На правах рукописи

Князев Николай Александрович

Статическое моделирование временных характеристик работы СБИС с использованием вычислительных систем с общей памятью

Специальность 05.13.18

Математическое моделирование, численные методы и комплексы программ

АВТОРЕФЕРАТ

Диссертации

на соискание ученой степени

кандидата физико-математических наук

Москва

2013

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институт прикладной математики им. М.В. Келдыша РАН.

Научный руководитель:	доктор физико-математических наук ФГБУН Институт прикладной математики им. М.В. Келдыша РАН, начальник сектора Якобовский Михаил Владимирович
Научный консультант:	кандидат физико-математических наук ЗАО «Интел А/О», старший научный сотрудник Малинаускас Костас Костович
Официальные оппоненты:	доктор технических наук, профессор Нижегородский государственный университет им Н. И. Лобачевского, декан факультета Вычислительной Математики и Кибернетики Гергель Виктор Павлович
	Кандидат физико-математических наук Федеральное государственное бюджетное учреждение науки Институт Прикладной Математики им. М.В. Келдыша РАН, заведующий сектором. Болдарев Алексей Сергеевич
Ведущая организация:	Федеральное государственное бюджетное учреждение науки Научно-исследовательский институт системных исследований РАН

Защита диссертации состоится 17 октября 2013 г.в 14.00 часов на заседании диссертационного совета Д 002.024.03 в Федеральном государственном бюджетном учреждении науки Российской академии наук Институте прикладной математики им. М.В. Келдыша по адресу: 125047, г.Москва, Миусская пл., д.4А.

С диссертацией можно ознакомиться в библиотеке ИПМ им. М.В. Келдыша РАН

Автореферат разослан сентября 2013 г.

Учёный секретарь

диссертационного совета Д 002.024.03, Н.В.Змитренко
доктор физико-математических наук

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы.

В диссертации рассматривается возможность применения многоядерных вычислительных систем (систем с общей памятью) к задаче статического моделирования временных характеристик схемы (Статического временного анализа, СВА). СВА – распространенный подход для оценки быстродействия синхронных цифровых схем и определения максимально-допустимой тактовой частоты работы схемы. Полное моделирование переходных процессов в современных СБИС слишком трудоемко, поэтому упрощенное моделирование методами СВА получило широкое распространение на практике. Поиск критических путей распространения сигнала является одним из ключевых этапов статического временного анализа и временной верификации цифровых схем. Для содержащих несколько миллионов транзисторов СБИС время работы алгоритмов временного анализа может занимать более суток, что существенно осложняет разработку схем. В некоторых случаях для сокращения времени работы можно использовать инкрементальные методы, однако, они применимы к ограниченному классу задач, не всегда работают достаточно быстро. Многопоточная параллелизация существующих методов потенциально может существенно ускорить время работы. Однако, когда разрабатывались алгоритмы временного анализа, многопоточного исполнения обычно не предполагалось. Разработка эффективно параллелизуемых методов СВА и их масштабирование для работы со сверхбольшими интегральными схемами являются на сегодняшний день актуальной проблемой статического моделирования временных характеристик цифровых схем.

Цели и задачи диссертационной работы

1. Разработка параллельного метода, построения критических путей

в задаче статического моделирования временных характеристик схемы для систем с общей памятью

2. Анализ масштабируемости распространенных методов построения критических путей в задачи статического моделирования временных характеристик схемы для реальных промышленных схем
3. Реализация разработанного метода в рамках системы автоматического проектирование и проведение экспериментов на реальных промышленных схемах

Научная новизна

Научная новизна диссертационной работы состоит в новом подходе к параллелизации схем содержащих комбинационную логику. В диссертационной работе предложен параллельный метод для статического моделирования временных характеристик схем с последовательной логикой, позволяющий достичь ускорения до 14 раз на 16 потоках. В работе впервые проведена теоритическая оценка масштабируемости методов статического моделирования временных характеристик схем и получены верхние оценки масштабируемости метода для реальных схем на неограниченном числе потоков.

Практическая значимость

Результаты диссертационной работы были внедрены в экспериментальную систему автоматического проектирования, разрабатываемую в ЗАО «Интел А/О». Реализация метода была использована для оценки масштабируемости параллельных методов СВА на используемых в промышленности схемах.

Апробация работы

Результаты работы докладывались и обсуждались на следующих конференциях:

1. Международная конференция «Научный сервис в сети Интернет: экзафлопсное будущее», г. Новороссийск, 2011.

2. Международная конференция-школа «Современные проблемы прикладной математики и информатики», г. Дубна 2012
3. Международная конференция «Нано- и Микроэлектронные системы», г. Москва, 2012.

На международной, проводящийся раз в два года конференции «Нано и Микроэлектронные системы», работа была отмечена как лучшая статья по тематике методы систем автоматического проектирования

Достоверность представленных результатов

Достоверность представленных результатов подтверждается экспериментами на более чем 80 промышленных схемах и сравнением результатов с заранее известными данными анализа данных схем используемыми в промышленности инструментами.

Реализация результатов

Предложенный в работе алгоритм был внедрен в экспериментальную систему моделирования и отображения временных характеристик и структуры схемы.

Личный вклад автора

Автором был разработан и реализован предложенный в работе параллельный алгоритм, проведены эксперименты и анализ масштабируемости метода.

Объем и структура диссертации

Диссертация состоит из 3х глав, введения, заключения и списка литературы. Диссертация содержит 80 страниц, 35 рисунков. В Списке литературы присутствует 36 наименований.

