СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 22 № 1 Год 2012

СОДЕРЖАНИЕ

ФИКСАЦИЯ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ В РЕКУРРЕНТНОМ ОПЕРАЦИОННОМ УСТРОЙСТВЕ*

***Р. А. Зеленов*[1], *А. А. Прокофьев*[2], *Ю. А. Степченков*[3], *В. Н. Волчек*[4]**

Аннотация: Обозначены проблемы, связанные с фиксацией исключительных ситуаций (ИС) в многоядерной потоковой вычислительной системе. Анализируются возможные варианты обнаружения ИС и формирования отладочной информации для ее передачи на обработку. Сформулированы принципы создания логики фиксации и обработки ИС, применимые для любого вида вычислительных систем.

Ключевые слова: исключительные ситуации; прерывания; потоковая архитектура; рекуррентность

1 Введение

В Институте проблем информатики Российской академии наук ведется разработка многоядерной потоковой рекуррентной вычислительной системы (МПРВС) с нетрадиционной архитектурой, являющейся развитием потокового подхода. В основе парадигмы вычислений лежит графодинамическое представление алгоритмов, рекуррентно свернутых до момента инициации исполнения и саморазворачивающихся в ходе выполнения задач. В вычислительном процессе задействован единый поток самодостаточных данных, который хранит в себе рекуррентно сжатый алгоритм решения конкретной задачи [1].

Особенность МПРВС заключается в том, что в ней осуществляется взаимодействие двух вычислительных систем с различной архитектурой: управляющего уровня, базирующегося на стандартных фон-неймановских принципах, и операционного уровня, реализованного в виде рекуррентного операционного устройства (РОУ). Следовательно, необходимо решить следующие проблемы взаимодействия этих уровней:

(1) обмен данными (решена путем введения буферной памяти (БП) [2]);

(2) обработка ИС — реакция аппаратуры на ошибки и отладочные события.

*Работа выполнена при частичной финансовой поддержке по Программе фундаментальных исследований ОНИТ РАН на 2011 г., проект 1.5.

[1]Институт проблем информатики Российской академии наук, graf.developer@gmail.com

[2]Институт проблем информатики Российской академии наук, a.a.prokofyev@mail.ru

[3]Институт проблем информатики Российской академии наук, YStepchenkov@ipiran.ru

[4]Институт проблем информатики Российской академии наук, v_volchek@inbox.ru

Обработка ИС является сложной и чрезвычайно актуальной задачей, которая включает в себя обнаружение ошибок, сбор отладочной информации, выполнение функций обработки возникших ситуаций, а также внедрение поддержки отладочных процедур в вычислительную систему. В данной статье представлен анализ проблем реализации функций фиксации и обработки ИС в РОУ, а также возможных вариантов их решения.

2 Особенности реализации многоядерной потоковой рекуррентной вычислительной системы

Как упоминалось выше, архитектура МПРВС является двухуровневой (рис. 1): в качестве управляющего выступает фон-неймановский процессор, в его задачи входит подготовка данных для операционного уровня и связь с периферийными устройствами; РОУ предназначено для эффективного исполнения параллельных алгоритмов.

В качестве программного обеспечения (ПО) для РОУ выступают капсулы [2]. Это программы, представляющие собой набор элементов самодостаточных данных (операндов), — инструкции по настройке определенных блоков, данные и команды для их обработки.

