ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК МАСШТАБИРУЕМОСТИ МЕТОДА МОДЕЛИРОВАНИЯ ЦУ С НЕИСПРАВНОСТЯМИ ДЛЯ РАСПРЕДЕЛЁННЫХ ВС

УДК 004.3:681.518

ИВАНОВ Дмитрий Евгеньевич
к.т.н., доцент, с.н.с. отдела ТУС ИПММ НАН Украины.
Научные интересы: техническая диагностика цифровых систем,
эволюционные вычисления в технической диагностике, параллельные алгоритмы, генетические алгоритмы.
e-mail: ivanov@iamm.ac.donetsk.ua

ВВЕДЕНИЕ

Сложность современных СБИС всё ещё растёт в соответствии с законом Мура, предсказывающем удвоение числа транзисторов на кристалле каждые два года. Согласно Международной Дорожной Карте [1] число транзисторов в SOC-системах потребительского уровня увеличится в 17 раз к 2024 году. С другой стороны, длина входных последовательностей для тестирования одиночных константных неисправностей (ОКН) увеличится с нынешних 30 тысяч входных векторов в 16 раз к указанному времени. В связи с этим ускорение процедур моделирования цифровых устройств (ЦУ) с неисправностями является одной из ключевых задач, стоящих перед разработчиками. Подчёркивает этот факт и то, что такие процедуры применяются тысячи раз в процессе разработки [1]. Кроме проверки качества тестовых последовательностей моделирование с неисправностями применяется в методах генерации входных идентифицирующих последовательностей, которые основаны на моделировании. Сюда относятся различные эволюционные методы построения тестов, в частности генетические алгоритмы [2].

Одним из путей ускорения процедур моделирования является использование ресурсов параллельных вычислительных систем (ВС) различных классов, доступных разработчику [3-4]. Исследователями разрабатываются несколько направлений решения данной задачи. Разбиение схемы с целью параллельного моделирования её частей [5-6] эффективно в том случае, когда число вентилей очень велико, и характерно для того периода развития ВС, когда описание ЦУ не удавалось целиком разместить в доступной памяти. В настоящее время этому подходу уделяется меньше внимания. Подходы с разбиением списка неисправностей [7] и тестовой последовательности [8] более популярны, поскольку показывают хорошую производительность и масштабируемость. Такие методы также являются основой для асинхронных методов моделирования [9-10], дополнительной целью которых является обеспечение баланса загрузки процессоров в процессе работы. В последнее время интерес к параллельным методам моделирования ЦУ с неисправностями возобновился с развитием многоядерных процессоров [3, 11-13] и графических акселераторов с большим числом потоковых процессоров [14-15].

Анализ работ в данном направлении показывает, что исследование часто завершается приведением результатов численных экспериментов на доступной параллельной ВС. Однако важным также является свойство масштабируемости разрабатываемых методов. Без его обеспечения нельзя говорить о применении метода при расширении вычислительной мощности ВС путём добавления вычислительных узлов/процессоров.

Целью данной статьи является: на основании экспериментальных данных изучить свойства масштабируемости метода моделирования ЦУ с неисправно-

стями для параллельной ВС с распределённой памятью, основанного на статическом разбиении списка моделируемых неисправностей.

ПАРАЛЛЕЛЬНЫЙ МЕТОД МОДЕЛИРОВАНИЯ ЦУ С НЕИСПРАВНОСТЯМИ ДЛЯ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

Вычислительные кластеры, в которых узлы являются обычными рабочими станциями, а связь реализована по технологии FastEhternet, являются доступной альтернативой специализированным параллельным ВС. Они получили распространение в последнее десятилетие благодаря как небольшой стоимости, так и доступности компонент. Такой кластер содержит ряд вычислительных узлов и высокоскоростные линии связи между ними на основании протокола TCP/IP. Каждый вычислительный узел содержит процессор и оперативную память. Для разработчиков такой кластер представляет собой ВС с разделяемой памятью (системы класса МИМД по классификации Флинна).

Базовый метод моделирования с неисправностями, который исследуется в данной работе, подробно описан в [16]. Для ясности дальнейшего изложения приведём его краткое описание. Основная идея заключается в том, что при построении метода параллельного моделирования необходимо один из узлов кластера выбрать в качестве головного. Данный узел инициализирует весь процесс вычислений, производит взаимодействие с процессорами-клиентами, которые непосредственно выполняют моделирование поведения ЦУ с неисправностями. Таким образом, исходя из функционального назначения вычислительных узлов в системе, метод распределённого моделирования схем с неисправностью распадается на два независимых.

