МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ПРОБЛЕМИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рекомендовано до друку Вченою радою
Херсонського національного
технічного університету
(протокол №8 від 29 червня 2011 року)

Журнал «Проблеми інформаційних технологій»
включено до Переліку наукових фахових видань
ВАК України (Постанова Президії ВАК України
№3-05/6 від 06.10.2010 р.), у яких можуть
публікуватися результати дисертаційних робіт на
здобуття наукових ступенів доктора
та кандидата наук

ISSN 1998-7005

#01(009) грудень 2011

/ СЕКЦІЯ 2 /
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

| В.Е. Ходаков, Д.В. Хапов |
Информационная технология оценки инвестиционной привлекательности и устойчивости развития региона

86

| Э.Г. Петров, Е.В. Губаренко |
Требования к системам комплексного мониторинга социально-экономических систем

98

| В.В. Москаленко, В.В. Кондращенко |
Модели и информационная технология построения схем финансирования инвестиционных проектов

104

| Д.Е. Иванов |
Автоматизированная система моделирования и идентификации цифровых устройств асмид-evolution

114

| А.М. Петрушенко |
Про природу (першопричину) складності обчислювальних систем та деякі ймовірні шляхи її подолання

125

| Г.В. Веселовська, А.Д. Чеклін, І.І. Кибалко |
Методи та інформаційні технології оптимізації взаємодії користувачів із електронними інформаційними ресурсами галузі знань «інформатика та обчислювальна техніка»

131

| Е.В. Высоцкая, А.П. Порван, Л.И. Рак, В.Г. Антоненко, Е.Е. Болибок, О.А. Сватенко |
Информационная технология определения систолической дисфункции миокарда у подростков

138

| М.М. Верес, Є.О. Демківський, О.А. Омельченко |
Масово розподілений пошуковий робот

147

АВТОМАТИЗИРОВАННАЯ СИСТЕМА МОДЕЛИРОВАНИЯ И ИДЕНТИФИКАЦИИ ЦИФРОВЫХ УСТРОЙСТВ АСМИД-EVOLUTION

УДК 681.518: 681.326.7

ИВАНОВ Дмитрий Евгеньевич

к.т.н., доцент, старший научный сотрудник института прикладной математики и механики НАН Украины.

Научные интересы: техническая диагностика, моделирование и тестирование цифровых устройств, генетические алгоритмы, эволюционные вычисления, параллельные алгоритмы.

АКТУАЛЬНОСТЬ ПОСТРОЕНИЯ И НАЗНАЧЕНИЕ СИСТЕМЫ.

Перечень функций, которые возлагаются на компьютерную технику, требует повышения надёжности её функционирования. Данное свойства современных цифровых устройств (ЦУ) закладывается при проектировании, которое в настоящее время полностью основано на применении современных автоматизированных систем. Центральное место в системах автоматизированного проектирования (САПР) занимают средства верификации, идентификации и оптимизации проектируемых устройств. В данной статье описывается структура, математическая основа и алгоритмическое наполнение новой версии системы моделирования и идентификации АСМИД.

Данная версия системы разработана в Институте прикладной математики и механики НАН Украины и является четвертым её поколением. Предыдущие версии функционировали на ЕС ЭВМ [1], в среде MS-DOS [2] и Windows [3]. В настоящее время ведётся работа над версией 4.1 системы, которая носит название АСМИД-Evolution, поскольку в ней широко представлены алгоритмы идентификации ЦУ, основанные на эволюционных вычислениях.

Система АСМИД предназначена для работы с комбинационными и с последовательностными схемами. При этом число элементов в схемах, а также число линий обратных связей не ограничивается. Набор функций, которые реализует система, позволяет использовать её на всех этапах «жизненного цикла» ЦУ: при проектировании, изготовлении, диагностике и эксплуатации. К функциям системы относятся:

- текстовый ввод описания цифровой схемы и предпроцессинг;
- логическое моделирование исправной схемы на заданной входной последовательности с целью верификации исправного поведения и получения временных диаграмм его поведения;
- построение проверяющих и диагностических тестов;
- моделирование работы неисправных ЦУ с целью определения диагностических свойств тестов;
- верификация эквивалентности двух заданных ЦУ с целью проверки корректности этапов оптимизации;
- построение инициализирующих последовательностей;
- определение параметров рассеивания тепла и пикового рассеивания тепла;
- построение тестов с минимальным рассеиванием тепла (энергоэффективных тестов);
- просмотр результатов работы, включая временные диаграммы.

К компонентам, которые используют новые алгоритмические решения на основе эволюционных вычислений относятся генераторы входных

идентифицирующих последовательностей и блок программ построения энергоэффективных тестов.

СТРУКТУРА СИСТЕМЫ.

Структурно система АСМИД состоит из нескольких подсистем, каждая из которых имеет свою функциональную направленность (рис. 1).

