Вопросы к зачету по дисциплине «Программируемые логические интегральные схемы». ФАВТ, 4-й курс. 2010-11 уч.г.

1. Программные и аппаратные способы реализации вычислительных алгоритмов.

2. Элементная база цифровой электронной техники. Классификация цифровых интегральных микросхем. ПЛИС в иерархии цифровых ИМС.

3, 4, 5. Концепция программирования структуры интегральных схем. Классификация ПЛИС. Области применения ПЛИС. Программируемые постоянные запоминающие устройства как ПЛИС. Классификация ПЛИС по виду памяти конфигурации.

6, 7. CPLD: общая структура, функциональные блоки. Блоки ввода/вывода, программируемая матрица соединений.

8. FPGA: общая структура, виды функциональных блоков.

~~9. FPGA: матрица межсоединений, блоки ввода/вывода, встроенные блоки памяти.~~

10. Системы синхронизации и тактирования. PLL, DLL, CDR.

11. Системы-на-кристалле: основные понятия, классификация. Требования к ПЛИС, используемым для построения систем на кристалле. Hard- и soft-ядра. Концепция применения IP-ядер.

12. Оценка логической емкости ПЛИС.

13. Оценка быстродействия ПЛИС.

14. Проектирование радиоэлектронной аппаратуры: цель, задачи, концепция, методология, этапы.

15. Области (функциональная, структурная, геометрическая) и уровни представления ПЛИС. Проектирование как процесс последовательного спуска по уровням областей представления (диаграмма Гайского-Кана).

16. Маршрут (поток) проектирования цифровых устройств на ПЛИС. - см. 3,4,5.

17. Иерархия языков проектирования аппаратуры.

18. Интерфейс JTAG: назначение, общие принципы граничного сканирования.

~~19. Интерфейс JTAG: структура цепочки, BSC и устройства управления.~~

~~20. Интерфейс JTAG: основные команды, диаграмма состояний TAP-контроллера.~~

21. Системы автоматизированного проектирования цифровых устройств на ПЛИС. Состав и назначение программных компонент. Особенности САПР Quartus II фирмы Altera.

22. Микропроцессоры. Классификация процессоров. Понятия архитектуры и структуры. Варианты архитектур. Конвейерное выполнение команд.

23. Микропроцессоры. Регистровая модель. Способы адресации операндов.

24. Микропроцессоры. Группы команд. Примеры команд основных групп.

25. Особенности архитектуры микроконтроллеров. Популярные семейства микроконтроллеров.

1. Программные и аппаратные способы реализации вычислительных алгоритмов.

Программный. Пусть нам надо реализовать логическую функцию $y = x_1 \cdot x_2 + x_3 \cdot x_4$. Будем считать, что *x* вводятся через порты, и *y* должен быть выведен через порт.

IN port A; A←(port A) — содержимое порта A пересылается в аккумулятор.

MOVE B, A; B←(A) — содержимое аккумулятора заносим в регистр B.

IN port B; A←(port B)

AND B, A; A←(A)·(B)

MOV C, A; C←(A)

IN port C; A←(port C)

MOVE B, A; B←(A)

IN port D; A←(port D)

AND B, A; A←(A)·(B)

OR A, C; A←(A)+(C)

OUT port D; port D ←(A)

2. Схемотехническая (аппаратная): два элемента И и один элемент ИЛИ.

Преимущества и недостатки.

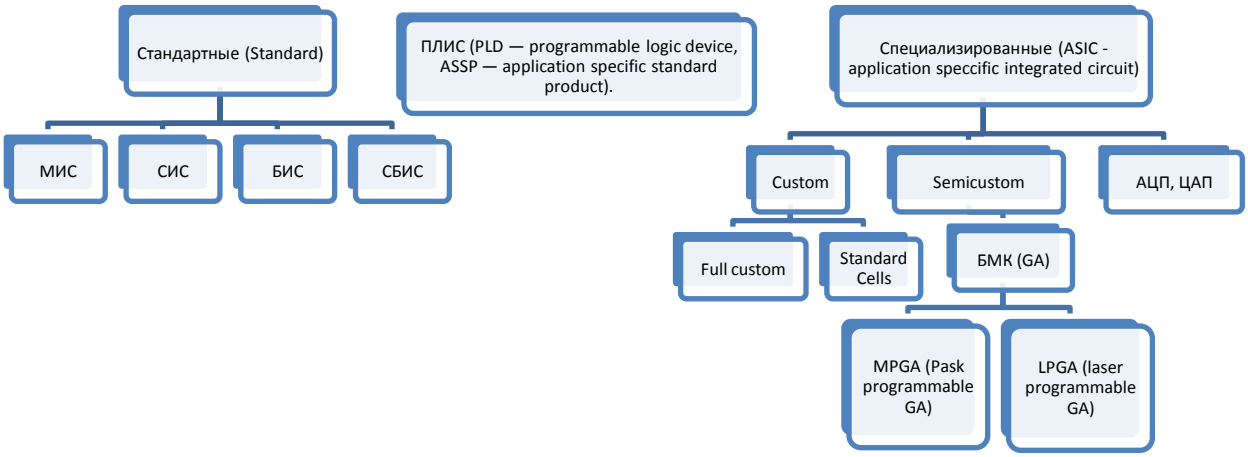
- Программа может выполняться только последовательно. Это минус программной реализации: меньшее быстродействие. В аппаратной же видим параллельное выполнение вычислительных операций.

- В случае простых задач стоимость меньше у аппаратной реализации. Для программной реализации всегда нужны одни и те же аппаратные ресурсы (микропроцессор, память, тактовый генератор и т. д.), что в ряде случаев экономически неэффективно.

- Высокая гибкость программной реализации: не надо ничего перепаивать.

Мы видим, что и аппаратная, и программная реализация имеют свои преимущества и недостатки.

2. Элементная база цифровой электронной техники. Классификация цифровых интегральных микросхем. ПЛИС в иерархии цифровых ИМС.



МИС — малой степени интеграции.

БМК — базовые матричные кристаллы, GA — gate array.

Самая дорогостоящая микросхема — полностью заказная. Эти микросхемы используются только там, где есть массовое производство. На стандартных ячейках они чуть дешевле — там используются некие заготовки. Полузаказные: БМК — полупроводниковые кристаллы, покрытые элементами, например, 2И-НЕ. Они изначально либо никак не соединены, либо соединено всё, и при проектировании надо либо сделать соединения, либо перерезать ненужные. Стоимость подготовительного этапа существенно ниже, поэтому полузаказные можно изготовлять при меньших тиражах. Особенность: техническое задание подготавливает заказчик, а изготовление микросхем производится на заводе-изготовителе.

В чём радикальное отличие ПЛИС от БМК? И там и там есть неспециализированная болванка, но БМК программируется на заводе, а ПЛИС может быть запрограммирован пользователем. В этом большое преимущество и удобство. Невозможно организовать большие тиражи, но зато это удобно. ПЛИС можно многократно перепрограммировать. ASSP: специализированная для конкретного приложения, но в то же время стандартный продукт (изначально неспециализированные).

