Для цитирования: Бененсон М.З., Дивин А.П. Методы построения тестового программного обеспечения многопроцессорных вычислительных систем // Вопросы радиоэлектроники. 2018. № 5. С. 95–103.
DOI 10.21778/2218-5453-2018-5-95-103
УДК 004.272.43

М.З. Бененсон[1], А.П. Дивин[1]

[1] АО «Научно-исследовательский институт вычислительных комплексов им. М.А. Карцева»

МЕТОДЫ ПОСТРОЕНИЯ ТЕСТОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Статья посвящена различным аспектам создания тестового программного обеспечения для многопроцессорных вычислительных систем (платформ). Все более актуальной становится задача разработки эффективных тестовых проверок, позволяющих определить работоспособность и функциональные характеристики гетерогенных вычислительных систем. Для разработки тестов были использованы разнообразные инструментальные средства: кроссплатформенный компилятор QT 5.2, библиотека boost, технология CUDA, программный пакет MPICH2, библиотека QWT, командный процессор Bash. Проектирование тестового программного обеспечения велось с помощью шаблонов, разработанных с целью построения требуемой программной архитектуры. В статье описаны методы построения управляющей программы «тестовый диспетчер», которая предоставляет возможность запуска тестов в многопоточном режиме. Рассматриваются методы построения тестовых программ для проверки различных модулей гетерогенной многопроцессорной вычислительной платформы. Результаты проведенных приемосдаточных испытаний подтверждают высокую эффективность предложенных методов тестирования многопроцессорной вычислительной системы.

Ключевые слова: *тесты, программное обеспечение, многопроцессорная вычислительная система, тест производительности, распараллеливание операций, характеристики вычислительных систем.*

Введение

В последние годы широко и все более быстрыми темпами осуществляется разработка, выпуск и внедрение в автоматизированные системы обработки информации и управления многопроцессорных вычислительных систем (платформ). Многопроцессорные вычислительные системы отличаются повышенной гибкостью (модульная структура) и характеризуются высокими показателями производительности, надежности и живучести.

Актуальной задачей становится разработка для таких систем тестового программного обеспечения, которое позволяет определить работоспособность и функциональные характеристики системы.

Результаты исследований

В данной статье рассматриваются методы построения тестовых программ для проверки многопроцессорной вычислительной платформы (МВП), предназначенной для решения различных прикладных задач.

МВП является гетерогенной вычислительной системой, в которой создание проблемно-ориентированной конфигурации достигается за счет выбора и установки в вычислительную платформу набора модулей, необходимого для достижения максимальной эффективности выполнения алгоритмов решаемых задач. Все аппаратные компоненты МВП являются модулями отечественного производства.

На рис. 1 показан один из вариантов (базовая конфигурация) структурной схемы МВП [1].

В МВП входят следующие модули:

- модуль коммутации KIC551, позволяющий на одной шине объединять процессорные модули, построенные на базе разнородных архитектур;
- модуль CPC512 с архитектурой x86, созданной на базе четырехядерного процессора Intel Ivy Bridge с оперативной памятью 8 Гбайт;
- модуль процессора «Эльбрус»;
- модуль процессора «Байкал»;
- модули VIM556, реализованные на базе высокопроизводительной графической карты NVIDIA;
- модуль FPU500, созданный на базе программируемых логических интегральных схем (ПЛИС);
- модуль интерфейсный оптический KIC552 отечественного производства;
- модуль KIC550 для поддержки жестких дисков с интерфейсами SATA и SATA II.

