УДК 004

ПРИМЕНЕНИЕ ФПП ПРИ ПРОЕКТИРОВАНИИ СИСТЕМ НА КРИСТАЛЛЕ

Кадышева Н.О, Кадышев Д.А.
Сибирский Федеральный Университет

Современная радио и микроэлектроника, микропроцессорные системы и радиотехнические устройства неразрывно связаны с жизнедеятельностью человеческого общества. Достижения практически всех существующих направлений в науке и технике немыслимы без применения микропроцессорной техники, цифровых и аналоговых интегральных схем и систем на их основе. На сегодняшний день в мировой практике освоено достаточно большое количество методов и технологий изготовления интегральных схем (ИС), систем на кристалле (СнК) и в корпусе (СвК).

Неуклонное развитие данного направления диктует жесткие условия для создания новейших методологий проектирования ИС. Здесь основное внимание уделяется проектированию однокристальных, специализированных микропроцессорных систем со сложной архитектурой. Постоянно совершенствующиеся технологии производства кристаллов и переход к новым технологическим нормам производства обуславливают активное внедрение передовых достижений микроэлектроники. Развивается производство трехмерных кристаллов, появляются однокристальные, многопроцессорные системы для параллельных вычислений, активно развивается направление производства сверхбольших интегральных схем (СБИС) с динамически реконфигурируемой архитектурой и т.д.

Но оборотной стороной прогресса является целый ряд проблем связанных с разработкой размещаемых на кристалле систем. И здесь, в первую очередь следует выделить глобальную проблему разрыва между числом логических элементов предлагаемых технологией на одной подложке и числом элементов, которые могут быть реально спроектированы и верифицированы в экономически целесообразные сроки.

Кроме того, при разработке СБИС решаются задачи системного и алгоритмического плана при условии территориального разделения коллектива разработчиков, задачи оптимизации проектных решений в условиях ограниченности вычислительных ресурсов, задачи поиска архитектурных решений при условии временных ограничений на разработку диктуемых финансовыми соображениями и др.

С увеличением объемов проекта возникают серьезные разногласия отдельных групп разработчиков, поскольку алгоритмисты, инженеры и проектировщики оперируют собственными понятиями и зачастую в процессе работы над проектом просто «перестают понимать друг друга».

С увеличением объемов проекта на первый план выдвигаются проблемы верификации и тестирования готовых систем. Сегодня, в маршруте проектирования верификация достигает 60-80% общего времени разработки.

При этом остаются и основные инженерные задачи по обеспечению надежности, ограничениям функционирования в режиме реального времени, тяжелых условиях эксплуатации и т.д.

К сожалению, на сегодняшний день в достаточной степени разработаны и автоматизированы только заключительные, низкоуровневые этапы создания проекта СБИС. Так же, удовлетворительными можно считать результаты, достигнутые в областях высокоуровневого модульного, секционного и платформенного подходов. Однако большинство задач системной организации процесса проектирования, задач эффективной архитектурной организации однокристальных систем параллельной обработки потоков и систем с динамически реконфигурируемой архитектурой до сих пор остаются нерешенными.

В отличие от заключительных этапов, при описании проекта на верхних уровнях иерархии закладываются концепции общесистемного взгляда на организацию всего процесса проектирования. Таким образом, на первый план выходит развитие маршрутов и

технологий, базирующихся на принципах высокоуровневого, архитектурно независимого проектирования, позволяющих осуществлять формирование комплексного подхода к организации всех фаз создания проекта.

Исследованиям методов проектирования вычислительных систем на кристалле посвящены работы А.Л. Стемпковского, С.Г. Русакова, А.Н. Терехова, В.В. Топоркова, А.К. Кима, А.Е. Платунова, В.Г. Немудова, специалистов ИППМ РАН, НИВЦ МГУ, МИЭТ, МИФИ, ИПУ РАН, ИСП РАН, ПОМИ им. В.А.Стеклова, СПбГУ ИТМО, ИВМ СО РАН, ФГУП НИИМА «Прогресс», ОАО НПЦ «Элвис» и др. Из зарубежных специалистов в первую очередь следует отметить работы А. Санджованни-Винсентелли, Е. Ли, А. Феррари, Г. Мартина, Г. Аха, А. Джеррайи и др.

