Algorithms for MIS Vector Generation and Pruning

Kenneth S. Stevens
Electrical and Computer Engineering
University of Utah
Salt Lake City, Utah
kstevens@ece.utah.edu

Florentin Dartu
Synopsys, Inc.
Advanced Technology Group
Hillsboro, OR 97124
fdartu@synopsys.com

ABSTRACT

Ignoring the effect of simultaneous switching for logic gates causes silicon failures for high performance microprocessor designs. The main reason to omit this effect is the run time penalty and potential over-conservatism. Run times are directly proportional to the vector sizes. Efficient algorithms are presented that prune the multiple input switching (MIS) vector set to a worst-case covering using a boolean logic abstraction of the gate. This non-physical representation reduces the vector size to approximately n vectors for an n-input gate. This is effectively the same vector set size as the optimal single input switching vector set. There are no errors for 88% the simulations using a Monty-Carlo coverage on a 90nm static library, and the magnitude of the errors are less than 5% on average.

1. INTRODUCTION

Accurately modeling silicon may seem feasible considering the progress made in computational power and timing analysis models and algorithms. Unfortunately, exponentially larger circuits are being built in silicon technologies that increasingly emphasize unwanted effects such as short channel, capacitive coupling, inductance, process variations, etc. As a result, all timing analysis tools still have to trade accuracy for run time.

Multiple input switching (MIS) is a very good example of an important effect that can substantially modify timing, but is commonly ignored in static timing tools. This delay variation can be measured in all gates with more than one input, as demonstrated in Figure 1. If `in2` switches long before `in1`, the delay from `in1` to the output remains constant. However, as the separation between `in1` and `in2` approaches zero, the delay from `in1` to the output can increase substantially. Figure 1 presents the maximum delay as a percentage of the single input switching (SIS) delay of `in1`. Observe that when the inputs switch concurrently (same 50% point) the delay pushout is larger than 25%. It is not difficult to find cases where the delay variation from SIS to MIS is as high as 70-80%.

Given such significant MIS delay variation, it is difficult to claim that static timing analysis always computes an upper delay bound. Indeed, MIS delay pushout has been confirmed as a source of failure in high performance microprocessor silicon. MIS has also been shown to significantly change the performance properties of certain circuit configurations[5].

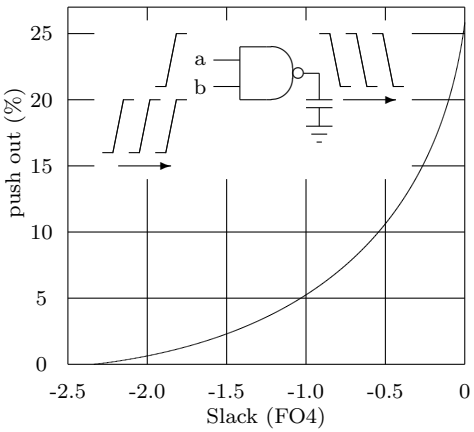

Figure 1: MIS Maximum Delay Pushout Effect

The series/parallel relationship between transistors is the primary cause of delay variation due to simultaneous switching. Second order effects include transistor sizing and ordering, charge sharing, voltages on non-switching transistors, legging, internal node voltages, and parasitic capacitance due to layout configurations. The current state of the art is to prune multiple input switching vectors based on the physical layout and topology of the gate [1, 2, 3, 4, 6]. This paper presents a significantly different approach in that no explicit topological information is used to create vectors.

The topology of transistors in a gate is inexorably related to the logic function of the gate: AND'ed literals require series transistors and OR'ed literals must be in parallel devices. Hence, given a boolean equation for a gate, one can easily extract the most significant topological relationship of transistors that cause MIS delay variation. This paper takes the novel approach of applying pure logic to create vectors based on the implied topological relationships.

This purely logical approach, if successful, has a number of advantages. The algorithms are very efficient and have been applied to the MIS characterization of a large microprocessor design using transistor level static timing analysis (STA). Since vectors are generated directly from a logic definition, the layout and schematics are not needed in cell-based design (CBD) methodologies. The vector generation algorithms are very flexible and correctly generate vectors for static gates, domino logic, sense amps, and other classes of gates. Finally, if more accuracy is required, this approach can rapidly create initial vector sets for further pruning based on more com-

ICCAD'06, November 5-9, 2006, San Jose, CA

plicated topological searches taking into account transistor sizes, parasitics, internal node voltages, etc.

Certain topological relationships are not exposed in the logical domain, such as transistor sizing and ordering, legging, etc. This can result in coverage errors in the vector set. Therefore a commercial 90nm static library was evaluated to determine the quality of the vector set produced by these algorithms.

