Custom circuit design as a driver of microprocessor performance

by D. H. Allen
S. H. Dhong
H. P. Hofstee
J. Leenstra
K. J. Nowka
D. L. Stasiak
D. F. Wendel

This paper presents a survey of some of the most aggressive custom designs for CMOS processor products and prototypes in IBM. We argue that microprocessor performance growth, which has traditionally been driven primarily by CMOS technology and microarchitectural improvements, can receive a substantial contribution from improvements in circuit design and physical organization. We predict that in future microprocessor designs the floorplan and wire plan will be as important as the microarchitecture, more control logic will be structured and become indistinguishable from dataflow elements, and more circuits will be designed and analyzed at the level of single transistors and wires.

1. Introduction

The traditional recipe for improving microprocessor performance, measured in instructions per cycle divided by cycle time, relies on improving both the frequency and the amount of useful computation per cycle. The total amount of work per cycle is increased by operating on multiple instructions in parallel and by avoiding stall conditions through speculation and out-of-order processing. Two main mechanisms are traditionally used to improve frequency—a contribution from CMOS technology, and adding stages to the microprocessor pipeline to reduce the amount of work per stage or, equivalently, per cycle. Unlike most microarchitectural mechanisms that aim to increase the amount of work per cycle, improving the frequency has an easily predictable benefit to overall performance and therefore resonates most in the marketplace.

Figure 1 shows an overview of the maximum frequency times CMOS lithography dimension for processors shipped in a number of recent 64-bit server systems. Since transistor delays are (to first order) proportional to channel length, and channel length is roughly proportional to the general lithography resolution, the graph provides (to first order) a technology-invariant measure of frequency. Except for the Alpha® processors, which set the standard for high-frequency processor design [1, 2] but have recently focused on CPI rather than processor frequency, the graph shows most 64-bit architectures improving steadily in frequency, even when the contribution from technology is factored out. It should be noted that technological improvements beyond lithography scaling, such as copper interconnects (IBM 0.22- and 0.18-μm technologies), and silicon-on-insulator, or SOI (used in the 0.22-μm 64-bit PowerPC*) have not been compensated for in this graph.

Technology advances have driven much of the performance improvement in commercial microprocessors.

©**Copyright** 2000 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

0018-8646/00/$5.00 © 2000 IBM

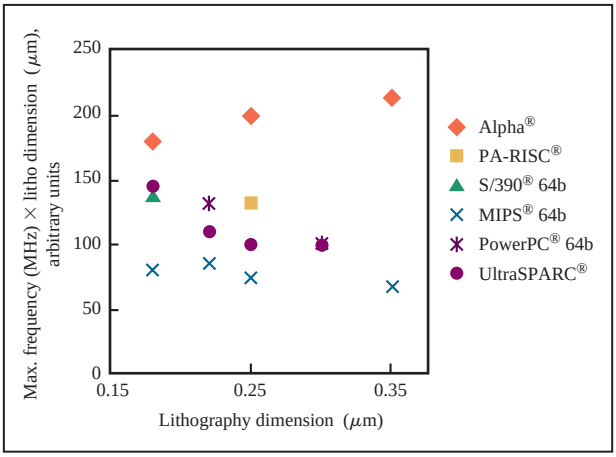


Figure 1

Technology-normalized frequency of 64-bit processors.

Table 1 presents the recent IBM CMOS technologies. With every generation, chip cycle time, as measured with a more accurate technology-invariant metric such as the number of fixed-fanout inverter delays in a cycle, has continued to decrease (**Figure 2**), and for the most aggressive designs now approaches that of 14 fanout-4 balanced inverters. With a constant clocking and latching overhead per cycle, and ever-increasing complexity to efficiently operate the longer pipelines, it appears that superpipelining much beyond today's pipeline depths of approximately 20 stages provides little additional performance. Also, the improvements in transistor performance cannot continue forever. Increasing power dissipation and increasing transistor leakage currents may be an early indicator of a maturing CMOS technology. If we add to this the fact that the price for developing a new microarchitecture (let alone a new industry-standard architecture) and the price for developing new technology are so high that innovation may become limited by economics, it becomes clear that pursuing techniques which promise to increase frequency for a given technology without changing pipeline depth may be necessary in order to maintain the current rate of performance growth.

This paper focuses on physical design as a different mechanism for improving frequency. Scaled wire delays are made to track scaled FET delays, but cross-chip delays do not scale and are now of the same magnitude as cycle times; hence, transfer cycles are more and more common in processor microarchitectures. Also, with the growing number of wiring levels available to the designer, the disparity in the characteristics of the different layers increases steadily. This implies that the performance of a processor microarchitecture cannot be evaluated without an accompanying floorplan and wire-level plan, as well as the traditional analysis of the delays of the various components.

When cycle times approach the delay of 14 fanout-4 inverters, clearly every transistor counts. Designing control at the gate level or above is no longer adequate. The entire processor must be designed at the level of transistors and wires. Design budgets for both control and dataflow must include wire delays up front. Array-like and dynamic circuit topologies that do more useful work per unit of delay are preferred. Eliminating just a single transistor from a critical path has a measurable effect on overall performance.

The paper is organized as follows. Section 2 discusses the state of the art in transistor and wire-level design methodology within IBM. Sections 3, 4, and 6 present a number of case studies that show what can be achieved with careful physical design. Section 3 discusses the use of dynamic circuits in recent commercial PowerPC processors. It shows a progression of adder designs as an example of increasing sophistication in physical design. Section 4 discusses two short-pipe PowerPC prototype processors realized entirely in self-resetting and delayed-reset dynamic circuits. Control is realized predominantly in array-like structures. In Section 5 we compare the data cache access paths of a commercial PowerPC processor and one of the PowerPC prototypes in the same

Table 1 CMOS process technology characteristics.

Technology	*Material*	*L(drawn)* (μm)	*L(effective)* (μm)	T_{ox} (nm)	*Wiring layers (excluding local interconnect)*	*Metal*	*M1 contacted pitch* (μm)	*M2–M4 contacted pitch* (μm)	*Supply voltage* (V)
CMOS 6S	bulk Si	0.25	0.18	5	6	Al	0.98	1.26	2.5
CMOS 6X	bulk Si	0.25	0.18	5	6	Al	0.98	1.26	1.8
CMOS 7S	bulk Si	0.22	0.12	3.5	6	Cu	0.63	0.81	1.8
CMOS 7S SOI	SOI	0.22	0.12	3.5	6	Cu	0.63	0.81	1.8
CMOS 8S	bulk Si	0.15	0.09	2.7	7	Cu	0.5	0.63	1.5
CMOS 8S2 SOI	SOI	0.15	0.08	2.3	7	Cu	0.5	0.63	1.5

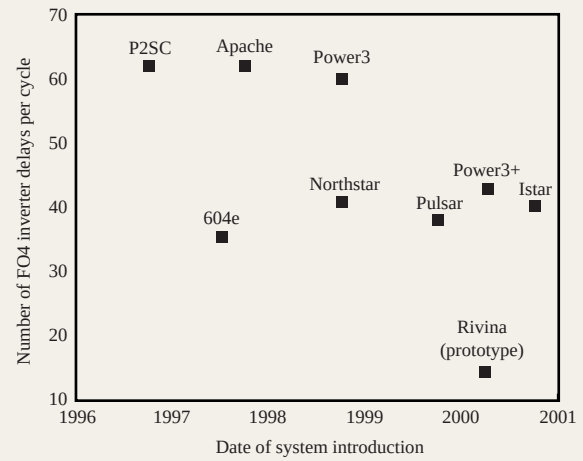


Figure 2

Technology-normalized delay of IBM server and workstation processors (courtesy M. A. Johnson, IBM Server Group).

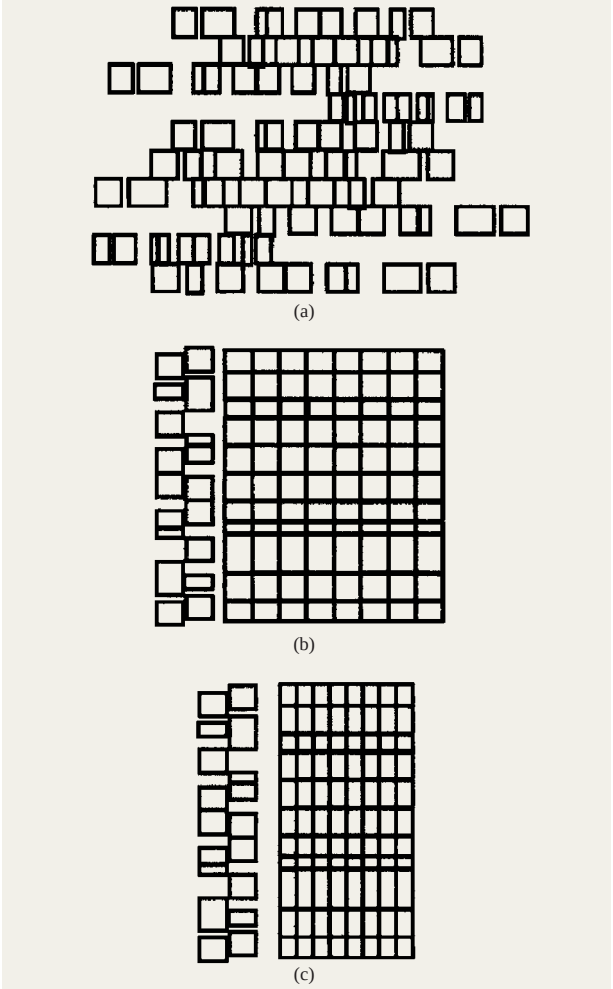


Figure 3

Floorplan as a function of the degree of customization: (a) automated placement; (b) customized placement; (c) custom circuit design.

technology in an attempt to quantify the improvements that are possible. Section 6 discusses a prototype of the central component of a high-frequency out-of-order superscalar processor. It shows the advantages that can be obtained by integrating control and dataflow functions, and provides another example of careful transistor-level design. Section 7 looks briefly at the future.

2. Transistor- and interconnect-level design

To achieve the highest frequency in a given technology, a circuit macro must be designed to provide an optimal balance among delay, power, and area. This requires an understanding of the physical and electrical context of the macro within the overall design. In addition, the designer must have access to tools that allow manipulation of the design database to effect the optimal solution, while maintaining schedule and minimizing risk.

- *Custom design*

In practical terms, "custom design" means control over the circuit style and topology, device sizes, and the physical design of both transistors and interconnects. Minimizing both parasitics and the number of stage delays incurred to implement a particular function are keys in attaining a high frequency. Computer-aided design tools do not always provide the flexibility to allow the designer enough control to arrive at the optimal solution. Accordingly, a significant amount of custom design tends to be manually driven, resulting in far less productivity (using a metric such as transistors per day, for instance) than that of a designer using a cell-based methodology. Therefore, it is imperative that early in the design cycle the designer identify which portions of the design are appropriate targets for a custom design solution and which are better left to a more automated approach. In a processor design, a large fraction of the logic is irregular and cannot be separated into multiple instances of similar or identical logic. However, a sizable remainder of the logic is dedicated to operating on data streams that are 32 or 64 bits or wider, providing a golden opportunity for reusing customized physical and circuit designs.

Figure 3 conceptually illustrates the varying levels of customization that can be used to reduce the physical

dimensions of a given block of dataflow logic to improve the maximum operating frequency of the design. In Figure 3(a), a cell-based methodology has employed design automation tools to place predesigned elements such as logic gates and latches taken from a library ("cells"). In order to maintain wirability, placement automation tools typically achieve an active area (sum of the areas of the bounding boxes of constituent cells) that is about 50 to 80% of the area of a bounding box encompassing the limits of the macro. Packing the cells closer together reduces the number of available wiring tracks and can result in non-optimal routing ("scenic routes") or even failure of the automated wiring tool to achieve any solution at all.

Achieving the density of the design shown in Figure 3(b) requires manual intervention by the designer. Here, the regularity of the logic function has been exploited by organizing the elements of a given logical data flow ("bit slice") into a vertical stack and replicating this slice for the width of the function ("bit stack"). In this example, data flows from the top of the stack toward the bottom of the stack, so inputs to the macro and circuits implementing logic earlier in the dataflow would be placed nearer the top of the stack, and outputs and logic near the end of the dataflow would be placed nearer the bottom. Ideally, the length of any interconnect is minimized by intelligent placement of the circuits within the dataflow. Achieving the packing density implied by the figure demands a high level of understanding of the connectivity of the dataflow and typically requires a heavy investment in time on the part of the designer, who is likely to arrive at an optimal solution only after a lengthy period of trial and error. This investment pays off only because the regularity of the dataflow allows the designer to reuse the design of the bit slice many times over the dataflow.

In this example, some amount of logic is shown off to the side and not incorporated into the stack. This represents control logic that is usually not regular and therefore does not require or benefit from customized placement. In Figure 3(c) customized placement is augmented by custom circuit design. Instead of limiting the circuit implementation to only those elements that are available in the predesigned library, the custom design can employ whatever circuit design style best fits the logic function. The designer can improve the wirability of the design by planning the allocation of wiring tracks, including ordering cells within a stack to minimize congestion and customizing cell pin locations so that they align. More efficient routing and closer proximity of cells within a stack result in shorter interconnects and less parasitic capacitance, allowing the designer to reduce device sizes or even eliminate entire levels of buffering.

- *Custom design tools*

Designing at the transistor level places demands on designer skills, design methodology, and tools. Tools for custom circuit design generally fit into one of two categories: tools that allow a designer to construct electrical and physical representations of a circuit (synthesis tools), and tools that allow a designer to verify the expected behavior of a circuit (analysis tools).