Тем не менее, известные работы не позволяют однозначно утверждать о создании эффективных методологий высокоуровневого проектирования сложных однокристальных систем с динамически реконфигурируемой архитектурой не имеющих привязки к конкретной целевой платформе реализации. В большинстве случаев проектируемая модель системы разрабатывается под конкретную архитектуру на прикладном языке на основе стандартных или рекомендованных производителем библиотек, используемых для конкретного воплощения в целевой кристалл или платформу. При этом имеются два основных подхода к использованию существующего программно-алгоритмического инструментария проектирования.

Первый основан на применении интегрированных систем низкоуровневого проектирования достаточно широко представленных на рынке ведущими компаниями данного сегмента, например Cadense, Mentor Graphics и др. Такие системы позволяют эффективно решать большинство существующих задач, однако с увеличением сложности проекта и при решении задач проектирования систем с динамически реконфигурируемой архитектурой влекут за собой весь шлейф проблем связанных с системной организацией процесса проектирования. Как следствие - увеличение сроков проектирования и снижение экономической целесообразности разработки.

Второе направление базируется на попытках использования высокоуровневых программных средств, которые ранее использовались либо с традиционными вычислительными системами либо были ориентированы на решение совершенно других задач, например MathLab, LabView и т.д.

Такие системы, несмотря на их мощность, не обеспечивают эффективного переноса разработанных алгоритмов на архитектуру программируемых логических интегральных схем (ПЛИС) за счет большого семантического разрыва между алгоритмическим описанием прикладной задачи и архитектурной организации целевого кристалла. Кроме того при данном подходе используется понятие «крупноблочного проектирования», что при создании проекта однокристальной системы параллельной обработки информационных потоков ведет к решениям на базе традиционных принципов распараллеливания, например применение несколько процессорных ядер или использование конвейерного распределения потоков.

Таким образом, несмотря на значительные успехи, достигнутые в области традиционного проектирования СБИС и перспективных направлений высокоуровневой разработки архитектурных решений, при разработке систем параллельной обработки данных и систем с реконфигурируемой архитектурой существует еще целый ряд задач:

В первых: Отсутствуют технологии эффективной выработки архитектурных решений для однокристальных систем параллельной обработки информационных потоков с динамически реконфигурируемой архитектурой, не зависящих от конечной формы реализации.

Во вторых: отсутствуют инструментальные средства, обеспечивающие эффективный перенос высокоуровневого описания решаемых прикладных задач на архитектуру целевого кристалла.

В третьих: Отсутствуют методы и средства формальной высокоуровневой верификации архитектурных решений для СБИС.

В четвертых: За редким исключением, применяемые на современном этапе для описания однокристальных систем параллельной обработки данных, языки программирования, либо предназначены для схемотехнического описания, либо ориентированы на традиционное программирование.

Известные попытки решения означенных вопросов в основном направлены на устранение семантического разрыва между высокоуровневым и низкоуровневым описанием систем с использованием универсальных языков программирования или разработкой собственных языковых средств от специализированного ассемблера до высокоуровневого языка параллельного представления, например COLAMO. Такие подходы не позволяют однозначно утверждать о решении всего ряда проблем. Кроме того известные инструментальные средства не наши широкого применения и, как правило, не выходят за рамки академических проектов.

В итоге, разработка инструментальных средств обеспечивающих архитектурно независимое описание однокристальных вычислительных систем с реконфигурируемой и динамически реконфигурируемой архитектурой, обладающих естественным параллелизмом является актуальной задачей.

