Two Virtex-II FPGAs Deliver Fastest, Cheapest, Best High-Performance Image Processing System

Dillon Engineering not only exceeded their client's performance specifications, but they also delivered the solution under budget by an order of magnitude.

by Tom Dillon
President, Dillon Engineering Inc.,
tdillon@dilloneng.com

In the early part of 2001, one of our clients asked us to consult on a sophisticated digital image processing system requiring a combination of high resolution and high frame rates. A key element in this application was to perform FFTs (Fast Fourier Transforms) on a huge amount of data at very high speed.

The client's existing solution for the FFT portion of the system was based on more than 40 high-end, fourth-generation (G4) IBM® PowerPC™ CPUs. This system had been intended to achieve at least 30 fps (frames per second), but in fact, it never delivered more than 15 fps.

After evaluating a number of design alternatives, we determined an implementation based on two Xilinx® Virtex™-II FPGAs would satisfy our client's immediate requirement for 120 fps – and would be scalable to 240 fps in the future. In addition to being one of the fastest – if not the fastest – FFT processor in the world, our solution would also cost only a fraction of our client's existing implementation.

In this article, we will first describe the scale of the challenge, then look at the alternatives we considered, and finally, reveal the solution we implemented, using our proprietary ParaCore Architect™ core generation utility, Mentor Graphic's LeonardoSpectrum™ 2001 synthesis technology, and two Xilinx Virtex-II XC2V6000 Platform FPGAs.

The Challenge

Unfortunately, our nondisclosure agreement with our client bars us from revealing their identity or even the nature of their extremely competitive business. Thus, we cannot describe the actual application in great detail or show images of the final hardware implementation. We must, therefore, leave any possible applications to your imagination.

What we can say is the most challenging part of this image processing system was to accept high-resolution digital images at 120 fps and generate a corresponding two-dimensional (2D) FFT for each frame. By generating these 2D FFTs, the images were transformed into frequency domain representations that can be used to analyze various features of the object being viewed.

The processing requirements associated with such a system are tremendous. The combination of 16-bit pixel data, a resolution of 2K x 2K pixels (2,048 x 2,048 = four megapixels), and a required frame rate of 120 fps results in 480 megasamples of 16-bit data per second. This huge amount of raw data must undergo extensive processing to convert it into the final 2D FFT, as depicted in Figure 1.

The system comprises two FFT processors. The first processor is used to generate a one-dimensional (1D) FFT of the frame. The second processor then converts this 1D FFT into a 2D FFT. The first FFT processor works on a row-by-row basis. The FFT for each row also contains 2,048 pixels, but in this case each pixel now represents a component in the frequency domain. Each of the 2,048 rows forming the frame requires an associated FFT to be generated. The result is a 1D FFT of the whole image.

