Platform FPGA design for high-performance DSPs

This tutorial explains how Platform FPGAs can be used to meet the challenges of today's demanding high-performance real-time DSP applications.

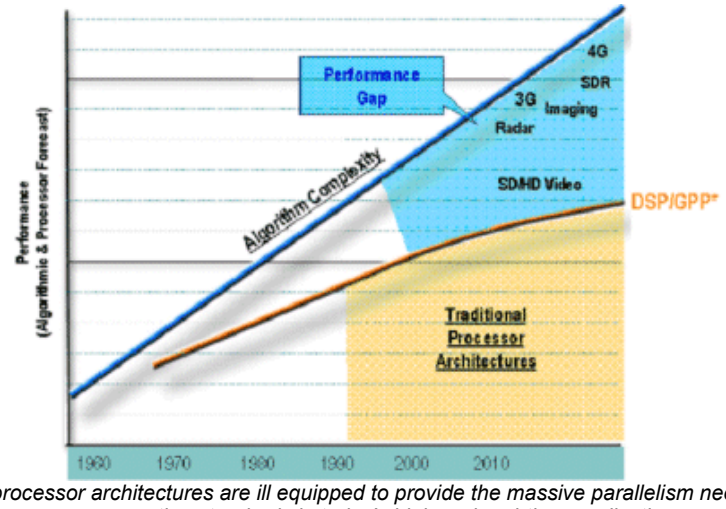
By Anil Telikepalli, Xilinx, and Etienne
Fiset, Lyrtech

As DSP technologies find their way into high-end, high-complexity commercial and consumer applications, developers are discovering the limitations of traditional Digital Signal Processors (DSPs) to satisfy their performance requirements. The processing complexity in many of these applications requires massive parallelism to perform complex DSP functions in real time. The cost of using large DSP farms to implement such parallelism is often far too prohibitive in terms of price, power and form factor, leaving the system designer no choice but to seek alternative solutions.

This paper discusses the unique capabilities platform FPGA-based designs offer when used to help meet the challenges of today's demanding high-performance real-time DSP applications, both as stand-alone solutions and as complementary solutions for traditional DSPs. A detailed investigation of the use of model-based design methodologies reveals the ease with which platform FPGAs can be employed to accommodate these highly complex DSP implementations. Finally, an example in the wireless arena – Multiple Input, Multiple Output (MIMO) technology – is used to illustrate the concepts.

Changing standards
For the past two decades, the networking, imaging and defense industries have searched for new and more efficient ways to process and deliver content-rich data across disparate networks of commercial and consumer-level systems and devices. In the context of DSP, the wired and wireless communications markets are experiencing this "standards revolution" with WiFi, WiMAX, 3G, 4G, and WCDMA system standards, application standards such as OFDM and MIMO, as well as technology standards such as Software Defined Radio (SDR). Additionally, a continuous stream of changes is also erupting from the audio, video, and broadcast markets as well as the defense electronics industry.

Processing complexity
Every standard represents a substantial opportunity and challenge for companies in the DSP design community. However, the DSP processing requirements of many of these standards has produced a performance gap that traditional DSPs and general-purpose processors (GPPs) cannot cross on their own (*Fig 1*).



1. Traditional processor architectures are ill equipped to provide the massive parallelism needed to support new-generation standards in today's high-end real-time applications.
(Click above image to see a larger, more detailed version)

These new applications have driven processing complexity and computational workload from tens of mega-samples per second (MSPS) to 100s of MSPS. The digital convergence phenomenon (video, voice, data) and adaptation to user's individual needs has created the simultaneous demand for more sophisticated real-time processing and higher bandwidth to support on-demand content-rich multimedia data.

System Integration
Treating DSP system design as solely a signal processing problem addresses only part of the issue. For many applications, DSP function is not a separate entity, but rather one of several system-on-a-chip (SoC) functions, raising the need to address a host of system integration issues as well. These issues include:

- Connectivity using standards such as RapidIO, PCIe ATCA, 1G/10G Ethernet, and DDR/DDR2 memories.
- System control issues such as statistics monitoring, QoS/CoS, and traffic priorities.
- The need to meet reliability requirements by keeping power consumption within the standards set by Networking Equipment Building Systems (NEBS) and European Technical Standards Institute (ETSI), and signal integrity requirements set by bit error rates (BERs).

How DSP solutions compare
Amid the confusion of challenges, issues and concerns, it's easy to understand why making the correct selection of the

technologies.

		Programmable DSP e.g., TI, ADI	DSP ASIC e.g., NEC	DSP ASSP e.g., Broadcom	MPU DSP e.g., Renesas	FPGA-based DSP e.g., Xilinx
Performance		<2 MSPS	>100 MSPS	Often Optimized	<1 MSPS	>100 MSPS
Design Flexibility		S/W only	H/W once & S/W many times	None or Little	S/W only	H/W many times S/W many times
Design Effort		C-based Flow	ASIC design Flow	Drop in Solution	C-based Flow	Simulink or HDL based
Power	per Device	OK for portable	OK for portable	OK for portable	Few Watts	Lots of Watts
	per Channel	Not scalable	Scalable	A little scalable	Not scalable	Scalable
Cost	per Device	Start at <$2	>250KU	Often optimized	Start at $5+	<250KU
	per Channel	Not scalable	Scalable	A little scalable	Not scalable	Scalable

Best OK Worst

2. FPGAs offer the best combination of performance, design flexibility and scalability; combining programmable DSPs and FPGAs is the best all-around solution.
(Click above image to see a larger, more detailed version)