Creative topologies and optimized device sizes are important factors in minimizing circuit delay, but careful management of interconnect resources is just as important to successful high-frequency design. A custom circuit design must consider both the delay and the noise effects of interconnects on circuit behavior. Long interconnects can introduce significant capacitance, resistance, and even inductance to a critical path and may even dominate the delay. Transitions on neighboring wires can result in capacitive coupling to an otherwise "quiet" net and appear as noise on the input of a circuit. If the noise exceeds the margin of the circuit, functionality can be threatened. Therefore, the circuit designer's "toolbox" must include tools for engineering high-speed interconnects, using techniques such as shielding, non-minimum widths, and isolation to minimize parasitics and interactions between wires. The designer must also have tools to analyze the design, including extracting parasitics, predicting noise events, and modeling the effects of parasitics on delay.

- *Transistor-level static timing*

Meeting aggressive frequency goals requires iterating and tuning a circuit design to minimize delay while abiding by area and power constraints. The ability to quickly determine the slowest paths and accurately predict the delay through a circuit is key in achieving an optimal design. Block-based static timing, while appropriate in a cell-based methodology, requires generating a timing model that is accurate over a broad range of environmental and usage conditions. This may include temperature, voltage, input transition rates, capacitive loading, and other factors. This is overkill for custom design, since the designer is interested only in a particular application of a circuit and knows in advance the exact usage conditions. In particular, the designer requires quick turnaround to enable rapid convergence on an acceptable solution. Static timing at the transistor level is a means for a designer to quickly and accurately predict the delay through a circuit, and EinsTLT fills that role in IBM's suite of circuit design tools. EinsTLT is a transistor-level static timing system based on the EinsTimer [3] static timing engine. Simulation and modeling routines running with EinsTLT provide fast RC modeling for interconnects and simulate devices dynamically using fast, piecewise-linear techniques. EinsTLT runs on a flat netlist that is based either on schematics or on the extracted physical

design. In addition to providing timing for design iteration and development at the transistor level, EinsTLT generates a timing model that can be used to time a higher level of the design hierarchy (for example, chip level) at the block level, allowing further balancing of run time versus accuracy.

- *Design checking*

Designing at the transistor level provides circuit designers with a degree of freedom not allowed in a cell-based methodology. However, this freedom comes with a price. Expanding the design space to include additional circuit topologies and families demands additional rigor in checking so that improperly designed circuits do not find their way into manufacturing. The definition of "best design practices" is subjective and depends upon many factors (including the skill and experience of the design team and the program objectives); generally it is a set of rules and guidelines that have been agreed upon by the design team to limit the range of possible circuit designs and combinations to those believed to provide an acceptable balance of risk while still enabling the design to meet the program objectives. EinsCheck [3] provides a means for discriminating between legal and illegal circuit constructs and device sizes, and performs basic checks on electrical relationships such as noise margins, capacitive coupling ratios, interconnect current-carrying capacity, and soft-error immunity.

- *Cell and macro physical design*

Customizing physical design is as important in designing high-frequency circuits as selecting circuit styles and device sizes. Minimizing parasitics by proper floorplanning, placement of inputs and outputs, and use of shielded and isolated interconnects is the goal of any physical designer. Time-to-market demands efficient use of design resources. GYM is a proprietary physical design entry and editing application with additional capabilities that allow a circuit designer to quickly arrive at an optimal physical design. Aside from basic shape manipulation, GYM provides a circuit generator for quickly generating common circuit topologies (stacked devices, transmission gates, etc.) and a compactor for minimizing layout area. GYM also provides assistance in placement of circuit components based on schematics. For help in wiring, GYM contains interfaces to a grid-based router and a gridless router.

- *Test vector generation*

Defects introduced during wafer fabrication impair the function of a circuit or the maximum frequency at which it may operate. In order to maintain quality and reliability standards, manufacturing defects must be efficiently detected and parts containing those defects eliminated from the production flow. Testing circuits with functional patterns (for example, checking that an adder adds two operands properly) is not always effective and can lead to very long and expensive test times. One of the goals of the custom designer is to deliver a set of test vectors that provide an acceptable level of coverage (or at least deliver a testable design for which vectors can be generated at a higher level of the design hierarchy). To facilitate test vector generation, GateMaker [4] provides a means for translating a transistor-level design into an equivalent Boolean representation. GateMaker is capable of recognizing the function of advanced circuit design styles such as domino and CPL, as well as the ubiquitous static CMOS. Once the Boolean representation is generated, TestBench[1] is used to create test vectors which can be applied through the scan interface, as well as functional patterns. The combination of scan test and functional test patterns, applied at the chip level during wafer sort and module testing, provides both the ac and dc test coverage required to attain the high level of quality and reliability required.

3. Case study 1: Star series of PowerPC processors

- *Overview*

To better understand the power of custom design, it is worth taking a look at a few examples. This section describes the application of custom design techniques in a series of successful commercial microprocessor designs. We first describe the development of the Star series processors, including the original design and several derivatives. The second subsection focuses on a particular critical macro used in the Star series and describes how custom design techniques provided the performance required for these high-frequency designs.

The Star series of PowerPC processors [5–8], now shipping in IBM *e*Server pSeries* and iSeries* servers, provides an example of aggressive but controlled application of custom design techniques to enhance the performance of a family of high-frequency microprocessor designs.

The first processor in the series, dubbed Northstar, replaced a multichip processor from a previous generation. Judicious use of custom physical design in dataflow (particularly in the fixed-point and floating-point execution units) allowed what had previously been implemented in seven chips (PU, FPU, PIU, and four MSCU) to be consolidated on a single processor chip. The increased level of integration made possible by custom design lowers not only the cost of the processor, but also the overall cost by easing power dissipation and delivery, making it

[1] TestBench is a UNIX**-based set of test design automation tools developed by IBM for internal use. It was made commercially available in 1994 by the IBM Microelectronics Division.

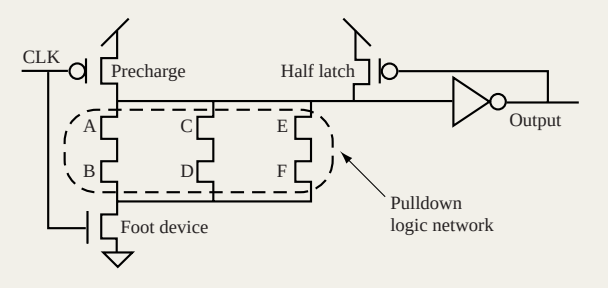


Figure 4

Basic dynamic gate.

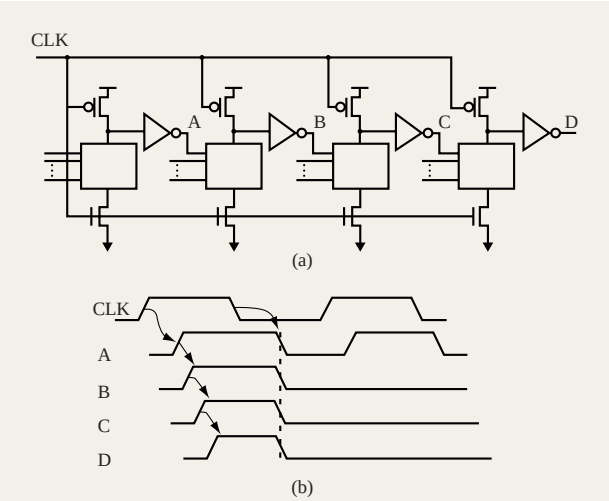


Figure 5

Basic dynamic domino circuit: (a) schematic; (b) timing diagram.

possible for more computing power to occupy a given volume in a system.

In the Northstar processor, custom circuits comprise approximately 600K transistors of a total of 2M transistors used to implement control logic and dataflow (the remainder of the 12.5M transistors were in instruction and data caches). About 170K transistors are found in dynamic circuits and the balance in custom static circuits. The comparatively small number of transistors used in dynamic circuits reflects the fact that fewer transistors are needed to implement a function in dynamic circuits, because the complementary p-FET network required for static circuits is replaced by a single precharge device in dynamic circuits. The use of custom dynamic circuits in the Northstar processor provided a significant performance advantage over CMOS static circuits.

Table 2 Star series processor evolution.

	Technology	*Frequency* (MHz)	*Die size* (mm^2)	*Transistor count*
Northstar [6]	CMOS 6S2	350	162	12.5M
Pulsar	CMOS 7S	450	139	34M
Istar [7]	CMOS 7S SOI	550	139	34M
Sstar [8]	CMOS 8S SOI	660	128	44M

Dynamic domino circuits are the predominant style of dynamic circuits used in the Star series of processors. **Figure 4** is an example of a domino dynamic gate. The logic function is implemented in n-FET transistors only; this results in lower input capacitance and lower delay than complementary CMOS gates. Following a computation, the precharge p-FET transistor resets the state of the dynamic node and forces the output voltage low in preparation for the subsequent computation. The foot n-FET transistor prevents the dynamic node from being discharged prematurely and prevents a direct path from the V_{DD} supply to ground during the precharge operation. The half-latch or keeper p-FET transistor holds the dynamic node high, and thereby the output low, for conditions in which the precharge p-FET is off and there is no open path from the dynamic node to ground.

Figure 5 presents one method by which domino dynamic gates can be connected. A common clock simultaneously initiates precharge in all of the domino gates. The computation ripples through the levelized domino gates. The basic dynamic domino circuit can be improved by changing to two-phase domino logic, in which the circuit is partitioned into two halves, the first clocked by CLK and the second by an additional clock that is out of phase with CLK. The two halves should be separated by a mid-cycle latch. This ensures that the first-phase logic signals are maintained at the input to the second-phase gates.

An investment in a custom processor design need not be a one-shot proposition. Scaling and other advanced developments in wafer fabrication technology (multithreshold devices, copper interconnects, silicon-on-insulator) can be employed to enhance the frequency and improve the integration of a custom design so that it remains competitive for several years after the original design matures. The engineering effort required to migrate a design is a fraction of that of the original design, and the same custom design methodology and skills that were used to produce the original can be used to "tune" the design in the new technology. **Table 2** illustrates the evolution of the original Northstar processor into subsequent generations. Each design point achieves a

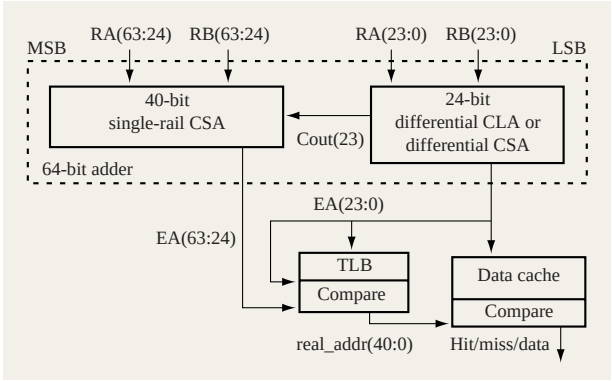


Figure 6

Data cache access path.

higher clock frequency and increased performance through integration of larger caches and other function.

- *Northstar/Pulsar/Istar/Sstar adder*

This 64-bit adder is used to compute the effective address in a PowerPC processor. The adder was originally designed in CMOS 6S2 bulk technology and later converted to CMOS 7S bulk, CMOS 7S SOI [9], and CMOS 8S2 SOI. This section focuses on the adder's migration and circuit modifications through these technologies.

In a high-speed microprocessor, cycle time is typically limited by a number of critical paths, one of which involves a cache memory access by a load or store instruction. For instance, an instruction of the form *lwaux RT*, *RC*, *RB* (PowerPC load word algebraic with update indexed) requires the value in register RC to be added to the value in register RB to produce the effective address (EA). The low-order 24 bits of the effective address, EA(23:0), are used to access the translation lookaside buffer (TLB) and the cache memory and are the most timing-critical, as shown in **Figure 6**. The chip floorplanning keeps the adder close to the TLB and cache by putting the adder near the center of the chip (**Figure 7**).

The adder is designed as a custom macro to allow fast TLB and cache memory accesses. In this implementation, the dynamic macro provides the low-order 24 bits of the Effective Address faster than the high-order 40 bits. This is done in the 6S2 adder by using a 64-bit carry-select adder (CSA) in which the low-order 24 bits are implemented in dual-rail dynamic and the high-order 40 bits in single-rail dynamic. In later designs, the effective address is implemented by a high-order 40-bit single-rail dynamic carry-select adder and a low-order 24-bit dual-rail dynamic carry-lookahead adder (CLA). See [10] for an overview of CSA and CLA adders.

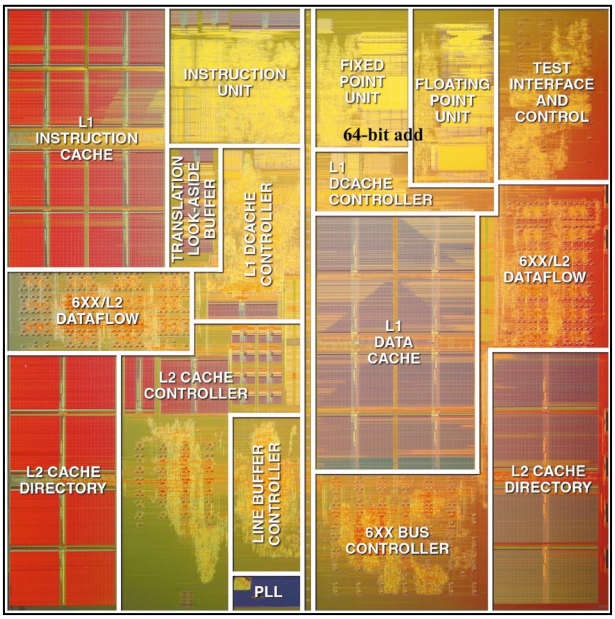


Figure 7

Dynamic adder placement (chip die photo courtesy of Thomas Way, IBM Microelectronics Division).

