

# РАСПРЕДЕЛЁННОЕ ПАРАЛЛЕЛЬНОЕ МОДЕЛИРОВАНИЕ ЦИФРОВЫХ СХЕМ С НЕИСПРАВНОСТЯМИ

Скобцов Ю.А., д.т.н., проф., кафедра АСУ, Донецкий национальный технический университет,  
Иванов Д.Е., к.т.н., с.н.с., отдел ТУС, Институт Прикладной Математики и Механики.  
E-mail: [skobtsov@kita.dgtu.donetsk.ua](mailto:skobtsov@kita.dgtu.donetsk.ua), [ivanov@iamm.ac.donetsk.ua](mailto:ivanov@iamm.ac.donetsk.ua)

*Fault simulation is one of the most highly compute-intensive tasks in technical diagnostics. One of the ways to speed-up this process is parallelization on the calculation cluster. In this paper a distributed algorithm for fault simulation of digital circuits is presented. It is based on the well-known «master-slave» approach in which one processor is nominating as a master and rules all calculation on the all slave's processors. To reach the maximal utilization of the processors in the cluster it is used schema with static fault list partitioning.*

## 1. Введение.

Одна из центральных задач технической диагностики – моделирование цифровых схем с неисправностями. Данное моделирование используется в основном для определения диагностических свойств тестовых последовательностей. В настоящее время предложено ряд быстрых алгоритмов моделирования цифровых схем с неисправностями [1, 2, 3]. Все они для повышения быстродействия используют стратегии параллельного одиночного распространения неисправностей и динамического сжатия списка неисправностей. В последней неисправность удаляется из списка в тот момент времени, когда она обнаруживается, и дальнейшее её моделирование не осуществляется. Однако непрерывное увеличение размерности проектируемых устройств оставляет задачу моделирования среди наиболее актуальных. Один из способов, с помощью которого возможно повышение скорости данного процесса, является расширение существующих алгоритмов для работы на многопроцессорных вычислительных системах (кластерах).

Распределённое моделирование схем с неисправностями возможно осуществить двумя путями. Первый заключается в том, что выделяется один процессор-сервер, который управляет списком неисправностей. Он передаёт процессорам-клиентам для моделирования часть данного списка и получает от них результаты моделирования. Задача процессора-клиента получить описание схемы, список неисправностей, произвести моделирование и передать на сервер его результаты. В работах [4, 5] предложены алгоритмы данного типа, которые позволяли увеличить скорость моделирования от двух до шести раз на восьмипроцессорном кластере. Для небольших схем увеличение быстродействия приближалось к единице.

Предпринимались также попытки построения распределённых алгоритмов моделирования для многомашинной платформы с общей памятью, в которых моделирование работы всей схемы осуществляется разбиением схемы на части с последующим моделированием каждой на своём процессоре [6, 7]. Однако предложенные алгоритмы не позволяли достичь одинаково сильной загрузки процессоров в кластере. Известны также работы, в которых изучается эффективность передачи потоков данных между сервером и клиентами в зависимости от разбиения схемы и применяемых протоколов связи [8].

Предлагаемый в данной статье алгоритм моделирования построен по принципу разбиения списка моделируемых неисправностей с выделением одного процессора в качестве сервера. В качестве среды обмена информацией выступает протокол TCP/IP, что позволяет включать в вычислительный кластер как машины с одной вычислительной платформой, так и построенные на разных платформах. Алгоритм распределённого моделирования схем с неисправностью распадается на два независимых алгоритма. Первый алгоритм реализуется на головном процессоре, который называется «сервером», и осуществляет управляющие функции, а также

файловый ввод-вывод. Второй алгоритм непосредственно выполняет моделирование схем с неисправностями и работает на нескольких доступных процессорах-«клиентах». Число доступных процессоров в кластере определяется сервером до начала моделирования. В предлагаемом алгоритме каждый клиент выполняет моделирование заданной тестовой последовательности только на части полного списка неисправностей. Разбиение полного списка неисправностей осуществляется сервером до начала моделирования и, следовательно, является статическим.

Статья имеет следующую структуру. В введении обосновывается актуальность поставленной задачи. Во втором разделе кратко описывается разработанный ранее авторами алгоритм параллельного моделирования схем с неисправностями, который используется в качестве основного на процессорах-клиентах. В третьем разделе описывается алгоритм распределённого моделирования схем для многопроцессорной системы с разделённой памятью. В четвёртом разделе описываются проведённые машинные эксперименты и полученные данные, а также изучаются вопросы эффективности применения предложенного алгоритма при моделировании больших схем. В заключении делаются выводы и предлагаются направления дальнейших исследований.

