УДК 004.421.2

Автоматизация проектирования сетей-на-кристалле со специализированной топологией

С.О. Быков

Владимирский государственный университет, sobykov@gmail.com

***Аннотация* — Концепция сети-на-кристалле была предложена как альтернатива шинной архитектуре при проектировании сложных систем, состоящих из нескольких десятков блоков. Для однородных систем хорошо подходят стандартные топологии, однако в общем случае эффективней разрабатывать сеть со специализированной топологией, которая будет учитывать особенности конкретной системы. В данной статье описан алгоритм автоматизированного проектирования сетей со специализированной топологией.**

***Ключевые слова* — сеть-на-кристалле, автоматизация проектирования, специализированная топология.**

I. Введение

Сеть-на-кристалле – подход, предложенный для создания эффективных многопроцессорных решений. Он заключается в создании коммутационной среды, предоставляющей единый интерфейс для подключения компонентов. Элементами сети являются коммутаторы, каналы и интерфейсы между ресурсами и сетью. Ресурсами являются различные системы, интегрированные в эту архитектуру. Идея состоит в разделении области инфраструктуры и области приложений уже на физическом уровне. Данный подход позволяет создавать хорошо масштабируемые системы, а также эффективно использовать IP-блоки, преобразуя их интерфейс подключения к интерфейсу сети. Причем составные части коммутационной среды также могут являться готовыми IP-блоками, что позволяет существенно сократить время проектирования всей системы.

Одним из ключевых свойств сети-на-кристалле является топология. В соответствии с этим все подходы к проектированию можно поделить на две большие группы: подходы, использующие стандартные топологии (Сетка, Дерево, Тор и другие), и подходы, основанные на создании специализированной топологии под конкретную систему. Стандартные топологии эффективны при проектировании однородных систем (многоядерные процессоры и другие вычислительные платформы), однако в общем случае они являются избыточными. Примеры решений на базе таких топологий рассмотрены в работах [1], [2] и [8]. Создание специализированной топологии позволяет учесть особенности конкретной системы и добиться лучших характеристик, однако увеличивает затраты на проектирование. Разработка систем на базе специализированной топологии рассмотрена в работах [3] и [4].

В данной статье описан алгоритм автоматизированного проектирования сетей-на-кристалле со специализированной топологией на базе топологии «Сетка».

II. Описание алгоритма

Проектируемая система представляется в виде списка связей между компонентами. При этом для каждой связи указывается требуемая пропускная способность, например:

N1 – N2 500 Мб/с.

Перед началом работы данный список сортируется в порядке убывания пропускной способности. Это позволяет размещать активно взаимодействующие компоненты в более выгодных условиях.

Весь процесс проектирования разбивается на 3 стадии: начальное размещение компонентов, размещение коммутаторов (создание топологии) и создание каналов с временным разделением. Далее подробно описаны все стадии.

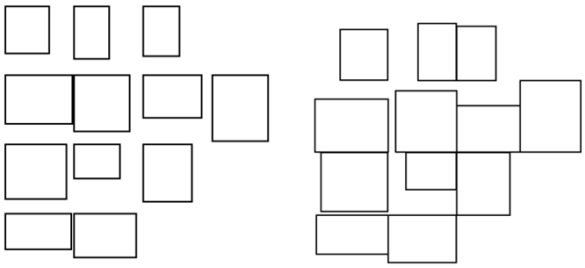
A. Начальное размещение компонентов

Задачей данной стадии является размещение активно взаимодействующих компонентов рядом друг с другом для уменьшения загруженности всей сети. Для решения этой задачи используется алгоритм, основанный на алгоритме размещения для топологии «Сетка» [5]. Получаемое решение не является оптимальным с точки зрения использования площади кристалла, однако позволяет получить эффективное решение в приемлемые сроки и упрощает анализ системы на следующих стадиях.

На первом шаге алгоритма создается топология «Сетка» с размером ячейки, который может вместить любой компонент системы. После этого компоненты размещают в сформированной исходной структуре в соответствии с алгоритмом для используемой топологии.

Следующим шагом является уплотнение компонентов на основании их реальных размеров. При этом высота ячеек для каждой строки устанавливается равной максимальному значению высоты в этой строке, а ширина ячеек для каждого столбца устанавливается равной максимальному значению ширины в этом столбце. В завершение данной стадии все компоненты сдвигаются к центру топологии, что позволяет умень-

шить занимаемую площадь. Пример уплотнения приведен на рис. 1.



а) до уплотнения б) после уплотнения

Рис. 1. Пример уплотнения компонентов

После получения окончательного размещения компонентов алгоритм переходит к следующей стадии.

