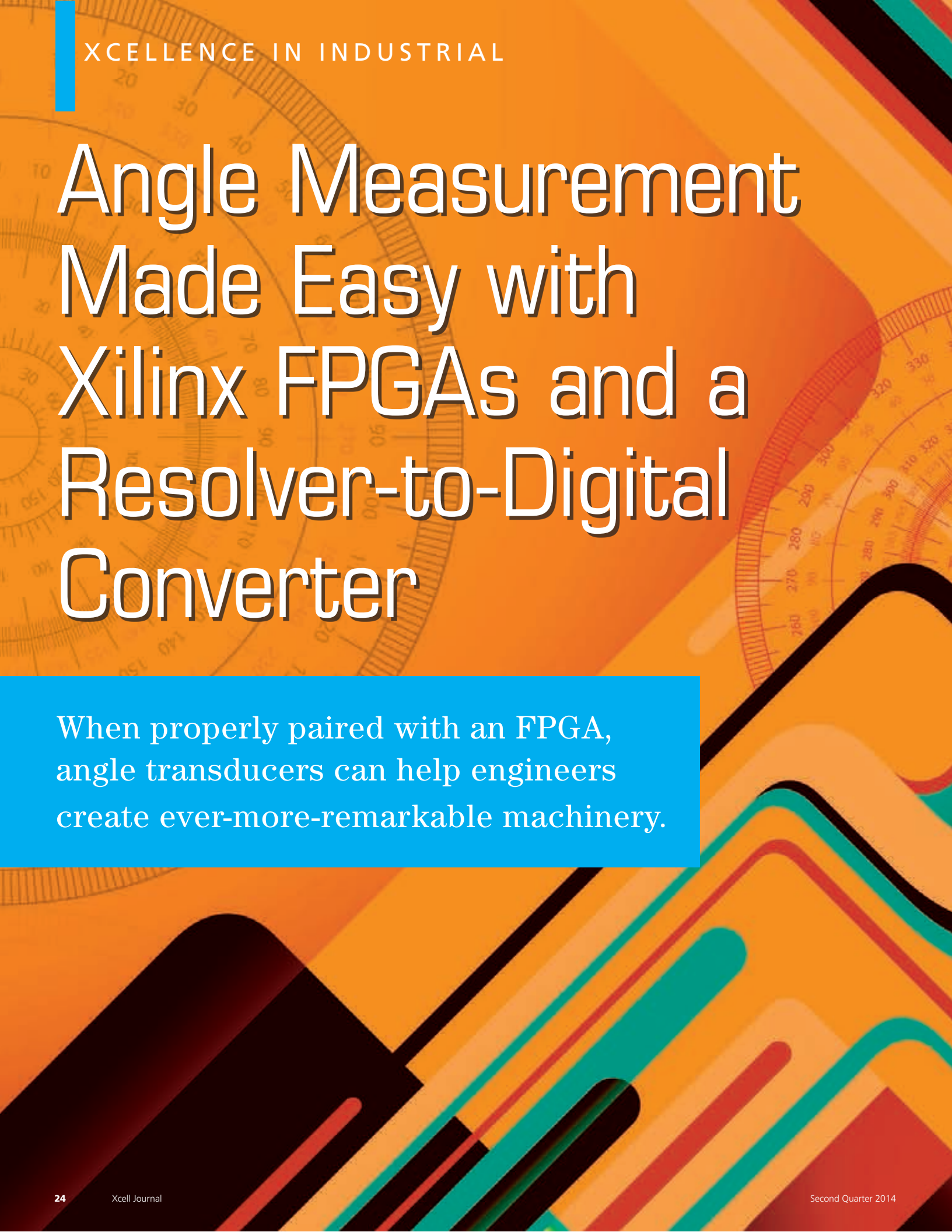
Angle Measurement Made Easy with Xilinx FPGAs and a Resolver-to-Digital Converter



When properly paired with an FPGA, angle transducers can help engineers create ever-more-remarkable machinery.

by N. N. Murty
Scientist "F"

S. B. Gayen
Scientist "F"

Manish Nalamwar
Scientist "D"
nalamwar.manishkumar@rcilab.in

K. Jhansi Lakshmi,
Technical Officer "C"
Radar Seeker Laboratory
Research Centre IMARAT
Defense Research Development Organization
Hyderabad, India

Ever since humans invented the wheel, we have wanted to know, with varying degrees of accuracy, how to make wheels turn more efficiently. Over the course of the last few centuries, scientists and engineers have studied and devised numerous ways to accomplish this goal, as the basic principles of the wheel-and-axle system have been applied to virtually every mechanical system, from cars to stereo knobs to cogs in all forms of machinery, to the humble wheelbarrow [1].

