Известия Гомельского государственного университета
имени Ф. Скорины, №3(18), 2003

УДК 681.3

Метод синтеза микропрограммных автоматов

И.В.Коршунов

Основным недостатком традиционных средств проектирования цифровых устройств на программируемых микросхемах является семантический разрыв между представлением разработчика о проектируемой системе (как правило, алгоритм функционирования) и средствами представления проектируемых устройств в традиционных системах проектирования (как правило, сеть из стандартных элементов средней степени интеграции).

Таким образом, традиционные средства проектирования никак не поддерживают этот наиболее сложный этап проектирования – по заданному алгоритму функционирования устройства построить соответствующую функциональную схему (корректную!). Они лишь позволяют, имея корректную функциональную схему устройства, минимизировать соответствующие логические функции и запрограммировать микросхему так, чтобы она выполняла эти логические функции.

Многие традиционные средства проектирования предоставляют системы моделирования входных описаний для проверки корректности заданной функциональной схемы. Однако в этих системах чрезвычайно трудоемким является задание тестовых последовательностей и анализ результатов моделирования, поскольку они ориентированы на имитацию привычных для разработчиков инструментов: генератора сигналов, осциллографа, логического анализатора. Например, основным средством анализа результатов моделирования являются временные диаграммы, а также числовые значения сигналов на шинах и линиях в произвольный момент времени. Очевидно, что поиск и исправление ошибок в спроектированной функциональной схеме такими средствами являются чрезвычайно трудоемкими. И эта проблема встает тем более остро, чем более сложное устройство проектирует разработчик.

В свою очередь индустрия программируемых микросхем, развиваясь, делает все более привлекательным их использование, поскольку одновременно растут и быстродействие программируемых микросхем и допускаемая к реализации сложность проектируемых устройств. Кроме того, многие традиционные системы проектирования обеспечивают автоматическое каскадирование программируемых микросхем в случае, если проектируемое устройство не может быть размещено на одной программируемой микросхеме. Это означает, что все более и более сложные устройства потенциально эффективно исполнять на программируемых микросхемах.

С другой стороны, перед разработчиками все более остро встает проблема резкого повышения производительности, при проектировании цифровых устройств для уменьшения времени разработки конечного продукта.

Резюмируя все вышеизложенное, можно с уверенностью утверждать, что использование самых разнообразных САПР для описания чисто аппаратных (даже однокристальных) систем на базе ПЛИС неудовлетворительно по срокам и качеству, поскольку основная тяжесть работы переносится на низкоуровневое проектирование (не выше уровня регистровых передач).

Сущность предлагаемого метода

Сущность принципиально нового подхода заключается в том, что мы предлагаем автоматическую генерацию корректных функциональных схем по отлаженным микропрограммам, что приводит к ликвидации всех указанных в предыдущем пункте недостатков процесса проектирования цифровых устройств на базе программируемых микросхем и соответственно

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

к резкому сокращению сроков их проектирования и изготовления (как минимум в 10 раз). Более того, при использовании такого подхода удается совместить быстроту проектирования устройств по их алгоритмам функционирования со сверхпроизводительностью спроектированных устройств, поскольку соответствующие управляющие автоматы реализуются в жесткой логике и с максимальным распараллеливанием исполнения микроопераций алгоритма.

Ниже описываются основные возможности инструментальной системы поддерживающей предлагаемый подход к проектированию устройств на программируемых микросхемах.

Прежде всего разработчику предлагается язык для описания алгоритма функционирования устройства (а не функциональной схемы этого устройства!). Язык по синтаксису близок к языку ассемблера микропроцессора i8086. Множество инструкций включает арифметические, логические и сдвиговые инструкции, а также инструкции пересылки данных, условной и безусловной передачи управления, вызова и возврата из подпрограмм. Более того, система позволяет эффективно добавлять новые инструкции с целью наиболее удобной записи алгоритмов устройств. Например, для разработки устройств цифровой обработки звуковых сигналов могут быть полезными инструкции, выполняющие те или иные стандартные преобразования над входным звуковым сигналом. Также имеются инструкции, позволяющие явно описывать участки микропрограммы, которые должны исполняться параллельно. Использование этих инструкций может привести к значительному (для некоторых задач на несколько порядков) ускорению работы устройства.

