An efficient and scalable STA tool with direct path estimation and exhaustive sensitization vector exploration for optimal delay computation

Salvador Barceló, Xavier Gili, Sebastià Bota, Jaume Segura
University of Balearic Islands (UIB)
Palma de Mallorca, Spain
salva.barcelo@uib.es

Abstract— ***We present a STA tool based on a single-pass true path computation that efficiently determines the critical path list. Given that it does not rely on a two-step process it can be programmed to find efficiently the N true paths from a circuit. We also report and analyze the dependence of complex gates delay with the sensitization vector and its variation (that gets up to 15% in 65nm technologies), and consider such effect in the path delay estimation. Delay is computed from a simple polynomial analytical description that requires a one-time library parameter extraction process, making it highly scalable. Results on combinational ISCAS synthesized for three technologies (130nm, 90nm and 65nm) provide better results in computation time, number of paths reported and delay estimation for these paths compared to a commercial tool.***

Keywords: delay-model, timing-analysis

I. Introduction

Timing analysis is a key step in the VLSI design flow whose significance and complexity increases with technology scaling due to new physical phenomena appearing in nanometer technologies [1][2]. The yield of the manufacturing process can increase considerably using a highly accurate timing analysis tool capable of finding true critical paths, and identifying those gates having higher sensibility to process variations and environmental conditions [3].

When a circuit design is synthesized using standard cells, CAD algorithms are designed to reduce circuit area, power consumption and propagation delays in addition to optimizing other parameters. To accomplish this goal synthesis tools use complex gates, i.e. circuit structures that combine primitive logic functions like NOT, AND, OR, NAND, NOR, in a single CMOS structure that reduces the number of transistors required to perform a given logic function. Typically, complex gates comprise a combination of few primitive functions (as detailed in Section II) although more complicated functions like full-adder or multiplexer are also common. In the context of timing analysis, a typical characteristic of complex gates vs. basic gates is that, in general, it is possible to find a set of vectors that sensitize each gate input while single gates have typically only one sensitization vector [4]. In this work we show that the gate delay when propagating a transition through a given input of a complex gate may vary significantly depending on the input vector used to sensitize such an input with the consequent impact on the circuit-level timing computation.

In some works, complex gates are converted to primitive gates prior to timing analysis thus applying the delay model to basic gates [5]. This methodology may be a source of inaccuracies since the circuit used for simulation has a topology that differs from the actual circuit structure being finally manufactured. Other works analyze the delay of complex gates through a transistor-level approach providing good accuracy at the cost of very complex expressions that result in a slow computation time at the circuit level [6][7].

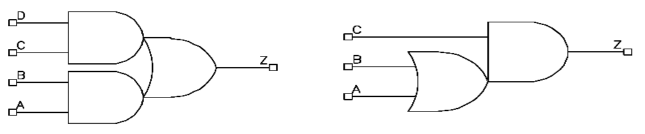
Most critical path algorithms operate in a two-step procedure such that first look for structural paths and compute their delay, and then try to sensitize iteratively the longest paths until the longest true path is found [8]. In this way, the delay is computed independently of the particular input vector used to sensitize each complex gate, which may introduce a relatively large uncertainty in overall delay estimations.

In this work we analyze the impact of the sensitization input vector on the propagation delay for complex gates showing that delay variations may get up to 20% depending on the technology used. We provide an insight about the root causes of such variations through a careful transistor level analysis. We conclude that, given the delay variation observed depending on the sensitization vector, timing analysis must be done considering which sensitization vector is used.

The rest of the paper is organized as follows: In section 2 we show two examples of input-vector-dependent delay and describe its causes. Section 3 explains the method used for delay estimation, while the results obtained with this method are detailed in Section 4. Finally the conclusions and future work are discussed in Section 5.

II. Complex gates delay variation examples

Without lose of generality, we illustrate the delay dependence with the sensitization vector using two complex gates included in almost all standard cell libraries. One of such gates is AO22 (referred to as AO2N in some technologies), being a four input gate that implements the logic function in (1), and whose logic symbol is shown in Fig. 1a. The other complex gate considered is



a) AO22 b) OA12

Figure 1: Two examples of complex gates.

$$Z = A * B + C * D \tag{1}$$

$$Z = (A + B) * C \tag{2}$$

OA12 (AO7N in some technologies), being a three input gate for which only one of its inputs has multiple input vectors to sensitize the gate. The gate logic function is given by (2) and its symbol is shown in Fig. 1b.

