УДК 681.3.06

Г.А. ПОЛЯКОВ[1], В.В. СКЛЯР[2], Д.А. ТОЛСТОЛУЖСКИЙ[3], Е.Г. ТОЛСТОЛУЖСКАЯ[4], В.С. ХАРЧЕНКО[5]

[1] ***Академия наук прикладной радиоэлектроники, Россия***
[2] ***Государственный НТЦ ядерной и радиационной безопасности, Украина***
[3] ***Харьковский национальный университет им. В.Н. Каразина, Украина***
[4] ***Харьковский университет Воздушных Сил им. И. Кожедуба, Украина***
[5] ***Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина***

ПРОБЛЕМЫ МНОГОВЕРСИОННОГО ПРОЕКТИРОВАНИЯ ВЫСОКОНАДЕЖНЫХ ПАРАЛЛЕЛЬНЫХ ПРОГРАММНЫХ СРЕДСТВ ДЛЯ СИСТЕМ УПРАВЛЕНИЯ КРИТИЧЕСКИМИ ТЕХНОЛОГИЯМИ И ОБЪЕКТАМИ

Рассматриваются основные проблемы проектирования высоконадежных программных средств для систем управления (АСУ) критическими технологиями и системами (КТС), состав и семантика задач в рамках отдельных проблем, обсуждаются возможные подходы к проектированию высоконадежного программного обеспечения, основанные (в отличие от известных подходов) на концепциях многоверсионности программных средств, множественности методов параллельной обработки, автоматического проектирования параллельных программ и учета предъявляемых к системам управления временных требований, требований к достоверности и ограничений на сложность средств реализации.

многоверсионность, гарантоспособность, параллельная обработка, параллельная программа, верификация, тестирование, контроль, диагностика

Введение

Постановка проблемы проектирования высоконадежных программных средств. Характерным для современного научно-технического прогресса является получение все более глубоких знаний законов материального мира и их практическое использование в интересах развития общества. Особенностями научно-технического прогресса являются:

– создание новых технологий и систем для решения задач все более высокоответственных, критических областей применения (энергетические комплексы, аэрокосмические комплексы, системы телекоммуникации, робототехнические системы и т.п.) в крайне ограниченные сроки;

– все большая опасность для общества катастроф, связанных с некорректным функционированием таких технологий и систем, которые могут иметь региональный, континентальный или планетарный характер.

Это обусловливает исключительную научную и практическую актуальность проблемы обеспечения достоверности и устойчивости управления критическими технологиями и системами, в том числе – достоверности и устойчивости их аппаратного и программного обеспечения.

Решение проблемы создания высоконадежных и гарантоспособных аппаратных и программных средств систем управления возможно при реализации следующих концепций их построения и проектирования [1 – 4]:

– **избыточность** (программная, аппаратная, данных) – как основа и необходимое условие создания высоконадежных аппаратных и программных средств АСУ КТС;

– **многоверсионность в сочетании с мажоритарной логикой** – как основа повышения достоверности функционирования аппаратного и программного обеспечения АСУ с помощью сравнительного анализа результатов параллельной обработки данных;

– **применение различных методов параллельной обработки** – как основной путь удовлетворения жестких временных требований/ограничений (на основе одновременного выполнения операций) и требований к достоверности (на основе совмещения выполнения

вычислительных операций и операций контроля, диагностики и коррекции ошибок);

– **применение встроенных средств контроля, диагностики и коррекции** – как путь повышения достоверности функционирования параллельных аппаратных и параллельных программных средств АСУ;

– **реконфигурация (перестраиваемость)** – как средство достижения адаптивности и устойчивости аппаратных и программных средств к изменениям внешней среды и к причинам нарушения корректности функционирования АСУ;

– **автоматический характер проектирования** параллельных аппаратных и программных средств – как единственно возможный путь повышения качества проектирования, снижения сроков проектирования и решения проблемы проектирования при постоянно возрастающей сложности АСУ КТС и существенном росте требований к надежности/ достоверности и устойчивости функционирования.

