УДК 621.3.049.771.14 DOI: 10.31114/2078-7707-2021-4-52-58

Алгоритм логического синтеза сбоеустойчивых схем в технологическом базисе

Тельпухов Д.В., Надоленко В.В.

Институт проблем проектирования в микроэлектронике РАН, г. Москва, nofrost@inbox.ru

***Аннотация* — Данная работа посвящена разработке алгоритма синтеза комбинационных логических схем в заданном технологическом базисе с оптимизацией по параметру сбоеустойчивости. Входными данными для алгоритма являются схема в формате AIG и библиотека элементов, выходными – синтезированное Verilog описание схемы. В качестве показателя надежности предлагается использовать одну из нескольких метрик различной степени точности, учитывающих вероятность логического маскирования. В основе предложенного алгоритма лежит классический алгоритм покрытия дерева, адаптированный под оптимизацию предложенных метрик сбоеустойчивости. Учет возможных путей распространения сигналов по схеме для различных покрытий дерева позволяет быстро и точно оценивать целевую функцию, что является ключевым фактором работоспособности алгоритма. Тестирование проводилось на схемах из набора ISCAS'85. AIG для данных схем были получены при помощи инструмента ABC. Также была протестирована эффективность алгоритма сбоеустойчивого ресинтеза на схемах, полученных в ходе данного исследования.**

***Ключевые слова* — случайные сбои, покрытие дерева, логический синтез, ресинтез, сбоеустойчивость, наблюдаемость, ODC.**

I. Введение

Известно, что поток высокоэнергетичных частиц (в основном протонов) способен вызывать различные эффекты в электронных устройствах, приводящие к временным сбоям в их работе [1]. Данные эффекты особенно опасны для оборудования, работающего в условиях интенсивного радиационного облучения: в космических аппаратах, некоторых медицинских устройствах, на атомных электростанциях и т.д. Все эти области требуют разработки радиационно-стойкой электроники.

Традиционные подходы к обеспечению радиационной стойкости включают в себя различные методы на системном уровне [2][3], использование специальных библиотек элементов [4][5], добавление схем функционального контроля [6] и усиление различных механизмов маскирования ошибок [7]. Последнее, как правило, подразумевает внесение локальных изменений в структуру схемы - например, кратное резервирование участков схемы или итеративный ресинтез [8]. Эффективность подобных изменений зависит от начальной структуры.

В данной статье предложен алгоритм логического синтеза комбинационных схем в заданном технологическом базисе, выполняющий оптимизацию по критерию сбоеустойчивости. В главе 2 описан общий подход к логическому синтезу и выполнен обзор существующих работ по теме сбоеустойчивого синтеза. В главе 3 приведены возможные метрики сбоеустойчивости и способ их применения в процессе логического синтеза. Глава 4 содержит предложенный алгоритм и его отличия от существующего аналога. В главе 5 представлены результаты экспериментов, рассмотрена эффективность алгоритма как отдельно, так и в связке с последующим ресинтезом [8]. Глава 6 заключает статью.

II. Логический синтез

Логический синтез выполняется над высокоуровневым поведенческим описанием устройства. В результате получается нетлист логической схемы – список логических элементов и их соединений. Процесс синтеза состоит из двух этапов [9]. Сначала исходное описание представляется в виде абстрактного ориентированного графа, где узлами являются логические функции, а направление ребер противоположно порядку вычислений. Как правило, на данном этапе используется ограниченное множество операций, позволяющее производить оптимизационные преобразования над графом. Например, все функции могут быть представлены в ДНФ. Также используются AIG (And-Inverter Graph) и другие подобные структуры. Второй этап представляет собой покрытие полученного графа логическими элементами из заданной технологической библиотеки. Участки графа последовательно заменяются элементами с соответствующими функциями (см. рис. 1). Процесс завершается, когда граф не содержит абстрактных узлов, т.е. полностью состоит из библиотечных элементов.

