
FPGA-based FIR Filter Using Bit-Serial Digital Signal Processing

Introduction

This application note describes the implementation of an FIR (Finite-Impulse Response) Filter with variable coefficients that fits in a single AT6002 FPGA. The filter uses a bit-serial arithmetic approach to the digital signal processing and is based upon the Atmel AT6000 series FPGAs. This note discusses the bit-serial arithmetic used for compact and efficient implementation of real-time DSP applications and details the implementation of an 8-tap FIR filter. The input and coefficient word-lengths in the included design example are both 8 bits. Each word-length can be adjusted to any size via the design techniques shown herein. The internal precision and output word-length is the sum of the input and coefficient word-lengths.

Benefits of Bit-serial Machines

Parallel arithmetic structures, such as single-cycle multipliers and wide adders, are often inefficient in traditional FPGAs. In cases where the element fits, the routing resources are overtaxed and the resulting design operates too slowly to meet the application's requirements. Parallel structures process all the bits simultaneously at a significant hardware cost. Bit-serial machines, by comparison, process the input one bit at a time. The advantage is that all the bits pass through the same logic, resulting in a huge reduction in the required hardware. Typically, the bit-serial approach requires 1/nth of the hardware required for the equivalent n bit parallel design.

The cost of this logic reduction is that the serial hardware takes n clock-cycles to execute, while the equivalent parallel structure executes in one clock-cycle.

However, the speed/logic-cost product is usually better than in equivalent parallel designs because the logic delays between registers are generally significantly smaller. This means that the serial machine can operate at much higher clock frequencies. In FPGAs, signal routing contributes sizable propagation delays and consumes resources. Serial structures tend to have very local routing, often to one destination. In contrast, parallel machines require many signals to span the width of the processing element. In some cases, the overall throughput for a serial design implemented in an FPGA can actually exceed that of an equivalent parallel design in the same device.

AT6000 Architecture

The Atmel AT6000 series FPGA architecture is tuned for high-speed computing and data-path applications. The FPGA is SRAM-based with a symmetrical, fine-grained array of logic-based cells. Routing within the array is either by bus or by direct connection between neighboring cells. Direct connections are most useful in tight spaces and over short distances where maximum speed is required. There are two kinds of busses: local and express. Local busses are the link between the array of cells and bussing network. Express busses are not directly connected to cells and hence provide higher speeds.



FPGA-based FIR Filter

Application Note



Connective units, called repeaters, spaced every eight cells, divide each bus, both local and express, into segments spanning eight cells. Repeaters can:

- isolate bus segments
- connect local-bus segments
- connect express-bus segments, and
- provide local/express transfers.

These busses and cells run in two dimensions, and an array of 8 x 8 cells surrounded by the repeater units is called a sector. A sector is pictured in Figure 1.

The core cell is simple and small, yet provides many important and commonly used logic functions, including those most useful in pipelining and arithmetic: a D-type register, a NAND/XOR pair, and a two-input multiplexer. The register is essential in pipelining, which is the most effective way to achieve high system throughput. The high register count of the AT6000 architecture (there are 3136 registers in the AT6005) makes pipelining simple and efficient. The NAND and XOR provide efficient implementation of full-adders, counters, and other arithmetic functions. The core cell is

shown in Figure 2. For more detailed information on the AT6000 FPGAs, please consult the Atmel Configurable Logic Databook.

Because its four sides are identical, each cell is completely symmetrical. Cell symmetry, together with array symmetry, high cell connectivity, and fine-grained architecture, makes the design of hard macros (circuits with fixed layouts) especially easy. More specifically, it facilitates the creation of bit-slice macros, hard macros in which the inputs/outputs on one side are aligned with the outputs/inputs on the opposite side. A larger circuit is constructed by simply abutting these bit-slice macros.

The Atmel IDS Component Generator Tool utilizes this technique to parametrically create data-path and other regular structures with any arbitrary user-defined width. More important, these automatically generated macros, although hard macros in a relative sense, can be moved, copied, and oriented anywhere on the symmetrical array without changing the timing parameters of the original instance.