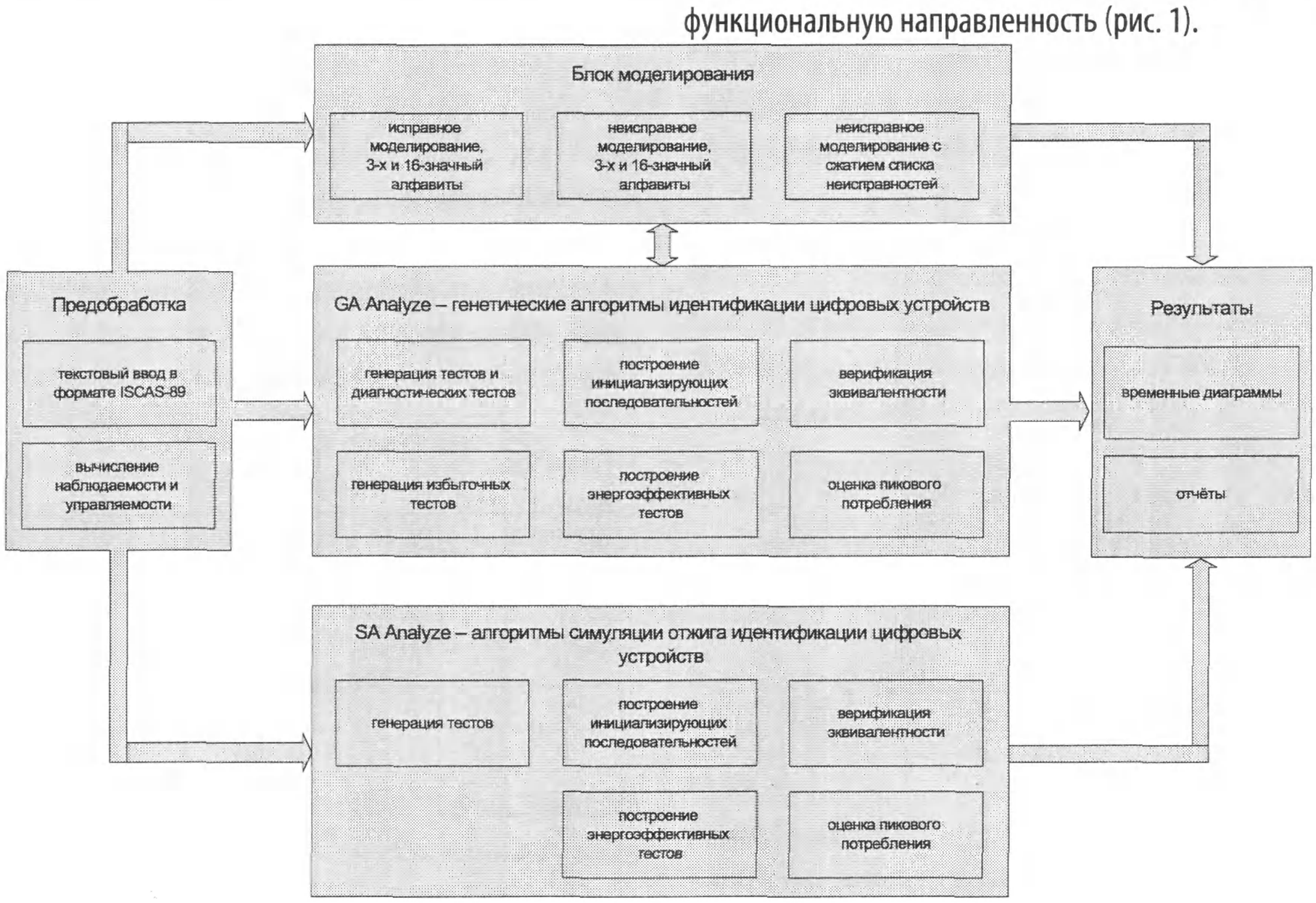

Рисунок 1 – Укрупнённая структура системы АСМИД

Подсистема предпроцессора предназначена для подготовки описания схемы к дальнейшей работе. Она содержит следующие элементы: программа транслятор входного описания, программа вычисления характеристик наблюдаемости и управляемости элементов схемы; программа построения полного списка неисправностей, программа сжатия списка неисправностей по эквивалентности.

Подсистема моделирования включает следующие компоненты: программа моделирования работы исправных ЦУ в 3-х и 16-значных алфавитах, программа моделирования ЦУ с неисправностями в 3-х и 16-значных алфавитах, программа быстрого моделирования ЦУ с неисправностями в 3-значном алфавите с динамическим сжатием списка неисправностей с целью анализа полноты теста [4].

Подсистема «GA-analyze» включает генетические алгоритмы (ГА) идентификации цифровых устройств, а именно: генератор тестов [5], генератор диагностических (локализующих) тестов [6], программа построения инициализирующих последовательностей [7], программа для верификации эквивалентности двух заданных схем [8], генератор избыточных тестов, программа построения энергоэффективных тестов [9], программа оценки пикового потребления энергии для заданной схемы.

Подсистема «SA-analyze» по назначению и составу программ практически полностью аналогична подсистеме «GA-analyze». Описание её компонент приведено в [10-12]. Основным отличием является то, что в данной подсистеме для решения указанных задач используются алгоритмы симуляции отжига (СО,

simulating annealing). Включение в систему подсистем с аналогичными функциями, но с различным алгоритмическим наполнением повышает гибкость разработчика в выборе компонент для работы.

Подсистема выходного анализа (постпроцессинг) позволяет просматривать отчёты всех программ, входящих в систему, а также временные диаграммы работы исправного ЦУ.

МАТЕМАТИЧЕСКАЯ ОСНОВА.

