УДК 004.31

Петренко Н.Г., Софиюк А.А.

Об одном подходе к переводу интерпретатора информационных структур в ПЛИС-реализацию

Введение

Проблема продления "жизненного цикла" хорошо зарекомендовавшей себя радиоэлектронной аппаратуры стоит уже давно. Постоянное повышение вычислительных возможностей современной элементной базы, в первую очередь микросхем с программируемыми структурами, привлекает все большее число разработчиков для решения указанной выше проблемы. При этом, немаловажную роль играют постоянно расширяющиеся возможности систем автоматизированного проектирования (САПР).

В данной статье рассмотрен пример перевода цифрового устройства (интерпретатора информационных структур), реализованного на СИС и МИС, в реализацию на программируемых интегральных логических схемах (ПЛИС) фирмы Xilinx.

Постановка задачи

Интерпретатор информационных структур (ИИС) или процессор базы знаний функционирует в составе машины базы знаний (МБЗ), включающей кроме него, стандартный компьютер.

Структурная схема 16-ти разрядного ИИС, реализованного в старом элементном базисе, приведена на рис.1, где приняты следующие обозначения [1]:

- МУУ – микропрограммное устройство управления;
- ОпБ – операционный блок;
- ОЗУ БЗ – ОЗУ базы знаний;
- ОЗУ Пр – ОЗУ признаков;
- ОЗУ Д – ОЗУ данных;
- АС – адаптер связи;

- ЛШ Пр – локальная шина признаков;

- ЛШ Д – локальная шина данных;

- ISA – интерфейсная шина ISA стандартного компьютера;

- $У_{ОпБ}$ – микроприказы ОпБ;

- $У_{ОЗУ\ БЗ}$ – микроприказы ОЗУ БЗ;

- $У_{АС}$ – микроприказы АС;

- $Пр_{ОпБ}$ – признаки, вырабатываемые ОпБ;

- $Пр_{АС}$ - признаки, вырабатываемые АС.

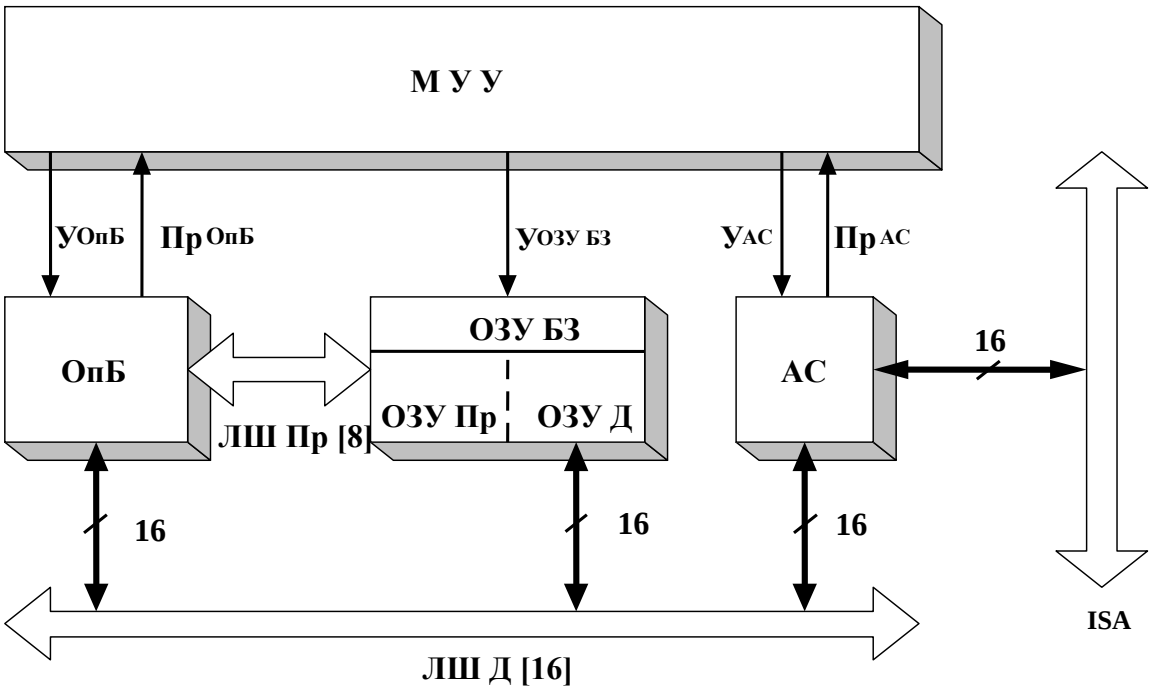

Рис.1. Структурная схема 16-ти разрядного ИИС, реализованного в старом элементном базисе

Подробную информацию, относящуюся к назначению и проектированию ИИС, можно найти в [2,3]. Архитектура МБЗ включает стандартный компьютер, интерпретатор информационных структур и адаптер связи между ними.

Решение задач пользователя обеспечивается двуединым процессом обработки пространства данных и пространства информационных структур (знаний), выполняемых соответственно на ПЭВМ и ИИС [2].

Интерпретатор информационных структур работает стандартным образом, принятым для микропрограммных процессоров, описанных, например в [4] по специальным алгоритмам, которые приведены в [1].

Недостатками реализации ИИС на старой элементной базе являются:

- конструктивная сложность устройства, заключающаяся в использовании большого количества микросхем малой и средней степени интеграции, что приводит к увеличению объема, потребляемой мощности и снижению надежности работы устройства. Кроме того, становится невозможной расширяемость конфигурации ПЭВМ в связи с отсутствием резерва как по месту в корпусе ПЭВМ так и по мощности источника питания;

- невозможность реализации современных методов обмена информацией между ПЭВМ и ИИС (ИИС может быть только ведомым);

- невозможность сколько-нибудь существенного изменения архитектуры и вычислительной мощности интерпретатора без изменения изготовленного образца устройства. При изменении архитектуры необходимо заново повторять техническое проектирование от разработки технической документации до отладки опытных образцов;

- необходимость разработки собственного микропрограммно - программного обеспечения для отладки аппаратных средств (самоконтроль и диагностика).

