Jonathan Hill

The Soft-Core Discrete-Time Signal Processor Peripheral

In just the last few years, the personal electronics market has seen explosive growth. Along with entirely new consumer products, there are concomitant developments in computer engineering. Although the system on a chip (SoC) concept is mature, the increasing density and sophistication of the field programmable gate array (FPGA) is introducing new design methodologies and higher levels of integration. While a single FPGA alone can implement an entire microprocessor system, manufacturers also provide the option of having an actual microprocessor integrated into the FPGA chip. In both situations the FPGA fabric (defined in this article as a two dimensional array of identical logic blocks in an interconnect resource) implements custom peripherals and other devices. The FPGA has essentially become a breadboard, allowing for truly rapid product development.

With SoC systems, there is a growing need for modest performance, low-cost solutions that make minimum use of FPGA resources and make the most effective use of the features in an FPGA. For such systems, it can be undesirable to provide an entire digital signal processing (DSP) core either on the same die or in a separate package. This article contends that for products having such modest signal processing requirements, the FPGA provides an ideal means to customize a peripheral in a system to the given application. To exemplify this point, two examples utilizing a discrete time signal processing (DTSP) toolkit are presented.

FPGAS AND THE SoC CONCEPT

The SoC is a computer system where all the components are in a single chip. The point of achieving such a level of integration is to reduce costs and improve performance. The SoC concept is well understood; references include [1], [2], and [3]. The chip itself may be a FPGA, an application specific integrated circuit (ASIC), a full or semi-custom design, or a mixture of these technologies. In a SoC, the notion of a core is that of a building block so that each system can be implemented a number of ways and interconnected on the same chip using a common bus. Microcontrollers typically use a generic central processing unit (CPU) that is a full custom designed core, integrated with memory and selected peripheral devices.

[THE FPGA HAS ESSENTIALLY BECOME A BREADBOARD, ALLOWING FOR TRULY RAPID PRODUCT DEVELOPMENT.]

The logic blocks of the FPGA contain lookup tables to represent arbitrary combinational logic as well as flip-flops. At the periphery of the FPGA fabric are blocks that handle input and output. In addition there may be special function blocks such as random access memory (RAM) and hardware multipliers. Some FPGAs also provide special interconnect routes used with fast-carry in performing arithmetic. FPGA device manufacturers include Actel, Altera, Atmel, Lattice, and Xilinx, among others. A microprocessor or other device described with a hardware description language and implemented solely in the fabric of an FPGA is referred to as being a soft core. In contrast, actual microprocessors such as the PowerPC, Atmel AVR, and others that are integrated directly in an FPGA chip are referred to as being hard cores.

The FPGA is particularly useful in research and product design. In using an FPGA, the designer can flexibly write a hardware description in a process similar to writing software. Significant changes can be made to a description, then simulated, and immediately tested in real hardware without incurring any tooling charges. With an FPGA, the notion of prototyping is primarily in reference to the development board on which the FPGA is mounted. Without a recurring tooling charge, the FPGA improves the feasibility of products that are highly technical yet low in volume. This is particularly true in research where only a few examples may be manufactured for laboratory use.

As a professor, I use manufactured development boards in my research and in teaching. The Spartan-3 Starter Board provides expansion connectors for use with application specific interface adapter cards. The Xilinx Embedded Development Kit (EDK) is also used to write the system hardware description as well the application software. Given that such a system must necessarily include both hardware and software, to avoid confusion I use the term “configure” to refer to the act of programming hardware and the term “program” in reference to application software. Applications that we have developed include an electrocardiogram, stethoscope, motor controller, and of course, the discrete-time signal processing toolkit that is described in this article.

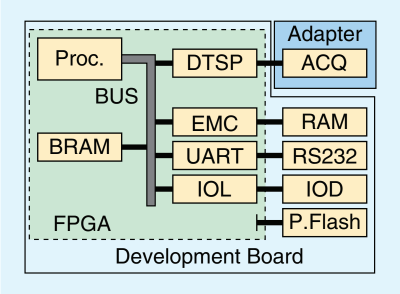
Due to its flexibility in design, the FPGA provides an ideal means to design peripherals that are customized to a given application and use a minimum of resources. Consider that, in serial

Digital Object Identifier 10.1109/MSP.2008.931090

data communications, if there is no need to change the signaling rate and without a need for parity checking, these features can be removed from the hardware description.

