by discussions of the results and the noise sources, and a comparison of the test bench results and the simulation results. The performance of the twelve variations of the resolver emulator on the bench is consistent with the performance in simulation; however, the output accuracy is worse because of noise.

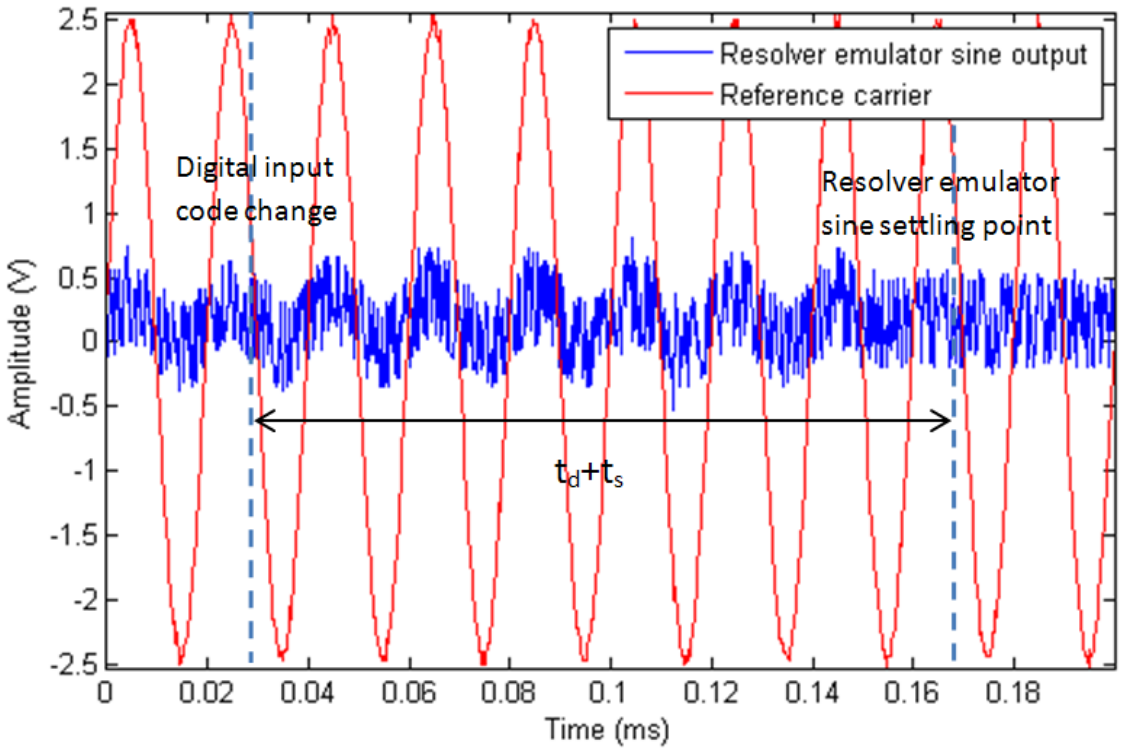
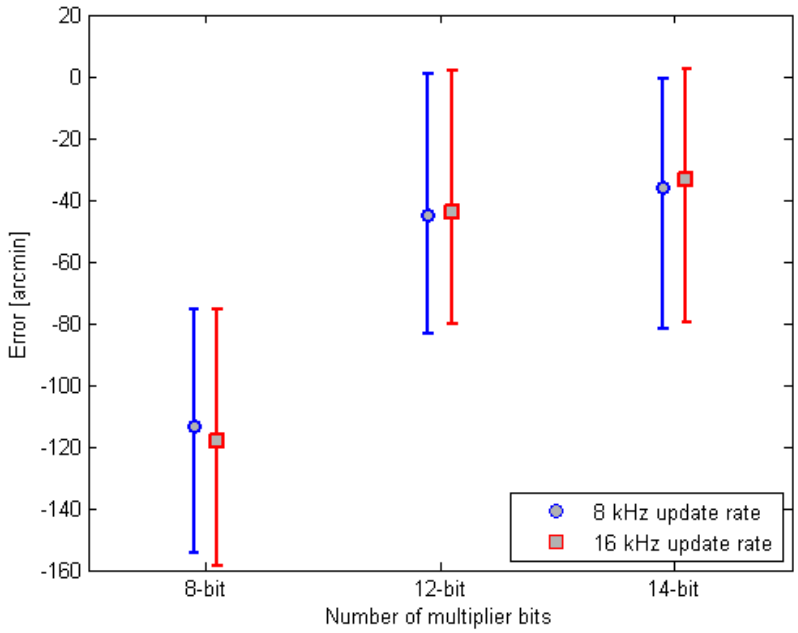