Первый метод реализуется на головном процессоре-сервере и осуществляет управляющие функции, а также файловый ввод-вывод. Второй метод непосредственно выполняет моделирование ЦУ с неисправностями и работает на нескольких доступных процессорах-клиентах. Причём каждый клиент выполняет моделирование заданной тестовой последовательности только на части полного списка неисправностей. Разбиение полного списка неисправностей осуществляется сервером до начала моделирования один раз и, следовательно, в данной схеме является статическим.

Диаграмма потоков обмена информацией между внешней средой (файловой системой), сервером и клиентами в процессе работы алгоритма приведена на рис. 1.

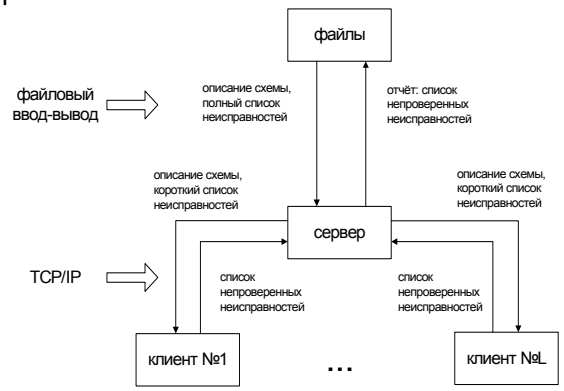

Рисунок 1 – Взаимодействие компонент метода распределённого моделирования ЦУ с неисправностями

Алгоритмическая реализация метода работы сервера в виде псевдокода представлена ниже.

Алгоритм А1

```
РаспределённоеМоделированиеСервер(имя_ЦУ, имя_теста)
{
 ЧислоКлиентов=ПоискКлиентов();
 if( ЧислоКлиентов != 0 )
 {
   A0 =ВводОписанияЦУ(имя_ЦУ);
   S =ВводТеста(имя_теста);
   F
=ПостроениеСпискаМоделируемыхНеисправностей(
 A0 );
  СортировкаСпискаНеисправностей( F );
  РазбиениеСпискаНеисправностей( F
,ЧислоКлиентов);
  for( int i=0 ; i< ЧислоКлиентов ; i++ )
   ПередатьКлиенту_i_ОписаниеЦУ( A0 );
  for( int i=0 ; i< ЧислоКлиентов ; i++ )
   ПередатьКлиенту_i_Тест( S );
  for( int i=0 ; i< ЧислоКлиентов ; i++ )
   ПередатьКлиенту_i_СписокНеисправностей( Fi );
  // передача данных завершена
  // переход к получению резальтатов моделирования
  for( int i=0 ; i< ЧислоКлиентов ; i++ )
   ПолучитьОтКлиен-
```

та_i_РезультатыМоделирования();
СформироватьОтчёт();
}
} // конец алгоритма – Сервер Моделирования

Дадим пояснение к данной реализации метода. Сервер начинает работу с поиска доступных клиентов. Дальнейшая работа выполняется в том случае, если найдены доступные для работы клиенты.

После ввода описания ЦУ A_0 и тестовой последовательности S происходит построение полного списка моделируемых неисправностей F. Далее данный список сортируется в процедуре *СортировкаНеисправностей()*. Цель данной процедуры заключается в том, чтобы неисправности, которые принадлежат одному пути распространения включались, по возможности, в одни и те же подсписки неисправностей. В [17] показано, что выбор метода сортировки в соответствии со стратегией «от внешних выходов во внутрь схемы» позволяет уменьшить число событий а, следовательно, и время моделирования на 5-9%. Далее список неисправностей разбивается на подсписки пропорционально числу найденных процессоров-клиентов: $F_1,...,F_l$, где l – число найденных клиентов.

После этого в цикле для каждого процесса-клиента выполняются следующие действия. Клиенту передаётся описание ЦУ A_0, подсписок неисправностей F_i и тестовая последовательность S.

Далее процесс-сервер переходит в режим ожидания результатов моделирования от клиентов. После получения списков непроверенных неисправностей от каждого из процессов-клиентов процесс-сервер формирует полный отчёт о моделировании: полнота тестовой последовательности, условная полнота, время моделирования на каждом процессе-клиенте, время обмена информацией с каждым процессом-клиентом и общее время работы.

Далее в виде псевдокода представлена реализация работы клиента.

