

An FPGA implementation of a digital Coriolis mass flow metering drive system

Mayela Zamora and Manus P. Henry, *Member, IEEE*

Abstract—Coriolis mass flow metering provides direct measurement of mass flow, and is generally regarded as the most accurate and precise flow technology in common use in industry. This paper describes the role of FPGA hardware, programmed using the Handel-C language, in the implementation of a “digital” Coriolis meter which replaces the conventional analog positive feedback system used to maintain flowtube oscillation. The FPGA is coupled to a microprocessor which carries out conventional measurement tasks, and selects the drive parameters to be used by the FPGA. The resulting meter is able to maintain operation in more difficult process conditions, including two-phase flow, which has previously caused Coriolis meters to cease oscillation. The system described in the paper is used in a commercial meter which has been successfully applied to two-phase industrial applications.

Index Terms—Coriolis flow meter, instrumentation, digital processing, digital delay, FPGA, Handel-C, vibration, resonance.

I. INTRODUCTION

CORIO LIS mass flow metering is a well established technique for industrial flow measurement. A Coriolis meter (Fig. 1) consists of an (essentially mechanical) vibrating flowtube through which the process fluid is passed, and an (essentially electronic) transmitter which maintains flowtube vibration via one or more drivers and performs measurement calculations based on signals from two sensors. The frequency of oscillation (varying from 50Hz to 1kHz for different flowtube designs) indicates the density of the process fluid, while the phase difference between the sensor signals provides the mass flow rate [1]. Coriolis meters offer many benefits, including high accuracy (to 0.1% for static flow rates), and good turndown (100:1 or better), while their limitations have included an inability to maintain operation when dealing with aerated fluids. A user perspective on Coriolis as an “almost perfect” flowmeter is given in [2].

The operation of a Coriolis mass flow meter is dependent upon the proper oscillation of the flowtube. This is controlled

by the drive signal(s) generated by the transmitter. The oscillation of the flowtube (as indicated by the sensor signals) is typically sinusoidal and hence characterized in terms of frequency, phase and amplitude. The drive signal is also often sinusoidal, or at least a regular waveform (e.g. square wave) for which similar attributes can be defined: the frequency, phase (relative to the sensor signal) and amplitude of the drive signal need to be determined and generated for optimal operation of the flowtube. A commonly-used criterion for optimal operation is that the flowtube should oscillate at its natural frequency of vibration, at a fixed amplitude. As measurement algorithms assume constant amplitude of oscillation over the calculation interval (typically 5 – 500ms), amplitude stability is relevant for measurement quality [3],[4].

For oscillation at the natural frequency, it is necessary [4] for the driving force to be 90° out of phase with the motion of vibration. Conveniently, the most commonly used sensor, based on an electromagnetic coil, measures velocity, hence the sensor signal is 90° out of phase with the motion of the flowtube. Thus an optimal drive signal has the same frequency of oscillation and phase as the sensor signal, with a drive amplitude selected to maintain a constant sensor amplitude.

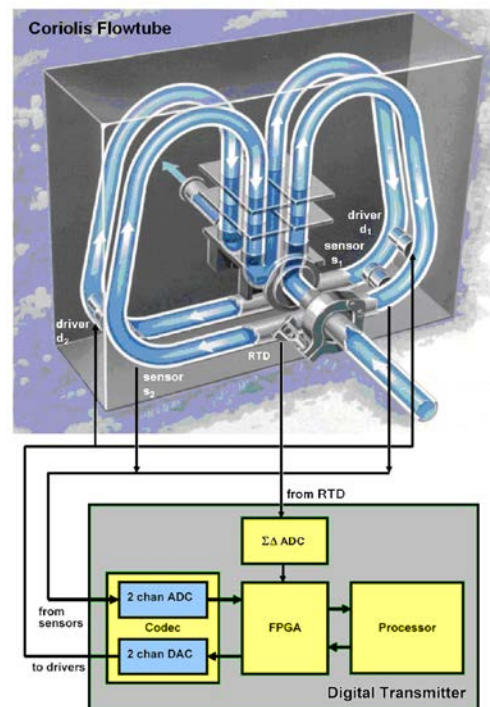


Fig. 1. Coriolis Mass Flow Meter

Manuscript received March 8, 2007, accepted Feb 7, 2008. This work was supported by Invensys.

Copyright (c) 2007 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

The authors are with Oxford University, Invensys University Technology Centre for Advanced Instrumentation, Department of Engineering, Science Parks Road Oxford OX1 3PJ, UK, e-mail: mayela.zamora@eng.ox.ac.uk and manus.henry@eng.ox.ac.uk.

Matching the drive output to the exact phase of the sensor signal is challenging. With small levels of phase offset, and with benign process conditions, the consequences are small – the drive signal power requirement increases. With more significant phase offset between driver and sensor, the flowtube oscillation becomes forced rather than natural, the drive energy requirement become significantly higher, and the drive frequency can drift away from its natural value. Finally, with large phase offset the meter may cease vibrating entirely (“stalling”), or begin to oscillate in another mode of vibration, typically at a frequency where the phase offset between driver and sensor is closer to an integral multiple of 360 degrees. Analogous issues are seen in power electronics design: for sinusoidal inputs and outputs, digital delay in the control circuitry can lead to inefficiencies [5],[6].

The most common technique for generating a drive signal has been *analog positive feedback*, whereby the sensor signal (containing the desired frequency and phase characteristics) is multiplied by a drive gain factor (either by analog or digital means) [3]. The drive gain required to maintain the desired amplitude of oscillation is proportional to the mechanical damping on the flowtube. Assuming negligible delay in the analog feedback circuitry, this approach ensures phase matching between sensor input and drive output. Positive feedback is straightforward to implement, but it provides only partial control of the drive waveform, and cannot prevent unwanted components in the sensor signal (e.g. other modes of vibration) from being fed back into the drive signal. In particular, in the presence of two-phase flow, drive systems based on analog feedback are prone to stalling [7]. The mechanical damping on the flowtube rises by two orders of magnitude with two-phase flow, and this damping varies rapidly. Most analog drive systems are unable to track and respond to damping under two-phase flow. Some designs have a maximum drive gain which, if exceeded by the damping, leads to catastrophic collapse in oscillation. High and variable damping leads to low and variable sensor amplitudes, and it is possible to lose track of the sensor signals, especially if they are contaminated with other modes of vibration.

An all-digital drive system avoids many of the pitfalls associated with analog positive feedback. The alternative approach presented in this paper is *drive waveform synthesis*, whereby the transmitter generates the drive waveform digitally, for example a pure sine wave or square wave, with the required amplitude, frequency and phase characteristics, in order to provide a highly adaptable and precise drive signal. This has several advantages over positive feedback, including full control over the drive waveform, and an ability to maintain operation even in two-phase flow, but the challenge is to match the phase of the sensor signal in real time given the inevitable delays of a digital implementation.

This paper describes a digital Coriolis transmitter design which implements drive waveform synthesis and which compensates for digital delays to ensure phase matching between the sensor input and drive output signals. Two

versions of the transmitter have been developed, the research prototype (700MHz PC-104 architecture) and a commercial variant (233 MHz PowerPC). Both use the VxWorks real-time operating system, employ a common code base in C++, and use a Field Programmable Gate Array (FPGA) to implement the critical real-time control of the driver and sensor interfacing. FPGAs are increasingly used for real-time control applications [8], and have an increasingly sophisticated set of design tools [9]. In this application, the FPGA is programmed in a C-like language called Handel-C [10].

