УДК 681.518

© 2009 г. [1]Д.Е. Иванов, [2]Р. Зуауи

1Институт прикладной математики и механики НАН Украины, Донецк
2Донецкий национальный технический университет, Донецк

ВЕРИФИКАЦИЯ ЭКВИВАЛЕНТНОСТИ ЦИФРОВЫХ СХЕМ С ИСПОЛЬЗОВАНИЕМ СТРАТЕГИИ СИМУЛЯЦИИ ОТЖИГА

В статье предложен новый алгоритм верификации эквивалентности цифровых последовательностных схем. Он основан на эволюционной стратегии симуляции отжига, которая использует итеративное улучшение свойств одной входной последовательности. Для оценки качества строящихся решений используется исправное моделирование цифровых схем. Эффективность предлагаемого алгоритма показана путём апробации на схемах из каталога ISCAS-89.

In this paper new algorithm for the verification of the equivalence of the sequential digital circuits is proposed. It is based on the new evolutionary strategy of the simulating annealing. This approach uses an iterative improvement of the properties of the one input sequence. Fault-free simulation of the digital circuits is used for the estimating the quality of the potential solutions. The effectiveness of the proposed algorithm is shown by its approbation on the ISCAS-89 benchmarks.

1. Введение

Разработка цифровых схем является сложным итеративным процессом. На каждой итерации схема проходит ряд преобразований как между уровнями представления (функциональный, логический и т.д.), так и внутри одного уровня, когда происходит оптимизация различных параметров: число логических вентилей, рассеиваемая тепловая энергия и т.п. При таких преобразованиях необходимо следить за тем, чтобы не изменилось эталонное функционирование схемы. Это достигается путём сравнения поведения начальной и преобразованной схем. Таким образом, задача проверки эквивалентности является актуальной в жизненном цикле разработки последовательностных цифровых схем.

Построение входных последовательностей с различными диагностическими свойствами является предметом изучения не одной группы авторов. При этом разработан целый ряд подходов для решения задач подобного типа. Для комбинационных и небольших последовательностных схем хорошо подходят методы, основанные на построении деревьев обходов [1]. К сожалению, с увеличением проектируемых схем резко падает их эффективность, что связано с экспоненциальным ростом размера таких деревьев и, как следствие, приводит к переполнению стеков возвратов поисковых процедур. Ко второй группе относятся методы, основанные на символьном преобразовании представления обрабатываемых схем [2]. Практическое применение данных алгоритмов также ограничено схемами небольшой и средней размерности из-за аналогичных ограничений. Третья группа методов [3] преодолевает эти недостатки, поскольку берёт необходимую для построения последовательностей информацию непосредственно из процедур моделирования. Поскольку задача моделирования решена для больших схем (в том числе последовательностных) [4], то и практическое применение данных алгоритмов шире. К данной группе, в частности, относятся подходы, основанные на генетических алгоритмах [5-6].

Авторы также уделяли огромное внимание эволюционной стратегии генетических алгоритмов. На её основе разработан ряд алгоритмов построения идентифицирующих последовательностей: тестовых [7], инициализирующих [8] и верифицирующих эквивалентность двух заданных схем [9]. Однако вне внимания осталась другая эволюционная оптимизирующая стратегия - симуляции отжига [10]. Её коренное отличие от генетических алгоритмов заключается в том, что здесь происходит эволюция только одного потенциального решения. В связи с этим отпадает механизм обработки популяций, который является существенной частью генетических алгоритмов. Таким образом, открывается возможность построения принци-

пиально более простых алгоритмов идентификации цифровых схем, которые используют стратегию симуляции отжига. Под упрощением построения алгоритмов также понимается вся связанная с их программной реализацией часть: кодирование и отладка.

В данной статье авторы впервые предлагают алгоритм построения верифицирующих последовательностей цифровых схем, который основан на стратегии симуляции отжига. Предложенный алгоритм реализован программно и проведена его апробация на схемах из международного каталога ISCAS-89, которая показывает лучшую эффективность в сравнении с другими алгоритмами.

