A Low-Cost, High Performance Data Acquisition & Control Card Using LatticeECP/EC FPGAs and Lattice ispPAC Power Manager

A Lattice Semiconductor White Paper

April 2005

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Introduction

Off-the-shelf data acquisition and control cards with an on-board DSP CPU often perform at only a fraction of their benchmark-based expectations. Increasing the number of channels, or increasing the sampling rate, begins to show signs of DSP CPU overload well before reaching full channel capacity.

Present day market pressures are forcing data acquisition and control card providers to increase the number of channels on a data acquisition card, increase the sampling rate and, at the same time, reduce cost. Further, the actual function of the card is determined either during manufacturing or configured in-system. Sometimes the same card will be expected to perform different functions at different slots on the same system.

The use of DSP CPU on board, while providing the flexibility of hardware reuse across a wide range of applications, also becomes a bottleneck for board level performance.

DSP CPU Efficiency Decreases with Increased Channel Loading

As the number of channels increases, the load on the DSP engine also increases. This is due to the serial processing nature of current DSP CPUs, which are fetch/execute engines. In most applications, most of the DSP CPU's bandwidth is consumed in front-end pre-processing tasks (offset shifting, gain adjustment, preliminary filtering, etc.). Developing this code is also difficult and time consuming because, for efficiency reasons, it is usually coded in assembly language. Even though the transfer of samples from each channel to different memory locations is handled by DMA, CPU performance suffers due to the reduced availability of memory bandwidth. The faster the sample rate, the less time is available for the processor to perform the actual DSP functions such as signal analysis, video processing, compression, etc.

Since the DSP processor has to switch contexts from one channel to the next, the cache thrashing further reduces the available bus bandwidth. In addition to processing the data, the DSP CPU also has to manage data (buffering, data movement, etc), transfer the processed data to the host processor through the backplane, backplane protocol and management, and so on.

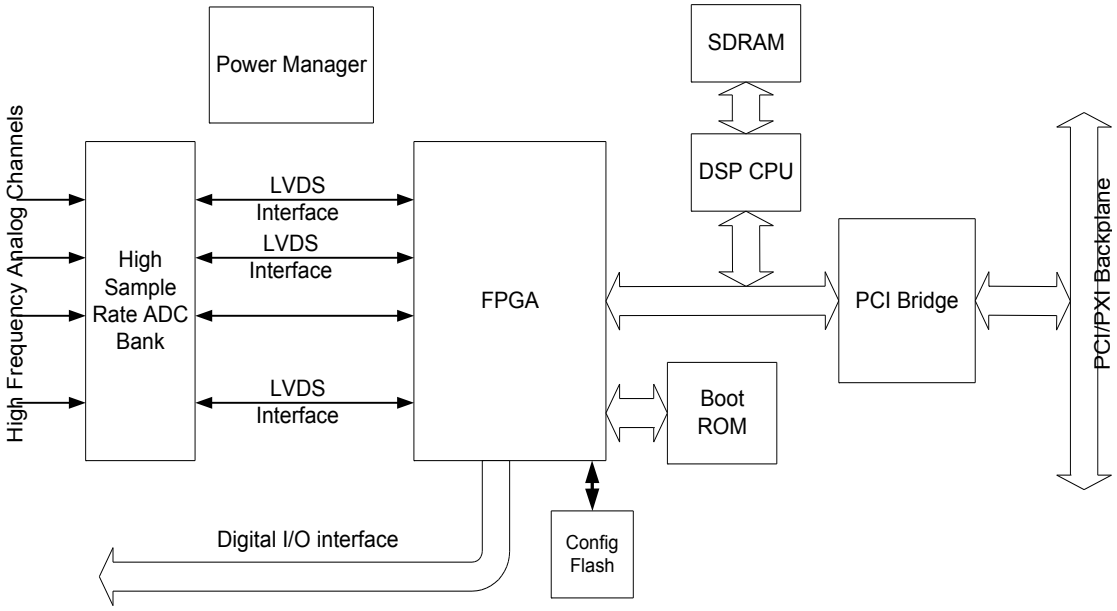
To satisfy the processing demands of an increased number of channels, the DSP CPU performance should be increased exponentially. The increased demand for DSP performance is mitigated either by increasing both the processor operating frequency (resulting in faster bus transfer frequency, use of faster memory, and peripheral interface chips) and/or by using a more powerful processor. Either way, the result is much higher board cost.