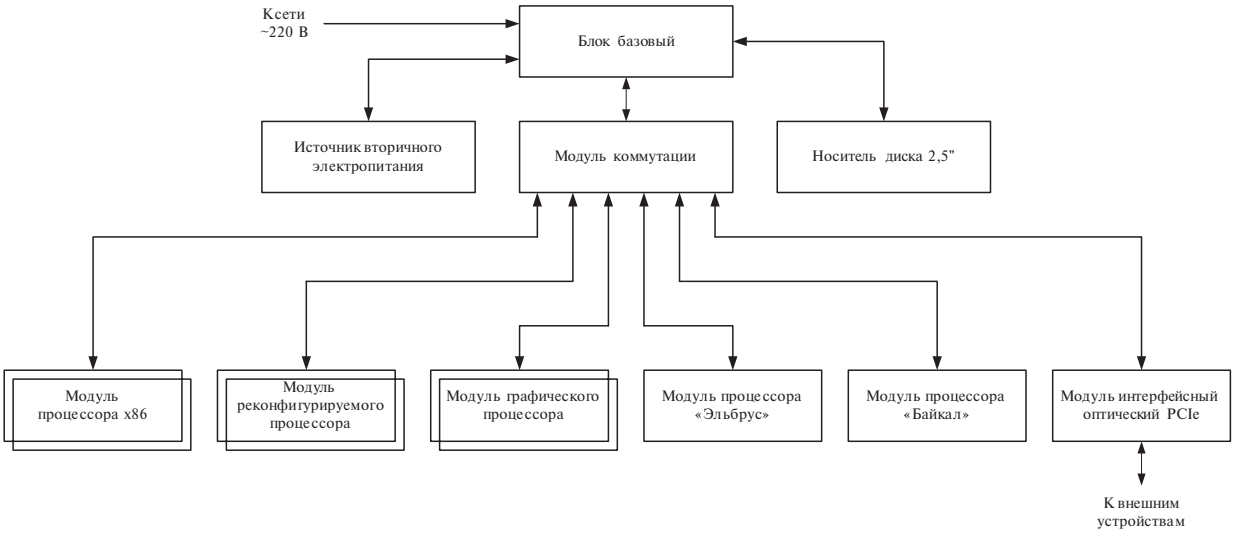

Рисунок 1. Структурная схема МВП

К каждому модулю CPC512 через порт SATA может быть подключен свой жесткий диск.

Связь между процессорами осуществляется через открытый стандарт PCI Express 3.0 с помощью модуля коммутации KIC551; кроме того, на модуле KIC551 установлен коммутатор 1Gb Ethernet, позволяющий реализовывать управление модулями системы.

Базовое программное обеспечение МВП включает в себя:

- ОС Linux одного из возможных дистрибутивов (Ubuntu, Debian, AstraLinux и другие), предназначенную для функционирования на средствах вычислительной техники с процессорной архитектурой x86–64;
- драйверы, которые создают между процессорами CPC512 стандартное сетевое (Ethernet) соединение, обеспечивающее передачу пакетов по шине PCI Express, и позволяют работать с контроллером шины PCI Express таким же образом, как с обычной сетевой картой.
- ОС «Эльбрус», основанная на базе ядра Linux 2.6.33 и рекомендованная разработчиком для процессорных модулей на базе процессоров с архитектурой «Эльбрус».

В связи с тем, что МВП является гетерогенной многопроцессорной системой, в процессе разработки были использованы инструментальные программные средства:

- Кроссплатформенный компилятор QT 5.2.
- Многофункциональная библиотека boost.
- Программная система CUDA, предназначенная для организации параллельных вычислений на процессорах NVIDIA.
- Программный пакет MPICH2 стандарта MPI 2.1, предназначенный для обмена сообщениями между процессорами x86–64.
- Библиотека технической графики QWT.
- Командный процессор Bash.

Разработанное тестовое программное обеспечение представляет собой программный комплекс, состоящий из двух программных подсистем:

- Проверочные тесты. Позволяют определить работоспособность аппаратуры МВП как в обычных условиях, так и в тяжелых режимах функционирования при высокой температуре и наличии механических воздействий.
- Функциональные тесты. Предназначены для проверки комплектации и производительности аппаратуры МВП на соответствие техническим условиям.

Подсистема проверочных тестов должна включать тесты различных типов устройств, входящих в состав МВП: модуля коммутации, модулей центральных процессоров (МЦП), оперативной памяти МЦП, накопителей на дисках МЦП, модулей графических процессоров (МГП), модуля интерфейса оптического (МИО), модуля дискового накопителя (МНД), модуля «Эльбрус», модуля «Байкал» и модуля реконфигурируемого процессора (МРП).

Для управления выполнением и обработки результатов тестовых проверок устройств МВП необходимо было разработать управляющую программу – тестовый диспетчер.

В процессе реализации подсистемы проверочных тестов использовался шаблон архитектуры «Уровни» [2]. Разрабатываемые модули были

разделены на три уровня, причем каждый уровень может взаимодействовать только с соседними уровнями.

Благодаря такому разделению разработка основного функционала каждого уровня велась независимо от других, что существенно повышает модифицируемость системы.