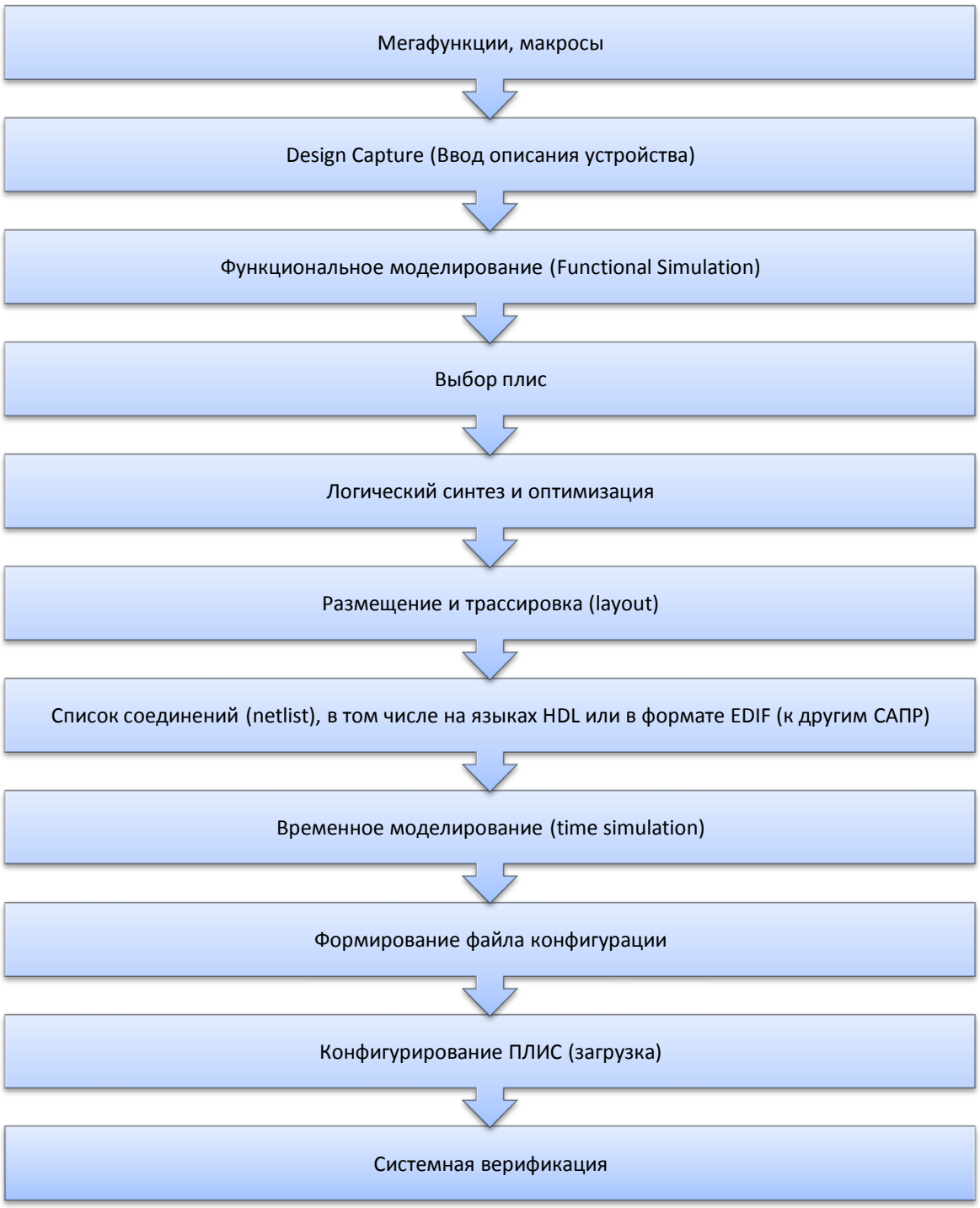
Коммерческое применение получили в 1980-х гг., в России — в 1990-х. Отечественные ПЛИС широкого распространения получить не успели.

3, 4, 5. Концепция программирования структуры интегральных схем. Классификация ПЛИС. Области применения ПЛИС. Программируемые постоянные запоминающие устройства как ПЛИС. Классификация ПЛИС по виду памяти конфигурации.

Рабочий поток (маршрут) проектирования цифровых устройств на ПЛИС

Проектирование ведётся на компьютере с использованием специализированного программного обеспечения.

Рабочий поток — design workflow:



Способы описания устройства:

- табличное описание;
- графическое описание (электрическая принципиальная схема);
- временные диаграммы;
- текстовое (например, на языке VHDL);
- аналитическое (логические уравнения);
- с помощью графов переходов;
- ...

Обычно программная среда поддерживает несколько способов описания, при этом в разработке одного проекта может сочетаться несколькими способами, например, на одном уровне иерархии графический способ, на другом — текстовый и т. п.

Синтез — перевод проекта в базис выбранной микросхемы.

Размещение и трассировка: программный пакет прикидывает, в каком месте кристалла что будет располагаться и как это будет соединено. Можно открыть вид на кристалл и всё посмотреть. Можно вручную подвигать.

Эти два этапа — синтез и размещение с трассировкой — называют компиляцией (compile).

HDL — Hardware design language — язык разработки аппаратуры.

Эти язык универсальные, их понимают программные среды разных производителей. Все САПРы понимают и HDL, и EDIF.

Временное моделирование учитывает задержки распространения сигналов в выбранной ПЛИС.

System verification: может оказаться, что на самом деле всё работает не так. Системная верификация — это не просто снятие осциллограмм с ножек. Можно данные даже с внутренних точек ПЛИС вывести обратно в компьютер. Не так может получиться в предельных случаях: по частоте (когда тактовая частота приближается к максимальной для данной ПЛИС) и в случае приближения к заполнению ПЛИС (когда ресурсы кристаллов исчерпываются).

Области применения ПЛИС

1. Быстродействующие устройства обработки сигналов, работающие в реальном времени. То есть там, где не справляется микропроцессор. Например, обработка видеопотоков высокой чёткости в реальном времени.

2. Аппаратура, выпускаемая малыми сериями (например, профессиональная ТВ аппаратура). Там, где невыгодно применять заказные специализированные микросхемы.

3. Прототипирование. Допустим, заказали специализированную микросхему, а в проекте ошибка. Придётся микросхемы выбросить. Хорошо бы сначала проект обкатать. ПЛИС — платформа для тестирования. И после этого уже делается заказ на изготовление специальной микросхемы.

4. Построение реконфигурируемых систем. ПЛИС можно перепрограммировать. Сначала внутри ПЛИС можно реализовать одно устройство, потом — другое.

Фирмы Xilings, Altera и Actel производят как ПЛИС, так и средства разработки проектов. Altera не такая крупная, но в России больше распространена, так как изначально софт у Alterы был

бесплатный. Actel производят ПЛИС специального назначения, обладающие особенными характеристиками.

BGA-корпус: снизу он покрыт полушариями с припоем по всей площади корпуса. Шарики устанавливаются на контактную площадку, и ПЛИС прижигается. Шаг шариковых выводов составляет доли миллиметра. Высокие требования к монтажу.

Много выводов — много сигналов. Выводы общего назначения могут быть запрограммированы как входы, как выходы, так и двунаправленные. Зачастую приходится обрабатывать параллельные потоки данных, поэтому много ножек — это хорошо.

Одно из направлений развития ПЛИС — увеличение количества выводов.

Классификация ПЛИС

1. По уровню интеграции и архитектурным признакам
   1.1. Простые. SPLD — simple programmable logic device. Пригодны для реализации только комбинационных устройств.
      1.1.1. программируемые логические матрицы, ПЛМ. PLA — programmable logic array.
      1.1.2. программируемая матричная логика, ПМЛ. PAL — programmable array logic. Не имеют элементов памяти.
   1.2. Сложные. Complex.
      1.2.1. CPLD — complex PLD.
      1.2.2. FPGA — field programmable gate array. Вентильные матрицы, программируемые пользователем.
      1.2.3. комбинированные архитектуры (сочетают черты двух предыдущих).
   1.3. Системы на кристалле. SoC — system on chip. На одном кристалле располагается несколько разнородных функциональных устройств. Например, вычислитель ДКП и вычислитель вектора движения. Или, например, два ядра процессора.
2. По типу памяти конфигурации (по кратности программирования).

Задача программирования ПЛИС заключается в соединении функциональных блоков в нужной последовательности. Как? Первая ситуация: изначально все программируемые ключи разомкнуты, и при программировании замыкаются нужные — установление электрических связей. Противоположная ситуация: изначально все ключи замкнуты, и в процессе программирования некоторые ключи размыкаются. Эта логика справедлива для всех ПЛИС, разница лишь в реализации.