Корректная оценка маскирующих свойств будущей логической схемы на первом этапе не представляется возможной, поскольку не определены границы элементов, т.е. не известно, какие литералы в функциях будут связаны с реальным физическим сигналом, подверженным сбоям. Установление этих связей происходит на втором этапе и влияет на вероятность маскирования случайных сбоев в схеме. Таким образом, для повышения сбоеустойчивости схемы при логическом синтезе необходимо обратиться к алгоритмам покрытия.

В общем случае задача оптимального покрытия относится к классу NP. Однако существует линейный оптимальный алгоритм покрытия для древовидных графов, поэтому распространенным приемом является разбиение произвольного графа на деревья. Узлы с количеством входящих ребер 2 и более, а также реализующие непосредственно функцию одного из первичных выходов схемы, становятся корнями деревьев. Входящие ребра для них удаляются, и таким образом граф приобретает форму леса (см. рис. 2). Затем для каждого дерева применяется алгоритм покрытия, удаленные ребра возвращаются и соединяют получившиеся фрагменты схемы.

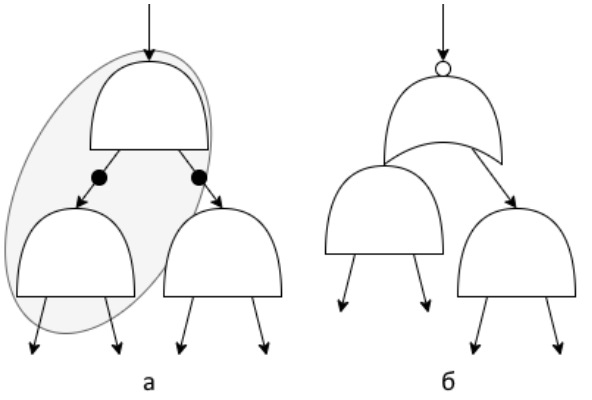

Рис. 1. Пример замены участка AIG библиотечным элементом: а) исходный фрагмент AIG; б) тот же фрагмент с элементом AOI21

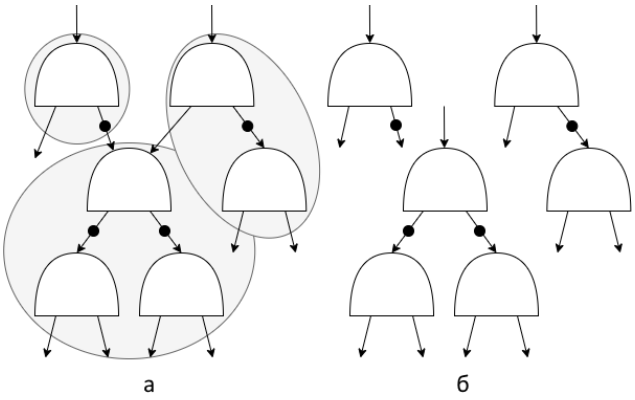

Рис. 2. Пример разбиения AIG на деревья: а) исходный фрагмент AIG; б) разбиение на древовидные участки

```
V = tree vertices
sort V topologically from leafs to top
set leaf costs to 0

for each v_i in V:
  min cost = +inf
  for each match m_k at v_i:
    match cost = cell cost + sum of input costs
    if match cost < min cost:
      min cost = match cost
      remember m_k and match cost for v_i
```

Рис. 3. Алгоритм оптимального покрытия дерева

На рис. 3 приведен алгоритм покрытия дерева (блоки циклов и условий обозначены отступами, как в языке python). Для каждого узла производится перебор подходящих элементов и вычисляются значения целевой функции, наилучший вариант запоминается. При этом учитывается также стоимость входного конуса каждого элемента, которая всегда известна за счет обхода узлов в топологическом порядке от листьев к корню. Важно отметить, что функции логических элементов также должны быть представлены в виде деревьев в соответствующем базисе для их совмещения с узлами дерева-цели. Например, при покрытии AIG невозможно использовать элемент xor, поскольку он не представим в виде дерева в базисе операций and и inv. Имеется ограничение и для целевых функций: необходимо свойство аддитивности, т.е. каждый отдельный элемент должен вносить свой вклад, не зависящий от других элементов. Примером такой функции является суммарная площадь элементов. В [10] также представлена модификация алгоритма для оптимизации по задержке на критическом пути, позволяющая учитывать зависимость вклада элемента от емкостной нагрузки на его выходе. В [11] было предложено применение алгоритма покрытия дерева с оптимизацией по критерию сбоеустойчивости.

