ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК519.677

DOI: 10.21822/2073-6185-2024-51-1-113-122 Оригинальная статья /Original article

Применение генетического алгоритма для оптимизации процесса автоматической генерации тестовых шаблонов

В.И. Кураедов
Национальный исследовательский университет
«Московский институт электронной техники,
124498, г. Москва, г. Зеленоград, Площадь Шокина, дом 1, Россия

Резюме. Цель. Всесторонняя проверка интегральных схем имеет решающее значение для предотвращения дорогостоящих ошибок и задержек в цикле разработки продукта. Это включает в себя тестирование взаимодействия и совместимости всех различных компонентов, составляющих микросхему, таких как центральный процессор, память и различные периферийные устройства. Для проверки соответствия интегральной схемы всем функциональным требованиям необходимо около 70% от всего времени проектирования. Очевидной задачей, которая стоит перед производителями интегральных схем, является исследование и разработка методов снижения сложности проектирования и сокращения сроков их изготовления. Необходимо исследовать возможность применения генетического алгоритма для оптимизации процесса ATPG при проектировании интегральных схем и предложить новый метод для тестирования сбоев перекрестных помех. **Метод**. Проведены исследования в области применения генетического алгоритма для своевременного обнаружения ошибок, которые могут повлечь брак готового изделия. **Результат**. Получены данные о количестве жертв и целевых ошибок для всех сбоев типа stuck-at-0 и stuck-at-1 для схем из наборов ISCAS'85 и ISCAS'89. Обнаружено, что предлагаемый метод эффективнее по сравнению со случайными векторами для различных эталонных схем в зависимости от количества обнаруженных целевых сбоев. **Вывод.** Полученные результаты позволяют использовать представленный алгоритм в процессе проектирования ИС, позволив сократить время, затрачиваемое на тестирование и улучшить качество тестовых решений.

Ключевые слова: дефект, логические схемы, ATPG, генетический алгоритм, stuck-at-fault, тестовое покрытие

Для цитирования: В.И. Кураедов. Применение генетического алгоритма для оптимизации процесса автоматической генерации тестовых шаблонов. Вестник Дагестанского государственного технического университета. Технические науки. 2024; 51(1):113-122. DOI:10.21822/2073-6185-2024-51-1-113-122

Applying Genetic Algorithm for test pattern generation process optimization

V.I. Kuraedov
National Research University "Moscow Institute of Electronic Technology",
1 Shokin Square, Moscow, Zelenograd 124498, Russia

Abstract. Objective. Comprehensive integrated circuit (IC) verification plays a crucial role in preventing costly errors and delays in product development cycle. It includes testing interaction and compatibility of different system elements, such as central processing unit, memory and various peripheral devices. Validating IC’s compliance to the functional requirements list may take up to 70% of the design process duration. This proportion grows with complexity and size of the device under development. Consequently, the essential tasks that integrated circuit developers face are research and development of methods for cutting design complexity and reducing implementation time. **Method.** Conducted research regarding usage of genetic algorithm for error discovery, which could lead to a failure in the end product**. Result.** Collected data regarding the amounts of victims and target errors

for each fault of types stuck-at-0 and stuck-at-1 for circuits from ISCAS'85 and ISCAS'89 benchmarks. It has been discovered that the proposed method is more effective in comparison to random vector generation to an extent of target errors amount for every benchmark. **Conclusions.** Accumulated results allow for the algorithm usage during IC design in order to reduce time consumption for circuitry validation and improve on test kits quality.

Keywords: Faults, logical circuits, ATPG, genetic algorithm, stuck-at-fault, fault coverage

For citation: V.I. Kuraedov. Applying Genetic Algorithm for test pattern generation process optimization. Herald of Daghestan State Technical University. Technical Sciences. 2024; 51(1): 113-122. DOI:10.21822/2073-6185-2024-51-1-113-122

Введение. В результате технологических достижений, которые привели к увеличению плотности микросхем, увеличению числа слоев межсоединений и улучшению временных характеристик, проверки только на статические stuck-at сбои становится недостаточно [1], и теперь также требуется иметь дело с физическими дефектами, которые влияют на временное поведение данной схемы. Различные источники шума, такие как перекрестные помехи и шум источника питания, оказывают значительное влияние на временные характеристики проектируемых систем [2]. Увеличение числа транзисторов в микросхеме приводит к одновременному переключению большего числа устройств, что приводит к помехам в источнике питания, что в свою очередь снижает уровни напряжения устройства и увеличивает задержку сигнала.