ПЛИС – реализация описанного выше ИИС обусловлена прежде всего перспективой избежать практически всех, описанных выше недостатков, и кроме того, обеспечить параллельный обмен данными и признаками между ИИС и стандартным компьютером за счет увеличения разрядности (по сравнению со старым парком 16-ти разрядных компьютеров) интерфейсной шины PCI. Кроме того, существенно снижена “зависимость” разработчика при выборе оптимальной структуры вычислительного устройства в рамках варьируемых параметров: затраты на оборудование и время решения задач пользователя.

Решение задачи

Структурная схема ПЛИС-реализации ИИС представлена на рис.2, где приняты следующие обозначения (за исключением аналогичных, принятых на рис.1):

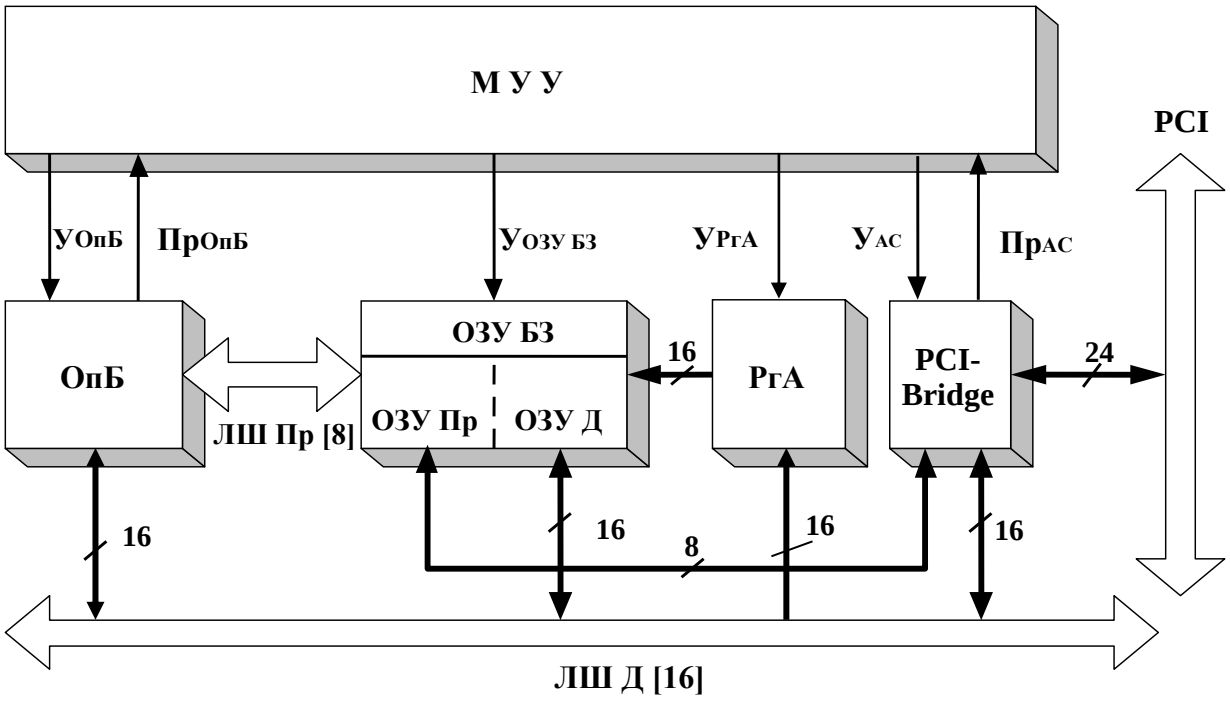

Рис.2. Структурная схема ПЛИС-реализации ИИС

- РгА – регистр адреса ОЗУ БЗ;

- PCI-Bridge – мост PCI;

- PCI – шина PCI стандартного компьютера;

- У $_{РгА}$ – микроприказы РгА;

- У $_{PCI\text{-}Bridge}$ – микроприказы PCI-Bridge;

- Пр $_{PCI\text{-}Bridge}$ – признаки, вырабатываемые PCI-Bridge.

Для реализации ИИС был использован кристалл 2S50PQ208-5 серии Spartan-II фирмы Xilinx.

Архитектура серии Spartan-II основана на архитектуре, получившей большое распространение серии Virtex с напряжением питания ядра кристалла 2,5 В [5]. ПЛИС серии Spartan-II обладают сравнительно небольшой стоимостью и могут быть использованы в качестве альтернативы специализированным интегральным схемам ёмкостью до 200000 вентилей и системным быстродействием до 200 МГц. Стоимость кристаллов Spartan-II эквива-

лентна стоимости заказных специализированных интегральных схем в партиях от 100000 штук и в 3-4 раза меньше, чем стоимость ПЛИС серии Virtex при розничных объёмах. Такое снижение цены по сравнению с серией Virtex достигнуто благодаря использованию новой технологии производства кремния и уменьшению номенклатуры корпусов. Серия Spartan-II состоит из 6-ти кристаллов, отличающихся логической ёмкостью. Основными особенностями архитектуры кристаллов серии Spartan-II являются свойства гибкости и регулярности. Кристаллы состоят из матрицы CLB (Configurable Logic Blocks – конфигурируемые логические блоки), которая окружена программируемыми блоками ввода –вывода (IOB). Все соединения между основными элементами (CLB, IOB) осуществляются с помощью набора иерархических высокоскоростных программируемых трассировочных ресурсов. Наличие таких ресурсов позволяет реализовывать на ПЛИС серии Spartan-II большие по объему и сложности проекты.

Кристаллы Spartan-II обеспечивают более высокую производительность, чем предыдущие поколения FPGA. Проекты могут работать на системных частотах до 200 МГц и частотах внутри кристалла, превышающих 350 МГц. Блоки ввода-вывода Spartan-II полностью соответствуют спецификациям PCI шины, поэтому микросхемы позволяют реализовывать интерфейсные схемы, работающие на частоте 33 МГц или 66 МГц.

