УДК 004.7

СТРУКТУРНАЯ И АРХИТЕКТУРНАЯ ОРГАНИЗАЦИЯ IP-БЛОКОВ ДЛЯ СЕТИ-НА-КРИСТАЛЛЕ С ПОДДЕРЖКОЙ ДИНАМИЧЕСКОЙ РЕКОНФИГУРАЦИИ

© 2016 Е.А. Суворова

Санкт-Петербургский государственный университет аэрокосмического приборостроения

Статья поступила в редакцию 29.09.2016

Динамическое реконфигурирование Сети-на-кристалле (СенК) позволяет расширить их функциональность (и тем самым область применения) и может быть использовано для парирования ошибок. В этой статье рассматривается процесс динамического реконфигурирования СенК, изготавливаемых по технологии ASIC, требования и ограничения, которые необходимо учитывать при проектировании IP-блоков для таких систем. Предлагается подход к проектированию IP-блоков, позволяющий использовать их для проектирования СенК с различной степенью гранулярности реконфигурации, правила проектирования IP-блоков, позволяющие исключить возникновение ошибок (потеря данных, некорректная передача / обработка данных) в процессе реконфигурации.

Ключевые слова: *Системы-на-кристалле (СнК), Сети-на-кристалле (СенК), реконфигурируемые СенК*

Реконфигурирование Системы-на-кристалле (СнК) может быть использовано для:
- расширения области применения;
- адаптации характеристик системы в соответствии с набором решаемых задач;
- реализации большего набора функциональности;
- парирования ошибок.

В настоящее время существует много подходов к динамическому реконфигурированию СнК и СенК, однако большинство из них ориентировано на системы, реализованные на базе FPGA [1, 2]. В данной статье мы рассмотрим процесс динамического реконфигурирования СенК, реализованных на базе ASIC. СБИС, изготовленные по технологии ASIC, широко используются в отечественных аэрокосмических системах [1, 2]. Процесс динамического реконфигурирования ASIC очень существенно отличается от динамического реконфигурирования FPGA. FPGA изначально представляет собой реконфигурируемую структуру. На этапе разработки FPGA, а не конкретного проекта, определены:
- перечень конфигурируемых элементов;
- место хранения конфигурационной информации;
- связи между зоной хранения конфигурационной информации и соответствующими элементами FPGA;
- схема загрузки конфигурационной информации.

Суворова Елена Александровна, кандидат технических наук, заведующая лабораторией Систем-на-кристалле. E-mail: wildcat15@yandex.ru

FPGA разделена на зоны, являющиеся минимальными единицами реконфигурирования. Перед разработчиком стоит задача расположения элементов проекта, для которых возможно независимое конфигурирование, по этим зонам. В том случае, если используется технология ASIC, разработчик на этапе разработки проекта сам определяет перечень конфигурируемых элементов, способы передачи, хранения и загрузки конфигурационной информации. Поэтому подходы, разработанные для FPGA в общем случае не применимы для ASIC, т.к. при проектировании реконфигурируемых ASIC требуется решать совершенно другие задачи.

Большинство существующих подходов к реконфигурированию СенК ориентировано на полное прекращение передачи данных даже при частичном реконфигурировании [5, 6]. Однако для систем, ориентированных на решение задач с требованиями реального времени, к которым относится большинство систем аэрокосмического назначения, такое временное прекращение функционирования зачастую является не приемлемым. Мы будем рассматривать процесс динамического реконфигурирования ряда компонентов, входящих в состав СенК без останова ее функционирования.

Процесс реконфигурирования системы включает в себя:
- определение необходимости реконфигурирования;
- определение новой конфигурации системы;
- реконфигурирование системы.

Механизмы реконфигурирования системы включают в себя:
- передачу конфигурационной информации;

- замену конфигурационной информации в конфигурируемых компонентах.

В этой статье мы рассмотрим структурную и архитектурную организацию реконфигурируемых компонентов и процесс замены конфигурации.

Динамическое реконфигурирование СенК, требования и ограничения. Динамическое реконфигурирование СенК осуществляется под управлением менеджера конфигурации. Менеджер конфигурации может быть реализован программно (например, представлять собой один или несколько процессов, входящих в состав ядра операционной системы, управляющей функционированием СенК) или аппаратно (представлять собой один или несколько конечных автоматов). Возможны так же и аппаратно/программные варианты реализации. Менеджер конфигурации может быть реализован как централизованный, распределенный или частично распределенный компонент. Менеджер конфигурации может быть связан с конфигурируемыми компонентами с использованием выделенной системы (сети) связей или использовать основную сеть передачи данных СенК [7, 8]. Реконфигурация СенК может выполняться в ходе штатной смены режима функционирования. Например, часть задач в системе была выполнена до конца, и в систему поступают новые задачи. Реконфигурация так же может выполняться вследствие выявления ошибок в СенК.

Если реконфигурация выполняется в ходе штатной смены режима функционирования, то в процессе смены конфигурации потоки (пакетов) данных, соответствующие предыдущей конфигурации, не должны присутствовать в системе в процессе перехода на новую конфигурацию. Менеджер конфигурации располагает информацией о том, в какие моменты времени завершается конфигурирование модулей. Передача потоков данных, соответствующих новой конфигурации, может быть разрешена только после завершения процесса конфигурирования. При этом в СенК должны корректно передаваться остальные потоки данных.