   2.1. Однократно программируемые ПЛИС. Как правило, сравнительно невысокой степени интеграции (сотни — тысячи логических вентилей (2И-НЕ)). Радиационно устойчивые. Устойчивы к перепаду температур. Вообще, их надёжность сравнительно высока. Используются в космической технике, авиации. Они дорогие. В специальных корпусах.
      2.1.1. С плавкими перемычками. Fuse. Там с помощью специального программатора в нужных местах создаются повышенные токи, которые вызывают переплавление соединительных линий.
      2.1.2. С диэлектрическими электрически пробиваемыми перемычками. Antifuse. Программатор создаёт повышенные напряжения в нужных местах, пробивающие участки диэлектрика в нужных местах.

2.1.3. На МОП-транзиторах с плавающими затворами без стирания зарядов. EPROM-OTP — electrically programmable read-only memory — one time programmable.

2.2. Репрограммируемые с существенно ограниченным числом перезаписи (десятки — сотни раз).

2.2.1. ПЛИС с памятью конфигурации с ультрафиолетовым стиранием. EPROM — electrically programmable ROM. Память конфигурации — память, где хранятся конфигурационные данные. Программируется электрически специальным программатором. Здесь тоже МОП-транзисторы с плавающими затворами, на которые инжектируются заряды, только стереть память здесь можно через специальное прозрачное окошко специальной кварцевой лампой. Дорогие и не очень удобные.

2.3. Репрограммируемые с ограниченным числом перезаписи (до сотен тысяч раз).

2.3.1. С памятью конфигурации типа EEPROM — electrically erasable PROM — память с электрическим, а не ультрафиолетовым стиранием. В программаторе сначала стирание повышенными токами/напряжениями, а затем записывается новая информация.

2.3.2. Флэш-память конфигурации. Отличие для пользователя от EEPROM: у flash нет произвольного доступа к ячейкам, перезапись только блоками.

2.4. Оперативно репрограммируемые с неограниченным числом перезаписей. SRAM-based — static random access memory. Статическая ОЗУ. Строится на базе триггеров.

Триггер управляет ключом, построенном на полевом транзисторе. Если в триггере 0, транзистор закрыт, соединения между функциональными блоками ФБ отсутствует. В отличие от всех предыдущих ПЛИС, память которых основана на ПЗУ, здесь ОЗУ, после отключения питания конфигурационные данные будут потеряны. Следовательно, чтобы устройство заработало после включения питания, где-то должно находиться некое ПЗУ с конфигурационными данными. По этому признаку разделяют их на два класса:

2.4.1. Оперативно репрограммируемые перезагрузкой.

Рядом с ПЛИС некое ПЗУ с конфигурационными данными, по определённому интерфейсу перекачивающиеся в ПЛИС. ПЛИС и память существуют раздельно. Либо в разных микросхемах, либо в разных областях кристалла. На пересылку данных требуется время. Несоизмеримо с длительностью такта.

2.4.2. Динамически репрограммируемые ПЛИС.

Прямо на ПЛИС рядом со статическим триггером ставится ПЗУ для хранения одного бита. Такая конфигурация уже быстрая, ведь загрузка данных происходит одновременно. Здесь возможно динамическое изменение конфигурации прямо в процессе работы устройств. Такие системы строятся для сложных многоэтапных задач. Для каждого этапа можно реконфигурировать ПЛИС по-своему.

3. По зависимости задержек распространения сигналов от путей их прохождения по кристаллу.

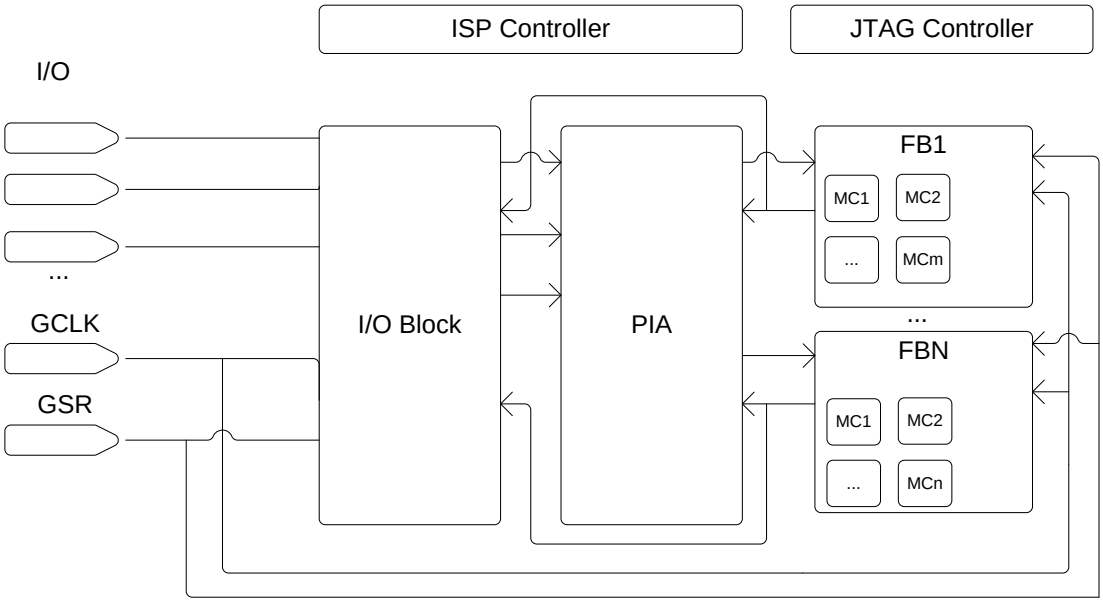
3.1. Задержки не зависят от путей распространения сигналов. То есть, из какой точки и в какую точку сигнал бы ни шёл, задержка будет одинакова. А значит, задержки предсказуемы ещё на этапе проектирования. С задержками распространения приходится считаться в случаях появления различных паразитных ложных импульсов

(иголки), которые могут попасть на счётчик и вызвать его срабатывание. Особенно приходится учитывать задержки распространения при проектировании высокоскоростных схем (при приближении к сотне МГц). Тут проблемы даже не собственно в задержках, а в их различиях. Так вот в таких ПЛИС все задержки выровнены, и о выравнивании на этапе проектирования можно не беспокоиться.

3.2. Задержки зависят от путей распространения сигналов. Тут при проектировании приходится вручную выравнивать задержки распространения. Придётся рассмотреть топологию кристалла и вручную подвигать расположение элементов. Впрочем, в наше время эти задачи решает система автоматизированного проектирования САП.

6, 7. CPLD: общая структура, функциональные блоки. Блоки ввода/вывода, программируемая матрица соединений.

Complex PLD.



PIA — Programmable Interconnect Array — глобальная программируемая матрица межсоединений. Каналы, пересекающие весь кристалл.

FB — Functional Block. Выполняет логические операции с сигналом.

MC — Macrocell.

GCLK — Global clock. Тактовый сигнал важен, чтобы все устройства срабатывали синхронно.

GSR — Global set/reset — внутри ПЛИС есть триггеры, устанавливающиеся и сбрасывающиеся.

ISP — in-system programmability — без извлечения платы.

JTAG — интерфейс отладки и тестирования.

Какие цифровые устройства можно реализовать на CPLD? Комбинационные. Последовательностные — сложно и неэффективно, обратную связь надо вести самому, а микросхема быстро закончится, поскольку одна макроячейка — это всего один бит.