Gates AO22 and OA12, and in general all complex logic cells, have more than one input vector that sensitizes each input and allows propagating a transition through such input toward the gate output. The sensitization vectors for each input are computed easily from the gate logic function. For some gate inputs, in some cases only one input vector allows propagating a transition through such an input, but in most cases more than one sensitization vector is found. Tables 1 and 2 list all sensitization vectors for each complex gate input. The logic value "T", represents a transition either rising or falling.

Table 1 shows that gate AO22 has three sensitization vectors for each input, leading to a total of 12 different delay propagation values. For gate OA12 only one input (input C) shows multiple sensitization vectors as shown in Table 2.

We carried extensive electrical simulations to compute the gate delays through each input for all the sensitization vectors for three CMOS technologies (130nm, 90nm and 65nm) at nominal supply voltage and 25ºC. Each gate was loaded with a gate of the same type.

Table 3 shows the delay results obtained when propagating a transition through input A for gate AO22 for its three sensitization vectors, and Table 4 provides the results when propagating a transition through input C for gate OA12. For each gate the Case 1 delay is taken as a reference value to which the delay of Cases 2 and 3 are referred.

Results in Tables 3 and 4 show propagation delay variations with the input sensitization vector that reach up to 22% depending on the gate structure and technology. For the 65nm technology, delay variation may get to almost 12 % (Case 2 vs Case 1 for gate AO22).

Table 1. Propagation table AO22

	A	B	C	D	Z
Case 1	T	1	0	0	T
Case 2	T	1	1	0	T
Case 3	T	1	0	1	T
Case 1	1	T	0	0	T
Case 2	1	T	1	0	T
Case 3	1	T	0	1	T
Case 1	0	0	T	1	T
Case 2	1	0	T	1	T
Case 3	0	1	T	1	T
Case 1	0	0	1	T	T
Case 2	1	0	1	T	T
Case 3	0	1	1	T	T

Table 2: Propagation table OA12

	A	B	C	Z
Case 1	T	0	1	T
Case 1	0	T	1	T
Case 1	1	0	T	T
Case 2	0	1	T	T
Case 3	1	1	T	T

Table 3. AO22 Propagation delay (Input A) (delays in ps)

		Case 1	Case 2	Case 3	%diff 2	%diff 3
130nm	In Rise	121,29	125,62	121,51	3,57%	0,18%
	In Fall	131,45	157,25	149,15	19,63%	13,47%
90nm	In Rise	60,10	63,13	59,16	5,04%	-1,56%
	In Fall	76,37	92,86	85,71	21,59%	12,23%
65nm	In Rise	110,23	109,85	107,40	-0,35%	-2,57%
	In Fall	116,87	131,01	125,30	12,10%	7,21%

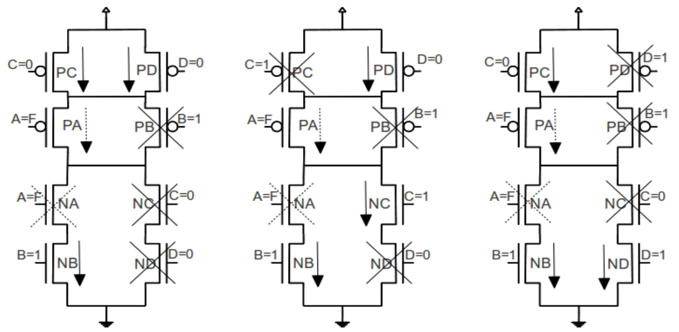
Table 4. OA12 Propagation delay (Input C) (delays in ps)