Также в исследуемой области параллельных технологий автором работы выпущена монография:

Князев Н. Сальников А "Планирование запуска программ на суперкомпьютерах" Монография/ LAP LAMBERT Academic Publishing. 2011 ISBN-13: 978-3-8433-0712-3 ISBN-10:3843307121 EAN: 9783843307123

СОДЕРЖАНИЕ РАБОТЫ

Диссертация состоит из трех глав, введения, заключения, списка литературы и двух приложений.

Во **введении** обоснована актуальность рассматриваемых в работе проблем, сформулированы основные цели диссертации и дано ее краткое содержание по главам. Приведено общее описание используемых в статическом моделировании временных характеристик (Статическом временном анализе, СВА) математических моделей. Проведен обзор существующих методов и подходов к поиску критических путей в статическом моделировании временных характеристик СБИС.

В **первой главе** приведено подробное описание используемой модели, описывается предлагаемый подход к параллелизации методов, который сравнивается с существующими методами.

Основная задача статического моделирования временных характеристик (или статического временного анализа, СВА) схемы – определить предельную тактовую частоту, на которой будет работать схема и построить критические пути прохождениях сигнала[1]. В статическом временном анализе задержки вычисляются путем упрощенного моделирования отдельных элементов схемы и их соединений с использованием сценариев худшего случая распространения сигналов. В качестве результата СВА принято рассматривать набор критических путей во временном графе[2], определяющих быстродействие схемы. В реализациях методов СВА рассматривают несколько этапов – загрузка модели схемы в оперативную память (построение модели на основе описания схемы), расчет схемы и вывод результата. В данной работе рассматривается второй этап – расчет схемы. Время, затрачиваемое инструментами на каждый из этапов, сильно зависит от конкретной реализации и предъявляемых требований

[1] **Гаврилов С., Стемпковский А., Глебов А.** Методы логического и логико-временного анализа цифровых СБИС // Москва, Наука. 2007 г.

[2] **Hassoun S., Sasao T.** Logic synthesis and verification // Springer pp. 373-401. 2002 г.

(способ хранения схемы, принцип вывода критических путей и т.д.).

В качестве **математической модели** для поиска критических проводящих путей используется построение графа задержек распространения изменений уровня сигнала по схеме. В работе используются понятия:

Событие – изменение уровня сигнала (логического «0» или «1») в узле схемы (на входе или выходе логического элемента). Событие *e* характеризуется шестью характеристиками *t(e), v(e), s(e), f(e), p(e), ref(e)*

Задержка события t(e) – время наступления события относительно некоторого начала отсчета.

Узел схемы v(e) – узел схемы, на котором наступает событие.

Фронт переключения события (Тип события) f(e) – направление переключения уровня сигнала (с 1 на 0 или с 0 на 1)

Время фронта переключения s(e) – Время, за которое изменяется уровень сигнала.

Родительское событие p(e) – событие на соседнем узле, порождающие данное событие

Событие генератора тактовых импульсов ref(e) – синхронизирующее событие, относительно которого считается задержка.

Временной граф схемы – направленный граф(V,G), где в качестве множества вершин G выступают узлы схемы, а множество направленных дуг V соответствуют причинно-следственным связям между возможными событиями в соседних узлах. В процессе СВА на временном графе происходит распространение событий от входов к выходам схеме.

(Временной) путь – последовательность событий, где очередное событие порождается предыдущим в смежном узле, т.е. такая последовательность событий $e_1 \dots e_n$,где $\forall i, i > 1: p(e_i) = e_{i-1}$

Задержкой пути, $e_1 \dots e_n$ заканчивающегося событием e_n, называется задержка этого события $t(e_n)$.

Критический путь – это такой путь $e_1 \dots e_n$, что $\forall$ события e на временном графе выполняется следствие:

$v(e) = v(e_п)$ $f(e) = f(e_п) \Rightarrow t(e) \le t(e_п)$

и

Запас – разность между фактической задержкой события (пути) и задержкой, предельно допустимой для корректной работы схемы.

Отрицательный путь – путь, имеющий отрицательный запас.

Особое значение в интегральной схеме имеют элементы памяти, триггеры. Именно через эти элементы поступают сигналы синхронизации. В корректно спроектированной комбинационной схеме отсутствуют замкнутые временные пути. Однако, в случае наличия триггеров на временном графе схемы могут появиться циклы. Для статического временного анализа есть важное различие между триггером с динамическим и триггером со статическим управлением

Триггером с динамическим управлением (flip-flop, с управлением по фронту) В данном типе триггеров событие передается в момент возникновения управляющего события синхронизирующей цепи, поэтому выходное событие порождается событием синхронизации, и во временном графе отсутствует дуга от входа данных к выходу.

Триггер со статическим управлением (latch, или с управлением по уровню) – триггер, в котором условием передачи сигнала служит состояние входа синхронизации (0 или 1), определяющее «прозрачность» триггера. Триггер прозрачен (открыт) между открывающим и закрывающим событиями синхронизирующей цепи, в остальное время триггер непрозрачен (закрыт). Обозначим время распространения события со входа данных на выход как *D*, а время распространения события синхронизации на выход – как *Dc*, опишем событие на выходе $e_{вых}$ через входящее событие $e_{вх}$ и события синхронизации $e_{откр}$ и $e_{закр}$