Постановка задачи. Электрически изолированные устройства могут создавать помехи друг другу, что приводит к функциональным проблемам. Одно из таких взаимодействий, вызванных паразитной связью между проводами, известно как перекрестные помехи [3]. Эти шумовые эффекты могут привести к сбоям в работе полностью проверенного чипа и снижению производительности изготовленной микросхемы. Существует два основных типа эффектов перекрестных помех: импульсы, вызванные перекрестными помехами [4], и задержка, вызванная перекрестными помехами. Тип эффекта перекрестных помех, рассматриваемый в этой статье, - задержка, вызванная перекрестными помехами. Задержка перекрестных помех возникает, когда две линии, линия агрессора и линия жертвы, имеют одновременные или почти одновременные переключения сигналов, которые могут вызвать нежелательные эффекты, включая сбои, увеличение или уменьшение задержки сигнала. Если обе линии проходят в одном направлении, эффективная задержка уменьшается, что приводит к перекрестным помехам ускорения. Если агрессор и жертва перемещают сигнал в противоположном направлении, то задержка увеличивается, что приводит к перекрестным помехам замедления. У разработчика есть два варианта устранения ошибок, вызванных перекрестными помехами: либо путем изменения размера источника, экранирования соединительных линий и перенаправления сигналов, либо путем разработки методов генерации тестов на перекрестные помехи [5]. Разработчики часто выбирают последний вариант, поскольку переделывание готового дизайна может быть коммерчески затратным.

Методы исследования. В этой статье описывается метод генерации тестовых последовательностей на основе генетического алгоритма [6] для сбоев задержек, вызванных перекрестными помехами. Эксперименты проводились на эталонных схемах из наборов ISCAS'85 и ISCAS'89 [7]. Генетический алгоритм генерирует варианты тестовых векторов, а средство моделирования сбоев задержек, вызванных перекрестными помехами вычисляет пригодность вариантов тестовых векторов.

Количество перекрестных помех между всеми возможными комбинациями линий-агрессоров и линий-жертв очень велико, и их непрактично обнаруживать в больших сложных цепях [8]. Некоторые сбои не могут быть протестированы или в их тестировании нет необходимости. Следовательно, уменьшенный набор сбоев задержек, вызванных

перекрестными помехами, определяется статическим временным анализом схемы. Количество критических путей и линий, которые лежат на критическом пути, вычисляются с использованием топологической информации и временной информации. Затем получаются списки целевых сбоев, которые меньше набора всех возможных комбинаций сбоев.

Алгоритм расчета сокращенного целевого списка сбоев таков:

1. Вычисляется самое позднее и самое раннее время перехода для каждой линии.
2. По максимальному значению самого позднего времени перехода определяется самый длинный путь (критический путь). Находятся строки в самом длинном пути. Они формируют набор линий-жертв.
3. Окно синхронизации выбранной линии-жертвы сопоставляется с окном синхронизации линии-агрессора, и если у окон имеется наложение, то выбранная пара линий агрессора и жертвы добавляется в список целевых сбоев. Список входных сбоев для системы моделирования представляет собой сокращенный набор целевых сбоев.

Учитывая последовательность тестовых векторов в качестве входных данных, задача системы моделирования сбоев состоит в том, чтобы определить, какие из этих сбоев обнаружены. Обнаруженные сбои - это те, которые вызывают логическую ошибку, то есть сбой на внешних выходах [9]. Также отмечается эффект задержки на выходе. Система моделирования сбоев способна обнаруживать сбои типа «несколько агрессоров»/ «одиночная жертва».

Моделирование сбоев на основе автоматической генерации тестовых шаблонов (Automatic Test Pattern Generation (ATPG)) включает [10]:

1. Моделирование исправной схемы для случайных векторов.
2. Считывание сбоя из списка сбоев.
3. Моделирование схемы для активации сбоя для каждого вектора.
4. У жертвы и агрессоров направление движения сигналов должно быть противоположным. Тогда сбой активируется.
5. Внесение неисправности, задерживая только жертву на временной шаг, равный единице задержки. Другие события исполняются как обычно для исправной схемы. Продолжайте моделирование до конца времени симуляции.
6. Сравнение значения на всех внешних выходах и последовательных входах с исправной схемой.
7. Если исправная и неисправная цепи различаются, то обнаруживается сбой, и пары, включающие жертву и активированные агрессоры удаляются из списка сбоев.
8. Повторение данных шагов для всех сбоев в списке сбоев.