Проблема надежности аппаратно-программного обеспечения не нова, но актуальность ее решения не только не уменьшилась, а наоборот, возросла. Это обусловлено следующими причинами:

– быстрым ростом потенциальной опасности критических технологий и объектов;

– значительным усложнением проблемы обеспечения надежности с расширением состава учитываемых факторов, увеличением сложности алгоритмов и ростом жесткости предъявляемых к АСУ КТС требований и ограничений;

– появлением новых, пока почти не исследованных, возможностей решения проблемы надежности/безопасности и одновременно удовлетворения жестких временных требований и ограничений на основе использования различных методов параллельной обработки информации;

– необходимостью разработки методов и средств автоматического проектирования параллельных аппаратных и программных средств, вызванной практической невозможностью решения рассматриваемой задачи вручную;

– отсутствием полного решения проблемы проектирования высоконадежного программного обеспечения даже для случая традиционных последовательных программ.

Анализ результатов разработки теории и прикладных вопросов проектирования параллельных программ [5 – 11] показывает, что вопросы обеспечения достоверности параллельного программного обеспечения исследованы пока недостаточно. Это обусловлено отсутствием учета ряда факторов, отличающих параллельные программы от традиционных последовательных программ и оказывающих существенное влияние на достоверность их функционирования. К таким факторам можно отнести следующие:

– увеличение количества операторов в параллельных программах задачи по сравнению с последовательной программой, необходимых для реализации различных методов параллельной обработки данных;

– зависимость изменения состава типов и количества таких «дополнительных» операторов от фактического состава применяемых методов параллелизма;

– возможность несоответствия схемы временного взаимодействия операторов программы схеме статических связей операторов по данным и по управлению и т.д.

Отметим, что применительно к критическим технологиям и объектам проблема высокой надежности функционирования переходит в проблему гарантоспособности – гарантированного выполнения системами управления необходимых функций в условиях воздействия факторов различного происхождения [12, 13].

Цель данной работы – анализ проблем и технологий проектирования высоконадежных параллельных программных средств с использованием принципа многоверсионности.

Проблемы многоверсионного проектирования высоконадежных параллельных программных средств

В отличие от общепринятой трактовки параллельных программ, определим параллельную программу как формальную конструкцию, содержащую спецификации следующих категорий

информации [14, 15]:

– множество объектов (данных), над которыми должны выполняться действия, и их спецификации;

– множество действий (операторов/операций), которые должны выполняться над данными для решения задачи;

– множество статических связей, задающих отношения упорядоченности операций по данным и по управлению;

– упорядоченность операций в динамике вычислительного процесса, определяемую моментами начала выполнения операторов и их длительностью;

– разделение множества операций на временные фрагменты, включающие некоторое множество операций и характеризующиеся одновременным началом их выполнения.

Отметим, что архитектура параллельных процессоров, мультипроцессорных ЭВМ и спецпроцессоров различных классов (рис. 1) оказывает существенное влияние на структуру и технологию проектирования параллельных программ вообще, и особенно – высоконадежных параллельных программ.

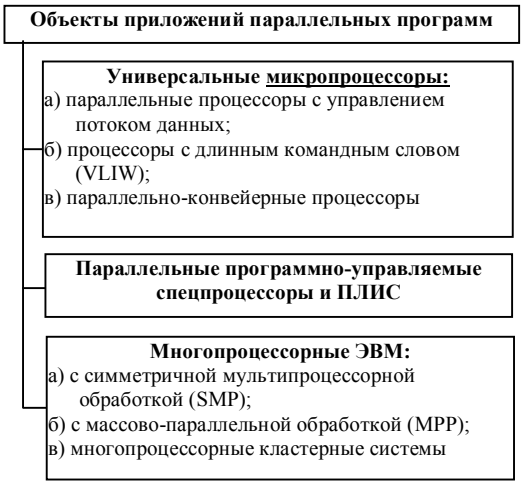

Рис. 1. Классы параллельных вычислительных средств с организацией вычислительного процесса на основе параллельных программ:
VLIW – Very Large Instruction Word;
SMP – Symmetric Multiprocessing;
MPP – Massively Parallel Processing