Алгоритм А2

РаспределённоеМоделированиеКлиент()
{
сервер_найден=ПоискСервера();
if(сервер_найден == true)
{
A_0 =ПолучитьОписаниеЦУ();
F =ПолучитьСписокНеисправностей();
S =ПолучитьТест();
// непосредственно моделирование с неисправностями
МоделированиеСНеисправностями(A_0 , F , S)
ПередатьРезультатыМоделирования();
}
} // конец алгоритма – Клиент Моделирования

После запуска вычислительного процесса происходит поиск сервера вычислений. Если сервер найден, то клиент переходит в фазу ожидания данных. От сервера он получает следующие данные для моделирования: описание моделируемого ЦУ A_0, краткий список неисправностей F и входную последовательность S. После получения необходимых данных, происходит непосредственно моделирование работы ЦУ с неисправностями. В виде процедуры здесь реализован метод параллельного по разрядам машинного слова моделирования, разработанный авторами в [18]. Именно параллельное моделирование неисправностей в разрядах машинного слова отличает данный подход от описанного в [19]. По завершении процедуры моделирования на сервер передаются следующие данные: список непроверенных неисправностей, время работы клиента и время обмена информацией. После обмена данными с сервером клиент переходит в режим ожидания и готов для получения новых данных для моделирования.

РЕАЛИЗАЦИЯ МЕТОДА И РЕЗУЛЬТАТЫ ЧИСЛЕННЫХ ЭКСПЕРИМЕНТОВ

Для проверки эффективности предложенного подхода методы работы сервера и клиента распределённого моделирования ЦУ с неисправностями реализованы программно в среде программирования CodeGear. Программный код процесса сервера составил приблизительно 1000 строк. В качестве основы кода клиента брался код алгоритма [18], который претерпел незначительные изменения. В частности, файловый ввод-вывод описания схемы, списка неисправностей и

входной последовательности заменён на сетевой обмен. В качестве технологии взаимодействия использованы блокирующие сокеты. Данная технология означает, что процесс вычислений в точках приёма информации будет остановлен до тех пор, пока передающая сторона не отправит необходимые данные. Таким образом, в рассматриваемом подходе процедуры приёма информации от клиентов являются точками синхронизации.

Для проведения экспериментов использовался вычислительный кластер, построенный на базе локальной сети по технологии 100BaseT (со скоростью обмена информацией 100 Мбит/с). Элементы кластера, включая сервер, имеют следующие характеристики: процессор Intel Celeron 2400 МГц, объём ОЗУ 256 Мб.

С целью изучения условий эффективного применения предложенного метода, в процессе проведения машинных экспериментов проводился замер следующих временных характеристик: время передачи описания ЦУ процессу-клиенту, время передачи списка неисправностей процессу-клиенту, время передачи входной тестовой последовательности процессу-клиенту, время моделирования части списка неисправностей на каждом процессе-клиенте.

В табл. 1 приведены численные результаты работы параллельного метода моделирования ЦУ с неисправностями (алгоритмы А1 и А2) для пяти наибольших схем каталога ISCAS-89 [20]. Число клиентов в экспериментах равнялось восьми. В качестве входного теста выступала случайно сгенерированная последовательность, содержащая 1000 входных наборов. В данной таблице для каждой из контрольных схем приведены временные характеристики работы всех восьми клиентов и алгоритма в целом.

Для каждого клиента приведены следующие данные:

- время обмена информацией с сервером: время передачи описания ЦУ, передачи тестовой последовательности, передачи подсписка неисправностей данного клиента;
- общее время работы клиента – T;
- число событий моделирования исправного ЦУ – $N^{испр}$;
- число событий моделирования с неисправностями – $N^{неиспр}$.

Для метода в целом приведены:

- общее время работы метода для восьмипроцессорной реализации T_8;
- общее число событий моделирования исправного ЦУ A_0 и ЦУ с неисправностями.

Из приведённых данных видно, что существенные ограничения накладывает скорость обмена данными по соединительным каналам. Время передачи описания ЦУ в общем времени работы клиентов варьируется в среднем от 10% для наименьшей схемы s9234.ben до 1.3% для наибольшей схемы s38417.ben.

В табл. 1 время передачи тестовой последовательности для клиентов неодинаково и уменьшается с ростом его номера. Это связано с тем, что сюда дополнительно включено время ожидания клиентом начала передачи теста от момента завершения передачи описания ЦУ. Таким образом, для i-го клиента данное время включает время передачи описания ЦУ для клиентов $i+1,...,8$ плюс непосредственно время передачи теста клиенту i. Без учёта времени ожидания данное время для всех схем составляет менее одной секунды.