Если реконфигурация выполняется вследствие выявления ошибок, то в ходе ее выполнения в сети могут передаваться данные, соответствующие предыдущей конфигурации. (Даже в том случае, если выполнение задачи, являющейся источником данных, путь передачи которых подлежит реконфигурации, может быть на время приостановлено менеджером конфигурации, какое-то количество данных от нее может передаваться через сеть в момент начала конфигурирования.) В большинстве случаев потеря этих данных является допустимой, однако они должны быть стерты (не должны навсегда остаться в сети, привести к возникновению блокировок).

В обоих случаях реконфигурация должна осуществляться таким образом, чтобы исключить нахождение СенК в противоречивом (inconsistent) состоянии, потерю и искажение данных передаваемых через СенК, а так же спонтанное появление новых данных в коммуникационной системе [9-11]. Процесс реконфигурации является распределенным во времени (не одномоментным). Если выполняется реконфигурация нескольких компонентов СенК, например, маршрутизаторов, то она может осуществляться последовательно (компонент за компонентом) или параллельно (конфигурирование нескольких компонентов осуществляется одновременно, или с частичным перекрытием во времени.) Если процесс конфигурирования выполняется последовательно, то система довольно ощутимое время находится в состоянии, когда в часть компонентов уже загружена новая конфигурация, а часть компонентов еще функционируют в соответствии с предыдущей конфигурацией. Если конфигурационная информация рассылается параллельно нескольким компонентам, то для каждого из них момент начала конфигурирования может определяться временем получения конфигурационной информации. Процесс конфигурирования отдельных компонентов может начинаться и заканчиваться в псевдослучайные моменты времени. В этом случае система также может находиться в состоянии, когда в часть компонентов уже загружена новая конфигурация, а часть компонентов еще функционируют в соответствии с предыдущей. По сравнению со схемой последовательной загрузки, время нахождения системы в этом состоянии будет существенно меньше, однако система все равно некоторое время может находиться в противоречивом состоянии.

Конфигурирование одного (отдельно взятого) компонента также может быть распределенным во времени. Это необходимо учитывать при проектировании реконфигурируемой СенК в целом, компонентов для нее и алгоритмов реконфигурации.

Структура реконфигурируемой СенК и IP-блоков для ее формирования. Обобщенная структура реконфигурируемого компонента представлена на рис. 1. Элемент памяти должен быть расположен как можно ближе к функциональному компоненту [12, 13]. Разрядность элемента памяти и разрядность линии связи между элементом памяти и функциональным компонентом равна разрядности вектора конфигурации. Если функциональный компонент имеет сложную внутреннюю структуру, разрядность вектора конфигурации велика, то элемент памяти

может быть на физическом уровне реализован как совокупность (множество) подэлементов, в каждом из которых хранится подвектор, соответствующий одной из частей функционального компонента. Разрядность линии связи, по которой осуществляется запись новой конфигурации, может быть различной.

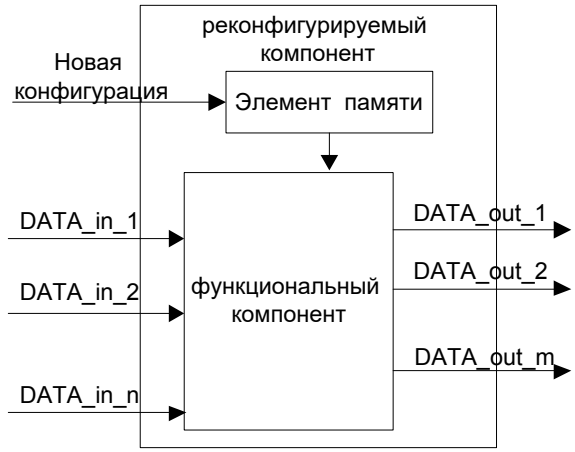


Рис. 1. Обобщенная структура реконфигурируемого компонента

Разработчик может определять различную степень гранулярности конфигурирования СенК в соответствии с решаемыми задачами и технологическими ограничениями. Минимальной реконфигурируемой единицей/модуль (reconfiguration unit) может быть [14, 15]:
- (1) фрагмент СенК (кластер из нескольких терминальных узлов и маршрутизаторов / коммутаторов) (пример СенК представлен на рис. 2);
- (2) отдельный терминальный узел, маршрутизатор, коммутатор (пример СенК представлен на рис. 3);
- (3) часть терминального узла, маршрутизатора, коммутатора (пример СенК представлен на рис. 3, однако в отличие от предыдущего варианта возможна смена конфигурации для отдельных блоков, отдельный контроллер конфигурации для каждого блока, как правило, не реализуется, т.к. это приводит к недопустимо большим аппаратным затратам).

Вариант (1), как правило, используется в СенК, в которых какое-либо влияние выполняемых задач друг на друга должно быть исключено. В этом случае для каждой задачи выделяются один или несколько реконфигурируемых модулей (в одном модуле в один момент времени выполняется только одна задача). Варианты (2) и (3) могут использоваться для систем, в которых подобные требования отсутствуют.