Over these many eras, it turns out the most essential element in making a wheel turn efficiently is not the wheel itself (why reinvent it?) but the shaft angle of the wheel. And the most effective way to measure and optimize a shaft angle today is through the use of angle transducers. There are many types of angle transducers that help optimize wheel efficiency through axle monitoring and refinements, but by applying FPGAs to the task you can achieve remarkable results and improve axle/wheel efficiencies in a broad number of applications.

Before we get into the details of how engineers are doing this optimally with Xilinx® FPGAs, let's briefly review some basic principles of angle transducers. Today there are two widely used varieties: encoders and resolvers.

TYPES OF ENCODERS AND RESOLVERS

Encoders fall into two basic categories: incremental and absolute. Incremental encoders monitor two positions on an axle and create an A or B pulse each time the axle passes those positions. A separate external electric counter then interprets those pulses for speed and rotational direction. Incremental counters are useful in a number of applications, but they do have some disadvantages. For example, when the axle is powered off, an incremental encoder must first calibrate itself by returning to a desig-

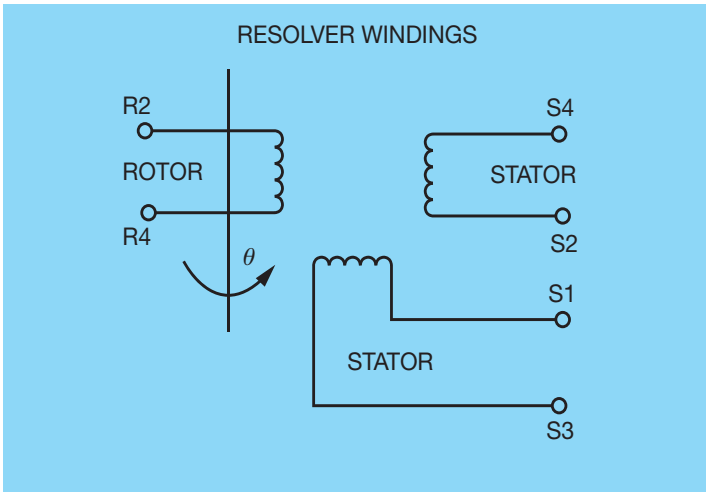


Figure 1 – Excitation to the rotor of a resolver

nated calibration point before beginning operation. Incremental counters are also susceptible to electrical interference, which can result in inaccuracies in pulses they send to the system and thus in rotation counts. Moreover, many incremental encoders are photoelectric devices, which precludes their use in radiation-hazardous areas, if that is a concern for your targeted application.

Absolute encoders are sensor systems that monitor the rotation count and direction of an axle. In an absolute-encoder-based system, users typically attach a wheel to an axle that has an electrical contact or photoelectric reference. When the axle is in operation, the absolute-encoder-based system records the rotation and direction of the operation and generates a parallel digital output that is easily translated into code, most commonly binary or Gray code. Absolute encoders are useful in that they need to be calibrated only once—typically in the factory—and not before every use. Moreover, they are typically more reliable than other encoders. That said, absolute encoders are typically expensive, and they are not great with parallel data transmission, especially if the encoder is located far away from the electronic system measuring its readings.

A resolver, for its part, is a rotary transformer—an analog device whose output voltage is uniquely related to the input shaft angle it is monitoring. It is an absolute-position transducer with 0° to 360° of rotation that connects directly to the axle and reports speed and positioning. Resolvers have a number of advantages over encoders. They are robust devices that can withstand harsh environments marked by dust, oil, temperature extremes, shock and radiation. Being a transformer, a resolver provides signal isolation and a natural common-mode rejection of electrical interference. In addition to these features, resolvers require only four wires for the angular data transmission, which suits them for everything from heavy manufacturing to miniature systems to those used in the aerospace industry.