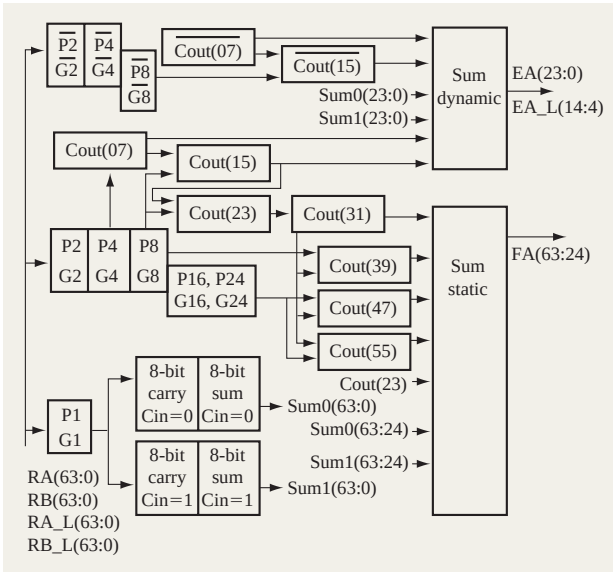


Figure 8

Northstar adder organization.

Figure 8 shows the internal block diagram for the CMOS 6S2 64-bit CSA. The adder is broken into 8-bit

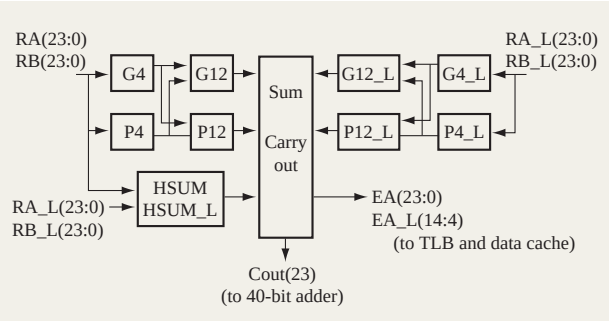


Figure 9

Pulsar adder organization.

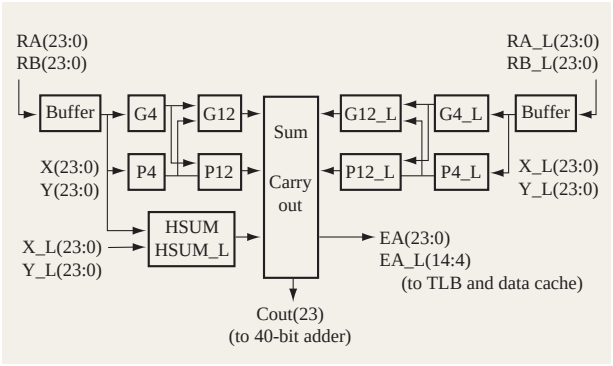


Figure 11

Istar adder organization.

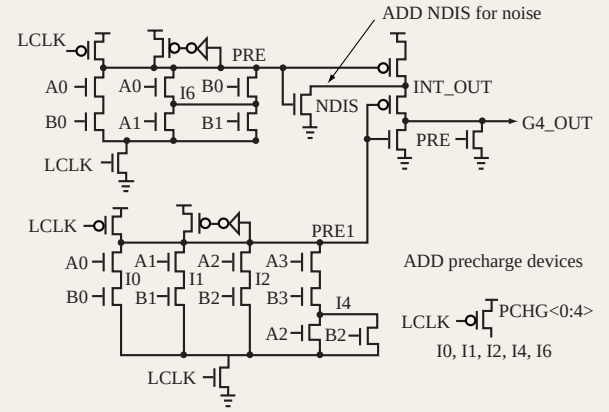


Figure 10

CMOS 7S 4-bit generate circuit.

sections. Each 8-bit section generates a sum assuming a carry-in of 1 and 0; Sum0(63:0) and Sum1(63:0) are the concatenated buses for these eight 8-bit sections. The global carry signals, Cout(7, 15, 23, 31, 39, 47, 55), select the sum that is the correct result for their respective 8-bit sections. The circuits in this adder are simple-function dynamic circuits: ao22, oa22, and2, or2, etc. Within the 24-bit dual-rail section, both phases are generated using separate circuits. The final sum circuit is dynamic for speed. This is possible because the global carries are dual-rail dynamic signals. Within the 40-bit CSA, single-rail is used, since the outputs are not as critical and the final sum is a static circuit.

The adder migration to CMOS 7S did not meet the performance requirements, so circuit changes were required. The low-order 24 bits, EA(23:0), are generated by a dual-rail CLA. The 24-bit CLA is a three-level adder (**Figure 9**). Positive logic is shown on the left and negative logic on the right. The remaining high-order 40 bits, EA(63:24), are generated by the same CSA logic as in the previous adder.

The 4-bit generate circuit (**Figure 10**) is the most complex circuit for the CMOS 7S adder and allows a three-logic-level adder. This circuit is a compound domino structure in which the inverter is replaced with a NOR gate [11], which gives both buffering and logic for faster performance. The NDIS transistor is a noise-suppressing discharge transistor that prevents a charge-sharing event between node INT_OUT and the output G4_OUT. The precharge devices, PCHG(0:4), are added to precharge internal nodes and prevent charge sharing. This circuit is the first level of the adder. It receives its inputs from latches, so foot devices are required in the pull-down network. Foot devices are n-FETs in the pulldown network gated by the local clock.

In addition to making the adder faster, the final level of the adder generates both phases of the effective address used by the data cache, EA(14:4) and EA_L(14:4). During the precharge, EA(14:4) and EA_L(14:4) are low; during the evaluate phase, EA or EA_L goes high for each bit. The data cache array begins its access when it detects that all address bits have been evaluated. This self-timed data cache solves the chip's worst cycle-time-limiting path.

The technology migration to 7S SOI met the performance requirements, but circuits were changed to combat SOI noise effects due to the floating body [12–15]. The 24-bit CLA is a three-level adder with an additional dynamic buffer as the first level (**Figure 11**). Custom design work focused on noise immunity to ensure that

bipolar effects were contained and lower noise immunity maintained.

The G4 circuit in 7S SOI is the second level of logic within the adder (Figure 11); the first level is the dynamic buffers. This allows the removal of the foot devices, which eliminates one of the bipolar conditions and increases the speed of this circuit to offset the additional delay from the dynamic buffers. The path delay did not change, yet the G4 gained noise immunity. Although removing the foot devices helps, it does not solve the entire noise problem. SOI bipolar leakage from the precharge node to intermediate nodes can still occur. Therefore, internal nodes within the pulldown network of G4 are discharged during the precharge phase to reduce SOI bipolar effects. Discharging these internal nodes guarantees charge sharing within the G4 circuit. However, this effect is reduced in SOI technology because of lower drain/source capacitance. In **Figure 12**, six discharge devices, PDIS(0:5), are added to the G4 circuit. Note that circuit data inputs (A0, A1, A2, and A3) drive these discharge devices, except in the case of PDIS(1), which must use the CLK input to avoid a dc path from V_{DD} to ground. This causes minimal effects on performance while preventing SOI bipolar effects.

When the adder was converted into 8S2 SOI [16], timing requirements were met through the technology, but changes were necessary because of scaling effects. Using G4 as an example (Figure 12), the NOR circuit requires P0 and P1 to be large since they are in series and part of the evaluate path, making the beta ratio of the NOR high. A problem occurs for this technology when the PRE node discharges and the PRE1 node falls slightly from noise effects. In this state, an output noise pulse can be generated because of the high beta ratio of the NOR. To solve this problem, two additional n-FETs, N2 and N3, are added to the static NOR [11]. These n-FETs alter the beta ratio of the NOR, allowing more noise immunity for PRE1. The intent is to have NDRIVE high during burn-in or stress testing and low during normal operation. NDIS also performs this ratio balancing, but for noise immunity on the PRE node.

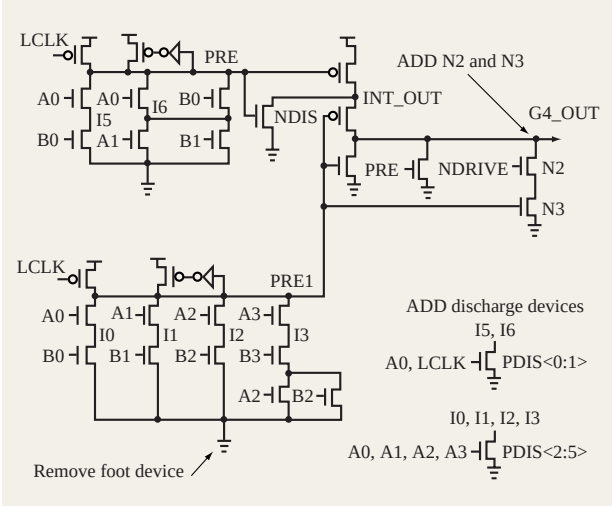


Figure 12

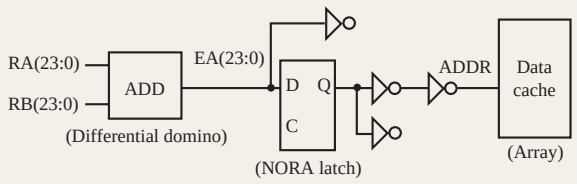
Istar G4 circuit.

SOI technology provides greater speed than bulk technology because of reduced drain/source capacitance, increased current due to lower threshold voltages, and a reduction in the reverse-bias effect on stack transistors. **Table 3** compares this adder's 6S2, 7S, 7S SOI, and 8S2 SOI delays from SPICE simulation. EA(23) and Cout(23) are the MSB and carry-out of the 24-bit adder. EA(63) is the MSB of the entire 64-bit adder. For second-generation SOI, EA(23) is determined in less than 278 ps, while EA(63) is complete in less than 435 ps. There is a 28% improvement moving into the first generation of SOI and a 21% improvement going to the second generation of SOI. The bottom row shows the timing results from a

Table 3 Star series path delay comparisons.

Path	*CMOS 6S2 bulk delay* (ps)	*CMOS 7S bulk*		*CMOS 7S 1st SOI*		*CMOS 8S2 2nd SOI*	
		Delay (ps)	*A* (%)	*Delay* (ps)	*B* (%)	*Delay* (ps)	*C* (%)
Cout(23)	753	426	−43	337	−21	283	−16
EA(23)	625	487	−22	351	−28	278	−21
EA(63)	942	615	−35	489	−21	435	−11
FO4	94	65	−31	52	−20	38	−26

A Improvement compared to CMOS 6S2.
B Improvement compared to CMOS 7S.
C Improvement compared to CMOS 7S SOI.



CMOS 8S2 SOI: 681 ps simulated
CMOS 8S2 SOI: 675 ps hardware

Figure 13

D-cache address timing path.

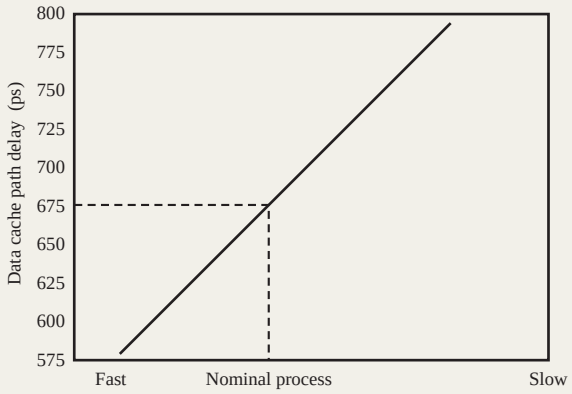


Figure 14

Hardware delay measurement.

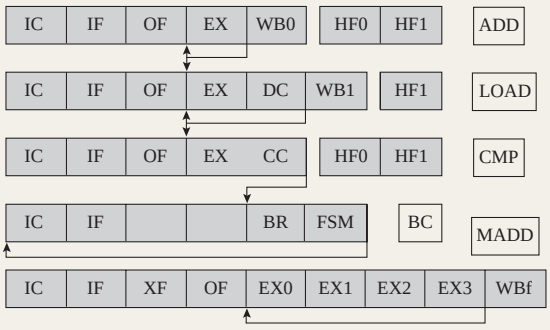


Figure 15

Pipeline of the Rivina prototype processor shown in Figure 17.

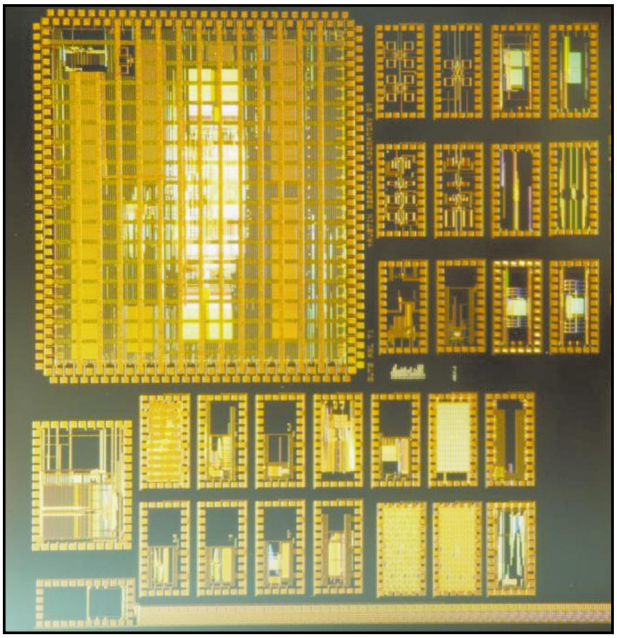


Figure 16

Test site for guTS.

well-known generic circuit, fan-out 4 (FO4),[2] which serves as a reference for speed comparison across technologies. The FO4 circuit is an inverter driving four inverters of similar size. FO4 in CMOS 6S2 uses normal-threshold devices, whereas the other three delay numbers use low-threshold devices. Device sizing was done by using the driver strength versus load ratio and wire rise/fall time requirements.

The values in Table 3 are simulated. To validate these times to hardware, the path from the adder to the data cache is measured in the hardware (**Figure 13**). This path passes through the adder, a latch, and several inverters to move across the chip to the arrays. The timing tools and methodology simulate this path at 681 ps. The hardware measurement for this path at the same conditions is 675 ps. **Figure 14** shows the best-fit line of data measurements on a number of parts across the spread of the process. The left axis shows the delay for the path from the adder to the data cache. The bottom axis shows the process, in which fast chips are on the left and slow on the right. The dotted line indicates the nominal process and points to the hardware speed of 675 ps for the data cache path.

