УДК 004.052.32

Реконфигурация маршрутов в RapidIO системе при наличии неисправных соединений

Г.А. Лавринов

ФГУ ФНЦ НИИСИ РАН, lavrinov@cs.niisi.ras.ru

***Аннотация* — Многопроцессорные системы на базе интерфейса RapidIO играют значительную роль в промышленной и коммерческой сфере. Одним из аспектов, влияющих на их надежность, является исправное состояние всех соединений в системе, в частности, соединений между RapidIO устройствами. В статье рассматриваются способы инициализации системы при возникновении нарушений в соединениях RapidIO устройств, и их сравнение по основным функциональным характеристикам. Подробно рассмотрена статическая инициализация на основе базовых блоков, а также ее применение для тестирования RapidIO систем.**

***Ключевые слова* — многопроцессорная система, коммуникационный интерфейс, способы инициализации, конфигуратор базовых блоков, алгоритм Дейкстры.**

I. Введение

Коммуникационный интерфейс RapidIO (RIO) применяется уже более десяти лет [1]. Многопроцессорные системы на базе интерфейса RIO могут насчитывать до 65536 устройств и надежность логической связи между отдельными процессорами имеет для них критическое значение. Коммутационная среда RIO базируется на коммутаторах, в которые при инициализации среды записываются таблицы маршрутов, задающие пути прохождения по среде сообщений, направляемых конкретному процессору. Большие RIO-системы обычно избыточны по связям и число возможных маршрутов между двумя процессорами больше 1. Для систем с напряженным трафиком выбор маршрута из множества возможных может быть предметом некоторой задачи оптимизации. В данной работе предполагается, что характер решаемых RIO-системой задач допускает выбор из какого-то множества альтернативных маршрутов.

Спецификацией на интерфейс RIO предусмотрено два физических уровня:

- LP-LVDS (PRIO) – канал точка-точка представляет собой параллельный 8- и 16-разрядный дуплексный интерфейс с максимальной частотой тактового сигнала 1000 МГц [2];
- LP-Serial (SRIO) – канал точка-точка представляет собой последовательный дуплексный интерфейс, состоящий из 1, 2, 4, 8 или 16 подканалов (соответственно, режимы 1x, 2x, 4x, 8x или 16x) с максимальной скоростью передачи битового потока по каждому подканалу 10,3125 Гбод [3].

В случае неисправности одного из подканалов SRIO происходит переключение режима, например, из режима 4x в 1x, с последующим снижением скорости передачи данных через данное соединение. Поэтому в системах, где важно иметь определенную скорость передачи данных, диагностируя соединение можно определить переключение в режим с меньшим числом подканалов, которое будет указывать на неисправность соединения.

Имеет смысл различать неисправности в RIO соединениях по времени их появления / обнаружения:

1) После включения или сброса системы, когда на этапе инициализации коммуникационной среды RIO выясняется, что конфигурация системы отличается от заявленной в спецификации на систему.

2) В уже настроенной системе, когда в системе была произведена инициализация коммуникационной среды и, возможно, запущена задача.

Во втором случае, в отличие от первого, коммутатор RIO, в котором произошло нарушение связи в одном из портов, производит отправку пакета MAINTENANCE PORT-WRITE [4]. Данный тип пакета относится к служебным операциям, наряду с операциями MAINTENANCE READ и WRITE, и выполняет функцию оповещения удаленного устройства о произошедшем событии в коммутаторе. Получив информацию о неисправном соединении устройству-инициатору необходимо снова произвести инициализацию коммуникационной среды системы, причем с сохранением прежних номеров идентификаторов (ID) устройств. Далее рассматриваются многопроцессорные системы с одним инициатором, причем каждое оконечное устройство (Endpoint) системы имеет географический адрес (физическое положение в системе), доступный по RIO.

II. Способы инициализации коммуникационной среды RapidIO

Перед началом обмена между оконечными устройствами необходимо инициализировать коммуникационную среду. Инициализация состоит из двух этапов:

- Настроить таблицу маршрутизации для каждого коммутатора, которая обеспечивает определенный маршрут доставки пакета.
- Назначить ID оконечным устройствам.