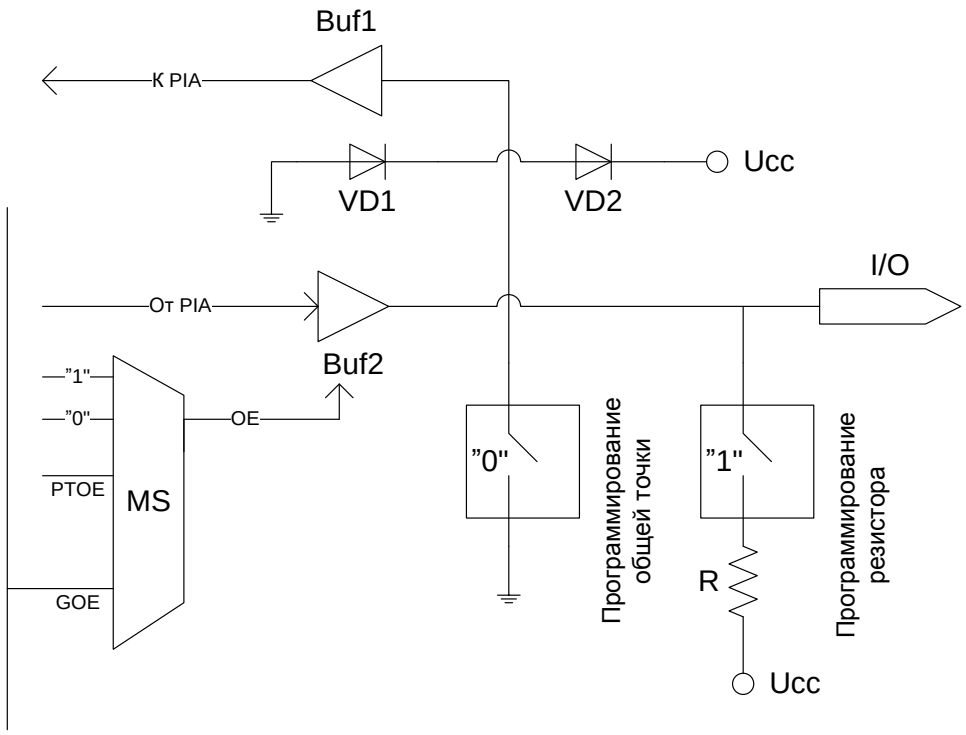
У них есть глобальные трассировочные каналы по всему кристаллу. Конфигурирование заключается в настройке мультиплексоров.

Примеры ПЛИС с архитектурой CPLD

Altera: Max 3000, Max 5000, Max 7000,Max 9000.

Xilings: XC 7000, XC 9500.

I/O Block



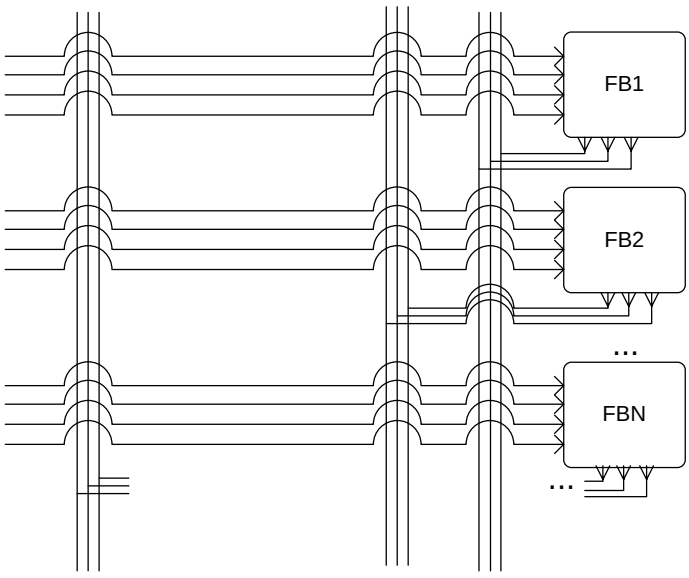
OE — разрешение выхода (дискретное изменение коэффициента передачи буфера Buf2).

PTOE — product term output enable — сигнал отпирания/запирания этого буфера, формируемый в ПЛИС.

GOE — global output enable. Влияет на все ножки МС.

VD1, VD2 — диоды защитные, ограничивают диапазон входного сигнала до 0…Ucc.

PIA



Матрица И позволяет получить только логические произведения. А логические функции, как правило, состоят не из одного логического произведения. Если их записать в форме СДНФ (совершенной дизъюнктивная нормальная форма) — запись в виде суммы произведений. Поэтому макроячейка должна иметь элемент ИЛИ. Там логическая функция реализована, остальное — дополнительное. Если логическая функция очень сложная, и ресурсов макроячейки не хватает, чтобы её реализовать, часть этой функции можно реализовать на предыдущей макроячейке и подать на эту, либо в этой часть реализовать и подать на следующую.

Исключающее ИЛИ представляет собой здесь управляемый инвертор. Ведь при выполнении этой операции с нулём, будет сохраняться поданная переменная, а с нулём — инвертированная.

Выход макроячейки можно снять с триггера, а можно в обход его. Существует комбинационный выход (в обход триггера) и регистровый выход (сигнал снимается с триггера). Триггер — одноразрядный регистр памяти.

8. FPGA: общая структура, виды функциональных блоков.

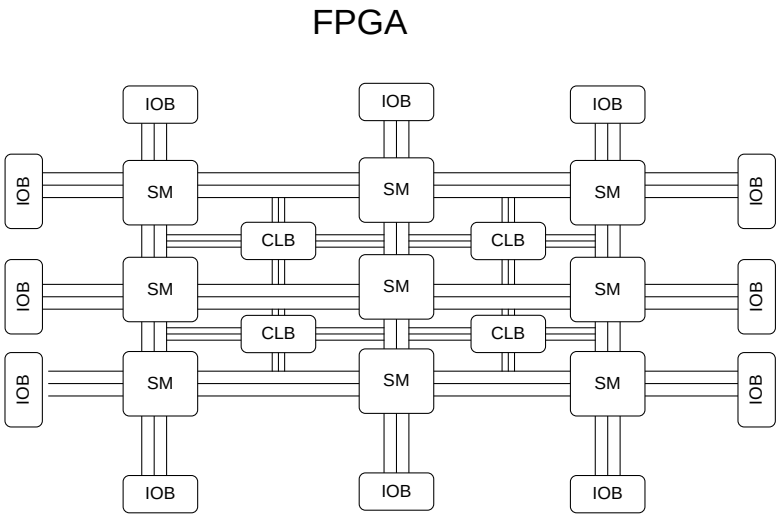
FPGA — Field Programmable Gate Array. ППВМ — программируемые пользователем вентильные матрицы.

Это существенно более распространённый класс ПЛИС. Основная доля промышленно выпускаемых.

Прародителем FPGA были базовые матричные кристаллы, БМК, или вентильные матрицы (gate array).

Отличие FPGA в том, что они программируются непосредственно разработчиком, а не на заводе. Но тем не менее они во многом сохранили черты БМК.

Зарисуем общую структуру FPGA (она же структура БМК).



IOB — input-output block.

SM — switch matrix. Обеспечивает коммутацию трасс прохождения сигналов.

CLB — configurable logic block.

Несмотря на то, что названия другие, по сути это то же, что мы видели в CPLD.

Принципиальное отличие от CPLD: нет глобальных трассировочных линий. Следовательно, задержки распространения сигналов неодинаковы.

Задержка появляется не из-за прохождения по линии. Задержка возникает в элементах на пути. Повторитель, например. Если таких элементов нет, то и задержек не будет. Количество таких элементов на пути прохождения сигналов в FPGA непредсказуемо, зависит от траектории, соответственно, задержки будут разными.

Кроме блоков, указанных на рисунке, в составе FPGA может также быть: 1. контроллеры ISP (in-system programmability), JTAG; 2. специальные средства для тактирования — PLL, DLL, CDR; 3. контроллеры высокоскоростных последовательных интерфейсов. Позволяют вводить и выводить потоки со скоростями в гигабиты.

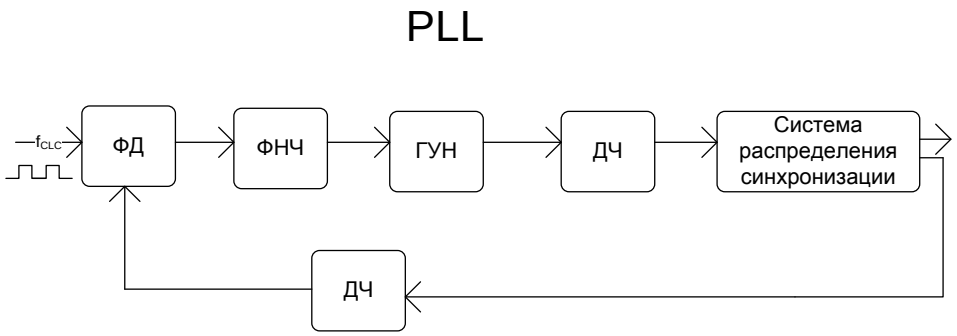
В современных FPGA реализуется иерархическая система трассировки.

