A 1.0-GHz Single-Issue 64-Bit PowerPC Integer Processor

Joel Silberman, *Member, IEEE*, Naoaki Aoki, David Boerstler, Jeffrey L. Burns, Sang Dhong, Axel Essbaum, Uttam Ghoshal, David Heidel, Peter Hofstee, *Member, IEEE*, Kyung Tek Lee, David Meltzer, *Member, IEEE*, Hung Ngo, Kevin Nowka, *Member, IEEE*, Stephen Posluszny, Osamu Takahashi, *Member, IEEE*, Ivan Vo, and Brian Zoric

***Abstract*—The organization and circuit design of a 1.0-GHz integer processor built in 0.25-μm CMOS technology are presented. A microarchitecture emphasizing parallel computation with a single late select per cycle, structured control logic implemented by read-only-memories and programmable logic arrays, and a delayed reset dynamic circuit style enabling complex functions to be implemented in a few levels of logic are among the key design choices described. A means for at-speed scan testing of this high-frequency processor by a low-speed tester is also presented.**

***Index Terms*— Computer architecture, CMOS integrated circuits, high-speed integrated circuits, integrated circuit design, logic design, microprocessors.**

I. Introduction

GREAT progress in raising the operating frequency of microprocessors has been demonstrated in recent years [1]–[4]. Indeed, the design roadmap of the Semiconductor Industry Association [5] forecasts steadily increasing frequencies exceeding 1000 MHz when 0.18-μm technology becomes the pervasive technology. The challenge in high-frequency processor design is to balance options that maximize both frequency and the useful work performed per cycle.

This paper describes a processor design undertaken to explore high-frequency circuit and microarchitecture options that permit a high clock rate while maintaining the features of machine design that support high performance. Built in 0.25-μm technology (Table I), this 64-bit processor implements a 96-member subset of PowerPC fixed-point instructions that includes compare, logical, arithmetic, and rotate-merge-mask instructions, conditional and unconditional branches, and a variety of loads and stores. Because the subset includes all the commonly encountered instructions, the processor implements over 90% of the instructions actually executed by typical applications. To facilitate high performance, this single-issue prototype supports full forwarding of data for back-to-back execution of dependent instructions, a short four-stage pipeline, and single-cycle execution of core integer operations as well as low-latency cache access.

Manuscript received April 6, 1998; revised June 8, 1998.

J. Silberman, D. Heidel, and D. Meltzer are with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10589 USA (e-mail: lber@us.ibm.com).

N. Aoki, D. Boerstler, J. L. Burns, S. Dhong, A. Essbaum, U. Ghoshal, P. Hofstee, K. T. Lee, H. Ngo, K. Nowka, S. Posluszny, O. Takahashi, and I. Vo are with the IBM Austin Research Laboratory, Austin, TX 78758 USA.

B. Zoric is with IBM Microelectronics, Austin, TX 78758 USA.

Publisher Item Identifier S 0018-9200(98)07041-3.

TABLE I
PROCESS TECHNOLOGY

L(drawn)	0.25 μm
L(effective)	0.15 μm
Tox	4 nm
Wiring layers	6 + local interconnect
Metal	Al
M1 contacted pitch	0.7 μm
M2–M5 contacted pitch	0.9 μm
M6 contacted pitch	1.8 μm
Supply voltage	1.8 V

Measurement of the limiting cycle time for correct operation was performed during wafer probe testing. When a small on-chip read-only memory (ROM) is used as a built-in self-test to exercise the core logic of the processor, correct operation with a 0.9-ns minimum cycle time (25°C, 2.4 V) is observed. If the processor is instead tested by loading a program in the on-chip 4-kByte instruction cache and fetching and executing the stored program, the measured cycle time is 1.0 ns (25°C, 1.8 V), corresponding to a 1.0-GHz operating frequency. This second measurement exercises the core logic as before, but it also characterizes the operation of the instruction cache macro and the instruction fetch path. The same 1-ns cycle-time macro used for the instruction cache was used to implement the data cache. Loading on the output of the data cache, however, increases the processor cycle time when executing programs that include load and store instructions to 1.15 ns (25°C, 1.95 V).

Processor organization and circuit design style contribute to achieving such a high operating frequency. The processor organization emphasizes parallel computation with a single selection point at the end of the cycle to choose the data and controls needed by the processor in the next cycle. An overview of the processor microarchitecture, floorplan, and clock generation and distribution is given in Section II. The use of custom dynamic circuits allows functions to be realized in a small series of complex logic gates with good gate delay and drive capability. In many cases, this circuit style facilitates merging functions for efficient use of area or for a savings in delay. Section III describes the dynamic circuit design

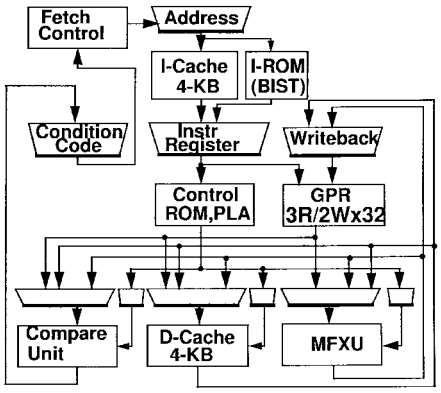


Fig. 1. Logical organization of the processor. This single-issue 64-bit processor implements a four-stage pipeline with full forwarding of results.

style and details functional unit architecture and circuit design choices for a multiplexed-input latch, fixed-point execution unit, and data cache macro. The approach used to support full-speed scan testing and test results appear in Section IV.