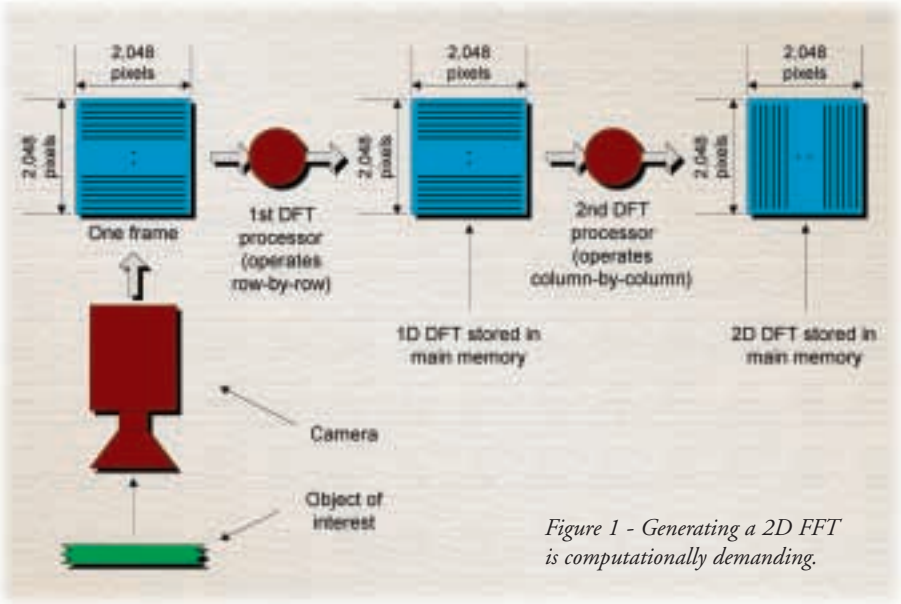

Figure 1 - Generating a 2D FFT is computationally demanding.

Once the 1D FFT associated with a frame has been produced, it is stored in main memory. If we visualize the first FFT processor as working "horizontally" across all of the pixels forming a row, we can consider the second FFT processor to work "vertically" and process the columns formed by taking the equivalent pixels in each row. That is, the second FFT processor will start working on the first column formed by pixel 0 on row 0, pixel 0 on row 1, pixel 0 on row 2, and so on. Once the second FFT processor has completed this column, it will commence working on the column formed by pixel 1 on row 0, pixel 1 on row 1, pixel 1 on row 2, and so on.

Note that as soon as the 1D FFT has been generated and stored in main memory, the camera is free to take another image. The first FFT processor then starts working on this new frame at the same time as the second FFT processor is converting the 1D FFT into its 2D equivalent.

The smallest computational element used to generate an FFT is called a "butterfly," which consists of a complex multiplication, a complex addition, and a complex subtraction as shown in Figure 2. In turn, the complex multiplication requires four simple multiplications and two simple additions, while the complex addition and complex subtraction each require two simple additions. This means that each butterfly requires a total of four simple multiplications and six simple additions.

Processing a single 2,048 pixel row (or column) requires a total of 11,256 butterflies organized in eleven "ranks," where the outputs from the butterflies forming the first rank are used to drive the butterflies forming the second rank, and so forth. Thus, processing a single row requires 45,025 simple multiplications and 67,536 simple additions. In order to generate the FFT for an entire frame, this process has to be repeated for each of the 2,048 rows (or columns) forming the frame. This means that in order to achieve a frame rate of 120 fps, the processing associated with each row (or column) must be completed within 4 µs (microseconds). This leads to a time budget of 90 ps (picoseconds) per simple multiplication and 60 ps per simple addition.

Alternative Solutions

The first stage of this project involved doing our "homework" into the current state-of-the-art for FFT processing. We had to become intimately familiar with the myriad reduction techniques and computational "tricks" available. We then started to evaluate a range of alternative implementation strategies.