Figure 1. Atmel AT6000 Series SRAM-based FPGAs

Architecture Optimized for High-Speed Computing

- Symmetrical, Fine-grain Architecture
- Logic-based Cells
- Hierarchical Bussing Networks
- Deterministic Timing
- No Fanout Restrictions
- Reconfigurable "on the fly"
 - Full (800 μ s for 5K gates)
 - Partial (200 ns/cell)
 - Without register data loss
 - No disruption of clocks or I/O's
 - Built-in data checking circuitry

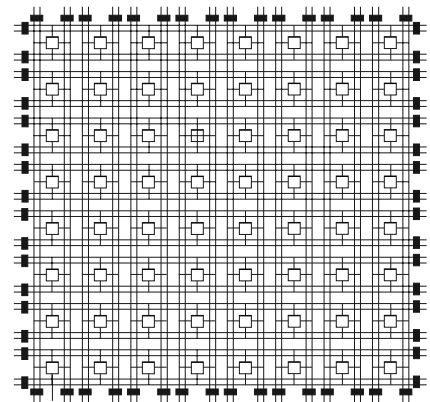
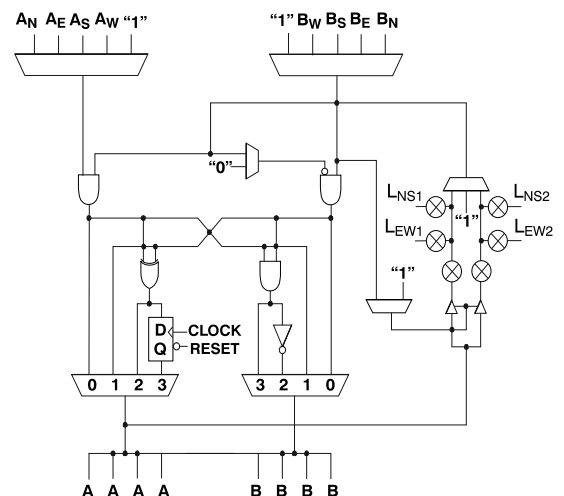


Figure 2. Atmel Series Cell Structure

- Lots of cell states (72)
 - XOR/NAND
 - MUX
 - Register
 - Tri-state
 - Wire
 - Combinations
- Fast

– Wire:	1.2 ns	
– NAND:	2.2 ns	
– XOR:	2.4 ns	
– Register:	Tsetup:	2.0 ns
	Thold:	0.0 ns
	Tco:	2.0 ns



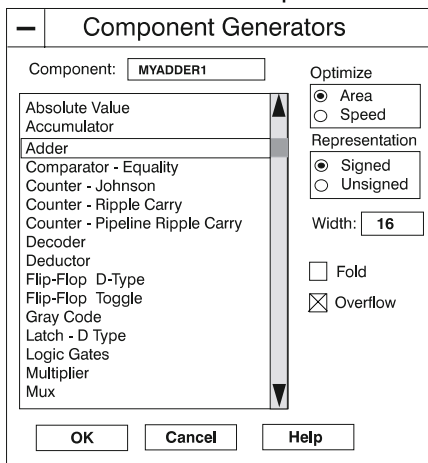
The Automatic Component Generators

A key part of the Atmel FPGA Integrated Development System, the Automatic Component Generators permit rapid deployment of datapath and logic functions. Designed to exploit Atmel's register-rich, dynamically reconfigurable FPGAs, high-speed custom logic functions can be created and implemented, resulting in significantly improved performance for compute intensive applications. This software, when used with Atmel's FPGA family, facilitates quick and easy implementation of compute-intensive logic functions, such as multipliers, adders, counters, and accumulators. Users can specify individual parameters, including width,

pipelining, and other function-specific options. The Component Generator Window is shown in Figure 3.

Within minutes, the design tool will automatically create a "hard layout" with worst case speed, area, and power consumption information, as well as generate a schematic symbol for the function. The end result of this process is a fully specified, reusable circuit that can be used for any size Atmel FPGA, in any location or orientation, without affecting its speed or function. Consult the Automatic Component Generator Handbook for specific details and application metrics.