Т.к. программирование на языке ассемблера является приемлемым не для всех разработчиков (особенно при разработке сложных устройств), то в настоящий момент ведутся работы по созданию компилятора с языка высокого уровня Си в язык описания микропрограммных автоматов. Предполагается, что компилятор будет содержать два оптимизатора. Оптимизатор, обрабатывающий внутреннее представление Си-программы (будут использованы большинство хорошо известных алгоритмов оптимизации), и специальный оптимизатор кода, учитывающий все особенности архитектуры микропрограммных автоматов. Важной частью компилятора будет являться анализатор, выявляющий блоки программы, которые могут исполняться параллельно. Т.е. пользователь полностью освобождается от решения традиционно сложных проблем, связанных с написанием параллельных программ. Для разработчиков, которых не удовлетворит работа оптимизаторов, будет предоставлена возможность реализации особо критических участков кода на языке ассемблера.

Для обеспечения корректности введенных программ предоставляется мощная современная интерактивная среда отладки. Эта среда отладки обладает стандартным интерфейсом и полным набором средств отладки, сопоставимым с лучшими из современных отладчиков для языков программирования высокого уровня, что позволяет производить поиск и устранение ошибок в программах для проектируемых устройств.

После отладки микропрограммы разработчик получает возможность сгенерировать функциональную схему микропрограммного автомата, выполняющего заданный алгоритм. Далее система HLCCAD[2] позволяет получить VHDL-описание сгенерированной схемы. Используя это VHDL-описание, любая САПР позволяет получить прошивку соответствующей ПЛИС. Например, для ПЛИС фирмы Altera Acex 1K EP1K50FC256 прошивку можно получить при помощи САПР MAX+plus II.

Организация микропрограммного автомата

В общем виде микропрограммный автомат (МПА) представляет собой управляющую систему с обратной связью. В простейшем виде МПА делится на две взаимодействующие части: Управляющий Автомат (УА) и Операционный Автомат (ОА), которые, взаимодействуя друг с другом, реализуют определенный алгоритм. УА представляет собой конечный автомат, который в зависимости от полученных микроусловий Xi и своего текущего состояния переходит в следующее состояние. В каждом состоянии УА выдает операционному автомату сигналы Yi – разрешения определенных в данном состоянии микроопераций. Т.е. микропро-

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

граммный автомат может выполнять несколько микроинструкций в одно и то же время, что является огромным преимуществом этого метода. ОА представляет собой совокупность функциональных блоков F_i, каждый из которых реализует определённую микрооперацию f_i. Некоторые блоки F_j являются блоками условных переходов. Эти блоки формируют сигналы истинности микроусловий X_i. В ОА также расположен блок памяти, который состоит из регистров и блоков ОЗУ, которые служат для хранения промежуточных и итоговых результатов вычислений.

Схема работает синхронно: по переднему фронту синхронизирующего сигнала срабатывает управляющий автомат, а по заднему происходит запись результатов микроопераций, выполняющихся в операционном автомате. Продолжительность тактового импульса определяется таким образом, чтобы успела сработать самая длинная по времени микрооперация ОА и самый длинный переход в УА.

