A Systematic Approach to Optimizing and Verifying Synthesized High-Speed ASICs

Trevor C. Landon, Maximo H. Salinas, Robert H. Klenke, James H. Aylor, Sally A. McKee, Kenneth L. Wright

Computer Science Report No. CS-95-51
December 11, 1995

This work was supported in part by a grant from Intel Supercomputer Division and by NSF grants MIP-9114110 and MIP-9307626.

A Systematic Approach to Optimizing and Verifying Synthesized High-Speed ASICs

T.C. Landon, M.H. Salinas, RH. Klenke, J.H. Aylor, S.A. McKee, K.L. Wright
Center for Semicustom Integrated Systems
Department of Electrical Engineering, University of Virginia
Charlottesville, Virginia 22903
(804) 924-6101, (804) 924-8818 (fax)
{landont | msalinas | klenke | jha | mckee | wright}@virginia.edu

Abstract -- This paper describes the design process used in developing a Stream Memory Controller (SMC)*. The SMC can reorder processor-memory accesses dynamically to increase the effective memory bandwidth for vector operations. A 132-pin ASIC was implemented in static CMOS using a 0.75µm process and has been tested at 36MHz.

I. Introduction

One of the promises of hardware synthesis is providing logic designers with little or no experience in ASIC design a means of designing complex ASICs successfully. Often, the design and fabrication environment are not ideally matched to that described by tool vendors, however, and discrepancies must be overcome by creative solutions in order to allow tools to deliver on their promise. In this paper, we describe the methodology we designed and put into practice which enabled us to design and fabricated a 71,000 transistor ASIC using graduate students to do the actual logic design.

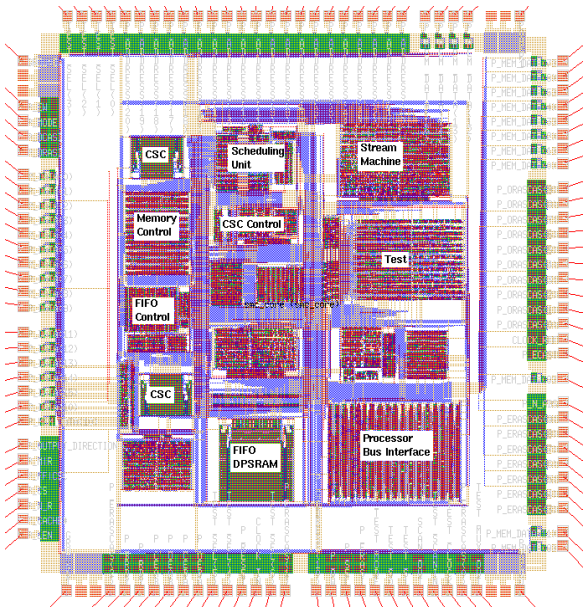

Figure1: SMC IC

The ASIC discussed in this paper was produced in the second phase of a multi-year project. The complexity of the second phase IC required that careful attention be paid to the design and revision process. As new functionality was added to the IC, a number of critical circuit paths were introduced that needed to be modified to meet timing requirements. Relying on synthesis alone proved inadequate for meeting our ambitious timing specifications [1], [2], [3]. In order to reduce design time and expense and to increase our confidence in the functionality of the completed ASIC, we developed a systematic approach to synthesized-circuit optimization. This paper discusses the methodology used to design and verify the ASIC and to optimize it to meet the timing requirements of the Intel i860 host processor [4].

*** This work was sponsored by the National Science Foundation under Grant MIP-9307626.**

II. SMC Architecture Overview

Processor speeds are increasing much faster than memory speeds. While microprocessor performance has improved steadily at a rate of 50-100% per year over the past decade, DRAM performance has increased at an annual rate of less than 10%. Not only is the current problem serious, but the difference in performance is growing at an exponential rate, and caching alone cannot bridge the processor-memory performance gap. Streaming computations -- including vector processing, multi-media (de)compression, encryption, signal processing, image processing, text searching, some database queries, some graphics applications, and DNA sequence matching -- suffer particularly acutely.