Это связано с необходимостью учета при проектировании количественного состава функциональных модулей и процессоров, длительностей выполнения вычислительных операций, типа межпроцессорных коммуникаций (полносвязная система, общая шина, кольцо, гиперкуб и т.д.), организации иерархической памяти, степени параллелизма доступа к памяти и времени доступа, временных затрат на межпроцессорный обмен данными и на синхронизацию и т.п.

Большое влияние оказывает на структуру и технологию проектирования параллельных программ также состав поддерживаемых при проектировании программных средств методов параллельной обработки данных (рис. 2) [1].

Основными факторами, порождающими многоверсионность параллельных программ, являются:

– возможность решения одной и той же задачи вычислительными средствами различных классов (**уровень вычислительных архитектур**);

– возможность решения одной и той же задачи на основе применения различных численных методов (**уровень численных методов)**;

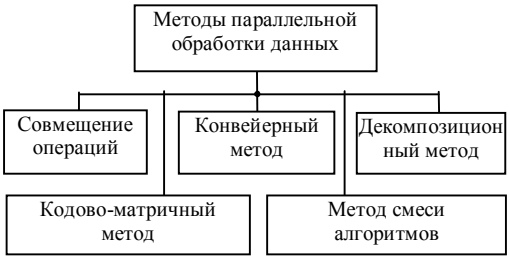

Рис. 2. Состав методов параллельной обработки данных, поддерживаемых при проектировании параллельных программ

– существование различных стилей и систем программирования: традиционный стиль, объектно-ориентированное проектирование и т.п. (**уровень технологий программирования);**

– множественность языков программирования и их избыточность (**уровень языков программирования)**;

– множественность методов параллельной обработки информации и возможность варьирования их составом (**уровень параллельной обработки)**;

– возможность изменения в широких пределах требований и ограничений, предъявляемых к аппаратным и программным средствам конкретными областями критических приложений (**уровень требований/ограничений)**.

Параллельные программы являются более общим (по сравнению с последовательными программами) объектом проектирования программного обеспечения, характерным для которого является расширение состава причин некорректного функционирования.

К причинам некорректности статического характера можно отнести:

– невыявленные ошибки в исходном тексте последовательной программы (в составе операторов, связях по данным, связях по управлению и т.д.);

– ошибки, обусловленные некорректностью учета архитектуры и временных характеристик параллельных процессоров, спецпроцессоров и многопроцессорных ЭВМ;

– ошибки некорректного выбора методов параллельной обработки данных, используемых при разработке временной модели параллельной программы, и перехода от последовательной программы к тексту параллельной программы (например, некорректное формирование фрагментов независимых операторов, выполняемых одновременно);

– ошибки типа «предшествование – наследование» операторов при статическом планировании модели выполнения параллельной программы;

– ошибки разделения параллельной программы на последовательные или параллельные процессы – «нити», выполняемые различными процессорами;

– ошибки распределения ресурса (при наличии ограничений на количество оборудования) между различными «нитями»;

– ошибки, связанные с некорректным вводом операторов обмена данными между параллельными процессами - нитями;

– ошибки, связанные с некорректным вводом операторов синхронизации «программных нитей» и т.д.

Примерами причин некорректности динамического (временного) характера могут являться:

– ошибки некорректного выбора методов параллельной обработки данных при разработке параллельной программы, приводящие к невыполнению временных требований/ограничений;

– ошибки вычисления значений данных;

– ошибки типа «предшествование – наследование» операторов при планировании модели (в частном случае – сетевого графика) параллельной программы в процессе функционирования;

– ошибки динамического формирования «нитей» параллельной программы (при планировании процесса в ходе выполнения программ);

– ошибки выполнения операторов обмена данными между параллельными процессами – нитями;

– ошибки выполнения операторов синхронизации «программных нитей».

