Design methodology for the S/390 Parallel Enterprise Server G4 microprocessors

by K. L. Shepard
S. M. Carey
E. K. Cho
B. W. Curran
R. F. Hatch
D. E. Hoffman
S. A. McCabe
G. A. Northrop
R. Seigler

This paper describes the design methodology employed in the design of the S/390® Parallel Enterprise Server G4 microprocessors. Issues of verifying design metrics of area, power, noise, timing, testability, and functional correctness are discussed within the context of a transistor-level custom design approach. Practical issues of managing the complexity of a 7.8-million-transistor design and encouraging design productivity are introduced.

1. Introduction

The fourth generation of the S/390* CMOS microprocessor is a 17.35-mm × 17.35-mm chip with 7.8 million transistors, which has been successfully operated above 300 MHz at a supply voltage of 2.5 V [1].

The design methodology of this microprocessor follows in the tradition of other successful methodologies [2, 3] in simultaneously addressing four goals:

- Verify that a design meets all of several metrics of quality such as area, functionality, timing, power, noise immunity, reliability, and testability.
- Manage complexity.
- Encourage productivity.
- Coordinate a parallel design process.

Technology scaling and ever-increasing demands for performance shape many aspects of the design methodology. Technology scaling has had several major consequences, of which the simplest is the growth in the complexity of the designs as more transistors are available for a given silicon area. Interconnection widths are scaling lower, while interconnection lengths have remained virtually the same as additional function or larger caches have been added in lieu of making smaller chips. Total wire capacitance is decreasing as a result, but wire resistance is increasing faster. As a result, *RC* delays of interconnections are increasing. At the same time, wiring capacitance dominates the load on many nets. Coupling capacitances, in particular line-to-line coupling capacitances, have become a significant source of noise on the chip, which means that they can produce glitch-induced failures or have a significant effect on wiring delay. Threshold voltages have also scaled to maintain drive in the presence of scaling supplies. This has

0018-8646/97/$5.00 © 1997 IBM

implications for noise, power, and I_{ddq} (quiescent supply current) testing. In addition to technology scaling, the demands of ever-increasing performance are driving designs to the use of dynamic circuits, which create further complexity in noise and timing analysis.

- *Methodology themes*

With these technology and performance trends as the driving force, several methodology themes underlie the approach we have taken in the design methodology for the S/390 G4 microprocessors:

- The demands of performance have required a fundamentally transistor-level focus in the design methodology. All tools and processes allow a design to be customized and verified at the device level.
- A two-level hierarchical approach is essential for simultaneously managing complexity and parallelizing the design process. The increasing complexity of the designs has necessitated abstraction, while the closer electrical interaction of circuit and interconnections creates challenges in accurately modeling hierarchical boundaries.
- Static analysis techniques are key. Transistor-level static analysis techniques are used for timing analysis, noise analysis, Boolean equivalence checking, and fault-model generation. Techniques employing binary decision diagrams (BDDs) are an important aspect of this approach.
- Interconnections must be carefully designed and analyzed. This includes wire-width tuning and buffer insertion to control *RC* delays.
- Design abstractions must be stored and controlled from a common database.
- Cycle simulation is key to verifying register-level with high-level behavioral models of the architecture. This is the only way to achieve the simulation performance required to verify design of rapidly increasing complexity.
- Noise is a design metric of importance comparable to, if not greater than, area, power, and timing.
- Semicustom implementations that preserve the leverage of transistor-based design are crucial to achieving global timing convergence and managing rapidly evolving logic changes.

- *Metrics for design quality*

Several metrics of design quality must be analyzed as part of any microprocessor design methodology:

- *Area* The physical size of the chip.
- *Power* The amount of power that the chip dissipates and how that is handled by the thermal environment of the package. This is discussed in more detail in Section 8.
- *Noise immunity* This is perhaps the most important new metric; it is discussed in detail in Section 9.
- *Timing* The design must meet latch setup and hold requirements for proper sequential operation. In addition, the use of multiphase dynamic logic requires additional timing checks to guarantee correct circuit operation.
- *Functionality and correctness* This involves a verification chain connecting the design abstractions. Simulation is used to verify the VHDL[1] against an architectural specification. A combination of switch-level simulation and Boolean equivalence checking verifies the VHDL against the transistor-level circuit schematic, and logical-versus-physical (LVS) checking verifies that the layout matches the circuit schematic.
- *Testability* This involves building a separate logical description of the implementation, called a *fault model*, which is used for testability analysis of single stuck-at fault coverage and for test-pattern generation [4].
- *Reliability* Electromigration analysis is the most significant component of this metric.

- *Design abstraction*

Design abstraction is one of the key methodology tools used to manage complexity. In the G4 microprocessor methodology, these abstractions are stored in a central database. All analysis and verification are accomplished with a two-level hierarchical approach which involves identifying groups of 10000–200000 transistors as *macros*. Macros are individually laid out and floorplanned on the chip and form the main unit of the division of labor that allows the design processes to be parallelized. At the macro level, one would typically find the following design abstractions in the central database:

- *Symbol* Schematic representation of the ports of the macro and their directionality.
- *Entity* The VHDL entity for the design, automatically created from the symbol.
- *Schematic* A schematic representation of the transistor-level implementation of the macro. The schematic may in itself be a hierarchy of other submacro symbols and schematics.
- *Architecture* A VHDL architecture description of the function of the macro used for simulation and Boolean equivalence checking.
- *Timing graph* A timing graph abstraction created by transistor-level timing.

[1] VHSIC Hardware Description Language, IEEE Standard 1076.

- *Logical constraints view* This view contains Boolean satisfiability constraints in the implementation, which are tested through BDD techniques.
- *Layout* The physical design of the macro, which may be a hierarchy of other submacro symbols and schematics.
- *Fault model* A schematic of logic and sequential primitives used for generating test patterns and determining single stuck-at fault coverage.
- *Power view* An abstraction of the current demands of each macro on the supply and ground distributions. This is used to determine the chip power dissipation and to estimate power supply noise.
- *Abstract* This is a simplified view of the layout, which can be used for floorplanning, place and route, and global extraction. The amount of shapes information varies during the course of the design process.

The continued development of static noise analysis [5] will result in an additional view:

- *Noise abstract* A noise abstraction created by transistor-level noise analysis.

Above the macro level is a hierarchy of schematics, symbols, and layouts which constitute the *global* interconnections and physical design of the chip. Two abstractions of the global environment are brought down to the macro level to guide macro-level implementation.

- *Shadow* This is a representation of the global wires overlaying a macro that is used to guide macro physical design and for macro extraction.
- *Timing assertions* This is information on the global timing at macro interfaces—arrival times with phase tags on inputs, required arrival times with phase tags on outputs, primary input resistances, and primary output capacitances.

The ways in which design abstractions are created and used are discussed in detail in the remainder of the paper. Section 2 discusses the use of VHDL in the G4 microprocessor design. Section 3 discusses how the circuits are verified against their corresponding VHDL simulation models. Because of the importance of interconnect modeling, Section 4 discusses extraction and interconnect modeling as it is used in timing, power, and noise analysis. Section 5 discusses the timing methodology, while Section 6 discusses the semicustom logic synthesis approaches used in the G4 designs. Section 7 describes the physical design of the chip, both macro and global layout and physical design planning. Section 8 discusses the power, electromigration, and noise analysis methodologies.

2. VHDL design and verification

The G4 microprocessor was designed using VHDL 1076 as the register-transfer-level description language [6]. There were three principal requirements on the use of the language:

- Must be mappable to cycle simulation.
- Must be able to check the VHDL logic design for Boolean equivalence against a circuit implementation.
- Must be able in some cases to guide synthesis to an implementation.

In this section, we describe how the VHDL is entered and stored and the coding styles employed. We show how the VHDL guides cycle-simulation model builds, scan-chain connections, initial values for registers, global Boolean satisfiability constraints, and logical structure for synthesis.

- *Design entry*

VHDL is entered only for the macros; this is done as a structurally flat description, with the exception of special latch and array primitives discussed in the next subsection. The design above the macro level exists only as a schematic and is netlisted as structural VHDL for the purpose of logic simulation and verification. This guarantees correct-by-construction correspondence between the VHDL and circuit above the macro level.

- *Language subset for macro architectures*

The IEEE `std_logic_1164` package is employed and augmented with an expanded set of logical, relational, and arithmetic operators in a separate `std_logic_support` package. In the case of the = and \= operators, the `std_logic_1164` package contains declarations for these functions which are implicit with all enumerated types in VHDL. We replace these with explicit declarations in the `std_logic_support` package, relying on compatibility flags in the VHDL analyzers to allow these nonstandard function definitions to be declared in a separate package. All of the functions are carefully coded to propagate 'X' and 'U' states of the `std_ulogic` type. The entire concurrent VHDL language is allowed. In addition, process statements that explicitly represent combinational logic are permitted. To meet this criterion, the process statements must be activated by every input. In addition, conditionals must explicitly cover all cases to avoid implying registers. As an example of a valid process, consider the example shown in **Figure 1**: The process codes the combinational piece of the state machine shown in **Figure 2** with a single two-bit state register. Each conditional based on the input `x` has an `else` statement, and the process is activated by both the input `x` and the state vector values.

```
combin: process(current_state, x)
begin
  case current_state is
    when "01" =>
      z <= '0';
      if x = '0' then
        next_state <= "01";
      else
        next_state <= "11";
      end if;
    when "10"
      z <= '0';
      if x = '0' then
        next_state <= "00";
      else
        next_state <= "01";
      end if;
    when "11" =>
      z <= '1';
      if x = '0' then
        next_state <= "00";
      else
        next_state <= "10";
      end if;
    when others =>
      z = '1';
      if x = '0' then
        next_state <= "00";
      else
        next_state <= "01";
      end if;
  end case;
  end process;
```

Figure 1

VHDL process statement used to code combinational logic.

All sequential logic is handled by explicit instantiation of a set of latch and array primitives. In some cases, simple transparent latches are created from level-sensitive guarded block assignments. The G4 designs use six primitive latch components, each parameterized with a set of generics. Three types of parameterized array primitives are used—read-before-write, write-before-read, and read-only. Read-only primitives, used to model on-chip ROMs, take a read address of variable length as input and return an output word, also of variable length. The contents of the read-only primitives are loaded at simulation startup.

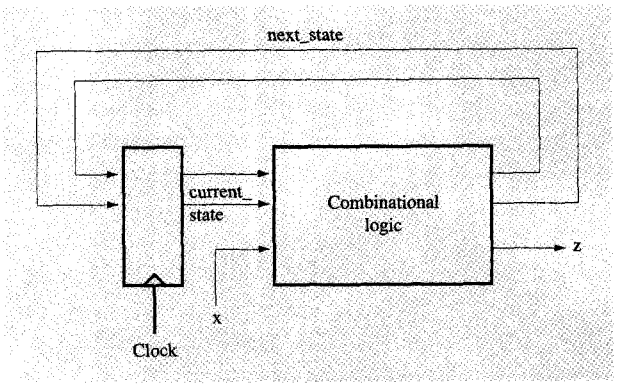


Figure 2

Example of a finite-state machine. A process statement can conveniently code the combinational logic of the state machine.

Both the read-before-write and write-before-read array primitives have an "asynchronous" read in which the data word is available at the output as soon as the read address is available. The writes are clocked. In the write-before-read primitive, if the write and read addresses are the same, the written data "flush" through and are immediately available for the read. In addition, tristate driver–receivers are also modeled with two special library components because of the inability of the synthesis tools used in cycle simulation to model high-impedance states. Explicit latch and array instantiation also allow these elements to be "snipped out" for the purposes of Boolean verification, discussed in more detail in Section 3. The VHDL model is explicitly full-function; that is, it models all of the logical functioning of the chip including test functions and contains full and complete scan-chain connections. Only cycle boundary latches are modeled in the VHDL. Mid-cycle latches, where they exist, are not modeled in the VHDL, as this is considered an implementation issue. The same applies to other latch structures which do not store machine state.

The G4 processor is initialized through scan. Initial values for the scan process are passed to the latch primitives through generics. A global VHDL variable determines whether these initial values are applied at $t = 0$ in VHDL simulation or whether the latch values are left at an `std_ulogic` value of `'U'`. These initial values are also exported as part of the cycle-simulation model-build process as a *model initialization file* (MIF) which can be used to initialize the cycle-simulation model as well as to determine the scan sequences required to initialize the processor for service processor code development. Scan-chain connections were coded in the VHDL in a manner to allow easy scan-chain reordering. Two local vector

signals, scan_connect_in and scan_connect_out, are declared in each macro. The scan_connect_in signal connects to the scan input of each latch, while the scan_connect_out signal connects to the scan output of each latch. All of the scan connections are then done as a block of signal assignments of bits of scan_connect_out to bits of scan_connect_in. This signal assignment block can then be replaced following scan-chain optimization as described in Section 6.

- *Logic simulation*

Two types of simulation are used on the G4 design—event-driven VHDL simulation and cycle simulation. VHDL simulators are event-driven; that is, they maintain an *event queue*, sequenced by real simulation time as well as "delta delays." In the coding style used for the G4 design, no explicit times are coded in the VHDL, except for the times used to establish the waveforms of the clocks. No attempt is made to use logic simulation to verify timing. This is done in static timing analysis, described in detail in Section 5. All signal assignments, therefore, occur as a cascade of delta delay events following a clock edge in VHDL simulation. We refer to this as a *clock-edge-triggered* logic specification. This controlled use of the language along with the explicit instantiation of latch and array primitives allow the register-transfer-level VHDL to be mapped to one of two cycle-simulation models very efficiently. In cycle simulation, one makes explicit use of the fact that the design is *clock-edge-triggered* to improve the performance of simulation [7]. Certain signals are identified as "registers" and change state only on the basis of their input values at the cycle boundary. Combinational logic is "flattened" to easily evaluated Boolean equations. The two cycle-simulation models built for the G4 design are

- A single-cycle simulation model. This model allows only a single state change per machine cycle, which is sufficient for modeling typical machine operation and therefore forms the basis for the main simulation engine for verification with instruction traces.
- A two-cycle simulation model. This model allows two state changes per machine cycle. In this model, latches are divided into two sets, those that evaluate on "even" cycles (or, equivalently, those latches that evaluate on the rising edge of the global system clock), and those that evaluate on "odd" cycles (latches that evaluate on the falling edge of the global system clock). This enables modeling of certain test functions that require this type of detail in the sequential modeling.

The cycle-simulation model-build process consists of three steps:

1. The VHDL is processed through the synthesis tools to produce a structural representation of the design.
2. Standardized primitives are used to replace elements of the structural representation.
3. The model is optimized and code generated for the cycle-simulation engine.

In step 1, latch and array primitives are "black-boxed"; that is, the VHDL architectures of these primitives are not processed by synthesis. Combinational logic is represented as a structural netlist of generic logical primitives.

In step 2, predefined cycle-simulation models for the primitives are used. The one-cycle and two-cycle simulation models are distinguished by the models for the latch primitives used in the model-build process. For most array macros, two separate VHDL descriptions also exist for the one-cycle and two-cycle models. The model-build process chooses between these two architectures in the VHDL netlisting operation through a "switch" view list. The one-cycle VHDL contains only the basic read and write functions which can be modeled with one-cycle granularity, while the two-cycle model contains details of boundary-scan and self-test functions, for example, which require two-cycle sequential granularity.

Consider the cycle-simulation representation of one of the latch primitives of the G4 design, a d-latch, with the VHDL description shown in **Figure 3**. In a single-cycle simulation model, this latch is modeled as shown in **Figure 4(a)**. In this case, the logic function is significantly simplified to model the basic system-state storing function of the latch. In particular, none of the test function of the VHDL is modeled. All register and array primitives can potentially change state every cycle. In this example, when clkg is '1', the latch changes state on the cycle boundary. In simulation, clkg is raised to '1' and held there. **Figure 4(b)** shows the two-cycle implementation of the latch model. In two-cycle models, registers and arrays can be of either "master" or "slave" type. We refer to this as *two-latch* behavioral modeling. Master-type latches, which evaluate on even cycles, are denoted with an M, while slave-type latches, which evaluate on odd cycles, are denoted with an S. With this level of sequential granularity, the full function of the latch can be modeled. In this case, clkg is toggled every cycle. In addition, the scan clocks, a_clk and b_clk, have at least twice the period of clkg to produce correct operation. In the cases in which individual transparent latches are used, a VHDL attribute is used to specify whether the two-cycle simulation mapping is of the master or slave type. The same attribute is also used for two-cycle mapping of array primitives. As an additional example of how master and slave declarations affect transparent latch modeling in cycle simulation, consider the example shown in **Figure 5**, in which two master latches are clocked by the same clock.

```
ctl1: block (clkg = '1')
begin
  control1 <= guarded scan_enable;
end block;
ctl2: block (clkg = '0')
begin
  control2 <= guarded control1;
ebd block;
c1 <= control2 and clkg;
c2 <= not(clkg) and (control1 or b_clk);
l1: block (c1 = '1')
begin
  l1_latch <= data;
end block;
L1s: block (a_clk = '1')
begin
  l1_latch <= scan_in;
end block;
12: block (c2 = '1')
begin
  l2_latch <= l1_latch;
end block;
```

Figure 3

VHDL description of a d-latch.

Applying the two-latch behavior to both transparent latches in the model, we see that the data are flushed through both latches in the same (i.e., "even") cycle. Replacing the rightmost latch in Figure 5 with a slave latch introduces a one-cycle delay from the master to the slave.

In step 3 of the cycle-simulation model-build process, the standardized flattened primitives created by the one-cycle or two-cycle synthesis and mapping steps undergo a variety of Boolean logic optimizations to improve run time and performance. Three target internal cycle simulators were used on the G4 design: TEXSIM, ZFS, and EVE. The optimizations performed by the first simulator (TEXSIM) include constant elimination, pin dropping, and expression merging. Code generation for this simulator results in an object module which uses an *oblivious evaluation algorithm*. By this, we mean rank-order simulation in which an expression is evaluated only after all of its predecessor expressions have been evaluated. TEXSIM simulation is used extensively for models of sections of the chip.

For larger models, including full chip and system, a second cycle simulator (ZFS) is used. Build for ZFS runs the TEXSIM optimizer, stopping prior to final code generation. The flattened and now optimized primitives are combined with additional parts and reoptimized, using a similar set of algorithms, including AND/OR/XOR gate merging. A fundamental difference between ZFS and TEXSIM is that ZFS treats all signals as single bits, ignoring any bundling that might have been present in the original design. Code generation for ZFS results in an object model which uses an event-driven evaluation algorithm of the optimized structural primitives.