На рис. 1 путем статического временного анализа было обнаружено, что схема на примере состоит из шести самых длинных путей, а именно: [3, 7, 10, 14], [4, 7, 10, 14], [3, 7, 10, 15], [3, 7, 11, 15], [4, 7, 10, 15] и [4, 7, 11, 15]. Набор жертв - [3, 4, 7, 10, 11, 14, 15]. Общее количество целевых сбоев перекрестных помех для схемы равно 42. На первичные входы логической схемы подаются два набора данных (00001, 01000). Набор агрессоров /жертв, принимаемый к рассмотрению, таков: [6, 7, 11, 14, 15/10].

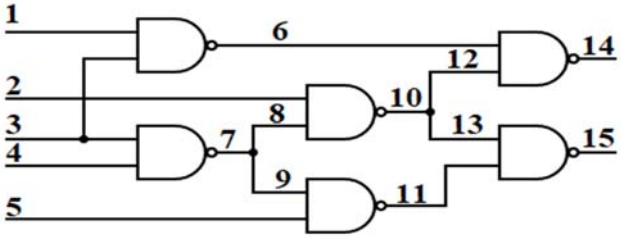

Рис. 1. Схема для выявления сбоев путём статического временного анализа
Fig. 1. Circuit for fault detection using static timing analysis

Агрессоры (11) и (14) активированы. Следовательно, жертва (10) получает задержку в одну единицу в логическом конусе. На (15) наблюдается сбой, а на (14) наблюдается эффект

задержки. Активированные агрессоры удаляются из списка сбоев.

Другие агрессоры учитываются при следующем тестировании. Неисправности считываются из списка сбоев, и выполняется моделирование сбоев. Весь процесс повторяется для всех тестовых векторов в наборе тестов.

В области тестирования СБИС генетические алгоритмы успешно используется при тестировании сбоев вентильной задержки и ошибок типа stuck-at-0 и stuck-at-1 [11]. Простота, надежность, эффективность и результативность генетического алгоритма делают его многообещающим инструментом для сложных систем. Он поддерживает набор потенциальных решений, называемых строками или хромосомами. Каждая строка связана со значением пригодности, определяемым пользовательской функцией пригодности.

На рис. 2 показана блок-схема генетического алгоритма.

Рис. 2. Блок-схема генетического алгоритма
Fig. 2. Genetic algorithm flow-chart

Он начинается с популяции, обычно генерируемой случайным образом [12], и эволюционного процесса, который состоит из этапов размножения, скрещивания и мутации и используется для создания совершенно новой популяции из уже существующей. Новая популяция и существующая популяция конкурируют за членство в очередном поколении. Отбор хромосом для новой популяции регулируется стратегией замещения. Старая популяция не учитывается. В результате отбора перекрестная мутация завершает цикл одного поколения. Генетический алгоритм прогрессирует через поколения, пока не будет достигнута цель, установленная пользователем цель, например фиксированное количество поколений.

Целью генерации теста для сбоев задержек, вызванных перекрестными помехами, является создание сокращенного набора тестов, который содержит как можно меньше тестовых векторов. По сути, это процесс, исследующий пространство пары тестовых векторов. Таким образом, генетический алгоритм можно использовать для оптимизации процесса исследования. Каждый тестовый вектор может рассматриваться как отдельный объект или строка.

Таким образом, мы можем рассматривать количество пар тестовых последовательностей как отдельную популяцию. Для оценки каждой пары тестовых векторов в популяции в любом поколении подходит система моделирования сбоев задержек, вызванных перекрестными помехами. Алгоритм ATPG выполняется в два этапа. На первом этапе исходная совокупность

тестовых векторов и последовательностей генерируется псевдослучайным процессом. На втором этапе тестовые векторы эволюционируют на основе функции пригодности Блок-схема процесса ATPG, основанного на генетическом алгоритме, показана на рис. 3.

