FPGAs: FAST TRACK TO DSP

Revised February 2009

ABSRACT:

Given the prevalence of digital signal processing in a variety of industry segments, several implementation solutions are available depending on cost, performance, and design requirements. These solutions include DSP processor cores, application-specific integrated circuits (ASICs) or field-programmable gate array devices (FPGAs). While general purpose digital signal processors (DSPs) are popular, their performance fall short for many applications. While custom ASIC implementations offer high performance and power efficiency, their fabrication costs can be prohibitive for medium-to-low volume products. Field Programmable Gate Arrays (FPGAs), along with their re-programmability, strike a balance between cost and performance that makes them suitable for many DSP applications.

This paper will examine the advantages and disadvantages of these silicon implementations, why FPGAs are often the most practical alternative, and the importance of a vendor-independent synthesis flow to take advantage of the DSP technologies available in the latest platform FPGAs.

Authors:

Douang Phanthavong

Ehab Mohsen

INTRODUCTION

To many engineers, digital signal processors (DSPs) are proprietary devices used to implement signal processing functions. These conventional pieces of silicon have been used for years to quickly and efficiently digitize analog signals, arithmetically alter digital information, and convert the data back into the analog world.

DSPs are found throughout the world of electronic equipment, from cell phones to flat-screen TVs. The use of dedicated DSP cores is still a widely popular means of achieving digital signal processing requirements. In no more than a few days time, a programmer can take a DSP device and write an algorithm that efficiently performs the processing task required. But as CPUs have incorporated arithmetic coprocessors and extensions that optimize digital signal processing tasks, and as other silicon alternatives have emerged, the necessity of a separate DSP device is no longer viewed as inevitable. In fact, there is increasing evidence that the negatives of discrete DSP devices outweigh their benefits.

ISSUES OF USING STAND-ALONE DSP DEVICES

Among the disadvantages of using a stand-alone, general purpose DSP chip is the issue of performance. Compared to embedded DSP functionality in either a custom ASIC or FPGA, a general-purpose DSP is by far the slowest option. Whereas a stand-alone DSP typically employs serial processing, the parallel capacities inherent to either ASIC or FPGA implementation will always give them both a significant edge over a stand-alone DSP.

The second problem is power. If the end product relies on portability, the power drain of most general purpose DSPs can also become a problem. While DSPs are often quick to program into a system during the development phase, their power requirements make them unattractive for mobile devices.

THE EMERGENCE OF ASIC DSP SOLUTIONS

Historically, when the performance or power disadvantages of stand-alone DSP chips made them unviable for certain applications, most designers turned to custom ASIC implementations. Being hard-wired for a specific application, an ASIC can be equipped to run as fast and as powerfully as allowed by the upper limits of the technology node. It is faster and more power efficient than its general-purpose DSP counterpart.

DSP vendors have been painfully aware of their lost market share due to more powerful ASIC solutions.

Their response has been to create hardware-accelerated engines to increase performance gain. While engines such as dedicated Viterbi decoders and matrix multipliers enhance a DSP chip’s parallelism, the performance gain is less than 100% per additional engine and the DSP device becomes more expensive.

FPGAs AS AN ALTERNATIVE

As well as ASICs stack up in benchmark comparisons, their high fabrication costs make them impractical for medium-to-low volume products. In today's fast-paced environment of semiconductor innovation, long lead times and guarantees of high-volume production are often in short supply. In fact, a change in requirements or a discovery of a bug often results in costly return trips to the fab.

As a result, more designers are turning to programmable logic devices, namely FPGAs, as a way to meet the demands of their DSP applications. The latest platform FPGAs are enhanced for digital-signal-processing, allowing extensive customization without the additional work required for custom ASICs. While even FPGA vendors admit that FPGAs typically do not outperform ASICs in either performance or power, the differences are inconsequential for many applications.

Cost is among the important factors when considering an FPGA as a viable alternative. While ASICs have high manufacturing costs associated with them, high volume may justify the up-front expense. However, the cost comparison is incomplete without considering the additional development, physical design, and verification costs associated with a custom ASIC, as well as post-market expense if the device requires an upgrade for any reason.

A photo mask set at the 65 nm node costs roughly $1.5 million, making leading edge ASIC design a high-stakes endeavor. If the design has hidden glitches that only manifest themselves after fabrication, the re-work and re-spin may be costly and impact time-to-market.

FPGAs WITH EMBEDDED DSP BLOCKS

More designers are turning to FPGAs for digital signal processing due to their flexible architecture, high performance, and low cost.

However, this was not always the case. In fact, older generation FPGAs did not implement DSP operations efficiently. A simple math equation, such as

```
Y = (C ± (A * B) + CIN)
```

used to be synthesized into the generic FPGA fabric and lead to multiple levels of logic with severe area and delay penalties.