The architecture described in this white paper enables an economical increase in the number of channels and the sample rate per channel for a given DSP CPU. This is achieved using low-cost FPGAs, such as the LatticeECP and LatticeEC, as a coprocessor (off-load-engine) to the main DSP CPU, minimizing the pre-processing and non-DSP operational load while remaining flexible to address a wide variety of applications. As these FPGA devices provide ample local storage, it is possible to realize a programmable data flow architecture that further enhances performance: data flow architecture helps the DSP to perform its computation once all the operands are available, instead of performing the computation sequentially as the operands arrive.

The following section describes two approaches to implementing data acquisition. The first method uses an FPGA to interface ADC bank, generate digital I/O interface and manage the data transfer between the ADC and DAC and the SDRAM memory. The second method uses an FPGA with DSP math processing abilities not only to interface the CPU bus to the ADC and DAC, but also to perform pre-processing on the acquired digital samples (DSP Co-processor).

Board Architecture Description

The following block diagram illustrates the architecture of a data acquisition and control card. The card is designed to process 20 Msample from the ADC Bank.



DSP Board With Integrated
4 Channel Data Acquisition With Playback

Block Diagram Description

Starting from the left, multiple high frequency analog signals are quantized by the ADC back and transferred to the FPGA via an LVDS interface. The FPGA transfers the data directly to the SDRAM attached to the DSP CPU through DMA. The DSP CPU processes the data in the memory and transfers analyzed data to the host processor through the PCI backplane. The FPGA generates the control signals for communication with the external digital subsystem through digital I/O at the CPU's command. This board's benchmarked throughput was able to handle 4 channels of 5 MSamples each. At that rate, there was no processing power left for supporting either analog control (driving DAC using Digital I/O), or

for implementing improved signal analysis. Because the DSP CPU performed all the processing (including the pre-processing functions, communication with the host processor, etc), any increase in the processing requirement could only be mitigated by the use of a next generation data acquisition board with a more expensive DSP CPU. Alternatively, the additional processing could be performed by reducing the number of input channels.

The proposed approach explores the use of FPGAs with embedded hardware accelerators for performing pre-processing DSP functions, resulting in a significant reduction of the processing burden on the DSP CPU. The resulting design not only offers higher performance, but also frees the DSP CPU to perform more sophisticated DSP processing and control functions.

FPGA with Hardware DSP Math Processing Blocks

Digital signal processing is performed through mathematical operations such as multiplication, addition and subtraction. It is both expensive and inefficient to implement mathematical functions in an FPGA using general purpose FPGA fabric. However, FPGAs with hardware blocks with multiplication, addition and subtraction ability are capable of offloading the DSP CPUs economically and efficiently. These multiple blocks enable the FPGA to process multiple streams of data simultaneously, substantially improving the performance of the system.

This section briefly describes one such hardware math processing block implementation, called sysDSP, on a Lattice ECP-DSP FPGA. The high performance sysDSP block can be configured to address a wide variety of DSP operations.

There are 4 to 10 sysDSP blocks per device, enabling parallel operation of DSP processing across multiple channels. Each of the sysDSP blocks can be configured as one of the following modes using software:

