Создание специализированных микропрограммных автоматов на базе специализированных ИС.

Семенов Н.Н., Каршенбойм И.Г.

Введение.

При разработке и создании все более сложных и объемных цифровых схем появляется проблема повышения эффективности работы разработчиков и гибкости проекта, а основным требуемым критерием эффективности разработки является время, затраченное на разработку, или, иначе говоря, «time to market» - время от начала реализации проекта до выхода готового продукта на рынок.

Другая (и не менее важная) проблема разработчиков – это выбор наиболее подходящей архитектуры проекта, которая обеспечивала бы наивысшую надежность работы микросхемы, приемлемую стоимость и легкость модификации проекта в будущем.

В настоящей статье рассматривается возможность построения электронных цифровых схем в виде микропрограммных автоматов (МА), оптимизация структуры таких автоматов и адаптация системы команд для заданных условий. Использование МА позволяет решить большинство вышеописанных проблем, создавая высокопроизводительные процессоры с ограниченной функциональностью внутри микросхемы.

Так как МА имеют стандартную архитектуру, применяя которую с небольшими изменениями для разрабатываемого проекта, можно существенно облегчить проектирование специализированных ИС и в несколько раз уменьшить время от начала разработки до выхода конечного продукта. Весь процесс разработки в таком случае заключает в себя следующие этапы:

1. Описание (на уровне блоков) алгоритма.
2. Создание необходимой системы команд.
3. Подключение к микропрограммному автомату необходимых библиотек.
4. Написание микропрограммы.
5. Отладка микропрограммы.

Объединение в одном устройстве сразу нескольких МА позволяет построить конвейерную обработку большого объема данных, распределенные вычисления, нейронную сеть и, что самое важное, реализовать все это внутри одной микросхемы, создавая так называемые System-On-Chip устройства.

1. Современные средства автоматизации проектирования цифровых схем и проблемы их использования.

Стремясь к достижению высоких технических характеристик и потребительских качеств своей продукции, разработчики электронных устройств используют специализированные ИС (СПИС). Их применение обеспечивает следующие преимущества :

- Сокращение габаритов устройства.
- Повышение технических характеристик (повышение быстродействия и сокращение потребляемой мощности)
- Повышение надежности (так как вероятность поломки прямо пропорциональна количеству ИС, использование СПИС значительно его сокращает).
- Обеспечение защиты разработки (скопировать устройство, содержащее СПИС практически невозможно, что позволяет обеспечить авторские права разработчика).
- Повышение гибкости модификации (модификация СПИС не требует переработки остальных узлов, переразводки печатных плат и т.д.).

2. Построение блок-схемы микропрограммного автомата.

Для построения устройств ввода и обработки данных обычно используется архитектура, показанная на рис. 1.

Рисунок 1

Центральный процессор через общую шину оперирует устройством ввода-вывода, обрабатывает полученные данные и отправляет их дальше. Но для управления реальными устройствами ввода-вывода необходимо производить большое число операций, не требующих сложных вычислений, но жестко ограниченных по времени. Получается, что центральный процессор вместо того, чтобы заниматься реальной обработкой полученных данных должен обеспечивать своевременное обслуживание устройства ввода-вывода. Нагрузка на общую шину тоже оказывается значительной из-за большого количества мелких операций между центральным процессором и устройством ввода-вывода. Упростить эту структуру можно разбив процесс обработки на несколько частей, как это показано на рис. 2.

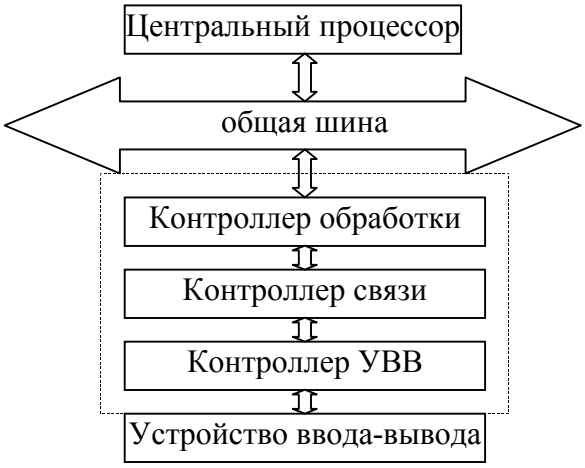

Рисунок 2

В этом случае «Контроллер УВВ» обеспечивает нормальное функционирование устройства ввода-вывода, обрабатывая критичные ко времени обработки запросы, «Контроллер связи» готовит данные к передаче и проверяет их целостность на приеме, «Контроллер обработки» занимается предварительной обработкой данных, чтобы разгрузить центральный процессор и общую шину от потока ненужной информации.