Аналогично, время передачи списка неисправностей каждому клиенту составляет менее одной секунды.

Также из таблицы видно, что число событий исправного моделирования для каждого из клиентов совпадает. Это связано с тем, что для определения проверяемости неисправностей на каждом клиенте помимо поведения ЦУ в присутствии неисправности необходимо знать точное поведение исправного ЦУ A_0. Поэтому на каждом клиенте производится моделирование поведения исправного ЦУ на заданной последовательности. Очевидно, что данная процедура в сравнении с однопроцессорной реализацией является избыточной с коэффициентом равным числу клиентов l.

Поведение каждого ЦУ в присутствии некоторой неисправности восстанавливается только на одном из клиентов. Поэтому рост числа событий моделирования с неисправностями должен быть относительно небольшим в сравнении с однопроцессорной реализацией.

Таблица 1 –

Результаты численных экспериментов алгоритма параллельного моделирования для схем из каталога ISCAS-89 при 8-процессорной реализации

имя схемы	обмен информацией		T, время работы, час.:мин.:сек	$N^{èñïð}$ - событий исправного моделирования	$N^{íåèñïð}$ - событий неисправного моделирования
	№ клиента	время передачи схемы / теста / неисправностей, сек.			
s9234	1	5 / 29 / 0	0:00:45	475129	59579219
	2	5 / 25 / 0	0:00:33	475129	43443725
	3	5 / 20 / 0	0:00:46	475129	58057854
	4	5 / 16 / 0	0:00:34	475129	43006081
	5	5 / 12 / 0	0:00:44	475129	56582162
	6	5 / 7 / 0	0:00:50	475129	63362723
	7	5 / 3 / 0	0:00:48	475129	62159082
	8	5 / 0 / 0	0:00:47	475129	59934347
	общее:		0:01:19	3801032	446125193
s13207	1	8 / 48 / 0	0:02:10	824041	134008066
	2	8 / 41 / 0	0:02:15	824041	134818432
	3	8 / 34 / 0	0:02:18	824041	142230941
	4	8 / 27 / 0	0:01:42	824041	102489140
	5	8 / 20 / 0	0:01:55	824041	116594471
	6	8 / 13 / 0	0:01:58	824041	118528980
	7	8 / 6 / 0	0:02:11	824041	133645846
	8	8 / 0 / 0	0:02:08	824041	128636522
	общее:		0:03:06	6592328	1010952398
s15850	1	9 / 61 / 0	0:04:14	1866392	297158465
	2	9 / 52 / 0	0:04:03	1866392	284583638
	3	9 / 43 / 0	0:03:13	1866392	224627174
	4	9 / 35 / 0	0:02:54	1866392	202233845
	5	9 / 26 / 0	0:03:40	1866392	256308655
	6	9 / 17 / 0	0:03:24	1866392	237100484
	7	9 / 8 / 0	0:03:40	1866392	256308922
	8	9 / 0 / 0	0:03:10	1866392	220354201
	общее:		0:05:24	14931136	1978675384
s35932	1	17 / 116 / 0	0:02:33	13219306	212636188
	2	17 / 99 / 0	0:08:26	13219306	723036895
	3	17 / 88 / 0	0:09:16	13219306	795805264
	4	17 / 66 / 0	0:09:00	13219306	774449580
	5	17 / 49 / 0	0:07:37	13219306	655311389
	6	17 /32 / 0	0:08:08	13219306	699907857
	7	17 / 15 / 0	0:08:42	13219306	746424994
	8	17 / 0 / 0	0:06:30	13219306	555461619
	общее:		0:11:31	105754448	5163033786
s38417	1	22 / 147 / 0	0:18:31	5911504	1017125962
	2	22 / 126 / 0	0:27:43	5911504	1522022582
	3	22 / 104 / 0	0:26:37	5911504	1455146297
	4	22 / 83 / 0	0:26:49	5911504	1470782477
	5	22 / 62 / 0	0:26:58	5911504	1481424307
	6	22 / 41 / 0	0:26:24	5911504	1449450364
	7	22 / 20 / 1	0:27:33	5911504	1507772565
	8	22 / 0 / 0	0:27:28	5911504	1510352828
	общее:		0:30:11	47292032	11414077382

ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК МАСШТАБИРУЕМОСТИ ПАРАЛЛЕЛЬНОГО МЕТОДА МОДЕЛИРОВАНИЯ ЦУ С НЕИСПРАВНОСТЯМИ

В качестве сравнительной базы для анализа масштабируемости предложенного параллельного метода бралась работа метода [18] на персональном компьютере с конфигурацией, которая соответствует конфигурации клиента в экспериментах выше и теми же входными последовательностями. Численные результаты экспериментов приведены в табл.2.