Математической основой алгоритмов в системе АСМИД являются булевы алфавиты различной значности, а также основанные на них многозначные логики. Простейшим из применяемых алфавитов является $B_2 = \{0,1\}$. Он достаточно точно отражает поведение устройства в статике, однако совсем не учитывает переходные процессы, которые возникают в схеме при смене значений входных сигналов. Поэтому в качестве базового алфавита выбирается 4-значный алфавит $B_4^2 = \{00,01,10,11\}$, элементы которого имеют следующую интерпретацию: $00(11)$ – значения сигналов одинаковы, $01(10)$ - значения сигналов отличны в различные моменты времени или в разных технических состояниях ЦУ. В [13] показано, как из данного алфавита строится 16-значный алфавит B_{16} (табл.1.), являющийся основным для рассматриваемой системы. Там же показано, что данный алфавит включает в себя все основные алфавиты, используемые на практике для моделирования и генерации тестов ЦУ. В этом смысле алфавит B_{16} является универсальным. До недавнего времени использование многозначных (более 4 символов) алфавитов при моделировании не получало широкого распространения, поскольку при увеличении значности алфавита из-за сложности многозначных моделей логических элементов резко падала скорость моделирования. Разработка быстрых алгоритмов моделирования ЦУ с неисправностями [14,4] позволила снять данную проблему.

В [13] предлагается метод кодирования алфавита B_{16} и единая система моделей компонентных функций, которые позволяют существенно повысить скорость процесса моделирования. С каждым элементом алфавита B_{16} ассоциируются четыре переменные кодирования X^{00}, X^{01}, X^{10}, X^{1}. Тогда поведение многозначной функции $\tilde{f}(\tilde{X})$, где $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$ описывается с помощью четырёх булевых функций: $f^{00}(X^{00}, X^{01}, X^{10}, X^{11})$, $f^{D'}(X^0, X^{D'}, X^D, X^1)$, $f^{D}(X^0, X^{D'}, X^D, X^1)$, $f^{1}(X^0, X^{D'}, X^D, X^1)$. Для выбранного метода кодирования функции f^0, f^{01}, f^D, f^1 одинаковы для наиболее распространённых многозначных алфавитов. Для основных вентилей данные функции приведены в табл.2.

Таблица 1 –

16-значный алфавит B_{16} и его кодирование

Значение	X^0	$X^{D'}$	X^D	X^1	Значение	X^0	$X^{D'}$	X^D	X^1
∅	0	0	0	0	0	1	0	0	0
1	0	0	0	1	C	1	0	0	1
D	0	0	1	0	F0	1	0	1	0
G1	0	0	1	1	H	1	0	1	1
D	0	1	0	0	G0	1	1	0	0
F1	0	1	0	1	E	1	1	0	1
D*	0	1	1	0	D0	1	1	1	0
D1	0	1	1	1	U	1	1	1	1

Таким образом, множество компонентных функций f^0, f^{01}, f^D, f^{11} является универсальной математической моделью, которая позволяет повысить скорость моделирования в многозначных алфавитах и сделать её приемлемой для практических целей.

Алгоритмы. В системе АСМИД используется целый ряд алгоритмов моделирования и генерации тестов, часть из которых является уникальными. Мы не будем останавливаться на описании алгоритмов, которые перешли в данную версию системы из предыдущих. Они подробно описаны в [4,6,13].

Принципиально новым в системе является:

- широкое применение эволюционных алгоритмов в задачах построения входных идентифицирующих последовательностей;
- разработка параллельных версии ряда генетических алгоритмов и алгоритмов моделирования ЦУ с неисправностями;
- разработка ряда алгоритмов проектирования энергоэффективных схем.

ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ.

В предыдущей версии системы [3] впервые был предложен генетический алгоритм построения тестов. Он заключается в эволюции набора (популяции) потенциальных решений, которые представлены двоичными входными последовательностями. В данной версии системы генетический подход также применён к решению задач инициализации и верификации эквивалентности ЦУ.

Задача логической инициализации возникает в тех случаях, когда в спецификации проектируемого ЦУ не заложен механизм аппаратного сброса. При построении ГА решения данной задачи аналогично задаче построения тестов в качестве особи выбирается одиночная последовательность и задаются генетические операции [5]. Оценка качества последовательности S строится на её способности снимать неопределённость с линий обратных связей в схеме, что задаётся выражением:

$$f(s) = f(n_1, n_2, n_3) = (c_1 * n_1 + c_2 * n_2) * c_3{}^{n3}$$

где: n_1 – отношение числа инициированных триггеров к их общему числу; n_2 – активность схемы или число событий моделирования; n_3 – длина последовательности; c_1, c_2, c_3 – нормализующие константы.

Эволюция популяции решений происходит до тех пор, пока не будет снята неопределённость со всех триггеров схемы.

Верификация эквивалентности поведения двух заданных ЦУ проводится после проведения оптимизации дизайна с целью проверить: является ли поведение оптимизированного и неоптимизированного ЦУ эквивалентным. Формальное доказательство такой эквивалентности часто затруднено либо из-за размеров обрабатываемого ЦУ, либо из-за того, что для оптимизированного ЦУ не строится формальная модель. Однако для разработчика важно не столько показать формальную эквивалентность, сколько увидеть, что в процессе оптимизации не произошёл сбой, т.е. увидеть различие в поведении двух заданных устройств. В этом случае возможно переформулировать задачу на противоположную: необходимо построить такую последовательность, которая покажет неэквивалентность поведения двух заданных ЦУ. В такой постановке построение алгоритма верификации эквивалентности практически полностью аналогично ГА инициализации, а оценка качества потенциального решения строится на степени различия поведений двух заданных ЦУ. При этом в качестве оценки последовательности s выступает выражение:

$$f(s) = f(n_1, n_2, n_3) = n_1 + c_1 \cdot n_2 + c_2 \cdot n_3,$$

где: n_1 – число различных значений на внешних выходах двух анализируемых схем; n_2 – число различных псевдовыходов схем; n_3 – число вентилей двух схем с различными значениями сигналов; c_1, c_2, c_3 – нормализующие константы.