Реализовать такую структуру можно разными способами: добавив в устройство один или несколько микропроцессоров, но это может быть неудобно по конструктивным, ценовым параметрам или из соображений быстродействия, или создать все эти контроллеры в виде нескольких МА внутри специализированной ИС. Последний вариант является предпочтительным, так как в центре большинства современных устройств лежит большая специализированная ИС, в функции которой входит коммутация сигналов на плате и распределение управляющих сигналов. Внутри этой ИС обычно можно расположить все необходимые МА.

Но это не единственный пример использования МА. Их можно объединять последовательно, параллельно, организовывать из них целые сети, как это показано на рис.3, все зависит только от требований проекта.

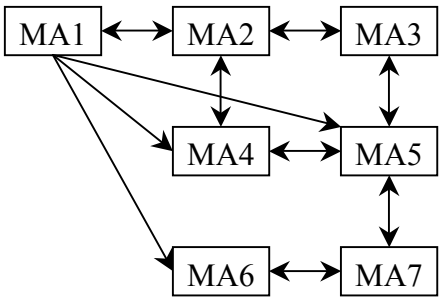

Рисунок 3

3. Описание этапов построения МА.

Выбор системы команд

Минимальный набор, позволяющий писать микропрограммы с ветвлениями, состоит из следующих команд :

NOP	Нет операции
MOV Reg, Reg	Запись содержимого одного регистра в другой
MOV Reg, [Mem]	Запись содержимого регистра в память
MOV [Mem], Reg	Запись из памяти в регистр
INC Reg	Увеличение значения регистра на 1
DEC Reg	Уменьшение значения регистра на 1
JMP Addr	Переход по абсолютному адресу
IF [Reg] JMP Addr	Переход по абсолютному адресу по условию [Reg<>0]
IF Flag JMP Addr	Переход по абсолютному адресу по состоянию флага
CALL Addr	Вызов подпрограммы (с записью указателя в стек)
RET	Возврат из подпрограммы (по содержимому стека)

Его можно дополнять и расширять. Арифметические операции сознательно убраны из списка, так как в большинстве МА они не нужны.

Построение ядра МА

Хотя для каждой конкретной реализации МА конструкция будет отличаться, структура останется неизменной, такой, как показано на рис. 4.