Figure 3. Automatic Macro Component Generators



- Parameterized Functions
- Hard Placement
- Automatic Schematic Generation
- Automatic Symbol Generation
- Automatic Timing Analysis/Power Estimation
- Back Annotation
- Pipelining Option
- Excellent for Structured Logic and Data-path Functions

Basic Bit-serial Building Blocks

The basic functions required for almost any signal processor include addition, negation, and storage registers. These basic functions can then be used to construct the more complex structures such as accumulators and multipliers. In most cases, using a bit-serial architecture simplifies the hardware required since all of the data bits pass through a single bit-wide element. For bit-serial arithmetic, the two fundamental building blocks are the bit-serial adder and the two's complement circuit.

Bit-serial Adder

A bit-serial adder is sometimes referred to as a carry-save adder because of the nature of its operation. Refer to the bit-serial adder schematic in Figure 4. It is constructed using a full-adder with registers on both its carry and sum outputs. The registered carry output is fed back to the carry input of the full-adder. In operation, the two words to be added are simultaneously shifted, least significant bit first, into the main inputs of the full-adder. The carry output from the addition of each bit is stored and used as the carry-in for the addition of the next bit. Essentially, the carries remain stationary as the inputs are shifted through the adder. The output of the circuit is registered, performing bit

pipelining. The latency is one bit-time. The bit-serial adder must be initialized before each new word by clearing the carry save register or adding an AND gate to the carry feedback path. This insures that the first (LSB) addition is performed properly. The input words must be of the same length. Three AT6000 core logic cells are used to implement the bit-serial adder. The FDHA cell illustrates the AT6000 compute-orientation, a registered half-adder in a single cell, providing a fundamental building block for addition, accumulation, and counting.

Bit-serial Two's Complementor

The second fundamental function is a circuit which computes the two's complement of the input word, shown in Figure 5. The serial algorithm for two's complement generation starts with the least significant bit, copies each bit until the first one is encountered, and then all remaining bits are inverted. The circuit uses two flip/flops, an XOR gate, and an OR gate. The detection flip-flop must be reset before each input word, since it causes the XOR to invert the input continuously after the first logic one is detected. The output is registered, so the function has a one bit-time latency.