Данное исследование также опирается на идею покрытия деревьев, однако имеет несколько существенных отличий от [11]. Для оценки сбоеустойчивости предлагается использовать одну из трех целевых функций в зависимости от имеющихся входных данных. При определении стоимости элемента учитываются маскирующие свойства его выходного конуса вплоть до первичных выходов схемы, что позволяет дать более точную оценку. И наконец, рассматривается проблема возможного наличия у деревьев одновременно прямого и инверсного выходов и предлагается гибкое решение.

III. Оценка сбоеустойчивости

Сбоеустойчивость в широком смысле означает стойкость устройства к внешним воздействиям и достигается различными методами на различных уровнях проектирования, как было упомянуто в главе 1. Данное исследование сосредоточено на структурной оптимизации комбинационной логики, поэтому здесь под сбоеустойчивостью схемы понимается ее способность корректно функционировать при возникновении случайных сбоев – инверсий сигнала на каких-либо узлах. Такие сбои имеют практическое значение в том случае, когда они изменяют состояние элементов памяти, однако этому мешают логическое, электрическое и временное маскирование. Вероятности срабатывания различных механизмов маскирования имеют слабую корреляцию и потому могут оцениваться отдельно [7]. В то же время логическое маскирование подвержено наибольшему влиянию со стороны структуры, поэтому представленные метрики сбоеустойчивости сосредоточены на нем.

Вероятность логического маскирования характеризуется наблюдаемостью элемента, на котором произошел сбой:

$$P_{LM}(G_i) = 1 - O(G_i)\,, \quad (1)$$

где P_{LM} – вероятность логического маскирования,

O – наблюдаемость,

G_i – i-ый элемент схемы.

Формула (1) следует из определения: наблюдаемость элемента – это вероятность того, что изменение состояния данного элемента в случайный момент приведет к изменению состояния выходов схемы. При некоторой малой вероятности сбоя ε на каждом элементе схемы и в предположении отсутствия многократных сбоев получим:

$$P_{LM} = 1 - \varepsilon \sum_i O(G_i) \quad (2)$$

В отсутствие какой-либо технологической информации об элементах библиотеки для оценки надежности логической схемы может быть использован коэффициент чувствительности (sensitivity factor) [12]:

$$SF = \sum_i O(G_i) \quad (3)$$

Как видно из формулы (2), минимизация коэффициента чувствительности увеличивает вероятность логического маскирования, делая схему более надежной.

Информация о площадях библиотечных ячеек позволяет применить немного более точную метрику – чувствительную площадь (sensitive area):

$$SA = \sum_i A(G_i) * O(G_i)\,, \quad (4)$$

где A – площадь элемента.

Наиболее точной на данный момент метрикой, совместимой с предложенным алгоритмом, является эффективная чувствительная площадь [13]:

$$ESA = \sum_i \left[A(G_i) \sum_j O\left(G_i \middle| V_j\right) * P(V_j)\right], \quad (5)$$

где V_j – j-ый входной вектор элемента G_i, $P(V_j)$ – его вероятность, получаемая при характеризации.

Метрика (5) требует не только технологической информации о библиотечных ячейках, но и их предварительной характеризации на предмет устойчивости к дестабилизирующим воздействиям. В [13] показано, что метрики (3), (4) и (5) имеют высокий коэффициент корреляции, поэтому при отсутствии необходимых данных или ресурсов допустимо использовать менее точные формулы.

