УДК 621.3.049.771.14

Временной анализ цифровых схем с учетом сложных логических корреляций

А.Л. Глебов, А.А. Миндеева, В.В. Шереметов

Национальный исследовательский университет «МИЭТ»

il0za@yandex.ru

***Аннотация* — Статический временной анализ (СВА) является одним из основных инструментов проектирования цифровых СБИС. В настоящей работе предлагается версия алгоритма временного анализа, значительно более точная по сравнению с известными. Указанное уточнение связано с учетом ложных путей распространения сигнала на основе предварительно вычисленных логических корреляций между сигналами в цифровой схеме.**

***Ключевые слова* — статический временной анализ, ложные пути, логические импликации.**

I. Введение

Проблема учета ложных путей при расчете задержек имеет давнюю историю. Однако до настоящего времени ее нельзя считать решенной. Авторы предлагают оригинальный подход к ее решению, связанный с использованием логических корреляций между сигналами схемы.

Простейшим видом таких корреляций являются простые логические импликации (ПЛИ). Табл. 1 показывает, насколько массовым явлением в цифровых схемах являются ПЛИ.

В последние годы все более актуальными являются задачи повышения эффективности существующих алгоритмов проектирования СБИС. Решение этих задач позволяет значительно повысить качество СБИС без реального изменения их технологии, что может быть охарактеризовано как “виртуальное улучшение технологии”. Все это в полной мере относится к алгоритмам временного анализа.

Временной анализ является важным этапом в процессе временной верификации СБИС. Его задачами являются выявление критических проводящих путей и определение максимально допустимой тактовой частоты для заданной схемы [1, 2].

Для разработки устройств, проектируемых с использованием современных технологий, статический временной анализ (СВА) оказывается единственным методом, пригодным для временной верификации, т.к. способен за приемлемое время справиться со схемами очень большого размера [3, 4].

Основу СВА составляет алгоритм обработки ориентированного взвешенного графа, соответствующего некоторому комбинационному блоку синхронной СБИС. Цель анализа – выявление самых длинных и самых коротких путей от первичных входов блока до его первичных выходов. Результатом работы программы СВА обычно является некоторое заданное число наиболее критических проводящих путей в схеме [5].

Существующие программы СВА исключают из рассмотрения логическую структуру схемы. Поэтому среди отмеченных критических путей могут существовать и ложные проводящие пути, т.е. пути, которые не реализуются в процессе нормального функционирования схемы ни при одной заданной последовательности входных сигналов. Для обнаружения ложных путей в данной работе предлагается использовать логические импликации [6, 7], которые представляют собой подмножество логических ограничений в схеме. При этом в отличие от работы [8] мы используем не только ПЛИ, но также и более сложные импликации (тройные или 3-ЛИ). Это позволяет существенно увеличить точность результатов работы программы статического анализа.

Данная работа организована следующим образом. В разделе 2 дано краткое описание традиционного подхода к СВА. Раздел 3 описывает предлагаемый метод обнаружения ложных путей распространения сигнала (включая результаты численных экспериментов), а в разделе 4 приведены выводы.

II. Традиционный подход к СВА

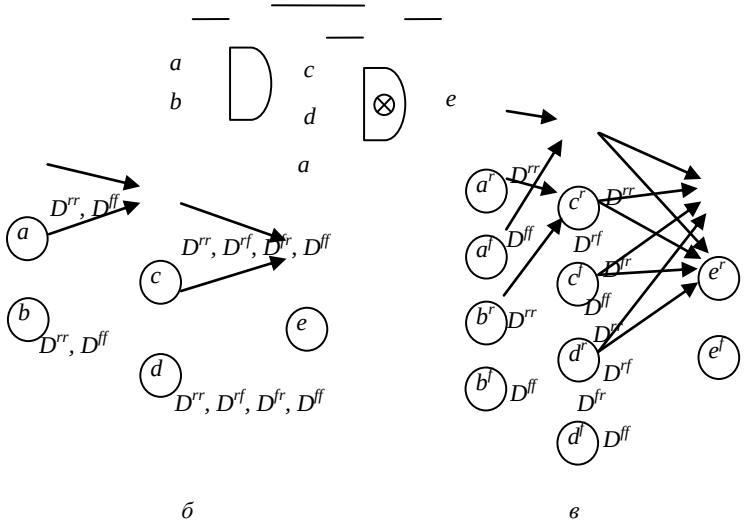
Традиционное применение СВА – это оперативная оценка быстродействия цифровой схемы. Рассмотрим цифровую комбинационную схему, которая является частью синхронной СБИС. Предположим, что нам необходимо оценить наихудшее быстродействие всей схемы, т.е. максимально возможное время прибытия сигнала (latest arrival time (LAT)) на один из первичных выходов схемы.