Figure 4. Bit Serial (Carry Save) Adder

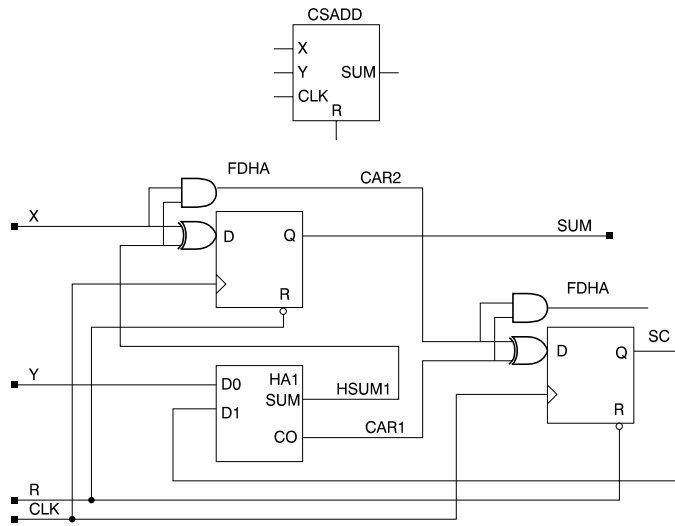


Figure 5. Two's Complement

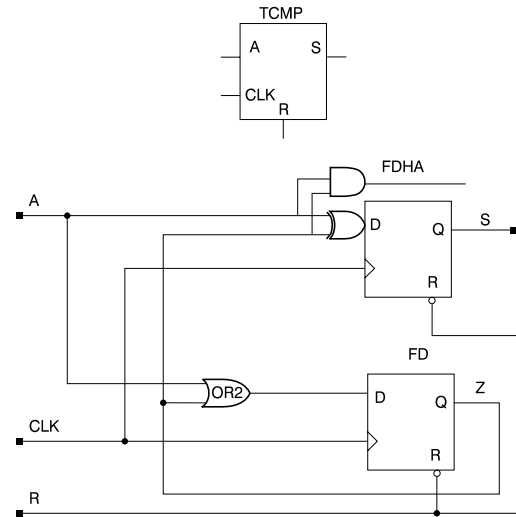
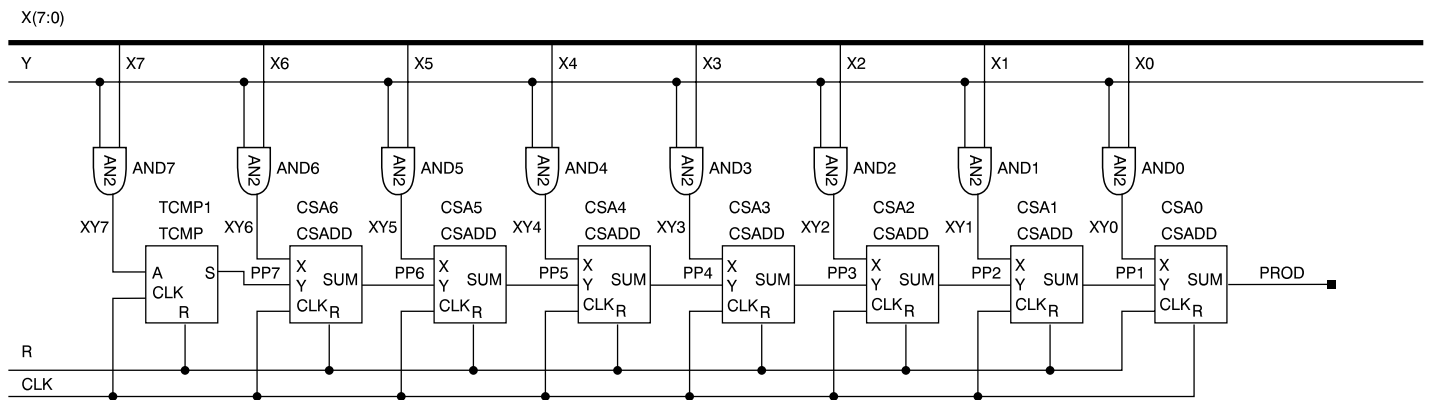


Figure 6. 8-bit Serial-parallel Multiplier



Delays

The last elementary function is the bit delay (a bit time is one clock cycle). The delay is required to align words, produce word delays needed for many algorithms, and propagate control signals that must also be aligned. Word delays are usually implemented as shift registers of the proper length. As will be shown later, depending on the input word length, appropriate delays are required between arithmetic operations. The Atmel Component Generator can automatically produce shift registers of any length, reducing the design task to mere specification.

Higher-level Functions

Most of the higher-level math functions are constructed from the above elements. It can be shown that many custom math functions can be constructed using the basic building blocks of bit-serial arithmetic, such as accumulators, deductors, and squaring circuits. The FIR filter

requires multipliers and a serial column adder, which adds the tap values simultaneously. Their operation is discussed below.

Serial-parallel Multiplier

Multipliers are essential to most digital signal processing algorithms. The simple serial-by-parallel multiplier (SPM) is particularly well suited for FPGA implementation because all of its routing is to nearest neighbor cells with the exception of the input. This multiplier has one serial input, one parallel input (used for coefficients), and a serial output. The number of building blocks is a function of the width of the parallel input. An 8-bit SPM is shown in Figure 6. This multiplier performs the familiar shift-add algorithm: the parallel input is multiplied by each bit of the serial input as it is shifted in, and each partial product is added to the shifted accumulation of the previous products. The bitwise products are the logical AND of the input bit with each bit of the parallel input. The shifting accumulator is constructed by

chaining a series of bit-serial adders together so that the inputs are bit parallel and the sum is downshifted on each clock cycle. The serial output is taken from the output of the least significant bit adder. The output has the same weight as the previous serial input bit with a latency of one bit-time. The number of bits in the output is equal to the sum of the number of serial input bits and the parallel input bits. Since the serial input has to be of the same length as the output, it is sign-extended.