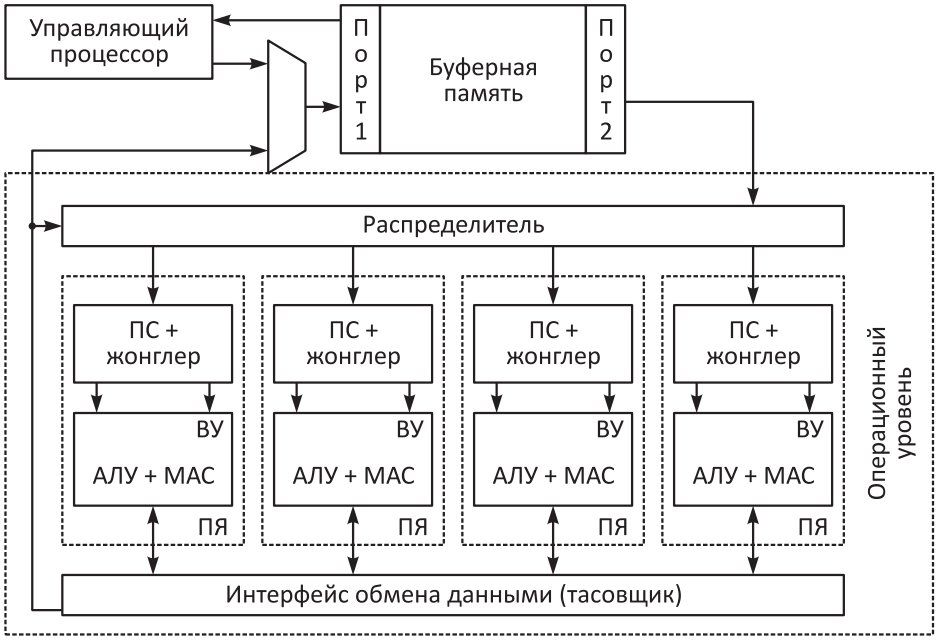

Рис. 1 Структура МПРВС

Взаимодействие двух уровней осуществляется через БП. С целью сокращения объема данных, поступающих со стороны управляющего уровня, перед инициализацией вычислительного процесса в БП записываются шаблоны капсул, содержащие только функциональные поля операндов, в процессе же вычислений идет заполнение лишь содержательных частей (непосредственно данных). Таким образом, в процессе многократного выполнения капсул вся информация о ходе вычислений остается неизменной, меняются только обрабатываемые данные [3].

В текущей реализации операционное устройство (ОУ) имеет в своем составе четыре процессорных ядра (ПЯ). Распределение входного потока самодостаточных данных по секциям осуществляется функциональным блоком (ФБ) — распределителем. Поступающие на вход ПЯ данные сначала находят свою пару в памяти совпадений (ПС), затем в зависимости от режима работы и кодов операций жонглер определяет, на какое плечо вычислительного устройства (ВУ) подать операнды для их дальнейшей обработки [4].

3 Фиксация исключительных ситуаций в рекуррентном операционном устройстве

Как и в любой вычислительной системе, в МПРВС возможно возникновение ошибок, связанных с различного рода ошибками и сбоями, которые невозможно предусмотреть на этапе программирования [5, 6]. Подобные события неизбежно приводят к нарушению нормального хода вычислительного процесса, поэтому встает задача их обнаружения, обработки, восстановления корректности вычислений. Представим структуру функции обработки ИС в виде дерева (рис. 2).

Двухуровневая организация МПРВС накладывает специфику на последовательность действий при работе с ошибками и служебными процедурами. Требуется разделение функций, представленных на рис. 2, между управляющим уровнем (УУ) и РОУ.

Внедрение логики фиксации исключительных ситуаций требует применения аналитического подхода с оценкой различных возможных вариантов исполнения, так как необходимо удовлетворять критериям:

- полноты покрытия ИС: должны обнаруживаться все возможные ИС, которые могут возникнуть в ходе работы РОУ, в том числе и вследствие сбоев в аппаратуре;
- полноты отладочной информации, передаваемой на УУ: в зависимости от места и времени возникновения ИС управляющий уровень будет производить обработку разными способами;
- минимизации аппаратных затрат на реализацию в устройстве;
- раннего обнаружения ИС: если ошибка, приводящая к ИС, может быть обнаружена в нескольких местах, то ее необходимо обнаруживать на как