Для проведения СВА необходимо отобразить схему на временной граф. Этот граф содержит узел для каждого узла схемы и ребро для каждой пары (вход вентиля, выход вентиля). Для каждого узла временного графа возможны переключения двух типов: 0 -> 1 (rise) и 1 -> 0 (fall). Каждому ребру приписаны до четырех различных задержек, соответствующих парам (rise,rise), (rise,fall), (fall,rise), (fall,fall) (см. рис. 1.б).

Таблица 1

Количество простых логических импликаций в логических схемах

Схема	c432	c1355	cla1	cnt_0	cnt_1	cnt_ones	cnt_zeros	testckt
#узлов	248	559	333	83	87	97	99	474
#ПЛИ	20210	32802	5672	1466	1516	2248	2098	86444
#ПЛИ/ пару	0.33	0.10	0.05	0.21	0.20	0.24	0.21	0.38



б *в*

Рис. 1. Пример схемы (*а*), ее временной граф (*б*), модифицированный временной граф (*в*)

Для того чтобы сделать более удобным временной анализ и особенно – перечисление околокритических путей, можно модифицировать временной граф следующим образом. Расщепим каждый узел n на два узла n^{rise}, n^{fall}, соответствующие каждый своему типу переключения. В результате каждое ребро (n_i, n_j) временного графа порождает до четырех ребер. Ребро в модифицированном временном графе соответствует единственной паре переключений с единственной приписанной к нему задержкой (см. рис.1в).

Предположим, что первичные входы схемы переключаются в момент времени t = 0. В этом случае СВА заключается в распространении по узлам схемы (в направлении от первичных входов к первичным выходам) максимальных времен распространения сигнала.

Будем считать, что для каждого вентиля схемы и для каждой пары вход-выход вентиля (i – индекс входа, j – индекс выхода) известны задержки gate.max_delay[i][j].

Результатом временного анализа для каждого из узлов схемы является поле node.max_AT структуры node, содержащей информацию об узле.

Алгоритм описан здесь в сильно упрощенном виде. В современных программах СВА учитываются также направления переключения сигналов, зависимость задержки вентиля от входного фронта и нагрузки на выходе вентиля. Оцениваются не только максимально возможное время прибытия сигнала, но и минимально возможное время прибытия сигнала. Рассматриваются не только комбинационные участки схем, но и блоки памяти. В расчетах также участвуют RC-цепочки на узлах схемы и.т.д.

По данному графу и известным LAT для каждого из узлов схемы легко восстановить самый длинный критический путь. Однако несмотря на высокую эффективность алгоритмов нахождения критического пути, разработчикам СБИС часто оказывается недостаточно информации только об одном критическом пути, чтобы исправить все временные нарушения в схеме. Для получения большего количества временной информации о схеме были разработаны алгоритмы перечисления путей [4, 5]. Несмотря на то, что такие алгоритмы дают больше информации о временных нарушениях в схеме, они могут выдавать слишком много информации, т.к. в худшем случае количество путей может зависеть экспоненциально от размера схемы. Поэтому в современных средствах временной верификации СБИС используется подход, основанный на выводе K наиболее критических путей в схеме, упорядоченных по возрастанию или убыванию времени задержки сигнала на данном пути [5]. Значение параметра K задает разработчик схемы.

Рассмотрим алгоритм нахождения K максимальных проводящих путей в схеме, основанный на рекурсивном обходе графа решений в глубину. Поиск начинается с первичных выходов (POUT) и идет в обратном направлении к первичным входам (PIN). В

процессе поиска используется критерий, который позволяет обрывать поиск на некоторых итерациях алгоритма. Таким образом, лишь небольшая часть проводящих путей в схеме обходится полностью.

Положим, что позднее требуемое время прибытия сигнала для всех POUT равно LRT(POUT) = 0. Для узла схемы N зададим следующие обозначения: LAT(N) = (максимальная длина пути от PIN до N); LRT(N) = - (максимальная длина пути от N до POUT); Lslack(N) = LRT(N) - LAT(N) = - (максимальная длина пути, проходящего через N).