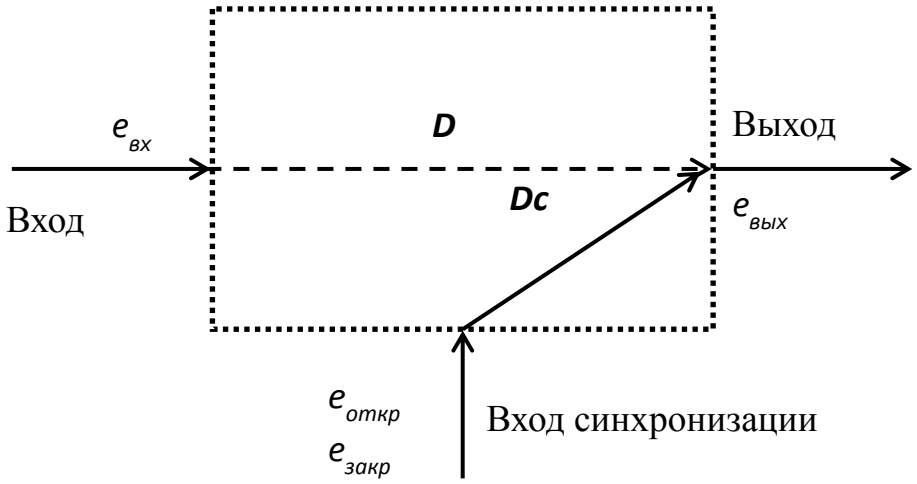


Рисунок 1 Триггер, тактируемый уровнем

Рассмотрим различные варианты соотношения задержек событий

1. Триггер прозрачен : $d(e_{закр})>d(e_{вх}) >d(e_{откр})$
   i) $d(e_{вых}) = d(e_{вх}) + D + d(ref(e_{вх})) - d(ref(e_{откр}))$ (фазовый сдвиг)
   ii) Изменение события синхронизации, относительно которого рассчитывается задержка триггера называется фазовым сдвигом: $ref(e_{вых}) = ref(e_{откр})$;
   iii) $p(e_{вых}) = e_{вх}$
2. Триггер закрыт: $d(e_{вх}) <d(e_{откр})$
   i) $d(e_{вых}) = d(e_{откр}) + Dc$
   ii) $ref(e_{вых}) = ref(e_{откр})$;
   iii) $p(e_{вых}) = e_{откр}$ (распространяется $e_{откр}$)
3. Восстановление сигнала: $d(e_{вх}) >d(e_{закр})$. В случае если событие произошло позже закрывающего события синхронизирующей цепи, то в рамках модели допускается «восстановление» события, и считается что событие на входе произошло в одновременно с закрывающим событием. Однако данная ситуация, как правило, является нарушением и о ней необходимо сообщить проектировщику схемы
   i) $d(e_{вых}) = d(e_{закр}) + Dc$
   ii) $ref(e_{вых}) = ref(e_{откр})$;
   iii) $p(e_{вых}) = e_{вх}$ (распространяется $e_{вх}$ с задержкой $d(e_{закр}) + Dc$)

Пусть события $e_{вых1}$,$e_{вых2}$ произошли на одном узле и входят в один путь, и при этом событие $e_{вых1}$ произошло раньше. Такой путь называется *критическим циклом*, если $d(e_{вых2}) - d(ref(e_{вых2})) > d(e_{вых1}) - d(ref(e_{вых1}))$

Критический цикл является ошибкой проектирования, т.к. задержка на нем накапливается, что приводит к сбою

В качестве входных данных для СВА выступает набор событий на входах схемы, при распространении событий используется *прореживание*: в каждом узле схемы выбираются события с наихудшей (наибольшей или наименьшей, в зависимости от типа анализа) задержкой, и дальше по схеме распространяются только эти события.

Далее в главе рассматриваются предлагаемые автором методы поиска критических путей на графе схеме, эти методы рассчитаны на параллельную реализацию для систем с общей памятью (описание реализации приводится в следующей главе). Данные методы используют разные известные методики теории графов, такие как алгоритм Йена[3] или параллельный поиск в ширину[4]. В Главе 1 доказывается корректность предложенных алгоритмов, применительно к задаче и модели статического временного анализа.

В параллельных реализациях СВА для схем, состоящих из комбинационной логики и не содержащих временных циклов используется разделение вершин по уровням. При этом синхронизация различных потоков происходит при переходе с одного уровня на другой. Вычисление нового уровня не начинается, пока не рассчитаны все вершины предыдущего. Такой подход порождает неравномерную загрузку вычислительных мощностей. В данной статье для таких схем предлагается «асинхронная» параллелизация, когда элемент начинает рассчитываться, как только для него готовы данные. Для обеспечения «асинхронной» параллелизации каждому элементу ставится в соответствие счетчик, соответствующий числу входов, на которых еще не

[3] **Yen S., Du D., Ghanta S.** Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis // In Proc. Design Automation Conference (DAC). 1989 г.

[4] **Schultze P., Korf R. E.** Large-Scale Parallel Breadth-First Search// Menlo Park, CA; Cambridge, MA; London. 1994 г.

готовы события сигнала. Далее описывается каждый этап алгоритма

На этапе **инициализации счетчиков** необходимо на всех элементах схемы инициализировать счетчики равными числу входов элемента. Здесь может использоваться поиск в ширину из начальных элементов.

На этапе **расчета элементов** производится непосредственно расчет схемы, т.е. распространение событий от входов схемы к ее выходам и расчет критических путей. Физический расчет элементов выполняется при помощи библиотеки RICE[5] и остается за пределами данной работы. Этап расчета элементов выглядит следующим образом:

Вход: граф схемы с установленными счетчиками

Выход: рассчитанные задержки событий на вершинах графа, с учетом прореживания

1. Поместить в очередь начальные элементы.
2. Выбрать элемент из очереди.
3. Рассчитать элемент, распространить события со входов элемента на его выходы.
4. Осуществить прореживание сигналов.
5. Рассчитать межсоединения, распространить события с выходов элемента на входы элементов, соединенных с выходами данного.
6. Для элементов, входы которых соединены с выходами данного, уменьшить счетчик на 1.
7. Если счетчик какого-либо из элементов стал равен 0, то добавить его в очередь.
8. Если очередь пуста, закончить алгоритм, иначе перейти на пункт 2.