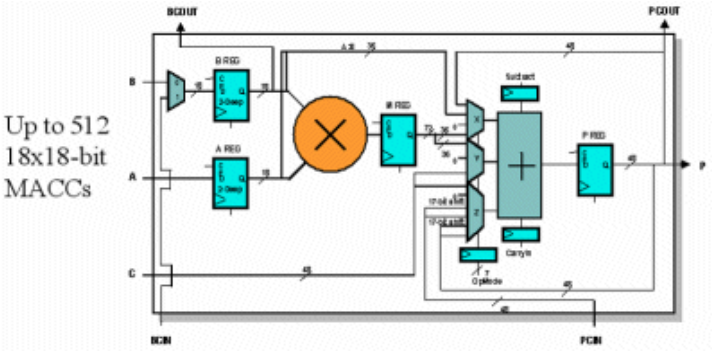
As performance and complexity requirements increase, the massive parallelism available in FPGA-based DSP designs becomes the critical selection criterion. Add to this the FPGA's design flexibility to help adapt to changing standards and its ability to scale power and cost based on the particular algorithm being implemented, and it is easy to see why FPGAs have gained such prominence as a leading DSP technology.

Making the right selection is dictated in large part by the particular application. In some cases, the best choice is a combination of programmable DSP and FPGA, where the FPGA is used either as a pre-processor to the DSP, or as a co-processor.

DSP design with platform FPGAs
Over the past two decades FPGAs have evolved from a collection of gates for programmable logic to platform FPGAs integrating system-level capabilities:

- Clock management.
- Memories.
- Parallel and serial I/Os.
- Ethernet MACs.
- Microcontroller(s) and microprocessor(s).

Platform FPGAs provide numerous multiply-accumulate (MAC) functions – the fundamental DSP building block (*Fig 3*). In contrast, a traditional Digital Signal Processor provides only one to four such MACs.



3. The Xilinx Virtex-4 DSP block offers up to 512 18x18 MACs.
(Click above image to see a larger, more detailed version)

With the Virtex-4 family, Xilinx introduced a new concept called the multi-platform FPGA. Built upon an architectural innovation called the ASMBL architecture, the multi-platform FPGA enables tuning the ratio of key features on the FPGA to match the requirements of an application domain; e.g., the SX family of Virtex-4 FPGAs is optimized to deliver more MACs per dollar and lower power for DSP applications.

The Virtex-4 family contains a new element called the DSP48 slice that integrates a high-performance arithmetic and accumulation unit along with a multiplier. The DSP48 slice comprises four main sections:

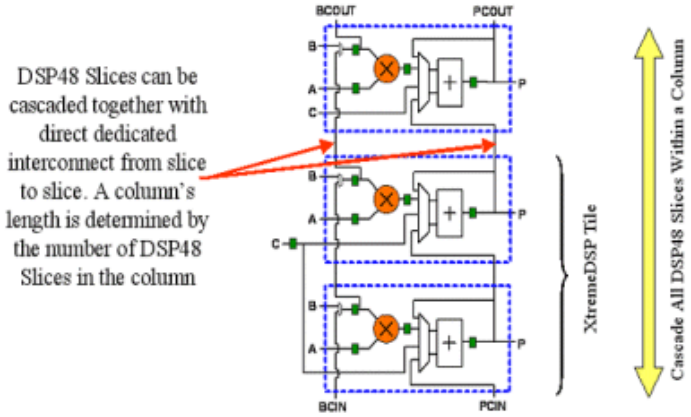
- I/O registers.
- 18 x 18 signed multiplier.
- Three input adder/subtractor blocks.
- OPMODE multiplexers.

The I/O registers deliver the maximum clock performance at no area cost, ensuring the highest possible sample rates. The OPMODE multiplexers are key to the functionality of the DSP48 structure to support over 40 unique operations. For example, to create a simple FIR filter, the X and Y MUXes are set to multiply, and the feedback path is selected from the registered output P as the Z MUX input to the arithmetic unit.

The DSP48 slice maintains 500 MHz operation regardless of the particular operation selected. Table 1-7 on pages 32-33 of the XtremeDSP User Guide provides a list of the various modes available including adder, subtracter, counter, comparator, multiplier, accumulator, etc. Changing operation from one OPMODE to another requires a single clock cycle.

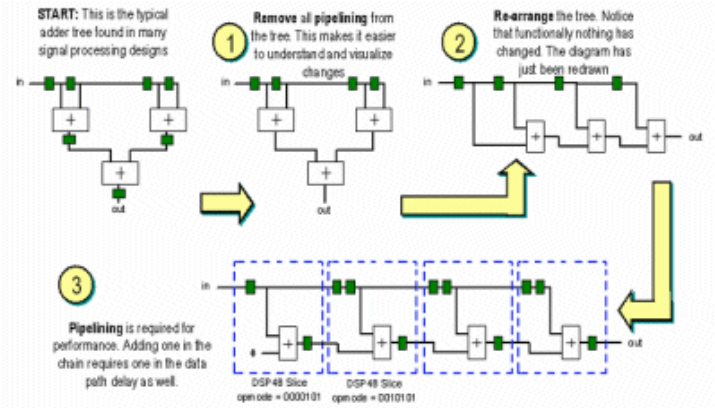
DSP48 slices are cascadable

DSP48 slices are organized into tiles – called XtremeDSP tiles – with sets of two slices per tile. The slices in each tile are arranged in a columnar fashion, both slices sharing the C input within that tile (*Fig 4*).