For the very largest system models, a hardware accelerator known as EVE is used [8]. Build for EVE follows after a model has been built for ZFS. The

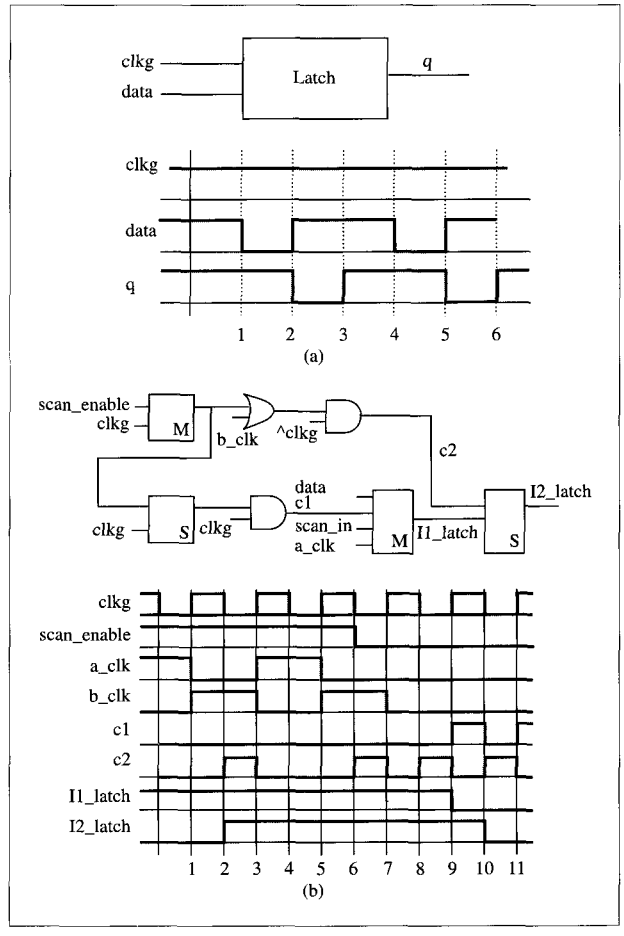


Figure 4

Cycle-simulation models of a G4 d-latch: (a) Schematic representation of the one-cycle model. Sample waveforms for this latch behavior are also shown. clkg acts as an enable signal (i.e., it is not toggled). (b) Schematic representation of the two-cycle model. Sample waveforms for latch behavior are also shown for the case of scan_in = 1 and data = 0; clkg is toggled to produce correct latch functioning.

optimized primitives are expanded into a four-input, one-output technology, optimized again, and then partitioned and scheduled for the EVE hardware. The object module runs on the EVE hardware, which performs highly parallel oblivious evaluation of the four-input, one-output primitives.

The test-case environment in the G4 design allows the designer the flexibility of moving between these simulators with *test-case transparency*, allowing the use of the simulator which is best for the specific model and test case.

The cycle simulators are explicitly two-valued simulators, propagating only '0' and '1' logic values. For nets driven by tristate driver–receivers, the Z state is recognized, and it is additionally checked that these nets are not simultaneously driven to conflicting logic values. Simulation checks to determine whether an uninitialized system state exists after a processor-wide scan-chain initialization is performed with VHDL simulation in which all the latches are left at the `std_ulogic 'U'` value at $t = 0$.

- *Assertions*

A Boolean function is said to be satisfiable if an assignment of Boolean value to the variables in the function results in a logic '1' value for the function. We refer to conditions expressed as a function which must be satisfiable as a *Boolean satisfiability constraint*. The variables in the function are referred to as a *constraint group*. Boolean satisfiability constraints constitute an important feature of the G4 design methodology. They are used for three main purposes: to express a "don't-care" set safely for a VHDL macro architecture, to allow the use of circuits that require certain logical conditions on their inputs for correct operation, and to eliminate false paths in static timing. Each of these uses is described in more detail in the sections that follow. At macro primary inputs and outputs, we use VHDL `assert` statements to express these constraints to immediately invoke simulation checking with their use. There are four types of conditions that can be expressed in this manner, each identifiable in the VHDL through the use of keywords in the message string:

- *Strong assertions* Assertions are logical conditions that we assume to be true and verify either formally or through simulation. Strong assertions are assertions which are true for any state that can be scanned into registers. These assertions can exist only on primary inputs of macros and *must* be accompanied by a test on the primary output of the driving macro where they can be formally verified. This creates the limitation that the constraint group for the strong assertion must be driven from a single macro and, consequently, a single test.

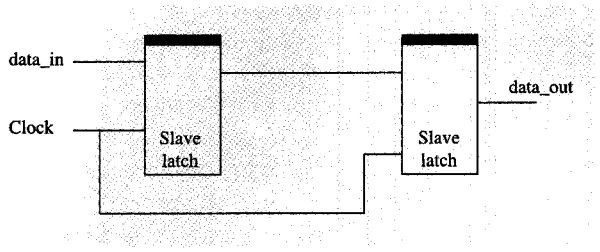


Figure 5

A configuration of two transparent latches of "master" type. Data flush through both latches on even cycles when the clock is '1'.

With further advances in formal combinational logic verification, flat logical verification of the design may be possible, in which case this limitation will be removed. This type of assertion must be used whenever there is a circuit that requires a condition for determinant function derived from primary inputs and is a thoroughly safe mechanism for expressing logical "don't-cares"; that is, all other conditions not covered by the assertion belong explicitly to the don't-care set.

- *Weak assertions* These are assertions which are true only for validly reachable machine states, but which will not be true for any possible machine state scanned into registers. These assertions must also exist at primary inputs, but are sequential in nature and can be verified only through simulation. These assertions are used in Boolean equivalence checking of VHDL and circuit at the macro level; conditions not covered by the assertion are also explicitly placed in the don't-care set. This assertion type may *not* be used when a circuit produces an indeterminant behavior in the absence of this assertion. It is preferable to convert weak assertions to strong assertions, except in cases where this would add unnecessary additional logic and reduce performance.
- *Strong tests* Tests are logical conditions that must be verified formally or through simulation. Strong tests are the tests which accompany strong assertions. The constraint group for a strong test must contain only primary output signals. In addition, strong tests must hold for any state which can be scanned into registers and must be verified formally. Strong tests are combinational in nature.
- *Weak tests* Weak tests are tests which are true only for validly reachable machine states. These are used only for simulation as a convenience to logic designers to flag unexpected conditions. Such tests are sequential in nature, and their failure implies incorrect operation of the machine.

• *Synthesis issues*

In some cases, the VHDL coding is used to help guide some pieces of the design to better initial structuring for synthesis. A common example of restructuring that might be done in the VHDL is to move timing-critical signals forward into the cone of logic by means of a Shannon expansion [9]. Let x_i denote a general Boolean variable and x_i' its complement. Consider a general Boolean function $f(x_1, x_2, \cdots, x_i, \cdots)$. The cofactor of f with respect to the variable x_i is given by $f_{x_i} = f(x_1, x_2, \cdots, 1, \cdots)$, while the cofactor of f with respect to the variable x_i' is given by $f_{x_i'} = f(x_1, x_2, \cdots, 0, \cdots)$. The Shannon expansion of f for the variable x_i is then given by $f(x_1, x_2, \cdots, x_i, \cdots) = x_i f_{x_i} + x_i' f_{x_i'}$. For example, if a critical path exists from input a to output g, and $g = f(a, b, c, \cdots)$, the VHDL is recoded as follows:

```
g0 <= f('0', b, c, . . .);
g1 <= f('1', b, c, . . .);
with a select
g <= g0 when '0',
     g1 when '1',
     'X' when others;
```

VHDL code is also frequently modified to eliminate encoders and decoders in critical paths by latching and distributing unencoded buses. Designers have direct control in the synthesis process of the extent to which the logical structure of the VHDL is preserved through optimization and mapping. The details of the synthesis process are described in Section 6.

Latch replication is also employed as a cloning technique not available to synthesis. This allows larger loads to be driven from latches without the need for buffers. Frequently, one latch is used to drive critical loads, while a cloned latch is used to drive noncritical ones. Retiming is also employed as a manual process in cases where this optimizes timing [10].

3. Equivalence checking

The functional verification methodology relies on simulation of the VHDL model through event-driven VHDL simulation and cycle simulation and an equivalence-checking methodology which ensures that the circuit implemented in silicon matches the VHDL description. The circuit must also be verified against the fault-model description used for single-stuck-at fault coverage and test-pattern generation. Because the design is represented as a single netlist representation above the macro level, the design is correct by construction above the macro level of hierarchy. Therefore, a necessary and sufficient condition for correspondence is that the macro circuits are Boolean-equivalent to the macro VHDL and the macro fault models. This is accomplished with a formal Boolean comparison of the circuit and VHDL, and of the circuit and fault model, augmented with switch-level verification of latch and array primitives.

IBM's Verity tool, used to perform the Boolean comparison, has been described in detail elsewhere [11, 12]. Verity relies on *canonical* reduced-ordered binary decision diagram (ROBDD) representations [13] of the logic from the fault model or as extracted from the VHDL by IBM's synthesis tool, BooleDozer [14, 15], and a logic representation of the circuit extracted from a simple switch model. Verity also incorporates a general configurable time-slice approach in which independent functions for different phase domains can be extracted and combined. This allows Verity to be used for the verification of multiphase dynamic implementations. The latch and array primitives discussed in Section 2 were preserved as "black boxes" to Verity in circuit, VHDL, and fault-model representations and were independently verified through switch-level simulation. These sequential elements are "cut out" as part of the verification process, in effect creating new outputs at the latch inputs and new inputs at the latch outputs. For large, complex designs where the ROBDDs grow too large, cut-point nodes are introduced to reduce the ROBDD size. In the circuit-to-VHDL comparison, strong and weak assertions (as described in Section 2) on the primary inputs are used to limit the care set of the comparison. In the circuit-to-fault-model verification, only the strong assertions are used, since weak assertions do not hold in general for any patterns scannable into registers, and therefore cannot restrict the care set of the comparison between the circuit implementation and the fault model.

Verity forms an important part of the Boolean constraint methodology in the G4 design. Verity is used globally to verify that every strong assertion is accompanied by a satisfying strong test for global signals. In addition, each macro, in general, has an associated logic constraints view which contains additional Boolean satisfiability constraints for the macro circuit, which are also verified by Verity as part of the comparisons. For debugging purposes, Verity generates a counterexample table showing all valid input states which produce incorrect outputs, failing satisfiability constraints, or failing consistency checks [16].

As an example of the canonical reduced-ordered BDD approach to equivalence checking, consider the VHDL and circuit model shown in **Figure 6(a)**. Verity computes the two final functions, f^1 and f^0, the function for which the output is driven to a 1 and the function for which the output is driven to a 0. The VHDL, of course, gives $f^1 = (x_1 \& x_2) | (x_3 \& x_4) | (x_5 \& x_6)$ and $f^0 = \bar{f}^1$. For the domino circuit, the final functions at the dynamic node are given by $f_d^1 = f_d^1[evaluate] | (f_d^1[precharge] \& \bar{f}_d^0[evaluate])$ and $f_d^0 = f_d^0[evaluate] | (f_d^0[precharge] \& \bar{f}_d^1[evaluate])$, where $f_d^1[evaluate]$ and $f_d^0[evaluate]$ are the functions driving the

output to 1 or 0, respectively, in the "evaluate" time slice, and $f_d^1[precharge]$ and $f_d^0[precharge]$ are the functions driving the output to 1 and 0, respectively, in the "precharge" time slice. These logical relationships between the time-slice elements are specified through a Verity control file. Since $f_d^1[evaluate] = 0$, $f_d^1[precharge] = 1$; and $f_d^0[precharge] = 0$, $f_d^0 = (x_1 \& x_2)|(x_3 \& x_4)|(x_5 \& x_6)$ and $f_d^1 = \overline{f_d^0}$. The static inverter at the domino output results in the final function $f^1 = f_d^0$ and $f^0 = f_d^1$. These final functions are the same as those specified in the VHDL; f^1 has the canonical ROBDD representation shown in **Figure 6(b)**.

Key to the verification process for large designs is a highly robust and efficient batch-submission system designed for running all of the macros within a unit in one submission. This includes building models, automatic creation of the Verity control file, and the submission of Verity jobs for every macro. In addition, comparison between circuit switch-level simulations and VHDL simulations for latch and array primitives are automated, with random patterns generated for valid clock and control signal sequences.

4. Extraction and interconnection modeling

In this section, we discuss the extraction and interconnection modeling used for various aspects of the G4 microprocessor design. Following the two-level hierarchical approach used for key analysis processes, extraction and interconnection modeling are divided between the macro and global levels, with special considerations for the interaction between these levels.

The resistance and capacitance extraction is rule-based, with lumped-element extraction, and involves the combined use of the vendor tools Dracula** and Preview** as well as internal tools. Rule-based approaches calibrated by finite-element calculations are the only techniques with the performance required for extraction calculation. Resistance is extracted using the sheet resistivity of the metal layer, with geometrical corrections for junctions. The capacitance extraction is done using coefficients derived from the two-dimensional configurations shown in **Figure 7**. Capacitances are calculated using a grid-based solution to Laplace's equation [17]. Line-to-line coupling capacitance is fitted to a single parameter, d, the spacing between the metal lines as shown for conductors 1 and 2 in Figure 7(a) with a piecewise-constant function with five to eight steps. Minimum-width lines and complete metal coverage on the planes above and below the lines are assumed for this characterization. Spacing between these metal coverage planes is denoted by H1 in the figure. Nonoverlapping line-to-line capacitance between interconnections on different levels, which we refer to as *distant fringe*, is also characterized with a single parameter, d, fitted to an equation of the form

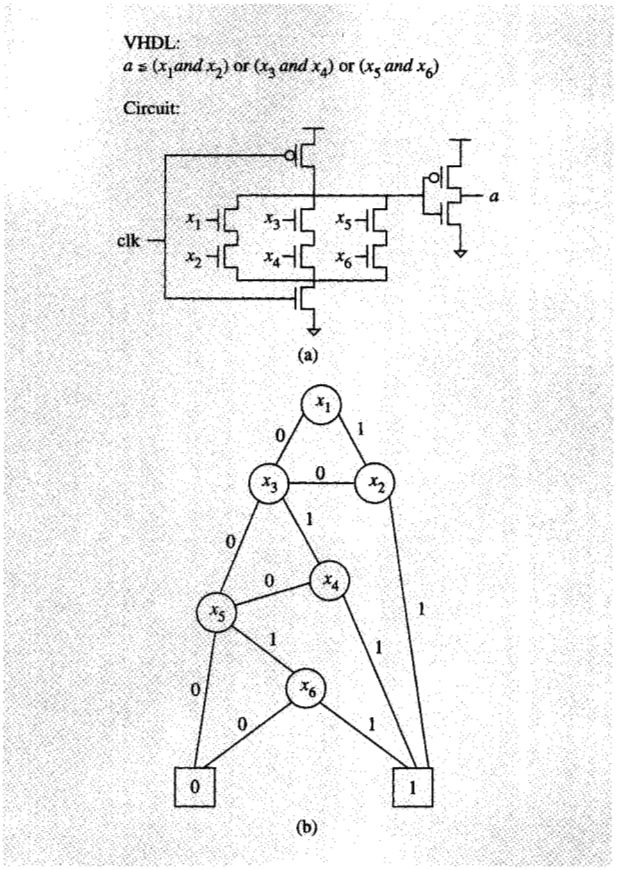


Figure 6

(a) Boolean-equivalent VHDL and circuit representations. (b) Canonical ROBDD representation for the 1-function (f^1).

$$C = K_1 e^{-K_2 \sqrt{d^2 + h^2}},$$

as shown for conductors 1 and 2 in Figure 7(b) (h is the dielectric thickness between the metal layers). The third component of capacitance is *area and fringe capacitance* between overlapping metal layers, as shown for conductors 1 and 2 in Figure 7(c). A piecewise-constant function is also used in this case, with a single parameter d, the distance to a neighboring conductor on the same level, which acts to reduce the fringing capacitance. For each value of d, capacitance for several values of conductor width, W, is calculated and the results fitted to $C = K_1 * W + K_2$, where K_2 is the fringe capacitance and K_1 is the area capacitance. Since this rule-based approach is fundamentally two-dimensional, three-dimensional effects can be handled only heuristically. For example, to handle the three-dimensional effects associated with shielding due to intervening layers in fringe and area

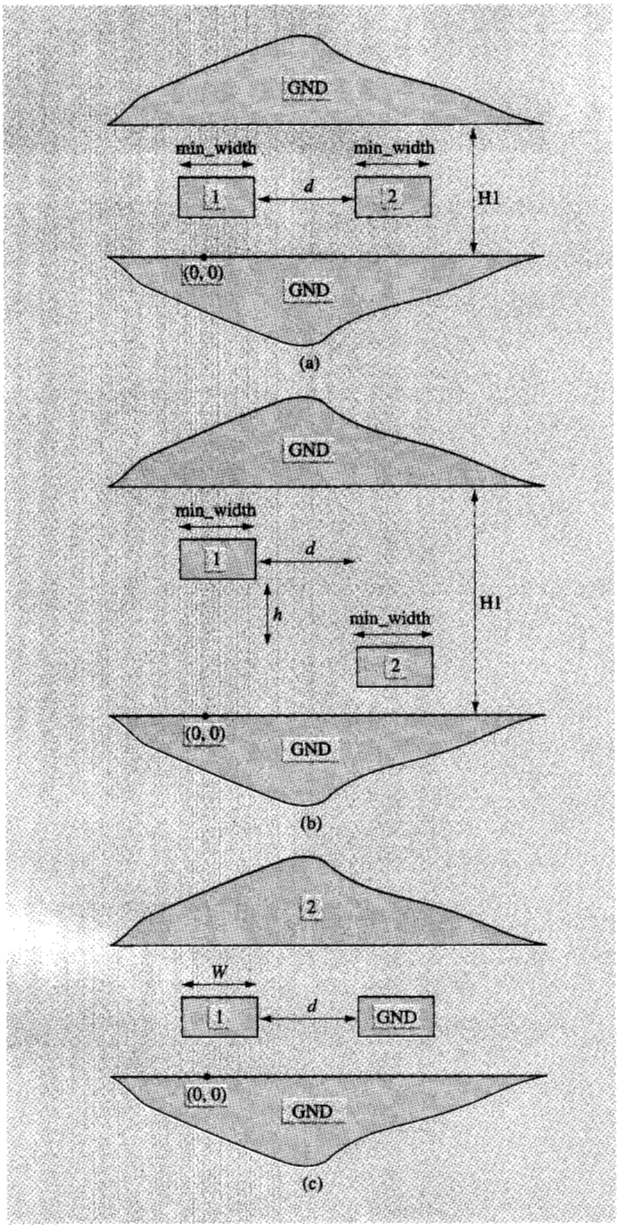


Figure 7

Capacitance geometries used to calculate rule-based coefficients for extraction: (a) Geometry for calculating line-to-line coupling, capacitance between conductors on the same interconnection layer. (b) Geometry for calculating distant fringe, capacitance between nonoverlapping conductors on different layers. (c) Geometry for calculating area and fringe capacitance for overlapping conductors on different layers.

calculation, the intervening layers are "expanded" to take into account the greater shielding effect these layers have than their geometric overlap would indicate.