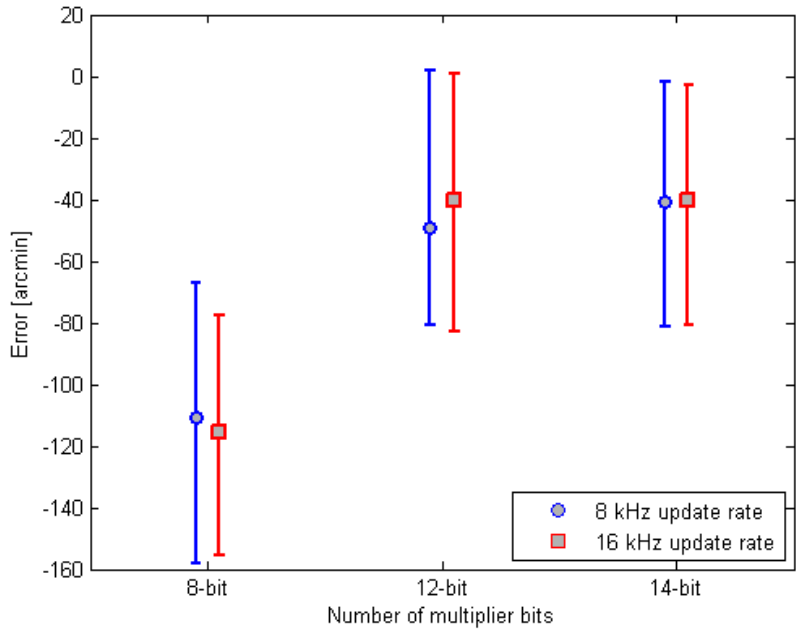
Figure 5.7: Resolver emulator output settling time

Table 5.3: Output settling time of the resolver emulator

	Time delay (µs)
Response time, t_d	114
Slew time, t_s	30.9
Output settling time, ($t_d + t_s$)	144.9