4. Cascading multiple DSP48 slices within the same column.
(Click above image to see a larger, more detailed version)

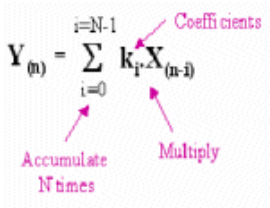
This column structure includes dedicated cascade logic between each block on both the input and the output. This dedicated routing enables the efficient use of chaining (*Fig 5*) to create filters and other functions, rather than a typical Direct Form Type I adder tree configuration. The chain configuration makes more efficient use of the XtremeDSP tile configuration and also enables the output of one adder to be further multiplied before being sent to the next adder.

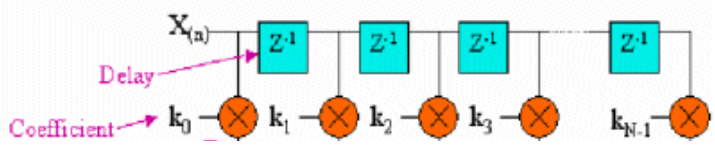


5. Chaining is used in place of the adder tree configuration to take advantage of the greater resource efficiency when building filters and other complex functions.
(Click above image to see a larger, more detailed version)

Building a FIR filter

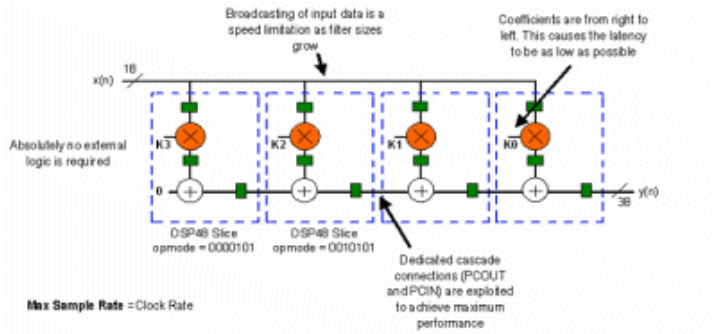
Having reviewed the functionality of the DSP48 engine, let us consider how to build some DSP functions with it. One of the most common DSP functions is the Finite Impulse Response filter, or FIR filter. The general FIR filter equation is a summation of products (also known as an inner product), which is defined in the equation and depicted by the block diagram in *Fig 6*. In this equation, a set of N coefficients is multiplied by N respective data samples, and the results are summed together to form an individual result. The values of the coefficients determine the characteristics of the filter (e.g., low-pass filter, band-pass filter, high-pass filter).





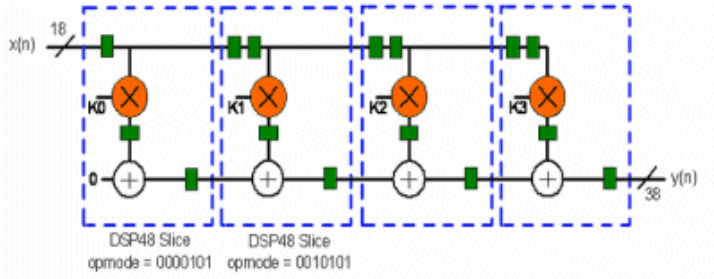
combination of the aforementioned techniques using three to four multipliers to build larger FIR filters.

Fig 7 shows a Transposed FIR filter (or Direct Form Type II) implemented in a DSP48 slice that operates at 400 MSPS, while *Fig 8* shows a Systolic FIR filter (or Direct Form Type I with pipelining) also implemented in a DSP48 slice that operates at 450 MSPS.



7. The Transposed FIR Filter structure is easily mapped to the DSP48 slice without additional external logic, achieving 400 MSPS.
(Click above image to see a larger, more detailed version)

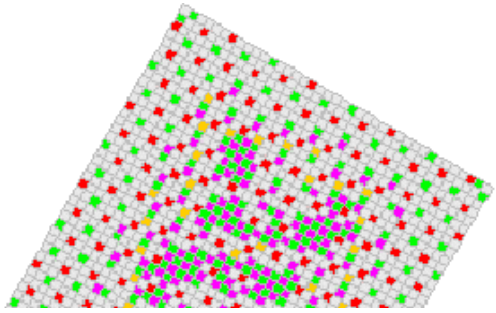
To put this into perspective, using a dedicated DSP operating at a 1 GHz clock frequency with four coefficients would produce a sampling rate one-fourth the operating frequency, or a maximum of only 250 MSPS.

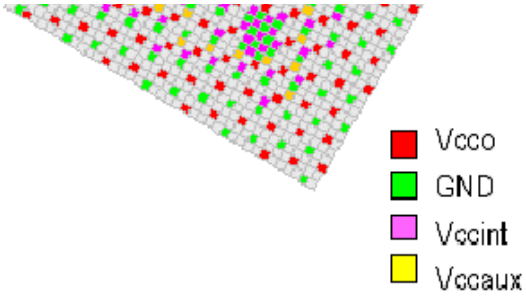


8. The Systolic FIR Filter offers the highest performance in a DSP48 slice – 450 MSPS.
(Click above image to see a larger, more detailed version)

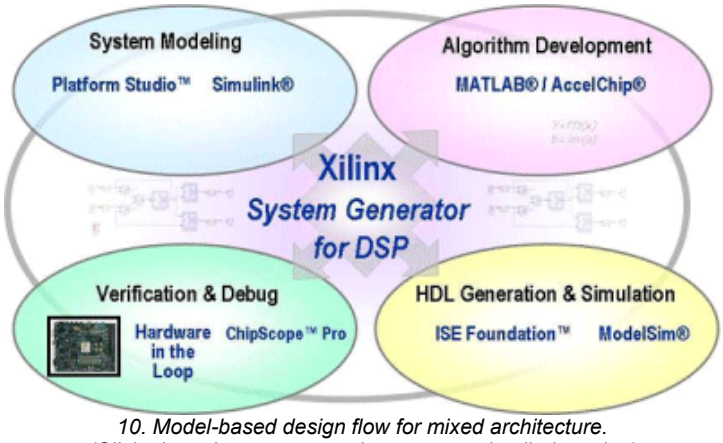
Simplifying system integration
As mentioned previously, DSP system designers need to take care of system design aspects in addition to DSP functionality and performance challenges. These include connectivity, system control, power, and signal integrity.