Рисунок 2 Алгоритм СВА для схемы без циклов

В данном алгоритме рассчитывать и добавлять элементы в очередь

[5] **Pillage L.T., Ratzlaff C. L.** RICE rapid interconnect circuit evaluation // 28th ACM/ IEEE Design Automation Conference. 1994 г.

можно в параллельном режиме. Все затратные по времени операции алгоритма (установка счетчиков и расчет элементов) есть возможность выполнять параллельно, что делает минимальной последовательную часть (только функции инициализации), и что согласно закону Амдала позволяет потенциально достичь хорошего ускорения.

В случае, если у элемента несколько выходов, то обработку выходов можно выполнять параллельно (использовать параллельный цикл по выходам в пунктах 3 и 5 алгоритма СВА для схемы без циклов). Таким образом, существует возможность встроить в алгоритм «низкоуровневый» параллелизм на этапе анализа одного элемента. Так как распространение событий через выходы можно производить независимо, синхронизации требуется только при создании новых событий на одном узле графа схемы(для этого можно использовать потокобезопасные контейнеры). Это позволит ускорить анализ схем, содержащих элементы с большим количеством выходов. В данной главе также показывается **корректность алгоритма.** Для этого последовательно доказываются следующие утверждения:

Лемма 1

Если в процессе выполнения алгоритма для комбинационных схем этап распространения событий не выполнен для какого-либо элемента (элемент не был проанализирован). То хотя-бы один из входов этого элемента соединен с элементом, для которого этот этап также не был выполнен.

Лемма 2

Если на путь $v_1 \ldots v_n$ является критическим, то для любого k, $k<n$ путь $v_1 \ldots v_k$ также является критическим.

Лемма 3

Построенные на этапе распространения событий пути на выходах элемента после осуществления прореживания являются критическими путями.

Стоит отметить, что Лемма 3 остается верной, только в случае, когда не учитывается длина фронта переключения сигнала. Событие с «пологим» фронтом переключения и малой задержкой на выходе схемы может

«накопить» задержку больше, чем событие с большей задержкой, но с «крутым» фронтом переключения[6], тем не менее учитывать величину фронта переключения при прореживании сигнала невозможно без расчета путей всех входящих сигналов, что делает а анализ слишком затратным, а прореживание бессмысленным.

Утверждение 1

В процессе выполнения алгоритма для комбинационных схем в корректной схеме (схеме, не содержащей циклов) этап распространения событий выполнится для каждого элемента схемы

Утверждение 2

Корректность алгоритма для комбинационных схем

Пусть $v_1 \ldots v_n$ - критический путь в комбинационной схеме, от входа v_1 к некоторой вершине v_n, тогда в рамках работы предложенного алгоритма будет создан критическая путь $e_1 \ldots e_n$, такой, что $v(e_1) = v_1 \ldots v(e_n) = v_n$.

В данной главе показывается **оценка сложности предлагаемого алгоритма**

Сложность алгоритма складывается из сложностей его составных частей. Пусть количество достижимых элементов в схеме N. Как было сказано ранее, инициализация счетчиков занимает намного меньше времени, чем расчет схемы. Во время второго этапа алгоритма каждый элемент вычисляется только один раз, таким образом, сложность равна $O(N)$.

Далее в первой главе приводится разработанный простейший метод инкрементального пересчета событий в случае увеличении задержки в одном из узлов схемы. После изменения одного элемента в схеме, во временном анализе возникает задача быстрого перерасчета событий на элементах. На практике часто допускают, что структура худших путей не изменилась. В этом случае выполнять выше описанный алгоритм нет необходимости, можно ограничиться перерасчетом уже известных путей сигнала. Перерасчет

[6] **Blaauw S., Zolotov D., Sundareswaran V.** Slope Propagation in Static Timing Analysis // Proceedings of the 2000 IEEE/ACM international conference on CAD. 2000 г.

событий на конусе схемы намного более простая задача, чем полный статический временной анализ. В данном алгоритме также используется поиск в ширину, однако, в отличие от описанного алгоритма СВА на графе без циклов, нет этапа расстановки счетчиков. Этот этап здесь не нужен, т.к. граф критических путей представляет собой дерево (у каждого события только одно родительское событие) и синхронизации для выполнения прореживания не требуется. В данном алгоритме обрабатывать и добавлять элементы в очередь можно в многопоточном режиме, что делает возможным реализацию алгоритма для многоядерных ЭВМ.