Рис. 3. Блок-схема процесса ATPG, основанного на генетическом алгоритме
Fig. 3. Flow-chart of ATPG process, based on GA

Используемая функция пригодности такова: FIT = Num_faults, (1)

где: Num_faults - количество обнаруженных сбоев.

На этом этапе на основе псевдослучайного процесса генерируются исходные последовательности, состоящие из N векторов. Для этих последовательностей происходит моделирование сбоев для сбоев из списка. Если последовательность обнаруживает сбой, то он удаляется из списка сбоев и соответствующая последовательность добавляется в набор решений. Если ни одна из последовательностей не обнаруживает сбоев, то в набор решений добавляется последняя последовательность, сгенерированная в соответствующем цикле. Этот процесс повторяется настраиваемое пользователем количество раз, называемое I_{max}. Блок-схема первой фазы показана на рис. 4.

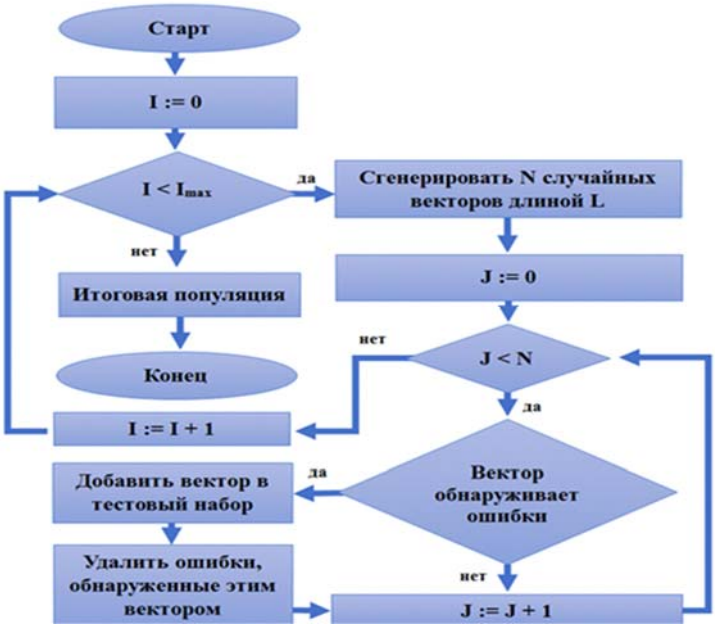

Рис. 4. Блок-схема первой фазы процесса ATPG, основанного на генетическом алгоритме
Fig. 4. Flow-chart of the first phase of ATPG process, based on GA

Обсуждение результатов. Исходная популяция генетического алгоритма состоит из последовательностей, сгенерированных в первой фазе. Чтобы сгенерировать новую популяцию из существующей, отбираются две особи (родители) и скрещиваются для создания двух полностью отдельных особей (дочерних), и каждый дочерний вид мутирует с некоторой небольшой вероятностью мутации. Оператором выбора является выбор на основе ранга. При выборе на основе ранга решения сортируются в соответствии с их пригодностью от худшего (ранг 1) до лучшего (ранг N). Каждому элементу в отсортированном списке присваивается пригодность, равная рангу решения в списке. После этого применяется оператор пропорционального отбора с ранжированным значением пригодности, и выбираются лучшие решения для заполнения новой популяции. Предыдущая популяция исключается. Используемое скрещивание - одноточечное. Вероятность скрещивания равна 1, а вероятность мутации равна 0,01, данные значения используются для всех наборов. Количество поколений равно 8, чтобы сократить время выполнения. Во время тестовой генерации используется значение размера популяции, равное 16. Блок-схема для второй фазы представлена на рис. 5.

