

PROGRAMMABLE SYSTEM LEVEL INTEGRATION ON THE DESKTOP

FPSLIC™
Field Programmable System Level ICs



is a registered trademark of Atmel Corporation

2325 Orchard Parkway, San Jose, 95131

TABLE OF CONTENTS

INTRODUCTION	2
THE ASIC SOLUTION	2
MULTIPLE APPROACHES TO PROGRAMMABLE SYSTEM LEVEL INTEGRATION	3
HIGH-DENSITY FPGAS	3
<i>Price</i>	<i>3</i>
<i>Long Design Cycles</i>	<i>4</i>
<i>The Role of Intellectual Property</i>	<i>4</i>
<i>Silicon Inefficiency</i>	<i>5</i>
<i>Power Consumption Issues</i>	<i>7</i>
ASIC/FPGA HYBRIDS	9
ATMEL'S FIELD PROGRAMMABLE SYSTEM LEVEL IC	8
THE FPGA	10
THE MICROCONTROLLER	11
THE MEMORY	11
EDA TOOLS	9
DESIGNING WITH FPSLIC	10
FPSLIC POWER CONSUMPTION	11
<i>MORE EFFICIENT USE OF LOGIC RESOURCES</i>	<i>11</i>
<i>INTEGRATION</i>	<i>11</i>
<i>FPGA CLOCKING TREE STRUCTURE</i>	<i>12</i>
<i>HIGH THROUGHPUT MCU</i>	<i>12</i>
IN-SYSTEM RECONFIGURABILITY	12
RECONFIGURABLE PDA APPLICATION	13
DESIGN TOOLS	13
SYSTEM DESIGNER CO-VERIFICATION EDA TOOL SUITE	14
FPGA DESIGN TOOLS	18
<i>Macro Generators for Re-usable IP Design</i>	<i>16</i>
<i>HDL Planner™</i>	<i>16</i>
FPSLIC FIRMWARE DESIGN AND DEBUGGING	17
CO-VERIFICATION ENSURES SPEEDY DESIGN CYCLE	17
CONCLUSION	18

INTRODUCTION

System level integration (SLI) is rapidly becoming the preferred way to implement electronic designs. Integrating all the system functionality in a system-level IC increases performance, reduces power consumption, cuts unit production costs and allows smaller products. These are particularly important benefits in telecommunications, multimedia and networking applications.

A “system” usually consists of three main digital building blocks: a processor, memory and logic. Usually, the processor is used for control flow logic, the memory is used for program and data storage and the logic is used for datapath logic. A true system-level solution must contain all three elements, eliminating multiple IC’s and inter-chip signal bussing. The FPSLIC™ contains all elements, and as set forth in this article is the best and most cost efficient solution.

The ASIC Solution

Until now, system level integration has been implemented in cell-based or masked Gate Array ASICs because these were the only solution available with sufficient density to handle system-level designs. Unfortunately, ASICs have high NRE costs, long lead times and significant minimum order quantities. As a result, system-level ASIC implementations have been accessible only to the highest volume designs with relatively long product life cycles. Minimum volume requirements for system-level ASIC devices are often more than \$500K per design per year. Designs with short product life cycles, low to medium volumes, time-to-market pressures or rapidly evolving standards can not afford the lengthy development cycle, risk and high NRE charges associated with an ASIC solution.

Even when the volume/dollar criteria are met for an ASIC solution, any change in the design to correct an error or improve it will leave the developer with a large inventory of possibly useless parts and another lengthy ASIC design cycle. This is particularly problematic for rapidly evolving designs such as those in telecommunications, networking and multimedia. For these and other designs a programmable solution is preferable because the design can be changed at will both during development and in the field. ASIC solutions are not an option for these designs.

A substantial number of these rapidly evolving designs are implemented in a combination of programmable logic, discrete standard products (microcontrollers and memories) and Application Specific Standard Products (ASSPs), such as T1 interface, ATM, 10/100 PHY, and video/audio codecs. Although this approach offers the flexibility to evolve designs rapidly it does not offer the performance, power, space and reliability advantages of a monolithic system level integrated circuit. A single-chip, programmable solution is clearly the preferable alternative.