Основной метод представленный в работе, это параллельный метод анализа для схем, содержащих триггеры тактируемые уровнем. В методе используется свойство схем, что любой цикл во временном графе содержит хотя бы один триггер, тактируемый уровнем. Таким образом, запретив распространение событий на всех триггерах, временной граф схемы приводится к ациклическому. Статический временной анализ является итеративным процессом с чередующимися этапами анализа ациклической части графа без триггеров и анализа только триггеров. Алгоритм выглядит следующим образом:

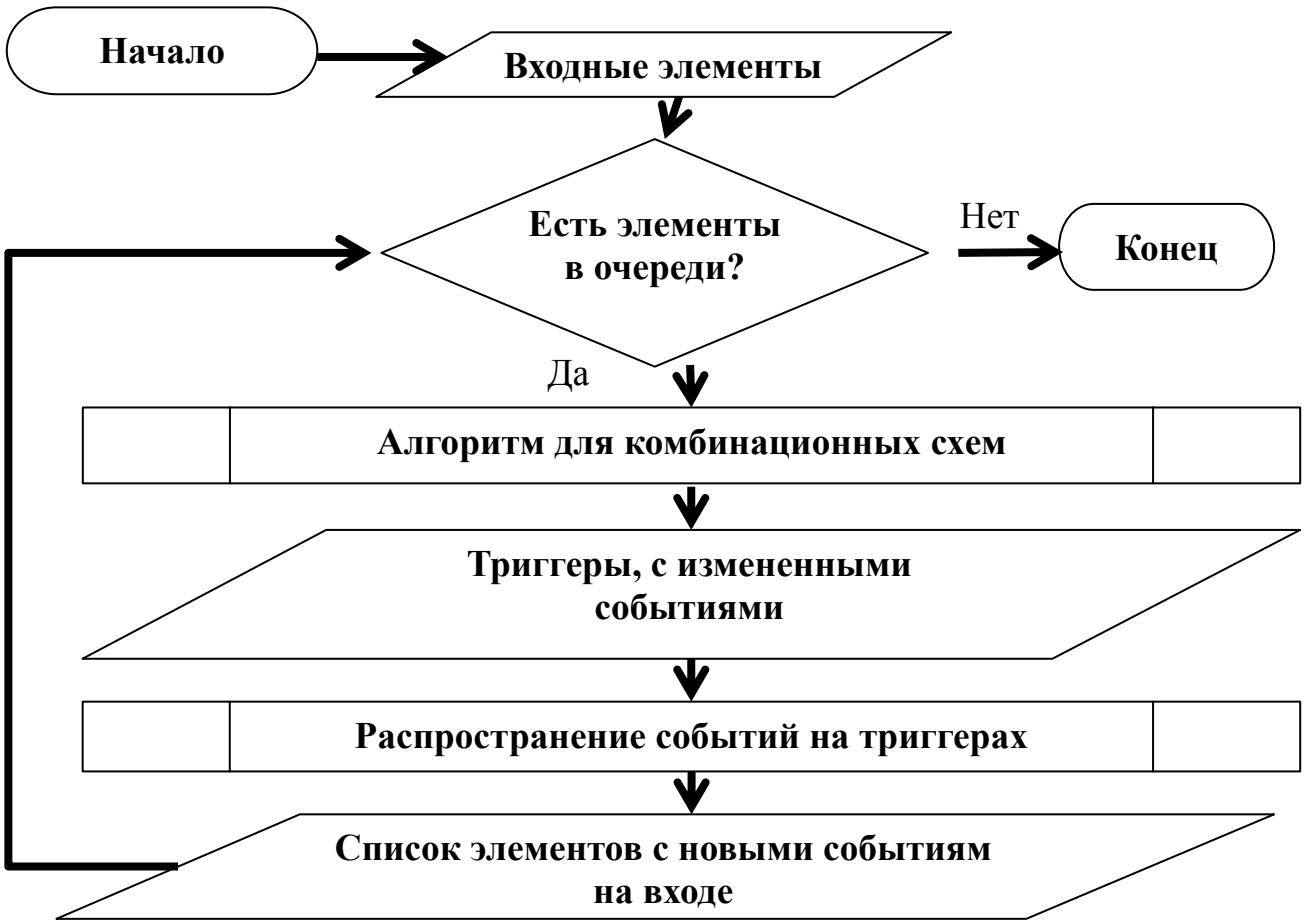


Рисунок 3 Алгоритм поиска критических путей в задаче статического моделирования временных характеристик схем с триггерами

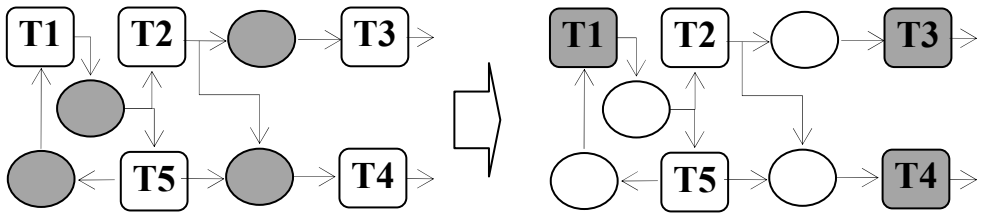
Для вычисления критических путей в ациклическом графе используется описанный выше алгоритм для комбинационных схем. После его выполнения анализируются триггеры, на входах которых изменились худшие задержки, события распространяются через эти триггеры. Далее элементы комбинационной логики, на входах которых задержки изменились, используются в качестве начальных для повторного выполнения алгоритма для комбинационной логики. Итерации повторяются, пока задержки событий не стабилизируются. Обнаружение и разрыв критических циклов будут рассмотрены ниже.

В данном алгоритме предусмотрено **обнаружение циклов,** которое происходит на этапе прореживания событий. Для этого просматривается назад путь, оканчивающийся новым событием. Если он посещает текущий узел повторно, то обнаружен критический цикл. Для разрыва цикла первый

встреченный триггер помечается при обратном просмотре как запрещенный. События на запрещенных триггерах не распространяются.

