A 256 Channel Control System Using FPGAs and a PLD

Dave DeLauter, Consultant
DeltaT

This paper describes the development process and the latest design iteration of a simple Pulse Width Modulation (PWM) Digital to Analog Converter (DAC) system. From an original micro based design to the high speed serial implementation and the extensions described in this paper, the design seems to continuously evolve with each step opening new possibilities. The unique flexibility of PLDs and FPGAs allows the design to have the capability of implementations from a few 16 bit PWM DACs in an FPGA up to 256 DACs on a printed circuit board. It also allows the use of more DACs of less resolution per FPGA such as 8 bit or even smaller. Using a fast PLD for the microwire-like interface and then dispersing data to up to 256 DACs implemented in slower FPGAs is the current level of evolution.

Background

Since we knew that PWMs would work in our application, our original design called for using a micro and its PWM capabilities to control few analog signals. As the design continued we found more places where PWM control would be adequate and would allow software control of sections of the product that were not even considered earlier. This additional level of control would allow software customization of each implementation while increasing the reliability of the system.

As the number of possible PWMs being considered grew, we also decided that it would be useful to drop the micro from certain implementations of the design and use a parallel PC port interface with the PWMs in an FPGA. When we approached our customer with this idea, he reacted with more possible applications which would include as many as 256 PWM DACs.

Design

The design was then changed to have one “master” FPGA controlling the parallel port and distributing data to as many “slave” FPGAs for the PWM DACs as the implementation required (Figure 1). In some previous designs we had used PWM DACs in an Actel 1280. In this case we decided initially to put as many 12 bit PWM DACs in an Actel 1020 as possible and then clone that part several times on the board. This structure was developed for the prototype. The actual design for PWM DACs is well known as is the design of various parallel port interfaces. The use of the multiple 1020s to add as many DACs as needed for a particular customer application allows us to use one printed circuit board for this design. The Actel 1020s change to support specific implementations.

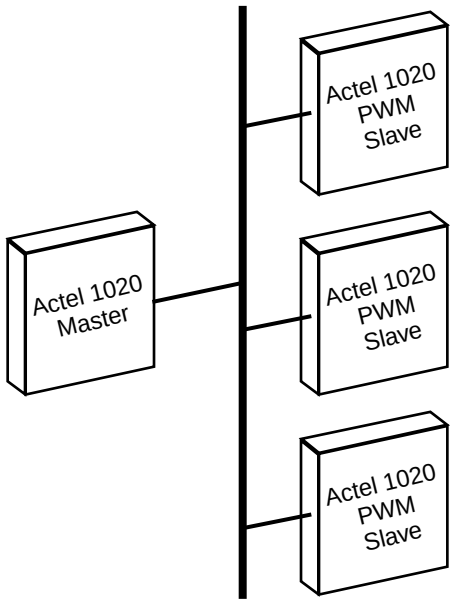

Figure 1 • *Parallel Port Diagram*

PWM Design

As indicated earlier there is nothing unique about the PWM structure we are using. Our PWMs consist of a counter, a latch, a comparator, and a flip flop. The latch data is compared to the counter on each clock cycle and when they are equal the flip flop is set (Figure 2). The flip flop gets reset by the carry out of the counter and the cycle begins again. In the Actel implementation, setting the flip flop drives the enable of a tristate pin that has ground as its input. This pin in the simplest case drives a resistor-capacitor combination to convert the asymmetry of the waveform into a DC level with a 50 percent duty cycle being the nominal midpoint. The latches we use are reset to a 50 percent duty cycle for convenience. With this structure, a zero in the latch will force the output pin to always be ground, so that the external resistor-capacitor combination (i.e., the DAC output) will