Performance comparisons indicate a 28% gain from bulk to first-generation SOI, followed by an additional gain of 21% to second-generation SOI. Overall, the adder

[2] D. Harris, R. Ho, G.-Y. Wei, and M. Horowitz, "The Fanout-of-4 Delay Metric" (private communication).

speedup is more than 43%, giving a complete 24-bit add in less than 280 ps. The less critical 64-bit path speedup is more than 30%, with path delay less than 440 ps. The entire area of the 64-bit adder in CMOS 8S2 measures 0.357 mm^2. Challenges in SOI dynamic circuits are solved by the methods discussed above. Additional improvements in noise immunity can be obtained by reordering the inputs to the pulldown network and by adding additional feedback [16].

4. Case study 2: The guTS and Rivina PowerPC prototypes

- *Introduction*

This section discusses two prototype PowerPC processors that were designed to demonstrate the performance gains that can be obtained with a focus on physical design. Both designs have a cycle time equivalent to about 14 fanout-4 balanced inverters, and thus could be expected to have very deep pipelines. Instead, these processors have very short pipes (**Figure 15**) and do more work per cycle by using aggressive circuit techniques, a carefully crafted floorplan, and a PD-aware microarchitecture.

The first of these prototypes, "guTS" [17] (**Figure 16**), consists of a GHz integer core with 4KB caches. A subset of about 100 integer instructions from the PowerPC 64-bit ISA is supported. This prototype effort taught us how to develop GHz-class delayed-reset and self-resetting dynamic components and helped develop a discipline and methodology necessary for integrating these components without sacrificing frequency. The second prototype, "Rivina" [18] (**Figure 17**), implements the complete 64-bit PowerPC instruction set, including floating-point and address translation. It also achieved a more balanced design by replacing the single-cycle 4KB caches with 64KB, two-cycle caches. The Rivina processor required a new control methodology, built around dynamic PLAs, so that control functions could be implemented to match the delays of the dataflow elements.

- *Timing closure by design*

With a cycle time that is no more than the delay of about 14 fanout-4 inverters, including wire delays and clocking uncertainties, it is clear that cycle time must be budgeted to an extraordinary degree. For example, merely inserting a redrive buffer (pair of inverters) without planning for it up front can cause the cycle-time target to be missed by up to 10%. To guarantee eventual timing closure, all logic was designed to follow the strict partitioning of the cycle shown in **Figure 18**. In addition, up-front floorplanning and global wiring and timing runs before the macros were fully realized allowed us to identify areas requiring modification early in the process. Finally, macro schematics were constructed to include all wires longer than a few

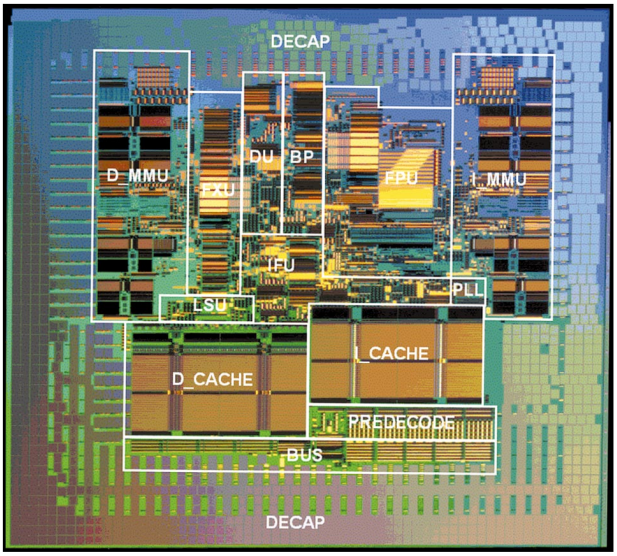


Figure 17

Rivina prototype processor (photo courtesy of Thomas Way, IBM Microelectronics Division).

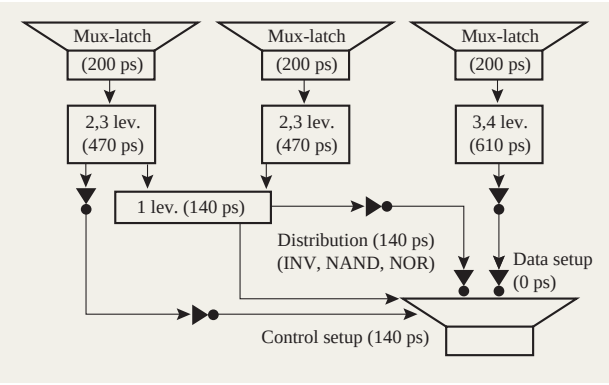


Figure 18

Rivina timing template.

hundred microns. These "hand-extracted" schematics guarantee that transistor sizes are optimized appropriately and that final delay models extracted from the layout closely match what is designed.

- *Delayed-reset and self-resetting dynamic circuits*

With a delay budget of no more than 610 ps, or about nine fanout-4 inverter delays, to realize the dataflow macros such as register files or a fixed-point unit, even well-designed two-phase dynamic circuits do not meet the

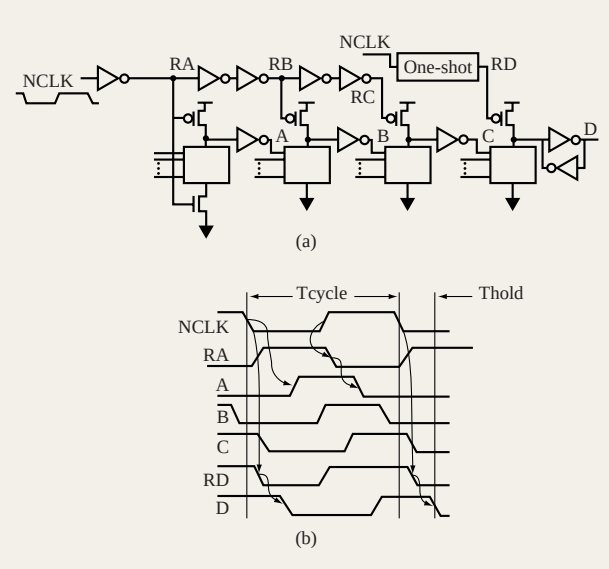


Figure 19

Delayed-reset circuit timing: (a) schematic; (b) timing diagram.

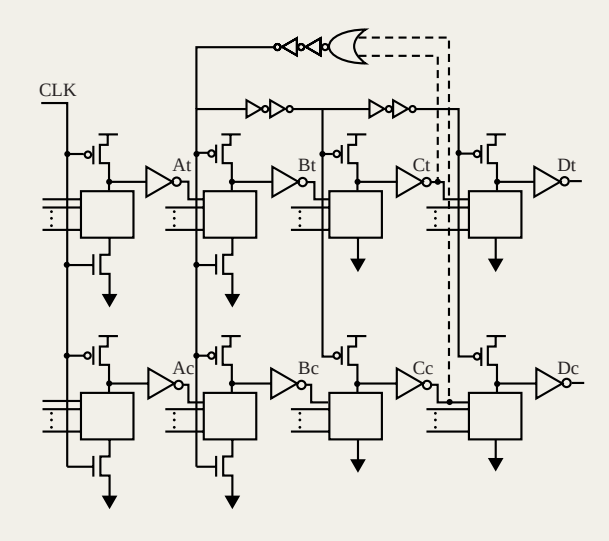


Figure 20

Self-reset circuit organization.

latency requirements. Also, at least as important, resetting multiple stages of dynamic logic at the same time leads to unnecessarily large reset devices.

Delayed-reset dynamic circuits [17, 19, 20] were used predominantly, as illustrated in **Figure 19**. Staging the precharge signals allows foot devices to be eliminated, improving circuit performance by 5 to 15%. Combining a multiplexor with the latch and eliminating the mid-cycle latch further improves evaluate latencies. Cycle time improves, because it is now limited by the reset and evaluate delays of a single stage, plus clocking and processing uncertainties. One-shots are used, both in the sampling multiplexor–latch and in the reset of the final stage of dynamic logic. The former provides the logic stages with a fixed-width pulse, which tends to widen through the stages of a latency-optimized design, whereas the latter guarantees a fixed hold time into the next cycle, which simplifies integration. Short fixed-width pulses, however, stress certain devices for only a small fraction of the long cycle times typical for burn-in (stress-testing). Additional clock signals, selected only during burn-in, allow all devices to be stress-tested for a fixed fraction of the cycle.

Self-resetting circuits [21, 22] were used in the floating-point multiplier. A self-resetting dynamic circuit is shown in **Figure 20**. In this example, the dual-rail outputs Ct and Cc are ORed to obtain a completion signal, which is inverted and delayed to drive the precharge signals for the domino levels that form the outputs B, C, and D. Self-resetting circuits were used to provide timing robustness in the floating-point multiplier. Self-resetting and delayed-reset dynamic circuits distribute the precharge load both spatially and temporally, which improves power-supply noise and eliminates long precharge signal wires.

- *Array-based control*

Although aggressive dataflow macros and a well-controlled floorplan are necessary to vastly improve frequency, they will succeed only if there is also a methodology for generating control elements that can match their performance. Most current designs are realized with custom dataflows and static synthesized control. With a budget of only 470 ps, or about seven fanout-4 inverters, for control logic, it soon became clear that this methodology would not support control for the short-pipe designs we were aiming to construct. On the other hand, individually designed custom dynamic macros for control would not have allowed us to overlap the physical design phase with the logic design phase, and would have added significantly to the schedule and design cost. Instead, all logic was constructed to fit in a set of custom-designed but automatically personalized PLAs, followed by at most one more stage of complex dynamic logic before controlling the multiplexor latch. Logic is partitioned by hand to meet the constraints of the largest realizable PLA. Logic optimization, PLA type selection, schematic and layout personalization, and logic-to-layout verification are all automated to create a PLA synthesis methodology. Like all other macros, PLAs are manually placed. In the Rivina

processor, 191 PLAs and a total of 88 different PLA personalizations were used.

• *Merged functionality and parallel computation*
Relative to static circuits, logic functions are most optimally implemented in dynamic circuits by using fewer levels of higher complexity. In order to fully exploit the benefits of dynamic circuits, the microarchitecture must be designed to this approach. This results in functional blocks which combine functions that are traditionally composed of separate structural units. Examples include the multiplexing latch and the MFXU, which is a complete fixed-point unit in a single macro [17]. In addition to using fewer, more complex gates, the unit reuses wires, and therefore allows a tighter integration, reducing wire delays. A third example can be found in the Rivina register file. There the multiplexor that supports forwarding from the write ports to support read-after-write operation has been widened to include immediate generation of operands at a minimal area and delay penalty, saving a separate immediate generation unit and separate buses for distribution of immediate operands. Final examples of merged functionality are the sum-addressed caches, developed independently at IBM [17] and Sun Microsystems [23], that allow elimination of the address-generation cycle preceding cache access. In addition to sum-addressed caches, Rivina included a sum-addressed address-translation unit.

With short cycle times and the resulting few levels of logic which can be accommodated, parallel computation of multiple possible results with late selection of the correct results can become necessary. For instance, in the Rivina floating-point unit, three possible result exponents are formed in parallel. The critical path is the logic that performs the selection of the correct result. Parallel computation is also used to speed highly critical portions of a computation. For example, an optimized structure is used to determine the time-critical sign bit of the alignment shift amount in the floating-point unit faster than it is produced by the shift-amount adder. Another example is the guTS compare unit, which computes the condition codes for most arithmetic operations in a separate unit prior to the completion of the ALU operation [24]. (A traditional organization would first complete the ALU operation and then compute the condition codes in a subsequent cycle.)

• *Microarchitectural innovation*
While the floorplan and preliminary timing results based on estimated macro delays, loads, and drive strengths were being studied, two paths emerged that were unlikely to close. The first was the path to preclude updates of the architected state in the case of a cache miss. The second was the hold control-signal on the copies of the instruction buffer, driven by the dependency-detection circuitry. Both led to modifications in the microarchitecture: a history-file mechanism and dependency-prediction circuitry (**Figure 21**), respectively. The important point here is that neither of these modifications seemed necessary from an analysis based only on logic complexity. Only when the floorplan and signal loading are taken into account does the problem become obvious. We believe that either one could have caused us to miss our target by a wide margin, and that our experience stresses the importance of designing the floorplan and logic hand in hand.

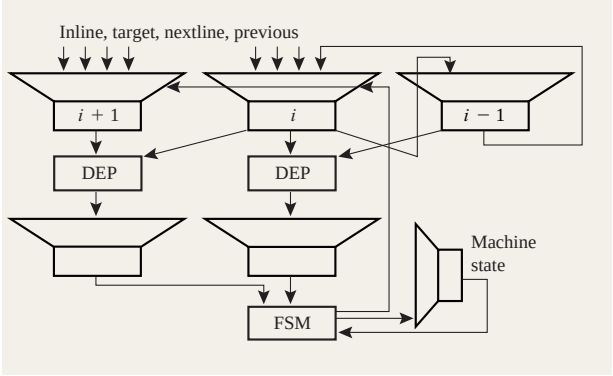
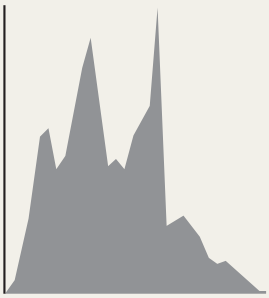


Figure 21

Dependency prediction.

• *Fast-turnaround chip integration and timing*
With a significant fraction of the cycle time devoted to data forwarding and distribution, macro and inverter placement and wire-level assignments must be designed, then subsequently fine-tuned. Fine-tuning is an iterative process, and the more iteration, the better the result. We were able to process incremental changes to the design and generate a new timing report within 24 hours [25].