		Case 1	Case 2	Case 3	%diff 2	%diff 3
130nm	In Rise	120,30	105,46	99,89	-12,33%	-16,97%
	In Fall	151,23	146,71	149,16	-2,99%	-1,37%
90nm	In Rise	60,47	51,98	50,13	-14,05%	-17,11%
	In Fall	96,68	92,62	94,48	-4,20%	-2,27%
65nm	In Rise	99,07	93,60	89,94	-5,53%	-9,22%
	In Fall	91,13	88,78	90,25	-2,57%	-0,97%

III. Transistor level analysis

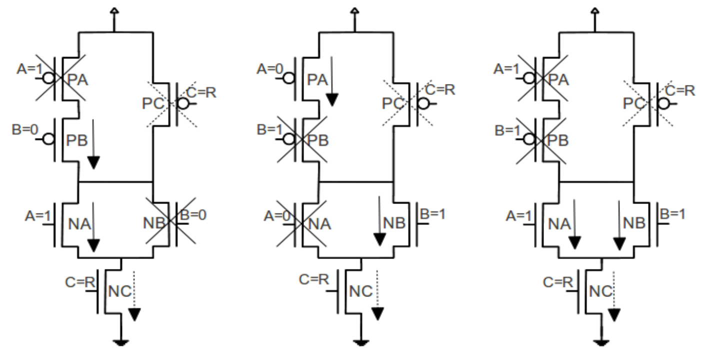
We investigated the root cause of the delay variations with the sensitization vector to get insight on this phenomenon through a transistor-level analysis. The two complex gates considered implement non-inverting functions, and require an output inverter for a CMOS implementation. Such an inverter does not influence the delay variation with the sensitization vector and therefore it is not considered in the transistor-level analysis. Fig. 2 shows the transistor-level analysis for gate AO22 and represents the three input vectors that propagate a falling transition through input A. A non-dashed cross on a transistor indicates that such device is OFF, while a non-dashed arrow close to a device indicates that such transistor is on. A dashed cross or arrow represents that such a transistor makes a transition and indicates the final state once the switching input is at its final state (i.e. a dashed arrow indicates a transistor that switched from off to on, while a dashed cross indicates a transistor that changed from OFF to ON).

Results in Table 3 show that Fig. 2a corresponds to the fastest transition, while Fig. 2b corresponds to the slowest one. As shown in the Figures, the current charging the output node must



a) Case 1 *b) Case 2* *c) Case 3*

Figure 2. Gate AO22 transistor-level schematic and current paths for each sensitization vector.



a) Case 1 *b) Case 2* *c) Case 3*

Figure 3. Gate OA12 transistor-level schematic and current paths for each sensitization vector.

pass through transistor pA. In the fastest case, both parallel transistors pC and pD are ON, allowing a higher current to pass through pA, leading to a quicker charging of the output node. In the other two cases only one of the two top parallel transistors (either pC or pD) is ON. The relative delay difference between Case 2 and Case 3, is due to the transistor nC being ON in Case 2 and creating an additional current path to charge internal parasitic capacitors. Such an additional current is taken from the current coming from the pMOS devices that is therefore not contributing to switch the output. As a result, the output transition is slower in this case due to such a current component.

The behavior of gate OA12 is analogous to the AO22 case. Fig. 3 shows the transistor-level diagram for each sensitization vectors that pass a rising transition at input C toward the gate output. Fig. 3c corresponds to the fastest transition. For this input vector transistors nA and nB are both ON, increasing the current available through nC with respect to the other two cases where only nA or nB are ON. Case 2 (Fig. 3b) shows a delay slightly larger than Case 1 (in both cases only one nMOS transistor is ON in the parallel structure) since transistor pB is ON increasing the amount of charge that must be drained from the output node when charging the internal parasitic capacitors.

The analysis carried over in this section together with the results shown in Tables 3 and 4 highlight that if a logic gate has more than one sensitization vector for a given input it is important to consider which input vector is actually applied to sensitize such input to the gate when performing timing analysis.

IV. Delay model, heuristic and tool

We developed a timing analysis tool that combines a specific delay model and algorithm to find true paths in a combinational circuit. The delay model is analytical through a polynomial expression similar to SPDM [9][10]. Such a polynomial model is used to estimate both the gate propagation delay and the output transition time, since the latter is required to compute the propagation delay of the next gate within the path. The second component of the timing analysis tool is the algorithm developed to find true paths in a combinational circuit. Such an algorithm is based on the RESIST algorithm [11], and was specifically developed to consider the dependence of the delay with the input vector for complex gates.