Выбранный шаблон архитектуры включает следующие операции:

- Подсистемы разбиваются на уровни: пользовательский, тестовый, инструментальный.
- Разрабатывается интерфейс межуровневого взаимодействия (взаимодействие происходит только между соседними уровнями: пользовательский взаимодействует с тестовым, тестовый взаимодействует с пользовательским и инструментальным).
- Разделение подсистемы на уровни показано на рис. 2.

В качестве используемого шаблона проектирования был выбран шаблон «Команда» [2], разделяющий систему на следующие объекты:

- Инициирующий операцию – тестовый диспетчер.
- Исполняющие операцию – тесты аппаратных модулей.

Реализация шаблона для подсистемы тестирования:

- Диспетчер отправляет команды о проведении тестирования (пользователь задает свойства тестирования и выбирает необходимые тесты).
- В зависимости от настроек и выбора пользователя происходит выполнение тестовых программ, результат которых передается в диспетчер.

Шаблон хорошо подходит для систем, управляемых событиями: при появлении события (запроса) необходимо выполнить определенную последовательность действий (тестирование аппаратных модулей).

Тестовый диспетчер

Тестовый диспетчер (ТД) предназначен для управления запуском и прохождением проверочных тестов.

Проверочные тесты выполняют выбранные пользователем тесты циклически, пока не выполнится указанное количество циклов или не произойдет инициируемое пользователем прерывание.

Входными данными для работы диспетчера являются файлы конфигурации, задающие модульный состав проверяемой МВП и режимы функционирования проверочных тестов.

Файлы конфигурации содержат следующие параметры:

- Состав конфигурации МВП. Содержит маску на типы тестируемых модулей, входящих в МВП.
- Путь к утилите mpirun, выполняющей запуск заданного числа параллельных процессов.
- Количество циклов выполнения выбранной тестовой последовательности (в случае задания неограниченного числа циклов предусмотрена возможность прерывания тестовой последовательности пользователем).

Тестовый диспетчер предоставляет возможность запуска тестов в многопоточном режиме (распараллеливание задач). В таком режиме тесты выполняются одновременно на всех процессорах одного типа. Вывод результатов выполнения теста каждого процессора выполняется независимо друг от друга.

Для реализации многопоточного режима в подсистеме используется утилита распараллеливания

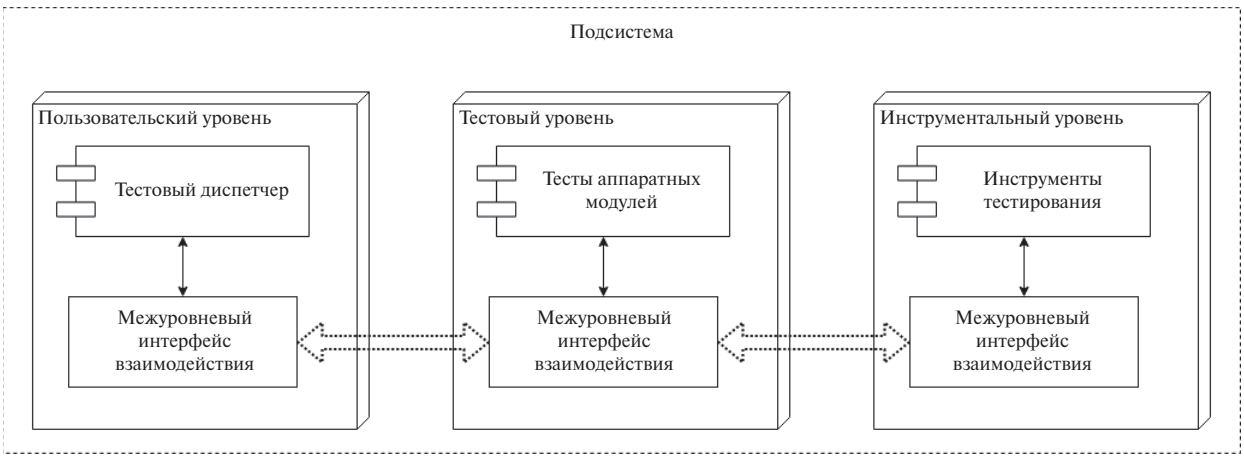

Рисунок 2. Разделение подсистемы на уровни

процессов по МЦП mpirun. Тестовый диспетчер содержит в себе функцию вызова утилиты mpirun с заданным числом параллельных процессов.