The resulting transmitter is a platform for a range of innovative developments in both the research laboratory and industrial applications. For example:

- The dynamic response of the meter can be measured by the time required to indicate a step changes in flow. This transmitter has demonstrated a response time of 4ms, between 1 and 2 orders of magnitude faster than other reported Coriolis meters [11],[12]. This has found industrial application in, for example, short proving runs for custody transfer applications [13], and filling applications [14].
- The two-phase problem (the “dirty little secret” of Coriolis meters [2]) has been transformed into a useful two-phase measurement capability, with numerous industrial applications [7],[15],[16]. Most recently, this meter has been used successfully in the first reported extended field trial for wet gas [17]. The significance of the two-phase capability in the commercial device has been recognized by an award in a leading industrial journal [18].

Presently, some 19 US and UK patents have been granted based on this transmitter technology (e.g. [19],[20]); a further 23 US patent applications have been published. Given the commercial sensitivities of the industrial flow measurement market, there is often a reluctance to publish implementation details for innovative designs; in the current case this has led to a deferral of publication to the present time. Accordingly, some aspects of the design discussed in the paper (such as the choice of processor and FPGA) are no longer state-of-the-art; however, the resulting meter remains at the forefront of flow measurement innovation, and this paper, in presenting a detailed explanation of its design, is thus providing an contribution to the open literature.

Section 2 provides an overview of the hardware design and Section 3 explains the principle of the drive operation. Section 4 provides a brief introduction to Handel-C, which is used to describe the digital drive implementation in Section 5. Sections 6 explains the mechanisms used to ensure phase synchronisation in the FPGA and the processor respectively, and finally Section 7 describes how the resulting design has been used to extend the operation range of industrial Coriolis meters.

II. DIGITAL TRANSMITTER

In addition to starting and maintaining flowtube vibration, the transmitter calculates the fluid mass flow, density and other

parameters from measurements of flowtube vibration and temperature. In the research prototype, measurements are communicated using frequency modulated pulse output [21] or via ethernet. The commercial transmitter, the CFT50 [14],[15], communicates through its LCD screen and keypad, or via contact, current output, or standard industrial communication protocols.

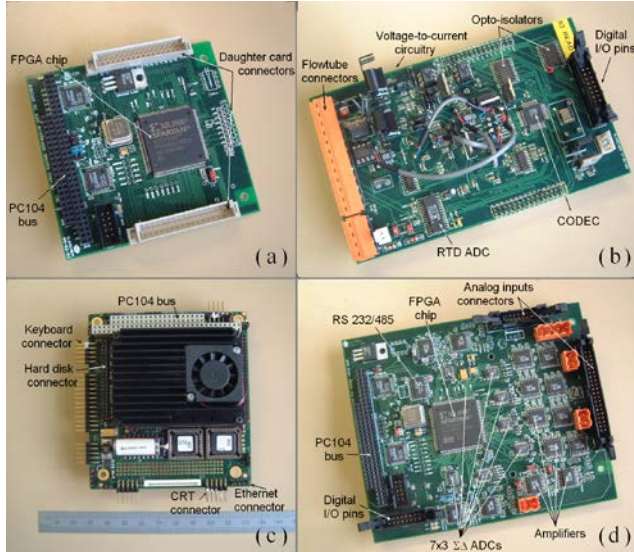


Fig. 2. Research transmitter's printed circuit boards. (a) Main FPGA board; (b) Daughter card: Front end circuitry and CODEC; (c) Processor board; (d) External I/O card (optional)

2.1. Architecture Description

Fig. 2 shows the PCBs used in the research transmitter, based on a PC-104 architecture, which emerged from a project to develop a general instrumentation prototyping platform [22]. It consists of:

- **Front end circuitry and stereo CODEC:** The CODEC is a dual analog-to-digital / digital-to-analog converter (ADC/DAC) device which provides two 24 bit ADC and two 24 bit DAC channels, operating at a sampling rate F_s of 48kHz, which are connected to the flowtube sensors and drivers respectively. There are two additional ADC channels provided by a sigma-delta converter connected to resistance temperature detectors (RTDs) to monitor the fluid and flowtube temperature. These components are located on a daughter board (Fig. 2b) of the main FPGA board (Fig. 2a), along with opto-isolated digital inputs / outputs (I/O) and voltage-to-current converters to drive the flowtube.

- **FPGA:** The Xilinx Spartan-IIIE device provides 300k gates of configurable logic. It serves as a digital interface to the ADC/DAC front-end circuitry, performing critical real-time tasks, including sensor signal filtering, waveform synthesis, buffering and peripheral control, under the direction of the processor. These tasks are more effectively designed and executed in true parallel on dedicated hardware rather than by time-slicing on a single processor. The FPGA board communicates with the processor board through a PC104 bus (Fig. 2a). Other forms of communication, such as high precision frequency output, are also generated within the FPGA. For the Coriolis application, 99% of FPGA slices were

used, with 57% of flip-flops and 78% of 4-input LUTS utilised. The design operates successfully with a 40MHz clock rate.

- **PC104 Processor board:** Based on a 700MHz Pentium III processor (Fig. 2c), this executes high precision, computationally intensive, measurement and control algorithms in full floating point. It runs under the VxWorks Real-Time Operating System. Measurements are updated every half-cycle of the sensor signal. For a typical 80Hz flowtube, this corresponds to 160Hz measurement update rate. However, for so-called straight flowtube geometries, the resonant frequencies can be as high as 800Hz. The research transmitter has successfully driven such devices while still providing two measurement updates per drive cycle, i.e. up to 1.6kHz [11].

An optional FPGA board (Fig. 2d), with 21 voltage or current analog input channels plus additional digital I/O, can be connected to the processor board. This has been used to monitor additional process variables in research and development, for example, a variety of environmental variables in two-phase flow experiments, and to communicate with other instruments using the Modbus protocol.

2.2. Measurement Calculation Overview

The frequency of oscillation is calculated by precise measurement of the time difference between zero crossings on the sensor signals, while Fourier techniques are used to calculate the amplitude and phase of the flowtube vibration [3],[12]. Temperature compensation for changes in flowtube stiffness is performed for both mass flow and density calculations. A non-linear amplitude control algorithm [4] provides stable oscillation, while a sustainable set-point for the amplitude of oscillation is selected during highly damped operation (e.g. two-phase flow), as illustrated in section 7. In single phase conditions the meter demonstrates very fast dynamic response [11], and repeatability as low as 0.01%, with accuracy within 0.15% [13]. In batching-from-empty and two-phase flow conditions, the digital transmitter is not only capable of maintaining oscillation but also of compensating for errors in the measurements, achieving 0.05% repeatability and 1% to 5% accuracy for gas void ratios up to 80% [13],[16].

These novel measurement applications have previously been described in the literature; the purpose of this paper is to provide a detailed description of the transmitter drive mechanism that enables such innovations.

III. DIGITAL DRIVE: OVERVIEW OF OPERATION

The sensor signals are read by two ADC channels in the CODEC (see Fig. 3), controlled by the FPGA. After low pass filtering and buffering in external RAM, the data is sent to the processor for analysis. Having calculated the mass flow and density [3], the processor determines parameter values for maintaining flowtube oscillation. These are sent to the FPGA, which synthesises the drive signals and passes them to the CODEC. Although the parameters are shown in Fig 3 passing independently from the processor to the FPGA, in practice

they are combined into one or at most two (section 6.4) packets for efficient communication over the PC-104 bus.

