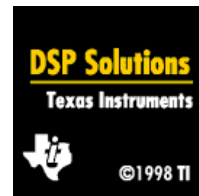




HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.demon.co.uk
URL: <http://www.hunteng.co.uk>



Choosing DSP or FPGA for your Application

Rev 1.0 R.Weir 10-01-2001

Introduction

The HERON range of modular DSP systems supports processing in either FPGA or DSP. The two approaches are markedly different. Here we look at when to use FPGA, and when to use DSP.

The Two Solutions

The DSP is a specialised microprocessor - typically programmed in C, perhaps with assembly code for performance. It is well suited to extremely complex maths-intensive tasks, with conditional processing. It is limited in performance by the clock rate, and the number of useful operations it can do per clock. As an example, a TMS320C6201 has two multipliers and a 200MHz clock – so can achieve 400M multiplies per second.

In contrast, an FPGA is an uncommitted “sea of gates”. The device is programmed by connecting the gates together to form multipliers, registers, adders and so forth. This is done at a block-diagram level. Many blocks can be very high level – ranging from a single gate to an FIR or FFT. Their performance is limited by the number of gates they have and the clock rate – so for example, a 200K-gate Virtex device with a 200MHz clock could implement ten 16-bit multipliers. This offers huge performance – especially as the 200K-gate device is considered small, with 1-M gate devices available.

Where they Excel

Beyond a sampling rate of a few MHz, a DSP can only implement a very simple operation on the data. Implementing that simple operation using the FPGA would be easy, with the option of far higher sampling rates. At low sampling rates, that situation is reversed – the DSP can implement massively complex programs, which would be hard to replicate in the FPGA.

With low-rate events, the DSP scores. It can queue these, ensuring that they all get processed – but there may be some latency before they are handled. In contrast, the FPGA cannot handle as many events – each one must have dedicated hardware – but each event can be handled at the same time as all the others.

If a major context switch is required, the DSP can implement this by branching to a new part of the program. In contrast, an FPGA needs to build dedicated resources for each configuration. If the configurations are small, then several can exist in the FPGA at the same time. Larger configurations mean the FPGA needs reconfigured – a process which can take some time.

The DSP can take a standard C program and run it. This C code can have a high level of branching and decision making – for example, the protocol stacks of communications systems. This is difficult to implement within an FPGA.

Finally, an FPGA is programmed in a “block-diagram” style, where it is easy to see the dataflow. A DSP is programmed with a sequential flow of instructions. Most signal processing systems start life as a block diagram of some sort. Actually translating the block diagram to the FPGA may well be simpler than converting it to C code for the DSP.

Making a Choice

There are a number of elements to the design of most signal processing systems, not least the expertise and background of the engineers working on the project. These all have an impact on the best choice of implementation. In addition, consider the resources available – in many cases, HERON I/O modules have FPGAs on board. Using these with a DSP processor may provide an ideal split.

As a rough guideline, try answering these questions:

1. What is the sampling rate of this part of the system? If it is more than a few MHz, FPGA is the natural choice.
2. Is your system already coded in C? If so, a DSP may implement it directly. It may not be the highest performance solution, but it will be quick to develop.

3. What is the data rate of the system? If it is more than perhaps 20-30Mbyte/second, then FPGA will handle it better.
4. How many conditional operations are there? If there are none, FPGA is perfect. If there are many, a software implementation may be better.
5. Does your system use floating point? If so, this is a factor in favour of the programmable DSP. None of the Xilinx cores support floating point today, although you can construct your own.
6. Are libraries available for what you want to do? Both DSP & FPGA offer libraries for basic building blocks like FIRs or FFTs. However, more complex components may not be available, and this could sway your decision to one approach or the other.

In reality, most systems are made up of many blocks. Some of those blocks are best implemented in FPGA, others in DSP. Lower sampling rates and increased complexity suit the DSP approach; higher sampling rates, especially combined with rigid, repetitive tasks, suit the FPGA.

Some Examples

Here are a few examples of signal processing blocks, along with how we would implement them:

1. First decimation filter in a digital wireless receiver. Typically, this is a CIC filter, operating at a sample rate of 50-100MHz. A 5-stage CIC has 10 registers & 10 adds, giving an “add rate” of 500-1000MHz.

At these rates any DSP processor would find it extremely difficult to do anything. However, the CIC has an extremely simple structure, and implementing it in an FPGA would be easy. A sample rate of 100MHz should be achievable, and even the smallest FPGA will have a lot of resource left for further processing.

2. Communications Protocol Stack – ISDN, IEEE1394 etc; these are complex large pieces of C code, completely unsuitable for the FPGA. However the DSP will implement them easily. Not only that, a single code base can be maintained, allowing the code stack to be implemented on a DSP in one product, or a separate control processor in another; and bringing the opportunity to licence the code stack from a specialist supplier.

3. Digital radio receiver – baseband processing. Some receiver types would require FFTs for signal acquisition, then matched filters once a signal is acquired. Both blocks can be easily implemented by either approach. However, there is a mode change – from signal acquisition to signal reception.

It may well be that this is better suited to the DSP, as the FPGA would need to implement both blocks simultaneously. Note that the RF processing is better in an FPGA, so this is likely to be a mixed system.

(Note – with today’s larger FPGAs, both modes of this system could be included in the FPGA at the same time.)

4. Image processing. Here, most of the operations on an image are simple and very repetitive – best implemented in an FPGA. However, an imaging pipeline is often used to identify “blobs” or “Regions of Interest” in an object being inspected. These blobs can be of varying sizes, and subsequent processing tends to be more complex. The algorithms used are often adaptive, depending on what the blob turns out to be... so a DSP-based approach may be better for the back end of the imaging pipeline.

Summary

FPGA and DSP represent two very different approaches to signal processing – each good at different things. There are many high sampling rate applications that an FPGA does easily, while the DSP could not. Equally, there are many complex software problems that the FPGA cannot address.

As a result, the ideal system is often to split the work between FPGAs and DSPs. This is easily accomplished using the HERON system, and in many cases can be done simply using I/O and processor modules without any dedicated FPGA resource.