СенК, проектируемые в соответствии с вариантами (1), (2), (3) могут реализовывать сходные наборы функций, поэтому для их реализации потенциально могут быть использованы одни и те же функциональные IP-блоки (например, IP-блоки маршрутизаторов, коммутаторов каналов, процессорных ядер, и др., а также IP-блоки с меньшей гранулярностью). Далее мы рассмотрим, как должны проектироваться IP-блоки, чтобы обеспечить такую возможность.

Каждый реконфигурируемый модуль должен включать в себя память для хранения одной (или нескольких конфигураций). Эта память с одной стороны должна быть доступна менеджеру конфигурации, с другой стороны, хранимая в ней конфигурация должна поступать на элементы реконфигурируемого модуля. В простейшем случае менеджер конфигурации может записывать данные непосредственно в элемент памяти, входящий в состав реконфигурируемого компонента (рис. 4А). В этом случае элемент памяти может быть снабжен интерфейсом, обеспечивающим пословную запись конфигурации. В других случаях может использоваться отдельный блок памяти, доступный менеджеру конфигурации. Из этой памяти вариант конфигурации под управлением контроллера конфигурации загружается в элемент памяти, входящий в состав реконфигурируемого компонента, (рис. 4Б).

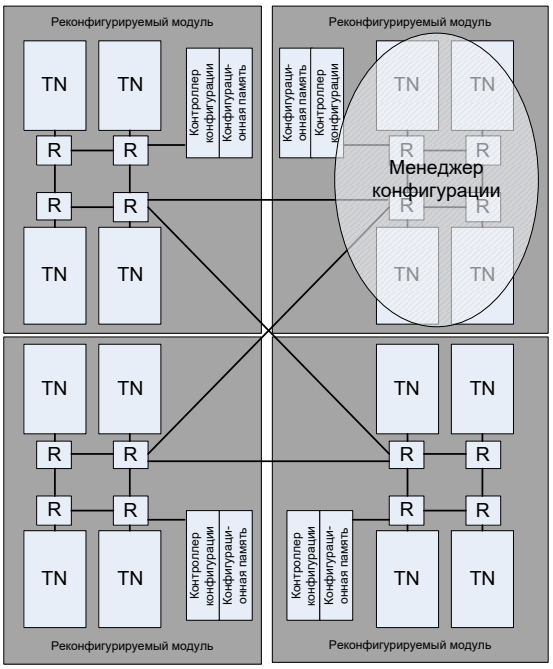


Рис. 2. Пример СенК с реконфигурируемыми модулями, включающими в себя несколько маршрутизаторов и терминальных узлов

Доступ к памяти может быть организован по-разному. В примере на рис. 2 менеджер конфигурации может обращаться к ней через основную сеть передачи данных СенК, в примере на рис. 3 – через выделенную сеть (она показана серым цветом).

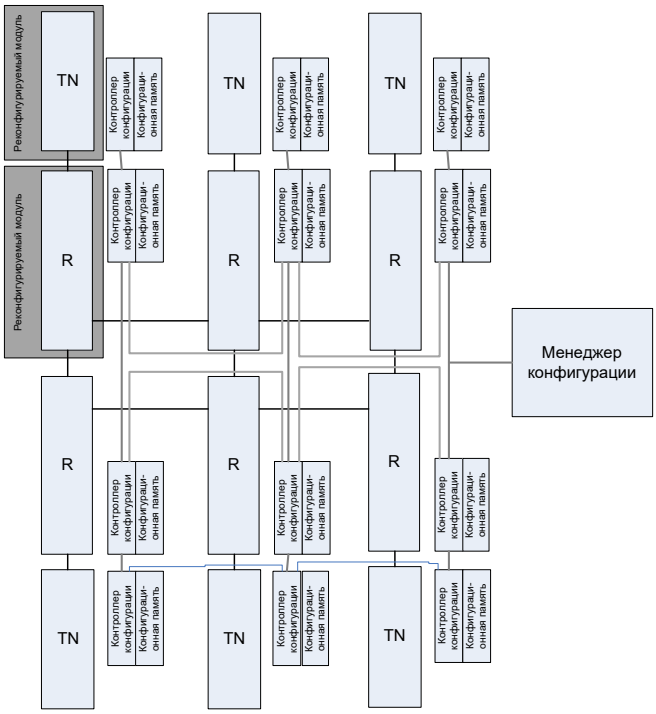


Рис. 3. Пример СенК, в которой отдельными реконфигурируемыми модулями являются маршрутизаторы и терминальные узлы

