Design Methodology for the High-Performance G4 S/390 Microprocessor

K. L. Shepard, S. Carey[1], D. K. Beece, R. Hatch[1], G. Northrop
IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY
[1]IBM S/390 Division, Poughkeepsie, NY

Abstract

This paper describes the methodology employed in the design of the G4 S/390 microprocessor. Issues of verifying design metrics of power, noise, timing, and functional correctness are discussed within the context of a performance-driven transistor-level custom design approach. Semi-automated techniques to encourage designer productivity consistent with the objectives of a high-frequency deep submicron design point are presented as are the practical issues associated with managing the complexity of an 8 million transistor design.

1 Introduction

The Generation-4 S/390 CMOS microprocessor is a 17.35-mm × 17.35-mm chip with 7.8 million transistors and has been successfully operated at frequencies up to 400 MHz[1]. Because of overall system performance requirements, a high-frequency design point was pursued with aggressive use of an advanced $0.2\mu mL_{eff}$ CMOS process. The methodolgy had to enable transistor-level optimization while dealing with the unique challenged of deep submicron design. Aggressive semi-automated techniques to encourage designer productivity and maintain schedule integrity were also an essential part of the design process. Issues of power, noise, timing, and interconnect modelling required the development of new methodologies and tools consistent with the stringent demands of reliability and testability required for parts for the high-end server market.

We first describe the manner in which the design data was organized, stressing abstraction and controlled hierarchical organization of the design as the means to manage complexity. In the following section, we describe the logic verification methodology which combines aggressive use of cycle simulation and Boolean comparison. In the next two sections, we describe circuit and chip-level design methodologies, emphasizing the synergy between semi-automated synthesis processes and pure custom design. We then devote several sections to the methodologies associated with interconnect modelling, timing, power, and noise analysis. In each case, we address the unique challenges of deep submicron design within the context of a transistor-level focus.

2 Design organization

Design abstraction is one of the key methodology techniques used to manage complexity. In the G4 microprocessor design, all analysis and verification are accomplished with a two-level hierarchical approach which involves identifying groups of 10,000 to 200,000 non-array transistors as *macros*. The design also consists of several large SRAM macros, a ROM macro, and several multi-port register file or scannable array macros. Macros are individually laid out and floorplanning 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 sub-macro 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.
- logical constraints view. This view contains Boolean satisfiability constraints in the implementation which are are tested through BDD techniques and used by timing and noise analysis.
- layout. The physical design of the macro which may be a hierarchy of other sub-macro symbols and schematics.
- faultmodel. 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.

- 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* interconnects 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.

Details of how the design abstractions are created and used will be discussed in detail in the remainder of the paper.

3 Logic Verification

Verifying the correct logical functionality of the microprocessor design, the fundamental goal of the methodology, involves a tight three-link *verification chain.* A register-transfer level description of the processor is verified with extensive simulation. This description is then compared against the circuit implementation with a formal Boolean comparison methodology. The layout is then verified against this circuit schematic with a layout-versus-schematic (LVS) engine. In this section, we examine the first two links in this chain.

The G4 microprocessor designed is captured completely in an RTL VHDL description[2]. The 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. In addition to the RTL model of the processor, there is a high-level microarchitectural model, also captured in VHDL, which is used for millicode (vertical microcode) verification and a C++ testcase generator, AVPGEN, which contains a "golden" behavioral description of the S/390 architecture[3].

In developing the RTL model of the processor, VHDL is entered for the macros and is done so as a structurally flat description with the exception of special latch and array primitives. The use of primitive components for sequential elements avoids errors in the design due to improper latch coding and facilitates mapping the design to cycle simulation and "cutting out" latches for Boolean comparisons. The entire concurrent VHDL language is allowed. In addition, process statements that explicitly represent combinational logic are permitted. The IEEE `std_logic_1164` package is employed and augmented with an expanded set of logical, relational, and arithmetic operators. Because the processor is initialized through scan, initial values for the scan process are passed to the latch primitives through generics. In addition, scan chain connections are coded in the VHDL in a manner which allows easy scan-chain reordering following physical design. 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.

