Методы повышения производительности вычислений при расчете метрик надежности комбинационных логических схем

А. Л. Стемпковский, Д. В. Тельпухов*, Р. А. Соловьев, М. В. Мячиков

Институт проблем проектирования в микроэлектронике РАН, Москва, Зеленоград, Россия

*Контактный e-mail: nofrost@inbox.ru

Представлен ряд методов повышения производительности вычислений при расчете метрик надежности комбинационных логических схем. В основе предлагаемых подходов лежат стандартные способы оптимизации вычислений, такие как кэширование, кэширование с измененным порядком вычислений и векторизация, позволяющие ускорить вычисления за счет более интенсивного использования памяти. На широкой выборке сгенерированных тестовых схем экспериментально показана высокая эффективность представленных методов.

Ключевые слова: сбоеустойчивость, комбинационная схема, наблюдаемость вентиля, коэффициент чувствительности, кэширование.

Введение

Вопросы, связанные с обеспечением надежности электронных компонентов, в настоящее время находятся в центре внимания научного сообщества и промышленности. Традиционно повышенные требования к надежности предъявляются в конкретных сегментах рынка, это космическая промышленность, авионика и атомная энергетика. В таких приложениях микросхемы находятся под влиянием внешних воздействий (например, ионизирующего излучения), которые могут вызвать сбои в комбинационной логике и последовательных элементах, что может приводить к отказам аппаратуры. В последние годы процесс непрерывного повышения степени интеграции в микроэлектронной промышленности создает новые проблемы в отношении надежности цифровой логики. Из-за малых размерностей транзисторов работа КМОП-логики из детерминированной области переходит в вероятностную с большими вариациями параметров транзистора. Из-за уменьшения узловой емкости новые технологии гораздо более чувствительны к воздействию радиации.

Комбинационная логика как источник возникновения сбоев много лет не вызывала опасений, и предыдущие исследования по большей части фокусировались на ошибках программного обеспечения и памяти (SRAM и DRAM). Однако при переходе к глубоко субмикронному технологическому процессу ошибки логической природы начинают составлять значительную часть в общем объеме сбоев схемы [1]. Увеличение тактовых частот, напряжения питания, уменьшение размеров транзисторов приводят к экспоненциальному росту интенсивности сбоев в комбинационных схемах. Кроме того, часто используемая суперконвейеризация уменьшает количество логических элементов

между последовательными стадиями, что приводит к ослаблению эффектов электрического маскирования, а также усугубляет уязвимость логических схем при сбоях [2]. В [3] показано, что за последние годы интенсивность сбоев комбинационной логики на уровне чипов значительно увеличилась. Более того, прогнозируется, что интенсивность логических сбоев будет доминировать даже на низких частотах по мере дальнейшего уменьшения технологических размеров транзисторов.

Несмотря на достигнутые успехи в разработке кодовых методов защиты, используемых при хранении, передаче, а также арифметической обработке данных, для обеспечения требуемого уровня отказоустойчивости логических схем до сих пор, как правило, используют методы кратного резервирования [4]. Ключевым аспектом при решении проблемы повышения сбоеоустойчивости комбинационных схем является задача создания методов оценки эффективности разрабатываемых подходов, которая в свою очередь не может быть решена без создания методов моделирования цифровых СБИС на логическом уровне.

Статья посвящена разработке методов ускорения вычислений метрик сбоеустойчивости комбинационных схем. Описаны базовая метрика сбоеустойчивости комбинационных схем и базовый алгоритм вычисления предлагаемой метрики, предложены различные методы повышения производительности вычислений при расчете метрик надежности комбинационных логических схем, приведены результаты вычислительных экспериментов.

1. Метрика сбоеустойчивости логических схем и базовый алгоритм ее вычисления

Определим метрику сбоеустойчивости, предложенную в [5], вычисление которой будем оптимизировать ниже. Введем некоторые обозначения. Пусть N и M — это число входов и число вентилей схемы соответственно. Следуя модели ошибок, предложенной Джоном фон Нейманом [6], ошибкой в элементе будем называть сбой, случайно возникающий независимо во всех элементах с некоторой одинаковой вероятностью p и приводящий к инверсии на выходе элемента. Ошибку в i-м элементе схемы обозначим e_i. Если на i-м элементе возникла ошибка, то $e_i = 1$, иначе $e_i = 0$. Вектор входных значений обозначим через $\mathbf{X} = (x_1, x_2, \ldots, x_N)$, а вектор $\mathbf{e} = (e_1, e_2, \ldots, e_M)$ будем называть вектором ошибки.

