Implementing Cache Logic® with FPGAs

The Cache Logic Concept

Atmel Corporation has developed an enabling technology to make adaptive hardware possible for electronics systems. This capability, trademarked as *Cache Logic*, was developed and patented by Atmel Corporation.(1)

Cache Logic is a cost-saving way of implementing logic more efficiently. The active functions of an application are performed by a field programmable gate array (FPGA) that can be reconfigured as it operates, while inactive functions are stored in an inexpensive configuration memory – an EPROM, for example. As new functions are required, they are written over old ones.

A single application is made up of many smaller macro-level operations, like counters, multipliers, shift registers, and multiplexers. When an application is broken down into its sub-operations, two things become apparent. First, functionality overlaps. A single function may be used a number of different times. Second, there is a high degree of functional latency. At any given moment, only a small portion of an application's operations are active; only a few functions are used at the same time.

By consolidating functionality, eliminating redundancy, and tracking the occurrence of each sub-operation, functions can be organized such that a relatively small, inexpensive logic device is reconfigured as it operates to perform a complex function. In a 10,000-gate application, for example, only 2,000 gates might be active at once. By caching the extra 8,000 gates for later use, a 2,000-gate device replaces a more expensive 10,000-gate device.

Cache Logic Implementation

Cache Logic implementation is conceptually similar to cache memory. In cache memory, the highest speed memory (usually SRAM) is used to store active data, while the bulk of data resides in lower-cost storage, such as DRAM, or EPROM, disk, etc. Cache Logic works in a similar fashion. Only a small fraction of the circuitry – those functions which are loaded into the logic cache – is active in a system at any given time, while unused functions or variations reside in lower-cost system memory. It is even possible to compile variations of a design in real time. As logic functions are required, they can be loaded into cache logic, replacing or complementing the logic already present.

Figure 1 shows the block diagram for the Atmel AT6000 FPGA, which is an ideal medium for cache logic. The ability to implement cache logic requires FPGAs that are capable of being dynamically reconfigured in-system, either completely or partially, without disrupting the operation of the balance of logic in the device. Another requirement is architecture symmetry. This is necessary to make possible the arbitrary placement of generic blocks in a location that is available at the time required. It is also

Note: 1. The method for exploiting Cache Logic was pioneered by the University of Strathclyde in Scotland and is described in Lysaght, P. and Dunlop, J., "Dynamic Reconfiguration of Field Programmable Gate Arrays", in *More FPGAs*, W. Moore and W. Luk, Eds., Abingdon EE&CS Books, England 1994.

Field Programmable Gate Array

Application Note

Rev. 0461C–09/99

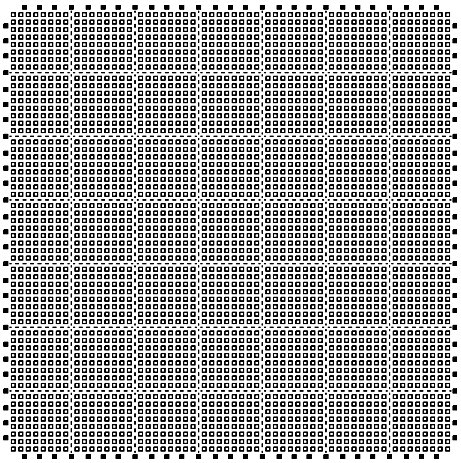
necessary to allow for easy modeling of device characteristics for the artificial intelligence required in the partitioning of a design. The symmetry also simplifies the creation of arrays of devices to create a larger digital medium for the implementation of cache logic.

Predetermined and Dynamic Cache Logic

There are two types of cache logic which have been defined: *predetermined* cache logic and *dynamic* cache logic. *Predetermined* cache logic involves the use of predefined functions and macros that are stored in external, nonvolatile memory (EPROM, EEPROM, disk, CD-ROM, or even memory remote from the system loaded over a communications link). These functions have already been placed and routed and have bit streams which have been previously generated (Figure 2). The implementation of these functions is controlled by a resident manager in the logic cache, or in an external control such as a microcontroller/processor routine. New functions may be downloaded to the logic cache in the background without disrupting the operation of the cache (logic, I/O, and register data), as shown in Figure 3. In fact, data in the registers is not lost even in the area being overwritten.