A further refinement is the brushless resolver, which does not require slip-ring connections to the rotor. This type of resolver is therefore even more reliable and has a longer life cycle.

Resolvers use two methods to obtain output voltages related to the shaft angle. In the first method, the rotor winding, as shown in Figure 1, is excited by an alternating signal and the output is taken from the two stator windings. As the stator windings are mechanically positioned at right angles, the output signal amplitude is related by the trigonometric sine and cosine of the shaft angle. Both the

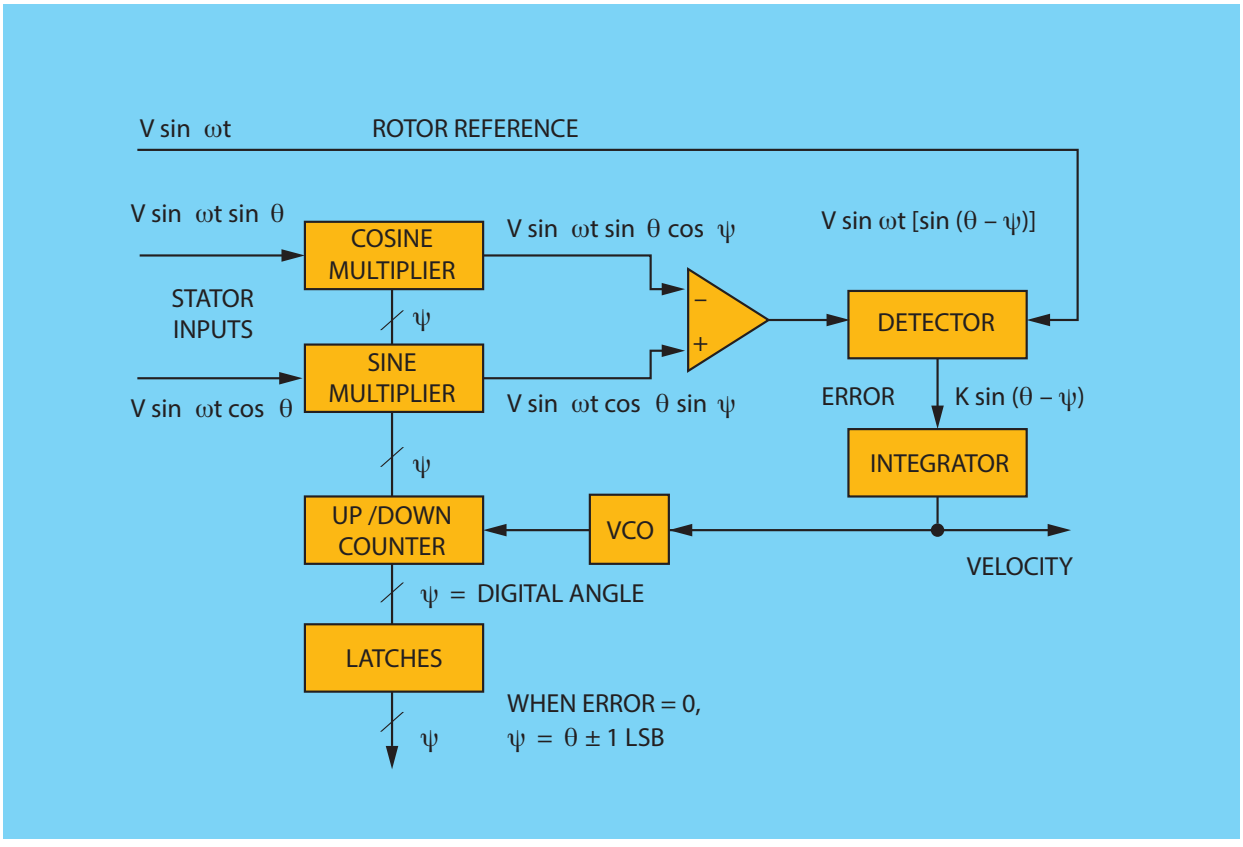


Figure 2 – Resolver-to-digital converter (RDC) block diagram

sine and cosine signals have the same phase as the original excitation signal; only their amplitudes are modulated by sine and cosine as the shaft rotates.