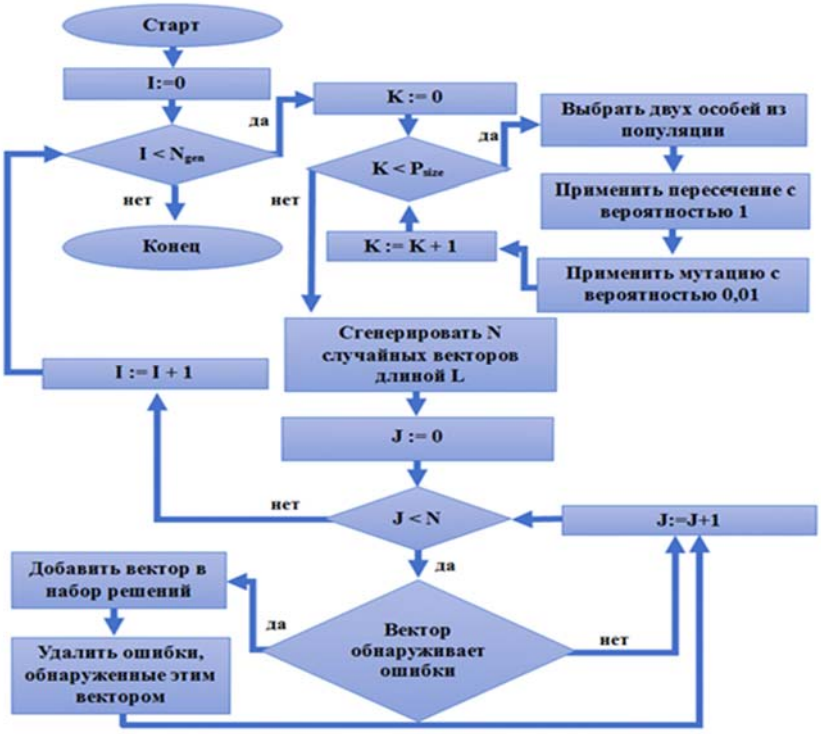

Рис. 5. Блок-схема второй фазы процесса ATPG, основанного на генетическом алгоритме
Fig. 5. Flow-chart of the second phase of ATPG process, based on GA

Система моделирования сбоев задержек, вызванных перекрестными помехами и генератор тестов на основе генетического алгоритма реализованы в MATLAB [13] в среде LINUX. ATPG на основе случайной генерации векторов и на основе генетического алгоритма применяются к некоторым схемам из наборов ISCAS'85 и ISCAS'89.

В табл. 1 приведены характеристики данных схем. После названия схемы указано количество первичных входов и первичных выходов, количество вентилей, количество путей и количество критических путей. Пути всей схемы анализируются с использованием древовидной структуры данных. Общее количество путей в схеме вычисляется с использованием алгоритма поиска в глубину [14], который использует процедуру рекурсивного поиска. Критические пути - это пути, задержка которых превышает заданный процент от наибольшей задержки распространения в схеме. Выбор производится с использованием статического анализа синхронизации и заданной задержки вентиля. Предполагается, что задержка вентиля для статического анализа синхронизации составляет одну единицу измерения времени.

Таблица 1. Характеристики схем ISCAS'85 и ISCAS'89
Table 1. Circuit characteristics of ISCAS'85 and ISCAS'89 sets

Схема Scheme	Количество вентилей Number of valves	Количество входов Number of inputs	Количество выходов Number of outputs	Количество критических путей Number of critical paths	Количество результирующих путей Number of resulting paths
S1494	647	14	25	1	976
S1488	653	14	25	1	962
S1238	508	32	32	30	3558
S1196	529	32	32	9	3097
S820	289	23	24	11	492
S526	193	24	27	1	410
S510	211	25	13	1	369
S420.1	218	34	17	1	474
C386	159	13	13	10	207
S349	161	24	26	1	365
S344	160	24	26	1	355
S298	119	17	20	1	231
S208.1	104	18	9	1	142
S208	96	19	10	2	145
S27	10	7	4	6	28
C880	383	60	26	92	8642
C449	202	41	32	14	9440
C432	160	36	7	2199	83926
C17	6	5	2	6	11

В табл. 2 показано покрытие сбоев, вызванных перекрестными помехами, полученное для нескольких эталонных схем.

Таблица 2. Эффект сбоя на первичном выходе
Table 2. Fault effect on the primary output