Проектирование высоконадежных параллельных программных средств систем управления с учетом изложенных концепций и факторов обусловливает необходимость решения следующих проблем:

– **формирования многоверсионности** и ее применения для реализации принципа мажорирования при создании программных средств высокой достоверности для систем управления [16];

– **оптимизации «разделения алгоритмов»** (для реализации задач с помощью аппаратных или программно-управляемых средств);

– **проектирования параллельных моделей алгоритмов**, обеспечивающих требуемые характеристики выполнения алгоритмов (время запаздывания, тактовая частота обработки данных, производительность, сложность, надежность/достоверность) и ориентированных на различные архитектуры вычислительных средств;

– **проектирования параллельных программ**, ориентированных на различные архитектуры вычислительных средств (параллельных процессоров, спецпроцессоров и мультипроцессоров) и удовлетворяющих заданной системе требований/ ограничений;

– **верификации и тестирования** результатов проектирования параллельных программных средств;

– **контроля, реконфигурации и восстановления** в реальном времени корректного

функционирования параллельных программных средств;

– **оценки показателей эффективности** (временных, надежности и стоимости) высоконадежных параллельных программных средств;

– **оперативной визуализации** статических и динамических объектов проектирования и показателей эффективности параллельных программных средств;

– **разработки формализованных методов и создания инструментальных средств**, обеспечивающих автоматическое выполнение всех этапов проектирования высоконадежных параллельных программных средств АСУ КТС.

Технология многоверсионного проектирования высоконадежных параллельных программных средств

Рассмотрим возможную архитектуру технологии многоверсионного проектирования параллельного программного обеспечения, состав задач различных этапов и возможные подходы к их решению, ориентированные на возможно более полную автоматизацию проектирования высоконадежных параллельных программ для АСУ КТС (рис. 3).

Этап 1 (рис. 3, символ 1). Цель этапа – подготовка исходных данных, необходимых для всесторонней автоматизации проектирования параллельных программ с учетом жестких требований/ограничений к временным характеристикам, достоверности и допустимой сложности технической реализации. Среди особенностей подготовки исходных данных следует отметить:

– целесообразность формирования строгой формальной спецификации задачи, обеспечивающей возможность создания средств формализации и автоматического синтеза текста программы на языке программирования;

– предоставление заказчиком спецификации единиц измерения физических величин, необходимой для семантической верификации подлежащей разработке эталонной программы (а также программ – версий);

– предоставление информации по областям определения исходных данных и результатов задачи, разработка заказчиком и предоставление теста;

Рис. 3. Этапы многоверсионного проектирования параллельных программных средств АСУ КТС

– задание (в количественном измерении) требований заказчика (время выполнения программы, необходимая тактовая частота, требуемое значение достоверности, возможные допуски, ограничение сложности);

– тип вычислительных средств, на которых должно функционировать параллельное программное обеспечение (универсальные микропроцессоры, спец-процессоры, мультипроцессоры класса «симметричный мультипроцессор» – SMP, «массово-

параллельный мультипроцессор» – MPP, «кластерная архитектура» – Cluster), их конфигурация (количество модулей или процессоров, архитектура межмодульных /межпроцессорных связей и т.д.) и технические характеристики (времена выполнения операций/тактовая частота, архитектура и времена обращения к памяти и т.д.), характеристики сложности.

Этап 2 (рис. 3, символ 2). Целью этапа является синтез текста эталонной последовательной программы на языке программирования высокого уровня, исходя из формальной спецификации задачи (рис. 3, символ 1). В настоящее время решение этой задачи обеспечивает, как правило, программист. Требование высокой достоверности программных средств (и в частности – эталонной программы) ставит задачу разработки формальных (и автоматических) методов перехода от спецификации к тексту программы. Решение такой задачи следует искать, по-видимому, в направлении обеспечения высокой стандартизации и формализации описания задачи, с одной стороны, и разработке механизмов формальной текстовой интерпретации элементов спецификации, с другой стороны.