Реализация

К настоящему моменту реализовано два вида управляющих автоматов: наиболее распространённый УА в виде схемы Уилкса-Стринжера и разработанный автором УА "сдвиговый регистр". Каждый из этих вариантов обладает рядом преимуществ и недостатков. Зачастую предпочтение тому или иному варианту можно отдать только после синтеза соответствующего автомата. Как показала практика, схема Уилкса-Стринжера наиболее употребительна при синтезе малых микропрограмм при отсутствии жестких требований к быстродействию УА. При синтезе устройств по большим микропрограммам оказалось выгоднее использовать схему "сдвиговый регистр", т.к. она в этом случае обеспечивает большую экономию резервируемых под устройство ячеек ПЛИС. Если же сравнивать эти варианты с точки зрения быстродействия, то в большинстве случаев схема "сдвиговый регистр" позволяет работать УА на более высокой частоте. Выбор схемы УА все-таки рекомендуется производить после синтеза конкретного микропрограммного автомата и сравнения его основных характеристик: количества ячеек занимаемых в ПЛИС и частоты, на которой может работать схема.

Схема Уилкса-Стринжера состоит из счётчика, дешифратора и комбинационной схемы переходов. Текущее состояние УА сопоставляется со значением, хранимым в счётчике. Дешифратор служит для преобразования состояния УА в выходные сигналы управляющего автомата Y_i. При такой реализации только на одной линии Y_i может быть установлена единица. Для поддержки возможности параллельного исполнения микроинструкций ОА ставит в соответствие с Y_i несколько функциональных блоков F_j. Это отступление от классической теории микропрограммных автоматов (когда каждому Y_i строго соответствует только один F_i) сделано для удобства синтеза и никоим образом не влияет на характеристики микропрограммного автомата.

Поясним механизм работы схемы Уилкса-Стринжера. В случае линейного исполнения микропрограммы просто инкрементируется счётчик. В случае же условного или безусловного переходов в счётчик записывается новое состояние УА, полученное от схемы переходов. Оно вычисляется по текущему состоянию УА и микроусловию X_i, соответствующему микрооперации условного перехода, исполняемой в данный момент, полученному от ОА. В случае безусловного перехода следующее состояние УА однозначно определяется по текущему. Таким образом, УА не затрачивает ни одного такта на исполнение инструкции безусловного перехода. Таким же образом исполнение условных переходов можно полностью перенести на ОА. То есть функциональные блоки условных переходов будут работать всегда, а не только когда получен соответствующий данному блоку разрешающий сигнал от УА. В этом случае УА сможет вычислять адреса переходов без затрат времени на исполнение микроопераций условного перехода. Но реализация этой идеи ведёт к существенному усложнению УА. И вопрос о целесообразности реализации данной идеи все ещё остаётся открытым. На первый взгляд эта идея даёт очевидный выигрыш в скорости исполнения, так как на операции условных переходов время не затрачивается. Примером удачного применения этой идеи является программа, в ко-

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

торой в критическом месте находится большая инструкция switch. При обычном подходе требуется минимум log(n) тактов для исполнения этой инструкции (n – количество вариантов в switch-инструкции). Но в силу усложнения схемы УА очевидно, что его общая производительность должна упасть. То есть реализация этой идеи должна повысить быстродействие исполнения некоторых микропрограмм, но в общем случае это не так.

Схема "сдвиговый регистр" названа так из-за подобия схеме сдвигового регистра. Она состоит из совокупности триггеров, каждый из которых соответствует какому-либо одному состоянию УА. В каждый момент времени только один триггер хранит единицу, остальные – нули. Номер триггера с единицей и соответствует состоянию УА. Выход каждого триггера связан с соответствующей линией Y_i.

Механизм работы схемы "сдвигового регистра" заключается в следующем. В случае линейного исполнения микропрограммы триггеры соединяются последовательно, то есть вход триггера T_{i+1} соединяется с выходом триггера T_i, то есть аналогично, как и в сдвиговом регистре. В случае условного или безусловного переходов вход триггера, который соответствует состоянию, в которое должен перейти УА, соединяется с линиями Y_i (i – номер состояния, после которого должен следовать безусловный переход в данное состояние) или X_i (i – номер логического условия, при истинности которого должен следовать условный переход в данное состояние). Если таких линий несколько, то они объединяются через логический элемент ИЛИ. При такой схеме УА его быстродействие, очевидно, должно быть максимальным, т.к. оно определяется лишь временем срабатывания логического элемента ИЛИ и триггера.