Paper published in the 1995 PLD Conference Proceedings

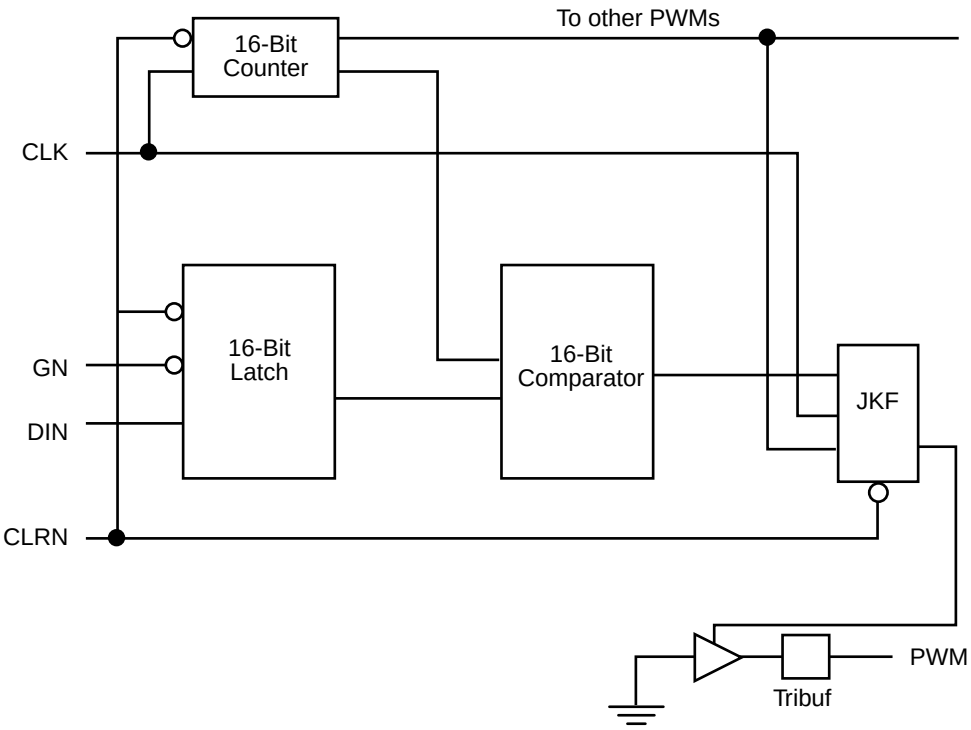

Figure 2 • *PWM Structure*

also be ground. As the latch increases in value, the output pin stays high longer which causes a corresponding increase in the DAC output voltage. The frequency of the counter sets the basis of the ripple of the DAC and also the response time.

Only one counter is required to support as many comparators and latches as the FPGA will hold. There are also tricks that can be done such as in a mostly 16 bit FPGA adding a carryout from the counter at 8 bits to have some faster frequency 8 bit DACs. In our implementations, we have not used a loadable counter since this changes the range of the DAC. There are more minimal gate implementation, but the software for them is more complicated as well. In this implementation, an 8 bit PWM requires 21 modules, a 12 bit, 30 modules, and a 16 bit 39 modules. The latch-comparator-flip flop combination is replicated usually to a nice number or if only a few parts are used to the maximum per FPGA. In some systems the input muxes are added to allow reading the latch data and in some cases the counter output by the software. No attempt has been made yet to read in the microwire design.

Microwire

The latest evolution of the design is to allow this customer to move his system or systems further from the computer by using a high speed dedicated microwire interface to replace the parallel port interface. The original idea was to have a simple 8 bit microwire type port and then feed data to the Actel 1020s for the PWM control.

When we designed the original microwire version we simply converted the parallel port FPGA to microwire. This we realized would have speed limitations which would slow down the control rate. This paper is the result of rethinking that portion of the design and using a high speed PLD to allow a faster microwire for those situations. In this sense it also reflects a design that is evolving. We could have used a shift register for the very high speed section and then moved bytes at the byte rate to the FPGA. Instead we are using a PLD which will allow us to include more control and sync logic as well as quickly changing the data path width without changing our board.

When we were thinking about this paper, we realized that the whole structure can be applied in more general applications with each FPGA being different if that is required. Some PWMs could be 8 bit which would allow more per part and/or a higher frequency versus others at 16 bit for better resolution. We have only begun to look at that structure, but the inherent nature of the FPGA allows for that kind of change without changing the pcb every time. Also, for this paper we have looked at using a pinout for the ACT 1 PLCC84 that would also work for the Actel 1240 or even 1280 for some more complex design changes by adding cuts and jumpers. This has not been tested.

The initial high speed version of the board will use a PA7024 for the microwire. The pcb will also be set to allow jumpering directly to the master control FPGA allowing for a slower

speed minimal implementation with a master Actel 1020. The microwire version of the design is completely new and has yet to be tested outside the lab.