2. VECTOR SETS AND RUN TIME

Static timing analysis trades off some accuracy through applying worst-case switching conditions for a considerable improvement in run time. The run time for library characterization is sensitive to two parameters: the number of vectors and the number of simulations required for every vector. Exhaustive simulation to characterize any n-input gate requires $2^{2n}k^p$ simulations (k^p simulations per vector for each of the 2^{2n} vectors), where p is the number of simulation parameters for k simulation points per parameter. Exhaustive SIS simulation requires $n2^nk^p$ simulations. This balloons to $(2^n - n - 1)2^nk^p$ simulations for MIS, where p will likely be larger than for SIS characterization.

To reduce the run time and CBD table size, model order reduction is applied to library characterization reducing p to two parameters: input slope and the output load modeled as effective capacitance C_{eff}. The number of simulation points k for each parameter is chosen such that a table with a simple interpolation between these points gives sufficient accuracy across the electrically valid slope and C_{eff} range. Five simulation points are usually sufficient (best, worst, nominal and the two points in between), resulting in $k^p = 5^2 = 25$ simulations per vector. Under MIS characterization, each input can have an independent slope and delay offset from the latest arriving signal. If both of these additional parameters are modeled for three switching signals, then $p = 2n+1 = 7$, requiring 78,125 simulations per vector and a substantially larger interpolation table. New model order reduction techniques can be developed to reduce the number of MIS parameters and required simulations. However, run time remains highly sensitive to vector sizes.

Transistor level STA does not need any precharacterization. The actual slopes, loads, and signal separations are used for simulation. However, all applicable worst-case vectors must be simulated on every gate in the design.

The minimum number of SIS vectors required to simulate an n-input gate is $2n$ – with each input pin rising and falling. Our algorithm computes, for a static cell library, an average of approximately $2n$ MIS vectors per gate.

DEFINITION 1. *A* **simulation vector** *v consists of a pair of boolean n-cubes where each literal value is a member of the set $\{0, 1\}$. This pair is represented as a single* **transition vector** *where each literal in the n-cube is a member of the set $\{\texttt{0}, \texttt{1}, \texttt{r}, \texttt{f}\}$. If the literal is unchanged in both vectors it retains its value of* `0` *or* `1`*. If the value in the first cube is* `0` *and the second is* `1`*, then the literal value in the transition cube is* `r` *(rising), otherwise it is* `f` *(a falling literal).*

The exhaustive 2^{2n} vectors can be broken down into three classes: $n2^n$ SIS vectors, $(2^n - n - 1)2^n$ MIS vectors, and 2^n stable vectors. The $n2^n$ SIS vector set for a two input gate is $\{\texttt{0r}, \texttt{1r}, \texttt{0f}, \texttt{1f}, \texttt{r0}, \texttt{r1}, \texttt{f0}, \texttt{f1}\}$, while the $(2^n - n - 1)2^n$ MIS vector set is $\{\texttt{rr}, \texttt{fr}, \texttt{rf}, \texttt{ff}\}$. The 2^n stable vectors are $\{\texttt{00}, \texttt{10}, \texttt{01}, \texttt{11}\}$.

2.1 First Order Affects

There are two aspects of vector pruning: logical (functional) constraints and topological constraints. The complete vector set 2^{2n} is substantially pruned by the simple functional requirement that each input vector must toggle an output. The 2^n stable vectors are not useful for any gate and are deleted. The remaining $2^{2n} - 2^n$ vectors are evaluated based on the logic function of the gate, and vectors that don't switch an output are also removed. Topological constraints cannot be used to reduce the valid single input switching vectors remaining from the $n2^n$ candidates. These vectors require physical information, such as transistor ordering, legging, device sizing, parasitic charge sharing, etc. However, when multiple signals switch, topological relationships largely determine if the gate will speed up, slow down, or be relatively unaffected by MIS in comparison to a single input switching on one of the input pins.

Delay variation in a gate due to simultaneous switching is primarily affected by the topological serial and parallel nature of a gate. Take the NAND gate of Figure 1. If the two inputs fall concurrently the circuit will speed up considerably. This occurs because the two PMOS transistors are in parallel and effectively double the drive of the gate. If the inputs rise concurrently then the output will be delayed. This is due the increased miller and body effect in the series pulldown stack. This allows us to make a first-order pruning of the the MIS vector set based on logical information.

The algorithms in this paper apply to any single diffusion connected network (DCN) such as a static and ratioed CMOS gates. Cells with pass gates as inputs are not admitted. Cells with multiple DCNs are not allowed except for a few special cases: when a second DCN is part of a feedback cycle, or it is an inverter that is on an output. While these algorithms could be applied to some cells with multiple DCNs (some XOR implementations), the disparity between topology and logic could introduce significant error. These two restrictions do not seem to represent a major limitation for many standard gate libraries.

The algorithms presented here support full MIS vector generation from 2 to n inputs switching for an n-input gate. Additional flexibility is provided because the algorithms accept an upper bound on the number of inputs that are allowed to switch at the same time. This adds flexibility in the run time/accuracy trade off in algorithms that apply these vectors to static timing analysis. As expected, the more inputs that are allowed to switch simultaneously, the larger the number of potential MIS vectors but the smaller the probability of such a case actually occurring in the circuit. For clarity, we have set this parameter to limit multiple input switching to 2 signals in the reported results.