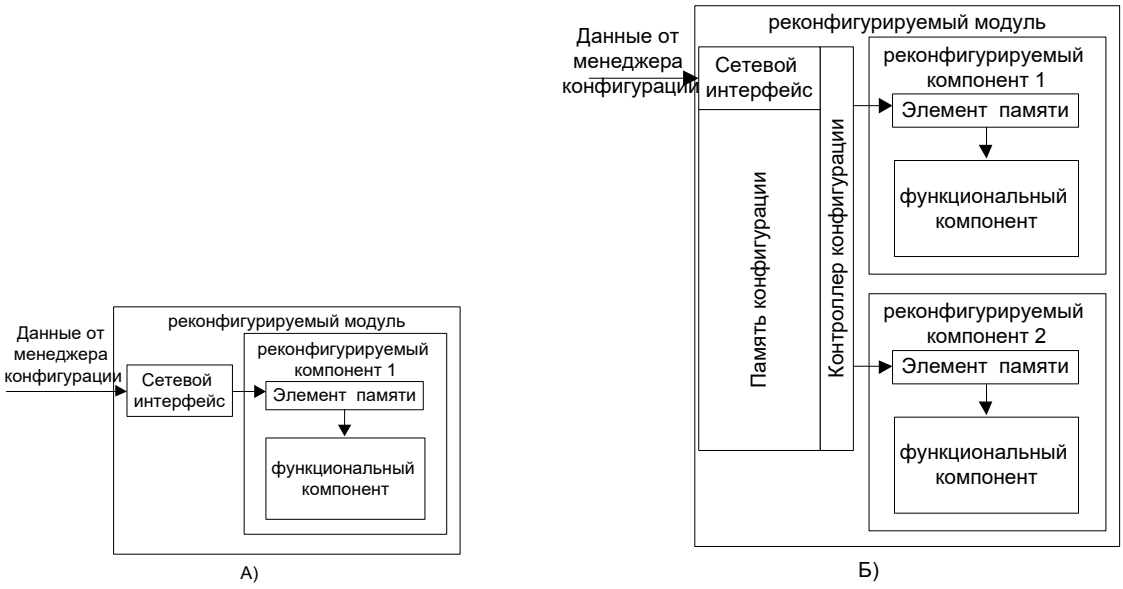


Рис. 4. Примеры структурной организации памяти конфигурации

Память может иметь «плоскую» или иерархическую структуру: при использовании плоской структуры в памяти хранится вектор конфигурации, который непосредственно передается на соответствующие входы элементов реконфигурируемого модуля. При использовании иерархической структуры в памяти могут храниться отдельно «ссылки»/номера элементов конфигурации, которые должны быть использованы и отдельно векторы/подвекторы конфигурации, соответствующие этим номерам. В соответствии с номерами определяется, какие именно векторы/подвекторы передаются на входы управления конфигурацией элементов реконфигурируемого модуля. Иерархическая структура может иметь один или несколько уровней иерархии.

Если используется схема, представленная на рис. 4А, то новая конфигурация загружается в элемент памяти непосредственно менеджером. Если загрузка конфигурации осуществляется не

одномоментно (длина конфигурационной последовательности может существенно превосходить разрядность слова в памяти конфигурации), то менеджер конфигурации должен загружать конфигурацию в такой последовательности, чтобы конфигурируемый модуль не оказался в противоречивом состоянии. Последовательность, обеспечивающая исключение возможности перехода модуля в противоречивое состояние, может быть обеспечена не для всех конфигурируемых модулей.

Это можно проиллюстрировать на примере коммутатора каналов. Конфигурация данного компонента включает в себя настройку связей между входными и выходными портами. В простейшем случае, для каждого входного порта настройка может включать в себя флаг разрешения работы и номер выходного порта, в который должны передаваться данные. В ходе реконфигурирования новые настройки могут быть определены для всех портов или только для какой-то группы портов. Во втором случае, как правило, по остальным портам передача данных в ходе реконфигурирования не должна нарушаться. (Может допускаться непродолжительный перерыв в ходе передачи данных по этим портам.) Например, до конфигурации данные из входного порта 1 должны были передаваться в выходной порт 2, а данные из входного порта 2 должны были передаваться в выходной порт 1 (рис 5А). В новой конфигурации данные из входного порта 1 должны передаваться в выходной порт 1, данные из входного порта 2 должны передаваться в выходной порт 2 (рис. 5Б). В обоих конфигурациях данные из порта 3 должны передаваться в порт 4, из порта 4 в порт 3. Например, порты 1 и 2 использовались комплектом задач 1, которые закончили свое функционирование, и вместо них должен начать выполняться комплект задач 3. Порты же 3 и 4 используются для передачи данных между задачами из комплекта 2, который продолжает свое функционирование. В процессе реконфигурирования рассматриваемый коммутатор каналов может оказаться в противоречивом состоянии.

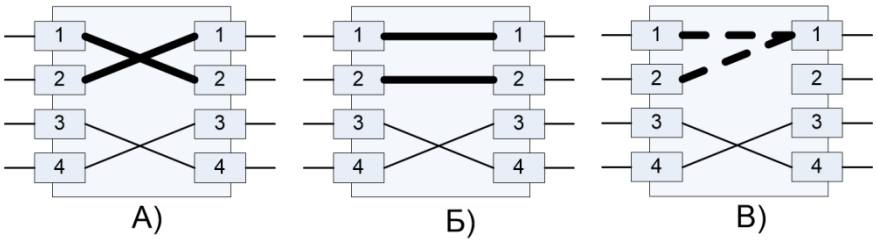


Рис. 5. Варианты конфигурации коммутатора каналов

Настройки каждого из портов могут храниться в отдельном слове элемента памяти. Запись в эти слова осуществляется последовательно. В результате, в ходе конфигурирования если запись настроек осуществляется в порядке возрастания номеров портов, коммутатор каналов окажется в состоянии, когда данные из входного порта 1 и данные из входного порта 2 должны передаваться в выходной порт 1 (рис. 5В). Если запись настроек осуществляется в порядке убывания номеров портов, то оба входных порта окажутся связанными с выходным портом 2. Соответственно, в данном случае для менеджера конфигурации не может существовать последовательности записи, которая позволит исключить переход коммутатора каналов в противоречивое состояние.