SPECIALTY PERIPHERAL DEVICES

Figure 1 is a generic block diagram for both example systems presented later in this article. The primary difference between the examples is the description of the DTSP peripheral device controlling the acquisition card (ACQ). The FPGA is a Xilinx Spartan 3, XC3S1000 device that provides roughly the equivalent of one million gates and also has 54 kB of Block RAM. The processor (Proc.) is a Microblaze soft-core device. The external memory controller (EMC) provides access to an additional 1 MB of onboard memory (RAM). The Universal Asynchronous Receiver/Transmitter (UART) and the interface logic (RS232) provide serial communications. Input-output logic (IOL) and input-output devices (IOD) comprising buttons, switches, light-emitting diodes (LEDs), and a four digit seven-segment display provide a simple user interface. At system start up and following reset, the image stored in the platform Flash (P.Flash) is loaded to configure the FPGA. Any unused resources are not shown in Figure 1.

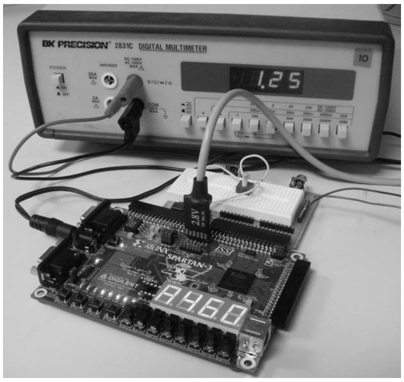


[FIG1] **DTSP processor system example.**

In considering signal processing the question may arise, “Why not just use a DSP chip?” An answer is that not all applications require the flexibility and power that a DSP provides. In SoC applications having modest signal processing needs, a custom soft-core peripheral that uses a minimum of resources is quite adequate. While the peripherals in Figure 1 are relatively simple, other devices can be quite complicated, perhaps containing a microprocessor. Examples include a disk drive controller, video display controller, Ethernet adapter, and keyboard interface. The point is that peripherals are tailored devices and the FPGA is a natural fit in their design, allowing the designer to incorporate just the right amount of processing power for a given application.

DTSP

The DTSP toolkit is being developed to demonstrate, as well as provide a means to investigate, DSP, both for research and instructional purposes. As an introductory tool, the toolkit provides students with a mechanism for deductive investigation of an existing system. Such a toolkit will help students develop an intuition for DTSP. The toolkit demonstrating analog to digital and digital to analog conversion is shown in Figure 2. The acquisition card in the center of the figure has a potentiometer providing the input and the voltmeter measures output.



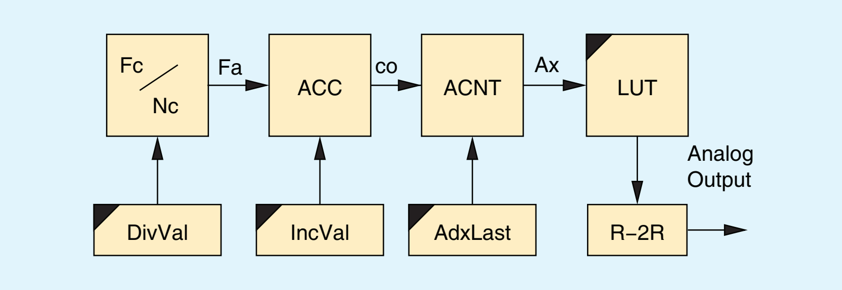
[FIG2] **DTSP toolkit and voltmeter.**

As outlined earlier, the toolkit is based on the Spartan-3 Starter Board using a XC3S1000 FPGA chip. The acquisition card is our own design. Analog to digital conversion is performed with the Analog Devices AD7819 chip which produces 8-b samples at rates up to 200,000 samples/s. Digital to analog conversion is performed with a conventional R-2R ladder network. Details regarding the toolkit are available at the project Web site [4]. The FPGA development board itself is an off-the-shelf product and the acquisition card can be constructed or breadboarded with the standard parts. In applying the GNU general purpose license to our code, artwork, and documentation, the material is free for your use.