Таблица 2 –

Результаты численных экспериментов алгоритма параллельного моделирования для схем из каталога ISCAS-89 при однопроцессорной реализации

имя схемы	T_1^A, T_1^S, T_1^F сек.	T_1 час.:мин.:сек	$N_1^{испр}$	$N_1^{неиспр}$
s9234	5 / <1 / <1	0:05:46	475129	439582370
s13207	8 / <1 / <1	0:16:34	824041	999031455
s15850	9 / <1 / <1	0:28:15	1866392	1954774483
s35932	17 / <1 / 1	0:59:09	13219306	5075464453
s38417	22 / <1 / 1	3:30:41	5911504	11363924430

Здесь:

- T_1^A, T_1^S, T_1^F – время передачи описания ЦУ A_0, тестовой последовательности S и списка неисправностей F соответственно;
- T_1 – время работы последовательной версии алгоритма (однопроцессорная реализация);
- $N_1^{испр}$ – число событий при моделировании исправного ЦУ;
- $N_1^{неиспр}$ – число событий моделирования неисправного ЦУ.

В табл. 3. приведены результаты сравнения работы параллельного метода моделирования для 8 клиентов и однопроцессорной реализации (при обозначении характеристик масштабируемости методов будем использовать общепринятые обозначения [21]):

- H_8 – ускорение работы алгоритма, индекс показывает число клиентов;
- $N_8^{испр} / N_1^{испр}$ – увеличение числа событий исправного моделирования ЦУ;

- $N_8^{неиспр} / N_1^{неиспр}$ – увеличение числа событий моделирования ЦУ с неисправностями;
- N_8 / N_1 – общее увеличение числа событий моделирования ЦУ;
- E_8 – эффективность использования процессоров;
- f_8 – доля последовательного кода.

Таблица 3 –

Сравнение параллельной и последовательной версий алгоритма моделирования с неисправностями

Имя схемы	H_8, **раз**	$\frac{N_8^{испр}}{N_1^{испр}}$, **раз**	$\frac{N_8^{неиспр}}{N_1^{неиспр}}$, **раз**	$\frac{N_8}{N_1}$, **раз**	E_8	f_8
s9234	4.38	8	1.01	1.02	0.55	0.12
s13207	5.34	8	1.01	1.02	0.67	0.07
s15850	5.23	8	1.01	1.02	0.65	0.08
s35932	5.14	8	1.02	1.04	0.64	0.08
s38417	6.98	8	1.00	1.01	0.87	0.02

Из приведённых числовых данных можно сделать следующие выводы.

Параллельная версия метода показывает существенное ускорение работы в сравнении с последовательной: H_8 изменяется от 4.38 до 6.98. При этом с ростом размерности схем ускорение H_8 также растёт, что связано с уменьшением накладных расходов на пересылку данных клиентам. Этот факт также позволяет надеяться на дальнейшее повышение эффективности метода при росте размерности ЦУ.

Число событий моделирования поведения исправного ЦУ $N_8^{испр}$ увеличивается пропорционально числу клиентов, что связано с необходимостью восстанавливать исправное поведение на каждом из них.

Число событий моделирования ЦУ с неисправностями $N_1^{неиспр}$ и общее число событий N_8 практически не растут: коэффициенты роста 1.00-1.04. При этом для больших ЦУ данный коэффициент имеет тенденцию к снижению.

Доля последовательного кода f_8, которая влияет на максимальное возможное ускорение, также существенно снижается при росте размерности ЦУ. Это позволяет надеяться, что на практике данный алгоритм будет показывать ещё большую эффективность, поскольку реальные схемы существенно больше рассматриваемых здесь (схема максимальной размерности s38417.ben в каталоге ISCAS-89 имеет около 25 тысяч логических вентилей и около 38 тысяч неисправностей в списке).

Основными факторами, которые ограничивают масштабируемость являются следующие.

1. Избыточность моделирования поведения исправного ЦУ, выполняемое на каждом из клиентов. Выше отмечено, что данный фактор с ростом размерности ЦУ постепенно уменьшает своё влияние, однако в схеме параллелизации с разбиением списка неисправностей принципиально не устраним.

2. Скорость передачи информации по линиям связи процессоров в кластере.