Схема Scheme	Количество целевых ошибок Number of target errors	Количество жертв Number of victims	Эффект сбоя на первичном выходе в зависимости от типа ATPG Effect of primary output		Процессорное время, (c) Processor time, (s)
			Случайная Random	Генетический алгоритм Genetic algorithm	
S1494	4283	18	4280	4282	45.03
S1488	4305	18	4302	4304	29.76
S1238	5822	45	2764	4556	20.36
S1196	10630	55	5791	9150	20.03
S820	7738	43	7362	7728	11.61
S526	891	10	729	886	4.21
S510	1098	13	1091	1098	4.76
S420.1	1276	14	927	1257	4.05
C386	4195	49	4152	4189	2.45
S349	1197	21	1192	1196	7.56
S344	1190	21	1185	1189	7.53
S298	537	10	532	533	2.32
S208.1	558	12	371	512	1.26
S208	743	18	559	726	1
S27	74	10	70	72	0.03
C880	9279	70	5722	8922	24.68
C449	21879	207	17937	20806	22.7
C432	9327	103	6629	7918	125.35
C17	42	7	42	42	0.2

Эффект сбоя на первичном выходе показывает количество сбоев, для которых на

первичном выходе был замечен эффект задержки, вызванный перекрестными помехами.

Обнаруженные сбои - это те, которые вызывают логическую ошибку на первичном выходе. Поскольку для больших цепей нецелесообразно тестировать все сбои, сокращенный набор целевых сбоев рассчитывается путем выбора жертв, которые находятся на критическом пути, и для каждого сбоя временное окно агрессор-жертва должно накладываться. Входной список сбоев для каждой цепи состоит из каждой комбинации пар одиночный агрессор/одиночная жертва. Предполагается, что вентили имеют единичную задержку, и вводимое значение задержки перекрестных помех также предполагается равным единице времени.

В табл. 2 для схем c449 и c880 показано, что 95% сбоев задержек, вызванных перекрестными помехами, приводили к задержке на первичном выходе. Для 13 из 15 схем из набора ISCAS'89 98% сбоев задержек, вызванных перекрестными помехами, приводили к задержке на первичном выходе. Схемы s1196, s1238 и c432 с критическими путями длиной 9, 30 и 2199, соответственно, вызвали эффект задержки только для 86%, 78,2% и 84,8% сбоев задержек, вызванных перекрестными помехами, соответственно. Это может быть связано с тем фактом, что основой для нахождения критических путей и устранения целевых сбоев является статический временной анализ, который не учитывает ложные пути.

В табл. 2 количество сбоев, обнаруженных с использованием ATPG на основе генетического алгоритма, было намного больше, чем при использовании случайных векторов для 17 из 19 протестированных эталонных схем. При работе со случайными векторами количество обнаруженных сбоев составило около 70% для схем c432 и c449, а для c880 охват сбоев составил 48%. Также при использовании случайных векторов количество обнаруженных сбоев составило от 24% до 66% для 10 схем из набора ISCAS'89.

Низкий охват отказов не является чем-то необычным, поскольку из-за противоречивых логических условий возникает множество сбоев задержек, вызванных перекрестными помехами, которые невозможно распознать или распространить на первичный выход. Более того, введенная единичная задержка может быть недостаточной, чтобы вызвать логическую ошибку на первичном выходе. Для большинства тестовых схем были получены компактные тестовые векторы. Время выполнения центрального процессора, показанное в табл.2, является временем генерации тестов на основе генетического алгоритма. Оно было немного выше для более крупных схем.

График, показанный на рис. 6, дает сравнительную диаграмму, показывающую эффективность генетического алгоритма по сравнению со случайными векторами для различных эталонных схем в зависимости от количества обнаруженных целевых сбоев.