10. Системы синхронизации и тактирования. PLL, DLL, CDR.

Системы глобальной синхронизации ПЛИС

Проблема: расхождение синхроимпульсов по фазе из-за различия в задержке времени распространения. Получается, что разные модули срабатывают в разные моменты времени. Это явление называется clock skew. Основная задача системы синхронизации: обеспечить синфазность прихода синхроимпульсов в разные точки кристалла. Вторая задача: синтез частот.

1. PLL — phase locked loop. Это ФАПЧ. (Altera) Это аналоговые устройства, которые в определённом количестве имеются на кристаллах. Можно использовать их в проекте.



k, l в ДЧ задаёт разработчик с целью изменять частоту.

Оттого, что это аналоговое устройство, вытекают:

Плюс: непрерывное (не дискретное) отслеживание изменения фазы.

Минус: в некоторых случаях система может оказаться неустойчивой из-за наличия обратной связи.

2. DLL — delay locked loop (xilinx).

Это почти то же самое. Но устройство уже цифровое.

Разница в управляемой линии задержки. Длительность изменяется дискретно, погрешность Δτ. Это минус.

Плюс в том, что устойчивость выше, чем у аналоговой системе.

3. CDR — clock data recovery. У PLL и DLL требования были в синфазности тактовых импульсов во всех точках кристалла. Но это требование не является настолько уж обязательным. Главное, чтобы они были правильно сфазированы относительно полезных данных.

Здесь используется механизм CDS — clock data synchronization. Последовательно формируются несколько тактовых сигналов с разными фазами. В составе информационных данных передаётся известное слово-эталон, оно принимается несколько раз при использовании нескольких синхросигналов в качестве тактирующих. Это слово передаётся в нужную точку кристалла, приём производится несколько раз, каждый раз используется тактовая последовательность с другой фазой. И отбирается синхропоследовательность, обеспечивающая безошибочный приём.

11. Системы-на-кристалле: основные понятия, классификация. Требования к ПЛИС, используемым для построения систем на кристалле. Hard- и soft-ядра. Концепция применения IP-ядер.

System on Chip. Два вида:

1) Система на кристалле с блочной архитектурой:

Плюсы: сравнительно высокое быстродействие фиксированных ядер (на 20…70% больше, чем в программных ядрах, реализованных на ПЛИС); меньшее энергопотребление; меньшая занимаемая площадь на кристалле. Эти преимущества связаны с тем, что эти модули разрабатываются и изготавливаются непосредственно производителем микросхемы, которому о своём кристалле всё известно, значит, они построены оптимально.

Минусы: ограниченная функциональная гибкость (если есть процессор или интерфейс, другого процессора и интерфейса не будет); ядра жёстко фиксированы на кристалле, как следствие, может быть затруднено размещение и трассировка.

Доля таких систем уменьшается.

2) Система на кристалле с однородной структурой (generic SoC).

По сути, это обычные большие ПЛИС с миллионами вентилей. На этой ПЛИС реализуется несколько функциональных блоков.

Минусы: меньшее быстродействие ядер; более высокие энергопотребление и занимаемая площадь. Потому что ядра, реализуемые программно, не могут быть настолько хорошо оптимизированы. Во-первых, эти ядра зачастую разрабатываются не производителем ПЛИС.

Плюсы: более высокая функциональная гибкость; возможность использования т. н. IP-ядер (intellectual properties): готовые модули хорошо отлажены, а если нет, можно предъявить претензии продавцу; улучшенные возможности размещения и трассировки кристалла.

Процессорное ядро Nios (Altera) — бесплатный vhdl-файл, реализующий простенький микропроцессор. Ещё MicroBlaze, PicoBlaze (Xilinx).

Требования к ПЛИС, используемых для реализации Generic SoC:

1. высокая логическая ёмкость (большое число эквивалентных вентилей);

2. возможность реализации нескольких тактовых доменов, то есть возможность тактирования разных областей кристаллов разными тактовыми частотами.

Преимущества систем на кристалле как таковых по сравнению с реализацией на нескольких микросхемах:

1. улучшенные массогабаритные показатели;

2. более высокая надёжность: основные отказы из-за отсутствия контакта либо лишнего контакта, вероятность нарушения внутри микросхемы в штатном режиме ничтожно мала;

3. меньшее энергопотребление.

12. Оценка логической емкости ПЛИС.

Оценка логической ёмкости ПЛИС

Здесь существуют разные подходы.

а) просто число эквивалентных вентилей (2И-НЕ)

$$Q_{\text{эв}} = \frac{S_{\text{кр}}}{S_{\text{2И–НЕ}}}$$

- total (берётся вся площадь кристалла) — неправдоподобно;

- usable (в рассмотрение берётся только часть кристалла) — в техдокументации.

б) PREP — programmable electronics performance corporation

$$Q_{\text{эв}} = \frac{1}{10}\sum_{i=1}^{10} q_i N_i$$

10 — набор из десяти эталонных схем, в качестве которых используются некоторые типовые функциональные узлы (дешифраторы, счётчики и т. д.)

q_i — логическая сложность *i*-й эталонной схемы, реализованной на образцовом базовом матричном кристалле. В качестве него одно время использовался LCA 300K.

N_i — максимальное число реализаций *i*-й эталонной схемы на оцениваемом кристалле.

Зачем привлекать образцовый кристалл? На разных ПЛИС логическая сложность может оказаться разной. Для реализации одного устройства может понадобятся разные ресурсы. Для единообразия выбрали базовый матричный кристалл. Известно, какую логическую сложность имеет эталонная схема, если она реализуется на нём. Затем смотрят, сколько раз эта схема может быть реализована на исследуемом кристалле. И так по каждой эталонной схеме из 10. И усредняют. Цифра будет поменьше, чем usable.

Эти две методики никак не учитывают наличие памяти на кристалле. После того как появились кристаллы со встроенной памятью, стали раздельно рассчитывать области.

в) логическая ёмкость ПЛИС, состоящей из логического массива и блоков памяти.

$$Q_{\text{лм}} = q_{\text{лб}} N_{\text{лб}}$$

$$Q_{\substack{\text{зу по прямому назначению} \\ \text{(в количестве экв.вентилей)}}} = q_{\text{бит}} M$$

$q_{\text{бит}}$ — количество эквивалентных вентилей образцового БМК, требуемое для хранения одного бита.

M — ёмкость памяти в битах.

Вторая возможность использования памяти — для табличной реализации.

$$Q_{LUT} = q_{\text{мп}} N_{\text{мп}}$$

$q_{мп}$ — эквивалентная логическая ёмкость одного модуля памяти, подсчитанная с привлечением образцового БМК.

$$Q_{зу} > Q_{LUT}$$

Потому что для хранения бита на БМК понадобится четыре вентиля. Реализовывать память на БМК неэффективно. Поэтому $q_{бит}$ большая. А память изначально предполагалась не для логических функций, а для хранения, она оптимизирована под хранение. А когда она используется для табличного преобразования, это комбинационное устройство, $q_{мп}$ выходит небольшим.

Как же считать? Надо считать с учётом того, в каком режиме память работает. Обычно для реальных проектов можно применить формулу

$$Q_{ПЛИС} = Q_{лм} + dQ_{зу} + (1 - d)Q_{зу}$$

d показывает долю памяти, используемой в качестве запоминающего устройства. Для большинства проектов $d = 0{,}65 \ldots 0{,}85$.

13. Оценка быстродействия ПЛИС.

CPLD (задержки распространения сигналов одинаковы) vs FPGA

CPLD: $t_{зр}$.