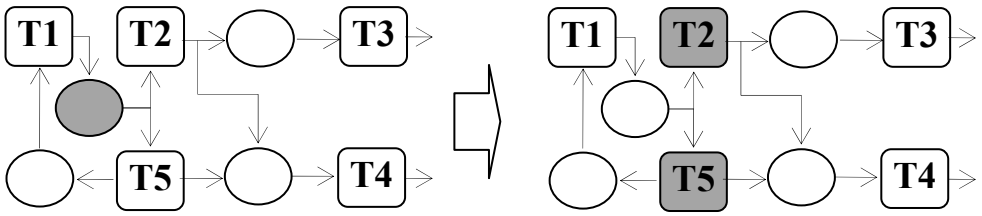
1-я итерация. Анализ комбинационной логики.

1-я итерация. Активизация триггеров T1, T3, T4.



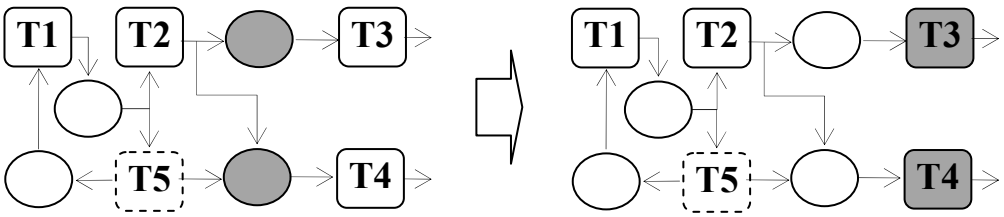
2-я итерация. Анализ комбинационной логики.

2-я итерация. Активизация триггеров T2 и T5. На одном из выходов T5 обнаружен цикл.



3-я итерация. Анализ комбинационной логики.

3-я итерация. Активизация триггеров T3 и T4. Новых событий на комбинационной логике не появляется.



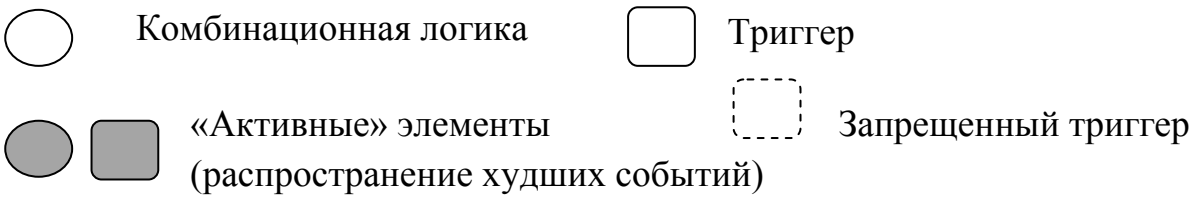


Рисунок 4. Пример работы алгоритма для схем, содержащих триггеры

В работе доказывается корректность используемого алгоритма, основываясь на уже доказанной корректности алгоритма для комбинационной схемы. Для доказательства корректности доказываются утверждения, из них ключевые:

Утверждение 3

Доказательство корректности алгоритма для последовательных схем

Пусть $v_1 \ldots v_n$ –критический путь на схеме, не содержащей критических циклов временном графе последовательной схемы, от входа v_1 к некоторой вершине v_n. Тогда в рамках работы алгоритма для последовательных схем будет создан критический путь $e_1 \ldots e_n$, такой, что $v(e_1) = v_1 \ldots v(e_n) = v_n$.

Также доказывается сходимость алгоритма:

Утверждение 4

Количество итераций основного цикла алгоритма для последовательных схем не превосходит количество триггеров в самом длинном по числу триггеров пути схемы, т.е. конечно.

В Главе исследуется сложность предлагаемого метода. Сложность итерации оценивается как $O(N+M)$, где N – количество элементов комбинационной логики, а M – количество триггеров. Количество итераций алгоритма равно числу триггеров L в самом длинном критическом пути схемы. Следовательно, суммарная сложность алгоритма – $O((N+M)*L)$.

Во **второй главе** приводится описание реализации предложенных методов для систем с общей памятью. В главе приводится классификация вычислительных систем и обзор методов параллелизации алгоритмов. Приводятся распространённые проблемы и методы используемые в параллельном программировании для систем с общей памятью. В качестве технологии реализации используется библиотека Intel© Threading Building Blocks, которая предлагает подход ориентированный на параллельное выполнение независимых задач. Программисту необходимо описать задачи и выбрать один из примитивов параллельного выполнения. Некоторые задачи при этом могут создаваться в результате выполнения других задач.

Технология Intel© Threading Building Blocks также предоставляет потокобезопасные контейнеры для реализации доступа разных задач, которые выполняются параллельно на разных потоках к общей памяти. Для распределения задач по потокам используется специальный планировщик, который поддерживает «воровство задача». Таким образом, абстрагируясь от конкретных потоков, возможно представить нашу задачу поиска критических путей в качестве множества мелких задач.

Параллельная реализация представляет собой несколько параллельных циклов созданных при помощи примитивов Intel© Threading Building Blocks. Алгоритмы реализованы с помощью шаблона parallel_do библиотеки Intel© Threading Building Blocks. Данный шаблон позволяет добавлять задачи в общий пул, откуда они с помощью планировщика задач распределяются по потокам. В качестве задачи выступает распространение событий на одном элементе и через исходящие из данного элемента межсоединения дальше, на элементы входы которых соединены с выходом анализируемого элемента. Анализ элемента производится с помощью библиотеки численных методов моделирования задержки RICE (описание примитивов Intel© Threading Building Blocks описано в Приложении 1) В алгоритме для последовательных схем помимо параллелизации расчета комбинационной части также возможно параллельно выполнять расчет триггеров на этапе распространения событий на триггерах. На этом этапе нет зависимости по данным, и синхронизация не требуется. Таким образом можно использовать шаблон tbb::parallel_for для параллельного выполнения распространения событий. Шаблон tbb::parallel_for позволяет параллельно выполнить наперед заданное количество задач (количество триггеров, на которых изменились события на входах нам известны). Как отмечалось в описании алгоритма, распространение событий через выходы схемы также можно выполнять параллельно. Поскольку количество выходов элемента заранее известно, то можно использовать шаблон tbb::parallel_for для параллельного распространения событий.