B. Размещение коммутаторов

На данной стадии происходит размещение коммутаторов и расчет необходимого их количества. Коммутаторы размещаются по углам ячеек. Вследствие начального размещения, это позволяет сократить длину соединений между ресурсами и коммутаторами, поскольку в углах проще обеспечить подключение нескольких ресурсов одновременно.

Задача размещения коммутаторов формулируется и решается в виде задачи линейного программирования (ЗЛП). Формулировка задачи приведена далее.

Пусть $V = \{v_1, v_2, \dots, v_N\}$ – набор компонентов системы.

Пусть $E = \{e_1, e_2, \dots, e_M\}$, где $e_k = \{v_i, v_j\}$ – набор связей в системе.

Для каждого компонента v_i, пусть $R_i = \{r_{i1}, r_{i2}, r_{i3}, r_{i4}\}$ означает набор коммутаторов, к которым может быть подключен v_i.

Теперь определим следующие переменные:

Пусть $X_{i,j}$ будет целочисленной переменной, которая равна 1, если компонент i подключен к коммутатору j, иначе равна 0. Всего получим $4 * |V|$ таких переменных.

Пусть $X_{i,j,k,l}$ будет целочисленной переменной, которая равна 1, если компонент i подключен к коммутатору j и компонент k подключен к коммутатору l, иначе равна 0. Эта переменная определена, только если $(v_i, v_k) \in E$ или $(v_k, v_i) \in E$, т.е. для каждой связи в системе. Таким образом, получим $16 * |E|$ таких переменных.

Целевая функция. Целью является минимизация общих затрат на организацию передачи данных, которые могут быть выражены следующим образом:

$$\min Z = \min(\sum_{(v_i,v_k)\in E} \sum_{j\in R_i} \sum_{l\in R_k} a_{i,j,k,l} * X_{i,j,k,l}),$$

где параметр $a_{i,j,k,l}$ является производным от требуемой пропускной способности для связи (v_i, v_k) и Манхэттенского расстояния между коммутаторами j и l. Значение $a_{i,j,k,l}$ может быть выражено как $a_{i,j,k,l} = \omega_{i,k} * dist_{j,l}$, где $\omega_{i,k}$ – требуемая пропускная способность для связи (v_i, v_k), а $dist_{j,l}$ – Манхэттенское расстояние между коммутаторами j и i. Величина $dist_{j,l}$ рассчитывается на основании начального размещения.

Следует заметить, что в целевой функции учитываются только те пары компонентов, между которыми существует связь.

Теперь определим следующие ограничения:

1) Каждый компонент должен быть подключен только к одному коммутатору, т.е.

$$\forall v_i \in V, \sum_{j\in R_i} X_{i,j} = 1.$$

2) Если компонент i подключен к коммутатору j, то все данные от него должны проходить через данный коммутатор, т.е.

$$\forall (v_i, v_k) \in E, \forall j \in R_i \sum_{l\in R_k} X_{i,j,k,l} = X_{i,j}, \quad (1)$$

$$\forall (v_i, v_k) \in E, \forall l \in R_k \sum_{j\in R_i} X_{i,j,k,l} = X_{k,l}. \quad (2)$$

Если использовать только переменные $X_{i,j,k,l}$, то первое ограничение (1) может быть представлено следующим равенством:

$$\forall (v_i, v_k) \in E, \forall (v_i, v_n) \in E, \forall j \in R_i$$

$$\sum_{l\in R_k} X_{i,j,k,l} = \sum_{m\in R_n} X_{i,j,n,m}.$$

При таких же условиях второе ограничение (2) может быть представлено следующим равенством:

$$\forall (v_i, v_k) \in E, \sum_{j\in R_i} \sum_{l\in R_k} X_{i,j,k,l} = 1.$$

Описанная формулировка может быть использована в любом средстве решения ЗЛП для получения результирующей топологии. При этом коммутаторы, по итогам решения располагающиеся на соприкасающихся углах компонентов, объединяются и в дальнейшем рассматриваются как один узел.

C. Создание каналов

На данной стадии происходит расчет таблиц маршрутизации для всех коммутаторов, обеспечивающий создание каналов передачи данных соответственно входному описанию системы.

Для максимального использования ресурсов сети для организации передачи данных было выбрано решение на основе создания виртуальных каналов с временным разделением (Time Division Multiplexing Virtual Circuits, TDM VC)[6]. В данном решении пакеты передаются без задержек, при условии, что их поступление в сеть происходит с предварительно рассчитанной скоростью, а коммутаторы работают по предварительно заданным таблицам маршрутизации. Из-за того, что перед началом работы сети необходима фаза кон-

фигурации, TDM VC неэффективно для систем, где требуется динамическое создание каналов передачи данных. Однако для конкретных систем с заранее известными параметрами фазу конфигурации можно производить на этапе проектирования, в результате чего сеть получается сразу готовой к работе.