Пусть Pn – это некоторый проводящий путь из N к POUT, тогда обозначим: pRT(N, Pn) = - (длина пути Pn); pSlack(N, Pn) = pRT(N, Pn) – LAT(N) = - (максимальная длина пути, содержащего Pn).

Полным путем назовем путь от одного из первичных входов PIN до одного из первичных выходов POUT. Алгоритм перебирает конечные части Pn полных проводящий путей и проверяет случаи, когда Pn оказывается полным путем. Пусть minDelay – это длина самого короткого из K самых длинных путей, найденных к данному моменту времени. Дальнейший поиск путей, содержащих Pn, прекращается, если –pSlack(N, Pn) < minDelay, т.е. длина любого пути, содержащего Pn, меньше, чем minDelay.

III. Метод обнаружения ложных путей распространения сигнала

Определение 1. Пусть *M* – узел временного графа. *Conj(M)* – узел временного графа (узел, сопряженный *(conjugated)* с *M*), который получается из того же узла схемы *N* и составляет с *M* пару *(N_rise, N_fall).*

Определение 2. Пусть *M*1 и *M*2 – узлы временного графа. Мы говорим, что существует поздняя ПЛИ (пПЛИ) *M*1 -> *M*2, если существует такое логическое ограничение между соответствующими узлами схемы *N*1 и *N*2, что если *N*1 = *v*1п, то *N*2 = *v*2п, где *v*1п и *v*2п – логические значения узлов *M*1 и *M*2 после переключения.

Пусть L_1, …, L_n – узлы, лежащие на некотором полном пути *P* во временном графе для некоторой схемы. Пусть *M* и *N* – некоторые два узла на этом пути. Сформулируем условия, при которых путь является ложным из-за наличия ПЛИ между *N, M* и некоторым другим узлом *S* (рис. 2).

Проводящий путь *P* состоит из трех подпутей: *Ppin,n, Pn,m, Pm,pout. Len(P) = Len(Ppin,n) + Len(Pn,m) + Len(Pm,pout)*, где *Len(P)* – длина пути *P*.

Пусть выполняется следующий ***набор условий*:**

1) существует ребро (*S, M*), не лежащее на пути *P*, т.е. *S* – это вход сбоку на путь *P*;

2) существуют пПЛИ *S* -> *M* и *N* -> *S*;

$$Len(Ppin,M) > LAT(S) + Ds,m, \quad (1)$$

где *Ds,m* – это задержка ребра временного графа (*S*, *M*).

В этом случае сигнал (переключение), идущий по пути *P*, не может быть на узле *M* позже, чем через $t = LAT(S) + Ds,m$. С другой стороны, время прибытия сигнала к узлу *M*, идущему по пути *P*, точно равно *Len(Ppin,m).* Таким образом, *P* является ложным путем (сигнал, идущий по этому пути, никогда не дойдет до *POUT*.

Теперь рассмотрим ***другой набор условий*** (рис. 2):

1) существует ребро во временном графе *(T, Conj(M))*;

2) существуют пПЛИ(*T, Conj(M)*) и пПЛИ(*N, T*);

$$Len(Ppi, M) > LAT(T) + Dt,Conj(m), \quad (2)$$

где *Dt,Conj(m)* – задержка ребра временного графа *(T, Conj(M)*).

В этом случае сигнал (переключение), идущий по пути *P*, не может быть на узле *M* позже, чем через $t = LAT(T) + Dt,Conj(m)$. С другой стороны, время прибытия сигнала к узлу *M*, идущему по пути *P*, точно равно *Len(Ppin,m).* Таким образом, *P* является ложным путем.

Имеется также ряд других критериев ложности пути, подобных критериям (1) и (2).

Табл. 2 показывает, насколько значительным может быть уточнение результатов временного анализа при учете ПЛИ.

Описанные выше методы и алгоритмы были рассмотрены в работе [8]. Рассмотрим теперь впервые предлагаемый нами в данной работе метод, основанный на использовании 3-ЛИ. В отличие от случая ПЛИ, хранить в памяти все 3-ЛИ нереально ввиду их огромного количества. Если всю рассматриваемую схему разбить на двухвходовые вентили AND и OR (с возможной инверсией сигнала на одном или обоих входах), то каждый такой вентиль имеет между своими выводами две ПЛИ и одну 3-ЛИ (назовем ее первичной 3-ЛИ).