36x36 Mode

- One 36x36 multiplier

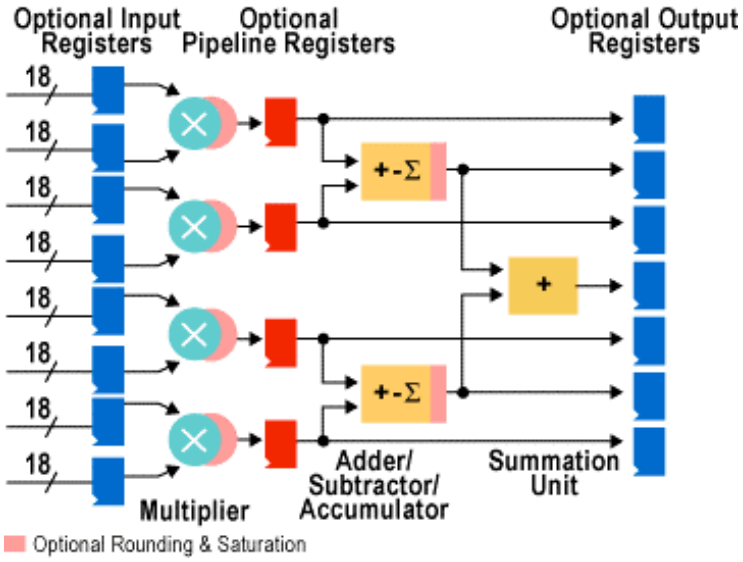
18x18 Mode

- Four Multipliers
- Two 52-bit MACs
- Two sums of two 18x18 multipliers each
- One sum of four 18x18 multipliers

9x9 Mode

- Eight Multipliers
- Two 34-bit MACs
- Four sums of two 9x9 multipliers each
- Two sums of four 9x9 multipliers each

The flexibility of the sysDSP block can be applied across a wide variety of DSP operations. The following block diagram shows the configuration of the sysDSP block in the 18X18 mode:



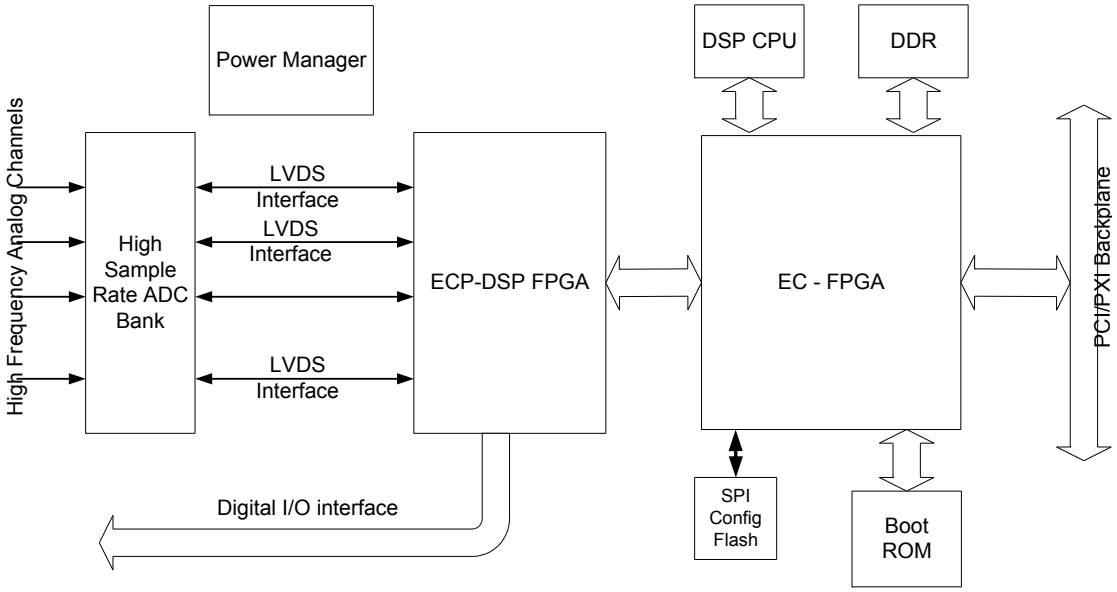
These blocks are able to perform all functions at a speed of 250 MHz, resulting in overall processing capacity of 10,000 MMACs (Mega Multiply Accumulate operations). As the data for processing is presented to these sysDSP blocks completely by hardware (as opposed to a microprocessor fetching the data from its memory), the performance of these blocks is close to benchmark-based expectations. The pipeline registers enable DSP processing operations at peak

operating speeds. The sysDSP block's ability to perform MAC, sum, add or subtract within the block, without the use of an external FPGA fabric, makes it immune to fabric routing delays.

The next step is to use this FPGA with the sysDSP building block in a circuit as a pre-processing DSP engine while maintaining the flexibility of the earlier architecture. The following section describes the architecture that uses 2 FPGAs. The FPGA with the sysDSP block, the LatticeECP-DSP FPGA (referred to as ECP-DSP), and the FPGA without the sysDSP block, the LatticeEC FPGA (referred to as EC), are used to improve overall data handling efficiency.