Основной задачей при оценке сбоеустойчивости по метрикам (3)-(5) становится вычисление наблюдаемостей элементов. Для ее выполнения в данном исследовании использовалось бит-параллельное Монте-Карло моделирование и симуляция инверсной ошибки [14]. Моделирование инверсной ошибки может быть выполнено для AIG так же, как и для схемы на основе логических элементов. Особенность AIG заключается в том, что для каждого узла существует его инверсная пара. В графе логической схемы, полученном после покрытия, узлы для прямого и инверсного сигнала могут существовать одновременно. Для определения наблюдаемостей обоих узлов необходимо распространить инверсную ошибку через их выходные конусы. Однако алгоритм покрытия на любом этапе может использовать инверсный узел вместо прямого (или наоборот) и тем самым поменять пути распространения сигнала в схеме. Данная проблема не возникает для узлов в древовидных участках AIG, поскольку каждому из них может соответствовать только один узел в выходном нетлисте, и его выходной конус не зависит от полярности. Поэтому покрытие отдельного дерева можно выполнять в прямом топологическом порядке, а обход деревьев в схеме – только в обратном.

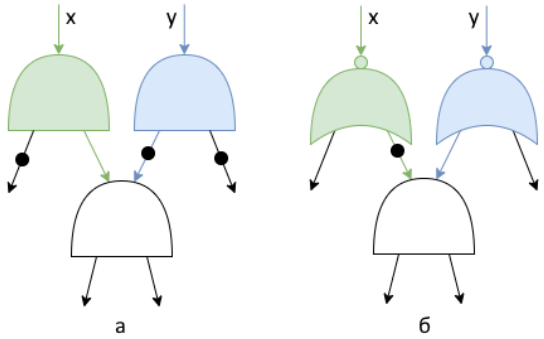

Рис. 4. Влияние покрытия на пути распространения сигнала: а) исходный фрагмент AIG, прямой сигнал идет по пути x, инвертированный – по пути y; б) частичное покрытие элементами NOR2, пути прямого и инверсного сигнала поменялись

IV. Алгоритм покрытия

Двойственность узлов AIG приводит также к невозможности напрямую применять алгоритм с рис. 3, поскольку один корень дерева AIG после покрытия может превратиться в два узла. Тогда соответствующая подсхема деревом являться не будет. Простым решением было бы выполнять покрытие только для одного выхода, получая второй с помощью инвертора. Однако в таком случае сбой в любом элементе дерева может распространяться через оба его выхода. С точки зрения сбоеустойчивости такое решение с высокой вероятностью не оптимально. Возможен и противоположный подход: выполнить полное покрытие для прямого и инверсного выходов и включить в схему оба дерева. Наблюдаемость отдельных элементов таким образом сводится к минимуму, но их удвоенное количество может негативно отразиться на чувствительности схемы к сбоям. Такое использование дополнительной площади избыточно и неэффективно.

Предлагаемое решение проблемы заключается в условном разбиении целевого AIG на общую и раздельную части (рис. 5). Раздельная часть – поддерево, корень которого совпадает с корнем целевого AIG. Для нее выполняются два покрытия,

реализующие прямую и инверсную функции. Общая часть содержит оставшиеся узлы целевого AIG и покрывается один раз. Одна из частей может отсутствовать, и тогда реализуется один из описанных выше крайних случаев – полное объединение или разделение покрытий.

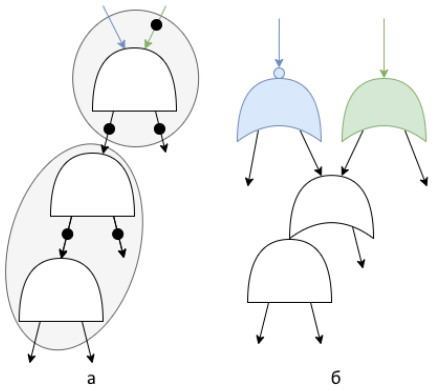

Рис. 5. Пример покрытия дерева AIG с учетом прямого и инверсного выхода: а) исходный фрагмент AIG; б) покрытие с общей частью и индивидуальными частями для выходов