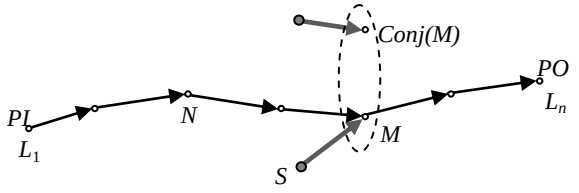


Рис. 2. Критерий обнаружения ложных путей

Таблица 2

Уточнение результатов временного анализа при учете ПЛИ

Схема	Max delay w/o SLI (ns)	Max delay with SLI (ns)	Max delay with SLI and 3-LI (ns)	Reduction %	# top false paths
c17	0.172	0.172	0.172	0	0
c432	2.05	1.14	0.89	56.7	>1000
c499	1.04	0.98	0.95	8.8	>1000
c1355	1.76	1.75	1.75	0.55	>1000
c1908	1.91	1.91	1.91	0	0
c2670	1.85	1.52	1.40	24.5	>1000
c3540	2.36	2.32	2.32	1.6	~500
c5315	2.49	2.48	2.47	0.88	~55
cnt_0	1.24	1.24	1.24	0	0
cnt_1	1.29	1.29	1.29	0	0
cnt_ones	1.32	1.31	1.31	0.65	1
cnt_z	1.20	1.20	1.20	0	0

Например, вентиль AND2 с входами a, b и выходом x порождает первичную 3-ЛИ: a, b -> x. Исходя из этой первичной 3-ЛИ и списков ПЛИ в узлах a, b, x, можно породить множество 3-ЛИ между различными узлами схемы.

Пусть L_1, …, L_n – узлы, лежащие на некотором полном пути *P* во временном графе для некоторой схемы. Пусть *M*, *N1*, *N2* – некоторые три узла на этом пути. Сформулируем условия, при которых путь является ложным из-за наличия 3-ЛИ между *N1*, *N2* и некоторым другим узлом *S*, а также ПЛИ между *S* и *M* (рис. 2).

Проводящий путь *P* состоит из четырех подпутей: *Ppin,n1, Pn1,n2, Pn2,m, Pm,pout. Len(P) = Len(Ppin,n1) + Len(Pn1,n2) + Len(Pn2,m) + Len(Pm,pout)*, где *Len(P)* – длина пути *P*.

Пусть выполняется следующий ***набор условий*:**

1) существует ребро *(S, M)*, не лежащее на пути *P*, т.е. *S* – это вход сбоку на путь *P;*

2) существуют пПЛИ *S -> M* и п3-ЛИ *N1, N2 -> S*;

$$Len(Ppin,M) > LAT(S) + Ds,m, \quad (3)$$

где *Ds,m* – это задержка ребра временного графа (*S*, *M*).

В этом случае сигнал (переключение), идущий по пути *P*, не может быть на узле *M* позже, чем через $t = LAT(S) + Ds,m$. С другой стороны, время прибытия сигнала к узлу *M*, идущему по пути *P*, точно равно *Len(Ppin,m).* Таким образом, *P* является ложным путем (сигнал, идущий по этому пути, никогда не дойдет до *POUT*.

Теперь рассмотрим ***другой набор условий*** (рис. 2):

1) существует ребро во временном графе *(T, Conj(M))*;

2) существуют пПЛИ *T -> Conj(M)* и п3-ЛИ

N1, N2 -> T;

$$Len(Ppi, M) > LAT(T) + Dt,Conj(m), \quad (4)$$

где *Dt,Conj(m)* – задержка ребра временного графа *(T, Conj(M)*).

В этом случае сигнал (переключение), идущий по пути *P*, не может быть на узле *M* позже, чем через $t = LAT(T) + Dt,Conj(m)$. С другой стороны, время прибытия сигнала к узлу *M*, идущему по пути *P*, точно равно *Len(Ppin,m).* Таким образом, *P* является ложным путем.

Последний столбец таблицы 2 показывает, насколько значительным может быть уточнение результатов временного анализа при учете ПЛИ и 3-ЛИ по сравнению с вариантом учета только ПЛИ.

IV. Выводы

В данной работе представлен алгоритм временного анализа цифровых схем, включающий значительные уточнения, вносимые алгоритмом обнаружения ложных путей (среди самых длинных путей в схеме). Критерии ложности путей основаны на использовании ПЛИ и (впервые) 3-ЛИ, т.е. сложных корреляций между сигналами в схеме.