Four distinct logic simulators were used in the verification of the RTL model: an event-driven VHDL simulator, two cycle simulators, and a hardware accelerator. VHDL simulation is used for "design simulation" on small VHDL models, macros or units. In the coding style used for the design, no explicit times are coded in the VHDL, except for the times used to establish the clock waveforms. All signal assignments, therefore, occur as a cascade of "delta" delay events following a clock edge, a *clock-edge-triggered logic specification.* For models of any significant size, including full chip models, cycle simulation is used. Cycle simulation makes explicit use of clock-edged-triggered design by identifying certain signals as "registers" which change state based on their input values at fixed cycle evaluation times. We build and simulate models with two types of sequential granularity in cycle simulation – a "two-cycle" model which uses two cycle-simulation cycles per machine cycle, essentially allowing independent modelling of master and slave and allowing simulation of all chip functionality, and a "one-cycle" model which uses one cycle-simulation cycle per machine cycle, which enables efficient simulation of normal system operation of the processor. For the very largest system models, a hardware accelerator is used[4]. The test case environment allows the designers the flexibility of moving between these simulator with *test case transparency,* allowing the use of the simulator which is best for the specific model and test case.

The second link in the verification chain is an equivalence checking methodology that ensures that the circuit implemented in silicon matches the function of the VHDL description. Because the design is represented as a single netlist above the macro level, the design is correct by construction above this macro level of hierarchy. Therefore, a necessary and sufficient condition for correspondence is that the macro circuits compare again the macro VHDL. This is accomplished with a formal Boolean comparison of the circuit and VHDL, augmented with switch-level verification of latch and array primitives. IBM's Verity tool[5], which relies on canonical reduced ordered binary decision diagram representations[6], is used to perform the Boolean comparison.

An important new feature of the G4 design methodology is the use of Boolean satisfiability constraints, logical conditions which can be expressed as a function which must be satisfied. They are used for four

main purposes – to express a "don't care" state safely for a VHDL macro architecture, to allow the use of circuits that require certain logical conditions on their inputs for correct operation, to eliminate false paths in static timing, or to reduce pessimism in noise analysis. Constraints associated with macro inputs and outputs are coded in the VHDL as `assert` statements. Constraints on inputs are known as *asserts*, condiditons which we assume to be true which are verified either formally (strong asserts) or through simulation (weak asserts). Constraints on outputs are known as *tests*, conditions that are verified to be true either formally (strong tests) or through simulation (weak tests). In addition, each macro, in general, has an associated logic constraints view which contains additional Boolean satisfiablity constraints for the macro circuit. Verity is used globally to verify that every strong assertion is accompanied by a satifying strong test for global signals. Verity also verifies the conditions contained within the logic constraints view.

4 Circuit and chip-level design methodologies

4.1 Custom macro methodology

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 semi-custom cell-based approach for control logic.

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-path cross sections. Estimates are also made for the capacitive loading at the outputs based on early chip floorplan estimates. 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 based on global timing. The final schematics and layout are hierarchical with no methodology limitation on the amount of hierarchy which may be used. The layouts have to conform with shadow views from the global environment, generated using either the blockage or contract methodology as described in Section 4.3.

The arrays in the G4 design are entirely custom designed. The use of self-resetting techniques precludes use of static timing analysis[7, 8]. 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.

4.2 Semicustom macro implementation

For the control logic of the design, stable logic definition do not occur until late in the design process. At the same time, logic restructuring and retiming offer the most benefit in optimizing performance. Because of these factors, a semiautomated schematic and layout generation system is required which preserves the benefits of transistor-level design.

We rely on BooleDozer, IBM's logic synthesis tool, to generate the schematics for our semicustom macros. The details of IBM's BooleDozer tool are described elsewhere[9, 10]. We exploit BooleDozer in innovative ways to achieve rapid implementation while maintaining the ability to control the logic structure and aggressively tune the design at the device level. These techniques include:

- The 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[9].
- Use of "don't cares" as defined by VHDL asserts to simplify logic implementation[11, 12].
- Use of "hill-climbing" based late timing correction; that is, individual transformation operating under greedy heuristics are allowed to make timing worse if a succession of these transformations ultimately improves timing, allowing the heuristics to escape from locally optimal timing solutions
- 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 several important ways. First, the rules were continuously parameterizable; that is, no fixed library cells were assumed. Second, they were parameterized by quantities directly related to delay, rather than size, which we refer to as *normalized gain* and *beta*. Consider the static CMOS inverter shown in Figure 1(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:

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

The β is given by:

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

In addition, we define something 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 3-input NAND gate shown in Figure 1(b) driving the same load capacitance c_{out}. 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. 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).