3.1. Weighted Sum of Sensor Signals

As mechanical systems, Coriolis flowtubes have multiple modes of vibration; indeed two modes of vibration are essential for the measurement of flow. The drive mode is the frequency at which the meter is intended to vibrate (typically the lowest mode), while the Coriolis mode is an adjacent mode of vibration that, in the presence of mass flow, causes phase difference to be generated *at the drive frequency* [13]. However, all other modes of vibration, including the Coriolis mode, are readily excited at their own natural frequencies, by external mechanical disturbances, flow disturbances or two-phase flow. Most higher modes of vibration can be excluded with signal processing by low-pass filtering in the FPGA (at the cost of introducing a phase delay). However, for the Coriolis mode itself there is a mechanical design tradeoff – the smaller the difference between the drive and Coriolis frequencies, the higher the sensitivity of the flowtube in terms of phase difference per unit flow rate. For example, for the flowtube design shown in Fig. 1, typical drive and Coriolis frequencies are 80Hz and 50Hz respectively for a water-filled flowtube, or 95Hz and 60Hz for an air-filled flowtube.

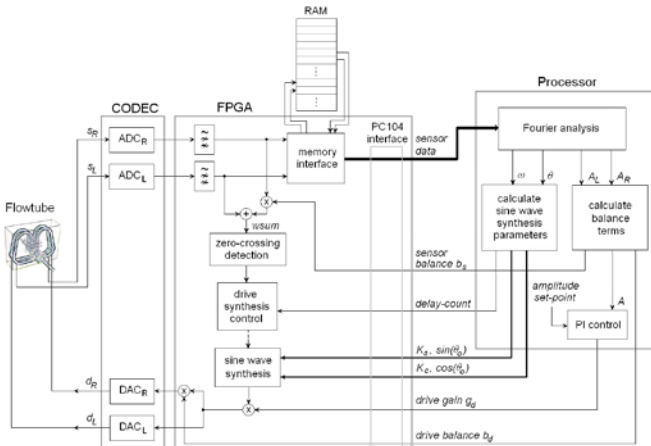


Fig. 3. Drive synthesis flowchart

With a base sample rate of 48kHz, low pass filtering cannot easily separate the two modes, and the presence of low levels of Coriolis mode vibration in the sensor signals is a significant issue for measurement calculations, requiring specialised signal processing techniques in the processor [3].

Fortunately, however, a simple technique can be used to reduce the influence of the Coriolis mode of vibration. Assuming zero flow, then the drive frequency component of the two sensor signals are exactly in phase. However, as adjacent modes of vibration have different wave shapes, the Coriolis mode vibrations are 180° out of phase between the two sensors. Summing the two sensor signals reinforces the drive mode component while largely cancelling out the Coriolis mode component.

With liquid flow, there is a phase difference between the sensor signals. The sum of the sensor signals still largely

eliminates the Coriolis mode component, while the resulting drive mode component has a phase at the midpoint between the two sensor signals. As the amplitudes of the two sensor signals are never identical, a further refinement is achieved by weighting one of the signals to have the same amplitude of the other (via the sensor balance term b_s).

The previously-stated goal of generating a drive signal to match the “phase” of the flowtube oscillation is thus revised as follows. The phase of the weighted sum signal, i.e. the midpoint of the phase of the two sensor signals, should be matched. This strategy effectively reduces one of the most significant noise sources, namely any Coriolis mode vibration.

Thus, in the drive system described here, the weighted sum is calculated and its zero crossings are used to provide a stable reference for synchronising the drive signal with the motion of the flowtube.

3.2. Steps in the Drive Process

The following tasks are performed (Fig. 3).

FPGA - acquire and pre-process the sensor data (at 48kHz):

- Acquire sensor signals s_L and s_R from CODEC ADC.
- Apply a low-pass elliptical digital filter to sensor signals, producing s_{Lf} and s_{Rf} .
- Calculate weighted sum of filtered sensor signals, $wsum$.
- Send filtered sensor signals s_{Lf} and s_{Rf} to RAM awaiting transmission to the processor.

Processor - analyse sensor data (typically at 160Hz):

- Fetch filtered sensor data for next measurement cycle.
- Perform Fourier analysis to determine frequency, phase and amplitudes of oscillation.
- Calculate delay terms (not shown).
- Calculate drive synthesis parameters.
- Calculate flow and density measurements (not shown).
- Calculate sensor and drive balance factors b_s , b_d .
- Perform amplitude control algorithm to select the drive gain factor, g_d .
- Send to FPGA the synthesis sine wave parameters, gain factor, sensor balance factor and drive balance factor.

FPGA - drive synthesis (at 48kHz):

- Synthesize new drive wave d based on current synthesis parameters.
- Multiply by gain g_d and apply drive balance factor b_d .
- Send scaled d_L and d_R values to CODEC DACs.
- Detect negative-to-positive zero crossing in $wsum$.
- Accept new synthesis parameters from processor.
- Wait $delay_count$ samples from zero crossing before applying new synthesis parameters, to ensure phase matching between input and output.

A more detailed explanation of the method used to find the phase offset and amplitude of the sensor signals, balance terms and drive gain, as well as calculation and compensation of mass flow and density measurements can be found in [3].

The benefit of using the FPGA for the low-level, real-time aspects of the application are that it is possible to write pieces of code, implemented in dedicated hardware, each providing a specific requirement (interfacing to the CODEC, bus or RAM, filtering etc). These all run in parallel so that I/O requests are dealt with immediately (and possibly simultaneously) by the relevant code segments, without the need for interrupts. This introduces an elegance and simplicity to real time I/O design which is simply not possible using a single processor.

Key aspects of the drive synthesis are now described in more detail, including their FPGA implementation using the language Handel-C.

IV. HANDEL-C

Handel-C is a language for defining FPGA functionality. It is based on a subset of ANSI-C, with some features excluded (e.g. floating point variables), but with additional language constructs provided to describe parallelism and synchronization, key aspects of FPGA operation. For example, “seq” and “par” statements specify which program steps are to be executed sequentially, and which are to be operated in parallel. The language has a well-defined semantics, based ultimately on the CSP computational model [23], which guarantees the action and timing of every program statement. Page [24] describes how Handel-C statements are mapped into FPGA hardware.

While the current standards languages for FPGA design are VHDL and Verilog, there is increasing interest in higher-level design tools which may generate as output either VHDL or FPGA netlists directly. These include Matlab/Simulink [25] and C++ [26], while SystemC [27] is a major initiative to develop a new language standard around a C-style syntax. Fleury et al. [28] provides a detailed description of the features of Handel-C, including a comparison with VHDL, while [29] gives an example of a circuit design simulation implemented in both in SystemC and VHDL. As discussed in [30], one of the key benefits of using such higher-level design languages is considerably reduced design time and effort, at the expense of some loss of efficiency in FPGA usage.

The following code section shows the top level Handel-C program, or main function, which consists simply of a set of calls to more specific functions in parallel. When the FPGA is loaded and the program commences, each of these functions is invoked in true parallel, operating within its respective section of FPGA hardware. As none of the individual functions terminates, neither does the main program.

```
set clock = external_ "P182" with {rate = 40.0};

void main (void)
{
    par {
        processor_interface(); // execute in parallel
        do_requests();          // read/write PC-104 bus
        do_rtd_adc();           // respond to PC requests
        do_rtd_adc();           // ADC interface to RTD sensor
        do_codec();            // ADC interface for codec
        do_sensor_and_drive();  // process sensor data
        mem_interface();        // interface to RAM hardware
        do_memory();            // higher level memory R/W
        do_digital_inputs();    // all other inputs
        do_digital_outputs();   // all other outputs
    }
```

```
}
```

Handel-C provides special constructs to interact with external hardware via the FPGA pins. These declarations are not shown, other than the clock interface definition which indicates which pin and what clock rate (40MHz) will be used. Similarly, communications between the FPGA and the processor are achieved via the processor bus through FPGA I/O pins. The `processor_interface()` function continually reads, and on occasion writes to, the bus pins, while the higher-level `do_requests()` function interprets and responds to commands and parameters sent from the processor.