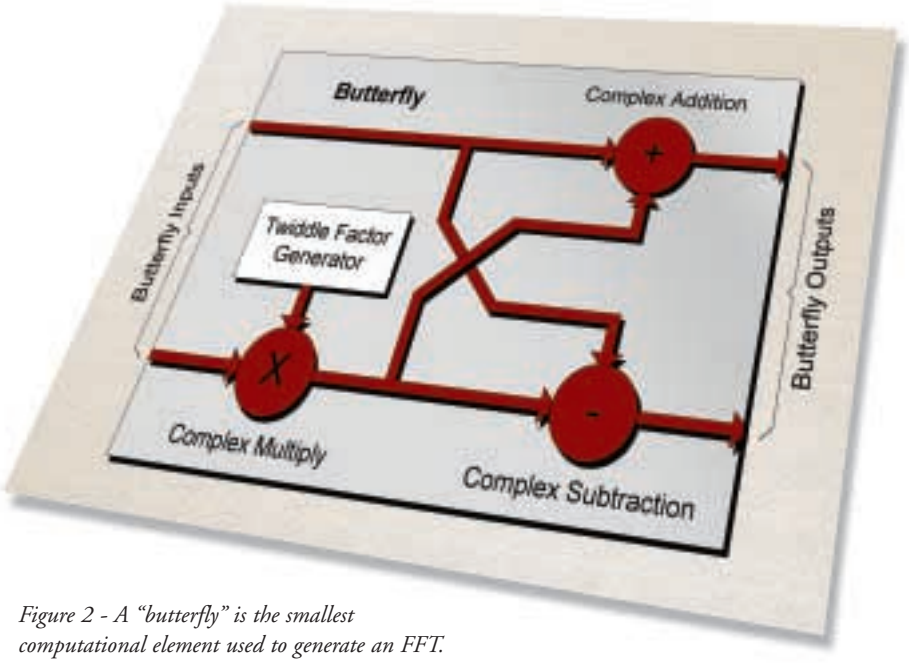

Figure 2 - A "butterfly" is the smallest computational element used to generate an FFT.

The first of these was to extend the customer's existing implementation, but we calculated that it would require more than 100 of the most powerful PowerPCs currently available to achieve 120 fps. In addition to being cost-prohibitive, this solution would not have been easily scaleable up to 240 fps in the future. Having ruled out a PowerPC solution, we moved on to consider off-the-shelf approaches – FFT processing boards, DSP-based solutions, and ASIC-based solutions – versus a custom FPGA-based design.

Off-the-Shelf Approaches

In the case of an off-the-shelf (commercially available) FFT processing board, the most appropriate option required 16 VMEbus boards, each costing $30K.

With regard to a DSP-based solution, high-end alternatives like the C67™ device from Texas Instruments or the Shark™ chip from Analog Devices took 300 µs and 900 µs to perform a 2k-point FFT, respectively (remember that our requirement was for 4 µs). Even using multiple devices, the end result would be slower than a PowerPC-based solution.

In the case of an off-the-shelf ASIC-based approach, the best options to implement a 2k-point FFT were the Pathfinder-2™ device from Catalina Research (31 µs), the PowerFFT™ chip from doubleBW (41 µs), and the DSP24™ component from DSP Architects (60 µs). A solution based on the fastest of these – the Pathfinder-2 device – would have required 32 of these components at $10K each, plus the cost of any supporting ICs.

Custom FPGA-based Solution

And so we came to consider a custom FPGA-based solution. We've been a member of the Xilinx XPERTS program since day one. Thus, it was no surprise to us that the Virtex-II family of Platform FPGAs leads the field for this class of computationally demanding, data-intensive application. In fact, using a Virtex-II XC2V6000, we managed to execute a 2k-point FFT in only 2.8 µs (well within the 4 µs budget required to satisfy our 120 fps design criteria). What's more, we achieved this with a clock frequency of only 125 MHz. The final product was a single VME board with two Virtex-II XC2V6000 devices (one each for the 1D and 2D FFTs) – at only one-twentieth the cost of the best off-the-shelf alternative as illustrated in Table 1.

ParaCore Architect

A critical factor in the design of the 2k-point FFT was our internally developed ParaCore Architect technology. Developed over a number of years, this technology facilitates the design of parameterized cores. The process begins by creating a source file containing a highly parameterized description of the design at an extremely high level of abstraction. The ParaCore Architect utility takes this description, combines it with parameter values specified by the user, and then generates an equivalent HDL representation. The resulting HDL is guaranteed suitable for use in any simulation and synthesis environment, so it isn't necessary to run any form of HDL rule-checking program.

The beauty of this type of highly parameterized representation is that it's extremely easy to target it toward a new application or an alternative device. For example, if we decide to change the length of the FFT from 2k to 1k points, setting a single parameter takes care of all of the details, including re-sizing the RAMS used to store

Solution	Comments	Total Cost
Off-the-shelf FFT processing board	Required 16 Cheetah™ boards from Catalina Research	$480,000
DSP-based solution	Required 5 PowerPC G4 VME boards from Mercury Computing (75 CPUs with DSP functionality)	$750,000
Off-the-shelf ASIC-based solution	Required 32 Pathfinder-2 ASICs from Catalina Research (plus supporting memory and logic)	$320,000+
Custom FPGA-based solution	Required 2 Xilinx Virtex-II XC2V6000 FPGAs plus SRAM on a single VME board	$20,000

Table 1 - Cost comparison of alternative implementation options

any internal results, and so forth. Similarly, another parameter can be used to select between fixed- and floating-point math formats (in the latter case, two further parameters are used to specify the size of the exponent and the mantissa).