Пользовательский интерфейс (ПИ) тестового диспетчера содержит элементы управления и вывода информации о результатах тестирования.

Элементы управления и вывода информации тестового диспетчера были разделены по своему назначению на отдельные группы. Группы и входящие в них элементы, а также их назначение представлены в таблице.

Вид пользовательского интерфейса тестового диспетчера подсистемы проверочных тестов представлен на рис. 3.

Вид ПИ ТД определяется конфигурацией проверяемой системы и режимом выполнения тестов. На рис. 4 приведен вид ПИ ТД в начале выполнения тестов (после нажатия кнопки «Выполнить»).

Вид пользовательского интерфейса ТД после прерывания пользователем выполнения тестов (после нажатия кнопки «Прервать») представлен на рис. 5.

При каждом запуске тестов создается лог-файл выполнения тестов с именем, которое формируется с учетом времени и даты запуска, что позволяет вести архив тестирования системы.

В лог-файл выполнения тестов из поля вывода сообщений тестового диспетчера записывается информация, содержащая сообщение о номере и времени запуска цикла тестирования, сообщение об исправном прохождении каждого теста или о возникшей при его выполнении ошибке, а также сообщение о времени завершения каждого цикла тестирования.

В случае возникновения неисправности в файл ошибок заносится подробная информация о зафиксированной во время выполнения теста ошибке: номер цикла, название теста, время начала теста, данные о зафиксированной ошибке.

Тесты модулей МВП

Для организации взаимодействия между процессорными модулями в подсистеме проверочных тестов был выбран наиболее распространенный в параллельном программировании интерфейс обмена данными – MPI [3].

В MPI используются как точечные взаимодействия, при которых один процесс передает сообщение, а второй его принимает, так

Таблица. Элементы управления тестового диспетчера

Группа	Элементы	Назначение
Наборы тестов	Индикаторы выбора тестов	Пользователь может выбрать тесты, которые необходимо выполнить
Состояние	Цветовые и текстовые индикаторы выполнения	Предоставляют информацию о выполнении тестирования. Состояния индикаторов (цвет – текст): • серый – тест неактивен (недоступен для выполнения); • желтый – тест готов к выполнению; • голубой – тест выполняется; • зеленый – тест выполнен; • оранжевый – тест прерван
Управление тестированием	Кнопка «Выполнить» Кнопка «Прервать»	Начинает процесс выполнения выбранных тестов Останавливает выполнение тестов
Параметры запуска	Поле «Путь к mpirun» Поле «Дополнительные параметры mpirun» Кнопка «Обзор» Счетчик «Число процессов»	Предназначено для ввода пути к программе распараллеливания Предназначено для ввода дополнительных параметров утилиты mpirun Позволяет выбрать путь к утилите mpirun Указывает, на сколько ветвей необходимо распараллелить программу
Ход выполнения	Индикатор прогресса (выполнения)	Отображает процесс выполнения тестов в процентах
Вывод сообщений	Поле вывода сообщений Кнопка «Очистить вывод»	Отображается оперативная информация о выполнении выбранных тестов Очищает поле вывода