FPGA: f_{max} — максимальная частота:

- тактирования всей схемы, реализованной на кристалле;

- тактирования отдельно взятого счетчика. Отдельно взятый счётчик может тактироваться с большей частотой, чем вся схема. $f_{\max сх} \approx \frac{1}{2} f_{ст}$.

Как ещё оценить максимальную частоту?

В современных ПЛИС возможна ситуация, когда ядро и периферия тактируются разными частотами.

$$f_{core} \begin{matrix} \leq \\ \geq \end{matrix} f_{i/o}$$

Современные частоты — 400…500 МГц. Современные ПЛИС обрабатывают видеосигналы высокой чёткости без дополнительных внешних микросхем.

14. Проектирование радиоэлектронной аппаратуры: цель, задачи, концепция, методология, этапы.

Целью проектирования является получение технической документации, позволяющей изготовить новый объект с заданными свойствами и с заданным функционированием в заданных условиях.

Составляющие проектирования:

1. Элементы: ПЛИС (техническая база, на которой будет реализовано устройство).

2. Инструментарий: САПР (то, с помощью чего ведётся проектирование).

3. Методика применения инструментария (как следует правильно использовать инструментарий для достижения целей).

4. Абстракции: синтаксис HDL (hardware design language) (термины, понятия, определения, используемые в процессе проектирования).

Этапы проектирования:

15. Области (функциональная, структурная, геометрическая) и уровни представления ПЛИС. Проектирование как процесс последовательного спуска по уровням областей представления (диаграмма Гайского-Кана).

Выделяют три области представления ПЛИС:

S — структурное, F — функциональное, G — топологическое (геометрическое) описание.

Каждая из областей имеет четыре уровня.

Всё начинается с уровня F1 — алгоритмический уровень. На нём определяется общий алгоритм функционирования устройства.

Затем идём на уровень S1 — уровень крупных структурных блоков (PMS-level — processor, memory, switcher level).

Затем G1 — определение плана кристалла.

Затем F2 — логический уровень. Описание в терминах алгебры логики. Пишем логические уравнения.

S2 — уровень регистровых передач (RTL, register transfer level).

G2 — уровень крупных функциональных блоков.

F3 — схемотехнический уровень. Но это слово тут не годится, потому что это функциональное описание. Эквивалентом принципиальной схемы являются дифференциальные уравнения. S3 — уровень логических вентилей (GL, gate level).

G3 — геометрия вентилей.

F4 — масочный, или топологический уровень. Описание маски для изготовления кристалла в каких-то терминах.

S4 — TL, transistor level.

G4 — геометрия отдельных транзисторов и маска для изготовления полупроводникового кристалла.

17. Иерархия языков проектирования аппаратуры.

Рассмотрим языки, относящиеся к семейству xHDL — hardware design language.

Уровни языка	язык программирования	HDL
языки реализации	машинный код	таблица соединений, списки цепей (netlist)
машинно-ориентированные языки	ассемблер, макроассемблер	AHDL (altera), ABEL (xilinx), PLDASM
процедурно-ориентированные языки (поведенческое, функциональное описание)	Pascal, Fortran, C, …	VHDL, Verilog
объектно-ориентированные языки (выполняют структурное описание)	C++, Java	Hardware C (мёртвый, нет компилятора. но нужен для моделирования)
языки, позволяющие описывать как программную, так и аппаратную части	System C	Superlog

18. Интерфейс JTAG: назначение, общие принципы граничного сканирования.

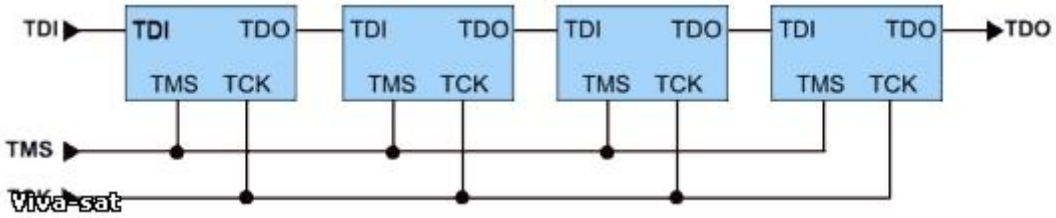
(Joint Test Action Group)

Существует два стандарта на этот интерфейс: IEEE Std. 1149.1 (1990), Std. 1149.1a (1993).

Изначально этот интерфейс был создан для тестирования. Раньше использовался метод контактирующих щупов. Но с появлением микросхем с большим количеством ножек метод стал невозможным. Тогда был разработан последовательный интерфейс JTAG, позволяющий вводить тестовые сигналы и считывать реакции с использованием всего двух выводов микросхемы. Для этого в микросхеме должны быть предусмотрены специальные модули, которые обеспечивают работу этого интерфейса.

Два аспекта стандарта:

1. Порядок обмена информацией между микросхемами, соединёнными в последовательную JTAG-цепочку:



TDI — test data input;

TDO — test data output;

TMS — test mode select — выбор состояния.

Микросхемы могут соединяться в цепочку, и данные могут передаваться по цепочке.

2. Стандартизация связи между внутренними модулями микросхемы и её выводами, ориентированными на тестирование. Как следствие, был разработан метод BST — boundary scan testing — метод граничного сканирования. Предназначен для:

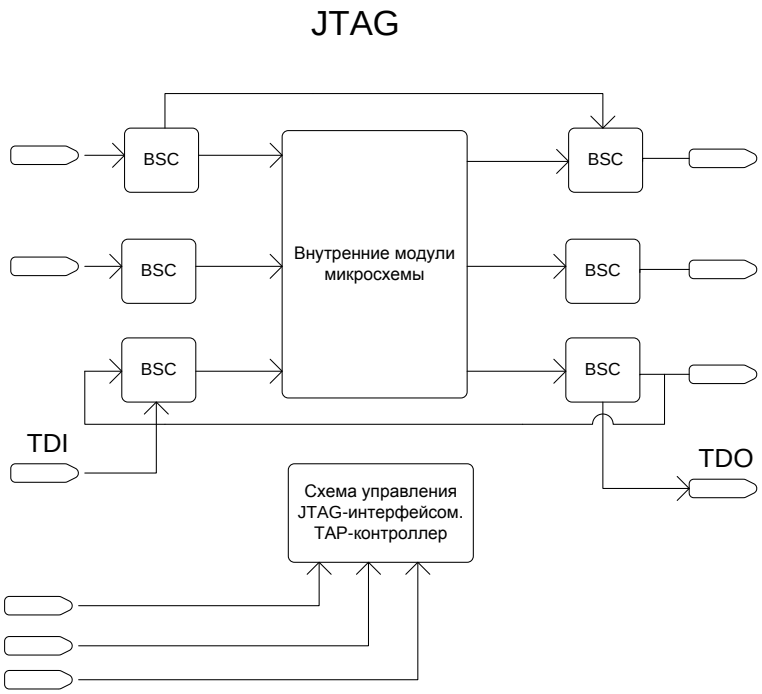
1) проверки работоспособности микросхемы с помощью встроенных специальных модулей — тестирование внутренностей микросхемы;

2) проверки качества монтажа микросхемы на печатной плате;

3) ввода или вывода данных в штатных режимах работы микросхемы.

Оказалось, что интерфейс годится не только для тестирования, но и для загрузки и выгрузки данных в микросхему. Можно конфигурировать микросхему, можно вести информационные потоки.

Логическая структура микросхемы с JTAG:



BSC — boundary scan cell. Такая ячейка есть около каждой ножки. Схема управления управляет всем обменом данным.

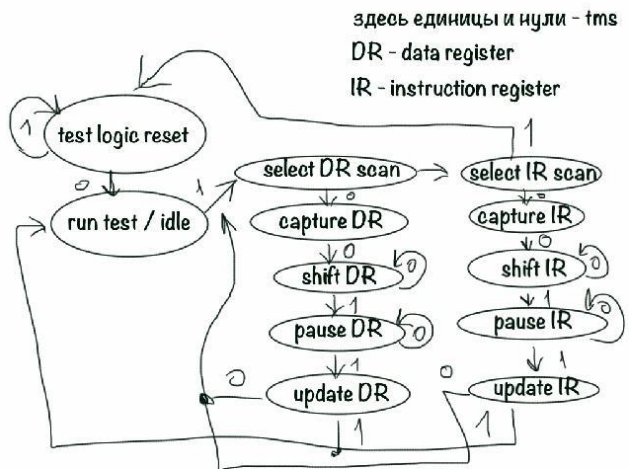
TAP — test access port.