Можно, конечно, полностью остановить работу коммутатора каналов на время перенастройки и любую перенастройку рассматривать как перенастройку этого компонента целиком. Однако сам по себе останов работы может привести к возникновению противоречивого состояния, поскольку в момент выполнения останова работы по портам 3 и 4 может идти передача данных. Останов должен быть выполнен таким образом, чтобы не пропало, не исказилось слово данных, передача которого может прийти на момент останова. Либо останов должен выполняться после того, как будет завершена передача текущих пакетов по портам 3 и 4. Во втором случае процесс перехода на новую конфигурацию может существенно затянуться из-за ожидания завершения передачи пакетов. Если для рассматриваемого конфигурируемого модуля последовательность записи, исключающая противоречивое состояние, не может быть обеспечена, то такой модуль должен включать в себя функции контроллера конфигурации, обеспечивающие управление процессом загрузки конфигурации, полученной от менеджера конфигурации и исключающие переход модуля в противоречивое состояние.

Для различных СенК проектировщик может выбирать вариант передачи данных через основную или выделенную сеть, различные варианты архитектуры памяти, с которой взаимодействует менеджер конфигурации, и, соответственно, будут использоваться контроллеры

конфигурации, которые могут очень сильно отличаться друг от друга. Для конкретной СенК может разрабатываться уникальный контроллер конфигурации. Для того, чтобы разрабатываемый реконфигурируемый функциональный IP-блок мог использоваться в дальнейшем в СенК с различными схемами реконфигурации, нами предлагается следующий подход.

Будем называть элементарным реконфигурируемым IP-блоком (блоком, который уже не включает в свой состав других реконфигурируемых IP-блоков). Элементарный IP-блок должен включать в себя функциональную часть и элемент памяти. Включение обоих этих частей в IP-блок обеспечит максимально близкое их расположение в топологии кристалла. Как было показано выше, в различных системах схема загрузки новой конфигурации может быть организована по-разному, разрядность шины данных так же может быть различной и выбираться в соответствии с допустимыми аппаратными затратами. Для того, чтобы обеспечить включение реконфигурируемого IP-блока в различные системы нами предлагается следующий интерфейс. Интерфейс загрузки новой конфигурации включает в себя (рис. 6А):

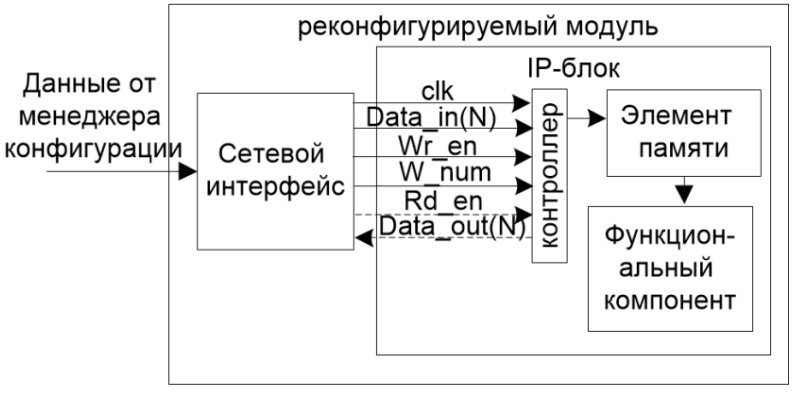
- сигнал тактирования (clk);
- линию данных на запись (Data_in);
- строб записи (Wr_en);
- номер записываемого (читаемого) слова (W_num);
- (опционально) строб чтения (Rd_en);
- (опционально) линию данных на чтение (Data_out).

Разрядность линии данных – размер слова данных, записываемого за один такт – задается параметрически (фиксируется на этапе разработки RTL модели). Она может не совпадать с разрядностью вектора конфигурации (быть как больше, так и меньше разрядности вектора). Номер записываемого (читаемого) слова указывает смещение записываемого слова относительно начала вектора конфигурации. Если количество битов между битом, с которого должно быть записано слово и концом вектора конфигурации меньше разрядности слова, то записываются только первые биты слова, аналогично при выполнении операции чтения недостающие биты заполняются нулями. Аппаратные затраты на реализацию предлагаемого контроллера интерфейса не велики (не превышают 1-5% от аппаратных затрат на реализацию блока памяти для хранения конфигурации). Элементарный реконфигурируемый IP-блок с таким интерфейсом может быть подключен к контроллерам конфигурации с различной архитектурой или напрямую к контроллерам сетевых интерфейсов СенК для обеспечения доступа менеджера конфигурации.

Несколько элементарных IP-блоков могут быть объединены разработчиком в более крупный реконфигурируемый IP-блок. В этом IP-блоке разработчик может использовать встроенный контроллер конфигурации (рис. 6Б) или вывести конфигурационные интерфейсы входящих в его состав IP-блоков на внешний интерфейс. Количество уровней вложенности при использовании этого подхода не ограничено.