Новым в данной системе является применение оптимизационной стратегии симуляции отжига. Кратко опишем её суть. Вводится понятие конфигурации K_i, которая является потенциальным решением поставленной комбинаторной задачи. Также выбирается кодирование для представления точек в пространстве поиска в виде конфигураций. С каждой конфигурацией K_i ассоциируется функция стоимости $C_i = C(K_i)$, показывающая целесообразность конфигурации для решения данной проблемы. Цель оптимизации – найти конфигурацию, которая максимизирует функцию стоимости. Дополнительно вводится понятие окружения конфигурации $O(K_i)$: множество потенциальных конфигураций, которые могут быть получены из K_i путём применения некоторых возмущающих операций. Тогда общая схема стратегии симуляции отжига может быть представлена в виде схемы на рис. 2. Видно, что как и ГА, алгоритмы СО являются итеративным процессом улучшения решения. Завершение работы происходит при нахождении приемлемого решения, либо достижении предельного числа итераций. Наиболее существенным отличием алгоритмов СО от ГА является то, что в данном случае происходит эволюция одного потенциального решения, а не популяции таких решений как в ГА.

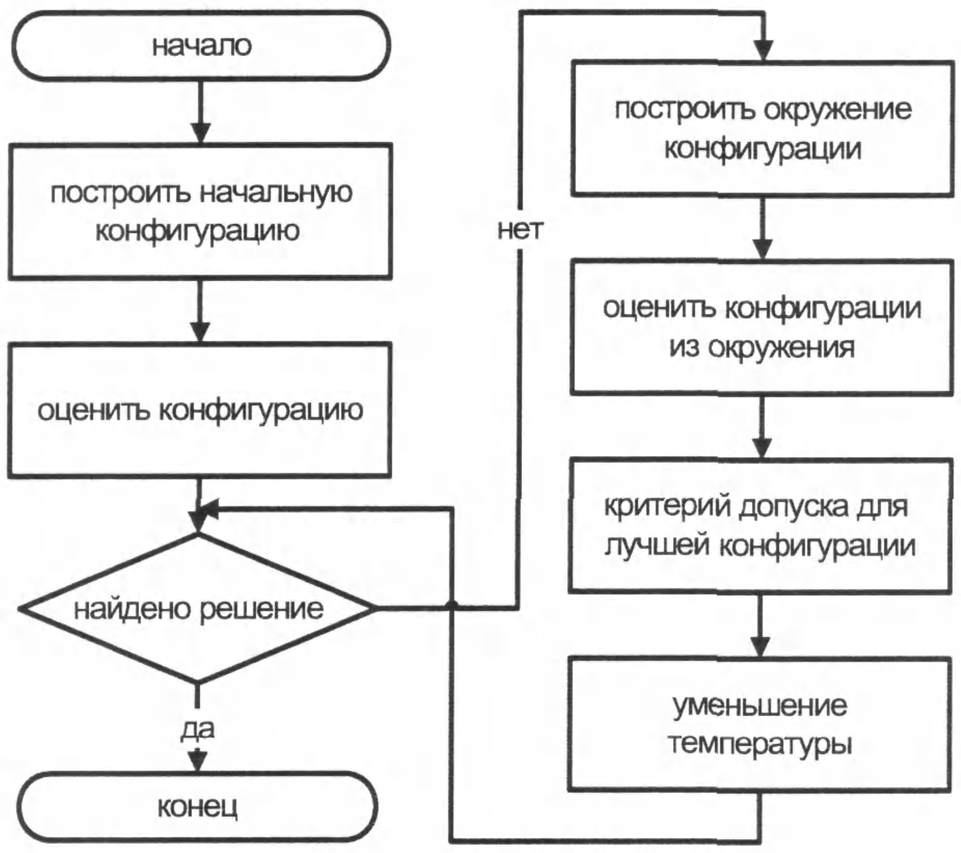

Рисунок 2 – Общая схема алгоритма симуляции отжига

На основании стратегии СО в системе реализованы алгоритмы генерации тестов, построения инициализирующих последовательностей и последовательностей для верификации эквивалентности двух заданных ЦУ. Несмотря на упрощение подхода, в ряде задач алгоритмы СО показывают лучшие экспериментальные результаты в сравнении с ГА [10-12].

В рассматриваемой версии системы АСМИД все алгоритмы решения задач, которые основаны на СО объединены в подсистему «SA Analyze». Набор решаемых задач совпадает с подсистемой «GA Analyze». Таким образом, на основании двух данных подсистем для разработчика сформирована альтернатива в выборе алгоритма решения той или иной задачи.

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ.

Развитие многоядерных процессоров и их массовое внедрение в инструментальные ЭВМ разработчиков заставляет соответствующим образом развивать программное обеспечение. Если в случае роста тактовой частоты процессоров скорость работы программ росла соответствующим образом, то наращивание числа ядер не ведёт к автоматическому повышению производительности программ. Таким образом, алгоритмы, на которых строится программное обеспечение САПР, для работы на многоядерных рабочих станциях должны обладать свойством масштабируемости [15].

Учитывая данную необходимость, для системы АСМИД разработан ряд параллельных алгоритмов, в частности генетические алгоритмы построения идентифицирующих последовательностей и алгоритмы моделирования ЦУ с неисправностями.