V. DIGITAL DRIVE IMPLEMENTATION

While the Handel-C program operates with a 40MHz clock, the codec provides new data at 48kHz, thus allowing 830 clock ticks between codec updates. Within this interval, it is necessary to provide a new drive output value. The function `do_sensor_and_drive()` carries out tasks from the receipt of new sensor data to the generation of new drive values.

When new data are ready from the ADCs, both samples are filtered and stored, while a new drive signal sample is generated, multiplied by the gain and sent to the DACs.

```
static void do_sensor_and_drive(void)
{
    while (1) { // loop forever
        while (new_data != 1) { // new sensor data ready?
            delay(); // wait until next clock tick
        }
        new_data = 0; // reset data flag
        filter_and_calc_wsum(); // filter s1 and s2, get wsum
        do_sinewave_synthesis(); // calculate new drive outputs
        apply_drive_gain(); // scale drive outputs
        store_sensor_data(); // in circular buffer for PC
        update_gains(); // read updates from PC
    }
}
```

This code provides a high-level implementation of the drive synthesis flowchart (Fig. 3), while the individual functions define the implementation in more detail.

A basic requirement for the digital Coriolis drive system is the generation of a sine wave with frequency, phase and amplitude specified by the processor. Typically, the frequency is 50-1000Hz, while the synthesis rate is 48kHz. In the implementation described here, drive synthesis parameters are updated by the processor each drive cycle (typically at 80Hz). Thus the processor sends parameter values for the synthesis of several hundred consecutive DAC outputs at a time. The waveform synthesis takes place in two stages: a fixed amplitude sinusoid is generated with the specified frequency and phase, which is then multiplied by a gain factor to provide the specified amplitude. The latter step is straightforward and not described.

5.1. Digital Sine Wave

Dai and colleagues [31],[32] used direct digital synthesis [33],[34], a method consisting in storing the signal values in look-up tables, to generate sinewaves within an FPGA. An alternative approach, used in the Coriolis application, is to apply a recursive method. Turner [35], describes the fixed

amplitude, coupled, standard quadrature oscillator based on the trigonometry of the sum of two angles:

$$\begin{aligned}\sin(\theta + \delta) &= \sin(\theta) \cos(\delta) + \cos(\theta) \sin(\delta) \\ \cos(\theta + \delta) &= \cos(\theta) \cos(\delta) - \sin(\theta) \sin(\delta)\end{aligned}\quad (1)$$

where $\theta = \theta_0 + n\delta$, for $n = 0, 1, 2, \dots$ and $\delta = 2\pi f_{wsum} / F_s$. It is assumed that the initial value θ_0 (or phase offset) is close to zero as the synthesis parameters θ_0 and δ are updated at negative-to-positive zero crossings (see Fig. 4).

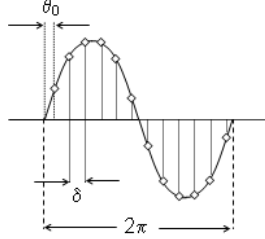


Fig. 4. Digital sinewave synthesis

Parameter δ controls the frequency of the synthesized wave, which is calculated by the processor every measurement cycle to match the sensor weighted sum frequency f_{wsum} . It is passed to the FPGA in the form of two values, $\sin(\delta)$ and $\cos(\delta)$, in a 24-bit format. Typical values of δ are very small, therefore $\sin(\delta) \approx 0$ and $\cos(\delta) \approx 1$. In order to achieve a better precision, $\cos(\delta)$ is sent as its difference from 1.0. Also, both parameters, $\sin(\delta)$ and $[1 - \cos(\delta)]$ are scaled so that the most significant bits known to be always zero are discarded and only the first 24 that can take values different from zero are sent to the FPGA.

The recurrent relation in (1) has been slightly manipulated to handle the parameter $[1 - \cos(\delta)]$:

$$\begin{aligned}\sin(\theta_{j+1}) &= \sin(\theta_j) + \Delta \sin \\ \cos(\theta_{j+1}) &= \cos(\theta_j) + \Delta \cos\end{aligned}\quad (2)$$

where $\theta_j = j\delta$, therefore $\theta_{j+1} = \theta_j + \delta$ and (2) can be rearranged as:

$$\begin{aligned}\Delta \sin &= \sin(\theta_{j+1}) - \sin(\theta_j) \\ \Rightarrow \Delta \sin &= -\sin(\theta_j) + \sin(\theta_j) \cos(\delta) + \cos(\theta_j) \sin(\delta) \\ \Rightarrow \Delta \sin &= -\sin(\theta_j) [1 - \cos(\delta)] + \cos(\theta_j) \sin(\delta)\end{aligned}\quad (3)$$

and

$$\begin{aligned}\Delta \cos &= \cos(\theta_{j+1}) - \cos(\theta_j) \\ \Rightarrow \Delta \cos &= -\cos(\theta_j) + \cos(\theta_j) \cos(\delta) - \sin(\theta_j) \sin(\delta) \\ \Rightarrow \Delta \cos &= -\cos(\theta_j) [1 - \cos(\delta)] - \sin(\theta_j) \sin(\delta)\end{aligned}\quad (4)$$

Equations (3) and (4) have been implemented in the function `do_sine_wave()`. In Handel-C, basic integer arithmetic functions $+$, $-$, $*$ and $/$ are provided. Variables are integers, of arbitrary bit length. However, multiplications and divisions on large operands can generate vast quantities of logic. By default, each $*$ operator generates a separate piece of FPGA silicon to perform the multiplication. For an efficient design, it is clearly desirable to reuse silicon where practical.

For example, in the Coriolis drive design, sensor and drive data is stored in 38-bit integer format (to allow for scaling over 3 orders of magnitude), while multiplying factors are 24-bit quantities. For the Coriolis calculations, usually two arithmetic operations take place in parallel for, where, for example, two parameters are all multiplied by the same factor. Accordingly, functions `do_mult()`, `do_sum()` and `do_neg()` provide dedicated registers to carry out multiplication, sum and subtraction operations. For the FPGA architecture targeted for this application (Xilinx Spartan IIE), the multiplication has been implemented using serial addition. More expensive and/or recent FPGAs provide dedicated hardware for multiplication; where such architectures are used a different design decision may be considered optimal.