- *Macro and global extraction*

Using the rule-based capacitance calculation described above, two types of extraction are done at the macro level:

1. Capacitance-only extraction, including coupling capacitors.
2. Resistance and capacitance extraction, in which all floating capacitors are broken as two capacitors tied to ground.

Depending on the stage of the design process, global coverage is modeled either as a statistical environment or as the shadow view passed down from the global level. Shadows used for macro extraction contain net-attributed shapes, allowing global net names to be used for floating-capacitor extraction. At the global level, four types of extraction are performed:

1. *Statistical* This is used when a quick interconnection estimation is required in timing analysis. Two statistical models are used. A worst-case model assumes 60 percent loading of all wires independent of their actual environment. A best-case model assumes only 30 percent loading of all wires. In both cases, two coefficients are used to characterize each interconnection layer, one that multiplies the area and another that multiplies the perimeter. If detailed routes are not available, a Steiner tree estimate of the wire length is used, along with a user-specified assumption of wire width which can be specified on a net basis. If a width is not specified, the minimum allowable wire width is used as the default.
2. *Detailed RC extraction without floating capacitors* In this case, a detailed capacitance calculation using the capacitance coefficients outlined in Figure 7 is used. Either abstracts or layouts are used for the macro shapes. All floating capacitors are broken and tied to ground.
3. *Detailed capacitance-only extraction with floating capacitors*
4. *Detailed RC extraction with floating capacitors*

Extraction techniques 2 and 4 produce tremendous amounts of resistance and capacitance data. Reduction techniques, described in the next section, are essential to successful analysis of these data for timing and noise analysis. Abstracts used for global extraction contain net-attributed shapes. For extractions 3 and 4, this allows macro net names to be used for floating capacitors. In extraction 4, resistances are not extracted for abstract shapes, since it is not possible to reconstruct the entire net topology necessary for correct analysis of distributed resistance.

• *Interconnection reduction*

The G4 design employs what we refer to as the *pi-model pole-residue macromodel* for the global interconnection. This technique is based on a state-space representation of the linear circuit equations that characterize the global interconnection. Let us first consider the circuit equations that correspond to calculating the current i_{driver} and the voltage v_{receiver} for the representative net shown in **Figure 8**. These equations can be written in matrix form as follows:

$$\mathbf{C}\dot{\mathbf{v}} = -\mathbf{G}\mathbf{v} + \mathbf{b}v_{\text{driver}},$$

or in the Laplace domain,

$$s\mathbf{C}\mathbf{v} = -\mathbf{G}\mathbf{v} + \mathbf{b}v_{\text{driver}},$$

where **C** is the capacitance matrix given by

$$\begin{pmatrix} C_1 & & \\ & C_2 & \\ & & C_3 \end{pmatrix},$$

and **G** is the conductance matrix given by

$$\begin{pmatrix} G_1 + G_2 & -G_2 & \\ -G_2 & G_2 & -G_3 \\ & -G_3 & G_3 \end{pmatrix}.$$

The *input vector* **b** is given by $\mathbf{b} = (G_1 \quad 0 \quad 0)^T$, and the *state vector* **v** is given by $\mathbf{v} = (v_A \quad v_B \quad v_C)^T$.

Let us first consider calculating the admittance of the network as seen by the driver. Temporarily ignoring the current through C_{node}, the capacitance to ground on the driver node itself, the current i_{driver} is given by

$$i_{\text{driver}} = \mathbf{l}^T\mathbf{v} + G_1 v_{\text{driver}},$$

where $\mathbf{l}^T = (-G_1 \quad 0 \quad 0)^T$. This gives the admittance

$$Y(s) = sC_{\text{node}} + \mathbf{l}^T(\mathbf{I} - s\mathbf{A})^{-1}\mathbf{r} + G_1,$$

including the admittance of C_{node}. $\mathbf{A} = -\mathbf{G}^{-1}\mathbf{C}$ and $\mathbf{r} = \mathbf{G}^{-1}\mathbf{b}$. Expanding in a Taylor series around $s = 0$,

$$Y(s) = s(C_{\text{node}} + \mathbf{l}^T\mathbf{A}\mathbf{r}) + s^2\mathbf{l}^T\mathbf{A}^2\mathbf{r} + s^3\mathbf{l}^T\mathbf{A}^3\mathbf{r} + \cdots.$$

These are the *moments* of the admittance. The elements of the pi-model shown in **Figure 9** are used to match for moments of the admittance to order s^3 [18]. Approximating transfer functions by their moments is the essence of asymptotic waveform evaluation (AWE) [19].

Similarly, the moments of the transfer function from the driver to each receiver are calculated:

$$H(s) = \frac{V_{\text{receiver}}(s)}{V_{\text{driver}}(s)} = \mathbf{l}^T(\mathbf{I} - s\mathbf{A})^{-1}\mathbf{r},$$

where $\mathbf{l} = (0 \quad 0 \quad 1)^T$.

In this case the moments are matched to a transfer function of the form

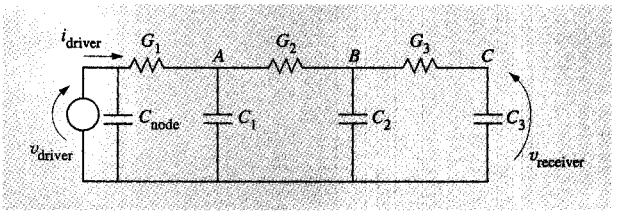


Figure 8

Representative RC interconnection structure.

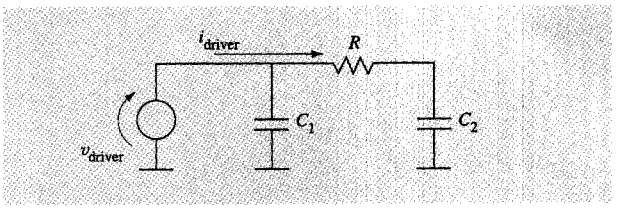


Figure 9

Pi-model used to model global interconnection load on the driver.

$$H(s) = \sum_i \frac{k_i/p_i}{s + p_i}.$$

This gives an output voltage for a unit step input of the form

$$V_{\text{receiver}}(s) = \sum_i \frac{k_i}{s + p_i} - \frac{1}{s},$$

where

$$\sum_i k_i = 1.$$

The pi-model pole-residue macromodels include the pi-model element values, C_1, C_2, and R, for each driver and the values of k_i and p_i for a given number of poles and residues for each receiver.

The pi-model pole-residue macromodels have several limitations:

- Receiver loads must be included in the reduction. As a result, it is not possible to separate the reduced-order model for the interconnections from the specifics of the receiver circuits.
- The approach is single-input, single-output. As coupled nets are included in the analysis, the number of ports will grow, lending computation efficiency to a multiport treatment.

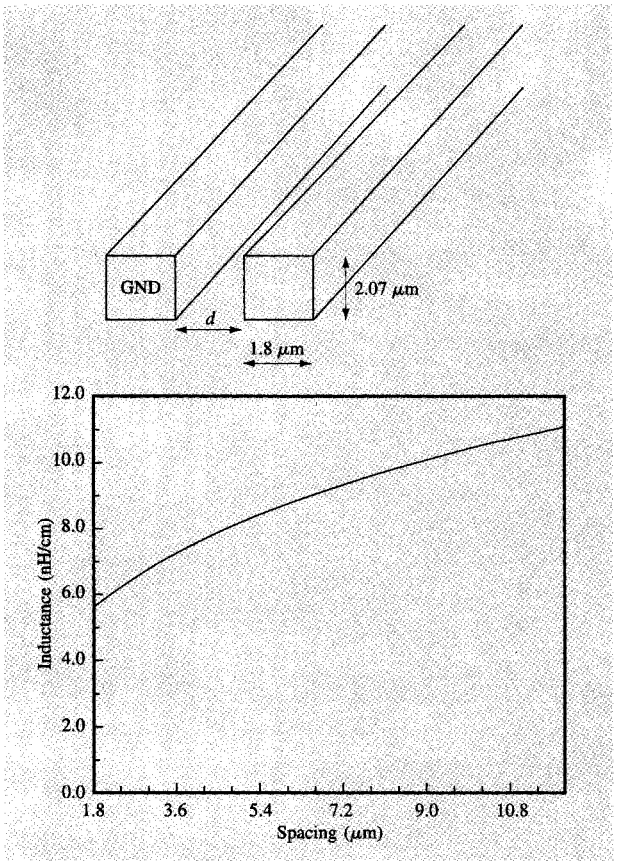


Figure 10

Inductance per unit length as a function of spacing between two wires with a skin depth of $\delta = 0.626\ \mu m$.

- Techniques such as AWE which rely on explicit calculation of the moments are numerically unstable.

To solve these difficulties, we are migrating our interconnection modeling to a *multiport driving-point impedance* formulation [20]. The impedance of an r-port, n-node, RC interconnection structure is given by

$$\mathbf{B}^T(\mathbf{G} + s\mathbf{C})^{-1}\mathbf{B},$$

where $\mathbf{B} \in \mathcal{R}^{n \times r}$ is given by $\mathbf{B}^T = (\mathbf{I}_r|0)$. $\mathbf{I}_r \in \mathcal{R}^{r \times r}$ is the identity matrix. Implicit Krylov subspace techniques such as Pade via Lanczos (PVL) can be applied to reduce these state equations, avoiding direct calculation of the moments [21].

On thick, low-resistivity, last-metal interconnections, we have found that inductance can have a noticeable effect on delay. The point at which inductance must be considered in interconnection analysis depends on the relative magnitude of three factors:

- Z_0, the characteristic impedance of the line

$$Z_0 = \sqrt{\frac{\mathcal{L}}{\mathcal{C}}},$$

  where $\mathcal{L}$ and $\mathcal{C}$ are the inductance and capacitance per unit length of the interconnection.
- R_{driver}, the effective resistance of the driver.
- R, the total resistance of the line.

For transmission-line effects to matter, $R_{driver} \ll Z_0$ and $R \ll Z_0$ [22, 23]. Inductance can easily be included in the linear interconnection analysis. The difficulty is that it is in general very difficult to calculate inductance, since the current return path is rarely well defined in the on-chip interconnection. Fortunately, however, inductance has only a weak logarithmic dependency on the distance to the current return, as shown in the example of **Figure 10**. As a result, if efforts are made to ensure a certain porosity of the power and ground distribution, the self and mutual inductances can be estimated with the current return assumed to be through the nearest power or ground distribution [24]. A lower bound on the inductance can also be obtained from the infinite-frequency relationship between the inductance and capacitance matrices:

$$\mathbf{LC} = \mu\epsilon\mathbf{I},$$

where μ and ϵ are the permeability and permittivity of the interconnection dielectric.

In reducing the interconnection models at the macro level, the requirement exists to preserve an RC netlist representation of the data for circuit simulation and timing analysis. To accomplish this, one can preserve the original topology of the RC extracted netlist, treating each branch as a two-port network, in lieu of other partitioning schemes which destroy the original net topology [25, 26]. Each two-port network can then be reduced to one of the three representations shown in **Figure 11**. To accomplish this reduction, the moments of the admittance matrix $\mathbf{Y}$ of the original branch network are calculated [27, 28]. Let $Y_{ij}[n]$ denote the nth moment of the given matrix element of the 2-by-2 $\mathbf{Y}$ matrix. Determining the element values is done through explicit moment-matching. For example, for the two-capacitor implementation, we match

$$Y_{11}[1] = C_1 Y_{22}[1] = C_2 Y_{12}[0] = Y_{21}[1] = -\frac{1}{R}.$$

This initial reduction leaves many single-resistor point-to-point nets. These can be reduced to a lumped capacitance value [Figure 11(a)] based on a time-domain criterion. In this case, the loading capacitance at the ports must be considered, and $R(C_1 + C_{port1})$ and $R(C_2 + C_{port2})$ must be less than t_{min}, the RC delay accuracy desired: typically 10 ps.

5. Timing

Static timing analysis is a major component of the G4 microprocessor design methodology [29]. Unlike timing *simulators*, static timers require no input patterns and find longest and shortest paths through a circuit with preconditioning assumptions at each gate to produce the worst-case or best-case delay. Static timing analysis also depends on the ability to abstract real switching voltages as linear saturate ramps, characterized only by a delay and a slew. As part of this abstraction, the 50% point of the real switching waveform is used to characterize the delay, and the difference between the 10% and 90% values is extrapolated to determine the slew.

There are some underlying methods and assumptions of static timing analysis that require special mention. Key to the approach is the construction of a *timing graph* from the circuit representation, as shown in **Figure 12**. Timing graphs are made up of *timing points*, which are connected by directed *propagate segments* or *test segments*. The propagate segments contain information on how *arrival times* (AT) and slews are propagated "forward" across the segment. Test segments describe setup or hold checks between signals. In the presence of a test, a *required arrival time* is also calculated (the arrival time that would be required to just satisfy the test). These times are propagated "backward" through the graph. Graph propagation can be done in either *late mode*, in which the latest of the arrival times is taken at each timing point, or *early mode*, in which the earliest of the arrival times is taken at each timing point. The *slack* is the difference between the required arrival times and the arrival time in late mode or the difference between the arrival time and the required arrival time in early mode. More details of these definitions can be found in Reference [29].

Another important aspect of static timing analysis is the idea of *cycle adjusts*, that is, determining whether a signal should be tested against a clock in the current cycle or the following cycle. This is accomplished in practice by "adjusting" the arrival time of the clock according to a methodology based on *phase tagging* of all data signals to indicate a reference clock edge, as shown in **Figure 13**. In this example, there is a single reference clock, denoted as `C1`. The clock phase associated with a positive active clock is denoted as `C1+`, while data launched from the leading edge of the active clock are denoted as `C1+R`. In this case, the cycle adjust is the difference between the next subsequent clock reference edge and the data reference edge as determined by the tagging. In the case of designs with transparent latches, "flush" loops may exist in the design. These loops are broken at one of the transparent latches, where the clock edge is used to determine the arrival time. The arrival time that wraps around the loop is subsequently compared to this clock reference edge. In the case of a violation of this "loop test," the launching arrival time may be adjusted forward into the period of latch transparency in an attempt to remove the violation [30]. Fundamental to static timing analysis, therefore, is that all data edges have an associated clock reference edge. In particular, loops in a timing graph must be "controllable" by a clock. These limitations make the

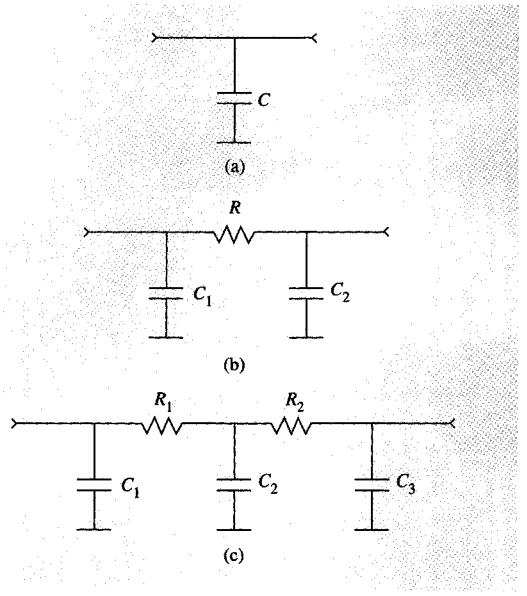


Figure 11

Possible two-port reductions of a single branch of an *RC* netlist.

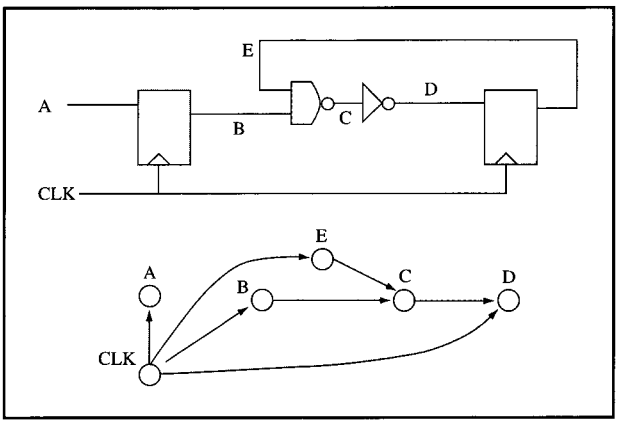


Figure 12

Timing graph abstraction. From the circuit shown at the top is abstracted a graph of timing points connected by propagate segments (shown in blue) and test segments (shown in red).

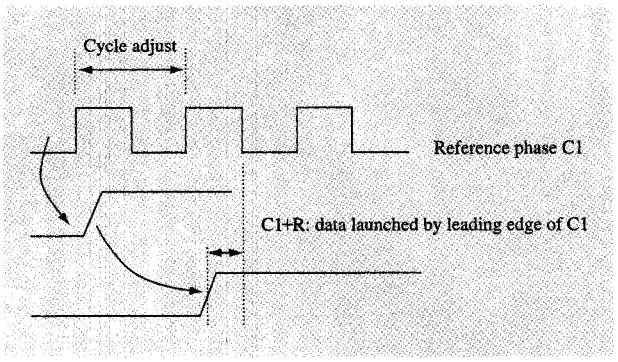


Figure 13

Determining the cycle adjust from clock tags. This figure shows the one-period cycle adjust calculated for a data signal launched by the leading edge of the C1 clock and subsequently tested to the falling edge of the same reference clock in the following cycle.

current algorithms generally difficult to extend to self-timed or self-resetting circuits [31].

The timing methodology for the G4 design is implemented using the tool Pathmill** from EPIC Design Technology and IBM's EinsTimer*. As in all key analysis processes on the G4 design, a hierarchical approach is used. Macros are individually abstracted from transistor-level analysis and are combined with global interconnection models in chip-level timing runs. The hierarchical approach allows faster turnaround of full-chip timing runs, since only those macros which changed since the last timing run must be re-abstracted. In addition, quick analysis of proposed global wiring changes can be made without detailed timing analysis at the macro level.