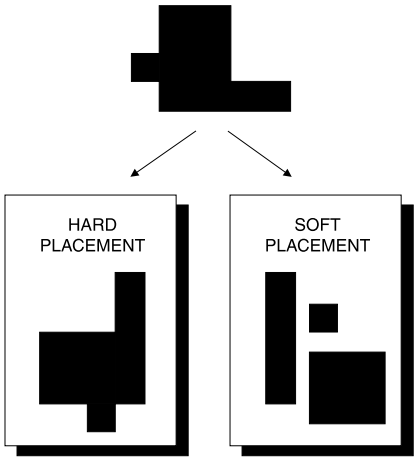
The second type of cache logic, *dynamic*, is the basis for building adaptive hardware. Dynamic caching involves the determination of logic, placement and routing of the logic, bit stream generation, and programming the logic cache in real time. The major issues to be addressed in the development of this capability include (but are not limited to) the scheduling and allocation of functions, random-logic collection, and collision handling and avoidance within the cache. Dynamic cache logic exists as a concept today; the physical implementation issues described above have not yet been fully addressed.

Figure 1. AT6000 Array



- Symmetrical Array
- Identical Cells
- 8-by-8 Cell Sectors
- Programmable Interconnects
- Surrounded by I/O
- No Dedicated Functions
- Reconfigurable On-the-fly
- Full
- Partial
- Without Data Loss

Figure 2. Macro Library



- Over 200 Hard Macros
- Fast
- Fully Specified
- Fixed Routing
- All Can be Softened
- Flexible Placement
- User-defined Macros
- Create Own Library
- Use on Future Designs
- Test Macros
- For Debug/System Test
- Super Macros
- Major Predefined Functions
- Specialized for Markets

Cache Logic may be applied in many applications. The concept of *virtual products* will be introduced, which utilizes the flexibility of programmable logic. Virtual products do not require cache logic programmability but, as we see, the use of cache logic greatly reduces the amount of programmable digital media needed to implement a virtual product.

Virtual Products

A *virtual product* is a combination of a "tangible asset," such as a data acquisition board, and a service, such as product customization. The first thing to understand about virtual products is what the end customer wants, and how system developers can match their core competencies with these needs.

There are two issues raised in the manufacture of virtual products:

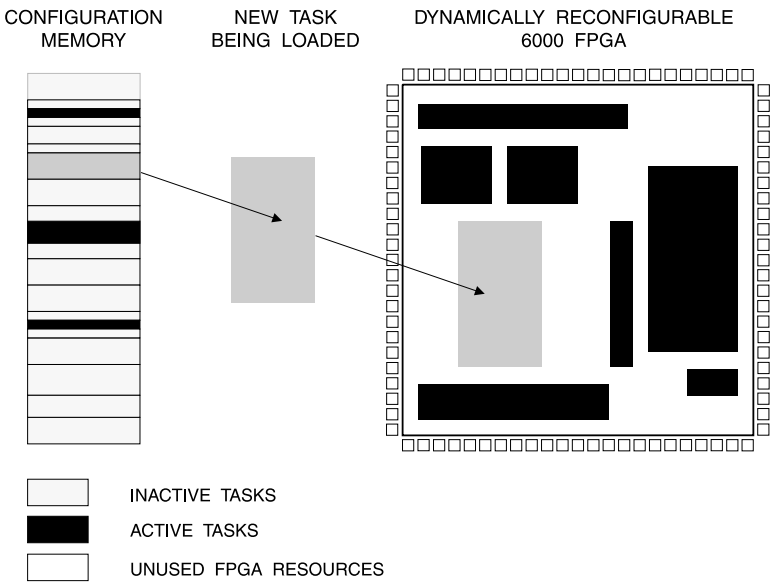
1. How to balance economy of scale achieved in volume manufacturing with special features that customers are willing to pay for; and
2. How to create diversity while maintaining a level of quality associated with standard high-volume production.