Multiple Approaches to Programmable System Level Integration

FPGA and other IC vendors have developed a variety of approaches to providing programmable system level integration. These include “pure play” high-density FPGAs and hybrid devices that combine both FPGA and fixed logic functionality.

High-density FPGAs - The mostly widely promoted means of achieving programmable SLI today are FPGA families that boast as many as a million gates. Although industry analysts believe these gate counts may be substantially overstated, these devices are still large enough to support system level integration of designs that might otherwise go into a masked- or cell-based ASIC. FPGAs now compete with masked-ASICs in terms of both density, and in the case of low density FPGAs, price. High density FPGAs are being proposed as a programmable, single-chip solution for system level integration. Although the programmability of the large FPGAs is very attractive, they have some significant drawbacks:.

Price - Although deep sub-micron process technologies have reduced the prices of low and medium density FPGA so that in many cases FPGA prices are on a par with those of ASICs, high gate count programmable devices are extremely expensive, some devices currently sell for over \$4000.00 each. The extremely high prices of these devices limits their use to ASIC prototyping and production runs of a handful of very high priced products. When one considers that a masked-ASIC of comparable density cost about <\$50, these large FPGAs are out of reach for most volume designs/applications.

Long Design Cycles – Although FPGA devices can cut the ASIC development cycle in half, the complexity of large FPGAs mandates a significant design and development process for system level designs. Today “time-to-market” is the difference between success and failure of a product. Designing a million gates of FPGA logic takes a great deal of time. Frequently, intellectual property cores are used to reduce the design cycle. However, integrating vendor supplied soft IP into a design is in itself often a cumbersome and time-consuming process.

Simulation is another problem with large FPGAs. HDL simulations are notoriously slow for simulating large designs, especially ones using complex soft IP cores. Simulating a 1M gate FPGA design can take so long that many designers simulate less thoroughly than is desirable or not at all. The result is that these designs are more likely to have undiscovered bugs that extend the debug cycle, further delaying product introductions.

This problem is further amplified if a microcontroller soft IP core is being used in a large FPGA design. Conventional MCU design methodologies are not available to the designer in the large FPGA based design flow. Typically, microcontroller designers have code development and debugging tools that are used to debug the microcode. These tools are often not available for soft IP cores used on a large FPGA, so code development and debugging are problematic if not impossible. Furthermore, because of the lack of code development and debugging tools available for processor cores, the integration and debug of the microcontroller portion of a these designs is extremely difficult. Similar arguments can easily be made for timing analysis on system-level FPGA designs.

The Role of Intellectual Property - One solution to the complexity of designing system-level FPGAs is to use “drop-in” soft intellectual property cores. Memory, logic and a limited number of processor IP cores can be purchased from third-party vendors and dropped into large FPGAs. However, soft IP cores are expensive, difficult to integrate in the design, and tend to be silicon inefficient. The difficulty of integrating IP cores from different third-party vendors

can significantly extend the product development cycle. As an example, one 8051 core costs over \$10,000 to license and uses 1010 logic elements or 16.4% of a large FPGA at the list price of \$4,298 for each FPGA, the silicon cost of the 8051 core is \$704, excluding the amortized cost of the core itself.

There are additional design problems associated with building the interfaces between the various cores and correcting timing problems. Studies have shown that up to one half of the typical design project is spent in the integration and test phase, which in reality becomes an exercise in correcting the accumulation of errors from the front end of the design cycle. These problems are magnified when multi-site, multi-engineer development teams work on large FPGA-based systems. The errors often reach all the way back to the specification and partitioning phase, where ambiguities in the hardware/software interface were introduced and then amplified during the hardware/software implementation phase. Often, the remedy of these errors is forced into software due to the long lead times and high costs of ASIC or large FPGA turns – even though a software fix may result in compromised performance or functionality in the final product.

Silicon Inefficiency - Although FPGAs represent an efficient means of implementing data-path functions, control logic is better suited to a CPLD or microcontroller architecture. FPGA implementations of control logic are not silicon efficient. This is demonstrated by the architectural changes some FPGA companies implemented in their next generation large FPGAs. The inclusion of CPLD blocks of logic in these large devices clearly shows a weakness in implementing control flow logic in current homogeneous large FPGA solutions. The inclusion of CPLD structures helps control flow in the those FPGA architectures, however, the CPLD solution is still less silicon efficient than a microcontroller at implementing control flow and decision making.