II. Processor Overview

A. Organization

The processor organization, shown in block diagram form in Fig. 1, reflects the design choice allowing only a single decision point per cycle to choose among multiple paths computed in parallel. Each cycle-bounding latch in the design includes a multiported input multiplexer. (The latch is described in Section III.) This arrangement maximizes the time allocated for control logic to determine which port to select, and data flow elements that compute the data inputs to the multiplexer/latch can also utilize the full latch-to-latch time for computation and forwarding. The machine processes a single instruction per cycle in a four-stage pipeline consisting of instruction fetch, decode and general-purpose register file (GPR) access, execution, and write backstages. Instructions, predecoded to a 64-bit format, are fetched from either a 4-Kbyte direct-mapped cache or read from a 64-entry ROM programmed with a sequence of instructions for use in a self-test mode. One of these two sources or a hard-wired no-op instruction is selected by the multiplexer (mux) input to the instruction register. Next-fetch-address logic chooses between the next sequential address and a target address in the case of a branch resolved to be taken. The target address can be the result of a branch address calculation or the contents of the link register.

During the decode/GPR cycle, all potential source register address fields within the instruction are used to access the three-read-port, two-write-port 32-entry GPR. At the same time, the source register fields are compared to the target register addresses for the instruction currently executing and for the instruction at the write-back stage of the pipeline. These comparisons are used to forward results needed as operands or to enable a GPR bypass. Simultaneously, one ROM interprets a subfield of the instruction op-code to determine what types of source operands the instruction actually requires: registers or immediate data. The outputs of this 64-entry by 64-bit ROM that indicate that the source operand needed is a register value pass through one additional dynamic AND gate before acting as the mux select input for the operand mux/latch at the end of the decode/GPR cycle. The AND gate conditions the select to reflect whether the required register data is to be obtained from the GPR or forwarding bus.

In parallel with the computation required to route the operands, another ROM interprets a second op-code subfield to determine the type of operation the instruction represents. This second ROM then produces the execution controls required for each operation. These controls are also latched at the end of the decode cycle. As a result, all control signals needed by the execution units are available from latches at the beginning of the cycle, and the execution unit designers could plan their paths with this in mind. Additional decoding made use of a small programmable logic array (PLA) built using precharged NOR gates with strobed outputs for the AND plane and footless dynamic OR gates to generate the PLA output.

Instructions execute in one of three execution units. The fixed-point unit executes arithmetic, rotate-mask-and-merge, and logical operations in a single cycle. The data cache unit implements the address generation, cache access, and data alignment needed to execute load and store instructions as single-cycle operations. A third unit, the compare unit [6], executes explicit compare operations, setting the condition code bits in the condition register to reflect whether one of the two operands compared is greater than, less than, or equal to the other operand. This unit also supports the efficient implementation of the recording form of arithmetic operations. These recording operations both produce an arithmetic result and update the condition register to reflect how the computed value compares to zero. Rather than perform the comparison after the arithmetic result is calculated, the compare unit computes the correct condition register bits in parallel with the production of the arithmetic result in the fixed-point unit. Accelerating the update of the condition register improves the delay in resolving conditional branches that depend on a preceding recording arithmetic instruction.

B. Floorplan

The processor was floorplanned as a single data flow (see Fig. 2) anchored at one end by the instruction cache and at the other by the data cache. The design comprises approximately one million transistors and occupies a region 1.6 by 6.1 mm within the 7 by 8 mm test-pad cage. When operating at 1.0 GHz, the chip consumes 6.3 W. The position of individual components in the design was manually adjusted during an iterated process involving placement, routing, and global timing. The timing optimization included insertion of inverter pairs to repower heavily loaded signals. Control of coupled noise played a major role in the use and placement of these buffers. Macro outputs that required buffering to drive across the chip were repowered with a single inverter near the source end and inverted again with a receiver floorplanned near the sink of the net. The receiver protects the noise-sensitive inputs of dynamic circuits. Within a macro, wire spacing, inverter ratios, and half-latch strength are chosen to mitigate coupled noise.

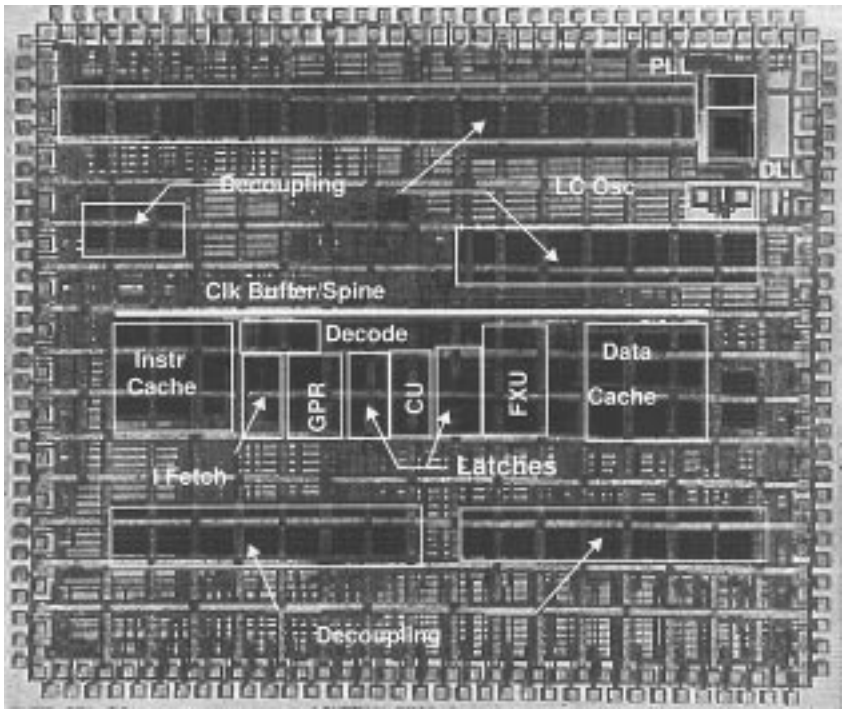


Fig. 2. Chip micrograph and overlay indicating the location of major elements in the processor. Test-pad cage is 7 by 8 mm and includes the 10-mm^2 processor, clock generation circuitry, and decoupling capacitors.

