A Hierarchical Transistor and Gate Level Statistical Timing Flow for Microprocessor Designs

Adil Bhanji, Chandu Visweswariah[*]*, Debjit Sinha*, *Gary Ditlow*[*]*, Kerim Kalafala, Natesan Venkateswaran, and Sachin Gupta*
IBM Systems and Technology Group, Hopewell Junction, NY 12533
[*]IBM T J Watson Research Center, Yorktown Heights, NY 10598

This paper presents a hierarchical statistical timing flow for microprocessor designs with minimal run-time and memory overhead compared to a deterministic timing flow. Application of the flow to robust custom circuit design under variability is highlighted.

I. Context

High performance microprocessors contain custom designed circuit macros to achieve aggressive frequency targets. These custom designed circuits are typically timed using circuit simulation engines. Microprocessor designs can contain upwards of one billion transistors. Circuit simulation, while highly accurate, is run-time intensive and is not practical to use in a timing flow where chip level timing runs are made daily during the design cycle of the chip.

This has led to the development of hierarchical *contract-based* timing where custom parts of the design are timed using transistor level timing tools [1] with circuit simulation type accuracy; followed by the generation of timing abstract models that reflect in a simpler and more compact form, the timing characteristic of the custom logic.

The timing characteristics are captured by the use of slew and load dependent tables. Timing abstraction employs techniques to reduce the size of the timing graph by performing pruning as well as arc compression. These techniques can reduce the number of timing arcs to be analyzed at the next level of hierarchy (unit or chip level) significantly. Model reductions of 400% are common. At the chip level, custom logic macros are now represented by these abstracts and can be timed quickly without the use of circuit simulation.

II. Motivation

Variability plays an increasingly important role within chip timing today, especially for high speed circuits like those used in microprocessor designs. Timing analysis considering variability (e.g., statistical timing) is now required as part of sign-off timing methodologies. In the case of designs containing standard cell libraries, modeling of statistical effects can be described within the characterized delay models. Statistical timing is traditionally based on finite differencing of timing quantities (like delays, slews, or timing waveforms) computed at multiple process corners. While this approach is usable in timing flows where delay calculations can be performed extremely rapidly, for example, using a table lookup delay calculator like .lib, such an approach is not feasible in custom transistor level timing due to the extreme simulation run-times. Additionally, most transistor level timing tools are not variation aware; and are only able to perform deterministic timing today. Guard bands in the form of large timing margins are instead applied to ensure chip functionality at the desired clock frequency under variability. However, this coarse way of modeling variability leads to pessimistic timing analysis for most timing paths and increasingly makes the chip design closure process more difficult, while leaving performance on the table. Another important consideration for variability analysis at the macro level is that the information be passed correctly up to the chip level for accurate variability aware hierarchical timing analysis. This work describes an approach for hierarchical statistical timing analysis with the following goals (individually described in Sections III-A, -B, and -C, respectively).

1. *Minimal run-time and memory overhead compared to deterministic timing analysis at the macro level*;
2. *Aid robust circuit design or optimization under variability;*
3. *Preserve timing variability information in macros for chip level timing with minimal growth in timing abstract model sizes.*

III. Contribution

A. Efficient statistical timing analysis at macro level

Given a deterministic (or process corner based) transistor level timing analysis tool, and the prohibitive run-time overhead of computing sensitivities of timing quantities to sources of variation using finite differencing, our approach employs asserted sensitivities for modeling timing variability.

In this approach, the first-order sensitivity of any timing quantity to a source of variation is specified as a simple function (e.g., fraction) of the nominal delay. An asserted sensitivity contains a correlated part, and an uncorrelated or independent part; the former representing sensitivity to global die-to-die variability that alters performance of all applicable timing quantities in the design in a systematic way, and the latter, representing sensitivity to local within-die variability of each applicable timing quantity in an independent way. The numeric values of these sensitivities are technology specific, and are decided and refined by design leads based on hardware data from the foundry. Asserted sensitivities may be used for both front-end (e.g., L_{eff}, *VT_TYPE*, *CHNL_WIDTH* and *GATE*) and back-end (e.g., *WIRE*) sources of variation.