A. The delay model

The delay model includes multiple variables like input transition time, output load, temperature and supply voltage, and can be easily extended to accommodate additional variables. The analytical nature of the model provides some advantages over the widely used LUT (Look-Up Table) based approaches. The main advantages are: a faster computation time due to interpolations required by LUT methods, and less memory space required to store the model data.

Equation (3) shows the basic form of the analytical model used to compute propagation delay and output transition time of a gate. Fo is the equivalent fanout (defined below), t_{in} the input transition time, T is the temperature and V_{DD} the supply voltage. The parameters of the model, represented by P_{ijkl} in (3) are obtained from electrical simulations of the cell.

$$f(Fo, t_{\text{in}}, T, V_{DD}) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{o} \sum_{l=0}^{p} P_{ijkl} \cdot Fo^{i} \cdot t_{\text{in}}^{j} \cdot T^{k} \cdot V_{DD}^{l} \quad (3)$$

The equivalent fanout (Fo) of a gate G, is the ratio between the capacitance at the gate output Cout (considering all the actual gates connected) vs. its input capacitance. Its value would correspond to the number of gates of the same type than G that should be connected to G output to obtain Cout. The equivalent fanout is computed from the input capacitance of each gate type estimated by integrating the input current during an input transition. Such value divided by the supply voltage value provides an estimation of the gate input capacitance. Electrical simulations showed that this value is independent of input transition time, temperature and supply voltage, but takes different values for rising and falling edges.

The electrical simulations from which the model parameters are obtained, are done automatically and systematically for a given technology library, and consist of a set of iterative simulations. Each iteration uses a different combination of values for each variable considered, for which the propagation delay and output transition time for rising and falling input transition are determined. Such an iterative simulations are repeated for each gate input and each input vector that sensitizes that input. This is done to account for the dependence described in Section II.

Once the simulations are done, a recursive polynomial regression procedure is applied to extract the model parameters. The maximum order for each variable (indexes *m, n, o, p* in (3)), are adjusted during the extraction process to provide the desired accuracy in the estimation.

An application was developed to perform the whole process automatically: determining all sensitization vectors for each gate input, generating the scripts for the iterative electrical simulations, and finally extracting the model parameters from simulations.

B. The path finding algorithm

The second component of the STA tool is the algorithm that finds all paths that propagate a transition from each input of a combinational circuit to its output (i.e finds the true paths). Most existing tools are based on a two-step procedure that first identifies a set of structural paths without checking if they are true paths and then compute their delay [8]. On a second step, the tool verifies if the set of paths ordered from longest to shortest are true paths, ending up with a list of ordered true paths. Such procedure has the disadvantage of ignoring how many paths must be included in the initial list to find a number N of slowest true paths. Moreover, if the delay of each gate is estimated before computing the sensitization vector, the gate delay value might be incorrectly estimated contributing to an accumulative error in the delay estimation process because of the delay dependence with the sensitization vector reported in previous section.

In this work we develop a path finding algorithm that sensitizes the path while computing its traverse through the circuit. The algorithm preserves as different paths those having the same course (i.e. traversing the same sequence of gates) but using different sensitization vectors at any of the gates. In this way, the information about the delay dependence with the sensitization vector is maintained.

The algorithm starts at a circuit input and advances node to node until an output is reached. If the node being analyzed has a fanout greater than 1 (i.e. is a fanout stem), or the next gate has multiple sensitization vectors, the process state is saved. The algorithm tries to sensitize the next gate and justify the logic values assigned until the inputs of the circuit are reach. If a logic incompatibility is found, all the paths that sharing the current sub-path are discarded and the algorithm jumps to the last saved point. If no incompatibility is found, then the output node of the sensitized gate becomes the new current node and the process is repeated.