При построении параллельных генетических алгоритмов в основном используется схема «рабочий-хозяин» [16-18]. Данная схема предполагает, что в параллельном ГА существует только один основной цикл построения популяций, который соответствует процессору «хозяину». Все остальные процессоры являются «рабочими» и они реализуют процедуры вычисления оценок особей. Заметим, что именно данные процедуры несут основную вычислительную нагрузку, поскольку оценка производится путём моделирования поведения работы схемы на заданной входной последовательности, представляющей особь в генетическом алгоритме.

Фактически, распараллеливанию подвергается процедура оценки особей в популяции. Инструментарием построения параллельных приложений в многоядерных и многопроцессорных вычислительных системах с общей памятью являются потоки исполнения. Псевдокод параллельной версии функции оценки особей в терминах потоков приведён ниже.

```
ОценитьПопуляцию(Популяция, ЧислоОсобей)
{
 for( int i=0 ; i<ЧислоОсобей/ЧислоПотоков ; i++ ){
  j=i*ЧислоПотоков;
  выполнять_параллельно_для j, j+1, … , j+ЧислоПотоков-1
  {
   ОценитьОсобь(Популяция->Особь[j]);
  }
  ЖдатьОкончанияПотоков j, j+1, … , j+ЧислоПотоков-1;
 }
}
```

Апробация параллельных ГА построения входных последовательностей проводилась на рабочей станции под управлением MS Windows Server 2008 R2, которая содержала два процессора Intel(R) Xeon CPU X5650 с частотой 2.67 Ггц (каждый по 6 вычислительных ядер), технология HyperThreading – выключена, объем оперативной памяти 16 Гб. Достигнутое ускорение работы алгоритмов составило от 9 до 13 раз при числе одновременных вычислительных потоков от 80 до 120.

Также параллельные версии разработаны для алгоритма моделирования ЦУ с неисправностями [19-21]. Первоначальная параллельная версия алгоритма, описанная в [19-20], основывается на схеме разбиения списка неисправностей: полный список неисправностей F разбивается на несколько подсписков $F_1, F_2, ..., F_n$, каждый из которых передаётся на отдельный процессор системы, где и выполняется его анализ путём моделирования на заданной входной последовательности. Однако данная схема имеет существенный недостаток: списки неисправностей на некоторых процессорах могут моделироваться существенно дольше, чем на остальных. Последние в такой ситуации вынуждены простаивать. В новом алгоритме [21] удалось избежать указанного недостатка за счёт того, что список неисправностей будет разбиваться динамически для каждого входного набора. При этом для каждого вычислительного ядра формируется группа неисправностей. Число неисправностей в такой группе равно числу разрядов машинного слова инструментальной ЭВМ. Неисправности в каждой группе моделируются параллельно по разрядам машинного слова, дополнительно повышая быстродействие. Укрупнённый псевдокод данного алгоритма приведён ниже.

```
ПараллельноеМоделирование1(Схема,
СписокНеисправностей, ВходнаяПоследовательность,
ЧислоНаборов, ЧислоПотоков)
    {
     while( НомерНабора < ЧислоНаборов )
     {
      SVI=МоделироватьИсправнуюСхему(Схема,
Набор);
      while( в списке есть непроверенные
неисправности )
       {

for(НомерПотока=0;НомерПотока<ЧислоПотоков;Номе
рПотока++)
        {

Группа=СформироватьГруппу(СписокНеисправностей);
         ПотокМоделирования[НомерПотока]-
>Запустить
         (Набор,SVI,Группа);
        }

for(НомерПотока=0;НомерПотока<ЧислоПотоков;Номе
рПотока++)
         ПотокМоделирования [НомерПотока]-
>ЖдатьОкончания();
  }// конец цикла по неисправностям
     } // конец цикла по входным наборам
    }  // конец ядра моделирования
```

Апробация данного алгоритма проводилась на рабочей станции с указанными выше характеристиками. Достигнутое ускорение работы программы составило от 4.61 до 6.72 раз для больших схем.

Разработанный параллельный алгоритм моделирования ЦУ с неисправностями используется в системе как самостоятельно для проверки полноты тестов, так и в качестве вспомогательного в генетических алгоритмах построения тестов.

АЛГОРИТМЫ ПРОЕКТИРОВАНИЯ ЭНЕРГОЭФФЕКТИВНЫХ УСТРОЙСТВ.

Впервые в данной версии системы представлены алгоритмы, цель которых – помочь в проектировании энергоэффективных ЦУ [9,12]. В настоящее время разработаны следующие алгоритмы:

- алгоритм оценки рассеивания тепла при работе на заданной последовательности;
- алгоритм построения энергоэффективных тестов;
- алгоритм оценки пикового рассеивания тепла для заданного ЦУ.

Построение энергоэффективных тестов основано на понятии избыточного тестирования состоит из трёх этапов.

Этап 1 решения задачи заключается в построении входной последовательности (набора подпоследовательностей) $S = \{s_1, s_2, ..., s_l\}$, которая обладает требуемой избыточной полнотой при фиксированной избыточности r. При этом для каждой неисправности гарантируется существование не менее чем r входных подпоследовательностей, принадлежащих S, таких, что каждая из них обнаруживает данную неисправность. Гарантирование избыточности для некоторой неисправности позволяет в дальнейшем в зависимости от целей задачи выбирать в качестве окончательного решения ту или иную из последовательностей.