In the second method, a stator winding is excited with the alternating signals, which are in phase quadrature to each other. Then a voltage induces in the rotor winding. The winding's amplitude and frequency are fixed, but its phase shift varies with the shaft angle.

The resolver can be positioned where the angle needs to be measured [2]. The electronics, generally a resolver-to-digital converter (RDC), can be positioned where the digital output needs to be measured. Analog output from the resolver, which contains the angular position information of the shaft, is then transformed in digital form using the RDC.

FUNCTIONALITY OF THE TYPICAL RDC

In general, the two outputs of a resolver are applied to the sine and cosine multiplier of the RDC [3]. These multipliers incorporate sine and cosine lookup tables and function as multiplying digital-to-analog converters. Figure 2 shows their functionality.

Let us assume, in the beginning, that the current state of the up/down counter is a digital number representing a trial angle, ψ. The converter seeks to adjust the digital angle, ψ, continuously to become equal to and track θ, the analog angle being measured.

The stator output voltage of the resolver is:

$V1 = V \sin\omega t \sin\theta$ Eq. 1

$V2 = V \sin\omega t \cos\theta$ Eq. 2

where θ is the angle of the resolver's rotor. The digital angle ψ is applied to the cosine multiplier and its cosine is multiplied by V1 to produce the term:

$V \sin\omega t \sin\theta \cos\psi.$ Eq. 3

The digital angle ψ is also applied to the sine multiplier and multiplied by V2 to produce the term:

$V \sin\omega t \cos\theta \sin\psi.$ Eq. 4

These two signals are subtracted from each other by the error amplifier to yield an ac error signal of the form:

$(V \sin\omega t \sin\theta\cos\psi - V \sin\omega t \cos\theta \sin\psi)$ Eq. 5

$V \sin\omega t (\sin\theta \cos\psi - \cos\theta \sin\psi)$ Eq. 6

From trigonometric identity, this reduces to:

$V \sin\omega t [\sin(\theta - \psi)]$ Eq. 7

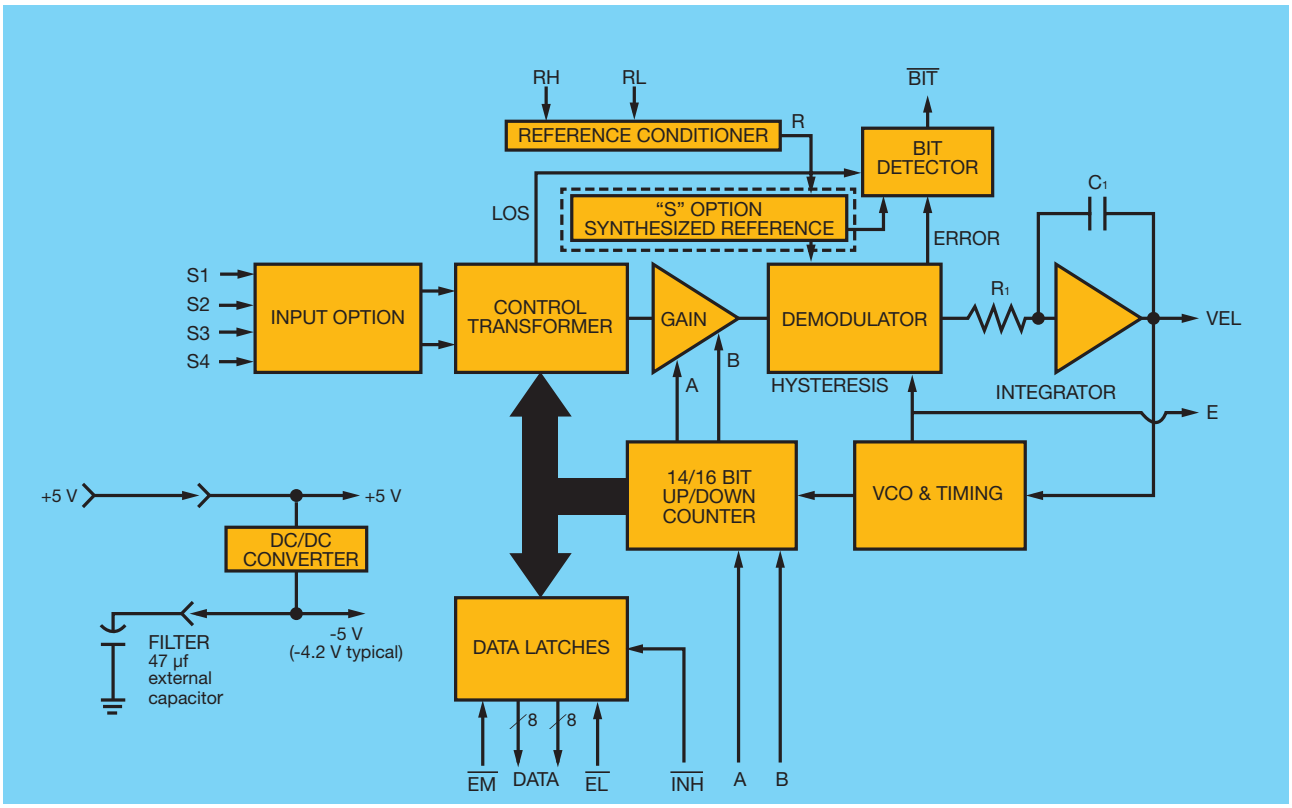


Figure 3 – SD-14620 block diagram (one channel)

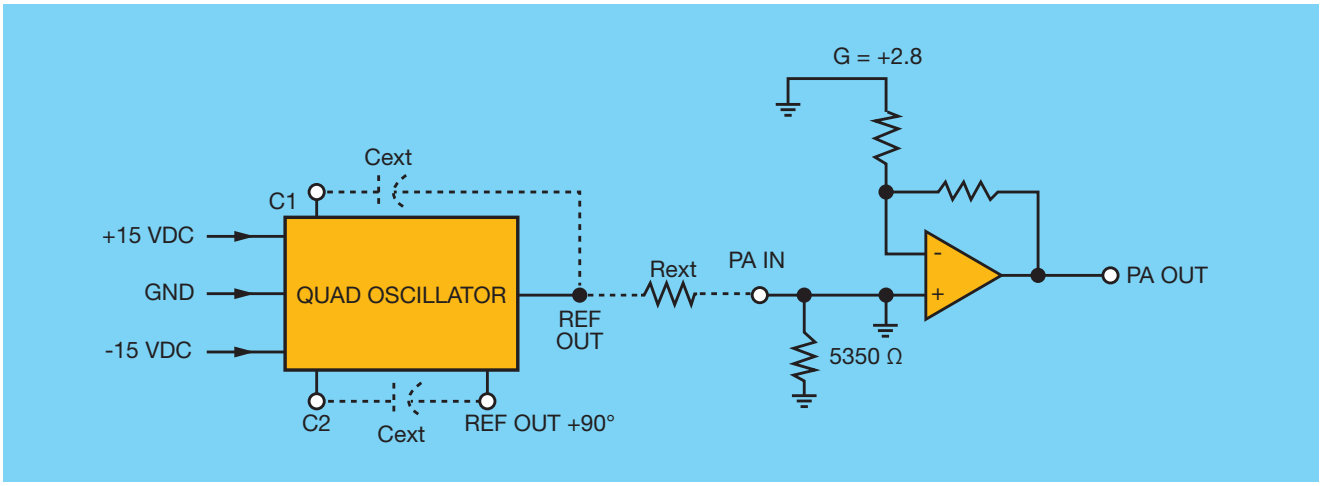


Figure 4 – Block diagram of the OSC-15802 reference oscillator

The detector synchronously demodulates this ac error signal, using the resolver's rotor voltage as a reference. This results in a dc error signal proportional to sin (θ -ψ).

The dc error signal feeds an integrator, the output of which drives a voltage-controlled oscillator. The VCO, in turn, causes the up/down counter to count in the proper direction to cause:

$$\sin(\theta - \psi) \rightarrow 0. \quad \text{Eq. 8}$$

When this result is achieved,

$$\theta - \psi \rightarrow 0, \quad \text{Eq. 9}$$

and therefore

$$\theta = \psi \quad \text{Eq. 10}$$

in one count. Hence, the counter's digital output, ψ, represents the angle θ. The latches make it possible to transfer this data externally without interrupting the loop's tracking.