Structural VHDL provides a netlist for wiring and a list of macros. New inverters and PLAs are generated and placed. A dataflow compiler adjusts the placement of macros in placed stacks. A quick-running initial global route provides representative wires that can be used in the wire-extraction process. This initial route and extraction took about eight hours for Rivina. A detailed route and final extraction ran overnight. The extracted delays and capacitances, along with extracted macro input capacitances and simplified macro timing equations based on load and simulated delay, allow an accurate timing run in about 15 minutes. Also, existing IBM clock distribution methodology based on a tuned H-tree driving a grid [26] was extended to speed up integration. Clock distribution skew was designed to be less than 15 ps [18]; thus, even in



IC + IF (data)	1902 ps
IF (ctrl)	952 ps
OF (data)	908 ps
OF (ctrl)	963 ps
EX (data)	953 ps
EX + DC (data)	1902 ps
CC set	921 ps
Branch Res.	976 ps

Simulation at 1.62 V (nom − 10%), 85°C, 50% sort point

Figure 22

Rivina timing histogram.

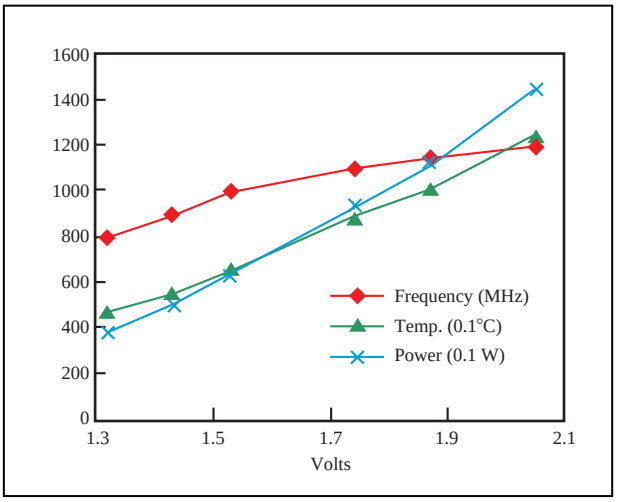


Figure 23

Measured results on fast hardware.

this high-frequency design, ignoring clock skew until the very end constitutes a valid approach.

- *At-speed scan-based testing*

Both the guTS and the Rivina processors include a serializing–deserializing tester interface [27] that enables at-speed functional testing of the processor using test equipment that runs at a fraction of the processor frequency. Both scan data and scan control are serialized so that all synchronization with the tester occurs at the slower (tester) frequency. One of the eight multiplex inputs of the multiplexing latch is dedicated to support scanning at a minimal cost to the overall design.

- *Predicted and measured results*

The static-timing slack histogram in **Figure 22** shows critical paths for both data and control in most pipeline stages to be about 950 ps. The critical path of the machine is in the branch resolution logic. Resizing a comparator circuit output stage, in which a cell was reused to reduce design time, would fix the problem. From the per-stage critical path delays, and the slack histogram, it is obvious that improving the cycle time below 950 ps requires hard work in nearly all pipeline stages. This is how it should be; i.e., the design is well balanced.

Figure 23 shows the results of functional test on the packaged modules for fast hardware. At a module supply of 1.87 V, the processor runs at 1.15 GHz, dissipates 112 W, and operates at 101°C. At just above 1.5 V, the processor tops out at 1 GHz at about 65 W and 65°C. The processor is still functional at 1.3 V on the module, and runs at 800 MHz, 40 W, at about 50°C. Rising chip temperatures limit the peak performance of 1.2 GHz.

- *Summary*

These short-pipe processors at a delay equivalent to only 14 fanout-4 inverters demonstrate the advantages that can be obtained by focusing on physical design. It is also demonstrated that these gains do not have to come at the expense of either productivity or reliability. The Rivina processor contains 19 million transistors and is described by some 325 000 lines of VHDL, yet it was realized by a team of, on average, ten people over two years. The design supports scan-based testing, and the dynamic circuits have been designed to operate under burn-in conditions.

5. A comparative study of the load latency in the Pulsar and Rivina processors

- *Introduction*

In this section we analyze the Rivina and Pulsar data cache access paths in some detail. The comparison is interesting because both processors provide a two-cycle access path to a cache of approximately the same physical size (the Pulsar cache is 128 KB and the Rivina cache is 64 KB but dual-ported). Both caches are two-way set-associative. Both processors have similar numbers of execution units. Both processors were realized in IBM CMOS 7S technology, but we present the comparison of delays in technology-independent units. The Rivina two-cycle load instruction requires 29.1 fanout-4 (FO4) delays, whereas the Pulsar two-cycle load instruction requires 68.1 FO4 delays, a substantial difference. The comparison is intended to help substantiate and quantify the main premise of this paper, namely that innovation in floorplanning, physical design, and "pico"architecture can lead to significant improvements in cycle time. (The term

Table 4 Pulsar data cache access timing (data courtesy Charles Wait, IBM Rochester).

1. Internal fixed-point logic (multiplexor)	3.54 FO4
2. RA/RB K0 data-in to data-out and setup time	4.00 FO4
3. 24-bit ADD	7.38 FO4
4. Re-power and wire	1.31 FO4
5. DC_ADDR K1 data-in to data-out	1.85 FO4
6. Re-power and wire	6.00 FO4
7. DC array access	20.61 FO4
8. First level of merge logic and wire	4.61 FO4
9. DC_DATA K1 data-in to data-out	2.46 FO4
10. Cache data multiplexors	11.69 FO4
11. Wire and re-power to fixed-point unit	2.46 FO4
Clock skew and jitter	2.16 FO4
Total (two cycles)	68.07 FO4

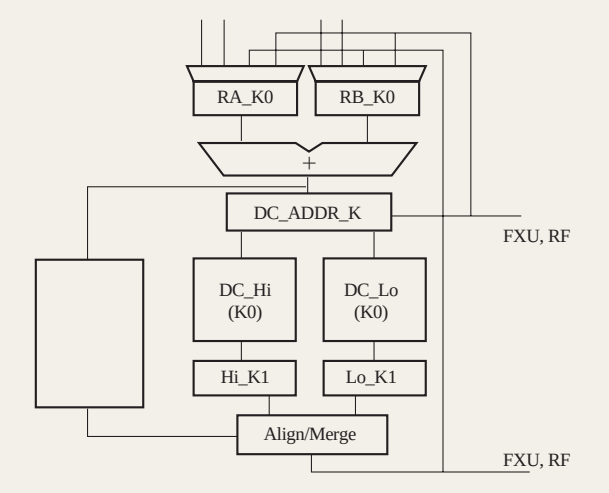


Figure 24

Pulsar data cache organization.

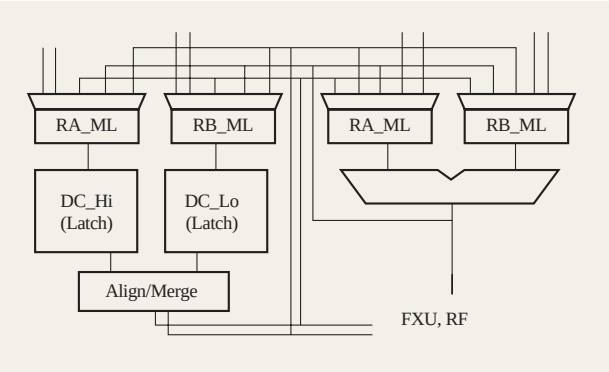


Figure 25

Rivina data cache organization.

picoarchitecture refers to improvements in organization somewhat below the level of the microarchitecture.) The types of architectural innovation we describe do not change the pipeline structure of the machine, or the order in which instructions are executed, but improve performance by enabling efficient circuit implementations and an efficient floorplan. In this section we limit ourselves to the execution of load instructions, but similar picoarchitectural innovations [17, 18, 24] have led to comparable improvements in the fixed-point execution unit and branch processing.

- *Picoarchitectural improvements*

Figures 24 and **25** show the high-level organization of the Pulsar and Rivina load-access paths. The most significant improvement in the organization of the Rivina data cache is the elimination of the effective address adder through the use of a "sum-addressed" [17, 23, 28] data store. In this organization, the word lines are selected through a combination of a carry-free adder and a traditional decoder circuit. The penalty to the access delay of the cache is minimal, amounting to less than one additional inverter delay versus 7.4 FO4 delays for the adder (**Table 4**). A second improvement is the (near) elimination of the address translation hardware from the cache access path. In a traditional organization, the TLB/directories must be accessed in order to perform the selection between the data of the two sets. The Rivina cache uses set prediction integrated with the address decode circuitry [28]. This implies that the set prediction bits must be stored twice, but it relieves the designer from the duty of floorplanning the address translation tables physically close to both the data cache and the address latches. In Rivina, a small store queue and a history file allow recovery of the overwritten architectural state for a limited number of cycles. As a result, the timing of the cache miss signal, which originates in the address translation macros, is not critical, and further allows us to floorplan the address translation macros wherever it suits us.

- *Improvements in the floorplan*

Figures 26 and **27** show the locations of the different units and wires involved in the data-cache access paths tabulated in Table 4 and **Table 5**. With the elimination of both the effective address adder and the address translation hardware from the critical path of the load instruction, the designer can focus on placing the address latches and the target latches (units) for the load data as close to the cache as possible. In Rivina the address latches are placed directly on the perimeter of the data

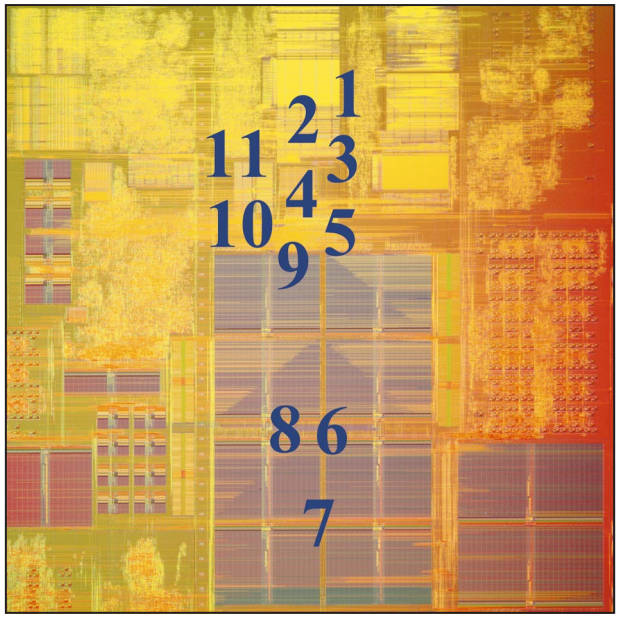


Figure 26

Pulsar data cache access floorplan.

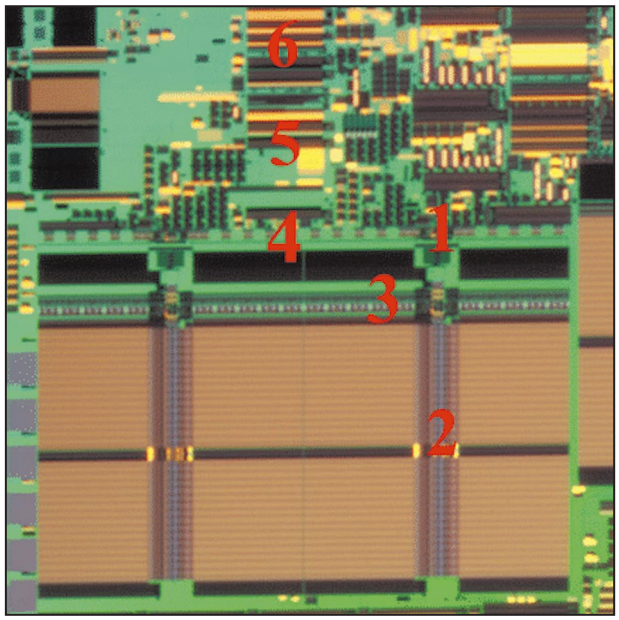


Figure 27

Rivina data cache access floorplan.

cache, which saves 1.31 FO4 in repowering and wire delay, and the latches to which the data must be forwarded are centered with respect to the cache macro for minimal delay. The total delay in wires and buffers (inverters) outside the cache macro in the Pulsar load path is 9.77 FO4, and 2.65 FO4 in the Rivina load path. The net saving is about 7.1 FO4.

Table 5 Rivina data cache access timing.

1. Multiplexor-latch data-in to data-out	3.24 FO4	3.24 FO4
2. Sum-addressed array access	16.39 FO4	16.39 FO4
3. Three-level align–merge	4.25 FO4	4.25 FO4
4. Wire and final merge	1.54 FO4	1.54 FO4
5. Re-power and wire to latches	1.68 FO4	
6. Re-power and wire to RF		5.56 FO4
Clock adjustment on RF latch		−2.91 FO4
Clock skew and jitter (2×)	1.00 FO4	1.00 FO4
Total (two cycles)	28.10 FO4	29.07 FO4

- *Improvements in the latches and clocking*

A significant improvement in latency is obtained by using (edge-triggered) sampling latches rather than level-sensitive latches. The use of edge-triggered latches allows the mid-cycle (K1) latches to be removed. The latency is further optimized by incorporating the (8:1) multiplexor function in the Rivina latch. In both Pulsar and Rivina one of the latches is hidden in the storage array [28]. The combination of the multiplexor and the latches outside the array accounts for a total delay of 10.0 FO4 in Pulsar and 3.2 FO4 in Rivina. A low-skew clock distribution and low-jitter PLL minimize the clock skew and jitter penalty from 2.16 FO4 to 1.0 FO4. Together the differences in latching and clocking penalties amount to 8.0 FO4.

- *Improvements in the circuits*

The remaining differences in latency, totaling about 16.5 FO4, stem from the extensive use of delayed-reset and self-resetting circuits in the Rivina data cache and merge and align circuitry. It should be noted that the Pulsar effective address adder implementation and self-timed cache are quite aggressive, and that a comparison against a fully static implementation would lead to a more significant difference.