An example is next presented to illustrate statistical modeling and analysis using asserted sensitivities with the following assumptions:

- Propagation delay through high VT (voltage threshold) devices has per sigma correlated and independent sensitivities of *2%* and *1%* of the mean delay, respectively, to *VT_TYPE*;
- Propagation delay through regular VT (voltage threshold) devices has per sigma correlated and independent sensitivities of *1%* and *0.5%* of the mean delay, respectively, to *VT_TYPE*;
- Propagation delay through interconnects in metal layer 2 has per sigma correlated and independent sensitivities of *2%* and *0.5%* of the mean delay, respectively, to *WIRE*; and
- Deterministic delays are worst-case library (3σ corner) based.

A path through a high VT device (*segment S1)*, followed by a wire routed in metal layer 2 (*segment S2)*, and finally through a regular VT device (*segment S3)* is considered, with deterministic delays across the three segments being *10* units each. The statistical delay for *S1* is expressed as: $m_{S1} + 0.02 m_{S1} \Delta VT\_TYPE + 0.01 m_{S1} \Delta R_{S1}$ [2] and obtained as follows (delays for *S2*, *S3* are computed similarly).

$$\text{Det. delay}_{S1} = 10 = m_{S1}[1 + 0.02\times 3 + 0.01\times 3] \;\Rightarrow\; m_{S1} = 9.17 \quad (1)$$

$$\text{Stat delay}_{S1} = 9.17 + 0.18\Delta VT\_TYPE + 0.09\Delta R_{S1} \quad (2)$$

The total statistical path delay through the three segments is then computed as $28.05 + 0.28\Delta VT\_TYPE + 0.18\Delta WIRE + 0.11\Delta R$, which has a worst case (3σ projected) delay of *29.1* units, while the deterministic delay model yields a path delay of *30* units. This example illustrates pessimism reduction by modeling variability statistically and highlights that asserted sensitivities allow significant flexibility in modeling variations. In fact, *"what-if"* experiments can

be performed even in the absence of precise variability information. Timing quantities may have different sensitivities to a source of variation depending on their characteristics (e.g., varying amount of sensitivity based on ranges of channel width of a device). In addition, the sources of variability may be correlated.

Since this approach performs statistical analysis based on the deterministic delays obtained from the circuit simulator, there are no additional calls to the simulator; thus eliminating a significant overhead in the turnaround time for such an analysis. Table 1 presents run-times and memory footprints of five macros (*M1-M5*) from a 45 nanometer industrial microprocessor design timed with and without statistical analysis. The second column denotes the number of Channel Connected Components (CCCs) in each macro. A CCC represents a combined set of transistors whose sources and drains are interconnected. Wall-times are used to compare the total run-times for the timing flow of these macros since it indicates what an end user notices when a timing job is submitted and takes into consideration I/O activity. As shown in the table, the run-time overhead averages less than *0.2%* with *5.6%* memory increase. The negligible run-time overhead and pessimism reduction motivate the deployment of statistical timing analysis in the macro timing flow.

B. Robust custom circuit design under variability

Microprocessor designs contain custom logic like dynamic circuits to meet aggressive frequency targets. From a timing perspective, these circuits are more complex since timing requirements or *tests* must be met for every logic stage unlike traditional static logic gates where the data and clock path delays need to meet given aggregated timing requirements. Fig.1 shows a typical dynamic circuit, and Fig.2 illustrates some timing tests for this circuit. The number of tests in a custom macro containing these kinds of circuits is thus very large. In addition to ensuring that all these tests meet their slack goals at a nominal or specified process corner, a designer needs to ensure that no test will fail under variability. A commonly used metric to denote the *robustness* of a test is the ratio of mean to the standard deviation of the slack distribution of that test. From Fig. 3, it is immediate that a robustness metric value greater than *3* indicates a *99.7%* yield or better for that test and is desired. A designer is also interested in the *joint* robustness metric for all tests in his/her macro.