Более широкое поле поиска дает доступ к лучшим решениям, но в то же время усложняет саму задачу оптимизации. Количество подобных разбиений дерева растет экспоненциально с увеличением его размера, делая полный перебор непрактичным. На рис. 6 представлен эвристический итеративный алгоритм, сочетающий покрытие дерева с поиском разбиения. На подготовительном этапе вычисляются наблюдаемости всех узлов целевого дерева. В зависимости от покрытия сигнал с каждого узла может распространяться через прямой и/или инверсный выход, поэтому существует три возможных значения наблюдаемости. Для корня дерева эти значения вычисляются с помощью моделирования инверсной ошибки, а для остальных узлов – методом обратного распространения, что позволяет ускорить работу алгоритма, не теряя точности. Более подробно этот метод раскрывается в [8], где он применяется на участках логической схемы. В основном цикле поочередно выполняется покрытие для прямого и инверсного выходов. При этом сохраняется информация о «противоположном» покрытии. Его узлы изначально считаются общими, и стоимость их использования приравнивается к 0. Стоимость добавления к покрытию новых узлов рассчитывается так же, как в базовом алгоритме (рис. 3). Нововведением является стоимость разделения. Этот показатель рассчитывается для неиспользованных узлов «противоположного» покрытия. Данные узлы перестают быть общими, их наблюдаемость меняется, а вместе с ней меняется и стоимость соответствующих элементов покрытия. Именно стоимость разделения позволяет алгоритму находить оптимальную общую часть. Однако решение оптимально только для заданного «противоположного» покрытия, которое в общем случае не содержит все узлы исходного дерева и потому сокращает поле поиска. Для рассмотрения большего количества вариантов алгоритм выполняется итеративно, пока он вносит в покрытие какие-либо изменения.

```
V = tree vertices
sort V topologically from top to leafs
t = tree top
t* = ~t
for each vi in V:
  compute O(vi | vi -> t)
  compute O(vi | vi -> t*)
  compute O(vi | vi -> t,t*)

sort V topologically from leafs to top
C = ∅
C* = ∅

repeat
  swap C, C*
  swap t, t*
  C0 = C
  clear C
  for each vi in V set cost to +inf
  set leaf costs to 0
  for each vi in C* set cost to 0

  for each vi in V \ t*:
    for each match mk at vi:
      separation cost = 0
      for each vj covered by mk:
        if vj in C*:
          separation cost += (vj cell cost | vj -> t*) -
                             (vj cell cost | vj -> t,t*)
      match cost = (vi cell cost | vi -> t) +
                   sum of input costs +
                   separation cost
      if match cost < vi cost:
        remember mk and match cost for vi
        add vi to C
until C == C0
```

Рис. 6. Алгоритм покрытия дерева AIG для прямого и инверсного выхода

V. Экспериментальные результаты

Предложенный алгоритм был протестирован на схемах из набора ISCAS'85 и технологической библиотеке Nangate Open Cell Library 45 nm без элементов, не представимых в виде деревьев AIG (xor и mux). Первый этап синтеза и создание AIG были выполнены с помощью программы ABC. Результаты работы алгоритма сравнивались с результатами команды map в ABC [15]. Однако особый интерес представляет другое сравнение. Идея разбиения исходного AIG на деревья сама по себе существенно ограничивает множество решений и отдаляет возможный результат от оптимального. Главная ценность предложенного алгоритма заключается в способности обеспечить лучшее начальное приближение для итеративной оптимизации на логическом уровне. Поэтому ко всем схемам также был применен алгоритм ресинтеза на основе локальных замен [8]. В качестве метрики сбоеустойчивости был

выбран коэффициент чувствительности (3) как не требующий дополнительной технологической информации. В таблице 1 представлены значения коэффициента чувствительности схем с применением предложенного алгоритма покрытия (reliable map) и стандартного (ABC map), до и после ресинтеза. Некоторые схемы пропущены ввиду того, что слишком малый размер древовидных участков в них не позволяет увидеть какую-либо разницу между алгоритмами покрытия.