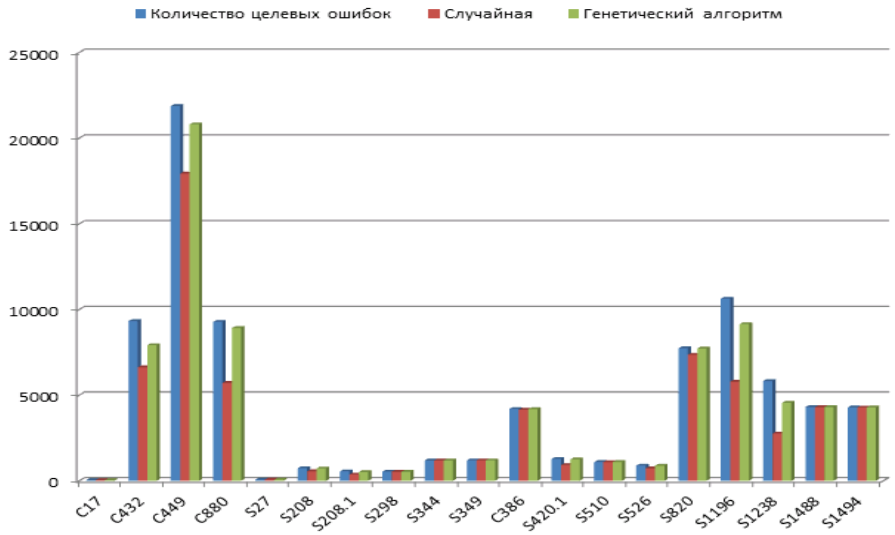

Рис. 6. Сравнение эффективности разных подходов ATPG для различных схем
Fig. 6. Comparison of different ATPG approaches for various circuits

Вывод. Повышенная плотность проектирования в субмикронных интегральных схемах приводит к более значительным помехам между сигналами из-за емкостной связи или перекрестных помех, которые могут приводить к сбоям типа stuck-at-0 и stuck-at-1. Чтобы гарантировать производительность проектирования, методы ATPG должны учитывать, как перекрестные помехи влияют на задержки распространения сигналов.

В этой статье разработана система моделирования сбоев задержек, вызванных перекрестными помехами, с использованием генетического алгоритма. Избыточные сбои задержек, вызванные перекрестными помехами, которые никогда не влияют на производительность схемы, отфильтровываются.

Система моделирования сбоев задержек, вызванных перекрестными помехами, способна обнаруживать сбои типа n-агрессоров/одиночная жертва.

Представлены результаты для сбоев задержек, вызванных перекрестными помехами, которые приводят к эффекту задержки на первичном выходе, а также для тех, которые приводят к логической ошибке на первичном выходе. Проверки на эталонном наборе схем приводили к эффекту задержки при 72-99% сбоев задержек, вызванных перекрестными помехами. Для сбоев задержек, вызванных перекрестными помехами, приводящих к логической ошибке на первичном выходе, метод обеспечил покрытие сбоев от 48% до 72% на четырех схемах из набора ISCAS’85 и от 12% до 66% на 15 схемах из набора ISCAS’89.

Библиографический список:

1. S. Hasan, A.K. Palit, W. Anheier, Test pattern generation and compaction for crosstalk induced glitches and delay faults, in Proceedings of the 23rd International Conference on VLSI Design (2010).
2. S. Jayanthy, M.C. Bhuvaneswari, S. Keesarapalli, Test generation for crosstalk-induced delay faults in VLSI circuits using modified FAN algorithm. VLSI Des. 2012. 2012;10 (Article ID 745861). https://doi.org/10.1155/2012/745861.
3. S. Jayanthy, M.C. Bhuvaneswari, M. Prabhu, Simulation based ATPG for low power testing of crosstalk delay faults in asynchronous circuits. Int. J. Comput. Appl. Technol. 2013; 48(3): 241–252. ISSN: 1741-5047.
4. W. Chen, S. K. Gupta, and M. A. Breuer, “Analytic models for crosstalk delay and pulse analysis under non-ideal inputs,” in Proceedings of the IEEE International Test Conference, November 1997; 809–818.
5. S. Chun, T. Kim, and S. Kang, “ATPG-XP: test generation for maximal crosstalk-induced faults,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 9, Article ID 5208481, 2009;28(9):1401–1413.
6. Darrel Whitley: A Genetic Algorithm Tutorial; November 10, 1993; Technical Report CS-93-103 (Revised); Department of Computer Science, Colorado State University, Fort Collins, US
7. D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," North Carolina State University, vol. 25, 1985.
8. Pasca, V., Anghel, L. & Benabdenbi, M. Kth-Aggressor Fault (KAF)-based Thru-Silicon-ViaInterconnect Built -In Self-Test and Diagnosis. J Electron Test 2012; 817–829. https://doi.org/10.1007/s10836-012-5322-3.
9. Lee S., Cobb B., Dworak J., Grimaila M. R., Mercer M. R. A new ATPG algorithm to limit test set size and achieve multiple detections of all faults. Design, Automation and Test in Europe Conference and Exhibition, 2002; 94-99. DOI: 10.1109/DATE.2002.998255.
10. M. Fujita, N. Taguchi, K. Iwata, and A. Mishchenko. Incremental atpg methods for multiple faults under multiple fault models. In Sixteenth International Symposium on Quality Electronic Design, 2015; 177–180.
11. Jutman A., Ubar R. Design error diagnosis in digital circuits with stuck-at fault model. Microelectron Reliab, Dec. 2000; 40(2):307–320.
12. Rudnick, E.M., Patel, J.H., Greenstein, G.S., Niermann, T.M.: A genetic algorithm framework for test generation. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 1997; 16(9), 1034–1044.
13. Xue, D., Press, T.U. MATLAB Programming: Mathematical Problem Solutions. De Gruyter STEM. De Gruyter. 2020; 21.
14. Aggarwal, A.; Anderson, R. J. (1988), "A random NC algorithm for depth first search", Combinatorica, 8 (1): 1–12, doi:10.1007/BF02122548