Most modern memory devices provide special capabilities that make it possible to perform some access sequences faster than others, e.g. page-mode DRAMs [5]. These capabilities can be exploited via *access ordering* to improve effective memory bandwidth for streaming computations. Our team is developing a combined hardware/software scheme for implementing access ordering dynamically at run-time. The hardware component of this approach is the Stream Memory Controller (SMC), a 132-pin ASIC implemented in a 0.75 µm process (see Figure 1).

The SMC provides a set of FIFOs which can be set up to buffer data between the processor and system memory for constant stride sequential read or write accesses. By ordering memory accesses appropriately, the SMC memory controller minimizes the number of DRAM page misses incurred as FIFOs are serviced.

The architectural components of the SMC IC are shown in Figure 2. The FIFOs are implemented using a dual-ported SRAM and a FIFO controller state machine which generates the address for both Bank Controller and processor accesses to the FIFOs. The Command, Status and Control (CSC) register is

also implemented using a dual ported SRAM and is used to store the base, length, and stride information for each stream. The Processor Bus Interface (PBI) state machine provides the logic necessary to interface the SMC with the i860 processor bus.

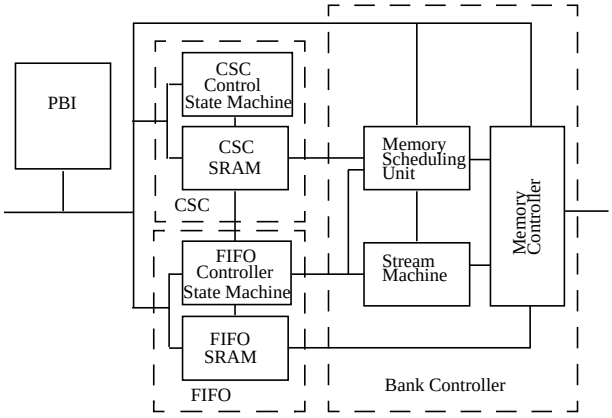

Figure2: SMC Architecture

The Bank Controller logic handles the interleaved memory system interface and fills or drains the FIFOs as required. The Bank Controller also provides support for scalar accesses to the SMC memory space. The FIFOs buffer data between the processor bus and the memory system bus and can be accessed by both simultaneously. The present version of the SMC contains fixed depth FIFOs, but future versions are planned with a software programmable depth (deeper is not always better as discussed in [6]. The FIFO controller logic provides full, near full, and empty signals to both the Bank Controllers and the PBI for each FIFO. These signals allow the PBI to determine if a given access can be serviced, and the Bank Controllers to determine if a FIFO needs servicing by the memory system [7].

In the current implementation, only one FIFO may be serviced at a time, and the Bank Controller determines which FIFO will be serviced next. Eligible FIFOs are selected in round-robin fashion; eligible FIFOs are defined as read FIFOs that are not full and write FIFOs that are not empty. When a FIFO is selected, the Bank Controller services the FIFO until one of the following three conditions occurs:

- the FIFO is full or empty, for a read or a write FIFO, respectively
- there is a page miss on a required DRAM access
- the stream operation on the FIFO is complete

Architectural level simulations indicated using a round-robin selection algorithm along with the above three conditions to determine the FIFO's service interval approached optimal performance.

III. Overall Project Design Process

The primary logic designers for the SMC IC were graduate students with little prior experience in ASIC design. Extensive virtual prototyping of the SMC was used to verify its functionality and to complete the design quickly. In addition, hardware synthesis was used for the majority of the design. Furthermore, an overall SMC project design process was devised in which ICs are designed and fabricated in phases in order to reduce the risk of fabricating faulty ICs.

The overall project development process is shown in Figure 3. The first phase chip is described in [7] and was used to verify that the synthesis and silicon compilation tools available can produce a working design at the required speeds using fabrication processes available through MOSIS. The first