Once the algorithm reaches an output node, the path is saved, and the algorithm returns to the last saved state and continues sensitizing the next gate having current node as an input. If there are no states saved, the process starts from the next circuit input node until the last input node is analyzed. Each time that a logic value is assigned to a node, such value is propagated through all the gates having such node as an input. This procedure helps in early detection of logic inconsistencies and improves the algorithm performance because it is less complex than a justification process [12].

To perform this logic propagation step efficiently, the algorithm uses a logic system with semi-undetermined values that allow identifying a logic incompatibility before all implied nodes are set (e.g. a falling transition applied to input A of an AND2 gate with an undetermined value to the B input, leads to a state that starts with an unknown value, but ends with a logic 0, this is a semi-undetermined logic value represented as "X0") [13]. Moreover, the logic system developed has the property of considering simultaneously both transitions on a given node (i.e. rising and falling), we call this dual value logic system. Using this technique the algorithm computes simultaneously both transitions through a given path the same step considering only one stored value per node. This method leads to an increase in the algorithm speed to trace all true paths, and avoids passing twice through the same path (one for rising input transition and another for falling input transition).

V. Results

A. Test circuit

We first report initial results on a simple circuit shown in Fig. 4 to illustrate how the developed algorithm works compared to a commercial tool. The critical path of the sample circuit in Fig. 4 passes through input A of an AO22 complex gate (shown in dashed box). The easiest way to sensitize the complex gate leads to the smaller propagation delay for this path, although it is also possible to sensitize the gate with an input vector that provides a larger delay. The commercial tool correctly provides the critical path that propagates a falling edge through nodes N1-n10-n11-n12-N20, as expected. The input vector used to sensitize the critical path is:

N1=F N2=1 N3=1 N4=1 N5=1 N6=0 N7=X

corresponding to the easiest option that assigns a logic 0 to node N6 and therefore doesn't require to assign n13 nor justifying its value to an input node. Setting N6 to 0 leads to the shortest way to sensitize the AO22 gate, but ignores another case having a larger propagation delay for that path, that can be obtained sensitizing gate AO22 with a vector that results in a larger delay. This second vector requires a more complex justification process to assign logic values until reaching an input node.

The tool developed provides two paths passing through the same nodes and starting with a falling transition, each with different input vector. One is the same vector provided by the commercial tool, and the second one is:

N1=F N2=1 N3=1 N4=1 N5=1 N6=1 N7=0

Table 5 provides the delay obtained from electrical simulations of the critical path for the two input vectors. It is shown that the additional path provided by the tool developed exhibits a delay increase of 7% with respect to the one given by the commercial tool. Such an erroneous estimation is due to not considering the multiple sensitization vectors of complex gates. The tool proposed in this work identifies correctly the path with larger delay.

Table 5. Delay vs Input vector for the simple circuit in Fig 4.

Input vector	Delay (ps)
N1=F, N2=1, N3=1, N4=1, N5=1, N6=1, N7=0	387,553
N1=F, N2=1, N3=1, N4=1, N5=1, N6=0, N7=X	361,06