References

1. S. Hasan, A.K. Palit, W. Anheier, Test pattern generation and compaction for crosstalk induced glitches and delay faults, in Proceedings of the 23rd International Conference on VLSI Design (2010).

2. S. Jayanthy, M.C. Bhuvaneswari, S. Keesarapalli, Test generation for crosstalk-induced delay faults in VLSI circuits using modified FAN algorithm. VLSI Des. 2012. 2012;10 (Article ID 745861). https://doi.org/10.1155/2012/745861.
3. S. Jayanthy, M.C. Bhuvaneswari, M. Prabhu, Simulation based ATPG for low power testing of crosstalk delay faults in asynchronous circuits. Int. J. Comput. Appl. Technol. 2013; 48(3): 241–252. ISSN: 1741-5047.
4. W. Chen, S. K. Gupta, and M. A. Breuer, "Analytic models for crosstalk delay and pulse analysis under non-ideal inputs," in Proceedings of the IEEE International Test Conference, November 1997; 809–818.
5. S. Chun, T. Kim, and S. Kang, "ATPG-XP: test generation for maximal crosstalk-induced faults," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Article ID 5208481, 2009;28(9):1401–1413.
6. Darrel Whitley: A Genetic Algorithm Tutorial; November 10, 1993; Technical Report CS-93-103 (Revised); Department of Computer Science, Colorado State University, Fort Collins, US
7. D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," North Carolina State University, 1985; 25.
8. Pasca, V., Anghel, L. & Benabdenbi, M. Kth-Aggressor Fault (KAF)-based Thru-Silicon-ViaInterconnect Built - In Self- Test and Diagnosis. J Electron Test 28, 2012; 817–829. https://doi.org/10.1007/s10836-012-5322-3.
9. Lee S., Cobb B., Dworak J., Grimaila M. R., Mercer M. R. A new ATPG algorithm to limit test set size and achieve multiple detections of all faults. Design, Automation and Test in Europe Conference and Exhibition, 2002; 94-99. DOI: 10.1109/DATE.2002.998255.
10. M. Fujita, N. Taguchi, K. Iwata, and A. Mishchenko. Incremental atpg methods for multiple faults under multiple fault models. In Sixteenth International Symposium on Quality Electronic Design, 2015; 177–180.
11. Jutman A., Ubar R. Design error diagnosis in digital circuits with stuck-at fault model. Microelectron Reliab, Dec. 2000; 40(2):307–320.
12. Rudnick, E.M., Patel, J.H., Greenstein, G.S., Niermann, T.M.: A genetic algorithm framework for test generation. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 1997; 16(9), 1034–1044.
13. Xue, D., Press, T.U. MATLAB Programming: Mathematical Problem Solutions. De Gruyter STEM. De Gruyter. 2020; 21.
14. Aggarwal, A.; Anderson, R. J. "A random NC algorithm for depth first search", Combinatorica,1988; 8 (1): 1–12, doi:10.1007/BF02122548

Сведения об авторе:
Вадим Иванович Кураедов, аспирант, институт интегральной электроники имени академика К.А. Валиева (ИнЭл), vladimir96k@mail.ru
Information about author:
Vadim I. Kuraedov, Post-graduate Student, Institute of Integrated Electronics (InEl); vladimir96k@mail.ru
Конфликт интересов/Conflict of interest.
Автор заявляет об отсутствии конфликта интересов/The author declare no conflict of interest.
Поступила в редакцию/ Received 15.12.2023.
Одобрена после рецензирования / Reviced 14.01.2024.
Принята в печать /Accepted for publication 14.01.2024.