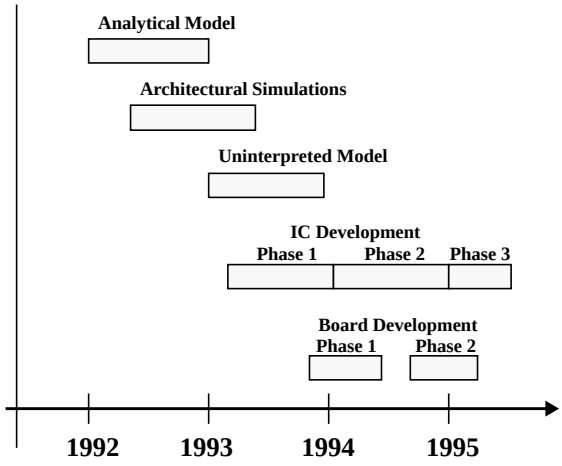

Figure 3: SMC Project Timeline

phase IC implemented the PBI, the CSC logic, FIFO controller and SRAM. In order to reduce costs, the IC was bit sliced so that each IC provides 16 of the 64 data bits to the processor. Since a full SMC for a 64-bit system with 2-way interleaved memory requires 198 pins for the data lines alone. The IC pin count approaches 300 when the control and power pins are added.

In the second (and current) phase of the SMC, the modules required to interface to the DRAM memory susbsystem and to generate stream accesses into this memory were added. In particular, the Stream Machine, the Memory Controller, and the Memory Scheduling Unit were included. The majority of the modules from the first phase IC were reused in the second phase IC with little modification. The reused modules were only modified where it was needed to improve the timing of some critical circuit paths.

The third phase of the IC will be include a redesign of the SMC FIFO architecture and provide software-programmable depth FIFOs to allow applications to fine tune SMC performance as needed.

IV. Design Methodology

An overall view of the design methodology used in this project is shown in figure 4. The various paths shown can be described by their primary function in the design cycle: functional simulation, gate-level simulation, static timing analysis, and back-annotated timing simulation.

Functional simulation consists of the path through Mentor Graphic Corporation's (MGC's) Design Architect into MGC's QuickSim II and back into MGC's Design Architect. This design cycle verified the operation of the ASIC against its specification and demonstrated that its expected performance agreed with that of previous bus-level software simulations. For these simulations, the state machines were described in functional VHDL, and the datapath components were modeled

at the gate level with unit timing delays. This simulation model was used for initial development because changes to any module could be incorporated and tested quickly.

When the functional model was believed to be correct, hardware synthesis was performed on the state machine VHDL using Cascade Design Automation's (CDA's) EPOCH Circuit Design System tool. The resulting description was used to verify that the synthesized versions of the behavioral VHDL components (i.e. the various controller state machines) operated correctly.

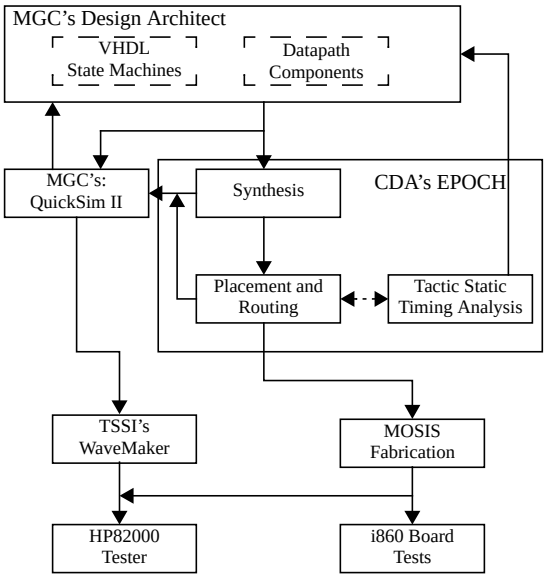

Figure 4: IC Design Process

The majority of our high-speed optimizations were made to address potential problems indicated by static timing analysis tools and by simulations including back-annotated timing information. The static timing analysis design cycle used the simulation models generated in the EPOCH place and route stage to characterize worst-case output delays, and identify critical paths for individual modules in the system. This cycle allowed the designer to determine exactly how each component in the system affected critical path timings. Unfortunately, many of the paths identified as critical by static timing analysis were not actually valid paths in that SMC operations would not require those paths to be traversed. However, static timing analysis does provide a rapid analysis of paths known to be critical and allows adjustments to be made and verified on the state machine or datapath descriptions.

The back-annotated simulation models were created by attaching delays calculated by EPOCH to the gate-level simulation model used by QuickSim II. These delays were based on capacitive loading and routing of the synthesized IC. The back-annotated simulation model was exercised using the same test patterns as the functional model in order to identify timing paths which could be critical in actual SMC operations.