C. Clocking

A variety of clock sources were integrated within the test-pad cage, including a phase-locked-loop (PLL), delay-locked loop, separate analog and digitally controlled voltage-controlled oscillators (VCO's), and a fixed-frequency inductor–capacitor oscillator, as well as receivers for the direct input of a high-frequency external clock. The PLL design [7] uses a delay-interpolating VCO to synthesize the processor clock. A differential design for the clock signal and control paths provides high noise immunity. Voltage-controlled current sources reduce the common mode response of the VCO, and replica biasing techniques reduce the VCO process and temperature sensitivity. With the processor actively executing instructions at a 1.046-GHz operating frequency, the rms jitter of the output of a divide-by-16 circuit driven by the clock was measured to be 10.64 ps using an HP58720D oscilloscope. In a quiet, stand-alone environment, measurements with an HP54120 oscilloscope directly on the buffered output of the PLL established an rms jitter of 2.9 ps. The PLL has a nominal lock range of 2 : 1 with measured maximum frequency of 1360 MHz for the integrated PLL with the processor active and 1560 MHz in the stand-alone configuration [7]. The output of the clock generator was routed differentially to a clock buffer located in the center along one edge of the processor data flow. This buffer drove a 51-μm-wide clock spine the length of the processor. Individual macros that used the clock tapped onto the clock spine and buffered the global clock with a single inverter to form the local clock signal. The effect of clock skew on cycle time was minimized by iteratively tuning the width of the wires from the spine to the macro based on global timing runs with extracted wire delays.

III. Functional Unit Circuit Design

The circuit design style used throughout the processor is based on delayed or cascaded reset dynamic circuits. A short chain of such gates is shown in Fig. 3(a). The global clock, NCLK, gates the output of the cycle bounding latch and launches a computation down a logic path such as the one shown. The latch outputs are dual-rail dynamic signals. If the logic equation represented by the NFET pull-down path of the gate whose output is labeled A in Fig. 3(a) is true, the precharged dynamic node falls and output A rises. This, in turn, may trigger the next gate, and the computation propagates through the logic to produce the macro output D. A one-shot circuit triggered by the same clock edge that launched the data from the latch produces a low-going pulse at some delay after NCLK falls. The delay is chosen such that if output A rose in the cycle, the output was high long enough to robustly switch the next gate. This low-going pulse resets the first gate in the chain, so that output A falls. The reset pulse is propagated down the logic path through a chain of inverters timed so that the reset signal is applied to each subsequent gate only after the inputs to the gate have returned to ground. The reset pulse must be long enough to insure that the dynamic node returns fully to the power rail and short enough that the reset signal is off before the next data input to the gate arrives.

One important consequence of the fact that both the rising and the falling edges of the dynamic gate output are determined by the clock is that the pulses produced by the gate are of fixed duration, independent of cycle time [see Fig. 3(b)]. The path is designed so that the pulsed output of the gate that drives the cycle bounding latch meets both the setup and the hold-time requirements of the latch at minimum cycle time.

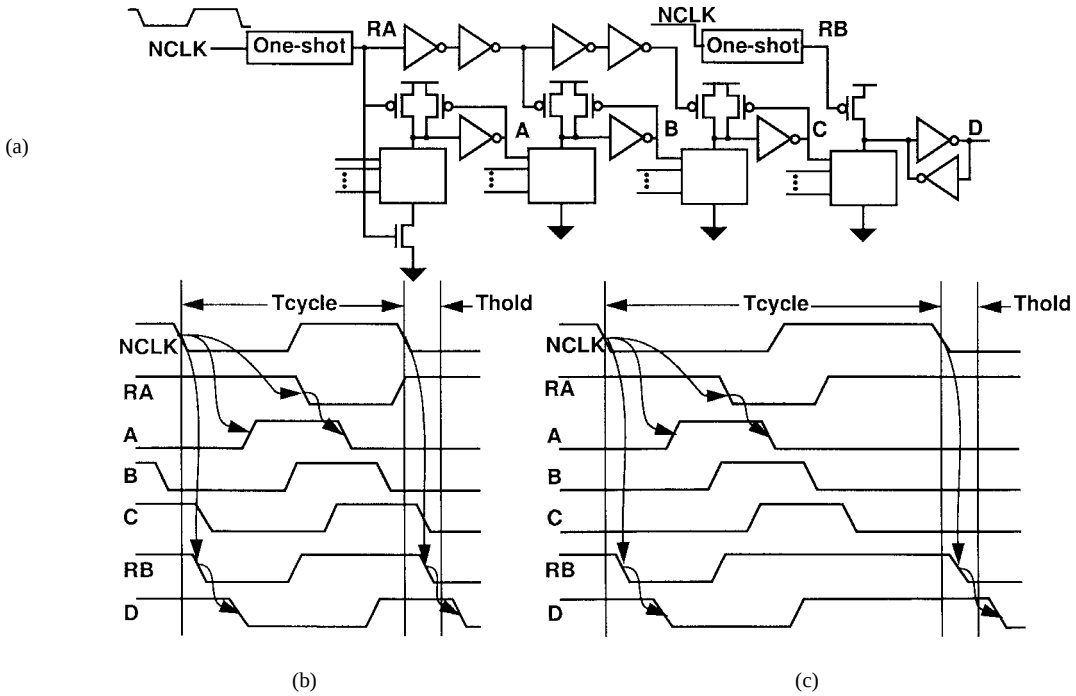


Fig. 3 (a) Illustration of cascaded reset dynamic circuit style used throughout this processor. (b) Global clock (NCLK) and output A–D waveforms produced as the computation and reset propagate through the logic chain and reset chain. (c) The last stage of logic is reset by the next clock edge so that output D stretches to meet the latch hold time as the cycle time increases.

At longer cycle times, these requirements would not be met if the pulse width were fixed. To overcome this limitation, the last gate in a chain of logic is not reset by the delayed reset pulse described above. Rather, the reset pulse for the last gate is derived from the clock edge that triggers the receiving latch. In this way [Fig. 3(c)], the output stretches as the cycle time increases to guarantee that the latch hold-time requirement is satisfied.