Для инициализации RapidIO используются служебные операции MAINTENANCE READ и WRITE, которые могут обращаться в служебную память устройства [4]. Служебная память – это набор регистров: регистры управления и состояния устройства (Control and Status Registers - CSR), регистры “возможностей” устройства (Capability Registers - CAR) и др.

Для настройки коммутаторов используется стандартный и иерархический метод маршрутизации. Стандартный метод применяется для многопроцессорной системы с числом устройств меньше 512. Иерархический метод позволяет маршрутизировать полный диапазон номеров устройств от 0 до 65535 [5].

Существует несколько способов инициализации системы:

1. Динамический способ – это способ инициализации, включающий в себя механизм динамического обхода среды, который представлен в спецификации [6]. Данный способ получил широкое распространение благодаря применению его к любым конфигурациям системы. Компания MontaVista Software внедрила поддержку данного подхода в ядро Linux [7], начиная с версии 2.6.
2. Статическая инициализация с предварительным применением динамического обхода среды. Данный способ также представлен в спецификации и включает в себя динамический способ, приведенный в пункте 1, с последующей оптимизацией маршрутов [6]. Данный способ может быть представлен в графическом виде, где оптимизация маршрутов выполняется оператором.
3. Инициализация на основе ПЗУ конфигурации коммутатора [8]. Используется аппаратная поддержка инициализации таблицы маршрутизации, с помощью которой после включения питания (сброса) системы производится запись заранее прописанных в ПЗУ значений в регистры коммутатора RIO. Благодаря данному подходу сокращается время на программную инициализацию коммуникационной среды.
4. Статическая инициализация на основе базовых блоков представлена в статье [9], где под базовым блоком понимается программное описание RIO устройств и их соединений в рамках модуля. Схема формирования маршрутов представлена на рис. 1. Данный подход используется в тестовой системе «TestMS» [10].

Все приведенные способы применимы к многопроцессорным системам с исправными соединениями между RIO устройствами. Но не все способы могут реагировать на изменения в системе, связанные с нарушениями соединений. Способы, указанные в пунктах 1 и 4, обеспечивают данное требование к инициализации системы (далее под динамическим способом будет пониматься модифицированный динамический способ, позволяющий повторно инициализировать систему, а ID задавать в соответствии с географическим адресом устройства).

В случае динамической инициализации при изменении конфигурации системы устройство инициатор переопределяет маршруты к устройствам, не обращая внимания на неисправное соединение. В статической же инициализации на основе базовых блоков фиксируется неисправное соединение и, при необходимости, пользователь может быть проинформирован о неисправности, после чего производится перенастройка маршрутов. В данном способе для формирования маршрутов используется конфигуратор базовых блоков (см. рис. 1).