Vector generation proceeds through three steps: 1. Create reduced superset of vectors. 2. Prune based on gate functionality. 3. Prune to the worst-case covering.

3. LOGICAL GATE REPRESENTATION

A gate representation is created using boolean logic with the structure $\mathcal{D} = \{I, O, B, L, o, s, r\}$ where I is set of input literals, O the output literals, B the feedback nodes, and L the inputs to the DCN where $L \subseteq I \cup B$. Boolean functions s and r operate over the literals in L and produce output $o(v) \in \{0, 1, *\}$ $\forall$ outputs $o \in O$. The algorithms create a vector set $v \in V$ mapping each literal in L to the set

$v_i \in \{0, 1, \mathtt{r}, \mathtt{f}, \mathtt{-}\}$, where '-' is a don't care value and '`r`' and '`f`' are rising and falling transitions respectively.

This model assumes a CMOS process. All passive devices are abstracted into either conducting terminals or open circuits with nodes being the conductive connections between terminals of the transistors. The model uses gates that can be extracted from a design as single DCNs where all source and drain terminals are all connected through the source and drains to power and ground.

A feedback node is a DCN input that can be traced back to an output of this DCN through at most one other DCN. Feedbacks are of two types: *direct feedbacks*, such as are found in sense amps, or *keeper feedbacks* as are found in domino gates. These signals have special behavior in our algorithms because they are both inputs and outputs to the DCN and hence cannot be freely given state assignments like unconstrained inputs due to their output dependency.

DEFINITION 2. *The* **set function** s *is* $\forall o \in \{1, *\}, \{x : s(x) = 1\}$ *asserts when the output is high or unknown (*$*$*). The* **reset function** r *is* $\forall o \in \{0, *\}, \{x : r(x) = 1\}$ *asserts when the output is low or unknown. The* **conflict set** z_c *is* $\{s \cap r\}$ *asserts when both* s *and* r *are asserted and the output is unknown. The ternary function for* o *is fully covered by* $s \cup r$.

The functions s and r are boolean representations of the transistor network between an output and the power rail, and the output and ground respectively. The definition for s and r supports ratioed gates and keeper feedbacks by abstracting the pullup and pulldown paths as demonstrated in Equation 1. A week pullup transistor will be ignored if it is turned on at the same time as a strong pulldown.

$$f = \begin{cases} 0 & \text{pullup} \ll \text{pulldown} \\ * & \text{pullup} \approx \text{pulldown} \\ 1 & \text{pullup} \gg \text{pulldown} \end{cases} \quad (1)$$

The precondition of a transition vector $\bullet v$ is the starting boolean cube generated by mapping `r` to 0 and `f` to 1. The postcondition $v\bullet$ creates the destination cube swapping 1 and 0 in the precondition.

There are many ways to structurally connect the gates, and there are many boolean representations. However, in this work we only use the canonical sum of products (SOP) and product of sums (POS) or conjunctive normal form. Thus we get a single logical representation that covers different topologies.

Both cell-based and transistor level flows are supported. For CBD, the functions s and r must be defined, and the signals sets I, O, B, and L are either provided or created. Our code automatically creates $\mathcal{D}$ from layout or schematics for the transistor level flow. The description of the transistor level code is omitted for brevity.

4. CREATING POTENTIAL VECTORS

This section describes the generation of a set of potential vectors using logical and topological information provided by the logical behavior of the gate.

Some gate topology is encoded in the canonical SOP format. Each minterm encodes the series relations between transistors. Parallelism exists between literals in different minterms. The algorithms use these topological implications

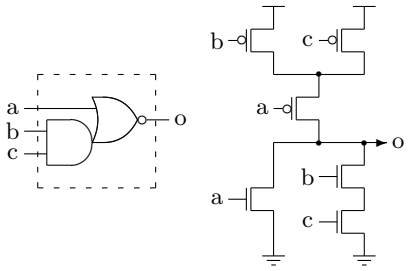

Figure 2: Static Gate $o = \overline{(a + bc)}$

to make worst case vector coverings for max-delay (pushout) and min-delay (pullin).

4.1 Remove Conflict and "tristate" Cubes

A superset of the valid vector set is created from the minterms in the s or r functions after removing the conflict cubes z_c and *keeper* cubes. Keeper cubes are all minterms that contain the output as a literal. Such cubes only retain the previous state, so they are not used for vector generation. Keeper cubes are removed from s and r by applying the Shannon cofactor on output $o \in O$ to the boolean functions, as $s_{\overline{o}}$ and r_o. This leaves the cubes which can actively switch the output up or down. The conflict cubes are removed through set subtraction, giving a simplified rising output function $s' = s_{\overline{o}} - z_c$, and the falling function $r' = r_o - z_c$.

This gives a set of minterms across which we can iterate to generate the superset of SIS and MIS vectors. These vector sets are pruned and refined in Section 5 using the full logic function to create the final valid vector set.