Для реализации проекта в кристалле использована система проектирования Xilinx Foundation Series, поддерживающая иерархическое проектирование и содержащая ряд инструментальных средств, выполненных в виде программных модулей и объединенных в единую систему посредством проектного менеджера (Project Manager). Схема процесса проектирования в этой системе представлена на рис.3 [6]. Как следует из схемы, проектирование начинается с описания проекта, выполняемого средствами схематического редактора, редактора диаграмм состояний или HDL-языка. Помимо системных библиотек примитивов и макроэлементов, используемых при синтезе схемы, применяется такой инструмент как Core генератор, позволяющий сформировать какой-либо макроэлемент с требуемыми параметра-

ми. При описании проекта на HDL-языке и его компиляции в качестве синтезатора используется программный модуль FPGA Express фирмы Synopsys. Наиболее удобным является смешанное описание, выполняемое различными средствами. Далее проводится верификация проекта по заданным контрольным точкам схемы путем функционального моделирования на вентильном уровне или поведенческого HDL-моделирования. Входные воздействия, инициализирующие требуемые входные сигналы, формируются разработчиком.

Формирование входных воздействий осуществляется в среде системы логического моделирования (Logic Simulator) с помощью утилит Stimulator Selection и/или Test Vector State Selection. При этом входные воздействия могут задаваться как с помощью формул, так и в графическом виде. Входные воздействия могут быть также подготовлены в текстовом виде текстовым редактором (Script Editor) с помощью мастера (или без его помощи) Script Wizard.

После функционального моделирования проводится реализация проекта в кристалле, включающая в себя стадии объединения списка цепей схемы с архитектурой кристалла, размещение и трассировку кристалла (Placement and Route – PAR), формирование данных для программирования кристалла (Bitstream), интерактивную отладку (при необходимости), формирование PROM-файла для “прожига” конфигурационного ПЗУ.

После размещения и трассировки проводится статический временной анализ, а также моделирование проекта с реальными задержками на вентильном уровне и поведенческое HDL-моделирование.

Этапы реализации, связанные с получением файла конфигурации, включают в себя:

- трансляцию данных (Translate) в формате EDIF, полученных в результате синтеза схемы, во внутренний формат базы данных (NGD);

- планировку (Map) кристалла, т.е. преобразования проектных логических элементов в физические элементы, такие как CLB и IOB;

- размещение физических элементов и трассировку связей между ними (Place&Route);

- определение временных задержек в кристалле (Timing);

- генерацию файла конфигурации (Bitstream).

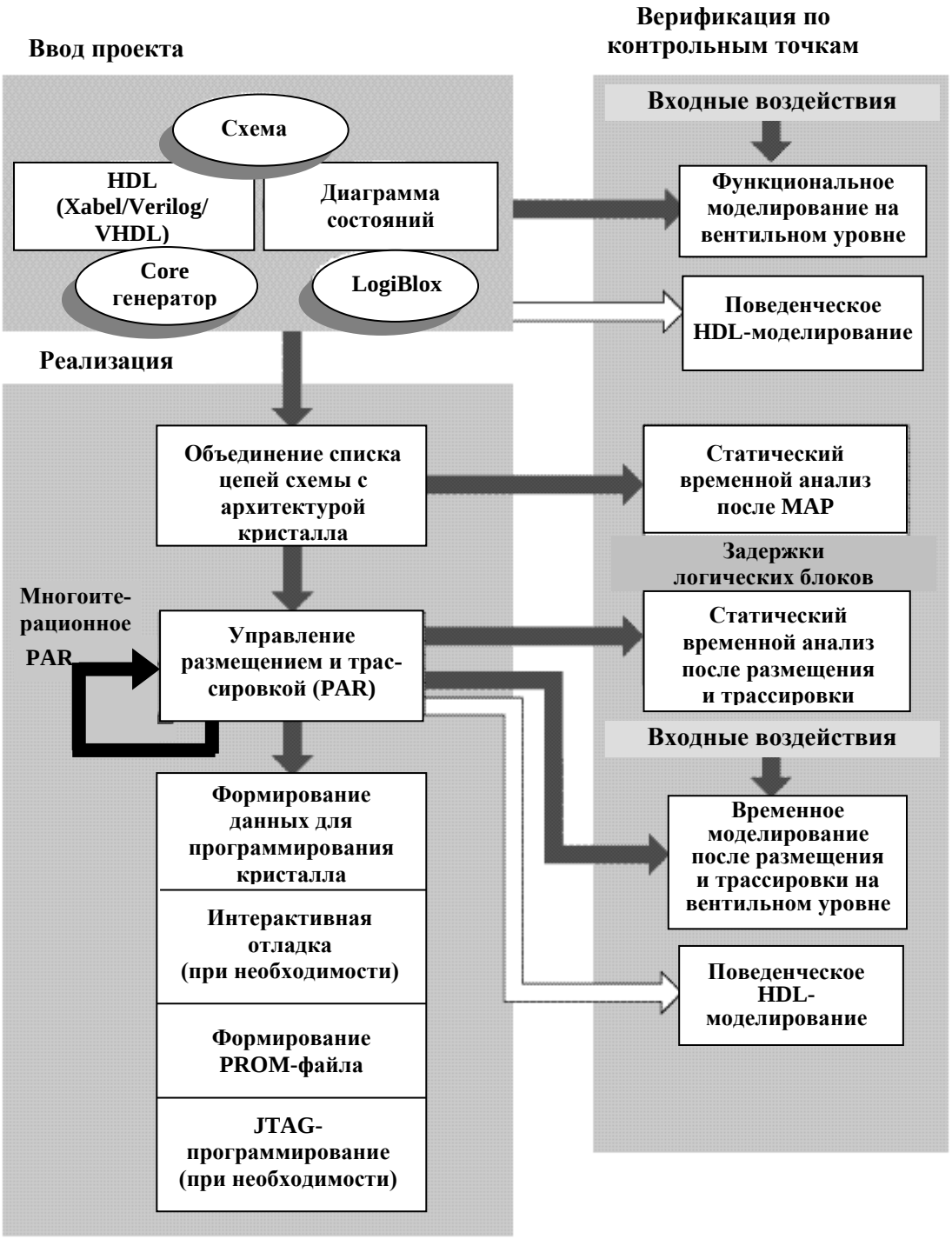

Рис.3. Схема процесса проектирования в системе Xilinx Foundation Series

Продуцируемый файл конфигурации содержит информацию о конфигурации кристалла, определяемую внутренней логикой и связями, а также спецификой используемого кристалла. Этот же файл используется для создания PROM-файла.