В связи с тем, что для размещения компонентов использовался алгоритм на базе алгоритма для топологии «Сетка», а коммутаторы расположены по углам ячеек, то для маршрутизации можно использовать простейший алгоритм *XY*-маршрутизации, когда пакет последовательно передается по осям координат.

В соответствии с выше изложенным был разработан следующий алгоритм создания каналов:

1) Для каждой связи в системе рассчитывается маршрут на основе *XY*-маршрутизации.

2) Для каждого маршрута в портах коммутаторов, через которые он проходит, резервируется пропускная способность, требуемая для соответствующей связи.

3) Для каждого маршрутизатора формируется циклическая таблица, определяющая прохождение пакетов через него. Размер этой таблицы рассчитывается на основании максимально загруженного порта для всех каналов, проходящих через рассматриваемый маршрутизатор. Порядок строк в таблице задают таким образом, чтобы обеспечить последовательную передачу пакетов от коммутатора к коммутатору для всех каналов в системе.

Результатами работы данной стадии являются рассчитанные таблицы маршрутизации для всех коммутаторов, а также минимальная пропускная способность портов, которая необходима для выполнения всех требований к системе. На основании этой пропускной способности могут быть подобраны соответствующие частота работы сети и разрядность линий связи.

III. Эксперимент

В данной главе представлены результаты проектирования с помощью разработанного алгоритма. В качестве проектируемой системы было выбрано PiP (Picture-in-Picture) устройство. Граф, описывающий данное устройство, представлен на рис. 2 [7]. Требуемая пропускная способность указана в МБ/с.

На базе графа формируется список связей в системе:

inp_mem1hs 128

inp_mem1 inp_mem2 64

hs vs 64

vs jug1 64

jug1 mem 64

inp_mem2 jug2 64

jug2 mem 64

mem op_disp 64.

Полученный список является исходными данными для процесса проектирования.

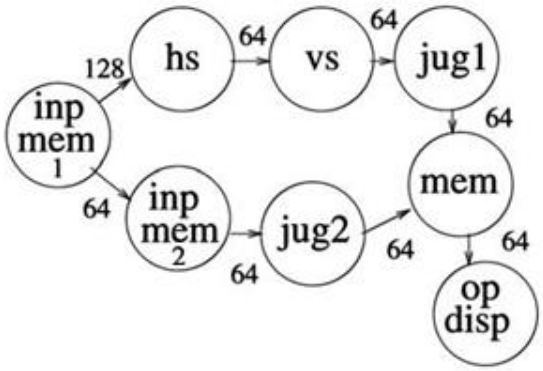


Рис. 2. Граф, описывающий PiP устройство

В соответствии с алгоритмом на первом шаге выполняется начальное размещение компонентов, после чего в полученную структуру помещаются коммутаторы. Результат размещения компонентов и коммутаторов представлен на рис. 3.

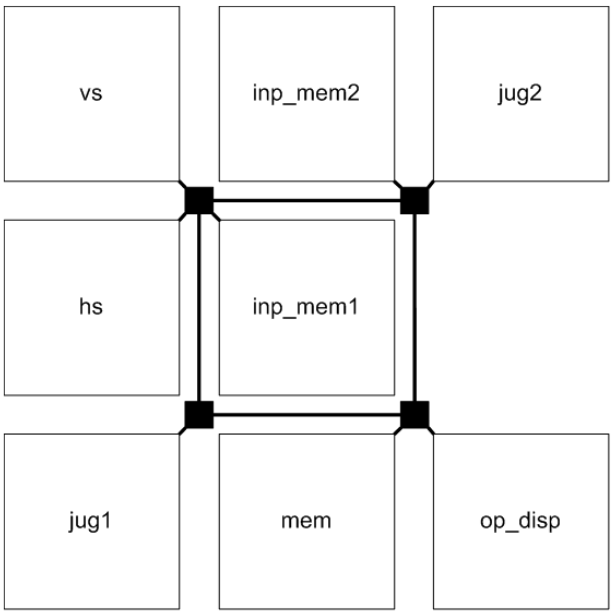


Рис. 3. Результат размещения компонентов и коммутаторов

Следующим шагом является создание каналов передачи данных. В связи с тем, что таблицы маршрутизации являются объемными, в данной статье представлена таблица только одного коммутатора, а именно того, к которому подключен компонент «inp_mem1». Данный коммутатор выбран из-за того, что компонент «inp_mem1» работает с двумя каналами шириной 128 МБ/с и 64 МБ/с, и ему требуется самая большая пропускная способность соединения: $128 + 64 = 192$МБ/с. Соответственно данная пропускная способность портов будет являться минимальной необходимой для выполнения заданных требований к системе. Данные таблицы маршрутизации рассматриваемого

коммутатора представлены в табл. 1. Нумерация портов производится с левого верхнего по часовой стрелке.