For example, a mail sorting system must acquire the visual data on each piece of mail using a high-speed camera, convert the data from analog to digital, pre-process it to identify the location and orientation of the address, process the address pixels to decode the address and then generate

machine readable code that can be read by the sorting machine. Just identifying the address in the image requires an enormous amount of data processing because each pixel has to be compared to all the pixels surrounding it. Since so much of the image is likely to be irrelevant, it would be inefficient and slow to process the entire image with a processor. FPGA-based DSP algorithms can be used to filter out those portions of the image that are not likely to be part of the street address. This process eliminates most of the image, thereby cutting the processing problem to a manageable size. FPGAs are ideal for these types of operations and can perform them significantly faster than a DSP processor.

A processor, on the other hand, performs the algorithms for image extraction, de-skewing, rotation and data interpretation much more efficiently than an FPGA. FPGAs are inadequate to these tasks because the amount of complex “decision” making involved (control flow). They are simply too slow at performing control flow tasks. Implementing a processor that is capable of the tasks outlined above using FPGA core cells is difficult and very silicon inefficient and therefore very expensive. Since all systems contain both datapath and control flow logic, system level integration in a general purpose FPGA cannot ever be terribly efficient. In short, on its own, a general purpose FPGA is not a solution.

Power Consumption Issues - Power consumption has three components: static power consumption, dynamic power consumption and I/O or system power consumption.

Static power consumption is a function of the number of transistors in the device and increases with the size of the FPGA. However, with careful design even large FPGAs can have very low static power consumption. It can be reduced further by re-using system resources in real-time through dynamic reconfiguration.

The second component of power consumption is dynamic. The combination of a large numbers of core cells for design implementation and the internal clock distribution tree significantly contribute to power consumption. Thus the larger FPGAs that must be used for SLI designs draw

proportionately more power. Programmable SLI in a large FPGA is likely to consume a great deal of power.

The third source of power consumption, is consumed in the I/O (input/output) structure. Significant power is dissipated each time an output switches from one logic state to another. Capacitive loading on the PCB (printed circuit board) is the reason for this power consumption. Reducing the number of parts in a system through integration will significantly reduce this aspect of system power consumption. Since most large FPGAs used still have to connect to the high-bandwidth microcontroller bus, significant power is consumed through this interface.

Because of the above issues, just increasing the density of FPGAs is probably not the most practical solution to achieving programmability with system level integration.

ASIC/FPGA Hybrids - Recently, a handful of fab-less IC start-ups have been announced that develop hybrid devices that integrate blocks of programmable logic with a hard-wired microprocessor cores. The introduction of these devices indicates the need for products that address the SLI needs of the systems architect.

These devices are too specialized to fall into the general purpose SLI category. For example, one configurable microcontroller is positioned as a replacement for a single-chip MCU with peripherals that are programmed into an on-chip FPGA with 5,000 to 30,000 gates. These devices provide embedded systems designers custom MCU derivatives without having to order the 100,000 units conventional MCU vendor require to consider creating a customer-specific derivative.

Another product which has announced a yet to be introduced “reconfigurable network processor” Although product details have not yet been disclosed, all indications point to a specialized product that addresses specific telecom/networking applications.

Other FPGAs have been announced that include specialized functions, such as PCI. However until now, no company has introduced a true general purpose system-level product with a processor, memory and datapath logic.

None of these announced solutions provides truly programmable system level integration namely (programmable) logic, microcontroller and memory, with design tools that support system-level design methodology.

ATMEL'S FIELD-PROGRAMMABLE SYSTEM-LEVEL IC (FPSLIC)

Atmel's response to the challenge of combining programmability with system level integration has been to develop a family of system-level ICs with specific dedicated functionality that provides a silicon efficient means of creating a system on a chip. Atmel's Field-Programmable System-Level ICs (FPSLIC) include an AT40K FPGA for datapath logic, the RISC-based AVR[®] microcontroller for control logic, a hardware multiplier, MCU peripherals and 32K bytes of SRAM – all the digital building blocks of a system. The FPSLIC architecture is ideal for the implementation of networking, telecommunications, multimedia, audio, handheld, portable and industrial control applications..