Управление последовательностью выполнения этапов осуществляется специальным проектным менеджером - Flow Engine.

Результаты по каждому этапу фиксируются в соответствующих отчетах. Так, на этапе трансляции формируется отчет, содержащий информацию об иерархических блоках, которые повреждены или не могут быть оттранслированы, некорректных или неполных временных ограничениях, противоречиях по выходам, ненагруженных выходах и входах, не имеющих источника. На этапе MAP формируются сообщения об удаленной логике, добавлении или расширении логики для скоростной оптимизации, количестве и процентном соотношении CLB, IOB, триггеров и “защелок”, использовании специфических архитектурных ресурсов, таких как глобальные буферы и логика пограничного сканирования. На этапе размещения и трассировки отчет содержит данные о результатах проведенных операций, количестве не полностью разведенных цепей, оценке временной производительности.

Принципы работы со схематическим редактором в основном идентичны для всех современных систем проектирования и достаточно хорошо описаны в технической литературе [7]. Ниже кратко рассматриваются основные аспекты синтеза схем на примере работы в системе Xilinx Foundation Series.

Описание схем с помощью схематического редактора.

В общем виде синтез схемы с использованием схематического редактора можно представить в виде следующей последовательности шагов:

- создание нового проекта;

- подготовка проектной (рабочей) библиотеки;

- размещение элементов схемы на поле чертежа;

- соединение выводов элементов между собой и присваивание имен сформированным связям;

- проверка схемы.

При создании нового проекта, средствами проектного менеджера системы создается пустая рабочая библиотека, наполняемая далее при формировании макроэлементов, создаваемых разработчиком. Разработчик имеет также доступ к системной библиотеке, содержащей примитивы (элементы нижнего уровня иерархии) и макроэлементы для заданного типа кристалла. Кроме системной и рабочей библиотек разработчик может включать в состав проекта библиотеки из других, ранее созданных проектов. Все операции при работе с библиотеками проводятся с использованием менеджера библиотек (Library Manager). Любой проект, создаваемый пользователем, как правило, является иерархическим и/или многолистовым. Схема, находящаяся на верхнем уровне иерархии, содержит примити-вы и/или макроэлементы, которые в свою очередь содержат примитивы и/или мак-роэлементы более низкого уровня иерархии. Многоуровневая иерархическая организация проекта по сравнению с многолистовой является более компактной и удобной для восприятия.

Подготовка рабочей библиотеки осуществляется путем разработки схем требуемых макроэлементов, на входах и выходах каждой из которых устанавливаются так называемые “терминалы” или иерархические соединители (Hierarchy Connector), указывающие на то, что данная схема является одним из уровней иерархии проекта. Иерархический соединитель может быть односигнальным, если он соединяется с одной цепью, и многосигнальным, если он соединяется с многопроводной связью (шиной). Далее из меню редактора вызывается процедура создания макросимвола из текущего листа схемы (Create Macro Symbol From Current Sheet), формирующая диалоговое окно создания макроэлемента (Create Symbol). Разработчик задает имя формируемого элемента и проверяет (при необходимости корректирует) количество, типы и имена выводов макроэлемента, определяемые иерархическими соединителями. Программа формирует внешний вид (оболочку) макроэлемента и заносит его в рабочую библиотеку. При необходимости разработчик может скорректировать оболочку с помощью редактора макроэлементов (Symbol Editor).

Размещение элементов схемы на поле чертежа производится путем выбора требуемого элемента из какой-либо библиотеки или задания его имени в поисковой строке (Search Box) окна схемных элементов (SC Symbols Toolbox).

Соединение элементов схемы между собой осуществляется путем проведения связей, которые делятся на однопроводные и многопроводные - шины. Каждой однопроводной связи и шине может быть присвоено имя, при этом именование шины производится в определенном индексном диапазоне в соответствии с правилами, зависящими от ее сложности. Шина может быть простой или сложной, если она состоит из нескольких простых шин или их сегментов.

Описание схем на языке VHDL.

Индустриальные стандарты становятся неотъемлемой частью рынка САПР. Они необходимы как для улучшения эффективности разработки программных средств САПР, так и для осуществления реальной возможности внедрения сообществом потребителей процессов оптимального проектирования электронных устройств. Это особенно актуально для развитого рынка коммерческих программных продуктов, которые должны быть интегрированы на основе единой системы стандартов, обеспечивающих проектирование и верификацию аппаратных средств информатики и электроники.

Одним из наиболее важных нововведений такого рода и по существу стандартом де-факто является язык VHDL [8]. Данный язык дает возможность описывать проект на верхнем уровне абстракции и позволяет реализовать высокоуровневое проектирование, используемое во взаимосвязи с автоматическим синтезом.

Язык VHDL предназначен для описания проектов различной степени сложности - от простейшего вентиля до целой системы, состоящей из аппаратных и программных частей. Он позволяет строить модели на различных уровнях абстракции, выполнять имитационное моделирование и генерировать временные диаграммы, вести строгое документирование

проекта, осуществлять синтез структуры по поведенческому описанию, верифицировать проект формальными методами, автоматически генерировать тесты.