Каждая BSC может данные, которые поступают с порта TDI, направить к следующей ячейке, или направить эти данные к внутренним модулям микросхемы. Также ячейки, обслуживающие выходы, могут принять данные как от других ячеек, так и от внутренностей микросхемы.

Таким образом, любые данные могут быть введены и выведены. Вместо того, чтобы данные подавать через определённую ножку, эти данные подаются через специализированный порт, доходят до нужной ноге и направляются внутрь кристалла.

Интерфейс не высокоскоростной, но удобный.

Диаграмма состояний контроллера порта доступа (tap-контрллера). Tap - test access port.



Run test - режим внутреннего самотестирования. В системе сканирования есть два регистра: данных и команд. Через регистр данных проходят все вводимые и выводимые данные. Д-триггеры, соединенные в последотельную цепочку, и образуют регистр данных.

Кроме регистра данных есть еще и регистр команд, где хранятся коды управляющих команд.

Можно выбрать работу с регистром данных или команд.

Capture - фиксация в триггере бита и в регистре данных некоторого слова данных (или в регистре команд кода очедной команды).

Shift - сдвиг. Мы двигаем биты от ячейки к ячейке. Если это состояние совершается несколько раз, можно по всей периферийной цепочке продвигать данные насколько угодно далек.

Update - захват нового состояния.

Команды граничного сканирования

С обязательными командами должны работать все устройствы с jtag. Перечислим их.

Bypass - обход при передаче данных в следующую микросхему.

Intest - тестирование микросхемы путем подачи данных от внешнего прибора. Им может быть компьютер.

И исходные данные, и результат заносятся в региср данных DR.

Extest - для тестирования внешних выводов. При этом внутренние бллоки отключаются от bsc-ячеек. То есть не отпаялась ли ножка, нет ли кз.

Sample/preload - контроль микросхемы в штатном режиме работы. То есть обмен данными с внешним миром происходит через все ножки микросхемы. Поэтому в регистр данных заносятся данные с внешних выводов.

В отличие от intest и extest, где данные вводятся в специальные ножки tdi/tdo.

Все остальные не являются обязательными, но отметим еще одну популярную.

HighZ - перевод всех выводов микросхемы в третье состояние.

Jtag со времем стал применяться не только для тестирования, но и для ввода и вывода данных, в частности, конфигурационных. Просто последовательный порт. Не очень скоростной, но хватает для обычных условий.

21. Особенности САПР Quartus II фирмы Altera.

При работе с микросхемами программируемой логики основным инструментом является САПР. Фирма Altera предлагает два САПР MAX+PLUS II и Quartus II. Каждый САПР поддерживает все этапы проектирования: Ввод проекта, Компиляция, Верификация и Программирование. Каждый САПР имеет Tutorial (Самоучитель), который устанавливается при инсталяции пакета. Tutorial состоит из занятий, в ходе которых проходится весь цикл проектирования от ввода проекта до программирования микросхем. При инсталяции также устанавливаюся файлы, описывающие проект так, что в ходе изучения Tutorial можно пропускать отдельные занятия и использовать готовые файлы. Например, можно пропустить "Ввод проекта" и перейти к "Компиляции" проекта, используя готовые файлы.

САПР MAX+PLUS II является более простым в освоении по сравнению с Quartus II. Он поддерживает семейства MAX, FLEX и ACEX, которые содержат микросхемы с 5В питанием и количеством функциональных преобразователей от 32 до 4992 и имеет меньшее количество настроек. Этот САПР фирма Altera не развивает и рекомендует переходить на Quartus II.

САПР Quartus II является основным. Фирма Altera активно его развивает. Он поддерживает все новые семейства микросхем и обладает особенностями, которых нет в MAX+PLUS II.

Бесплатные САПР фирмы Altera

Фирма Altera предлагает бесплатные версии САПР MAX+PLUS II BASELINE и Quartus II Web Edition, которые поддерживают все этапы проектирования от ввода проекта до программирования. Ограничениями являются количество поддерживаемых микросхем и некоторые функции для Quartus II. Подробнее см. таблицу Сравнение особенностей САПР фирмы Altera. Бесплатные версии САПР можно скачать с сайта фирмы Altera (https://www.altera.com/support/software/download/sof-download_center.html)

Чтобы начать работать с бесплатными версиями нужно получить на сайте фирмы Altera лицензионный файл, который генерируется по номеру сетевой карты. Лицензионный файл для MAX+PLUS II BASELINE можно также получить по номеру жесткого диска. Продолжительность действия лицензионного файла для Quartus II Web Edition 150 дней для MAX+PLUS II BASELINE 180 дней. По окончании этого срока нужно получать новый лицензионный файл.

22. Микропроцессоры. Классификация процессоров. Понятия архитектуры и структуры. Варианты архитектур. Конвейерное выполнение команд.

Классификация микропроцессоров:

1) общего назначения (универсальные). Тенеденция: наращивание прроизводительности. Достигается несколькими путями. Это повышение тактовых частот и повышение разрядности.

2) специализированные:

2.1. Микроконтроллеры. Особенность: ориентированы на задачу управления чем-либо. От них не требуется сверхвысокая производительность, но требуются широкие функциональные возможности. Поэтому редко бывают большой разрядности. Работают на невысоких тактовых частотах. Зато прямо на борту имеют дополнительные устройства: интерфейсные схемы (vga, usb, fire wire...), память.

2.2. Цифровые сигнальные процессоры (dsp). Предназначены для обработки сигналов в реальном времени. Оптимизированы для быстрого выполнения арифметических операций. Это дешевле, чем ставить компьютер. Высокая производительность из-за оптимизированной системы команд, архитектуры. Зачастую есть некоторорые аппаратные модули, наппример, апаратный умножитель. Который работает мгновенно. Иногда модули быстрого преобразования Фурье.

2.2.1. С фиксированной запятой. 2.2.2. С плавающей запятой.

Архитектура и структура микропроцессора

Архитектура ммикропроцессора - это набор его аппаратных и программных средств, доступных пользователю:

1. Программно доступные регистры;

2. Исполнительные устройства (АЛУ);

3. Система команд;

4. Размер адресного пространства и способы адресации;

5. Виды и способы обработки прерываний.

Два основных типа архитектуры микропроцессооров:

1. Принтстонская (фон Неймана). 1940е. Особенность: общая память программ и данных и одна общая шина.

Недостаток: только последовательная выборка команд и операндов из памяти, а значит, меньше быстродействие. Преимущества: простая структура, а также в том, что массив памяти можно перераспределить между командами и данными.

2. Гарвардская. Раздельная память программ и данных. Раздельные шины. Плюс: возможность одновременной выборки кода команды и операнда. Более высокое быстродействие. Минусы: более сложная структура и невозможность перераспределения памяти. Но они сошли на нет. Поэтому именно Гарвардская архитектура получает большее распространение.

Классификация микропроцессоров по типу систем команд.

1. Cisc-процессоры (complpex instruction set computer). Большой набор команд разной длины с различными способами адресации операндов.

Плюсы: найдется команда на любой случай. Частота применения команд будет разной.

Примеры: Pentium (внешне для программиста), Motorolla 680, AMD.

2. Risc-процессоры (reduced instruction set computer) - ограниченный набор команд одинаковой длины с ограниченным числом способов адресации операндов.

В микропроцессорах осуществляется конвейерное выполнение команд, то есть не единомоментно, а поэтапно.

Команда состоит из машинный циклов. ВК - выборка кода команды из памяти. ДК - дешифрирование команды в управляющем устройстве. ФА - формироввание адреса операнда. Воп - выбора операнда из памяти данных. ВО - выполнение операции. РР - размещение результата в памяти, регистре...