5.2. Sine Wave Code

The function `do_sine_wave()` updates the values of the global variables `sin` and `cos` for generating the next drive output values, based on equations (3) and (4) above. Parameters $[1 - \cos(\delta)]$ and $\sin(\delta)$ are obtained from the processor in the forms $kc = 2^b \cdot [1 - \cos(\delta)]$, and $ks = 2^a \cdot \sin(\delta)$. Scaling exponents a and b are used to maximise the number of significant bits, given that parameters kc and ks are close to zero, and are hardwired as constants `FIRST_SHIFT` and `SECOND_SHIFT`, where `FIRST_SHIFT = a` and `SECOND_SHIFT = (b - a)`.

```
signed int 38  sin, cos; // global variables - current
                        // sin/cos values
unsigned int 24  kc, ks; // factors to calculate next
                        // values

void do_sine_wave( void)
{
    signed int 38  shfcos, shfsin;

    par {
        // scale down multiplicands
        shfcos = cos >> FIRST_SHIFT;
        shfsin = sin >> FIRST_SHIFT;
    }

    par {
        // load do_mult() parameters
        mult_fact = ks;
        mult_in[0] = shfcos;
        mult_in[1] = shfsin;
    }
    do_mult(); // mult_res = [Ks*cos, Ks*sin]

    par {
        // load do_sub() parameters
        al[0] = cos;
        nl[0] = mult_res[1];
    }
    do_sub(); // sum[0] = cos - Ks*sin
    cos = sum[0];

    par {
        // load do_sum() parameters
        al[0] = sin;
        a2[0] = mult_res[0];
    }
    do_sum(); // sum[0] = sin + Ks*cos

    par {
        // load do_mult() parameters
        mult_fact = kc;
        mult_in[0] = shfcos >> SECOND_SHIFT;
        mult_in[1] = shfsin >> SECOND_SHIFT;
    }
    do_mult(); // mult_res = [Kc*cos, Kc*sin]

    par {
        // load do_sub() parameters
        al[0] = cos;
        nl[0] = mult_res[0];
        al[1] = sum[0];
        nl[1] = mult_res[1];
    }
}
```

```

do_sub();           // sum[0] = Kc*cos - Ks*sin
                   // sum[1] = Kc*sin + Ks*cos
par {
  cos = sum[0];
  sin = sum[1];
}

```

The revised value of `sin` is used to generate the next drive output. Repeated calls to `do_sine_wave()` generate an complete sine wave sequence, thus enabling the FPGA to synthesize the drive output at 48kHz with only occasional parameter updates from the processor.

VI. TIME AND PHASE SYNCHRONISATION

The task of matching phase between sensor input and drive output is achieved through time delay calculations in the processor and a synchronization mechanism within the FPGA.

6.1. Sources of Time and Phase Delay

The most important sources of time delay are the following:

- **ADC input delay.** Sigma-delta converters typically use a noise-shaper digital filter to facilitate high precision (say 24 bit) at a high sampling rate (say 48kHz), but at the cost of several samples delay between analog input and digital output. However, the delay in samples introduced by the ADC is constant for all frequencies in the device's bandwidth, as the noise-shaper filter has linear phase. For example, in the device used in this application, there is a delay of 51 samples.
- **Filtering delay and phase shift.** Of the 24kHz bandwidth sampled, only the lowest 150Hz is used. Sixth-order elliptical filters are implemented in the FPGA. Although the filtering delay is small – a few microseconds – significant phase offset is introduced, this being a non-linear function of the frequency of oscillation. This phase response is modeled within the processor.
- **Processor calculation.** The major source of delay during processor calculation is waiting for the next full cycle of sensor data to become available, as the Fourier techniques used to calculate amplitude and phase information require complete periods of drive cycle data.
- **Bus communication and processor delay.** These can introduce random variations in the timing of data processing and communication of parameters back to the FPGA. The drive control method devised here allows for flexibility in the timing of the communications.
- **DAC output delay.** Finally, there is (for the device employed in this design) a further 24 sample delay between sending a 24-bit value to the DAC and the corresponding analog voltage appearing at its output.

6.2. FPGA Synchronisation

Coarse synchronisation (to within one sensor sample) is provided by detecting each negative-to-positive zero crossing (zc) on the weighted sum within the FPGA. As previously discussed, this signal is particularly noise immune because the Coriolis mode components from each sensor cancel each other out. Each zc can be predicted by the processor and detected by the FPGA, so the zc is used to provide synchronization for the purposes of phase alignment between sensor input and drive

output. The processor calculates a delay count from the zc after which the new sine wave synthesis parameters are used. For example, suppose each sample corresponds to one degree of phase, and there is a calculated 60.2 degrees of (uncorrected) phase lag between input and output. When the processor sends a new set of drive synthesis parameters to the FPGA, it instructs the FPGA to defer using the new set until 300 samples after the next zc. Given the extra 60 samples delay, this causes the drive output signal to be in phase with the sensor input. Fine grain synchronisation is provided by the initial values of the sine and cosine parameters, for example corresponding to a phase offset of 0.2 degrees.

6.3. Calculating the Time and Phase Offsets

The processor fetches a full cycle of the filtered sensor signals from the FPGA, and identifies the relative location of the negative-to-positive zero crossings on each sensor signal by interpolation (Fig. 5).

The filter applied to sensor signals s_L and s_R has a phase delay which depends on the frequency of the signal at the filter input. This delay can be calculated from the filter coefficients, and converted into the corresponding number of samples delay. The sample delay is modelled by a 9th-order polynomial, with an error not exceeding ± 0.05 samples.

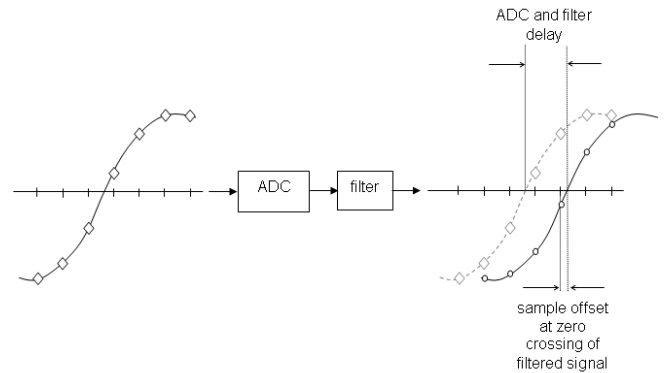


Fig. 5. Delays introduced by ADC and digital filter on sensor voltage signal

The DAC has a delay of 24 samples from a new value being written to its input register, to the resulting change at its analog output. This must be included in the delay calculation, but also influences the actual timing of when updated values are sent to the DAC, as shown in Fig. 6.

Processor measurement calculations are intensive and variable. By the time they are complete the sensor signals have advanced by several samples, possibly by more than a full cycle. The processor determines how far behind “real time” it has fallen by reading the FPGA circular buffer backlog. This shows how many ADC samples are waiting to be sent to the processor. Based on the current drive frequency, the processor can predict the occurrence of the next zero crossing to be observed by the FPGA, and hence an appropriate set of waveform synthesis and delay parameters.

Fig. 7 shows the timing of the steps starting from cycle i in the sensor signal through to the generation of a sine wave at the DAC output, based on the parameters calculated from

sensor cycle i . Even when the number of samples waiting in the FPGA buffer is less than one cycle, the drive output based on sensor cycle i cannot be sent in time to match the next cycle, i.e. cycle $i+1$, but rather at two periods later, at cycle $i+3$. Once a new set of synthesis parameters have been received from the processor, the FPGA waits until the next negative-to-positive zc . It then waits an additional number of samples (*delay-count*, calculated by the processor and included in the set of drive synthesis parameters for this cycle) to ensure phase synchronization between sensor input and drive output.