Использование VHDL позволяет не привязывать проект заранее к конкретному физическому способу реализации, одна и та же логическая VHDL-реализация является источником генерации различных физических.

В основе языка лежат следующие принципы построения [9]:

- поддержка функциональной декомпозиции (функциональная иерархия и рекурсия);

- поддержка структурной декомпозиции (структурная иерархия);

- представление системы в виде параллельно функционирующих взаимодействующих процессов;

- использование абстрактных типов данных;

- использование событийного моделирования;

- поддержка различных уровней абстракции и детализации представления проекта.

В соответствии с уровнями абстракции, язык допускает построение моделей трех типов - поведенческих, потоковых и структурных.

Применение аппаратных языков описания непрерывно расширяется в связи с усовершенствованием как элементной базы, так и систем синтеза и оптимизации.

Описание цифровых структур путем задания диаграмм состояний (StateCAD).

StateCAD или State Editor служит для создания диаграмм состояний последовательностных автоматов, которые затем преобразуются в коды какого-либо HDL языка, являющиеся исходными для синтеза и оптимизации [10].

Диаграмма состояний может содержать один или несколько конечных автоматов (Finite State Machine), располагаемых на одном листе, имеющих совместно используемый интерфейс (порты) и работающих параллельно. Каждый автомат, размещаемый на диаграмме, представляется в виде прямоугольного фрейма, ограниченного рамкой.

Основными элементами диаграммы состояний являются:

- *имя диаграммы* (diagram name). Указывается в верхней части листа, на котором располагается диаграмма;

- *блок операций* (diagram actions box). Располагается в левом верхнем углу диаграммы и предназначается для определения дополнительных утверждений, которые могут быть вставлены в генерируемый HDL-код;

- *порты* (ports), используемые для связи описываемого автомата с окружающей средой. *Порты* могут быть входными, выходными и двунаправленными. Входной *порт* может быть прочитан (операция чтения), выходному *порту* может быть присвоено какое-либо значение (операция записи), для двунаправленнго *порта* может быть использована как операция чтения, так и операция записи;

-*имя автомата* (mashine name), идентифицирующее автомат внутри диаграммы и используемое в качестве идентификатора сигнала, выполняющего хранение текущего состояния автомата (в генерируемом HDL-коде);

- *состояние* (state) является внутренним состоянием автомата, определяющим значение его выходов. Каждый автомат имеет, по меньшей мере, два различных *состояния*, представляемых на диаграмме обычно в виде окружностей. *Состояния* внутри автомата идентифицируются уникальными именами, генерируемыми по умолчанию и представляющими собой последовательность S1, S2, S3 и т.д.;

- *переход* (transition) соединяет между собой *состояния* и определяет порядок изменения текущего *состояния* автомата на другое *состояние* в момент действия активного фронта синхросигнала;

- *условие* перехода (condition) определяется булевым HDL-выражением и непосредственно связывается с *переходом*, изображаемым обычно в виде прямой линии или дуги со стрелкой. Если HDL-выражение выполняется, т.е. является истинным, то *переход* в новое *состояние* имеет место. Если одновременно могут выполняться два или более HDL-выражений, связанных с *переходами* из одного *состояния* в другие, то таким *переходам* назначают-

ся приоритеты. По умолчанию, *переход* на диаграмме состояния имеет приоритет 0, который является самым высоким из возможного интервала от 0 до 9;

- *операция* (action) представляет собой набор утверждений, синтаксически соответствующих выбранному языку HDL. Утверждения определяют новые значения, присваиваемые *портам, внутренним сигналам* и *переменным*. Размещение утверждений *операции* в генерируемом коде зависит от типа элемента, которому присваивается новое значение (комбинационный *порт/сигнал*, регистровый *порт/сигнал* или *переменная*) и типа *операции*. Имеется два основных типа *операций* - *состояния* (State actions) и *перехода* (Transition actions), а также два дополнительных типа *операций*, уменьшающих число *операций перехода* на диаграмме - *входа* (Entry actions) и *выхода* (Exit actions). *Операция состояния* соответствует модели автомата Мура и выполняется, когда автомат переходит в определенное состояние. Поведение комбинационных и регистровых *портов/сигналов*, включаемых в *операцию состояния*, одинаково и HDL-утверждения выполняются на активном фронте синхросигнала. *Операция перехода* соответствует модели автомата Милли и выполняется, когда автомат выполняет определенный *переход* из определенного *состояния*. Поведение *портов/сигналов*, включаемых в *операцию перехода*, различно для комбинационных и регистровых типов. Комбинационные *порты/сигналы* асинхронны по отношению к изменениям входов автомата, регистровые изменяют свои значения только на активном фронте синхросигнала. *Операция входа* определяется для *состояния* и выполняется на каждом *переходе* к этому *состоянию*. *Операция выхода* также определяется для *состояния* и выполняется на каждом *переходе* из этого *состояния*;

- *переменные* и *внутренние сигналы* (variables/internal signals). *Переменные* декларируются внутри описываемого процесса и локальны для автомата. Если диаграмма содержит более одного автомата, то к *переменным*, объявленным для одного автомата, нельзя обращаться при описании процессов другого автомата. *Сигналы* декларируются в архитектурном теле и глобальны для всех автоматов, определенных внутри диаграммы. Следовательно, они

могут использоваться для связи между автоматами и последние внутри диаграммы могут иметь свободный доступ к определенным *сигналам*. И *переменные* и *внутренние сигналы* обозначаются на диаграмме одним и тем же символом (в виде окружности небольшого диаметра). Отличие между *переменными* и *сигналами* определяется расположением символа. Если символ находится внутри фрейма, определяющего автомат, то он является *переменной*, за пределами фрейма – *сигналом*;

- *сброс* (reset) предназначен для *перехода* автомата в *состояние*, называемое *состоянием сброса* автомата (reset state), если *условие перехода*, называемое *условием сброса* (reset condition), истинно. Переключение в *состояние сброса* осуществляется независимо от текущего состояния автомата и выполняется немедленно, если *сброс* асинхронный или на активном фронте синхроимпульса, если *сброс* синхронный. У*словие сброса* может быть представлено любым булевым выражением.