Atmel can provide the FPSLIC solution because it is in the unique position of owning and producing a high-performance FPGA architecture, and several microcontroller core technologies. Atmel's substantial experience in embedded MCUs in both cell-based and in masked ASIC and Application Specific Standard Product (ASSP) development provides it with the technology and expertise to combine these functions into silicon efficient, cost-effective system level devices.

The AVR-based FPSLIC product is the first in a family of devices incorporating Atmel's AT40K embedded FPGA technology.

The FPGA - The AT40K FPGA core is a fully PCI-compliant, SRAM-based FPGA with distributed 10 ns programmable synchronous/asynchronous, dual-port/single-port SRAM, 68

external global clocks and two AVR processor global clocks, Cache Logic™ ability (partially or fully reconfigurable without loss of data) and 10,000 to 40,000 usable gates.

The Microcontroller - The embedded AVR core executes instructions in a single clock cycle, achieving throughput that approaches 1 MIPS per MHz. The substantial throughput of the AVR allows system architects to optimize power consumption versus processing speed. The 30 MIPS AVR microcontroller core is based on an enhanced RISC architecture that combines a rich instruction set with 32 general-purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughput up to ten times faster than conventional CISC microcontrollers.

The Memory - The AVR executes out of an on-chip SRAM. Both the FPGA configuration memory and AVR instruction code SRAM can be automatically loaded at system power-up using Atmel's in-system programmable AT17 series EEPROM configuration memories. By combining the three main system building blocks on a single programmable device, Atmel has created an a high-performance system-level product that is flexible enough and cost effective enough to be used as a general purpose SLI device.

EDA Tools - FPSLIC EDA tool suite, System Designer™ provides a truly integrated system design methodology and suite of design tools. FPSLIC is the only programmable solution to include co-verification tools as standard. Co-verification of the FPGA hardware and the AVR software facilitates the creation of virtual prototypes, which allows the problems typically encountered during system integration to be resolved much earlier in the design cycle, resulting in a shorter design cycle. Co-verification also allows rapid "what if" trade-offs to be performed, fostering better system efficiencies.

Both the FPSLIC silicon and the System Designer software have been engineered as a complete SLI solution to accelerate time to market.

DESIGNING WITH FPSLIC

By integrating all the functionality required to create a system level product in one programmable solution, FPSLIC devices present the user with a comprehensive and integrated solution. The FPSLIC device closely mimics a typical system-level architecture. It has the common interfaces between microcontroller memory and logic already implemented allowing the designer to focus on the value added parts of the system design without compromising flexibility or performance.

Use of existing standard design tools with the addition of hardware/software co-verification means that reliable bug-free microcontroller and FPGA development software is combined with software-based system-level simulation. The result is an easy to use productivity enhancing development tool (System Designer 2.0) that allow concurrent software and hardware design and dramatically accelerates design development and reduces a products time-to-market.

An AT94K FPSLIC devices implementation of the mail sorting example described earlier would distribute system tasks in the follow way: AT40K FPGA array in the FPSLIC device would pre-process the pixelized data to locate probable address candidates. The FPSLIC's AVR microcontroller and hardware multiplier would take over the image extraction, rotation and de-skewing tasks. Intermediate data would be stored in the on-chip SRAM, allowing maximum system throughput to be maintained.

This combination of memory (SRAM), logic (FPGA) and a microcontroller (AVR microcontroller) on a single SLI device provide for efficient implementation of the data-path (logic) and control flow (AVR microcontroller) aspects of SLI design. Instead of "shoe-horning" a design into a homogeneous FPGA solution the FPSLIC devices provide efficient implementation of all aspects of the system. Silicon efficiency results in smaller die size, fast development times, high-performance designs and lower power consumption. So dramatic is the

effect with Atmel FPSLIC devices that device costs are two orders of magnitude less than competing large FPGA solutions. The usual trade-off associated with silicon efficiency is a lack of flexibility – however the use of a high performance RISC microcontroller with FPGA and a dynamically-allocated SRAM memory provides both efficiency and flexibility.