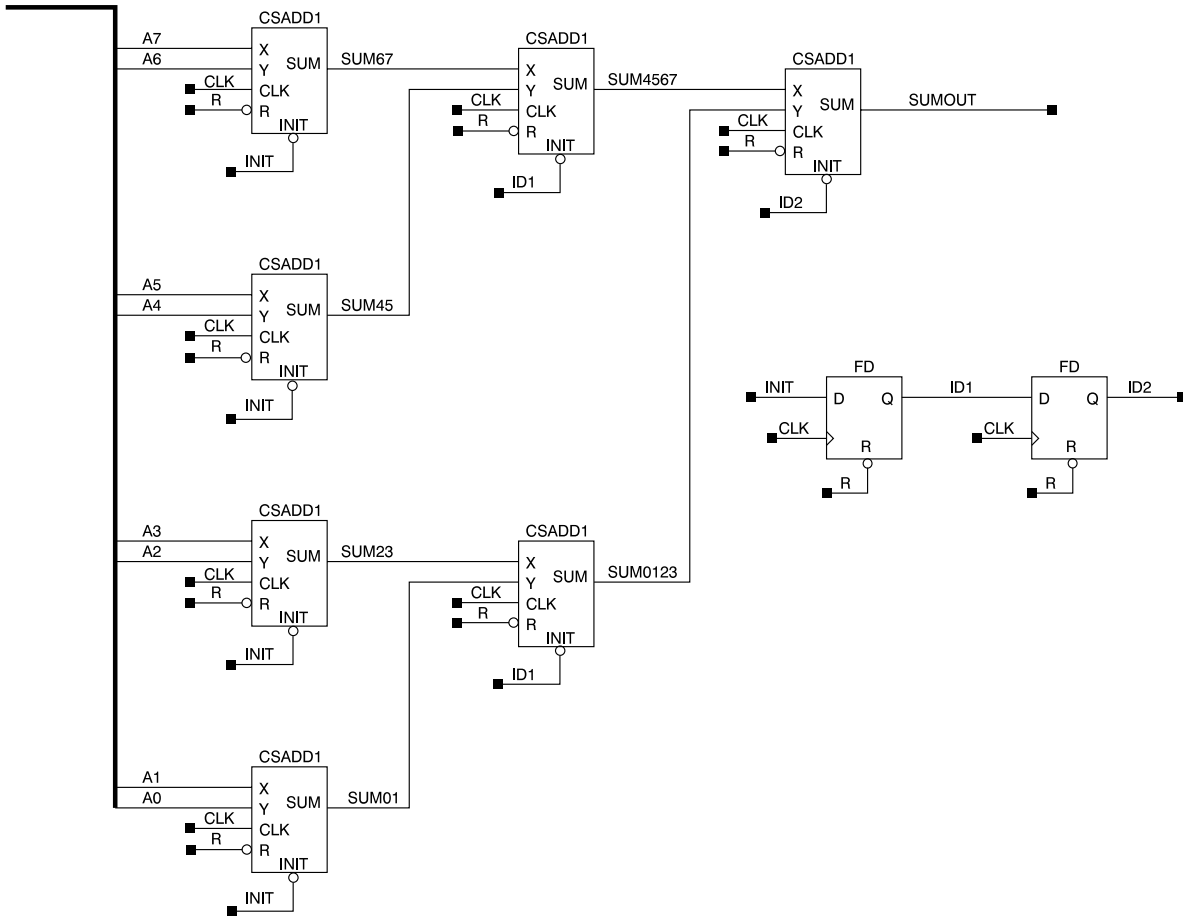
The serial multiplier must be cleared before a new word is input to prevent errors. This is especially true if the X (parallel) input is negative since the two's complement circuit cannot self clear. No other controls are required. The serial input has to be sign-extended by the number of bits in the parallel input (i.e., to make the number of bits in the serial input equal to the number of bits in the output). The output is always a full precision output.

Serial Column Adder

An adder structure capable of simultaneously adding more than two inputs is a desirable function. This is easily

accomplished by a tree of serial adders. Each serial adder (carry-save adder) combines two input streams into one output, hence each level of the tree structure reduces the number of serial streams by half, adding one bit-time of latency in the process. A serial column adder (SCADD) designed in this way allows an arbitrarily large number of inputs to be summed together without a sacrifice in the bit rate. If an odd number of inputs exist in a level, the odd input can be passed on to the next level via a delay register to maintain the bit-alignment. If overflow is to be avoided, one bit of growth must be allowed for each level in the adder. Since the input and output must have the same number of bits, the input must be sign-extended (guard bits) to prevent overflow. The number of levels and hence the latency and number of guard bits for an n-input serial column adder is equal to $\text{Log}(n)$ rounded up to an integer. As with single bit-serial adders, the inputs and outputs run least-significant bit first. The 8-input SCADD used in this design appears in Figure 7. The initialization signal resets the bit-serial adders. The control signal, INIT, is delayed and applied to the successive stages insuring that they are timed properly.