На рис.4 представлен пример описания триггера J-K с помощью State Editor и временная диаграмма работы триггера.

В соответствии с описанными процедурами была выполнена реализация ИИС в кристалле. На рис.5 приведена схема верхнего уровня иерархии, выполненная с помощью схематического редактора.

Схема состоит из двух макроэлементов: ядра контроллера (PCI-Bridge) шины PCI (PSIS) и собственно модулей ИИС, объединенных в одном макроэлементе PRO_BD.

Внешние выводы схемы, расположенные слева, предназначены для соединения с шиной PCI, справа – для соединения с внешней RAM, образующей ОЗУ БЗ.

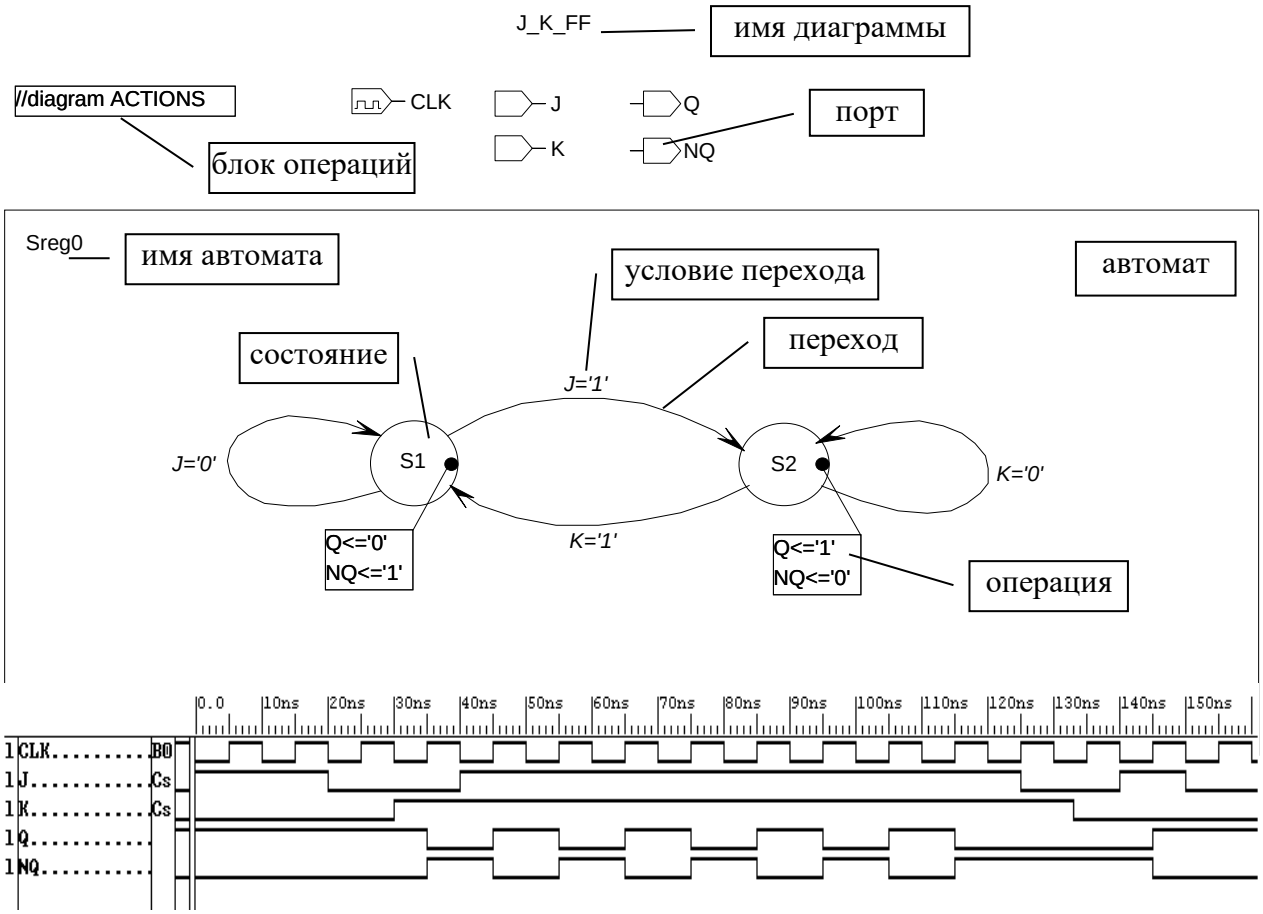

Рис.4. Пример описания триггера J-K с помощью State Editor и временная диаграмма работы триггера