The different simulation models required us to establish a revision process which would facilitate incorporating each design change with little disturbance to the rest of the IC. Our protocol consisted of first testing any new changes in the functional model to insure that they did not hinder the operation of the original system, or drastically effect system performance. The amended component was then synthesized on its own and incorporated into the gate level model. Once the gate level model was verified, all the components of the IC would be placed and routed anew and timing analysis would be performed again.

V. High-Speed Optimization Techniques

When critical paths were located, a number of techniques were used to shorten their total delays in our IC. The first technique was used in critical paths observed in unbalanced pipeline stages. These paths were typically found through the full timing simulations. In this situation, one stage in the pipeline would complete its required operation in a shorter time then that of a neighboring stage. These timing path problems were sometimes resolved by simply moving the registers which separated the stage and modifying the associated control circuitry. Although generally difficult to do, it was sometimes necessary to reorder pipeline stages to accommodate late arrival of control signals. Similarly, extra states were sometimes added to break critical paths, but this had the potential of affecting SMC performance by adding extra cycles to its control mechanism.

Static timing analysis using CDA's Tactic tool indicated other critical timing paths. In cases where a signal from a state machine was on a critical path, tailoring the VHDL input code to the synthesis tool allowed us to improve state-machine performance. One of the techniques frequently used was to modify the tool's state assignment with hand-optimized encodings so that some of the state bits could be used as Moore outputs. Our encodings improved performance by either decreasing the signal output delay for all signals, or by shortening the path for a particular critical signal. To improve overall performance various different state encoding were synthesized and analyzed to determine a near optimal encoding for the machine.

Timing optimizations could also be made by EPOCH by collapsing boundaries between functional blocks in order to perform logic minimization over larger portions of the circuit at any one time. This was particularly useful in cases where a critical net consisted of components in multiple functional blocks.

VI. Design for Testability

In order debug any problems which may arise in the fabricated IC, two testing mechanisms were incorporated into its design. First, a large multiplexer was used to observe any of 128 internal signals. The signal to be observed was selected by the value placed on seven external pins on the IC. This provided an easy way to observe a single internal signal during operation. However, because we had to avoid adding extra capacitance to critical paths, critical timing signals could not be directly observed through the test mux.

The second testing mechanism in the IC is a partial scan path. This test mechanism was used primarily on the state machines and provided a means of observing and/or modifying the state of any of the state machines in the IC. Because the testing circuitry consisted of hand-placed datapath elements, we had to encode our state assignments manually, even in cases where timing analysis did not force us to do so. This was because the VHDL enumerated data type used for automatic state

assignment encoding could not be directly connected to the bit-vector data inputs of the datapath elements. For various reasons, we were unable to use automated test circuitry insertion tools.

VII. IC Testing

The lower four blocks on figure 4. are primarily concerned with testing of the fabricated IC. The Test System Strategies, Inc.'s (TSSI's) WaveMaker tool stage [8] on the diagram converts the test suite used in the QuickSim II modeling of the design to a format that can be used on the HP82000 IC tester. This stage was intended and used primarily for functional verification of the IC.

After minimal functionality was verified with the HP82000, further testing proceeded in parallel on a custom SMC Daughter Card illustrated in Figure 5. SMC board tests were stored in and

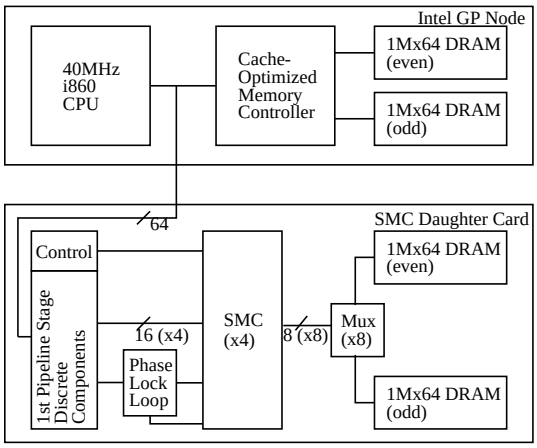

Figure 5: SMC Test Board

executed from the i860 host processor's boot PROM. This stage validated the interface between the Intel i860 and the SMC. There was some concern over this interface since the simulation models for the Intel i860 which were used during the design of this system only simulated external i860 bus transactions. Many bus cycles observed on the board, e.g. code fetches and cache fills, had not been included in our simulations. In addition the board testing provided further proof that the SMC concept can realize the effective bandwidth gains expected.