- *Summary*

The two-cycle Pulsar load delay is more than twice that of the Rivina two-cycle load delay. Of the difference in delay, 19% (7.4 FO4) is attributable to a difference in picoarchitecture, 18% (7.1 FO4) is attributable to the improvements in the floorplan that flow from this improved architecture, 21% (8.0 FO4) can be attributed to improved clocking and latches, and 42% (16.5 FO4) can be attributed to the use of an improved circuit family. Many of the improvements are interdependent; the

floorplan improvement stems in large part from the improved picoarchitecture, and the circuit-family and clocking/latching improvements cannot be obtained independently.

6. Case study 3: Instruction window buffer

• *Introduction*

This section discusses a so-called "instruction window buffer" (IWB) that manages instructions and interrupt processing, controlling the processing sequence for an out-of-order CPU. The out-of-order processor dispatches the instructions still in order but allows them to execute and complete out of order. The generic structure of the out-of-order CPU is shown in **Figure 28**. After instruction fetch, decode, and branch prediction, the instructions are stored in an instruction queue. In a single cycle, up to four instructions can be dispatched from the instruction queue into a 64-entry reservation station. During this dispatch process the target register is renamed to map the register name space onto the reorder buffer registers. This is done to avoid the so-called "write-after-read" and "write-after-write" hazards that otherwise would prevent the out-of-order execution of these instructions. In the reservation station the instructions wait for operands while dispatch proceeds. As operands become available, in each cycle up to four instructions are issued from the reservation station to the four execution units, possibly out of program order. After execution, the results are stored in allocated entries of the reorder buffer. The reorder buffer (ROB) stores the results until they are written to the register file in program order by the retire process.

The described blocks, storing the instructions until the operands are valid and reordering the results before the instructions are retired, are complex to control, especially since it is required to maintain precise exceptions (in-order execution appearance in case of an interrupt) and to recover from mispredicted branches. Because of this complexity and the required high frequency of operation, all design aspects from microarchitecture to logic, circuit design, and floorplanning have to be addressed from the beginning. This is required in order to achieve an optimal solution with respect to performance, power consumption, and cost. Otherwise, the control flow path limits cycle time (or additional pipeline stages must be introduced, leading to an overpipelined processor design).

As a feasibility study for future high-frequency IBM S/390* processors, the critical parts of an instruction window buffer (IWB) were designed, implemented, and fabricated in a 0.18-μm IBM CMOS 8S bulk silicon technology.

The IWB contains the renaming logic, reservation station, and reorder buffer, as shown in Figure 28. In the next subsections we describe how this IWB is developed and implemented. We then present the measurement results of the IWB hardware.

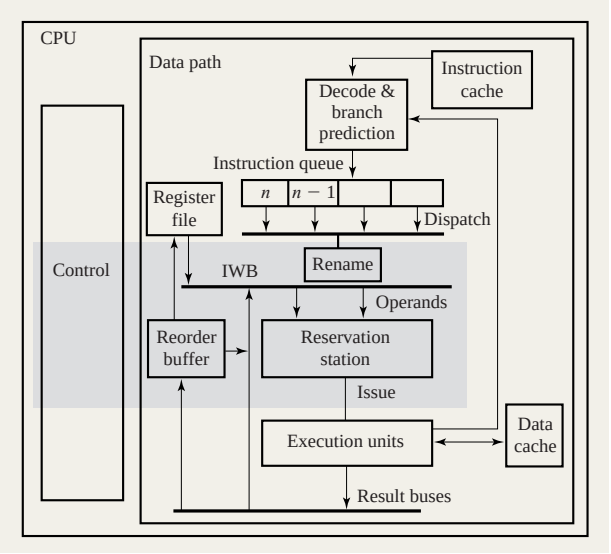


Figure 28

Out-of-order CPU structure.

• *IWB high-frequency design methodology*

After initial architectural design and its modeling in C++, the first step in the development process was the partitioning of the IWB into independent macros. In this process, logic and circuit designers worked together to cover all circuit and logic aspects down to the transistor level, and physical placement from the beginning. Initial design focused on how the dataflow could be mapped onto the reorder buffer and reservation station macros such that it could support a GHz+ frequency. The greater challenge, however, was subsequently to design a control structure that could support complex control for the data path and enable very high frequency. Thereby, the initial assignment of the dataflow was optimized (for example, by introducing bypasses) during the detailed design of the control flow.

One very important aspect in supporting such a high frequency was the introduction of distributed control structures. Instead of implementing a single control block that controls all parts, as shown in Figure 28, a structure was designed in which each part of the data path is controlled locally. This leads to a dataflow-like operation in which each entry in the reservation station and reorder buffer entry also has all of the status bits. From these status bits, the state of the instruction can be calculated without the need for any global control signals. For

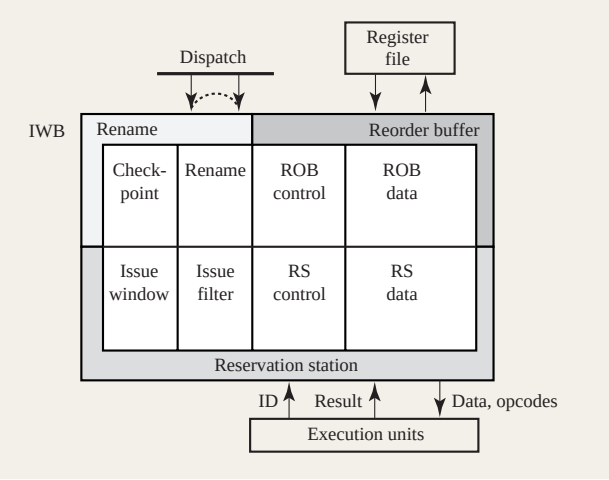


Figure 29

Partitioning of the IWB into array macros.

example, each reservation station contains the same issue logic for every entry. Only the final stage of control, selecting the entry that will be issued to a specific execution unit, receives the inputs of all entries. Stages that receive the inputs of all entries are kept as simple (and regular) as possible by doing all of the other tasks for each entry individually in front of those stages.

Another important aspect of the high-frequency design methodology was to design the part for regularity. Partitioning the circuits such that each entry is controlled locally, followed (or preceded) by a stage that connects all entries, leads to logic that can be realized by regular structures.

The final aspect of the high-frequency design methodology was that, because of the regularity of the design logic structure, each such macro could easily be implemented by array structures. The IWB macros contain array structures such as decoders, CAMs, comparators, greater–equal compares, and priority networks to implement tasks such as "selecting the entry in which the result must be written in the reorder buffer," "selecting the active window of the reservation station," and so on. The use of array structures has the advantage that their path delays can be estimated very accurately early in the design cycle by evaluating only the critical part of the array (the so-called cross section). Furthermore, they are the ideal place to use dynamic logic, since the timing can be very structured, path lengths are known, and physical dimensions are fixed. The introduction of dynamic logic in such an array structure improves the performance significantly without introducing significant additional costs. Additional costs are limited to additional design complexity due to less noise immunity and more complex clocking. The design process described above resulted in a design that was completely implemented by array structures. The resulting partitioning of the IWB into macros is shown in **Figure 29**.

Local to each dataflow array structure, there is a control array structure that controls the flow of data. The control wiring is orthogonal to the dataflow wiring. Control decisions are achieved in parallel for all entries, in synch. For example, each entry within the ROB control array probes the result coming back from execution to decide whether it has to be stored in the ROB data array. The reservation-station control array contains the associative search logic to select the operand locations that depend on the result data and to set their validity in parallel for each entry. The issue-filter array selects the oldest instruction waiting for issue based on the issue window calculated by the issue-window array. Finally, the renaming array and its associated checkpoint array implement the renaming function, together with the ability to reload the state of the rename logic from the checkpoint array. Such a reload of the checkpoint is performed in the case of a mispredicted branch, such that the IWB can immediately continue its operation after the instructions which follow the mispredicted branch have been purged.

- *Circuit style, timing, and clocking*

Besides performance and cost, reliability is a major design criterion for high-end server processors, affecting the implementation. As shown in [29, 30], essential parts of the S/390 processor are doubled, executing the instruction stream twice and comparing the two results before it is written back to the memory to achieve best possible failure detection and allow recovery. On a circuit level, stability, noise immunity, and a wide operation window are required. Turbo machines run with deep-sorted hardware in a chilled environment, while harsh quality screening requires function at burn-in conditions (supply voltage is 1.5 times nominal V_{DD} and 140°C). This is an extremely rigid requirement for fast hardware out of the best-case process arena. Power consumption can be a problem because of high dc currents at these conditions. Dynamic-circuit-specific problems include leakage at dynamic nodes and coupled noise due to the high V_{DD} and the low V_t of the devices resulting from the high operating temperature. For test and bring-up, the hardware must run from maximum frequency down to quite slow production-stress frequencies, perhaps even with single clock pulses. To minimize risk as well as design effort, standard static CMOS logic is used for combinatorial logic wherever possible. Having a circuit library with an almost continuous set of device strengths, together with sophisticated tools with integrated synthesis, placement,

routing, and timing capabilities, yields very dense and fast solutions [30]. However, the growing performance requirements make the use of dynamic circuits in selected places desirable. Array structures are an ideal place to use dynamic circuits. Timing can be very structured, path length is very well known, and physical dimensions are fixed. The timing is designed together with the floorplan. The circuit technique of choice for the IWB can be characterized as clocked, synchronous, unfooted, delayed-reset style, embedded into an LSSD clocking scheme.

This way, the dynamic logic appears encapsulated within clocked macros. The clocked macros are bounded at both input and output by latches, ensuring that all input and output signals to and from the clocked macro are static. This concept sprinkles dynamic logic into a sea of static logic. The analysis methods for timing and logic simulation are those used for static logic.

Figure 30 shows a standard LSSD configuration. All logic paths are bounded by master–slave latches. The cycle starts by launching data from the slave latch with the leading C2 clock edge. The data propagates through the combinatorial logic and is captured with the trailing edge of C1 into the master latch.

Figure 31 shows the logic representation of the embedded clocked macro used for high-level logic simulation and for the static timer. Data is presented to the C2 clocked macro from the C1 clocked master latch Q1. The C1 latch is transparent to the previous path until the trailing edge of C1. The data is received by the input latch with the leading-edge C2 clock, which is the starting point for the signal propagation through the logic.

The output of the macro is captured with the C1 latch Q2. The clocked macro acts as a slave latch for the master latch Q1. The maximum path delay for the clocked macro stretches from the leading edge of C2 to the trailing edge of C1. Combinatorial logic can be inserted between the output of the clocked macro and the receiving latch if the clocked macro does not fill the complete cycle. The restriction of this scheme is that there can be no logic or long wires between the sending latch Q1 and the clocked macro. The clocked macro must be the first component in the cycle. In this ideal scheme, input setup time to the macro is guaranteed by clock separation between the trailing edge of C1 and the leading edge of C2. Static paths and dynamic macros can both be driven from the same sending master–slave latch. The dynamic path is fed from the master, the static path from the slave of the same latch. It is also no problem to combine outputs of the clocked macros with signals from combinatorial paths and capture the result in the receiving master.

The structural realization of the clocked macro using dynamic circuits is shown in **Figure 32**. Data-in and data-out signals are static data in order to allow universal usage of the macro within a static circuit environment. The

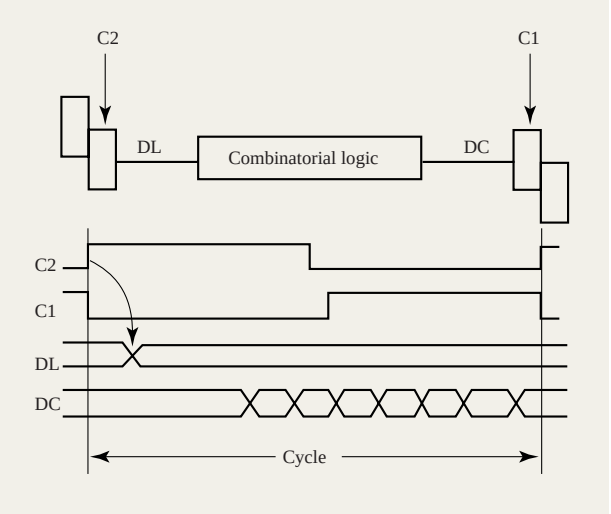


Figure 30

Standard LSSD configuration.