Параллельное программирование для конкретных архитектур широко использовалось в конце 70-х, начале 80-х годов. В это время были разработаны различные вычислительные системы, значительно отличающиеся друг от друга. В дальнейшем развитие технологии создания однокристальных микропроцессоров привело к резкому снижению эффективности подобных разработок и к нецелесообразности создания нетрадиционных архитектур. Это привело к использованию при создании высокопроизводительных систем стандартных аппаратных средств: симметричных мультипроцессоров, кластеров, массово-параллельных систем.

Использование подобных архитектур позволило унифицировать методы разработки параллельных программ, увязав их не с конкретными вычислительными системами, а с семействами архитектур ПВС. Появились легко переносимые инструментальные средства и пакеты, позволяющие выполнять одни и те же программы на вычислительных системах, имеющих разные системы команд. К подобным программным средствам следует отнести:

– OpenMP, Pthreads, ориентированный на параллельное программирование для систем с общей памятью;

– MPI, PVM ориентированные на передачу сообщений в системах с распределенной памятью;

– mpC, HMPI, предназначенные для организации вычислений в неоднородных компьютерных сетях.

Их применение поддерживалось и теоретическими работами, связанными с исследованиями принципов взаимодействия последовательных процессов. Следует отметить работы Хоара, результаты которой использовались при создании языка программирования Оккам и параллельных вычислительных систем на базе транспьютеров.

Вместе с тем языки параллельного программирования, использующие явное управление, хоть и позволяют описать максимальный параллелизм задачи, но обеспечивает это не самым удобным способом, так как:

– программист сам должен формировать все параллельные фрагменты и следить за корректной синхронизацией процессов;

– использование в языках такого типа "ручного" управления памятью может привести к конфликтам между процессами в борьбе за общий ресурс (программисту самому приходится тщательно следить за распределением памяти или явно соблюдать принцип единственного присваивания);

– разработанные программы оказываются жестко привязанными к конкретной архитектуре или к семейству архитектур (дальнейшая смена поколений вычислительных

систем или появление новых, более эффективных архитектур ведут к полной потере разработанного ПО и необходимости его переработки, что не раз случалось на различных этапах исторического развития);

– существует излишняя детализация управления вычислениями, так как, кроме управления, связанного с непосредственным преобразованием данных, программисту приходится иметь дела с механизмами, обеспечивающими использование специфических особенностей конкретной архитектуры.

Проблемы преобразования параллельных программ, написанных с использованием явного распараллеливания, оказались настолько серьезными, что данное направление не получило широкого практического развития. Вместе с тем следует отметить, что основные практические достижения параллельного программирования в настоящее время связаны с написанием программ, ориентированных на конкретные семейства архитектур.

Создание прикладных параллельных программ, ориентированных на численные вычисления и символьную обработку, удобнее осуществлять с применением функциональных языков параллельного программирования, в которых выполнение каждого оператора осуществляется по готовности его данных. Они не требуют явного описания параллелизма задачи. Достаточно указать информационную взаимосвязь между функциями, осуществляющими преобразование данных. Использование таких языков позволяет:

– создавать программы с параллелизмом на уровне операторов, ограниченным лишь методом решения задачи;

– обеспечивать перенос программы на конкретную архитектуру, не распараллеливая программу, а сжимая ее максимальный параллелизм (перенос подобных программ никоим образом не связан попытками преобразования структур управления и построением информационного графа, так как данный граф изначально построен);

– проводить оптимизацию программы по множеству параметров с учетом специфики архитектуры ВС, для которой осуществляется трансляция без учета управляющих связей программы.

Существуют различные модели параллельных вычислений и языки, построенные на их основе, которые, для описания параллелизма, используют только информационные зависимости между выполняемыми функциями. Работы в этой области условно можно разделить на два направления:

– разработка методов управления вычислениями по готовности данных;

– разработка языков и методов функционального программирования.

При проектировании сложных однокристальных систем требуется новый, архитектурно независимый подход, с одной стороны, обеспечивающий максимальное абстрагирование исходных алгоритмов от архитектуры целевого кристалла, с другой стороны, реализующий механизм перехода на RTL уровень с параллельной верификацией при проектировании без возврата к предыдущим уровням создания проекта.