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

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

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

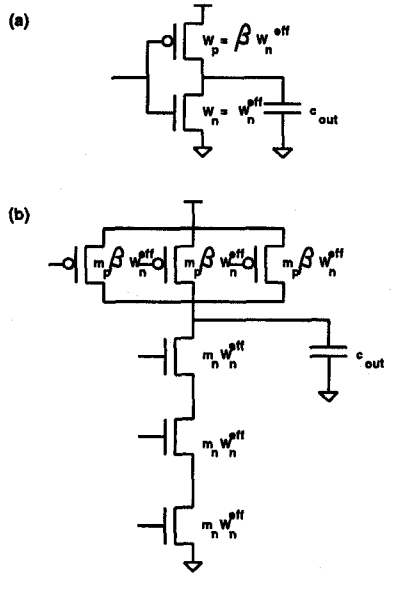

Figure 1: (a) Device widths for a static inverter in terms of W_n^{eff} and β. (b) Device widths for a 3-input static NAND gate.

Beta and *gain* parameterization in the timing rules enable heuristics for delay optimization which can be applied after an initial placement of the design. It is only after an initial placement that the interconnect capacitance can be estimated accurately enough through minimum width Steiner tree routes to enable detailed retuning. The changes that result from these postplacement optimization are handled as an "engineering change option" (ECO) to the original placement. In addition to retuning, postplacement optimizations done within BooleDozer include reordering of the scan chains, optimization of the clock distribution network within the macro, and short-path padding in the case of early-mode hold violations.

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 concentrates on efficient design of simple cells (the most complex being a 2x2 AO/OA), and allows customization of the cell image. 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 non-parameterized 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.

4.3 Chip integration

Chip integration, the top level of the two-level circuit and physical design process, consists of floorplanning and global wiring design. Guided by early global timing, the first step in the chip-level design process is to floorplan the macros, allocating macro area and optimizing pin placement. 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 areal power distribution pads as opposed to wire-bonded peripheral pads. The clock tree design is a balanced H-tree structure created with a specialized maze router that uses wire width as well as length tuning to achieve skew control of $\pm 25ps$ while simultaneously working to reduce latency. After power and clock routing, the I/O's are wired as are other timing critical buses in the design. Critical bus routing is done 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.

Once the initial floorplan with power, clock, and

prewires is complete, the rest of the interconnect design is managed through the use of a hierarchical physical design process to parallelize the design effort. 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 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 levesl of hierarchy and this contract is coded in both the shadow and abstract, which are negative images of each other. Prewires – clock, power, chip I/O, and critical nets – also appear in the shadow as blockages. This techniques does not create as efficient a use of wiring channels as the blockage method but allows the parallelization of the routing process.

Managing global RC delays and assuring that global net drivers are appropriately sized for their loads are essential parts of the chip integration process. We use the timing tools to identify global net receiver slews which exceed a target slew maximum. For those receivers showing violations, the slew at the driver is also examined. The general optimization methodology is a three-step process:

1. In the cases in which 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, then 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.

5 Electrical and physical verification

In this section, we describe the methodologies we employ for interconnect, timing, power, and noise analysis. Each of these methodologies is driven by the need to address deep submicron design within the context of a transistor-level focus.

5.1 Interconnect analysis

As with all other aspects of the methodology, extraction and interconnect modelling are divided between the macro and global with special considerations for the interaction between these levels. The resistance and capacitance extraction is rule-based, calibrated by finite element calculations. At the macro level, we perform two types of extraction, a capacitance-only extraction, including coupling capacitors, and a resistance and capacitance extraction in which all floating capacitors are broken and tied to ground. For the global level, three extractions are performed. A statistical extraction allows quick interconnect estimation for timing assuming a percentage loading on each wire. Steiner tree estimates are used for estimating unrouted designs. In addition, there is a detailed RC extraction with grounded coupling capacitance for timing and a complete coupling-capacitance extraction for noise analysis.