This circuit is equivalent to a Type 2 servo loop, as it has in effect two integrators. One is the counter, which accumulates pulses; the other is the integrator at the output of the detector. In a Type 2 servo loop with a constant-rotational-velocity input, the output digital word continuously follows, or tracks, the input, without needing externally derived conversion.

TYPICAL EXAMPLE OF AN RDC: THE SD-14621

The SD-14621 is a small, low-cost, RDC from Data Device Corp. (DDC). It has two channels with programmable resolution control. Resolution programming allows selection of 10-, 12-, 14- or 16-bit modes [4]. This feature allows low resolution for fast tracking or higher resolution for higher accuracy. Thanks to its size, cost, accuracy and versatility, this converter is suitable for high-performance military, commercial and position-control systems.

A single +5 V is required for device operation. The converter has velocity outputs (VEL A, VEL B) of a voltage range of ±4 V with respect to analog ground, which can be used to replace a tachometer. Two built-in test outputs are provided for two channels (/BIT A and /BIT B) to indicate loss of signal (LOS).

This converter has three main sections: an input front end, an error processor and a digital interface. The front end differs for synchro, resolver and direct inputs. An electronic Scott-T is used for synchro inputs, a resolver conditioner for resolver inputs and a sine-and-cosine voltage follower for direct inputs. These amplifiers feed the high-accuracy control transformer (CT). The other input of the CT is a 16-bit digital angle ψ and the output is an analog error angle, or a difference angle, between the two inputs. The CT performs the ratiometric trigonometric computation of SINθ COSψ - COSθ SINψ = Sin(θ-ψ) using amplifiers, switches, logic and capacitors in precision ratios.

Compared with a conventional precision resistor, these capacitors are used in precision ratios to get enhanced accuracy. Further, these capacitors (which are used with an op amp as a computing element) sample at high rates to eliminate drift and op-amp offsets.

The DC error processing is integrated, yielding a velocity voltage that drives a voltage-controlled oscillator. This VCO is an incremental integrator when it is combined with the velocity integrator: a Type 2 servo feedback loop.

REFERENCE OSCILLATOR

The power oscillator in our design, also from DDC, is the OSC-15802. This device is suitable for RDC, synchro, LVDT, RVDT and inductosyn applications [5]. The frequency and amplitude outputs are programmable with capacitors and resistors, respectively. The output frequency range is 400 Hz to

The I/O voltage of the FPGA is 3.3 V, while the RDC's voltage is 5 V. We used voltage transceivers to achieve voltage compatibility between the two devices.

10 kHz, with an output voltage of 7 Vrms. Figure 4 shows a block diagram of the device.

The oscillator output, which is given to the resolver and the RDC, works as a reference signal.

VIRTEX-5 FX30T FPGA AND RDC INTERFACE

For our design, we used a Xilinx Virtex®-5 FX30T FPGA [6]. The I/O voltage of the FPGA is 3.3 V, while the RDC's voltage is 5 V. We used voltage transceivers to achieve voltage compatibility between the two devices. Internal connections with the FPGA are established through the GPIO IP core provided by Xilinx, as seen in Figure 5.

For simplicity's sake, Figure 5 shows just one channel with the single resolver interface. You will find the pin details of the RDC and corresponding pin locking with the FPGA in the Xilinx Board Description (XBD) file that accompanies this article. The details are listed in Section 1 of that document.

DETAILS OF THE DEVICE DRIVER

In this case, we used an external input clock of 20 MHz for the FPGA. This FPGA has a hard PowerPC® 440 core that is running at a 200-MHz frequency. The timing diagram of the RDC is shown in Figure 6 and Figure 7.

In accordance with the timing diagram of the RDC, we developed, tested and confirmed correct functionality with the actual hardware [4]. The actual code of the device driver is included in the separate XBD file. As per the timing diagram, we generated required delays using for loops. When processing is running at 200 MHz, each count corresponds to a delay of 5 nanoseconds.