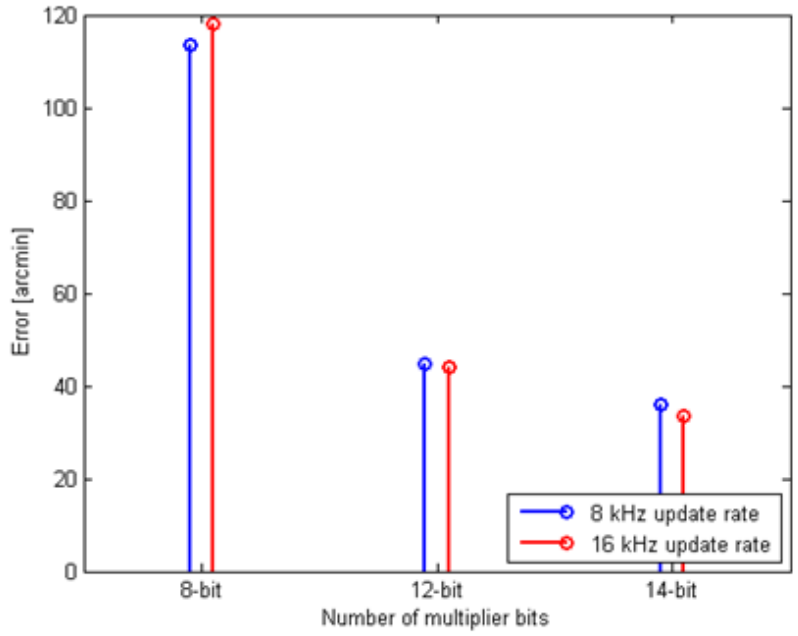


(a) Averages and ranges of errors for resolver emulators using a 12-bit digital input position

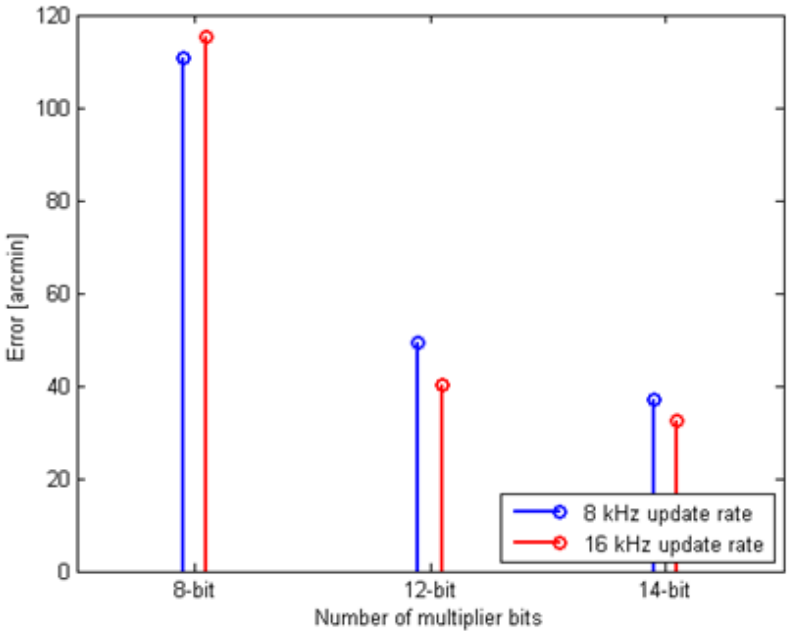


(b) Averages and ranges of errors for resolver emulators using a 16-bit digital input position

Figure 5.8: Comparison of average and ranges of errors for resolver emulator with the shaft position of 125°

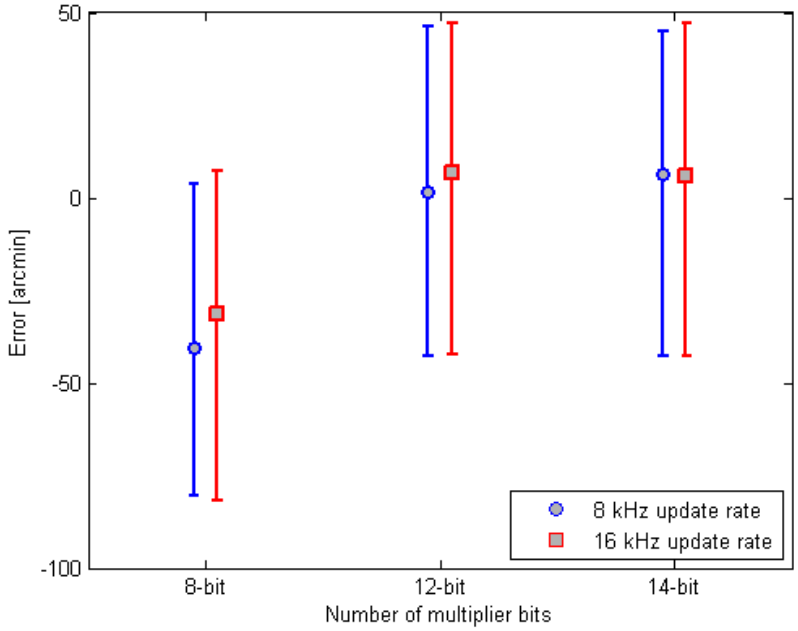


(a) Average error magnitudes for resolver emulators using a 12-bit digital input position