- Connectivity – Platform FPGAs must provide universal connectivity through parallel and serial system interfaces such as PCI, DDR2 SDRAM, RapidIO, XAUI, and PCIe.
- System Control – Essential to monitoring and decision making, soft-processor cores and industry-standard processors such as PowerPC and ARM are embedded in today's FPGAs.
- Power and Signal Integrity – 90nm Virtex-4 FPGAs use triple-oxide technology and power optimized hard IP blocks to reduce static and dynamic power by 50% vs. 130nm FPGAs. They also use new flip-chip packaging technologies with pinouts that provide PWR and GND return paths adjacent to user pins to mitigate inductive crosstalk (*Fig 9*).





processor. Bridging the gap between FPGA and DSP design flows requires a high-level, model-based design approach (*Fig 10*) that enables the designer to quickly articulate, explore, iterate and validate the entire system design – including the FPGA – from within the existing DSP design flow. Such a design flow already exists using MATLAB' and Simulink' (an interactive block-diagram simulation tool based on MATLAB).

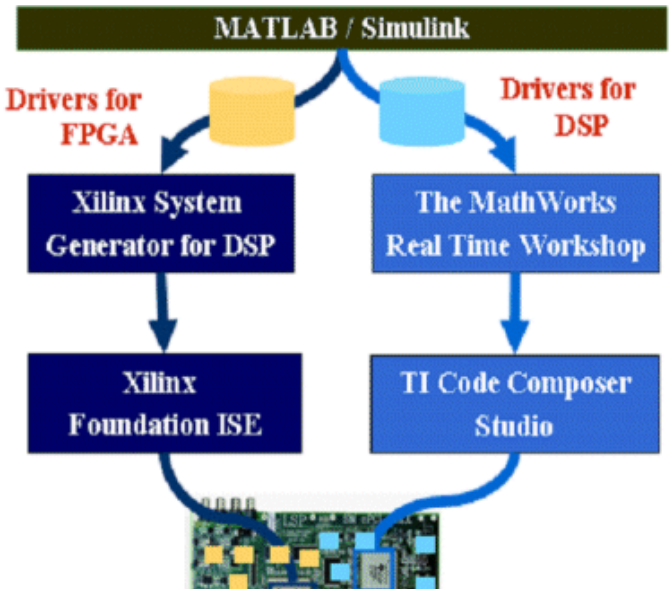


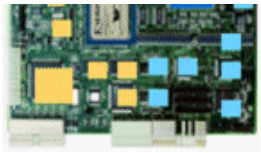
10. Model-based design flow for mixed architecture.
(Click above image to see a larger, more detailed version)

MATLAB is an excellent tool for algorithm development and data analysis. Simulink lets you graphically design the architecture and simulate the timing and behavior of the whole system. It augments MATLAB, allowing you to model the digital, analog and event driven components together in one simulation.

Using Simulink you can quickly build up models from libraries of pre-built blocks. Xilinx offers block libraries within their *System Generator for DSP* tool (containing bit-true and cycle-accurate models of their FPGAs' particular math, logic, and DSP functions) that plug into Simulink.

The AccelDSP tool allows DSP algorithm developers to create HDL designs from MATLAB and export them into System Generator.





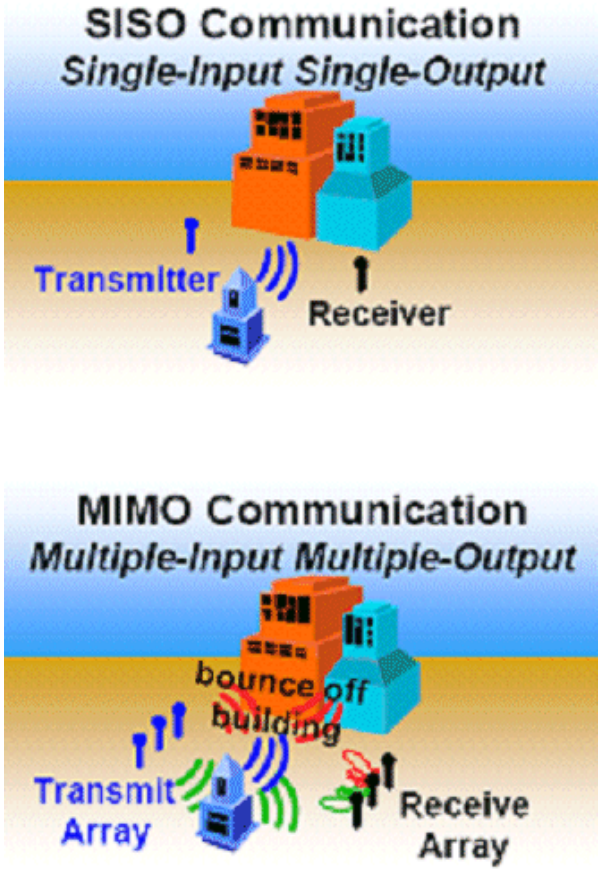
FPGA and DSP drivers in the development platforms.