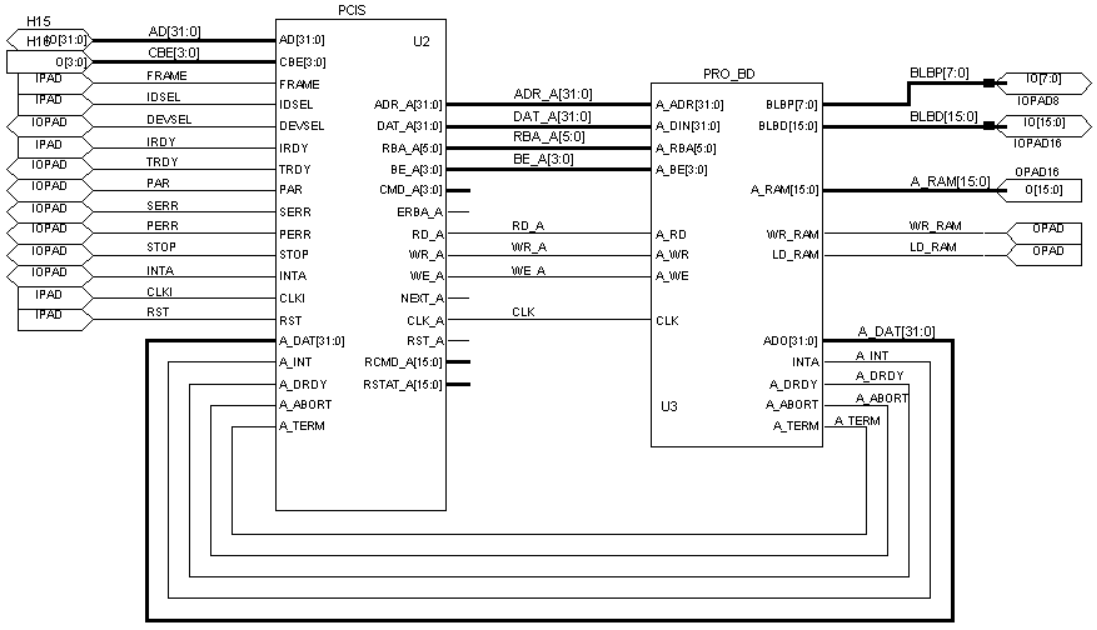

Рис.5. Схема верхнего уровня иерархии, выполненная с помощью схематического редактора

Ядро контроллера шины PCI реализует 32-разрядный обмен данными в соответствии с спецификацией [11] и выполнено в виде нескольких модулей, описанных на языке VHDL. Ядро синтезировано средствами пакета программ FPGA Express фирмы Synopsys [12]. Далее синтезированное ядро, являющееся файлом в формате EDIF (Electronic Data Interchange Format) - промышленном стандарте для обмена данными, включено в состав проекта и из него сформирован макроэлемент (PCIS), состоящий из оболочки (symbol) и списка цепей (netlist).

Макроэлемент PRO_BD содержит иерархические макроэлементы в соответствии с функциональной схемой ИИС, представленной на рис.6, и выполняет функции в соответствии с их описаниями.

Большинство разработанных макроэлементов включают в себя модули, сформированные с помощью Core генератора. На рис. 7 в качестве примера приведен внешний вид окна

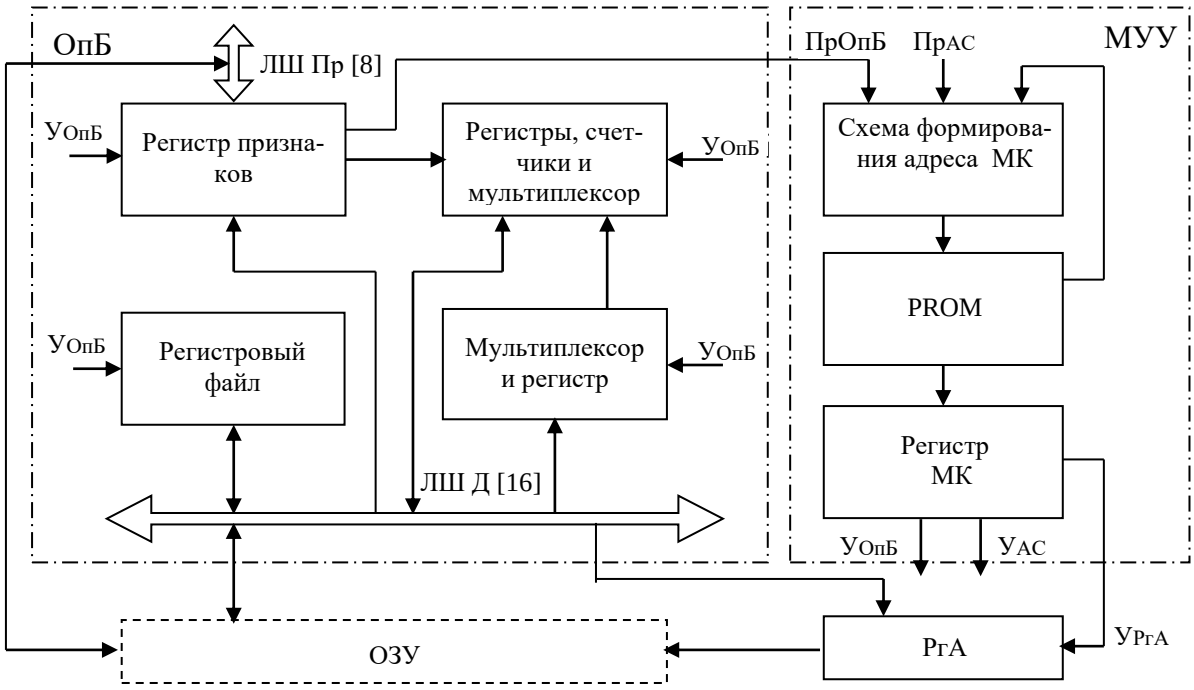

Рис.6. Функциональная схема ИИС

системы Core Generator, генерирующей функцию однопортовой блочной памяти, объемом 512 58-разрядных слов, выполняющей функцию PROM для хранения микрокоманд.

Отличительной особенностью структуры ИИС является наличие нескольких внутренних локальных шин, связывающих между собой модули, оперирующие с потоками данных и признаков. Архитектура кристалла Spartan-II содержит буферные элементы с 3-я состояния-

ми, позволяющими эффективно реализовать указанные шины. Кроме того, наличие внутреннего PROM обеспечивает высокое быстродействие устройства за счет уменьшения времени обращения к памяти при выборке макрокоманды. Использование модулей, сформированных Core генератором, существенно сокращает временные затраты на разработку проекта.

Ресурсы кристалла, затраченные на реализацию проекта:

Количество внешних выводов для глобальных CLK (GCLKIOBs) - 1 из 4 25%

Количество внешних выводов (IOBs) - 88 из 140 62%

Объем блочной памяти (BLOCKRAMs) - 8 из 8 100%

Количество слайсов (SLICEs) - 314 из 768 40%

Количество внутренних буферов с 3-я состояниями (TBUFs) - 352 из 832 42%

В таблице приведены сравнительные данные основных параметров ИИС, реализованных в ИМС средней степени интеграции и в ПЛИС серии Spartan-II фирмы Xilinx без учета RAM, выполняющей функции ОЗУ БЗ.