The device driver has three sections of code: RDC initialization; generation of a control signal and reading from channel A of the RDC; and generation of a control signal and reading from channel B. RDC initialization is the point at which the direction of the signal and default values are

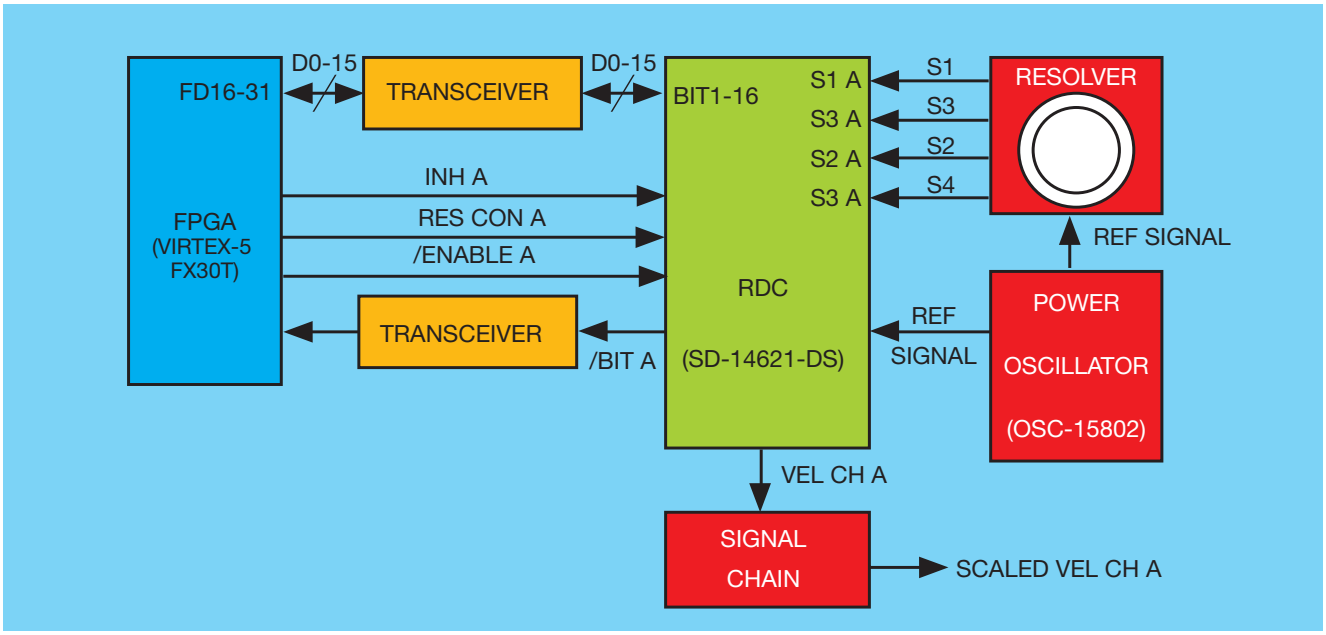


Figure 5 – RDC interface with the Virtex-5 FPGA (single channel)

set. For example, with the following statement, the direction is set as "out" from the FPGA to the RDC.

```
XGpio_WriteReg(XPAR_RESOLUTION_CNTRL_CH_A_
  BASEADDR,XGPIO_TRI_OFFSET,0x000);
```

With the next statement, the 16-bit resolution is set by writing "0x3" (that is, pulling high):

```
XGpio_WriteReg(XPAR_RESOLUTION_CNTRL_CH_A_
  BASEADDR,XGPIO_DATA_OFFSET,0x03);
```

Figure 8 shows a snapshot of the coding. Note: for simplification, we have included code for only one channel.

As we have seen, angle transducers help engineers create a better wheel and thus a plethora of more efficient machinery. Resolvers are an especially useful type of angle transducer, and when properly paired and controlled with an FPGA, can help engineers create even more remarkable machinery.

References

1. "Synchro/Resolver Conversion Handbook," Data Device Corp.
2. John Gasking, "Resolver-to-Digital Conversion: A Simple and Cost-Effective Alternative to Optical Shaft Encoders," AN-263, Analog Devices
3. Walt Kester, "Resolver to Digital Converter," MT-030, Analog Devices
4. SD-14620 Series Data Sheet, Data Device Corp.
5. OSC-15802 Data Sheet, Data Device Corp.
6. "Virtex-5 Family Overview," Xilinx