## 2. Алгоритм параллельного моделирования цифровых схем с неисправностями.

На каждом элементе кластера, который использовался для моделирования в качестве клиента, реализован алгоритм параллельного по неисправностям моделирования с динамическим сжатием неисправностей [3]. Данный алгоритм показывает хорошие временные характеристики для тестовых схем из каталога ISCAS-89 [9]. Он претерпел незначительные изменения, которые касаются только ввода схемы, списка неисправностей и тестовой последовательности. В данном случае ввод всех данных реализован не из файла, а с помощью технологии сокетов, позволяющей обмениваться данными по локальным и глобальным сетям с помощью протокола TCP/IP. Заметим также, что данный факт позволяет объединять в кластер

```
параллельное_моделирование( схема, тест, список_неисправностей )
{
  while ( есть тестовые наборы )
  { // цикл по тестовым наборам
    смена_тестового_набора(); // ввод тестового набора
    параллельное_моделирование_исправной_схемы ();
    пока( для данного тестового вектора есть неисправности )
    {
      формирование_группы_неисправностей(); //по неисправностям
      внесение_неисправностей();
      восстановить_значения_узлов_состояний();
      внесение_неисправностей_на_внешних_выходах();
      моделирование_на_наборе_неисправной_схемы();
      внесение_неисправностей_на_внешних_выходах();
      проверка_обнаружимости_неисправностей();
      запоминание_неисправных_узлов_состояний();
      перемещение_условно_проверяемых_неисправностей();
    }
  } // конец циклов по неисправностям и тестовым наборам;
  запись_непроверенных_неисправностей_в_файл_на_диск();
} // конец алгоритма
```

Рис.1 Алгоритм параллельного моделирования.

для решения поставленной задачи процессоры, работающие под управлением различных типов операционных систем, например Windows и Unix.

Опишем кратко алгоритм работы клиента. Алгоритм предназначен для моделирования константных неисправностей в комбинационных и синхронных последовательностных схемах. Для моделирования используется 3-значный алфавит  $E_3=\{0,1,u\}$ . Параллельное по тестовым наборам моделирование неисправных схем производится в соответствии с алгоритмом, представленным в форме псевдокода на рис.1.

Отметим несколько ключевых моментов, которые существенно влияют на быстродействие данного алгоритма:

- динамическое изменение списка неисправностей: в данном алгоритме неисправность удаляется из списка в тот момент времени, когда она была обнаружена; дальнейшее моделирование для данной неисправности не выполняется;
- сортировка неисправностей: в алгоритме применяется сортировка от «внешних выходов в глубину», что позволяет включать в группы для моделирования неисправности, которые вызывают одни и те же события;
- функциональное внесение неисправностей: данный метод позволяет избежать всех лишних проверок необходимости внесения неисправности для выводов вентиля; для вентилях, имеющих на выходах неисправности, создаются фиктивные элементы, позволяющие только для них обрабатывать проверку и внесение неисправностей;
- сохранение элементов состояний неисправных схем; для каждого входного набора производится сохранение элемента состояния (триггера) только в том случае, если его значение отлично от значения в исправной схеме.

### 3. Алгоритм распределённого моделирования цифровых схем с неисправностями.

Предлагаемый алгоритм распределённого параллельного моделирования относится к классу алгоритмов, которые для ускорения процесса моделирования разбивают полный список неисправностей на части. Сервер выполняет только управляющие функции: ввод описания