The key advantage of cascading the resets in this way is that most gates in the path can be built without a "foot" or ground-interrupt device, allowing this device to be used as an extra logic device in the NFET pull-down path or removed to obtain higher speed. Additionally, the precharge current is drawn from the supply throughout the cycle rather than only at the clock edges, as is the case with conventional domino dynamic circuits, reducing peak current demand. [8], [9]. The dynamic multiplexed-input latch, fixed-point unit, and data cache unit built with this circuit style are described below.

A. Multiplexed-Input Latch

Besides providing a multiplexing function to support the late-select-based machine organization, the mux/latch generates dual-rail outputs from single-rail inputs to drive dynamic circuits, provides a static output for latch-to-latch paths, and supports full scan testing. The leaf cell schematic and control circuitry for the cycle-bounding multiplexed-input latch are given in Fig. 4. The leaf cell consists of two dynamic 8 : 1 muxes with common select signals together with latching elements. The two muxes share a foot device. Depending on whether the single-rail data input to the selected path is high or low, either true output (OT) or complement output (OC) produces an active high pulse to drive downstream

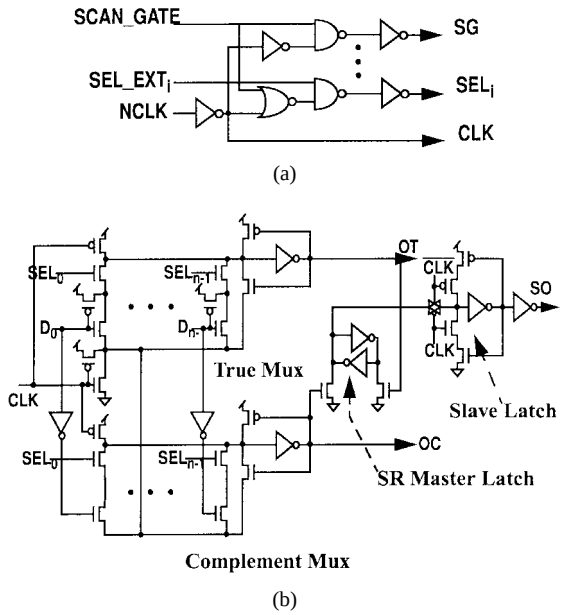
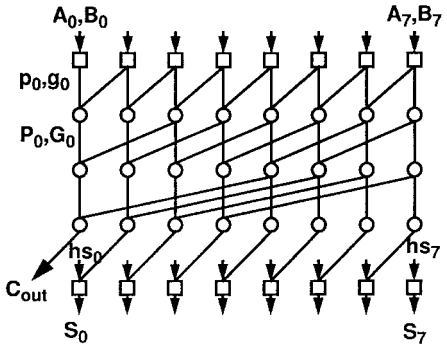
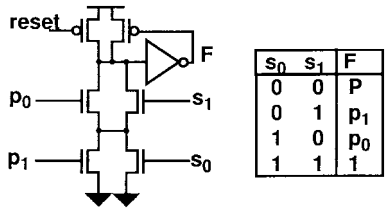


Fig. 4 (a) Register control circuitry for multiplexer/latch. The external input SEL_EXT_i activates multiplexer port i. SCAN_GATE chooses the scan-in port and has priority over other select inputs. (b) Multiplexer/latch leaf cell.

logic. These two signals also set or reset a master latch. The contents of the master latch are transferred to the slave latch at midcycle to produce a static output. By combining the function of a multiplexer with the latch—and, in particular, because the functional path through the macro is through the mux only—latching has very little cycle-time impact. The use of single-rail inputs instead of dual-rail reduces the number of global wires required for forwarding results and allows computations to produce a single-rail result, saving area. In



(a)



(b)

Fig. 5. (a) Cell-level diagram of an 8-bit carry look-ahead adder. Because the wiring of the three middle merge cells is the same as that of a shifter, the same network can be used for both addition and rotation functions. (b) Dynamic 2-bit propagate merge circuit modified by the addition of devices controlled by s_0 and s_1. As the truth table illustrates, this gate can be used either in an adder (propagate function) or as a multiplexer for a shifter.

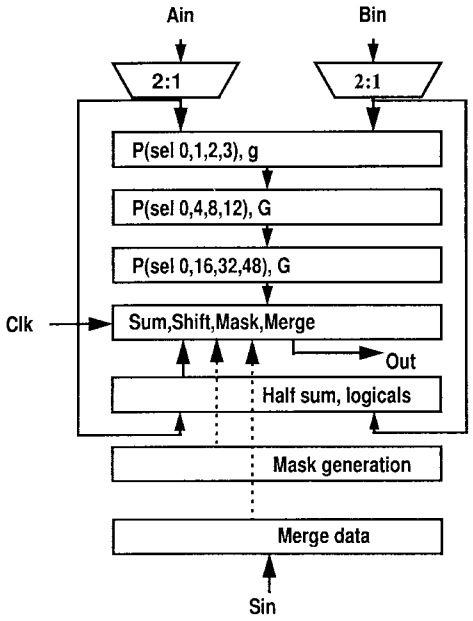


Fig. 6. Architecture of the fixed-point unit, which implements arithmetic, rotate-mask-and-merge, and logical operations. The last gate in the unit is reset by the clock to produce a pulse that stretches with cycle time.

addition, many of the paths in the processor contain strobed decoders that take advantage of the low bit-to-bit skew in the dual-rail outputs of the mux/latch.

Although the mux/latch is built with eight input ports, six are available as general-purpose inputs and two have dedicated uses. One mux port is used to implement an explicit hold function, and the static output is fed back to the corresponding data input for this port to accomplish this. A second mux port is dedicated to support scan testing and serves as the scan input to the latch. In this case, the static output is the scan output from the bit. The scan chain runs from least significant bit to most significant bit (MSB) in each register, and the scan output of the MSB is buffered and driven to the next element in the chain. The mux/latch toggles between scan and functional modes under the control of the globally distributed SCAN_GATE signal. This macro input to the control circuitry [see Fig. 4(a)] takes priority over all other selects, ensuring that the scan input is launched on OT and OC and latched in each leaf cell.