In general, RC extractions produce tremendous amounts of resistance and capacitance data. Reduction techniques are essential to successful analysis of this data for timing and noise analysis. At the global level, we employ what we refer to as a *pi model, pole-residue* macromodel as shown in Figure 2. We model the interconnect load on each driver as a pi-model, matching to order s^3 the moments of the driving-point admittance [13] as obtained by asymptotic waveform evaluation (AWE) techniques[14]. In addition, the model includes for each receiver the poles and residues of the transfer function of the unit step response for the driver to the receiver, also obtained through AWE techniques.

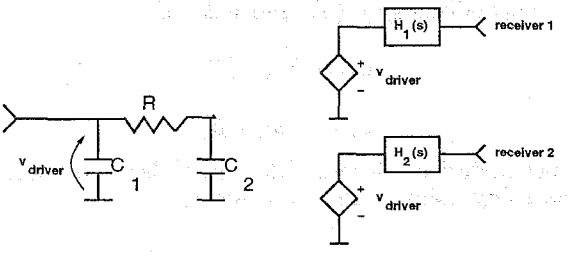

Figure 2: *pi model, pole-residue* interconnect macromodel.

For reducing the interconnect models at the macro level, the requirement exists to preserve the RC netlist representation of the data for circuit simulation and timing analysis. To do this, we preserve the original topology of the RC extracted netlist, treating each branch as a two-port network. A frequency-domain criterion is used to choose between two- and three-capacitor reductions by matching the moments of the two-port admittance. This initial reduction leaves many single resistor point-to-point nets which are selectively eliminated based on a time-domain accuracy requirement.

5.2 Timing analysis

The timing analysis of the G4 design is accomplished using static timing analysis techniques, implemented using 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 combined with global interconnect 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 have to be re-abstracted. In addition, quick analysis of proposed global wiring changes can be made without detailed analysis at the macro level.

In performing the macro-level timing analysis, individual channel-connected components are identified. Delays through these components are generally simulated under assumption that only one input switches at a time. In lieu of complex metastability analysis, heuristics are applied to determine setup and hold times at latches. Both late and early mode timing analysis is performed. All circuit in late mode are timed to nominal process, highest predicted on-chip temperature, and lowest possible on-chip voltage. Early-mode analysis is performed at a three-sigma fast process, lowest predicted on-chip temperature, and highest possible on-chip voltage. Timing abstractions presented to global timing from macro analysis can be either black or gray. In the case of the black box, no internal latch points are defined and setup and hold tests are presented at primary inputs. These require independent verification of latch-to-latch paths within the macro, which are not presented to global timing. Gray boxes are used for timing verification in the case of transparatent 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.

In global timing, the macro abstracts are combined with the interconnect macromodels. To calculate the driver waveforms, we employ a variant of the *effective capacitance* approach. The actual admittance of the interconnect is modelled at the driver as a pi model. The effective capacitance is given by the capacitance that produces the same total integrated current as the pi model through the driver through the 50 percent point of the driver voltage waveform. From this analysis, one finds:

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

where t_r is the slew time. This equation is solved iteratively with the driver slew characterization.

$$t_r = t_r^o + kC_{load}$$

The global timing runs are used to generate assertions to drive macro-level implementations. These assertions include effective capacitances on the outputs to indicate global wire loading, primary input resistance assertions to indicate driver strength and global wire resistive shielding, input arrival times, early and late mode, rising and falling, and output required arrival times, rising and falling. 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. Proper assertion management is key to timing convergence in a hierarchical timing environment.

5.3 Power analysis

Analyzing the power demands of the chip constitutes an important part of the methodology. Associated with this is the equally important analysis of the integrity of power supply and the electromigration reliability of the power network. 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. Several aspects of the problem stubbornly defy straightforward static analysis such as 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[15]. 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 analysis 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[16]. In SPECS, the simulation is event-driven. $i - v$ characteristics are assumed to be piecewise constant, branch currents are assumed to be piecewise constant in time, and branch voltage 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 ($<$ 50,000 transistors) hundred of patterns are analyzed while for larger macros (50,000 - 200,000 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, as shown in Figure 3, are attached to the power points in the extracted netlist 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 will result in power supply noise, which in turn affect 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