Таблица 1

Коэффициент чувствительности некоторых схем из набора ISCAS'85 после применения различных алгоритмов покрытия и ресинтеза.

Схема	До ресинтеза			После ресинтеза		
	ABC map	Reliable map	Δ, %	ABC map	Reliable map	Δ, %
c1908	181.58	157.88	-13.05	134.07	106.84	-20.31
c3540	270.23	231.49	-14.34	211.14	196.29	-7.04
c5315	512.51	472.11	-7.88	404.95	403.61	-0.33
c6288	1870.16	1586.53	-15.17	1837.84	1406.57	-23.47
c7552	713.93	581.86	-18.50	599.00	520.71	-13.07
c880	115.37	108.63	-5.84	84.66	83.14	-1.79

Как видно из таблицы 1, во многих случаях эффект от оптимизации по сбоеустойчивости на этапе покрытия сохраняется и после итеративного ресинтеза. Таким образом, подтверждается предположение, что для ресинтеза имеет значение в том числе и надежность исходной схемы. Также стоит отметить, что при ресинтезе выполнялось 600 попыток локальных замен. Для схем размером от сотен до нескольких тысяч элементов этого достаточно, чтобы приблизиться к локальному минимуму коэффициента чувствительности, т.е. большее число замен не оказало бы существенного влияния на представленные результаты.

Однако оригинальные схемы из набора ISCAS'85 содержат значительную функционально избыточную часть, не способствующую логическому маскированию. Сокращения избыточности уже достаточно для повышения их сбоеустойчивости. Поэтому для получения более показательных результатов были проведены еще два эксперимента. Их отличие состоит в том, что к исходным схемам на этапе синтеза в ABC применялись скрипты resyn2rs и compress2rs [16]. Как видно из таблиц 2 и 3, коэффициент чувствительности исходных схем действительно уменьшился, однако преимущество сбоеустойчивого покрытия также сохранилось.

Таблица 2

Коэффициент чувствительности некоторых схем из набора ISCAS'85, синтезированных скриптом resyn2rs, после применения различных алгоритмов покрытия и ресинтеза.

Схема	До ресинтеза			После ресинтеза		
	ABC map	Reliable map	Δ, %	ABC map	Reliable map	Δ, %
c1355	143.37	138.49	-3.40	128.03	112.69	-11.98
c1908	164.96	158.62	-3.84	137.01	102.13	-25.46
c3540	241.82	219.85	-9.09	212.39	191.22	-9.97
c499	142.85	139.62	-2.26	129.48	114.76	-11.37
c5315	469.69	412.83	-12.11	353.40	359.40	1.70
c7552	576.11	487.83	-15.32	483.67	446.30	-7.73
c880	126.37	113.21	-10.41	86.39	83.40	-3.47

Таблица 3

Коэффициент чувствительности некоторых схем из набора ISCAS'85, синтезированных скриптом compress2rs, после применения различных алгоритмов покрытия и ресинтеза.

Схема	До ресинтеза			После ресинтеза		
	ABC map	Reliable map	Δ, %	ABC map	Reliable map	Δ, %
c1355	145.03	138.85	-4.26	121.51	108.68	-10.56
c1908	176.50	159.50	-9.63	126.99	104.20	-17.94
c3540	235.06	217.10	-7.64	203.21	186.34	-8.30
c499	140.57	138.10	-1.76	124.70	115.88	-7.07
c5315	477.87	414.14	-13.34	392.08	377.36	-3.76
c7552	563.82	477.81	-15.25	457.28	439.16	-3.96
c880	129.72	113.30	-12.66	82.91	82.77	-0.17

VI. Заключение

Предложенный алгоритм позволяет усиливать логическое маскирование в комбинационных схемах на этапе синтеза. Основным его преимуществом является предоставление лучшей исходной схемы для итеративных алгоритмов ресинтеза логических схем, целевая функция которых связана со сбоеустойчивостью. Главный же недостаток – невозможность применять элементы xor и mux – отчасти компенсируется последующим ресинтезом, не имеющим подобных ограничений. Полное устранение данного недостатка является наиболее перспективным направлением развития темы. Также возможно обобщение алгоритма покрытия на произвольные схемы с несколькими выходами, не связанными отношением инверсии.