Чтобы понять влияние второго фактора на масштабируемость параллельной версии метода, оценим его параметры, предположив, что обмен служебной информацией будет происходить мгновенно. Данное предположение является идеальным, однако близким к реальности. Например, если бы рассматриваемый выше эксперимент проходил на ВС с линиями связей, построенными по технологии 1000BaseT (скорость передачи данных увеличена в 10 раз по сравнению с проведёнными экспериментами), то для самой большой из рассматриваемых схем s38417.ben весь обмен информацией сервера с клиентами занял бы не более 10 секунд, а для наименьшей s9234.ben – менее секунды.

В табл. 4. приведено сравнение значения максимального ожидаемого ускорения, получаемое из закона Амдаля [21], для реальный числовых данных и для идеального случая (для паракомпьютера), описанного выше:

- f_8 – доля последовательного кода при восьмипроцессорной реализации;
- H_∞ – ожидаемое ускорение работы алгоритма при бесконечном числе процессоров:

$$H_\infty = 1/(f + (1-f)/\infty) = 1/f$$

Результаты для паракомпьютера вычислялись из предположения, что общее время работы параллельного алгоритма равно максимальному времени работы одного из клиентов.

Таблица 4 –

Максимально возможное ускорение параллельного алгоритма для реальной ВС и паракомпьютера

схема	данные экспериментов		идеальный случай	
	f_8	H_∞	f_8	H_∞
s9234	0.12	8.33	0.022	44.85
s13207	0.07	14.29	0.016	63.00
s15850	0.08	12.5	0.028	35.10
s35932	0.08	12.5	0.036	27.58
s38417	0.02	50	0.0075	133

Сравнивая столбцы H_∞ для реального и идеального случаев, можно видеть, что при повышении скорости обмена информацией по линиям связи может быть достигнуто ускорение работы параллельной версии алгоритма, которое в несколько раз превышает значение для реального эксперимента.

Такие оценки важны при рассмотрении возможности масштабирования инструментальной ВС, либо при адаптации метода к другим вычислительным средам. Безусловно, они являются достаточно приблизительными, ведь общее время работы существенно зависит от качества тестовой последовательности. Чем выше полнота прилагаемого теста, тем быстрее будут удаляться из списка проверенные неисправности и общее время работы будет ниже. Однако оценка для случайно сгенерированной последовательности применяется достаточно часто [9]. А тот факт, что такие последовательности имеют относительно низкие проверяющие свойства, говорит об консервативности приведённых оценок.

Также можно проследить рост эффективности характеристик параллельной версии алгоритма при росте числа процессоров-клиентов. Для оценки выберем схему s9234.ben, показавшую консервативные результаты.

В табл.5 приведены числовые данные при проведении машинных экспериментов со схемой средней размерности S9234.ben из набора контрольных схем ISCAS-89. В таблице столбцы показывают:

- T_i^A, T_i^S, T_i^F – время в секундах передачи описания ЦУ, тестовой последовательности и списка неисправностей соответственно;
- T_i – время моделирования ЦУ при числе клиентов равном i;
- $N_i^{испр}$ – число событий исправного моделирования при числе клиентов равном i;
- $N_i^{неиспр}$ – число событий моделирования с неисправностями при числе клиентов равном i.

Таблица 5 –

Численные результаты экспериментов алгоритма параллельного моделирования для схемы s9234.ben при различном числе клиентов

число клиентов, i	T_i^A, T_i^S, T_i^F сек.	T_i час.:мин.:сек	$N_i^{испр}$	$N_i^{неиспр}$
1	5 / 0 / 0	0:05:46	475129	439582370
2	3 / 3 / 0	0:03:14	950258	440855071
3	3 / 5 / 0	0:02:18	1425387	441793031
4	3 / 9 / 0	0:01:46	1900516	441882827
6	3 / 14 / 0	0:01:26	2850774	444938892
8	5 / 29 / 0	0:01:19	3801032	446125193