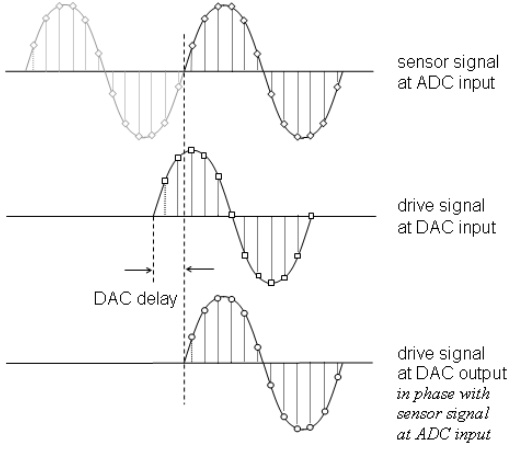


Fig. 6. Anticipating DAC delay on drive signal

6.4. Applying the Drive Gain

Two further refinements to the drive synthesis method are found to have practical benefit. Firstly, it is desirable to update the drive synthesis parameters at a zc to minimise any step change in output. There are two zero crossings per cycle, and the meter performs two measurement calculations per cycle. Given that the frequency of oscillation does not normally vary rapidly, the phase and frequency synthesis parameters are only updated once per cycle. However once reasonable phase matching has been successfully achieved, the drive gain, which determines the amplitude of the drive waveform, is the most influential parameter on amplitude control, and half-cycle updates of this parameter improve performance. Updated values are introduced at each zc , twice each drive cycle, as illustrated in section 7.

The second refinement is the ability to provide *negative* drive gain, i.e. to drive against the flowtube motion. As demonstrated in section 7 this can be beneficial in extreme conditions, such as two-phase flow. This is achieved very simply by using a single-bit parameter to indicate the polarity of the desired drive gain which, if set, results in the inversion of the drive output.

6.5. Sine Wave Initial Values

Along with the other synthesis parameters, the initial phase offset θ_0 is sent by the processor to the FPGA each cycle. More specifically, initial values $\sin(\theta_0)$ and $\cos(\theta_0)$, where $\sin(\theta_0)$ is the first output value of the synthesized sine wave, are passed in a 32-bit format to the FPGA. These provide fine

grain phase synchronisation to substantially less than one sample. Note that even for a fixed drive frequency, the value of θ_0 varies between cycles because the CODEC sample rate is never an exact multiple of the drive frequency. It is straightforward to calculate the phase step δ corresponding to a single sample. For a fixed drive frequency, the value of θ_0 will vary between 0 and δ due to the beating between the drive frequency and the CODEC update frequency. For example, for a CODEC with an update frequency of 48kHz and a drive frequency of 81.2Hz, then $\delta = 0.61^\circ$. Assuming a value of θ_0 of zero on cycle 1, Table I shows the values of θ_0 on subsequent cycles.

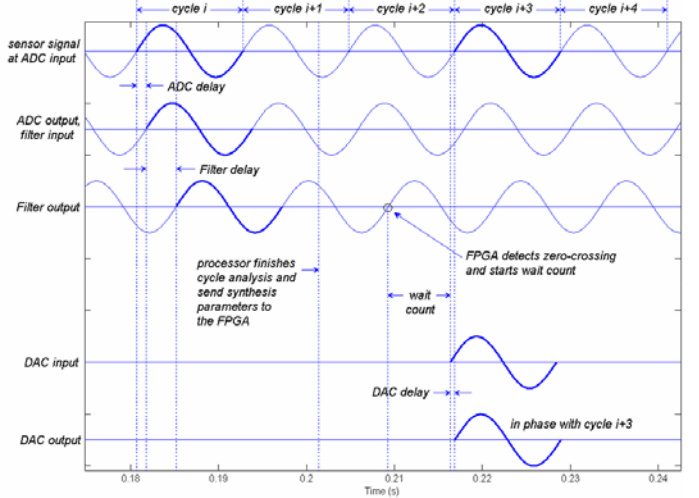


Fig. 7. Drive wave synthesis timing diagram

It can be seen that for a correct value of θ_0 it is important to co-ordinate cycle numbering between the processor and the FPGA, to ensure the right phase offset is used for each cycle. While it is unlikely that the required synthesis frequency will change very rapidly between cycles (and hence a delay in updating synthesis parameters is unlikely to generate an inappropriate drive frequency), Table II (with a drive frequency of 765Hz) demonstrates that the phase offset does vary significantly between consecutive cycles, so that poor co-ordination of phase offset could lead to a significant reduction in the quality of waveform synthesis.

TABLE I
VALUES OF θ_0 FOR A DRIVE FREQUENCY OF 81.2Hz AND A CODEC
FREQUENCY OF 48kHz

cycle index i	sample index k	time $t = kT_s$ (s)	phase offset θ_0 (degrees)
1	1	0.0000	0.0000
2	593	0.0124	0.5280
3	1184	0.0247	0.4470
4	1775	0.0370	0.3660
5	2366	0.0493	0.2850
6	2957	0.0616	0.2040
7	3548	0.0739	0.1230

VII. APPLICATION

The Coriolis meter digital drive system described in this paper has seen widespread application over several years. Laboratory prototypes have been used in experimental work in

a range of field trials, but the most extensive validation has been through the thousands of commercial devices applied to industrial applications since the launch of the CFT-50 product in 2002. One of the key features of the product is its ability to maintain flowtube oscillation through two-phase flow, which combined with novel measurement techniques have led to a range of new applications, especially in the oil and gas industry [16]. The ability to maintain oscillation through two-phase flow is attributable to the digital waveform synthesis system, described in this paper, together with a higher-level non-linear amplitude control algorithm, which selects the amplitude of the drive signal in order to maintain the amplitude of oscillation at the desired set-point [4].

TABLE II
VALUES OF θ_0 FOR A DRIVE FREQUENCY OF 765Hz AND A CODEC
FREQUENCY OF 48kHz

cycle index i	sample index k	time $t = kT_s$ (s)	phase offset θ_0 (degrees)
1	1	0.0000	0.0000
2	64	0.0013	1.4625
3	127	0.0026	2.9250
4	190	0.0040	4.3875
5	252	0.0053	0.1125
6	315	0.0066	1.5750
7	378	0.0079	3.0375

7.1. Amplitude Control

Fig. 8 shows the results of an amplitude control experiment in laboratory conditions on a 25mm diameter flowtube, of the design shown in Fig. 1. In this example, the process fluid is single phase water, and so there are no difficulties in maintaining oscillation. However, the set-point, or desired amplitude of oscillation, is adjusted by three orders of magnitude over the course of 45 seconds of operation.

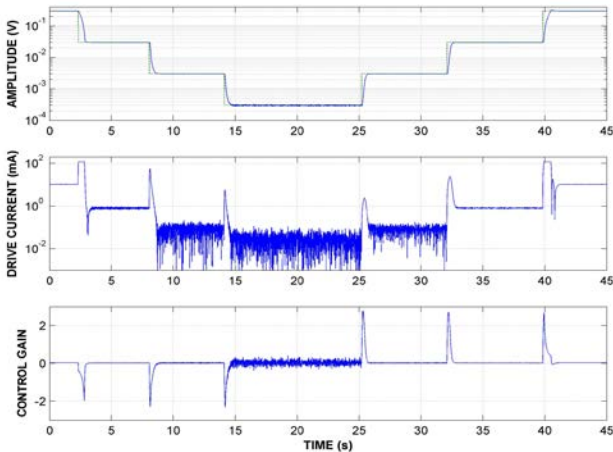


Fig. 8. Amplitude control experiment

The default set point for the amplitude of oscillation (0.3V) corresponds to a physical amplitude of 0.6mm. In the top graph, the set-point (dashed line) is reduced to 30mV, 3mV, and finally 0.3mV, before these steps are reversed. The actual amplitude is also shown, calculated every half cycle, at approximately 165Hz. At 0.3V, the cycle-by-cycle standard deviation (s.d.) of amplitude is approximately 0.001%, while 0.3mV, the s.d. is approximately 1%. Maintaining steady

amplitude is desirable for good flow measurement quality.