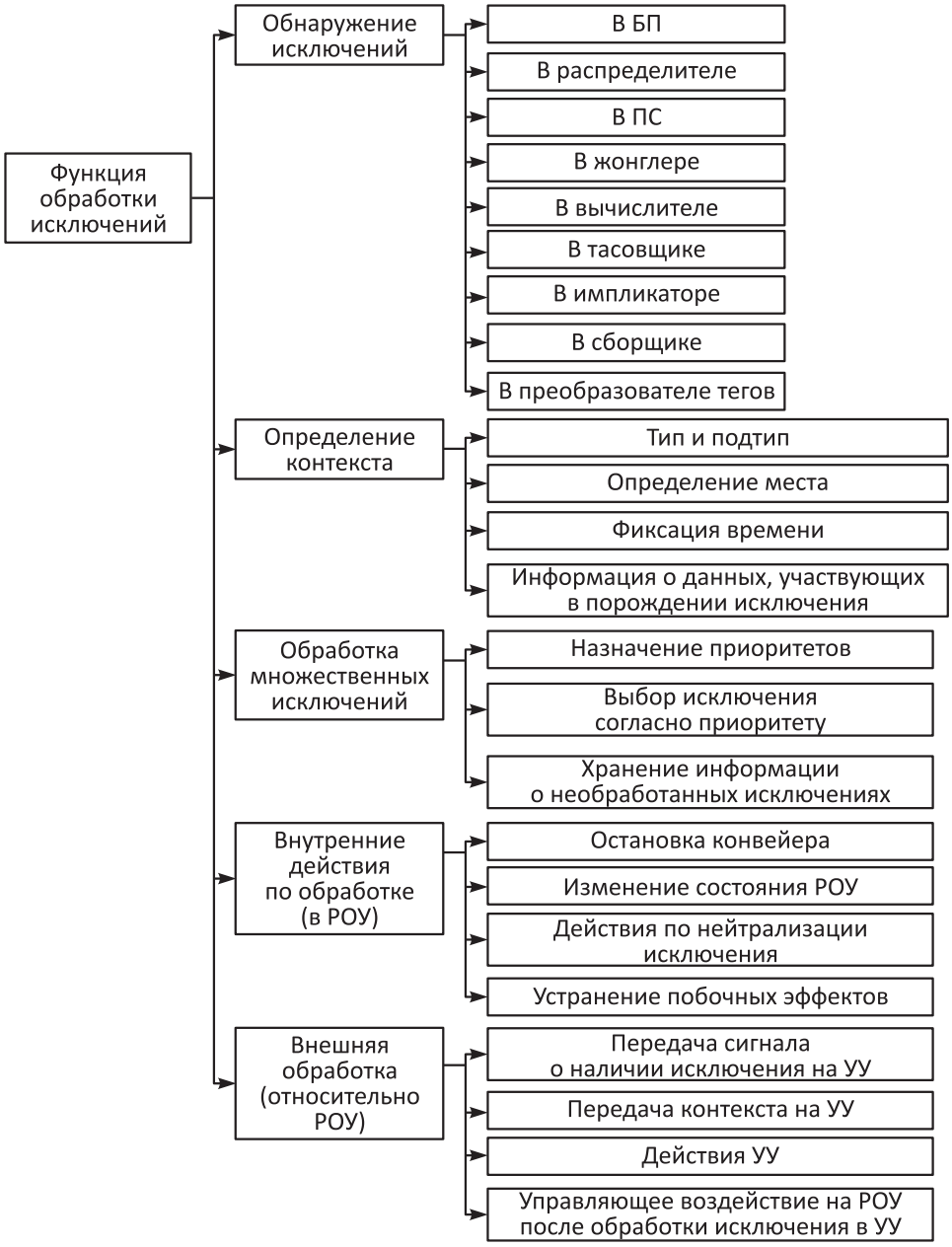

Рис. 2 Структура функции обработки ИС

можно более ранних этапах вычислительного процесса. Это позволит точнее определить причину ее возникновения.

Для реализации фиксации ИС требуется найти ответы на следующие вопросы:

1. Каков перечень ИС, их классификация и приоритеты?
2. Какую информацию о возникшей ИС требуется сохранить?
3. Что должно произойти с вычислительным процессом в РОУ при возникновении ИС?
4. Каким образом УУ получит информацию об ИС?
5. Каковы возможные варианты обработки ИС?

3.1 Перечень исключительных ситуаций

Для составления перечня ИС проанализируем работу каждого ФБ в отдельности с точки зрения спецификации и установим, какие ошибки возможны в ходе вычислительного процесса. Каждый из ФБ работает под управлением данных, поступающих на его вход. В этих данных могут передаваться коды настройки; следовательно, если код настройки не специфицирован, то поведение ФБ не определено и такая ситуация должна вызывать ИС. Подобного рода ИС можно отнести к типу «ошибка по допустимости». Однако будем последовательны и начнем анализ возможных ошибок в функциональных блоках с интерфейса взаимодействия РОУ с управляющим уровнем — буферной памяти.