Данная статья имеет следующую структуру. Во втором разделе в общем виде будет описан алгоритм симуляции отжига. В третьем разделе будет дана формальная постановка задачи верификации эквивалентности двух заданных цифровых схем и описаны компоненты решающего её алгоритма, который основан на стратегии симуляции отжига; приведены результаты машинных экспериментов и сравнения с другими алгоритмами, решающими данную задачу. В четвёртом разделе будет описана модификация предложенного алгоритма для двухядерных рабочих станций, которая позволяет существенно ускорить его работу. В заключении будут сделаны выводы и отмечены направления возможных дальнейших исследований.

2. Стратегия симуляции отжига

Стратегия симуляции отжига впервые разработана Метрополисом (Metropolis N.) в [10] на заре численных компьютерных экспериментов. Она предложена для статистической задачи нахождения состояния группы атомов при остывании нагретого слитка металла. И лишь спустя три десятилетия Кирпатрик (Kirpatrick S.) в [11] применил аналогичный подход к задачам дискретной оптимизации, в частности, к задаче трассировки печатных плат. Таким образом, в данной статье было показано, что данный подход является оптимизирующей стратегией в широком смысле. Практически одновременно появилась статья Керни (Cerny V.) [12], показывающая применение стратегии к решению задачи коммивояжера. С тех пор интерес к данной стратегии неукоснительно растёт и множится число её применений к практическим задачам.

Термины стратегии, как отмечено выше, были заимствованы из задач металлургии: отжиг, распределение температур, конфигурация, скорость охлаждения и т.д. Из названия видно, что данный метод поиска моделирует процесс восстановления. Под восстановлением понимается некоторый физический процесс, который состоит из нагрева и последующего медленного охлаждения исследуемой субстанции. Скорость охлаждения должна быть небольшой и подобрана таким образом, чтобы гарантировать построение прочной кристаллической структуры, в противоположность структуре с дефектами, которая возникает при быстром охлаждении. В задачах оптимизации структура представляет собой закодированное потенциальное решение задачи. А метафора с температурой используется для определения: как и когда следует принимать новые изменённые решения.

Введём некоторые базовые понятия, которые необходимы для описания стратегии симуляции отжига. Данная стратегия представляет собой итеративный алгоритм улучшения свойств некоторого потенциального решения - конфигурации, которое для шага i алгоритма обозначается K_i. Таким образом, конфигурация K_i есть закодированное потенциальное решение задачи в пространстве поиска. С каждой конфигурацией K_i соотносится функция, которая показывает качество данной точки с точки зрения решения задачи, и называется функцией стоимости: $C_i = C(K_i)$. Также дополнительно вводятся возмущающие операции, которые позволяют для точки K_i строить окружение – некоторое множество точек с изменёнными характеристиками. Вычисление функций оценки для точек из окружения может показать, что их качество ухудшилось по сравнению с исходной точкой. Принимать такие ухудшения или отклонять показывает распределение температур $\{T_i\}$. Смысл построения данного распределения заключается в том, что при больших температурах вероятность принять худшие решения выше, чем при более низких. Выбор распределения температур $\{T_i\}$ показывает скорость остывания субстанции и, таким образом, существенно влияет результаты поиска.

Дадим словесное описание алгоритма симуляции отжига.

1. Алгоритм начинает работу с построения начальной конфигурации $K_i = K_1$ и её оценки $C_1 = C(K_1)$. Также для выбранного распределения температур выбирается начальная $T_i = T_1$. Следующие шаги алгоритма повторяются итеративно вплоть до нахождения решения на одном из них, либо до достижения верхней границы числа итераций.

2. Для текущей температуры T_i путём применения модифицирующих операций к текущей конфигурации K_i строится её окружение, состоящее из N конфигураций: $O_i^n = O(K_i) = \{K_i^1, K_i^2, ..., K_i^N\}$

3. Для каждой конфигурации K_i^j из окружения O_i^N вычисляется её оценка $C_i^j = C(K_i^j)$, а также параметр улучшения оценки $\Delta C_i^j = C_i^j - C_i$, на основании которого происходит/не происходит замещение текущей конфигурации K_i:

$$K_{i+1} = \begin{cases} K_i^j, \textit{если } \Delta C_i^j < 0; \\ K_i^j \ \textit{с вероятностью} \\ \quad P = \exp(-\Delta C_i^j / kT_i), \ \textit{если } \Delta C_i^j > 0; \end{cases}$$

Изменяется счётчик итераций $i = i + 1$, а также в соответствии с выбранным распределением температур изменяется текущая температура: $T_{i+1} = \textit{обновить}(T_i)$.