В **третей главе** приводятся результаты экспериментов и анализ масштабируемости предложенного метода. Реализация алгоритма была протестирована на ряде промышленных схем. В ходе анализа было проверено соответствие результатам, применяемым в промышленности инструментам СВА, ускорение, достигаемое алгоритмом на разном количестве потоков.

Ввиду того, что различные системы предполагают различное представление данных схемы в памяти, фактические результаты работы алгоритма могут существенно отличатся. Чтобы по возможности абстрагироваться от конкретной реализации (это невозможно сделать целиком, так как необходимо оценивать реальное время исполнения) все эксперименты проводились после загрузки данных в оперативную память.

Была показана хорошая балансировка загрузки и приемлемый минимальный размер задачи. В результате экспериментов удалось установить, что 95% задач имеют время выполнения, превосходящее требуемое разработчиками системы Intel© Threading Building Blocks.

Для анализа эффективности реализации метода был рассмотрен рост ускорения алгоритма с увеличением числа потоков для различных типов схем. Пример роста ускорения для схемы произвольной логики, на которой ускорение алгоритма самое лучшее ускорение (~33000 элементов)

Рисунок 5 Ускорение алгоритма на различном числе потоков

Из приведенных данных делается вывод, что график ускорения разного размера схем не насыщается при росте числа потоков до 16ти. Для больших схем график близок к линейному. Это дает основания предполагать, что стоит ожидать роста ускорения на системах с 32 ядрами и более.

Далее в главе проводится **исследование пределов масштабируемости** предложенного метода параллелизации – параллельного моделирования элементов. Эффективность параллельного алгоритма может быть разной для различных типов схем. В общем случае на ускорение параллельного алгоритма влияет количество параллелизуемой работы. В случае поиска критических путей в СВА на схеме последовательной работы является построение модели для одного выхода элемента. Таким образом максимально достижимое для данной схемы ускорение определяется количеством элементов, которые можно анализировать независимо. Для определения максимального ускорения строится граф выполняемых работ. Вершинами этого графа являются работы, которые необходимо выполнить. Вес вершины – время затрачиваемое на выполнение данной работы. Ребра графа отражают зависимость связи между работами. В нашем случае вершинами графа будет анализ одного элемента. Две вершины соединяются ребром, в случае если данные элементы соединяются ребром на графе схемы. Таким образом граф работ совпадает с временным графом схемы, но веса вершин соответствуют времени анализа данного элемента.

Для определения максимального ускорения необходимо найти критический путь в графе работ. Этот путь соответствует самой длительной последовательности работ, которую нет возможности выполнить параллельно. Таким образом время работы алгоритма ограниченно снизу длиной этого пути в графе работ (обозначим это время за $T_{критич.}$). Остальные работы будут выполняться параллельно (в случае неограниченного числа потоков). Время выполнения алгоритма на одном потоке равно сумме значений всех вершин в графе (обозначим его за $T_{общ}$). Как отмечалось в описании алгоритма, при анализе одного элемента есть возможность для

разных выходов элемента параллельно распространять события со входов на выходы элемента и через межсоединения на входы следующих элементов. Таким образом при вычислении $T_{общ}$ необходимо учитывать все время распространения событий. Для вычисления $T_{критич.}$ можно учитывать только время распространение того выхода, время распространения событий через который является максимальным. Таким образом ускорение параллельной

части алгоритма вычисляется как

В общем случае, на схеме, обладающей более высоким максимальным ускорением, при расчете на небольшом числе потоков не обязательно будет достигнутое большее ускорение, чем на схеме, обладающей меньшим максимальным ускорением.

В главе показана зависимость максимального ускорения от размера схемы для различных типов схем: Произвольная (нерегулярная) логика, Схемы потоковый обработки данных, схемы памяти (регистры, ROM)