Figure 7. Serial Column Adder



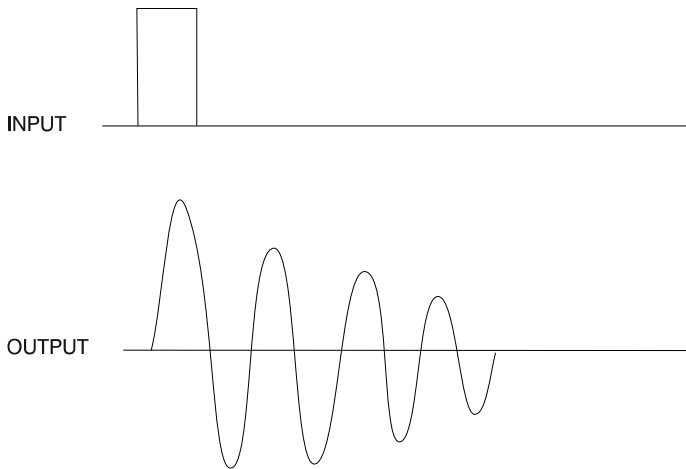
The FIR Filter

The FIR filter algorithm is a discrete convolution of the input signal with a set of coefficients. This type of digital filter is unconditionally stable due to the fact that its output is only a function of its input and will ultimately settle to zero. This is exemplified in the impulse response graph shown in Figure 8. Described mathematically, the FIR response can be defined as:

$$Y_{[k]} = \sum^n C_i X_{[k-i]}$$

The signal flow diagram of a typical FIR Filter consists of a series of multiplies, adds, and “Z” delay elements to store the input words as they transverse the processor. A word-delay shift register and serial-parallel multiplier are used for each tap in the filter, with the parallel inputs of the multiplier programmed with the value of the associated coefficient. The summation stages of the flow diagram are replaced by the serial-column adder with as many inputs as there are taps.

Figure 8. Transfer Function of Filter Response



Circuit Description

The 8-tap FIR Filter consists of an array of serial-parallel multipliers with coefficient storage, delay shift-registers, the serial column adder, and the initialization logic, as shown in Figure 9. The coefficients are stored as constant cells in the Atmel AT6000 FPGA architecture. This provides the compact and efficient storage of a fixed-coefficient scheme,

but is variable in real-time through the use of Cache Logic™ (dynamic partial reconfiguration of the FPGA). The locations of the coefficient constants are noted during the interactive floor-planning session and are used later to develop the coefficient bit-stream. Any one or more of the coefficients can be modified by sending the appropriate bit-stream(s) to the FPGA. While the updates occur, the filter continues to operate undisturbed, as does the rest of the circuitry on the array. This is another unique benefit of Atmel FPGAs.

In this application, the carry-save (bit-serial) adders and two’s complement circuits are used to build up the serial-parallel multiplier (SPM) and the serial column adders (SCADD). The multiplier is easily built since the number of bit-serial adder stages is a function of the parallel input (coefficient) word width. In this case, an 8-bit coefficient is needed, so one two’s-complementor followed by seven stages of bit-serial adders defines the basic multiplier structure. The AND gates perform the multiplication on a bit-by-bit basis as the serial input word is shifted into the multiplier. The SPM has been converted into a UDM, user-defined hard macro function, by using the interactive floor-planner to drop the cells into a logically-oriented physical unit. This SPM macro can now be used repeatedly, knowing that the layout is defined. Bit-serial arithmetic units layout in fine-grain FPGA architectures very efficiently. Using the cell-to-cell interconnection permits excellent device utilization coupled with high-speed performance.