- *Macro timing*

Pathmill was used for the macro-level timing analysis. Inputs include a netlist, configuration file, and characterization file. The netlist can be generated either from a schematic or from an extracted layout. The configuration file contains the assertion information generated from the global timing run relevant to the macro under analysis. In addition, it contains "hints" to Pathmill on how to handle difficult circuit topologies, such as clock-shaping circuits, complex latch structures, or certain pass-gate structures. These commands are applied to sets of devices identified from subgraph isomorphism with specified patterns [32]. Delays at each channel-connected component are generally made under the assumption that only one input switches at a time. Patterns are also developed in an effort to deal with the effects of simultaneous switching on early-mode timing. Patterns that match the most common static CMOS gates, such as NANDs, NORs, OAIs, and AOIs (two-, three-, and four-way) are used to reduce the best-case delay calculated for these circuits. Boolean satisfiability constraints in the form of inversion or orthogonality declarations are also passed to Pathmill in the configuration file and are used to eliminate false paths in the timing graph. These conditions are obtained from the *logical constraints view* and are verified by Verity, as described in Section 3. The characterization file specifies the input slew and output loading design point used for determining the sensitivity coefficients for delay and slew to these quantities in the timing abstractions.

In lieu of complex metastability analysis, heuristics are applied to determine setup and hold times at latches, as shown for example in **Figure 14(a)**. There are two possible types of heuristics that can be used, "trigger-to-trigger" and "trigger-to-latch." In "trigger-to-trigger" heuristics **[Figure 14(b)]**, which are the simplest to analyze but in most cases are prohibitively conservative, data are always launched at the latch trigger time, or at the leading edge of the active clock. Late data must be set up at the latch node before the leading edge of the early active clock arrives at the clock node. Early data must be held after the trailing edge of the late active clock arrives at the clock node. **Figure 14(c)** shows the case of trigger-to-latch heuristics. In this case, data are launched from the latch at the later of the data input arrival time or the trigger time for late mode. For early mode, the data are launched at latch trigger time. Late data must arrive at the latch node before the trailing edge of the early clock, and early data must be held after the trailing edge of the late active clock. If data arrive after the trailing edge of the clock, the trailing edge of the clock is used to launch data from the latch. This is referred to as *clipping*, and the data launched from the register are said to be at a *clock-limited* arrival time. Many of the registers of the G4 design are of a master–slave type. In this case, trigger-to-latch heuristics are used on the master, and trigger-to-trigger heuristics are used on the slave. When master and slave clocks are nonoverlapping, setup and hold checks are performed to the trailing edge of the master clock and data are launched from the leading edge of the slave clock. When master and slave clocks overlap, setup checks are performed to the leading edge of the slave clock, while hold checks are performed to the trailing edge of the master clock. Data are launched from the leading edge of the slave clock. In the case of transparent latch design, trigger-to-latch heuristics are always employed. To add conservatism to the latch analysis, the delay in actually setting the latch, that is, switching the cross-coupled inverters, is included in the data arrival time for setup checks.

Static timing analysis can also be applied to multiphase dynamic logic, with additional timing constraints that must be satisfied by timing analysis. As an example, consider the "footed" domino stage shown in **Figure 15**. "Footed" denotes the presence of a clocked evaluation transistor at the bottom of the n-FET stack. For this logic stage, there are four additional timing checks which must be performed:

- The dynamic node must fall before the falling edge of the clock (setup). The evaluate must occur during the current cycle's evaluation period.

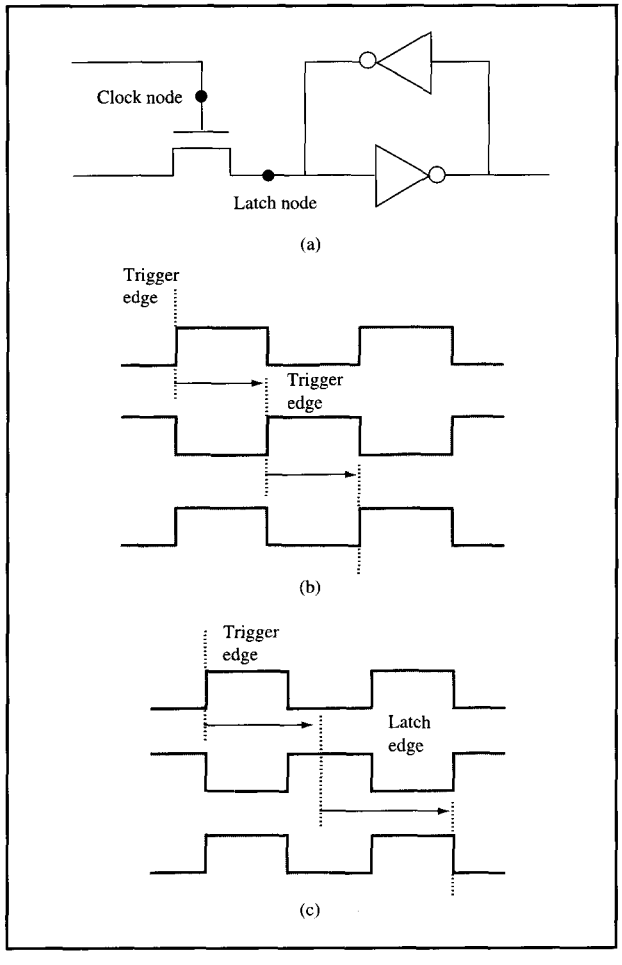


Figure 14

Latch heuristics: (a) Example transparent latch circuit, with clock-node and latch-node timing points identified. (b) Trigger-to-trigger heuristics in which data are launched at trigger time and tested at trigger time. (c) Trigger-to-latch heuristics in which data are launched at the later of the data input arrival time or the trigger time for late mode. Tests are performed against the latch time, the trailing edge of the clock.

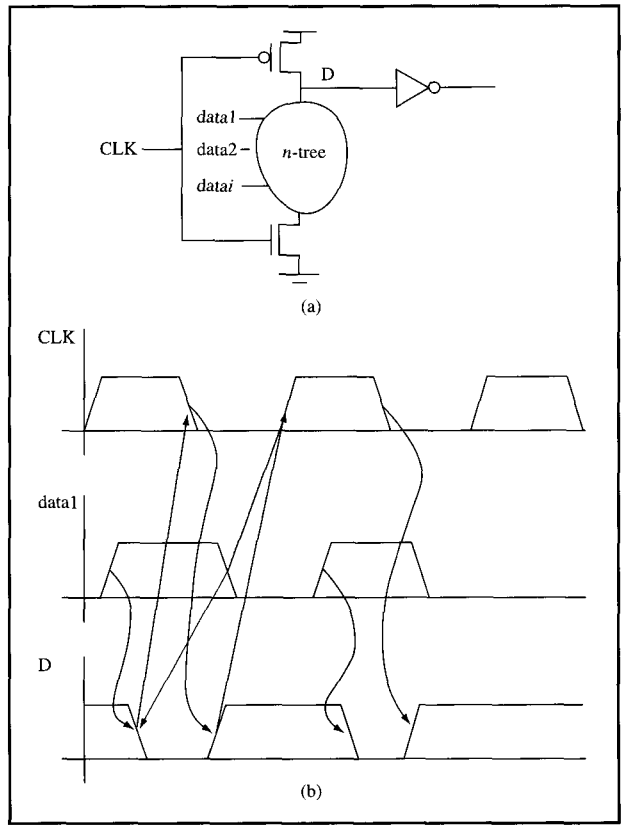


Figure 15

Timing checks for a standard "footed" domino stage: (a) Circuit topology for the stage. D denotes the dynamic node of the gate, and CLK is the clock. (b) Example waveforms for the data inputs, clock, and dynamic nodes. Red arrows denote setup tests. Green arrow denotes hold test. Black arrows denote delays.

- The data node must fall before the rising edge of the clock (setup). This ensures that the previous stage resets before the evaluation begins.
- The dynamic node must rise before the rising edge of the clock (setup). This ensures that the current stage resets before the evaluation begins.
- The falling edge of the data node must be held until after the dynamic node falls (hold). This ensures that the data "pulses" are wide enough to evaluate the gate.

Timing abstractions presented to global timing from macro analysis can be either black or gray, as shown in **Figure 16**. In the case of the black box, no internal latch points are defined, and setup and hold tests are presented at primary inputs. Black boxes can only be used in the case of static logic with nontransparent latches. In addition, black box abstraction requires independent verification of latch-to-latch paths within the macro, which

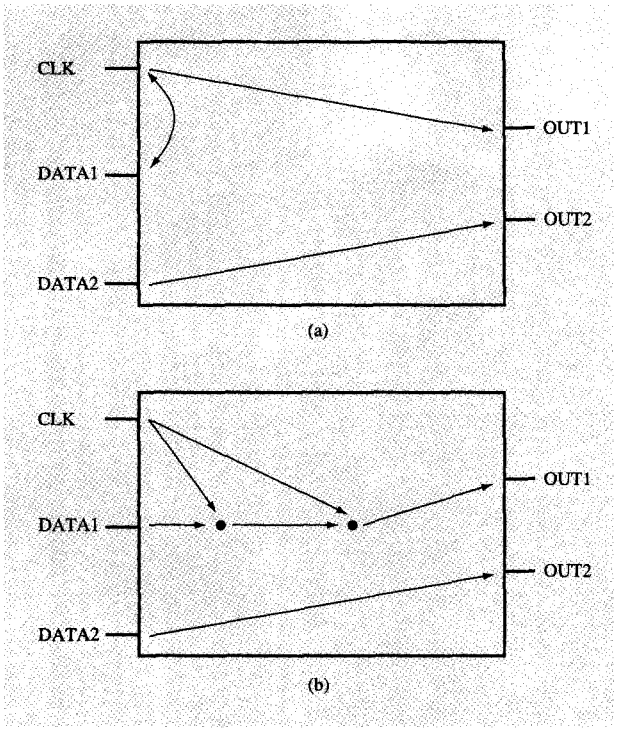


Figure 16

(a) Black box and (b) gray box modeling.

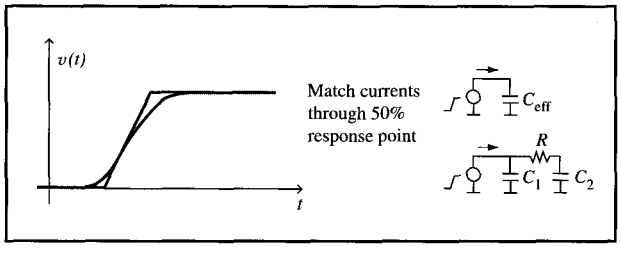


Figure 17

Calculating C_{eff}.

are not presented to global timing. Gray boxes are essential for timing verification in the case of transparent latches or domino logic. In this case, internal latch points are defined, and segments and tests to the internal latch points are included in the abstraction.

- *Global timing and assertion management*

To perform the global timing analysis, the black or gray models from macro timing are translated into DCL [33] and loaded into EinsTimer along with the pi-model pole-residue global interconnection models described in Section 4. In the early stages of the design, this information comes from largely estimated or partial routes. As the design progresses, the interconnection models increasingly reflect fully routed designs. The statistical model described in Section 4 is used throughout most of the design process, with "best-case" statistics used in early mode and "worst-case" statistics used in late mode.

To calculate the macro driver waveforms, we use the idea of the "effective capacitance," C_{eff} [34]. The effective capacitance is a single capacitance assertion, which is designed to yield an accurate delay and slew against a "k-factor" driver model. The actual admittance of the interconnection is modeled at the driver as a pi-model, as shown in **Figure 17**. C_{eff} is given by the capacitance that produces the same total integrated current through the driver through the 50% point of the driver voltage waveform. Let the slew (0–100%) at the driver be given by t_r. We consider a rising waveform, but the same discussion applies to a falling waveform. The total integrated current to the 50% response point for the C_{eff} driver load is

$$\int_0^{t_r/2} I\,dt = \frac{C_{eff}V_{dd}}{2}.$$

Combining the admittance of the pi-model with the Laplace transform of the saturate ramp waveform, one finds that the current flowing through the driver in the Laplace domain is given by

$$I(s) = \frac{V_{dd}}{t_r}\left(\frac{C_1 + C_2}{s} - \frac{C_2}{s + \dfrac{1}{RC_2}}\right)(1 - e^{-st_r}).$$

In the time domain, this becomes

$$I(t) = \frac{V_{dd}}{t_r}[(C_1 + C_2) - C_2 e^{-t/RC_2}] \qquad \text{for } t < t_r.$$

The total integrated current to the 50% response point for the pi-model driver load is

$$\int_0^{t_r/2} I\,dt = \frac{V_{dd}(C_1 + C_2)}{2} - \frac{RC_2^2 V_{dd}}{t_r}(1 - e^{-t_r/2RC_2}).$$

Equating the two integrated-current expressions, one obtains an expression for C_{eff} in terms of the slew time t_r,

$$C_{eff} = C_1 + C_2 - \frac{2RC_2^2}{t_r}(1 - e^{-t_r/2RC_2}).$$

To find C_{eff}, this equation is solved iteratively with the driver slew equation, which gives the slew as a function of C_{eff}. Convergence is achieved in a few iterations. We note

that our approach differs from that of Reference [34] in that only the slew at the driver is used; that is, no "block delay" is considered in the analysis.

To calculate the receiver waveforms from the pi-model pole-residue interconnection model, the driver saturate ramp waveshape is applied to the pi-model transfer function. For poles p_i and residues k_i, the step response in the Laplace domain is given by

$$V(s) = \sum_i \frac{k_i}{s + p_i} - \frac{k_{\text{sum}}}{s},$$

where

$$k_{\text{sum}} = \sum_i k_i,$$

and $-k_{\text{sum}}$ is the steady-state voltage value. The Laplace transform of the saturate ramp source is given by

$$\frac{1}{s^2 t_r}(1 - e^{-st_r}).$$

The voltage response at the receiver is then given by

$$V(s) = \frac{1}{st_r}\left[\sum_i \frac{k_i}{s + p_i} - \frac{k_{\text{sum}}}{s}\right](1 - e^{-st_r}).$$

Converting to the time domain, one finds

$$v_t = \begin{cases} \frac{1}{t_r}\left[\sum_i \frac{k_i}{p_i}(e^{-p_i t} - 1) - k_{\text{sum}} t\right] & \text{for } 0 \le t \le t_r, \\ \frac{1}{t_r}\left[\sum_i \frac{k_i}{p_i}(e^{-p_i t} - e^{-p_i(t - t_r)})\right] - k_{\text{sum}} & \text{for } t \ge t_r. \end{cases}$$

This is then converted back to a saturate ramp waveform by calculating the 50%, 10%, and 90% response points.

Assertions are generated from the global timing runs and are necessary to drive macro-level timing optimization and as characterization information for timing abstraction. For each macro, the following assertions are generated:

- Effective capacitances on the outputs.
- Primary input resistance assertions.
- Input arrival times (early and late mode, rising and falling) with phase tags.
- Output required times (rising and falling) with phase tags.

A "slack-apportionment" algorithm is employed during the early phases of the design process, before timing convergence is achieved, to apportion negative slack across multiple macros through modification of the actual arrival time and required arrival-time assertions. Proper assertion management is key to timing convergence in a hierarchical timing environment. In multicycle dynamic and separated latch designs, which are increasingly required in high-performance design, additional timing checks associated with the clock phases must be managed across hierarchical boundaries. Significant work is underway to address the challenges associated with managing these phase constraints and achieving overall phase convergence in the design.

The global chip timing run using EinsTimer also produces slack reports showing path traces associated with the worst slacks in the design, early or late mode. Another useful report is a list of nets that violate slew limits, with both driver and receiver slews presented in the violations report. Issues associated with managing *RC* delays in global interconnections are further discussed in Section 7.

Voltage, temperature, and process conditions, slack margins, and slew constraints were chosen to guarantee functionality of the G4 design. A late-mode slack margin was established to obtain sufficient yield of 300-MHz processor chips, taking into account the effects of phase-locked loop jitter, clock skew, coupled noise, and temperature and voltage swings within the multichip module environment. All circuits in late mode were timed to nominal process, highest predicted on-chip temperature, and lowest predicted on-chip voltage. Early-mode analysis was performed at a three-sigma fast process, lowest predicted on-chip temperature, and highest predicted on-chip voltage. Early-mode slack margins, protected by short-path padding in the design, were chosen to account for the effects of clock skew and simultaneous switching. Different clock skew values were assumed, depending on the receiving latch type and the relative locations of the latches in the clock distribution tree at the beginning and end of the path. A slew (10% to 90% transition time) limit was also enforced to reduce path delay sensitivity to manufacturing process variations and to reduce path delay sensitivity to coupled noise and ground-supply bounce. A slight delta in ground or supply potential between driving and receiving circuits translates into a variation in propagation delay proportional to slew rate. Global nets are allowed the highest slew limit only because the relatively high resistance of the interconnections forced a higher limit. Nets internal to a macro have a smaller slew limit, and even smaller slew limits are targeted for dynamic nets and internal latch nodes, since noise on these nets could affect the functionality of the chip.

6. Semicustom synthesis methodology

BooleDozer [14], IBM's logic synthesis tool, was an essential element of the G4 methodology for implementing major portions of the G4 microprocessor design, many of these containing timing-critical paths. In this section, we describe some of the ways we exploited

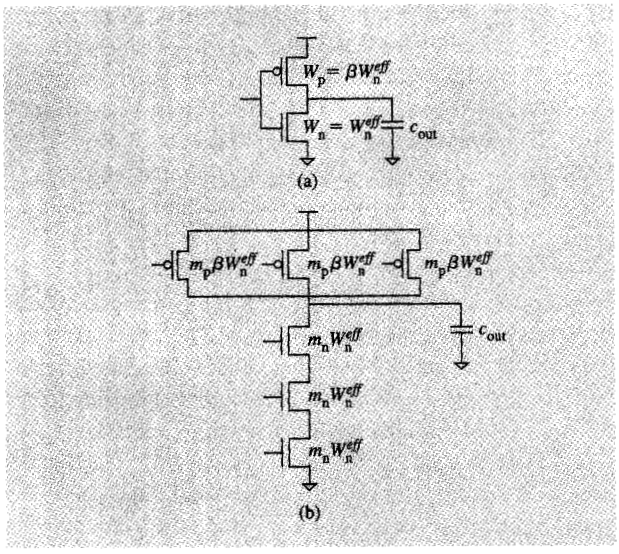


Figure 18

(a) Static CMOS inverter. (b) Three-input NAND gate.