During custom macro circuit design at early stages of the chip design flow, lack of accurate technology specific process variability information requires that designers verify the robustness metric of macros under different amounts of variability. SPICE is traditionally used for this analysis, but suffers from impractical run-times if used inside a Monte Carlo simulation. Moreover, Monte Carlo analysis using SPICE after every custom optimization step (like transistor sizing or replacing a regular VT device with a high VT device) is not feasible.

Our statistical macro timing analysis aids the designer in running numerous statistical analyses of the macro while changing the amount (or sensitivity) of the sources of variability in a reasonable time (see run-time overheads in Table 1). Based on the obtained results, the robustness metric of every test (or all tests) can be obtained as a function of variability. In Fig. 4, the robustness metric for a *Precharge* test (ensures output node of dynamic logic is completely pre-charged before the evaluate cycle) is shown as a function of *GATE* variability. It is observed that the metric crosses the value of *3* at *8.3% GATE* variability, and thus implies that the circuit will have less than *100%* timing yield for variability greater than this value; this is illustrated in Fig. 5 where timing yield plots for *2%*, *4.6%*, and *10%* variability are shown. These plots provide critical variability information for desired tests in the macro to a designer.

Our approach provides flexibility to model other (or a combination of) sources of variation as well and thus aids robust circuit design under variability.

C. Hierarchical statistical timing analysis using statistical abstraction with small model size growth

A timing abstract is typically generated in the last step of a transistor (macro) level timing flow (see Fig. 6). Since abstraction involves macro timing graph compression, specific details of timing arcs (e.g., an arc that denotes a wire, or one that denotes signal propagation through a CCC with high VT devices) are lost. It is thus obvious that asserted sensitivities would not work correctly on the abstract models due to lack of specific design data lost during abstraction.

Even in the simplest case, where there is a single source of variation that applies to all timing arcs, compression affects the sensitivities in a manner that cannot be trivially *re-constructed* during chip level timing. Fig. 7 presents data showing a histogram of errors in test-slack for a macro in such a scenario with one independent source of variability that is asserted the same sensitivity value in both the macro and chip level runs. The figure shows how slacks may be both under- and over-predicted in this simplest case without statistical abstraction. Our approach therefore employs statistical abstraction to capture the timing variability information in a macro.

Statistical abstraction stores sensitivity information for each timing quantity of every compressed timing arc in the abstract. Arcs that are topologically close to the primary inputs or outputs of the macro are characterized as a function of input slew or output load or both. For these arcs, the sensitivity information that is required to be stored becomes significant. Fig. 8a shows the results obtained from a delay characterization of a timing arc that is input slew dependent. A set of m input slews is used for characterization, wherein for each input slew $Slew_i$, the statistical delay of the timing arc is obtained in a parameterized form [2] with mean delay M_i, and sensitivity to each parameter X_j as S_{ij}. The size of the table to store this information is ($m + mn$), since a mean value is stored along with a set of n sensitivities (corresponding to n sources of variation) for each of the m input slews characterizations.

Our approach, as shown in Fig. 8b, facilitates model size versus accuracy trade-offs wherein the size of the table to store the above information is reduced to ($m + n$) instead of ($m + mn$). In this *reduced* delay characterization model, only the mean delay values M_i are stored as a function of the input slew $Slew_i$. The sensitivities are stored relative to the mean delays (as fractions of the mean delay) and not as absolute numbers. In addition, these relative sensitivities are not stored as a function of the varying input slews, but approximated to be constant using least square fitting.