Процесс замены конфигурационной информации, требования к архитектуре реконфигурируемых IP-блоков. Можно выделить следующие состояния реконфигурируемого модуля (IP-блоков, входящих в его состав) по отношению к процессу реконфигурации:

- (1) в момент, когда может быть начато реконфигурирование (в момент прихода новой конфигурации) модуль не выполняет каких-либо действий по передаче/обработке данных, в ходе выполнения конфигурации в него не поступают новые данные;
- (2) в момент прихода новой конфигурации модуль выполняет передачу/обработку данных;
- (3) в момент прихода новой конфигурации модуль не выполняет передачу/обработку данных, но в ходе выполнения реконфигурации в него поступают новые данные.



А)

Б)

Рис. 6. Примеры структурной организации реконфигурируемых модулей на базе IP-блоков

IP-блоки, входящие в состав реконфигурируемых модулей, можно разделить на:
- IP-блоки, являющиеся комбинационными схемами (функциональная часть не содержит элементов памяти), например, коммутатор каналов;
- IP-блоки, содержащие элементы памяти (например, конечные автоматы, указатели чтения и записи при реализации FIFO).

Рассмотрим возможность возникновения противоречивых состояний для IP-блоков, являющихся комбинационными схемами. Значения на выходах таких IP-блоков могут быть определены как Y=F(X,C), где X – вектор функциональных входов, C – вектор конфигурационных входов, F – функция определяющая зависимость между значениями на входах и выходах. Для такого IP-блока может быть осуществлена формальная проверка возможности перехода в противоречивое состояние, путем построения таблиц истинности. В общем случае размер таблиц истинности может быть очень большим при большой разрядности векторов X и C, что может затруднить их анализ. Однако при построении таблиц истинности вектор X, как правило, может быть разделен на две составляющие Xd – информационные сигналы и Xc – управляющие сигналы. Аналогично, вектор Y может быть разделен на Yd и Yc. Изменение информационных сигналов в системе анализируется только в привязке к изменению управляющих сигналов. Например, в блоке коммутатора каналов имеются входные сигналы данных и входные сигналы строба. Они транслируются между заданными в соответствии с текущей конфигурацией входами и выходами коммутатора каналов. При поступлении в следующие компоненты СенК (например, маршрутизатор или терминальный узел) сигналы данных анализируются только в случае, если сигнал строба установлен в активное значение. В соответствии с этим, система может перейти в противоречивое состояние, только в том случае, если в ходе изменения конфигурации произойдет некорректное изменение вектора Yc. Таким образом, формирование таблиц истинности необходимо только для Yc=F1(X,C). Для большинства систем (например, рассматриваемый выше коммутатор каналов) зависимость между Xd и Yc отсутствует, поэтому размер таблиц истинности может быть еще сокращен: Yc=F2(Xc,C). Поскольку для большинства систем разрядность информационных сигналов в разы (и даже на порядки) больше разрядности управляющих, это позволяет очень существенно сократить размер таблицы истинности, что в свою очередь делает возможным выполнение формальной проверки.

Для рассматриваемого в нашем примере коммутатора каналов противоречивые состояния могут возникать, только если по отношению к процессу реконфигурации он находится в состоянии (2) или (3). Рассмотрим, как можно исключить возникновение противоречивых состояний. Рассматриваемый компонент может быть помещен в специальную оболочку-контейнер (wrapper), обеспечивающую исключение передачи данных через компонент в процессе его реконфигурирования. Данная оболочка

обеспечивает установку всех выходов компонента, по которым передаются сигналы управления, в неактивные значения. Например, если интерфейс рассматриваемого нами коммутатора каналов включает в себя линии данных, линии признаков действительности данных и линии подтверждения приема данных, то выходные линии признаков действительности данных и выходные линии готовности приема данных в контейнере должны быть установлены в неактивное значение, когда компонент находится в процессе реконфигурации. Аппаратные затраты на реализацию данного метода очень невелики, контейнер должен включать в себя по одному логическому вентилю на каждый выход, по которому передаются сигналы управления. Он позволяет исключить возможность выдачи искаженных или некорректных данных на выходы компонента в ходе его реконфигурирования, позволяет исключить прием в реконфигурируемый компонент данных извне (что может привести к их потере или искажению). Эта структура иллюстрируется рис. 7А.

Если IP-блок используется для передачи/обработки объектов данных, поступающих в IP-блок целиком, то использование такого контейнера является достаточным для исключения противоречивых состояний. Однако, если IP-блок передается/обрабатывается не одномоментно, то такой контейнер не всегда позволяет исключить возникновение противоречивого состояния в системе в целом. Это можно проиллюстрировать следующим примером. Через рассматриваемый коммутатор каналов началась передача пакета, затем возникла пауза в передаче, во время которой поступила новая конфигурация. В этой конфигурации входной порт, по которому начал поступать пакет, связан с другим выходным портом по отношению к исходной конфигурации. В результате оставшаяся часть пакета будет передана в другой выходной порт. Исключение подобных ситуаций требует введения понятия состояния IP-блока (для нашего примера может быть определено два состояния: «идет передача пакета» и «не идет передача пакета»). Это невозможно для IP-блока, являющегося комбинационной схемой. Таким образом, реконфигурируемые IP-блоки, передача данных через которые осуществляется не одномоментно, должны разрабатываться как блоки с элементами памяти. Подходы к исключению противоречивых состояний для них будут рассмотрены далее.