Проект выполнен при финансовой поддержке гранта РНФ №17-19-01645.

Литература

[1] Gaillard, Rémi. Single Event Effects: Mechanisms and Classification // Soft Errors in Modern Electronic Systems. 2010. P. 27-54.

[2] Ivannikov A.D., Levchenko N.N., Okunev A.S., Stempkovsky A.L., Zmejev D.N. Dataflow Computing Model - Perspectives, Advantages and Implementation // Proceedings of 2017 IEEE East-West Design and Test Symposium, EWDTS 2017.

[3] Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Использование модели вычислений с управлением потоком данных и реализующей ее архитектуры для систем экзафлопсного уровня производительности // Проблемы разработки перспективных микро- и наноэлектронных систем - 2012. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2012. С. 459-462.

[4] G. Anelli et al. Radiation tolerant VLSI circuits in standard deep submicron CMOS technologies for the LHC experiments: practical design aspects // IEEE Transactions on Nuclear Science. Dec. 1999. Vol. 46. №6. P. 1690-1696.

[5] Singh, Rahul Kumar. Silicon on Insulator Technology Review // International Journal of Engineering Sciences & Emerging Technologies. 2011. Vol. 1.

[6] Гаврилов С.В., Гуров С.И., Жукова Т.Д., Рухлов В.С., Рыжова Д.И., Тельпухов Д.В. Методы повышения сбоеустойчивости комбинационных ИМС на основе избыточного кодирования // В сборнике: Прикладная математика и информатика. труды факультета ВМК МГУ имени М.В. Ломоносова. Москва, 2016. С. 93-102.

[7] B. Ghavami, M. Raji. Introduction: Soft Error Modeling // Soft Error Reliability of VLSI Circuits: Analysis and Mitigation Techniques. 2021. P. 1-7.

[8] Надоленко В.В., Тельпухов Д.В., Битков Ю.В. Разработка маршрута ресинтеза комбинационных логических схем с целью повышения маскирующих свойств // Проблемы разработки перспективных микро- и наноэлектронных систем(МЭС). 2018. Вып. 1. С. 50-56.

[9] Thomas, Donald E., Moorby, Philip R. Logic Synthesis // The Verilog® Hardware Description Language. 2002. P. 35-71.

[10] Herve J. Touati. Performance-Oriented Technology Mapping. 1990.

[11] Wo, Zhaojun & Koren, Israel. Technology mapping for reliability enhancement in logic synthesis. // ISQED 2005. P. 137-142.

[12] 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 // Вычислительные технологии. 2016. Т. 21. № 2. С. 53-62

[13] Stempkovskiy A.L., Telpukhov D.V., Solovyev R.A. and Nadolenko V.V. Increasing the Accuracy of Reliability-aware Resynthesis with Standard Cell Reliability Characterization // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). P. 2035-2039.

[14] Стемпковский А.Л., Тельпухов Д.В., Соловьев Р.А., Мячиков М.В. Методы повышения производительности вычислений при расчете метрик надежности комбинационных логических схем // Вычислительные технологии. 2016. Т. 21. № 6.

[15] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/

[16] A. Mishchenko, R. K. Brayton. Scalable logic synthesis using a simple circuit structure // Proc. IWLS '06. P. 15-22.

Reliability-Driven Logic Synthesis Using Arbitrary Standard Cell Library

Telpukhov D.V., Nadolenko V.V.