Fortunately, today's advanced FPGAs have dedicated DSP blocks, or specialized regions of the device optimized for implementing arithmetic operations as the one described above. DSP architectures for Altera's Stratix-IV family and Xilinx's Virtex-5 family are shown in Figure 1 and Figure 2, respectively.

Figure 1: Altera Stratix-IV Embedded DSP Architecture

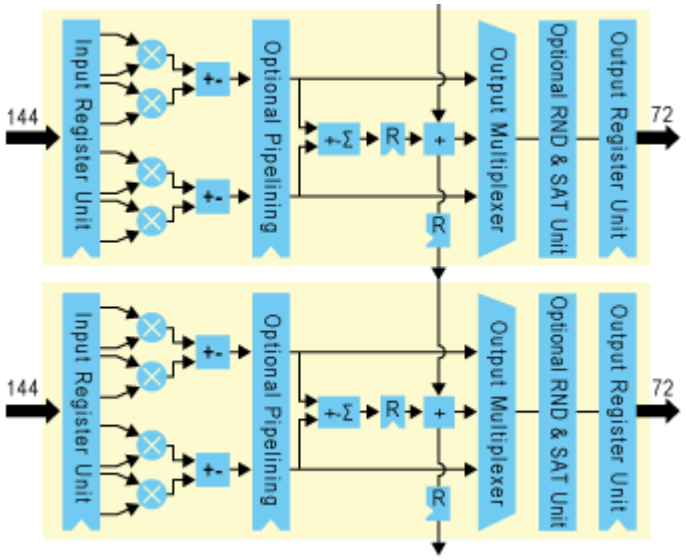
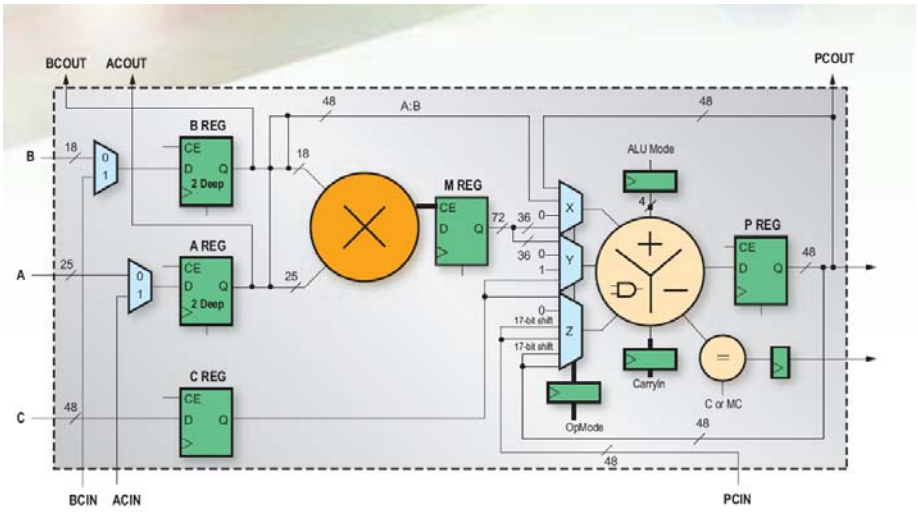


Figure 2: Xilinx Virtex-5 DSP48E Slice



FPGAs now incorporate embedded features that enable multiplication, accumulation, addition/subtraction and summation, all of which are commonly used for DSP functions. With these basic arithmetic functionalities, designing the overall DSP-based application becomes fast, flexible and efficient. At the core of a typical DSP block is a multiplier feeding an adder. DSP blocks have additional features that can be utilized for improved resource utilization and performance:

- Pipelined registers in between the multiplier and adder.
- Built-in registers at the inputs and output of the DSP block.
- Dedicated input or multiplier output (or a combination) can synchronously load the output.

- DSP blocks can be cascaded so that the output of an input stage goes to the next block (this is especially suitable for FIR filter implementation, described in a later section).

However, the versatility just described, raises questions in terms of use-model. How does a designer ensure that certain functions are mapped to specialized DSP resources as desired? What kind of learning curve and changes to the design are involved?

VENDOR INDEPENDENT SYNTHESIS

There are two primary ways for a designer to control usage of an FPGA's specialized DSP resources. The first is to physically instantiate the vendor-specific DSP technology cells within the HDL design. FPGA vendors typically provide push-button utilities to appropriately configure these cells and generate pre-synthesized netlists to be imported with the rest of the design.

The second method is to describe functionality in standard HDL language and let the synthesis tool infer the embedded DSP technology. Figure 3 shows a simple multiply-accumulate function described in VHDL. An advanced synthesis tool will typically infer the correct DSP resource of the device, as well as detailed configuration such as pipeline registers and cascade levels as they are described in the HDL design.