VIII. CAD Tool Considerations

Several issues arose due to limitations in the available CAD tools. Our concerns fall into two broad categories: inconvenience in procedures and shortcomings in functionality. First, it must be pointed out that a driving concern in our selection of the tools used was the ability to design in a fast IC process available through MOSIS. Specifically, MOSIS offers a 0.75µm HP process. Secondly, we already had experience with and access to the full MGC tools suite. However, since no ASIC design kit was/is available for this process for MGC's AutoLogic synthesis tools, we decided to use CDA's EPOCH as a somewhat more limited but suitable alternative. Although CDA is an open-door partner to MGC and does provide a link to their tools, the interface is somewhat crude and could be improved. For example, there is a lengthy stage which must be completed before using any of the MGC simulation tools after EPOCH's processing. This procedure converts the simulation models from the native CDA format to the MGC format. In our experience, this stage requires two to three times the computing resources as the actual synthesis, placement and routing stages combined. We found that this step was very sensitive to network traffic and by storing copies of the design on a local disk, it could be reduced by two-thirds, down to an acceptable two hours. However, this performance increase limited the design process to a single workstation at any one time.

Regarding the shortcomings in the CAD tools, we ran into two problems with the simulation models from Synopsys' Logic Modeling Group [9]. The first concerned the previously mentioned processor bus model. A useful addition to this model would be simulated cache line fills during instruction fetches. This would have allowed us to debug a potentially critical situation on the IC before fabrication. The other problem which arose centered around a zero nanosecond setup time on one of the control signals to the DRAM. After observing some problems during board testing, we verified in simulation, that in the worst case, this signal barely makes the setup time. However, on the on the test board the signal would sometimes miss the required setup time. In order to help avoid this situations where the timing models used in IC and board design are not perfectly accurate a user-defined "uncertainty region" which would control a minimum and maximum range for all timing values would be useful.

IX. Conclusions

A 71,000 transistor ASIC has been designed and fabricated, and is currently being tested and used to verify expected SMC performance gains. Our results indicate that the fabricated SMC can deliver the expected bandwidth improvements for inner loops of important streaming computations [6], [10], [11]. Our need to use graduate students, our experience and access to MGC tools, and the necessity to use a particular IC fabrication process (0.75µm HP through MOSIS) forced us to use tools that were not tightly integrated. This led to the development of the design and revision process described here.

References

[1] *EPOCH User's Manual*, Cascade Design Automation, EUM-3.1, 1993.

[2] *Cascade Delay Calculation Manual*, Document No. 93-0071-Rev 2, Cascade Design Automation, May 12, 1994.

[3] *System-1076, QuickSim II User's Manual, Mentor Graphics Corp, 1993.*

[4] i860 *Hardware Reference Manual*, Intel Corporation, 1993.

[5] Quinnell, R., "High-speed DRAMs", EDN, May 23, 1991.

[6] McKee, S.A., Wulf, Wm.A., Landon, T.C., "Bounds on Memory Bandwidth in Streamed Computations", Lecture Notes in Computer Science 966 (Proc. Europar'95, Stockholm, Sweden, August, 1995), Springer-Verlag, 1995.

[7] McGee, S.W., Klenke, R.H., Aylor, J.H., Schwab, A.J., "Design of a Processor Bus Interface for the Stream Memory Controller", ASIC'94, Rochester, NY, September, 1994.

[8] *TDS WaveMaker User's Guide*, Test System Strategies, Inc., 1991.

[9] *Smartmodel Library Reference Manual*, Logic Modelling Corp, 1992.

[10] McKee, S.A., Klenke, R.H., Schwab, A.J., Wulf, Wm.A., Moyer, S.A., Hitchcock, C., Aylor, J.H., "Experimental Implementation of Dynamic Access Ordering", Proc. HICSS-27, Maui, HI, January 1994.

[11] McKee, S.A., Moyer, S.A., Wulf, Wm.A., Hitchcock, C., "Increasing Memory Bandwidth for Vector Computations", Lecture Notes in Computer Science 782 (Proc. PLSA, Zurich, Switzerland, March 1994), Springer Verlag, 1994.