Improving Performance Through Repartitioning Using ECP-DSP and EC FPGAs

The following block diagram uses the same DSP CPU and addresses all the increased performance requirements, while doubling the channel capacity, at a much lower cost compared to the board that used the newer DSP CPU.



Proposed Architecture: 40 MSample DSP Board With Integrated 4/8 Channel Data Acquisition With Playback & Control

Starting from the left, the ADC Bank can now sample up to 8 high frequency analog signals and communicate with the ECP-DSP FPGA using the LVDS signaling interface. The ECP-DSP FPGA performed all the signal pre-processing functions, real-time, on all channels simultaneously. The cleaned and ready to process sample data was written into the DDR memory through the FPGA.

The CPU bus interface (64-bit), implemented in the EC FPGA, enabled the transfer of data and instructions between high performance DDR memory and CPU cache in burst mode. The EC FPGA also provided internal memory to buffer the data for processing, reducing the non-sequential access penalty of the DDR memory. While the CPU is processing data in its cache, the intelligent bus arbiter and switch logic implemented in the EC FPGA enabled the data transfer to and from peripheral devices (EC FPGA, PCI Backplane, Digital I/O interface).

This architecture improves DSP CPU efficiency by

- Reducing the pre-processing load on the DSP CPU
- Reducing time critical channel and context switching interruptions
- Reducing the non-DSP operation load
- Increasing the memory bandwidth by using faster DDR memory
- Reducing non-sequential access to DDR during processing

The increased availability of the DSP CPU is used to increase the number of input and output channels.

Functions Performed by the ECP-DSP FPGA

- High speed ADC interface
- Per channel real time pre-processing, implementing functions such as offset shifting, gain adjustment, FIR filter, etc.
- Digital I/O interface that can be configured as the application demands
- The digital interface can also drive DAC with buffering
- High speed communication interface with the EC FPGA

Functions Performed by the EC FPGA

- 64-bit DSP CPU interface
- DDR interface

- Local memory storage for partially processed data
- cPCI/PXI interface
- Logic to configure the ECP-DSP device using the configuration stored on the host CPU
- High speed communication interface with the ECP-DSP FPGA
- Logic to transfer the boot code to DDR memory for the CPU to execute
- Update boot code Flash with the updated code received from the host CPU
- Intelligent bus arbitration logic functions required for managing data transfer among LatticeECP, Memory, cPCI back plane, etc.

Configuring the Board for Different Applications

This board can be configured either in-system through the backplane interface or during manufacturing.

Configuration from the Backplane

The code in the boot ROM for the DSP CPU enables communication with the host processor, and the SPI configuration flash attached to the LatticeEC FPGA is loaded with configuration data to perform all the functions described above.

After the card is plugged into the slot, the FPGA configures, transfers the ROM code to DDR memory and signals Power Manager to release CPU Reset. The DSP CPU communicates with the host processor and initiates the transfer of the DSP algorithm into DDR memory, as well as the pre-processing configuration to be loaded directly into the DSP FPGA by the sysConfig port controlled by the FPGA. The DSP CPU and the DSP FPGA are then ready to process signals as required by that slot.

Configuration During Manufacturing

The DSP algorithm is stored in the boot ROM and the FPGA configurations are stored in their respective SPI Config Flash memory.

After the card is plugged into the slot, the FPGA device configures itself from the SPI Flash (during this time the Power Manager holds the CPU in Reset condition) and transfers the contents of boot ROM into the DDR memory. Simultaneously, the LatticeECP-DSP FPGA configures itself from its own SPI

Config Flash device. After both FPGAs are configured, the Power Manager is signaled to release the CPU Reset.

Advantages of the Proposed Architecture

The performance of the enhanced data acquisition board with the proposed architecture more than doubled, compared to the old implementation, while using the same DSP CPU. The increase in performance was achieved primarily by eliminating time-consuming, assembly-language-coded, difficult-to-maintain, pre-processing functions from the old design. Additional performance improvement was achieved by switching to faster and less expensive DDR memory.

The resulting design is significantly less expensive, offers higher performance and is more flexible than boards with a traditional architecture.

###