Требование высокой достоверности эталонной программы обусловливает необходимость проведения различных видов верификации, в том числе:

– верификации корректности структуры эталонной программы с помощью средств стандартных компиляторов с языков программирования или сочетания стандартного компилятора с компиляционным верификатором – в случае числового и графического представления эталонной программы [10];

– семантической верификации – проверки совпадения специфицированных заказчиком и фактически полученных единиц измерения результатов выполнения эталонной программы [11];

– декомпиляционной верификации, включающей восстановление исходной формальной спецификации, исходя из текста последовательной эталонной программы, и сравнение полученной и исходной спецификаций (рис. 3, символ 2).

Высокая достоверность программы – эталона определяется положительным исходом компиляционной, семантической, декомпиляционной верификаций, а также положительными результатами тестирования тестом заказчика и эталонным тестом. Средством дополнительного повышения достоверности эталонной программы является использование средств контроля выполнения операций, диагностики и автоматической коррекции влияния (не обнаруженных статическими средствами) ошибок в процессе функционирования программы.

Этап 3 (рис. 3, символ 3). На данном этапе формируются множества логически эквивалентных последовательных программ – версий для исходной задачи, соответствующих исходной спецификации и рассматриваемых как кандидаты на включение в мажоритарную логику. При ручном программировании – это программы, разработанные различными программистами для одного или различных численных методов решения задачи. При автоматическом синтезе это может быть множество программ – версий, автоматически формируемых (в текстовом или числовом формате) на основе одного или различных численных методов решения задачи с использованием различных стилей и систем программирования и избыточности языков программирования.

Этап 4 (рис. 3, символ 4). Задачей этапа является формальный (в перспективе – автоматический) синтез временных параллельных моделей для последовательных программ – версий. Возможные подходы и методы формализованного решения задачи проектирования параллельных моделей задач содержатся в ряде работ [1, 19, 21]. В качестве исходных данных выступают текст последовательной программы, требования и ограничения (время выполнения, тактовая частота, сложность), состав известных методов параллельной обработки данных, класс и характеристики архитектуры параллельных вычислительных средств. Основными этапами синтеза являются

преобразование текстов программ в числовой и графический формат, выбор методов параллельной обработки данных, адекватных алгоритмам задач и временным требованиям/ограничениям, синтез максимально параллельных временных моделей, параллельных моделей с заданными характеристиками (время, такт, сложность) или последовательных временных моделей (в числовом и графическом форматах), оценку их количественных характеристик (количество операций, количество маршрутов в модели, состав операций различных маршрутов, время выполнения различных маршрутов и т.д.), оценку показателей эффективности моделей.

Отметим, что при ручном параллельном программировании эта задача не решается. Это обусловлено, как минимум, двумя причинами:

– общепринятая трактовка параллельной программы не включает в качестве необходимого параметра фактор времени, определяющий разбиение (временную параметризацию) множества операторов программы на фрагменты операций, начинающих выполняться одновременно в **конкретные моменты времени** (формируются лишь фрагменты информационно независимых операторов, выполнение которых **может совмещаться** во времени);

– общепризнано, что качественные параллельные программы (даже без учета фактора времени) принципиально не могут быть созданы программистами при количестве процессоров, превышающем 3, 4.

Этап 5 (рис. 3, символ 5). Задачей этапа является проектирование для исходной задачи множества логически эквивалентных параллельных программ – версий, являющихся потенциальными претендентами на включение в схему мажоритарной логики.

При ручном параллельном программировании “разработку параллельной программы можно представить себе как последовательность из трех шагов [5]:

– **написание** традиционной последовательной программы, выполняющей необходимые вычисления;

– **мысленное проектирование** схемы параллелизма, т.е. распределение данных и вычислений по процессорам.

