Many ASICs Can Easily be REPLACED with FPGAs

by Austin Lesea, Principal Engineer,
austin.lesea@xilinx.com

One of the first things you usually notice about a printed circuit board design is the use of the "big chips." It is not hard to find the microcontroller, the memories, the FPGAs, and the ASICs. And, sometimes the unused capacity in one FPGA is enough to replace several ASICs. If you want to reduce your overall development/ manufacturing/test costs it often makes sense to incorporate ASIC functions into your FPGAs.

Telecommunications

I'll discuss three different common tasks performed in telecommunications: framing, multiplexing, and performance monitoring.

In a typical digital communications application, there may be a T1 or E1 framer ASIC. These devices usually cost about $20 in quantity, and they may be underutilized by the application. Sometimes, only the transmit section, or just the receive section are used, or sometimes the pattern being generated is fixed, and the device is not being used to its fullest. The description of these functions is available from a number of sources such as ITU G-series documents (G.703 for example), or from ANSI, ATIS T1 documents.

"If you want to reduce your overall development/ manufacturing/test costs it often makes sense to incorporate ASIC functions into your FPGAs."

Digital multiplexers/demultiplexers (MULDEM's) are also common ASICs. They take a number of T1 or E1 signals, multiplex and demultiplex them, to and from a higher rate signal. Again, you can refer to the ITU or ANSI standards to get all the information you need to implement the function in an FPGA. Quite frequently, if the application also involves a fiber optic channel, or a radio link, other logic is required, which can easily be implemented on the same FPGA.

Performance monitoring consists of accumulating bit errors, coding violations, out of frame conditions and CRC errors. It must also keep track of these in light of error statistics. In these applications, monitoring bit streams at rates from 1 to 50 megabits per second is not a task that is easily performed by a microcontroller. To detect and keep track of the events, and then present them in a digested fashion to the microcontroller, allows for more features and higher performance. These circuits are all simple counters, shift registers and multiplexers, easily implemented in an FPGA.

Signal Processing

Digital finite impulse response filters (FIRs) are common elements of any communications system as well as control systems. Depending on the speed, resolution and number of taps required, an FPGA may be a good choice.

One such application is the root Nyquist transmit filter. In any channel, to provide for zero inter-symbol interference (ISI), you need to filter the transmit symbols. Many designs take the optimal ISI-free filter for a channel and split it into two parts: half at the transmit end, and half at the receive end. This also minimizes transmit bandwidth in the channel. Taking half of a filter is the same as the square root of the response, hence the name root Nyquist filter.

> *"By designing ASIC functions into an FPGA, you usually save money, power, and board space. Once designed, the function becomes part of your company's intellectual property, and can be re-used."*

A typical FIR structure for the transmit filter is a shift register which is clocked at three times the symbol rate (or more), where the outputs or taps of the flip-flops in the register pass through resistors to a summing junction. The resistor values are chosen to set the gains in the taps of the filter. The sign of the value to be summed is chosen by selecting the normal, or the inverting output of the register. In one example, a 22-tap FIR is easily implemented in the I/O blocks of the FPGA along one side of the device. The output of the summing junction need only pass through a simple low pass filter to remove the sampling clock and the harmonics. Such filters are commonly used in all digital radio systems. Each bit of a modulation format's symbol requires such a filter, so for QAM, two such filters are required, and for 16QAM, four such filters are required.

Resistor/register FIR structures are useful to symbol rates up to a few mega symbols per second, and are easily implemented in an FPGA.

General Purpose Applications

Another good example of the "no more ASICs" design philosophy is in forward error correcting. Most communication channels have errors that occur (fiber, radio, magnetic, or metallic based channels) and need some amount of error correction to make the channel useful. Rather than buying ASICs to do the job, again it makes sense to perform the functions in an FPGA. Some forward error correction algorithms require a large memory block, but using an external RAM device is still less expensive than the ASIC alternative.

Some error correcting schemes are fairly easy to implement, and require only feedback shift register structures, such as Reed-Solomon codes. The simple schemes have a high overhead; they do not correct many errors per block or byte in relation to the extra bits required. These codes typically require 50% or more bits as check bits.

More powerful error correcting codes are popular where bandwidth or more bits becomes a liability. These are the Bose-Chaudhuri-Hocqueghem (BCH) codes. These codes can correct both random and burst errors, and can recognize when they cannot correct the errors. The more powerful the error correction scheme, the more memory is required. Some BCH codes, 511 bits for a 493-bit block for example, will correct one, two, or three bit errors in any block. The efficiency of this code is that it only adds less than 4% more bits to the channel.

Frequency synthesis is another area where the often expensive and single-sourced parts may be easily replaced by an FPGA. Fractional synthesizers, pulse swallowers, direct digital frequency synthesizers, as well as the phase detectors for phase locked loops, are all easily implemented in FPGAs.

Conclusion

By designing ASIC functions into an FPGA, you usually save money, power, and board space. Once designed, the function becomes part of your company's intellectual property, and can be re-used. For example, if an application needs to change from being a T1 to and E1 design, often only the FPGA program needs to change, and the board remains the same. In fact, some designs initialize as T1 or E1 depending on configurations stored in memory, once the application is selected.

If at some point the standards change, or the competition adds some highly desirable feature (or you want to add a new feature), the FPGA also gives you a future - it can be easily changed. If an ASIC is part of the design, you will end up with a board re-layout, and probably have to add an FPGA to "band-aid" the design. Why not get the design right the first time? Σ