Cache Logic and FPGAs help the manufacturers achieve these two requirements of virtual products. A virtual product line is one with characteristics which meet the needs of a class of customers. An example would be a PC-based data-acquisition product. Such a product has certain physical requirements consistent with a PC-bus card standard. The board would also have a series of standard data gathering features such as multiple-channel A-to-D converters, digital I/O ports, D-to-A converters, and high-speed clock counters. These features are typically accomplished by highly integrated well-designed ICs readily available to all manufactures. The complexity of such products is in the data path and protocol which connects the PC to the standard IC products. The structure of this data path is prejudiced by optimum system performance, cost, and customer preference, the key item being customer preference for a successful virtual product, or for that matter any successful product.

The traditional approach to creating a data-acquisition product, like most products, is to create a board with a standard bus footprint, use industry standard A-to-D and D-to-A circuits, and then create a custom data path. The manufacturer then has to trust marketing studies and instinct to determine the best data path approach. It is possible to hedge the bet by adding redundancy. This redundancy has two detrimental effects: added cost, and added complexity for the end user. The selection of wrong data path protocol or excessive complexity caused by redundancy results in dissatisfied or nonexistent customers.

A virtual product does not mean that a manufacturer would be able to offer one product which was all things to all people. The use of programmable logic would allow a manufacturer to create an extensive catalog of products, but only have a small number of tangible assemblies to tool for manufacturing. The manufacturer would use FPGAs and cache logic FPGAs to create diversity in its product line. The cost of diversity to the manufacturer is the cost of service, or "personalization engineering", required to create a niche design on a standard assembly. The advantage for the customer is a mass-produced product which meets their specific needs.

Figure 3. Cache Logic Concept



The virtual product approach allows a manufacturer to perfect a single assembly. The FPGA's ability to be configured for self-test could even enhance the quality of the assembly. Atmel has developed the IEEE1148 boundary-scan supermacro. Utilizing the reconfigurable logic capability of the AT6000 family, the boundary-scan function may be loaded into the device and diagnostics performed, then the device can be reconfigured for other logic functions. A single Atmel device may be used for testing and logic, with no overhead or speed penalty, as is the case for all other FPGAs and other ASIC devices.

The result would be an inventory of nearly identical raw-product assemblies, which through virtual design becomes a catalog full of products when shipped to the customer. With a solid design, most customer problems can be traced to the virtual-design personalization process, and be repaired in the field with FPGA configuration updates. It is also possible to introduce new features into virtual products as soon as they are invented and proven, rather than wait until a new hardware product is designed, tooled, and manufactured.

Cache Logic Benefits

There are several benefits derived from cache logic design:

- New functionality may be added to existing hardware, without having to make modifications to the board.
- The hardware may be tailored to the application, resulting in higher system performance across a broad range of applications.
- The FPGA density limitations are eliminated.
- Overall system reliability is improved by reducing the number of physical products manufactured and utilizing boundary scan macros for manufacturing and system testing.

Overall product life cycle costs are significantly reduced by using reusable software and hardware:

- Lower development costs
- Lower inventory costs
- Quicker time to market
- Fewer parts on the board
- Lower power consumption
- Lower total system cost
- Reusable designs

Summary

To many people, the ideal of adaptive hardware or virtual products is a futuristic concept. The AT6000 family is capable of implementing cache logic and virtual products today. The Atmel FPGA and its abilities to implement cache logic make it a foundation for adaptive hardware and virtual products. Successful design with this new technology has been commercially demonstrated. Today's design methodology, that requires a new product for each new function, will be replaced by adaptive hardware products that meet the needs of both customers and suppliers with customized products and improved quality, while reducing product development time and overall life cycle costs.

Examples of Cache Logic Applications

Power and Space-sensitive Applications	**Compute Intensive Applications**
Portable Computers Battery-Operated Instrumentation Portable Communications Portable Medical Equipment	Computer Graphics Image Processing Data Compression Speech Recognition Pattern Recognition
Reprogrammable Hardware	**Application Acceleration**
Test Equipment Industrial Control Instrumentation Special-purpose Computers Connectors	CAD Database Spreadsheet Multimedia

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

0461C–09/99/xM