B. Fixed-Point Execution Unit

The 64-bit fixed-point unit achieves 550-ps delay by exploiting the powerful complex gates that the cascaded reset circuit style affords and by merging together rotation and addition functions in a single circuit. The basis for combining these functions is illustrated in Fig. 5(a), which shows a symbolic representation of an 8-bit adder constructed according to the parallel prefix algorithm of Kogge and Stone [10], [11]. This carry look-ahead adder organization results in cells with low fan-out and highly uniform size. In Fig. 5(a), the first row of cells produces the usual propagate and generate signals for each bit of the input operands A and B. These are then merged two bits at a time in the next three rows of cells to produce the carry-out at each bit position, which is then combined with the half-sum to produce the sum output. The low fan-out per cell is attractive for high speed.

Because the signal routing in the merge portion of the adder in Fig. 5(a) is the same as required by a logarithmic shifter, it is possible to build a special merge cell that allows both addition and shifting functions with the same network. Such a cell and the corresponding truth table for the gate are shown in Fig. 5(b). A 2-bit propagate function built as a dynamic gate requires six of the transistors shown in the figure. By the addition of the two devices whose gates are controlled by inputs s_1 and s_0, the circuit can be used to implement the propagate function (when both s_0 and s_1 are low) or to select input p_0 or p_1 (by the exclusive application of s_0 or s_1, respectively).

The attraction of modifying an adder of the form shown in Fig. 5 to perform both shifting and addition functions is that both the devices and the wires can be shared between the two operations. In addition, producing either result with the same macro saves delay by eliminating the need for an explicit result mux in the path. To implement a full rotator, the merging network needs to be supplemented with the wires to bring bits from left to right, and the input and sum stages must be suitably modified to support both operations.

Fig. 6 diagrams the resulting architecture of the fixed-point execution unit. Input operands A and B are conditionally complemented through 2 : 1 muxes to support subtraction and a variety of logical instructions. Merging for addition and selection for rotation are accomplished in 4-bit groups. The first row of merge circuits computes group propagates (or rotations) but bit-wise generates. Staggering the generate and propagate circuits in this way enables replacing a four-high stack of NFET's needed to compute product terms such as $p_0 \cdot$

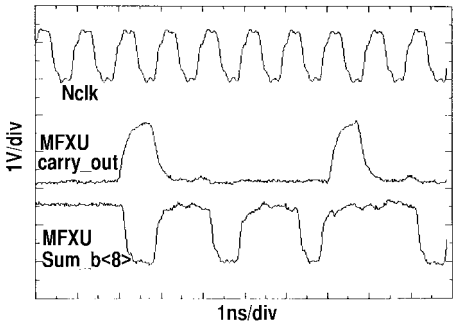


Fig. 7. Measured waveforms at 1 GHz of the global clock, fixed-point unit carry-out, and inverted sum output (bit 8). The falling edge of NCLK represents the start of the cycle. The delay from NCLK to the adder output includes buffering by a single inverter to generate the local clock, launch of data from a mux/latch, the delay through the 2:1 mux, and the adder.

$p_1 \cdot p_2 \cdot g_3$ by two devices in series, since the product of p's was evaluated earlier in the path. This reduction in stack height, in turn, allows the calculation of the carry-out for each bit to be combined with production of the sum output, resulting in a total of four dynamic gates in the path through the adder. For rotate-mask-and-merge instructions, mask generation occurs in parallel to rotation of the data. Logical operations reach the macro output through the half-sum path. The final stage of the macro is reset by the clock to produce a stretched pulse, ensuring that the output is captured by the receiving latch. The measured carry-out and inverted sum output waveforms plotted together with the global clock signal in Fig. 7 show operation of the fixed-point unit at a frequency of 1 GHz.

C. Data Cache

Like the fixed-point unit, the data cache exploits merging functions to achieve low latency for the tasks of address generation, cache access, and data alignment. To reduce the delay in address generation, the design uses a carry propagation-free adder in place of the binary adder normally used. Although such an adder produces an output that is not a unique representation of the sum of its inputs, combining such an adder with a decoder results in the capability to address a memory location by the sum of the two input operands [12] and reduces delay over a conventional adder and decoder [13]. Cache-access latency was reduced through the use of separate read and write ports and single-ended, full-swing readout from a seven-transistor, two-port cell. A full 32-byte line was accessed in both even and odd word-line banks [12]. Bank selection, column selection, and flexible routing functions were merged to a single level of parallel 64:1 muxes, one per output byte.

The access path for this direct-mapped 4-Kbyte cache is illustrated in Fig. 8. Two input operands are launched from mux/latches to start a load or store execution cycle. The lower 12 bits of these inputs are combined in the carry propagation-free adder realized as a single dynamic gate. The signals corresponding to the upper six of the 12 bits are used for row decoding, while the remaining bits and additional signals designating the number of requested bytes and other output formatting controls are used for column selection and output alignment control. As depicted in Fig. 8, the output of the carry propagation-free adder drives a self-strobing NOR decoder to activate the read word line. Four SRAM cells are dotted onto the precharged local bit line, which falls or remains precharged depending on whether the addressed cell contains a one or zero. Two such local bit lines are ORed in a NAND gate to drive an NFET connected to a precharged global bit line. The eight-way dotting on this node results in a total of 64 cells per column per bank. A source-driven 64:1 multiplexer selects one byte from the 32-byte line from either the even or odd bank and routes it to the output. Eight copies of the mux are used to generate the full double-word output. This mux is built hierarchically like the read bit-line path, so that fast precharged ORing predominates. The final output stage is reset by the clock to produce a stretched pulse.