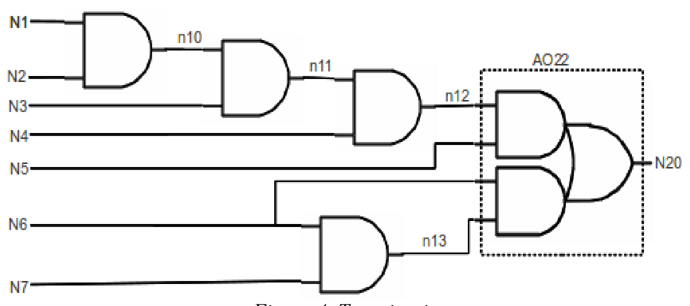


Figure 4. Test circuit

Table 6. Technology independent critical path identification results

	Developed tool			Commercial tool							
ISCAS Circuit	Input vectors	MultiInput Paths	CPU Time (s)	Backtrack limit	CPU Time (s)	#Paths	#True paths	#False paths	Backtrack limited	False path ratio	Worst delay prediction ratio
c17	32	8	< 1	1000	< 1	8	8	0	0	0,0%	62,5%
c432	10628	2018	15	1000		1000	680	62	258	32,0%	27,2%
c499	16752	4828	16,03	1000	7340	4828	593	0	4235	87,7%	56,7%
c880a	96172	5010	8,6	1000	23,4	1000	519	0	481	48,1%	10,0%
c1355	2120	0	3,667	--	--	--	--	--	--	--	--
c1908	10838	3128	2,59	1000	433	1000	234	85	681	76,6%	33,3%
c2670	137344	3472	66,25	1000	1397	1000	176	41	783	82,4%	4,2%
c3540	52348	8154	49,66	1000	2397	1000	266	8	726	73,5%	5,3%
c5315	773374	13082	161,63	1000	2315	1000	249	3	748	75,1%	0,0%
c6288	1154	54	111,04	1000	25,4	54	24	1	29	55,6%	75,0%
				5000	77	54	28	1	25	48,1%	78,6%
				10000	139	54	30	1	23	44,4%	80,0%
				25000	320	54	33	1	20	38,9%	81,8%
c7552	87542	11632	126	1000	642	500	132	0	368	73,6%	25,0%
				5000	1798	500	148	0	352	70,4%	22,3%

Table 7: 130nm delay comparison vs electrical simulation

	Developed tool				Commercial tool			
ISCAS Circuit	Mean path error	Max path error	Mean gate error	Max gate error	Mean path error	Max path error	Mean gate error	Max gate error
c17	1,92%	4,61%	1,91%	5,26%	9,94%	21,16%	8,63%	24,16%
c432	1,24%	2,59%	6,02%	28,96%	6,76%	7,53%	17,22%	44,11%
c499	3,31%	5,20%	6,44%	32,47%	4,11%	4,12%	11,70%	25,37%
c880a	2,11%	7,38%	4,63%	68,03%	3,31%	7,11%	13,78%	64,13%
c1908	1,66%	3,65%	4,13%	26,67%	7,39%	8,71%	17,99%	61,60%
c2670	0,59%	1,08%	4,10%	29,09%	8,95%	27,89%	15,31%	306,95%
c3540	3,04%	5,63%	5,33%	20,05%	5,10%	5,10%	19,45%	109,07%
c5315	6,31%	7,41%	6,32%	29,38%	10,60%	13,59%	17,75%	53,62%
c6288	2,39%	7,86%	3,50%	31,99%	10,59%	22,66%	15,38%	82,24%
c7552	5,38%	9,67%	7,24%	22,35%	11,59%	21,17%	16,23%	58,45%

In this work we only consider steady logic values applied to the inputs of complex gates, future versions of the tool are currently developed to consider multiple simultaneous transitions, as well as considering parameter variations on the delay model. Given that the tool is designed to rely on analytical delay descriptions only the delay model needs to be included. This process is already designed.

B. ISCAS circuits

The proposed STA tool, composed by the critical path algorithm and the delay estimation engine, was developed in C++ and ran on a Core2 Quad processor. The focus of this work is on the delay variation with the input vector for complex gates, and therefore results are focused on analyzing the delay of the paths having more than one sensitization input vector due to complex gates. We tested the tool developed using the ISCAS combinational circuits synthesized on three CMOS technologies, 130nm, 90nm and 65nm.

To generate the results we first determined the paths having more than one sensitization vector. Then the tool generated a script for the commercial tool to explore these paths, and import the report generated. With this information, we compared the delay estimation and the input values assigned to the complex gates within each path, to those generated by the developed tool and the electrical simulations. Finally, we computed the percentage of paths for which the commercial tool identifies correctly the input vector that provides the larger delay. Each path obtained was simulated electrically with *Spectre* to verify that it was really a true path and to determine the input vector providing the larger delay.