Table 2 illustrates the relative model size of the sensitivity information in a statistical abstract to that of the deterministic timing information; the proposed trade-off approach limits the statistical model size overhead on the average to *12.9%*. When statistical timing is enabled in a hierarchical context, these sensitivities are applied to the deterministic delays already modeled in the timing abstract. Statistical abstraction qualification in our microprocessor timing environment indicate within *3%* accuracy of the statistical abstract models in comparison to the transistor timing models under technology specific slew and load limits.

IV. Summary

This work presents a hierarchical statistical timing flow for microprocessor designs with minimal run-time and memory overhead compared to a deterministic timing flow. Application of the flow to robust custom circuit design under variability is highlighted.

References

[1] Bard *et al.*, "Transistor-level tools for high-end processor custom circuit design at IBM," in *Proc. IEEE, Vol. 95, No. 3*, 2007, pp. 530–554.

[2] Visweswariah *et al.*, "First-order incremental block-based statistical timing analysis," in *DAC*, 2004, pp. 331–336.

FIGURES AND TABLES

Table1: Run-time and memory overheads for statistical timing with asserted sensitivities

Macro name	Num. CCCs	Wall time (hh:mm:ss)		Memory used (Gb)	
		Complete run	Statistical timing **OVERHEAD**	Complete run	Statistical timing **OVERHEAD**
M1	33.9K	00:46:00	00:00:07	2.50	0.25
M2	6K	02:29:00	00:00:10	3.60	0.10
M3	2K	03:58:00	00:00:10	3.20	0.06
M4	18.7K	05:35:00	00:00:14	3.96	0.28
M5	12.2K	03:52:00	00:00:27	6.62	0.40
	Average	**100%**	**< 0.2%**	**100%**	**5.6%**

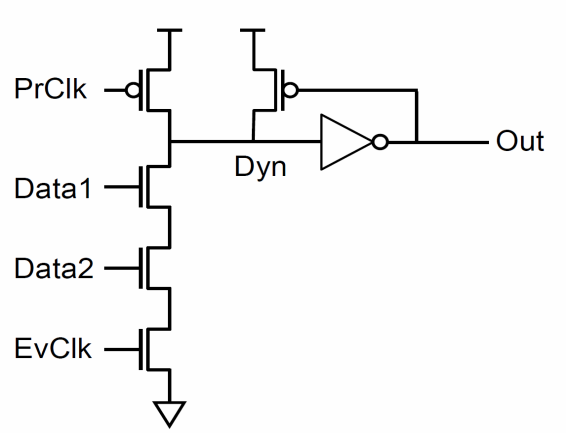

Fig. 1: A dynamic circuit

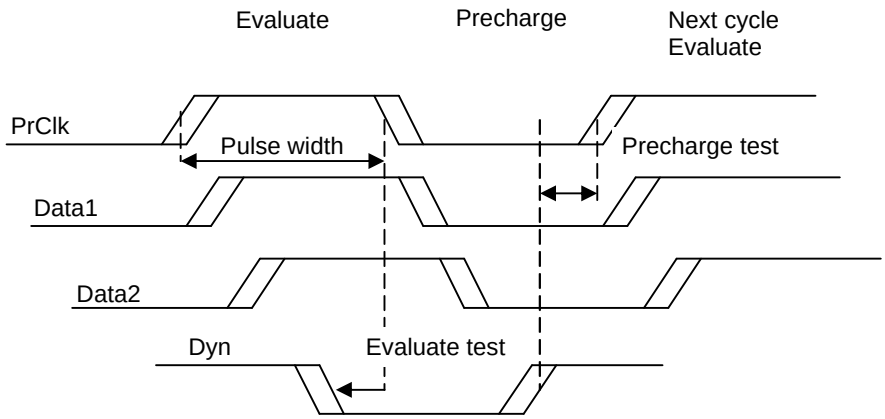

Fig. 2: Timing tests for a dynamic circuit

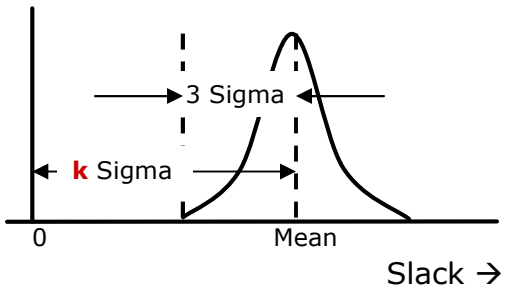

Fig. 3: Slack distribution showing relevance of robustness metric (k) and why it is desired to have k > 3