– **переписывание** программы в параллельном виде с использованием конкретной инструментальной системы параллельного программирования, имеющее целью введение необходимых операторов обмена данными между процессами – нитями и операторов временной синхронизации процессов – нитей”.

Отметим, что задача формирования **множества** логически эквивалентных параллельных программ до настоящего времени не ставилась. В то же время отмечается, что “для достижения предельно возможного ускорения приходится применять существенно более сложные методы исследования и преобразования программ, чем те, которые традиционно включаются в компиляторы. Сейчас они настолько сложны, что об их ручном применении не может быть и речи” [6].

Альтернативой ручному параллельному программированию [5 – 11] является формализованное (и автоматическое) проектирование параллельных программ, различные аспекты которого рассматриваются в работах [1, 14, 15, 17, 19 – 23]. Исходные данные:

Си-спецификация функциональной задачи, требования/ограничения (время выполнения, тактовая частота, сложность), класс и характеристики архитектуры вычислительных средств. Основными этапами проектирования при этом подходе являются:

– синтез числовой спецификации Си-программы исходной задачи;

– синтез числовых спецификаций параллельных временных моделей программ;

– синтез параллельных /последовательных программных нитей в рамках параллельных временных моделей программ с учетом минимизации временных затрат на обмен между нитями;

– организация обмена данными между программными нитями путем введения в состав

нитей операторов обмена данными (например, оператора ввода данных «**in»,** вывода данных «**out»,** двустороннего обмена **«in out» –** в случае использования многопроцессорных ЭВМ класса SMP) или операторов – директив обмена сообщениями (специфицированных в различных библиотеках систем параллельного программирования, например, MPI, Open MP, PVM – при использовании ЭВМ класса MPP с массовым параллелизмом);

– обеспечение синхронизации во времени совместного выполнения программных нитей путем введения в состав нитей операторов «ждать», «стоп», «пуск», использование семафоров и т.п. (также специфицированных в различных библиотеках параллельного программирования).

В целом следует отметить, что проблема формализации и автоматического проектирования параллельных программ (даже без учета требований высокой достоверности и средств ее достижения) не имеет в настоящее время полного решения и характеризуется диаметрально противоположными мнениями относительно перспектив ее успешного решения от мнения пессимистов, что "… методов автоматического превращения программ для традиционных компьютеров в программы для параллельных машин, дающие при исполнении скоростной выигрыш, не просто не существует, а не может существовать в принципе, по крайней мере, в ближайшем будущем"[5], прагматических взглядов – "Трудно рассчитывать на то, что можно создать полностью автоматическое средство анализа и преобразования (программ). Выход один – должна быть предусмотрена возможность исследования в любом режиме – от полностью автоматического преобразования "текст – текст" до интерактивного режима"[6], и оптимистических прогнозов, основанных на результатах работ [1, 14, 15, 17, 19 – 23].

Этап 6 (рис. 3, символы 6,7). Задачей этапа является верификация и тестирование параллельных программных версий, оценка количественных показателей качества и формирование на основе их анализа подмножества параллельных/последовательных программ – версий, непосредственно используемых для реализации принципа мажорирования обработки.

В зависимости от жесткости требований к программам – версиям могут применяться различные стратегии реализации этапа, различающиеся степенью детализации рассмотрения, сложностью логики выполнения, временными затратами и обеспечиваемой достоверностью результатов.

Минимальная стратегия включает тестирование программ – версий тестом заказчика, фиксацию для каждой версии множества некорректных откликов, формирование подмножеств, содержащих ошибки, общие для $n \geq 2$ (n-кратные ошибки) версий, включение в состав мажоритарного множества необходимого количества версий, имеющих минимальную мощность n-кратных ошибок, и оценку достоверности.

Номинальная стратегия предусматривает дополнительное тестирование версий эталонным тестом, диагностику ошибок, коррекцию обнаруженных ошибок, повторное тестирование эталонным тестом, формирование подмножеств откликов с n-кратными ошибками и формирование результата на основе версий, имеющих минимальную мощность n-кратных ошибок и обеспечивающих наибольшее значение достоверности.