Таблица 1

Таблица маршрутизации

Номер такта цикла	Коммутируемые порты		Требуемая пропускная способность, МБ/с
1	5	3	128
	1	4	64
2	5	3	128
	1	4	64
3	3	2	64
	5	1	64

На примере пятого порта рассмотрим принцип организации коммутации нескольких каналов в один. Для этого цикл работы коммутатора разбивается на три такта. При этом пропускная способность на каждом такте становится равной: $192 \div 3 = 64$ МБ/с. После этого два такта отводятся для соединения с шириной канала 128 МБ/с и один – для соединения с шириной канала 64 МБ/с. Таким образом, учитывая полученную пропускную способность на одном такте, обеспечиваем выполнение заданных требований.

Для оценки эффективности полученного решения эта же система была спроектирована на базе топологии «Сетка». Результат представлен на рис. 4.

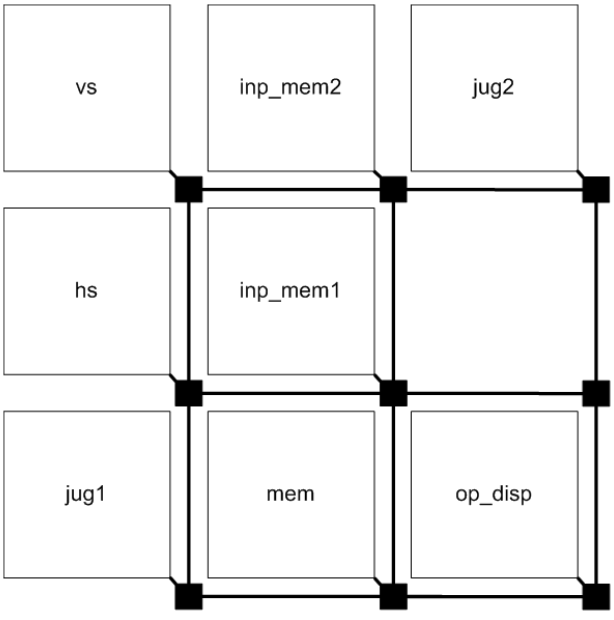


Рис. 4. Решение на базе топологии «Сетка»

Сравнение характеристик двух решений представлено в табл. 2.

Таблица 2

Сравнение решений на базе специализированной топологии и на базе топологии «Сетка»

Параметр	Спец. топология	«Сетка»
Количество коммутаторов	4	9
Количество линий связи	12	20
Минимальная пропускная способность порта (МБ/с)	192	192

Как видно из таблицы, решение на базе специализированной топологии позволило сократить количество коммутаторов на 56%, количество линий связи на 40%, при этом полученная производительность сети позволяет обеспечить такие же характеристики для данной системы. Однако следует отметить, что из-за уменьшения количества коммутаторов в специализированной топологии уменьшается количество резервных путей и, следовательно, надежность системы.

IV. Заключение

Сети-на-кристалле являются активно развивающимся направлением, и проблема автоматизации их проектирования остается актуальной. В данной статье описан алгоритм автоматизированного проектирования сети-на-кристалле со специализированной топологией. На примере проектирования PiP устройства была показана эффективность полученного решения по сравнению с решением на базе стандартной топологии «Сетка».

Литература

[1] Jantsch A., Tenhunen H. Networks on Chip. Kluwer Academic Publishers, 2003. 312 p.

[2] Rantala V., Lehtonen T., Plosila J.Network on Chip Routing Algorithms // TUCS Technical Report. 2006. №779. 34 p.

[3] Srinivasan K., Chatha K. S., Konjevod G. Application Specific Network-on-Chip Design with Guaranteed Quality Approximation Algorithms // Proc. ASP-DAC '07. 2007. P. 184–190.

[4] Srinivasan K., Chatha K.S. ISIS : A Genetic Algorithm based Technique for Custom On-ChipInterconnection Network Synthesis // Proc. VLSID '05. 2005. P. 623–628.

[5] Bykov S.O. Algorithm for automated custom Network-on-Chip topologies design // Proc. EWDTS '13. 2013. P. 1–3.

[6] Gebali F., Elmiligi H., El-Kharashi M.W. Networks-on-chips : theory and practice. Taylor & Francis Group, LLC. 2009. 355 p.

[7] Bertozzi Et Al. Noc Synthesis Flow For Customized Domain Specific Multiprocessor Systems-On-Chip // IEEE Transactions On Parallel And Distributed Systems. 2005. V. 16. № 2. P. 113–129.

[8] M. Schoeberl, F. Brandner, J. Sparsø A Statically Scheduled Time-Division-Multiplexed Network-on-Chip for Real-Time Systems // Proc. NOCS'12. 2012. P. 152-160.