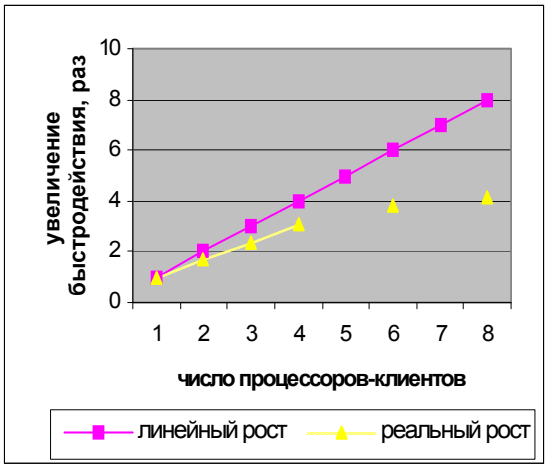

Рисунок 2 – Сравнительный рост реального и линейного быстродействия метода

На основании представленных числовых данных рассчитываются показатели масштабируемости параллельного алгоритма в зависимости от числа клиентов. Такие числовые результаты приведены в табл. 6. Здесь заголовки столбцов соответствуют таковым в табл. 3 с тем отличием, что параметры рассчитаны для i клиентов, а не для восьми.

Таблица 6 –

Значения параметров параллелизации алгоритма моделирования в зависимости от числа клиентов

число клиентов i	H_i, **раз**	$\frac{N_i^{испр}}{N_1^{испр}}$, **раз**	$\frac{N_i^{неиспр}}{N_1^{неиспр}}$, **раз**	$\frac{N_i}{N_1}$, **раз**	E_i	f_i
2	1.62	2	1.00	1.00	0.81	0.23
3	2.51	3	1.01	1.01	0.84	0.10
4	3.26	4	1.01	1.01	0.82	0.08
6	4.02	6	1.01	1.02	0.67	0.10
8	4.38	8	1.01	1.02	0.55	0.12

Приведённые числовые данные говорят о том, что:

- число событий исправного моделирования растёт пропорционально числу клиентов;
- число событий моделирования с неисправностями и общее число событий практически не растут: увеличение составляет около 2%;
- эффективность загрузки ядер имеет тенденцию к уменьшению с ростом числа клиентов;
- параметр доли последовательного кода в данном примере не репрезентативен.

Для наглядности на рис. 2 приведен график роста быстродействия при изменении числа процессоров-клиентов от 1 до 8 по данным моделирования для этой же схемы. В качестве сравнения на диаграмме присутствует график линейного роста быстродействия с коэффициентом равным 1.

ЗАКЛЮЧЕНИЕ

В статье на основании результатов численных экспериментов изучены свойства масштабирования метода параллельного моделирования ЦУ с неисправностями, основанном на схеме с разбиением списка моделируемых неисправностей и реализованном на вычислительном кластере с распределённой памятью.

В общем следует отметить, что изучаемый метод параллельного моделирования ЦУ с неисправностями обладает достаточно высокими характеристиками масштабируемости, позволяя существенно увеличить быстродействие данного процесса при реализации на параллельных ВС с распределённой памятью. Оценки показывают, что рост числа процессоров в кластере обеспечит дальнейший рост быстродействия для всех контрольных схем.

Основным сдерживающим фактором, который препятствует линейному росту быстродействия, является скорость обмена по линиям связи в вычислительном кластере. Произведена оценка эффективности метода для паракомпьютера, позволяющая говорить об улучшении характеристик масштабируемости при реализации на параллельной ВС с большей пропускной способностью каналов связи между узлами.

В противоположность оценкам [9] наши числовые данные показывают, что необходимость моделирования поведения ЦУ на каждом из узлов кластера не является существенным ограничителем роста быстродействия. Вклад данного фактора составляет от 1 до 2% в общем числе событий моделирования.

Также для исследуемого метода имеется тенденция к повышению характеристик параллелизации при росте размерности моделируемых ЦУ. Это даёт основания надеяться, что на реально проектируемых ЦУ данные параметры будут ещё выше.

ЛИТЕРАТУРА:

1. ITRS 2010 technology roadmap [jelektronnyj resurs] //http://www.itrs.net/Links/2010ITRS/Home2010.htm/. – Zagl. s jekrana.- (1.06.2013).
2. D.E. Ivanov Geneticheskie algoritmy postroenija vhodnyh identificirujushhih posledovatel'nostej cifrovyh ustrojstv /Ivanov D.E. – Doneck, 2012. – 240 s.
3. Kochte M.A. Efficient Fault Simulation on Many-Core Processors /M.A. Kochte, M. Schaal, H.-J. Wunderlich, C.G. Zoellin //Proceedings of the 47th Design Automation Conference ACM, New York, NY, USA. – 2010. – Pp.380-385.
4. Dömer R. Multi-core parallel simulation of system-level description languages /R. Dömer, W. Chen, X. Han /Proceedings of the 16th Asia and South Pacific Design Automation Conference, 2011. – P.311-316.
5. Mueller-Thuns R.B. Portable parallel logic and fault simulation /R.B. Mueller-Thuns, D.G. Saab, R.F. Damiano, J.A. Abraham //Proc. Int. Conf. CAD. – 1989. – Pp.506-509.