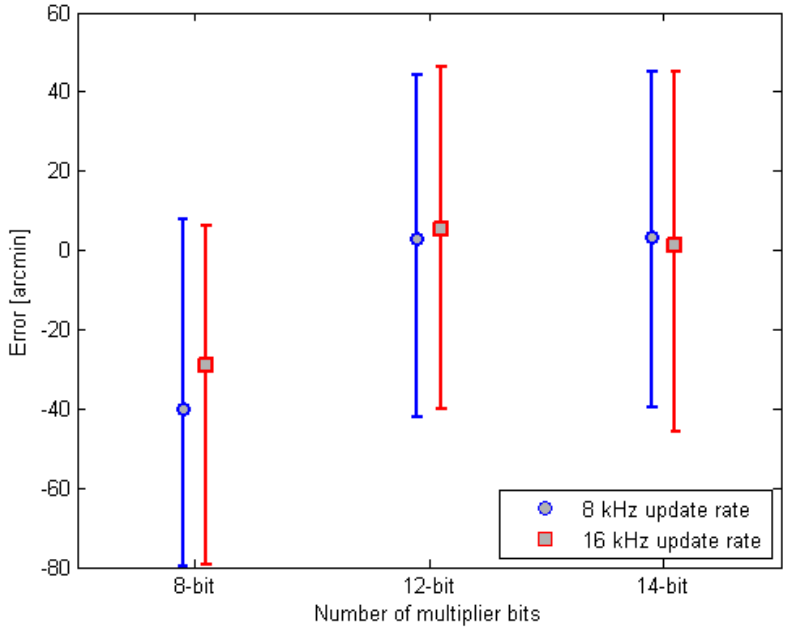


(b) Average error magnitudes for resolver emulators using a 16-bit digital input position

Figure 5.9: Comparison of average error magnitudes for resolver emulator with the shaft position of 125°

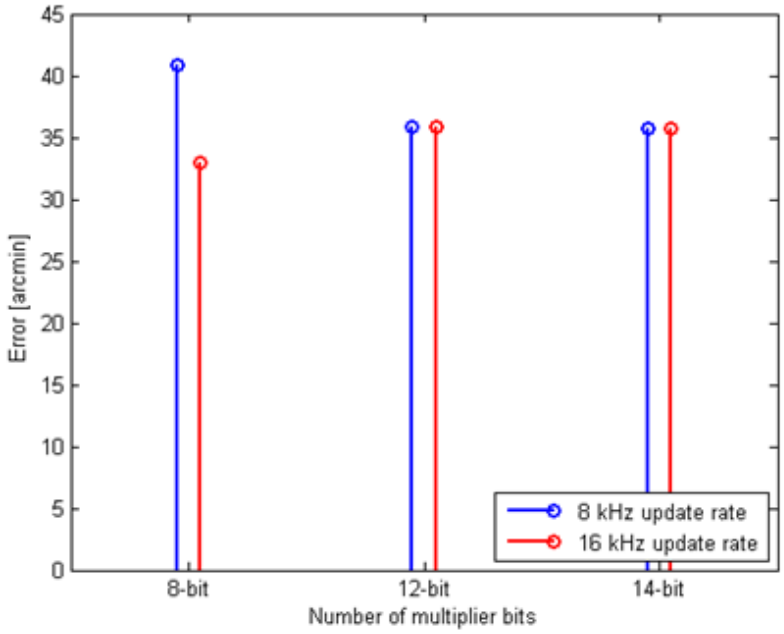


(a) Averages and ranges of errors for resolver emulators using a 12-bit digital input position