The initial microwire design is set up so that an ICT PA7024 receives data from the high speed serial bit link and then transfers bytes to an ACT 1 FPGA that does chip select, control, and buffering of the DAC data and address (Figure 3). This data and address is then passed to one of the individual slave PWM FPGA modules on the board. Each of the FPGA slaves contains its own counter which is run from the master clock on the board.

The PA7024 fully buffers the high speed serial data. The FPGAs run on a much slower independent clock. The speed of the FPGA clock is a function of the desired frequency of the PWMs. Higher frequencies will have less ripple and/or quicker response. Since in our application most signals do not change or do not change very often, we use a slower 8 Mhz clock. There is a lot of flexibility here. At 24 Mhz with a 3 byte per channel overhead the rate of change of one channel is 1 µsec so changing all channels is a 256 µsec minimum. In a parallel port system with a nominal 30K byte per second capacity, these numbers become 10K channels per second max or 100 µsec per channel. In some applications we have used block io software commands to output when all channels have to be written, but for the most part very few channels in our system change at once so that the actual data rate to the channels is fine at 10K or even less. In one instance where we needed more speed we reduced the data width and number of channels so that the standard parallel port worked. It is the ability to adapt the design that makes the PLDs and the FPGAs of value here by extending generic circuit designs without completely new boards for each design.

While we have not needed to change yet, we anticipate using fast parallel ports, speeding up the software, and using the higher speed PLD design discussed here for future implementations. In addition several remote microwire sites could be controlled from the central control location. The PLD speed also opens the possibility for future designs involving interfaces to coax or fiber. At the moment it seems more practical to let the PC handle this level of interface if it is required.

Summary

What started as a limited scope application has evolved into a larger structure which is being designed with the goal of minimal board changes components and maximum flexibility of implementation. The use of both PLDs and FPGAs increases this flexibility. In addition each stage of the design seems to open more possibilities for the customer as well as the designers.

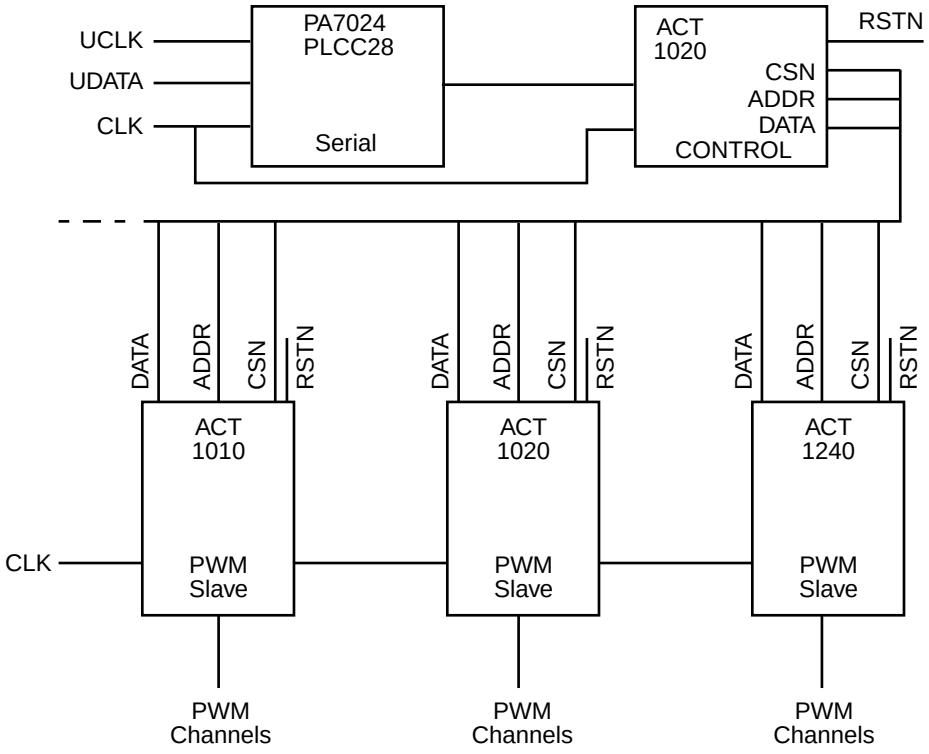

Figure 3 • *Microwire Diagram*