Figure 3: VHDL code describing multiple-accumulate function

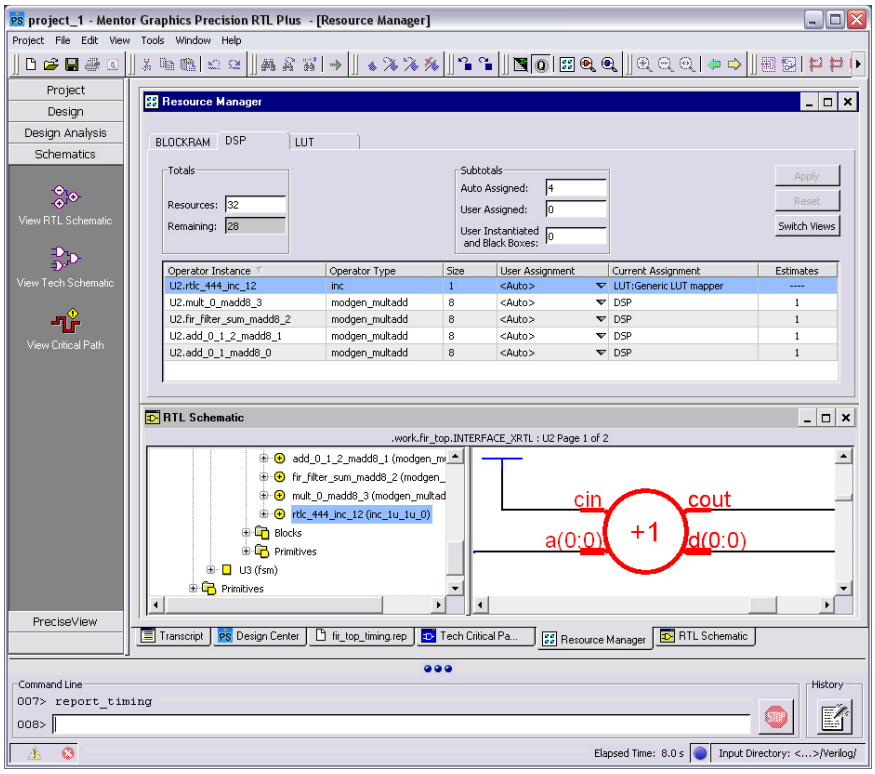
```
entity mac is
port(       clk : in std_logic;
          a: in std_logic_vector (7 downto 0);
          b: in std_logic_vector (7 downto 0);
          result: inout std_logic_vector (16 downto 0));
end mac;

architecture behav_mac of mac is
signal prod_result : std_logic_vector (15 downto 0);
begin
prod_result <= a * b;

process (clk)
begin
    if (clk'event and clk = '1') then
       result <= result + prod_result;
    end if;

end process;
end behav_mac;
```

The inference method just described has two primary benefits, the first of which is that the user has control of resource allocation. In a real-world, DSP intensive design, you may be trying to infer more embedded DSP cells than are available on the device. In such a case you want to carefully choose which DSP is to be implemented in the generic logic fabric—perhaps one of the

smaller functional units or perhaps one not on the critical timing path of the design. Keeping DSP functionality in generic HDL allows you to make this choice.

Precision RTL Plus, Mentor Graphics' leading FPGA synthesis tool, provides a resource management graphical user interface to make allocation decisions easy. As shown in Figure 4, Precision displays all embedded DSP blocks available on the device and displays the design blocks that are assigned to these resources or can potentially be assigned to them. Through this interface, you can control how specialized resources are assigned. These design blocks can be cross-probed back to the original HDL or design schematics within the Precision graphical user-interface for more analysis. Though most tools do a great job in finding the optimal use of the available device resources, as a designer, you may find advantage in assigning certain functional blocks in a specific way. This flexibility is not available when instantiating vendor-dependent technology cells.

Figure 4: Resource Manager in Precision RTL Plus



The other benefit of the inference method is that it allows your design to be FPGA vendor independent. FPGA vendors frequently leap frog each other in terms of capabilities in their devices, meaning the vendor you use for this project may not be the best vendor for your next generation product. By using a vendor independent synthesis tool such as Precision RTL Plus, which supports all devices from leading FPGA vendors, you can re-target your next project to another device with fewer modifications to the design and tool flow. Using the instantiation method described above would require a complete re-generation of vendor-specific DSP technology cells when switching FPGA vendors.

CONCLUSION

The need for companies to differentiate their FPGA-based products via value-added features, combined with the short life cycle of many products, makes FPGAs an attractive platform for many DSP applications.

Dedicated DSP blocks, combined with the advanced inference and synthesis capabilities in Precision RTL Plus, provide the ability to seamlessly implement various high-performance DSP functions for any FPGA vendor.