The changing world of DSP
Clearly, the DSP world has benefited from dramatic improvements on all fronts. Whether in terms of performance, complexity, system integration or design methodology, the challenges imposed by numerous DSP application domains have driven the DSP industry to keep pace with innovative solutions that enable the system designer to overcome those challenges. As such, the platform FPGA has become an important source of DSP functionality due to its inherent parallel processing capabilities, design flexibility and system integration benefits.

A good example of the type of application in which FPGA-based DSPs will play a vital role is the wireless communications domain. A new technology is emerging from within this community that promises enormous performance and reliability improvements to the applications that embrace it. This new technology is called MIMO, or Multiple Input, Multiple Output antenna configurations.

Understanding MIMO
In the world of wireless communications the term "multipath wave propagation" has, in the past, evoked a great deal of frustration and consternation. It refers to the phenomenon caused by obstructions such as hills, canyons, buildings, and utility wires that scatters portions of the wireless signal, thereby creating problems such as fading, cut-out and intermittent reception.

Recently, however, a number of new wireless standards are emerging that hope to turn this phenomenon into a benefit. Whereas conventional wireless systems that employ Single Input, Single Output (SISO) antenna configurations, (i.e., one transmit antenna and one receive antenna) are susceptible to the detrimental affects of multipath wave propagation, MIMO uses two or more antennas to transmit and receive multiple, simultaneous signals (two or more radio waveforms) in a single frequency channel to exploit multipath propagation, thereby multiplying the spectral efficiency of the system (*Fig 12*).



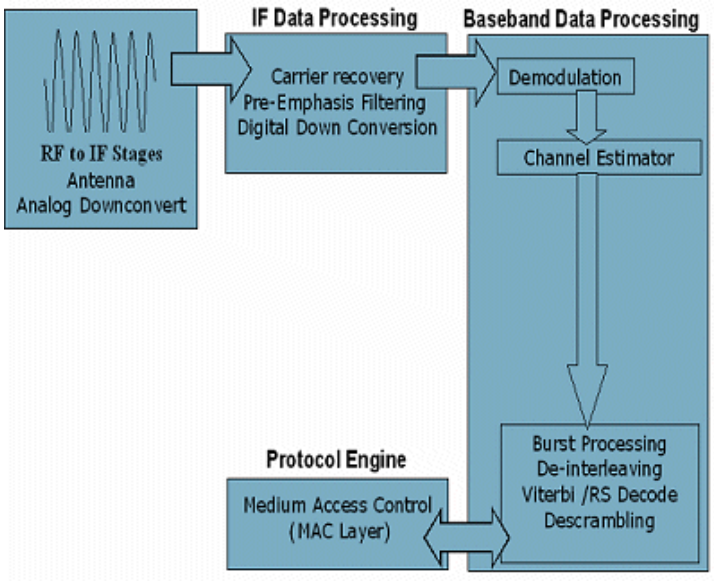
12. MIMO exploits the effects of multi-path wave propagation to improve throughput, range and reliability.

Accomplishing this requires the use of space-time coding techniques. As a result, a wireless network that has adopted

DSP complexity, performance, and changing standards that require platform FPGAs.

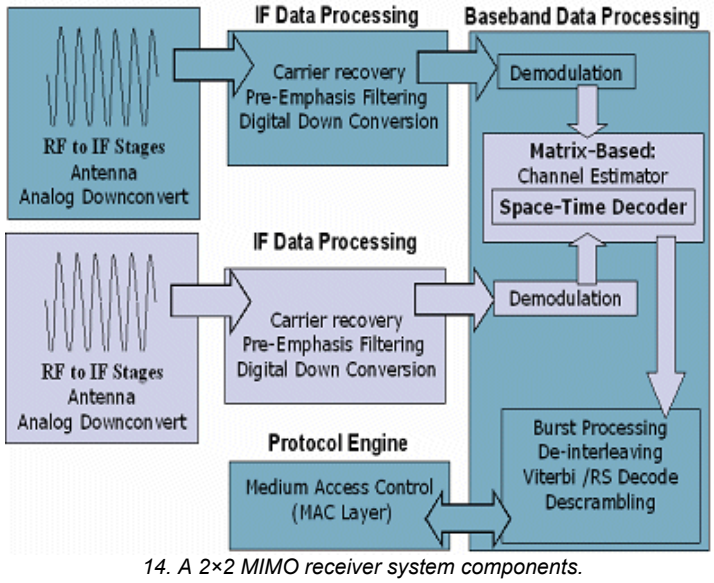
A number of applications have begun work on new standards to implement MIMO technology. Among these are: WiFi / IEEE 802.11n, WiMAX / IEEE 802.16e, 4G Mobile and Super 3G.

Implementing MIMO
To understand the fundamental requirements for implementing a MIMO-based system, it is helpful to first consider the elements of a SISO-based system. Figure 13 below depicts the basic elements of a SISO implementation.