Необходим такой способ представления алгоритмов на самом верхнем уровне иерархии, который, при последующем нисходящем проектировании, обеспечивал бы сохранение параллелизма, задаваемого на самом верхнем уровне, и допускал сквозную верификацию в ходе формирования топологии ПЛИС без возврата к предыдущим уровням иерархии.

В связи с этим перспективна задача поиска таких методов описания параллелизма исходных алгоритмов, которые в дальнейшем могли бы только «сжиматься» с учетом специфики ресурсных ограничений.

Языки описания аппаратуры на нижнем уровне – это языки, ориентированные на описание графа, состоящего из узлов-элементов, обменивающихся данными и связей между ними, обеспечивающих перетекание этих данных. Поэтому и на верхнем уровне целесообразно формировать описание на уровне потоков данных, а не использовать императивный стиль. Следовательно, при реализации данного подхода необходимо ориентиро-

ваться не на традиционное программирование с применением императивных языков, а использования функционально-потоковое представление программ.

Особый интерес представляет замена существующих методов высокоуровневого (прикладного) описания решаемой задачи на языковые и инструментальные средства, обеспечивающие поддержку архитектурно-независимого параллельного программирования.

Для поддержки архитектурно-независимой разработки параллельных программ в предложена функционально-потоковая парадигма, положенная в основу языка Пифагор. Предлагаемый язык обеспечивает описание разнообразных алгоритмов и имеет развитые средства по представлению параллелизма программ и данных. Использование неявного управления по готовности данных позволяет описывать максимальный параллелизм решаемой задачи. Архитектурная независимость используемой в языке модели вычислений базируется на следующих принципах.

Считается, что виртуальная машина, предназначенная для выполнения функционально-потоковых параллельных программ, имеет неограниченные вычислительные ресурсы. Это позволяет выделять для каждой операции новый вычислительный ресурс.

Организация циклов обеспечивается за счет рекурсии. Это позволяет избавиться от ресурсных ограничений, присущих циклическим фрагментам. Наличие асинхронных списков в сочетании с рекурсией позволяет формировать алгоритмы с динамически изменяемым параллелизмом.

Вместе с тем следует отметить, что используемые в языке Пифагор конструкции должны, в конечном итоге, отображаться на архитектуры реальных параллельных вычислительных систем. Формирование подобного отображения является нетривиальной задачей. Аналогичные проблемы возникают и при преобразовании функционально-потоковых параллельных программ в топологические структуры ПЛИС. Решение данной задачи позволит повысить эффективность разработки заказных интегральных схем.

Новый подход к формированию топологии ПЛИС, основанный на использовании функционально-потоковой парадигмы, обеспечивающей архитектурно независимое описание реализуемых параллельных алгоритмов позволит осуществлять описание системы на алгоритмическом уровне без привязки к конкретной реализации и разделения на программную и аппаратную составляющие, но с учетом изначального параллелизма решаемой задачи. Наряду с отсутствием явного описания параллелизма облегчается отладка и верификация параллельных программ. Разработка инструментальных средств, обеспечивающих непосредственную трансляцию с рассматриваемого языка в языковые средства, используемые при проектировании ПЛИС, позволит повысить эффективность процесса разработки.

Список использованных источников

1. Маршруты и технологии архитектурно-независимого проектирования при разработке сложных однокристальных схем специального назначения, О. В. Непомнящий, А. И. Легалов, Н. Ю. Сиротинина О. В. Непомнящий, А. И. Легалов, Н. Ю. Сиротинина, Сибирский Федеральный университет
2. Функциональный язык для создания архитектурно-независимых параллельных программ. Легалов А.И. – Вычислительные технологии, № 1 (10), 2005. С. 71-89
3. Функциональная модель параллельных вычислений и язык программирования «Пифагор». Легалов А.И., Казаков Ф.А., Кузьмин Д.А., Привалихин Д.В.