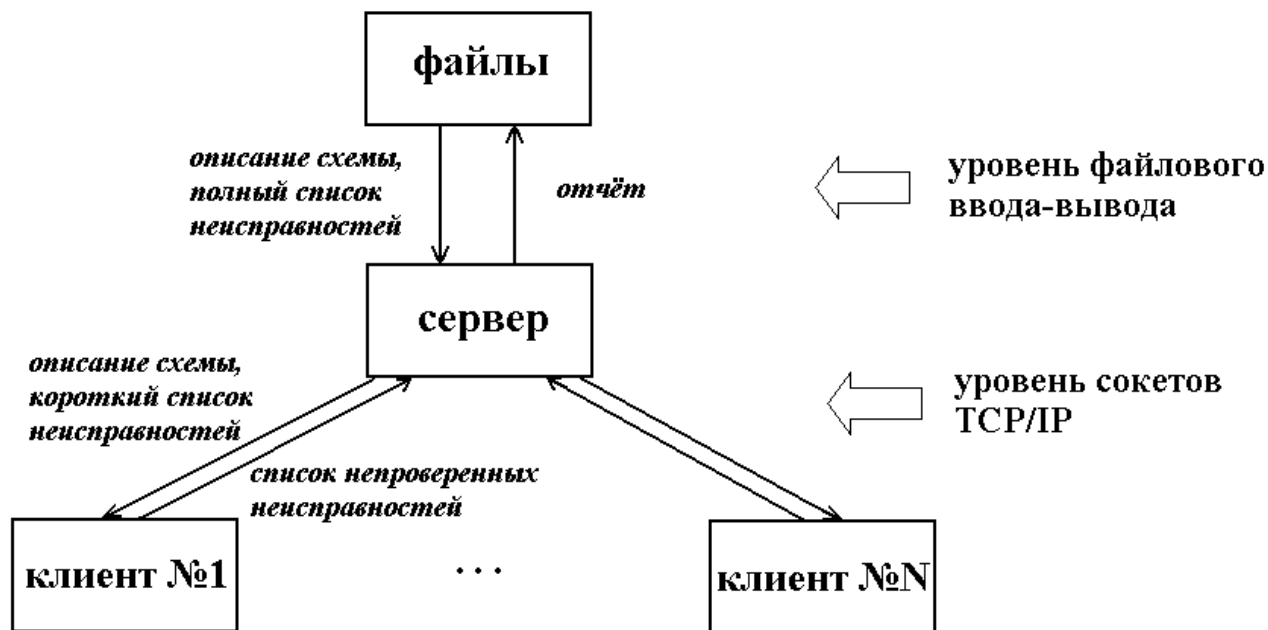


Рис.2 Диаграмма потоков данных при распределённом моделировании.

схемы, передача схемы и списка неисправностей клиентам, получение от них результата и формирование отчёта. Каждый из доступных клиентов выполняет параллельное моделирование на данных, полученных от сервера, и возвращает список непроверенных неисправностей. Опишем подробнее работу сервера и клиентов в предложенном алгоритме.

Диаграмма обмена информацией в процессе работы алгоритма приведена на рис.2. Передача данных в алгоритме происходит на двух уровнях. Файловый ввод-вывод применяется процессом-сервером для получения описания схемы, полного списка неисправностей и входной последовательности. Весь обмен информацией с процессами-клиентами происходит с помощью технологии TCP/IP сокетов, позволяющей обмениваться информацией между вычислительными процессами на любых платформах с настроенным протоколом TCP/IP. В предложенном алгоритме обмен происходил в локальной сети учебной аудитории со скоростью 100Мбит/сек.

Псевдокод работы серверного процесса приведён на рис.3. Сервер начинает работу с ввода описания схемы, полного списка неисправностей и тестовой последовательности. Данные процедуры осуществляются с помощью файлового ввода-вывода. После этого происходит запуск сокета-сервера и поиск сокетов-клиентов. Если во время описка обнаружены процессы-клиенты, то полный список неисправностей разбивается пропорционально их числу. Далее в цикле для каждого процесса-клиента выполняются следующие действия. Клиенту передаётся описание схемы, часть списка неисправностей и тестовая последовательность. После чего процес-сервер переходит в режим ожидания результатов. После получения списков непроверенных неисправностей от каждого из процессов-клиентов процесс-сервер формирует полный отчёт моделирования: полнота тестовой последовательности, условная полнота, время моделирования на каждом процессе-клиенте, время обмена информацией с каждым процессом-клиентом и общее время работы.

Псевдокод работы процесса-клиента приведён на рис.4. После запуска вычислительного

```
распределённое_моделирование(схема, тест)
{
    число_клиентов = поиск_клиентов();
    если( число_клиентов != 0 )
    {
        ввод_схемы();
        ввод_теста();
        построение_полного_списка_неисправностей();
        разбиение_списка_неисправностей();
        для( i=0 ; i< число_клиентов ; i++ )
        {
            передать_клиенту_i_описание_схемы();
            передать_клиенту_i_часть_списка_неисправностей();
            передать_клиенту_i_тест();
        }
        для( i=0 ; i< число_клиентов ; i++ )
        {
            получить_список_непроверенных_неисправностей();
        }
        формирование_отчёта();
    }
}
```

Рис.3 Алгоритм работы процесса-сервера.

```

моделирование_с_неисправностями()
{
    поиск_сервера();
    если( сервер_найден )
    {
        получить_описание_схемы();
        получить_список_неисправностей();
        параллельное_по_наборам_моделирование(схема, тест, список)
        передать_список_непроверенных_неисправностей();
    }
}

```

Рис.4 Алгоритм работы процесса-клиента.