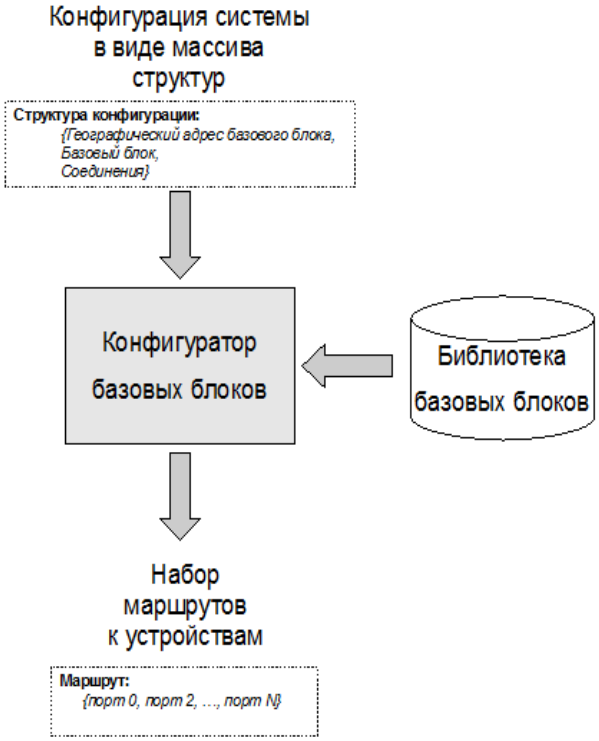

Рис. 1. Схема получения маршрутов к устройствам

Рис. 2. Схема конфигуратора базовых блоков

III. КОНФИГУРАТОР БАЗОВЫХ БЛОКОВ

Главным элементом в статической инициализации на основе базовых блоков является конфигуратор (см. рис. 2). Его задачей является формирование маршрутов к каждому устройству в системе. Под маршрутом понимается набор портов коммутаторов, через которые служебными операциями настраиваются таблицы маршрутизации к определенному устройству. На вход конфигуратора поступает массив структур {географический адрес базового блока, модель базового блока, соединения}. Структура заполняется в соответствии со спецификацией на многопроцессорную систему. В структуре указана модель базового блока, которая хранится в подготовленной библиотеке базовых блоков. Соединение представляет собой либо объединительную плату, на которой расположены базовые блоки, либо кабель.

До того, как будут сформированы маршруты, производится обработка входного массива структур в несколько этапов. На первом этапе конфигуратор определяет связи и нумерует все устройства, входящие в базовые блоки, которые указаны в каждой входной структуре. На данном этапе конфигуратор рассматривает конфигурацию абстрактной системы (система, конфигурация которой задана массивом структур). Для определения устройства абстрактной системы, с которого следует начать формирование маршрутов, конфигуратором считывается географический адрес устройства реальной системы, с которого и будет в дальнейшем произведена инициализация. Для формирования маршрутов на данном этапе используется механизм динамического обхода. Другими словами, проверяется по порядку каждое соединение базового блока для доступа к следующему блоку. В итоге результатом первого этапа является набор нумерованных устройств и связей.

В алгоритме динамического обхода длина маршрута, измеряемая количеством коммутаторов, не фигурирует. Поэтому следующим этапом является оптимизация путей в зависимости от решаемой на многопроцессорной системе задачи.

Маршруты, полученные для абстрактной системы, применяются к реальной системе. Производится проверка каждого маршрута в системе и, соответственно, проверка исправности соединений и коммутирующих RIO устройств. Также могут проверяться географические адреса устройств, определенные во входном массиве структур. Если все маршруты проверены успешно, то на выход конфигуратора базовых блоков передается весь полученный набор маршрутов. Если же один из маршрутов нарушен в системе, то производится удаление обнаруженного неисправного соединения из абстрактной системы, и затем снова происходит этап оптимизации маршрутов и т.д. (см. рис. 2).

Когда все маршруты сформированы и проверены, осуществляется полноценная инициализация системы. Алгоритм инициализации системы приведен на рис. 3.

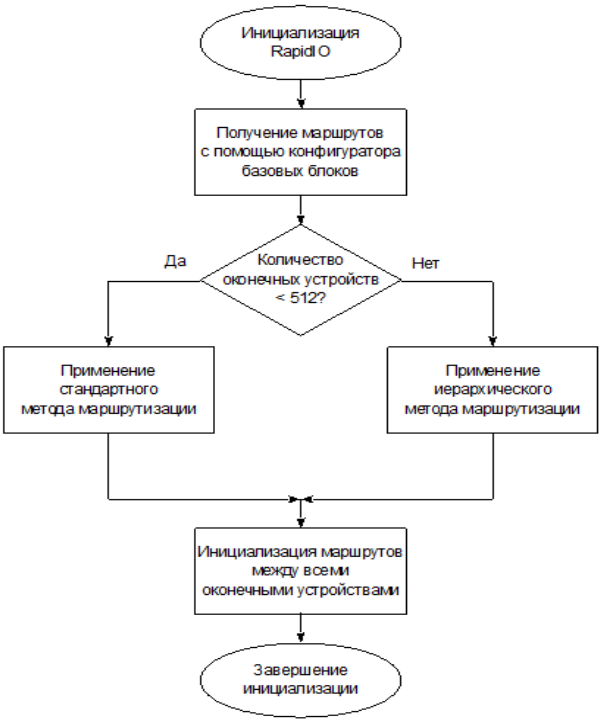

Рис. 3. Алгоритм статической инициализации на основе базовых блоков

IV. Сравнение способов инициализации

Основными функциональными характеристиками для сравнения способов инициализации являются: универсальность применения, наличие информации о физическом составе системы, возможность инициализации многопроцессорной системы в составе других систем. Сравнение способов инициализации, позволяющих реконфигурировать маршруты при появлении неисправных RIO соединений, приведено в табл. 1.