Далее, через $E(\mathbf{X}, \mathbf{e})$ обозначим характеристическую функцию множества пар векторов (входных сигналов $\mathbf{X}$ и векторов ошибок $\mathbf{e}$), при которых на выходе схемы возникает ошибка [7, 8]:

$$E(\mathbf{X}, \mathbf{e}) = \begin{cases} 1, & \text{если набор } (\mathbf{X}, \mathbf{e}) \text{ приводит к ошибке;} \\ 0, & \text{иначе.} \end{cases}$$

В этих обозначениях используемая метрика сбоеустойчивости записывается следующим образом:

$$\alpha = \frac{1}{2^N} \sum_{\bar{X}, \bar{e}, |\bar{e}|=1} E(\mathbf{X}, \mathbf{e}), \tag{1}$$

где суммирование ведется по всем входным комбинациям и всем комбинациям векторов ошибок, вес которых равен единице. Данная метрика получила название коэффи-

циента чувствительности комбинационной схемы к одиночным сбоям. Анализируя (1), нетрудно видеть, что данная метрика, по сути, является линейным коэффициентом полинома ошибки — функции, полностью характеризующей вероятность сбоя схемы в зависимости от вероятности сбоя вентилей. Использование данной метрики оправдано предположением о том, что вероятность сбоя вентиля в действительности очень мала [9], а следовательно, мало и влияние степеней полинома, отличных от первой. Так, коэффициент чувствительности определяет касательную к графику полинома ошибки в точке нуль, определяя наиболее точную аппроксимацию в условиях, когда вероятность сбоя вентиля стремится к нулю. Обратившись к (1), можно сформулировать еще одну интерпретацию предложенной метрики. Коэффициент чувствительности задает сумму наблюдаемостей всех вентилей логической схемы и может быть интерпретирован как среднее число "ненадежных" элементов в схеме по отношению к одиночным сбоям. Кроме того, предлагаемый коэффициент не зависит от вероятности сбоя вентиля, что позволяет использовать его на ранних этапах проектирования сбоеустойчивых схем, а также при разработке методов повышения сбоеустойчивости, когда не определены элементная база и условия эксплуатации схемы.

Приведем используемые структуры данных и язык реализации описываемых алгоритмов. В качестве языка для реализации алгоритмов принят Python 3.4. Структура схемы в нашей реализации задается списком элементов, каждый из них представляет собой упорядоченную пару: (операция, [список аргументов]), в которой список аргументов состоит из индексов элементов, выходы которых соединены со входами этого элемента. Предполагается также, что список является результатом топологической сортировки соответствующего схеме орграфа. Время сортировки линейно от суммы количества элементов и связей в схеме и не влияет на оценку сложности алгоритмов, приведенных ниже.

Базовый алгоритм состоит в простом переборе всех комбинаций в сумме из (1) и вычислении $E(\mathbf{X}, \mathbf{e})$ с аккумуляцией результата. На языке Python он записывается следующим образом:

```
def baseline(sch):
"""
:param sch: исследуемая схема – объект, реализующий метод process,
    возвращающий для вектора входных значений и вектора ошибки вектор
    выходных значений
:return: коэффициент надежности
"""
n = sch.inputs()  # количество входов в схеме
m = sch.elements()  # количество элементов в схеме

def count_errors(outputs_true, outputs_error):
    """
    :param outputs_true: список ожидаемых значений на выходах схемы
    :param outputs_error: список реальных значений на выходах схемы
    :return: 1, если хотя бы в одной позиции есть различия, иначе 0
    """
    if any(i ^ j for i, j in zip(outputs_true, outputs_error)):
        return 1
    else:
```

```
            return 0

errors = 0
for inps in inputs_combinations(n): # перебор всех векторов длины n
    inputs_values = list(inps)
    errors_values = [0] * m
    for error_number in range(m):

        errors_values[error_number-1] = 0
        errors_values[error_number] = 1

        outputs_true = sch.process(inputs_values)
        outputs_error = sch.process(inputs_values, errors_values)
        errors += count_errors(outputs_true, outputs_error)

return Fraction(errors, 2**n)
```

Легко видеть, что сложность этого алгоритма относительно операции вычисления значения на выходе одного вентиля составляет $O(2^N M^2)$.