Этап 2 заключается в предварительной оценке подпоследовательностей, которые входят в S: для каждой подпоследовательности $s_i \in S$ вычисляется числовая оценка $E(s_i)$, которая показывает рассеивание тепловой энергии при моделировании работы схемы на заданной входной последовательности.

Этап 3 заключается в выборе подмножества последовательностей $S' \subset S$ такого, что полнота теста не ухудшается $P(S') = P(S)$ и рассеивание тепла минимально $E(S') \to \min$.

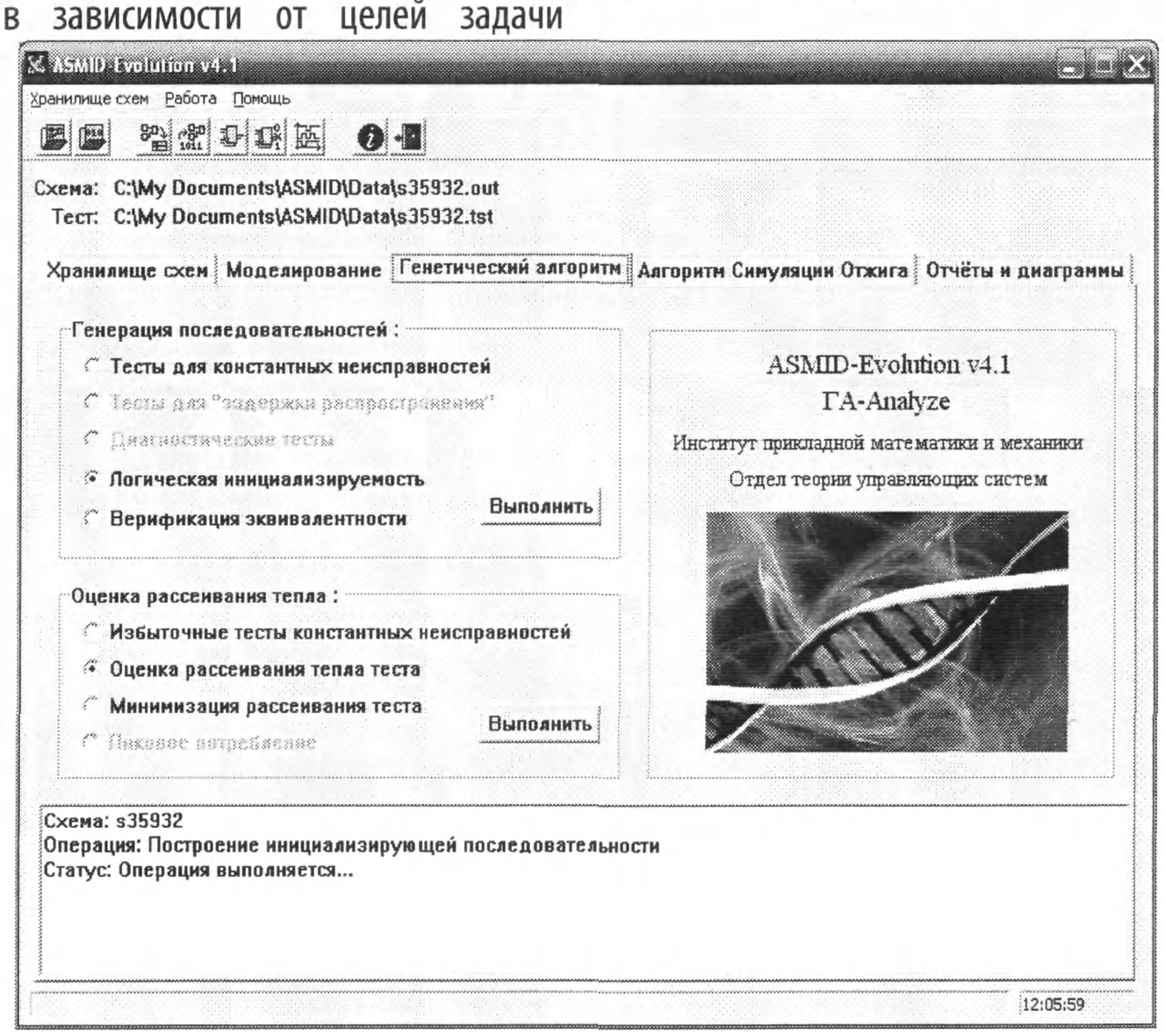

Рисунок 3 – Главная экранная форма системы АСМИД-Evolution

Первые два этапа являются подготовительными, а третий является задачей комбинаторной оптимизации. В системе реализовано решение задач этапов 1 и 3 как с помощью алгоритмов ГА, так и с помощью алгоритмов СО. Решение задачи второго этапа состоит в моделировании работы схемы на заданных последовательностях с получением нужной оценки. При этом следует заметить, что данный алгоритм также имеет самостоятельную ценность.

Проведённые эксперименты с реализацией подхода позволяют уменьшить рассеивание тепла в фазе тестирования в среднем на 86% для набора контрольных схем ISCAS-89.

Задача оценки пикового рассеивания тепла заданным ЦУ ставится, как задача поиска такой тройки объектов (S, I_{t-1}, I_t), что при инициализации элементов состояний устройства состоянием S и последовательной подачей на вход наборов I_{t-1}, I_t

показатель рассеивания тепла будет максимальным $P(S, I_{t-1}, I_t) \to \max$. При этом оценка такого показателя рассеивания тепла производится с помощью моделирования. Видно, что задача носит переборный характер и в настоящее время разрабатывается ГА её решения.