Анализ структуры и функционирования БП выявил следующие возможные нарушения в корректности работы:

– попытка обращения по несуществующему адресу БП (тип «ошибка по допустимости», так как адрес превышает допустимое адресное пространство):
  - со стороны УУ — неверное значение на шине адреса;
  - через индексные регистры БП — выход за границы адресного пространства при считывании и записи данных;
– запись данных со стороны УУ в непредназначенное для них место: попытка записать функциональную часть операнда в область памяти, предназначенную для хранения содержательной части, и наоборот («ошибка по назначению»);
– отсутствие обязательного конфигурационного операнда по адресу, соответствующему началу капсулы («ошибка по отсутствию»).

Такие события могут происходить, если возник сбой в работе при записи операндов капсулы в БП управляющим уровнем.

После БП операнды попадают в распределитель. Для обеспечения раннего обнаружения логично именно на входе распределителя проверять на соответствие спецификации значения функциональных полей операндов («ошибка по допустимости»). Но данный анализ не может гарантировать полного отсутствия ошибок в функциональных полях, так как вследствие определенных причин значения полей будут специфицированными, но отличными от задуманных программистом, что приведет к ошибкам в ходе вычислительного процесса.

Кроме того, в распределителе могут возникать еще два типа нарушений корректности работы:

(1) по причине неправильного программирования или сбоя возможны ситуации, когда несколько операндов будут направлены на один и тот же путь передачи данных одновременно («ошибка по коллизии»);

(2) вследствие неверных настроек рассылки операндов они не могут быть корректно направлены в места своего назначения («ошибка по репликации»).

Других типов ИС в распределителе не предусмотрено, так как остальные варианты его работы специфицированы и могут корректно обрабатываться.

При определенной настройке жонглера нужно фиксировать ИС, а именно при обнаружении несовместимости в значениях отдельных функциональных полей сцепившихся операндов, когда:

– невозможно определить, нужно ли направлять результирующий операнд на шину обмена данными с распределителем и БП («ошибка по экспозиции»);

– коды операций несовместимы («ошибка по операции»);

– невозможно определить, нужна пересылка результирующего операнда вообще, а если да, то в свою или соседнюю секцию («ошибка по пересылке»).

В ПС может быть зафиксирована ИС, связанная с обращением к несуществующему адресу («ошибка по допустимости»).

Вычислитель выполняет операции над данными и, подобно большинству процессоров, отслеживает сопутствующие операциям события и характеристики получаемых результатов (нулевое значение результата, переносы, переполнения, вхождения в циклы и т. д.). Проведя их анализ, можно прийти к выводу, что только факт переполнения может негативно повлиять на ход вычислительного процесса, а это означает, что при его наступлении должна вырабатываться ИС — «ошибка по переполнению». Переполнения возможны в трех местах вычислителя: арифметико-логическом устройстве (АЛУ), регистре B, регистре C.

Результат вычислений должен через интерфейс обмена данными попадать на межсекционные шины; следовательно, здесь возможна «ошибка по коллизии».

В остальных узлах РОУ фиксации ИС не требуется, так как их функционирование не допускает появления ИС.

3.2 Классификация исключений

Получив перечень ИС, необходимо провести их классификацию, так как это позволит разработать стратегию по их фиксации и дальнейшей обработке. На рис. 3 приведена схема классификации ИС по трем основным признакам:

1. **Критичность** означает возможность или невозможность дальнейшего продолжения вычислительного процесса. Критичные ИС должны немедленно генерировать прерывания на управляющий уровень.

   В процессе отладки может потребоваться переход в пошаговое исполнение при возникновении определенных условий, в то время как в процессе вычислений достаточно только зафиксировать факт возникновения этих условий без генерации прерываний, поэтому некритичные ИС могут иметь возможность настройки своего поведения:

   – логирование информации об ИС;

   – генерация прерываний.

2. **Приоритет** позволит определить очередность и возможность обработки ИС в случае, когда в различных узлах устройства одновременно возникают ошибки. При этом необходимо сохранить информацию о наиболее важных из них. Приоритет может обозначаться числом (увеличение числа соответствует уменьшению приоритета, табл. 1), и при одновременном возникновении обработке будут подлежать ИС с большим приоритетом.

3. **По назначению** ИС делятся на два вида:

   (а) использующиеся постоянно для фиксации ошибок;

   (б) определяемые пользователем для целей отладки.

Перечень типов ИС и их классификация представлены в табл. 1.