In FPSLIC devices, the intellectual property required for system level integration is an inherent part of the device. Additional IP blocks can be added to FPGA (logic) part of the design (e.g. from the library of parameterizable macro generators). Unlike large FPGAs, the simulation, placement and timing challenges of integrating soft IP processors have mostly been removed from the FPSLIC parts. This accelerates time to market and allows the designer to focus on the value added aspects of the SLI design. Logic based soft IP cores can be used on the FPSLIC device if required in the logic part of the design. Atmel’s worldwide network of FPSLIC/AT40K FPGA design consultants provides both application and product knowledge on the FPSLIC/AT40K FPGA devices and related IP.

FPSLIC POWER CONSUMPTION

Multiple features in FPSLIC devices cut power consumption dramatically, compared to the FPGA or discrete solution.

More Efficient Use of Logic Resources - By including a “hard” microcontroller core for the implementation of control logic, FPSLIC save, power draining logic resources. A “soft” microcontroller core in a large FPGA requires substantially more logic gates and consume substantially more power.

Integration - Since FPSLIC devices integrate all the required system blocks, FPSLIC eliminates the I/O ports and capacitive loading associated with inter-device PCB connections significantly saving system power.

FPGA Clocking Tree Structure - FPSLIC devices include the FPGA clocking tree structure, which is partitioned into small segments so the FPSLIC device only drives clocks lines to

registers, as required. The clock partitioning can save 50%+ of the dynamic power consumption in the FPGA part of the FPSLIC device on a typical design.

High Throughput MCU - The AVR core in the FPSLIC can achieve 30+ MIPS throughput, allowing it to be used in “burst-mode” processing. Burst-mode processing allows the AVR to perform the processing in very short periods of time and then be put into power-down mode for the majority of the time, saving substantial power.

IN-SYSTEM RECONFIGURABILITY-The combination of FPGA and microcontroller in FPSLIC devices allows partial reconfiguration of the FPGA core. Thus, a single design to be reconfigured to serve several purposes, saves both silicon and power.

This capability is particularly useful for designs which must be able to implement multiple standards, such as “soft radios” being developed for third-generation mobile communications. A single FPSLIC could contain multiple mobile phone (base-band) standards that allow it to operate in any location or environment. In a location where W-CDMA is the standard, the W-CDMA design would be loaded into the FPSLIC device. When the phone user travels to a location where GSM is the standard (Europe for example), the FPSLIC device can be reconfigured on the fly with the GSM design. This would be entirely transparent to the user, but it would allow a single FPSLIC device to be used for many different system level mobile phone standards.

Atmel’s System Designer EDA tool suite supports reconfigurable computing by supporting incremental design changes, extensive library controls and bitstream utilities. Additional reconfigurable computing tools will be available later in 1999.

RECONFIGURABLE PDA APPLICATION-In the near future, personal digital assistants (PDAs), mobile phones, pagers and global positioning by satellite will all be squeezed into a single, hand-held PDA-type device. The high performance, reconfigurability and very low power

consumption of FPSLIC devices make them ideal for these portable applications.

PDA's typically operate in different, dedicated, sequential modes. For example, in one mode the PDA captures the pen input. In another it performs infrared data transfer. In a third mode, it could support modem transfers. By using an in-system programmable SLI device such as FPSLIC, many modes can be supported in a single system-level IC efficiently and with minimum power consumption.

In *pen capture* mode, the FPSLIC's FPGA would scan the screen and process the raw data, while the AVR MCU would handle decisions making and data display. If the PDA user decided that he/she needed to beam data to/from another user, the infrared logic mode (*IrD link*) design would be loaded into the FPGA, replacing the pen capture mode. The reconfigured PDA would beam the data, using the AVR microcontroller to handle data packaging and compression, and use the FPGA to handle CRC checking, the physical layer logic and handshaking. After the transfer is complete, the *IrD link* logic would not be needed and the FPSLIC device might be reconfigured to transfer data received serially to a back-up PC or printer by loading the FPGA with a high-performance UART (the FPSLIC on-chip UART could be also used.) In short, a single piece of FPGA silicon in the FPSLIC device can be re-used many times in different applications in the system. Reconfiguring the FPSLIC FPGA results in a smaller, lower power and more cost-effective solution.