2. Методы ускорения вычислений коэффициента чувствительности комбинационной схемы

2.1. Простое кэширование

В основе первой предлагаемой оптимизации лежит тот факт, что у двух последовательных входных комбинаций некоторые значения совпадают, а значит, значения на выходе элементов, зависящие только от них, можно вычислить на предыдущей операции и сохранить, чтобы использовать на следующей. Предполагается, что получение результата логической операции является гораздо более затратной вычислительной операцией, чем чтение сохраненного результата из памяти, что не всегда верно.

При инициализации алгоритма производится топологическая сортировка элементов схемы для определения порядка выполнения логических операций и каждому элементу ставится в соответствие множество элементов, непосредственно использующих результат на выходе этого элемента в вычислениях. Затем на каждой итерации по разнице между последовательными входными комбинациями определяется список элементов, значения на выходах которых могли измениться. Значения на выходах вычисляются в порядке, определенном топологической сортировкой. При этом значение на выходе каждого обработанного элемента сравнивается со значением на предыдущей итерации и в случае несовпадения результатов список с сохранением топологического порядка дополняется элементами, для которых выход элемента с изменившимся значением является входом.

На примере схемы, изображенной на рис. 1, рассмотрим несколько итераций алгоритма вычисления коэффициента надежности с применением простого кэширования.

Значения на входах схемы и выходах элементов на двух последовательных итерациях с номерами m, $m+1$ приведены на рис. 2. В данной ситуации во входной комбинации изменяется только один входной бит и значения на выходах элементов, не зависящих от этого входа, можно не вычислять снова, а считать из кэша. Изменение входной

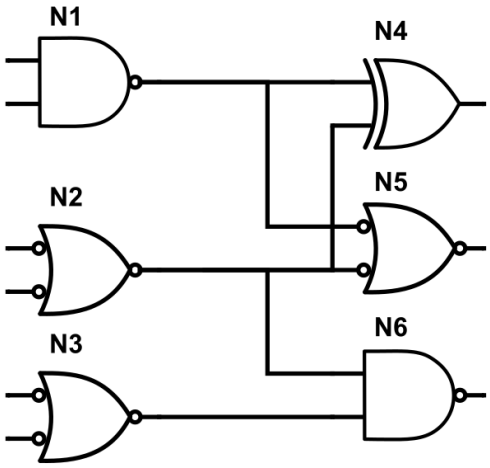

Рис. 1. Тестовая комбинационная логическая схема

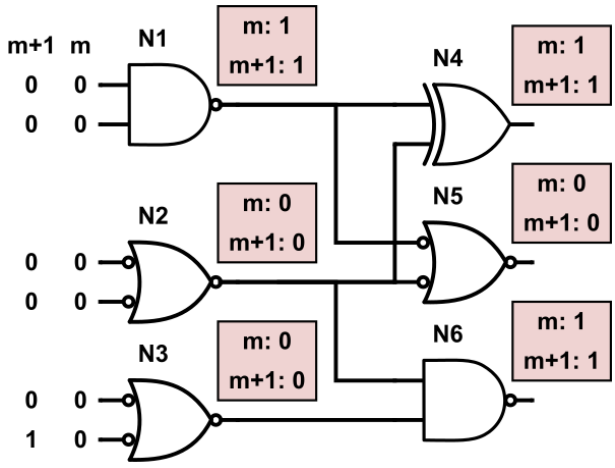

Рис. 2. Простое кэширование

комбинации в данном примере не меняет значений на выходах элементов схемы, а следовательно, результат работы схемы на $m + 1$-й итерации может быть сразу получен из памяти.

2.2. Кэширование с измененным порядком входных комбинаций

Приведенная выше оптимизация может быть естественным образом усовершенствована, если изменить порядок входных комбинаций таким образом, чтобы каждое изменение входной комбинации влекло за собой как можно меньше вычислений на следующей итерации.