Рассмотрим возможность возникновения противоречивых состояний для реконфигурируемых IP-блоков, содержащих элементы памяти. Значения на выходах таких IP-блоков могут быть определены как Y=Fy(X,C,S), где S – текущее состояние IP-блока. S(t+1)=Fs(X,C,S(t)). Для таких IP-блоков использование формальной проверки возможности перехода в противоречивое состояние по таблицам истинности в общем случае не представляется возможным (размер таблицы может быть бесконечным). Гарантированное исключение противоречивых состояний может быть достигнуто за счет организации конечных автоматов. Конечные автоматы, входящие в состав IP-блока в явном виде должны включать в себя состояния, обеспечивающие процесс реконфигурации и сохранения целостности данных в процессе реконфигурации. В автомат должно быть добавлено, как минимум, одно состояние, в котором он должен находиться во время смены реконфигурации. Должны быть определены переходы из всех остальных состояний в это состояние. В ряде случаев прямой переход в состояние реконфигурации невозможен (влечет за собой потерю целостности данных) и требуется введение дополнительных состояний, позволяющих ее исключить. Разработчик должен определить также, в какое состояние автомат должен перейти после завершения смены конфигурации.

Например, рассмотрим автомат контроллера входного порта коммутатора/ маршрутизатора, реализованный с использованием этого метода. Исходный вариант автомата состояний представлен на рис. 7Б. Команда реконфигурации потенциально может приходить во время передачи очередного пакета через рассматриваемый входной порт. Разработчик должен определить схему поведения входного порта в этом случае: пакет может быть либо передан до конца, либо хвост пакета должен быть стерт, а после уже переданной части пакета должен быть передан маркер конца пакета с индикацией ошибки. Выбранная схема должна быть реализована в конечном автомате. В конечном автомате должно присутствовать состояние, в котором выполняется его реконфигурация. В это состояние автомат должен переходить только в том случае, если обработка текущих данных завершена в соответствии с выбранным сценарием. В этом состоянии на управляющие выходы компонента должны выдаваться нейтральные значения, обеспечивающие исключение поступления в IP-блок новых данных. После завершения смены конфигурации автомат должен перейти в начальное состояние (в состояние, в которое он попадает после выхода из состояния сброса/включения питания). В автомат также может быть добавлено состояние, в котором выполняется стирание хвоста пакета. (Модифицированный вариант автомата состояний представлен на рис. 7В). Возникает вопрос, если используется автомат состояний, в который включены состояния, обеспечивающие реконфигурацию, нужно ли использовать контейнер? Если на выходы IP-блока подаются значения, формируемые в комбинационной схеме (например, если используется автомат Мили), то в ходе конфигурирования на них могут возникнуть изменения значений, поэтому в данном случае контейнер необходим.

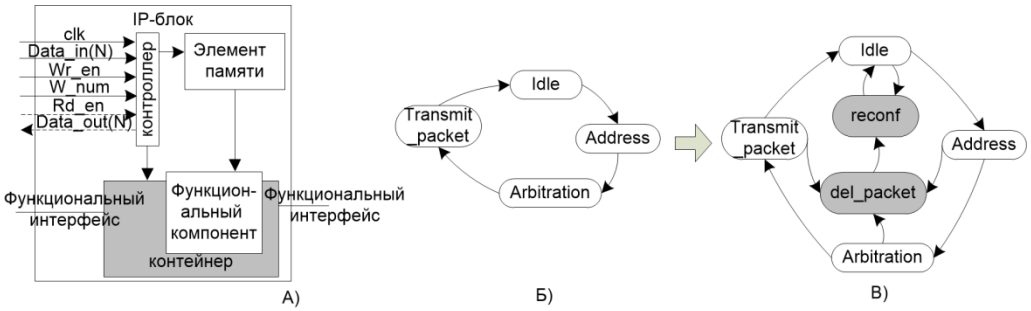


Рис. 7. Структурная и архитектурная организация реконфигурируемых IP-блоков

Не для всех реконфигурируемых IP-блоков с памятью состояние (вектор состояния) существует в явном виде, он может иметь составную структуру. В качестве простейшего примера такого блока можно рассмотреть реконфигурируемое FIFO. В нашем примере реконфигурируемое FIFO включает в себя один блок памяти, на базе которого реализовано два FIFO: блок памяти логически разделен на две части и имеется два комплекта указателей чтения и записи. В процессе реконфигурации может быть перемещена логическая граница между частями. Состояние этого IP блока определяется значениями счетчиков чтения и записи. С точки зрения сохранения целостности данных, однако, важны не сами их значения, а соотношения значений счетчиков чтения и записи в паре, соответствующей одному FIFO – это соотношение показывает, есть ли в FIFO данные, или оно пусто. Если FIFO используется для передачи/хранения объектов данных (например, пакетов), размер которых превосходит размер одной ячейки FIFO, то, даже если в рассматриваемый момент времени FIFO пусто, для корректного определения состояния необходимо учитывать также информацию, было ли передано через FIFO целое количество объектов данных или последний объект данных передан частично. Разработчик должен определить правила поведения автомата, если новая конфигурация поступает в момент, когда FIFO не пусто, или, когда через него не полностью прошел объект данных. Например, дождаться, когда из FIFO будут прочитаны все объекты данных, которые в него начали записываться на момент поступления новой реконфигурации.