4. Переход к шагу 2.

Видно, что в структуре алгоритма присутствует большое число эвристик, от значений которых существенно зависит эффективность адаптации алгоритма к конкретной задаче: начальная и конечная температуры, скорость изменения температур, распределение вероятностей принятия отрицательных изменений, способ построения кодирования конфигураций и вычисления их оценок. Именно задание кодировки конфигурации и указанных параметров означает построение алгоритма симуляции отжига для решения некоторой практической задачи.

3. Алгоритм верификации эквивалентности и экспериментальные данные

Сформулируем более формально постановку задачи верификации эквивалентности двух заданных схем.

Определение 1. Пусть заданы две цифровые последовательностные схемы A_0 и A_1. Будем называть схему A_0 исправной, а схему A_1 - модифицированной (оптимизированной, неисправной). Тогда схемы называются последовательностно эквивалентными (или просто эквивалентными), если при произвольном одинаковом начальном состоянии выходные реакции данных схем на произвольные входные последовательности s являются одинаковыми, т.е.

$$A_0 = A_1 \ \Leftrightarrow \ \forall s : \quad A_0(s) = A_1(s).$$

При этом, исходя из практического смысла, считается, что схема A_1 получена в результате некоторой оптимизации внутренней структуры схемы A_0. Другими словами структура их элементов состояний одинакова, тогда как логические структуры, описывающие комбинационные блоки, различны.

Задача теоретического доказательства эквивалентности схем в такой постановке является очень трудной, а для практически проектируемых схем невыполнимой. Однако разработчику часто нет необходимости знать, что схемы эквивалентны. Для оценки правильности этапа оптимизации ему достаточно знать, что схемы являются неэквивалентными, т.е. в процессе оптимизации произошёл сбой.

Поэтому, данную задачу построения входных последовательностей, как и другие, которые основаны на недетерминированных подходах, возможно переформулировать. Для этого изменим её на противоположную: мы не будем доказывать эквивалентность заданных схем, а будем строить последовательность, которая будет стараться различить их поведение. Таким образом, алгоритм будет показывать неэквивалентность двух схем. Сформулируем задачу в такой постановке.

Определение 2. Если существует такая последовательность s_k, на которой при одинаковых начальных состояниях схемы A_0 и A_1 имеют различные выходные реакции, то данные схемы не являются эквивалентными.

$$A_0 \neq A_1 \ \textit{если} \ \exists \ s_k : A_0(s_k) \neq A_1(s_k)$$

Алгоритм симуляции отжига для решения задачи верификации эквивалентности последовательностных схем строится на основании общего алгоритма (раздел 2) путём определения его компонент: кодирование конфигураций, определения операций возмущения и вероятностей их применения,

роятностей их применения, построения последовательности температур и на его основе распределения принятия отрицательных решений. Авторы уже описывали подобный алгоритм симуляции отжига, который строил инициализирующие последовательности последовательностных схем [13]. Предлагаемый алгоритм очень схож с описанным, поэтому мы дадим только его краткое описание, больше внимания уделив численным экспериментам и его оптимизации.

В качестве модели алгоритм использует синхронные последовательностные схемы с дальнейшим их преобразованием в псевдокомбинационный эквивалент [14].

В описываемом алгоритме в качестве конфигурации выступает двоичная входная последовательность $K_i = s_i$, для которой заранее не известна оптимальная длина. Ширина входной последовательности равна числу внешних входов рассматриваемых схем. Кодирование конфигураций соответствует кодированию особей в ГА построения тестов [7].

