происходит потеря информации.

Далее матрицы коэффициентов субполос переводится в вектор при помощи "зигзаг"-сканирования. Полученный вектор свертывается с помощью алгоритма группового кодирования (RLE). Алгоритм RLE заключается в следующем: любой последовательности повторяющихся входных символов ставится в соответствие набор из двух выходных символов: первый байт, определяющий длину входной последовательности, второй — сам входной символ.

На следующем этапе проводится Кодирование по Хаффману. В основе алгоритма кодирования Хаффмана лежит довольно простой принцип: символы заменяются кодовыми последовательностями различной длины. Чем чаще используется символ, тем короче должна быть кодовая последовательность. Кодовая таблица строится на основе статистического анализа имеющейся информации. Код Хаффмана обладает свойствами префиксных кодов и легко может быть свернут обратно в последовательность длин серий.

Сжимая файл по алгоритму Хаффмана, первое, что нужно сделать - подсчитать сколько раз встречается каждый символ. После подсчета частоты вхождения каждого символа, необходимо построить таблицу кодов и сформировать мнимую компоновку между кодами по убыванию. Для восстановления первоначального файла, нужно иметь декодирующую таблицу, так как они будут различны для разных файлов. Следовательно, необходимо сохранять таблицу вместе с файлом.

Для реализации вышеописанных алгоритмов был использован язык технических вычислений MatLab.Таким образом, была разработана модель кодера изображений на основе параунитарного банка фильтров в ряде случаев превосходящая по скорости вычислений и степени сжатия уже существующие кодеры.

Список использованных источников:

1. Петровский, А. А. Цифровые банки фильтров: анализ, синтез и применение в мультимедиа системах : учебно-методическое пособие / А.А. Петровский [и др.]. – Мнинск, БГУИР, 2006. – 82 с.
2. Парфенюк, М. Параунитарные банки фильтров на основе алгебры кватернионов: теория и применение/ Парфенюк М., Петровский А.А - Мнинск, БГУИР, 2008. – 14с.
3. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

ОПИСАНИЕ ПРОЦЕССОРНОГО ЯДРА MOTOROLA MC6800 НА ЯЗЫКЕ ОПИСАНИЯ АППАРАТУРЫ VHDL

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Винокуров В. В.

Герасимович В. Ю. – аспирант, профессор

Рассмотрен один из методов проектирования процессорного ядра Motorola 6800 с использованием языка описания аппаратуры, показаны основные проблемы на различных этапах проектирования.

Введение

Микропроцессоры представляют собой сложную структуру связанных между собой различных компонентов цифровой техники. Проектирование процессорного ядра является очень трудоемкой задачей, которая занимает колоссальное количество времени и человеческого труда. Без соответствующих САПР, этот процесс занимало бы не один год и увеличивал себестоимость в несколько раз, что в современном темпе развития техники, делает продукт полностью не конкурентоспособным. Одним из главных методов проектирования аппаратуры, является проектирования с использованием языков описания интегральных схем.

Основная цель моей работы – на практике, опробовать VHDL в качестве инструмента для проектирования микропроцессоров, ознакомиться с основными проблемами, с которыми сталкиваются начинающие системотехники. Проектируемая модель в точности повторяет архитектуру известного микропроцессора Motorola 6800. Motorola 6800 — микропроцессор, разработан и выпущен компанией Motorola в 1974 году. Это был первый микропроцессор с индексным регистром.

Основная часть

Перед началом реализации процессора на VHDL, требовалось как можно точнее изучить внутреннюю архитектуру и принципы работы ядра MC6800. В качестве основного источника информации использовалась официальная документация на микропроцессор, в которой были описаны основные режимы работы процессора, а также в общих чертах показана его структура. Вот тут и появляется первая проблема, для молодых специалистов.

Любая крупная IT фирма всегда утаивает ключевые аспекты построения своей архитектуры. Motorola не исключение. В частности, отсутствовала какая-либо информация об организации внутренних шин ядра и взаимодействии между собой основных компонентов. В таком случае проектирование происходит на основе

общего понимания организации ЭВМ.