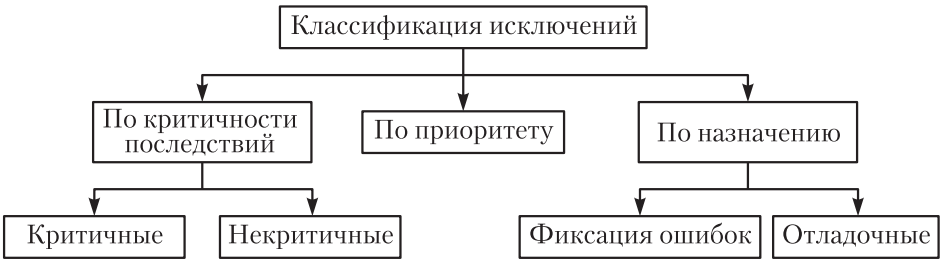

Рис. 3 Классификация ИС

Таблица 1 Перечень и классификация ИС

Тип ИС	Место возникновения	Классификация				Приоритет
		Критичность		Назначение		
		Критичная	Некритичная	Фиксация ошибок	Отладка	
Ошибка по допустимости	БП	+		+		1
	Распределитель	+		+		2
	ПС	+		+		3
Ошибка по отсутствию	БП	+		+		4
Ошибка по назначению	БП	+		+		5
Ошибка по коллизии	Распределитель	+		+		6
	Тасовщик	+		+		7
Ошибка по экспозиции	Жонглер	+		+		8
Ошибка по операции	Жонглер	+		+		9
Ошибка по пересылке	Жонглер	+		+		10
Ошибка по репликации	Распределитель		+	+	+	11
Ошибка по переполнению	Вычислитель: АЛУ		+	+	+	12
	Вычислитель: регистр B		+	+	+	13
	Вычислитель: регистр C		+	+	+	14
	БОИС		+	+	+	15
Пользовательская ИС (отладочная)	Определяется разработчиком		+	+	+	16

3.3 Отладочная информация об исключительной ситуации

Основной проблемой при фиксации ИС является сбор и сохранение информации о возникшем событии. Сохраняемая об ИС информация должна обеспечивать возможность определить:

(1) где возникла ИС (ФБ и при необходимости номер ПЯ);

(2) тип ИС;

(3) время возникновения ИС: вершина в графе вычислений.

Решение первых двух проблем достаточно очевидно: ввести регистры для сохранения кода ФБ, номера ПЯ и кода типа ИС. Сохранение же временно́й характеристики требует анализа структуры РОУ. Исходя из особенностей реализации БП, можно сделать вывод, что состояние индексных и ряда других управляющих регистров точно определяет, в каком месте графа алгоритма находится процесс вычисления. Но при этом остается неизвестен номер цикла

исполнения алгоритма, поэтому необходимо ввести соответствующий счетчик. Таким образом, путем объединения информации из регистров БП и счетчика циклов формируется временной параметр.

3.4 Функционирование блока обработки исключительных ситуаций в рекуррентном опреционном устройстве

Из вышеприведенного анализа следует, что в каждом из ФБ необходимо реализовать проверку условий возникновения ИС и разработать ФБ, отвечающий за сбор и хранение информации о возникших ИС (блок обработки исключительных ситуаций, БОИС) и обеспечивающий связь с УУ.

Из предложенной классификации ИС следует, что критичные ошибки должны приводить к немедленному завершению вычислений и передаче управления на УУ. При возникновении же некритичных ИС возможно продолжение работы с логированием информации о возникшем событии. Кроме того, любая ИС может использоваться в отладочных целях, например останавливать конвейер и переводить РОУ в режим пошагового исполнения. В связи с этим целесообразно вводить следующие режимы работы БОИС:

– при возникновении некритичных ИС:
  - игнорировать их возникновение;
  - логировать ИС без изменения хода вычислительного процесса;
  - активировать режим отладки;
– при возникновении критичных ИС:
  - остановить конвейер и генерировать прерывание;
  - активировать режим отладки.

Для минимизации промежутка времени между фиксацией ИС и переходом в режим отладки принято решение расширить функциональность БОИС способностью выполнять отладочные процедуры (например, пошаговую отладку, чтение/запись памяти и внутренних регистров), контролируемые УУ или пользователем.

Логирование информации требует введения памяти в БОИС, размер которой будет зависеть от преследуемых разработчиком целей. На ранних этапах необходимо накапливать как можно больше отладочной информации, тогда как в будущем, когда МПРВС пройдет приемо-сдаточные испытания, размер может быть уменьшен. Факт переполнения данной памяти может генерировать ИС «ошибка по переполнению».