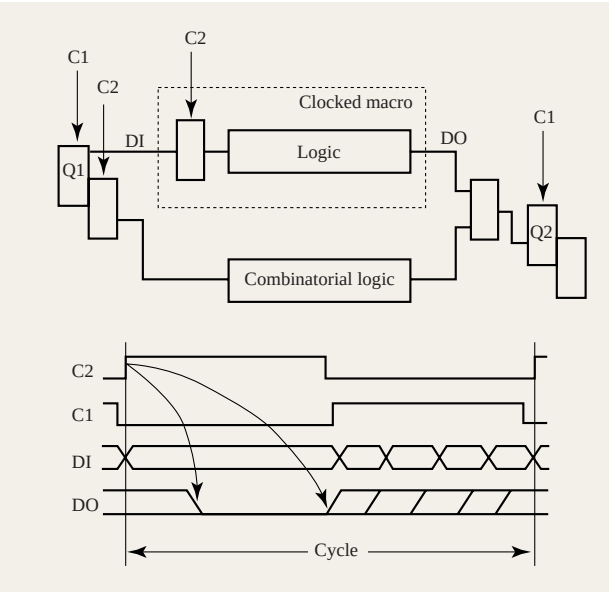


Figure 31

High-level logic model.

macro internal clock is generated through a clock chopper. All internal timing depends on this internal clock, and therefore on the leading edge of the macro clock. External signal timings such as setup, hold, and delay are all specified relative to the leading-edge macro clock. The first input stage of the clocked macro performs the wave

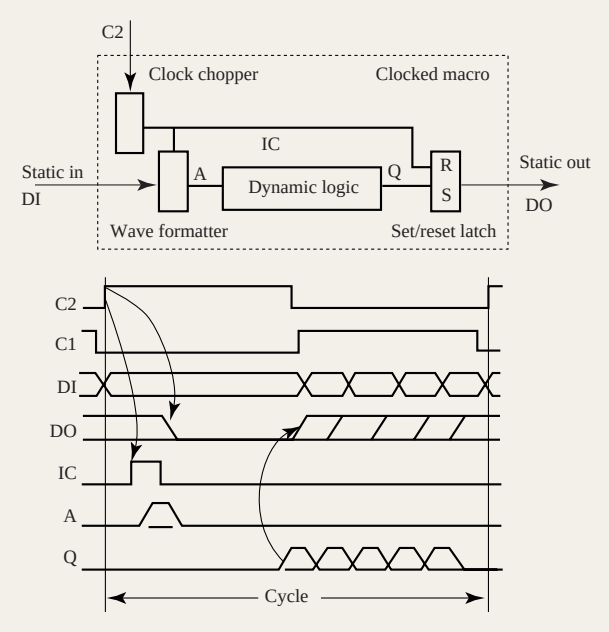


Figure 32

Structural realization.

formatting. Typically this is a true-complement generator, but it can also be a logic stage, gated by the clock. Since the evaluation trees comprise n-MOSFETs, the signals propagating through the dynamic logic have to be formatted to a return-to-zero waveform. The general logic circuit concept is "delayed-reset." The precharge, restore circuitry is default "on," and opens an evaluate window only if requested. The evaluate window is shifted through the cycle, allowing the different circuit stages to evaluate for only that specific time during which the dynamic nodes are kept by the keepers only. The evaluate window is independent of the cycle time. This solves potential leakage problems on dynamic notes operated on a long cycle at high temperature and high voltage. The circuit type used is always the one best fitting the required function [31]. Where possible, biased static circuits benefiting from the return-to-zero waveform are used. The general dynamic circuit type is unfooted domino. The first circuit after long wires is always static in order to be as immune as possible from coupled noise. The output of the clocked macro is driven by a set–reset latch. The latch is reset by the macro clock at the beginning of the cycle and set by the result of the evaluation through the logic path. To prevent false switching of the output latch, the reset pulse can be a combination of macro clock and evaluation request, or it can be timed equivalent to the expected set pulse with the set overwriting the reset.

- *Testability*

The goal of the implementation is to support full LSSD testing, including random self-test, at any cycle time equal to or longer than the functional cycle. An exception is made for scan. Scanning is done at a lower frequency to keep power during test acceptable. The clocked macro can be isolated from the surrounding logic during test, to allow individual test sequences for the clocked macro and the macro-surrounding logic. The latches feeding the clocked macro as well as the S/R macro output latches get an additional slave latch to allow independent scanning. The master as well as the slave scan clock and the macro clock are independent of C1, C2 clocks during test. To test the macro only, the respective stimulus patterns are scanned into the latches driving the macro, and the macro clock is fired. The result is received in the output latch and can be observed after scanning the data out. The macro gets into a defined state after power-on, since the default state of all restore devices is on, precharging all dynamic nodes. No additional test signals, such as reset after power-on, are necessary. Single clocked cycles, such as the typical test sequence consisting of scan-in, execute one cycle, and scan-out, are possible.

- *Functions implemented in hardware*

The macros making up the critical path of the instruction window buffer have been designed and laid out in the IBM 0.18-μm CMOS 8S bulk silicon technology and put on a test site for verification. The design supports a cycle time equivalent to 17 balanced fanout-4 inverters. Circuit design details and measured data are reported elsewhere [32].

The macros that were implemented are the four macros (described above) on the lower part of the IWB, as shown in Figure 29: reservation station data part, reservation station control part, issue window control, and issue filters.

- *Summary*

A method has been discussed that enables the design of a GHz+ instruction window buffer (IWB). The key aspects of designing for such high frequency are the following:

- A design process in which logic and circuit designers work together from the beginning to cover all requirements from microarchitecture down to the transistor level.
- Using distributed control.
- Mapping the logic functions onto regular structures.
- Implementing the logic with array structures using dynamic circuits.

The function of the IWB implementation as suggested here has been successfully demonstrated by hardware. It

has been shown that the chosen design for the instruction window buffer supports very competitive frequencies.

7. A look at the future

In this section we briefly discuss a few topics that will affect microprocessor design in the near future. Power dissipation is a major concern; hence, we examine power efficiency. We also discuss some technology developments that promise to improve performance, but fall outside the domain of improvements due to device and interconnect scaling.

- *Power and power efficiency*

Power dissipation increasingly limits what can be achieved. CMOS power densities are rapidly increasing [33] (**Figure 33**). Many of the packaging and cooling techniques originally developed for bipolar processors are again being applied. The power budget for a microprocessor is becoming a design constraint of similar importance to target technology, target area, and target performance. It is therefore crucial to evaluate circuit techniques and microarchitectural improvements with regard not only to their effect on performance, but also to their net effect on power.

The most effective means available to control power dissipation in a microprocessor is (dynamic) voltage scaling[3] [34]. In a processor with dynamic voltage control, supply voltage and operating frequency are adjusted to the workload. When the frequency is reduced by 1%, the voltage can be reduced by 1% as well, resulting in a 3% decrease in power, since to first order power is proportional to CV^2f. Thus, first-order, 1% performance is worth 3% power. The scaling law holds over the region in which supply voltage and maximum frequency are proportional, which is from about three times the threshold voltage upward. Over this range, $MIPS/W^{1/3}$, or equivalently, *energy-delay*^{2} [35], is a constant, making this an appropriate measure of energy efficiency. This metric favors performance more heavily than the commonly used energy-per-operation and energy-delay [36] metrics. An example of how this works can be found in Figure 23. At high voltage, the Rivina processor does not compare favorably on either the energy or the energy-delay metrics with the Pulsar processor, which dissipates 22 W at 450 MHz [8], yet at 1.3 V, 40 W, and 800 MHz, the Rivina processor provides better performance at equivalent energy per operation.

The rule can be used to guide decisions about circuit design[3] as well as microarchitecture, and applies to frequency as well as IPC. That being said, it is important to realize that unless one is willing to manage multiple power supplies, the scaling can only be applied globally; that is, a design that does not balance the per-stage delays is always wasting energy. In some stages of a pipeline, reducing delays may have a significant enough impact on overall performance to warrant spending the additional power, whereas in other parts of the design a low-power multistage or low-power, low-speed circuit-family solution may be optimal.

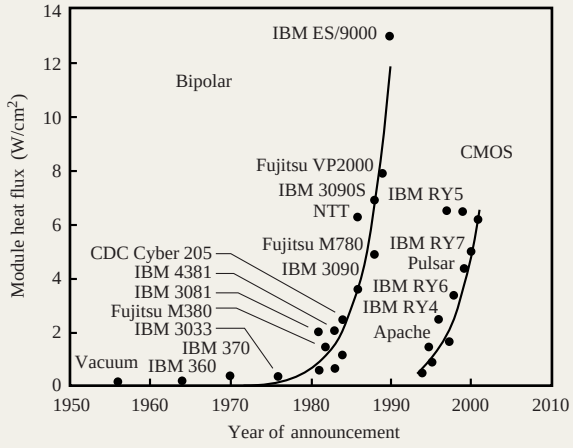


Figure 33

Bipolar and CMOS power densities (courtesy Ghoshal and Schmidt [33]).

- *Technology considerations*

In this section we briefly discuss two other techniques that hold promise for increasing performance in the future.

The first is cooling technology. IBM mainframes are sold in "turbo" [29] versions, in which performance is improved by cooling the modules to improve frequency. Cooling has the additional benefit of dramatically reducing off-currents and improving reliability. Since the off-current is one of the limiting factors in device design, a reduced off-current can be traded for additional performance if one tailors a process to low-temperature operation [37]. Whereas traditionally systems have been cooled by refrigerators that include pumps, which add significantly to overall system cost, advances in thermoelectric cooling [33] hold promise for low-cost, low-temperature systems. It is important to note, however, that a thermoelectric cooler increases the total amount of heat that must be removed from the system. If a (multiprocessor) system is limited by the amount of heat that must be removed from the system, it is not obvious that thermoelectric cooling results in a net performance advantage.

[3] K. Nowka, P. Hofstee, and G. Carpenter, "CMOS VLSI Power Efficiency Under Voltage Scaling Assumptions," *J. Solid-State Circuits*, in press.

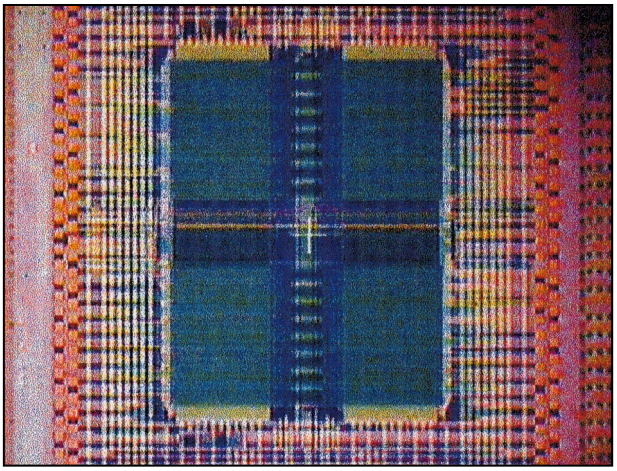


Figure 34

Blazer 1MB fast embedded DRAM macro [38].

If one wants to predict what a next generation of microprocessors might look like, examining the system in which they are used usually gives a good clue. Present-day designs have migrated L2 caches and multiple cores onto the chip. It is thus interesting to examine integrating DRAM on the die with the processor. IBM has demonstrated the capability of combining high-performance DRAM, the next off-chip component, with high-performance logic on the same die [38] (**Figure 34**). When logic transistor performance no longer suffers, the power and density advantages of DRAM make it a good candidate for an on-chip, higher-level cache. For a large enough cache size, a DRAM can even outperform an SRAM in latency because the delay to cover the distance over the array to the processor dominates the time needed to access the subarray. Promising as this technology may be, it is by and large complementary to the improvements that are expected in the core of the processor.

8. Conclusions

The various designs discussed in this paper, from product-quality adders and an instruction window buffer to a complete prototype processor, demonstrate the advantages that can be obtained by increasing the focus on physical design. The work on the Star series commercial microprocessors has shown that it is possible to integrate aggressively designed components in testable and manufacturable products. According to the trends shown in Figure 2, we should expect cycle times, as measured in fanout-4 inverter delays, to dip well below 20 in the near future. We have shown, by prototyping an entire short-pipe processor and the reorder structures needed to implement a more complex microarchitecture, that such cycle times are feasible today if the circuits, logic, floorplan, and microarchitecture are designed hand in hand. We expect engineering and architecting for low power to be a major design consideration in the future. Even though the power dissipation of an aggressively designed microprocessor such as the Rivina prototype processor is large, its power efficiency is in fact competitive with those of previous designs in the same technology.

Acknowledgments

This paper has reported on the work of many groups and a large number of individuals at multiple locations within IBM. Their names can be found on the many IBM papers we have referenced. Without their work this paper could not have been written.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of the Open Group or X/Open Company Ltd.

Alpha, PA-RISC, UltraSPARC, and MIPS, respectively, are registered trademarks of Compaq, Information Technologies Group, Hewlett-Packard Corporation, Sun Microsystems, Inc., and MIPS Technologies, Inc.

References

1. D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, D. Bertucci, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, S. Hassoun, G. Hoeppner, K. Kuchler, M. Ladd, B. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala, and S. Santhanam, "A 200-MHz 64-b Dual-Issue CMOS Microprocessor," *IEEE J. Solid-State Circuits* **27,** No. 11, 1555–1567 (1992).
2. W. Bowhill, S. Bell, B. Benschneider, A. Black, S. Britton, R. Castelino, D. Donchin, J. Edmondson, H. Fair, P. Gronowski, A. Jain, P. Kroesen, M. Lamere, B. Loughlin, S. Mehta, R. Mueller, R. Preston, S. Santhanam, T. Shedd, M. Smith, and S. Thierauf, "Circuit Implementation of a 300-MHz 64-Bit Second-Generation CMOS Alpha CPU," *Dig. Tech. J.* **7,** No. 1, 100–118 (1995).
3. *EinsTimer Users Guide and Language Reference*, IBM Microelectronics Division, Hopewell Junction, NY, 1995.
4. S. Kundu, "GateMaker: A Transistor to Gate Level Model Extractor for Simulation, Automatic Test Pattern Generation and Verification," *Proceedings of the 1998 IEEE International Test Conference*, 1998, pp. 372–381.
5. J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel, "A Multithreaded PowerPC Processor for Commercial Servers," *IBM J. Res. Develop.* **44,** No. 6, 855–898 (2000, this issue).
6. S. Storino, A. Aipperspach, J. Borkenhagen, R. Eickemeyer, S. Kunkel, S. Levenstein, and G. Uhlmann, "A Commercial Multi-Threaded RISC Processor," *ISSCC Digest of Technical Papers*, 1998, pp. 234–235.
7. A. Aipperspach, D. Allen, D. Cox, N. Pham, and S. Storino, "A 0.2μm 1.8v SOI 550MHz 64-Bit PowerPC