ARBITRARY WAVEFORM GENERATOR

Figure 3 outlines the digital arbitrary waveform generator (DAWG) peripheral. Each box that has its upper-left corner marked is accessible from the host microprocessor. To use the DAWG, values are first written to the lookup table (LUT). The clock (*Fc*) is divided by DivVal, producing *Fa* used to enable the accumulator (ACC). The ACC is a phase accumulator like that in a digital frequency synthesizer, and based on the increment value (IncVal) provides very fine frequency adjustment capability. The carry out (*co*) enables the address counter (ACNT), producing addresses (*Ax*) from zero to AdxLast, used by the LUT.

DivVal and AdxLast each are 8-b values and the LUT produces 8-b values. The value N_R is the number of bits used to represent the accumulated phase in ACC. While IncVal is a 32-b value, to ensure normal behavior the accumulated phase is stored in a 33-b register. The rate



[FIG3] **Waveform generator.**

the ACC is updated in (1). The frequency step size for the accumulator output is in (2). With a 50-MHz system clock and DivVal set to ten, each unit in IncVal corresponds to 0.5821 mHz. The frequency of *co* driving the address counter ACNT is in (3) and the period is (4).

$$F_a = \frac{F_c}{\mathrm{DivVal}}, \qquad (1)$$

$$F_{\mathrm{cs}} = \frac{F_a}{2^{N_R}}, \qquad (2)$$

$$F_{\mathrm{co}} = \frac{F_c \cdot \mathrm{IncVal}}{2^{N_R} \cdot \mathrm{DivVal}}, \qquad (3)$$

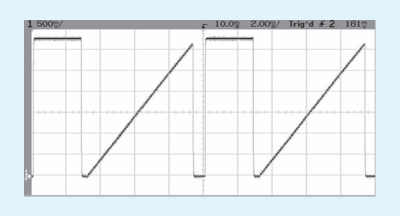
$$T = \frac{2^{N_R} \cdot \mathrm{DivVal}}{F_c \cdot \mathrm{IncVal}} (\mathrm{AdxLast}+1). \qquad (4)$$

To produce a waveform 100 samples long in a 10-ms period with the values above, AdxLast was set to 99 and IncVal was selected for the period of *co* to be 0.1 ms. A small program in C first writes sample values into the LUT and then instructs the device to generate the waveform, shown in Figure 4.

In reviewing the synthesis report, without optimizing the design and including the bus access logic DAWG consumes roughly 4% of the block RAM and roughly 2% of the fabric of a XC3S1000 device. In comparison, a generic Microblaze system constructed for this development board consumes roughly 13% of the fabric resources. Since Microblaze uses RAM in sizes that are a power of two, memory is left over for use with this peripheral.

FINITE IMPULSE RESPONSE FILTER

For some applications, a finite impulse response (FIR) type filter [5] is required. Rather than using the brute-force approach of directly mapping to hardware the resources in the block diagram for a FIR filter, given the speed of the hardware multipliers we can still achieve respectable sample rates by directly implementing the convolution sum. While well known, a brief outline of the convolution sum provides insight into the filter structure. As in (5), when the input x is a unit impulse, the output y is the corresponding impulse response h. We assume here that the impulse response is causal or zero for n less than 0. We next assume the system is linear and time invariant so that if the input is a scaled and delayed impulse, then in (6) the output is scaled and delayed in the same fashion.



[FIG4] Example waveform.

$$x[n] = \delta[n],\ y[n] = h[n], \qquad (5)$$

$$\begin{aligned} x[n] &= a[k]\,\delta[n-k] \\ y[n] &= a[k]\,h[n-k]\cdot \qquad (6) \end{aligned}$$

In applying superposition, given a sampled input x waveform that is a sum of scaled and delayed impulses, the output is the sum of scaled and delayed impulse responses (7). We also assume the impulse response $h[n]$ is finite in length with M samples, so that changing variables leads to (8).