Функция оценки традиционно для алгоритмов данного типа строится на основании результатов моделирования. Поскольку цель алгоритма - построить последовательность, которая должна показать различие в функционировании схем, то оценка должна основываться на различии поведения внутри сравниваемых схем.

Общий вид функции оценки для конфигурации $K_i = s_i$ и двух заданных схем A_0 и A_1:

$$C(K_i) = f(A_0, A_1, s_i) = \sum_{j=1}^{\text{длина } s_i} f(A_1, A_0, s_{ij}) =$$

$$= \sum_{j=1}^{\text{длина } s_i} (c_1 \cdot n_1 + c_2 \cdot n_2 + c_3 \cdot n_3),$$

где $c_1 - c_3$ - некоторые нормирующие константы, n_1 - число различий на внешних выходах сравниваемых схем, n_2 - число различий на псевдовыходах схем (элементах состояний), n_3 - число различий на множестве контрольных точек. Подробно построение данной функции оценки описано в [9].

Значения эвристических параметров, например, расписание температур, подбираются экспериментальным путём.

Сконструированный таким образом алгоритм построения входных последовательностей реализован программно в среде программирования CodeGear (студенческая версия). Объём программного кода составил около 1200 строк, включая событийный алгоритм исправного моделирования работы последовательностных схем.

Для проведения машинных экспериментов с программной реализацией выбрана стратегия, схожая с [9]. Для проверки эффективности алгоритма следует проводить эксперименты на схемах, в которых произведены минимальные изменения. Причём считается, что последовательностная структура (структура элементов состояний) является одинаковой в сравниваемых схемах. А различными являются комбинационные блоки, реализующие функции состояний и переходов. Для того, чтобы усложнить работу алгоритма, мы будем вносить некоторые миноритарные изменения. При построении двух последовательностных схем с очень близкой логической структурой мы пользовались следующими умозаключениями. Оптимизатор вносит изменения в некоторый логический блок. Чем меньше этот блок, тем меньше различия в схемах и тем труднее построить различающую последовательность. Минимальным логическим блоком, в котором можно произвести изменения, для вентильного уровня представления является логический элемент. Поэтому схема для сравнения будет строиться из оригинальной путём случайного изменения одного типа вентиля в схеме. Для двух полученных таким образом схем мы производим запуск алгоритма верификации.

Апробация реализации алгоритма проводилась на схемах из международного каталога ISCAS-89 [15]. Данный каталог содержит описания цифровых последовательностных схем различной размерности (от единиц до десятков тысяч вентилей) и различной сложности. На схемах данного каталога принято проводить апробацию эффективности алгоритмов идентификации последовательностных схем.

С целью изучения свойств алгоритма решать поставленную задачу для каждой из контрольных схем проводилось по 25 экспериментов.

Значения эвристических параметров, подобранные на основании машинных экспериментов: $T_{нач} = 120$, $T_{кон} = 1$, шаг изменения температуры $\Delta T = 1$, мощность окружения $N = 50$, константа Больцмана $k = 1*10^{-5}$, число итераций без улучшения оценки $N_{итер} = 50$.

Табл.1. Результаты численных экспериментов.

название схемы	число вентилей / триггеров в схеме	эксперименты			
		всего	различено схем	не различено схем	ответ под вопросом
s298	145 / 14	25	25	0	0
s344	198 / 15	25	25	0	0
s349	199 / 15	25	25	0	0
s382	191 / 21	25	23	1	1
s386	182 / 6	25	25	0	0
s967	465 / 29	25	21	4	0
s1196	578 / 18	25	25	0	0
s1238	557 / 18	25	25	0	0
s1423	756 / 74	25	25	0	0
s1488	689 / 6	25	23	0	2
s1494	683 / 6	25	23	0	2
s3271	1731 / 116	25	25	0	0
s3330	2037 / 131	25	22	2	1
s3384	1940 / 183	25	25	0	0
s4863	2514 / 104	25	25	0	0
s5378	3045 / 179	25	24	0	1
s6669	3460 / 239	25	25	0	0
	Всего:	425 (100%)	411 (96,71%)	7 (1,645%)	7 (1,645%)

Числовые результаты экспериментов с указанными значениями параметров приведены в табл.1.