Microprocessor with Copper Interconnects," *IEEE J. Solid-State Circuits* **34,** 1430–1435 (1999).
8. T. Buchholtz, A. Aipperspach, D. Cox, N. Phan, S. Storino, J. Strom, and R. Williams, "A 0.18μm 1.5v SOI 660MHz 64-Bit PowerPC Microprocessor with Copper Interconnects," *ISSCC Digest of Technical Papers*, 2000, pp. 88–89.
9. G. Shahidi, A. Ajmera, F. Assaderaghi, R. Bolam, E. Leobandung, W. Rausch, D. Sankus, D. Schepis, L. Wagner, K. Wu, and B. Davari, "Partially-Depleted SOI Technology for Digital Logic," *ISSCC Digest of Technical Papers*, 1999, pp. 426–427.
10. N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, VLSI Systems Series, Addison-Wesley Publishing Co., Reading, MA, 1985, pp. 310–338.
11. H. Ngo, S. Dhong, and J. Silberman, "High Speed Binary Adder Having Inclusive Propagates and Generates," IBM U.S. Patent Pending 0116-AD-AT9-97-509.
12. A. Davies and D. Stasiak, "Method and Apparatus to Prevent Output Noise in NOR Complex Domino Dynamic Circuits," IBM Research Disclosure No. 427, November 1999, pp. 1537–1538; IBM Research Division, Yorktown Heights, NY 10598.
13. D. Allen and D. Stasiak, "Method and Apparatus to Control Noise in a Dynamic Circuit," U.S. Patent 6,002,292, 1999.
14. A. Davies and D. Stasiak, "Method and Apparatus for Using Circuit Input to Reduce Bipolar Current Effects in Dynamic Circuits Designed in Silicon On Insulator," IBM U.S. patent pending, 1999.
15. C. T. Chuang, P. F. Lu, and C. J. Anderson, "SOI for Digital CMOS VLSI: Design Considerations and Advances," *Proc. IEEE* **86,** No. 4, 689–720 (1998).
16. D. Stasiak, J. Tran, F. Mounes-Toussi, and S. Storino, "A 2nd Generation 440ps SOI 64 Bit Adder," *ISSCC Digest of Technical Papers*, 2000, pp. 288–289.
17. J. Silberman, N. Aoki, D. Boerstler, J. Burns, S. Dhong, A. Essbaum, U. Ghoshal, D. Heidel, P. Hofstee, K. T. Lee, D. Meltzer, H. Ngo, K. Nowka, S. Posluszny, O. Takahashi, I. Vo, and B. Zoric, "A 1.0 GHz Single-Issue 64-Bit PowerPC Integer Processor," *IEEE J. Solid-State Circuits* **33,** No. 11, 1600–1608 (1998).
18. P. Hofstee, N. Aoki, D. Boerstler, P. Coulman, S. Dhong, B. Flachs, N. Kojima, O. Kwon, K. T. Lee, D. Meltzer, K. Nowka, J. Park, J. Peter, S. Posluszny, M. Shapiro, J. Silberman, O. Takahashi, and B. Weinberger, "A 1 GHz Single-Issue 64b PowerPC Processor," *ISSCC Digest of Technical Papers*, 2000, pp. 92–93.
19. K. Nowka and T. Galambos, "Circuit Design Techniques for a Gigahertz Integer Microprocessor," *Proceedings of the IEEE International Conference on Computer Design*, 1998, pp. 11–16.
20. R. Heald, K. Aingaran, C. Amir, M. Ang, M. Boland, A. Das, P. Dixit, G. Gouldsberry, J. Hart, T. Horel, W.-J. Hsu, J. Kaku, C. Kim, S. Kim, F. Klass, H. Kwan, R. Lo, H. McIntyre, A. Mehta, D. Murata, S. Nguyen, Y.-P. Pai, S. Patel, K. Shin, K. Tam, S. Vishwanthaiah, J. Wu, G. Yee, and H. You, "Implementation of a 3rd-Generation SPARC V9 64b Microprocessor," *ISSCC Digest of Technical Papers*, 2000, pp. 412–413 and slide supplement.
21. T. Chappell, B. Chappell, S. Schuster, J. Allen, S. Klepner, R. Joshi, and R. Franch, "A 2-ns Cycle, 3.8-ns Access 512-kb CMOS ECL SRAM with a Fully Pipelined Architecture," *IEEE J. Solid-State Circuits* **26,** No. 11, 1577–1584 (1991).
22. R. Haring, M. Milshtein, T. Chappell, S. Dhong, and B. Chappell, "Self-Resetting Logic Register and Incrementer," *IEEE Symposium on VLSI Circuits*, *Digest of Technical Papers*, 1996, pp. 18–19.
23. R. Heald, K. Shin, V. Reddy, I.-F. Kao, M. Khan, W. L. Lynch, G. Lauterbach, and J. Petolino, "64kB Sum-Addressed-Memory Cache with 1.6ns Cycle and 2.6ns Latency," *IEEE J. Solid-State Circuits* **33,** No. 11, 1682–1689 (1998).
24. K. Nowka and J. Burns, "Parallel Condition-Code Generating for High Frequency PowerPC Microprocessors," *IEEE Symposium on VLSI Circuits*, *Digest of Technical Papers*, 1998, pp. 112–115.
25. S. Posluszny, N. Aoki, D. Boerstler, P. Coulman, S. Dhong, B. Flachs, P. Hofstee, N. Kojima, O. Kwon, T. Lee, D. Meltzer, K. Nowka, J. Park, J. Peter, J. Silberman, O. Takahashi, and P. Villarrubia, "Timing Closure by Design; A High-Frequency Microprocessor Design Methodology," *Proceedings of the ACM Design Automation Conference*, 2000, pp. 12–17.
26. P. Restle, T. McNamara, D. Webber, P. Camporese, K. Eng, K. Jenkins, D. Allen, M. Rohn, M. Quaranta, D. Boerstler, C. Alpert, C. Carter, R. Baile, J. Petrovick, B. Krauter, and M. McCredie, "A Clock Distribution Network for Microprocessors," *IEEE Symposium on VLSI Circuits*, *Digest of Technical Papers*, 2000, pp. 184–187.
27. S. Dhong, P. Hofstee, K. Nowka, and J. Silberman, "At Speed Scan Testing," U.S. Patent 6,014,763, 2000.
28. J. Silberman, N. Aoki, N. Kojima, and S. Dhong, "A 1.6 ns Access, 1 GHz Two-Way Set-Predicted and Sum-Indexed 64-kByte Data Cache," *IEEE Symposium on VLSI Circuits*, *Digest of Technical Papers*, 2000, pp. 220–221.
29. C. F. Webb, C. J. Anderson, L. Sigal, K. L. Shepard, J. S. Liptay, J. D. Warnock, B. Curran, B. W. Krumm, M. D. Mayo, P. J. Camporese, E. M. Schwarz, M. S. Farrell, P. J. Restle, R. M. Averill III, T. J. Siegel, W. V. Huott, Y. H. Chan, B. Wile, P. G. Emma, D. K. Beece, C. T. Chuang, and C. Price, "A 400MHz S/390 Microprocessor," *ISSCC Digest of Technical Papers*, 1997, pp. 168–169.
30. G. Northrop, R. Averill, K. Barkley, S. Carey, Y. Chan, Y. H. Chan, M. Check, D. Hoffman, W. Huott, B. Krumm, C. Krygowski, J. Liptay, M. Mayo, T. McNamara, T. McPherson, E. Schwarz, L. Sigal, T. Slegel, C. Webb, D. Webber, and P. Williams, "600MHz G5 S/390 Microprocessor," *ISSCC Digest of Technical Papers*, 1999, pp. 88–89.
31. Kerry Bernstein, "7.x Basic Logic Families," in *Design of High Performance Microprocessor Circuits*, Anantha Chandrakasan, Frank Fox, and William Bowhill, Eds., IEEE Press, Piscataway, NJ, 1999.
32. J. Leenstra, J. Pille, A. Mueller, W. Sauer, R. Sautter, and D. Wendel, "A 1.8GHz Instruction Window Buffer," International Solid-State Circuits Conference 2001 (to appear).
33. U. Ghoshal and R. Schmidt, "Refrigeration Technologies for Sub-Ambient Temperature Operation of Computing Systems," *ISSCC2000 paper TD13.2* (slide supplement), 2000.
34. D. R. Ditzel, "Transmeta's Crusoe: A Low-Power x86-Compatible Microprocessor Built with Software," *Cool Chips III*, April 25, 2000.
35. A. J. Martin, A. Lines, R. Manohar, M. Nyström, P. Penzes, R. Southworth, U. Cummings, and T. K. Lee, "The Design of an Asynchronous MIPS R3000 Microprocessor," *Proceedings of the 17th Conference on Advanced Research in VLSI*, IEEE Computer Society, 1997, 164–181.
36. R. Gonzales and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors," *IEEE J. Solid-State Circuits* **31,** No. 9, 1277–1284 (1996).
37. I. Aller, K. Bernstein, U. Ghoshal, H. Schettler, S. Schuster, Y. Taur, and O. Torreiter, "CMOS Circuit Technology for Sub-Ambient Temperature Operation," *ISSCC Digest of Technical Papers*, 2000, pp. 214–216.

38. O. Takahashi, S. Dhong, M. Ohkubo, S. Onishi, R. Dennard, R. Hannon, S. Crowder, S. Iyer, M. Wordeman, B. Davari, W. B. Weinberger, and N. Aoki, "1 GHz Fully Pipelined 3.7ns Address Access Time 8k*1024 Embedded DRAM Macro," *ISSCC Digest of Technical Papers*, 2000, pp. 396–397.

Received February 3, 2000; accepted for publication October 23, 2000

David H. Allen *IBM Server Technology Development, 3605 Highway 52 N, Rochester, Minnesota 55901 (dhallen@us.ibm.com).* Mr. Allen received the B.S. degree in electrical engineering from the University of Kansas in 1980. He was involved in the design of high-speed memory, logic, and custom circuits at Micron Technology (Boise, Idaho), VTC (Bloomington, Minnesota), and Intel (Portland, Oregon) before joining the IBM Enterprise Server Group in Rochester, Minnesota, in 1992. Since then, he has developed custom 64b PowerPC processors for AS/400 and RS/6000 servers, and has been involved in developing custom circuit techniques, clocking and latch design, and processor design methodology. Mr. Allen is currently leading the implementation of a >1GHz PowerPC processor design in SOI technology.

Sang H. Dhong *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (dhong@us.ibm.com).* Dr. Dhong received the B.S.E.E. degree from Korea University, Seoul, in 1974 and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley in 1980 and 1983, respectively. He joined the IBM Research Division in Yorktown Heights, New York, in 1983 as a Research Staff Member involved in the research and development of silicon processing technology, mainly bipolar devices and reactive ion etching (RIE). From 1985 to 1992, he was engaged in research and development of DRAM cell structures, architectures, and designs spanning many generations of IBM DRAMs from 1 Mb to 256 Mb. Key contributions in this area are high-speed DRAMs, low-power DRAMs, and n-MOS-access transistor trench DRAM cells. After spending three years in development of one of IBM's PowerPC microprocessors as a Branch/I-cache circuit team leader of 15 designers, he worked on a simple but fast processor core based on the PowerPC architecture and on high-speed embedded DRAM (eDRAM) in the Austin Research Laboratory, leading, managing, and growing the high-performance VLSI design group to a peak of 25 people from 1995 to 1999. Their work set a major milestone for the microprocessor industry by prototyping 1GHz PowerPC processors. Also, work on the high-speed eDRAM provided justification for the logic-based eDRAM foundry/ASIC technology offering by IBM as well as the design basis for eDRAM macros of DRAM-like density with SRAM-like high speed. Since becoming the chief technologist of the Austin Research Laboratory in 1999, Dr. Dhong has continued to work in three areas: fast low-power embedded PowerPC, superpipelined multigigahertz PowerPC servers, and high-speed eDRAM. He is a member of the IBM Academy of Technology.

H. Peter Hofstee *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (hofstee@us.ibm.com).* Dr. Hofstee received his doctorandus degree in theoretical physics from the Rijks Universiteit Groningen, The Netherlands, in 1988, and his M.S. and Ph.D. degrees in computer science from the California Institute of Technology in 1991 and 1994, respectively. After two years on the faculty at the California Institute of Technology, he joined the IBM Austin Research Laboratory in 1996. At the Austin Research Laboratory he has participated in the designs of two 1GHz PowerPC prototypes, focusing on microarchitecture, logic design, and chip integration issues. Dr. Hofstee has also participated in numerous studies to help define future server microprocessors, together with the RS/6000, AS/400, and S/390 processor development organizations.

Jens Leenstra *IBM System/390 Division, Boeblingen Development Laboratory, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (leenstra@de.ibm.com).* Dr. Leenstra received his M.S. degree from the University of Twente, The Netherlands, in 1986 and his Ph.D. degree from the University of Eindhoven, The Netherlands, in 1993. In 1988 he joined the Institute for Microelectronics Stuttgart, where he headed the Signal Processing Department. He joined IBM in 1994 at the IBM Development Laboratory in Boeblingen, Germany. Dr. Leenstra has worked in several areas of the S/390 development organization, including logic design and verification of I/O chips, multiprocessor system verification, and processor design. He is currently working on the design of future IBM microprocessors. Dr. Leenstra's current interests focus on computer architecture, high-frequency design, and design for testability.

Kevin J. Nowka *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (nowka@us.ibm.com).* Dr. Nowka received his B.S. degree in computer engineering from Iowa State University in 1986 and his M.S. and Ph.D. degrees in computer science from Stanford University in 1988 and 1995, respectively. In 1996 he joined the IBM Austin Research Laboratory, where he has specialized in VLSI logic and computer arithmetic for application to the design of high-frequency and low-power CMOS processors.

Daniel L. Stasiak *IBM Server Technology Development, 3605 Highway 52 N, Rochester, Minnesota 55901 (stasiak@us.ibm.com).* Mr. Stasiak received his B.S. degree in electrical engineering from the University of Missouri at Rolla in 1987 and his M.S. degree in electrical engineering from the University of Minnesota at Minneapolis in 1990. He joined IBM Rochester in 1987. Mr. Stasiak has used his ASIC and custom design skills in the design of high-performance processors. He is the author of several papers and holds three U.S. patents.

Dieter F. Wendel *IBM System/390 Division, Boeblingen Development Laboratory, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (wendel@de.ibm.com).* Mr. Wendel received his B.S. degree in electrical engineering from the University of Wuerzburg, Germany, in 1981. He joined IBM the same year to work on the very large scale integration fellowship team in Boeblingen. After a two-year assignment from 1984 to 1986 at the IBM Research Laboratory in Yorktown Heights, New York, he joined the S/390 microprocessor development organization at the IBM Boeblingen Laboratory to work in several areas including test, array design, and custom logic design. Mr. Wendel is currently working on the definition of next-generation IBM microprocessors. His current interests focus on concepts in high-frequency circuit design and the exploitation of new technologies.