The balance of the FIR filter elements are the input word delay stages that are implemented as shift registers and the coefficient storage. As mentioned above, the SPM and other bit-serial arithmetic functions usually have a latency of one-bit time. Consequently, the delay stages need to compensate to this latency by providing additional bit-time delays in addition to the product delay. The total delay requirements will vary depending on the architecture and functions used. In this example, we are summing the outputs from the eight taps using the SCADD circuit. This adder tree has three levels of bit-serial adders and hence a latency of three bits. Therefore, the total delay word-length requirement is 20 bits, 16 for the product length, one for the SPM, and three for the SCADD. These registers are specified to the Component Generator and the shift register macro is created in minutes.

The AT6002 Design Example

The design example, shown in Figure 10, is an 8-stage FIR Filter with 8-bit data words and 8-bit coefficients. An 8-bit parallel input port and 16-bit parallel output port is provided for interface to standard systems. This interface can be easily double-buffered (or more) to accommodate variable data rates, system clocks, and processing speed. A control circuit synchronizes the input data load signal and generates the initialization, shift, and other control signals.

The interface circuitry was also produced with help from the Component Generator. The serial-to-parallel and parallel-to-serial registers, input and output registers, and actual input and output pad elements are all generated parametrically and instantiated into the top schematic sheet.

Language-based Design Support

VHDL-based design entry is also supported by the component generation technique. Top-level architectures are typically structural, and the component declarations and instantiations enable the use of generated macros. The Component Generators also produce "Component Declarations" in VHDL, based upon the input parameters. This file can be pasted into a VHDL architecture body, facilitating code generation. The Place & Route Tool reads the

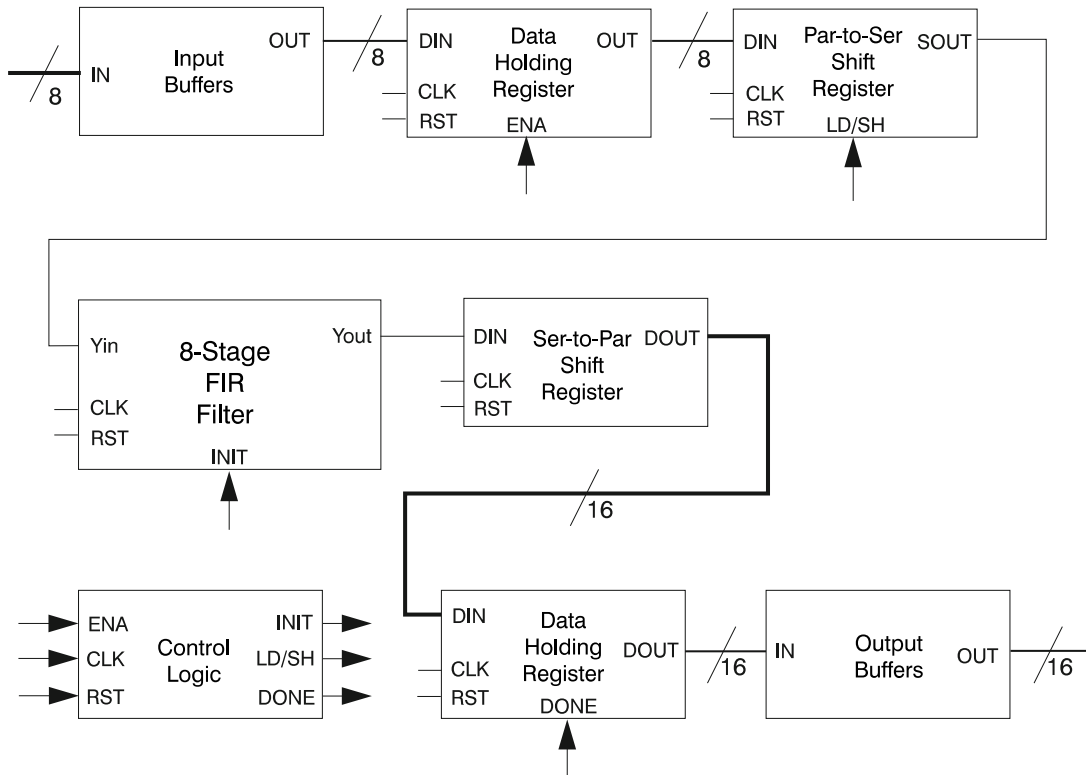
user-defined macro library to determine if the instantiated components are available as hard macros and uses them instead of any synthesized versions.