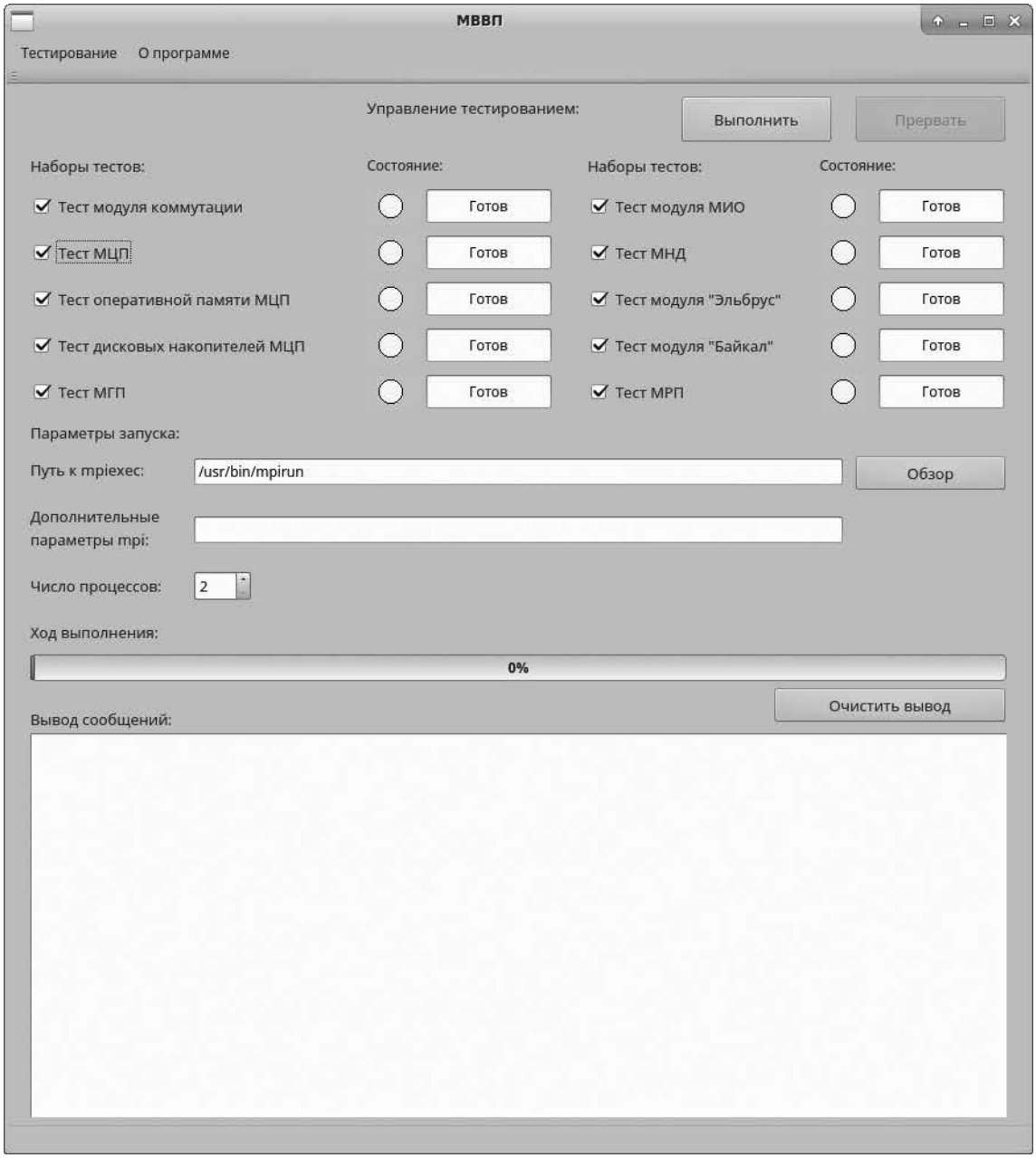

Рисунок 3. Вид пользовательского интерфейса тестового диспетчера

и коллективные взаимодействия типа «один-ко-всем», «все-к-одному» и «все-со-всеми».

В основе построения гетерогенной МВП лежит решение, позволяющее на одной шине объединять модули, построенные на основе разных архитектур. Связь между модулями осуществляется на основе открытого стандарта PCI Express. Обменом информацией между модулями по шине PCI Express управляет модуль коммутации KIC551.

Проверочный тест модуля коммутации состоит в проверке выполнения передачи и приема данных (шахматного кода 55 и AA) для всех пар процессоров МЦП. Перечень тестируемых операций включает одиночные обмены, параллельные одиночные обмены и коллективные операции интерфейса MPI.

Проверочный тест МЦП состоит в вычислении и сравнении с эталоном скалярного произведения векторов большой размерности *N* (в этом тесте число *N* было выбрано равным 330×1024).

Использование интерфейса MPI позволяет распараллелить тестовую задачу на все процессоры x86 и, таким образом, использовать линейку МЦП как единый вычислительный ресурс.

Модули графических процессоров VIM556 МВП реализованы на базе высокопроизводительной графической карты NVIDIA.

В современных процессорах NVIDIA используется архитектура CUDA, которая включает в себя набор команд над числами с плавающей точкой, позволяющий не только выполнять традиционную

Рисунок 4. Вид пользовательского интерфейса тестового диспетчера в начале выполнения тестов

обработку графических изображений, но также проводить математические вычисления общего назначения. В связи с этим графическим процессорам разрешен произвольный доступ к памяти для чтения и записи, а также доступ к программно-управляемому кэшу.