Stores are executed in a read-modify-write manner. During the execution cycle of a store, the target line is read from the cache, the store data are aligned to the proper location and merged with the read data, and the modified line is latched. The active word line in the addressed bank is also latched in decoded form. At the start of the next cycle, the write word line at the latched location is activated and the modified line driven on the differential, static bit lines. The cell is written early in the cycle before the next read operation so that loads to the same address can follow stores without conflict. The use of a read-modify-write approach rather than selective writing avoids disturbance of internal cell nodes in unwritten cells, which would degrade read access in the case of loads following stores that address the same cache line.

IV. At-Speed Scan Testing

All scan paths on the processor as well as the distribution network for SCAN_GATE were designed so that data could be scanned at the full operating frequency of the chip. Within a register, this requirement was easily satisfied since the scan output of a bit only needs to drive the input to the adjacent bit. When the propagation delay of the scan signal along a long wire between registers made the scan path exceed the cycle time, a clocked 1-bit register was simply inserted in the wire. The design to scan at speed was motivated by the desire to perform single- and multiple-cycle ac testing with the clock and circuitry toggling and drawing power at the rate characteristic of the actual operating environment. A flexible means of wafer probe testing the part was developed to allow scan testing of the high-speed processor using a lower speed tester [14]. Scan data were brought onto the chip through a 16-bit parallel-to-serial interface allowing data to be scanned into the chip at 1 GHz but only requiring the tester to provide 16 bits of data at a rate of 62.5 MHz per pin. Similarly, the register contents were scanned from the chip into a serial-to-parallel converter and interpreted by the tester at a rate that was one-sixteenth of the processor clock rate. To synchronize these operations, the high-frequency processor clock was divided on-chip by 16 and driven to the tester. The tester was locked to this lower frequency signal.

Control of scan versus functional mode by the global signal SCAN_GATE also relied on parallel-to-serial conversion to enable the slower tester to precisely control the toggling of operating mode. The tester delivered to the processor a bit vector defining the SCAN_GATE signal for the next 16

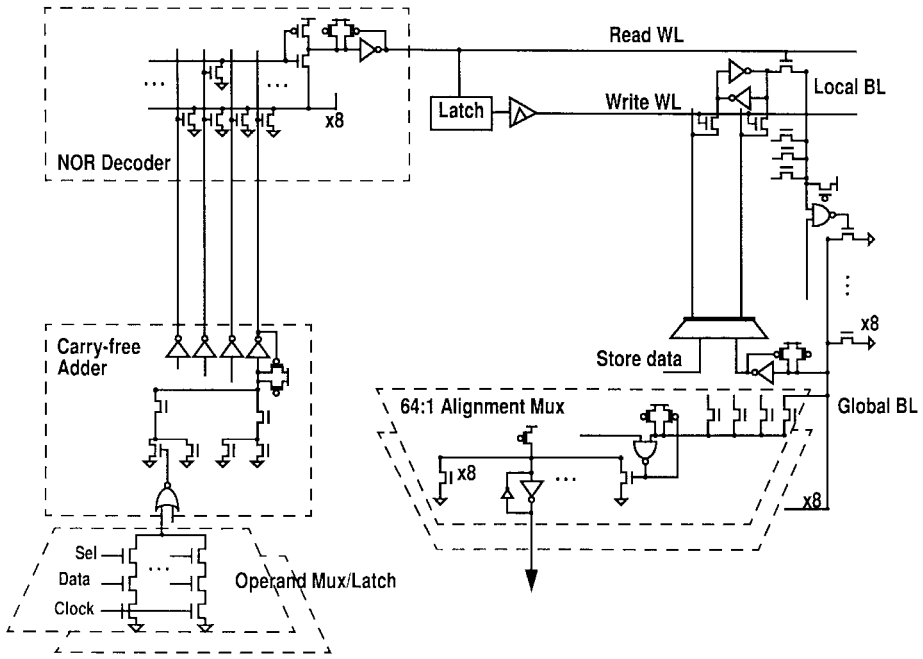


Fig. 8. Illustration of the cache-access path, which includes address generation, array access, and data alignment in a single cycle.

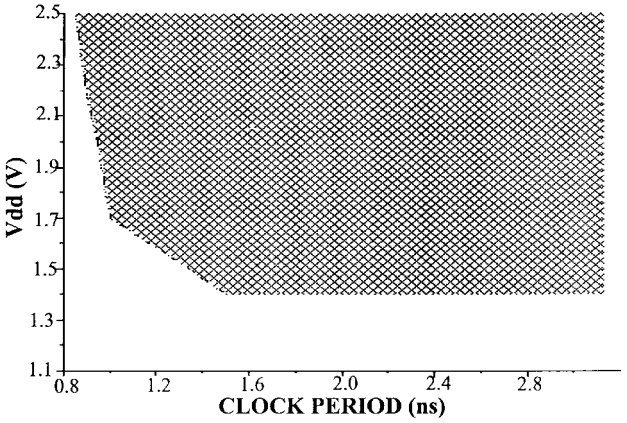


Fig. 9. Shmoo plot indicating the operating region of the processor core logic and forwarding paths exercised by a short loop of instructions fetched from a fast, on-chip ROM.

cycles through a second parallel-to-serial interface. The serial output of the parallel-to-serial converter used to receive this control vector was then buffered by a fan-out tree of latches to distribute the global SCAN_GATE control simultaneously to the control inputs of all latches in the processor. This method of establishing the operating mode means that the sub-nanosecond timing required to synchronize the processor operation was accomplished within chip itself, greatly relaxing the precision required of the tester. An external write enable signal asserted to load a 256-bit instruction cache line from a scannable register to the cache was synchronized to the scan data through a third 16-bit parallel-to-serial input in the same way as the SCAN_GATE signal.