Учет импликаций более сложных, чем 3-ЛИ, не имеет смысла, поскольку эти импликации не являются независимыми, но могут быть получены как комбинации различных 3-ЛИ и ПЛИ. В то же время набор критериев ложности пути, описанных в данной

работе, не является исчерпывающим и может быть значительно расширен [8].

ЛИТЕРАТУРА

[1] Amon T., Borriello G. An approach to symbolic timing verification. // DAC, 1992. - P. 410-412.

[2] Gladstone B. Accurate timing analysis holds the key to performance in today's system designs. // EDA. - 1993.

[3] Hitchcock R.B. Timing verification and the Timing analysis Program. // DAC, 1982. - P. 594-604.

[4] Reddi R., Chen C. Hierarchical Timing Verification System. // Computer Aided Design. - Vol. 18. - 9, November. - P. 467-477.

[5] Yen S., Du D., Ghanta S. Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis. // DAC, 1989. - P. 649-654.

[6] Glebov A., Gavrilov S., Blaauw D., et. al. False-Noise Analysis Using Logic Implications. // ICCAD, 2001. - P.515.

[7] Glebov A., Gavrilov S., Blaauw D., Zolotov V. False-noise analysis using logic implications. // ACM Trans. on Design Automation of Electronic Systems (TODAES). - 2002. V.7. P.474.

[8] Соловьев Р.А., Глебов А.Л., Гаврилов С.В. Обнаружение ложных путей в цифровых схемах на основе логических импликаций. – Изв. ВУЗов, Электроника, 2007, №2, с.78-84.

Timing analysis of digital circuits basing on logic correlations

A.L.Glebov, A.A.Mindeeva, V.V.Sheremetov

National Research University «MIET»

il0za@yandex.ru

***Keywords* — Static timing analysis, false paths, logic implications.**

ABSTRACT

Timing analysis is an important stage in the process of VLSI timing verification. Its aim is finding signal critical paths and calculation of maximum clock frequency for the given circuit [1, 2].

There exist multiple methods of circuit timing analysis that may be divided into two types: dynamic and static. Dynamic methods are based on simulation of the circuit operation, for the given input testvector sequence. Various methods of this type differ only in degree of adequacy of the model under use. On the other hand, static timing analyzer is based on different principle. Instead of simulation, it considers all possible signal paths in the circuit. This requires larger computer memory, but is orders of magnitude faster than dynamic analysis. In modern VLSI design, static timing analysis (STA) is the only method suitable for timing verification, since it is able to cope with circuits of very large size in acceptable time [3, 4].

Within STA algorithm, the directed weighted graph is processed that corresponds to some combinational block of synchronous VLSI circuit. The aim of analysis is to detect the longest and shortest paths from primary inputs to primary outputs of the block. The result of STA algorithm is usually some specified number of most critical paths in the circuit [5].

Existing STA programs do not consider the logic structure of the circuit. Therefore, among marked critical paths the false paths can exist, i.e. paths that do not actually propagate switching for any transitions of primary inputs. For detecting the false paths we propose here to use the logic implications, which compose the subset of all logic constraints in the circuit [6, 7]. This allows to improve the accuracy of STA results substantially.

It should be noted that, unlike the work [8], in this paper we consider not only simple logic implications (SLI), but also complex ones, i.e. triple logic implications (3-LI).

REFERENCES

[1] Amon T., Borriello G. An approach to symbolic timing verification. // DAC, 1992. - P. 410-412.

[2] Gladstone B. Accurate timing analysis holds the key to performance in today's system designs. // EDA. - 1993.

[3] Hitchcock R.B. Timing verification and the Timing analysis Program. // DAC, 1982. - P. 594-604.

[4] Reddi R., Chen C. Hierarchical Timing Verification System. // Computer Aided Design. - Vol. 18. - 9, November. - P. 467-477.

[5] Yen S., Du D., Ghanta S. Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis. // DAC, 1989. - P. 649-654.

[6] Glebov A., Gavrilov S., Blaauw D., et. al. False-Noise Analysis Using Logic Implications. // ICCAD, 2001. - P.515.

[7] Glebov A., Gavrilov S., Blaauw D., Zolotov V. False-noise analysis using logic implications. // ACM Trans. on Design Automation of Electronic Systems (TODAES). - 2002. V.7. P.474.

[8] Soloviev R.A., Glebov A.L., Gavrilov S.V. Detecting false paths in digital circuits basing on logic implications. // Izvestiia vuzov, Electronics, 2007, №2, P. 78-84.