Таблица 2 –

Экспериментальные данные программ генерации тестов на основе алгоритма СО, построения инициализирующих последовательностей на основе ГА и СО

схема	тесты*, СО			инициализирующие последовательности**, ГА			инициализирующие последовательности*, СО		
	полнота, %	длина	время, мин:сек	инициировано / всего триггеров	длина	время, мин:сек	инициировано / всего триггеров	длина	время, мин:сек
s344	96.20	438	0:16	15 / 15	2	<1	15 / 15	2	0:07
s349	95.71	566	0:07	15 /15	2	<1	15 / 15	2	0:07
s386	72.57	1189	0:23	6 / 6	2	<1	6 / 6	2	0:07
s641	86.50	1517	0:34	19 / 19	1	<1	19 / 19	1	<1
s713	83:07	1716	0:42	19 / 19	1	<1	19 / 19	1	<1
s1196	99.95	1994	1:43	18 / 18	1	<1	18 / 18	1	<1
s1238	93.27	3866	1:51	18 / 18	1	0:01	18 / 18	1	<1
s1488	73.62	2202	7:11	6 / 6	1	<1	6 / 6	1	<1
s1494	74.30	1947	7:45	6 / 6	1	<1	6 / 6	1	<1
s3271	98.13	2807	1:58	116 / 116	14	0:11	116 / 116	11	0:28
s3330	67.39	8312	1:37:51	132 / 132	3	0:06	132 / 132	3	0:14
s3384	90.71	3191	37:28	183 / 183	8	0:07	183 / 183	7	0:19
s5378	67.37	6913	22:00	179 / 179	40	0:61	179 / 179	14	0:55
s6669	98.34	7640	44:56	239 / 239	5	0:19	239 / 239	2	0:28
s35932	74.28	597	1:09:54	1728 / 1728	1	0:08	1728 / 1728	1	<1

*Инструментальная платформа: Intel CoreQuad Q6400 2.4Ггц, 2Гбайт ОЗУ.

**Инструментальная платформа: Intel Celeron 2.8Ггц, 512Мбайт ОЗУ.

РАБОТА С СИСТЕМОЙ И ЭКСПЛУАТАЦИОННЫЕ ХАРАКТЕРИСТИКИ.

Система АСМИД-Evolution реализована в среде программирования CodeGear и предназначена для работы в операционной системе Windows. При программировании использовался объектно-ориентированный подход, который позволяет существенно ускорить программную реализацию новых алгоритмов за счёт технологии повторного использования кода, поскольку методы различных объектов ГА и СО очень часто совпадают.

Управление системой интерактивное и реализовано в виде системы меню. После запуска головной программы на экран выводится главная форма, которая позволяет выбрать схему и задать режимы дальнейшей работы (рис. 3).

Выбор схемы для работы производится на вкладке «Хранилище схем». При необходимости здесь же выполняется трансляция описания во внутреннюю структуру данных [13]. Входным форматом описания схем является текстовый формат ISCAS-89 [22]. Несмотря на свою простоту, данный формат является широко используемым и в настоящее время. Более того, при развитии исследований на уровне VHDL был выпущен новый набор контрольных схем на данном уровне описания. Однако позднее разработчики привели описание новых схем и на логическом уровне в формате ISCAS-89. Это объясняется, как необходимостью выполнять диагностику на логическом уровне, так и необходимостью соотнесения результатов исследований, полученных на двух данных уровнях. Результатом работы программы-транслятора из формата ISCAS-89 является файл с расширением

*.out, в котором описание схем хранится в двоичном виде в виде системы связных таблиц [13]. Результатом программы оценки параметров наблюдаемости и управляемости является файл с расширением *.cop, который содержит данные в двоичном виде.

Таблица 3 –

Экспериментальные результаты программ построения энергоэффективных тестов*

схема	избыточные тесты, ГА, r=5			Оптимизация ГА			Оптимизация СО		
	полнота, %	длина	время работы, мин:сек	уменьшение рассеивания тепла, %	уменьшение длины, раз	время работы, мин:сек	уменьшение рассеивания тепла, %	уменьшение длины, раз	время работы, мин:сек
s298	85.71	1467	0:21	90.77	8.84	<1	92.02	9.85	1
s344	96.19	3621	1:01	88.30	6.58	<1	92.97	12.57	1
s349	95.42	1960	0:27	91.04	9.95	<1	91.73	12.10	1
s382	89.47	8661	3:36	86.46	6.24	<1	91.02	10.13	1
s386	73.95	17826	5:34	89.23	9.10	<1	90.77	10.49	1
s400	88.21	16617	10:43	92.51	10.75	<1	93.08	11.48	<1
s444	79.32	3676	3:00	87.32	7.02	<1	88.72	8.47	1
s526	8.65	70	0:16	86.09	7.00	<1	72.53	3.50	<1
s635	0.15	40	53:00	73.51	4.00	<1	54.43	2.00	<1
s641	44.75	57372	1:28:25	88.91	9.04	<1	88.91	8.04	3
s713	81.93	11841	3:35	89.02	773	<1	91.56	9.71	1
s832	48.05	3135	1:58	89.21	8.93	0:01	89.74	9.28	4
s938	4.40	1952	1:45:26	86.20	122.00	<1	86.31	122.00	<1
s967	7.41	190	1:05:13	89.56	9.50	<1	89.56	9.50	1
s1196	91.22	16117	1:16	76.48	4.48	0:24	84.19	5.91	23
s1238	77.86	7025	0:56	74.92	3.69	0:07	76.33	3.91	6
s1269	17.87	647	2:55	79.75	4.83	<1	87.77	8.29	1
s1423	73.47	76075	2:01:21	80.78	4.93	0:10	81.89	5.23	5
s1488	92.19	16968	6:22	82.44	5.08	0:14	83.25	5.42	20
s1494	95.22	10966	3:45	88.32	8.26	0:14	89.55	8.61	9
s1512	4.86	130	3:13	84.45	6.50	<1	84.75	6.50	1
s2081	8.29	480	12:03	93.25	15.00	<1	93.28	15.00	<1
s3271	96.57	11422	2:15	86.92	6.52	0:54	89.60	7.41	64
s3330	66.45	5345	16:13	78.84	3.94	0:09	88.29	3.94	5
в среднем:				**85.6**	**12.04**		**86.34**	**12.89**	