4.2 Creating Complete SIS Vector Superset

This algorithm iterates across every minterm in either the canonical SOP s (when output $o\uparrow$ rises) or r. The superset of SIS vectors is created by setting each literal in every cube to rise or fall.

if $o\uparrow$ then $f \longleftarrow s_{\overline{o}} - z_c$ else $f \longleftarrow r_o - z_c$

$\forall\, c \in f$

$\forall\, l_i \in L$ if $l_i \in c \wedge l_i \notin B \cup O$ create v:

$$v[j] = \begin{cases} \mathtt{r} & j = i \wedge c[i] = 1 \vee l_j = o \wedge o\uparrow \\ \mathtt{f} & j = i \wedge c[i] = 0 \vee l_j = o \wedge o\downarrow \\ \mathtt{0} & j \neq i \wedge l_j \in c \wedge c[j] = 0 \\ \mathtt{1} & j \neq i \wedge l_j \in c \wedge c[j] = 1 \\ \mathtt{-} & l_j \notin c \wedge l_j \neq o \end{cases}$$

This procedure applies Shannon decomposition to remove the keeper, and subtracts conflict minterms. A superset of SIS vectors are generated for every remaining minterm cube $c \in f$. A new vector is created for each literal in the cube that is not an output or feedback. The vector is the size of the cardinality of the input to the DCN $|L|$. For each vector, the literal l_i in v is set to rising if the literal is asserted in the cube, else it is set to fall. If the output o is an element of the vector, then it is set to rise (fall) when using function s (r). For all other literals l_j, the value of $v[j]$ is set to `0` or `1` if the literal l_j is in the cube based on the literal's value.

If the literal l_j is not in the cube then its value in the vector is don't care.

Refer to the gate in Figure 2. The reset function $r = \{a, bc\}$, the conflict and feedback sets z_c and B are empty, the output set $O = \{o\}$, and the input literals are $L = \{a, b, c\}$. This produces the three vectors $\{\texttt{r--}, \texttt{-r1}, \texttt{-1r}\}$ with a vector signal order of "abc".

4.3 Complete MIS Vector Superset

The MIS vector superset set is generated from the SIS vectors. All possible vectors are created by allowing don't care and controlling values to rise or fall as permitted by their minterm cubes. MIS vectors can have anywhere from 2 to n literals in the set L switching where $2 \leq n \leq |L|$. These algorithms take parameter n to limit the maximum number of signals that can switch concurrently. This limit is set to 2 in the results of this paper without loss of generality. The MIS vector set M (initialized $\emptyset$) is calculated with the recursive function $g(v, i, k, n)$ where v is a SIS vector, i is the literal index, and k is the current switching count, n is the max switching ceiling:

For all vectors v in SIS vector set, call $g(v, 0, 1, 2)$:

if $i = |L|$ then when $k \geq 2$, $M = M \cup v$
else
$g(v, i+1, k, n)$
if $v[i] \notin \{\texttt{r}, \texttt{f}\} \wedge k < n$
if $v[i] \neq 0$
$g(v : v[i] \longleftarrow \texttt{r}, i+1, k+1, n)$
if $v[i] \neq 1$
$g(v : v[i] \longleftarrow \texttt{f}, i+1, k+1, n)$

For each SIS vector v, a recursive call is made using the next literal i. When we have iterated across all literals in the vector, we add the MIS vector to our vector set if it has at least two signals switching. If the current literal is not rising or falling and the switching ceiling has not been reached, then we recursively call this function on the next literal up to two times, incrementing the MIS count k. If the value of the current literal in the SIS vector is not zero, we set the current value in the MIS vector to rising and recurse. If the value of the literal is not one, we set the literal to falling and recurse.

The complete MIS pulldown vector set for the circuit of Figure 2 is $\{\texttt{rr-}, \texttt{rf-}, \texttt{r-r}, \texttt{r-f}, \texttt{f1r}, \texttt{fr1}, \texttt{-rr}\}$. This was created from the three SIS vectors in the example from Section 4.2.

4.4 Maximum Delay MIS Vector Superset

To the first order, the maximum delay for a DCN operating under MIS occurs when transistors in a series stack are switching. The topological information in the SOP format is used to create a set of vectors covering all combinations of series transistors that can simultaneously switch. The function $e(v, c, i, k, n)$ is called creating a transition vector v for the set or reset function. The function produces the MIS vector set M (initialized $\emptyset$) using minterm cube c, literal index i, transition count k, and MIS concurrency ceiling of n.

if $o\uparrow$ $f \longleftarrow (s_{(\overline{o} \cup B)} \cap s_{\overline{B}}) - z_c$
else $f \longleftarrow (r_{(o \cup B)} \cap r_{\overline{B}}) - z_c$

$\forall\, c \in f$ call $e(v, c, 0, 0, 2)$

if $i = |L|$ then when $k \geq 2$, $M = M \cup v$
else

$$v[i] = \begin{cases} \texttt{-} & l_i \notin c \\ \texttt{h} & c[i] = 1 \\ \texttt{l} & c[i] = 0 \end{cases}$$