Для того чтобы задействовать все средства архитектуры графических процессоров, фирмой NVIDIA был предложен язык CUDA C [4], который и был использован для разработки теста МГП.

Тест модулей МГП состоит в вычислении и сравнении с эталоном скалярного произведения векторов большой размерности.

В тестах как МЦП, так и МГП все операции выполняются над числами с плавающей точкой.

Результаты прохождения тестов сравниваются с эталоном, вычисляемым по известной формуле, согласно которой величина скалярного произведения векторов со значениями координат i и $2i$, где $i = 0, 1, \ldots N-1$, может быть вычислена как удвоенная сумма квадратов целых чисел от 0 до $N-1$.

Каждый МЦП содержит блок оперативной памяти. В проверочном тесте оперативной памяти выполняются побайтовая роспись каждого блока памяти «бегающим нулем» и «бегающей единицей», побайтовое чтение информации из памяти и сравнение прочитанной информации с эталоном. Использование интерфейса MPI позволяет проводить одновременную проверку всех блоков памяти МЦП.

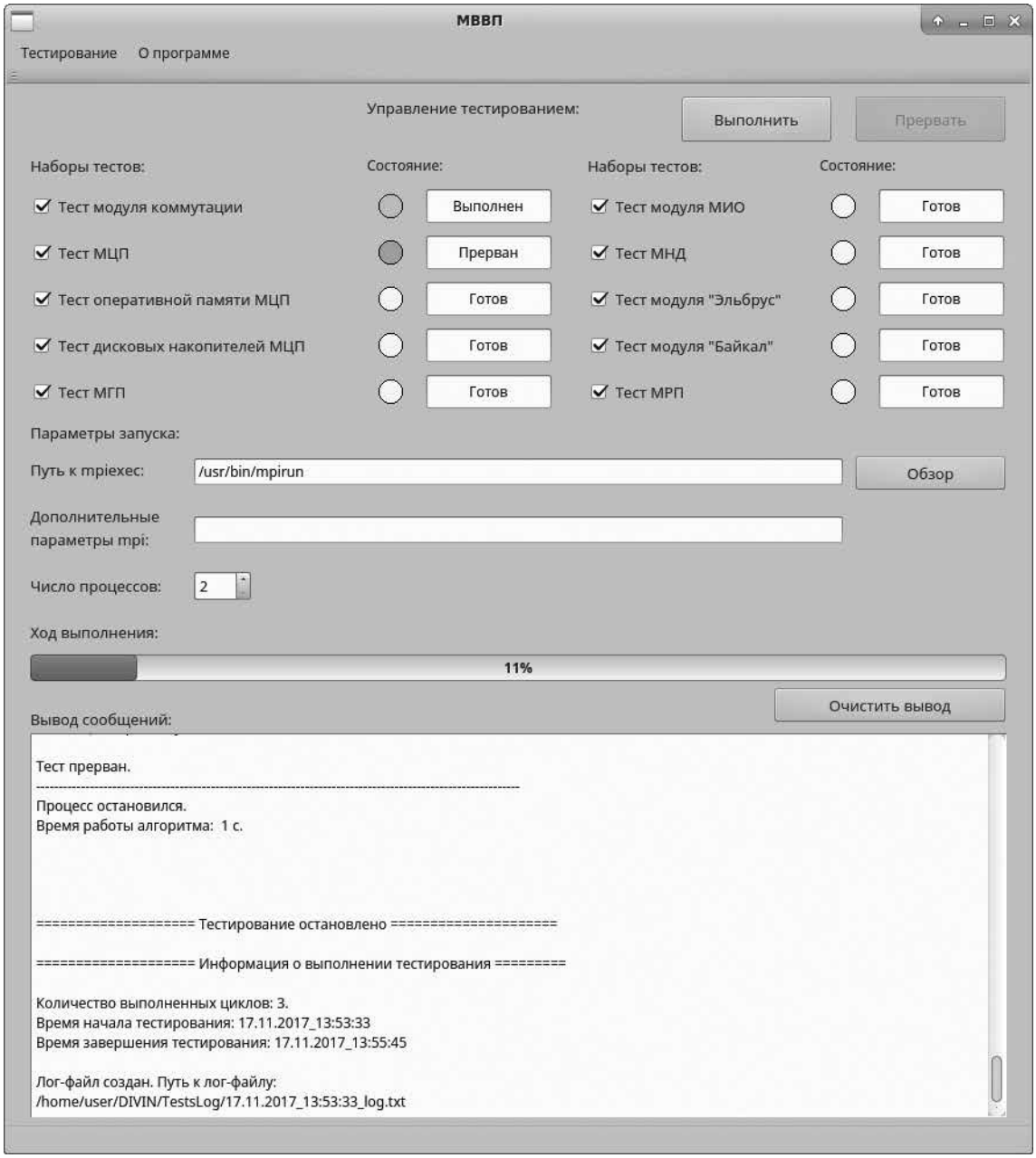

Рисунок 5. Вид интерфейса тестового диспетчера после остановки выполнения тестов пользователем

Файловая система МВП может включать в себя накопители на внешних дисках МЦП и независимые модули накопителей на внешних дисках. Тест накопителей на внешних дисках состоит в заполнении ячеек эталонного массива их адресами, создании на проверяемом диске файла заданного размера, записи в созданный файл эталонной информации, чтении информации из файла и сравнении считанной информации с эталоном.

Использование интерфейса MPI позволяет организовать параллельную проверку накопителей на дисках МЦП.

В тестах интерфейса модулей «Эльбрус» и «Байкал» используется кросс-платформенная клиент-серверная программа iperf3, которая позволяет генерировать трафик различного типа. В данных тестах используется двунаправленное тестирование со стороны клиента. При этом в качестве клиента выбирается МЦП, а в качестве сервера – модуль «Эльбрус» («Байкал»). Запуск утилиты iperf3 с необходимыми параметрами осуществляется из командного файла, выполняемого интерпретатором bash.