Главным достоинством динамической инициализации является универсальность применения. Другими словами, не требуется вносить изменений в ПО в случае изменения конфигурации системы. В статической инициализации на основе базовых блоков необходимо вносить изменение во входной массив структур, а также добавлять в библиотеку новые базовые блоки при необходимости. Но, благодаря входному массиву структур, известен физический состав системы. Данная информация необходима, например, для локализации ошибок в многопроцессорной системе. Кроме того, в вычислительном комплексе можно производить инициализацию только сконфигурированной подсистемы, не затрагивая работу других подсистем.

Таблица 1

Сравнение способов инициализации

Функциональная характеристика	Динамическая инициализация	Статическая инициализация на основе базовых блоков
Универсальность применения	Да	Нет
Информация о физическом составе системы	Нет	Да
Инициализация системы в составе других систем	Нет	Да

V. Применение способа инициализации на основе базовых блоков

В ФГУ ФНЦ НИИСИ РАН разработана тестовая система «TestMS», которая осуществляет начальную функциональную проверку процессоров серии 1890 [11], модулей и многопроцессорных систем на их основе [12], включая тестирование коммуникационной сети RIO. При первоначальном тестировании многопроцессорных систем возможно наличие неисправных соединений. Поэтому, чтобы при каждом обнаружении неисправности в соединениях не останавливать тестирование RIO устройств, в тестовой системе используется способ статической инициализации на основе базовых блоков для поиска новых маршрутов. Также, благодаря данному способу, заранее известен состав системы, и это позволяет локализовать нарушения в системе.

В тестовой системе «TestMS» управление тестированием осуществляется одним RIO устройством, который также производит инициализацию проверяемой системы. Поэтому в конфигураторе базовых блоков, применяемом в тестовой системе, для оптимизации маршрутов используется алгоритм Дейкстры. Алгоритм позволяет определить кратчайшие маршруты от устройства инициатора до всех остальных устройств системы. Многопроцессорная система на базе RIO, с точки зрения алгоритма, представляет собой взвешенный ориентированный граф G=(V, E) с весом ребра w[ij] равным 1, где V – множество вершин графа (устройства RIO), E – множество ребер графа (соединения устройств RIO), а ij – ребро.

VI. Заключение

Для реконфигурации маршрутов в многопроцессорных системах на базе RapidIO при нарушении соединений между RIO устройствами можно использовать рассмотренные в работе динамический способ инициализации и статический на основе базовых блоков. Последний эффективно использовать в тестовых задачах и в многопроцессорных системах, где требуется инициализировать часть системы. В случае использования статической инициализации в различных многопроцессорных системах, состоящих из однотипных вычислительных модулей, снижаются затраты на подготовку инициализации из-за наличия уже готовой библиотеки базовых блоков (достаточно сформировать массив структур).

Литература

[1] Fuller S. RapidIO. The Embedded Systems Interconnect. Wiley Publ. Inc., 2005, 380 p.

[2] RapidIO Interconnect Specification. Part 4: Physical Layer 8/16 LP-LVDS Specification. Revision 3.0. RapidIO Trade Association, 2013.

[3] RapidIO Interconnect Specification. Part 6: LP-Serial Physical Layer Specification. Revision 3.0. RapidIO Trade Association, 2013.

[4] RapidIO Interconnect Specification. Part 1: Input/Output Logical Specification. Revision 3.0. RapidIO Trade Association, 2013.

[5] Лавринов Г.А. Способ инициализации сети RapidIO с оконечными устройствами, имеющими разную разрядность идентификатора // Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты. М.: Изд-во НИИСИ РАН, 2014, том 4. №2. С. 39-41.

[6] RapidIO Interconnect Specification. Annex 1: Software/System Bring Up Specification. Revision 3.0. RapidIO Trade Association, 2013.

[7] Porter M. RapidIO for Linux // Proceedings of the Linux Symposium. 2005. V. 2. P. 35-47.

[8] URL: http://www.idt.com/document/man/tsi578-user-manual (дата обращения: 27.04.2016).

[9] Лавринов Г.А. Способы инициализации многопроцессорной системы // Программные продукты и системы. 2014. №4. С. 37-40.