Нет необходимости дожидаться выполнения всех циклов. Можно начать следующую, не дожидаясь. Это и есть конвеерное выполнение команд. Используется практически во всех современнных микропроцессорах. Почему это возможно? Потому что циклы независимы.

Интервал выполнения всей команды называется командным циклом.

Если все команды имеют одинаковый размер, конвеер работает хорошо, в нем нет простоев и ожиданий. Простои снижают общую производительность. Поэтому risc-процессры и получили популярность. Поэтому современные пентиумы должны выглядеть для пользователя как cisc (для совместимостью с 8080), но внутри они risc.

Примеры: SPARC (Sun).

3. VLIW - very large instruction word. Очень большие команды. Если в risc пошли путем убыстрения конвеера, то здесь одна команда обеспечивает выполнение сразу нескольких операций в нескольких исполнительных устройствах. То есть происходит распараллеливание операций. За счет этого тоже повышается производительность. Не получила большого распространения. Пример: Itanium (HP+intel), некоторые dsp, PA 8500 (Hewlett Packard).

Есть так называемые суперскалярные процессоры, содержащие несколько исполнительных устройств и конвейеры и одновременно выполняют несколько команд.

23. Микропроцессоры. Регистровая модель. Способы адресации операндов.

Регистровая модель

В составе микропроцессора есть регистры, которые делятся на

1. регистры общего назначения. Они образуют так называемую регистровую память. Она самая быстродействующая.

2. служебные регистры:

- program counter. Его состояние увелививается на единицу, когда извлекается новая команда. Является указателем адреса очередной команды.

- status register - регистр состояния, где хранятся флаги.

- регистр управления режимом работы процессора control register. Процессор может обратиться к нему, посмотреть значение и изменить свою работу в соответствии с обнаруженным числом.

- регистры тестирования test registers.

- ...

В служебные регистры данные так просто записывать нельзя. Вся совокупность регистров образует регистровую модель данного процессора.

Основные способы адресации операндов

Операнды хранятся в памяти данных. Их надо как-то извлекать, а результаты в память записывать. Память данных - память с произвольным доступом, обращение к ячейкам которой производится по их адресам.

1) непосредственная адресация: операнд размещается в самой команде;

2) прямая: в команде размещается адрес ячейки памяти. Mov A, addr.

3) регистровая. В команде имя регистра, содержащего операнд. Mov A, B.

4) косвенно-регистровая. В команде содержится имя регистра, содержащего адрес ячейки памяти с операндом.

24. Микропроцессоры. Группы команд. Примеры команд основных групп.

1. Команды пересылки: mov, in, out...

2. Арифметико-логические команды. Они формируют флаги:

- z - zero - флаг нулевого результата. Устанавливается, если результат равен нулю.
- c - carry - флаг переноса.
- n - negative - флаг отрицательного результата.

Флаги хранятся в регистре состояний.

3. Команды сдвига. Сдвиг какого-либо числа в регистре влево или вправо. Нужно для умножния и деления на степени двойки.

4. Команды сравнения. Устанавливают флаги.

5. Команды битотвых операций. Это операции над отдельными разрядами. В том числе над флагами. Это инвертирование, например.

6. Команды управления программ.

6.1. Это, например, команды безусловных переходов, когда, например,

```
Instr n-1
Instr n
Instr n+1
jmp M
...
M instr k
```

6.2. Есть команды условных переходов. Когда проверяется некое условие и переход осуществляется только в случае выполнения его.

JZ M; если (z)=1, происходит переход на команду с меткой M.

6.3. А также команды обращения к подпрограмме.

Call addr; следущее значение счетчика команд (pc)+1 заносится в стек (адрес следующей команды); addr -> pc. Управление передается подпрограмме, располагающейся по адресу addr. В конце подпрограммы располагается команда ret, возвращающая из стека адрес следующей команды.

6.4. Команда организации программных циклов с заданным числом выполнений. Когда надо сделать, чтобы фрагмент программы циклически выполнялось заранее известное количество раз.

Loop

6.5. Команды прерываний.

7. Команды управления процессом.

nop - нет операции. Пустая команда. Для временного промежутка.

Stp, hlt - stop, halt.

25. Особенности архитектуры микроконтроллеров. Популярные семейства микроконтроллеров.

Микроконтроллер - специализиванный микропроцессор, ориентированный на задачи управления. Особенность его в тотм, что он должен обладать широкими функциональным возможностями, поэтму там есть ряд дополнительных узлов, блоков и модулей.

Микроконтроллеры часто имеют внутреннюю память. Могут иметь и ПЗУ, и ОЗУ. ПЗУ может педназначаться и для хранения программ, и констант.

Прерывание: представим, что микропроцессор выполняет какую-то задачу. Вдруг внешнему устройству понадобилось к нему обратиться. Постейший способ - без прерываний. Микропроцессор опрашивает устройство раз в миллисекунду, например. Это нехорошо. Приходится отвлекаться. Поэтому лучше сделать так, чтобы внешнее устройство само могло постучаться. Для этого нужны прерывания. При необходимости внешнеее устройство посылает специальный сигнална специальный вывод процессора - запрос прерывания. Процессор прекращает выполнение текущей задачи и переходит к подпрограмме обработки прерывания. Затем возврат к решению основной задачи с места остановки.

Бывает многоуровневая система прерываний. Будет обрабатываться прерывавшие с высшим приоритетом.

Обычно обмен с памятью происходит через процессорное ядро. Что опять-таки отвлекает его. Чобы этого не было, устанавливают контроллер прямого доступа к памяти.

Параллельный порт обычно один. Последовательных портов несколько. От контроллеров периферийных устройств они отличаются тем, что последние предназначены для подключения устройств с нестандартным интерфейсом.

Таймеры - это счетчики, предназначенные для измерения временных интервалов. Или формирования сигналов с заданными временными параметрами.

Основные режимы работы микропроцессоров

1. Выполнение основной программы.

2. Вызов и исполнение подпрограммы.

3. Обслуживание прерываний. Есть кроме всего прочего прерывания программные, которые тоже вызывают обращение к подпрограмме.

4. Прямой доступ к памяти.

Классификация по физическому состоянию процессора:

1. Активный режим.

2. Режим пониженного энергопотребления. Скачки сигнала происходят в моменты перезарядки паразитных емкостей, на что расходуется мощность. Снижая тактовую частоту, энергропотребление понижаем.

3. Режим сброса. (Начального запуска).

Два вида архитектуры микропроцессора: открытая (шины выведены на внешние выводы микроконтроллера, что позволяет в первую очередь использовать внешнюю память) и закрытая (шины не подключены к выводам).

Помимо модулей, указанных на рис., в структуру микроконтроллера входят:

- модули контроля питания;

- модули диагностики;

- модули программирования и внутрисхемной отладки;

- модули тактированя.

UART - universal asynchronous receiver/transmitter - универсальный асинхронный приемопередатчик.

Выполняет сериализацию и десериализацию и ввод/вывод.

Основные семейства микроконтроллера.

1) Самое распространенное ядро - Intel MCS-51 (1980). Производится до сих пор. Похож на процессор 8080. Есть встроенное ПЗУ от 4 до 32 кбайт. Оперативки 128 кб. Частота тактирования шины от 1 до 3 МГц. fCLK/fBUS=12. Четыре порта ввода/вывода, несколько таймеров. Микроконтроллер восьмиразрядный. Это разрадность шины данных (не адреса! Ее определяет размер адресуемого пространства). Здесь классическая cisc-архитектура.

2) PIC (microchip) - первый микроконтроллер с риск-архитектурой. Семейства 16, 24, 32-разрядные. Система из 33 команд, выполняемых за один машинный цикл. fCLK/fBUS=4. Применяются широко.

3) Ядро AVR (atmel) - risc-ядро с гарвардской архитектурой. Система из 120 команд. ПЗУ до 128 кб, RAM до 4 кб. В составе АЛУ имеются аппаратные умножители. То есть ядро повзоляет выполнять и DSP. fBUS=fCLK!