The central graph of Fig. 8 shows the drive current, corresponding to the amplitude of the synthesised waveform. The lowest graph shows the control gain, which is the ratio of the drive current to the sensor amplitude. Its mean value is approximately constant when the amplitude is steady. During set-point changes it exhibits large swings, and may become negative, requiring the use of negative gain as discussed above. Negative gain is useful for effecting set-point changes, and for maintaining flowtube stability at low amplitudes (e.g. for $t = 15\text{--}25\text{s}$) and with high damping (e.g. two-phase flow).

At the full amplitude of oscillation the drive output time series is unremarkable, consisting of a very steady sine wave in phase with the sensor signals, or more precisely their weighted sum. The amplitude of the drive output varies typically by 0.01% s.d., in order to ensure a sensor amplitude stability of 0.001% s.d. However, at the lower amplitudes of oscillation the unique features of the digital drive become more apparent.

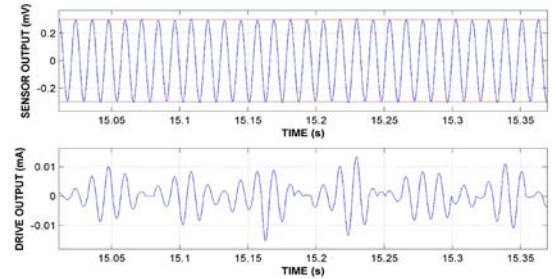


Fig. 9. Amplitude control at 0.3mV. 0.1% of normal set-point.

Fig. 9 shows a typical time series for the sensor voltage and drive output when the amplitude of oscillation is 0.3mV. Several features of the drive synthesis algorithm are demonstrated. The drive and sensor signals are substantially in phase with each other. The amplitude of the drive signal is updated every half cycle, and these updates occur at the zero crossing points, so that the transitions are smooth. Occasionally a negative gain value is used, in which case consecutive half-cycles of the drive output have the same polarity. These techniques are effective: the sensor amplitude stays within 1% of the set-point, despite operating at only 0.1% of the normal amplitude of oscillation. The ability to operate with precision and agility over a 1000:1 amplitude range is made possible by the 38-bit registers used to store and process sensor and drive data in the FPGA.

7.2. Two-phase Flow – Industrial Application

A Control Engineering article [14] describes an industrial batching application where two 50mm Coriolis meters placed in series are subject to two-phase transitions between a process liquid and air. Fig. 10 shows the resulting behaviour of the two meters. The upper two graphs show the density and mass flow readings from the CFT-50 digital Coriolis meter, while the lower graph shows the mass flow reading from the other commercial meter. At the start of each batch both flowtubes are empty, as indicated by the low density reading. When the flow begins (18s) the CFT-50 responds immediately, while the other meter requires an additional 16s to recover from the

hydraulic shock caused by the onset of flow, during which time approximately 200kg of product has passed. Further examples of two-phase flow performance are given in [7],[15],[16].

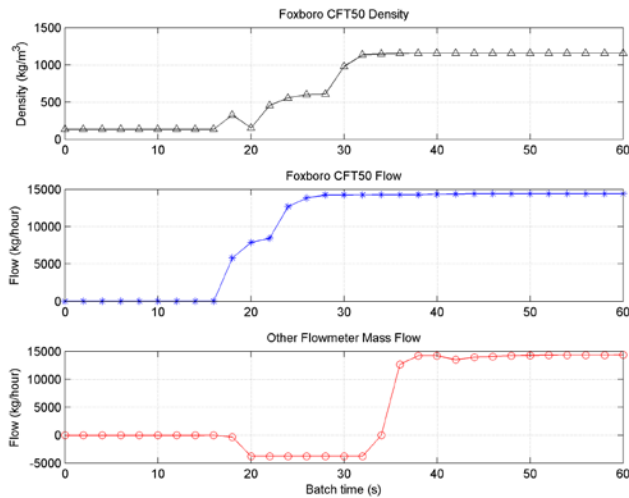


Fig. 10. Comparison between digital and conventional Coriolis meter during industrial batch application; both meters start empty.

VIII. SUMMARY AND FUTURE WORK

This paper has described the FPGA-based digital drive for the Coriolis mass flow meter, including aspects of its implementation using the Handel-C language. The benefits of this design have been illustrated by examples of how the meter is able to maintain operation through two-phase flow.

However, an essential aspect of Coriolis meter operation that has not been considered in this paper is flowtube start-up: with no signal from the flowtube, and hence no knowledge of the appropriate frequency and phase at which to drive the flowtube, how is oscillation to be initiated? A start-up capability has been provided through two additional techniques for driving the flowtube [19]. The first provides open-loop flowtube excitation, limited to the desired range of resonant frequencies, and is generated by appropriate filtering of a random sequence. The second is a form of digital positive feedback, which is used to rapidly drive up the amplitude of oscillation of the desired mode of vibration to the point where its frequency can be identified, so that full waveform synthesis can begin. These techniques will be described more fully in a subsequent paper.

FPGA technology has advanced rapidly since this design was first implemented, which raises the question of how the split in functionality between processor and FPGA could be redefined using the latest FPGA technology. It is unlikely that all Coriolis transmitter functionality would be implemented entirely in dedicated hardware: there are extensive requirements for floating point calculations, and the current VxWorks application code size is in excess of 100,000 lines of C++. However, the lowest level of measurement calculation, consisting of computationally intensive Fourier integration [3], could be implemented in dedicated FPGA hardware, bringing a number of benefits, including a significant reduction in

processor bus bandwidth while enabling higher ADC sampling rates and ultimately improved dynamic response [11]. In addition, the use of an embedded, possibly “soft” processor within the FPGA would provide seamless integration of the hardware and software aspects of the design, all within a single “system-on-chip” FPGA device.

ACKNOWLEDGMENT

The authors would like to thank all their colleagues at the Invensys UTC in Oxford for their support, especially Jez Bowles for his electronic design work.