13. SISO receiver system components.

Starting at the receive antenna, the RF signal is down-converted to an Intermediate Frequency (IF). The IF Data Processing circuitry separates the carrier and data signals and applies some filtering before finally performing the digital down conversion and delivering the signal to the Baseband Data Processing (BDP) circuitry. The BDP comprises chip-rate processing functions (such as the demodulator, channel estimator), and symbol-rate processing functions, including a host of other DSP algorithms to perform de-interleaving, decoding and descrambling. Ultimately, the data is passed to the Protocol Engine, which contains the appropriate Media Access Controller (MAC) for the system. When comparing the SISO receiver to the 2x2 MIMO receiver block diagram in *Fig 14*, we see the addition of another completely separate receiver channel with its own RF and IF stages, as well as a separate demodulation stage in the BDP. But the real complexity introduced by the MIMO implementation is the need for a matrix-based channel estimator with a highly sophisticated space-time decoder circuit. The remainder of the circuitry remains exactly the same as in the SISO system.



14. A 2×2 MIMO receiver system components.

It is readily apparent by this comparison that the processing requirements for the MIMO receiver are substantially greater than that of the SISO system. Besides the necessity to process twice the amount of data, the matrix-based channel estimator and space-time decoder introduce a substantial new layer of complexity and processing demands that far exceed the capabilities of traditional DSP processors, necessitating the use of FPGAs.

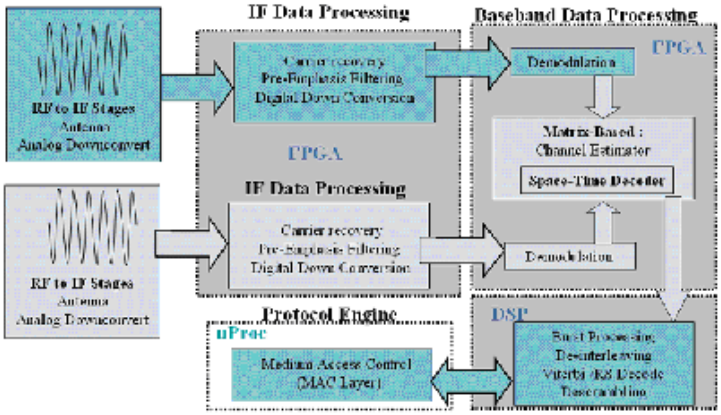
Other system integration challenges for implementing MIMO solutions include:

- MIMO is a recent technology, so it's still in a state of flux and evolving very fast, requiring a programmable framework to adapt to changes.
- To reach higher data rates, evolving MIMO standards must employ more complex algorithms, such as low-density parity check (LDPC) and Turbo codes.
- Besides achieving more processing power, designers must also manage costs, flexibility and time-to-market.
- Finding optimal components for a MIMO application can be challenging and costly.
- No single processor type can do all the processing required.

The last issue regarding the need for multiple DSP processor types working together introduces the issue of properly partitioning the various functions, relegating each to the particular DSP processor best suited to the task. For the high-performance, low-processing complexity IF Data Processing section, an FPGA offers the parallelism needed to efficiently perform these functions.

As previously explained, the channel estimation and space-time decoder functions are matrix-based operations, an inherently parallel algorithmic challenge. However, their complexity suggests the use of floating point C code to reduce development time. Thus, one must decide whether to use a DSP or FPGA processor for this section of the algorithm. A key aspect to bear in mind is that MIMO is a very recent technology, and is still evolving at a very fast rate. As it evolves, the amount of processing power required is likely to increase exponentially.

For example, most MIMO systems are moving from 2×2 (2 TX, 2 RX antennas) to 4×4 implementations (4 TX, 4 RX antennas). Doubling the matrix size from 2×2 to 4×4 requires at least eight times more processing power. Clearly, DSPs will have a hard time providing the required processing power of future MIMO chip-rate processing algorithms. Thus, FPGAs are best suited to implement these algorithmic functions.



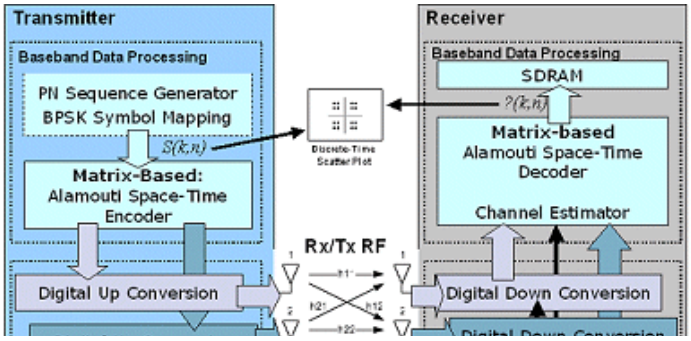
15. Partitioning a MIMO receiver.
(Click above image to see a larger, more detailed version)

On the other hand, a conventional DSP processor can easily accommodate the symbol-rate processing section of the Baseband Data Processor and would enable the designer to take advantage of legacy code for the various algorithms such as Viterbi and Reed-Solomon used in this area. (Note that, if desired, designers might also use pre-designed soft IP for these algorithms and target FPGAs.) Our implementation in *Fig 15* assumes that the user has existing C code targeted at a DSP for these algorithms. Again, using FPGA processors for symbol-rate processing might become mandatory to support future algorithms required by next-generation MIMO standards. For example, Viterbi and Reed Solomon are likely to be replaced with higher-performance coders such as low-density parity check (LDPC) and Turbo codes algorithms, which will require more processing power.

Finally, the MAC layer protocols are most easily implemented using either a discrete general-purpose microprocessor or one integrated into an FPGA. *Fig 15* illustrates this partitioning strategy as well.

As we have shown, one of the major challenges when designing a MIMO system is implementing complex algorithms on FPGA processors in order to support future evolutions of the MIMO standards. The next section describes just such an algorithm – the Alamouti Space-Time Encoder.