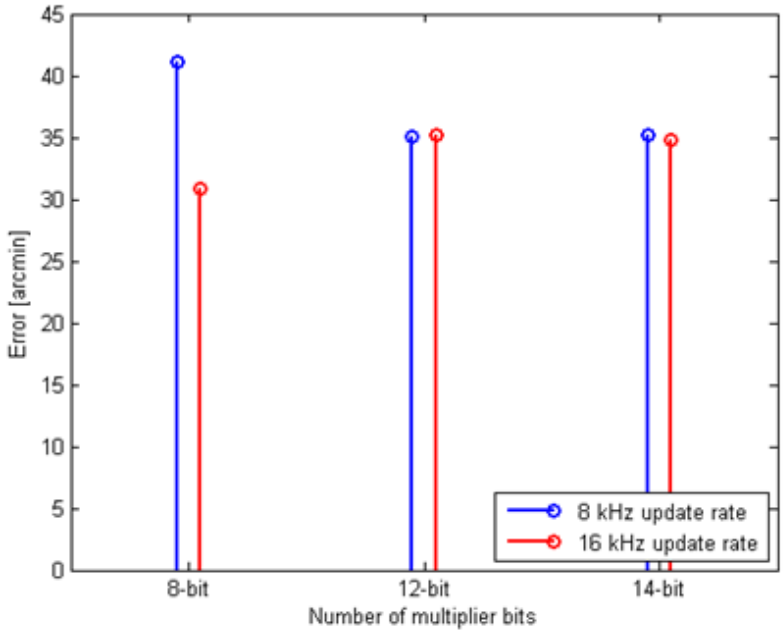


(b) Averages and ranges of errors for resolver emulators using a 16-bit digital input position

Figure 5.10: Comparison of average and ranges of errors for resolver emulator with the shaft position of 310°



(a) Average error magnitudes for resolver emulators using a 12-bit digital input position



(b) Average error magnitudes for resolver emulators using a 16-bit digital input position

Figure 5.11: Comparison of average error magnitudes for resolver emulator with the shaft position of 125°

CHAPTER VI

CONCLUSIONS

6.1 Summary

A low-cost high-accuracy resolver emulator is developed using an architecture based on a microcontroller and a multiplying digital-to-analog converter. The microcontroller is used to look up the sine and cosine representations of the digital input position. Values are then communicated to the mDAC, which uses them to modulate a reference carrier in order to produce the resolver emulator outputs.

The key design parameters for the resolver emulator architecture are the number of bits in the digital input position of the microcontroller, the number bits in the multiplier in the LUT, and the update rate of the mDAC. Each of the design parameter plays a role in determining total error in the outputs of the resolver emulator. The simulations of the resolver emulator show that it is important to ensure that sufficiently high number of multiplier bits is used in order to achieve high accuracy resolver emulator outputs. The number of digital input bits and the update rate can further be increased to obtain higher accuracy. In general, the update rate should be at least 360 times larger than the rate of shaft rotation to keep the error due to the update rate under one degree.

A bench test is conducted to gain further insights into the relationships between the key design parameters and the output accuracy of the resolver emulator in a non-ideal environment. Two additional sources of error that affect the resolver emulator outputs on the bench test are analyzed; an enviromental noise with a frequency of 12 MHz and a peak-to-peak voltage of 150 mV, and error coming from mismatch and inaccuracy of the discrete components in the current-to-voltage conversion and the single-ended to differential conversion. The RF noise is determined to cause about 111 arcminutes of error in the shaft angle indicated by the resolver emulator outputs, while mismatch and inaccuracy in the discrete components caused approximately 57 arcminutes of errors. These errors are added to the sources of error analyzed from the design parameters. The results from the test bench are proven to match the results from the simulation as well as the analysis. The bench test shows that an accuracy of within approximately 40 arcminutes was realized with the design parameters of 16 digital input bits, 14 multiplier bits, and 16 kHz update rate when environment noise of the type seen on the test bench is present, and for the discrete components used.