Institute for Design Problems in Microelectronics of RAS, Moscow, nofrost@inbox.ru

***Abstract* — This paper presents logic synthesis algorithm with reliability optimization using arbitrary standard cell library. It takes a circuit in AIG format and cell library as input and returns synthesized Verilog netlist. Several reliability metrics to drive the algorithm are presented. These metrics take into account logical masking mechanism and differ by their precision levels. Presented algorithm is based on tree covering modified to optimize reliability characteristics. Tracking possible signal paths dynamically enables fast and precise goal function evaluation which is crucial for scalability. ISCAS'85 benchmark circuits were used for algorithm testing. AIGs for those circuits were generated via ABC tool. Finally, test circuits were used as input for resynthesis algorithm improving reliability. Experimental results show that modifying logic synthesis step favors resynthesis efficiency.**

***Keywords* — SET, tree covering, logic synthesis, resynthesis, reliability, observability, ODC.**

The project was carried out with the financial support of the Russian Science Foundation grant No. 17-19-01645.

REFERENCES

[1] Gaillard, Rémi. Single Event Effects: Mechanisms and Classification // Soft Errors in Modern Electronic Systems. 2010. P. 27-54.

[2] Ivannikov A.D., Levchenko N.N., Okunev A.S., Stempkovsky A.L., Zmejev D.N. Dataflow Computing Model - Perspectives, Advantages and Implementation // Proceedings of 2017 IEEE East-West Design and Test Symposium, EWDTS 2017.

[3] Levchenko N.N., Okunev A.S., Stempkovsky A.L. The usage of dataflow computing model and architecture realizing these for exaflops performance system // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2012. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2012. P. 459-462.

[4] G. Anelli et al. Radiation tolerant VLSI circuits in standard deep submicron CMOS technologies for the LHC experiments: practical design aspects // IEEE Transactions on Nuclear Science. Dec. 1999. Vol. 46. №6. P. 1690-1696.

[5] Singh, Rahul Kumar. Silicon on Insulator Technology Review // International Journal of Engineering Sciences & Emerging Technologies. 2011. Vol. 1.

[6] Gavrilov S.V., Gurov S.I., Zhukova T.D., Rukhlov V.S., Ryzhova D.I., Telpukhov D.V. Metody povysheniya sboeustojchivosti kombinacionnyh IMS na osnove izbytochnogo kodirovaniya // In: PRIKLADNAYA MATEMATIKA I INFORMATIKA. trudy fakul'teta VMK MGU imeni M.V. Lomonosova. Moscow, 2016. P. 93-102.

[7] B. Ghavami, M. Raji. Introduction: Soft Error Modeling // Soft Error Reliability of VLSI Circuits: Analysis and Mitigation Techniques. 2021. P. 1-7.

[8] Nadolenko V.V., Telpukhov D.V., Bitkov U. Development of Resynthesis Flow for Improving Logical Masking Features of Combinational Circuits // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2018. Issue 1. P. 50-56. doi:10.31114/2078-7707-2018-1-50-56

[9] Thomas, Donald E., Moorby, Philip R. Logic Synthesis // The Verilog® Hardware Description Language. 2002. P. 35-71.

[10] Herve J. Touati. Performance-Oriented Technology Mapping. 1990.

[11] Wo, Zhaojun & Koren, Israel. Technology mapping for reliability enhancement in logic synthesis. // ISQED 2005. P. 137-142.

[12] Stempkovskiy A.L., Telpukhov D.V., Solov'ev R.A., Myachikov M.V., Telpukhova N.V. Razrabotka tehnologicheski-nezavisimyh metrik dlja ocenki maskirujushhih svojstv logicheskih shem // Vychislitel'nye tehnologii. 2016. Vol. 21. № 2. P. 53-62.

[13] Stempkovskiy A.L., Telpukhov D.V., Solovyev R.A. and Nadolenko V.V. Increasing the Accuracy of Reliability-aware Resynthesis with Standard Cell Reliability Characterization // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). P. 2035-2039.

[14] Stempkovskiy A.L., Telpukhov D.V., Solov'ev R.A., Myachikov M.V. Metody povyshenija proizvoditel'nosti vychislenij pri raschete metrik nadezhnosti kombinacionnyh logicheskih shem // Vychislitel'nye tehnologii. 2016. Vol. 21. № 6.

[15] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/

[16] A. Mishchenko, R. K. Brayton. Scalable logic synthesis using a simple circuit structure // Proc. IWLS '06. P. 15-22.