Table 8: 90nm delay comparison vs electrical simulation

	Developed tool				Commercial tool			
ISCAS Circuit	Mean path error	Max path error	Mean gate error	Max gate error	Mean path error	Max path error	Mean gate error	Max gate error
c17	2,93%	5,12%	3,21%	7,28%	19,92%	40,58%	18,42%	42,08%
c432	4,87%	8,87%	8,76%	52,22%	17,92%	20,23%	24,39%	73,58%
c499	11,20%	26,70%	7,96%	41,26%	16,15%	19,05%	24,77%	136,92%
c880a	6,21%	9,67%	6,74%	49,45%	18,31%	53,27%	19,71%	97,03%
c1908	2,88%	5,33%	6,59%	29,67%	17,67%	21,76%	28,55%	98,82%
c2670	2,32%	3,52%	5,17%	25,12%	16,26%	29,64%	21,71%	231,97%
c3540	4,11%	6,87%	6,21%	26,14%	15,89%	31,45%	26,87%	66,88%
c5315	5,64%	9,13%	5,16%	27,19%	18,52%	28,79%	20,36%	60,09%
c6288	3,55%	8,61%	4,94%	18,46%	13,25%	23,74%	18,56%	58,34%
c7552	7,36%	12,04%	7,62%	17,58%	16,23%	39,25%	23,34%	61,87%

Table 9: 65nm delay comparison vs electrical simulation

	Developed tool				Commercial tool			
ISCAS Circuit	Mean path error	Max path error	Mean gate error	Max gate error	Mean path error	Max path error	Mean gate error	Max gate error
c17	9,01%	12,71%	8,80%	16,59%	29,91%	59,99%	28,20%	59,99%
c432	7,82%	9,75%	9,29%	29,53%	29,09%	32,94%	31,56%	103,06%
c499	3,94%	5,91%	9,35%	35,27%	28,20%	33,99%	37,84%	248,47%
c880a	1,65%	3,79%	8,86%	30,87%	33,32%	99,43%	25,64%	129,93%
c1355	4,10%	7,01%	9,05%	32,68%	27,95%	34,82%	39,11%	136,04%
c1908	4,05%	5,96%	6,24%	21,16%	23,57%	31,39%	28,11%	156,99%
c2670	2,35%	6,81%	5,67%	16,34%	19,87%	29,60%	21,01%	49,58%
c3540	3,98%	7,61%	9,33%	30,07%	25,67%	40,12%	32,11%	77,43%
c5315	5,87%	10,04%	8,81%	22,34%	31,01%	57,64%	26,28%	81,42%
c6288	3,29%	8,75%	7,81%	20,14%	23,47%	62,37%	34,69%	64,58%
c7552	5,42%	11,01%	8,43%	19,68%	26,33%	42,11%	33,84%	67,12%

Table 6 shows the results about the ability to identify the input vector that induces the worst-case delay for each path for both the developed and commercial tool. The second column gives the total number of input vectors reported by the developed tool that can sensitize a true path, and the third column indicates the number of paths having more than one sensitization vector, being the paths of interest in this work. The fourth column is the computation time in seconds required by the tool to find the paths and the input vectors. The fifth column gives the backtrack limit used in the commercial tool and the next column shows the cpu time in seconds. Column “#Paths” gives the number of paths explored, column "#True paths" shows how many of these paths are identified as true paths by the commercial tool, and the column "#False paths" corresponds to the number of paths that the commercial tool misidentifies as false paths. The next column gives the number of paths for which the tool arrives to the backtrack limit without finding any input vector. The eleventh column provides the ratio between the paths for which the commercial tool is unable to find a sensitization input vector versus the total number of paths explored. Finally, the last column shows the percentage of paths for which the commercial tool provides the input vector that actually corresponds to the worst delay. Theses results show the inefficiency of not considering the delay variation due to the sensitization vector for complex gates. In many cases the commercial tool simply finds the case for which the complex gate input assignations are easier to justify instead of exploring all the possibilities. As shown in Table 6, the proposed tool identifies correctly some paths that are considered false by the commercial tool. In addition, the computation time required to find the paths is considerably shorter that the one required by the commercial tool.

The algorithm used in this work explores all possible input vectors for each path, unlike the commercial tool that only gives one input vector for each true path. Therefore, the tool proposed identifies correctly the worst delay for each path. The results in the last column of Table 6 show that if the delay variation with the input vector is not considered, the estimation of the worst delay for each path is quite poor, obtaining only a mean value of 40% of paths correctly estimated.