Возможны три результата работы алгоритма:

- алгоритм построил необходимую последовательность, т.е. сумел различить поведение схем, результаты представлены в столбце «различено схем»;
- алгоритм не построил последовательность и функция оценки в процессе поиска осталась раной нулю, т.е. на всех рассмотренных входных последовательностях не удалось увидеть различия поведения даже внутри схемы, результаты представлены в столбце «не различено схем»;
- алгоритм не построил требуемую последовательность, но различия наблюдались внутри схемы, однако не были распространены на внешние выходы, в этом случае, возможно, построить последовательность удалось бы за счёт углубления поиска, результаты представлены в столбце «ответ под вопросом».

На основании приведённых данных можно сделать вывод о высокой эффективности предложенного алгоритма: число различенных схем 96,71%. Для сравнения приведём данные по другим алгоритмам верификации эквивалентности: генетический алгоритм верификации эквивалентности – 94,7% [9], алгоритм VEGA – 88,35% [16], алгоритм AQUILA – 65,00% [17].

4. Оптимизация алгоритма для многоядерных рабочих станций

В настоящее время стандартом «де-факто» стали многоядерные рабочие станции. Это должно учитываться при разработке новых алгоритмов, в том числе и идентификации цифровых схем. Многоядерная структура процессоров, по сути, является компактной реализацией многопроцессорных систем с общей памятью и, следовательно, позволяет локально реализовывать параллельные алгоритмы. Хотя число ядер в современных процессорах невелико (2-4), уже вышло поколение, которое за счёт технологии HyperThreading даёт возможность пользоваться восемью ядрами без физического построения вычислительного кластера: коммутационная среда, среда передачи сообщений и т.д.

В данном разделе будет предложена одна модификация описанного выше алгоритма, которая существенно ускоряет его работу на двухядерных рабочих станциях и улучшает параметр производительности процессоров в пересчёте на один транзистор [18], который резко

падает в современных вычислительных системах, показывая малоэффективное применение их вычислительных ресурсов. Данное ограничение связано с тем, что принципиально алгоритм симуляции отжига является последовательным по своей структуре, в отличии, например, от генетических алгоритмов, в которых параллелизации эффективно подвергается блок оценки особей в популяции.

Однако и в предложенном алгоритме можно найти участок, позволяющий эффективную параллельную организацию. Это функция вычисления оценки одной конфигурации. В описанном выше алгоритме она реализуется на основе моделирования работы двух сравниваемых схем. В описанном выше варианте алгоритма данное моделирование выполняется последовательно для обеих схем: эталонной и оптимизированной. Нами предлагается модификация, которая будет выполнять моделирование параллельно для обеих схем.

Инструментом реализации такого подхода являются потоки. Применяя терминологию потоков, последовательное вычисление функции оценки можно представить следующим образом.

```
ПоследовательноеВычислениеФитнесс
(схема1,схема2,последовательность)
{
  Поток1=СоздатьПотокМоделирования
      (схема1,последовательность);
  Поток1->ЖдатьЗавершения();
  Поток2=СоздатьПотокМоделирования
      (схема2,последовательность);
  Поток2->ЖдатьЗавершения();
  Фитнесс=СравнениеСхем
      (схема1,схема2);
}
```

Видно, что моделирование работы второй схемы не начинается до тех пор, пока не будет завершено моделирование поведения первой схемы. После этого происходит сравнение результатов моделирования и вычисление на его основе оценки качества входной последовательности.

Такая реализация является совершенно естественной для однопроцессорной системы. Очевидно также, что такая реализация функции в двухядерных системах приводит к простою второго ядра: сначала, когда выполняется моделирование работы первой схемы, и далее во время моделирования второй схемы. Следовательно, в многоядерных системах можно не дожидаться окончания моделирования для первой схемы, а параллельно выполнять аналогичную задачу для второй схемы на другом ядре. В терминах потоков моделирования модифицированная процедура вычисления оценки входной последовательности приведена ниже.

Табл.2. Результаты численных экспериментов.