Для проверки работоспособности модулей «Эльбрус» и «Байкал» к проверяемому модулю подключаются клавиатура и монитор, после чего под управлением ОС выбранного модуля «Эльбрус» («Байкал») выполняется автономная система тестовых проверок соответствующего модуля.

Для выполнения теста интерфейсного оптического модуля (МИО) передающий и принимающий

модули должны быть соединены оптическим кабелем, при этом один из МИО должен быть установлен в режим «master», а другой – в режим «slave».

В тесте МИО утилита iperf3 используется для выполнения двунаправленного тестирования со стороны клиента. В качестве клиента был выбран передающий МИО, а в качестве сервера – принимающий МИО, соединенный оптическим кабелем с передающим модулем.

В МВП предусмотрена возможность применения модулей реконфигурируемых процессоров (МРП), созданных на базе ПЛИС. В этом случае на универсальные процессоры возлагается больше функций реализации человеко-машинного интерфейса и системы управления, а на ПЛИС и графические процессоры возлагаются функции расчета и обработки больших данных по определенным алгоритмам – например, кодирования, декодирования, сжатия, фильтрации, ускорения индексирования и поиска по массивам данных [5].

В тесте модуля МРП ПЛИС (FPGA) выполняется первичная обработка исходной матрицы радиолокационных сигналов, записанной в бинарном файле, и сравнение с эталоном информации, полученной в результате обработки на ПЛИС исходной матрицы.

Выводы

В работе предложена оригинальная методика проектирования программного тестового обеспечения, в соответствии с которой была выполнена идентификация объектов проектируемой системы и разработана модель взаимодействия объектов системы между собой. Выбранный подход позволяет существенно повысить наглядность и модифицируемость программного кода.

В статье приведены методы построения управляющей программы «тестовый диспетчер», предназначенной для запуска тестов в многопоточном режиме. Разработаны методы построения проверочных тестов вычислительных модулей, имеющих различную аппаратную архитектуру.

Приведенные результаты приемосдаточных испытаний многопроцессорной вычислительной платформы подтверждают высокую эффективность и надежность разработанной системы тестовых программ.