Tables 7, 8 and 9, provide the error in the delay estimation given by the tool developed and the commercial tool, when compared to electrical simulations. These tables contain the mean and maximum delay error for the entire path and for an individual gate. Results show that the delay model used to estimate the gate propagation delay provides more accurate results than the commercial tool considered.

In all the cases studied the polynomial model provides better delay estimations than the look-up table model used by the commercial tool, even using a first order model. The analytical form of the model reduces considerably the computation time leading to faster delay estimations.

VI. Conclusions

We have shown the importance of considering the input vector used to sensitize a complex gate in the delay estimation reporting delay variations up to 15% for a 65nm technology at the gate level. A detailed transistor-level analysis has been included to understand the root cause for such a variation providing results from electrical simulations.

A specific delay tool based on analytical delay description has been presented. It uses a single pass through the circuit to get a list of true paths instead of the traditional two-pass scheme. This allows the tool to account for all sensitization vectors in each complex gate and compute the gate delay accurately. Results from combinational ISCAS circuits show that the delay model considered provides a good estimation of the delays, and demonstrate the ability of the algorithm developed to find all input vectors for a given path, identifying correctly the worst input vector for each path. Such a feature is not supported in the commercial tool that doest not account for multiple sensitization vectors in complex gates and assigns the vector whose justification is simpler. Results for all technologies considered show that the tool developed provides better results than the commercial tool as it reports more paths with a more accurate delay requiring less computation time.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation under grant BES-2006-11943 and by the European Regional Development Fund (ERDF) and the Spanish Ministry of Science and Innovation under Project TEC2008-04501/MIC.

References

[1] I. Keller, K. Ho Tam and V. Kariat, "Challenges in Gate Level Modeling for Delay and SI at 65nm and Below", Design Automation Conference (DAC), pp. 468-473, June 2008.

[2] S.R. Nassif, "Modeling and forecasting of manufacturing variations", Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 145-149, 2001.

[3] D. Blaauw, K. Chopra, A. Srivastava and L. Scheffer, "Statistical Timing Analysis: From Basic Principles to State of the Art", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 589-607, April 2008.

[4] T. El Motassadeq, V. Sarathi, S. Thameem and M. Nijam, "SPICE versus STA tools: Challenges and tips for better correlation", IEEE international SOC Conference (SOCC), pp. 325-328, September 2009.

[5] H. Yaun-chung, C. Hsi-chuan, S. Shangzhi and D.H.C. Du, "Timing Analysis of Combinational Circuits Containing Complex Gates", In proceedings of International Conference on Computer Design: VLSI in Computers and Processors (ICCD), pp. 407-412, October 1998.

[6] C. Ting-Wei, C.Y.R. Chen and Wei-Yu Chen, "An Efficient Gate Delay Model for VLSI Design", 25th International Conference on Computer Design (ICCD), pp. 450-455, October 2007.

[7] J. Xue, D. Al-Khalili and C.N. Rozon, "A Normalized Intrinsic Delay Model of Static CMOS Complex Gates for Deep Submicron Technologies", Notheast Workshop on Circuits ans Systems (NEWCAS), pp. 17-20, June 2004.

[8] Shihheng Tsai and Chung-Yang Haung, "A False-Path Aware Formal Static Timing Analyzer Considering Simultaneous Input Transitions", Design Automation Conference (DAC), pp. 25-30, July 2009.

[9] S. Wen-Tsong and W. Wanalertlak, "An Advanced Cell Polynomial-Base Modeling for Logic Synthesis", IEEE international SOC Conference (SOCC), pp. 393-396, September 2003.

[10] Feng Wang and Shir-Shen Chiang, "Scalable Polynomial Delay Model for Logic and Physical Synthesis", ICDA, August 2000.

[11] RESIST: a recursive test pattern generation algorithm for path delay faults considering various test classes", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, pp. 1550-1562, August 2002.

[12] H. Fijiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms", 25th International Symposium on Fault-Tolerant Computing, pp. 350, June 1995.

[13] S. Bose, P. Agrawal and V.D. Agrawal, " Deriving logic systems for path delay test generation", IEEE Transactions on Computers, pp. 829-846, August 1998.