процесса происходит поиск сервера вычислений. Если сервер найден, то клиент переходит в фазу ожидания данных. От сервера он получает следующие данные для моделирования: описание схемы, краткий список неисправностей и входную последовательность. После получения необходимых данных, происходит моделирование работы цифровой схемы с неисправностями на данном множестве неисправностей. Это происходит в функции «параллельное\_по\_наборам\_моделирование()». Отметим, что работа выполняется по алгоритму, описанному в разделе 2. По завершении 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 и экспериментальные данные.**

Предложенный алгоритм распределённого моделирования реализован программно с технологией блокирующих сокетов в среде программирования C++ Builder 6. Технология блокирующих сокетов означает, что процесс вычислений в точках приёма информации будет остановлен до тех пор, пока передающая сторона не отправит необходимые данные.

Код процесса сервера составил приблизительно 300 строк. В качестве основы кода клиента брался код алгоритма [3], который претерпел незначительные изменения. В частности, файловый ввод-вывод описания схемы, списка неисправностей и входной последовательности заменён на сетевой обмен с помощью технологии блокирующих сокетов. По этой же технологии добавлена передача отчёта на сервер кластера вычислений.

Для проведения экспериментов использовался вычислительный кластер, построенный на базе локальной сети 100 Мбит одной из учебных аудиторий. Элементы кластера, включая сервер, имеют следующие характеристики: процессор Intel Celeron 2000МГц, объём ОЗУ 256Мбайт, операционная система Windows XP.

С целью изучения условий эффективного применения предложенного алгоритма, в процессе проведения машинных экспериментов проводился замер следующих временных характеристик: общее время процесса моделирования, число событий при моделировании исправной схемы и схем с неисправностями, общее число событий. Для сравнительной базы бралась работа алгоритма авторов [3] на персональном компьютере соответствующей конфигурации.

В табл.1 приведены числовые данные при проведении машинных экспериментов со схемой средней размерности S9234.ben. Данная схема имеет следующие характеристики: число входов – 19, число выходов – 22, число D-триггеров – 228, число инверторов – 3570, число вентилях других типов – 2027, общее число элементов – 5866. В таблице в скобках при многопроцессорной реализации указано увеличение быстродействия (для времени моделирования) и измеряемого параметра при сравнении с однопроцессорной реализацией алгоритма [3].

Таблица 1. Экспериментальные данные для схемы S9234.ben

Характеристика	Однопроцессорная реализация	Многопроцессорная реализация (число процессоров)					
		1	2	3	4	6	8
длина теста	1000	1000					
общее время моделирования, сек.	330	336 (0,98)	194 (1,7)	138 (2,39)	107 (3,08)	86 (3,83)	79 (4,17)
общее число событий, млн.	441,51	440,05 (1,00)	441,81 (1,00)	443,21 (1,00)	443,78 (1,01)	447,79 (1,01)	449,93 (1,02)
число событий моделирования исправной схемы, млн.	0,48	0,48 (1,00)	0,95 (1,98)	1,42 (2,96)	1,90 (3,96)	2,85 (5,93)	3,80 (7,92)
число событий моделирования схем с неисправностями, млн.	441,03	439,58 (1,00)	440,86 (1,00)	441,79 (1,00)	441,88 (1,00)	444,94 (1,01)	446,13 (1,01)

При анализе видно, что число событий при моделировании работы схем с неисправностями практически не растёт (коэффициент  $\approx 1$ ). Это говорит о том, что при предложенной схеме работы алгоритма распараллеливания отсутствует избыточное моделирование схем с неисправностями. С другой стороны, число событий при моделировании работы исправной схемы растёт с коэффициентом близким к числу задействованных процессоров: от 1,00 до 7,92 при увеличении числа процессоров-клиентов от 1 до 8. Это обусловлено необходимостью моделирования работы исправной схемы на всех процессорах-клиентах. Общее же число событий растёт несущественно: коэффициент изменяется от 1,00 до 1,02 при увеличении числа процессоров-клиентов от 1 до 8. Это связано с тем фактом, что число событий при моделировании исправной схемы составляет менее 1% от общего числа событий.

На рис.5 приведен график роста быстродействия при изменении числа процессоров-клиентов от 1 до 8 по данным моделирования схемы S9234.ben. Для сравнения на диаграмме присутствует график линейного роста быстродействия с коэффициентом равным 1.

Приведённые экспериментальные данные подтверждает эффективность предложенной схемы параллелизации алгоритма моделирования. Однако полностью линейного увеличения быстродействия достичь не удастся. Основным препятствием для этого являются необходимость многократной (по числу задействованных процессоров-клиентов) передачи описания схемы, а также избыточное моделирование работы исправной схемы на каждом рабочем процессоре.