[10] Свид. 2014612379 Российская Федерация. Свидетельство о государственной регистрации программы для ЭВМ. Тест-мониторная система «TestMS» / Лавринов Г.А.; заявитель и правообладатель

НИИСИ РАН (RU). – №2013662154; заявл. 25.12.13; опубл. 25.02.14, Реестр программ для ЭВМ. 1 с.
[11] Пат. 2359315 Российская Федерация, МПК7 G06F 9/30, G06F 7/483, G06F 7/57. Микропроцессор гибридный / Бобков С.Г., Аряшев С.И., Барских М.Е., Бычков К.С., Зубковский П.С.; заявитель и патентообладатель НИИСИ РАН (RU). – №2007116220/09; заявл. 28.04.07; опубл. 20.06.09, Бюл. №17. 23 с.
[12] Сердин О.В., Бобков С.Г., Кондратьева Н.В., Еремин А.А. Разработка высоконадежных многопроцессорных модулей на базе высокоскоростных каналов rapidio // Программные продукты и системы. 2013. №4. С. 49-55.

Route reconfiguration in RapidIO system in case of faulty connections

G.A. Lavrinov

Scientific Research Institute of System Analysis, lavrinov@cs.niisi.ras.ru

***Keywords* — Multiprocessor system, communication network, methods of initialization, configurator of basic blocks, Dijkstra's algorithm.**

ABSTRACT

Multiprocessor systems take a significant place in the industrial and commercial sphere. Processors interact with each other via communicative interface. The article considers multiprocessor systems based on RapidIO. One of aspects affecting reliability is operative condition of all connections in the system, in particular, the connections between the RapidIO devices. The article describes two methods of system initialization in case of malfunction of connections of RapidIO devices and their comparison by basic functional characteristics. The first method of initialization is based on dynamic routing system mechanism. The second method of initialization is static initialization with the use of basic blocks. The article considers in detail the second method of initialization and its application for RapidIO testing systems.

REFERENCES

[1] Fuller S. RapidIO. The Embedded Systems Interconnect. John Wiley & Sons Publ. Inc., 2005, 380 p.
[2] RapidIO Interconnect Specification. Part 4: Physical Layer 8/16 LP-LVDS Specification. Revision 3.0. RapidIO Trade Association, 2013.
[3] RapidIO Interconnect Specification. Part 6: LP-Serial Physical Layer Specification. Revision 3.0. RapidIO Trade Association, 2013.
[4] RapidIO Interconnect Specification. Part 1: Input/Output Logical Specification. Revision 3.0. RapidIO Trade Association, 2013.
[5] Lavrinov G.A. Sposob inicializacii seti RapidIO s okonechnymi ustrojstvami, imejushhimi raznuju razrjadnost' identifikatora (Method to initialize RapidIO network with endpoints having different number of bits of the identifier), Moscow, SRISA RAS, 2014, vol. 4, no. 2, pp. 39-41. (in Russian).
[6] RapidIO Interconnect Specification. Annex 1: Software/System Bring Up Specification. Revision 3.0. RapidIO Trade Association, 2013.
[7] Porter M. RapidIO for Linux // Proceedings of the Linux Symposium, vol. 2, 2005, pp.35-47.
[8] IDT. Available at: http://www.idt.com/document/man/tsi578-user-manual (accessed 27.04.2016).
[9] Lavrinov G.A. Sposoby inicializacii mnogoprocessornoj sistemy (Methods of multiprocessor system initialization), Tver, 2014, no. 4, pp.37-40. (in Russian).
[10] Lavrinov G.A. Svidetel'stvo o gosudarstvennoi registratsii programm dlya EVM "Test-monitornaja sistema TestMS", № 2014612379, 25.02.2014 (Certificate of state registration of computer programs "Test-monitor system TestMS", no. 2014612379, 25.02.2014). (in Russian).
[11] Bobkov S.G., Arjashev S.I., Barskih M.E., Bychkov K.S., Zubkovskij P.S. Mikroprocessor gibridnyj [Hybrid microprocessor]. Patent RF, no. 2359315. 2009. (in Russian).
[12] Serdin O.V., Bobkov S.G., Kondratyeva N.V., Eremin A.A. Razrabotka vysokonadezhnyh mnogoprocessornyh modulej na baze vysokoskorostnyh kanalov rapidio (Design of high reliability multiprocessor modules based on high-performance rapidio interconnect architecture), Tver, 2013, no. 4, pp.49-55. (in Russian).