Таблица

Вид реализации / Параметр	Реализация ИИС в элементном базисе ИМС средней степени интеграции	Реализация ИИС в элементном базисе ПЛИС серии Spartan-II фирмы Xilinx
Кол-во корпусов ИМС, шт.	88	1+1(PROM для загрузки)
Быстродействие, нс/МГц	110/9,1	25,5/39,2
Потребляемая мощность, Вт	23,8	0,231

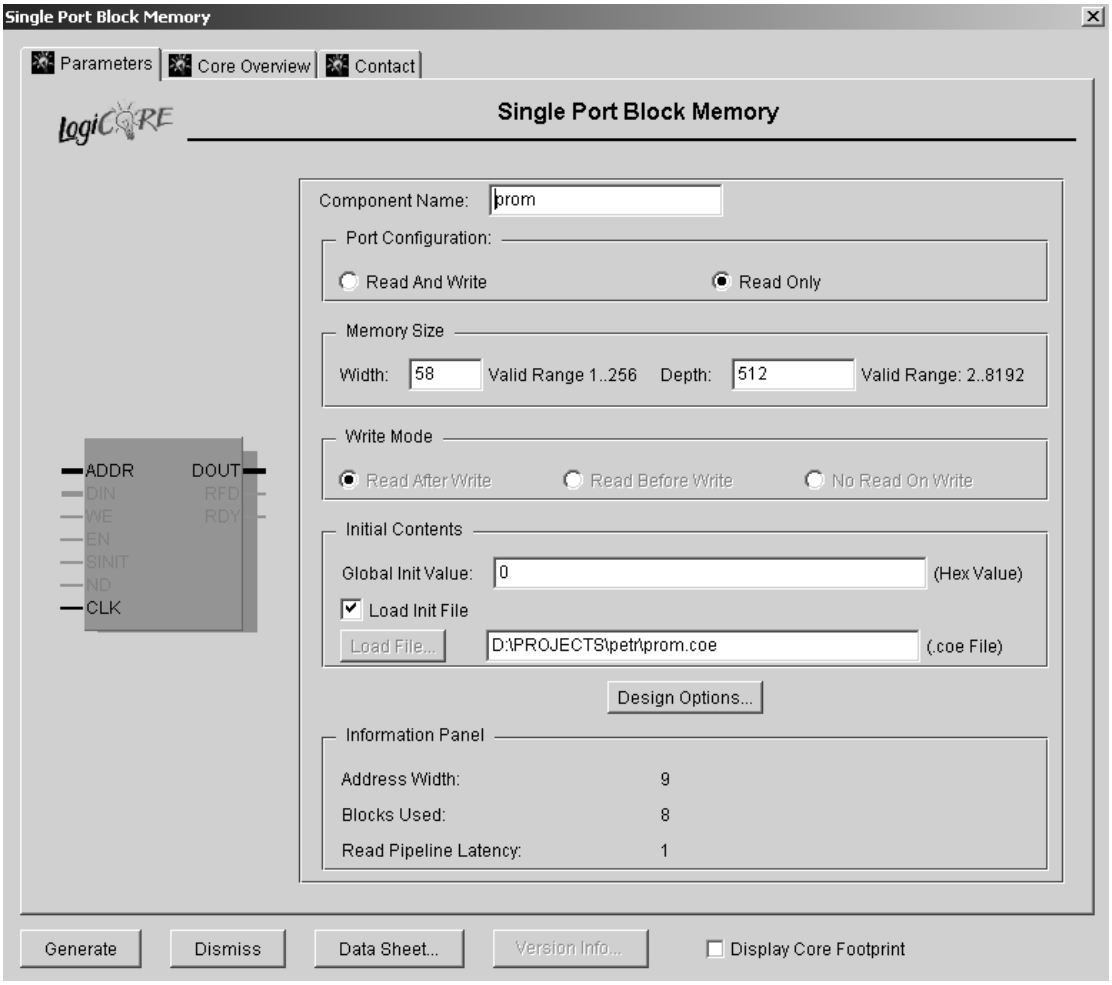

Рис.7. Внешний вид окна системы Core Generator

Заключение

Рассмотренные выше преимущества технологии проектирования цифровых устройств в элементном базисе ПЛИС и полученные сравнительные характеристики, приведенные в таблице, позволяют сделать вывод о практической ценности описанного в статье подхода.

Литература:

1.Пат. 2071111 РФ, RU C1 6 G06 F 9/00. Устройство управления / А.Ф.Кургаев, Г.Н.Дашкиев, Н.Г.Петренко. – Опубл. 27.12.96, Бюл. № 36. – 34с.

2. Кургаев А.Ф. Исследование архитектуры машины баз знаний // УсиМ. – 2000. - № 1. – С. 74-91.

3. Кургаев А.Ф., Петренко Н.Г. Синтез структуры процессора. // Кибернетика и системный анализ. – 1995. - № 4 – С.159-168.

4. Майоров С.А., Новиков Г.И. Структуры цифровых вычислительных машин. – Л.: Машиностроение, 1979. – 384 с.

5. Xilinx DataBook 2002. (http://www.xilinx.com).

6. Foundation Series 3.1i Quick Start Guide. (http://www.xilinx.com).

7. Афанасьев А.О., Кузнецова С.А. OrCAD 7.0...9.0.Проектирование электронной аппаратуры и печатных плат. Изд-во "Наука и техника", 2001 г., 464 стр.

8. VHDL'93. IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1993.

9. П.Н.Бибило. Основы языка VHDL. М.:”Солон-Р”, 2000.

10. State Editor Tutorial. Xilinx Foundation Series On-Line Help System. Xilinx Foundation F4.1i.

11. PCI Local Bus Specification Rev. 2.2, PCISIG. Available at http:// www.pcisig.com.

12. FPGA Compiler II/FPGA Express. VHDL Reference Manual. - 496 P.