Задача выбора оптимальной последовательности выходит за рамки этой статьи, однако в качестве хорошего приближения можно использовать следующую эвристику. Поскольку количество дополнительных вычислений тем больше, чем больше отличающихся входных значений между двумя комбинациями, следует свести их количество к минимуму. Этого позволяет достичь использование кода Грея, так как между двумя соседними комбинациями в нем всегда отличается только один входной бит. Схема алгоритма аналогична предыдущей с отличием только в порядке входных комбинаций.

2.3. Обработка всех векторов ошибки за один проход

Подход с кэшированием можно расширить также и на комбинации ошибок. Так, при вычислении результата на выходе схемы при определенной входной комбинации можно вычислять, как влияет ошибка в каждом элементе, сохраняя это значение для использования в последующих вычислениях, что позволяет снизить суммарное количество требуемых операций. Одна итерация алгоритма выглядит следующим образом. Для каждого элемента помимо результата на выходе следует вычислить то же значение при условии, что в элементе произошел сбой. Помимо этого вычисляется также то значение, которое было бы получено при условии ошибки в одном из предыдущих элементов. Таким образом, по завершении итерации алгоритма будут получены значения на вы-

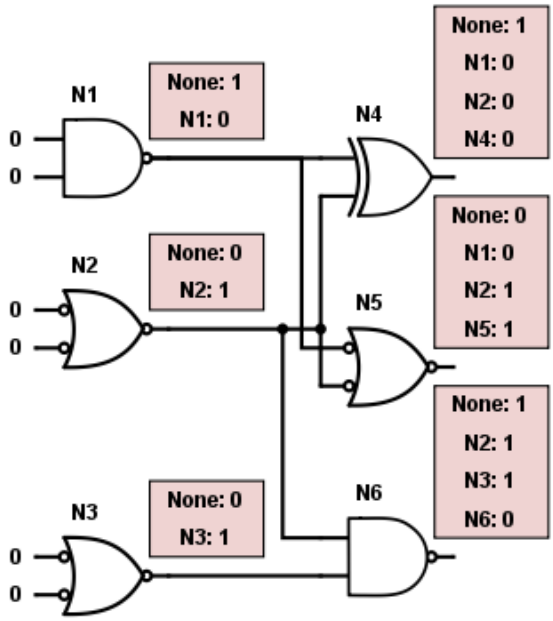

Рис. 3. Схема обработки всех векторов ошибки за один проход

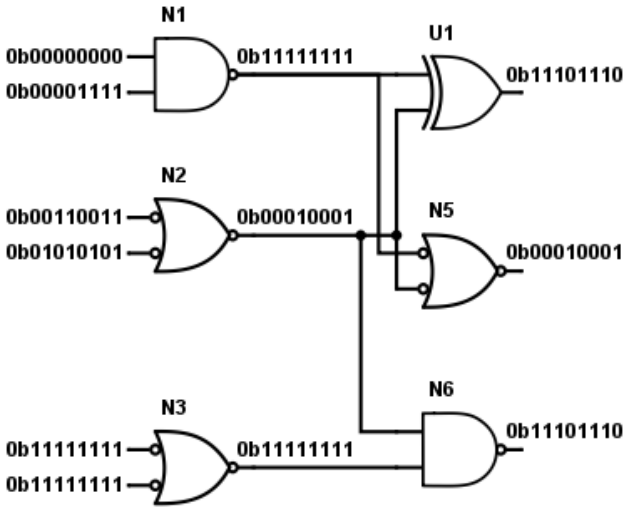

Рис. 4. Векторизация вычислений

ходах схемы для всех одинарных ошибок. На рис. 3 приведена схема, изображенная на рис. 1, с рассчитанными значениями на выходах элементов для одной итерации алгоритма.

Значения на выходах элементов, соответствующие безошибочной работе схемы, обозначены меткой None, а значения, возникающие при ошибке в одном из элементов, — меткой этого элемента.

2.4. Векторизация

В описанных выше вариантах алгоритма вычисления коэффициента надежности значения 0 и 1 хранятся в виде целых чисел, однако логические операции реально выполняются над числами, разрядность которых больше одного бита. Таким образом, производительность вычислений можно повысить, если объединить последовательные значения комбинаций на каждом входе в слова и выполнять побитовые логические операции уже над ними. Вычисления на одной из итераций для схемы, изображенной на рис. 1, будут выглядеть так, как показано на рис. 4.