Проблема возникает при поступлении на входы БОИС сведений о нескольких одновременно возникших ИС. Она связана с необходимостью либо увеличивать

аппаратные затраты на реализацию данного блока, либо увеличивать время его работы. В этом случае целесообразно введение двух вариантов работы БОИС:

(1) при наличии хотя бы одной критичной ИС обработке подвергается наиболее приоритетная;

(2) когда все ИС не являются критичными и режим работы предполагает логирование информации, БОИС остановит конвейер и в течение нескольких тактов запишет информацию обо всех ИС в свою память.

3.5 Взаимодействие с управляющим уровнем при возникновении исключительных ситуаций

Рекуррентное операционное устройство может обнаруживать появления ИС, но одного факта фиксации недостаточно — встает вопрос о передаче управления и информации об ИС на УУ. В первую очередь УУ должен узнать о том, что произошла ошибка, а затем, после некоторых процедур сохранения своего состояния, получить информацию об ИС и перейти к ее обработке.

Для получения УУ информации о факте ошибки существует несколько вариантов реализации:

(1) УУ с некоторой периодичностью опрашивает РОУ.

(2) интеграция с системой прерываний процессора, под управлением которого функционирует УУ.

Последний вариант является более предпочтительным, так как позволяет сократить расход вычислительных ресурсов и осуществить практически моментальную передачу информации о факте возникновения ИС на УУ.

После того как УУ оповещен об ошибке, он может либо сразу, либо после завершения текущей задачи считать подробную информацию об ИС из внутренних регистров РОУ, входящих в состав БОИС.

3.6 Обработка исключительных ситуаций на управляющем уровне

Управляющий уровень после получения информации об ИС должен корректно на нее отреагировать. Можно выделить два режима работы системы в целом: рабочий и режим отладки.

В рабочем режиме необходимо либо останавливать вычислительный процесс и выдавать пользователю сообщение об ошибке, либо, если это возможно, продолжать работу, фиксируя сведения о возникших ИС во внутренней памяти устройства. Выбор вариантов действий должен осуществляться через параметры настройки устройства.

В режиме отладки требуется выдавать сообщение об ошибке, останавливать вычислительный процесс и переходить к процедуре отладки.

По завершении обработки ИС необходимо восстановить вычислительный процесс, что невозможно без изменения внутреннего состояния РОУ. Перед тем как запустить на выполнение новый цикл вычислений, требуется провести работу по очистке конвейера: установить требующиеся значения флагов, управляющих регистров и памяти.

4 Среда разработки программного обеспечения для рекуррентного операционного устройства

При создании ПО для РОУ (капсул) разработчик должен учитывать множество нюансов, связанных с особенностями работы как всего РОУ в целом, так и его отдельных ФБ. Очевидно, что человеческий фактор (невнимательность, невозможность держать в памяти огромное количество информации из спецификации) может стать потенциальным источником ошибок в капсулах. Это означает, что для их разработки необходимо использовать средство, которое будет сводить к минимуму вероятность возникновения такого рода ошибок. Для решения этой проблемы была разработана система капсульного программирования и отладки (СКАТ) [2], которую можно отнести к такому классу ПО, как интегрированные среды разработки.

Главными задачами СКАТ являются: обеспечение разработчиков капсул инструментами для программирования и отладки; обнаружение возможных ошибок на как можно более раннем этапе разработки. К сожалению, во время создания капсулы невозможно определить все ошибки, которые могут возникнуть в процессе исполнения капсулы.

Процесс разработки ПО для РОУ итерационный, и его можно разделить на три повторяющихся этапа: программирование капсулы, исполнение ее на модели РОУ и отладка с помощью встроенных средств.

4.1 Использование системы капсульного программирования и отладки для работы с исключительными ситуациями

Основная задача СКАТ состоит в помощи разработчику при формировании капсулы, которая будет содержать минимальное количество ошибок. Ошибки синтаксиса при использовании СКАТ исключены, так как используется визуальный конструктор капсул [2].