REFERENCES

- [1] R. Cheesewright and C. Clark, “The effect of flow pulsations on Coriolis mass flow meters”, *J. Fluid. Struct.*, 12, pp. 1025-1039, 1998.
- [2] J. R. Reizner, “Coriolis – The almost perfect flow meter”, *IEE Computing & Control Engineering*, 14(4), pp 28-33, Aug. 2003.
- [3] M. P. Henry, D. W. Clarke, N. Archer, J. Bowles, M. J. Leahy, R. P. Liu, J. Vignos and F. B. Zhou, “A self-validating digital coriolis mass-flow meter: an overview”, *Control Eng. Pract.*, 8, pp 487-506, 2000.
- [4] D. W. Clarke, “Nonlinear control of oscillation amplitude of a Coriolis mass-flow meter”, *Eur. J. Control*, 4, pp 196-207, 1998.
- [5] W. le Roux and J. Daniel van Wyk, The Effect of Signal Measurement and Processing Delay on the Compensation of Harmonics by PWM Converters, *IEEE T. Ind. Electron.*, 47(2), pp 297-304, April 2000.
- [6] F. P. Dawson, and L. Klaffke, Variable-Sample-Rate Delayless Frequency-Adaptive Digital Filter for Synchronized Signal Acquisition and Sampling, *IEEE Trans. Ind. Electron.*, 46(5), pp 889-96, Oct. 1999.
- [7] M. Henry, H. Yeung, W. Mattar, M. Duta and M. Tombs, “How a Coriolis Mass Flow Meter can operate in Two-Phase (Gas/Liquid) Flow”, presented at ISA 2004, Houston.
- [8] E. Monmasson and M. N. Cirstea, “FPGA Design Methodology for Industrial Control Systems—A Review”, *IEEE T. Ind. Electron.*, 54(4), pp 1824-1842, Aug. 2007.
- [9] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, “Features, Design Tools, and Application Domains of FPGAs”, *IEEE T. Ind. Electron.*, 54(4), pp 1810-1823, Aug. 2007.
- [10] “DK Design Suite, Rapid path from C to FPGA implementation”, Agility Design Solutions, available: <http://www.agilityds.com>, Apr. 3, 2008.
- [11] C. Clark, M. Zamora, R. Cheesewright and M. Henry, “The dynamic performance of a new ultra-fast response Coriolis flow meter”, *Flow Meas. Instrum.*, 17(6), pp 391-398, Dec. 2006.
- [12] M. Hensen, “Three generations of Coriolis mass flow meter electronics”, National Engineering Laboratory Conference, Flow measurement 2001, Peebles, Scotland, May 2001.
- [13] M. Tombs, M. Henry, F. B. Zhou, R. M. Lansangan and M. Reese, “High precision Coriolis mass flow measurement applied to small volume proving”, *Flow Meas. Instrum.*, 17(6), pp 371-382, Dec. 2006.
- [14] “‘Breakthrough’ Coriolis meter allows filling machine to operate with exceptional precision”, *Control Engineering Europe*, pp 8-9, Jun. 2004.
- [15] M. P. Henry, “Precision polyethylene batch processing”, *Control Magazine*, pp 58, Jul. 2004.
- [16] M. Henry, M. Tombs, M. Duta, F. B. Zhou, R. Mercado, F. Kenyery, J. Shen, M. Morles, C. Garcia and R. Langansan, “Two-phase flow metering of heavy oil using a Coriolis mass flow meter: A case study”, *Flow Meas. Instrum.*, 17(6), pp 399-413, Dec. 2006.
- [17] Lansangan, R. Skinner, J and Henry, MP. “Wet Gas Measurement using Coriolis Mass Flow Metering – Wamsutter Field Trials”, Multiphase Measurement Roundtable, Houston, May 2007.
- [18] J. Montague and Control Engineering staff, “Finding gold”, supplement featuring 2002 Editors’ Choice award winners, *Control Eng.*, 50(1), pp 18, Jan. 2003.
- [19] M. P. Henry and M. E. Zamora, “Startup and Operational Techniques for a Digital Flowmeter”, US Patent No. 7,146,280, Dec. 2006.
- [20] M. S. Tombs, M. P. Henry, M. D. Duta, R. Langansan, R. E. Dutton and W. M. Mattar, “Multiphase Coriolis meter”, US Patent 7,207,229, Apr. 2007.

- [21] M. Zamora, M. P. Henry and C. Peter, "Generation of frequency output for instrumentation applications using digital hardware", *Sensor Review*, 23(2), pp 143-149, 2003.
- [22] M. Tombs, M. P. Henry, C. Peters, "From research to product using a common development platform", *Control Eng. Pract.*, 12, pp 503-510, 2004.
- [23] M. P. Henry, "Keynote Paper: Hardware compilation - a new technique for rapid prototyping of digital systems - applied to sensor validation", *Control Eng. Pract.*, 3(7), pp 907-924, 1995.
- [24] I. Page, "Constructing Hardware-Software Systems from a Single Description", *J. VLSI Signal Proc.*, 12(1), pp 87-107, 1996.
- [25] Y. F. Chan, M. Moallem, and W. Wang, "Design and Implementation of Modular FPGA-Based PID Controllers", *IEEE T. Ind. Electron.*, pp 1898-1906, 54(4), Aug. 2007.
- [26] M. N. Cirstea and A. Dinu, "A VHDL Holistic Modeling Approach and FPGA Implementation of a Digital Sensorless Induction Motor Control Scheme", *IEEE T. Ind. Electron.*, pp 1853-1864, 54(4), Aug. 2007.
- [27] Open SystemC Initiative OSCI, *SystemC Language Reference Manual*, 2006. [Online]. Available: www.systemc.org, Sep. 4, 2007.
- [28] M. Fleury, R. P. Self and A. C. Downton, "Hardware compilation for software engineers: An ATM example", *IEE Proc.-Softw.*, 148(1), pp. 31-42, Feb. 2001.
- [29] H. Al-Junaid, T. Kazmierski, P. R. Wilson and J. Baranowski, "Timeless Discretization of Magnetization Slope in the Modeling of Ferromagnetic Hysteresis", *IEEE T. Comput. Aid. D.*, 25(12), pp 2757-2764, Dec. 2006.
- [30] K. Benkrid, A. Benkrid, S. Belkacemi, "Efficient FPGA hardware development: A multi-language approach", *J. Syst. Architect.*, 53, pp 184-209, 2007.
- [31] F. F. Dai, C. Stroud, and D. Yang, "Automatic Linearity and Frequency Response Tests With Built-in Pattern Generator and Analyzer", *IEEE T. VLSI Syst.*, 14(6), pp. 561-572, Jun. 2006.
- [32] J. Qin, C. E. Stroud, and F. F. Dai, "FPGA-Based Analog Functional Measurements for Adaptive Control in Mixed-Signal Systems", *IEEE T. Ind. Electron.*, pp 1885-1897, 54(4), Aug. 2007.
- [33] "A technical tutorial on digital signal synthesis," Analog Devices, Inc., Technical Report, 1999.
- [34] L. Cordesses, "Direct digital synthesis: a tool for periodic wave generation (part 1)," *IEEE Signal Process. Mag.*, 21(4), pp 50-49, Jul. 2004.
- [35] C. S. Turner, "Recursive discrete-time sinusoidal oscillators," *IEEE Signal Process. Mag.*, 20(3), pp 103-111, May 2003.

flow metering. Dr Henry is a Fellow of the Institute of Measurement and Control, a visiting Professor of Chongqing University, and Deputy Chair of the IEEE Committee for System on Chip, and has 25 patents on flow measurement and self-validating systems.



Mayela Zamora

Mayela Zamora graduated with a B.Sc. degree in Electronic Engineering at Simón Bolívar University, Venezuela, in 1986. In 1996 she received her M.Sc. in Electric Engineering at Universidad Central de Venezuela, and in 2001 she completed her D.Phil thesis "The Study of the Sleep and Vigilance Electroencephalogram Using Neural Network Methods" at Oxford University, England. Since then, she has worked at the Invensys University Technology Centre (UTC) for Advanced

Instrumentation at Oxford, where she is carrying out research and developing software for the design of self-validating instruments.



Manus Henry (M'04)

Dr Manus Henry has worked at the Department of Engineering Science, University of Oxford, since 1987, researching into self-validating (SEVA) sensors, with a particular interest in Coriolis mass flow meters. His current position is Deputy Director of the Invensys University Technology Centre (UTC) for Advanced Instrumentation. In 1999 he won the IEE Control Division's Younger Engineer Award for the development of the digital Coriolis mass flow meter; commercialised in 2002 by Invensys Foxboro,

this device has won several prizes for its ability to measure two-phase flow. In 2007 the UTC won the IET Measurement Prize for its work on Coriolis mass