Поскольку в языке Python реализована поддержка так называемой “длинной арифметики”, для работы с целыми числами произвольной длины разрядность таких слов может быть выбрана произвольной, причем этот выбор существенно влияет на производительность результатов. Стоит также учитывать, что увеличение разрядности чисел, над которыми проводятся логические операции, приводит к росту затрат оперативной памяти. Векторизация также позволяет повысить эффективность кэширования, так как с увеличением разрядности используемых битовых слов растет время вычисления результата одной логической операции (увеличивается время обработки слов, при этом в пересчете на один бит скорость повышается). Время на чтение из памяти остается на прежнем уровне.

3. Экспериментальные результаты

Для тестирования эффективности предлагаемых методов расчет коэффициента надежности проводился на искусственно сгенерированной выборке из 2000 комбинационных схем с числом входов от 5 до 10, числом вентилей от 20 до 40 и числом выходов от 5 до 10. Тестовая выборка получена путем генерации описаний ряда схем, которые затем были синтезированы с помощью библиотеки yosys с последующим удалением схем, не удовлетворяющих количественным ограничениям на число входов, вентилей и выходов, заданным выше.

Описанные выше методы можно комбинировать для получения более высоких показателей производительности, при этом число таких комбинаций достаточно велико. В таблице приведен показатель производительности для наиболее эффективной комбинации методов, полученной в ходе экспериментов. Эта комбинация заключается в векторизации с длиной слов, равной 1024, и обработке всех векторов ошибки за один проход.

Представлены значения ускорения вычислений для рассматриваемых в данной статье методов и наиболее эффективной их комбинации. Указано среднее увеличение производительности вычислений при использовании соответствующего метода и среднее квадратичное отклонение (СКО) на тестовой выборке.

Как видно, наибольшую эффективность при работе с отдельными битовыми значениями демонстрирует метод обработки всех векторов ошибки за один проход. Методы кэширования с разным порядком входных комбинаций (стандартным и кодом Грея) различаются незначительно за счет более сложной вычислительной (в случае кода Грея) процедуры генерации входных комбинаций. Наиболее эффективный способ повышения производительности вычислений в целом — векторизация: ускорение вычислений при ее использовании пропорционально длине применяемых битовых слов. Недостатком этого метода является увеличение затрат необходимой памяти.

Сравнение эффективности предлагаемых методов повышения производительности вычислений

Метод	Ускорение, раз	СКО
Кэширование	1.79	0.08
Кэширование и код Грея	1.80	0.079
Single-pass	9.53	1.29
Векторизация и single-pass	442.72	133.17

Заключение

Рассмотрены практические методы повышения производительности при расчете надежности комбинационных логических схем. На широкой тестовой выборке эмпирически показана эффективность предлагаемых способов ускорения вычислений. Дальнейшая работа будет направлена на получение асимптотических оценок сложности предлагаемых методов, изучение возможности оптимизации работы с памятью для больших схем и поиск оптимальных последовательностей входных и ошибочных комбинаций при использовании кэширования.

Список литературы / References

[1] **Mahatme, N.N., Jagannathan, S., Loveless, T.D., Massengill, L.W., Bhuva, B.L., Wen, S.J. et al.** Comparison of combinational and sequential error rates for a deep submicron process // IEEE Trans. Nucl. Sci. (IEEE T NUCL SCI). 2011. Vol. 58, No. 6. P. 2719–2725.

[2] **Dhillon, Y., Diril, A., Chatterjee, A.** Soft-error tolerance analysis and optimization of nanometer circuits // Automation and Test in Europe. Proceedings. 2005. Vol. 1. P. 288–293.

[3] **Mahatme, N.N., Gaspard, N.J., Assis, T., Jagannathan, S., Chatterjee, I., Loveless, T.D., Bhuva, B.L., Massengill, L.W., Wen, S.J., Wong, R.** Impact of technology scaling on the combinational logic soft error rate // Intern. Reliability Physics Symp. (IRPS), 2014. P. 5F.2.1–5F.2.6.

[4] **Стемпковский А.Л., Тельпухов Д.В., Соловьев Р.А., Мячиков М.В.** Повышение отказоустойчивости логических схем с использованием нестандартных мажоритарных элементов // Информ. технологии. 2015. Т. 21, № 10. С. 749–756.
Stempkovskiy, A.L., Telpukhov, D.V., Solovyev, R.A., Myachikov, M.V. Improving the fault tolerance of logic circuits using unconventional majority voters // Inform. Technology. 2015. Vol. 21, No 10. P. 749–756. (In Russ.)