Все блоки ОА делятся на функциональные и условные. Функциональные реализуют микрооперации, которые не влияют на порядок выполнения микропрограммы. Например, операции пересылки, сложения и т.д. На выходе каждого функционального блока устанавливается тристабильный буфер, который пропускает результат микрооперации на блок памяти, только если соответствующий сигнал Y_i установлен. В противном случае на выходе тристабильного буфера устанавливается Z-состояние. То есть результат обработки данных функциональным блоком записывается в память только при установленном Y_i.

Условные блоки реализуют микрооперации условных переходов. Эти блоки представляют собой комбинационные схемы, которые выдают сигналы истинности микроусловий. Каждый блок выдаёт два сигнала X_{2n} и X_{2n+1}, где n – это номер микроусловия. Если установлен X_{2n}, то микроусловие истинно. Если установлен X_{2n+1}, то микроусловие ложно. Если УА находится в состоянии, в котором данная микрооперация не выполняется, то оба сигнала сбрасываются. Не существует ситуации, при которой оба сигнала могли бы быть установлены.

Заключение

На данный момент для реализации предлагаемого метода проектирования цифровых электронных схем в виде микропрограммного автомата предприняты следующие шаги:

- рассмотрены различные подходы к реализации цифровых схем в виде микропрограммного автомата;
- определена базовая версия языка описания микропрограммных автоматов (MPDL);
- реализована модель виртуального процессора микропрограммных автоматов для систем WINTER[3] и HLCCAD[2], которые обеспечивают разработку и отладку программного и аппаратного обеспечения соответственно;
- реализован транслятор микропрограмм, написанных на MPDL[4];
- реализован синтезатор микропрограммных автоматов. Синтезатор представляет собой программный модуль, подключаемый к системе IEESD-2000;
- начата разработка компилятора Си.

На текущий момент система имеет характер завершенного цикла проектирования, начинающегося с разработки и отладки алгоритма функционирования и завершающегося синтезом описаний спроектированных устройств на языке VHDL, по которому САПР следующего уровня могут прошивать физические микросхемы, например ПЛИС.

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

Разработанную систему можно эффективно использовать не только при разработке цифровых устройств, но и в учебном процессе, в таких дисциплинах, как “Проектирование цифровых устройств”, ”Физические основы ЭВМ” и т.д. Работая с системой, обучающиеся могут усвоить простейшие принципы построение микроконтроллеров.

Abstract

The method of microprogram automatons synthesis is described in this paper. Automatons are based on hard logic and can implement algorithmes of any complicity. Use of this method can sufficiently simplify the development of application-specific integrated circuits and reduce the time to market.

The software system consists of microprogram automaton assembler language translator, which syntax is similar to i8086 assembler language, debugger, simulator and microprogram automaton scheme generator, that automatically generates correct hardware description for the algorithm developed. The proposed method is very efficient when developing FPGA-based parallel processing devices.

Литература

1. Баранов С.И. Синтез микропрограммных автоматов, Л. // Энергия, Ленинград. отделение, 1979.

2. Литвинов В.А. Система высокоуровневого проектирования цифровых устройств (HLCCAD – High Level Chip Computer-Aided Design) // Труды международной конференции "Информационные технологии в бизнесе, образовании и науке". – Минск, 1999. – С. 179-182.

3. Ермолаев И.Ю. Технология создания интегрированной среды разработки программ для произвольного микроконтроллера // Электроника. 2000. – № 2. – С. 20-26.

4. Гончаренко И. И. RTAsm – ассемблер, настраиваемый на целевую архитектуру микропроцессора/микроконтроллера // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях, Материалы IV Республиканской научно-технической конференции студентов и аспирантов 19-22 марта 2001. С. 143-144.

Гомельский государственный
университет им. Ф.Скорины

Поступило 14.04.2003

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