BooleDozer to achieve rapid implementation while maintaining the ability to control the logic structure and aggressively tune the design at the device level. Some of the future directions in semicustom implementation are also addressed. The discussion involves several different aspects of the use of synthesis in the G4 design:

- Use of a continuously tunable, parameterized standard-cell library with logic functions chosen for performance.
- Designer controls on restructuring and technology mapping to this library.
- Use of "don't-cares" as defined by VHDL asserts to simplify logic implementation.
- Use of "hill-climbing"-based late-timing correction.
- Use of postplacement retuning and postplacement optimization of the macro clock distribution and scan chains.
- Use of tag-based partitioning to create design hierarchy to allow further customization of circuit and layout.

Traditionally, timing rules for standard-cell designs have been based on the actual size of the gate. In addition, each cell was available in a number of discrete sizes, or "power levels." The timing rules for the static CMOS library used in the G4 microprocessor design differ from these traditional libraries in three important ways. First, the rules were continuously parameterizable; that is, no fixed library cells were assumed. This has implications for the physical design of the library, which is discussed in Section 7. Second, the rules were parameterized by quantities directly related to delay, rather than size, which we refer to as *normalized gain* and *beta*. Finally, the parameterized logic functions chosen were limited to simple, single inverting stages, the most complex being a 2 × 2 AOI/OAI (AND-OR-INVERT/OR-AND-INVERT).

Let us first define the parameters normalized gain and beta. Consider the static CMOS inverter shown in **Figure 18(a)** driving a load capacitance of c_{out}. Let p_{cg} be the gate capacitance per unit width. The normalized gain, g, of the inverter is given by

$$g = \frac{c_{out}}{p_{cg}(W_p + W_n)}.$$

Beta (β) is given by

$$\beta = \frac{W_p}{W_n}.$$

In addition, we define a parameter called the *effective n-FET width*, W_n^{eff}, which is given by

$$W_n^{eff} = W_n$$

for the static inverter. In terms of β and W_n^{eff}, the normalized gain is given by

$$g = \frac{c_{out}}{p_{cg} W_n^{eff}(1 + \beta)}.$$

Now consider the three-input NAND gate shown in **Figure 18(b)**, driving the same load capacitance c_{out}. The equation derived above continues to apply. We introduce *FET multiplication factors* m_n and m_p, which are chosen for a particular book type so that the rising and falling delays of the gate match the rising and falling delays of a normalized gain-3 inverter. One might expect that $m_p = 1$ and $m_n = 3$. In actuality, $m_n = 2$ would be a typical value for the technology used in the G4 microprocessor design. This approach follows closely the previously published method of "logical effort" [35, 36]. In our formulation, the normalized gain of the gate is the same as the product of the logical effort and the electrical effort used in Reference [36].

The rule structure itself consists of interpolated tables which calculate delay (d_o) and slew (s_o) as a function of input slew (s_i), normalized gain (g), and beta (β):

$$d_o = f(s_i, g, \beta),$$

$$s_o = f(s_i, g, \beta).$$

Figure 19 shows a rising output delay of an inverter as a function of normalized gain and slew for a β of 1.5. The rule structure also allows calculation of delays for a sized gate set with a table which stores the value of W_n^{eff} for each fixed-size gate.

A parameterized domino library is also being developed using many of the same ideas. The domino library differs

from the static one in two principal ways. First, unlike the static library, in which performance drives the gate design to simpler logic function, domino gates are designed to achieve as much logic function as possible. Many complex functions are achievable by replacing the traditional output inverter with a static NAND or NOR gate, as shown in **Figure 20**. Second, gain is the only parameter which drives domino sizing. Noise considerations and precharge time requirements drive the rest of the device sizing and ratioing.

An important aspect of the use of BooleDozer in the G4 microprocessor design was designer control over *structural dominance*. By structural dominance, we mean the extent to which the logical structure in the VHDL dominates the mapping [14]. This is accomplished with a `SYN_CONTROL` keyword which is placed in the VHDL through an attribute on the block statement or, in some cases, on an entire design entity. `SYN_CONTROL` could have one of two values, `direct` or `dataflow`, both implying structural dominance of the logic as coded in the VHDL. The value `direct` denotes the highest degree of control from the VHDL. BooleDozer attempts to find a one-to-one mapping into the target technology. If none exists, the function is given the same treatment as the `dataflow` keyword implies. In the `dataflow` case, the technology-mapping algorithm attempts to find a covering that matches the original structure as closely as possible [37].

Explicit declaration of a "don't-care" set using VHDL assert statements provides another approach for optimization in BooleDozer [38, 39]. A common example is a fully decoded bus, in which the bits of the bus are known to be orthogonal. Consider the implementation of the following piece of VHDL:

```
assert(not(a(0) and a(1))
       or not a'stable(1 ns))
   report "dontcare: Orthogonality
           violation on net a"
   severity ERROR;
with a select
  d <= b when "01",
       c when "10",
       'X' when others;
```

The `a'stable(1 ns)` in the `assert` statement ensures that it is not activated in VHDL simulation while the signal `a` is settling. An `assert` statement of this form could reflect either a weak or strong assertion, as discussed in Section 2. Without the assertion, BooleDozer does an implementation that drives `d` to a `'0'` when `a` is `"00"` or `"11"`. In the presence of the assertion, however, BooleDozer is free to choose a more simplified logic implementation. BooleDozer uses a test generator and a redundancy-removal algorithm to perform the simplication, the details of which are described elsewhere [39, 40].

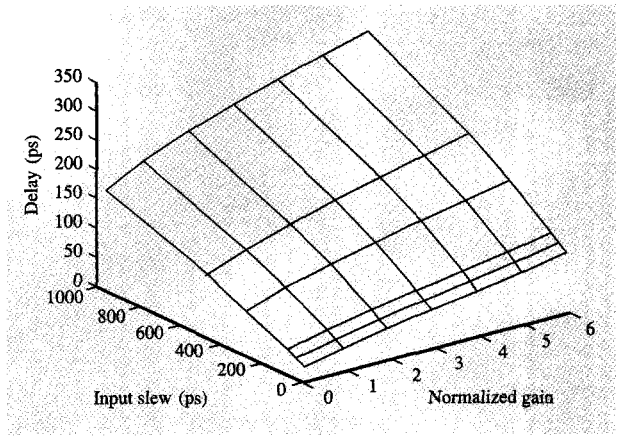


Figure 19

Inverter delay as a function of normalized gain and input slew.

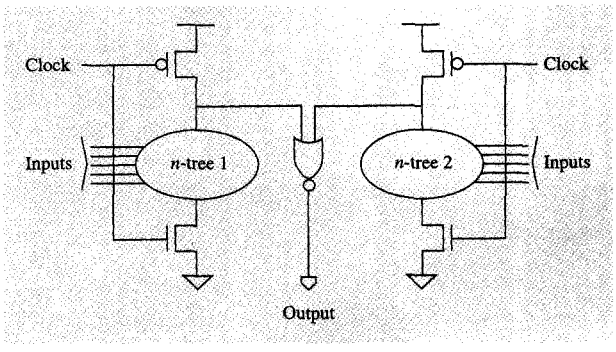


Figure 20

Complex domino gate consisting of two separate domino pulldown stacks combined in a static NAND or NOR gate.

The optimizations required by logic synthesis are very complex. As a result, most are accomplished through greedy heuristics which can never be guaranteed to produce an optimal result [14]. These heuristic timing optimizations are performed in BooleDozer after technology mapping, a stage in the synthesis process referred to as *late timing correction*. Late timing correction consists of several steps:

1. Capacitances are corrected to 200% of their specified limits through cloning and repowering.
2. Global delay optimization is performed in which all output pins with negative slack are collected. For each

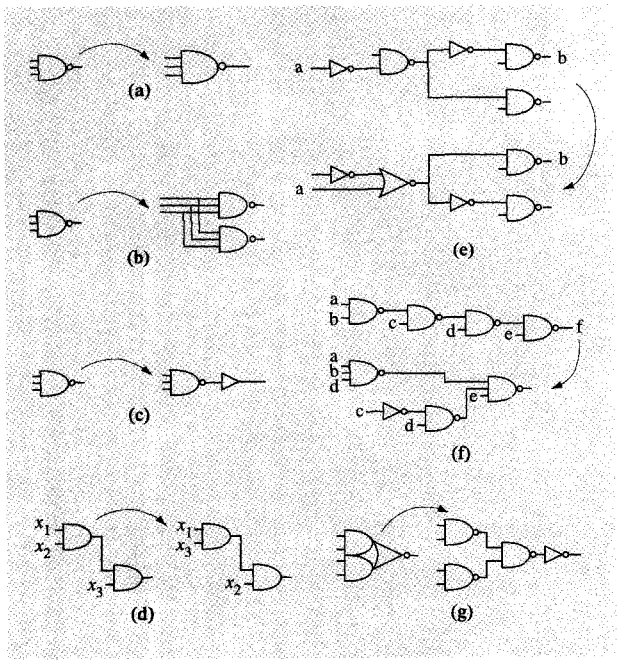


Figure 21

Delay optimization transformations: (a) Repowering; (b) cloning; (c) buffering; (d) pin swapping; (e) inverter pushing; (f) boundary moves; (g) expansion.

pass through the pin list, the delay optimization transformation that produces the greatest improvement in delay is performed. More details on the delay optimization transformations are presented below.

3. Capacitances are then corrected to 100% of their specified limits.
4. Critical-path optimization is performed. For each pass through a given critical path, the best transformation is performed on the output pin of the path that produces the best result.
5. On paths with positive slack, area is recovered where possible through repower and common-term elimination.
6. Slews are now corrected.
7. A final critical path optimization is performed.

Delay optimization transformation consists of repowering, cloning, buffering, pin swapping, inverter pushing, boundary moves, and expansion. An example of each of these is shown in **Figure 21**. Repowering means sizing a gate to achieve better timing in driving a load. Cloning, sometimes also called parallel repowering, involves duplicating a gate and dividing the fan-out between the copies. Pin swapping, also referred to as fan-in reordering, involves changing the pin assignment for commutative logic functions. The example shown in Figure 21(d) is *two-level* pin swapping, since critical signal x_2 is moved ahead one logic level in the swap. In Figure 21(e), the inverter at input a is pushed forward, resulting in fewer logic levels for the critical path from a to b. A boundary move is illustrated in Figure 21(f). In this case, a four-level NAND structure with critical path from a to f is converted to a two-level NAND structure for this critical path. Expansions may also be used to improve timing. An example of an expansion is shown in Figure 21(g). Late timing correction is also performed under "hill-climbing" conditions. This means that individual transformations are allowed to make timing worse if a succession of these transformations ultimately made timing better, allowing the heuristics to escape from locally optimal timing solutions. A checkpointing mechanism prevents the algorithm from ever ultimately producing a slower implementation.

Beta and gain parameterization in the timing rules as described above enable heuristics for delay optimization which can be applied after an initial placement of the design. Only after an initial placement can the interconnection capacitance be estimated accurately enough through minimum-width Steiner tree routes to enable detailed retuning. Timing correction in a postplacement environment must be more restrictive, incorporating only repowering, cloning, and buffering as delay optimizations. The changes that result from these postplacement optimizations are handled as an "engineering change option" (ECO) to the original placement. Details of this process are discussed in Section 7. These delay optimization transforms can be employed with the same late timing correction approach described above, with several notable exceptions enabled by beta–gain parameterization.

One notable difference is the way in which buffers are added to drive large loads from primary outputs. As part of the global delay optimization, we can calculate the *path effort* F for each critical path in the design, which we define as $F = \sqrt{C_{po}/C_{pi}}$ [36], where C_{po} is the capacitance being driven from the primary output of the macro as determined from global assertions, and C_{pi} is a reasonable input pin capacitance limit derived from the primary input resistance assertion. In addition, let n be the number of logic stages in the critical path and g_{opt} the optimal normalized gain for a given gate type (typically about 3). If $F^{1/n} > g_{opt}$, additional buffers are added to the primary output to bring $F^{1/n}$ below g_{opt}.

Repowering in the parameterized context separates gain and beta optimization as distinct optimization processes. In the case of gain, we define the *branching effort* as $b = C_{out}/C_{in}$. Then, for optimal repowering of a given gate, the normalized gain g should satisfy $gb = g_{opt}$ for minimum delay. This enables an immediate determination

of the locally optimal gate repowering, considerably improving run-time performance.

In addition to retuning, postplacement optimizations done within BooleDozer include reordering of the scan chains and optimization of the clock-distribution network within the macro. Following initial placement, which is done without regard to the scan-chain connectivity, the scan chains are reordered on the basis of latch placement to minimize the total scan-chain length. Following this reordering, the scan optimization program generates a new set of `scan_connect_out` to `scan_connect_in` VHDL signal assignments which are used in the VHDL architecture as described previously in Section 2.

Signal tagging can be used for design partitioning. In this case, signals are tagged to denote that the cone of logic associated with this signal is to be included in a specific partition. Logic in the cone is collected until a primary input or other tagged net is reached. This is used to "tag out" a piece of the design for a custom implementation or produce "submacros" for a more partitioned physical design.

The importance of a semicustom design process cannot be underestimated. It is extremely difficult to constantly adapt full-custom designs to continuous changes in the global timing and loading environment. Significant effort is now underway to expand the semicustom approach to handle multiphase domino logic implementations and the associated problems of phase assignment and convergence.

7. Chip circuit and physical design

We now consider some of the details of the circuit and physical design of the processor. The methodology follows the two-level paradigm with a macro level and a chip-integration level of design. Macro-level design consists of custom circuit and layout approaches for the dataflow stacks and arrays and the semicustom cell-based approach for control logic. Those macros implemented in the semicustom approach are referred to as random logic macros (RLMs).

- *Custom macro methodology*

There are three loosely delineated stages to the design of a custom macro—early schematic design and prototyping, interactive schematic refinement, and final schematic and layout. Custom macro design begins with a VHDL description of the logic function developed in concert with a transistor-level schematic implementation. Initial circuit and logic decisions are made with early circuit-simulation-based timing of critical cross sections. Estimates are also made for the capacitive loading at the outputs based on early chip floorplan estimates, as discussed in the subsection on chip integration physical design. An early floorplan of each custom macro is also done to ensure that sufficient area and wiring resources are available. This early physical design planning forms the basis for wire capacitance estimates placed in the schematic. "Layout-dependent" device models are also used which contain early estimates of source and drain diffusion capacitances based on predicted layout style. Once a complete schematic exists, static timing is used to verify the early cross-section selection and provide a timing abstraction to use in early global timing.

Iterative refinement of the design occurs as timing assertions are established on the basis of global timing. The timing assertion generation process was described in detail in Section 5. Area estimates are also updated as part of this process. At some point, the macro designs are "frozen." This is made known to the slack apportionment program so that all further timing improvements are required from the semicustom implementations. The macro then enters the final schematic and layout stage.

The final schematics and layout are hierarchical, with no methodology limitation on the amount of hierarchy which may be used. Layout and schematic hierarchies are encouraged, but not required, to match in order to enable hierarchical layout-to-schematic (LVS) verification. The custom circuit layout implementations used on the G4 design encompass a wide variety of layout organizations and design styles. The layout image is, in general, constrained only by the bit image of the data stacks, which specify wire usage and bit positions above first-level metal, leaving complete flexibility within technology ground rules and circuit style guidelines to FET layout and local interconnection. The layouts, of course, have to conform with shadow views from the global environment, generated using either the blockage or contract methodology described in the subsection on abstract and shadow methodology. FETs are formed from either polygons or the instantiation of parameterized device cells. Some use is made of device-level wiring tools, but most designs are wired manually, with highly regular wiring done with scripts. After circuit layout is complete, the detailed macro-level extraction is performed, as well as design rule checking (DRC) and LVS. Static timing analysis is then run directly on the extracted netlist. Final timing closure involves potential retuning of the layout.

The G4 design makes limited use of dynamic, or "weakly static," circuits. Weakly static circuits are circuits in which the dynamic node is held by a weak static half-latch device. Timing checks are done using static timing analysis. Noise is a major concern, particularly with dynamic circuits, an issue discussed in more detail in Section 9.

The arrays in the G4 design are entirely custom-designed. The use of self-resetting techniques [41] precludes use of static timing analysis. Regular structures in the arrays allow timing verification almost entirely through cross-section simulation. The timing abstractions

for the arrays are largely hand-generated from this analysis.

• *Random logic macro implementation*

Efficient implementation of the semicustom macros within performance requirements is an essential part of the G4 methodology. The goals of the semicustom implementation are twofold—provide a technique for automatically generating a complete circuit and layout from a VHDL description, while simultaneously preserving the benefits of transistor-level design. We have already discussed the parameterized libraries which were used in synthesis, in addition to a "conventional" standard-cell library. In this section, we complete the picture with discussion of the physical design of the parameterized library and discussion of the entire RLM methodology.

Parameterized cell generation

The use of parameterized cells or *soft libraries* requires development of a tool to generate layouts automatically as part of the design process. The library generator for static CMOS developed for the G4 design concentrates on efficient design of simple cells (the most complex being a 2×2 AO/OA), and allows customization of the cell image. The cell generator techniques are also being applied to complex domino logic gate implementations, as shown in Figure 20, by modularization of devices in the topology; that is, by doing the precharge devices, n-FET pull-down stack, and output stage as separate modules and combining them.

The cell generator is used in two ways. It is first used to create a standard set of sizes which are selected and shared over the entire chip, in effect creating a standard cell library with a large number of sizing options. This library is used for initial implementation and placement of all semicustom macros. In some cases, the cells are made a permanent part of the design hierarchy, matching a nonparameterized representation in the schematic. The more common approach is to tune away from the fixed library sizes. In this case, a cell library is created transiently corresponding to a user-specified "binning" of the continuously tuned schematic. After a placed-and-routed implementation from the soft library is completed, the layout is subsequently flattened, eliminating all references to the cell layout design. A parameterized schematic corresponds to this flattened layout. In this way, the original soft-library schematic and layout become part of a customized macro implementation. More details of the RLM methodology are described below.

• *Semicustom macro methodology*

The RLM methodology begins with a schematic that is created by synthesizing the VHDL description. In all cases, the initial implementation uses the fixed power-level cell set. A physical hierarchy of the design corresponding to the schematic hierarchy is constructed, using abstracts for both the standard-cell or parameterized library and custom-designed blocks embedded in the design and tagged out in the VHDL. A shadow view, containing an estimate of the macro size as well as macro pin placement, is used to create a macro floorplan. Each of the standard cells is automatically placed within circuit rows in the macro floorplan. The placement program optimizes the placement, with constraints on critical nets and routing congestion, using the Cell3** place-and-route engine. The initial placement, which is in turn given to BooleDozer to perform the postplacement optimizations, gives no weight to clock and scan-chain nets. Postplacement optimizations include reconnection of the clock distribution network, scan-chain reordering based on placement of latches, and, where necessary, continuous repowering and fan-out correction. Network changes are handled as an ECO on the initial placement solution, which is consequently routed. Upon completion of the routing, the actual routes are extracted and the design is retimed. This may result in additional retuning through ECOs. From this point, the design is tantamount to a transistor-level custom layout for all timing and electrical analysis.