Непосредственное проектирование VHDL модели выполнялось модульно. В структуру MC6800 входят: АЛУ, 2 8-разрядых аккумулятора, 2 16-разрядных индексных регистра, программный счетчик, регистр команд и устройство управления. Все модули, кроме УУ, описывались по стандартным методам и использовались шаблоны, встроенные в Xilinx ISE.

В конечном счете архитектура MC6800 следующая. Общая шина данных представляет собой 3 8-разрядные шины. Шина A и B связывает выходы регистров с информационными входами АЛУ соответственно, а шина C – выход АЛУ с входами регистров. Передача данных между шинами A(B) и C осуществляется через АЛУ. Так же шины B и C выходят на двунаправленный буфер данных.

Самый сложный компонент – устройство управления. Устройство представляет из себя конечный автомат, состояниями которого является этапы выполнения команд процессора. Вторая проблема для начинавших – разобраться в методах кодирования команд процессора. Как правило, каждая команда – набор элементарных действий, которые повторяются в разной последовательности и в разном количестве от команды к команде. Эти элементарные действия и их последовательность кодируются в команду. Понимание принципов кодирования помогает описать устройство управление очень лаконично, не расписывая каждую команду в отдельности. Реализация устройства управления MC6800 и была основана на конечном автомате, который реализовывал разные стадии выполнения программы. Конечный автомат определяет стадию выполнения команды и в заданные моменты времени подает сигналы управления на все модули (пример: записи данных в аккумулятор, нужную команду АЛУ). По окончанию выполнения команды - переходит на состояние выборки следующей команды.

Только к концу проектирования, были выявлены несколько методов оптимизации кода на основе кодирования команд, описанной выше.

В свете выше сказанного, хочется отметить что проектирование любого цифрового устройства является решением комплексной задачи аналитики и проектирования с применяем лучших решений в данной области. А применение разнообразных САПР и HDL позволяет оптимизировать работу проектирования, обеспечивает ее гибкость. Данная работа является прямым применением всех наших знаний, полученных за время обучения в нашем ВУЗе.

Список использованных источников:

1. Perry D.L. VHDL: Programming by Example. Fourth Edition. McCraw-Hill, 2002. – 476 p.
2. Бибило П.Н. Основы языка VHDL. Второе издание. — М.: Солон-Р, 2002. — 224 с

РАЗРАБОТКА ИГРОВОГО ПРОЕКТА НА UNREALENGINE 4

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Корсаков А. А.

Разработка игр является комплексной и весьма сложной задачей. Основная сложность заключается в том, что игровые проекты включают в себя множество областей: графика, анимация, обработка огромных объёмов данных в реальном времени, искусственный интеллект, геймдизайн, социальное программирование, повествование, взаимодействие с игроками и так далее. Над проектом, обычно, трудятся команды из огромного числа специалистов в совершенно разных областях.

Можно утверждать, что на данный момент геймдев как никогда близок к рядовым пользователям. С развитием индустрии начала появляться профильная литература, курсы и специальности в учебных заведениях, а пользователи всё чаще пытаются делать свои проекты, тем самым продвигая такое направление, как Инди-разработка.

Я и моя команда не стала исключением. Нам пришлось пройти долгий путь, впитывая в себя огромные объёмы информации, перенимая опыт успешных коллег из-за рубежа и, попутно, экспериментировать и выносить для себя собственные гипотезы и умозаключения.

Для своей работы мы выбрали игровой движок UnrealEngine 4, который, если можно так выразиться, является своеобразным САПРом для создание игр. Его преимуществами является бесплатная модель распространения и, что самое важное, открытый код, который позволяет переписывать любые аспекты системы под свои нужды. UE4 базируется на C++, что даёт ему ощутимый выигрыш в скорости работы относительно конкурентов.

Основными проблемами при разработке игровых проектов являются обеспечение наибольшего быстродействия, организация архитектуры таким образом, чтобы возможно было в кратчайшие сроки вносить изменения и расширять систему, необходимость создания большого количества сопутствующих продуктов, призванных повысить качество и скорость разработки.

Развитие игровой индустрии тесно связано с общим прогрессом в области IT. В частности, на данный