Of particular interest is the way in which our FFT algorithm is implemented. Consider the 11,256 butterfly operations required to implement a 2k-point FFT. If execution time were not a major factor, it would only be necessary to use a Virtex-II XC2V40 device with its 4x multiplier blocks, create a single butterfly structure (four simple multipliers and six simple adders) and to cycle all of the butterfly operations through this structure. The resulting structure would take 90 µs to generate each 2k-point FFT. Although this is extremely respectable, it falls well short of the 4 µs time budget required by our image processing application.

The easiest way to increase the speed of the algorithm was to increase the number of butterfly structures instantiated in hardware and to perform more of the processing in parallel. In the case of XC2V6000 devices with 6 million system gates, 144 x 18-bit multipliers, and 144 x 18-kilobit RAM blocks, it was possible to perform an entire 2k x 2k-point FFT fast enough to achieve our 120 fps system requirement. And using XC2V10000 components, we will be able to scale the system to achieve 240 fps. To target these different devices only requires setting a single ParaCore Architect parameter to specify the number of butterfly structures we require to be instantiated in hardware.

LeonardoSpectrum 2001

When you're designing an FPGA with 6 million system gates, it's very difficult to achieve optimum performance without (a) spending decades working on the problem by hand or (b) having great tools. Our client would not have accepted the first approach, so we decided to go with option (b).

Thus, another vital factor in our design was to use the LeonardoSpectrum 2001 synthesis tool from Mentor Graphics. We've been using Mentor's tools for eight years and have always been satisfied with their power and functionality, but we still took the time to consider other options that were available. In the case of this project, there were a number of factors that made Mentor the vendor – and LeonardoSpectrum the tool – of choice:

- The excellent relationship between Mentor and Xilinx means that LeonardoSpectrum synthesis technology is at the cutting edge of the Xilinx product offerings. In this case, it was ready for Virtex-II devices pre-silicon, which allowed us to start designing well ahead of the physical components becoming available.
- We didn't have to create any special constructs in our HDL, because the LeonardoSpectrum tool inherently understands functional elements like multipliers, adders, block RAM, and the like. The tool automatically instantiates the functional elements' equivalent hardware counterparts in Virtex-II devices. (Other tools we evaluated would have required us to go to extremes to code our HDL in strange and amazing ways to achieve the same effect).
- For initial design evaluations, the LeonardoSpectrum interface has a "single pushbutton" mode that we used to good effect. Later in the design process, we moved to using the more advanced modes that allow every synthesis attribute to be controlled step-by-step.
- Furthermore, the LeonardoSpectrum interface can be used to drive the Xilinx Foundation™ place-and-route software. Having a single user interface to drive both tools made our lives much easier.
- Last, but not least, LeonardoSpectrum's TimeCloser™ technology allowed us to "close the loop" by seamlessly importing timing data from Foundation place-and-route back into synthesis to re-synthesize critical paths.

Conclusion

As you can imagine, everyone at Dillon Engineering is extremely happy with the outcome of this project. In addition to a significant design win, we currently have what we believe to be the fastest 2k-point FFT processor in the world. Besides leveraging our core competency in FPGA-based, high-bandwidth, real-time digital signal and image processing, this project also allowed us to take full advantage of our ParaCore Architect utility.

Furthermore, in addition to solving our client's immediate problem by providing them with a 120 fps solution, this solution is an order of magnitude less expensive than their existing implementation (which can only achieve 15 fps on a good day).

Finally, our solution is scalable and can be enhanced to provide 240 fps using Virtex-II XC2V10000 devices – just as soon as they hit the market.