In some cases, the initial placement described above is performed using timing-driven placement of critical timing paths. The intent is to limit the amount of wiring capacitance along these paths to prevent excessive area utilization in subsequent retuning and fan-out correction. The most timing-critical nets are first identified in the macro long-path timing-slack report. A target capacitance limit is calculated for each of these nets. The calculation accounts for the timing slack of the net and the net's fan-out. The capacitance limit was set progressively higher for nets which were less timing-critical or which had greater fan-out. This approach has the most benefit on large (>80000-transistor) designs.

The system of combining soft libraries with mature place-and-route technology finds application in many macros that would have otherwise been done with full-custom design. In dataflow macros, particularly those which do not have a bit-slice architecture, a significant productivity advantage is obtained by manually implementing the design with parameterized gates, tuning each gate independently to optimize the critical path, and applying soft libraries and place-and-route for the layout. The semicustom layout approach is being driven by the increasing need for early physical design to predict performance and growing difficulty in estimating capacitive load and *RC* delays from schematic representations. Technology changes are also occurring many times in a design cycle, both in the devices and in the interconnections—a growing need exists to react rapidly to these changes in the physical design. We are currently

developing the semicustom approach to handle the needs of bit-slice layouts and multicycle domino implementations.

• *Chip integration physical design*

Chip integration, the top level of the two-level circuit and physical design process, consists of floorplanning and global wiring design. The first step in the chip-level design process is to floorplan the macros, allocating macro area and optimizing pin placement. Early abstractions describe the area and aspect ratio for each of the macros to the floorplanning tool. Pin placement is determined by the desire to reduce interconnection length as well as to ease routability constraints. The estimated interconnection models used in early timing analysis consider pin placement. Early global timing is used to help discover poor macro pin placements.

Once the initial floorplan is created, power and clock are routed. One of the largest strengths of the G4 on-chip power distribution is the use of C4 [42] areal power distribution pads as opposed to wire-bonded peripheral pads. As shown in **Figure 22**, the C4 periodicity was 900 μm each direction for power and ground. Large last-metal buses were used to distribute power in a twisted fashion. Figure 22 also shows the tight grid distribution used on the other interconnection levels. The rigidity of the power grid is further discussed in Section 8. The clock tree design is a balanced H-tree structure [43] created with a specialized maze router that uses wire width as well as length tuning to achieve skew control of $\pm$25 ps while simultaneously working to reduce latency. Latency translates directly into skew when process, temperature, and voltage variations are considered. The clock tree consists of two levels, as shown in **Figure 23**. The H-tree for this clock distribution is shown in **Figure 24**. The first is the balanced tree from the central phase-locked loop (PLL) and clock driver to preplaced sector buffers. Each sector buffer is placed directly under a top-level-metal power bus to minimize both delay variations within the chip and *IR* drops in the power distribution network. A second level of balanced routing connects each sector buffer to the local macro clock generators. The main clock wires are routed on the top two interconnection levels. The top interconnection level is thick with low sheet resistivity. Accurately predicting its delay requires consideration of inductance effects. In order to reduce coupling interaction with other wires and to provide good return paths to reduce inductance, top-level interconnection clock wires are routed with adjacent supply or ground.

After power and clock routing, the I/Os are wired, as are other timing-critical buses in the design. I/O routing is done first, since I/Os typically demand last-metal interconnections in congested areas of the chip. Critical

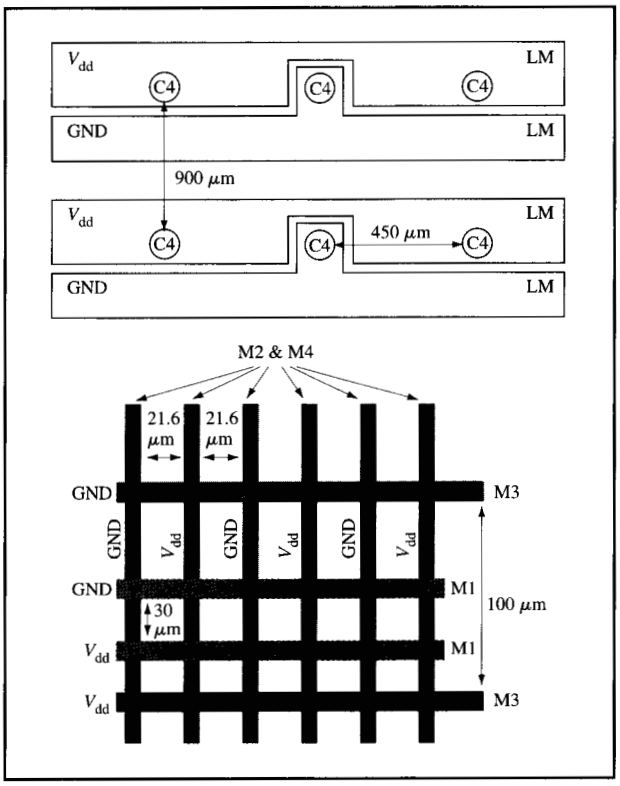


Figure 22

Power grid of the G4 processor.

bus routing is done next, with use of wide wires to minimize *RC* delays. Early critical bus routing is also done with consideration of capacitive coupling, which drives wider spacing between wires or alternate signal and power/ground routing.

• *Abstract and shadow methodology*

Once the initial floorplan with power, clock, and prewires is complete, the rest of the interconnection design is managed through the use of a hierarchical physical design process to parallelize the design effort and manage complexity. Two types of abstractions are used in the process of managing wiring resources across the macro-chip hierarchical boundary—shadows which pass wiring information from the global routes to the macros, and abstracts which pass wiring information from the macros to the globals.

Two basic methodologies are used to manage wiring resources. In the first, which we refer to as the *blockage method*, global routes are completed first, with no blockage restriction from the macro level. Shadows pass the actual global routes to the macro level of hierarchy. The shadow nets are attributed so that the macro routes

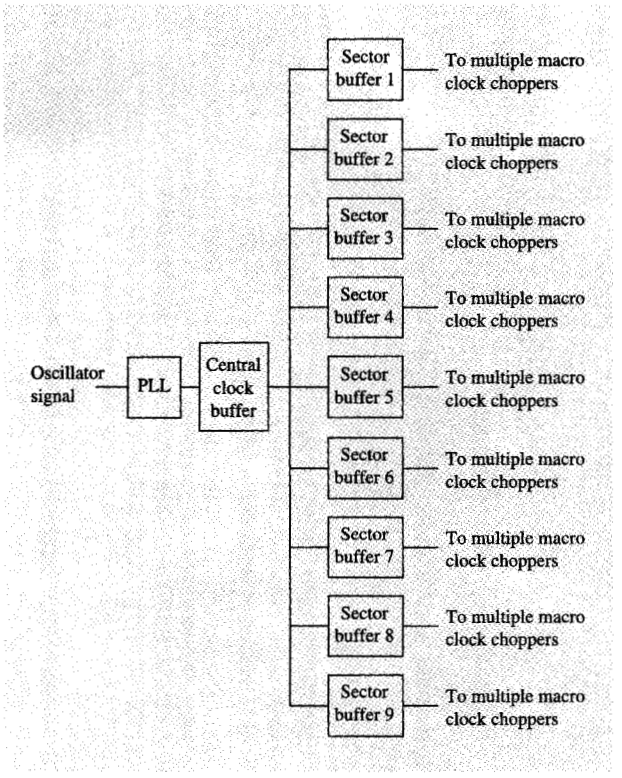


Figure 23

G4 clock distribution. The clock tree consists of two levels of buffering. A central clock buffer drives nine sector buffers which, in turn, drive local clock generators.

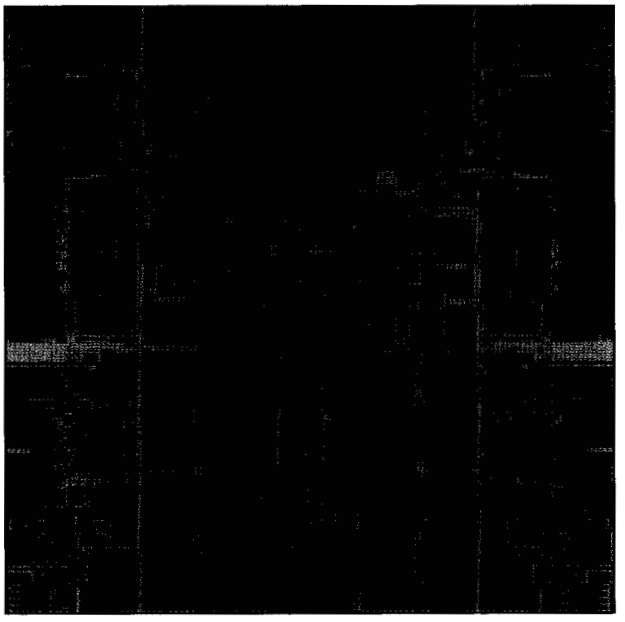


Figure 24

H-tree for clock distribution. This figure shows the actual structure of the clock tree from the central clock buffer to each of the nine sector buffers.

may tap into these where appropriate, further maximizing wiring utilization. This approach is used in selective areas of the chip where wiring resources are at a premium. In the second methodology, which we refer to as the *contract method*, the wiring tracks are divided *a priori* between the macro and global levels of hierarchy, and this contract is coded in both the shadow and abstract, which are negative images of each other. For example, if 80% of the wiring tracks are reserved for global routes, their channels will appear in the shadows as blockages. The other 20% reserved for the macro routes will appear in the abstract as blockages. Prewires—clock, power, chip I/O, and critical nets—also appear in the shadow as blockages. This technique does not create as efficient a use of wiring channels as the blockage method, but allows the parallelization of the routing process.

As the design progresses, the shadows and abstracts increasingly become representations of completed design, with net-attributed shapes representing actual routes replacing blockage shapes. These shadows and abstracts, therefore, naturally evolve into their role in parasitic extraction, as described in Section 4. In both the blockage and contract methods, abstracts frequently have *large pins*, big shapes or collections of shapes presented to the global level for connectivity to a given pin. The use of large pins helps to create flexibility in the global routing. However, this, along with the use of net-attributed shadows in the blockage method, produces situations such as those shown in **Figure 25**, in which the resulting hierarchical assignments of shapes produce difficult topologies for extraction. In Figure 25(a), a "large pin" is tapped by multiple global wires. The pin has multiple taps in the macro as well as serving as a "feedthrough" for the global routes. In Figure 25(b), a net-attributed shadow is used so that a global route has multiple tap points to the macro, more than the original number of pins in the design for this net. One possible solution to these hierarchical problems is to treat the tap points as *equivalent*, shorting them together in the resulting macro extraction, but keeping them distinct in the global extraction, choosing the best-case or worst-case global interconnection delay from among the equivalent pins.

• *Global interconnection optimization*

Managing global *RC* delays and ensuring that global net drivers are appropriately sized for their loads are essential parts of the chip integration process. These problems are

identified as slew violation at global net receivers. For those receivers showing violations, the slew at the driver is also examined. The general optimization methodology is a three-step process:

1. Where there is a slew violation at the driver, the driver is resized.
2. If the driver is appropriately sized but the receiver slew is still poor, there is excessive *RC* delay. The first approach to fixing this is to widen the wire, resizing the driver in the process to the larger capacitive load.
3. In cases where slack allows, a repeater may be used as an alternative or in addition to wide wires.

If the net is longer than ten millimeters and there is sufficient slack, insertion of repeaters is generally the preferred solution. If the net is less than ten millimeters, wire widening is generally used.

In the case of low-resistivity, last-metal interconnections, in which the wire acquires lossy transmission-line characteristics, one must also be careful not to *overdrive* the line. This can result in an impedance mismatch between the driver and the characteristic impedance of the interconnection which can produce ringing.

- *Verification of physical design*

Verification of the physical design is divided into two main areas: design rules checking (DRC) and layout versus schematic (LVS). In addition, the physical verification tools were also employed to perform such auxiliary functions as physical data compare and technology conversion.

The bulk of the DRC physical verification consisted of checking the design against the layout ground rules present in the technology guide, using manufacturing-approved rules. The same ground-rule checking tool was used at all stages of the design cycle. This required it to be robust enough to handle an entire chip's worth of data, yet with low enough overhead that it did not affect the turnaround time when checking small circuits. The checking tool was integrated into the design system, providing textual and graphical feedback upon completion of the run. This feedback was reported hierarchically, greatly reducing the amount of information that had to be acted upon by the designer. Both the textual and graphical error coordinates were normalized to those of the prime cell. This allowed for easy graphical import into the design database while simplifying the error-separation process. In addition to the ground-rule checks, there were other rule files available; these checked for potential design problems that were not specifically ground-rule-related, such as degenerate shapes, or non-technology-specific ground rules, such as I/O checks.

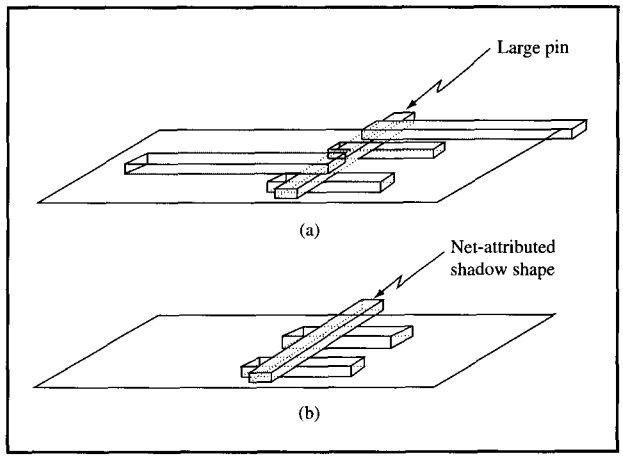


Figure 25

Hierarchical problems in extraction: (a) Large macro pin connects to multiple wires representing the same global net. (b) Net-attributed shadow shape is tapped in multiple places by macro nets.

Like DRC, the same LVS tool was used at all levels of the design hierarchy. The LVS submission tool was also completely integrated into the design system. In addition to the physical layout, LVS requires a schematic representation of the circuit as an input. This automated procedure is accomplished by taking the output of the electrical simulation and processing the text. Because of the wide variety of design styles present on the chip, LVS had to be flexible enough to check each correctly. This was facilitated by allowing the designers to select from a wide variety of options when submitting LVS jobs, allowing them to tailor the checking job to meet the specific requirements of their designs. One of the most important features of the LVS methodology was its ability to handle different design flows. Both top-down and bottom-up designs could be submitted and checked from within the design system. The top-down approach required some data preprocessing. Because of this, a macro designer could successfully check any combination of circuits and black boxes; a processor subunit integrator could check any combination of macros and black boxes; and a chip integrator could check any combination of processor subunits and black boxes. Having this ability permitted wiring errors to be caught early in the design cycle, thus greatly reducing the amount of rework.

Having a well-structured hierarchy was an important factor in the successful operation of the LVS program. It allowed for faster run times and more concise diagnostics. This dependence on hierarchy was a drawback, however, when dealing with the I/O terminating resistors. Because these resistors had to be tuned with regard to chip wiring,

they could not be assigned values until the end of the chip integration cycle. Placement of these resistors at the top level of the design disrupted the hierarchy to a considerable extent. It would not have been possible to LVS-check the design without a lengthy rework of the chip schematic. To get around this problem, a series of checking programs were run outside the boundaries of LVS to ensure that the resistors were placed correctly, while LVS was told to ignore the resistors in the network compare.

Besides DRC and LVS checking, the verification tool set was also employed to handle some special cases. Between major design releases, it was expedient to repair design bugs by modification of personalization levels only. In cases such as these, a layout data comparison program was used to ensure that only the personalization levels were altered, and that the silicon and local interconnection layers remained unchanged, allowing for a savings in the cost of these masks.

An even more significant use of the verification tools involved the conversion of existing design data to a more aggressive set of technology ground rules. This was accomplished in a manner completely transparent to the design team. They designed the entire chip in one technology, and the final data were transformed to the new technology. This resulted in cutting several months off the time it would have taken for the design team to convert to the new technology by hand. Such a methodology required a more extensive set of verification requirements, since it was necessary for the design data to be completely DRC- and LVS-clean for both the designed and converted technologies.

The final design data were checked completely using the existing technology ground rules. A two-stage conversion program was then employed to convert the data to the new technology. A single conversion program was initially used. Although this produced a ground-rule-correct design, the conversion caused considerable disruption in the chip hierarchy because of a special local interconnection level. This level had a large amount of interhierarchy interactions, leading to resultant shapes that had to be propagated several layers upward in the hierarchy. Using the current checking tools, it was impossible to run a complete LVS on the final design. A cell-by-cell approach was then tried, but this led to ground-rule violations on the local interconnection level. The conversion program was then converted into the two-stage approach. The first stage performed the bulk of the shapes conversion on a cell-by-cell basis. A hierarchical conversion was then performed on a portion of the local interconnection layer shapes that were in violation of the technology ground rules. The final data could be checked with all of the same rules used for the initial nonconverted design.

8. Electrical analysis

We now describe the methodology used to analyze the power demands of the chip as well as determine the power supply noise and the electromigration reliability of the power network.

- *Power calculation*

Static algorithms, such as those applied to timing analysis, rely on simulations at the gate level combined with a graph-based path search. The fundamental assumption of this approach is that correct characterization for the analysis in question can be done at the level of individual channel-connected components. This is, unfortunately, not true for determining the power demands of digital systems. Straightforward static analysis is stubbornly defied by the existence of glitches and incomplete transitions at gate inputs, the dependence of energy consumption on past history, and sensitivity of energy demands to the precise analog nature of waveshapes in the design [44]. As a result, the approach we take is an essentially two-level hierarchical analysis methodology, in which large, flat macro-level simulations are abstracted and combined statically to analyze the energy demands of the chip and determine the integrity of the power distribution network.