[5] **Стемпковский А.Л., Тельпухов Д.В., Соловьев Р.А., Мячиков М.В., Тельпухова Н.В.** Разработка технологически независимых метрик для оценки маскирующих свойств логических схем // Вычисл. технологии. 2016. Т. 21, № 2. С. 53–62.
Stempkovskiy, A.L., Telpukhov, D.V., Solovyev, R.A., Myachikov, M.V., Telpukhova, N.V. The development of technology-independent metrics for evaluation of the masking properties of logic // Comput. Technology. 2016. Vol. 21, No. 2. C. 53–62. (In Russ.)

[6] **Von Neumann, J.** Probabilistic logics and the synthesis of reliable organisms from unreliable components // Automata Studies / C.E. Shannon and J. McCarthy. (Eds). Princeton, NJ: Princeton Univ. Press, 1956. P. 43–98.

[7] **Стемпковский А.Л., Тельпухов Д.В., Соловьев Р.А., Соловьев А.Н., Мячиков М.В.** Моделирование возникновения неисправностей для оценки надежностных характеристик логических схем // Информ. технологии. 2014. № 11. С. 30–35.
Stempkovskiy, A.L., Telpukhov, D.V., Solovyev, R.A., Solovyev, A.N., Myachikov, M.V. Fault simulation technique for logic circuits reliability characteristics evaluation // Inform. Technology. 2014. No 11. P. 30–35. (In Russ.)

[8] **Тельпухов Д.В., Соловьев Р.А., Мячиков М.В.** Разработка практических метрик для оценки методов повышения сбоеустойчивости комбинационных схем // Тр. Междунар. науч.-техн. конф. “Информационные технологии и математическое моделирование систем 2015”. Москва, 2015. С. 79–81.
Telpukhov, D.V., Solovyev, R.A., Myachikov, M.V. Development of practical metrics to evaluate the methods for increasing the fault tolerance of combinational circuits // Proc. of the Intern. Scientific and Technical Conf. “Information Technologies and Mathematical Modeling of Systems 2015”. Moscow, 2015. P. 79–81. (In Russ.)

[9] **Ibrahim W., Beiu V., Beg A.** GREDA: A fast and more accurate gate reliability EDA tool // IEEE Trans. Comput. Des. Integr. Circuits Syst. 2012. Vol. 31, No. 4. P. 509–521.

Поступила в редакцию 6 июня 2016 г.

Methods for improvement of computing performance when calculating reliability metrics of combinational logic circuits

Stempkovskiy, Alexander L., Telpukhov, Dmitry V.*, Solovyev, Roman A., Myachikov, Mikhail V.

Institute for Design Problems in Microelectronics RAS, Moscow, 124365, Russia

*Corresponding author: Telpukhov, Dmitry V., e-mail: nofrost@inbox.ru

Purpose. The main objective of this work is to develop computational acceleration methods of fault-tolerance metrics evaluation for combinational circuits. The urgency of this goal stems from the fact that the computational complexity of direct methods for calculating fault tolerance characteristics of combinational circuits depends exponentially on the number of circuit inputs, making them unusable even for medium-sized schemes.

Methodology. The paper presents a number of methods for increasing computing capacity when calculating reliability metrics for combinational logic circuits. Standard methods for optimization of calculations such as caching, caching with the changed order of calculation and vectorization underlie the proposed approaches to speed up computations due to more intensive use of memory.

Findings. Within the framework of the research, it was found that highest efficiency, when working with the individual bit values, shows the single pass method of handling errors. Caching methods with a different order of input combinations, which consists of standard and Gray codes slightly differ due to a more complex Gray code input generation procedure for combinations. The most effective way to increase computing performance as a whole is vectorization: acceleration of computations was proportional to the used bit width. The disadvantage of this method is the requirement for the increased usage of memory.

Originality/value. In this article, the practical methods for performance improvement in the calculation of reliability of combinational logic circuits have been considered. The high efficacy of the proposed acceleration methods has been empirically demonstrated on a broad set of tests.

Keywords: fault-tolerance, combinational circuit, sensitivity factor, caching.

Received 6 June 2016