6. Ghosh S. A distributed algorithm for fault simulation of combinatorial and asynchronous sequential digital designs, utilizing circuit partitioning, on loosely coupled parallel processors /S. Ghosh //Microelectronic Reliability. – 1995. – №35(6). – P.947-967.
7. Han J. A Parallel Implementation of Fault Simulation on a Cluster of Workstations /J. Han, S.Y. Lee //Proc. IEEE International Symposium Parallel and Distributed Processing IPDPS. – 2008. – Pp.1-8.
8. Amin M.B. Data Parallel-Fault Simulation /M.B. Amin, B. Vinnakota //IEEE Trans. VLSI Systems. – 1999. – Vol.7. – №2. – Pp.183-190.
9. Krishnaswamy D. Asynchronous parallel algorithms for test set partitioned fault simulation /D. Krishnaswamy, P. Banerjee, E.M. Rudnick, J.H. Patel //ACM SIGSIM Simulation Digest. – 1997. – Volume 27, Issue 1. – P.30-37.
10. Ivask E. Distributed fault simulation with collaborative load balancing for VLSI circuits /E. Ivask, S. Devadze, R. Ubar //Scalable Computing: Practice and Experience. – 2011. – Vol.12, №1. – P.153-163.
11. Gillespie, M. Masshtabirovanie programmnyh arhitektur dlja mnogojadernyh vychislitel'nyh sistem budushhego [Jelektronnyj resurs] /Matt Gillespie //Intel® Software Network. – Rezhim dostupa: http://software.intel.com/ru-ru/articles/scaling-software-architectures-for-the-future-of-multi-core-computing/. – Zagl. s jekrana. – (1.06.2009).
12. Ivanov D.E. Parallel fault simulation on multi-core processors /D.E. Ivanov //«Radioelektronni i komp'juterni sistemi», 2009. – №6 (40). – S.109-112.
13. Ivanov D.E. Parallel'nyj algoritm modelirovanija cifrovyh shem s neispravnostjami dlja mnogojadernyh sistem s obshhej pamjat'ju /D.E. Ivanov //Jelektronnoe modelirovanie. – 2011. – T.33, №2. – S.93-106.
14. Gulati K. Towards acceleration of fault simulation using graphics processing units /K. Gulati, S. Khatri //Proceedings of the 45th annual Design Automation Conference, DAC '08. – 2008. – P.822-827.
15. Perinkulam A. Logic simulation using graphics processors /A. Perinkulam, S. Kundu //In Proc. ITSW, 2007. (v jelektronnom vide)
16. Ivanov D.E. Raspredeljonnoe parallel'noe modelirovanie cifrovyh shem s neispravnostjami /D.E. Ivanov, Ju.A. Skobcov, Jel'-Hatib A.I. //Naukovi praci Donec'kogo nacional'nogo tehnichnogo universitetu. Serija: «Obchisljuval'na tehnika ta avtomatizacija». Vipusk 107. – Donec'k: DonNTU. – 2006. – S.128-134.
17. Maamari F. A fault simulation method based on stem regions /F. Maamari, J. Raiski //Proc. of Int. conference on Computer Aided Design. – 1988. – Pp.170-173.
18. Ivanov D.E. Parallel'noe modelirovanie neispravnostej dlja posledovatel'nostnyh shem /D.E. Ivanov, Ju.A. Skobcov //Iskusstvennyj intellekt. – 1999. – №1. – S.44-50.
19. Parker S. A parallel algorithm for fault simulation based on PROOFS /S. Parker, P. Banerjee, J. Patel //Proc. IEEE Int. Conf. Computer Design. – 1995. – P.616-621.
20. Brgles F. Combinational profiles of sequential benchmark circuits /F. Brgles, D. Bryan, K. Kozminski //International symposium of circuits and systems, ISCAS-89. – 1989. – P.1929-1934.
21. Gergel', V.P. Osnovy parallel'nyh vychislenij dlja mnogoprocessornyh vychislitel'nyh sistem. Uchebnoe posobie /Gergel' V.P., Strongin R.G. – Nizhnij Novgorod: Izd-vo NNGU im. N.I. Lobachevskogo. – 2003. – 184 s.

Рецензент: *д.т.н., проф. Скобцов Ю.А., Донецкий национальный технический университет, Донецк.*