СПИСОК ЛИТЕРАТУРЫ

1. Реконфигурируемая вычислительная платформа с разнородной архитектурой / А.К. Барыбин, В.Н. Лобанов, М.И. Чельдиев, П.Б. Чучкалов // Вопросы радиоэлектроники. 2016. № 7. С. 70–77.
2. Бек К. Шаблоны реализации корпоративных приложений. М.: Вильямс, 2008. С. 23–33.
3. Антонов А.С. Параллельное программирование с использованием технологии MPI. М.: Изд-во МГУ, 2012. С. 4–11.
4. Сандерс Дж., Кэндрот Э. Технология программирования CUDA в примерах. Введение в программирование графических процессоров. М.: ДМК Пресс, 2015. С. 43–59.
5. Чудинов С.М., Сорокин А.П. Применение ПЛИС для организации параллельно-конвейерной обработки данных в гетерогенной вычислительной среде // Вопросы радиоэлектроники. 2017. № 9. С. 51–56.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Бененсон Михаил Залманович, к.т.н., главный научный сотрудник, АО «Научно-исследовательский институт вычислительных комплексов им. М.А. Карцева», 117437, Москва, ул. Профсоюзная, д.108, тел.: 8 (495) 330-81-48, 8 (916) 515-22-18, e-mail: mz_ben@mail.ru.

Дивин Антон Павлович, инженер, АО «Научно-исследовательский институт вычислительных комплексов им. М.А. Карцева», 117437, Москва, ул. Профсоюзная, д.108, тел.: 8 (495) 330-81-48, 8 (916) 175-97-65, e-mail: ant.divin@gmail.com.

For citation: Benenson M.Z., Divin A.P. Methods of developing test software of multiprocessor computer systems. Voprosy radioelektroniki, 2018, no. 5, pp. 95–103.
DOI 10.21778/2218-5453-2018-5-95-103

M.Z. Benenson, A.P. Divin

METHODS OF DEVELOPING TEST SOFTWARE OF MULTIPROCESSOR COMPUTER SYSTEMS

The article is devoted to various aspects of the creation of test software for multiprocessor systems (platforms). All the more urgent becomes the task of developing an efficient test checks to determine the operability and functional characteristics of heterogeneous computing systems. For test development was done using various tools: a cross-platform compiler QT 5.2, boost library, CUDA technology, the software package MPICH2, a library QWT, the command processor Bash. The design of the test software was performed using templates developed with the aim of building the desired software architecture. The article describes methods of constructing the control program Test Manager, which provides the ability to run tests in multi-threaded mode. Discuss methods of constructing test programs to test different modules of a heterogeneous multiprocessor computing platform. The results of the acceptance tests confirm the high efficiency of the proposed test methods in a multiprocessor computing system.

Keywords: tests, software, multiprocessor computer system, benchmark, parallel operations, characteristics of computing systems.

REFERENCES

1. Barybin A. K., Lobanov V. N., Cheldiev M. I., Chuchkalov P. V. A reconfiguration computing platform with diverse architecture. *Voprosy radioelektroniki,* 2016, no. 7, pp. 70–77 (In Russian).
2. Kent B. *Shablony realizatsii korporativnykh prilozhenii.* [Implementation Patterns]. Moscow, Vilyams Publ., 2008, pp. 23–33 (In Russian).
3. Antonov A. S. *Parallelnoe programmirovanie s ispolzovaniem tehnologii* [Parallel programming using MPI technology]. Moscow, Izd-vo MGU Publ., 2012, pp. 4–11 (In Russian).
4. Sanders J., Kandrot E. *Tehnologija programmirovanija CUDA v primerah. Vvedenie v programmirovanie graficheskih processorov* [CUDA by Example: An Introduction to General-Purpose GPU Programming]. Moscow, DMK Press Publ., 2015, pp. 43–59 (In Russian).
5. Chudinov S. M., Sorokin A. P. Application of FPGA for organization of parallel-conveyor data processing in the heterogeneous computing environment. *Voprosy radioelektroniki,* 2017, no. 9, pp. 51–56 (In Russian).

AUTHORS

Benenson Mikhail, PhD, chief researcher, M. A. Kartsev Scientific and Research Institute of Computing Systems, 108, Profsoyuznaya ulitsa, Moscow, 117437, Russian Federation, tel.: +7 (495) 330-81-48, +7 (916) 515-22-18, e-mail: mz_ben@mail.ru.
Divin Anton, engineer, M. A. Kartsev Scientific and Research Institute of Computing Systems, 108, Profsoyuznaya ulitsa, Moscow, 117437, Russian Federation, tel.: +7 (495) 330-81-48, +7 (916) 175-97-65, e-mail: ant.divin@gmail.com.