название схемы	время работы		ускорение, (раз)	эффективность загрузки ядер	доля последовательного кода
	последовательный вариант	параллельный вариант			
s298	24	18	1,33	0,67	0,5
s344	12	7	1,71	0,86	0,167
s349	26	16	1,63	0,81	0,231
s382	14	11	1,27	0,64	0,571
s386	28	16	1,75	0,88	0,143
s967	88	47	1,87	0,94	0,068
s1196	148	76	1,95	0,97	0,027
s1238	143	72	1,99	0,99	0,007
s1423	137	71	1,93	0,96	0,036
s1488	93	55	1,69	0,85	0,183
s1494	115	61	1,89	0,94	0,061
s3271	254	132	1,92	0,96	0,039
s3330	471	246	1,91	0,96	0,045
s3384	599	308	1,94	0,97	0,028
s4863	402	209	1,92	0,96	0,040
s5378	15	9	1,67	0,83	0,2
s6669	410	210	1,95	0,98	0,024
		в среднем:	1,78	0,89	0,139

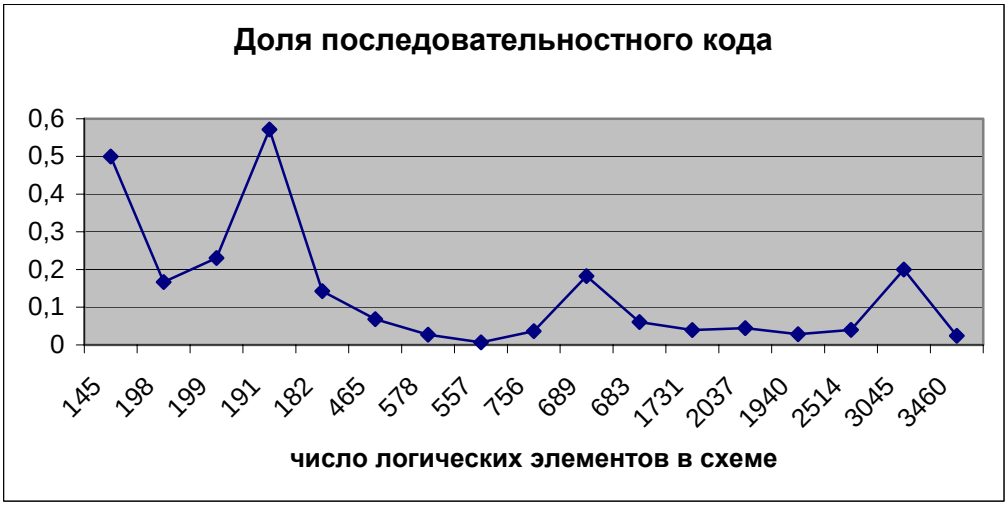

Рис.1. Доля последовательностного кода в параллельной реализации программы в зависимости от числа вентилей обрабатываемой схемы.

```
ПараллельноеВычислениеФитнесс
(схема1,схема2,последовательность)
{
  Поток1=СоздатьПотокМоделирования
       (схема1,последовательность);
  Поток2=СоздатьПотокМоделирования
       (схема2,последовательность);
  Поток1->ЖдатьЗавершения();
  Поток2->ЖдатьЗавершения();
  Фитнесс=Сравнение_схем
       (схема1,схема2);
}
```

Такое построение данной процедуры увеличивает загрузку ядер процессора, а следовательно, повышает эффективность их использования [19].

Отметим также, что при этом остальная часть алгоритма, которая реализует эволюционный поиск, остаётся неизменной, сохраняя высоким параметр повторного использования кода.

Как было отмечено выше, предлагаемая оптимизация не есть параллельная версия алгоритма в прямом его смысле. Она существенно улучшает временные характеристики только для двухядерных систем, для систем с большим числом вычислительных ядер рост этих характеристик прекращается. Поэтому для двухядерных рабочих станций имеет смысл привести необходимые для оценки степени параллелизации алгоритма характеристики: ускорение и эффективность использования ядер процессора [19].