#### **Заключение и дальнейшие исследования.**

В работе исследована актуальная научная задача: возможность распределённого моделирования цифровых схем с неисправностями на вычислительном кластере, состоящем из нескольких рабочих процессоров.

В работе предложен алгоритм распределённого моделирования цифровых схем с неисправностями, который использует схему «сервер-клиент» со статическим разбиением списка неисправностей и параллельным моделированием работы цифровой схемы на процессорах-клиентах. Предложенная схема показала высокую эффективность распараллеливания процесса моделирования и простоту реализации. В качестве перспективных дальнейших исследований можно отметить следующие направления:

- изучение зависимости скорости моделирования от способа группировки неисправностей, передающихся на моделирование одному клиенту;

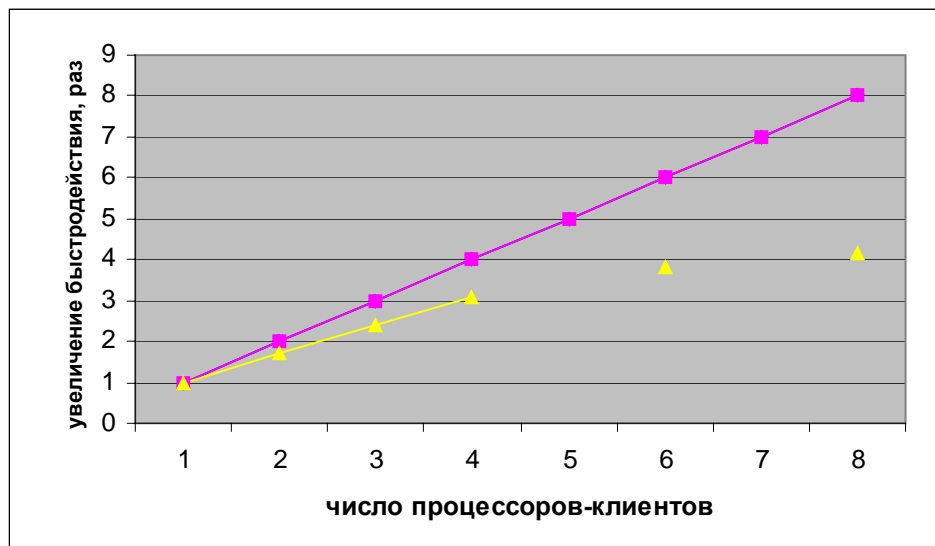


Рис.5 Рост быстродействия при увеличении числа процессоров-клиентов.

- изучение возможности применения построенных алгоритмов-клиентов для быстрого вычисления оценочных функций в генетическом алгоритме генерации тестов;
- изучение возможности применения схемы «клиент-мастер» для построения распределённого генетического алгоритма генерации тестов.

### Литература.

1. 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.
2. Kung C.P., Lin C.S. HyHope: A Fast Fault Simulator with Efficient Simulation of Hypertrophic Faults // Proc. of International Test Conference. - 1994. - P.714-718.
3. Иванов Д.Е., Скобцов Ю.А. Параллельное моделирование неисправностей для последовательных схем // Искусственный интеллект. – 1999. - №1. – С.44-50.
4. P.A. Duba, R.K. Roy, J.A. Abraham and W.A. Rogers, “Fault simulation in a distributed environment”, in Proceedings of the 25<sup>th</sup> ACM/IEEE Design Automation Conference, pp.686-691, June 1988.
5. T. Marcas, M. Royals and N. Kanopoulos, “On distributed fault simulation”, IEEE Computer, vol. 7, pp. 40-52, Jan. 1990.
6. S. Patil, P. Banerjee and J. Patel, “Parallel test generation for sequential circuits on general purpose multiprocessors”, in Proceedings of the 28<sup>th</sup> ACM/IEEE Design Automation Conference, (San Francisco, CA), June 1991.
7. S. Ghost, “NODIFS: a novel, distributed circuit partitioning based algorithm for fault simulation of combinational and sequential digital design on loosely coupled parallel processors”, tech. rep., LEMS, Division of Engineering, Brown University, Providence, RI, 1991.
8. Ладыженский Ю.В., Попов Ю.В. Программная система для исследования протоколов синхронизации при распределённом событийном логическом моделировании // Наукові праці Донецького національного технічного університету, Серія “Обчислювальна техніка та автоматизація”.- 2004.- Выпуск №74.- С.201-209.
9. Brgles F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89. – 1989. – P.1929-1934.