V. Conclusion

With this means of loading cache and register data, executing any number of functional cycles, and inspecting the latched results, detailed testing of the hardware with various frequencies, supply voltages, and patterns was readily accomplished. Fig. 9 presents a shmoo plot of correct operation for the core logic and forwarding paths of the chip when exercised by a short loop of instructions read from the on-chip instruction ROM. Demonstrated operation at 1.0 GHz results from the combination of delayed-reset custom dynamic circuits, merged functionality in latches and execution units, and a machine organization exploiting parallel computation with a single selection per cycle. These design choices enabled the implementation of a 64-bit single-issue integer processor supporting full forwarding with a four-stage pipeline in 0.25-μm technology.

Acknowledgment

The authors are indebted to K. Jenkins, M. Immediato, R. Robertazzi, and K. Stawiasz for their expertise and dedication in testing the hardware.

References

[1] B. A. Gieseke, R. L. Allmon, D. W. Bailey, B. J. Benschneider, S. M. Britton, J. D. Clouser, H. R. Fair III, J. A. Farrell, M. K. Gowan, C. L. Houghton, J. B. Keller, T. H. Lee, D. L. Leibholz, S. C. Lowell, M. D. Matson, R. J. Matthew, V. Peng, M. D. Quinn, D. A. Priore, M. J. Smith, and K. E. Wilcox, "A 600 MHz superscalar RISC microprocessor with out-of-order execution," in *ISSCC Dig. Tech. Papers*, Feb. 1997, pp. 176–177.

[2] J. Schutz and R. Wallace, "A 450 MHz IA32 P6 family microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 1998, pp. 236–237.

[3] N. Rohrer, C. Akrout, M. Canada, D. Cawthron, B. Davari, R. Floyd, S. Geissler, R. Goldblatt, R. Houle, P. Kartschoke, D. Kramer, P. McCormick, G. Salem, R. Schulz, L. Su, and L. Whitney, "A 480 MHz RISC microprocessor in a 0.12 μm Leff CMOS technology with copper interconnects," in *ISSCC Dig. Tech. Papers*, Feb. 1998, pp. 240–241.

[4] 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. Slegel, W. V. Huott, Y. H. Chan, B. Wile, T. N. Nguyen, P. G. Emma, D. K. Beece, C.-T. Chuang, and C. Price, "A 400-MHz S/390 microprocessor," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1665–1675, Nov. 1997.

[5] Semiconductor Industry Association, "The national technology roadmap for semiconductors," 1998, p. 40.
[6] J. Burns and K. Nowka, "Parallel condition-code generating for high frequency PowerPC microprocessors," in *1998 Symp. VLSI Circuits Dig. Tech. Papers*, to be published.
[7] D. Boerstler and K. Jenkins, "A phase-locked loop clock generator for a 1 GHz microprocessor," in *1998 Symp. VLSI Circuits Dig. Tech. Papers*, to be published.
[8] T. I. Chappell, B. A. Chappell, S. E. Schuster, J. W. Allen, S. P. Klepner, R. V. Joshi, and R. L. Franch, "A 2-ns cycle, 3.8-ns access 512-kb CMOS ECL SRAM with a fully pipelined architecture," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1577–1584, Nov. 1991.
[9] R. A. Haring, M. S. Milshtein, T. I. Chappell, S. H. Dhong, and B. A. Chappell, "Self-resetting logic register and incrementer," in *1996 Symp. VLSI Circuits Dig. Tech. Papers*, June 1996, pp. 18–19.
[10] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. C-22, pp. 786–793, Aug. 1973.
[11] K. Suzuki, M. Yamashima, T. Nakayama, M. Izumikawa, M. Nomura, H. Igura, H. Heiuchi, J. Goto, T. Inoue, Y. Koseki, H. Abiko, K. Okabe, A. Ono, Y. Yano, and H. Yamada, "A 500 MHz 32 bit, 0.4 μm CMOS RISC processor," *IEEE J. Solid-State Circuits*, vol. 29, pp. 1464–1473, Dec. 1994.
[12] Y.-H. Lee and S. H. Hwang, "Address addition and decoding without carry propagation," *IEICE Trans. Inform. Syst.*, vol. E380-D, no. 1, pp. 98–100, Jan. 1997.
[13] R. Heald, K. Shin, V. Reddy, I.-F. Kao, M. Khan, W. Lynch, G. Lauterbach, and J. Petolino, "64 kB sum-addressed-memory cache with 1.6 ns cycle and 2.6 ns latency," in *ISSCC Dig. Tech. Papers*, Feb. 1998, pp. 350–351.
[14] D. Heidel, S. Dhong, P. Hofstee, M. Immediato, K. Nowka, J. Silberman, and K. Stawiasz, "High speed serializing/de-serializing design-for-test method for evaluating a 1 GHz microprocessor," in *Proc. VLSI Test Symp.*, Apr. 1998, to be published.

Joel Silberman (M'97) received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1986.

That year, he joined the IBM T. J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member. He has been involved in high-frequency microprocessor design since 1993, including a two-year assignment at the IBM Austin Research Laboratory.

Naoaki Aoki received the B.S. and M.S. degrees in communication engineering from Osaka University, Osaka, Japan in 1987 and 1989, respectively.

He joined IBM's Yasu Technology Application Laboratory, Shiga, Japan, in 1989, where he was leading the development of LCD drivers. In 1996, he joined Austin Research Laboratory, Austin, TX. Since then, he has been working in the High-Performance VLSI Group.

Mr. Aoki is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

David Boerstler received the B.S. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, and the M.S. degree in computer engineering and in electrical engineering from Syracuse University, Syracuse, NY.

Since joining IBM in 1978, he has held a variety of assignments, including responsibility for the design of high-frequency phase-locked loops for clock generation and recovery, fiber optics, and other bipolar and CMOS circuit development projects. He currently is a Research Staff Member with the High-Performance VLSI Group at the IBM Austin Research Laboratory.

Jeffrey L. Burns received the B.S. degree in engineering from the University of California, Los Angeles, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley.

In 1988, he joined the IBM T. J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member, where he worked in the areas of IC layout automation and microprocessor design. Since 1996, he has been with the IBM Austin Research Laboratory, Austin, TX. His current research interests are in computer-aided design for custom design and circuits and microarchitectures for high-performance microprocessors.