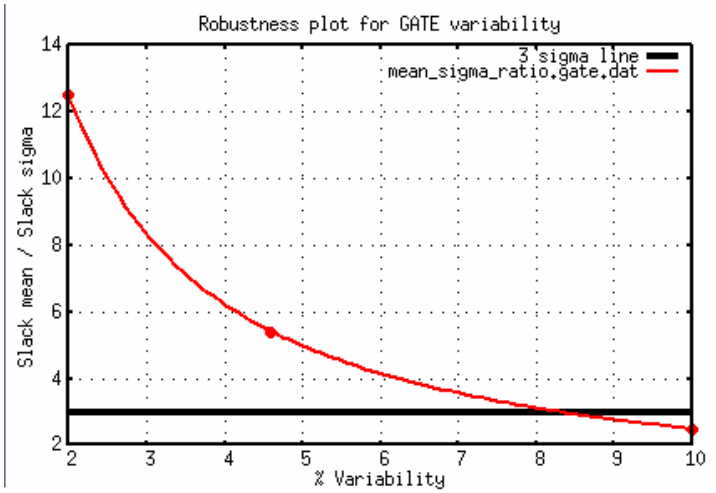

Fig. 4: Robustness plot for a Pre-charge test as a function of variability

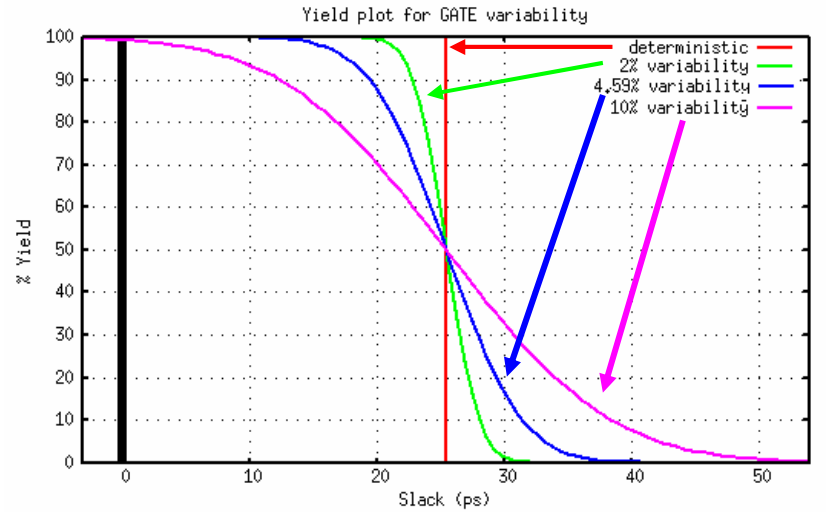

Fig. 5: Yield plot for a Pre-charge test as a function of variability

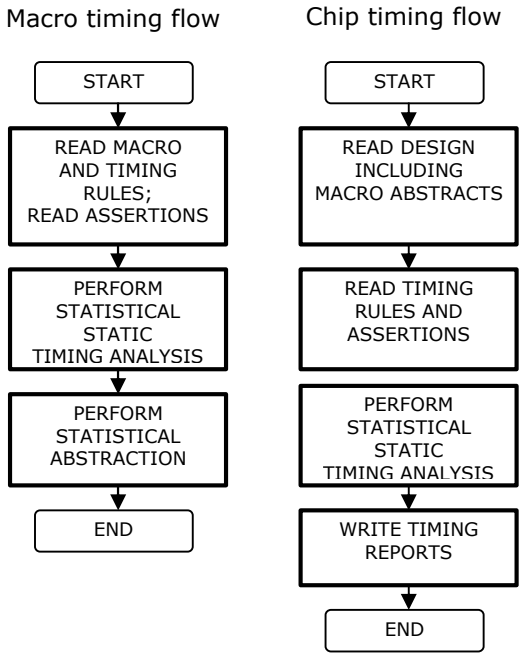

Fig. 6: Macro- and chip-level timing flows

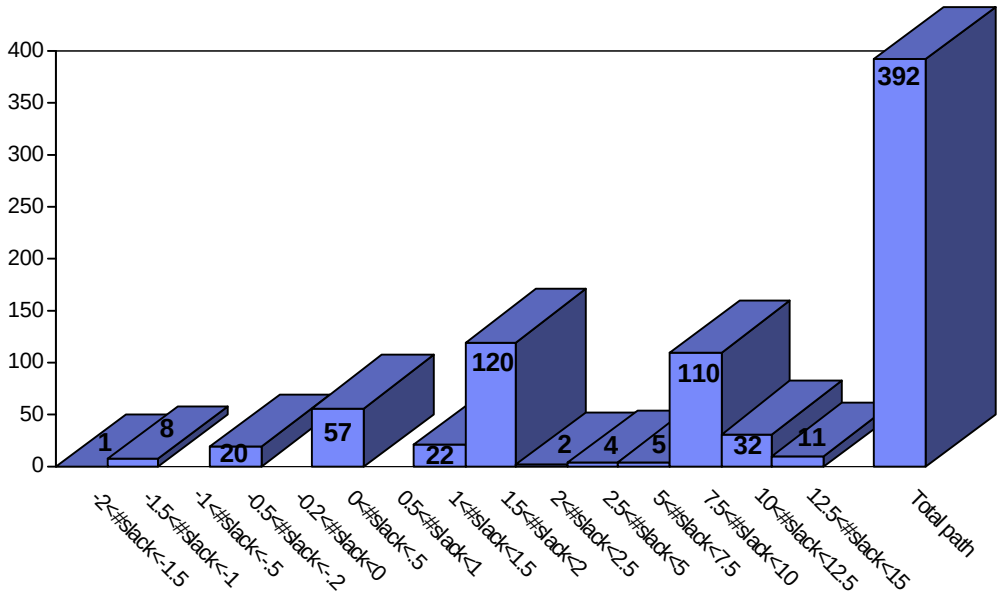

Fig.7: Histogram of number of tests inside a macro that have slack mismatch in macro and chip level timing due to lack of statistical abstraction

Original

	Mean delay	Sensitivity to parameter X_1	Sensitivity to parameter X_2	…	Sensitivity to parameter X_m
$Slew_1$	M_1	S_{11}	S_{12}		S_{1m}
$Slew_2$	M_2	S_{21}	S_{22}		S_{2m}
…					
$Slew_n$	M_n	S_{n1}	S_{n2}		S_{nm}

Reduced

	Mean delay	Sensitivity to parameter X_1 as % of mean	Sensitivity to parameter X_2 as % of mean	…	Sensitivity to parameter X_m as % of mean
$Slew_1$	M_1	S'_1	S'_2		S'_m
…					
$Slew_n$	M_n				

Fig.8: (a) Detailed statistical abstract table (b) Reduced statistical abstract table

Table 2: Run-time and memory overheads for statistical timing with asserted sensitivities

Macro name	Num. CCCs	Model size in terms of disk usage (Mb)	
		Deterministic abstract	Statistical abstract **OVERHEAD**
M1	33.9K	122.7	11.9
M2	6K	3644.7	773.3
M3	2K	2441.3	308.0
M4	18.7K	1866.9	273.0
M5	12.2K	3184.8	206.2
	Average	**100%**	**12.9%**