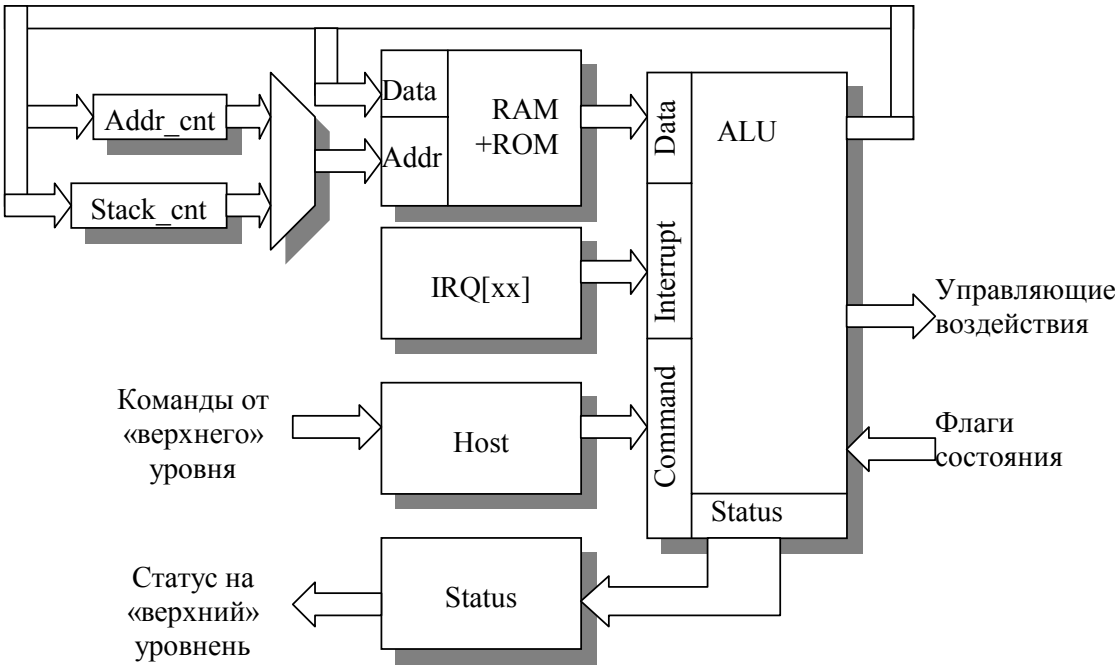

Рисунок 4

Блоки «Host» и «Status» обеспечивают интерфейс с верхним уровнем обработки. Блок «IRQ[xx]» предоставляет возможность организации работы по прерываниям, что совершенно необходимо для работы в реальном времени.
Блок «RAM+ROM» служит идеализированной моделью памяти. При практической реализации память команд, память данных и стек удобно разнести. Это оказывается возможно при небольшом размере памяти программ и стека, так как память программ хранится в виде констант, а размер стека определяется алгоритмом работы и может быть расположен на регистрах.
«Addr_cnt» - это указатель на следующую команду, которую нужно исполнить, или адрес данных, которые нужно считать или записать.
«Stack_cnt» - механизм работы со стеком.
«ALU» - ключевой механизм МА, состоящий из дешифратора команд, набора регистров и производящего определенные операции над этими регистрами. В простейшем случае ALU не требует контроллера прерываний и умеет только пересылать данные с места на место и сравнивать их с заданным условием. Но можно построить и такой ALU, который умеет производить сложные операции, такие как сложение, вычитание, умножение, деление, нахождение синуса и так далее, причем в качестве математического сопроцессора может выступать другой МА. Поэтому для каждой конкретной задачи необходимо определить, какие функции ALU являются необходимыми, и подключить к проекту только библиотеки с необходимыми функциями.

Написание микропрограммы

Для удобства написания программ для МА можно создать собственный ассемблер, ставящий в соответствие удобную для восприятия мнемонику с реальными командами МА. Сейчас каждый программист в программе обучения в институте проходит курс создания компиляторов, и написание ассемблера для любого процессора не является трудной задачей.
Так как МА выполняет ограниченный набор функций, размер микропрограмм обычно не превышает нескольких десятков-сотен команд. Вот пример части такой программы:

Адрес	**Код**	**Мнемоника**	**Комментарий**
0000	000000	MOV [0], X2	В X2 записали адрес из ячейки памяти
0001	110240	MOV 240, X1	В X1 записали 240 – длину пакета
0002	220003	MOV [X2], X3	Считали ячейку памяти
0003	430327	MOV X3, DMA	Записали в регистр DMA
0004	090002	INC X2	Увеличили адрес читаемой памяти
0005	080001	DEC X1	Уменьшили счетчик циклов
0006	710002	IF [X1] JMP 0002	Проверка условия выхода из цикла
0007	050000	RET	Возврат из прерывания

Отладка микропрограммы

Для отладки микропрограммы есть много различных путей, используя комбинацию из них, можно достичь быстрого положительного результата.
Во-первых, используемые для отладки специализированные ИС (FPGA) обычно перепрограммируемые, то есть в случае обнаружения ошибки всегда есть возможность исправить эту ошибку без переделки всего устройства.
Во-вторых, отладка микропрограммы происходит по частям, что позволяет легко локализовать ошибку и исправить ее.
В третьих, использование САПР, такого как Max+Plus, позволяет создавать тестовые входы и выходы МА, имитировать реальную работу всего устройства в симуляторе.

В процессе отладки можно использовать FPGA с большим количеством ячеек, чем нужно для функционирования проекта. Это позволит выделять отладочные ресурсы. К таким ресурсам можно отнести дополнительные счетчики, тестовые сообщения и эмуляторы. Эти ресурсы можно перераспределять для отладки отдельных частей микропрограммы.