We perform the macro circuit simulation using the SPECS (Simulation Program for Electronic Circuits and Systems) simulator [45]. In SPECS, the simulation is event-driven; *IV* characteristics are assumed to be piecewise constant, branch currents are assumed to be piecewise constant in time, and branch voltages are assumed to be piecewise linear in time. The clocks are toggled at system cycle time and patterns are applied with arrival times relative to the clock as determined from static timing analysis. Output loading is also obtained from the global timing environment. Patterns are either designer-chosen to maximize the power requirements of the circuit, or randomly generated. For smaller macros (<50000 transistors), hundreds of patterns are analyzed, while for larger macros (50000–200000 transistors), tens of patterns are simulated. We define a set of *power points*, pins within the power and ground network that separate the "local" power distribution from the global one. These are typically chosen on the via layer that connects the first- and second-level metal. The macro, including the local power grid up to the power point, is extracted. Current meters are attached to the power points in the extracted netlist, as shown in **Figure 26** for simulation. A fundamental assumption of this hierarchical approach to power analysis is that the global supply and ground can be assumed to have their nominal values when calculating the power-point currents. In actuality, macro power demands result in power supply noise which in turn affects the

power-point currents, an effect which is ignored in this analysis.

In order to abstract macro power data both temporally and spatially, we monitor the currents at the power points during simulation. The active edge of the global clock defines the cycle. Let $i_n^{peak}(m)$ be the peak current on power point n during cycle m, and let $i_n^{average}(m)$ be the average current on power point n over cycle m. We then find the cycle m for which

$$\sum_n i_n^{peak}(m)$$

is maximum, and the cycle m' for which

$$\sum_n i_n^{average}(m')$$

is maximum. We then store the $i_n^{peak}(m)$ and $i_n^{average}(m')$ values for each power point as a *power view*. Additional temporal resolution is possible by dividing the cycle into a number of "time buckets."

From logic simulation, we determine a switching factor f between 0 and 1 for each macro in the design during "average" and "worst-case" activity. We define f as the average fraction of inputs that change during a given machine cycle. **Figure 27** shows a power map of the chip for average switching activity, calculated by computing

$$V_{dd} f \sum_n i_n^{average}(m'),$$

where the sum is over all of the power points in the power view for the given macro.

• *IR/EM analysis*

The power abstracts are also used to determine the power-supply noise and evaluate electromigration constraints in the power distribution. Both analyses begin with an extraction of the multilevel power distribution network from the macro power points to the C4 pads. A full *RC* extraction of the power grid is performed [46]. The power-grid extraction includes the widths and via sizes associated with each resistor in the extraction. For the on-chip inductance extraction, the mesh plane of the multichip module (MCM) package is used as the ground plane [47].

We first perform a dc analysis of the power distribution to determine the *IR* drops in the power and ground distribution. For this analysis, we use the average power-point current and the "worst-case" switching factors to apply dc current sources to the global power and ground grids. The resulting resistive network is solved with a sparse LU factorization package. *IR* drop results are calculated for each power point, and branch currents are calculated for each resistor in the network. The branch currents are cross-referenced with the wire widths and via sizes to flag potential electromigration problems. If we

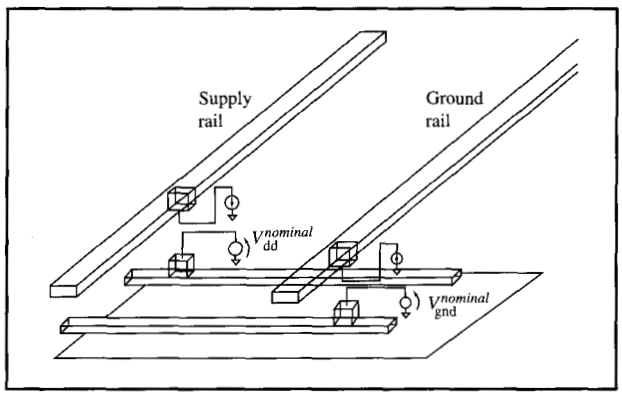


Figure 26

Power-point methodology. The supply and ground distribution is divided between macro (black) and global (red). Power points form the connection between these two levels of hierarchy. Independent voltage sources that supply nominal supply and ground voltages to the macro serve as current meters. Peak and average currents measured at these meters are subsequently applied to the global power grid to determine *IR* drops and delta-*I* noise.

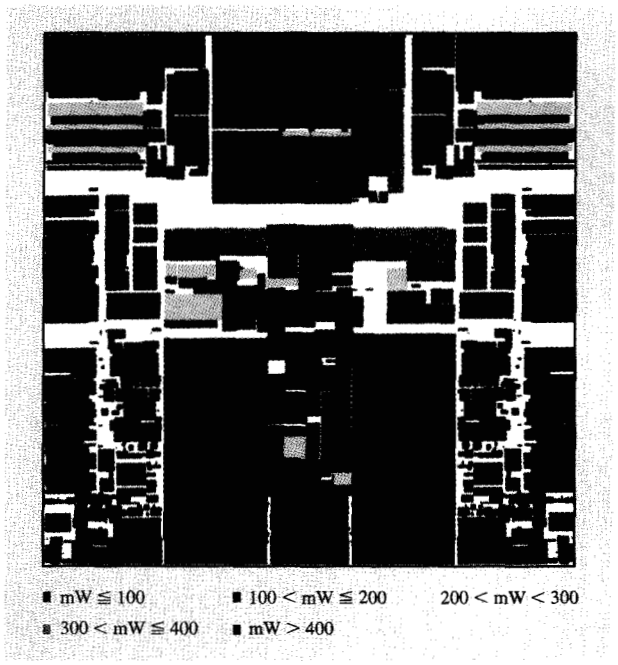


Figure 27

Power dissipation map of the G4 processor.

define the applied voltage $V_{applied}$ as the difference between the power and ground voltage, $V_{applied}^{nominal}$ is the nominal applied voltage, $V_{applied}^{average}$ is the average applied voltage, and $V_{applied}^{max}$

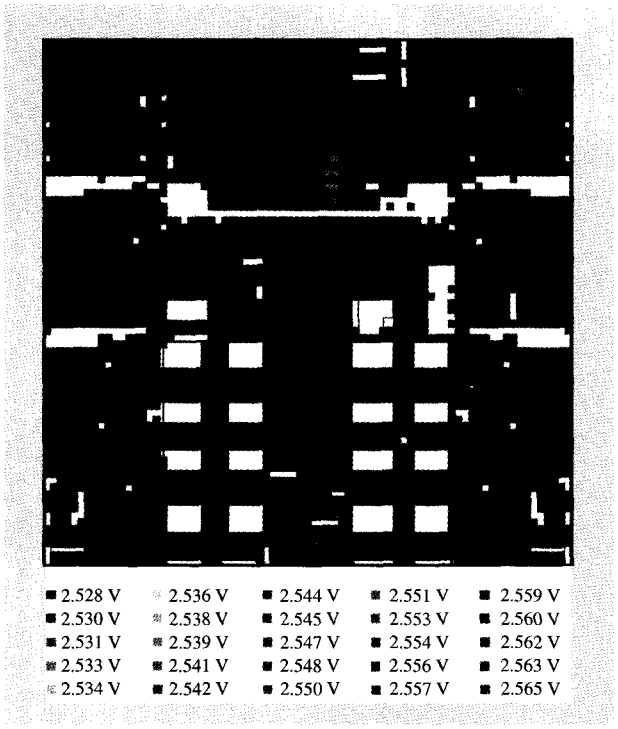


Figure 28

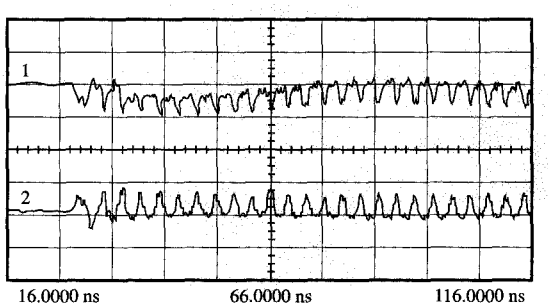
Power supply map of the G4 processor.

is the maximum value of the applied voltage. **Figure 28** shows V_{applied} calculated in this manner using $i_n^{average}(m')$ current values at each power point and average f values. The maximum applied voltage drop, $V_{\text{applied}}^{max} - V_{\text{applied}}^{nominal}$, is 37 millivolts, and the average applied voltage drop, $V_{\text{applied}}^{average} - V_{\text{applied}}^{nominal}$, is 10 millivolts.

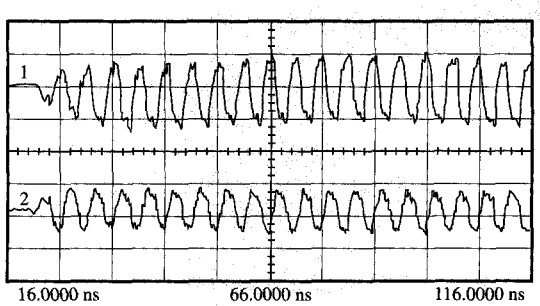
In addition to the variations in the dc power and ground levels due to the steady-state current demands of the chip, there are periodic variations due to simultaneous switching of off-chip drivers and internal circuits. This delta-*I* noise occurs when these "pulses" of current are sourced or sinked through inductance on the chip and package supply and ground wires. **Figure 29** shows delta-*I* noise on the supply and ground as actually *measured* on-chip. To analyze the delta-*I* noise, the power-point sources are applied to the complete *RLC* extraction of the power grid combined with a lumped-element model of the MCM. **Figure 30** shows a highly simplified view of this model for a single power/ground C4 pair. The current sources at the power points are assumed to switch as a spike with a slew time of 100 ps rising and falling and with a magnitude given by the peak current of the power point. Decoupling capacitors are added to the equivalent circuit, as are estimates of n-well and nonswitching circuit capacitance.

9. Noise analysis

We begin the discussion with a couple of definitions. An *evaluation node* is a circuit node that forms the connection between channel-connected components, or gates, in the design. *Noise*, therefore, is defined as anything that causes the voltage of an evaluation node to deviate from the nominal supply or ground rails when the node should represent a stable logic '0' or '1' value. We follow Reference [5] in characterizing noise sources by peak magnitudes relative to the nominal supply and ground rails. Noise sources that reduce an evaluation node voltage below the supply level (V_{dd}) are denoted V_H, while noise sources that increase an evaluation node voltage above the ground level are denoted V_L. Noise may also be bootstrapping if it increases a node voltage above the supply level (V_{H^*}) or below the ground level (V_{L^*}). Noise sources relevant to digital design include leakage noise, power-supply noise, charge-sharing noise, and crosstalk



(a)



(b)

Figure 29

Measured delta-*I* noise on the G4 microprocessor at a clock cycle time of (a) 3.5 ns; (b) 5 ns.

noise. Noise has two deleterious effects in digital systems: It can produce logic failures by causing a latch to falsely change state and can also have a direct effect on delay. Leakage noise and power-supply noise result in lower or higher supply levels, which reduce or enhance the current drive of a circuit and consequently increase or decrease the delay. Coupling noise can cause the effective line capacitance to increase or decrease in the presence of simultaneously switching noisy lines, increasing or decreasing the delay. We refer to this as the *interconnection Miller effect*.

The G4 design methodology analyzed only coupling noise at the global level and did so in a limited way using the capacitance-only coupling extraction of the global interconnection discussed in Section 5. We define the *victim* net as the static net onto which pulse noise is being coupled by one or more *perpetrator* nets. Coupling noise was calculated using the simple linear model shown in **Figure 31**. A threshold of coupling capacitance to victim self-capacitance was used to decide which perpetrator nets to include in the analysis. R_{driver} is the effective resistance of the driver. The "resistance" R_k of an individual FET k is modeled from the linear region of the I_{ds} versus V_{ds} current–voltage characteristic at $|V_{\text{gs}}| = V_{\text{dd}}$. R_{driver} is then

$$R_{\text{driver}} = \sum_k R_k$$

over the weakest static FET path in the driver. R_{net} is the total resistance of the net to the receiver. C_{ground} is the total capacitance of the victim net which is tied to ground. Capacitances C^i_{coup} couple the victim to each of the perpetrator sources v^i_{perp} which are modeled as saturate ramp waveforms of slew t^i_{slew}. This network ignores the distributed effects of resistance on the victim net. Instead, the entire net resistance is put in series with the driver, a pessimistic simplifying assumption. In addition, the distributed resistance of the perpetrator nets is also ignored. The $v_{\text{noise}}(t)$ response produced by the action of a single perpetrator source $v^i_{\text{perp}}(t)$ is given by

$$v_{\text{peak}} = \begin{cases} \dfrac{RV_{\text{dd}}}{t^i_{\text{slew}}}\left[1 - e^{-t/[R(C_{\text{coup}}+C_{\text{ground}})]}\right] & \text{for} \quad 0 \le t \le t^i_{\text{slew}}, \\ \dfrac{RV_{\text{dd}}}{t^i_{\text{slew}}}\left[1 - e^{-t^i_{\text{slew}}/[R(C_{\text{coup}}+C_{\text{ground}})]}\right] e^{-(t-t^i_{\text{slew}})/R(C_{\text{coup}}+C_{\text{g}})} & \text{for} \quad t \ge t^i_{\text{slew}}, \end{cases}$$

where $R = R_{\text{net}} + R_{\text{driver}}$. By superposition, the peak noise v_{peak} is given by

$$v_{\text{peak}} = \sum_{i=1}^{n} \frac{RC_{\text{coup}}V_{\text{dd}}}{t^i_{\text{slew}}}\left[1 - e^{-t^i_{\text{slew}}/[R(C_{\text{coup}}+C_{\text{ground}})]}\right].$$

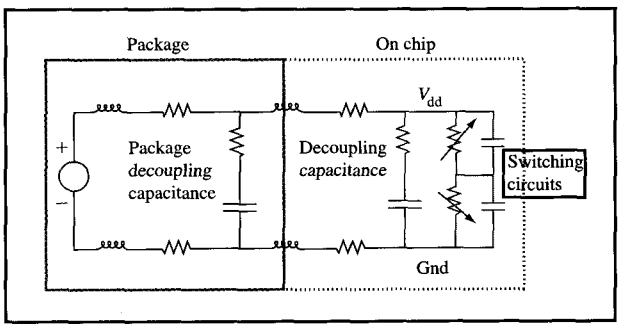


Figure 30

Equivalent circuit for delta-*I* calculations.

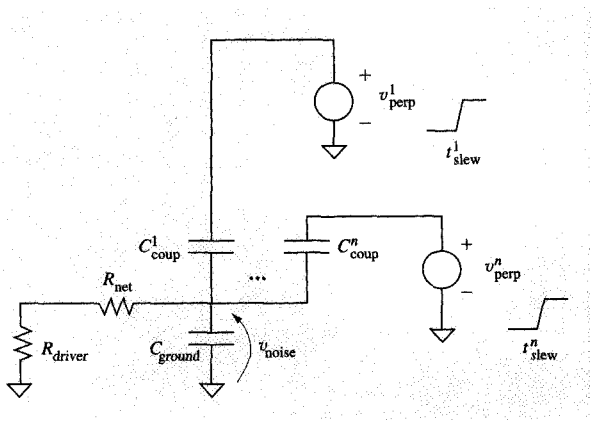


Figure 31

Simple circuit model for crosstalk coupling.

In addition, both timing windows and logical constraints were used to reduce pessimism. Arrival windows for the perpetrator net signals are obtained from static timing analysis and are defined by early- and late-mode propagation at the same process, temperature, and voltage. We then solve what we call the *optimal control problem* for the arrival times of the perpetrator net driver waveforms. We seek to find the arrival times for the voltage waveforms of the perpetrator net drivers which meet the arrival time constraints and which maximize the peak noise response on each victim net receiver. Some proactive attempts to avoid coupling problems were made on global routes with constraint-driven routing techniques [48].

Receiver sensitivity to noise was characterized by circuit topology. Latch topologies on receivers of long nets were naturally ascribed lower acceptable v_{peak} values than static CMOS gates. Latches with n-FET pass-gate inputs have an

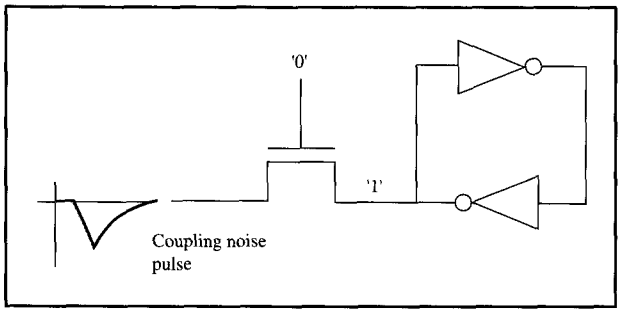


Figure 32

V_{L^*}-sensitive pass-gate latch structure. A V_{L^*} coupling even greater than the threshold voltage V_T causes the pass gate to turn on and switch the latch, with no possibility of recovery.

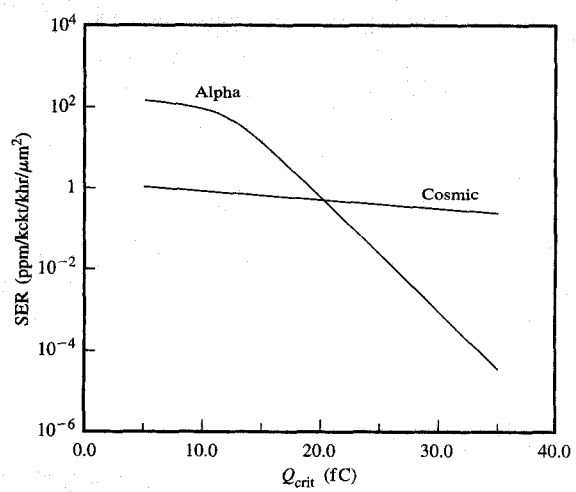


Figure 33

Soft-error rate as a function of the critical charge Q_{crit} for both alpha-particle and cosmic-ray sources for a representative 0.35-μm technology.

additional sensitivity to V_{L^*}[2] noise. Consider the case in which the gate of the pass gate is '0' and the latch stores a logic '1', as shown in **Figure 32**. A V_{L^*} coupling event greater than the threshold voltage V_T causes the pass gate to turn on and switch the latch. When the noise pulse disappears, the pass gate is off, and no mechanism exists to return the latch to its correct logic '1' state.

In development is a more detailed noise methodology to provide protection at the macro and global levels against glitch-induced logic transitions for all possible noise

[2] p-FET pass-gate inputs would have a similar sensitivity to V_{H^*} noise.

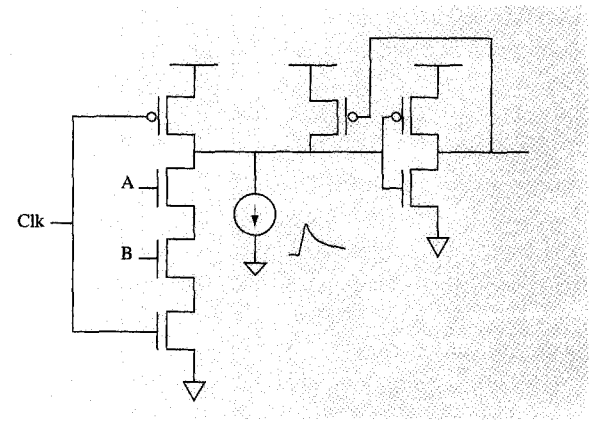


Figure 34

Equivalent circuit for determining Q_{crit} for soft-error failures of a dynamic circuit. A pulse current source with total integrated charge of Q_{crit} is applied.

sources [5]. The approach defines a metric of quality for noise known as *noise stability*. The global interconnection analysis takes advantage of the multiport impedance macromodels described in Section 4 and will include calculation of the effects of coupling on delay.