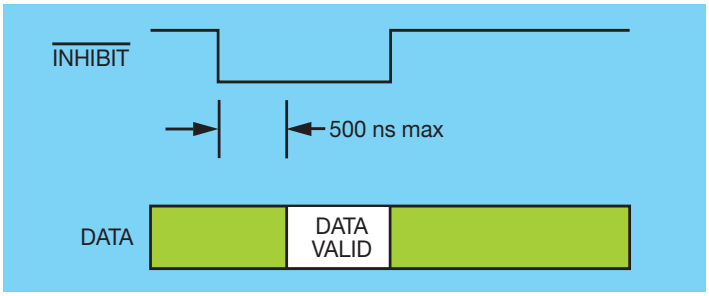


Figure 6 – INHIBIT timing

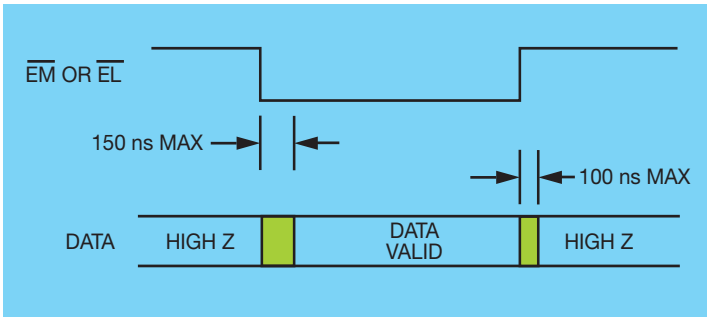


Figure 7 – ENABLE timing

```
seXGpio_WriteReg(XPAR_INHIBIT_CH_A_
BASEADDR,XGPIO_DATA_OFFSET,0x01);
for(i=0;i<=5;i++); //gives delay of 25 ns
          XGpio_WriteReg(XPAR_ENABLE_LSB_
CH_A_BIT_BASEADDR,XGPIO_DATA_OFFSET,0x01);

for(i=0;i<=5;i++);
XGpio_WriteReg(XPAR_INHIBIT_CH_A_BASEAD-
DR,XGPIO_DATA_OFFSET,0x00);
for(i=0;i<=2;i++);
XGpio_WriteReg(XPAR_ENABLE_LSB_CH_A_BIT_
BASEADDR,XGPIO_DATA_OFFSET,0x00);

for(i=0;i<=2;i++);
lsb_val=XGpio_ReadReg(XPAR_RDC_DATA_15_
TO_0_PINS_BASEADDR,XGPIO_DATA_OFFSET);

XGpio_WriteReg(XPAR_INHIBIT_CH_A_BASEAD-
DR,XGPIO_DATA_OFFSET,0x01);
      for(i=0;i<=5;i++);
XGpio_WriteReg(XPAR_ENABLE_LSB_CH_A_BIT_
BASEADDR,XGPIO_DATA_OFFSET,0x01);
for(i=0;i<=25;i++);

XGpio_WriteReg(XPAR_INHIBIT_CH_A_BASEAD-
DR,XGPIO_DATA_OFFSET,0x01);
for(i=0;i<=5;i++);
XGpio_WriteReg(XPAR_ENABLE_MSB_CH_A_BIT_
BASEADDR,XGPIO_DATA_OFFSET,0x01);
for(i=0;i<=5;i++);

XGpio_WriteReg(XPAR_INHIBIT_CH_A_BASEAD-
DR,XGPIO_DATA_OFFSET,0x00);
      for(i=0;i<=2;i++);
XGpio_WriteReg(XPAR_ENABLE_MSB_CH_A_BIT_
BASEADDR,XGPIO_DATA_OFFSET,0x00);
for(i=0;i<=2;i++);
msb_val=XGpio_ReadReg(XPAR_RDC_DATA_15_
TO_0_PINS_BASEADDR,XGPIO_DATA_OFFSET);

  lsb_val=lsb_val & 0x00ff;

  msb_val=msb_val & 0xff00;

  rdccount_cha = msb_val | lsb_val;

XGpio_WriteReg(XPAR_INHIBIT_CH_A_BA-
SEADDR,XGPIO_DATA_OFFSET,0X01);
for(i=0;i<=5;i++);

XGpio_WriteReg(XPAR_ENABLE_MSB_CH_A_
BIT_BASEADDR,XGPIO_DATA_OFFSET,0x01);
for(i=0;i<=20;i++);
```

Figure 8 – A snapshot of RDC device driver code