6.2 Recommendations

With the insight from simulations and bench test, a high accuracy resolver emulator can be developed, as long as the discrete components used in the architecture are sufficiently accurate. The errors come from inaccurate and mismatched discrete components in the current-to-voltage conversion and single-ended to differential conversion. Thus, special emphasis should be placed on the precision of the discrete components. An mDAC that can produce differential voltages directly can be used to

eliminate the need for external electronics built from discrete components on the backend of the resolver emulator, and therefore eliminate the possibility of mismatch. Specifically, use of an mDAC that produced differential voltages at its output would eliminate the need for the current-to-voltage converters and the single-ended to differential converters that contain the discrete components causing the mismatch error.

As observed during the bench testing, the number of multiplier bits and the update rate are limited by the internal memory and the execution time of the microcontroller, respectively. By taking advantage of the symmetry of the sine function, the space required for the lookup table is reduced to an eighth of the original space required for the full sine and cosine waves. However, the microcontroller's memory size is still insufficient to store the first quadrant of sine in the lookup table for when more than 14 multiplier bits are used. A more expensive microcontroller with a larger memory is needed to increase the number of multiplier bits. As for the update rate, the communication between the microcontroller and the mDAC can only go as fast as the execution time of the microcontroller allows. The use of a microcontroller that is designed to work with digital signals and has a higher number of instructions per clock cycle, such as a dsPIC, will enable the update rate to be higher.

Another way to improve the output accuracy of the resolver emulator is to implement the resolver emulator on a printed circuit board with a large ground plane. Redesign of the traces and the ground plane on the printed circuit board can reduce the environmental noise such as the 12 MHz RF noise seen in Chapter V.

6.3 Future research

Further improvements can be made to the resolver emulator. One improvement includes the ability to set the values of the design parameters of the resolver emulator either in software or hardware. In the current implementation, the microcontroller has to be reprogrammed manually every time a design parameter value changes; thus there is no option to change the design parameters without flashing the microcontroller's memory. Software setting of the design parameters can be implemented by having the resolver emulator perform an initialization at startup to select the desired design parameters using existing serial ports. This initialization can come from the position encoder, or from the system to which the resolver emulator is interfaced. Alternatively, hardware setting of design parameters can be implemented by using a set of jumpers on the printed circuit board to select the desired design parameters. Regardless of the method used for setting design parameters, extra system memory may be required to store look-up table data for all possible design parameter values.

A thorough study of the particular sources of noise affecting the outputs of the resolver emulator may reveal further insight into improvements that can mitigate those effects. As seen from the bench testing section in this thesis, the output accuracy of the resolver emulator is affected by the environment noise as well as error from mismatch and inaccuracy in the discrete circuit components. This makes the test bench results differ from the simulation results. More studies can be done on the topic, so that the designer can pinpoint specific details for improvement of the implemented resolver emulator circuit. An example improvement that might result would be the use of an optocoupler to

isolate the digital signals communicated between the encoder and the resolver emulator in order to reduce the effect of motor noise on the resolver emulator outputs when the motor is running.

Lastly, it is possible to implement the resolver emulator on a single integrated circuit. The design process will have to start at a much lower level than the current system level. A transistor-level schematic can be developed to convert the digital input position into analog voltages, as well as to modulate those voltages with a reference carrier to produce the outputs for the resolver emulator.

BIBLIOGRAPHY

[1] P. Novak, "Interface and Protocol for Industrial Sensors," *In XXVI ASR 2001 Seminar on Instruments and Control,* pp. 370-373, April 2001.

[2] iC-Haus, "BiSS Interface Protocol Description," [Online]. Available: http://www.ichaus.de/news/82. [Accessed December 2011].