DESIGN TOOLS

Designing, simulating and debugging an ASIC or ASSP design is a daunting task, at best. Designing a very large FPGA is equally as cumbersome. Even when the designer can, with confidence, drop in a soft IP core of a microcontroller, he/she is faced with some serious design tool challenges. Issues of place and route times, design complexity, interactions between IP from different IP vendors and design performance are just some of the immense challenges facing designers of large FPGAs.

Unlike large FPGA design tool suites, which have not changed to accommodate the additional challenges associated with implementing SLI, Atmel has developed the FPSLIC design tools concurrently with the FPSLIC architecture to ensure a seamless development environment between the programmable logic and microcontroller areas of the tool.

Atmel's approach to design tools for the FPSLIC family has been to evolve its established, "field-tested" design tools. The design methodology remains essentially unchanged from the methodology used for a discrete solution with a microcontroller, FPGA and memory. Atmel has modified its standard FPGA design tools to support FPSLIC FPGA cores. The same tools designers have been using to design AT40K FPGAs, such as macro generators, timing driven design, HDLPlanner™, static timing analysis, back annotation, push button APR all work the same way in the FPSLIC System Designer development tools as they do in Atmel's IDS FPGA design tools. The FPSLIC AVR development tools work identically to Atmel's AVR Studio. By using established state-of-the-art software, Atmel has created a comprehensive, verified software solution.

SYSTEM DESIGNER CO-VERIFICATION EDA TOOL SUITE

FPGAs are usually designed using hardware description languages (HDLs), such as Verilog or VHDL and then simulated using an HDL simulator. Microcontroller designs are usually done in the C-language or assembly and debugged using a software debugger or ICE (in-circuit emulator). The challenge Atmel faced was integrating these two solutions into an environment that not only allowed for easy product development and accelerated the designers time-to-market, but also allowed the designer to do extensive "what if" analysis between the hardware and software aspects of the design very early in the design process.

Atmel has solved the problem of software/hardware co-design by developing the System Designer Co-verification EDA tool suite. The need for hardware/software co-verification has grown out of the productivity and time-to-profit obstacles inherent in the conventional design cycle. Up to half of the typical design project is spent in the integration and test phase. In reality this is an exercise in correcting the accumulation of errors from the front end of the design cycle. These errors often reach all the way back to the specification and partitioning phase, where ambiguities in the

hardware/software interface were introduced and then amplified during the hardware/software implementation phase. Often, the remedy of these errors is forced into software due to the long lead times and high cost of ASIC turns – even though a software fix may mean compromised performance or functionality in the final product.

System Designer seamlessly integrates Atmel's FPGA design tools and a third-party hardware (verilog/VHDL) simulator with its AVR microcontroller instruction simulator and debugging tools. In addition, a co-verification framework fully synchronizes hardware and software execution, and source- and assembly-level software debugging. The tool suite provides full visibility of the AVR memory and registers and full hardware design visibility. System Designer allows designers to do the complete hardware and software design with complete confidence. Software/hardware trade-offs can be made and tested until an optimized implementation is arrived at. Design cycles can be cut by as much as 90%. In addition, combining the software development tools with logic simulation, the co-verification environment delivers high-performance co-verification months ahead of a discrete solution. The co-verification environment enables software and hardware development to be parallel activities, removing the software from the critical path, and reducing the risk of hardware prototype iterations resulting from integration errors.

FPGA Design Tools - The FPGA design software in System Designer is based on Atmel's IDS 7.0. It includes Macro Generators, HDLPlanner™ push button automatic place and route, floor planning, timing driven design, both static and interactive timing analysis, bitstream utilities, incremental design change capability, architecture mapping, back annotation, an interactive layout editor and a library manager.