Sang 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, Berkeley, in 1980 and 1983, respectively.

In 1983, he joined IBM's Research Division in Yorktown Heights, NY, as a Research Staff Member, where he was involved with the research and development of silicon processing technology, mainly bipolar devices and reactive ion etching. From 1985 to 1992, he was engaged in research and development of DRAM designs, spanning over many generations of IBM DRAM's, from 1-Mb DRAM's to 256-Mb DRAM's. After spending three years in development of one of IBM's PowerPC chips as a Circuit Designer, he currently is working on a simple but fast processor core based on the PowerPC architecture in the Austin Research Lab of the IBM Research Division, managing the high-performance VLSI design group.

Axel Essbaum received the B.S. degree in computer science from Carnegie-Mellon University, Pittsburgh, PA.

He joined IBM in 1992. He has worked on programmable logic array design and design automation software, logic synthesis, layout automation tools, and custom layout.

Mr. Essbaum received an Outstanding Technical Achievement Award in 1996 for his role in developing a dynamic synthesis methodology.

Uttam Ghoshal received the B.Tech. degree from the Indian Institute of Technology, Bombay, and the Ph.D. degree in electrical engineering from the University of California (UC), Berkeley, in 1994.

He was a Member of Technical Staff with the Microelectronics and Computer Technology Corporation, Austin, TX, for five years, where his work focused on methods for extracting interconnection circuit parameters and signal-integrity issues in multichip modules. From 1994 to 1995, he was a Member of Technical Staff with Conductus, Inc., and a Visiting Research Fellow at UC Berkeley. He joined the High-Performance VLSI Group at the IBM Austin Research Laboratory as a Research Staff Member in 1996. His current interests include CMOS circuit design for low-temperature operation and design of novel solid-state coolers.

David Heidel received the B.S. degree in physics from Miami University, Oxford, OH, in 1974 and the M.S. and Ph.D. degrees in physics from The Ohio State University, Columbus, in 1976 and 1980, respectively.

In 1980, he joined IBM's T. J. Watson Research Center, Yorktown Heights, NY, working on Josephson technology. Since 1984, he has been working on the design and testing of high-speed semiconductor devices and circuits. He is presently Manager of the High Speed Analog/Digital Test Group within the Research Division of IBM.

Peter Hofstee (M'96) received the Drs. degree from Rijks Universiteit Groningen, The Netherlands, and the M.S. and Ph.D. degrees from the California Institute of Technology, Pasadena.

He is a Research Staff Member with the IBM Austin Research Laboratory and a member of the High-Performance VLSI Group.

Kyung Tek Lee received the B.S. degree in electronics engineering from Seoul National University, Korea, and the M.S. degree in electrical and computer engineering from the University of Texas at Austin, where he currently is pursuing the Ph.D. degree.

He has been with the High-Performance VLSI Design Group at the IBM Austin Research Laboratory since 1996. He is working on logic and circuit design. He also has interests in timing verification and delay testing in VLSI digital circuits.

David Meltzer (S'70–M'71) received the B.E.E. degree from Rensselaer Polytechnic Institute, Troy, NY, and the M.S. and Ph.D. degrees from The Ohio State University, Columbus, all in electrical engineering.

He is a Research Staff Member in the VLSI Systems Group at the IBM T. J. Watson Research Center, Yorktown Heights, NY. He joined IBM in the Poughkeepsie, NY, product development laboratory and worked on the logic design, microarchitecture, performance, and instruction set architecture of several System/360, 370, and 390 products. Prior to joining the IBM Austin Research Lab, he worked on circuits and microarchitectures appropriate to the design of CMOS processors operating at low temperature (70 K). In Austin, he focused on the design of high-frequency microprocessors. His research interest continues to be the design of high-performance microprocessors, especially the interaction among circuits, instruction sets, and microarchitectures. He has received nine patents and currently has five additional patent applications on file.

Hung Ngo received the B.S. degree in computer science from New York University, the B.E. degree in electrical engineering from Cooper Union, New York, and the M.S. degree in computer engineering from Syracuse University, Syracuse, NY.

He has been involved as a Logic Designer and Circuit Designer for several system S/390 processor projects. He joined the IBM Austin Research Laboratory in 1996.

Kevin Nowka (S'84–M'85) received the B.S. degree from Iowa State University, Ames, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA.

He is a Research Staff Member with the IBM Austin Research Laboratory and is a Logic and Circuit Designer in the High-Performance VLSI Group.

Stephen Posluszny received the B.S. and M.S. degrees in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1981 and 1982, respectively.

He joined IBM, Fishkill, NY, in 1982, working in the computer-aided design organization on silicon compaction tools. In 1988, he transferred to IBM in Austin, TX, where he focused on logic synthesis tools. He joined the IBM Austin Research Lab in 1995 as a member of the Computer Aided Design and Analysis Group and is currently an IBM Senior Engineer. His current research interests include high-frequency methodologies, static timing analysis, and chip integration tools.

Osamu Takahashi (M'97) received the B.S. and M.S. degrees from the University of California, Berkeley.

He is a Staff Engineer with the IBM Austin Research Laboratory and is a Member of the High-Performance VLSI Group.

Ivan Vo received the B.S.E.E. degree from the University of Illinois, Urbana-Champaign.

He joined the Burlington Physical Design group of IBM in 1985. In 1987, he transferred to Austin, working on circuit design and chip integration for RS/6000 and PowerPC microprocessor products. He currently is with the IBM Austin Research Laboratory.

Brian Zoric received the B.S.E.E. degree from the University of Pittsburgh, Pittsburgh, PA.

He has worked on circuit design for Power I and Power II RS/6000 chip sets and on circuit design and chip integration on the PowerPC 601 microprocessor. His current interest is high-speed circuit design methodologies.

Mr. Zoric received an Outstanding Technical Achievement Award from IBM in 1996 for contributions to a dynamic standard cell synthesis methodology.