Обнаружение большинства ИС, перечень которых приведен выше, возлагать на СКАТ не имеет смысла. Во-первых, для некоторых из них это невозможно по причине необходимости запуска вычислительно процесса. Во-вторых, любые из ИС могут возникнуть в результате ошибок или аппаратного сбоя во время перемещений капсулы между УУ и РОУ, поэтому их в любом случае нужно проверять в аппаратуре.

Определенный семантический анализ капсул в СКАТ внедрить все же необходимо по причине сложности программирования и невозможности учета всех нюансов вычислительной системы. Только с помощью СКАТ могут быть выявлены следующие ошибки:

– выход за рамки допустимой избыточности в процессе развертывания алгоритма;

– нарушение структуры капсулы:

  - отсутствие обязательных операндов;
  - взаимная противоречивость отдельных функциональных полей в операндах.

Вследствие использования в СКАТ VHDL-модели РОУ для процесса тестирования и отладки капсул, представляется возможным возложить на СКАТ некоторые функции УУ, а именно:

– получение от модели РОУ информации о возникновении ИС;

– выдачу пользователю информации, которая сможет помочь ему в устранении ошибки, вызвавшей ИС.

Информация об ИС может быть показана разработчику не только в виде состояния внутренних регистров блока фиксации ИС. На основе этих данных СКАТ может в некоторых случаях автоматически определить, какой операнд содержит в себе ошибку.

Таким образом, в скором времени планируется дополнить СКАТ следующими компонентами:

– семантическим анализатором капсул, который позволит обнаруживать некоторые логические ошибки в капсулах;

– компонентом анализа ИС, который будет по результатам анализа отчета работы модели РОУ сообщать пользователю информацию о возникших ИС и, где возможно, указывать на причину их возникновения.

Внедрение данного функционала позволит не только снизить количество возможных ошибок в капсулах, но также в некоторой степени автоматизировать процесс отладки, что является актуальной задачей для сред разработки ПО низкого уровня абстракции.

5 Заключение

В данной статье проведен анализ проблем, возникающих при разработке логики фиксации и обработки ИС в РОУ, и рассмотрены варианты их решения.

Принципы создания подобной функциональности, которые могут быть использованы практически для любой вычислительной системы, можно представить в виде последовательности шагов:

(1) составление перечня возможных ИС на основе анализа функционирования каждого из блоков вычислительной системы;

(2) классификация ИС, которая необходима для выработки правил их обработки;

(3) определение минимально необходимого набора информации, который позволит эффективно проводить процедуры отладки;

(4) определение возможных вариантов работы с ИС, и в случае если их фиксация и обработка возложены на разные аппаратные модули, то разработка протокола их взаимодействия;

(5) внедрение аппаратной поддержки обнаружения и обработки ИС в вычислительную систему.

В настоящее время ведется апробация вышеизложенных вариантов решения проблем фиксации и обработки ИС для РОУ.

Литература

1. *Степченков Ю. А., Петрухин В. С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов // Системы и средства информатики. Доп. вып. — М.: ИПИ РАН, 2008. С. 118–129.
2. *Зеленов Р. А., Степченков Ю. А., Волчек В. Н., Хилько Д. В., Шнейдер А. Ю., Прокофьев А. А.* Система капсульного программирования и отладки // Системы и средства информатики. — М.: ТОРУС ПРЕСС, 2010. Вып. 20. № 1.С. 24–30.
3. *Степченков Ю. А., Волчек В. Н., Петрухин В. С., Прокофьев А. А., Зеленов Р. А.* Механизмы обеспечения поддержки алгоритмов цифровой обработки речевых сигналов в рекуррентном обработчике сигналов // Системы и средства информатики. — М.: ТОРУС ПРЕСС, 2010. Вып. 20. № 1.С. 30–46.
4. *Степченков Ю. А., Волчек В. Н., Петрухин В. С., Прокофьев А. А., Зеленов Р. А.* Цифровой сигнальный процессор с нетрадиционной рекуррентной потоковой архитектурой // Проблемы разработки перспективных микро- и наноэлектронных систем — 2010: Сборник трудов. — М.: ИППМ РАН, 2010. 694 с.
5. *Селлерс Ф.* Методы обнаружения ошибок в работе ЭЦВМ / Пер. с англ. Ф. Селлерс. — М.: Мир, 1972. 310 с.
6. *Клингман Э.* Проектирование микропроцессорных систем / Пер. с англ. В. А. Балыбердина, В. А. Зинченко. — М.: Мир, 1980. 576 с.