$e(v, c, i+1, k, n)$
if $k < n \wedge l_i \in c$

$$v[i] = \begin{cases} \texttt{r} & c[i] = 1 \\ \texttt{f} & c[i] = 0 \end{cases}$$

$e(v, c, i+1, k+1, n)$

This algorithm creates transition cubes by first removing keeper and feedback cubes with Shannon decomposition and subtracting the conflict set from either the rising or falling outputs. The function e is then called for each minterm cube. At the end of the recursive iteration, the vector is added to the set only if it has at least two transitioning literals. The vector index is set to don't care if its corresponding literal is not in the cube. Otherwise it is set to high or low based on the literal's value, and a recursive call is made incrementing the literal index. When the MIS ceiling has not been reached, the current literal is set to rising or falling, based on the value of the literal in the cube. A recursive call is then made incrementing the literal index and transition count.

Using the reset function $r = \{a, bc\}$ from Figure 2, the max delay MIS vector set $\{\texttt{-rr}\}$ is created.

4.5 Minimum Delay MIS Vector Superset

To the first order, the minimum delay occurs when multiple paths to power or ground are concurrently enabled. The minterms in a POS format create all parallel bisections in the gate topology. The algorithm for calculating maximum pushout in Section 4.4 can be used when slightly modified to create the maximum pullin.

Following the Shannon decomposition on functions s and r the function f is translated into POS format $\hat{f}$. The recursive function e is slightly modified into function $\hat{e}$ which replaces the initial vector assignment with its dual as shown in Equation 2; if the literal is asserted (unasserted) in the cube it is unasserted (asserted) in the vector. This ensures that the minterm is only enabled by rising or falling literals.

$$v[i] = \begin{cases} \texttt{-} & l_i \notin c \\ \texttt{l} & c[i] = 1 \\ \texttt{h} & c[i] = 0 \end{cases} \tag{2}$$

The POS format for the pulldown of Figure 2 is $\hat{r} = \{ab, ac\}$, equivalent to the CNF form $(a + b)(a + c)$. The maximum delay vectors for this function are $\{\texttt{rr-}, \texttt{r-r}\}$.

5. PRUNING POTENTIAL VECTORS

Four candidate vector supersets were generated in Section 4. Each vector set is pruned for functional validity using the s and r functions. Further optimizations or unique constraints for maximum or minimum delay effects are applied when applicable. Pruning can be called concurrently with vector generation to increase efficiency in an implementation.

5.1 Functional Vector Pruning

Every vector in a vector set must meet the following three functional correctness constraints. While these ensure a functionally correct transition, they do not require the transition to be *glitch-free* or *stable* at intermediate vector values.

1. correct initial voltage: if $o\uparrow$ then $f(\bullet v) = 0$ else $f(\bullet v) = 1$
2. vector ends in conducting state: If $o\uparrow$ then $v\bullet \subseteq s_{\overline{o}} - z_c$ else $v\bullet \subseteq r_o - z_c$
3. feedbacks not controlling: when $l_i \in B$ then $v\prime[i] \longleftarrow \bullet v[i]$ else $v\prime[i] \longleftarrow v[i]\bullet \wedge$ if $o\uparrow$ then $f(v\prime) = 1$ else $f(v\prime) = 0$

This requires that a rising vector must start low, end high, and cannot be controlled by the feedbacks. This will prune the example SIS set generated in Section 4.2 to the final falling set {`r0-`, `r10`, `0r1`, `01r`}. Don't care vectors are fully instantiated for simulation, so this gives in five pulldown ($o\downarrow$) vectors.

5.2 Functional MIS Vector Pruning

Functional pruning is sufficient for the complete SIS and MIS vector sets when evaluating static gates. An additional condition must hold when evaluating dynamic gates with feedback through a separate DCN. The auxiliary DCNs must also be functionally correct such that the pre- and postconditions generate the appropriate logic levels. This is not necessary if the keeper gate is an inverter, but is necessary for more complicated keeper logic.

5.3 Maximum Delay Vector Pruning

MIS vector delay is maximized when all switching transistors are topologically connected in series. Since the potential vector set was generated based on a single minterm, it is possible that multiple minterms can be concurrently enabled, speeding up the gate. An additional constraint is therefore added to ensure multiple minterms are not asserted. The function $g(v, f)$ is true when only one minterm in function f is asserted for the vector v. A second pruning condition ensures that the vector cannot enable the output until all rising or falling signals have switched. This holds if the vector transitions properly when all but one of the rising or falling literals are set to don't care in the precondition of the vector.

1. single asserted minterm: if $o\uparrow$ $g(v\bullet, s_{\overline{o}})$ else $g(v\bullet, r_o)$
2. stable until output: If $o\uparrow$ $f \longleftarrow r - z_c$ else $f \longleftarrow s - z_c$. $f(\bullet v\prime) = 1 : \forall\ v[i] \in \{\mathtt{r}, \mathtt{f}\} : (\forall\ v[j] : i \neq j \wedge v[j] \in \{\mathtt{r}, \mathtt{f}\}$ then $v\prime[j] \longleftarrow$ `-`)

For Figure 2 only vector {`0rr`} holds for falling outputs.