Максимальная стратегия может обеспечивать приращение достоверности за счет использования (кроме рассмотренных выше) средств контроля численных результатов выполнения операций (например, контроль на четность, контроль по модулям 3 или 7, коды с обнаружением ошибок и т.п.), применения контроля корректности семантики (единиц измерения операндов и результатов) и средств автоматической коррекции ошибок, обнаруживаемых в динамике выполнения программ.

Как следует из предшествующего, высокая достоверность параллельных программ может достигаться применением как статических методов

обеспечения «правильности программы» (компиляторы языков программирования, компиляционная, семантическая и декомпиляционная верификация), так и динамическими способами (встроенные средства контроля, диагностики и коррекции), в сочетании с возможно более полным тестированием программ. Отметим, что эффективным подходом к решению рассматриваемой проблемы в статике являются способы формального доказательства правильности программ, использующие формальное описание программы и формальное определение семантики языка программирования. Однако для динамических (временных) объектов, какими являются параллельные программы, вопросы применимости формальных методов пока не исследовались.

Выводы

1. Проблема обеспечения высокой гарантоспособности аппаратного и программного обеспечения АСУ КТС имеет исключительно важное научное и практическое значение.

2. Параллельное программное обеспечение характеризуют следующие особенности: возможность удовлетворения жестких временных требований, расширенные возможности реализации принципа мажорирования, более высокая сложность и потенциально большее количество ошибок. В связи с этим целесообразность практического применения параллельных программ должна определяться на основе количественных оценок возможности повышения достоверности с учетом противоречивого влияния перечисленных факторов.

3. Основными задачами, требующими решения при создании гарантоспособного параллельного программного обеспечения АСУ КТС, являются разработка формальных спецификаций задач и механизмов формальной текстовой интерпретации спецификаций, способов формального доказательства правильности программ, развитие средств компиляционной, семантической, декомпиляционной верификации и повышения эффективности тестирования, применение встроенных средств контроля, диагностики и динамической коррекции ошибок, методов синтеза параллельных временных моделей задач и параллельных программ, способов отбора версий для реализации мажоритарной логики, средств визуализации объектов проектирования параллельных программ.

4. Статистика соотношения ошибок аппаратных и программных средств как 30% к 70% позволяет считать перспективным путь аппаратной реализации алгоритмов АСУ КТС с помощью спецпроцессоров с жесткой или перестраиваемой аппаратно управляемой архитектурой. Это обусловливает необходимость создания методов преобразования параллельных программ в логически эквивалентные структуры многофункциональных перестраиваемых спецпроцессоров, которые могут быть реализованы на основе ПЛИС.

5. Объективная высокая сложность проектирования гарантоспособного параллельного аппаратного и программного обеспечения АСУ КТС, жесткость требований к временным характеристикам и достоверности, ограниченные возможности применения субъективного «ручного» подхода делают крайне актуальной разработку методов и средств, обеспечивающих возможно более полную автоматизацию решения рассмотренных выше задач.

Литература

1. Поляков Г.А. Адаптивные самоорганизующиеся системы с мультипараллельной обработкой данных – стратегия развития цифровой вычислительной техники в XXI веке // Прикладная радиоэлектроника. – Х.: АН ПРЭ, 2002. – Т. 1, № 1. – С. 57-69.

2. Харченко В.С. Многоверсионные системы, технологии и проекты. – Х.: НАКУ “ХАИ”. – 2002. – 528 с.

3. Харченко В.С., Токарев В.И. Проектирование отказоустойчивых и живучих компьютерных систем управления на основе концепции “3М” // Вісник Технологічного університету Поділля. – 2003. – № 3. – С. 29-32.