$$\begin{aligned} y[n] &= \sum_{k=0}^{n} a[k]\,h[n-k] \\ y[n] &= \sum_{\lambda=0}^{M-1} a[n-\lambda]\,h[\lambda], \qquad (7) \end{aligned}$$

where

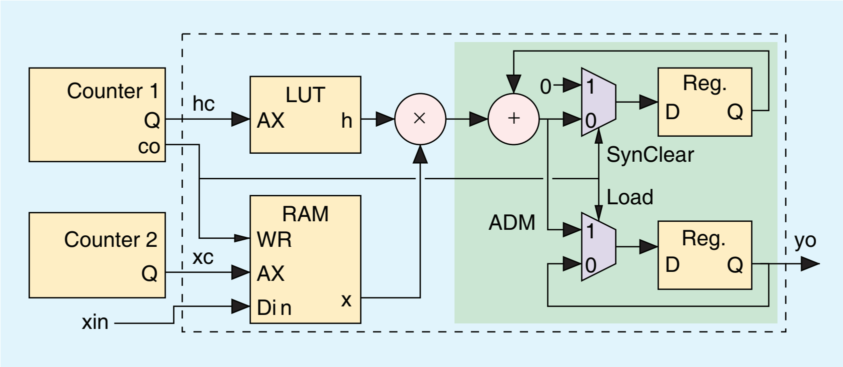
$$h[\lambda] = 0 \quad \text{for } \lambda < 0 \quad \text{and } \lambda \geq M. \qquad (8)$$

Equation (8) suggests a filter structure where one multiply-accumulate operation is performed during each clock cycle. The sample rate is limited by the length of the impulse response; this is improved by subdividing the convolution into stages, allowing for a cost-to-performance tradeoff. With clock frequency F_c and S stages, the maximum sampling rate $F_{s\,\max}$ is given in (9). With a 50-MHz clock and an impulse response with 500 samples, the sample rate for one stage is 100 kHz.

$$F_{s\,\max} = \frac{F_c}{M} S. \qquad (9)$$

In Figure 5, the dashed-line box contains one convolution stage. Counter 1 produces *hc* to read impulse response values from the LUT. The RAM stores input samples with address values from Counter 2. The impulse response is M samples long, Counter 1 is modulo M, and Counter 2 is modulo $M-1$. The output *yo* is produced with the accumulate-and-dump module (ADM) constructed with multiplexers and registers. The RAM and LUT are fully synchronous and chosen as they are fairly high density and separate resources so that their use does not compete for resources in the FPGA fabric.

For this example, I wrote a program to calculate samples of an exponentially decaying 8 kHz sine wave with a 0.25-ms time constant. Such an impulse response behaves as band pass filter. The sampling rate is 100 kHz and to represent four time constants; one hundred samples were written to the LUT. Without optimizing the design and including the bus access logic, the FIR filter consumes roughly 3% of the fabric in a XC3S1000 device.



[FIG5] Convolution stage.

CONCLUSIONS

The increasing density and sophistication of the FPGA is introducing new design methodologies to the SoC concept. The FPGA fabric is used to implement custom peripherals and other devices, as demonstrated in the examples presented in this article. With continuing improvements in FPGA technology and density, one may expect soft-core implementations to have an increasingly important role in signal processing applications of modest requirements.

AUTHOR

Jonathan Hill (jmhill@hartford.edu) is an assistant professor in the Electrical and Computer Engineering Department at the University of Hartford in Connecticut. He instructs graduate and undergraduate computer engineering courses, directs graduate research, and performs research involving embedded microprocessor based systems. Specific projects involve digital communications, signal processing, and intelligent instrumentation. He is a Member of the IEEE.

REFERENCES

[1] S. Hauck and A. DeHon, *Reconfigurable Computing*. San Mateo, CA: Morgan Kaufmann, 2008.

[2] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, 3rd ed. The Netherlands: Kluwer, 2002.

[3] R. Rajsuman, *System-on-a-Chip: Design and Test*. Norwood, MA: Artech House, 2000.

[4] J. Hill, *DTSP Toolkit* (13 Jan. 2009) [Online]. Available: http://uhaweb.hartford.edu/jmhill/projects/DTSPkit/index.htm

[5] A. V Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

Cooperative Communications and Networking
K. J. Ray Liu, Ahmed K. Sadek, Weifeng Su and Andres Kwasinski
Quickest Detection
H. Vincent Poor and Olympia Hadjiliadis
APPLIED SPEECH AND AUDIO PROCESSING
With Matlab® Examples
Ian McLoughlin
Fundamentals of Digital Imaging
H. J. Trussell and M. J. Vrhel