2×2 MIMO transceiver system
This section describes the design and implementation of a transmit/receive station based on two transmit and two receive antennas (2×2) with specific focus on the space-time decoder algorithm. Such a system is an excellent starting point to further develop a complete MIMO system such as an 802.11n wireless basestation.

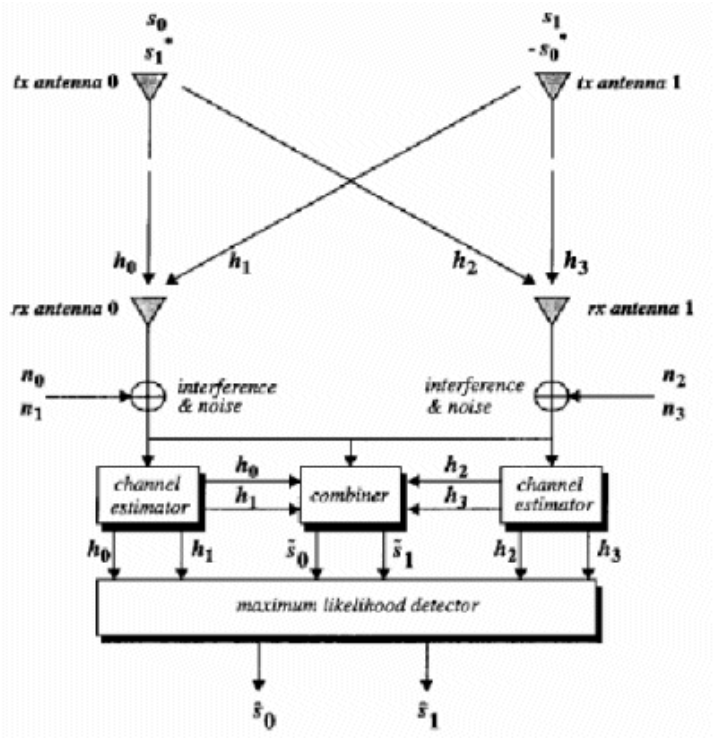


16. 2×2 MIMO transceiver system components.
(Click above image to see a larger, more detailed version)

Fig 16 presents the portions of both the transmitter and receiver systems that have been partitioned to an FPGA. For the purposes of this paper, we will focus in on the implementation of one particularly complex function on the receiver side of the system – the Alamouti space-time decoder. **Alamouti space-time decoder**
Space-time block coding (STBC) is a technique used in wireless communications to transmit multiple copies of a data stream across a number of antennas and to exploit the various received versions of the data to improve the reliability of data-transfer. The fact that transmitted data must traverse a potentially difficult environment with scattering, reflection, and refraction combined with the data corruption caused by thermal noise in the receiver makes some of the received copies of the data 'better' than others. Space-time coding combines all the copies of the received signal in an optimal way to extract as much information from each of them as possible. It was originally designed for a two-transmit antenna system and has the coding matrix:

$$C_2 = \begin{bmatrix} s_1 & s_2 \\ s_2^* & -s_1^* \end{bmatrix}$$

The description of an Alamouti 2×2 transceiver can be as follows (*Fig 17*):



17. Two branch transmit diversity scheme with two receivers.
(Click above image to see a larger, more detailed version)

As we can see, the received signals can be expressed as:

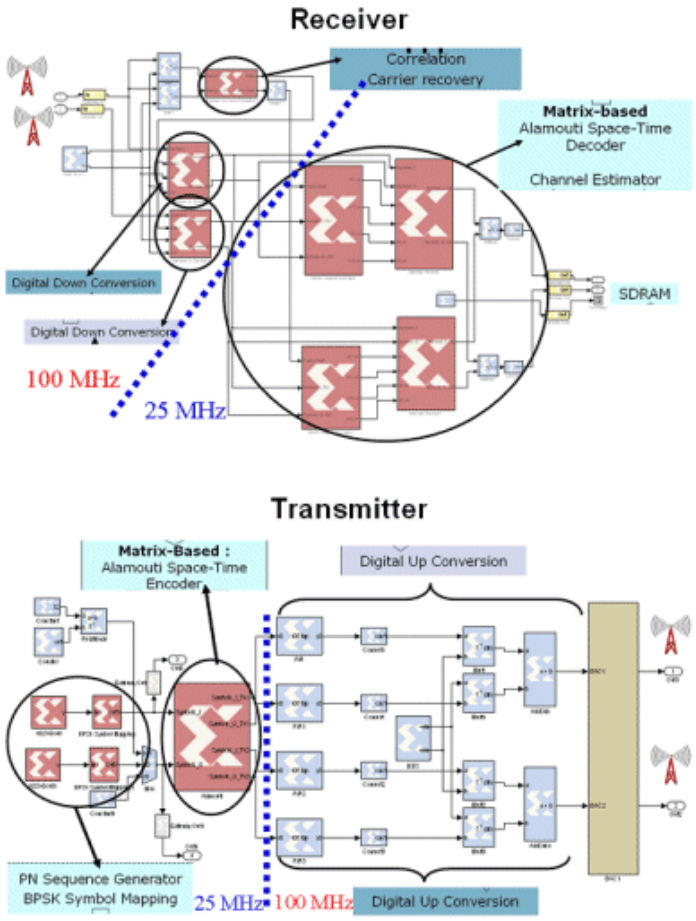
Rx antenna 0: r0= h0s0 + h1s1 + n0
r1= h0s1* - h1s0* + n1
Rx antenna 1: r2= h2s0 + h3s1 + n2
r3= h2s1* - h3s0* + n3