5.4 Minimum Delay Vector Pruning

Each vector is set to deliver maximum current to the output to minimize the delay. Each potential minimum delay vector from Section 4.5 is expanded to fully instantiate don't cares. The vector is kept if for each vector literal that is 1, the function is positive unate ($f_{\overline{l_i}} \subseteq f_{l_i}$) or not negative unate for that literal. Likewise, for each vector literal that is `0`, the function must be negative unate ($f_{l_i} \subseteq f_{\overline{l_i}}$) or not positive unate for that literal.

function	input pins	Full SIS sets	Full MIS sets	MIS max-del		MIS min-del	
				$o\uparrow$	$o\downarrow$	$o\downarrow$	$o\uparrow$
$\overline{ab}$	2	2	1	0	1	0	1
$\overline{a+b}$	2	2	1	1	0	1	0
$\overline{abc}$	3	3	3	0	3	0	3
$\overline{a+b+c}$	3	3	3	3	0	3	0
$\overline{a+bc}$	3	5	7	2	1	2	1
$\overline{ab+ac}$	3	5	7	1	2	1	2
$\overline{ab+ac+bc}$	3	6	6	3	3	3	3
$\overline{abcd}$	4	4	6	0	6	0	6
$\overline{abc+abd}$	4	8	16	1	6	1	6
$\overline{a+bcd}$	4	10	24	3	3	3	3
$\overline{a+bc+bd}$	4	10	22	6	2	4	2
$\overline{ab+acd}$	4	10	22	2	6	2	4
$\overline{ab+cd}$	4	12	26	4	6	4	2
$\overline{(a+b)(c+d)}$	4	12	26	6	4	2	4
$\overline{abcd+abce}$	5	11	28	1	12	1	12
$\overline{abc+abde}$	5	15	42	2	15	2	9
$\overline{abc+ade}$	5	19	54	4	18	4	6
$\overline{ab+acde}$	5	19	60	3	13	3	7
$\overline{ab+cde}$	5	23	70	6	16	6	4
$\overline{a+bcde}$	5	19	66	4	6	4	6
$\overline{a(b+c)(d+e)}$	5	21	62	6	12	2	12
$\overline{(a+b)(c+d)}$ $\overline{(e+f)}$	6	54	207	27	24	3	24
$\overline{abc+def}$	6	42	159	9	42	9	6
All vectors	94	630	1836	94	201	60	123
Average	4.1	27.4	79.8	4.1	8.7	2.6	5.3

Table 1: Vector results for 90nm static cell library. The function of the cell is listed with input pin count, the complete single input switching (SIS) and multiple input switching (MIS) vector set sizes for rising and falling outputs sets, the worst case max-delay vectors for rising outputs $o\uparrow$ and for falling outputs $o\downarrow$, and the worst case min-delay vectors for outputs that fall and rise.

1. maximal drive: if $o\uparrow$ $f \longleftarrow \hat{s}$ else $f \longleftarrow \hat{r} : \forall\ v[i]$ if $v[i] = 1$ then $f_{\overline{l_i}} \subseteq f_{l_i} \ \vee\ f_{l_i} \nsubseteq f_{\overline{l_i}}$. if $v[i] = 0$ then $f_{l_i} \subseteq f_{\overline{l_i}} \ \vee\ f_{\overline{l_i}} \nsubseteq f_{l_i}$.

This prunes the four possible vectors for our example to the final set {`rr1`, `r1r`}

6. RESULTS

The vector generation and pruning algorithms were run on 25 different cells consisting of 23 different logic functions. There were two different 2-input NAND and NOR topologies.

6.1 Vector Set Analysis

Table 1 summarizes the size of the vector sets for each of the cell families. The logic function of the gates are shown in the first column. The gates ranged from two to six input pins, as shown in the second column. The third and fourth columns show the size of the complete set of single input switching and multiple input switching vectors that toggle the gate's output. The full set sizes were the same for rising and falling outputs. The next two columns show the set

Class	Vectors for falling output $o\downarrow$
Full SIS	`r00 r01 r10 0r1 01r`
Full MIS	`rr0 rr1 rf0 r0r r0f r1r 0rr`
max-delay	`0rr`
min-delay	`rr1 r1r`

Table 2: Pruned worst-case vectors for circuit of Figure 2

Vector class	Full set base / rel.	Reduced set base / rel.	Reduced Percent
optimal SIS	1260 / 1.0	188 / 1.0	85%
max-delay	3672 / 2.9	295 / 1.6	92%
min-delay	3672 / 2.9	183 / 1.0	95%

Table 3: Vector Set Sizes from Table 1. Comparison of total vector set sizes between the minimum SIS vector set baseline and the worst case MIS vector sets. Full sets and SIS set multiplied by two for combined rise and fall comparison. Minimum SIS set size assumed to be equal to # input pins.

sizes for worst case max-delay pushout vectors for falling and rising outputs. The last two columns show the set sizes for worst case min-delay pullin for falling and rising outputs. The final two rows shows the total vectors and the average per gate. The minimum SIS vector set size is equal to the number of input pins for all cases (min and max rise and fall delay), since at least each input pin must switch.