Для получения указанных числовых данных были проведены дополнительные эксперименты со схемами ISCAS-89 (табл. 2). При проведении экспериментов мы фиксировали число итераций, а алгоритм симуляции отжига моделировал поиск последовательности с постоянно увеличивающейся длиной. Общее число рассмотренных точек в пространстве поиска составляло 2381 (для схемы s382 – 11901, для схемы s6669 - 1191). Для немодифицированной (последовательной) и модифицированной (параллельной) версий алгоритма измерялось время работы. По полученным данным вычислялись следующие характеристики: ускорение, эффективность загрузки процессоров (ядер в многоядерных системах), доля последовательного кода.

Ускорение рассчитывается для параллельной реализации алгоритма на системе с p процессорами (ядрами) и определяется формулой:

$$S_p(n) = \frac{T_1(n)}{T_p(n)} \qquad (4)$$

где p – число процессоров в параллельной реализации алгоритма, n – некоторый параметр вычислительной сложности алгоритма, $T_i(n)$ - время выполнения параллельного алгоритма на системе с i процессорами.

Эффективность использования процессоров при параллельной реализации алгоритма рассчитывается по формуле:

$$E_p(n) = \frac{T_1(n)}{p \cdot T_p(n)} = \frac{S_p(n)}{p} \qquad (5)$$

Для нашего подхода под числом процессоров понимается число ядер процессора. Следо-

вательно, параметр $E_p(n)$ будет выражать эффективность использования ядер процессора.

Доля последовательностного кода (доля последовательных вычислений) f служит для оценки свойств алгоритма к распараллеливанию и определяется из закона Амдаля:

$$f = \frac{p/S_p - 1}{p - 1} \qquad (6)$$

Численные результаты экспериментов и расчётов приведены в табл.2. Из таблицы видно, что с ростом числа логических элементов в схеме доля последовательного кода снижается, а параллельного – растёт. Чтобы наглядно оценить долю последовательного кода в зависимости от числа вентилей в исследуемой схеме, построим данную зависимость графически (рис.1). Видно, что по мере роста числа элементов в схеме, доля последовательного кода резко падает и составляет в среднем меньше 5% для всех схем (за исключением двух) с числом вентилей большим 500. Это объясняется тем, что в этом случае алгоритм подавляющую часть времени своей работы тратит на вычисление оценок конфигураций (т.е. моделирование работы сравниваемых схем), а не на непосредственное преобразование конфигураций. Отметим, что несмотря на простоту предложенной параллельной модификации, были получены очень высокие значения констант, характеризующих параллельные свойства алгоритма: ускорение работы – в среднем 1.78 раза; эффективность загрузки ядер – в среднем 0.89 и параллельной доли кода – в среднем 0.86. При этом лучшие значения, полученные для схемы s1238, равны 1.99, 0.97 и 0.993 соответственно.

5. Заключение

В статье впервые предложен алгоритм построения входных последовательностей для верификации эквивалентности цифровых схем. Данный алгоритм основан на оптимизирующей стратегии симуляции отжига. При его построении возможно применение компонент аналогичного по целям генетического алгоритма, что существенно сокращает время разработки. Несмотря на более простую стратегию оптимизации, которая основана на улучшении свойств одного потенциального решения, в целом предложенный алгоритм показал лучшую эффективность в работе с контрольными схемами в сравнении с уже известными алгоритмами. Также предложена параллельная модификация данного алгоритма для двухядерных рабочих станций, которая на контрольных схемах ISCAS-89 показывает повышение быстродействия в среднем в 1.78 раза.

В качестве дальнейших исследований можно отметить следующие:

- разработка алгоритма симуляции отжига построения входных тестовых последовательностей для одиночных константных неисправностей;
- разработка параллельных версий алгоритмов симуляции отжига.