Floor Planning and Design Layout

The Component Generator makes deployment of the design in silicon a straightforward task. The generator is used to produce many of the functional blocks of the design, as described above. The components are placed and routed macros in the Atmel architecture and predominantly use the cell-to-cell wiring to implement the interconnection of the logic cells comprising the function. There are two important benefits to this method:

1. The macro component is essentially transparent to the bus routing system of the FPGA, which means that placement of the component does not block the ability of the tools to route signals through the component. This is an intrinsic benefit of the hierarchical nature of the fine-grained architecture.
2. The relative timing and hence operational speed of the component is independent of its placed location. This is due to the symmetrical architecture of the FPGA.

Figure 10. 8-stage FIR Filter using Atmel AT6002



Multiple instances of components can be placed in arbitrary locations in the FPGA, and the timing performance remains the same. This makes design iterations a much more straightforward task, since re-engineering of the timing parameters is not necessary as in coarse-grained architectures.

Efficient utilization of the device is the result of interactive floor planning using the generated components. This task is netlist driven so that net congestion and routing requirements are visually presented to the user in the interactive placement tool. Since the components are pre-routed, the job of the router is simplified as well. After floor planning and placement of the SPM array and other macros, the automatic place and route tools are used to complete the physical design. Using the Interact Floorplanner/Layout Tool, this design was planned and placed in one-hour. The autorouter completed in two minutes.

Table 1. AT6002 Design Statistics for “App-FIR8”

Utilization Summary	Utilized
Number of Macros	71
Number of Nets	147
Number of Pins	322
Average Pins per Net	2.19
Maximum Pins per Net	6
Number of Flip-Flops	356
Number of Gates	289
Number of Turns	22
Number of Buses	106
Local Buses	102
Express Buses	4
Number of I/O's	25 of 64
Number of Cells	900 of 1024
Number of Clock & Reset Combinations	2
Number of Equivalent Gates	3818.0

Summary

This design fits into the AT6002, the smallest member of the Atmel FPGA product family. It consists of 3818 equivalent (ASIC) gates, and will run at clock frequencies up to 100 MHz. Since each sample takes 20 bit-times (clock cycles), an effective continuous sample-rate of 5M samples per second can be attained. If sample-rate requirements fall into the audio/telephony range (8-50K samples per sec.), FIR filters of 100 taps or more can be implemented in similar size arrays using “folding” techniques. The coefficients are fixed at design time; however, they can be modified via Cache Logic techniques.

In a future application note, the implementation of dynamic coefficient modification via partial reconfiguration of the FPGA will be discussed.

Conclusion

It has been shown that the right tools and techniques coupled with innovative features in silicon architecture can yield complete DSP functions in a single FPGA. These approaches have been extended to multi-FPGA systems that are in-system reconfigurable on-the-fly, permitting features and capabilities that were unachievable just a few years ago. The cost/feature trade-offs are further reduced by Cache Logic, a technique of time-sharing the resources of FPGA-based products and systems as well as a means of modifying the hardware in real time.

Bibliography

- [1] P. Denyer and D. Renshaw, VLSI Signal Processing: A Bit Serial Approach. Selected Readings, Addison-Wesley, 1985.
- [2] Atmel Corporation, Configurable Logic Databook, AT6000 Series Data Sheets, Atmel Corporation.
- [3] R. J. Andraka, FIR Filter fits into an FPGA using a Bit Serial Approach, Proceedings of the PLD Conference, 1993.
- [4] Frederick Furtek, An FPGA Architecture for Massively Parallel Computing, Concurrent Logic, Inc., 1990.



Atmel Headquarters

Corporate Headquarters

2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe

Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia

Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan

Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs

1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset

Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand

North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

BBS

1-(408) 436-4309

© Atmel Corporation 1999.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Marks bearing ® and/or ™ are registered trademarks and trademarks of Atmel Corporation.

Terms and product names in this document may be trademarks of others.



Printed on recycled paper.

0529C-09/99/xM