Therefore, the Alamouti decoder algorithm required to recover the transmitted symbols is:

s0= h0*r0 – h1r1* + h2*r2 – h3r3*
s1= h1*r0 + h0r1* + h3*r2 + h2r3*

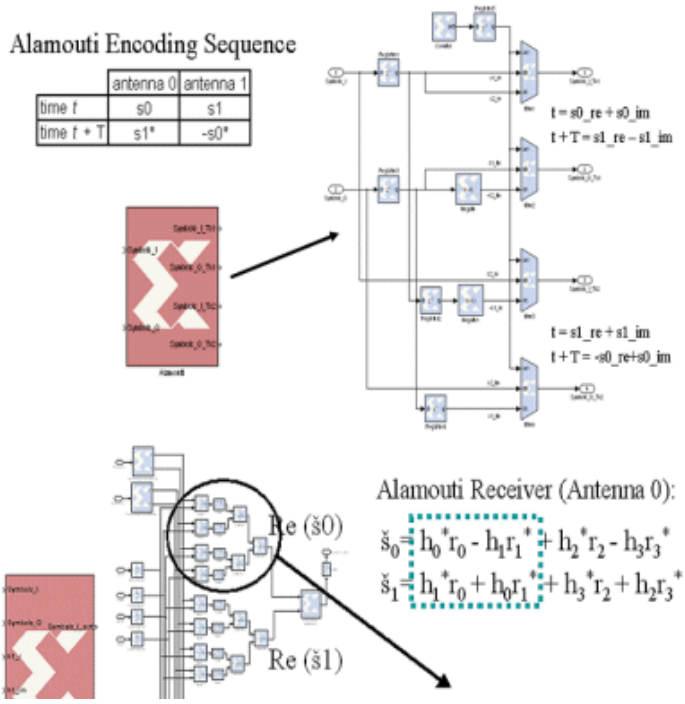
Our design of the complete 2×2 transceiver, including the Alamouti space-time decoder, was accomplished using blocks from the *Xilinx System Generator for DSP* tool, thus eliminating the need for VHDL programming.

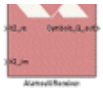
Fig 18 shows the 2×2 MIMO transmitter and receiver implementations. *Fig 19* shows the detailed implementation of the Encoder and "Real" component of the Alamouti Space-Time decoder. The imaginary component implementation is very similar, with slight modifications in + and – operations.

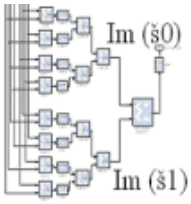


18. 2×2 MIMO transmitter and receiver FPGA implementations using the System Generator for DSP tool. (Click above image to see a larger, more detailed version)

Thanks to the bit-true and cycle-accurate simulation capability of System Generator for DSP, we can validate the design before implementing it on hardware. Once this is done, we can implement a real-time prototype of the 2×2 MIMO transceiver on an FPGA-based hardware platform.







$$\mathrm{Re}(\check{s}0) = \mathrm{Re}(h_0^* r_0 - h_1 r_1^*) =$$
$$((r0\_re \times h0\_re) + (r0\_im \times h0\_im))$$
$$-$$
$$((r1\_re \times h1\_re) + (r1\_im \times h1\_im))$$

can compare the results of a Virtex-II implementation to that of a Virtex-4 device.

The table in *Fig 20*. below shows a comparison of both implementations:

	Number of Logic Slices		
Design	Virtex-II XC2V3000	Virtex-4 SX25	Saving
Transmitter	2692	2865	3%
Receiver	8233	7254	12%

20. Summary of MIMO transceiver implementation results.

While this implementation puts the same design in both the FPGAs, if the user can take advantage of Virtex-4 DSP48 specific capabilities in the System Generator, the implementation in Virtex-4 devices will be more performance and area efficient.

Summary
The emergence of innovative technologies like MIMO that utilize complex DSP algorithms which require the massive parallel processing capabilities inherent to FPGAs, continues to precipitate the electronics industry's growing dependency on platform FPGAs as an integral source for emerging DSP processing solutions.

Furthermore, the FPGA's ability to deliver unmatched design flexibility and time to market make it the best choice where early implementation of evolving standards is a strategic imperative.

Finally, model-based design tools and methodologies greatly simplify the means by which FPGA-based DSP designs can be implemented and validated – even by engineers that are relatively unfamiliar with FPGA design. Taken together, these values firmly establish the role of the FPGA as a premier DSP processing platform.

Bibliography

1. XtremeDSP for Virtex-4 FPGAs, Xilinx User Guide, December 19, 2005.
2. Applications reference material from Niall Battson, Applications Engineer, Xilinx.
3. "A Simple Transmit Diversity Technique for Wireless Communications", Siavash M. Alamouti, IEEE JOURNAL on select areas in Communication, Vol. 16, No. 8, October 1998.
4. Robust MIMO Wireless Communication in the Presence of Interference Using Ad Hoc Antenna Arrays, Dr. Daniel W. Bliss & Amanda M. Chan, MIT Lincoln Laboratory.
5. DSP: Designing for Optimal Results, Xcell Books, April, 2005.
6. Virtex-4 FPGA User Guide, 2005.

Anil Telikepalli *is Sr. Manager, Virtex Solutions, Xilinx. Anil can be contacted at* anil@xilinx.com.

Etienne Fiset *is an Applications Engineer at Lyrtech. Etienne can be contacted at* etienne.fiset@lyrtech.com.

This article is excerpted from a paper of the same name presented at the Embedded Systems Conference Silicon Valley 2006. Used with permission of the Embedded Systems Conference. Please visit www.embedded.com/esc/sv.