*Инструментальная платформа: Intel CoreQuad Q6400 2.4Ггц, 2Гбайт ОЗУ.

Алгоритмы моделирования доступны на вкладке «Моделирование». Для их работы необходимо описание схемы во внутреннем формате (*.out) и файл с входными последовательностями (*.tst). Результатом работы для программ исправного моделирования является файл с реакциями исправной схемы *.rea, который доступен для визуального просмотра на вкладке «Отчёты и диаграммы».

Вкладка «Генетический алгоритм» содержит программы подсистемы «GA-Analyze». Аналогично, на вкладке «Симуляция отжига» доступна подсистема «SA-Analyze».

Программы построения идентифицирующих последовательностей формируют следующие выходные файлы: тесты - *.tst, инициализирующие последовательности - *.ini, верифицирующие

эквивалентность - *.ver. Все они имеют одинаковый внутренний формат [13].

Программа построения избыточных тестов формирует файл *.rts, в котором в отличие от файла *.tst дополнительно отмечается начало каждой подпоследовательности.

Для работы программы оценки рассеиваемой мощности необходим файл описания схемы и файл с избыточными тестами. Результатом её работы является файл *.rpw, в котором содержится следующая информация (в порядке появления): число рассмотренных подпоследовательностей; общее число неисправностей в схеме; для каждой подпоследовательности с новой строки: длина, оценка рассеиваемой мощности в условных единицах, число проверяемых неисправностей, список номеров проверяемых неисправностей.

Программа построения энергоэффективных тестов на основе данного файла и файла с избыточными тестами строит файл *.mpt с требуемой входной последовательностью, который имеет структуру аналогичную *.rts.

Просмотр всех типов файлов, которые используются в системе, а также временных диаграмм доступен на вкладке «Отчёты и диаграммы».

Приведём эксплуатационные характеристики для новых программных компонент системы. Для апробации использовались схемы каталога ISCAS-89. В табл.1 приведены результаты экспериментов для программ построения тестов на основе алгоритма CO, а также программ построения инициализирующих последовательностей (ГА и CO). Табл.2 содержит численные результаты экспериментов для программ построения энергоэффективных тесов (алгоритмы ГА и CO). Для краткости изложения мы не приводим полные данные для программ верификации эквивалентности. Стратегия проводимых экспериментов верификации, а также числовые результаты приведены в [7] и [11]. Отметим, что для ГА в среднем удалось показать неэквивалентность для схем с различием в одном типе вентиля в 94.70% экспериментов, тогда как для алгоритма CO – в 96,71% экспериментов.

Приведённые числовые результаты говорят о высоких эксплуатационных характеристиках системы, что достигнуто благодаря объединению новых эффективных эволюционных алгоритмов и многозначных логик.

ВЫВОДЫ.

В статье описана новая версия системы моделирования и идентификации АСМИД-Evolution, для которой характерно широкое применение эволюционных алгоритмов. Рассмотрены основные компоненты системы, их программное наполнение и алгоритмы решения задач. В соответствии с тенденциями развития многопроцессорных и многоядерных рабочих станций разработаны параллельные версии алгоритмов генерации тестов и моделирования, описаны основные принципы их построения. Разработан ряд алгоритмов проектирования энергоэффективных схем. Система показывает высокие эксплуатационные характеристики благодаря совместному применению ad-hoc алгоритмов идентификации, основанных на эволюционном подходе, и многозначной логике. Применение системы позволяет повысить эффективность автоматизированного диагностирования цифровых устройств на различных этапах жизненного цикла их проектирования и эксплуатации.

ЛИТЕРАТУРА:

1. Скобцов Ю.А. Автоматизированная система моделирования и диагностики цифровых устройств /Ю.А. Скобцов, Г.Г. Пономаренко, А.П. Шатохин //УсиМ. – 1988. – №2. – С.11-16.
2. Скобцов Ю.А. Система логического моделирования и генерации тестов АСМИД-П /Ю.А. Скобцов, Ю.В. Скобцов //УсиМ. – 1996. – №1/2. – С.39-45.
3. Иванов Д.Е Автоматизированная система моделирования и генерации тестов АСМИД-Е /Д.Е. Иванов, Ю.А. Скобцов //Техническая диагностика и неразрушающий контроль. – 2000. – №2. – С.54-59.
4. Иванов Д.Е. Параллельное моделирование неисправностей для последовательностных схем /Д.Е. Иванов, Ю.А. Скобцов //Искусственный интеллект. – 1999. – №1. – С.44-50.