[3] Heidenhain, EnDat 2.2 - Bidirectional Interface for Position Encoders, Heidenhain, September 2011.

[4] Sick|Stegmann, "Description of the Hiperface Interface," [Online]. Available: http://www.sick-automation.ru/images/File/pdf/Hyperface_e.pdf. [Accessed January 2012].

[5] Data Device Corporation, "DR-11525 16-bit High Frequency Hybrid Digital-to-Resolver Converter," 1999.

[6] National Instruments, "Encoder Measurements: How-To Guide," 2013.

[7] Optoresolver.com, "Tutorial: Resolver vs Encoder," [Online]. Available: http://www.optoresolver.com/help/tutorials/resolver_v_encoder.htm. [Accessed 18 February 2013].

[8] K. Bhattarai, "On the Use of Digital Communication Channel for Feedback in Position Control System," *Master Thesis,* 2012.

[9] J. Gasking, "Resolver-to-Digital Conversion - A Simple and Cost Effective Alternative to Optical Shaft Encoders," 1992.

[10] Dynapar, "Encoders and Resolvers: How to Choose the Right Feedback Option," [Retrived on January 2013].

[11] Piclist.com, "December 2003 MassMind Newsletter," [Online]. Available:

http://www.piclist.com/techref/new/letter/news0312.htm. [Accessed 18 February 2013].

[12] Data Device Corporation, "Synchro/Resolver Conversion Handbook," 1994.

[13] Axsys Technologies, "Pancake Resolvers Handbook," [Retrived on September 2012].

[14] North Atlantic Industries Inc, "Synchro and Resolver Conversion," 2006.

[15] W. Drury, Control Techniques' Drives & Controls Handbook, IEE Power & Energy Series, 35, 2001.

[16] Moog Components Group, "Synchro and Resolver Engineering Handbook," 2004.

[17] Data Device Corporation, "DRC-11522 Two-channel Digital-to-Resolver Converter," 1999.

[18] Data Device Corporation, "DRC-10520 High Power 16-bit D/R Converter," 1999.

[19] Digikey Corporation, "AD633JNZ-ND," [Online]. Available: http://www.digikey.com/product-detail/en/AD633JNZ/AD633JNZ-ND/750991. [Accessed 15 March 2013].

[20] Microchip Technology Inc, "AN655 D/A Conversion Using PWM and R-2R Ladders to Generate Sine and DTMF Waveforms," 1997.

[21] Microchip, "PIC24FJ128GA010 Family Data Sheet," 2007.

[22] Analog Devices, "AD5545 Dual, Current-Output, Serial-Input, 16-/14-Bit DACs," 2013.

[23] Burr-Brown Products, "OPA4277 High Precision Operational Amplifiers," 2005.

[24] National Semiconductor, "LME49724 High Performance, High Fidelity, Fully-Differential Audio Operational Amplifier," 2013.

[25] National Instruments, "NI USB-6361 X Series Data Acquisition," 2012.

[26] Advanced Micro Controls Inc, "What Is a Resolver?," [Online]. Available: http://www.amci.com/tutorials/tutorials-what-is-resolver.asp. [Accessed 18 February 2013].

[27] ICPE, "User Manual: The Resolver," [Retrived on September 2012].

[28] Cypress Perform, "AN2025 Analog - Sine Wave Generation with PSoC 1 (Demonstration with CTCSS)," 2007.

[29] Atmel Corporation, "AVR131: Using the AVR's High-speed PWM," 2003.

[30] G. K. McMillan, Process/industrial Instruments and Controls Handbook, McGRAW HILL, 1999.

[31] P. Horowitz and W. Hill, The Art of Electronics, Cambridge University Press, 1989.

[32] Maxim Integrated, "Effects of Digital Crosstalk in Data Converters," [Online]. Available: http://www.maximintegrated.com/app-notes/index.mvp/id/1761. [Accessed 30 July 2013].

-