- *Soft-error rates*

Soft errors are a leakage noise source caused by ionizing radiation, which generates minority carriers in the n-well or substrate that are collected at reverse-biased source–drain diffusions. These can produce failures in dynamic circuits, latches, and RAM cells. There are two main sources of this ionizing radiation: cosmic rays and alpha particles produced by radioactive decay of lead in C4 package technology. The impact of ionizing radiation on circuits is determined by measuring the critical charge, Q_{crit}, which will produce failure in a given circuit. Smaller feature sizes brought about by technology scaling mean smaller capacitances and lower values of Q_{crit}. **Figure 33** shows the error rate for this circuit for both alpha and cosmic rays in parts per million per thousand bits per thousand hours of usage per μm^2 of diffusion area. Adjustments in the failure rate are also made in the presence of error correcting.

To determine Q_{crit}, circuits susceptible to soft-error leakage are analyzed with an equivalent circuit similar to that shown in **Figure 34**. A pulse current source is applied to the evaluation node of the circuit so that the total integrated charge of the current pulse is Q_{crit}. For p diffusions, this charge is positive. For n diffusions, this charge is negative. A typical pulse shape for this analysis

has a 5-ps rise time and 30-ps time constant exponentially delayed falling edge.

10. Conclusions

In this paper we have reviewed the philosophies, techniques, and processes used in the design of the S/390 Parallel Enterprise Server G4 microprocessor. In doing so, we have emphasized some of the guiding themes of our approach. Cycle simulation is an essential element of any verification effort, and a methodology must exist to map the design from an event-driven HDL into a cycle-simulation model or, in many cases, into multiple cycle-simulation models. The design methodology must be fundamentally transistor-level to allow detailed optimization trade-offs among timing, power, and noise. At the same time, to manage the complexity, a consistent two-level hierarchical approach must be used for all key analysis processes, with design abstractions stored and controlled from a common database. Static techniques must be employed for these analyses wherever possible. In addition, one of these analyses must be noise, which has acquired overwhelming importance with technology scaling. Technology trends also mean that interconnections must be designed and analyzed with comparable importance to devices.

Acknowledgments

The authors gratefully acknowledge all of the members of the G4 design team and other individuals throughout IBM for their contributions to the methodology. We would like particularly to thank Leon Stok, Reinaldo Bergamaschi, Dan Brand, Andreas Kuehlmann, Chandu Visweswariah, Gary Ditlow, Joachim Clabes, Izzy Bendrihem, Andrew Sullivan, Nate Hieter, John Beatty, Pete Elmendorf, Alex Suess, Vinod Narayanan, Kelvin Lewis, Phil Restle, Mike Scheuermann, David Kung, Prabhakar Kudva, Lisa Lacey, Dan Beece, Allan Dansky, Pat Williams, Keith Barkley, Dennis Merrill, Scott Nealy, Steve Walker, Howard Chen, Dan Knebel, Steve Washburn, Kwok Eng, Larry Lange, and Paul Villarrubia.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Cadence Design Systems, Inc. or Synopsys, Inc.

References

1. C. F. Webb, C. J. Anderson, L. Sigal, K. 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, T. J. Slegel, W. V. Huott, Y. H. Chan, B. Wile, P. G. Emma, D. K. Beece, C. T. Chuang, and C. Price, "A 350 MHz S/390 Microprocessor," *1997 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 168–169, 449.
2. A. Cao, A. Adalal, J. Bauman, P. Delisle, P. Dedood, P. Donehue, M. Dell'OcaKhouja, T. Doan, M. Doreswamy, P. Ferolito, O. Geva, D. Greenhill, S. Gopaladhine, J. Irwin, L. Lev, J. MacDonald, M. Ma, S. Mitra, P. Patel, A. Prabhu, R. Puranik, S. Rozanski, N. Ross, P. Saggurti, S. Simovich, R. Sunder, B. Sur, W. Vercruysse, M. Wong, P. Yip, J. Zhou, and G. Zyner, "CAD Methodology for the Design of UltraSPARC-I Microprocessors at Sun Microsystems, Inc." *Proceedings of the ACM/IEEE Design Automation Conference*, 1995, pp. 19–22.
3. C. Roth, R. Lewelling, and T. Brodnax, "The PowerPC 604 Microprocessor Design Methodology," *Proceedings of the 1994 International Conference on Computer-Aided Design*, pp. 404–408.
4. W. V. Huott, T. J. Koprowski, B. J. Robbins, M. P. Kusko, S. V. Pateras, D. E. Hoffman, T. G. McNamara, and T. J. Snethen, "Advanced Microprocessor Test Strategy and Methodology," *IBM J. Res. Develop.* **41,** No. 4/5, 611–627 (1997, this issue).
5. K. L. Shepard and V. Narayanan, "Noise in Deep Submicron Digital Design," *Proceedings of the 1996 International Conference on Computer-Aided Design (ICCAD '96)*, San Jose, CA, November 1996, pp. 524–531.
6. *IEEE Standard VHDL Language Reference Manual*, IEEE Standard 1076–1987, IEEE Standards Board, 345 E. 47th St., New York, NY 10017, 1988.
7. C. Maxfield, "Digital Logic Simulation: Event-Driven, Cycle-Based, and Home-Brewed," *Electron. Design News* **41,** No. 14, 129–130, 132, 134, 136 (1996).
8. D. K. Beece, G. Deibert, G. Papp, and F. Villante, "The IBM Engineering Verification Engine," *Proceedings of the 25th Design Automation Conference*, Anaheim, CA, 1988, pp. 218–224.
9. C. E. Shannon, "A Symbolic Analysis of Delay and Switching Circuits," *Trans. AIEE* **57,** 713–723 (1938).
10. G. Even, I. Y. Spillinger, and L. Stok, "Retiming Revisited and Reversed," *IEEE Trans. Computer-Aided Design of Integrated Circuits & Syst.* **15,** No. 3, 348–357 (1996).
11. A. Kuehlmann, A. Srinivasan, and D. P. LaPotin, "Verity—A Formal Verification Program for Custom CMOS Circuits," *IBM J. Res. Develop.* **39,** No. 1/2, 149–165 (1995).
12. D. P. Appenzeller and A. Kuehlmann, "Formal Verification of a PowerPC Microprocessor," *Proceedings of the International Conference on Computer Design*, Austin, TX, October 1995, pp. 79–84.
13. Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits & Syst.* **5,** 677–691 (1986).
14. D. Brand, R. F. Damiano, and A. D. Drumm, "In the Driver's Seat of BooleDozer," *Proceedings of the International Conference on Computer Design*, Austin, TX, October 1994, pp. 518–521.
15. R. A. Bergamaschi, R. A. O'Connor, L. Stok, M. Z. Moricz, S. Prakash, A. Kuehlmann, and D. S. Rao, "High-Level Synthesis in an Industrial Environment," *IBM J. Res. Develop.* **39,** No. 1/2, 131–148 (1995).
16. A. Kuehlmann, D. I. Cheng, A. Srinivasan, and D. P. LaPotin, "Error Diagnosis for Transistor-Level Verification," *Proceedings of the 31st ACM/IEEE Design Automation Conference*, 1994, pp. 218–224.
17. W. T. Weeks, "Calculation of Coefficients of Capacitance of Multiconductor Transmission Lines in the Presence of a Dielectric Interface," *IEEE Trans. Microwave Theory Tech.* **MT-18,** 35–43 (1970).
18. Peter R. O'Brien and Thomas L. Savarino, "Efficient On-Chip Delay Estimation for Leaky Models of Multiple-Source Nets," *Proceedings of the Custom Integrated Circuits Conference*, 1990, pp. 9.6.1–9.6.4.
19. L. T. Pillage and R. A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. Computer-*

Aided Design of Integrated Circuits & Syst. **9,** No. 4, 352–366 (1990).
20. K. L. Shepard, V. L. Narayanan, P. C. Elmendorf, and G. Zheng, "Global Harmony: Coupled Noise Analysis for Full-Chip RC Interconnect Networks," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1997, to be published.
21. R. W. Freund and P. Feldmann, "Reduced-Order Modeling of Large Passive Linear Circuits by Means of the SyPVL Algorithm," *Proceedings of ICCAD '96*, San Jose, CA, November 1996, pp. 280–287.
22. A. Deutsch, G. V. Kopcsay, C. W. Surovic, B. J. Rubin, L. M. Terman, Jr., R. P. Dunne, T. A. Gallo, and R. H. Dennard, "Modeling and Characterization of Long On-Chip Interconnections for High-Performance Microprocessors," *IBM J. Res. Develop.* **39,** No. 5, 547–567 (1995).
23. H. B. Bakoglu, *Circuits, Interconnects, and Packaging for VLSI*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1990.
24. K. L. Shepard, "Practical Issues of Interconnect Analysis in Deep Submicron Integrated Circuits," *Proceedings of the IEEE International Conference on Computer Display*, October 1997; to be published.
25. H. Liao, W. W.-M. Dai, R. Wang, and F.-Y. Chang, "S-Parameter Based Macro Model of Distributed-Lumped Networks Using Exponentially Decayed Polynomial Function," *Proceedings of the Design Automation Conference*, 1993, pp. 726–731.
26. H. Liao and W. W.-M. Dai, "Partitioning and Reduction of RC Interconnect Networks Based on Scattering Parameter Macromodels," *Proceedings of the International Conference on Computer-Aided Design*, 1995, pp. 704–709.
27. Vivek Raghavan, J. Eric Bracken, and Ronald A. Rohrer, "AWESpice: A General Tool for the Accurate and Efficient Simulation of Interconnect Problems," *Proceedings of the 29th ACM/IEEE Design Automation Conference*, Anaheim, CA, June 1992, pp. 87–92.
28. Seok-Yoon Kim, Nanda Gopal, and Lawrence T. Pillage, "Time-Domain Macromodels for VLSI Interconnect Analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits & Syst.* **13,** 1257–1270 (1994).
29. R. B. Hitchcock, G. L. Smith, and D. D. Cheng, "Timing Analysis for Computer Hardware," *IBM J. Res. Develop.* **26,** 100–105 (1982).
30. T. M. Burks, K. A. Sakallah, and T. N. Mudge, "Critical Paths in Circuits with Level-Sensitive Latches," *IEEE Trans. VLSI Syst.* **3,** No. 2, 273–291 (1995).
31. V. Narayanan, B. A. Chappel, and B. M. Fleischer, "Static Timing Analysis for Self-Resetting Circuits," *Proceedings of the 1996 International Conference on Computer-Aided Design*, pp. 119–126.
32. M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, "SubGemini: Identifying Subcircuits Using a Fast Subgraph Isomorphism Algorithm," *Proceedings of the 1993 Design Automation Conference*, pp. 31–37.
33. *Draft of Procedural Interface and DCL Language*, CAD Framework Initiative, Inc., Austin, TX 78759, 1995.
34. Curtis L. Ratzlaff, Satyamurthy Pullela, and Lawrence T. Pillage, "Modelling the RC-Interconnect Effects in a Hierarchical Timing Analyzer," *Proceedings of the Custom Integrated Circuits Conference*, 1992, pp. 15.6.1–15.6.4.
35. E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic Decomposition During Technology Mapping," *Proceedings of the 1995 International Conference on Computer-Aided Design*, November 1995, pp. 264–265.
36. Ivan Sutherland and Robert Sproull, "The Theory of Logical Effort: Designing for Speed on the Back of an Envelope," in *Advanced Research in VLSI*, University of California at Santa Cruz, 1991.
37. D. S. Kung, R. F. Damiano, T. A. Nix, and D. J. Geiger, "BDDMAP: Technology Mapper Based on a New Covering Algorithm," *Proceedings of the 1992 Design Automation Conference*, pp. 484–487.
38. D. Brand, R. A. Bergamaschi, and L. Stok, "Be Careful with Don't Cares," *Proceedings of the 1995 International Conference on Computer-Aided Design*, pp. 83–86.
39. R. A. Bergamaschi, D. Brand, L. Stok, M. Berkelaar, and S. Prakash, "Efficient Use of Logic Don't Cares in High-Level and Logic Synthesis," *Proceedings of the 1995 International Conference on Computer-Aided Design*, pp. 272–278.
40. D. Brand, "Redundancy and Don't Cares in Logic Synthesis," *IEEE Trans. Computers* **C-32,** 947–952 (1983).
41. 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* **26,** No. 11, 1577–1585 (1991).
42. L. F. Miller, "Controlled Collapse Reflow Chip Joining," *IBM J. Res. Develop.* **13,** No. 3, 239–250 (1969).
43. H. B. Bakoglu, J. T. Walker, and J. D. Meindl, "A Symmetric Clock Distribution Tree and Optimized High Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits," *Proceedings of the IEEE International Conference on Computer Design*, 1986, pp. 118–122.
44. D. Brand and C. Visweswariah, "Inaccuracies in Power Estimation During Logic Synthesis," *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1996, pp. 388–394.
45. C. Visweswariah and R. A. Rohrer, "Piecewise Approximate Circuit Simulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits & Syst.* **10,** 861–870 (1991).
46. H. H. Chen, "Minimizing Chip-Level Simultaneous Switching Noise for High-Performance Microprocessor Design," *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1996, pp. 544–547.
47. B. J. Rubin, "An Electromagnetic Approach for Modeling High-Performance Computer Packages," *IBM J. Res. Develop.* **34,** No. 4, 585–600 (1990).
48. E. Malavasi, E. Charbon, E. Feit, and A. Sangiovanni-Vincentelli, "Automation of IC Layout with Analog Constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits & Syst.* **15,** No. 8, 923–942 (1996).

Received January 14, 1997; accepted for publication July 30, 1997

Kenneth L. Shepard *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (shepard@vnet.ibm.com).* Dr. Shepard received a B.S.E. in electrical engineering from Princeton University in 1987, and an M.S. and Ph.D. in electrical engineering from Stanford University in 1988 and 1992, respectively. He is currently a Research Staff Member and Manager at the IBM Thomas J. Watson Research Center in the VLSI Design Department, where he has worked since joining IBM in 1992. He is also an adjunct assistant professor of electrical engineering at Columbia University.

Sean M. Carey *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (scarey@vnet.ibm.com).* Mr. Carey has been employed by IBM Poughkeepsie since 1988. He has worked in the hardware development area on several S/390 projects, with emphasis on timing methodology support and development. Mr. Carey received a B.S.E.E. degree from Clarkson University in 1988 and is currently working on his M.S.E.E. from Syracuse University.

Ee Kin Cho *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (ee@vnet.ibm.com).* Mr. Cho has been working in the hardware development area in the System/390 Division since he joined IBM in 1981. He has worked in design verification and tools support areas on various S/390 projects. Mr. Cho received a B.S. degree in electrical engineering from the University of Massachusetts in 1981 and an M.S. degree in computer science from Union College in 1990.

Brian W. Curran *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (bcurran@vnet.ibm.com).* Mr. Curran received a B.S. degree in electrical engineering from the University of Wisconsin at Madison in 1984, joining the IBM Data Systems Division (now System/390 Division) that same year. Mr. Curran has designed logic and circuits for large system processors and memory subsystems. He was a member of the teams which developed the System/3090 and ES/9121 processors, in addition to his work on the G4 design. Mr. Curran has received several IBM Technical Achievement Awards; he has been issued seven patents relating to processor design and has three patents pending. He is currently a Senior Engineer in the Poughkeepsie Development Laboratory and is a technical leader in the development of a future high-frequency full-custom CMOS microprocessor.

Robert F. Hatch *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (bhatch@vnet.ibm.com).* Mr. Hatch is a Senior Engineer in the Custom Design Department of the S/390 Division. He received a B.S. degree in engineering from Southern Illinois University in 1976 and an M.S. degree in electrical engineering from Purdue University in 1978. Mr. Hatch joined IBM in 1978 at the East Fishkill facility, where he did custom PLA circuit design for a custom VLSI bipolar CPU chip set. His work has been in the areas of bipolar circuit design, MOS circuit design, macro design, VLSI chip design, and VLSI design tools. Mr. Hatch is currently working on the next generation of S/390 Division CMOS processors.

Dale E. Hoffman *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (daleh@vnet.ibm.com).* Mr. Hoffman received a B.S. in electrical engineering from Pennsylvania State University in 1981 and an M.S. in electrical engineering from Syracuse University in 1984; he is currently pursuing an M.S. in computer engineering at National Technological University. He is a Senior Engineer and a design manager responsible for test, chip integration, physical design, and back-end design methodology for high-frequency custom microprocessors. Mr. Hoffman joined IBM in 1981 at the East Fishkill facility, where he designed and managed advanced VLSI logic and memory test systems for eleven years. Mr. Hoffman holds five U.S. patents and has several publications.

Scott A. McCabe *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (samiam@vnet.ibm.com).* Mr. McCabe received a B.S. degree in electrical engineering from Rutgers University in 1984, joining IBM the same year. Mr. McCabe has worked in the area of physical verification, first at East Fishkill and since 1993 at Poughkeepsie.

Greg A. Northrop *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (gnorth@watson.ibm.com).* Dr. Northrop received the B.S. degree from Texas Tech University in 1977, and the M.S. and Ph.D. degrees in condensed-matter physics from the University of Illinois, Urbana, in 1979 and 1982, respectively. He joined the IBM Thomas J. Watson Research Center in 1984 as a member of the Semiconductor Physics and Devices Department, subsequently joining the VLSI Design Department in 1993. His current interests are in the areas of high-performance CMOS circuit design, including layout automation, standard-cell methodologies, and circuit tuning.

Rick Seigler *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (seigler@pk705vma.vnet.ibm.com).* Mr. Seigler joined IBM Poughkeepsie in 1980 after receiving B.S.E.E. and M.S.E.E. degrees from Rensselaer Polytechnic Institute. He worked as a logic designer and as a systems test engineer on 3090 systems, and as a Recovery/Serviceability Systems Test manager for 3090 follow-on systems and 902X systems. Prior to joining the Custom Design Department in 1992 where he now works as an Advisory Engineer, he also served one year on an IBM Faculty Loan assignment at the Georgia Institute of Technology in Atlanta.