Note that some of the MIS pruned vector sets contain no entries - such as the rising max-delay for the NAND gate in the first row. In such cases the pruning algorithm has determined there are no MIS vectors that will delay the output because there are no transistors in series.

The complete set of vectors for the falling output for the example gate of Figure 2 are shown in Table 2. This gate $\overline{(a + bc)}$ is in row 5 of Table 1.

The vector sizes directly impact the library characterization for static CBD timing and runtime for transistor level timing. Table 3 shows the average number of vectors for these gates. This assumes the minimum SIS vector set needing a single vector per input pin can be found that sufficiently covers the worst case delay for all slopes and loads. Note that the MIS worst case min-delay vector count is smaller than the SIS vector size. The max delay MIS vector size is about 60% larger than an optimal SIS vector set. The optimal SIS vector set has an 85% reduction in vector size from the full vector set. The worst case MIS min and max delay vector sets show a 92% and 95% reduction from the full vector set respectively.

6.2 Simulation Results

A simulation study of the 25 gates was performed to determine the quality of the reduced MIS vector set. The fully extracted parasitics for each cell were used in the simulations. Each cell had a range of sizes to support various output loads. Monte Carlo analysis was performed by randomly picking one of the cell types, a valid output load and an associated cell size, and an independent input slope for each switching input. The 50% transition point for all inputs were aligned. A complete simulation run for random parameters was performed on the cell for the full rising and falling MIS vector sets – 2×79.8 simulations per cell on average. The worst case delay and its vector were recorded for the complete vector set and the pruned vector sets for each of the simulation runs. This loop was iterated 25,000 times for approximately 4 million simulations. The result for the full set was compared against the pruned set to determine if the worst case was covered. If not, the size of the error was calculated.

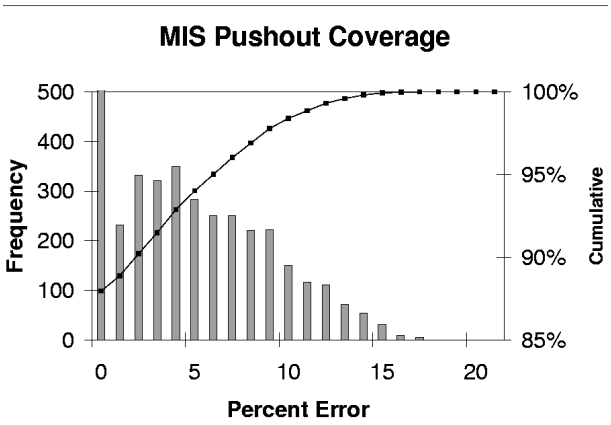

Figure 3: Graph of max-delay (pushout) errors.

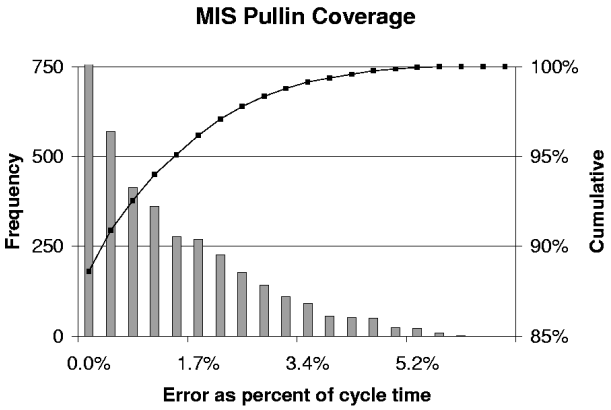

Figure 4: Graph of min-delay (pullin) errors.

The data for these runs is plotted in Figures 3 and 4. The error for the max-delay pushout is reported as a percentage difference from the actual worst case. For min-delay, the value is normalized by dividing the error by an aggressive clock frequency for this technology node. This shows that the largest max or min delays were contained in the reduced vector set in approximately 88% of the simulations. In approximately 94% of the cases the error was zero or relatively insignificant (within 5%). This shows that the automatic pruning algorithm did a good job of covering the worst case vectors.