Для того, чтобы избежать ошибок в ходе проектирования, для реализации этого алгоритма в IP-блоке мы рекомендуем в явном виде специфицировать конечный автомат, который будет управлять процессом реконфигурации. Кроме состояния реконфигурации он может включать в себя другие дополнительные состояния, например, состояние сброса. Применительно к данному примеру, после собственно загрузки конфигурации данный конечный автомат должен выполнить дополнительные действия: необходимо сбросить значения счетчиков чтения и записи в новые начальные значения (при изменении размеров частей, значения счетчиков могут оказаться за границами соответствующих частей).

Выводы: рассмотрен процесс динамического реконфигурирования СенК, реализованных по технологии ASIC. Определены требо-вания и ограничения, которые необходимо учитывать при разработке IP-блоков для динамически реконфигурируемых СенК. Предложена структура и организация конфигурационного интерфейса IP-блока, обеспечивающая возмож-ность включения его в реконфигурируемые модули с различной степенью гранулярности. Предложена структурная и архитектурная орга-низация функциональной части IP-блока, позволяющая исключить возникновение противо-речивых состояний системы (ошибок передачи/ обработки данных) в ходе динамического реконфигурирования.

Исследования выполнены при финансовой поддержке Министерства Образования и Науки Российской Федерации научно-исследовательской работы, выполняемой в рамках базовой части государственного задания в 2016 году по проекту № 1810.

СПИСОК ЛИТЕРАТУРЫ:

1. Partial Reconfiguration: A Simple Tutorial Richard Neil Pittman, Technical Report. February 2012, Redmond, WA 98052
2. *Weber, J.A* Reconfigurable Network-on-Chip Datapath for Application Specific Computing / *J. Weber, E. Oruklu* // Circuits and Systems. 2013. N4. P. 181-192.
3. *Осипенко, П.Н.* Микропроцессоры для космических применений // Электронные компоненты. 2010. №1. С. 66-69.
4. *Немудров, В.* Системы на кристалле. Проектирование и развитие / *В. Немудров, Г. Мартин*.— М.: Техносфера, 2014. 216 с.
5. *Murali, S.* Mapping and physical planning of networks on chip architectures with quality of service guarantees / *S. Murali* et al. // In Proc. ASP-DAC, 2005. P. 27-32.
6. Murali, S. Mapping and configuration methods for multi-use-case networks on chips / *S. Murali* et al. // In Proc. ASP-DAC, 2006. P. 146-151.

7. *Cota, E.* Reliability, Availability and Serviceability of Networks-on-Chip / *E. Cota* et al. – Springer, 2012. 209 p.
8. *Jafri, S.* Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes / *S. Jafri* et al. Microprocessors and Microsystems. 2013. V. 37, Issue 8. P. 811–822.
9. *Hansson, A.* Undisrupted quality-of-service during reconfiguration of multiple applications in networks on chip / *A. Hasson* et al. // in Proc. DATE, 2007. P. 954-959.
10. An Online and Real-Time Fault Detection and Localization Mechanism for Network-on-Chip Architectures
11. *Aisopos, K.* ARIADNE: Agnostic reconfiguration in a disconnected network environment / *K. Aisopos, A. DeOrio, P. Li-Shiuan, V. Bertacco* // In Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques (PACT'11). IEEE Computer Society, Washington, DC. P. 298–309.
12. International Technology Roadmap for Semiconductors (ITRS), 2013, URL: http://www.itrs2.net/2013-itrs.html
13. *Суворова, Е.А.* Методы динамической реконфигурации Сетей-на-кристалле на базе ASIC и FPGA / *Е.А. Суворова, Ю.Е. Шейнин* // Гагаринские чтения, ГУАП, 2015. С. 163-175.
14. *Qiaoyan, Yu* et al. Transient and Permanent Error Control for High-End Multiprocessor Systems-on-Chip. 2012. Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS) // May 9-11, 2012. P. 169-176.
15. *Vermeulen, B.* A network-on-chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs // *B. Vermeulen, K. Goossens* // in International Symposium on VLSI Design, Automation and Test (VLSI-DAT). 2009. P. 183-186.

STRUCTURAL AND ARCHITECTURAL ORGANIZATION OF IP-BLOCKS FOR DYNAMICALLY RECONFIGURABLE NETWORKS-ON-CHIP

© 2016 E. A. Suvorova

Saint-Petersburg State University of Aerospace Instrumentation

Dynamic reconfiguration of Network-on-chip (NOC) allows to essentially expand its functionality (and thereby application area) and can be used for fault mitigation. In this paper, we consider dynamic reconfiguration process of NOC produced as ASIC, requirements and constraints that should be taken into account by IP-blocks designers. We suggest an approach of IP-blocks development for NOC's with different granularity of reconfiguration modules. We suggest the rules allow to exclude faults (such as loss of data, incorrect transmission and processing of data) during configuration process.

Key words: *Systems-on-chip (SoC), Networks-on-chip (NoC), reconfigurable NoC*

Elena Suvorova, Candidate of Technical Sciences, Chief of the Systems-on-chip Laboratory. E-mail: wildcat15@yandex.ru