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

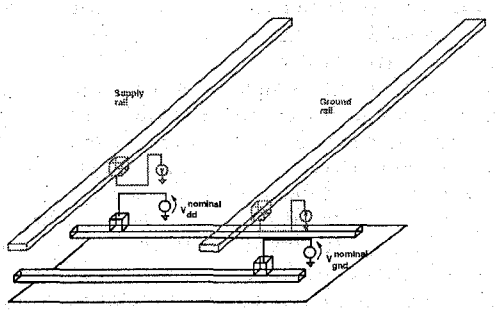

Figure 3: Power point methodology. The supply and ground distribution is divided between macro (dark) and global (light). Power points form the connection between these two levels of hierarchy. Independent voltage sources that supply nominal supply and ground voltages to the macro serve as current meters. Peak and average currents measured at these meters are subsequently applied to the global power grid to determine IR drops and delta-I noise.

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 up 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 4 shows *a power map* of the chip for average switching activity, calcalated by computing:

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

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

The power abstracts are also used to determine the power supply noise and evaluate electromigration contraints 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. 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-references with the wire widths and via sizes to flag potential electromigration problems.

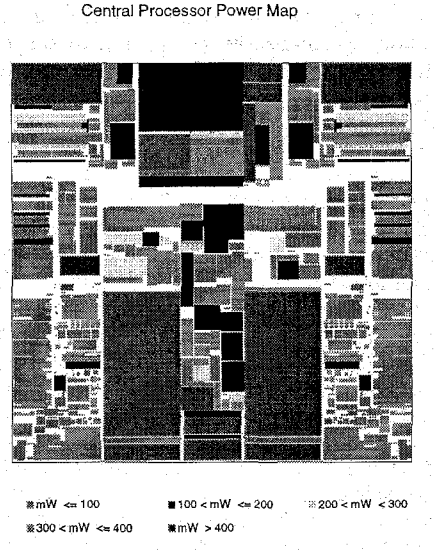

Figure 4: Power dissipation map of the G4 processor.

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. 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 5 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 $100psec$ 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 nwell and nonswitching circuit capacitance.

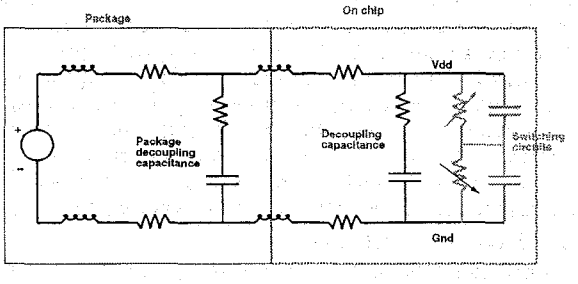

Figure 5: Equivalent circuit for delta-I calculations.

5.4 Noise analysis

Noise is an important new metric for design, of comparable importance to timing, area, and power. In the G4 processor design, we considered only global coupling noise, while actively working on a more comprehensive noise analysis strategy[17]. The coupling noise analysis uses the capacitance-only coupling extraction of the global interconnect. We define the *victim* net as the static net onto which pulse noise is being coupled by one or more *perpetrator* nets. Coupling noise is calculated using the simple linear model shown in Figure 6. 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 modelled from the linear region of the I_{ds} versus V_{ds} current-voltage characteristic at $|V_{gs}| = V_{dd}$. R_{driver} is then given by:

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

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 modelled as saturate ramp waveforms of slew t^i_{slew}. This network ignores the distributed effects of resistance on the victim net.

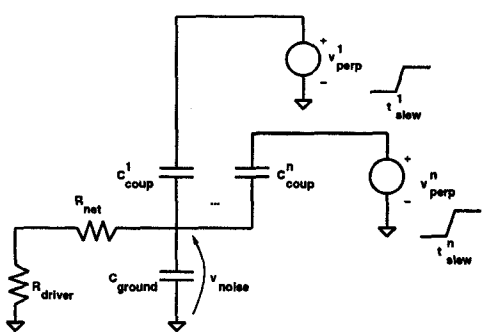

Figure 6: Simple circuit model for crosstalk coupling.