4. Скляр В.В., Харченко В.С.,

Ястребенецкий М.А. Цифровые информационные и управляющие системы атомных электростанций и ракетно-космических комплексов: Сравнительный анализ, тенденции развития, обеспечение безопасности // Ядерная и радиационная безопасность. – 2004. – Т. 7. – № 2. – С. 35-41.

5. Лацис А. Как построить и использовать суперкомпьютер. – М.: Бестселлер. 2003. – 240 с.

6. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – С.-Пб.: БХВ-Петербург, 2002. – 608 с.

7. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. – С.-Пб.: БХВ-Петербург, 2002. – 400 с.

8. Программирование на параллельных вычислительных системах / Р. Бэбб, Дж. Мак-Гроу, Т. Акселрод и др. – М.: Мир, 1991. – 376 с.

9. Липаев В.В. Проектирование программных средств.– М.: Высш. шк.,1990. – 303 с.

10. Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. – 320 с.

11. Chandy, K. Mani. Parallel Program Design: a foundation. – Jayadev Misra, 1988. – 516 p.

12. Avizienis A., Laprie J.-C., Randell B. Fundamental Concepts of Dependability // Techn. Report: UCLACSD Report no. 010028, LAAS Report no. 01-145, Newcastle University Report no.CS-TR-739. – 2002. – 31 p.

13. Харченко В.С. Гарантоздатність комп'ютер-них систем: проблеми та результати // Авіаційно-космічна техніка і технологія. – 2005 – № 7 (23). – С. 352-357.

14. Поляков Г.А. Синтез времяпараметризованных параллельных программ – новый подход к параллельному программированию для специализированных МВК АСУ реального масштаба времени / НТК "Программное обеспечение АСУ". Ч. 2: Методология разработки АСУ. – Калинин, 1980.

15. Поляков Г.А. Глобально-параллельные времяпараметризованные программы – новый подход к синтезу структур и выполнению параллельных программ в АСУ реального времени / НТК "Программное обеспечение ВС и систем реального времени". – К., 1981.

16. Kharchenko V., Yastrebenetsky M., Sklyar V. Diversity Assessment of Nuclear Power Plants Instrumentation and Control Systems // Proceeding by 7th Int. Conf. on Probabilistic Safety Assessment and Management and European Safety and Reliability Conf. "PSAM 7 – ESREL ´04".– Berlin (Germany). – 14-18 June 2004. – Vol. 3. – P. 1351-1356.

17. Поляков Г.А., Толстолужский Д.А. Компиляционная методика верификации статических и динамических объектов автоматического проектирования мультипараллельных цифровых устройств // Прикладная радиоэлектроника. – Х.: АН ПРЭ, 2005. – Т. 4, № 2. – С.161-167.

18. Калбертсон, Роберт, Браун, Крис, Кобб, Гэри. Быстрое тестирование. – М.: Вильямс, 2002. – 384 с.

19. Толстолужская Е.Г. Методика формализованного синтеза мультипараллельных архитектурно-ориентированных моделей решения задач // Моделювання та інформаційні технології. – К.: НАНУ, ІПМЕ ім. Г.Є. Пухова – 2003. – Вип. 22. – С. 206-215.

20. Поляков Г.А., Онищенко В.В. Визуализация статико-динамических объектов автоматического проектирования мультипараллельных цифровых устройств // Системи обробки інформації. – Х.: ХВУ. – 2004. – Вып. 7 (35). – 240 с.

21. Поляков Г.А., Умрихин Ю.Д. Автоматизация проектирования сложных цифровых систем коммутации и управления. – М.: Радио и связь, 1988. – 304 с.

22. Поляков Г.А. Синтезатор графов Си-программ. Свид. о регистрации программы для ЭВМ № 2004610165 от 12.01.2004. РФ, Москва.

23. Поляков Г.А. Синтезатор параллельных моделей Си-программ. Свид. о регистрации программы для ЭВМ № 2004610800 от 31.03.2004. РФ, Москва.

Поступила в редакцию 14.03.2006

Рецензент: д-р техн. наук, проф. Б.М. Конорев, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.