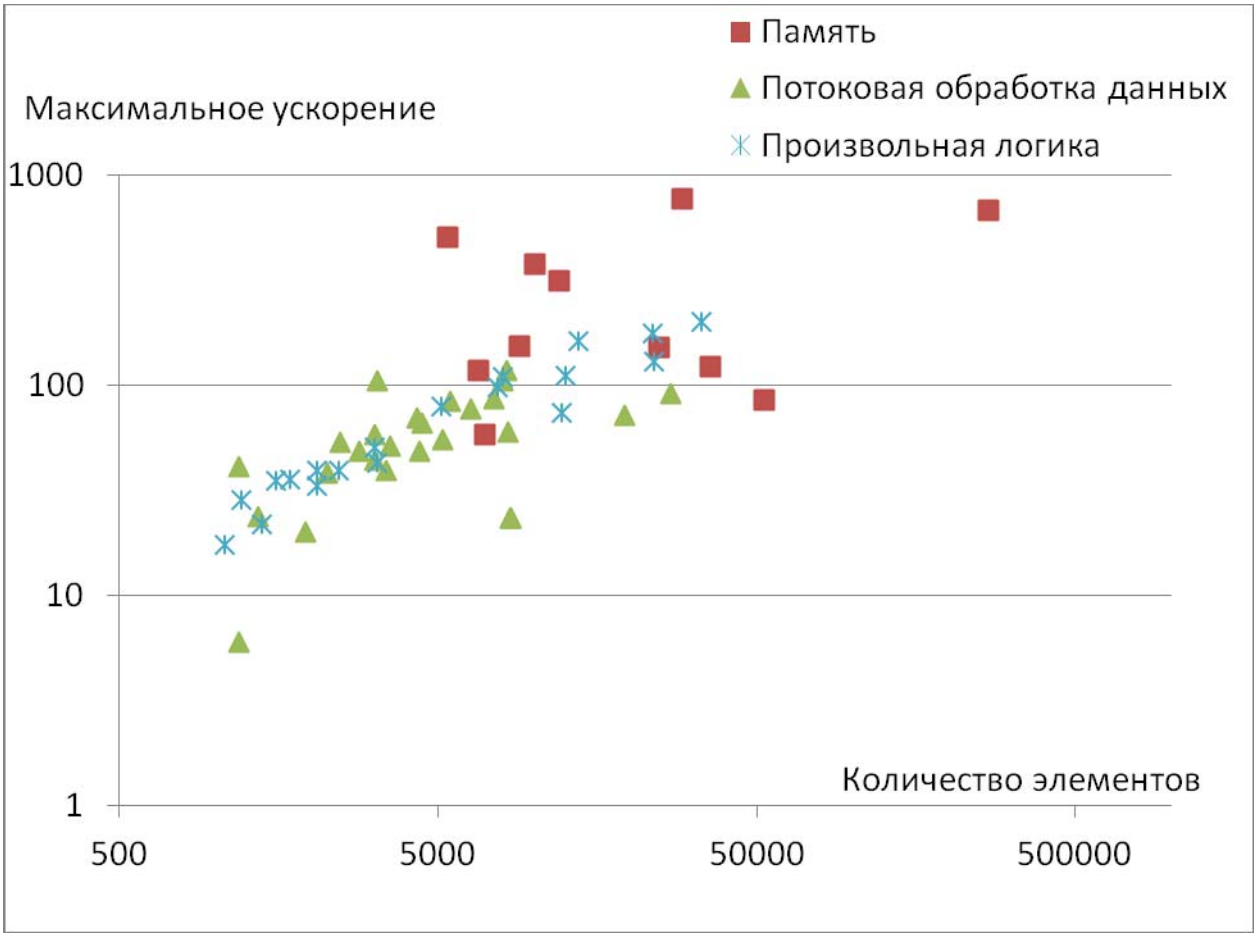


Рисунок 6 Максимальное теоритическое ускорение для различных типов схем

Как видно из графиков для всех типов схем характерно увеличение максимального ускорения с увеличением размера схемы.

В целом поиск Критических путей в задаче статического временного анализа представляет хороший потенциал для параллелизма (возможно ускорение в среднем до 90-100). Однако, зависимость по данным и необходимость прореживания является существенным ограничивающим параллелизм фактором. Снизить эту зависимость можно удалением некоторых цепей из рассмотрения (например: цепи питания или тестовые цепи, задержки сигнала в которых не влияют на максимальная частоту процессора), однако это требует определенного формата хранения данных о СБИС.

В главе показывается зависимость между максимальным потенциальным ускорением и реальным ускорением на 16ти потоках (вертикальная шкала линейная, а горизонтальная логарифмическая)

Рисунок 7 Зависимость между максимальным и реальным ускорением

Из графика видно, что с ростом максимального потенциального ускорения увеличивается и реальное ускорение на 16 потоках. Однако рост реального ускорения ожидаемо насыщается в районе значения 14. Исходя из приведенного исследования эффективности алгоритма, можно сделать вывод о большом потенциале к распараллеливанию задачи поиска критических путей в статическом анализе схем. Эффективность параллелизации прямопропорциональная размеру схемы, что делает подход применимым к большим схемам, время анализа которых занимает более суток.

В **Заключении** сформулированы результаты, выносимые на защиту:

1. Разработан параллельный метод и алгоритмы для вычислительных систем с общей памятью, реализующий поиск критических путей в задаче статического моделирования временных характеристик комбинационных и последовательных СБИС. Доказана корректность этих алгоритмов.

2. На основе разработанных алгоритмов создана параллельная программа, интернирована в экспериментальную систему анализа схем.

3. Проведены вычислительные эксперименты, показавшие, что полученные результаты заметно отличаются от промышленных инструментов СВА не более чем в 5% случаев. Получены оценки эффективности параллельных вычислений (эффективность составила 60-90%) и пределов масштабируемости метода для более чем 80 промышленных СБИС.

Публикации по теме диссертации

1. Князев Н.А. Малинаускас К.К. "Параллельный алгоритм поиска критических путей и циклов в задаче статического временного анализа для схем с последовательностной логикой" Сборник трудов конференции Микро и Нано Электронные Системы, Москва 2012 (список ВАК)

2. Князев Н.А. Малинаускас К.К. "Алгоритмы поиска критических путей в задаче статического временного анализа СБИС // Журнал «Информационные Технологии», 2012, №11, с. 2-9, г.Москва (список ВАК)

3 Князев Н.А. Статический временной анализ синхронных цифровых схем на многоядерных ЭВМ // Научный сервис в сети Интернет: экзафлопсное будущее: Труды Международной суперкомпьютерной конференции (19-24 сентября 2011, г. Новороссийск). 2011

4 Князев Н.А. Параллельный статический временной анализ цифровых схем с последовательностной логикой // Труды Конференции «Современные проблемы прикладной математики и информатики», г. Дубна 2012