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 $v_{noise}(t)$ response produced by the action of a single perpetrator source $v^i_{perp}(t)$ is given by:

$$v_{peak} = \begin{cases} \frac{RV_{dd}}{t^i_{slew}}\left(1 - e^{-t/R(C_{coup}+C_g)}\right) & \text{for } 0 \le t \le t^i_{slew} \\ \frac{RV_{dd}}{t^i_{slew}}\left(1 - e^{-t^i_{slew}/R(C_{coup}+C_g)}\right) \times e^{-(t-t_{slew})/R(C_{coup}+C_g)} & \text{for } t \ge t^i_{slew} \end{cases}$$

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

$$v_{noise} = \sum_{i=1}^{n} \frac{RC_{coup}V_{dd}}{t^i_{slew}}\left(1 - e^{-t_{slew}/R(C_{coup}+C_g)}\right)$$

In addition, we use timing windows 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[18]

6 Summary

We have reviewed some of the methodology considerations that went into the design of the G4 S/390 microprocessor. Several important themes thread through our approach which directly led to the success of the design. Cycle simulation is an essential element of logic verification given the size and complexity of designs and the need for high-performance simulation. The methodology must also be fundamentally transistor-level to allow detailed circuit tradeoffs between timing, power, and noise. Interconnect must be designed and analyzed with comparable importance to devices. Static techniques must be employed wherever possible, for example, in timing and noise analysis and in Boolean equivalence checking. Managing complexity inevitably involves the use of hierarchy and abstraction which must be handled strictly and consistently.

7 Acknowledgments

The authors gratefully acknowledge all the members of the G4 design team and other individuals through IBM for their contributions to the methodology. We would like to particularly 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, Kwok Eng, Larry Lange, Rick Seigler, Dale Hoffman, and Paul Villarrubia.

References

[1] C. F. Webb, *et al*, "A 350 MHz S/390 Microprocessor", in *Proceedings of the International Solid-State Circuits Conference*, 1997.

[2] *IEEE Standard VHDL Language Reference Manual, Std 1076-1987*, IEEE, NY, 1988.

[3] A. Chandra, *et al*, "AVPGEN: A test generator for architecture verification," *IEEE Transactions on VLSI Systems*, pp. 188-200, June, 1995.

[4] D. K. Beece, G. Deibert, G. Papp, and F. Villante, "The IBM Engineering Verification Engine," in *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 218-224, June, 1988.

[5] A. Kuehlmann, A. Srinivasan, and D. P. Lapotin, "Verity – a formal verification program for custom CMOS circuits," *IBM Journal of Research and Development*, pp. 149-165, January/March 1995.

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computers*, pp. 677-691, August 1986.

[7] 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 pipelines architecture," *IEEE Journal of Solid-State Circuits*, pp. 1577-1585, Nov, 1991.

[8] A. Pelella, et al., "A 2 ns access, 500 MHz 288kb SRAM macro," *Dig. Tech. Papers, Symp. VLSI Circuits*, pp. 128-129, 1996.

[9] D. Brand, R. F. Damiano, and A. D. Drumm, "In the Driver's Seat of BooleDozer," *Proceedings of the International Conference on Computer Design*, pp. 518-521, 1994.

[10] 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 Journal of Research and Development*, pp. 131-148, January/March 1995.

[11] R. A. Bergamaschi, D. Brand, L. Stok, M. Berkelaar, and S. Prakash, "Efficient use of large don't cares in high-level and logic synthesis," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 272-278, 1995.

[12] D. Brand, R. A. Bergamaschi, and L. Stok, "Be careful with don't cares," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 83-86, 1995.

[13] P. O'Brien, T. Savarino, "Efficient On-Chip Delay Estimation for Leaky Models of Multiple-Source Nets.", in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 9.6.1-9.6.4, 1990.

[14] L. Pillage, R. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis", *IEEE Transactions on Computer Aided Design*, pp. 352-366, April 1990.

[15] D. Brand and C. Visweswariah, "Inaccuracies in power estimation during logic synthesis," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 388-394, 1996.

[16] C. Visweswariah nad R. A. Rohrer, "Piecewise approximate circuit simulation," *IEEE Transactions on Computer-Aided Design*, pp. 861-870, July 1991.

[17] K. Shepard and V. Narayanan, "Noise in deep submicron digital design," *Proceedings of the International Conference on Computer-Aided Design*, pp. 524-531, 1996.

[18] E. Malavasi, E. Charbon, E. Feit, and A. Sangiovanno-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Transactions on Computer-Aided Design*, pp. 923-942, 1996.