СПИСОК ЛИТЕРАТУРЫ

1. *Niermann T., Patel J.H.* HITEC: A Test Generation Package for Sequential Circuits // Proc. European Design Automation Conf. – 1991. – P. 214-218.
2. *Ghosh, A., S. Devadas and A. R. Newton*. Sequential Logic Testing and Verification. – Kluwer Academic Publishers. – 1992. – 214p.
3. *Corno F., Prinetto P., Rebaudengo M., Sonza Reorda M.*, GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits // IEEE Transactions on Computer-Aided Design, August 1996. – Vol. 15, №8. – P. 943-951.
4. *Niermann T.M., Cheng W.-T., Patel J.H.* PROOFS: A Fast, Memory-Efficient Sequential Circuits Fault Simulator // IEEE Trans. CAD. – 1992.– P.198-207.
5. Goldberg D.E., Genetic Algorithm in Search, Optimization, and Machine Learning. – Addison-Wesley. – 1989.
6. *Скобцов Ю.А.*. Основы эволюционных вычислений.– Донецк: ДонНТУ, 2008. – 326с.
7. *Иванов Д.Е., Скобцов Ю.А.* Генерация тестов цифровых устройств с использованием генетических алгоритмов // Труды института прикладной математики и механики НАН Украины. – Т.4. – Донецк, ИПММ. – 1999. – С.82-88.
8. *Иванов Д.Е., Скобцов Ю.А., Эль-Хатиб А.И.* Построение инициализирующих последовательностей синхронных цифровых схем с помощью генетических алгоритмов. // Проблеми інформаційних технологій. – 2007. – №1. – с.158-164.
9. *Иванов Д.Е.* Генетический подход проверки эквивалентности последовательностных схем // Радіоелектроніка. Інформатика. Управління.- Запоріжжя, ЗНТУ. – 2009. – №1(20). – С.118-123.
10. *Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E.,* Equation of State Calculation by Fast Computing Mashines. // J. of Chem.Phys. – 1953. – Vol.21, No.6. – P.1087-1092.
11. *Kirkpatrick S., Gelatt C.D., Vecchi M.P.* Optimiza-

tion by simulating annealing. // Science, 220. – 1983. – P.671-680.
12. *Cerny V.* Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. // J. Optim. Theory Appl. – 1985. – v.45. – P. 41-51.
13. *Иванов Д.Е., Зуауи Р.* Применение стратегии симуляции отжига для задачи построения инициализирующих последовательностей цифровых схем // Вісник східноукраїнського національного університету ім.В.Даля. – 2009. – №1(131), Ч. 2. – С.161-168.
14. *Барашко А.С., Скобцов Ю.А., Сперанский Д.В.* Моделирование и тестирование дискретных устройств. – Киев: Наукова думка, 1992. – 288 с.
15. *Brgles F., Bryan D., Kozminski K.* Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89.– 1989.– P.1929-1934.
16. *F. Corno, M. Sonza Reorda, G. Squillero* VEGA: A Verification Tool Based on Genetic Algorithms // ICCD98, International Conference on Circuit Design, Austin, Texas (USA). – 1998. – P.321-326.
17. *Shi-Yu Huang , Kwang-Ting Cheng , Kuang-Chien Chen, Forrest Brewer, Chung-Yang Huang,* AQUILA: An Equivalence Checking System for Large Sequential Designs. // IEEE Transactions on Computers. – 2000. – v.49, N.5. – P.443-464,.
18. *Ю.С. Затуливер, Е.А. Фищенко* Компьютер ПС-2000:Многопроцессорная архитектура, опередившая время // Пленарные и избранные доклады 4-й Международной конференции «Параллельные вычисления и задачи управления», Москва, 27-29 октября 2008.
19. *Гергель В.П., Стронгин Р.Г.* Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие. – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184с.

Ivanov D.E., Zouaoui R. **Equivalence verification of the digital circuits with the strategy of the simulating annealing**

Іванов Д.Е., Зуаві Р. **Верифікація еквівалентності цифрових схем з використанням стратегії симуляції відпалу**

У статті запропоновано новий алгоритм верифікації еквівалентності цифрових послідовних схем. Він заснований на еволюційній стратегії симуляції відпалу, яка використовує ітеративне покращення властивостей однієї вхідної послідовності. Для оцінки якості отримуваних рішень використовується справне моделювання цифрових схем. Ефективність пропонованого алгоритму показана шляхом апробації на схемах з каталогу ISCAS-89.