Также можно воспользоваться сигнатурным анализом, то есть индикацией на светодиоде или в доступном регистре результата работы сложного алгоритма. Это позволяет сделать качественную оценку работоспособности всего алгоритма или его части.

Как результат, отладка микропрограмм в МА оказывается значительно проще переписывания всего конечного автомата в случае обнаружения в нем ошибки или смены алгоритма работы.

4. Сравнительная характеристика

Чтобы оценить преимущества использования МА перед другими способами, рассмотрим конкретный пример. В проекте создания шлюза IP-телефонии было необходимо с помощью центрального процессора i80486DX4-100 обрабатывать поток данных с сетевого интерфейса 10Мбит/с. Оказалось, что даже после оптимизации алгоритма процессор не справляется с обработкой 10-Мбитного потока данных. Пришлось использовать другой процессор – Intel Pentium-233. Это резко увеличило стоимость конечного продукта. Использование же конвейерной обработки позволяет настолько разгрузить центральный процессор, что даже у I486DX остаются ресурсы для работы операционной системы, обработки информации и так далее.

Для сравнения МА с конечным автоматом, который является альтернативой МА, можно привести следующее:

- Чтобы в новом проекте реализовать заданную последовательность действий можно либо каждый раз заново создавать конечный автомат, либо взять уже готовый МА, адаптировать его к заданным условиям, и, написав небольшую программу, запустить. Причем написание программы для МА намного проще написания конечного автомата на языках AHDL, VHDL и так далее.
- Чтобы исправить ошибку в конечном автомате, необходимо пересобирать весь проект, а в МА можно только перезаписать программу.
- Чтобы изменить алгоритм работы конечного автомата, необходимо его полностью переписывать, что требует много времени и сил, в МА достаточно изменить микропрограмму.
- Конечный автомат должен иметь ограниченное количество состояний, в то время как МА по количеству состояний не ограничен.

5. Обзор производимых МА в мире

Процесс создания и эксплуатации микропрограммных автоматов на FPGA идет в мире уже более 5 лет. Обычно это коммерческие проекты, поэтому о них мы можем судить только по рекламе.

Например, фирма “Hammer Cores” выпустила мегафункцию для FPGA фирмы Altera, которая включает CISC-процессор, работающий на 20 МГц, имеющий до 26 регистров с разрядностью 8, 16 и 24 бита и занимающий 367 ячеек при 7 регистрах, 3 прерываниях и разрядности 8 бит.

Более сложный процессор на FPGA фирмы Altera представила фирма “CAST” – это полный аналог 8-битного микроконтроллера C8051, работающего на 8 МГц и занимающего 2398 ячеек.

На FPGA фирмы Xilinx тоже ведутся работы по созданию МА, но они в основном направлены на построение универсального инструмента для аэрокосмических нужд и объединяются в глобальный проект “Adaptive Instrument Module”.

Есть и другие фирмы-производители функций микропрограммных автоматов, максимальная частота работы некоторых из них достигает 48 МГц. Это и копии

популярных микроконтроллеров и микропроцессоров, главное удобство которых в доступности средств разработки программного обеспечения, и совершенно новые МА, позволяющие в виде параметров для компиляции менять свою структуру под требования заказчика. Фирмы-производители снабжают своих клиентов большим набором отладочных средств и средств разработки ПО, поэтому проблем с использованием таких МА обычно не бывает.

Литература

1. **Баранов В.П.** Синтез микропрограммных автоматов. М.: Нолидж, 1997. 376 с.
2. **Долинский М.**С., Харрасов А.А. Средства разработки цифровых устройств методом синтеза микропрограммных автоматов. Электроника. 11-12, 1998, сс 19-23.
3. **Гультяев А.К.** Имитационное моделирование в среде Windows.— СПб.: КОРОНА принт, 1999.— 288 с.
4. **Баранов С.И.** Синтез микропрограммных автоматов. Л. : Энергия, 1979. 239 с.
5. **Altera**, Processor & Peripheral Megafunction, **AMPP**, http://www.altera.com/
6. **Xilinx**, AIM, http://www.xilinx.com/