Macro Generators for Re-usable IP Design - System Designer comes with push button macro generators that facilitate the design of fully parameterizable hard or soft intellectual property (IP) cores for the FPSLIC FPGA array. The macro generator calculates power consumption, area and pitch of the macros. All Atmel macros are optimized for the AT40K architecture. More than fifty macro generators are available in System Designer that can be used to create fully parameterized

IP cores of virtually any complexity. The macro generators include: adders, FIFOs, counters, comparators, decoders, flip flops, latches, RAMs, CRC, integer dividers, linear feedback shift registers, (LSRs), fast pre-scale counters, accumulators, deductors, multipliers, muxes, negation, shifters ROMs, subtractors and tri-state bus control. These functions can be parameterized for word-width, power or area and trade-offs can be made easily.

System Designer's macro generators are invoked from pull-down menus so the designer need only point and click to create the desired functionality. Once the macro generator is invoked, a dialog box lets the designer specify any parameters that are appropriate to the macro.

Designers can also download details of complex IP cores, such as FIR filters, IIR filters, convolvers and other functions that are available from Atmel's growing library of FPGA intellectual property from Atmel's web site.

HDLPlanner™ - Atmel's HDLPlanner tool automates the development of FPGA VHDL or Verilog descriptions by automatically generating syntactically correct Verilog or VHDL code. HDLPlanner can generate HDL descriptions from macros developed with System Designer's macro generator tools. Any design done using HDLPlanner is completely device and technology independent and can be synthesized, using industry standard synthesis-tools, for implementation in any ASIC or FPGA. HDLPlanner automatically instantiates components that are optimized for Atmel FPGAs. The instantiated components are completely transparent to the user. Although they are architecturally optimized for the AT40K FPGA logic, instantiated components have no affect on the technology independence of the HDL designs.

FPSLIC Firmware Design and Debugging - Atmel's AVR software design environment is an integral part of the System Designer EDA Tool Suite. It enables the development, execution and debugging of AVR programs using a built-in instruction set simulator.

AVR Studio provides a “Source” window with the program code and a pointer that marks the code currently being executed. It has a variety of views that assist in debugging the design. These include windows that:

- o display the values of defined symbols, including instance variables in a C-program;
- o display the contents of all 32 of the AVR's registers;
- o report messages issued by AVR Studio;
- o allow the user to view and modify the contents of all AVR memory resources;
- o show the address of the next instruction to be executed, the value of the stack pointer, and the number of clock cycles that have elapsed since the last reset;
- o show the status of peripheral devices;
- o display information about any AVR timer/counters;
- o show the three I/O registers on each of the AVR ports;
- o show the status of on-chip peripherals (UARTs, SPI).

Co-verification Ensures Speedy Design Cycle - FPSLIC Co-verification routines allow the HDL simulator and the AVR instruction set simulation to run simultaneously and interactively. Since the hardware and software are designed and debugged together, the likelihood of getting a design that works the first time is greatly increased. Total system development time can be cut by 10% to 90%. The FPSLIC System Designer software suite is unique in its completeness and ability to accelerate time to market.

CONCLUSION

With the introduction of the FPSLIC family of programmable system level devices, Atmel has developed a Field Programmable System Level Integrated Circuit solution that is unmatched in the industry in either hardware or software. Atmel’s FPSLIC family of programmable system-level ICs is the only device family on the market today that combines all the building blocks required for true programmable system-level integration—a high-performance reconfigurable

FPGA, an ultra high-performance RISC MCU with a hardware multiplier, and a substantial block of SRAM.

The FPSLIC's EDA environment is based on a systems approach to design methodology, including co-verification of the hardware and software.

Atmel is uniquely positioned to offer a complete programmable SLI solution because of its extensive experience and intellectual property in the areas of programmable logic, microcontrollers, programmable logic software development and system level integration. Other silicon solution providers today do not have the ability to combine these essential technologies to provide a true programmable SLI solution with the features required to be successful in the new programmable SLI marketplace.

© Atmel Corporation 1999. All rights reserved. Atmel, the Atmel logo, and combinations thereof are trademarks of Atmel Corporation. AVR, Field Programmable System Level IC, FPSLIC and the FPSLIC logo are a trademark of Atmel Corporation. Terms and product names in this document may be trademarks of others.