The vector that created the worst case delays for each random slope and load combination were recorded. Just as in the SIS case, a single vector is not sufficient to cover the worst case delays in a gate for all valid slope and load combinations. Table 4 shows the average number of MIS simulation vectors required to correctly cover the worst case condition. An average of two vectors per cell is required to cover worst case conditions for falling transitions; slightly less for

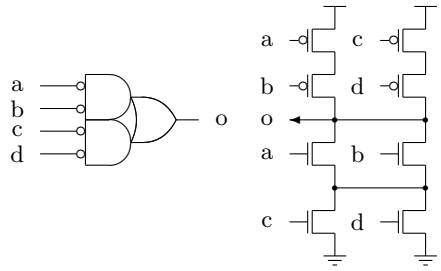

Figure 5: Gate $f = \overline{ac + ad + bc + bd}$

rising. Only one-third to one-fourth of the reduced MIS vectors generated by this algorithm were needed to cover the worst case conditions. This implies that other means, such as some structural pruning following the logical pruning, may be a useful means of further reducing vector sets.

Eight of the 25 cells contained one or more slope, load, and sizing condition where the worst case required a vector outside the pruned set. All errors occurred on the rising outputs; none on falling outputs. One gate contained only a single vector with an error of less than 0.1%. Two cells contained errors in all of their rising outputs.

Of the 3012 erroneous max-delay simulation sets, all but one were from vectors with both rising and falling literals. The algorithm for pruning MIS vectors did not consider these vectors as candidates, and therefore they were not included in the pruned set. The mechanism that appears to have resulted in these failures is demonstrated with the circuit of Figure 5, with the set function $s = \{\overline{a}\overline{b} + \overline{c}\overline{d}\}$ and the reset function $r = \{ac + ad + bc + bd\}$. One worst case vector is `1rf0`. Note that cube ac is initially pulling the gate low, and the other reset cubes are off. Cube ac de-asserts and cube $\overline{c}\overline{d}$ is asserted to pull the gate high. However, during this transition, reset cube bc has signal b rising and c falling. This creates a glitch on the output of this cube as this term partially asserts during the transition, sourcing current that fights the pullup cube $\overline{c}\overline{d}$. This condition is exacerbated with strong NMOS devices compared to the PMOS, particularly with slow transition times.

7. SUMMARY

An algorithm was described that uses logic to represent circuit topology and automatically create full and worst-case multiple input switching vector sets. This algorithm was tested on a modern static cell library at a 90nm design node. The algorithms presented here generate the full set of vectors that will flip the output under both single and multiple input switching conditions. This set is then automatically pruned to generate the worst-case rising and falling min- and max-delay vector sets with a reduction of 92-95% over the full vector set. The reduced set sizes are approximately the same size as the minimum single-input switching vector sets. Each set can be characterized and used independently.

The accuracy of the reduced vector set was evaluated against the complete MIS vector set using Monte Carlo simulation. Random input slopes, output loads, and cell sizes were selected. The reduced set showed satisfactory accuracy, as 88% of the simulation sets were error free, and in 94% of the cases the error was deemed insignificant. The maximum error was less than 16%. Examination of the erroneous vectors indicates that the errors all appear to be caused by second order effects such increased short circuit current from glitching cubes. The patterns of the error vectors from this study indicate that it may be possible to improve the error characterization of the reduced within the logic framework by adding an additional small set of vectors.

In many cells two or more vectors exhibited the worst-case delays for different rise time and load. However, this still only included a fraction of the worst-case vector sets as only 23–30% were needed to model the worst-case delays. This leaves significant room to improve this reduced vector set using either logical or structural methods, or a combination of both, if one is only searching for the worst case min or max delay for a cell.

	used $o\uparrow$	**used** $o\downarrow$	**used/ set size**	**used/ set size**
max-delay	1.2	2.0	30%	23%
min-delay	2.3	2.0	39%	69%

Table 4: Actual worst-case vectors from simulation versus pruned set

8. ACKNOWLEDGMENTS

The bulk of the work and all of the results presented here were obtained while the authors were with Intel Corp., Strategic CAD Labs, Hillsboro Oregon.

9. REFERENCES

[1] V. Chandramouli and K. A. Sakallah. Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time. In *33rd Design Automation Conference Proceedings 1996*, pages 617–622, June 1996.

[2] L.-C. Chen, S. K. Gupta, and M. A. Breuer. A New Gate Delay Model for Simultaneous Switching and its Applications. In *Design Automation Conference Proceedings*, pages 289–294, June 2001.

[3] Y.-H. Jun, K. Jun, and S.-B. Park. An Accurate and Efficient Delay Time Modeling for MOS Logic Circuits Using Polynomial Approximation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(9):1027–1032, Sept. 1989.

[4] L. McMurchie and C. Sechen. WTA: Waveform-Based Timing Analysis for Deep Submicron Circuits. In *International Conference on Computer-Aided Design (ICCAD'02)*, pages 625–631, November 2002.

[5] C. E. Molnar, I. W. Jones, W. S. Coates, J. K. Lexau, S. M. Fairbanks, and I. E. Sutherland. Two FIFO Ring Performance Experiments. *Proceedings of the IEEE*, 87(2):297–307, February 1999.

[6] K. T. Tang and E. G. Friedman. Delay Uncertainty Due To On-Chip Simultaneous Switching Noise in High Performance CMOS Integrated circuits. In *IEEE Workshop on Signal Processing Systems*, pages 633–642. IEEE, Oct. 2000.